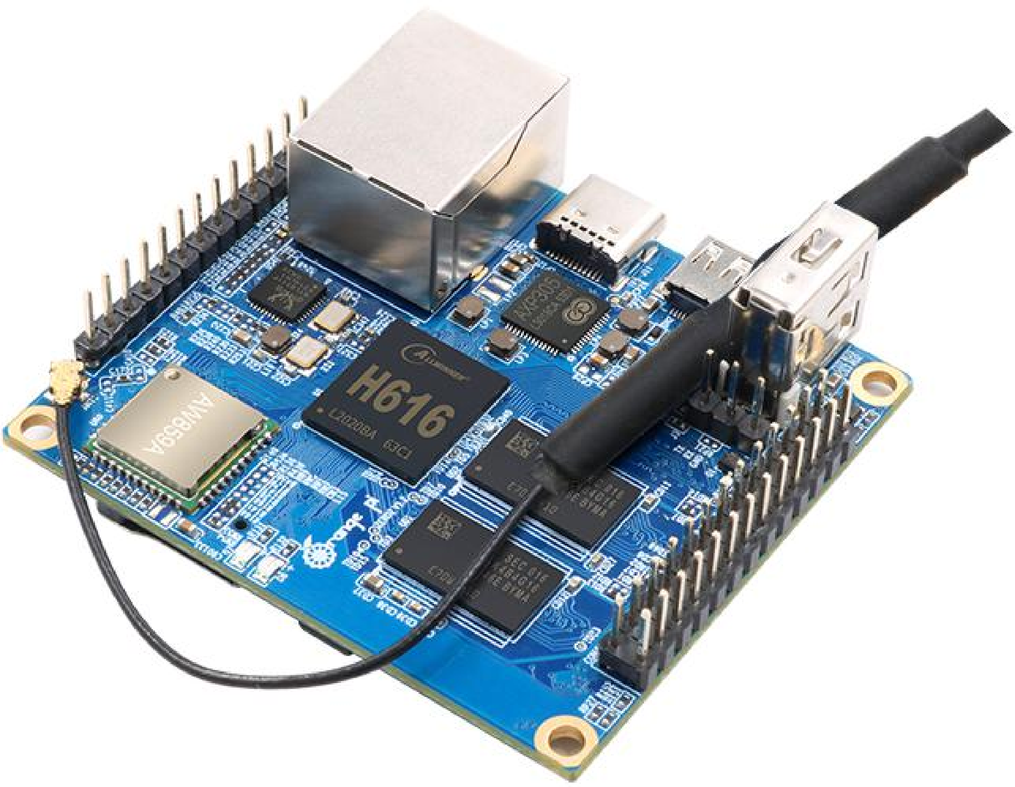
Orange Pi Zero 2
User Manual



Content

1. Basic features of Orange Pi Zero 2

1.1. What is Orange Pi Zero 2?

Orange Pi is an open-source single board computer, a new generation of ARM64 development board, which can run OS such as Android 10.0, Ubuntu and Debian, and so on. Orange Pi Zero 2 uses the Allwinner H616 system-on-chip and has 512MB/1GB DDR3 memory

1.2. The uses of Orange Pi Zero 2

We can use it to build:

●A computer
●A wireless server
●Games
●Music and Sounds
●HD video
●A Speaker
......

Pretty much anything else, because Orange Pi Zero2 is open source

1.3. Who's it for?

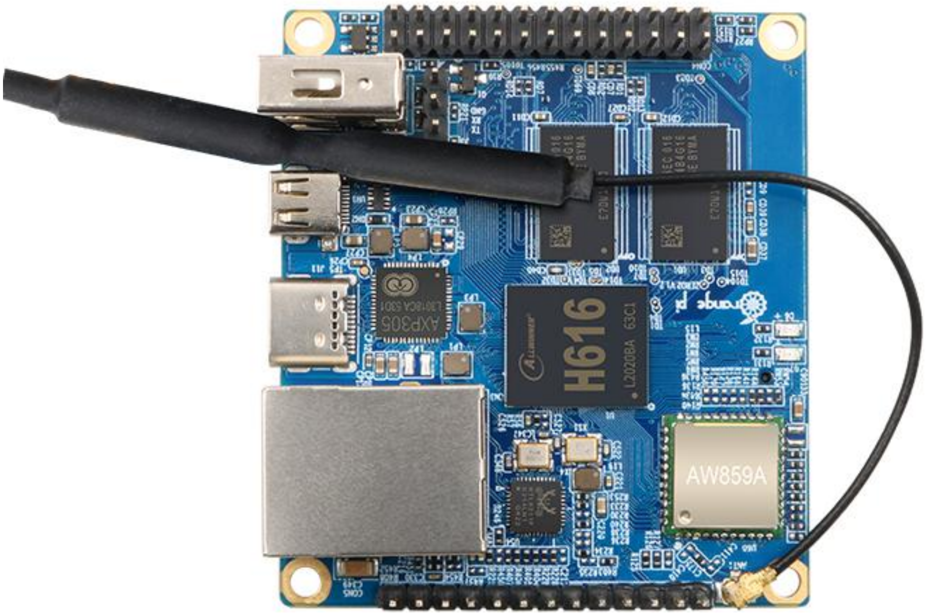
Orange Pi Zero 2 is for anyone who wants to start creating with technology – not just consuming it. It's a simple, fun, useful tool that you can use to start taking control of the world around you.

1.4. Hardware Specification of Orange Pi Zero 2

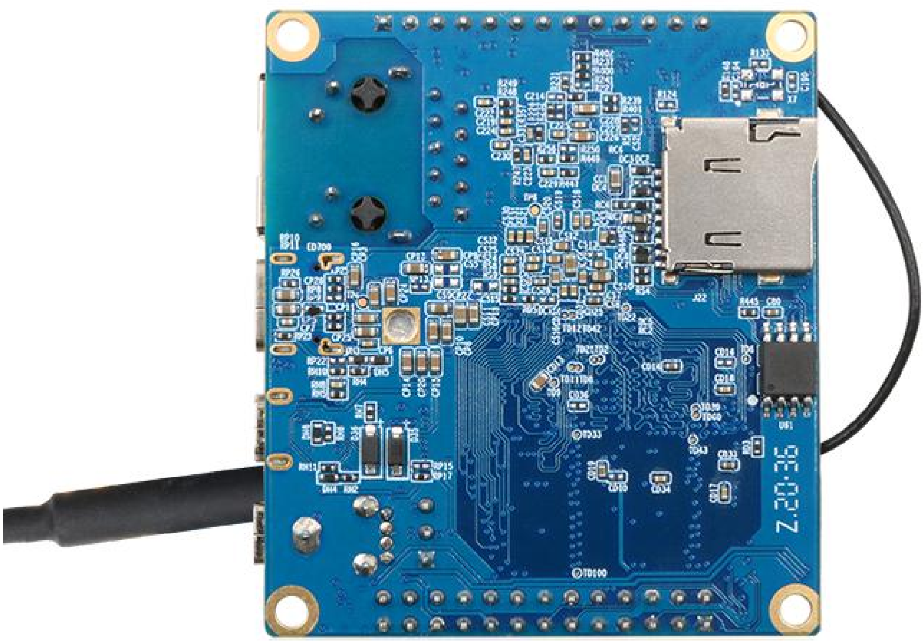
Hardware Specification	
CPU	Allwinner H616 64-bit 1.5GHz high-performance Quad-core Cortex-A53 processor
GPU	Mali G31 MP2 Supports OpenGL ES 1.0/2.0/3.2、OpenCL 2.0
Memory(SDRAM)	512MB/1GB DDR3 (Shared with GPU）
Onboard Storage	MicroSD card slot、2MB SPI Flash
Onboard Network	Support 1000M/100M/10M Ethernet
Onboard WIFI+BT	• AW859A Chip • Support IEEE 802.11 a/b/g/n/ac • Support BT5.0
Video Outputs	• Micro HDMI 2.0a up to 4K@60fps • TV CVBS output, Support PAL/NTSC （Via 13pin interface board）
Audio output	• Micro HDMI • 3.5mm audio port (Via 13pin interface board)
Power Source	Type-C interface 5V/2A input
USB 2.0 Ports	3*USB 2.0 HOST（Two of them are via 13pin interface board）
26pin header	with I2C, SPI, UART and multiple GPIO ports
13pin header	with 2*USB Host, IR pin, Tv-out、 Audio (no MIC) and 3 GPIO ports
Debug serial port	UART-TX、UART-RX and GND
LED	Power led & Status led
IR receiver	Support IR remote control (via 13pin interface board)
Supported OS	Android10.0、Ubuntu、Debian
Appearance specification	
Dimension	85mm×56mm
Weight	30g
range Pi™ is a trademark of the Shenzhen Xunlong Software CO., Limited	

1.5. Top and bottom views of Orange Pi Zero 2

Top view

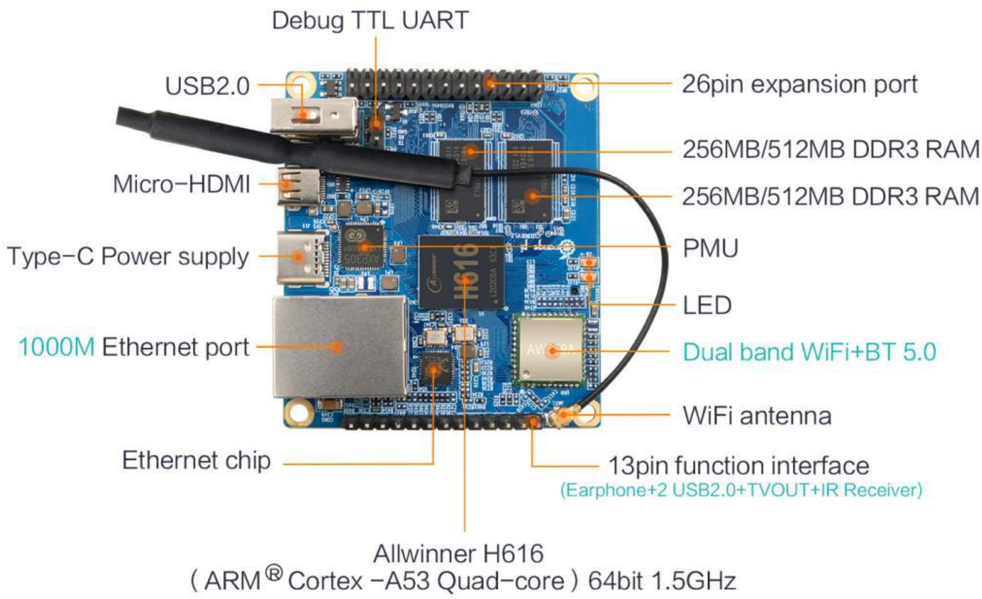


Bottom view:

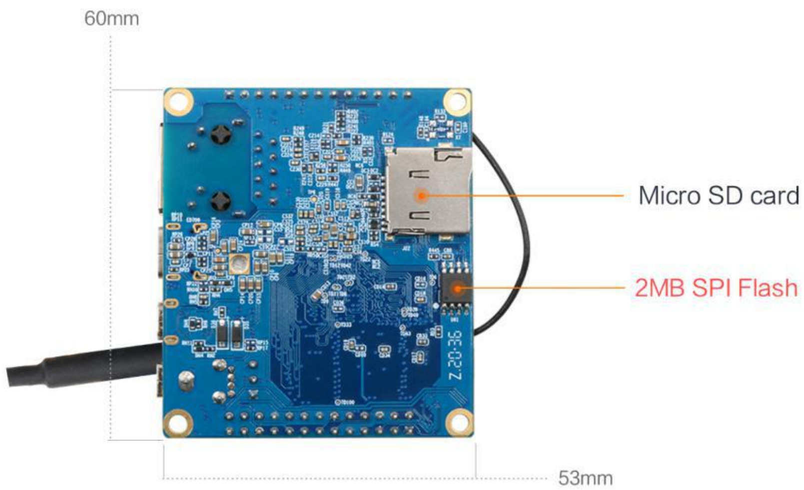


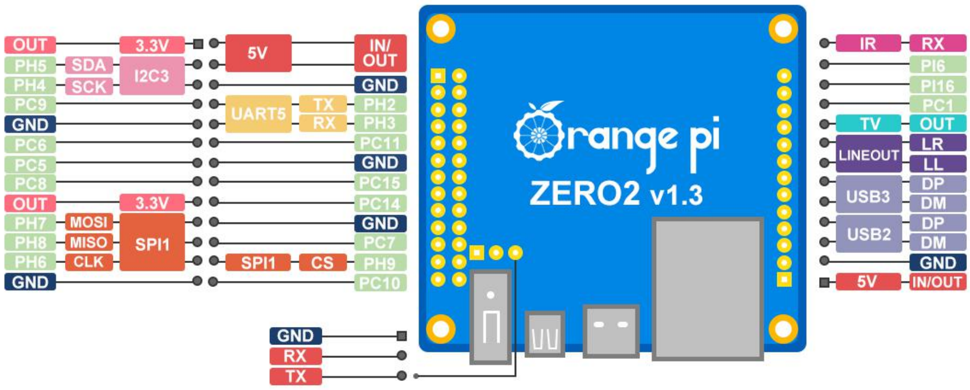
1.6. Orange Pi Zero 2 interface details

Top view



Bottm view





2. Introduction to the use of the development board

2.1. Prepare the necessary accessories

1) MicroSD card, a high-speed card above class 10 with a minimum capacity of 8GB, it is recommended to use SanDisk's MicroSD card. Orange Pi uses SanDisk's MicroSD card with all the tests. Other brands of TF cards may cause the OS to fail to start.

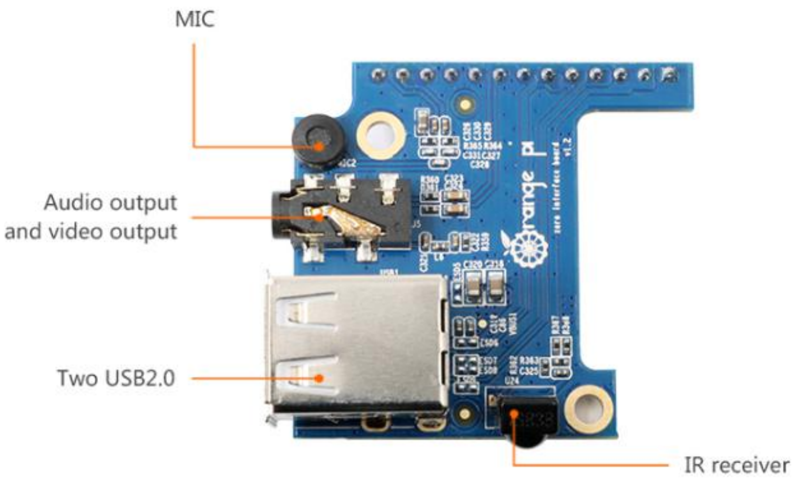
2) TF card reader used to read and write MicroSD card

3) Micro HDMI to HDMI cable used to connect the development board to an HDMI monitor or TV for display

4) Power Supply: Using a USB Type C interface data cable and a 5V/2A or 5V/3A high-quality power adapter

5) 13-Pin expansion board

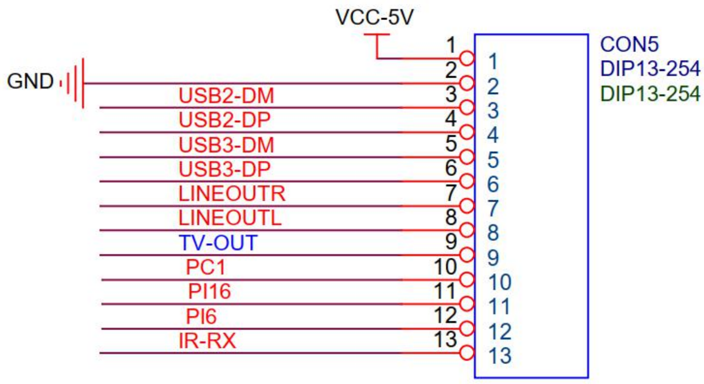
a. The expansion board is as shown in the following picture



b. The 13-Pin header on the Orange Pi Zero 2 development board can be connected to an expansion board to expand the functions that are not on the development board. The expansion board includes functions

1	MIC	**Not Support**
2	Analog audio and video output interface	Can be used to connect headphones to play music, or connect to the TV through AV cable to output analog audio and video signals (Android OS only)
3	USB2.0 x 2	Used to connect USB keyboard, mouse and USB storage device
4	IR function	Android OS can be controlled by IR remote control

c. The schematic diagram of the 13-Pin header on the Orange Pi Zero 2 development board is shown below



6) USB mouse and keyboard, as long as it is a standard USB mouse and keyboard, the mouse and keyboard can be used to control the Orange Pi development board

7) IR remote control mainly used to control the Android OS

8) 100M or 1000M network cable used to connect the development board to the Internet

9) AV cable, if you want to display video through the CVBS interface instead of HDMI interface, then you need to connect the development board to the TV through the AV cable

10) USB to TTL module and DuPont cable. When using the serial port debugging

function, you need USB to TTL module and DuPont cable to connect the development board and the computer

11) A personal computer with Ubuntu and Windows operating OS

1	Ubuntu14.04 PC	Optional, used to compile Android source code
2	Ubuntu18.04 PC	Optional, used to compile Linux source code
3	Windows PC	Used to flash Android and Linux images

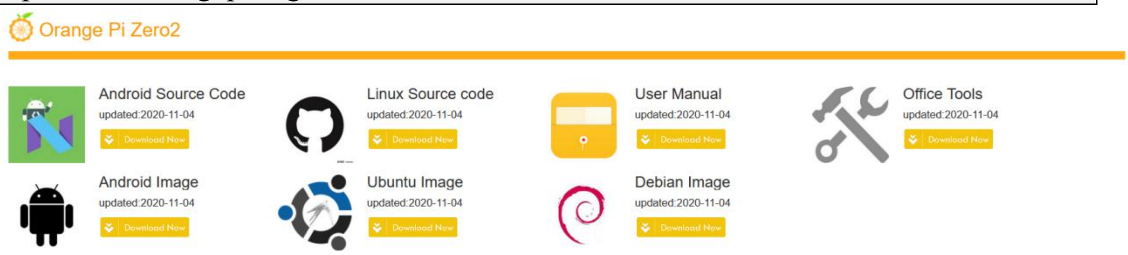
2.2. Download the image and relevant documents

1) The download URL of the Chinese version is

```
http://www.orangepi.cn/downloadresourcescn/
```

2) The download URL of the English version is

```
http://www.orangepi.org/downloadresources/
```



3) The information mainly contains

a. Android source code: saved on Baidu Cloud Disk and Google Cloud Disk

b. Linux source code: saved on GitHub, the link URL is

```
https://github.com/orangepi-xunlong/orangepi-build
```

c. User manual and schematic diagram: chip related data manual will also be placed here

d. Official tools: mainly include the software needed during the use of the development board
e. Android image: saved on Baidu Cloud Disk and Google Cloud Disk
f. Ubuntu image: saved on Baidu Cloud Disk and Google Cloud Disk
g. Debian image: saved on Baidu Cloud Disk and Google Cloud Disk

2.3. Method to flash Linux image to MicroSD card based on Windows PC

1) First, prepare a MicroSD card with 8GB or larger capacity. The transmission speed of the MicroSD card must be above class10. It is recommended to use a brand MicroSD card such as SanDisk.

2) Then use a card reader to insert the MicroSD card into the computer

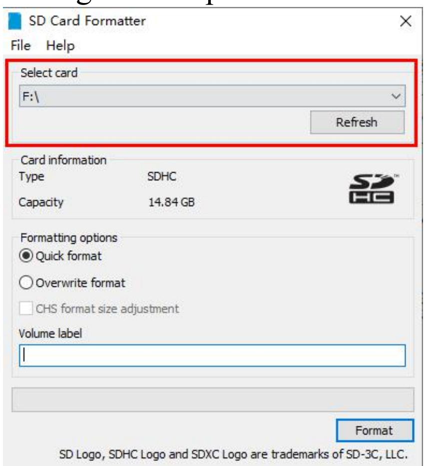
3) Then format the MicroSD card

a. You can use the SD Card Formatter software to format the MicroSD card, the download URL is

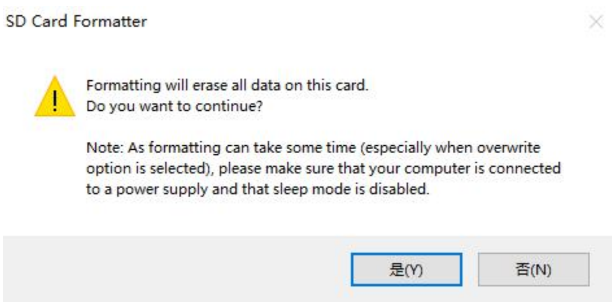
```
https://www.sdcard.org/downloads/formatter/eula_windows/SDCardFormatterv5_WinEN.zip
```

b. After downloading, just unzip and install, then open the software

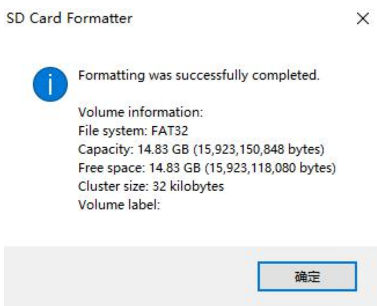
c. If the computer has only inserted a MicroSD card, the "Select card" column will display the drive letter of the MicroSD card. If the computer has multiple USB storage devices inserted, you can select the drive letter corresponding to the MicroSD card through the drop-down box



d. Then click "Format", a warning box will pop up before formatting, and formatting will start after selecting "Yes (Y)"



e. After formatting the MicroSD card, the message shown below will pop up, click OK



4) Download the Linux operating OS image file compression package you want to flash from download page of the Orange Pi official website, and then use the decompression software to decompress it. In the decompressed file, the file ending with ".img" is the operating OS image file, with a normal size Are above 1GB

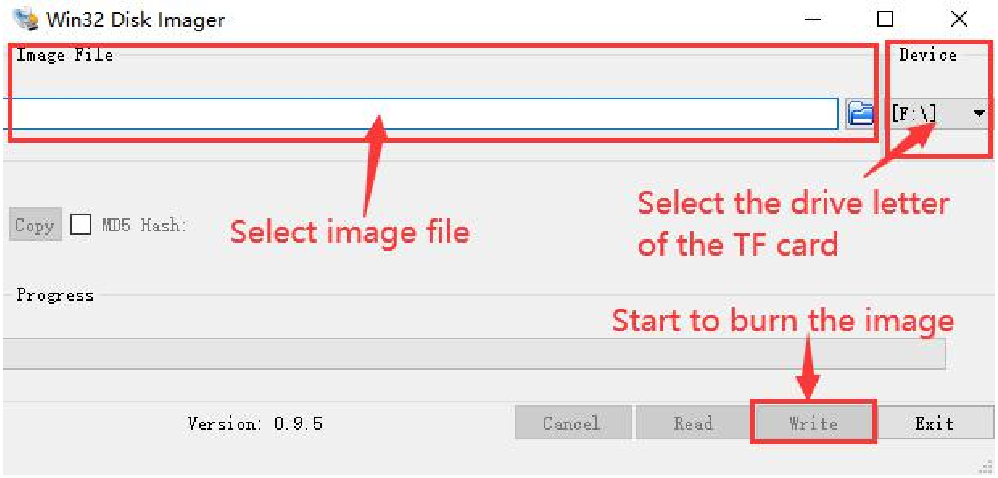
5) Use Win32Diskimager to burn Linux image to MicroSD card

a. The download page of Win32Diskimager is

http://sourceforge.net/projects/win32diskimager/files/Archive/

b. Install directly after downloading, the Win32Diskimager interface is shown below

a) First, select the path of the image file
b) Then confirm that the drive letter of the MicroSD card is consistent with the one displayed in the "Device" column
c) Finally, click "write" to start flash



c. After the image is written, click the "Exit" button to exit, and then you can pull out the MicroSD card and insert it into the development board to boot

2.4. Method to flash Linux image to MicroSD card based on Ubuntu PC

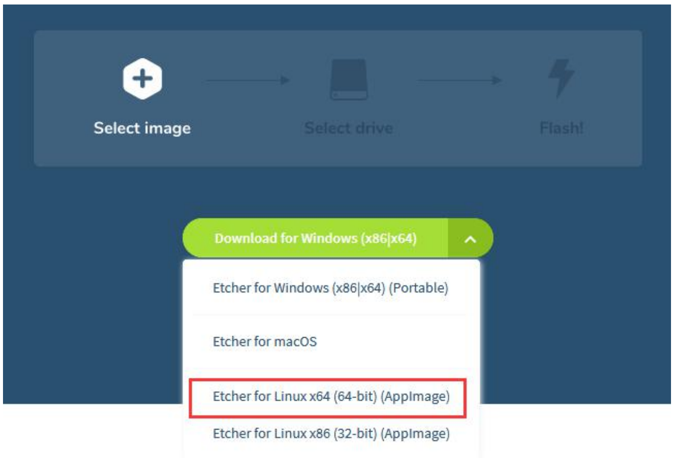
1) First, prepare a Micro card with 8GB or larger capacity. The transmission speed of the MicroSD card must be above class10. It is recommended to use a brand MicroSD card such as SanDisk.

2) Then use a card reader to insert the MicroSD card into the computer

3) Download balenaEtcher software, the download URL is

https://www.balena.io/etcher/

4) After entering the balenaEtcher download page, please select the Linux version of the software through the drop-down box to download



5) After downloading, use unzip to decompress. The decompressed balenaEtcher-1.5.109-x64.AppImage is the software needed to burn Linux image

```
test@test:~$ unzip balena-etcher-electron-1.5.109-linux-x64.zip
Archive:  balena-etcher-electron-1.5.109-linux-x64.zip
  inflating: balenaEtcher-1.5.109-x64.AppImage
test@test:~$ ls
balenaEtcher-1.5.109-x64.AppImage  balena-etcher-electron-1.5.109-linux-x64.zip
```

6) Download the Linux operating OS image file compression package you want to burn from the Orange Pi data download page, and then use the decompression software to decompress it. In the decompressed file, the file ending with ".img" is the operating OS image file. The size is generally above 1GB

a. The decompression command of the compressed package at the end of 7z is as follows:

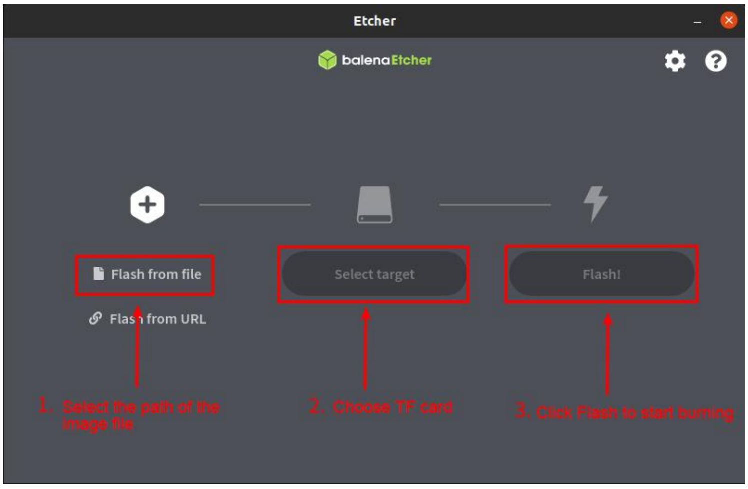
```
test@test:~$ 7z x Orangepizero2_2.0.8_ubuntu_bionic_server_linux4.9.170.7z
test@test:~$ ls Orangepizero2_2.0.8_ubuntu_bionic_server_linux4.9.170.*
Orangepizero2_2.0.8_ubuntu_bionic_server_linux4.9.170.7z
Orangepizero2_2.0.8_ubuntu_bionic_server_linux4.9.170.img.sha  # Checksum file
Orangepizero2_2.0.8_ubuntu_bionic_server_linux4.9.170.img   #Image file
```

7) After decompressing the image, you can first use the sha256sum -c *.sha command to calculate whether the checksum is correct. If the prompt is successful, it means that the downloaded image is correct, and you can safely flash it to the MicroSD card. If the checksum does not match, it means there is a problem with the downloaded image, please try to download again

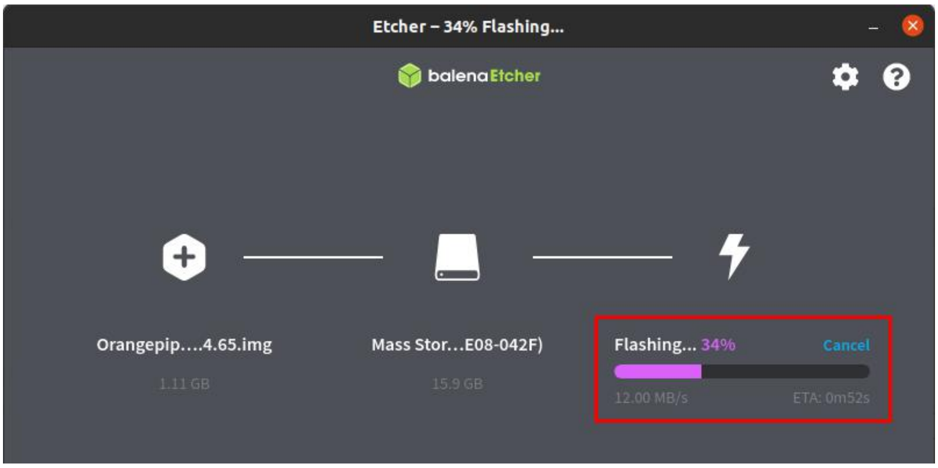
```
test@test:~$ sha256sum -c *.sha
Orangepizero2_2.0.8_ubuntu_bionic_server_linux4.9.170.img: success
```

8) Then double-click balenaEtcher-1.5.109-x64.AppImage on the graphical interface of Ubuntu PC to open balenaEtcher (no installation required), the opened interface is as shown in the figure below

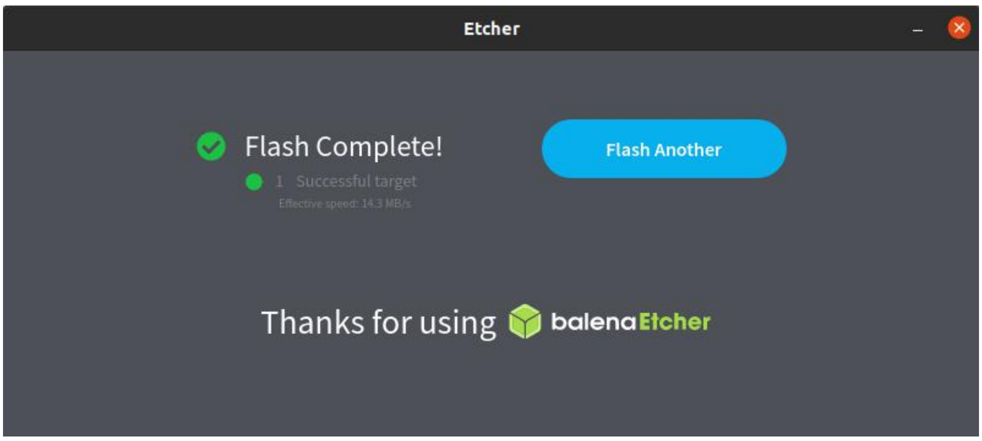
a. First, select the path of the Linux image file
b. Then select the device number of the TF card
c. Finally, click Flash to start burning



9) The writing process will prompt the writing speed and remaining time



10) After burning, the following interface will be displayed, then you can unplug the MicroSD card from the computer and insert it into the development board to start



2.5. Method of flashing Android firmware to MicroSD card

The Android image can only be flashed to MicroSD card using PhoenixCard

software under the Windows platform, but cannot be burned under Linux platform

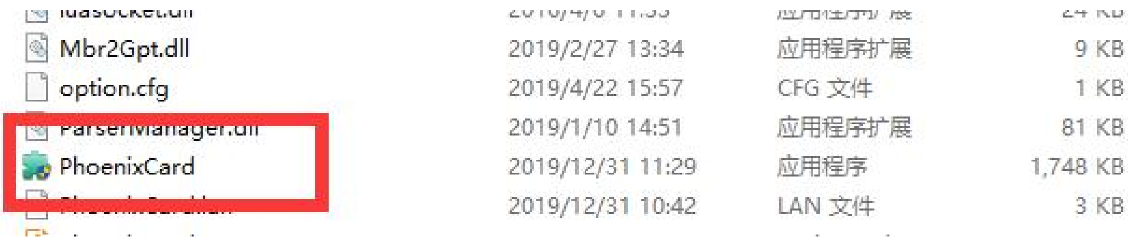
1) First, prepare a MicroSD card with 8GB or larger capacity. The transmission speed of the TF card must be above class10. It is recommended to use a brand MicroSD card such as SanDisk.

2) Then use a card reader to insert the MicroSD card into the computer

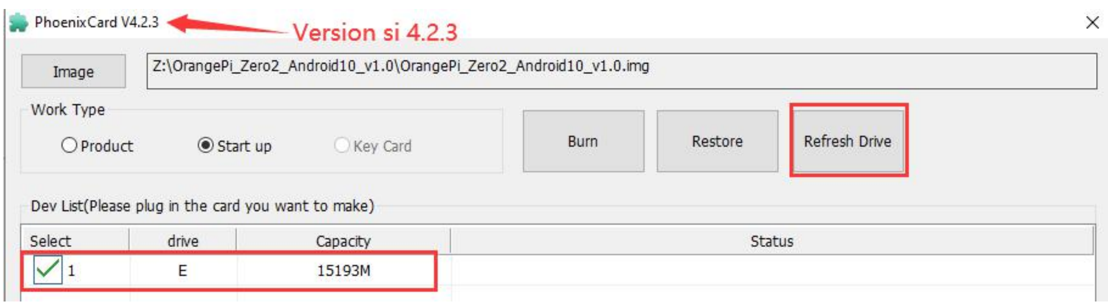
3) Download the Android 10 OS and PhoenixCard flashing tool from Orange Pi's data download page. Please make sure that the PhonenixCard tool version is PhonenixCard-v4.2.3. Tools below this version will have problems flashing to the Android 10 OS.

4) Use the decompression software to decompress the downloaded Android OS compressed package. In the decompressed file, the file ending with ".img" is the Android image file

5) Use decompression software to decompress PhonenixCard-v4.2.3.zip, this software does not need to be installed, you can find PhoenixCard in the decompressed folder and open it

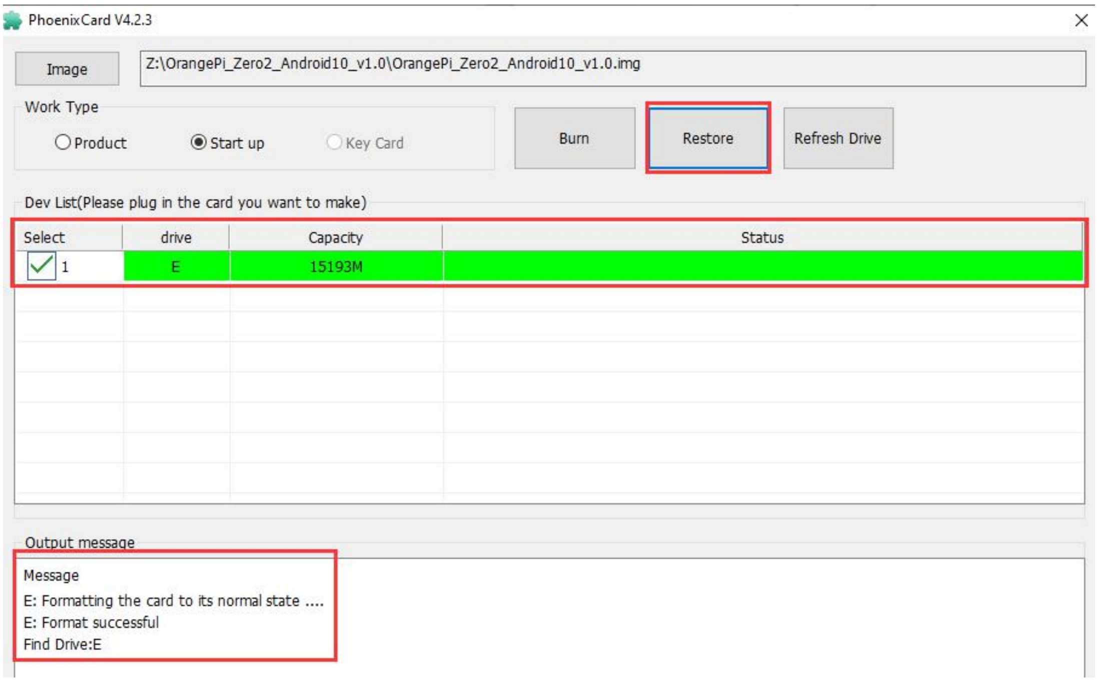


6) After opening PhoenixCard, if the MicroSD card is recognized normally, the drive letter and capacity of the MicroSD card will be displayed in the middle list. Please make sure that the displayed drive letter is consistent with the drive letter of the MicroSD card you want to burn. If there is no display, you can try to remove and insert the MicroSD card, or click the **Refresh Drive** letter button in PhoenixCard



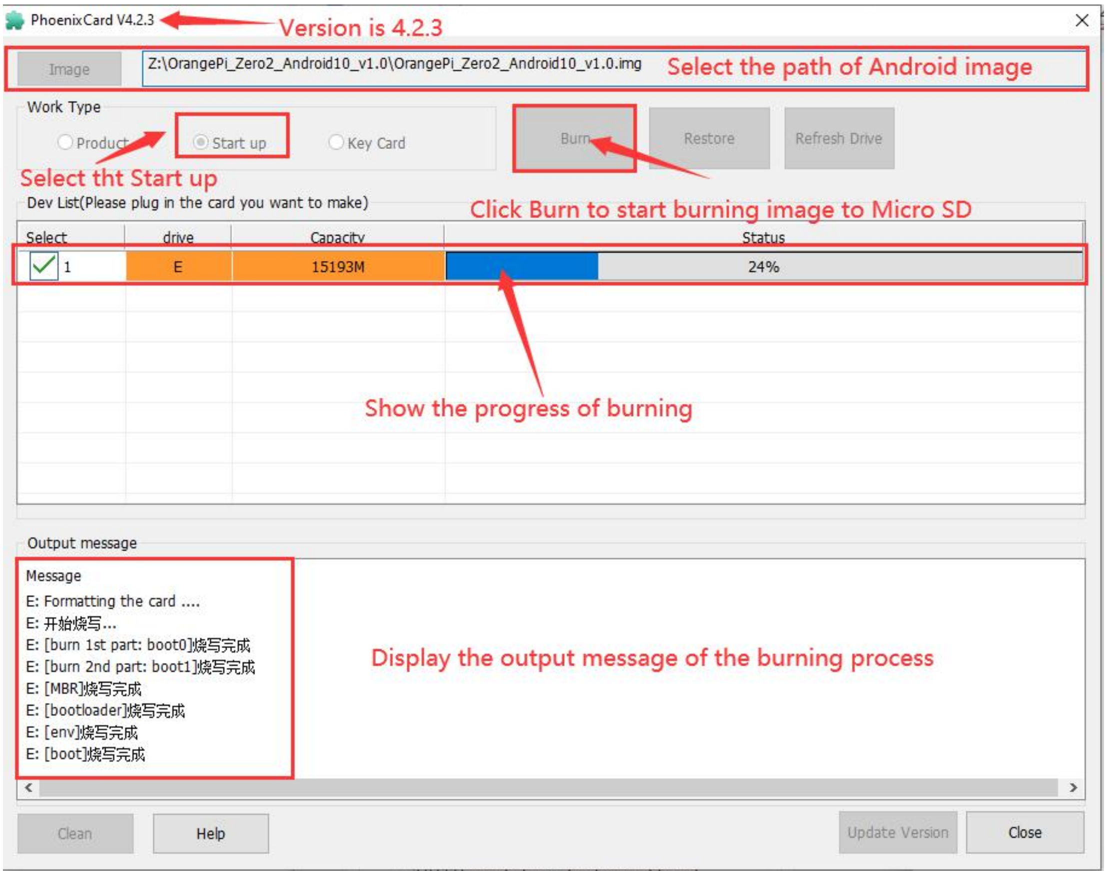
7) After confirming the drive letter, format the MicroSD card first, click the "Restore" button in PhoenixCard, or use the aforementioned SD Card Formatter to format the

MicroSD card

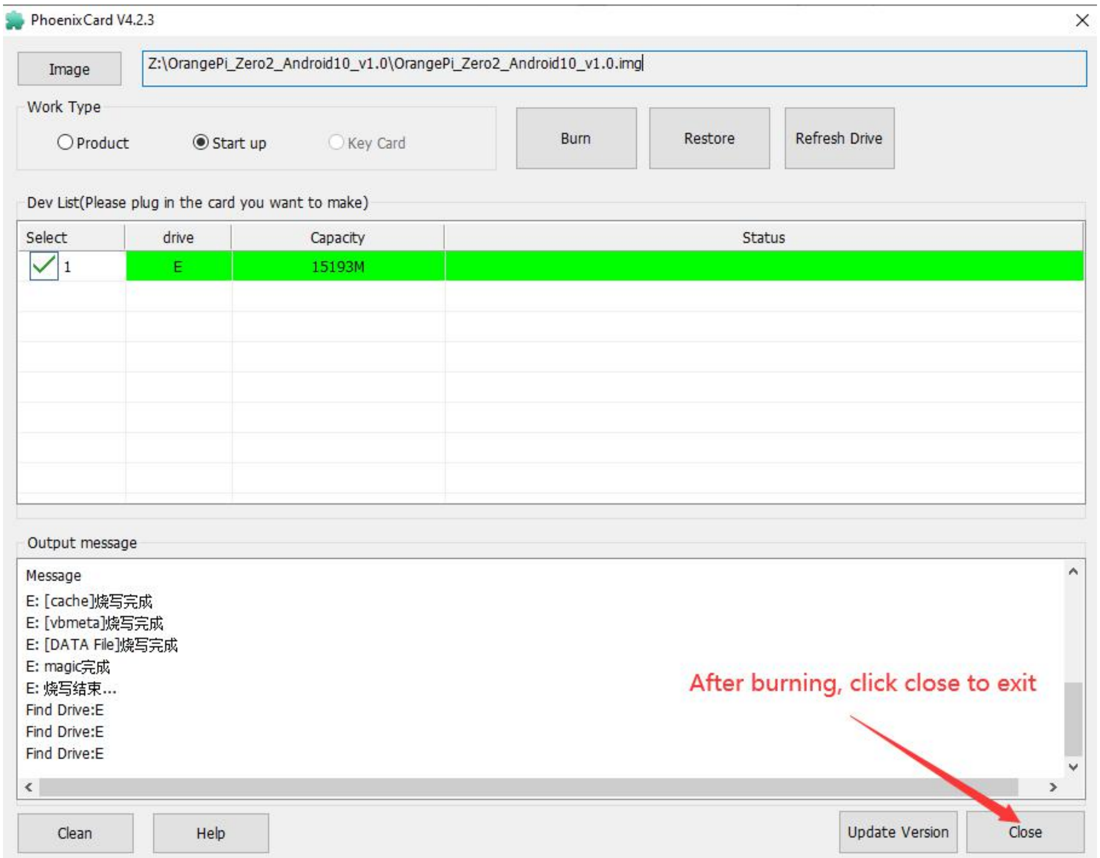


8) Then start to write the Android OS to MicroSD card

a. First, select the path of the Android image in the "**Image**" column
b. Select "**Start up**" in "Type of Making Card"
c. Then click the "**Burn**" button to start burning



9) After burning, the PhoenixCard will display as shown in the figure below. At this time, click the "**Close**" button to exit PhoenixCard, and then you can unplug the MicroSD card from the computer and insert it into the development board to start



2.6. Start the Orange Pi development board

1) Insert MicroSD card with well-flashed image into the MicroSD card slot of the Orange Pi development board
2) The development board has a Micro HDMI interface, you can connect the development board to a TV or HDMI display through a Micro HDMI to HDMI cable
3) Connect the USB mouse and keyboard to control the Orange Pi development board
4) The development board has an Ethernet port, which can be plugged into the network cable for Internet access
5) A high-quality power adapter with a 5V/2A (5V/3A is also available) USB Type C interface

a. Remember not to plug in the 12V power adapter, if you plug in the 12V power adapter, it will burn the development board

b. Many unstable phenomena during OS power-on and startup are caused by power supply problems, so a reliable power adapter and USB Type C data cable are very important

6) Then turn on the power adapter switch, if everything is normal, the HDMI display

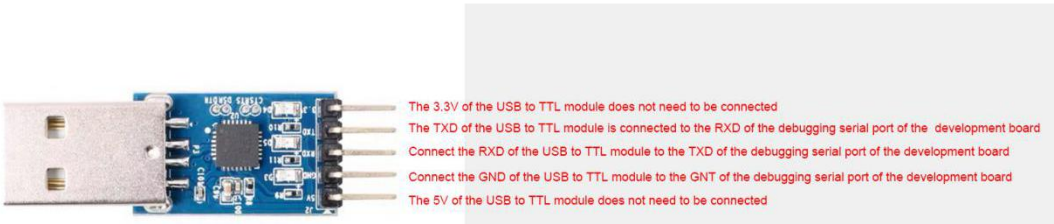
can see the OS startup screen at this time

7) If you want to view the output information of the OS through the debug serial port, please use the serial cable to connect the development board to the computer. For the connection method of the serial port, please refer to the section on the use of the debug serial port

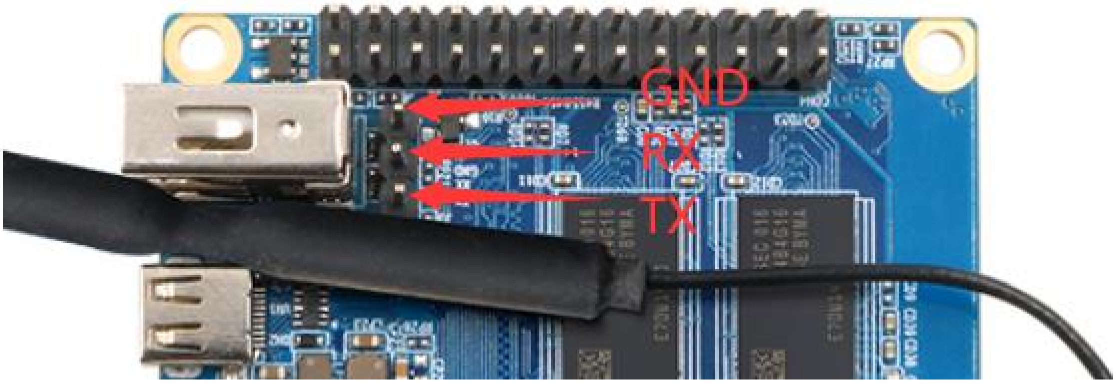
2.7. How to use the debug serial port?

2.7.1. Debug serial port connection instructions

1) First, you need to prepare a USB to TTL module. This module can be bought in Orange Pi stores. If there are other similar USB to TTL modules, you can also insert the USB end of the USB to TTL module into the USB port of the computer



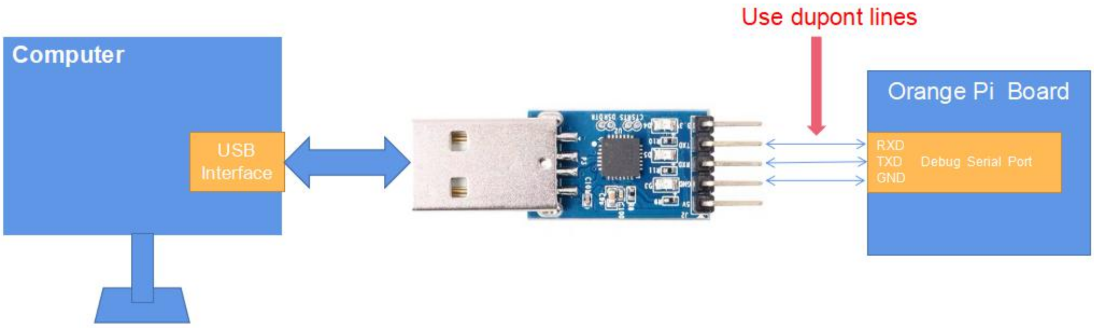
2) The corresponding relationship between the debug serial port GND, TX, and RX pins of the development board is shown in the figure below



3) The GND, TX, and RX pins of the USB to TTL module need to be connected to the debug serial port of the development board through a Dupont cable

a. Connect the GND of the USB to TTL module to the GND of the board
b. Connect the RX of the USB to TTL module to the TXD of the board
c. Connect the TX of the USB to TTL module to the RX of the board

4) The schematic diagram of connecting the USB to TTL module to the computer and the Orange Pi development board is shown below



Schematic diagram of connecting USB to TTL module
to computer and Orange Pi development board

2.7.2. How to use the debug serial port on the Ubuntu platform?

1) If the USB to TTL module is connected normally, you can see the corresponding device node name under /dev of Ubuntu PC, remember this node name, you will use it when setting up the serial port software later

```
test@test:~$ ls /dev/ttyUSB*
/dev/ttyUSB0
```

2) Many serial debugging tools that can be used under Linux, such as putty, minicom, etc. The following shows how to use putty

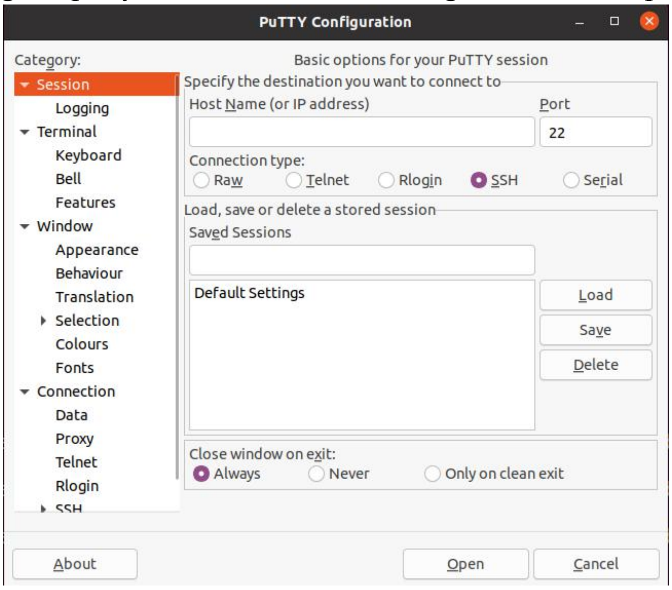
3) First, install putty on Ubuntu PC

```
test@test:~$ sudo apt update
test@test:~$ sudo apt install putty
```

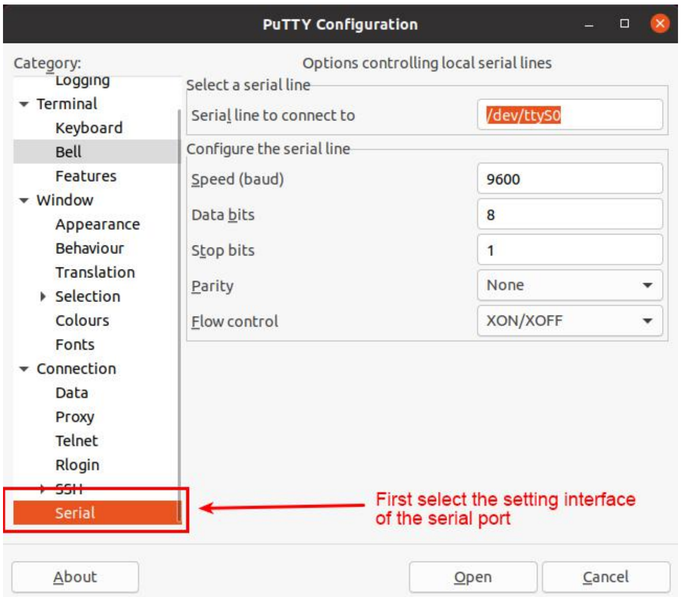
4) Then run putty, remember to add sudo permissions

```
test@test:~$ sudo putty
```

5) After executing the putty command, the following interface will pop up



6) First, select the setting interface of the serial port

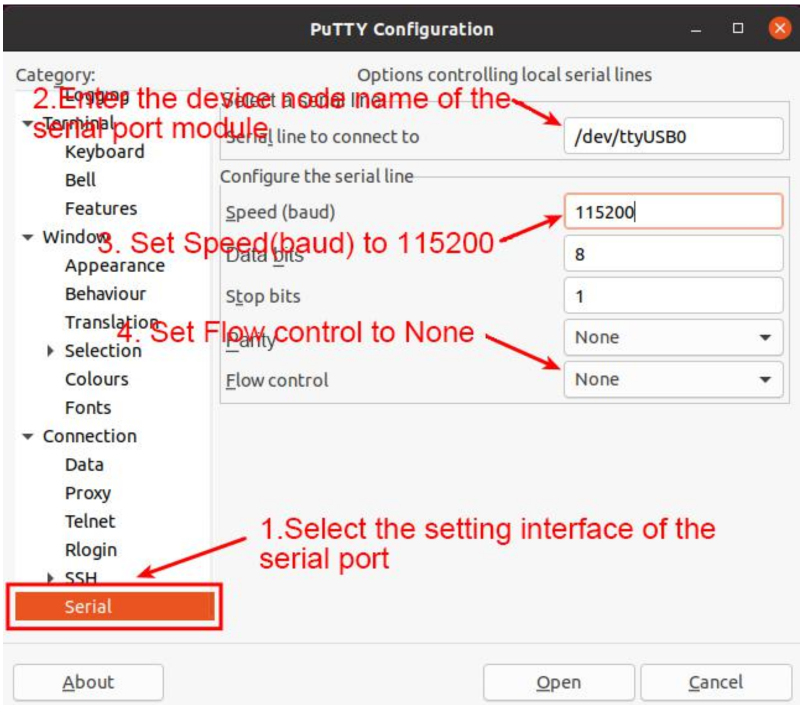


7) Then set the parameters of the serial port

a. Set the Serial line to connect to /dev/ttyUSB0 (modify to the corresponding node name, generally /dev/ttyUSB0)

b. Set Speed(baud) to 115200 (baud rate of the serial port)

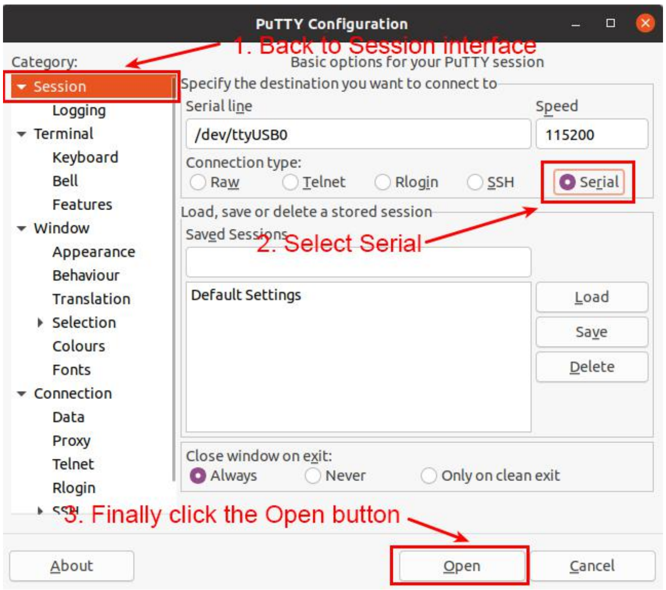
c. Set Flow control to None



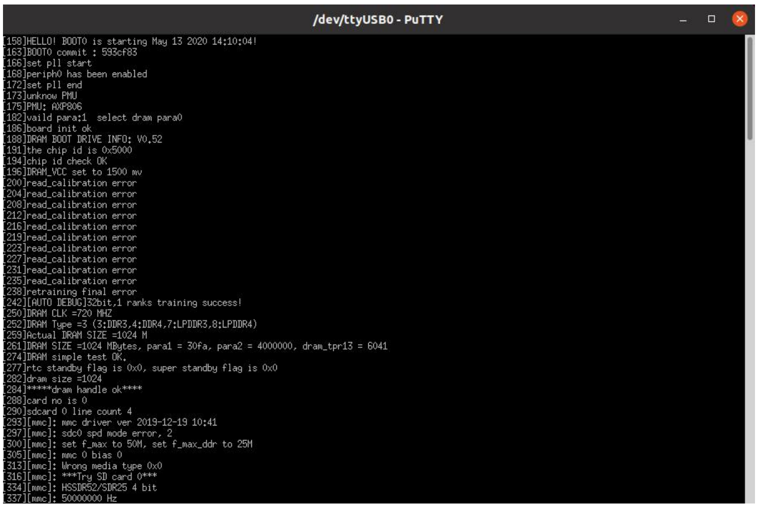
8)After setting the serial port setting interface, return to the Session interface

a. First, select the Connection type as Serial

b. Then click the Open button to connect to the serial port



9)After starting the development board, you can see the Log information output by the OS from the opened serial terminal



2.7.3. How to use the debug serial port on the Windows platform?

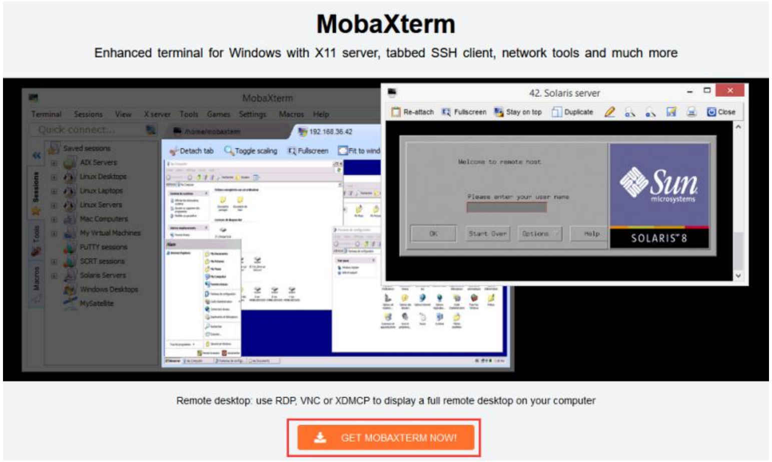
1) Many serial debugging tools that can be used under Windows, such as SecureCRT, MobaXterm, etc. The following shows how to use MobaXterm. This software has a free version and can be used without purchasing a serial number.

2) Download MobaXterm

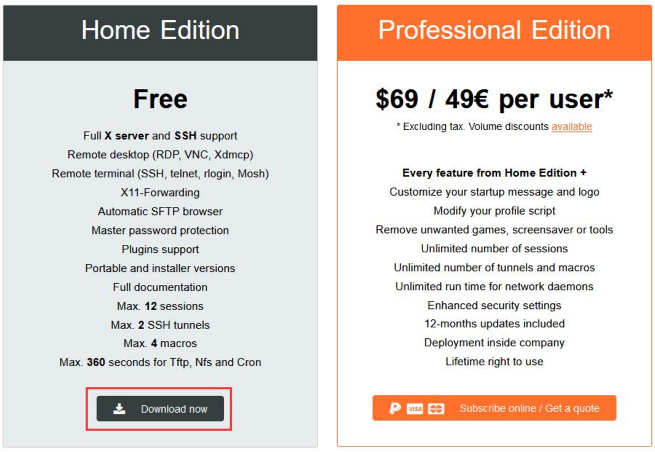
a. Download MobaXterm URL as follows

https://mobaxterm.mobatek.net/

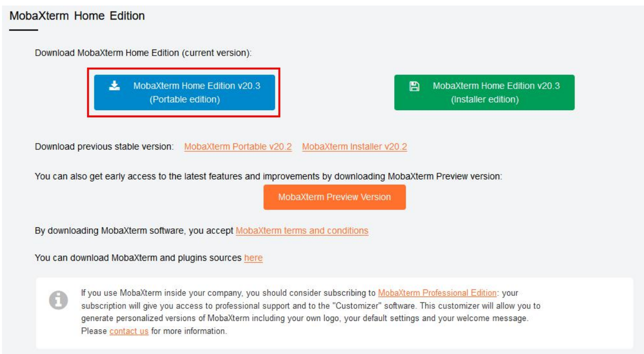
b. **After entering the MobaXterm download page, click GET XOBATERM NOW!**



c. Then choose to download the Home version



d. Then select the Portable version. After downloading, you don't need to install it, you can open it directly



3) After downloading, use the decompression software to decompress the downloaded compressed package, you can get the executable software of MobaXterm, and then double-click to open it

4) After opening the software, the steps to set the serial connection are as follows

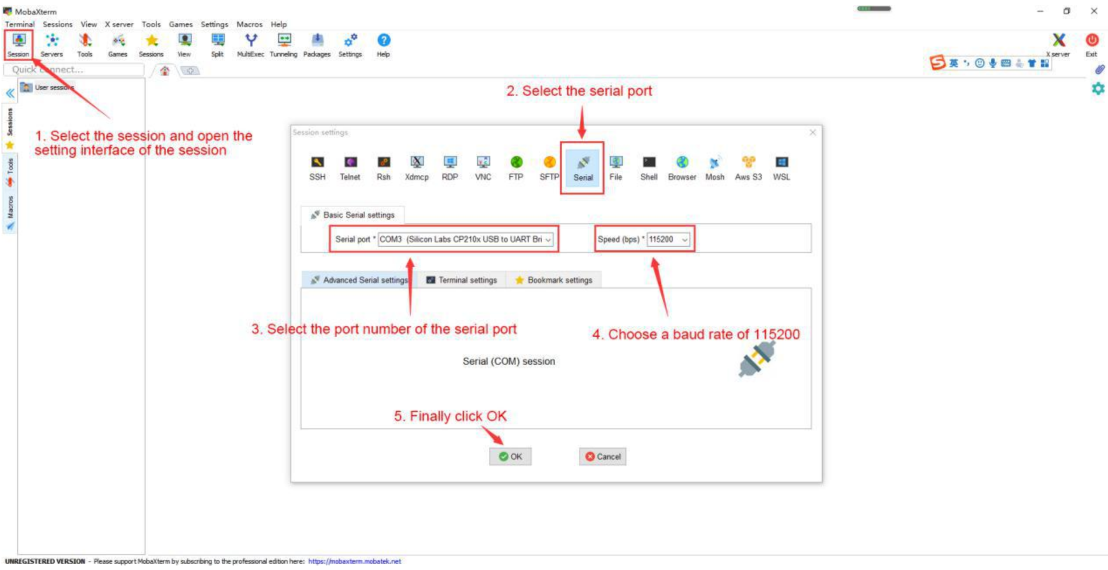
a. Open the session setting interface

b. Select the serial port type

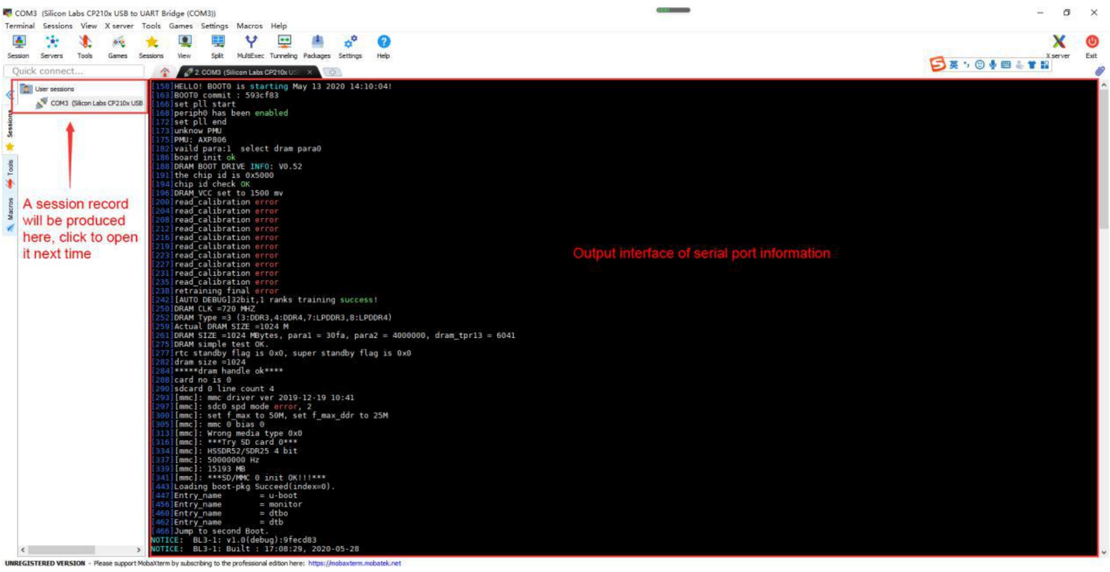
c. Select the port number of the serial port (choose the corresponding port number according to the actual situation), if you cannot see the port number, please use the 360 driver master to scan and install the driver for the USB to TTL serial chip

d. Select the baud rate of the serial port to be 115200

e. Finally click the "OK" button to complete the setting



5)After clicking the "OK" button, you will enter the following interface, and you can see the output information of the serial port after starting the development board



3. Linux OS instructions

3.1. Supported Linux distribution types and kernel versions

Release version	Kernel version	Server version	desktop version
Ubuntu 18.04	linux4.9	Support	Support
Ubuntu 20.04	linux4.9	Support	Support
Debian 10	linux4.9	Support	Support

3.2. linux4.9 kernel driver adaptation status

Orange Pi Zero 2 currently only supports the linux 4.9 version of the kernel, and the driver adaptation is shown in the table below

Function	Status
HDMI video	OK
HDMI Audio	OK
USB2.0 x 3	OK
TF card boot	OK
Network card	OK
IR receiver	OK
WIFI	OK
BT	OK
Headphone audio	OK
USB camera	OK
LED light	OK
26pin GPIO	OK
I2C	OK
SPI	OK
UART	OK
Temperature Sensor	OK
Hardware watchdog	OK
Mali GPU	NO
Video codec	NO

3.3. Linux OS default login account and password

Account	Password
root	orangepi
orangepi	orangepi

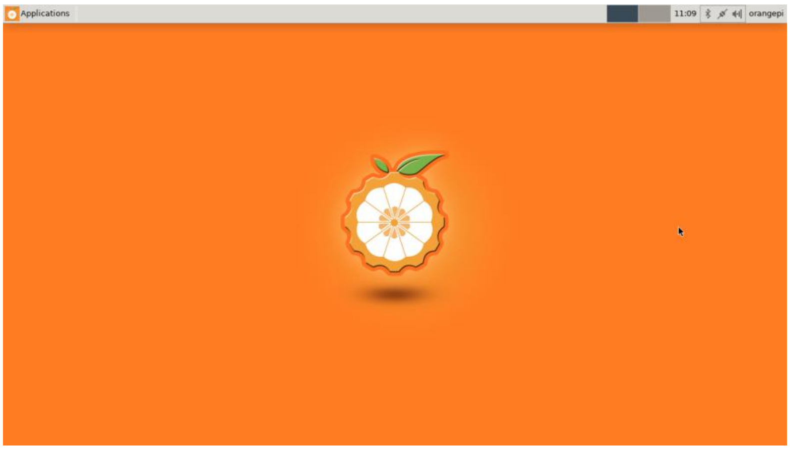
3.4. Onboard LED light display description

	Green Light	Red Light
u-boot startup phase	Turn off	bright

Kernel boot to enter system	bright	Turn off
GPIO port	PC13	PC12

3.5. Instructions for the automatic login of Linux desktop version OS

1) The desktop version of the OS will automatically log in to the desktop after it is started by default, without entering a password



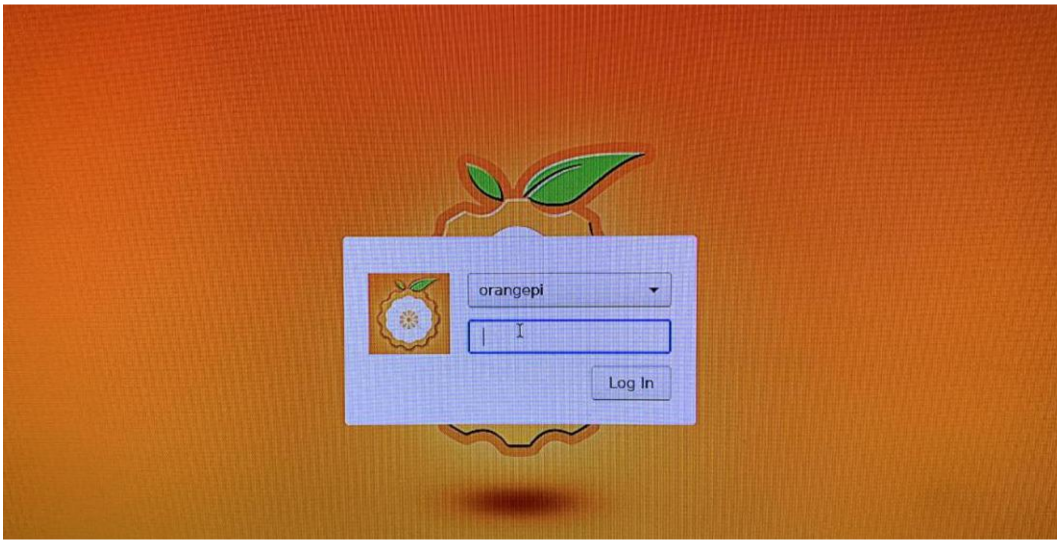
2) Modify the configuration in /etc/lightdm/lightdm.conf.d/22-orangepi-autologin.conf to prohibit the desktop version OS from automatically logging in to the desktop. The modification command is as follows, or you can open the configuration file to modify directly

```
root@orangepis:~# sed -i "s/autologin-user=orangepi/#autologin-user=orangepi/" /etc/lightdm/lightdm.conf.d/22-orangepi-autologin.conf
```

3) The configuration of /etc/lightdm/lightdm.conf.d/22-orangepi-autologin.conf after modification is as follows

```
root@orangepis:~# cat /etc/lightdm/lightdm.conf.d/22-orangepi-autologin.conf
[Seat:*]
#autologin-user=orangepi
autologin-user-timeout=0
user-session=xfce
```

4) Then restart the OS and a login dialog box will appear, at this time you need to enter a password to enter the OS



3.6. Start the rootfs in the auto-expanding TF card for the first time

1) **When the TF card starts the Linux OS for the first time, it will call the orangepi-resize-filesystem script through the orangepi-resize-filesystem.service systemd service to automatically expand the rootfs, so there is no need to manually expand**

2) After logging in to the OS, you can use the df -h command to check the size of rootfs. If it is consistent with the actual capacity of the TF card, it means that the automatic expansion is running correctly

```
root@orangepi:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            430M     0  430M   0% /dev
tmpfs           100M  5.6M   95M   6% /run
/dev/mmcblk0p1   15G  915M   14G   7% /
tmpfs           500M     0  500M   0% /dev/shm
```

3.7. How to modify the linux log level (loglevel)?

1) The log level of the Linux OS is set to 1 by default. When using the serial port to view the startup information, the kernel output log is as follows, basically all shielded

```
Starting kernel ...

Uncompressing Linux... done, booting the kernel.

Orange Pi 2.0.8 Bionic ttyS0
orangepi login:
```

2) When there is a problem with the OS startup, you can use the following method to modify the value of log level, to print more log information to the serial port display, which is convenient for debugging

```
root@orangepi:~# sed -i "s/verbosity=1/verbosity=7/" /boot/orangepiEnv.txt
root@orangepi:~# sed -i "s/console=both/console=serial/" /boot/orangepiEnv.txt
```

3) The above commands are actually to set the variables in /boot/orangepiEnv.txt. After setting, you can open /boot/orangepiEnv.txt to check

```
root@orangepi:~# cat /boot/orangepiEnv.txt
verbosity=7
bootlogo=false
console=serial
```

4) Then restart the development board, the output information of the kernel will be printed to the serial port for output

3.8. Ethernet port test

1) First insert the network cable into the Ethernet interface of the development board, and ensure that the network is unblocked

2) After the OS starts, it will automatically assign an IP address to the Ethernet card through DHCP

3) The command to view the IP address is as follows

```
root@orangepi:~# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.47  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::e56:c34d:62f0:8d6e  prefixlen 64  scopeid 0x20<link>
        ether 02:81:3e:a8:58:d8  txqueuelen 1000  (Ethernet)
        RX packets 2165  bytes 177198 (177.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 312  bytes 40435 (40.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 39
```

4) The command to test network connectivity is as follows

```
root@orangepi:~# ping www.baidu.com -I eth0
PING www.a.shifen.com (14.215.177.38) from 192.168.1.12 eth0: 56(84) bytes of data.
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=1 ttl=56 time=6.74 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=2 ttl=56 time=6.80 ms
```

```
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=3 ttl=56 time=6.26 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=4 ttl=56 time=7.27 ms
^C
--- www.a.shifen.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 6.260/6.770/7.275/0.373 ms
```

3.9. SSH remote login development board

Linux OS have SSH remote login enabled by default, and allow root users to log in to the OS. Before ssh login, you need to make sure that the Ethernet or wifi network is connected, and then use the ifconfig command or check the router to obtain the IP address of the development board

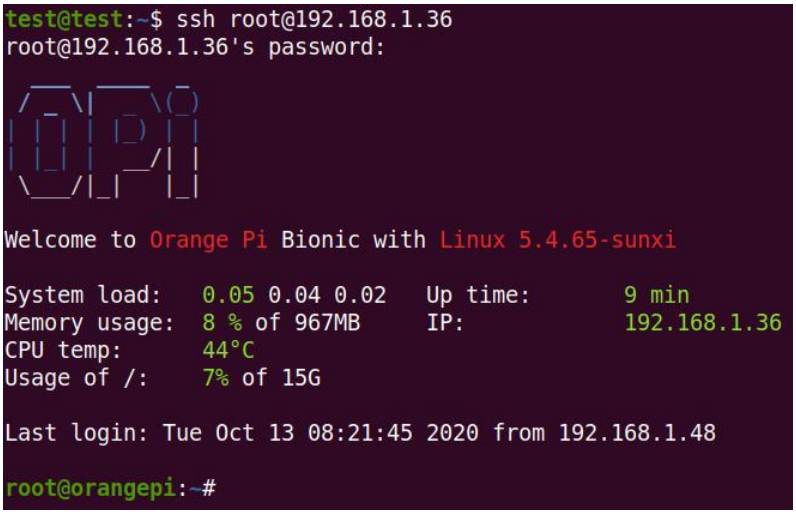
3.9.1. SSH remote login development board under Ubuntu

1) Get the IP address of the development board

2) Then you can log in to the Linux OS remotely through the ssh command

```
test@test:~$ ssh root@192.168.1.36          (Need to be replaced with the IP address of the development board)
root@192.168.1.36's password:    （Enter the password here, the default password is orangepi）
```

3) The display after successfully logging in to the OS is shown in the figure below



4) If the following error is prompted during ssh login

```
test@test:~$ ssh root@192.168.1.36
Connection reset by 192.168.1.149 port 22
lost connection
```

You can enter the following command on the development board and try to connect

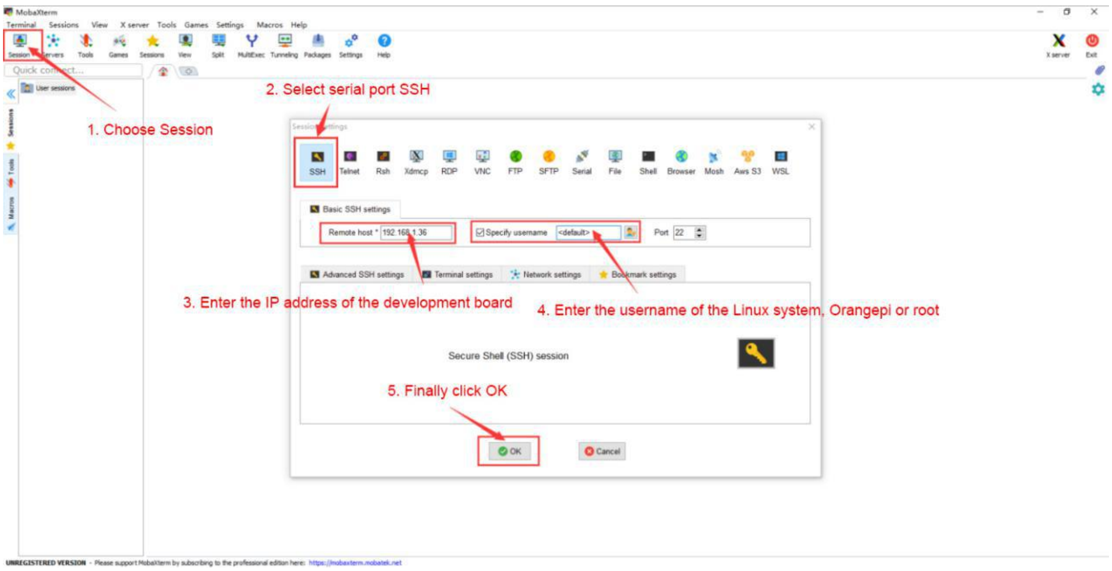
```
root@orangepi:~# rm /etc/ssh/ssh_host_*
root@orangepi:~# dpkg-reconfigure openssh-server
```

3.9.2. SSH remote login development board under Windows

1) First, get the IP address of the development board

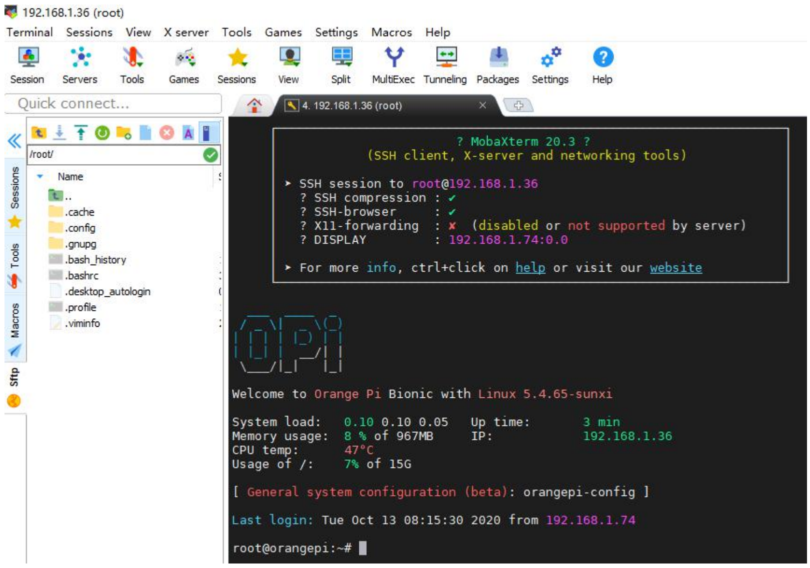
2) MobaXterm can be used to remotely log in to the development board under windows, first create a new ssh session

a. Open Session
b. Then select SSH in Session Setting
c. Then enter the IP address of the development board in Remote host
d. Then enter the username root or orangepi of the Linux OS in Specify username
e. Finally, click OK



3)Then you will be prompted to enter a password, the default passwords for root and orangepi users are orangepi

4)The display after successfully logging in to the OS is shown in the figure below



3.10. HDMI display test

1) Use Micro HDMI to HDMI cable to connect Orange Pi development board and HDMI display

2) If the HDMI display has image output after starting the Linux OS, it means that the HDMI interface is working normally

3.11. HDMI resolution setting

1) There is a disp_mode variable in the /boot/orangepiEnv.txt of the Linux OS, which can be used to set the HDMI resolution. The default resolution of the Linux OS is 1080p60

```
root@orangepi:/boot# cat orangepiEnv.txt
verbosity=1
```

```
console=both
disp_mode=1080p60
```

2) The disp_mode variable supports setting values as shown in the following table

disp_mode supported value	HDMI Resolution	HDMI refresh rate
480i	720x480	60
576i	720x480	50
480p	720x480	60
576p	720x576	60
720p50	1280x720	50
720p60	1280x720	60
1080i50	1920x1080	50
1080i60	1920x1080	60
1080p24	1920x1080	24
1080p50	1920x1080	50
1080p60	1920x1080	60

3) Modify the value of the disp_mode variable to the resolution you want to output, and then restart the OS, HDMI will display the set resolution

3.12. How to modify the width and height of Framebuffer?

1) There are two variables fb0_width and fb0_height in the /boot/orangepiEnv.txt of the Linux OS, which can be used to set the width and height of the Framebuffer. The Linux OS defaults to fb0_width=1280, fb0_height=720

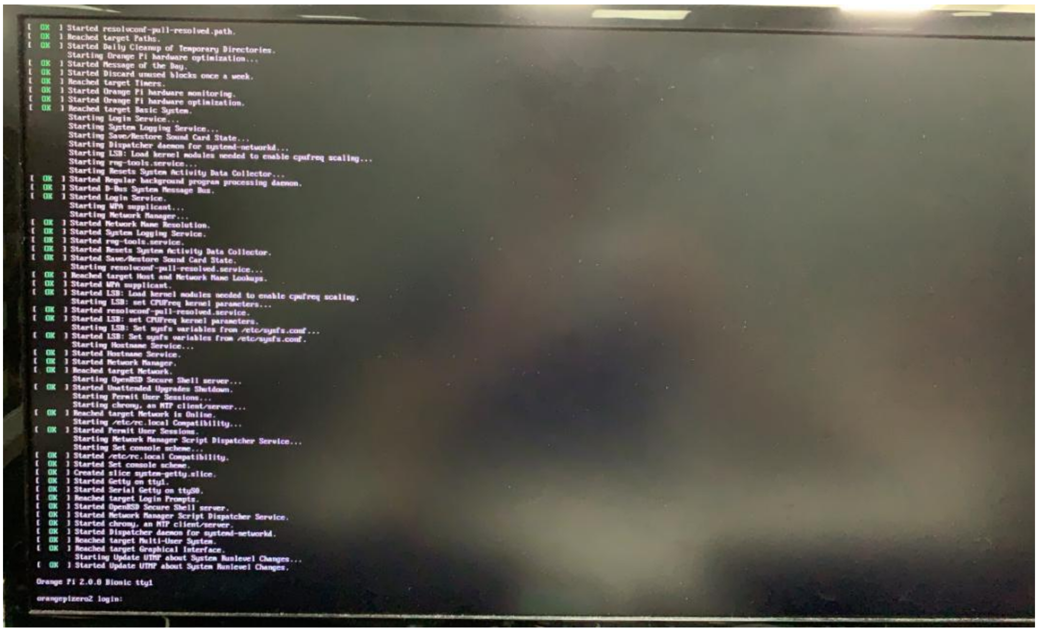
```
root@orangepi:~# cat /boot/orangepiEnv.txt
verbosity=1
console=both
disp_mode=720p60
fb0_width=1280
fb0_height=720
```

2) The reference values corresponding to different resolutions of fb0_width and fb0_height are as follows

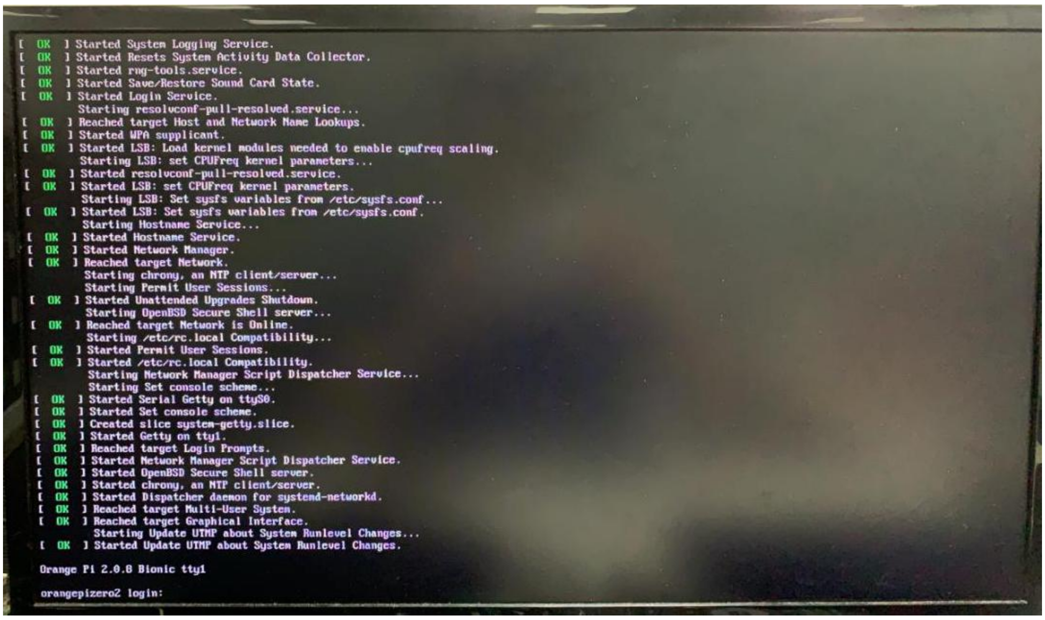
HDMI resolution	fb0_width	fb0_height
480p	720	480
576p	720	576
720p	1280	720
1080p	1920	1080

3) Under the same HDMI resolution, the display of different fb0_width and fb0_height is as follows

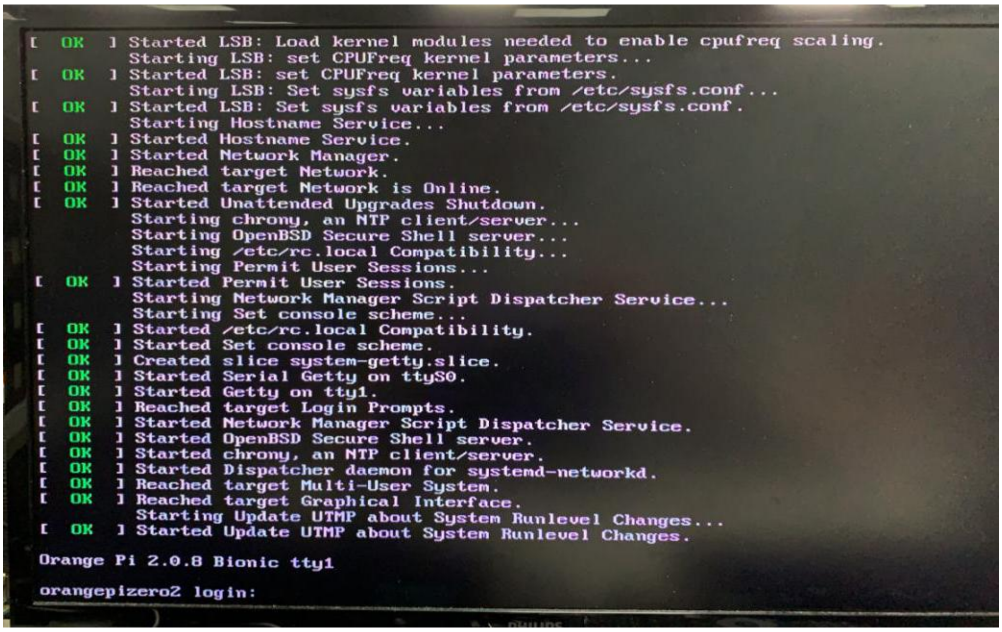
a. HDMI resolution is 1080p60, fb0_width and fb0_height are 1920x1080



b. HDMI resolution is 1080p60, fb0_width and fb0_height are 1280x720 display



c. HDMI resolution is 1080p60, fb0_width and fb0_height are 720x576



d. HDMI resolution is 1080p60, fb0_width and fb0_height are 720x480 display

3.13. WIFI connection test

3.13.1. Test method of Linux server version image

1) Log in to the Linux OS first, there are three ways

a. If the development board is connected to the network cable, you can log in to the Linux OS remotely via SSH

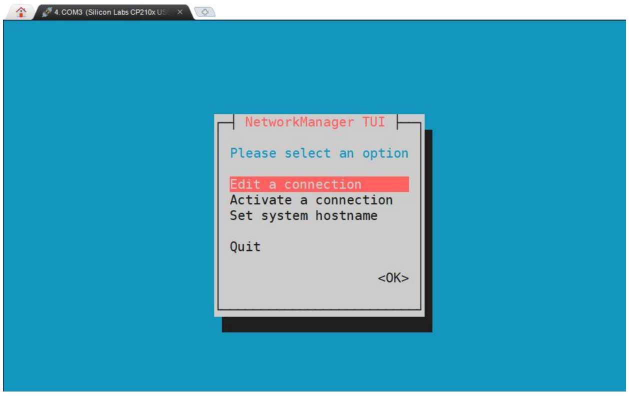
b. If the debug serial port is connected, you can use the serial terminal to log in to the Linux OS (use MobaXterm for the serial port software, and the graphical interface cannot be displayed using minicom)

c. If you connect the development board to the HDMI display, you can log in to the Linux OS through the HDMI display terminal

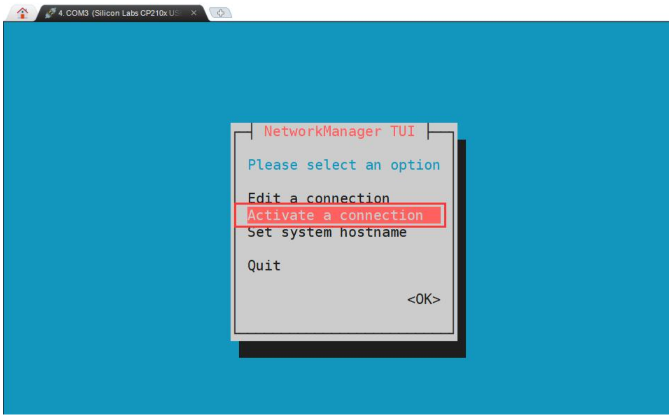
2)Then enter nmtui in the command line to open the wifi connection interface

```
root@orangepi:~# nmtui
```

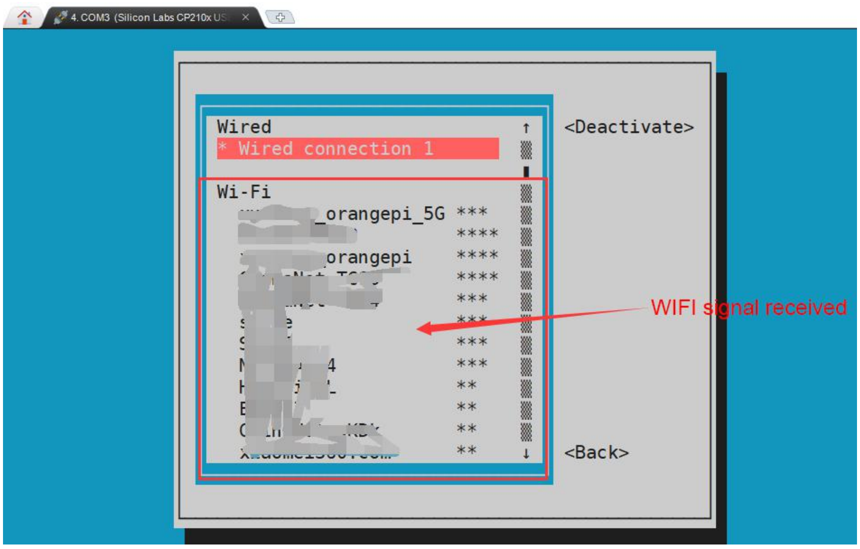
3)Enter nmtui to open the interface as shown below



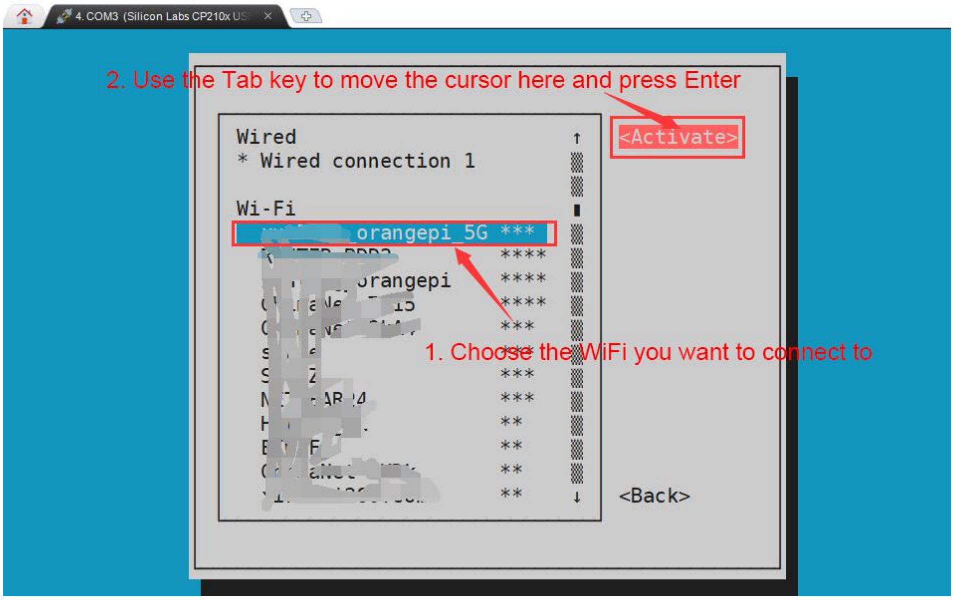
4)Select Activate a connect and press Enter



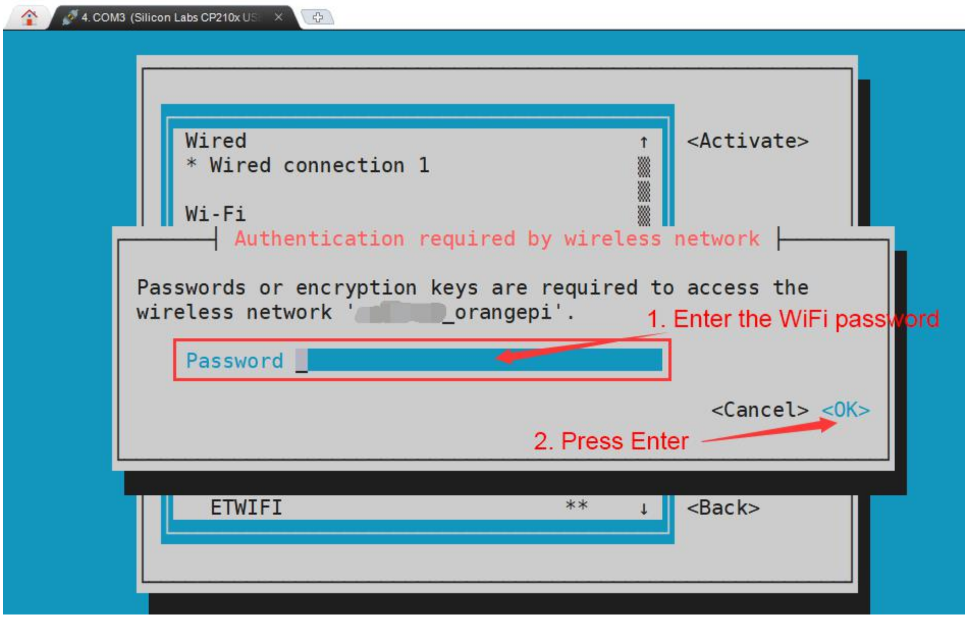
5)Then you can see all the searched WIFI hotspots



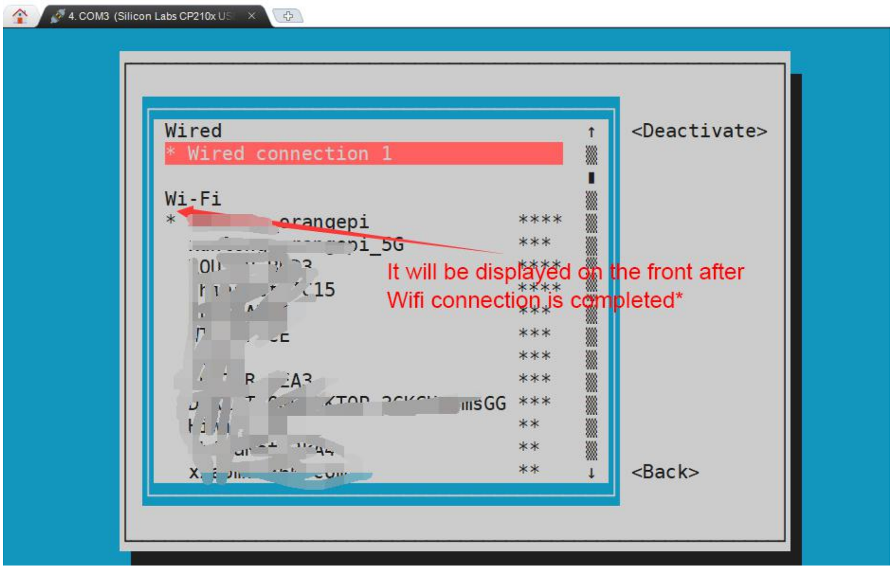
6)Select the WIFI hotspot you want to connect to, then use the Tab key to position the cursor on Activate and press Enter



7)Then a dialog box for entering the password will pop up, enter the corresponding password in Password and press Enter to start connecting to WIFI



8)After the WIFI connection is successful, a "*" will be displayed before the connected WIFI name



9)The wifi IP address can be viewed through the ifconfig command

```
root@orangepi:~# ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.49  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::76bb:f67d:ef98:2f9a  prefixlen 64  scopeid 0x20<link>
        ether 12:81:3e:a8:58:d8  txqueuelen 1000  (Ethernet)
        RX packets 185  bytes 109447 (109.4 KB)
        RX errors 0  dropped 61  overruns 0  frame 0
        TX packets 27  bytes 14783 (14.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

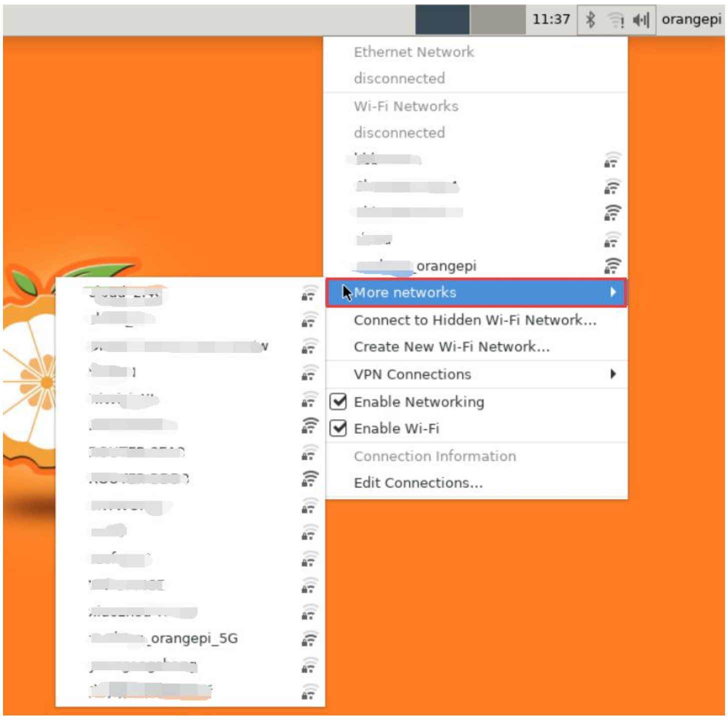
10)Use the ping command to test the connectivity of the wifi network

```
root@orangepipcplus:~# ping www.orangepi.org -I wlan0
PING www.orangepi.org (182.92.236.130) from 192.168.1.49 wlan0: 56(84) bytes of
data.
64 bytes from 182.92.236.130 (182.92.236.130): icmp_seq=1 ttl=52 time=43.5 ms
64 bytes from 182.92.236.130 (182.92.236.130): icmp_seq=2 ttl=52 time=41.3 ms
64 bytes from 182.92.236.130 (182.92.236.130): icmp_seq=3 ttl=52 time=44.9 ms
64 bytes from 182.92.236.130 (182.92.236.130): icmp_seq=4 ttl=52 time=45.6 ms
64 bytes from 182.92.236.130 (182.92.236.130): icmp_seq=5 ttl=52 time=48.8 ms
^C
--- www.orangepi.org ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 41.321/44.864/48.834/2.484 ms
```

3.13.2. Test method of Linux desktop version image

1) Click the network configuration icon in the upper right corner of the desktop (please do not connect the network cable when testing WIFI)

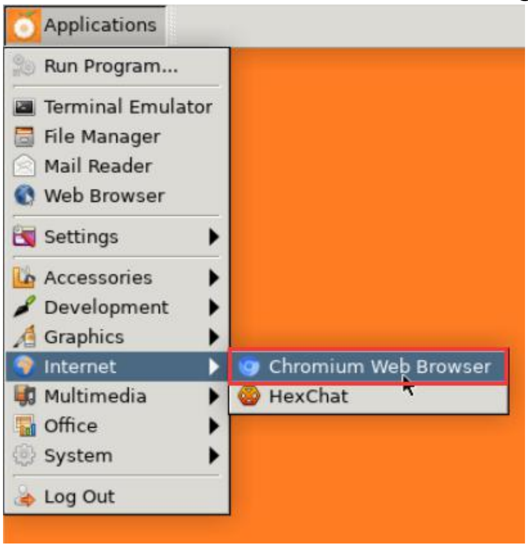
2) Click More networks in the pop-up drop-down box to see all scanned WIFI hotspots, and then select the WIFI hotspot you want to connect to



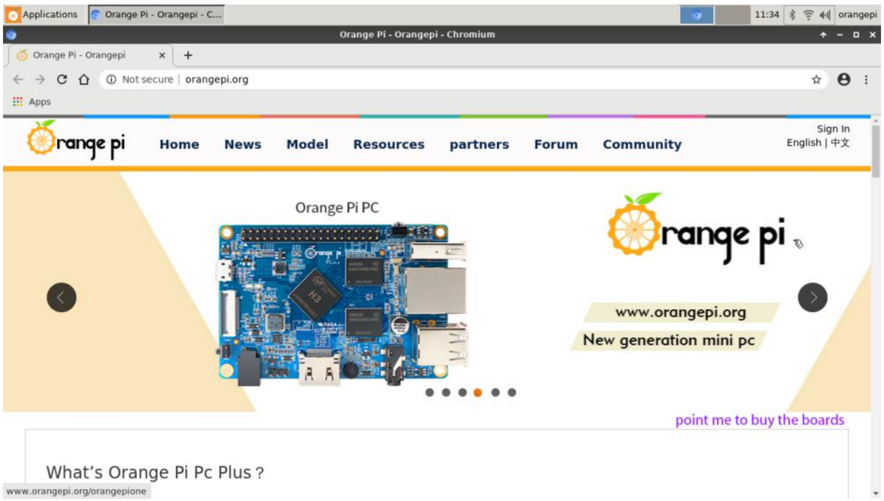
3) Then enter the password of the WIFI hotspot, and then click Connect to start connecting to WIFI

4) After connecting to the WIFI, you can open the browser to check whether you can

surf the Internet. The entrance of the browser is shown in the figure below



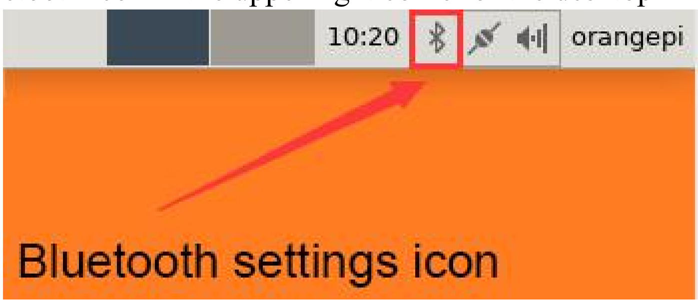
5) After opening the browser, if you can see the page of the OrangePi website, or you can open other web pages, the WIFI connection is normal



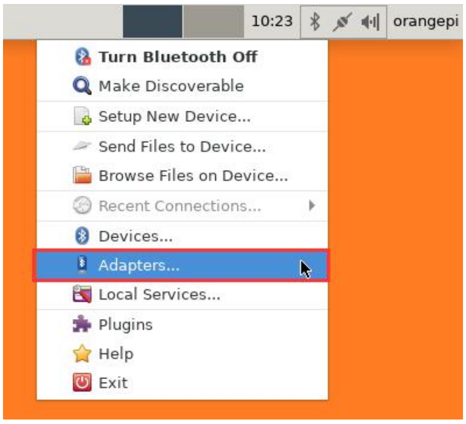
3.14. How to use Bluetooth？

3.14.1. Test method of desktop version image

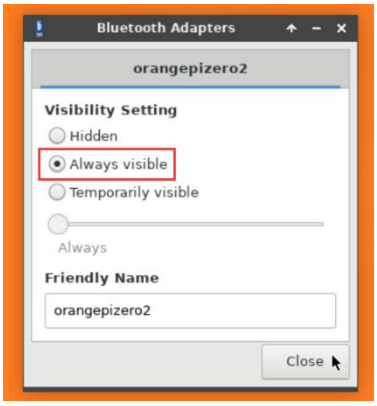
1) Click the Bluetooth icon in the upper right corner of the desktop



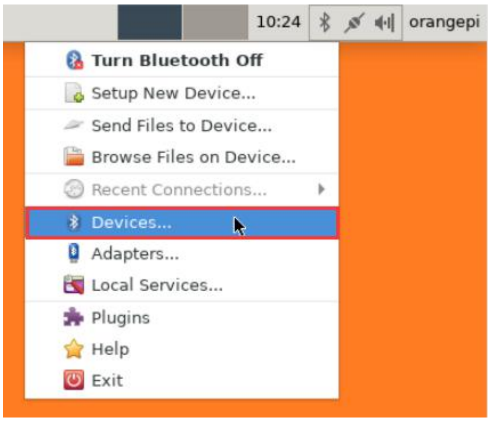
2) Then select the adapter



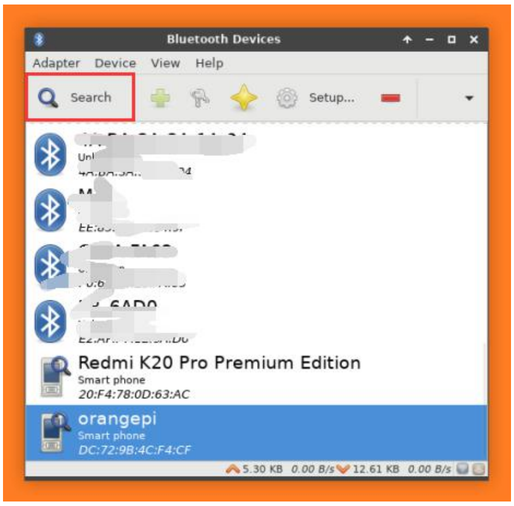
3) Set Visibility Setting to Always visible in the Bluetooth adapter setting interface, and then click close to close



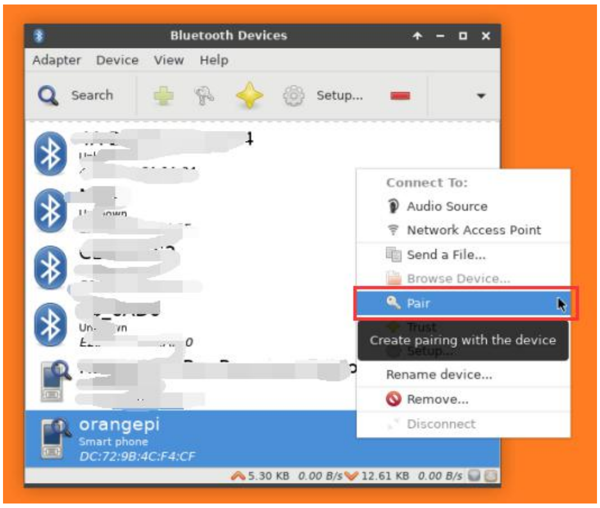
4) Then open the configuration interface of the Bluetooth device



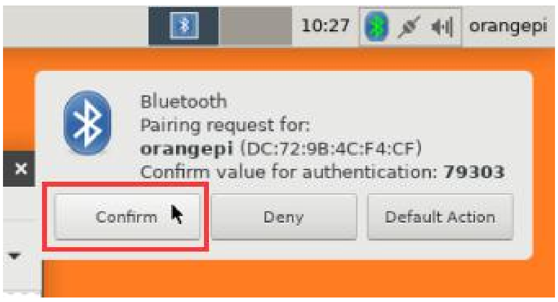
5) Click Search to start scanning surrounding Bluetooth devices



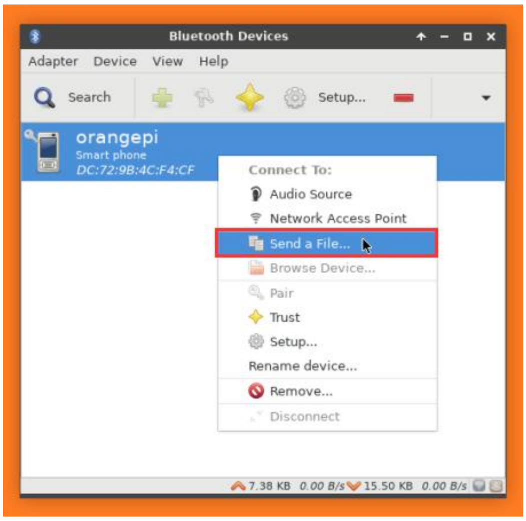
6) Then select the Bluetooth device you want to connect to, and then click the right mouse button to pop up the operation interface of the Bluetooth device. Select Pair to start pairing. Here is a demonstration of pairing with an Android phone



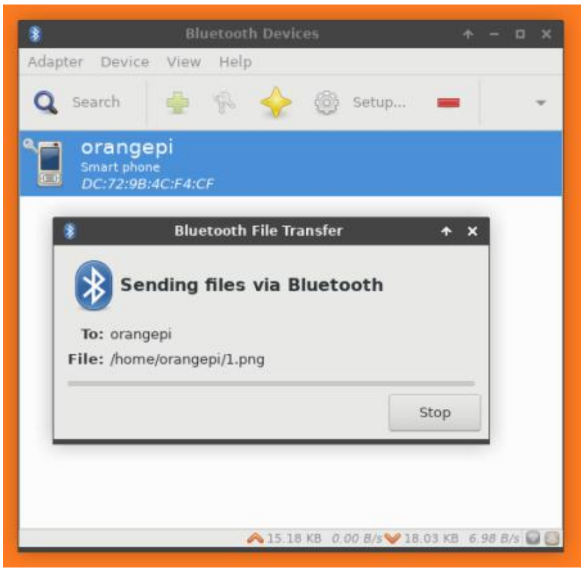
7) When pairing with a mobile phone, a pairing confirmation box will pop up in the upper right corner of the desktop, select Confirm to confirm. At this time, the mobile phone also needs to be confirmed



8) After pairing with the phone, you can select the paired Bluetooth device, then right-click and select Send a File to start sending a picture to the phone



9) The interface for sending pictures is as follows



3.14.2. How to use the server version image?

1) When the linux OS starts, the aw859a-bluetooth.service service will be run to initialize the Bluetooth device. After entering the OS, you can use the hciconfig command to check whether there is a Bluetooth device node. If it exists, the Bluetooth initialization is normal

```
root@orangepi:~# hciconfig -a
hci0:   Type: Primary  Bus: UART
        BD Address: 10:11:12:13:14:15  ACL MTU: 1021:8  SCO MTU: 240:3
        UP RUNNING
        RX bytes:646 acl:0 sco:0 events:37 errors:0
        TX bytes:2650 acl:0 sco:0 commands:37 errors:0
        Features: 0xbf 0xff 0x8d 0xfe 0xdb 0x3d 0x7b 0xc7
        Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
        Link policy:
```

```
Link mode: SLAVE ACCEPT
Name: 'orangepizero2'
Class: 0x000000
Service Classes: Unspecified
Device Class: Miscellaneous,
HCI Version: 5.0 (0x9)  Revision: 0x400
LMP Version: 5.0 (0x9)  Subversion: 0x400
Manufacturer: Spreadtrum Communications Shanghai Ltd (492)
```

2) Use bluetoothctl to scan for Bluetooth devices

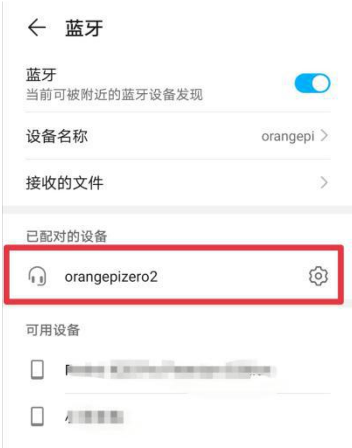
```
root@orangepi:~# bluetoothctl
[NEW] Controller 10:11:12:13:14:15 orangepizero2 [default]
Agent registered
[bluetooth]# power on          # Enable controller
Changing power on succeeded
[bluetooth]# discoverable on    # Set the controller to be discoverable
Changing discoverable on succeeded
[CHG] Controller 10:11:12:13:14:15 Discoverable: yes
[bluetooth]# pairable on   # Set the controller to be pairable
Changing pairable on succeeded
[bluetooth]# scan on       # Start scanning for surrounding Bluetooth devices
Discovery started
[CHG] Controller 10:11:12:13:14:15 Discovering: yes
[NEW] Device 76:60:79:29:B9:31 76-60-79-29-B9-31
[NEW] Device 9C:2E:A1:42:71:11 Xiaomi mobile phone
[NEW] Device DC:72:9B:4C:F4:CF orangepi
[bluetooth]# scan off        # After scanning the Bluetooth device you want to connect, you can turn off the scan, and then write down the MAC address of the Bluetooth device. The Bluetooth device tested here is an Android phone, the Bluetooth name is orangepi, and the corresponding MAC address is DC:72:9B:4C :F4:CF
Discovery stopped
[CHG] Controller 10:11:12:13:14:15 Discovering: no
[CHG] Device DC:72:9B:4C:F4:CF RSSI is nil
```

3) After scanning the device you want to pair, you can pair it. Pairing requires the MAC address of the device

```
[bluetooth]# pair DC:72:9B:4C:F4:CF     # Use the scanned MAC address of the Bluetooth device for pairing
Attempting to pair with DC:72:9B:4C:F4:CF
[CHG] Device DC:72:9B:4C:F4:CF Connected: yes
Request confirmation
[leeb1m[agent] Confirm passkey 764475 (yes/no): yes  # Enter yes here, you also need to confirm on the phone
```

```
[CHG] Device DC:72:9B:4C:F4:CF Modalias: bluetooth:v010Fp107Ed1436
[CHG] Device DC:72:9B:4C:F4:CF UUIDs: 0000046a-0000-1000-8000-00805f9b34fb
[CHG] Device DC:72:9B:4C:F4:CF ServicesResolved: yes
[CHG] Device DC:72:9B:4C:F4:CF Paired: yes
Pairing successful     #Prompt for successful pairing
[CHG] Device DC:72:9B:4C:F4:CF ServicesResolved: no
[CHG] Device DC:72:9B:4C:F4:CF Connected: no
```

4) After the pairing is successful, the Bluetooth interface of the mobile phone will be displayed as shown below



3.15. USB interface test

3.15.1. Connect mouse or keyboard test

1) Insert the keyboard of the USB interface into the USB interface of the Orange Pi development board

2) Connect the Orange Pi development board to the HDMI display

3) If the mouse or keyboard can operate normally, the USB interface is working normally (the mouse can only be used in the desktop version of the OS)

3.15.2. Connect USB storage device test

1) First, insert the U disk into the USB port of the Orange Pi development board

2) Execute the following command, if you can see the output of sdX, it means that the U disk is successfully recognized

```
root@orangepi:~# cat /proc/partitions | grep "sd*"
major minor  #blocks  name
   8        0   30044160 sda
```

```
8        1   30043119 sda1
```

3) Use the mount command to mount the U disk to /mnt, and then you can view the files in the U disk

```
root@orangepi:~# mount /dev/sda1 /mnt/
root@orangepi:~# ls /mnt/
test.txt
```

4) After mounting, you can view the capacity usage and mount point of the U disk through the df -h command

```
root@orangepi:~# df -h | grep "sd"
/dev/sda1        29G  208K   29G   1% /mnt
```

3.16. USB camera test

1) First insert the USB camera into the USB port of the Orange Pi development board

2) Then through the lsmod command, you can see that the kernel has automatically loaded the following modules

```
root@orangepi:~# lsmod
Module                  Size  Used by
uvcvideo              106496  0
videobuf2_vmalloc      16384  1 uvcvideo
videobuf2_memops       16384  1 videobuf2_vmalloc
videobuf2_v4l2         32768  1 uvcvideo
videobuf2_core         53248  2 uvcvideo,videobuf2_v4l2
```

3) Through the v4l2-ctl (note that l in v4l2 is a lowercase letter l, not a number 1) command, you can see that the device node information of the USB camera is /dev/video0

```
root@orangepi:~# apt update
root@orangepi:~# apt install v4l-utils
root@orangepi:~# v4l2-ctl  --list-devices
USB 2.0 Camera (usb-sunxi-ehci-1):
     /dev/video0
```

4) Use fswebcam to test the USB camera

a. Install fswebcam

```
root@orangepi:~# apt update
root@orangepi:~# apt-get install fswebcam
```

b. After installing fswebcam, you can use the following command to take pictures

a) -d option is used to specify the device node of the USB camera

b) --no-banner is used to remove watermark from photos
c) -r option is used to specify the resolution of the photo
d) -S option is used to skip the previous frame number
e) ./image.jpg is used to set the name and path of the generated photo

```
root@orangepi:~# fswebcam -d /dev/video0 --no-banner -r 1280x720 -S 5 ./image.jpg
```

c. In the server version of the Linux OS, after taking the picture, you can use the scp command to transfer the picture to the Ubuntu PC for image and viewing

```
root@orangepi:~# scp image.jpg test@192.168.1.55:/home/test （Modify the IP address and path according to the actual situation）
```

d. desktop version of Linux OS, you can directly view the captured pictures through the HDMI display

5) Use motion to test the USB camera

a. Install the camera test software motion

```
root@orangepi:~# apt update
root@orangepi:~# apt install motion
```

b. Modify the configuration of /etc/default/motion, change start_motion_daemon=no to start_motion_daemon=yes

```
root@orangepi:~# sed -i
"s/start_motion_daemon=no/start_motion_daemon=yes/" \
/etc/default/motion    (This is a command)
```

c. Modify the configuration of /etc/motion/motion.conf

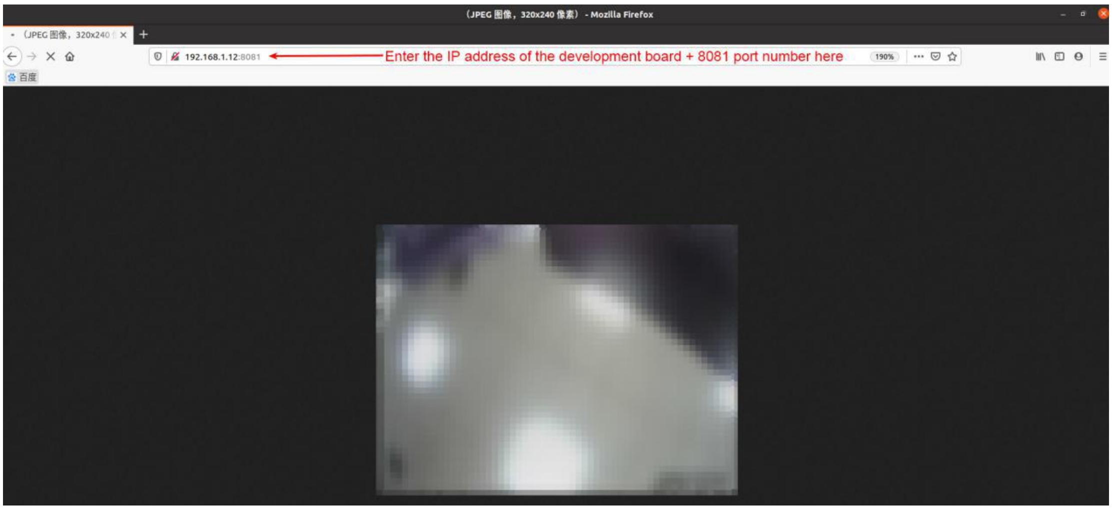
```
root@orangepi:~# sed -i "s/stream_localhost on/stream_localhost off/" \
/etc/motion/motion.conf    (This is a command)
```

d. Then restart the motion service

```
root@orangepi:~# /etc/init.d/motion restart
[ ok ] Restarting motion (via systemctl): motion.service.
```

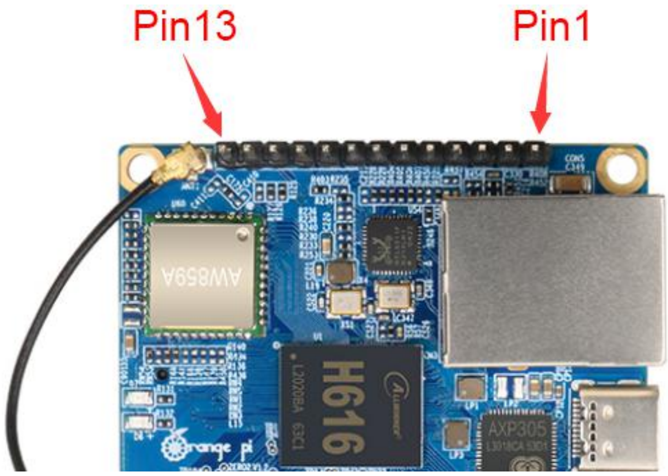
e. Before using motion, please make sure that the Orange Pi development board can connect to the network normally, and then obtain the IP address of the development board through the ifconfig command

f. Then enter the [IP address of the development board: 8081] in the Ubuntu PC or Windows PC on the same LAN as the development board or the Firefox browser of the mobile phone to see the video output by the camera

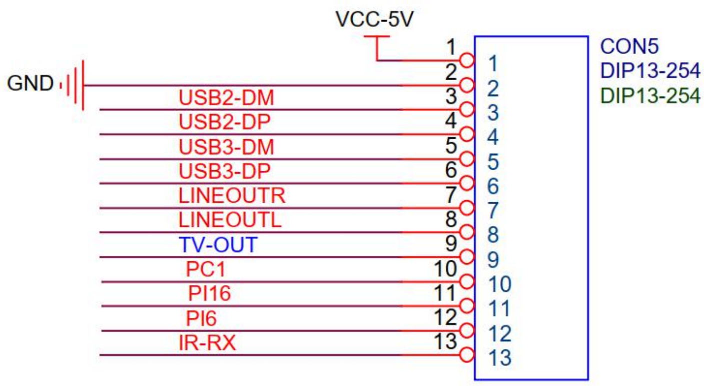


3.17. 13 Pin transfer board interface pin description

1) Please refer to the figure below for the sequence of the orange Pi Zero 2 development board 13 pin adapter board interface pins



2) The schematic diagram of the 13pin interface of the Orange Pi Zero 2 development board is shown below



3) The function description of the 13 pin adapter board interface pins of the Orange Pi Zero 2 development board is as follows

a. When the 13pin is connected to the adapter board, it can be additionally provided
    a) 2 USB2.0 Host
    b) Headphone left and right channel audio output
    c) TV-OUT video output
    d) IR receiving function
    e) After connecting the adapter board, pins 10, 11 and 12 of the 13pin interface cannot be used
    f) Also, note that the MIC on the 13pin adapter board cannot be used on Orange Pi Zero2

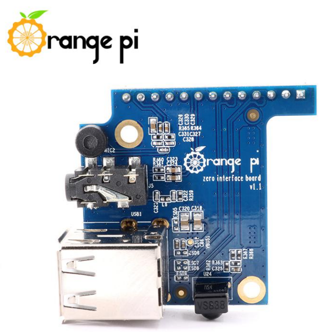
b. When the 13pin is not connected to the adapter board, the 10, 11, and 12 pins can be used as ordinary GPIO ports

GPIO serial number	Function	Pin
	5V	1
	GND	2
	USB2-DM	3
	USB2-DP	4
	USB3-DM	5
	USB3-DP	6
	LINEOUTR	7
	LINEOUTL	8
	TV-OUT	9
65	PC1	10
272	PI16	11
262	PI6	12
	IR-RX	13

3.18. Audio test

3.18.1. Headphone jack play audio test

1) First, you need to insert the 13pin adapter board into the 13pin interface of the Orange Pi development board, and then insert the headset into the audio interface



2) Through the aplay -l command, you can view the sound card devices supported by the Linux OS, where card 0: audiocodec is the sound card device required for headset playback

```
root@orangepi:~# aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: audiocodec [audiocodec], device 0: SUNXI-CODEC sun50iw9-codec-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

3) Upload the audio files that need to be played to the /root folder of the Linux OS. You can use the scp command to upload in the Ubuntu PC (the IP address in the command is the IP address of the Orange Pi development board), or copy it with a USB flash drive

```
test@test:~/AudioTest$ scp audio.wav  root@192.168.1.xx:/root (Modify the IP address and path according to the actual situation)
```

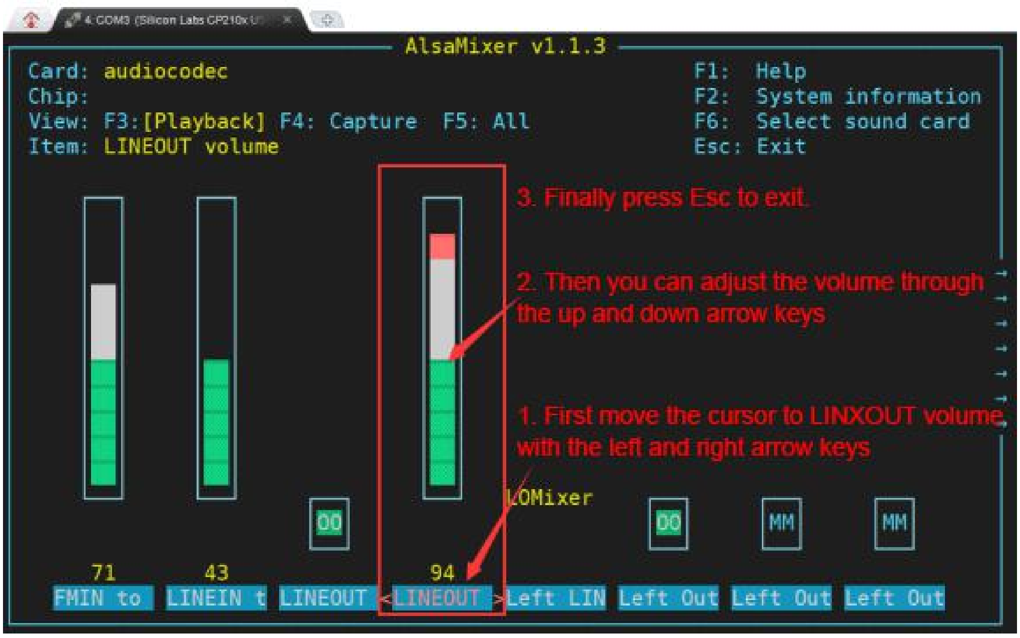
4) Then use the aplay command to play the audio, the headset can hear the sound

```
root@orangepi:~# aplay -D hw:0,0 audio.wav
Playing WAVE 'audio.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
```

5) The volume of the headset can be adjusted through the alsamixer command
   a. Enter the alsamixer command in the terminal

```
root@orangepi:~# alsamixer
```

   b. After entering the alsamixer command, the following audio setting interface will pop up, and then you can adjust the size of the audio through the up, down, left, and right direction keys. The specific steps are shown in the figure below



3.18.2. HDMI audio playback test

1) First use the Micro HDMI to HDMI cable to connect the Orange Pi development board to the TV (other HDMI displays need to ensure that they can play audio)

2) Upload the audio files that need to be played to the /root folder of the Linux OS. You can use the scp command to upload in the Ubuntu PC (the IP address in the command is the IP address of the Orange Pi development board), or copy using a USB flash drive

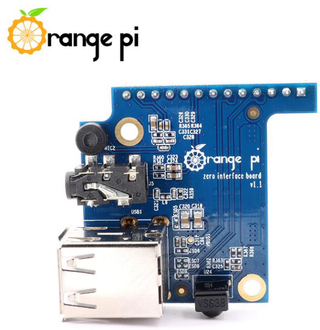
test@test:~/AudioTest$ **scp audio.wav root@192.168.1.xx:/root （Modify the IP address and path according to the actual situation)**

3) HDMI audio playback does not require other settings, just use the aplay command to play directly

root@orangepi:~# **aplay -D hw:1,0 audio.wav**

3.19. IR receiving test

1) First, you need to insert the 13pin adapter board into the 13pin interface of the Orange Pi development board. After the adapter board is inserted, the Orange Pi Zero 2 can use the IR receiving function



2) Install ir-keytable IR test software

root@orangepi:~# **apt update**
root@orangepi:~# **apt-get install ir-keytable**

3) Then execute ir-keytable to view the information of the IR device

```
root@orangepi:~# ir-keytable
Found /sys/class/rc/rc0/ (/dev/input/event1) with:
        Driver: sunxi-rc-recv, table: rc_map_sunxi
        lirc device: /dev/lirc0
        Supported protocols: lirc nec
        Enabled protocols: lirc nec
        Name: sunxi_ir_recv
```

```
bus: 25, vendor/product: 0001:0001, version: 0x0100
Repeat delay = 500 ms, repeat period = 125 ms
```

4) An IR remote controller needs to be prepared before testing the IR receiving function

5) Then enter the ir-keytable -t command in the terminal, and then use the IR remote control to press the button against the IR receiving head of the Orange Pi development board to see the received key code in the terminal

```
root@orangepi:/# ir-keytable -t
Testing events. Please, press CTRL-C to abort.
1598339152.260376: event type EV_MSC(0x04): scancode = 0xfb0413
1598339152.260376: event type EV_SYN(0x00).
1598339152.914715: event type EV_MSC(0x04): scancode = 0xfb0410
```

3.20. Hardware watchdog test

1) Download the code of wiringOP

```
root@orangepi:~# apt update
root@orangepi:~# apt install git
root@orangepi:~# git clone https://github.com/orangepi-xunlong/wiringOP
```

2) Compile wiringOP

```
root@orangepi:~# cd wiringOP
root@orangepi:~/wiringOP# ./build clean
root@orangepi:~/wiringOP# ./build
```

3) Compile the watchdog test program

```
root@orangepi:~/wiringOP# cd examples/
root@orangepi:~/wiringOP/examples# make watchdog
[CC] watchdog.c
[link]
```

4) Run the watchdog test programsystem

a. The second parameter 10 represents the counting time of the watchdog. If there is no dog feeding within this time, the will restart
b. We can feed the dog by pressing any key on the keyboard (except ESC). After feeding the dog, the program will print a line of keep alive to indicate that the dog is successfully fed

```
root@orangepi:~/wiringOP/examples# ./watchdog 10
open success
options is 33152,identity is sunxi-wdt
put_usr return,if 0,success:0
The old reset time is: 16
return ENOTTY,if -1,success:0
return ENOTTY,if -1,success:0
put_user return,if 0,success:0
put_usr return,if 0,success:0
keep alive
keep alive
keep alive
```

3.21. Temperature sensor

1) H616 has a total of 4 temperature sensors, the command to check the temperature is as follows

a. sensor0：CPU

```
root@orangepi:~# cat /sys/class/thermal/thermal_zone0/type
cpu_thermal_zone
root@orangepi:~# cat /sys/class/thermal/thermal_zone0/temp
57734
```

b. sensor1：GPU

```
root@orangepi:~# cat /sys/class/thermal/thermal_zone1/type
gpu_thermal_zone
root@orangepi:~# cat /sys/class/thermal/thermal_zone1/temp
57410
```

c. sensor2：VE

```
root@orangepi:~# cat /sys/class/thermal/thermal_zone2/type
ve_thermal_zone
root@orangepi:~# cat /sys/class/thermal/thermal_zone2/temp
59273
```

d. sensor3：DDR

```
root@orangepi:~# cat /sys/class/thermal/thermal_zone3/type
ddr_thermal_zone
root@orangepi:~# cat /sys/class/thermal/thermal_zone3/temp
58949
```

3.22. How to install Docker

1) Uninstall the old version of docker that may exist first

```
root@orangepi:~# apt remove docker docker-engine docker-ce docker.io
```

2) Then install the following packages

```
root@orangepi:~# apt update
```

```
root@orangepi:~# apt install -y apt-transport-https ca-certificates curl \
  software-properties-common
```

3) Add the key of Alibaba Cloud docker

```
root@orangepi:~# curl -fsSL http://mirrors.aliyun.com/docker-ce/linux/ubuntu/gpg \
 | sudo apt-key add -
```

4) Add the corresponding docker source in the OS source of ubuntu

```
root@orangepi:~# add-apt-repository "deb [arch=arm64] \
   https://mirrors.aliyun.com/docker-ce/linux/ubuntu $(lsb_release -cs) stable"
```

5) nstall the latest version of docker-ce

```
Iroot@orangepi:~# apt update
root@orangepi:~# apt install docker-ce
```

6) Verify the status of docker

```
root@orangepi:~# systemctl status docker
● docker.service - Docker Application Container Engine
  Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2020-08-24 10:29:22 UTC; 26min ago
    Docs: https://docs.docker.com
 Main PID: 3145 (dockerd)
   Tasks: 15
  CGroup: /system.slice/docker.service
          └─3145 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

7) Test docker

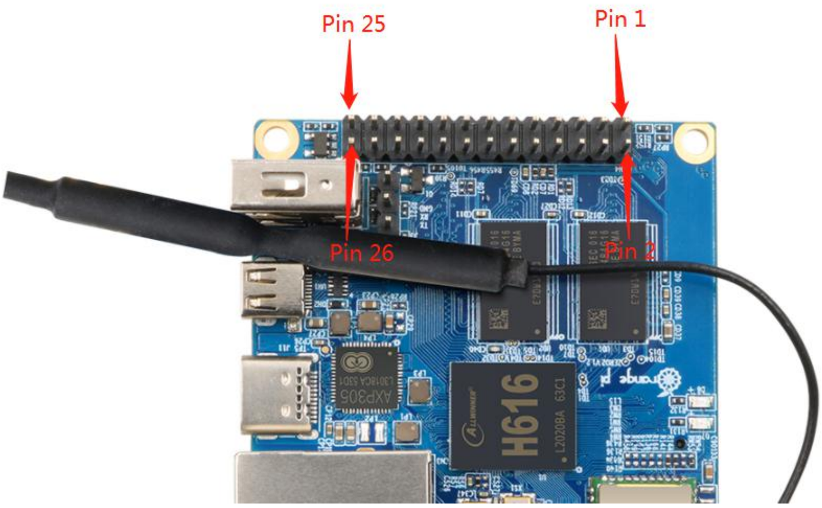
```
root@orangepi:~# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
256ab8fe8778: Pull complete
Digest:
sha256:7f0a9f93b4aa3022c3a4c147a449ef11e0941a1fd0bf4a8e6c9408b2600777c5
Status: Downloaded newer image for hello-world:latest
```

```
Hello from Docker!
This message shows that your installation appears to be working correctly.
```

3.23. 26pins GPIO, I2C, UART, SPI test

3.23.1. 26 Pins description

1) Please refer to the figure below for the sequence of the 26 pins of the Orange Pi Zero 2 development board



2) The function of the 26 pins of the Orange Pi Zero 2 development board is shown in the table below

GPIO No.	GPIO	Function	Pin	Pin	Function	GPIO	GPIO No.
		3.3V	1	2	5V		
229	PH5	TWI3-SDA	3	4	5V		
228	PH4	TWI3-SCK	5	6	GND		
73	PC9	PC9	7	8	UART5_TX	PH2	226
		GND	9	10	UART5_RX	PH3	227
70	PC6	PC6	11	12	PC11	PC11	75
69	PC5	PC5	13	14	GND		
72	PC8	PC8	15	16	PC15	PC15	79
		3.3V	17	18	PC14	PC14	78
231	PH7	SPI1_MOSI	19	20	GND		
232	PH8	SPI1_MISO	21	22	PC7	PC7	71
230	PH6	SPI1_CLK	23	24	SPI1_CS	PH9	233
		GND	25	26	PC10	PC10	74

3.23.2. Install wiringOP

1) WiringOP has been adapted to the Orange Pi Zero 2 development board, using wiringOP can test the functions of GPIO, I2C, UART, and SPI

2)Download the code of wiringOP

```
root@orangepi:~# apt update
root@orangepi:~# apt install git
root@orangepi:~# git clone https://github.com/orangepi-xunlong/wiringOP
```

3) Compile wiringOP

```
root@orangepi:~# cd wiringOP
root@orangepi:~/wiringOP# ./build clean
root@orangepi:~/wiringOP# ./build
```

4) The output of the test gpio readall command is as follows

a. There is a one-to-one correspondence between pins 1 to 26 and 26 Pins on the development board

b. Pin 27 corresponds to pin 10 of 13pins on the development board

c. Pin 29 corresponds to pin 11 of 13pins on the development board

d. Pin 31 corresponds to pin 12 of 13pins on the development board

e. 28, 30, 32 pins are empty, please ignore directly

```
root@orangepizero2:~/wiringOP# gpio readall
 +------+-----+----------+------+---+  Zero 2  +---+------+----------+-----+------+
 | GPIO | wPi |   Name   | Mode | V | Physical | V | Mode | Name     | wPi | GPIO |
 +------+-----+----------+------+---+----++----+---+------+----------+-----+------+
 |      |     |     3.3V |      |   |  1 || 2  |   |      | 5V       |     |      |
 |  229 |   0 |    SDA.3 |  OUT | 1 |  3 || 4  |   |      | 5V       |     |      |
 |  228 |   1 |    SCL.3 |  OUT | 1 |  5 || 6  |   |      | GND      |     |      |
 |   73 |   2 |      PC9 |  OUT | 1 |  7 || 8  | 1 | OUT  | TXD.5    | 3   | 226  |
 |      |     |      GND |      |   |  9 || 10 | 1 | OUT  | RXD.5    | 4   | 227  |
 |   70 |   5 |      PC6 |  OUT | 1 | 11 || 12 | 1 | OUT  | PC11     | 6   | 75   |
 |   69 |   7 |      PC5 |  OUT | 1 | 13 || 14 |   |      | GND      |     |      |
 |   72 |   8 |      PC8 |  OUT | 1 | 15 || 16 | 1 | OUT  | PC15     | 9   | 79   |
 |      |     |     3.3V |      |   | 17 || 18 | 1 | OUT  | PC14     | 10  | 78   |
 |  231 |  11 |   MOSI.1 |  OUT | 1 | 19 || 20 |   |      | GND      |     |      |
 |  232 |  12 |   MISO.1 |  OUT | 1 | 21 || 22 | 1 | OUT  | PC7      | 13  | 71   |
 |  230 |  14 |   SCLK.1 |  OUT | 1 | 23 || 24 | 1 | OUT  | CE.1     | 15  | 233  |
 |      |     |      GND |      |   | 25 || 26 | 1 | OUT  | PC10     | 16  | 74   |
 |   65 |  17 |      PC1 |  OUT | 1 | 27 || 28 |   |      |          |     |      |
 |  272 |  18 |     PI16 |  OUT | 1 | 29 || 30 |   |      |          |     |      |
 |  262 |  19 |      PI6 |  OUT | 1 | 31 || 32 |   |      |          |     |      |
 +------+-----+----------+------+---+----++----+---+------+----------+-----+------+
 | GPIO | wPi |   Name   | Mode | V | Physical | V | Mode | Name     | wPi | GPIO |
 +------+-----+----------+------+---+  Zero 2  +---+------+----------+-----+------+
root@orangepizero2:~/wiringOP#
```

3.23.3. Test common GPIO port

1) The following is an example of how to set the high and low level of the GPIO port with pin No. 7-corresponding to GPIO as PC9-corresponding to wPi number as 2-

```
root@orangepizero2:~/wiringOP# gpio readall
 +------+-----+----------+------+---+  Zero 2  +---+------+----------+-----+------+
 | GPIO | wPi |   Name   | Mode | V | Physical | V | Mode | Name     | wPi | GPIO |
 +------+-----+----------+------+---+----++----+---+------+----------+-----+------+
 |      |     |     3.3V |      |   |  1 || 2  |   |      | 5V       |     |      |
 |  229 |   0 |    SDA.3 |  OUT | 1 |  3 || 4  |   |      | 5V       |     |      |
 |  228 |   1 |    SCL.3 |  OUT | 1 |  5 || 6  |   |      | GND      |     |      |
 |   73 |   2 |      PC9 |  OUT | 1 |  7 || 8  | 1 | OUT  | TXD.5    | 3   | 226  |
 |      |     |      GND |      |   |  9 || 10 | 1 | OUT  | RXD.5    | 4   | 227  |
 |   70 |   5 |      PC6 |  OUT | 1 | 11 || 12 | 1 | OUT  | PC11     | 6   | 75   |
```

2) First set the GPIO port to output mode, and **the third parameter needs to input the serial number of the wPi corresponding to the pin**

root@orangepi:~/wiringOP# **gpio mode 2 out**

3) Then set the GPIO port to output low level. After setting, you can use a multimeter to measure the value of the pin voltage. If it is 0v, it means that the low level is set successfully

root@orangepi:~/wiringOP# **gpio write 2 0**

Use gpio readall to see that the value (V) of pin 7 has become 0

```
root@orangepizero2:~# gpio readall
 +------+-----+----------+------+---+  Zero 2  +---+------+----------+-----+------+
 | GPIO | wPi |   Name   | Mode | V | Physical | V | Mode | Name     | wPi | GPIO |
 +------+-----+----------+------+---+----++----+---+------+----------+-----+------+
 |      |     |     3.3V |      |   |  1 || 2  |   |      | 5V       |     |      |
 |  229 |   0 |    SDA.3 |  OFF | 0 |  3 || 4  |   |      | 5V       |     |      |
 |  228 |   1 |    SCL.3 |  OFF | 0 |  5 || 6  |   |      | GND      |     |      |
 |   73 |   2 |      PC9 |  OUT | 0 |  7 || 8  | 0 | ALT2 | TXD.5    | 3   | 226  |
 |      |     |      GND |      |   |  9 || 10 | 0 | ALT2 | RXD.5    | 4   | 227  |
 |   70 |   5 |      PC6 | ALT5 | 0 | 11 || 12 | 0 | OFF  | PC11     | 6   | 75   |
```

4) Then set the GPIO port to output high level. After setting, you can use a multimeter to measure the value of the pin voltage. If it is 3.3v, it means that the high level is set successfully

root@orangepi:~/wiringOP# **gpio write 2 1**

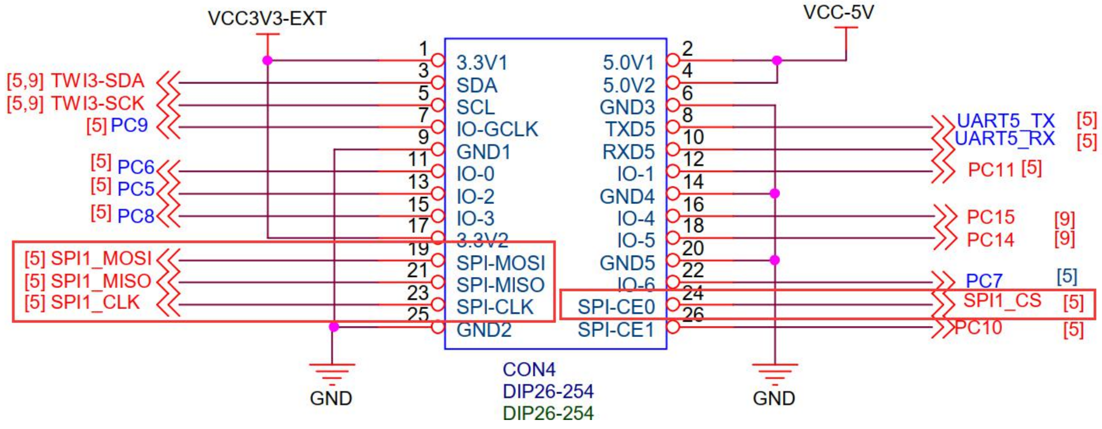
Use gpio readall to see that the value (V) of pin 7 has become 1

```
root@orangepizero2:~# gpio readall
 +------+-----+----------+------+---+  Zero 2  +---+------+----------+-----+------+
 | GPIO | wPi |   Name   | Mode | V | Physical | V | Mode | Name     | wPi | GPIO |
 +------+-----+----------+------+---+----++----+---+------+----------+-----+------+
 |      |     |     3.3V |      |   |  1 || 2  |   |      | 5V       |     |      |
 |  229 |   0 |    SDA.3 |  OFF | 0 |  3 || 4  |   |      | 5V       |     |      |
 |  228 |   1 |    SCL.3 |  OFF | 0 |  5 || 6  |   |      | GND      |     |      |
 |   73 |   2 |      PC9 |  OUT | 1 |  7 || 8  | 0 | ALT2 | TXD.5    | 3   | 226  |
 |      |     |      GND |      |   |  9 || 10 | 0 | ALT2 | RXD.5    | 4   | 227  |
 |   70 |   5 |      PC6 | ALT5 | 0 | 11 || 12 | 0 | OFF  | PC11     | 6   | 75   |
```

5) The setting method of other pins is similar, just modify the serial number of wPi to the serial number corresponding to the pin.

3.23.4. SPI test

1) According to the schematic diagram of the 26pins connector, the available SPI for Orange Pi Zero2 is SPI1



2) First check whether there is a **spidev1.1** device node in the Linux OS. If it exists, it means that SPI1 has been set up and can be used directly

```
root@orangepi:~/wiringOP/examples# ls /dev/spidev1*
/dev/spidev1.1
```

3) Compile the spidev_test test program in the examples of wiringOP

```
root@orangepi:~/wiringOP/examples# make spidev_test
[CC] spidev_test.c
[link]
```

4)Do not short-circuit the mosi and miso pins of SPI1 first, and the output result of running spidev_test is as follows, you can see that the data of TX and RX are inconsistent

```
root@orangepi:~/wiringOP/examples# ./spidev_test -v -D /dev/spidev1.1
spi mode: 0x0
bits per word: 8
max speed: 500000 Hz (500 KHz)
TX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF F0 0D  | ......@....▒..................▒.
RX | FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF  | ................................
```

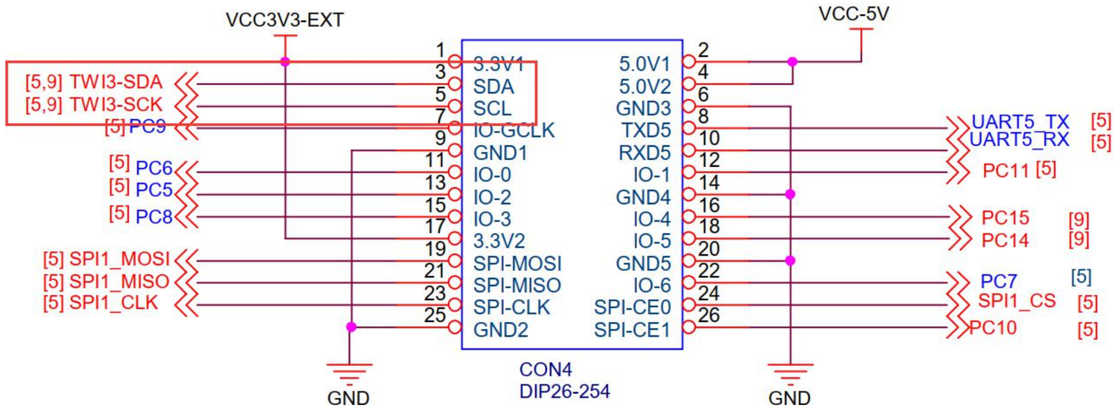
5) Then short-circuit the two pins of SPI1's mosi (the 19th pin in 26pins) and miso (the 21st pin in 26pins) and run the output of spidev_test as follows, you can see that the sent and received data are the same

```
root@orangepi:~/wiringOP/examples# ./spidev_test -v -D /dev/spidev1.1
spi mode: 0x0
```

```
bits per word: 8
max speed: 500000 Hz (500 KHz)
TX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF F0 0D  | ......@....▒.................▒.
RX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF F0 0D  | ......@....▒.................▒.
```

3.23.5. I2C test

1) According to the schematic diagram of 26pins, the i2c available for Orange Pi Zero 2 is i2c3



2) After starting the linux OS, first confirm that there is an i2c3 device node under /dev

```
root@orangepi:~# ls /dev/i2c-*
/dev/i2c-3  /dev/i2c-5
```

3) Then start to test i2c, first install i2c-tools

```
root@orangepi:~# apt update
root@orangepi:~# apt install i2c-tools
```

4) Then connect an i2c device to the i2c3 pin of the 26pins connector

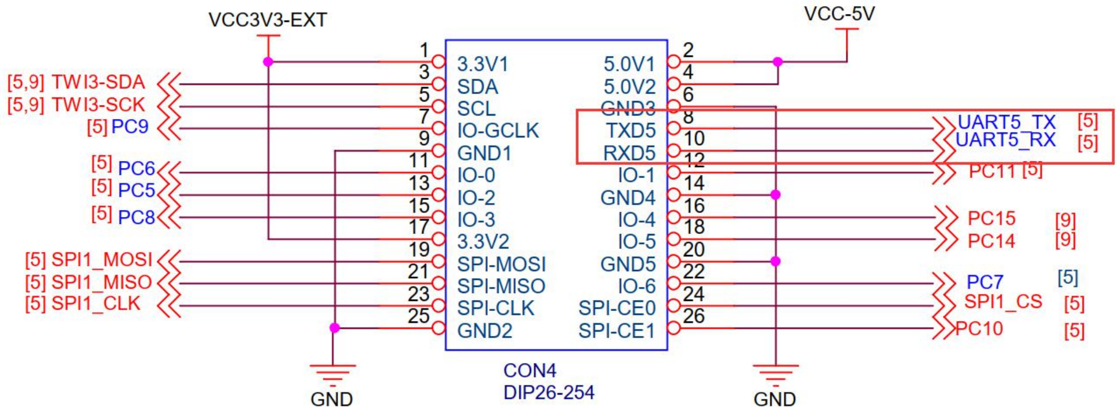
	i2c3
Sda Pin	Corresponding to pin 3
Sck Pin	Corresponding to pin 5
vcc Pin	Corresponding to pin 1
gnd Pin	Corresponding to pin 6

5) Then use the i2cdetect -y 3 commands if the address of the connected i2c device can be detected, it means that i2c can be used normally

```
root@orangepi:~# i2cdetect -y 3
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: 50 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

3.23.6. UART test

1) According to the schematic diagram of 26pins, the uart available for Orange Pi Zero 2 is UART5



2) After starting the linux OS, first confirm that there is a UART5 device node under /dev

```
root@orangepi:~# ls /dev/ttyS*
/dev/ttyS0  /dev/ttyS1  /dev/ttyS5
```

3) Then start to test the UART5 interface, first use the Dupont line to short-circuit the rx and tx of the UART5 interface to be tested

	UART5
tx Pin	Corresponding to pin 8
rx Pin	Corresponding to pin 10

4) Then modify the serial device node name opened by the serial test program serialTest in wiringOP to **/dev/ttyS5**

```
root@orangepizero2:~/wiringOP/examples# vim serialTest.c
```

```
int main ()
{
  int fd ;
  int count ;
  unsigned int nextTime ;

  if ((fd = serialOpen ("/dev/ttyS5", 115200)) < 0)
  {
    fprintf (stderr, "Unable to open serial device: %s\n", strerror (errno)) ;
    return 1 ;
  }
```

5) Recompile the serial test program serialTest in wiringOP

```
root@orangepi:~/wiringOP/examples# make serialTest
[CC] serialTest.c
[link]
root@orangepi:~/wiringOP/examples#
```

6) Finally, run serialTest, if you can see the following print, it means that the serial communication is normal

```
root@orangepi:~/wiringOP/examples# ./serialTest

Out:   0:  ->   0
Out:   1:  ->   1
Out:   2:  ->   2
Out:   3:  ->   3
Out:   4:  ->   4
Out:   5:  ->   5
Out:   6:  ->   6
Out:   7:  ->   7
Out:   8:  ->   8^C
```

3.24. Method of redirecting kernel console output to serial port ttyS5

The kernel console outputs to ttyS0 by default, which is the 3pin debug serial port on the development board. We can also set the kernel console output to be redirected to the 26pins connector UART5, the method is as follows:

1) Modify console=ttyS0 in /boot/boot.cmd to console=ttyS5

```
root@orangepi:~# vim /boot/boot.cmd
```

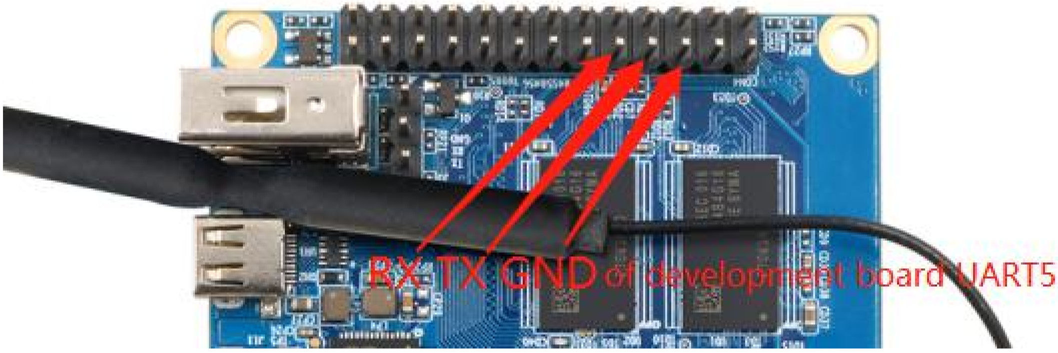
```
if test "${console}" = "display" || test "${console}" = "both"; then setenv consoleargs "console=ttyS5 115200 console=tty1"; fi
if test "${console}" = "serial"; then setenv consoleargs "console=ttyS5 115200"; fi
if test "${bootlogo}" = "true"; then setenv consoleargs "bootsplash.bootfile=bootsplash.orangepi ${consoleargs}"; fi
```

2) Then compile and recompile /boot/boot.cmd to /boot/boot.scr

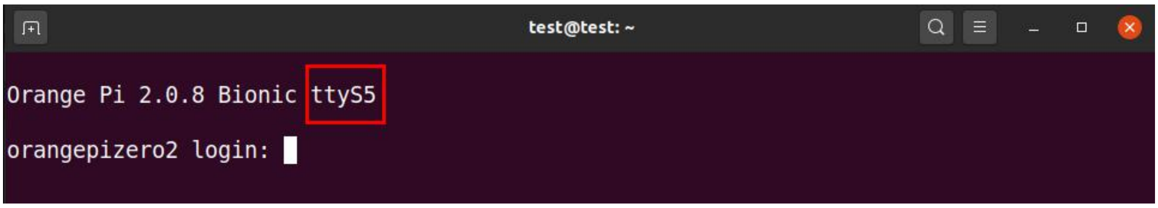
```
root@orangepi:~# mkimage -C none -A arm -T script -d /boot/boot.cmd
/boot/boot.scr
Image Name:
Created:      Tue Nov  3 01:45:17 2020
Image Type:   ARM Linux Script (uncompressed)
Data Size:    2247 Bytes = 2.19 KiB = 0.00 MiB
Load Address: 00000000
Entry Point:  00000000
Contents:
   Image 0: 2239 Bytes = 2.19 KiB = 0.00 MiB
```

3) Then connect the USB to TTL module to the 26pins UART5 pin through the DuPont cable

a. Connect the GND of the USB to TTL module to the GND of the 26pins of the development board
b. Connect the **RX** of the USB to TTL module to the TX of the development board UART5
c. Connect the **TX** of the USB to TTL module to the RX of the development board UART5

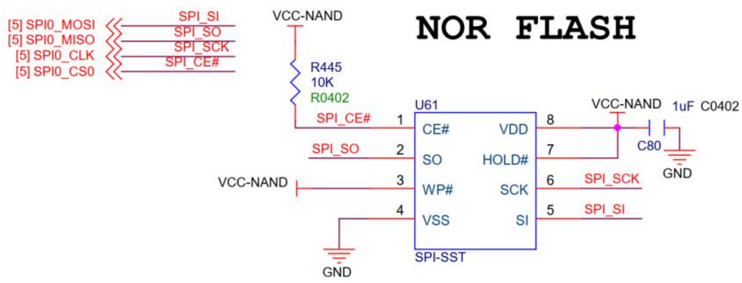


4) Then restart the development board, you can see that the kernel console outputs to ttyS5 by default. Note that the output log of u-boot is still output to ttyS0, not ttyS5



3.25. SPI Nor Flash test

1) In the Orange Pi Zero 2 development board, SPI0 is connected to the onboard 2MB SPI Nor Flash. WiringOP can be used to test it. At present, it can only test the data read and write of the SPI Nor Flash chip to ensure that the hardware is okay. , Can’t use SPI Nor Flash to run U-boot



2) The SPI Flash test program in wiringOP is w25q64_test.c

```
root@orangepi:~/wiringOP# cd examples/
root@orangepi:~/wiringOP/examples# ls w25q64_test.c
w25q64_test.c
```

3) First make sure that both SPI_CHANNEL and SPI_PORT in w25q64_test.c are set to 0

```
#define SPI_CHANNEL 0
#define SPI_PORT 0
```

4) Then check whether there is a device node of spidev0.0 in the Linux OS. If it exists, it means that SPI0 has been set up and can be used directly

```
root@orangepi:~/wiringOP/examples# ls /dev/spidev0*
/dev/spidev0.0
```

5) Then compile w25q64_test

```
root@orangepi:~/wiringOP/examples# make w25q64_test
[CC] w25q64_test.c
[link]
```

6) Test SPI Nor Flash, if you can see data output (not all 0 or ff), it means that SPI Nor Flash reads and writes normally

```
root@orangepi:~/wiringOP/examples# ./w25q64_test
Opening device /dev/spidev0.0
JEDEC ID : c2 20 16
Unique ID : ff ff ff ff ff ff ff
Read Data: n=256
-------------------------------------------------------
00000: 30 31 32 33 34 35 36 37 38 39 41 42 43 44 45 46 |a2
00010: 47 48 49 4a 4b 4c 4d 4e 4f 50 51 52 53 54 55 56 |e8
00020: 57 58 59 5a ff ff ff ff ff ff ff ff ff ff ff ff |56
00030: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
00040: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
00050: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
00060: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
```

```
00070: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
00080: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
00090: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
000a0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
000b0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
000c0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
000d0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
000e0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
000f0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
---------------------------------------------------------
     c1 c4 c7 ca 71 73 75 77 79 7b 84 86 88 8a 8c 8e |10
......
```

4. Linux SDK instructions

The compilation of the Linux SDK is done on a **PC with Ubuntu 18.04** installed. Other versions of Ubuntu OS may have some differences

4.1. Get the source code of Linux SDK

1) First download the code of orangepi-build, the code of orangepi-build is modified based on the armbian build OS

a. Currently the Orange Pi Zero 2 development board only supports the legacy branch
b. The kernel version is linux4.9
c. u-boot version is v2018.05

```
test@test:~$ sudo apt update
test@test:~$ sudo apt install git
test@test:~$ git clone https://github.com/orangepi-xunlong/orangepi-build.git
```

2) Orangepi-build will contain the following files and folders after downloading

a. **build.sh:** Compile the startup script
b. **external:** Contains the configuration files needed to compile the image, specific scripts, and the source code of some programs, etc.
c. **LICENSE:** GPL 2 license file
d. **README.md:** orangepi-build instruction file
e. **scripts:** general scripts for compiling linux images

```
test@test:~/orangepi-build$ ls
build.sh  external  LICENSE  README.md  scripts
```

3) The orangepi-build repository does not contain the source code of the Linux kernel and u-boot after the first download. They are stored in a separate git repository

(please do not download the source code of the kernel and u-boot separately, unless you know how to use), when orangepi-build runs for the first time, it will automatically download the kernel and u-boot source code

a. The repository where the Linux kernel source code is stored is as follows, where sun50iw9 is the code name of the H616 SOC chip

```
https://github.com/orangepi-xunlong/linux-orangepi/tree/orange-pi-4.9-sun50iw9
```

b. The repository where u-boot source code is stored is as follows, where sun50iw9 is the code name of the H616 SOC chip

```
https://github.com/orangepi-xunlong/u-boot-orangepi/tree/v2018.05-sun50iw9
```

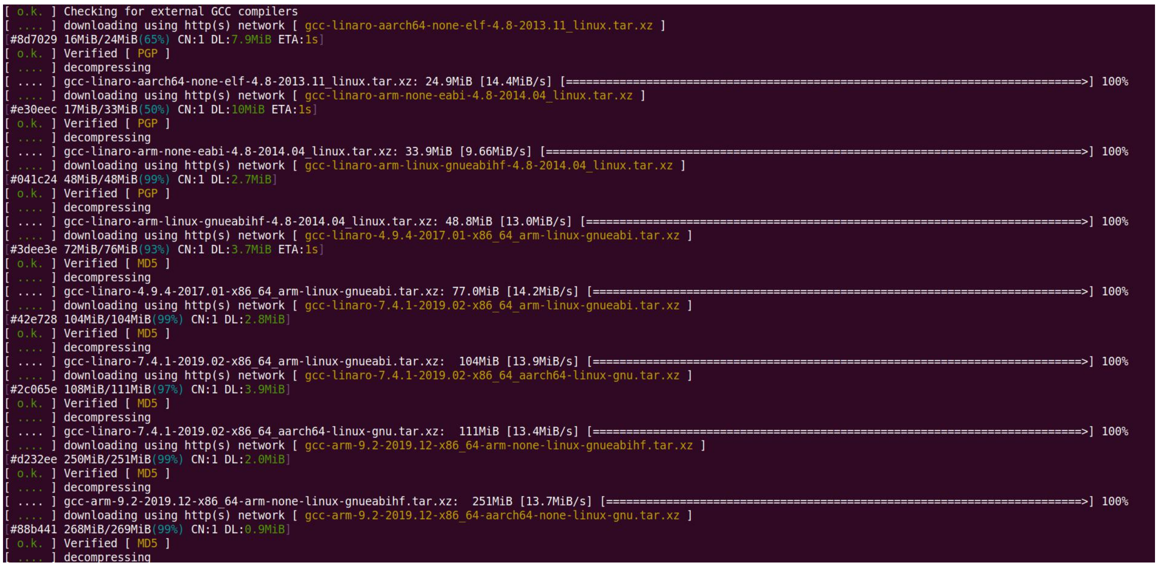
4) Orangepi-build will download the cross-compilation toolchain, u-boot, and Linux kernel source code during the process of compiling the image. After successfully compiling the image once, the files and folders that can be seen in orangepi-build are

a. **build.sh:** Compile the startup script
b. **external:** Contains the configuration files needed to compile the image, scripts for specific functions, and the source code of some programs. The rootfs compressed package cached during the compiling of the image is also stored in external
c. **external:** Stores the source code of the Linux kernel, the folder named orange-pi-4.9-sun50iw9 stores the kernel source code of Orange Pi Zero 2 (sun50iw9 is the code name of the H616 SOC chip)
d. **LICENSE:** GPL 2 license file
e. **README.md:** orangepi-build instruction file
f. **output:** store the compiled u-boot, linux and other deb packages, compilation logs, and compiled images and other files scripts: general scripts for compiling linux images
g. **toolchains:** store cross-compilation toolchains
h. **u-boot:** Store the source code of u-boot, the folder named v2018.05-sun50iw9 inside stores the u-boot source code of Orange Pi Zero 2 (sun50iw9 is the code name of the H616 SOC chip)
i. **userpatches:** store configuration files needed to compile scripts

```
test@test:~/orangepi-build$ ls
build.sh  external  kernel  LICENSE  output  README.md  scripts  toolchains
u-boot  userpatches
```

4.2. Download the compilation toolchain

1) When orangepi-build is run for the first time, it will automatically download the cross-compilation toolchains and place it in the toolchains folder. Every time the orangepi-build build.sh script is run, it will check whether the cross-compilation toolchain in toolchains exists. If it does not exist, it will restart the download, if it exists, it will be used directly, and the download will not be repeated



2) The image URL of the cross-compilation tool chain in China is the open source software image site of Tsinghua University

```
https://mirrors.tuna.tsinghua.edu.cn/armbian-releases/_toolchain/
```

3) After **toolchains** is downloaded, it will contain multiple versions of cross-compilation toolchains

```
test@test:~/orangepi-build$ ls toolchains/
gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu          gcc-linaro-7.4.1-2019.02-x86_64_arm-linux-gnueabi
gcc-arm-9.2-2019.12-x86_64-arm-none-linux-gnueabihf       gcc-linaro-aarch64-none-elf-4.8-2013.11_linux
gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabi          gcc-linaro-arm-linux-gnueabihf-4.8-2014.04_linux
gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu      gcc-linaro-arm-none-eabi-4.8-2014.04_linux
```

4) The cross-compilation toolchains used to compile the Orange Pi Zero 2 linux4.9 kernel source code is

```
gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu
```

5) The cross-compilation toolchain used to compile the Orange Pi Zero 2 u-boot v2018.05 source code is

```
gcc-linaro-7.4.1-2019.02-x86_64_arm-linux-gnueabi
```

4.3. Compile u-boot

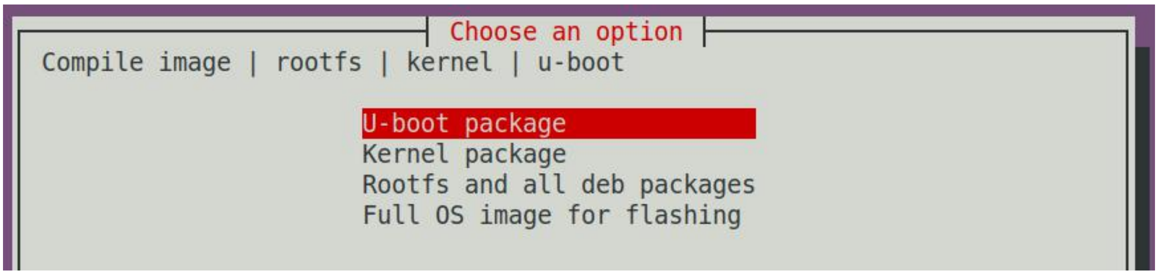
1) First of all, it should be noted that **there is no boot0 source code in the u-boot source code**, this part of the source code is currently not open, only the bin file of

boot0 is provided

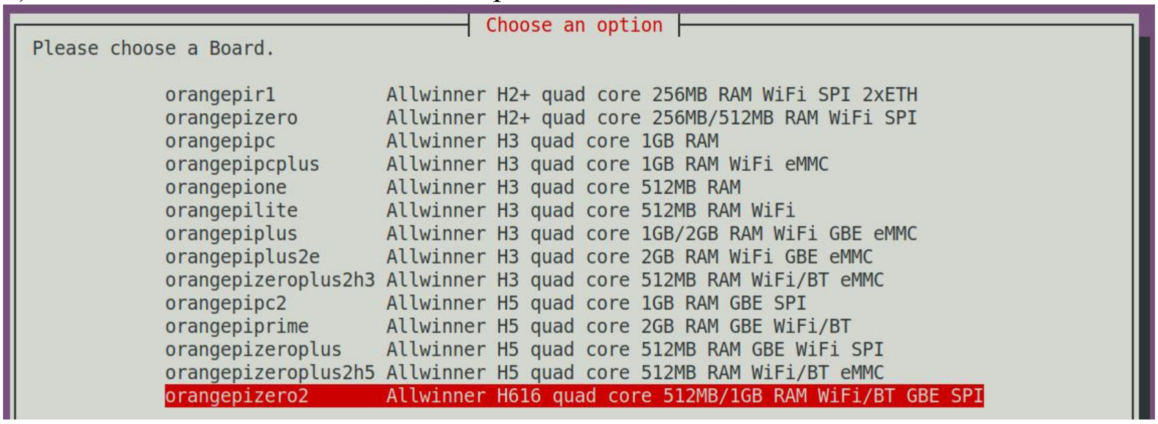
2) Run the build.sh script, remember to add sudo permissions

```
test@test:~/orangepi-build$ sudo ./build.sh
```

3) Select **U-boot package,** then press Enter



4) Then select the model of the development board



5) Then it will start to compile u-boot, some of the information prompted during compilation are explained as follows

a.u-boot source version

```
[ o.k. ] Compiling u-boot [ v2018.05 ]
```

b. The version of the cross-compilation toolchain

```
[ o.k. ] Compiler version [ arm-linux-gnueabi-gcc 7.4.1 ]
```

c. Compile the generated u-boot deb package path

```
[ o.k. ] Target directory [ orangepi-build/output/debs/u-boot ]
```

d. Package name of u-boot deb package generated by compiling

```
[ o.k. ] File name [ linux-u-boot-legacy-orangepizero2_2.0.8_arm64.deb ]
```

e. Compilation time

```
[ o.k. ] Runtime [ 1 min ]
```

f. Repeat the command to compile u-boot, use the following command without selecting through the graphical interface, you can start compiling u-boot directly

```
[ o.k. ] Repeat Build Options [ sudo ./build.sh  BOARD=orangepizero2
BRANCH=legacy BUILD_OPT=u-boot KERNEL_CONFIGURE=no ]
```

6) View the compiled u-boot deb package

```
test@test:~/orangepi-build$ ls output/debs/u-boot/
linux-u-boot-legacy-orangepizero2_2.0.8_arm64.deb
```

7) The files contained in the generated u-boot deb package are as follows

a. Use the following command to unzip the deb package

```
test@test:~/orangepi-build$ cd output/debs/u-boot
test@test:~/orangepi_build/output/debs/u-boot$ $ dpkg -x \
linux-u-boot-legacy-orangepizero2_2.0.8_arm64.deb .  (Note that there is a "." at
   the end of the command.)
test@test:~/orangepi_build/output/debs/u-boot$ ls
linux-u-boot-current-orangepizero_2.0.8_armhf.deb  usr
```

b. The decompressed file is as follows

```
test@test:~orangepi-build/output/debs/u-boot$ tree usr/
usr/
└── lib
    ├── linux-u-boot-legacy-orangepizero2_2.0.8_arm64
    │   ├── boot0_sdcard.fex
    │   └── boot_package.fex
    └── u-boot
        ├── LICENSE
        ├── orangepi_zero2_defconfig
        ├── orangepizero2-u-boot.dts
        └── platform_install.sh

3 directories, 6 files
```

8) When the orangepi-bulid build OS compiles the u-boot source code, it first synchronizes the u-boot source code with the u-boot source code of the github server, so if you want to modify the u-boot source code, you first need to turn off the download and update function of the source code **(You need to compile u-boot once to turn off this function, otherwise you will be prompted to find the source code of u-boot)**, otherwise, the changes made will be restored, the method is as follows:

Set the IGNORE_UPDATES variable in userpatches/config-default.conf to "yes"

```
test@test:~/orangepi-build$ vim userpatches/config-default.conf
IGNORE_UPDATES="yes"
```

9) When debugging u-boot code, you can use the following method to update u-boot in the linux image for testing

a. Upload the compiled u-boot deb package to the linux OS of the development board

```
test@test:~/orangepi-build$ cd output/debs/u-boot
test@test:~/orangepi_build/output/debs/u-boot$ scp \
linux-u-boot-legacy-orangepizero2_2.0.8_arm64.deb root@192.168.1.xxx:/root
```

b. Then log in to the development board, uninstall the installed deb package of u-boot

```
root@orangepi:~# apt purge -y linux-u-boot-orangepizero2-legacy
```

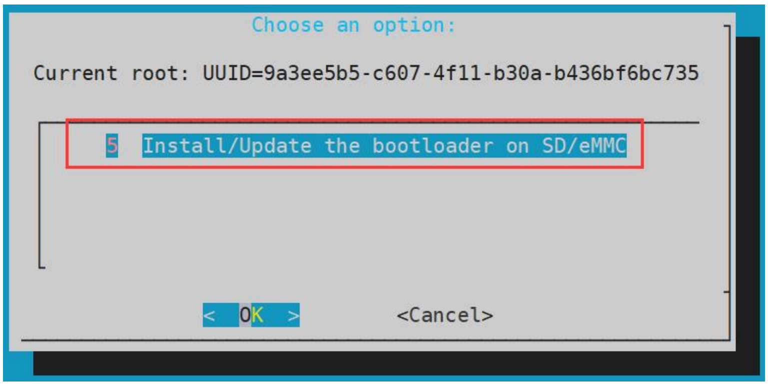
c. Install the new u-boot deb package just uploaded

```
root@orangepi:~# dpkg -i linux-u-boot-legacy-orangepizero2_2.0.8_arm64.deb
```

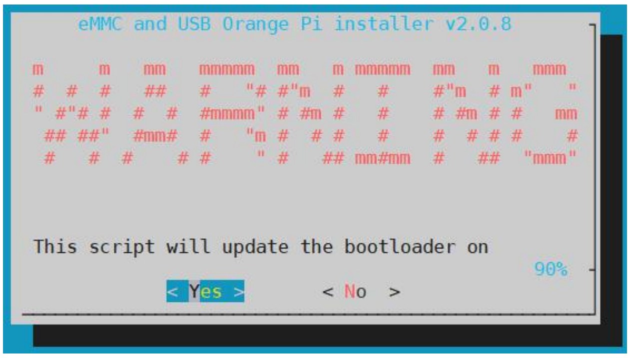
d. Then run the nand-sata-install script

```
root@orangepi:~# nand-sata-install
```

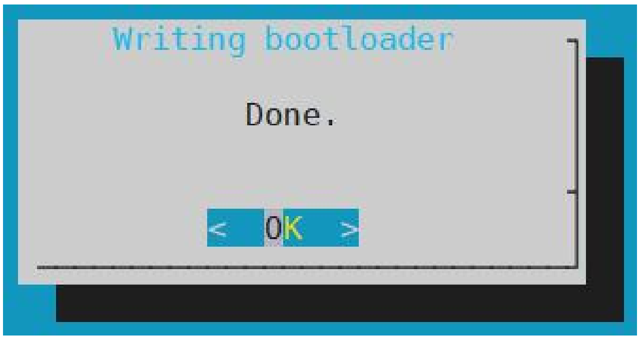
e. Then select **5 Install/Update the bootloader on SD/eMMC**



f. After pressing the Enter key, a Warring will pop up first



g. Press Enter again to start updating u-boot, and the following information will be displayed after the update



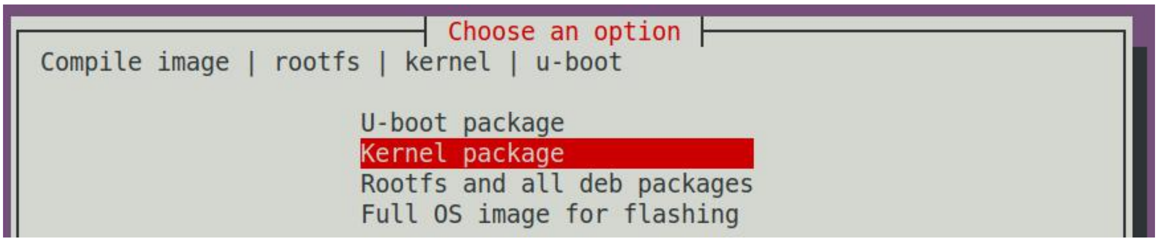
h. Then you can restart to test whether the u-boot modification has taken effect

4.4. Compile the Linux kernel

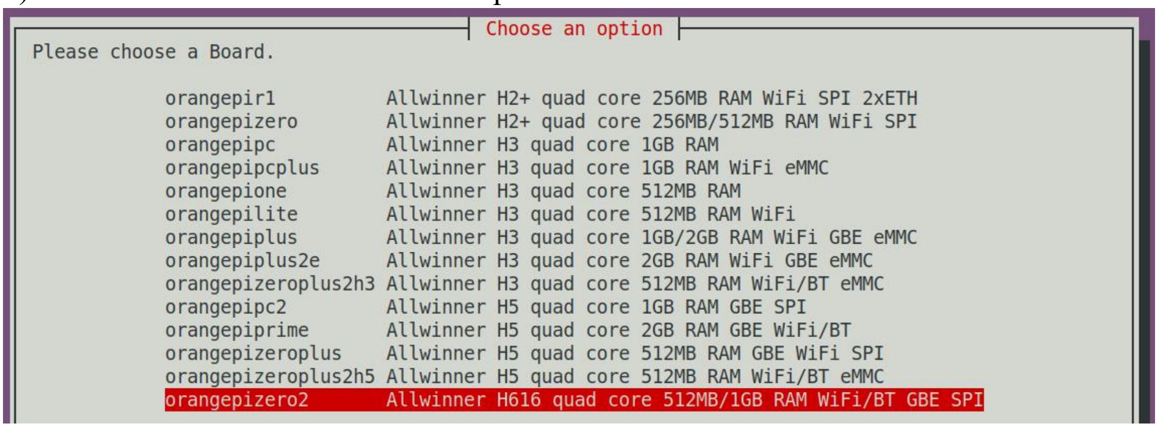
1) Run the build.sh script, remember to add sudo permissions

```
test@test:~/orangepi-build$ sudo ./build.sh
```

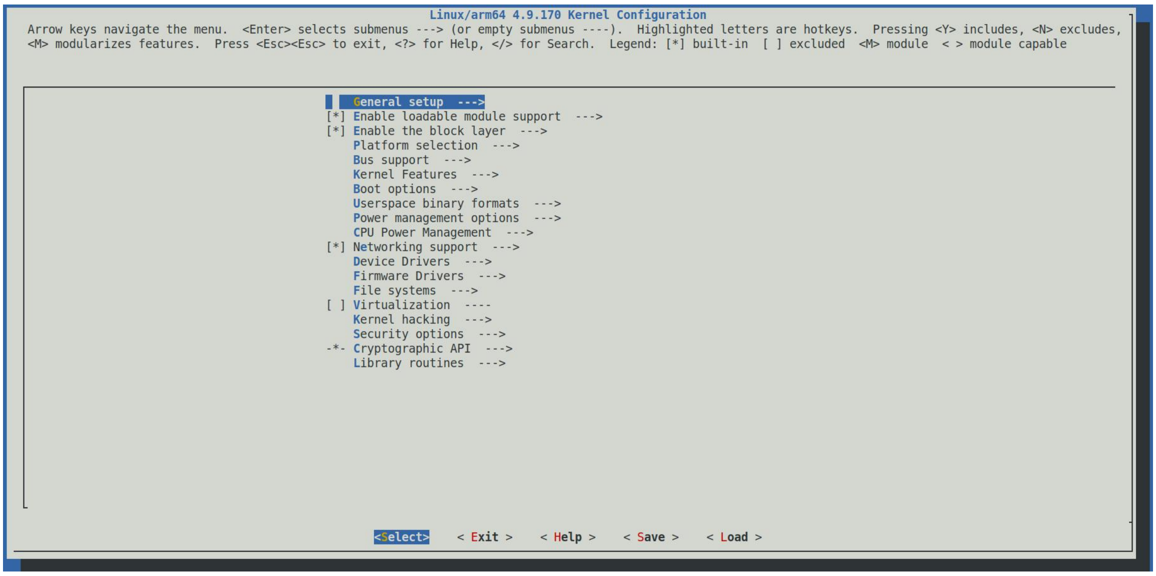
2) Select **Kernel package**, then press Enter



3) Then select the model of the development board



4) Then the kernel configuration interface opened through **make menuconfig** will pop up. At this time, you can directly modify the kernel configuration. If you don't need to modify the kernel configuration, just exit directly. After exiting, the kernel source code will be compiled



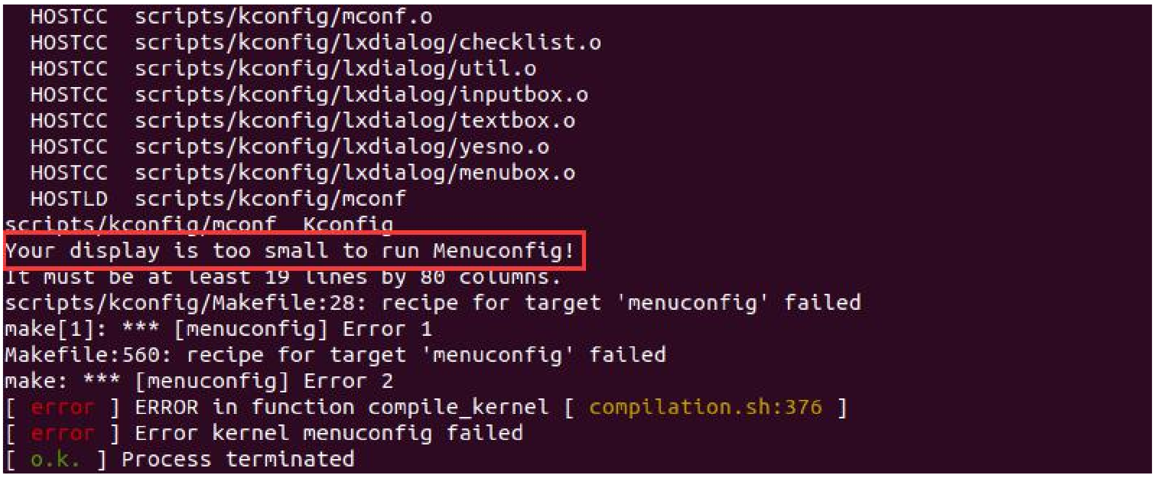
```
Linux/arm64 4.9.170 Kernel Configuration
Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
<M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

    General setup  --->
[*] Enable loadable module support  --->
[*] Enable the block layer  --->
    Platform selection  --->
    Bus support  --->
    Kernel Features  --->
    Boot options  --->
    Userspace binary formats  --->
    Power management options  --->
    CPU Power Management  --->
[*] Networking support  --->
    Device Drivers  --->
    Firmware Drivers  --->
    File systems  --->
[ ] Virtualization  ----
    Kernel hacking  --->
    Security options  --->
-*- Cryptographic API  --->
    Library routines  --->

<Select>    < Exit >    < Help >    < Save >    < Load >
```

a. If you do not need to modify the configuration options of the kernel, when you run the build.sh script, pass in **KERNEL_CONFIGURE=no** to temporarily block the pop-up kernel configuration interface

test@test:~/orangepi-build$ **sudo ./build.sh KERNEL_CONFIGURE=no**

b. You can also set **KERNEL_CONFIGURE=no** in the orangepi-build/userpatches/config-default.conf configuration file to disable this feature permanently

c. If the following error is prompted when compiling the kernel, this is because the terminal interface of the Ubuntu PC is too small and the make menuconfig interface cannot be displayed. Please adjust the terminal of the Ubuntu PC to the maximum, and then rerun the build.sh script



```
  HOSTCC  scripts/kconfig/mconf.o
  HOSTCC  scripts/kconfig/lxdialog/checklist.o
  HOSTCC  scripts/kconfig/lxdialog/util.o
  HOSTCC  scripts/kconfig/lxdialog/inputbox.o
  HOSTCC  scripts/kconfig/lxdialog/textbox.o
  HOSTCC  scripts/kconfig/lxdialog/yesno.o
  HOSTCC  scripts/kconfig/lxdialog/menubox.o
  HOSTLD  scripts/kconfig/mconf
scripts/kconfig/mconf  Kconfig
Your display is too small to run Menuconfig!
It must be at least 19 lines by 80 columns.
scripts/kconfig/Makefile:28: recipe for target 'menuconfig' failed
make[1]: *** [menuconfig] Error 1
Makefile:560: recipe for target 'menuconfig' failed
make: *** [menuconfig] Error 2
[ error ] ERROR in function compile_kernel [ compilation.sh:376 ]
[ error ] Error kernel menuconfig failed
[ o.k. ] Process terminated
```

5） When compiling the kernel source code, the following information will be prompted

a. The version of the kernel source code

[o.k.] Compiling legacy kernel [**4.9.170**]

b. The version of the cross-compilation toolchain

```
[ o.k. ] Compiler version [ aarch64-none-linux-gnu-gcc 9.2.1 ]
```

c. The default configuration file used by the kernel and its storage path

```
[ o.k. ] Using kernel config file [ config/kernel/linux-sun50iw9-legacy.config ]
```

d. If **KERNEL_CONFIGURE=yes,** the final configuration file .config used by the kernel will be copied to **output/config.** If the kernel configuration is not modified, the final configuration file is consistent with the default configuration file

```
[ o.k. ] Exporting new kernel config [ output/config/linux-sun50iw9-legacy.config ]
```

e. The path of the deb package related to the compiled kernel

```
[ o.k. ] Target directory [ output/debs/ ]
```

f. The package name of the compiled kernel image deb package

```
[ o.k. ] File name [ linux-image-legacy-sun50iw9_2.0.8_arm64.deb ]
```

g. Compilation time

```
[ o.k. ] Runtime [ 5 min ]
```

h. In the end, it will display the compiling command to recompile the kernel selected last time. Use the following command without selecting through the graphical interface, you can directly start compiling the kernel source code

```
[ o.k. ] Repeat Build Options [ sudo ./build.sh  BOARD=orangepizero2
BRANCH=legacy BUILD_OPT=kernel KERNEL_CONFIGURE=yes ]
```

6) View the deb package related to the kernel generated by the compilation

a. **linux-dtb-legacy-sun50iw9_2.0.8_arm64.deb** is not used yet, don't care about it
b. **linux-headers-legacy-sun50iw9_2.0.8_arm64.deb** contains kernel header files
c. **linux-image-legacy-sun50iw9_2.0.8_arm64.deb** contains kernel image and kernel module

```
test@test:~/orangepi-build$ ls output/debs/linux-*
output/debs/linux-dtb-legacy-sun50iw9_2.0.8_arm64.deb
output/debs/linux-headers-legacy-sun50iw9_2.0.8_arm64.deb
output/debs/linux-image-legacy-sun50iw9_2.0.8_arm64.deb
```

7) The files contained in the generated linux-image deb package are as follows

a. Use the following command to unzip the deb package

```
test@test:~/orangepi-build$ cd output/debs
test@test:~/orangepi_build/output/debs$ mkdir test
test@test:~/orangepi_build/output/debs$ cp \
linux-image-legacy-sun50iw9_2.0.8_arm64.deb test/
test@test:~/orangepi_build/output/debs$ cd test
test@test:~/orangepi_build/output/debs/test$ dpkg -x \
```

```
linux-image-legacy-sun50iw9_2.0.8_arm64.deb .
test@test:~/orangepi_build/output/debs/test$ ls
boot  etc  lib  linux-image-legacy-sun50iw9_2.0.8_arm64.deb  usr
```

b. The decompressed file is as follows

```
test@test:~/orangepi_build/output/debs/test$ tree -L 2
.
├── boot
│   ├── config-4.9.170-sun50iw9
│   ├── System.map-4.9.170-sun50iw9
│   └── vmlinuz-4.9.170-sun50iw9
├── etc
│   └── kernel
├── lib
│   └── modules
├── linux-image-legacy-sun50iw9_2.0.8_arm64.deb
└── usr
    ├── lib
    └── share

8 directories, 4 files
```

8) When the orangepi-bulid build OS compiles the linux kernel source code, it first synchronizes the linux kernel source code with the linux kernel source code of the github server, so if you want to modify the source code of the linux kernel, you first need to turn off the source code update function **(you need to complete the compilation once This function can only be turned off after the linux kernel source code, otherwise it will be prompted that the source code of the linux kernel cannot be found),** otherwise the changes made will be restored.

Set the IGNORE_UPDATES variable in userpatches/config-default.conf to "no"

```
test@test:~/orangepi-build$ vim userpatches/config-default.conf
IGNORE_UPDATES="no"
```

9) If you modify the kernel, you can use the following methods to update the kernel and kernel modules of the Linux OS on the development board

a. Upload the compiled linux deb package to the linux OS of the development board

```
test@test:~/orangepi-build$ cd output/debs
test@test:~/orangepi-build/output/debs$ scp \
linux-image-legacy-sun50iw9_2.0.8_arm64.deb root@192.168.1.207:/root
```

b. Then log in to the development board, uninstall the installed deb package of u-boot

```
root@orangepi:~# apt purge -y linux-image-legacy-sun50iw9
```

c. Install the new u-boot deb package just uploaded

```
root@orangepi:~# dpkg -i linux-image-legacy-sun50iw9_2.0.8_arm64.deb
```

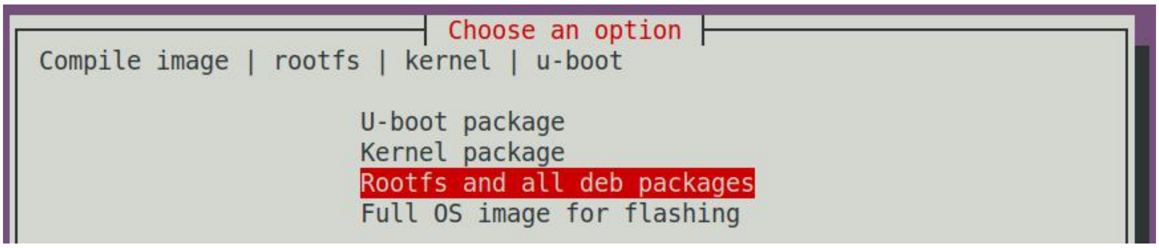
d. Then restart the development board, and then check whether the kernel-related changes have taken effect

4.5. Compile rootfs

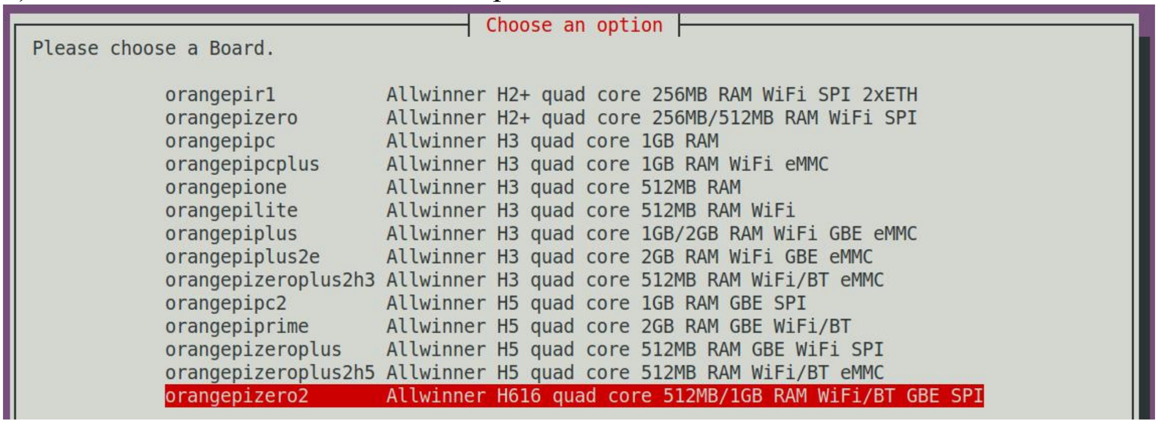
1) Run the build.sh script, remember to add sudo permissions

```
test@test:~/orangepi-build$ sudo ./build.sh
```

2) Select **Rootfs and all deb packages**, then press Enter



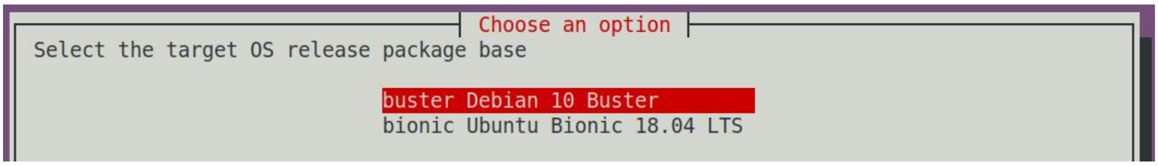
3) Then select the model of the development board



4) Then select the type of rootfs

a. **buster** means Debian 10

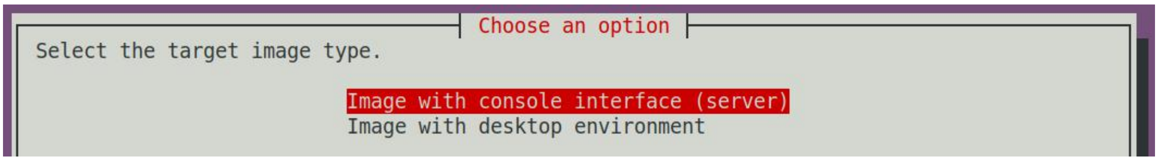
b. **bionic** means Ubuntu 18.04



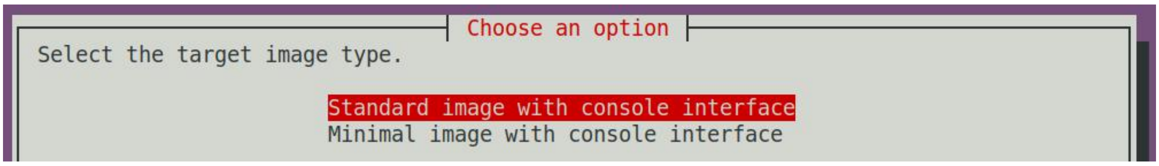
5) Then select the type of image

a. **Image with console interface** represents the image of the server version, which is relatively small

b. **Image with the desktop environm**ent means that the image with a desktop version is relatively large



6) If you are compiling the server version of the image, you can also choose to compile the Standard version or the Minimal version. The pre-installed software of the Minimal version will be much less than the Standard version



7) After selecting the type of image, rootfs will be compiled, and the following information will be prompted during the compilation

a.Type of rootfs

[o.k.] local not found [Creating new rootfs cache for **bionic**]

b. The storage path of the compiled rootfs compressed package

[o.k.] Target directory [**external/cache/rootfs**]

e. The name of the rootfs compressed package generated by the compilation

[o.k.] File name [**bionic-cli-arm64.153618961f14c28107ca023429aa0eb9.tar.lz4**]

f. Compilation time

[o.k.] Runtime [**13 min**]

g. Repeat the command to compile rootfs, use the following command without selecting through the graphical interface, you can directly start compiling rootfs

[o.k.] Repeat Build Options [**sudo ./build.sh BOARD=orangepizero2 BRANCH=legacy BUILD_OPT=rootfs RELEASE=bionic BUILD_MINIMAL=no BUILD_DESKTOP=no KERNEL_CONFIGURE=yes**]

8) View the compiled rootfs compressed package

a. bionic-cli-arm64.153618961f14c28107ca023429aa0eb9.tar.lz4 is a compressed package of rootfs, the meaning of each field of the name is

a. **bionic** represents the type of linux distribution of rootfs
b. **cli** means rootfs is the server version type, if it is desktop, it means the desktop version type
c. **arm64** represents the architecture type of rootfs
d. **153618961f14c28107ca023429aa0eb9** is the MD5 hash value generated by the package names of all software packages installed by rootfs. As long as the list of software packages installed by rootfs is not modified, this value will not change. The compilation script will use this MD5 hash value. Determine whether you

need to recompile rootfs

b. bionic-cli-arm64.153618961f14c28107ca023429aa0eb9.tar.lz4.list **lists** the package names of all packages installed by rootfs

```
test@test:~/orangepi-build$ ls external/cache/rootfs/
bionic-cli-arm64.153618961f14c28107ca023429aa0eb9.tar.lz4 bionic-cli-arm64.153618961f14c28107ca023429aa0eb9.tar.lz4.list
```

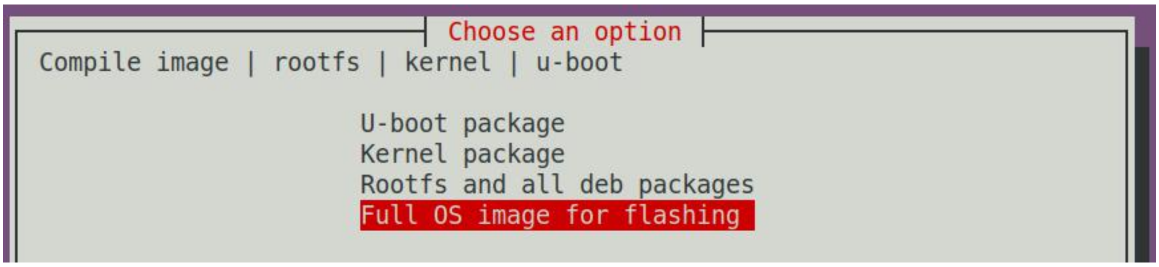
9) If the required rootfs already exists under **external/cache/rootfs,** then compiling the rootfs again will skip the compilation process and will not restart the compilation. When compiling the image, it will also go to **external/cache/rootfs** to check whether it is already Rootfs with cache available, if available, use it directly, which can save a lot of download and compilation time

4.6. Compile linux image

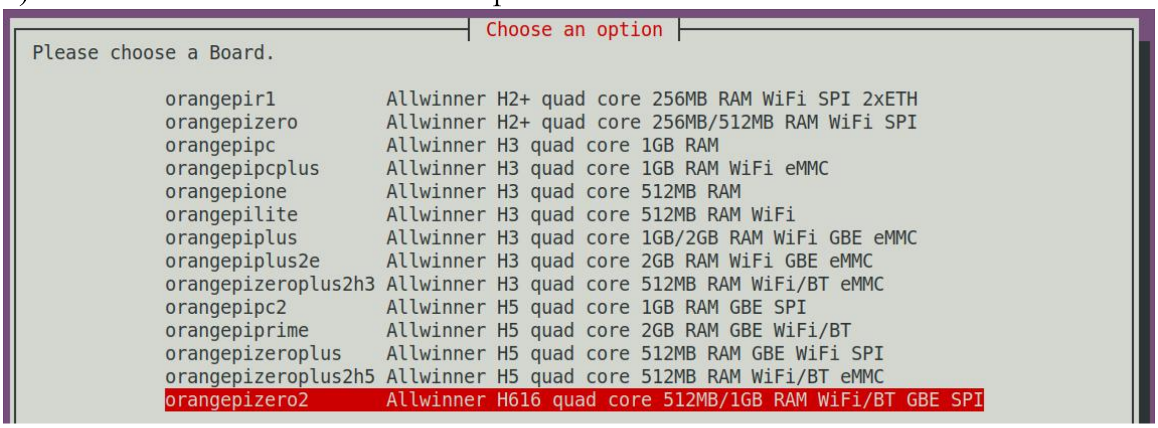
1) Run the build.sh script, remember to add sudo permissions

```
test@test:~/orangepi-build$ sudo ./build.sh
```

2) Select **Full OS image for flashing**, then press Enter



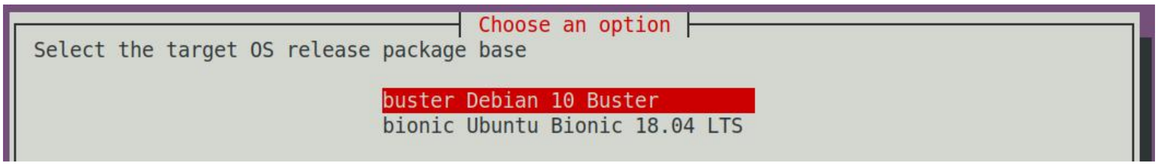
3) Then select the model of the development board



4) Then select the type of rootfs

a. **buster** means Debian 10

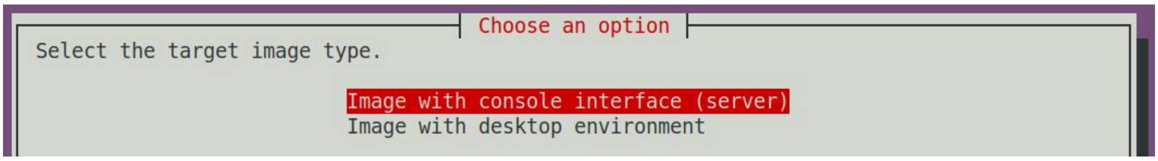
b. **bionic** means Ubuntu 18.04



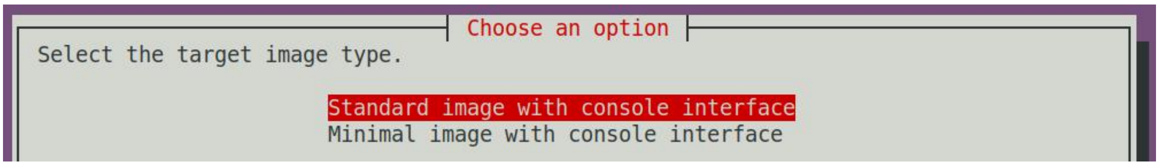
4) Then select the type of image

a. **Image with console interface** represents the image of the server version, which is relatively small

b. **Image with the desktop environment** means that the image with a desktop version is relatively large



5) If you are compiling the server version of the image, you can also choose to compile the Standard version or the Minimal version. The pre-installed software of the Minimal version will be much less than the Standard version.



6) After selecting the type of image, it will start to compile the Linux image. The general process of compilation is as follows

a. Initialize the compilation environment of Ubuntu PC and install the software packages needed for the compilation process

b. Download the source code of u-boot and linux kernel

c. Compile u-boot, generate u-boot deb package

d. Compile linux source code and generate linux related deb package

e. Make deb package of linux firmware

f. Make deb package of orangepi-config tool

g. Make board-level support deb package

h. If it is to compile the desktop version image, it will also make desktop related deb packages

i. Check whether the rootfs has been cached, if there is no cache, then re-create the rootfs, if it has been cached, directly unzip and use

j. Install the previously generated deb package into rootfs

k. Make some specific settings for different development boards and different types of images, such as pre-installing additional software packages, modifying configuration files, etc.

l. Then make an image file and format the partition, the default type is ext4

m. Copy the configured rootfs to the image partition

n. Then update the initramfs

o. Finally, the bin file of u-boot is written to the image through the dd command

7) After compiling the image, the following information will be prompted

a. The storage path of the compiled image

```
[ o.k. ] Done building
[ output/images/Orangepizero2_2.0.8_ubuntu_bionic_server_linux4.9.170/Orangepizero2_2.0.8_ubuntu_bionic_server_linux4.9.170.img ]
```

b. Compilation time

```
[ o.k. ] Runtime [ 19 min ]
```

c. Repeat the command to compile the image, use the following command without selecting through the graphical interface, you can directly start to compile the image

```
[ o.k. ] Repeat Build Options [ sudo ./build.sh  BOARD=orangepizero2 BRANCH=legacy BUILD_OPT=image RELEASE=bionic BUILD_MINIMAL=no BUILD_DESKTOP=no KERNEL_CONFIGURE=yes ]
```

5. Android OS instructions

5.1. Supported Android version

Android version	kernel version
Android 10.0	linux4.9

5.2. Android 10 feature adaptation

Features	Status
HDMI video	OK
HDMI audio	OK
USB2.0 x 3	OK
TF card boot	OK
Network card	OK
IR	OK
WIFI	OK
WIFI hotsport	OK
Bluetooth	OK
Bluetooth earphone	OK
BLE Bluetooth	OK
CVBS audio	OK
CVBS video	OK
USB camera	OK
LED lights	OK
Temperature Sensor	OK

Hardware watchdog	OK
Mali GPU	OK
Video codec	OK

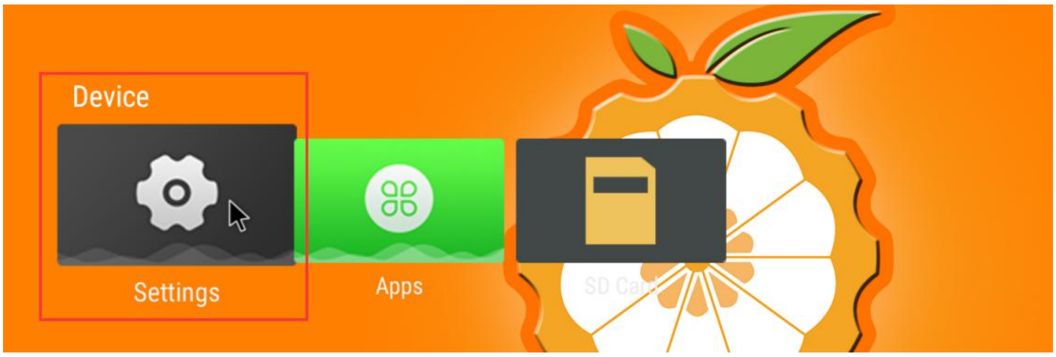
5.3. Onboard LED light display description

	Green light	Red light
U-boot startup phase	OFF	ON
Kernel boot to enter system	ON	OFF
GPIO	PC13	PC12

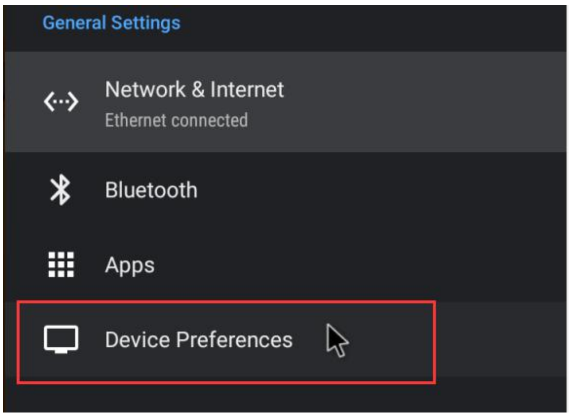
5.4. How to use ADB?

5.4.1. Open USB debugging option

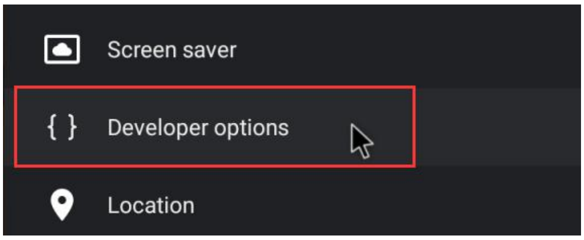
1) Select **Settings**



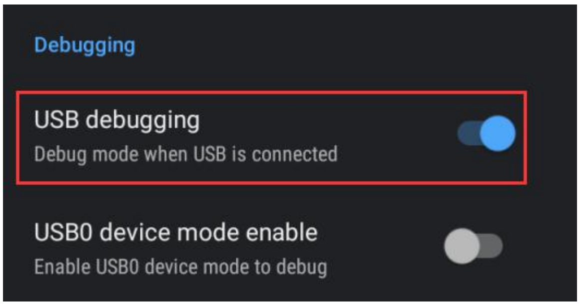
2) Then select **Device Preferences**



3) Then select **Developer options**



4) Finally find **USB debugging**, make sure it is turned on



5.4.2. Use data cable to connect adb debugging

1) First make sure to turn on the **USB debugging option**

2) Prepare a USB Type C interface data cable. One end of the USB interface is inserted into the USB interface of the computer, and the other end of the Type C interface is inserted into the power interface of the development board. In this case, the USB interface of the computer supplies power to the development board, so please ensure that the USB interface of the computer can provide the powerful to drive the development board.

3) Install adb tool on Ubuntu PC

```
test@test:~$ sudo apt update
test@test:~$ sudo apt install adb
```

4) View the identified ADB device

```
test@test:~$ adb devices
List of devices attached
8c00141167058911ccd     device
```

5) Then you can log in to the android OS through adb shell on the Ubuntu PC

```
test@test:~$ adb shell
cupid-p2:/ #
```

5.4.3. Use network connection adb debugging

1) Using the network adb does not require a USB Typc C interface data cable to connect the computer and the development board, but communicates through the network, so first make sure that the wired or wireless network of the development board is connected, and then obtain the IP address of the development board for to be used next

2) Make sure that the **USB debugging option** is turned on

3) Make sure that the **service.adb.tcp.port** of the Android OS is set to port number 5555

```
cupid-p2:/ # getprop | grep "adb.tcp"
[service.adb.tcp.port]: [5555]
```

5) If service.adb.tcp.port is not set, you can use the following command to set the port number of the network adb

```
cupid-p2:/ # setprop service.adb.tcp.port 5555
cupid-p2:/ # stop adbd
cupid-p2:/ # start adbd
```

6) Install adb tool on Ubuntu PC

```
test@test:~$ sudo apt update
test@test:~$ sudo apt install adb
```

7) Then connect to the network adb on Ubuntu PC

```
test@test:~$ adb connect 192.168.1.xxx    (The IP address needs to be modified to the IP address of the development board)
* daemon not running; starting now at tcp:5037
* daemon started successfully
connected to 192.168.1.xxx:5555

test@test:~$ adb devices
List of devices attached
192.168.1.xxx:5555       device
```

6) Then you can log in to the android OS through adb shell on Ubuntu PC

```
test@test:~$ adb shell
cupid-p2:/ #
```

5.5. How to use a USB camera

1) First insert the USB camera into the USB interface of the development board, and then confirm that the kernel module related to the USB camera has been loaded normally

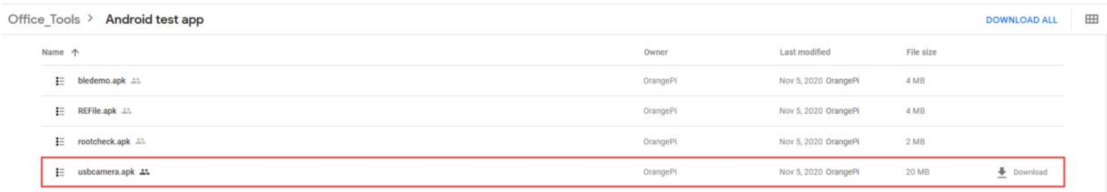
```
console:/ # lsmod
Module                  Size  Used by
sprdwl_ng             405504  0
sprdbt_tty             36864  2
uwe5622_bsp_sdio      274432  2 sprdwl_ng,sprdbt_tty
uvcvideo              102400  0
videobuf2_v4l2         28672  1 uvcvideo
videobuf2_vmalloc      16384  1 uvcvideo
videobuf2_memops       16384  1 videobuf2_vmalloc
videobuf2_core         49152  2 uvcvideo,videobuf2_v4l2
mali_kbase            532480  7
```

2) If the USB camera is recognized normally, the corresponding video device node will be generated under **/dev**

```
console:/ # ls /dev/video0
/dev/video0
console:/ # ls -l /sys/class/video4linux/ -lh
total 0
lrwxrwxrwx 1 root root 0 2020-11-02 20:46:01.187678078 +0800 video0 -> ../../devices/platform/soc/5200000.ehci1-controller/usb1/1-1/1-1:1.0/video4linux/video0
console:/ #
```

3) Then make sure that the adb connection between the Ubuntu PC and the development board is normal

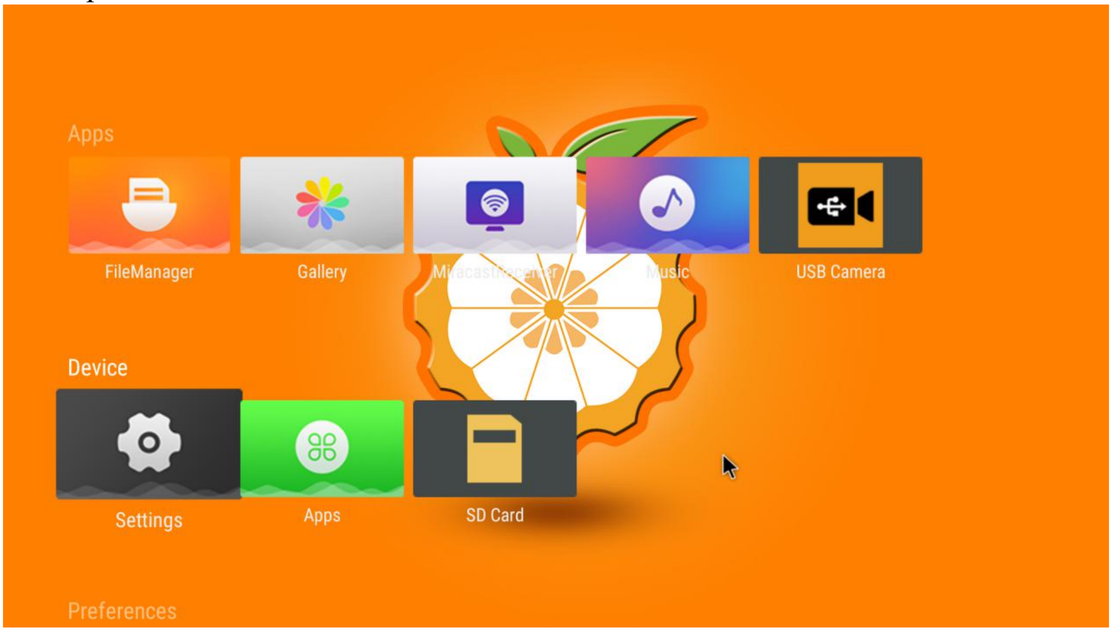
4) Download the USB camera test APP from the official tool on the page below the Orange Pi Zero 2 information



5) Then use the adb command to install the USB camera test APP to the Android OS, of course, you can also use the U disk copy method to install

```
test@test:~$ adb install usbcamera.apk
```

6) After installation, you can see the startup icon of the USB camera on the Android desktop



7) Then double-click to open the USB camera APP and you can see the output video of the USB camera

5.6. Android OS ROOT description

The Android 10.0 OS released by Orange Pi is already ROOT, you can use the following method to test

1) Download rootcheck.apk from the official tool on the Orange Pi Zero 2 data download page

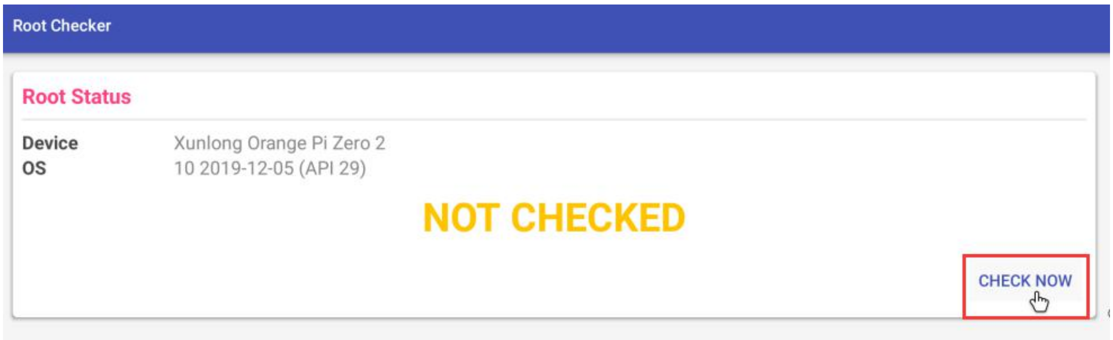
2) Then make sure that the adb connection between the Ubuntu PC and the development board is normal

3) Then use the adb command to install rootcheck.apk to the Android OS, of course, you can also use the U disk copy to install

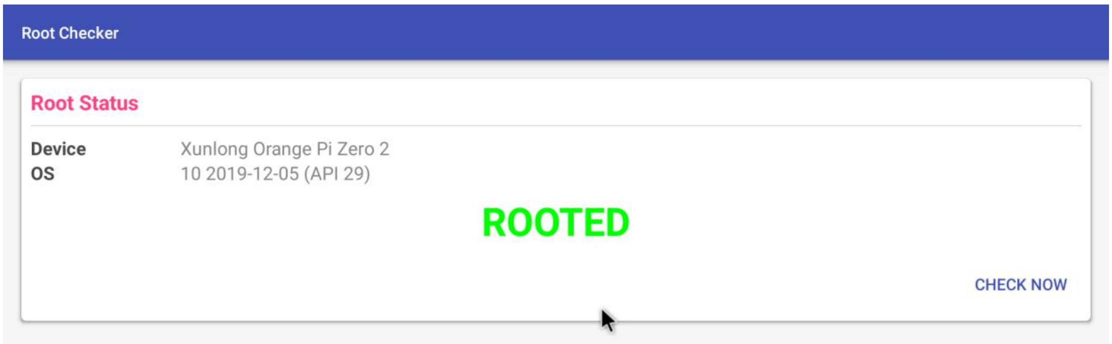
```
test@test:~$ adb install rootcheck.apk
```

4) After installation, you can see the startup icon of the ROOT test tool on the Android desktop

5) The display interface after opening the **ROOT test tool** for the first time is as shown in the figure below



6) Then you can click **"Check now"** to start the inspection of the ROOT status of the Android OS. The display after the inspection is as follows, you can see that the Android OS has obtained ROOT permission



6. Android SDK instructions

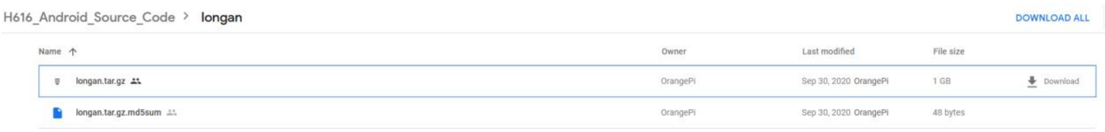
1) The compilation of the Android SDK is performed on a PC with Ubuntu 14.04 installed, and other versions of Ubuntu OS may have some differences

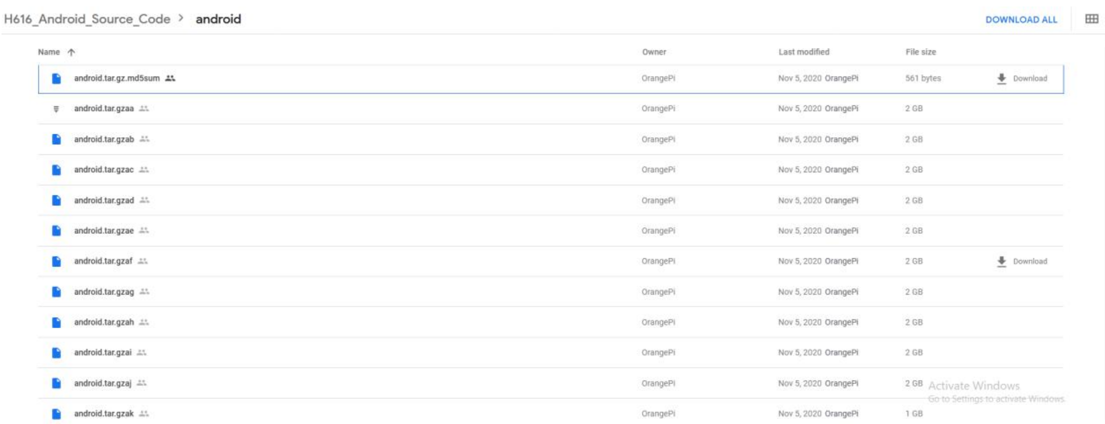
2) Android SDK is the original SDK released by the chip manufacturer. If you want to use the Android image compiled by the Android SDK on the Orange Pi development board, you need to adapt to different boards to ensure that all functions are used normally

6.1. Download the source code of android SDK

1) The android source code of H616 contains the following 4 files

a. **android.tar.gzxx:** android source code
b. **android.tar.gz.md5sum:** MD5 checksum file of android.tar.gzxx
c. **longan.tar.gz:** Contains u-boot, linux kernel source code (does not include boot0 source code)
d. **longan.tar.gz.md5sum:** MD5 checksum file of longan.tar.gz





2) After downloading the android source code, first check whether the MD5 checksum is correct, if not, please download the source code again

```
test@test:~$ md5sum -c android.tar.gz.md5sum
android.tar.gzaa: OK
android.tar.gzab: OK
android.tar.gzac: OK
android.tar.gzad: OK
android.tar.gzae: OK
android.tar.gzaf: OK
android.tar.gzag: OK
```

```
android.tar.gzah: OK
android.tar.gzai: OK
android.tar.gzaj: OK
android.tar.gzak: OK
test@test:~$ md5sum -c longan.tar.gz.md5sum
longan.tar.gz: OK
```

3) Then unzip the android source code

a. android: store android-related source code

b. longan: store the source code of the linux kernel and u-boot (not including the source code of boot0), and other configuration files

```
test@test:~$ cat android.tar.gza* | tar -zx
test@test:~$ tar -zxf longan.tar.gz
test@test:~$ ls
android  longan
```

6.2. Build android compilation environment

1) Install JDK

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

2) Configure JAVA environment variables

a. First determine the installation path of java, generally

```
test@test:~$ ls /usr/lib/jvm/java-8-openjdk-amd64
ASSEMBLY_EXCEPTION  bin  docs  include  jre  lib  man  src.zip
THIRD_PARTY_README
```

b. Then use the following command to export java environment variables

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

3) Use Ubuntu 14.04 to compile the source code of android 10, you need to ensure that Ubuntu 14.04 uses the **linux 4.4 kerne**l, otherwise an error will be reported when compiling, if the kernel is not linux 4.4, please upgrade the kernel

```
test@test:~$ uname -a
Linux ubuntu 4.4.0-142-generic #168~14.04.1-Ubuntu SMP Sat Jan 19 11:26:28
UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
```

4) Install platform support software

```
test@test:~$ sudo apt-get update
test@test:~$ 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

test@test:~$ sudo apt-get install u-boot-tools
```

6.3. Compile android image

6.3.1. Compile the kernel

1) First configure the compilation environment

```
test@test:~$ cd longan
test@test:~/longan$ ./build.sh config

Welcome to mkscript setup progress
All available platform:
   0. android
   1. linux
Choice [android]: 0
All available ic:
   0. h313
   1. h616
   2. h700
Choice [h616]: 1
All available board:
   0. fpga
   1. ft
   2. p1
   3. p2
   4. perf1
   5. perf1_axp152
   6. perf2
   7. perf3
   8. qa
Choice [p2]: 3
INFO: kernel defconfig: generate /wspace2/H616/Android_10/longan/kernel/linux-4.9/.config by /wspace2/H616/Android_10/longan/kernel/linux-4.9/arch/arm64/configs/sun50iw9p1smp_h616_android_defconfig
*** Default configuration is based on 'sun50iw9p1smp_h616_android_defconfig'
#
# configuration written to .config
```

```
#
```

2) Then start compiling

```
test@test:~/longan$ ./build.sh
```

3) The output after compilation is as follows

```
sun50iw9p1 compile Kernel successful

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

6.3.2. Compile android source code

1) The command to compile android is as follows

```
test@test:~$ cd android
test@test:~/android$ source build/envsetup.sh
test@test:~/android$ lunch cupid_p2-eng
test@test:~/android$ extract-bsp
test@test:~/android$ make -j8
```

2) After compiling, the following information will be printed

```
#### build completed successfully (01:51 (mm:ss)) ####
```

3) Then use the pack command to package and generate the android image

```
test@test:~/android$ pack
......
----------image is at----------

longan/out/h616_android10_p2_uart0.img

pack finish
use pack4dist for release
```

4) The storage path of the generated Android image is

```
longan/out/h616_android10_p2_uart0.img
```