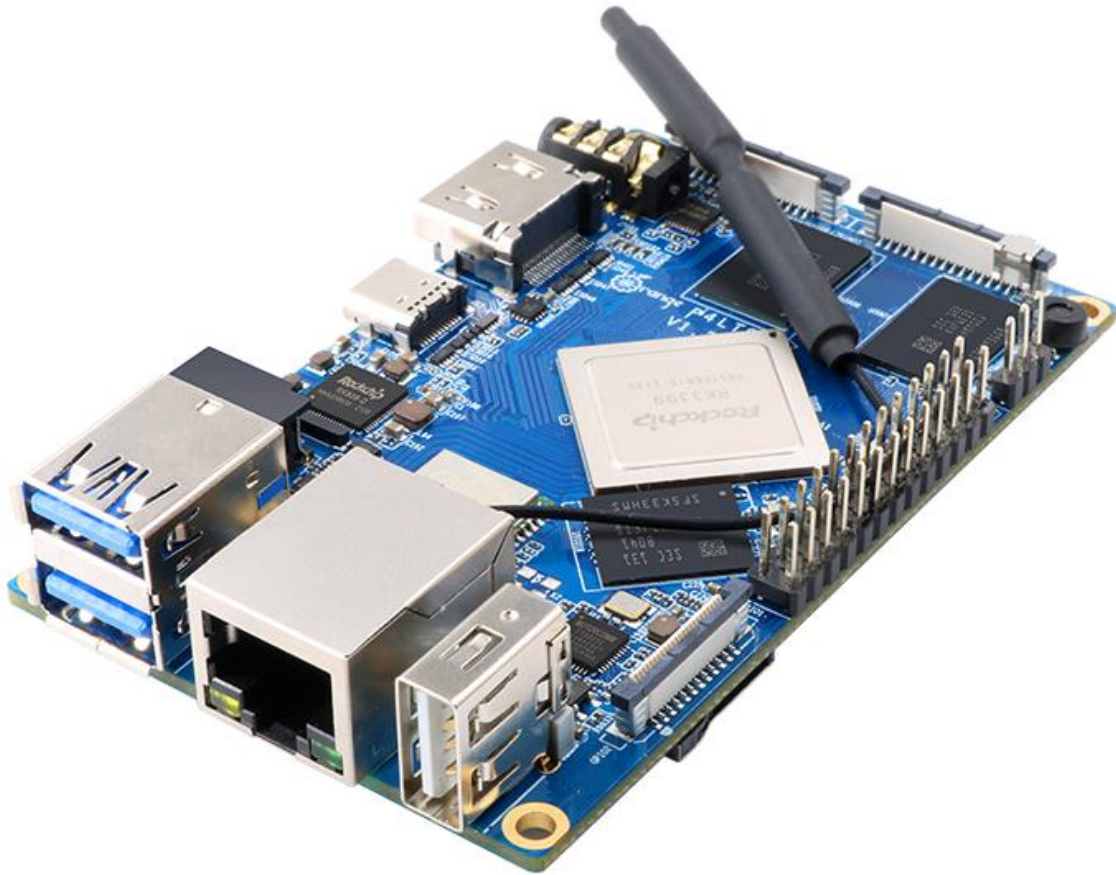




Orange Pi 4 LTS

User Manual





Contents

- 1. Basic Features of Orange Pi 4 LTS..... 1
 - 1.1. What is Orange Pi 4 LTS? 1
 - 1.2. Purpose of Orange Pi 4 LTS..... 1
 - 1.3. Who is Orange Pi 4 LTS designed for?..... 1
 - 1.4. Hardware specification..... 2
 - 1.5. Top view and bottom view of Orange Pi 4 LTS..... 4
 - 1.5.1. RK3399+4GB LPDDR4 with 16GB eMMC version..... 4
 - 1.5.2. RK3399+4GB LPDDR4 without 16GB eMMC version..... 5
 - 1.5.3. RK3399-T+3GB LPDDR4 with 16GB eMMC version..... 6
 - 1.5.4. RK3399-T+3GB LPDDR4 without 16GB eMMC version..... 7
 - 1.6. Interface details of Orange Pi 4 LTS..... 8
 - 1.6.1. RK3399+4GB LPDDR4..... 8
 - 1.6.2. RK3399-T+3GB LPDDR4..... 9
- 2. Introduction to the use of the development board..... 11
 - 2.1. Prepare the necessary accessories..... 11
 - 2.2. Download the image of the development board and related files..... 15
 - 2.3. Use the Android image pre-installed in eMMC to test the function of the development board..... 15
 - 2.4. Method of burning Linux image to TF card based on Windows PC..... 16
 - 2.4.1. How to use Win32Diskimager to burn Linux image..... 16
 - 2.4.2. How to use balenaEtcher to burn a Linux image..... 18
 - 2.5. The method of burning Linux image to TF card based on Ubuntu PC..... 22
 - 2.6. Method of programming Linux image to eMMC..... 26
 - 2.7. How to burn Android firmware to TF card..... 26
 - 2.8. The method of burning Android firmware to eMMC based on Windows PC..... 29
 - 2.8.1. Directly burn Android firmware to eMMC through Type C interface..... 29



2.8.2. Burn Android image to eMMC via TF card.....	35
2.9. Method of Burning Android Image to eMMC Based on Ubuntu PC.....	38
2.10. How to enter MaskRom mode.....	40
2.11. Start the orange pi development board.....	41
2.12. How to debug the serial port.....	43
2.12.1. Connection instructions for debugging serial port.....	43
2.12.2. How to use the debugging serial port on Ubuntu platform.....	45
2.12.3. How to use the debugging serial port on Windows platform.....	48
3. Instructions for use of Linux system.....	52
3.1. Supported linux distribution types and kernel versions.....	52
3.2. Linux kernel driver adaptation.....	52
3.3. Onboard LED light display description.....	53
3.4. Linux system default login account and password.....	53
3.5. Instructions for automatic login of Linux desktop system.....	53
3.6. Start the rootfs in the auto-expanding TF card for the first time.....	55
3.7. How to modify the linux log level (loglevel).....	57
3.8. Ethernet port test.....	58
3.9. How to set a static IP address.....	60
3.10. SSH remote login development board.....	67
3.10.1. SSH remote login development board under Ubuntu.....	67
3.10.2. SSH remote login development board under Windows.....	68
3.11. HDMI display test.....	69
3.12. Type C to HDMI display test.....	70
3.13. HDMI to VGA display test.....	70
3.14. WIFI connection test.....	71
3.14.1. The server version image is connected to WIFI through the command..	72
3.14.2. Connect WIFI graphically in the command line.....	73
3.14.3. How to use the Linux desktop to connect to WIFI.....	77



3.15. How to use Bluetooth.....	79
3.15.1. Test method of desktop version image.....	79
3.15.2. How to use the server version image.....	83
3.16. USB interface test.....	85
3.16.1. Connect mouse or keyboard test.....	85
3.16.2. Connect USB storage device test.....	85
3.17. USB camera test.....	86
3.18. Audio Test.....	89
3.18.1. Headphone jack audio playback test.....	89
3.18.2. Onboard MIC recording test.....	90
3.18.3. Headphone recording.....	90
3.18.4. HDMI audio playback test.....	91
3.19. Temperature sensor.....	91
3.20. How to use Mini-PCIE.....	92
3.21. GPU test method.....	93
3.22. MPV hardware decoding test.....	95
3.23. Chromium hardware acceleration test.....	97
3.24. How to install Docker.....	98
3.25. 26pin interface pin description.....	100
3.26. How to install wiringOP.....	101
3.27. 26pin interface GPIO, I2C, UART, SPI, PWM test.....	102
3.27.1. 26pin GPIO port test.....	102
3.27.2. 26pin SPI test.....	104
3.27.3. 26pin I2C test.....	105
3.27.4. 26pin UART test.....	106
3.27.5. 26pin PWM test.....	110
3.28. How to use 0.96-inch OLED module with I2C interface.....	111
3.29. Hardware watchdog test.....	114
3.30. How to program linux image to eMMC.....	114



3.31. How to use Linux 4.4 OV13850 camera.....	118
3.31.1. Camera connection instructions.....	118
3.31.2. Using a single camera.....	119
3.31.3. Dual camera usage.....	121
3.32. 10.1 inch MIPI screen usage.....	122
3.32.1. LCD1 interface instructions.....	122
3.32.2. LCD2 interface instructions.....	125
3.33. How to set up dual screen display.....	129
3.34. How to use the orange pi DS1307 RTC clock module.....	129
3.35. Reset and shutdown methods.....	134
3.36. Partial programming language test supported by Linux system.....	135
4. Instructions for use of Android system.....	138
4.1. Supported Android Versions.....	138
4.2. Android 8.1 function adaptation.....	138
4.3. On-board LED light display description.....	139
4.4. How to use ADB.....	139
4.4.1. Use data cable to connect adb debugging.....	139
4.4.2. Using network connection adb debugging.....	140
4.5. How to use OV13850 camera.....	141
4.6. How to use the HDMI interface.....	144
4.7. The method of displaying the system interface through the TypeC interface.....	146
4.8. 10.1 inch MIPI screen usage.....	146
4.9. How to use the USB camera.....	150
4.10. Instructions for use of Mini-PCIE.....	151
4.10.1. Instructions for connecting mini-PCIE to hard disk.....	151
5. Instructions for using the Linux SDK.....	157
5.1. Compilation system requirements.....	157
5.2. Get the source code of linux sdk.....	157



- 5.2.1. Download orangepi-build from github..... 158
- 5.2.2. Download the cross-compilation toolchain..... 159
- 5.2.3. Orangepi-build complete directory structure description..... 161
- 5.3. Compile u-boot..... 162
- 5.4. Compile the linux kernel..... 166
- 5.5. Compile rootfs..... 171
- 5.6. Compile the linux image..... 176
- 6. Instructions for using Android SDK..... 182
 - 6.1. Download the source code of Android SDK..... 182
 - 6.2. Build Android Compilation Environment..... 183
 - 6.3. Compile Android image..... 183
 - 6.3.1. Compiling u-boot..... 183
 - 6.3.2. Compile the kernel..... 184
 - 6.3.3. Compile android..... 184
 - 6.3.4. Packaging the full image..... 185



1. Basic Features of Orange Pi 4 LTS

1. 1. What is Orange Pi 4 LTS?

Orange Pi is an open source single-board card computer, a new generation of arm64 development boards, which can run operating systems such as Android 8.1, Ubuntu and Debian. Orange Pi development board (Orange Pi 4 LTS) uses Rockchip RK3399 or RK3399-T SoC, and has 3GB or 4GB LPDDR4 memory

1. 2. Purpose of Orange Pi 4 LTS

we can use it to build:

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

Of course there are many more features as Orange Pi is open source

1. 3. Who is Orange Pi 4 LTS designed for?


The Orange Pi development board is not just a consumer product, it is designed for anyone who wants to use technology to create and innovate. It's a very simple, fun, and useful tool that you can use to shape the world around you



1. 4. Hardware specification

Hardware specification	
CPU	<ul style="list-style-type: none"> • Rockchip RK3399 or RK3399-T • 6-core ARM® 64-bit processor • The main frequency of RK3399 is up to 1.8GHz, and the main frequency of RK3399-T is up to 1.6GHz • Based on big.LITTLE large and small core architecture: Dual-core Cortex-A72 (large core) + quad-core Cortex-A53 (small core)
GPU	<ul style="list-style-type: none"> • High-performance multi-core GPU Mali T864 • OpenGL ES 1.1/2.0/3.0 • OpenCL 1.0/1.1/1.2 • DirectX 9/11.1
RAM	3GB or 4GB LPDDR4
Onboard storage	<ul style="list-style-type: none"> • 16GB EMMC or Default Empty • TF card slot
Network	10/100/1000Mbps Ethernet (YT8531C)
WIFI+Bluetooth	<ul style="list-style-type: none"> • UWE5622, IEEE 802.11 a/b/g/n/ac • BT5.0
video output	HDMI 2.0 x 1(Type-A), support 4K@60 frame output DP 1.2 x1 (DisplayPort) , support 4K@60 frame output Supports dual channel MIPI-DSI (4 wires per channel)
video input	MIPI-CSI x2 camera interface (MIPI_RX0, MIPI_TX1/RX1)
Audio output	<ul style="list-style-type: none"> • 3.5mm headphone jack • HDMI
audio input	<ul style="list-style-type: none"> • Onboard MIC



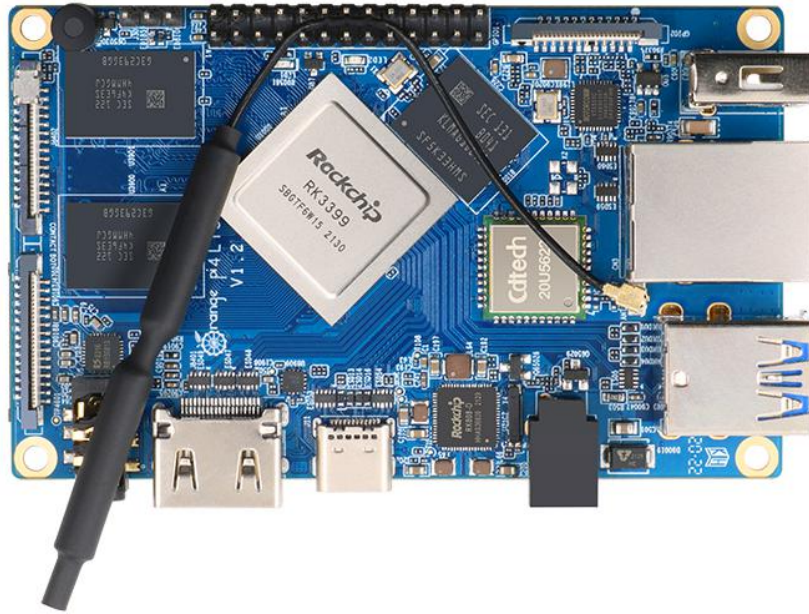
	<ul style="list-style-type: none"> • Headphone recording
power supply	<ul style="list-style-type: none"> • DC 5V/3A or DC 5V/4A • TYPE-C 5V/3A or TYPE-C 5V/4A
USB port	<ul style="list-style-type: none"> • USB2.0 HOST x 2 • USB3.0 HOST x 1 • USB3.0 Type-C x 1
26 pins header	with I2Cx2, SPIx1 or UARTx1 and multiple GPIOs
Mini-PCIE	24pin mini-PCIE interface
Debug serial port	UART-TX、 UART-RX and GND
LED	Power led & Status led
Button	Reset button x1, upgrade button x1
Supported OS	Android8.1、 Ubuntu、 Debian
Appearance Specifications	
Dimension	91mm×56mm
 range Pi™ is a trademark of the Shenzhen Xunlong Software CO., Limited	



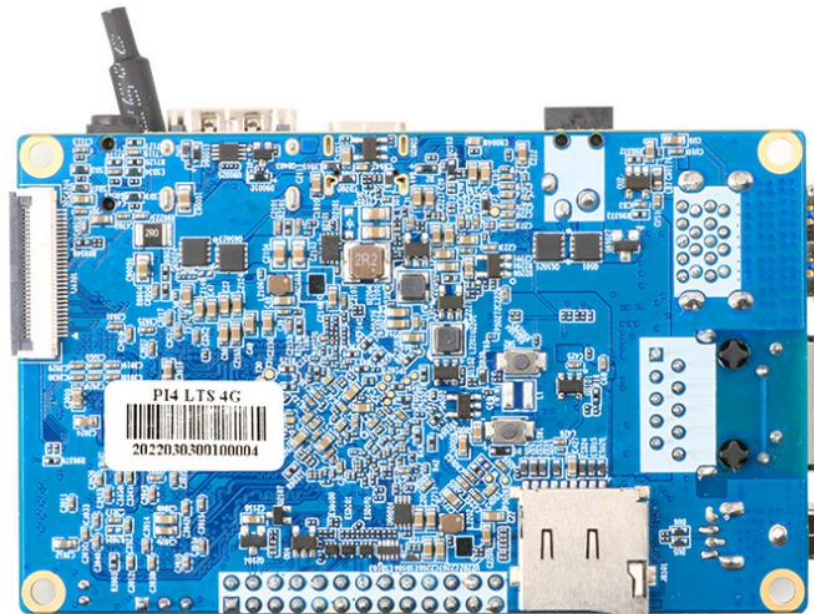
1. 5. Top view and bottom view of Orange Pi 4 LTS

1. 5. 1. RK3399+4GB LPDDR4 with 16GB eMMC version

Top view:



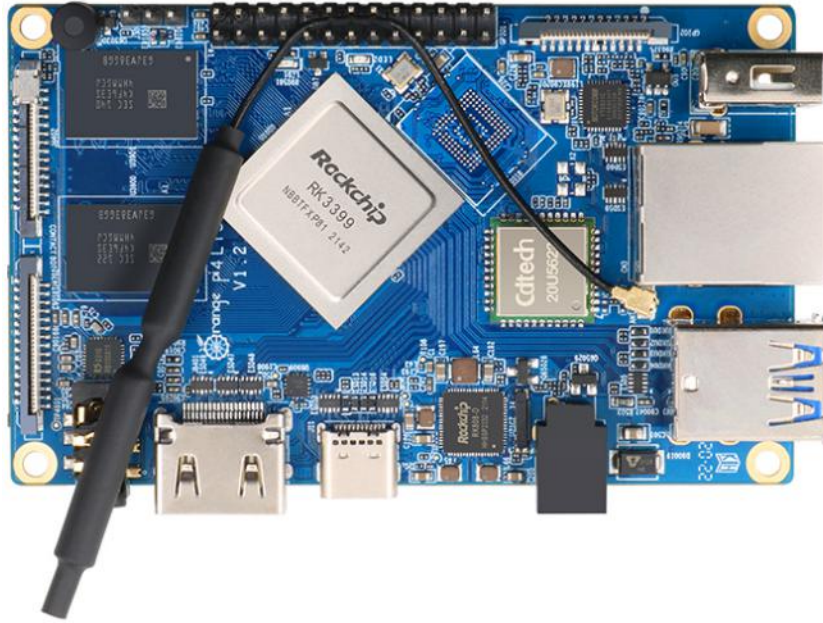
Bottom view:



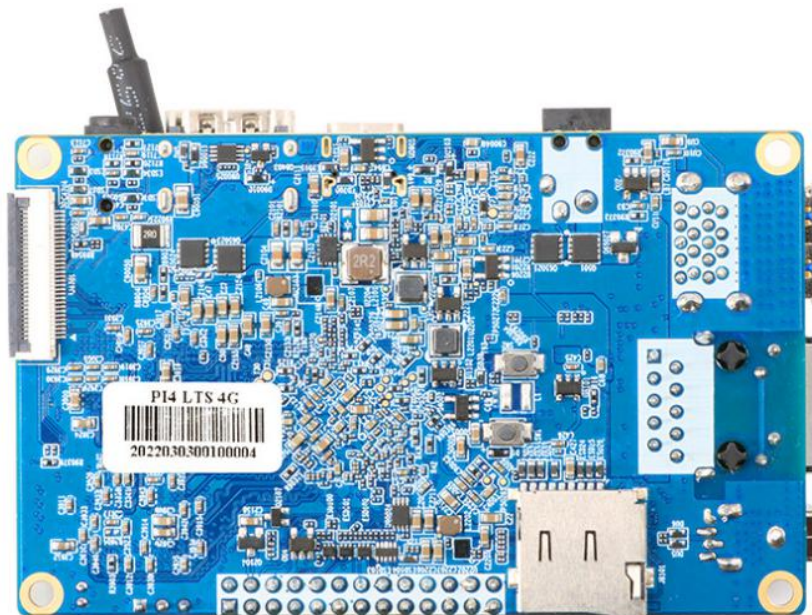


1. 5. 2. RK3399+4GB LPDDR4 without 16GB eMMC version

Top view:



Bottom view:



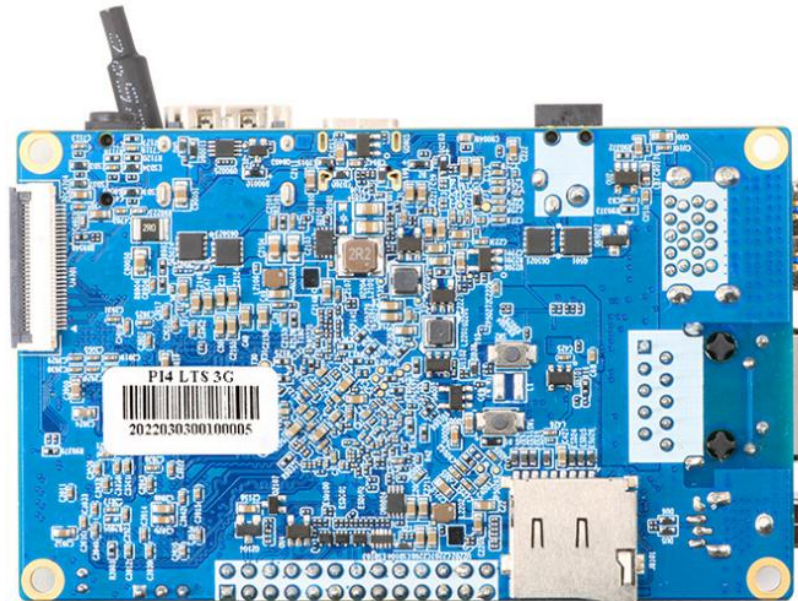


1. 5. 3. RK3399-T+3GB LPDDR4 with 16GB eMMC version

Top view:



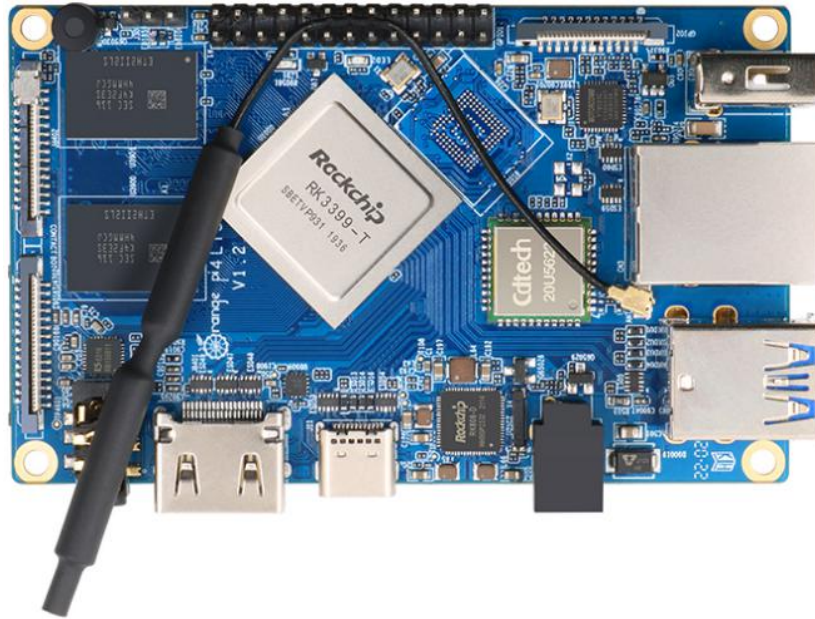
Bottom view:



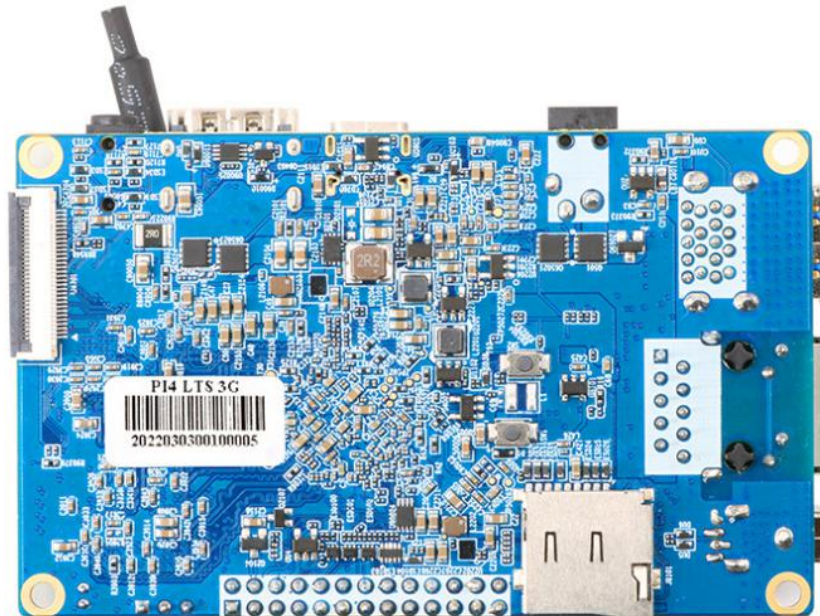


1. 5. 4. RK3399-T+3GB LPDDR4 without 16GB eMMC version

Top view:



Bottom view:

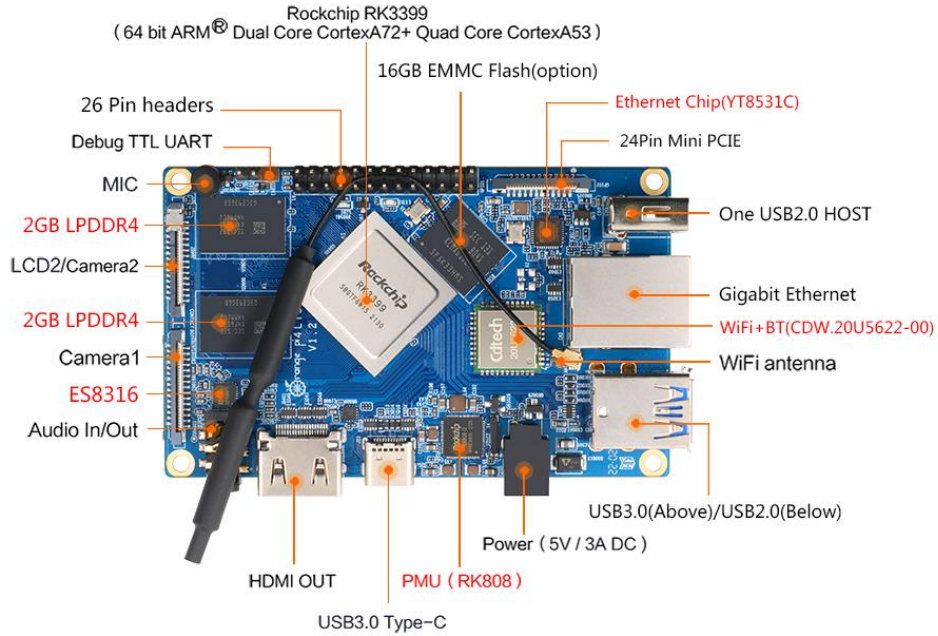




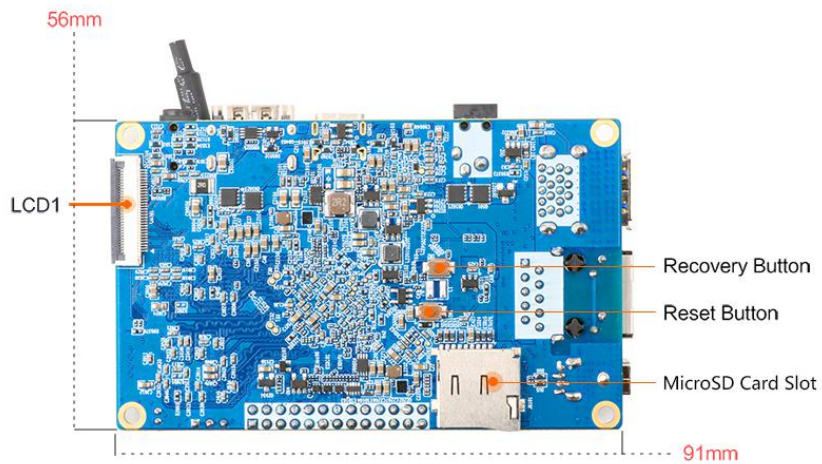
1. 6. Interface details of Orange Pi 4 LTS

1. 6. 1. RK3399+4GB LPDDR4

Top view



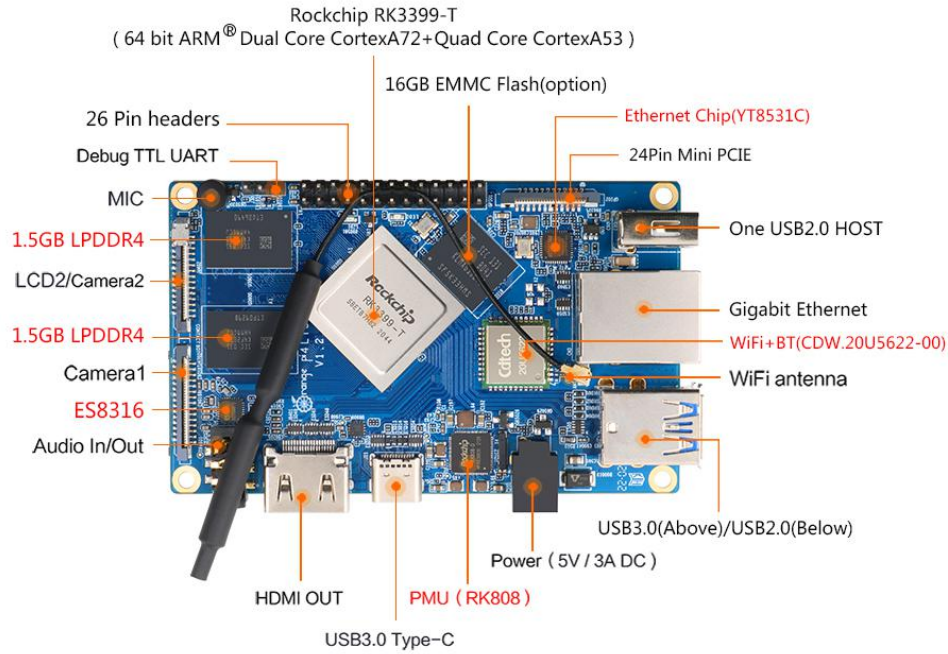
Bottom view



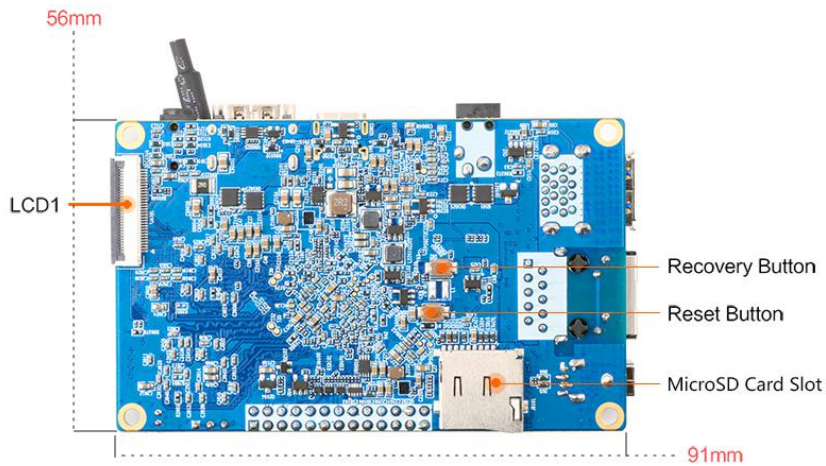


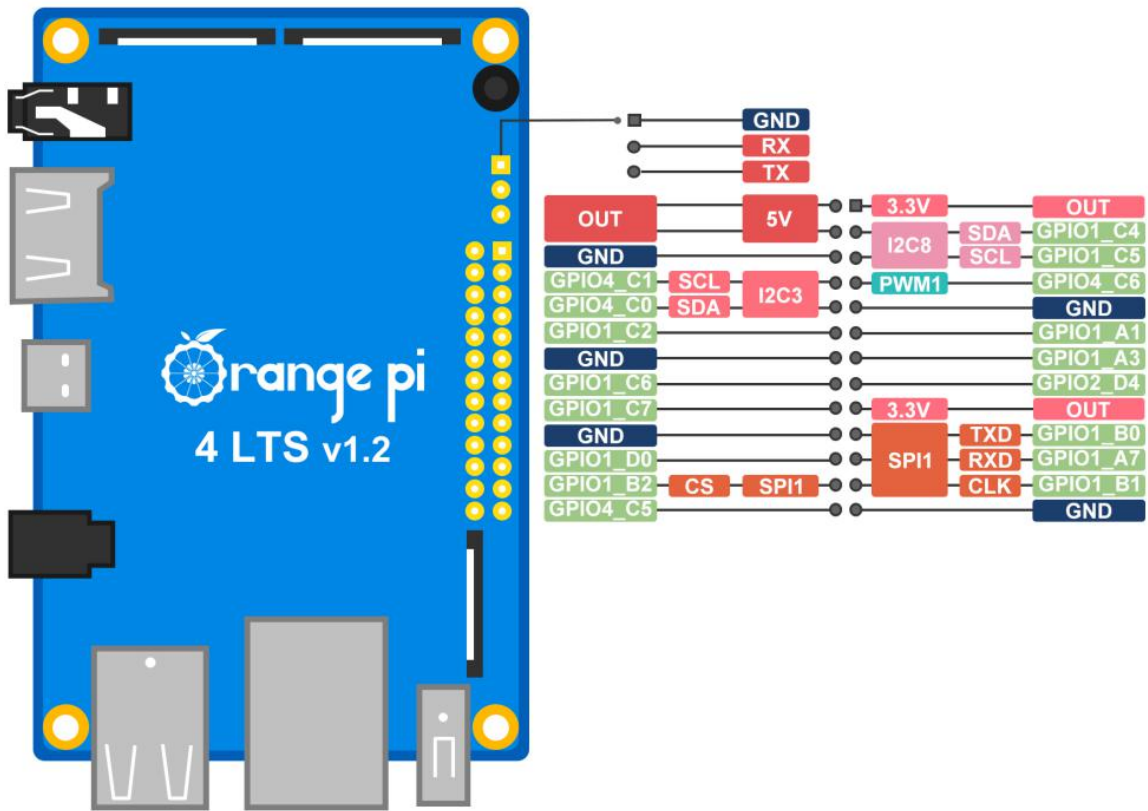
1. 6. 2. RK3399-T+3GB LPDDR4

|| Top view ||



|| Bottom view ||





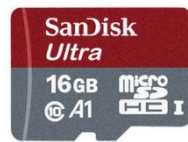


2. Introduction to the use of the development board

2.1. Prepare the necessary accessories

1) TF card, a high-speed card of class 10 or above with a minimum capacity of 8GB, it is recommended to use a SanDisk TF card, the Orange Pi test is to use a SanDisk TF card, other brands of TF cards may cause the system to fail to boot.

SanDisk 闪迪



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



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



4) Type-C to HDMI cable, connect the development board to HDMI display or TV through Type-C interface for display



5) Type-C to USB3.0 adapter, used to connect USB3.0 storage devices or USB devices such as mouse and keyboard



6) 10.1-inch MIPI screen, used to display the system interface of the development board



7) Power adapter, Orange Pi 4 LTS supports 5V/3A or 5V/4A DC power supply, and also supports 5V/3A or 5V/4A Type-C power supply



Note that: the Orange Pi 4 LTS cannot be powered through the 5v pin on the 26pin interface

- 8) USB interface mouse and keyboard, as long as it is a standard USB interface mouse and keyboard, the mouse and keyboard can be used to control the Orange Pi development board
- 9) Metal heat dissipation shell, Orange Pi 4 LTS matching metal shell is shown in the figure below, because the RK3399/RK3399-T chip generates a relatively large amount of heat, it is necessary to have a metal shell





10) If you don't buy a metal cooling case, it is recommended to add a 5V cooling fan. As shown in the figure below, there are 5V and GND pins on the 26pin interface of the development board that can be connected to the cooling fan. The spacing between the 26pin pin headers is 2.54mm. The power interface of the cooling fan can be purchased from Taobao according to this specification (the development board is plugged into the power supply) After the 5V pin can be used directly, no other settings are required)



11) Fast or Gigabit Ethernet cable to connect the development board to the Internet

12) OV13850 13MP camera, dedicated camera for Orange Pi 4 LTS, compatible with MIPI interface



13) 3.3V USB to TTL module and DuPont cable, when using the serial port debugging function, USB to TTL module and DuPont cable are required to connect the development board and computer



14) A PC with Ubuntu and Windows operating systems installed



1	Ubuntu14.04 PC	Optional, used to compile Android source code
2	Ubuntu21.04 PC	Optional, used to compile Linux source code
3	Windows PC	For burning Android and Linux images

2.2. Download the image of the development board and related files

1) The download URL of the Chinese version of the file is:

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

2) The download URL of the English version of the file is:

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

3) The information mainly includes

a. **Android source code**: saved on Baidu cloud disk and Google network disk

b. **Linux source code**: saved on github, the link address is

<https://github.com/orangepi-xunlong/orangepi-build>

c. **User manual and schematic diagram**: The data sheet related to the chip will also be placed here

d. **Official tools**: mainly include the software that needs to be used during the use of the development board

e. **Android image**: save on Baidu cloud disk and Google network disk

f. **Ubuntu image**: save on Baidu cloud disk and Google network disk

g. **Debian image**: save on Baidu cloud disk and Google network disk

2.3. Use the Android image pre-installed in eMMC to test the function of the development board

Note that if you purchase the version without eMMC, you cannot pass the Android image test pre-installed in eMMC, you can only burn the image to the TF card, and then start the system in the TF card to test the function of the development board

If you purchased the Orange Pi 4 LTS development board with 16GB eMMC, after getting the development board, you can use the Android 8.1 image pre-installed in the



eMMC to test the functions of the development board, and make sure that all hardware functions of the development board are OK. After that, burn the system you want to use.

2.4. Method of burning Linux image to TF card based on Windows PC

Note that: the Linux image mentioned here refers specifically to the image of a Linux distribution such as Debian or Ubuntu downloaded from the Orange Pi data download page

2.4.1. How to use Win32Diskimager to burn Linux image

1) First prepare a TF card with a capacity of 8GB or more. The transmission speed of the TF card must be above class 10. It is recommended to use a TF card from a brand such as SanDisk

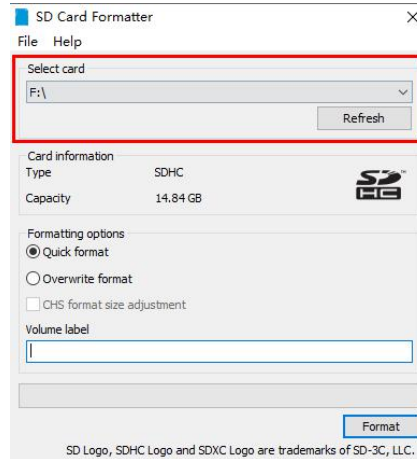
2) Then use the card reader to insert the TF card into the computer

3) Then format the TF card

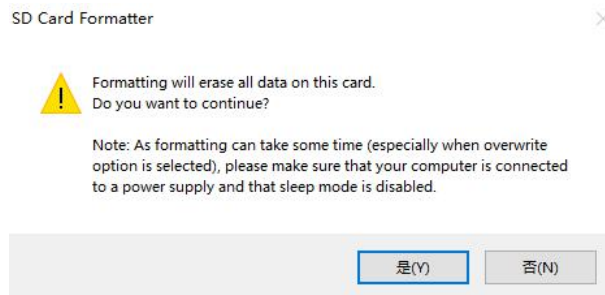
- a. The **SD Card Formatter** software can be used to format the TF card, and its download address is

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

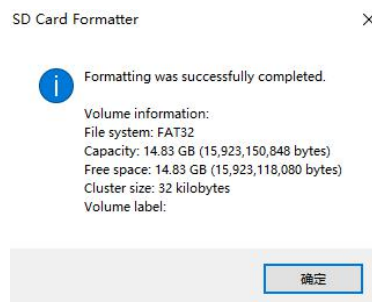
- b. After downloading, unzip and install directly, and then open the software
- c. If only the TF card is inserted into the computer, the “**Select card**” column will display the drive letter of the TF card. If multiple USB storage devices are inserted into the computer, you can select the drive letter corresponding to the TF card through the drop-down box.



- d. Then click "**Format**", a warning box will pop up before formatting, select "Yes (Y)" to start formatting



- e. After formatting the TF card, the information shown in the figure below will pop up, click OK.



4) Download the compressed package of the Linux operating system image file you want to burn from the data download page of Orange Pi, and then use the decompression software to decompress it. In the decompressed file, the file ending with ".img" is the image file of the operating system. The size is generally more than 1GB

5) Use **Win32Diskimager** to burn the Linux image to the TF card

- a. The download page of Win32Diskimager is

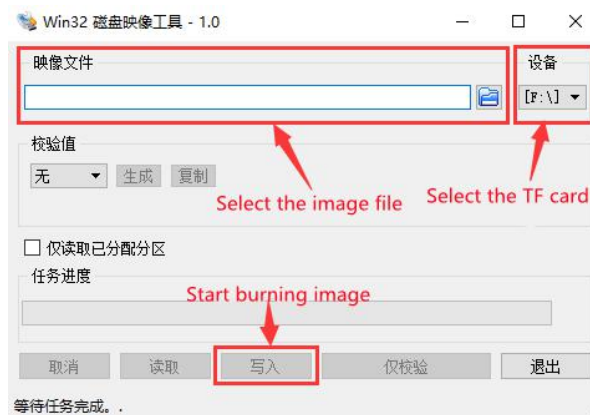
<http://sourceforge.net/projects/win32diskimager/files/Archive/>

- b. After downloading, install it directly. The Win32Diskimager interface is as follows

- a) First select the path of the image file

- b) Then confirm that the drive letter of the TF card is consistent with the one displayed in the "Device" column

- c) Finally click "Write" to start burning



- c. After the image writing is completed, click the "Exit" button to exit, and then you can pull out the TF card and insert it into the development board to start

2.4.2. How to use balenaEtcher to burn a Linux image

- 1) First prepare a TF card with a capacity of 8GB or more. The transmission speed of the TF card must be above class 10. It is recommended to use a TF card from a brand such as SanDisk

- 2) Then use the card reader to insert the TF card into the computer

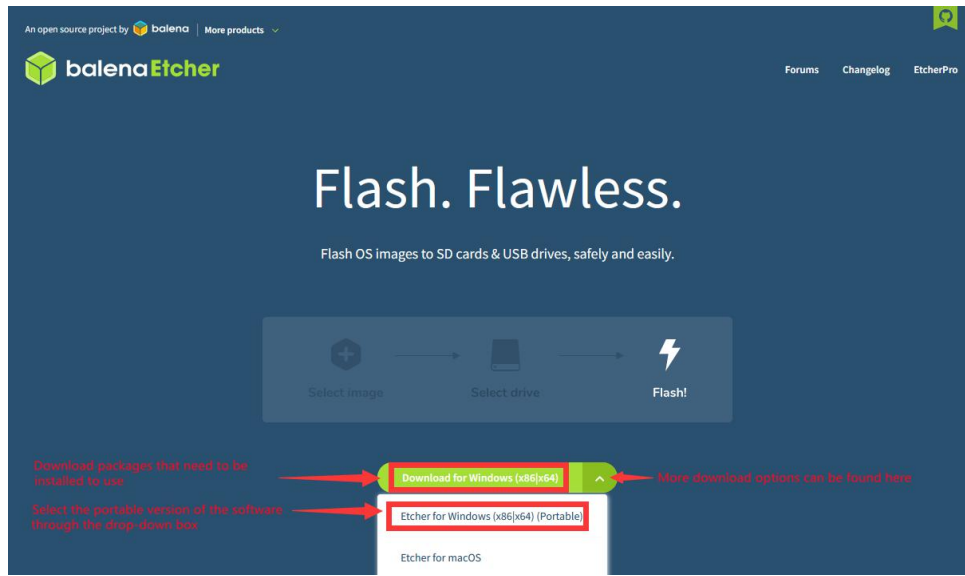
- 3) Download the compressed package of the Linux operating system image file you want to burn from the data download page of Orange Pi, and then use the decompression software to decompress it. In the decompressed file, the file ending with ".img" is the image file of the operating system. The size is generally more than 1GB

- 4) Then download the burning software of the Linux image - **balenaEtcher**, the download address is

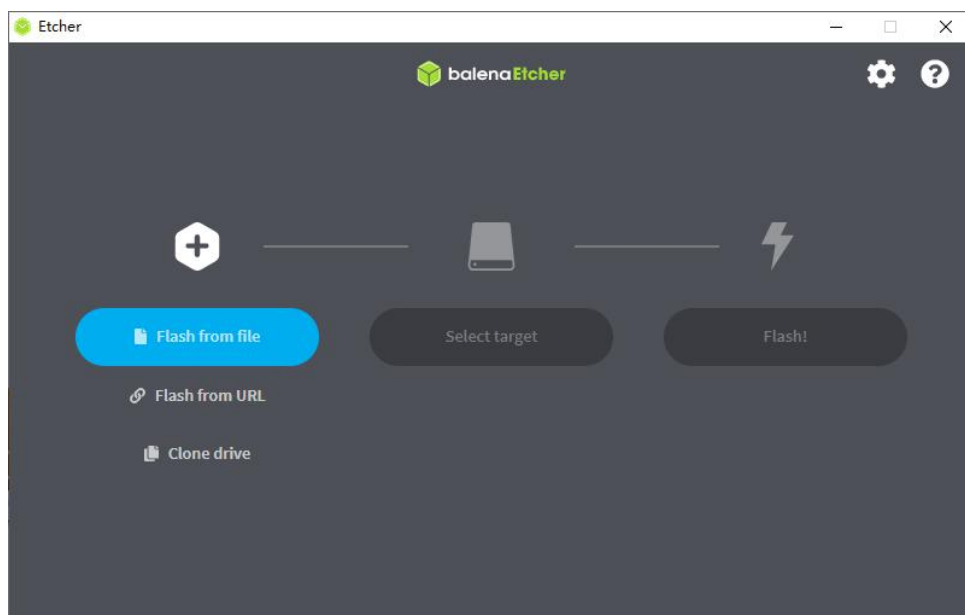


<https://www.balena.io/etcher/>

5) After entering the balenaEtcher download page, click the green download button to download the installation package of balenaEtcher. You can also select the Portable version of balenaEtcher through the drop-down box. The Portable version does not need to be installed. Double-click to open it and use it

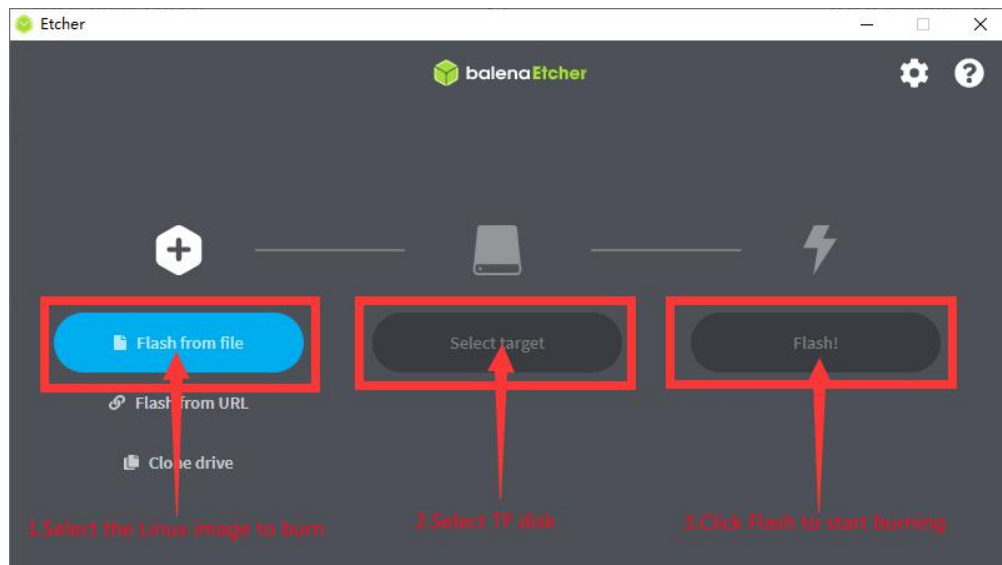


6) If you download a version of balenaEtcher that needs to be installed, please install it before using it. If you download the Portable version of balenaEtcher, just double-click to open it. The opened balenaEtcher interface is shown in the figure below.

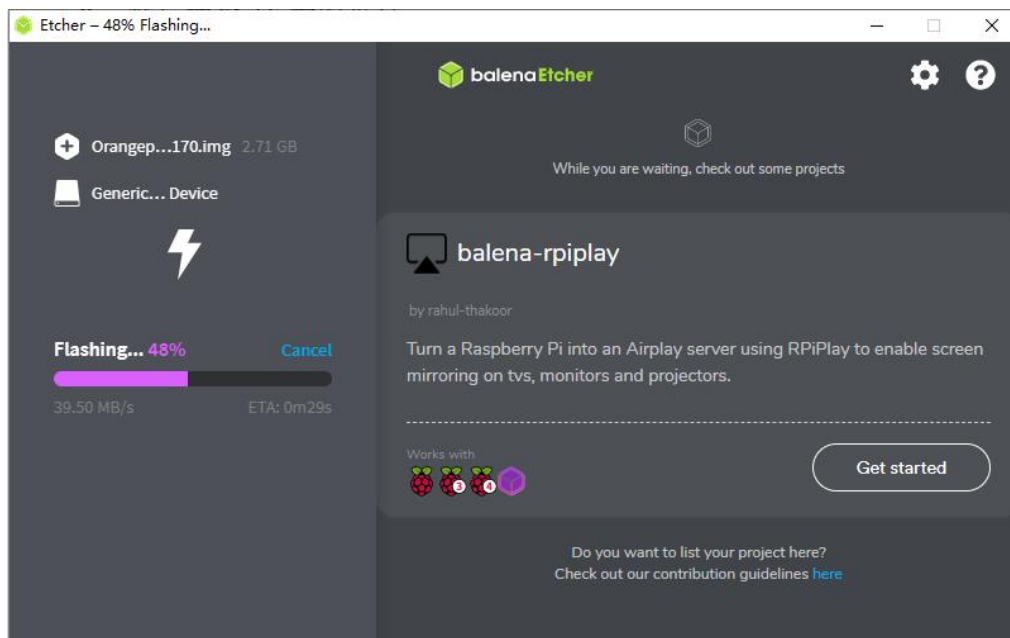




- 7) The specific steps to use balenaEtcher to burn a Linux image are as follows
 - a. First select the path of the Linux image file to be burned
 - b. Then select the drive letter of the TF card
 - c. Finally, click Flash to start burning the Linux image to the TF card

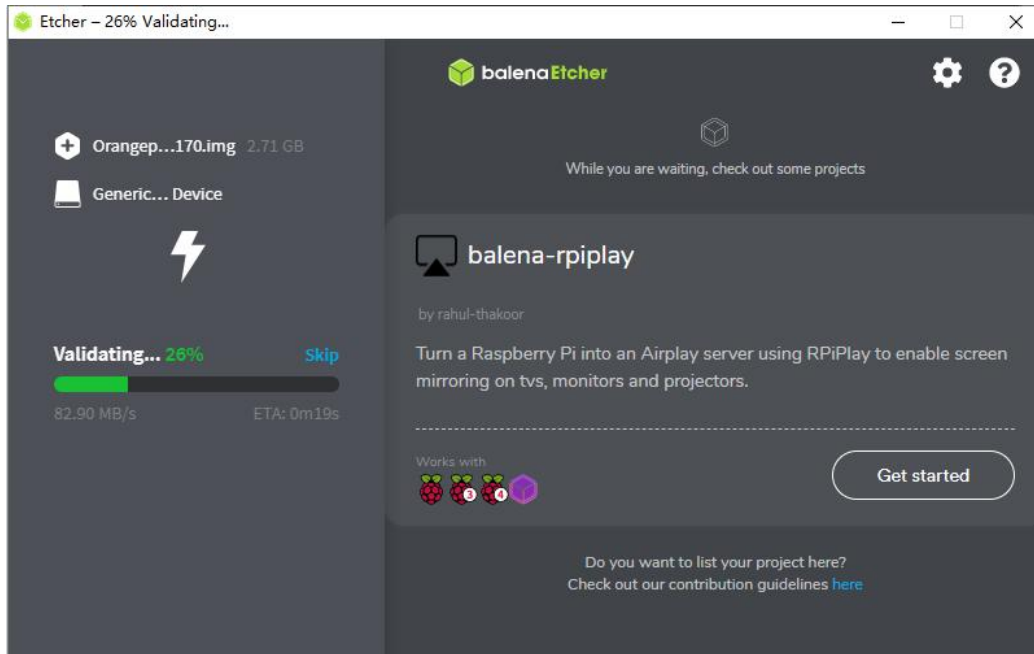


- 8) The interface displayed in the process of balenaEtcher burning the Linux image is shown in the figure below. In addition, the progress bar shows purple to indicate that the Linux image is being burned to the TF card.

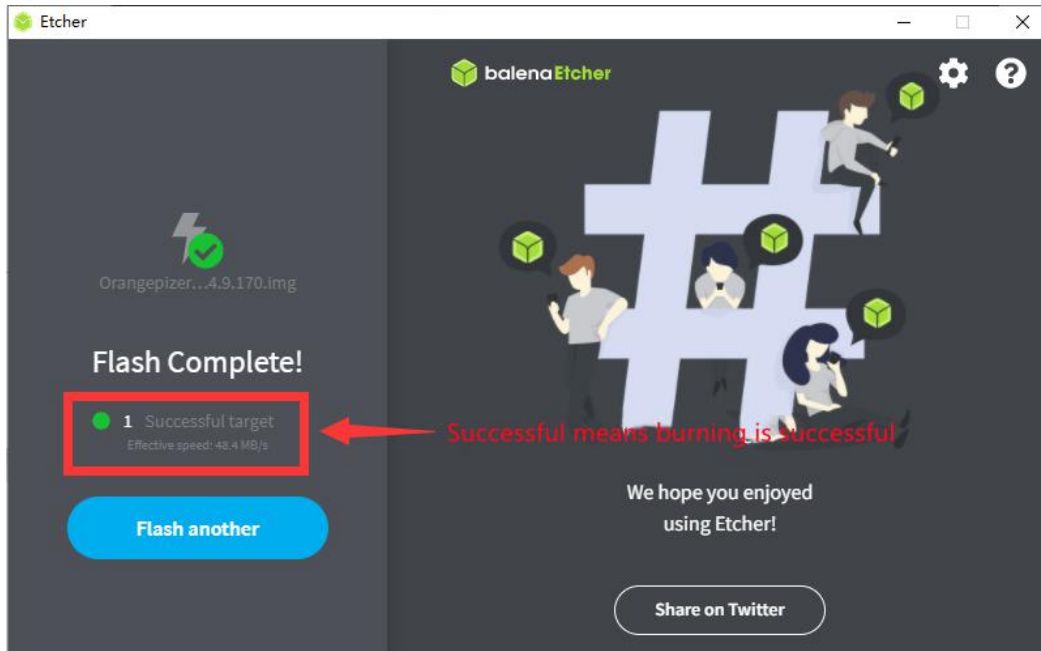




9) After the Linux image is burned, balenaEtcher will also verify the image burned to the TF card by default to ensure that there is no problem in the burning process. As shown in the figure below, a green progress bar indicates that the image has been burned, and balenaEtcher is verifying the burned image.



10) After the successful burning is completed, the display interface of balenaEtcher is shown in the figure below. If a green indicator icon is displayed, it means that the image burning is successful. At this time, you can exit balenaEtcher, and then pull out the TF card and insert it into the TF card slot of the development board.



2.5. The method of burning Linux image to TF card based on Ubuntu PC

Note that: the Linux image mentioned here specifically refers to a Linux distribution image such as Debian or Ubuntu downloaded from the Orange Pi data download page, and Ubuntu PC refers to a personal computer with Ubuntu installed.

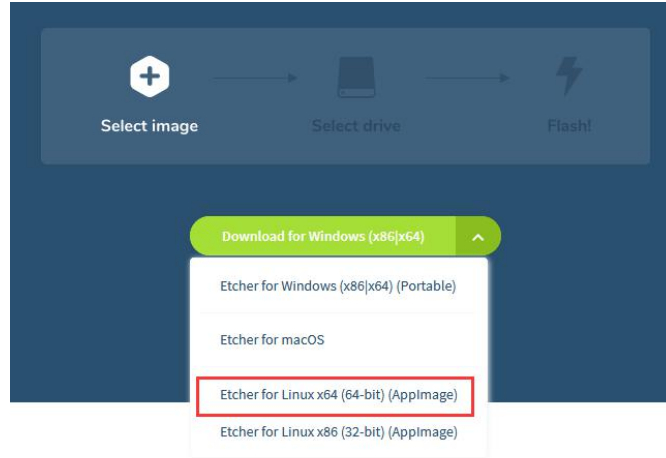
1) First prepare a TF card with a capacity of 8GB or more. The transmission speed of the TF card must be above class 10. It is recommended to use a TF card from a brand such as SanDisk

2) Then use the card reader to insert the TF card into the computer

3) Download balenaEtcher software, the download address 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, please use the **unzip** command to decompress the downloaded compressed package. The decompressed **balenaEtcher-1.5.109-x64.AppImage** is the software needed to burn the 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 compressed package of the Linux operating system image file you want to burn from the data download page of Orange Pi, and then use the decompression software to decompress it. In the decompressed file, the file ending with ".img" is the image file of the operating system. The size is generally more than 1GB

The decompression command for the compressed package ending in 7z is as follows

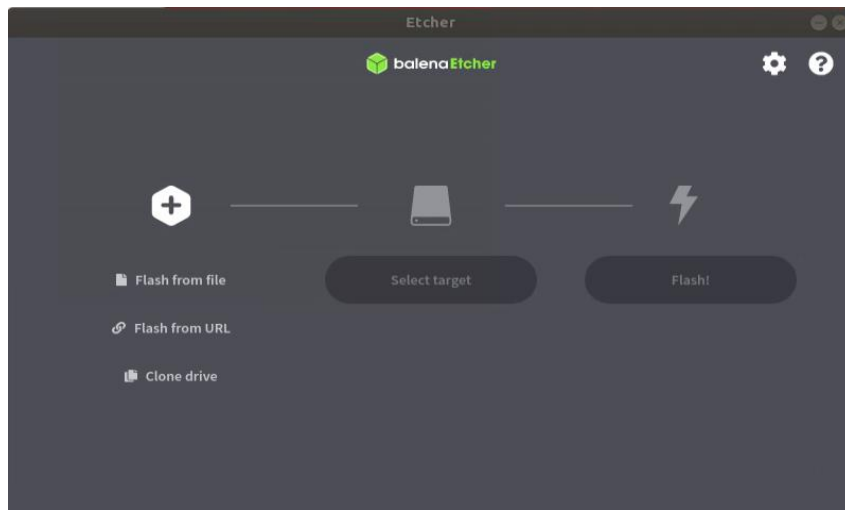
```
test@test:~$ 7z x Orangepi4-lts_2.2.0_debian_bullseye_linux5.10.43_xfce_desktop.7z
test@test:~$ ls Orangepi4-lts_2.2.0_debian_bullseye_linux5.10.43_xfce_desktop.*
Orangepi4-lts_2.2.0_debian_bullseye_linux5.10.43_xfce_desktop.7z
Orangepi4-lts_2.2.0_debian_bullseye_linux5.10.43_xfce_desktop.sha  #checksum file
Orangepi4-lts_2.2.0_debian_bullseye_linux5.10.43_xfce_desktop.img  #image file
```

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

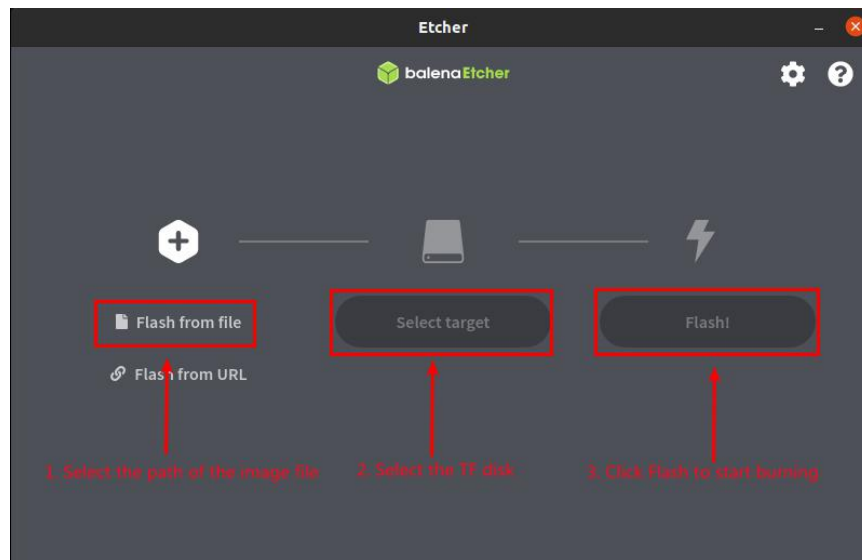


```
test@test:~$ sha256sum -c *.sha  
Orangepi4-lts_2.2.0_debian_bullseye_linux5.10.43_xfce_desktop.img: success
```

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



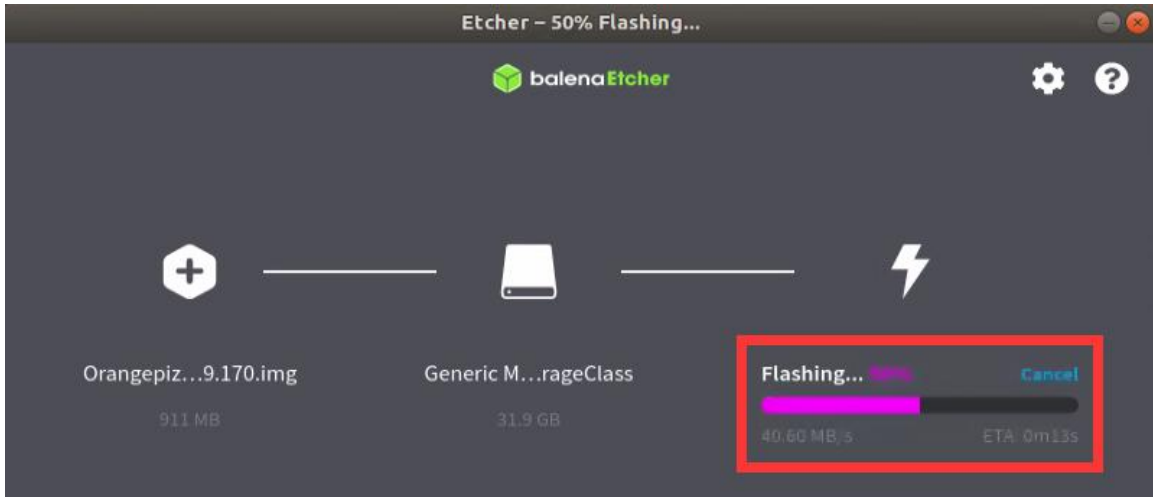
- 9) The specific steps to use balenaEtcher to burn a Linux image are as follows
- a. First select the path of the Linux image file to be burned
 - b. Then select the drive letter of the TF card
 - c. Finally, click Flash to start burning the Linux image to the TF card



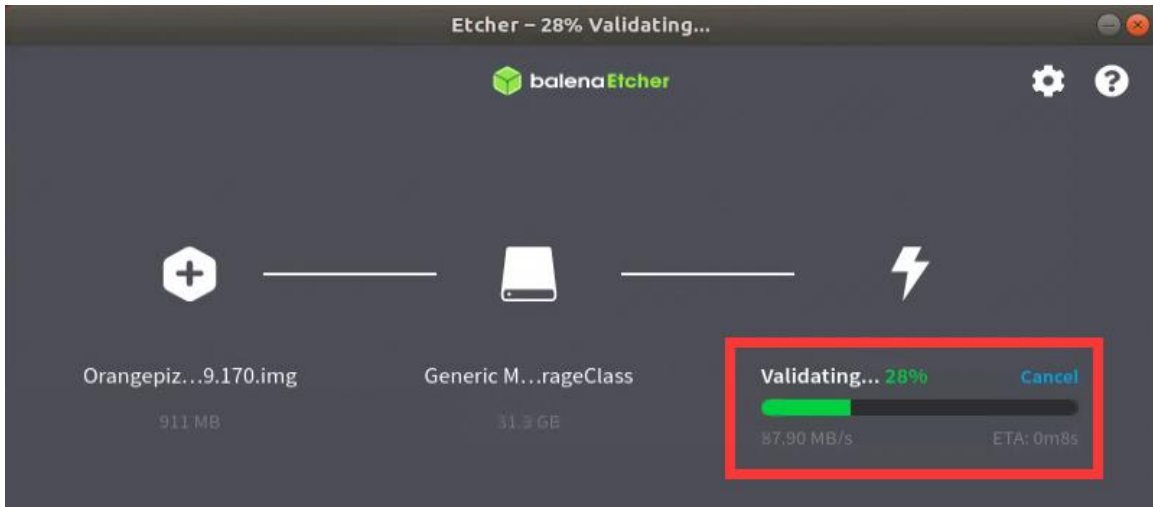
10) The interface displayed in the process of balenaEtcher burning the Linux image is



shown in the figure below. In addition, the progress bar shows purple to indicate that the Linux image is being burned to the TF card.



11) After the Linux image is burned, balenaEtcher will also verify the image burned to the TF card by default to ensure that there is no problem in the burning process. As shown in the figure below, a green progress bar indicates that the image has been burned, and balenaEtcher is verifying the burned image.



12) After the successful burning, the display interface of balenaEtcher is shown in the figure below. If the green indicator icon is displayed, it means that the image burning is successful. At this time, you can exit balenaEtcher, and then pull out the TF card and insert it into the TF card slot of the development board.



2. 6. Method of programming Linux image to eMMC

[See the method of burning linux image to EMMC](#)

2. 7. How to burn Android firmware to TF card

The Android firmware of the development board can only be burned to the TF card using SDDiskTool software under the Windows platform. In addition, the SDDiskTool software does not have a Linux platform version, so it is impossible to burn the Android system to the TF card under the Linux platform.

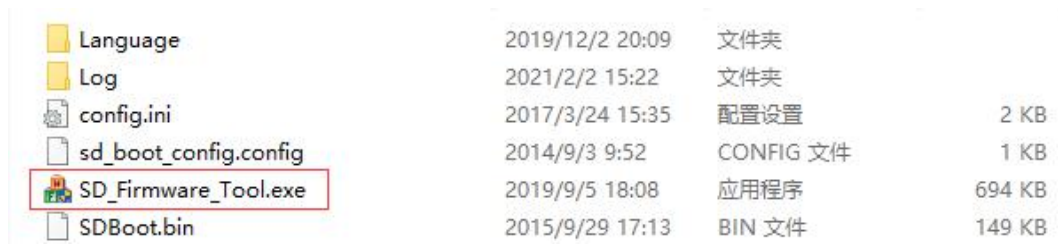
- 1) First prepare a TF card with a capacity of 8GB or more. The transmission speed of the TF card must be above class 10. It is recommended to use a TF card from a brand such as SanDisk
- 2) Then use the card reader to insert the TF card into the computer
- 3) Then download the SDDiskTool programming tool and Android firmware from the



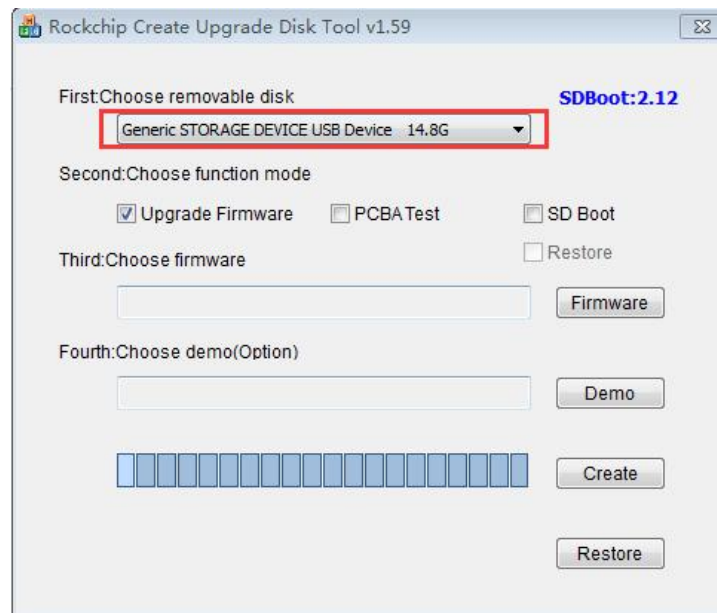
data download page of Orange Pi. The Android firmware on the official website has two versions: eMMC boot and TF card boot. Here you need to select the TF card boot version, and pay attention to the TF card boot version. The name of the Android firmware contains the word "SD"

4) Then use the decompression software to decompress the downloaded Android firmware compressed package. In the decompressed file, the file ending with ".img" is the Android firmware

5) Then use the decompression software to decompress **SDDiskTool_v1.59.zip**, this software does not need to be installed, find **SD_Firmware_Tool.exe** in the decompressed folder and open it

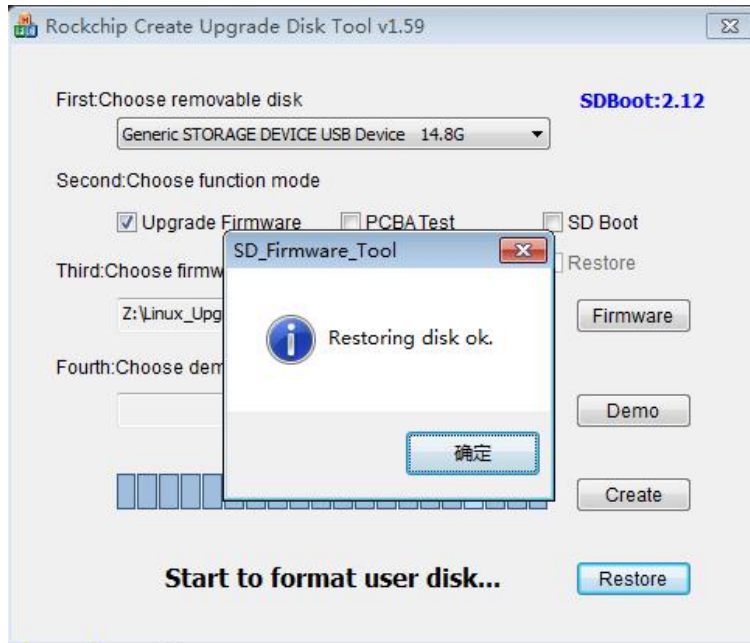


6) After opening **SDDiskTool**, if the TF card is recognized normally, the inserted disk device will be displayed in the "Select Removable Disk Device" column. **Please make sure that the displayed disk device is the same as the drive letter of the TF card you want to burn.** Yes, if there is no display, you can try to unplug the TF card

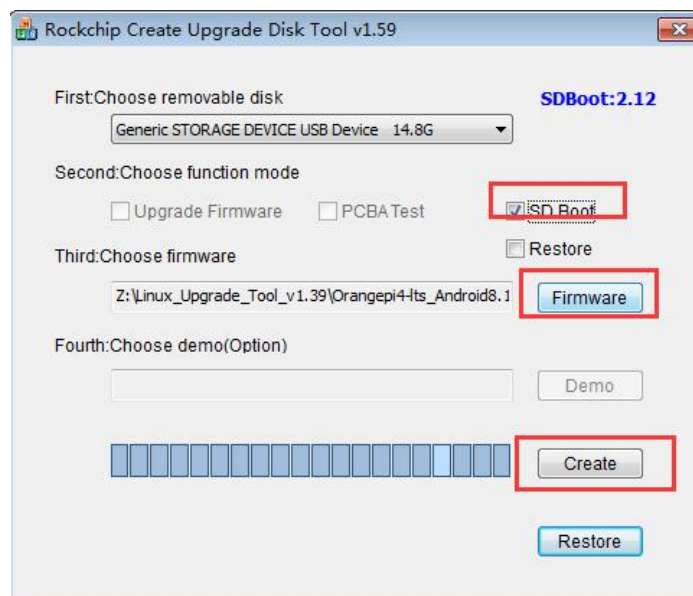




7) After confirming the drive letter, format the TF card first, click the **restore disk** button in SDDiskTool, or use the SD Card Formatter mentioned above to format the TF card



- 8) Then start writing Android image to TF card
 - a. First check "SD Boot" in "Select Function Mode"
 - b. Then select the path of the Android image in the "Select firmware upgrade" column
 - c. Finally, click the "Start Creation" button to start burning the Android image to the TF card





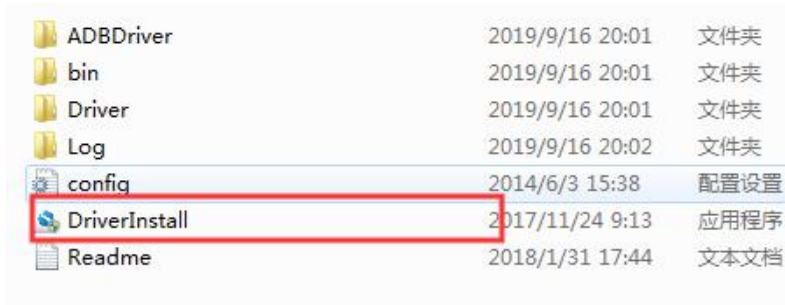
9) After burning, you can exit SDDiskTool, and then you can pull out the TF card from the computer and insert it into the development board to start

2.8. The method of burning Android firmware to eMMC based on Windows PC

Orange Pi 4 LTS has three upgrade modes, namely MaskRom mode, Loader mode and SD upgrade mode. The first two modes need to be burned through the Type C cable, and the latter mode is burned through the TF card. For the way of programming through Type C cable, if there is no programming system in eMMC, it will enter MaskRom mode by default. If a bootable system has been programmed in eMMC, you can enter Loader mode for programming. It should be noted that if the system in eMMC is damaged due to an accident in programming or other reasons, it will not be able to enter the Loader mode for programming. You need to enter the MaskRom mode according to the method of entering the MaskRom mode, and then use the TypeC cable to connect. Computer and development board for programming

2.8.1. Directly burn Android firmware to eMMC through Type C interface

- 1) First prepare a good quality Type C data cable
- 2) Then download the Rockchip driver DriverAssitant_v4.6, the burning tool AndroidTool and the firmware of Android8.1 from the data download page of Orange Pi. The Android firmware on the official website has two versions: eMMC boot and TF card boot, here you need to choose eMMC The boot version, note that the name of the Android firmware that supports eMMC boot does not contain the word "SD", and please ensure that the version of the AndroidTool tool is v2.58, please do not use AndroidTool software lower than v2.58 to burn Android 8.1 firmware of Orange Pi 4 LTS, AndroidTool tools lower than this version may have problems to program Android 8.1 system
- 3) Decompress `DriverAssitant_v4.6.zip` with decompression software, find the `DriverInstall.exe` executable file in the decompressed folder and open it



4) The steps to install Rockchip micro driver are as follows

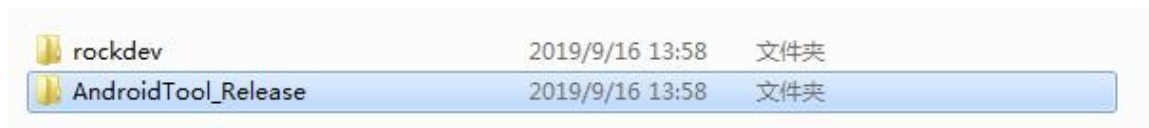
a. Click the "Driver Installation" button

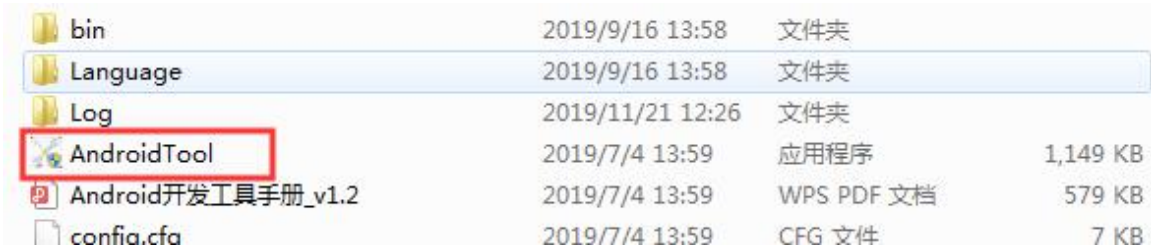


b. After waiting for a while, a pop-up window will prompt "Driver installed successfully"

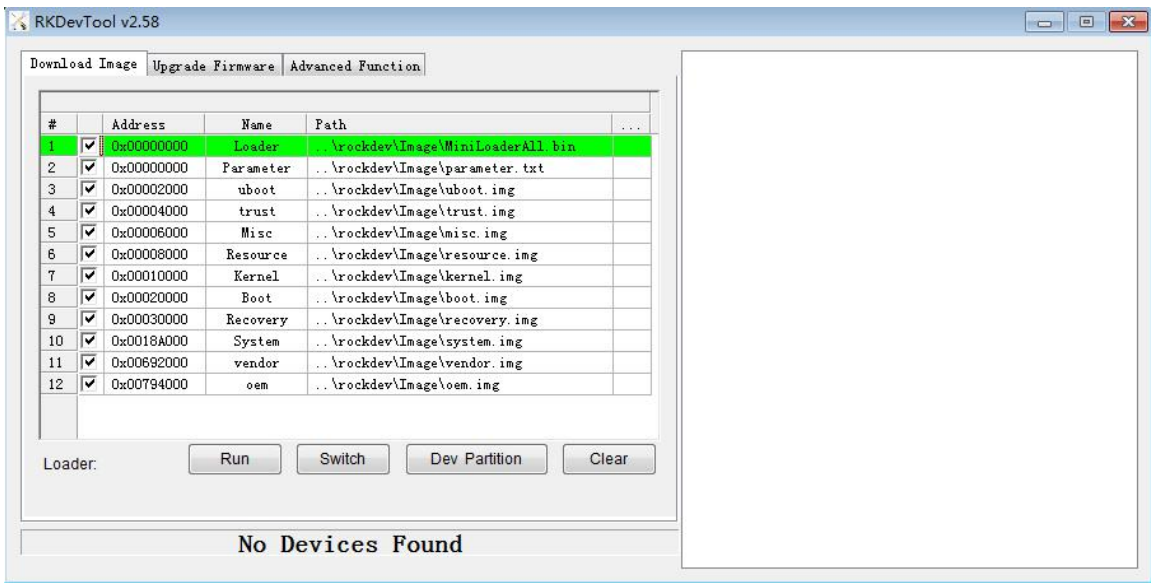


5) Unzip AndroidTool_v2.58.zip, this software does not need to be installed, just find AndroidTool in the unzipped folder and open it

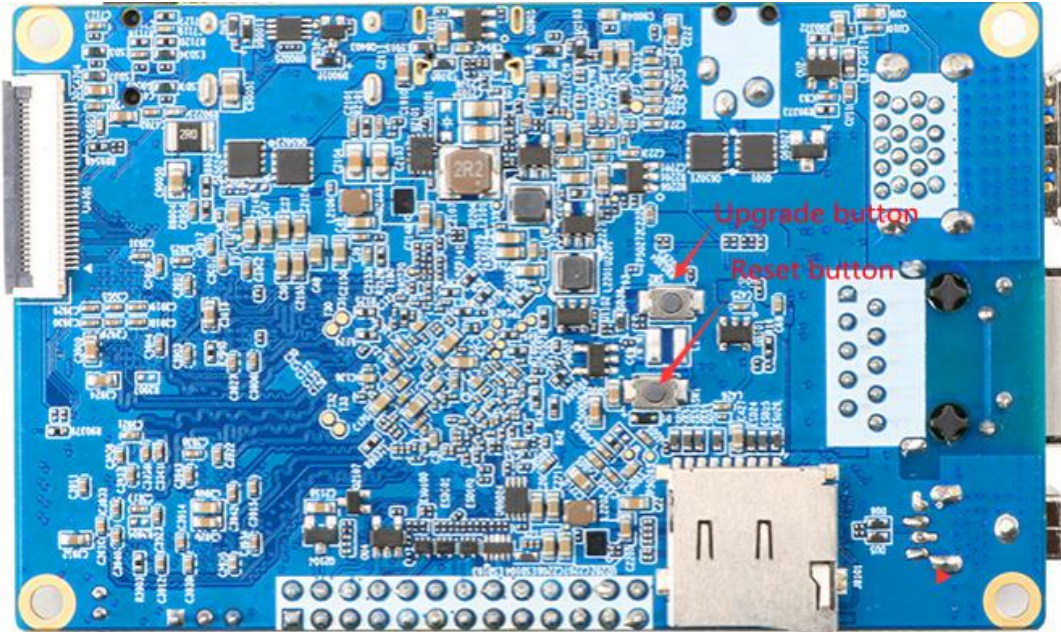




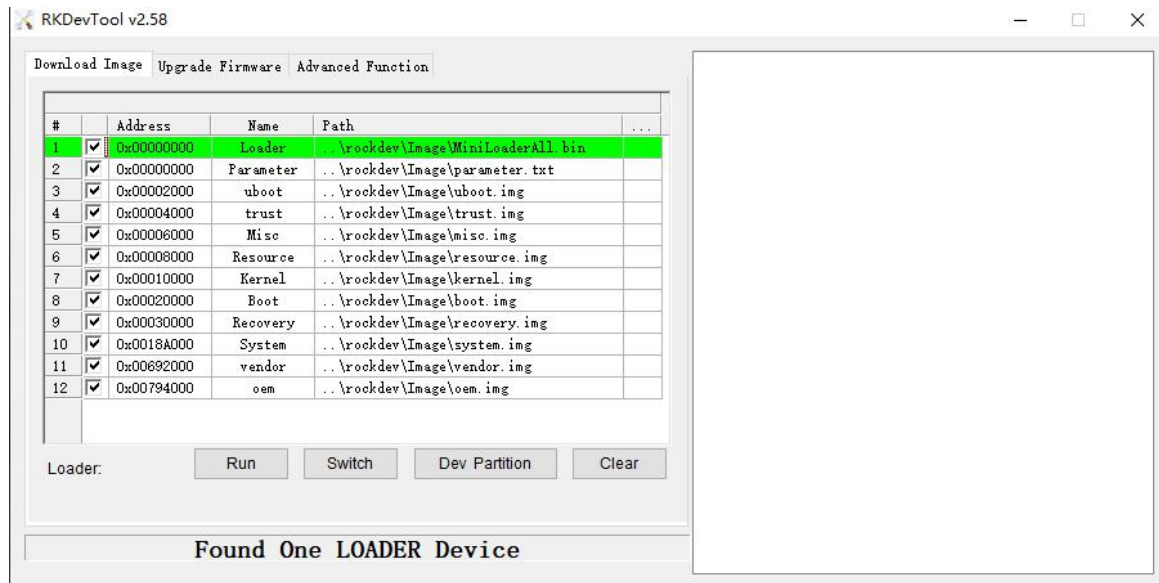
6) After opening the AndroidTool tool, because the computer has not been connected to the Orange Pi 4 LTS development board through the Type-C cable at this time, the lower left corner will prompt "No Devices Found"



- 7) Then start the burning of Android firmware
 - a. First connect the DC power adapter to the OrangePi 4 LTS development board, and make sure to unplug the TF card
 - b. Then connect OrangePi 4 LTS to Windows PC via Type-C cable
 - c. First press and hold the upgrade button of Orange Pi 4 LTS, then lightly press the reset button and release it immediately, wait for 3~5 seconds and then release the upgrade button. The position of the button on the development board is shown in the figure below.

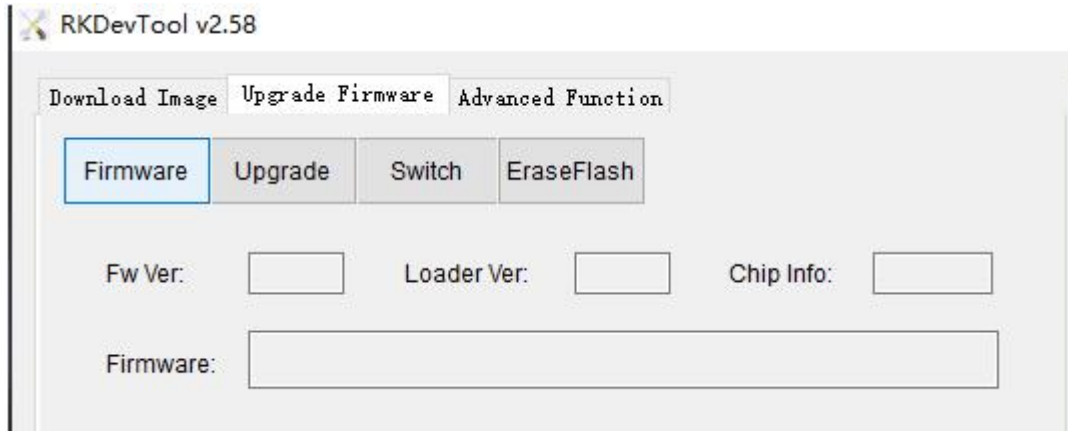


- a. If the previous steps are successful, the development board has entered the Loader mode, and "Found One LOADER Device" will be prompted on the interface of the AndroidTool tool

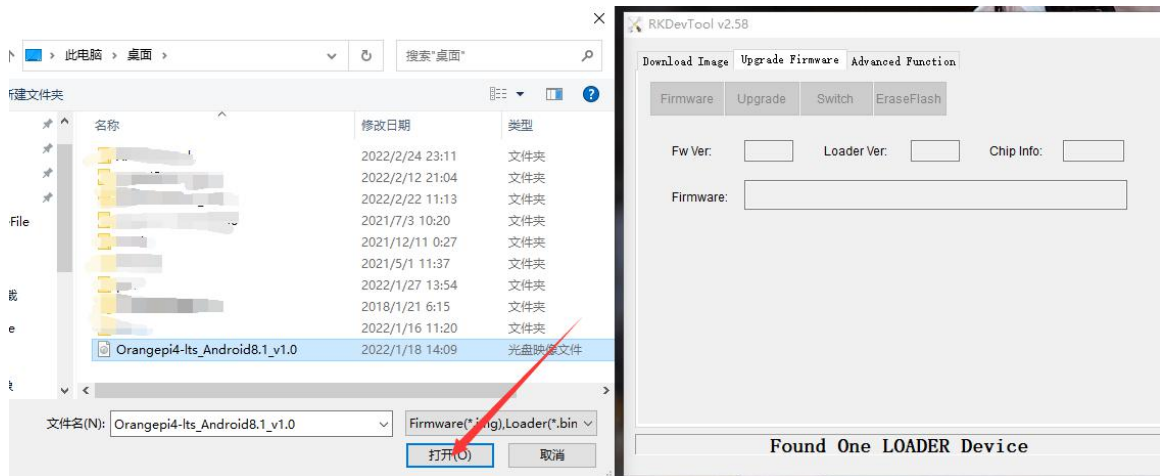


If no system is burned in eMMC, then AndroidTool will prompt "Found a MaskROM device"

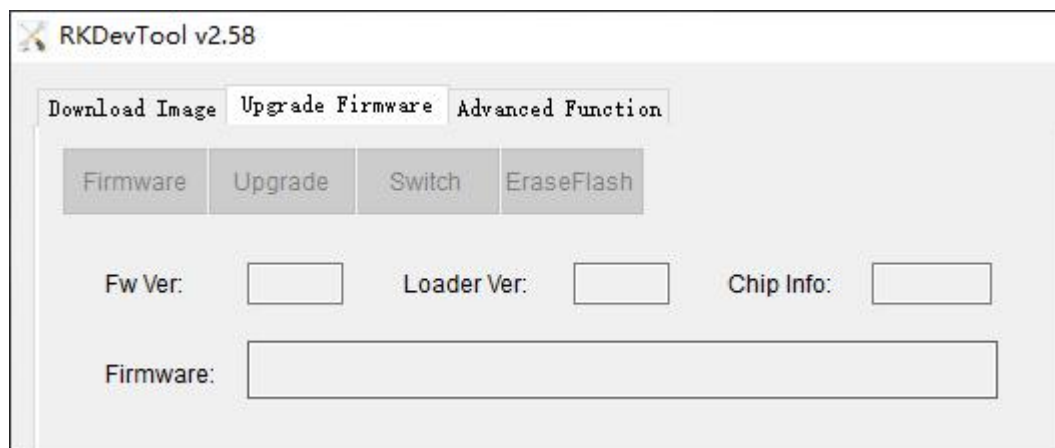
- b. Then click the "Upgrade Firmware" column of AndroidTool



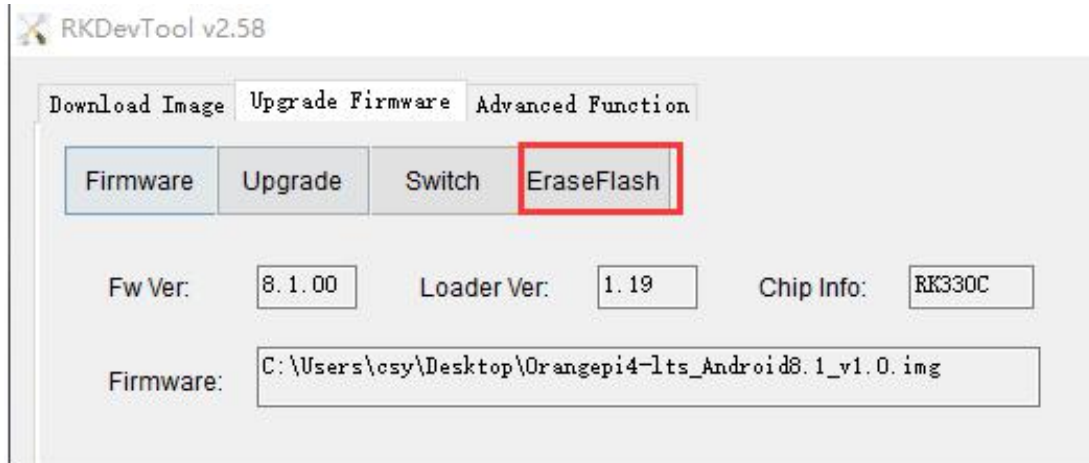
- c. Then click the "Firmware" button to select the path of the Android firmware, and then click "Open", as shown in the following figure



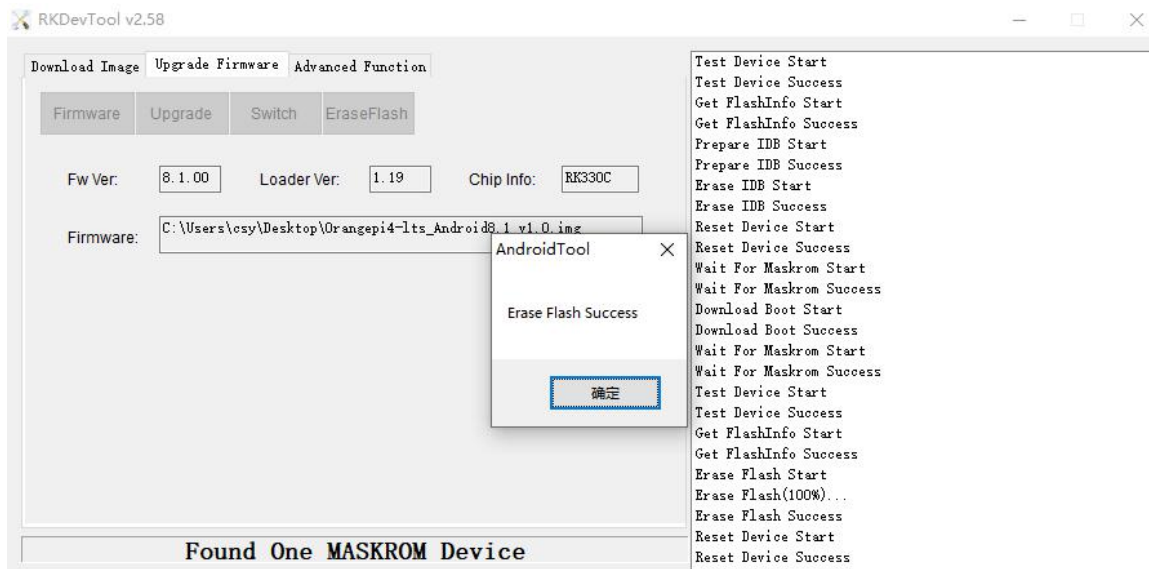
- d. After the Android firmware path selection is completed, the firmware will start to be loaded, and the button will turn into a gray unselectable state.



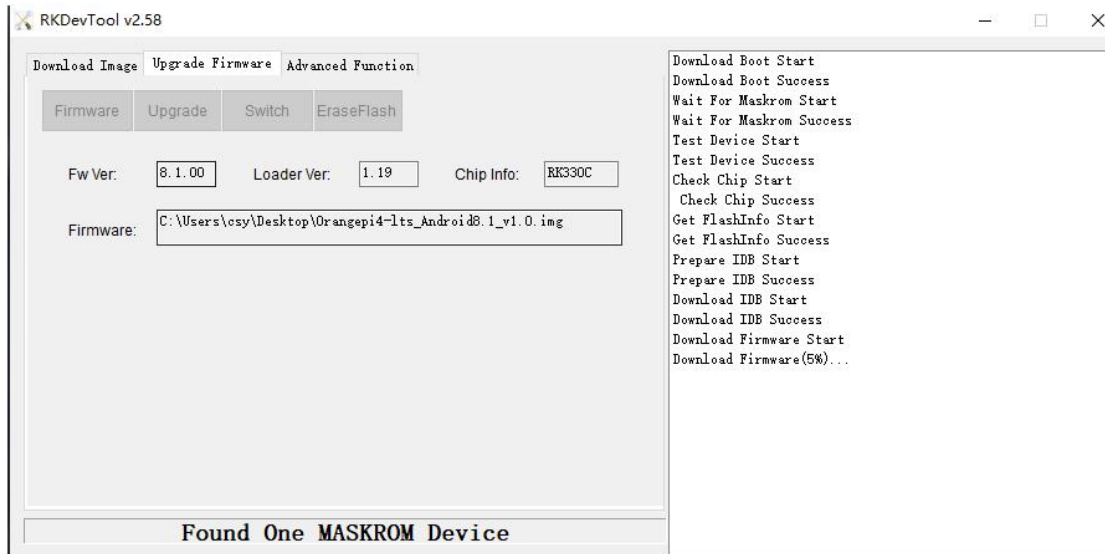
- e. After the firmware is loaded, the button becomes selectable, and then click "Erase Flash" to start erasing eMMC



- f. The interface of the successful erasing is shown in the figure below, because the system in eMMC has been erased, so you can see the prompt "found a MASKROM device"



- g. Finally, click the "Upgrade" button to burn. During the burning process, the AndroidTool is displayed as shown in the figure below, and the Android system will automatically start after the burning is completed.



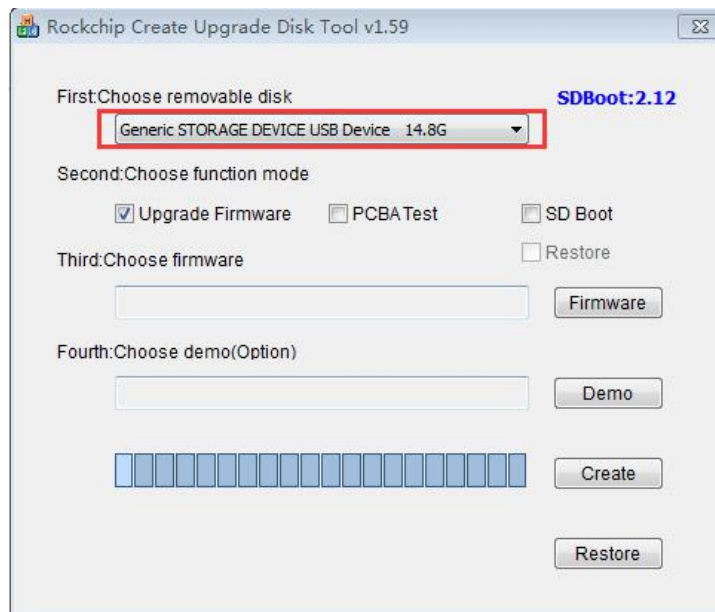
2. 8. 2. Burn Android image to eMMC via TF card

- 1) First prepare a TF card with a capacity of 8GB or more. The transmission speed of the TF card must be above class 10. It is recommended to use a TF card from a brand such as SanDisk
- 2) Then use the card reader to insert the TF card into the computer
- 3) Download the Android firmware and SDDiskTool programming tool from the data download page of Orange Pi. It should be noted that the firmware on the official website has two versions: eMMC boot and TF card boot. Here you need to select the eMMC boot version, and pay attention to support eMMC boot The firmware does not contain the word "SD", and please ensure that the version of SDDiskTool is v1.59
- 4) Then use the decompression software to decompress the downloaded Android firmware compressed package. In the decompressed file, the file ending with ".img" is the Android firmware
- 5) Use the decompression software to decompress **SDDiskTool_v1.59.zip**, this software does not need to be installed, just find **SD_Firmware_Tool.exe** in the decompressed folder and open it

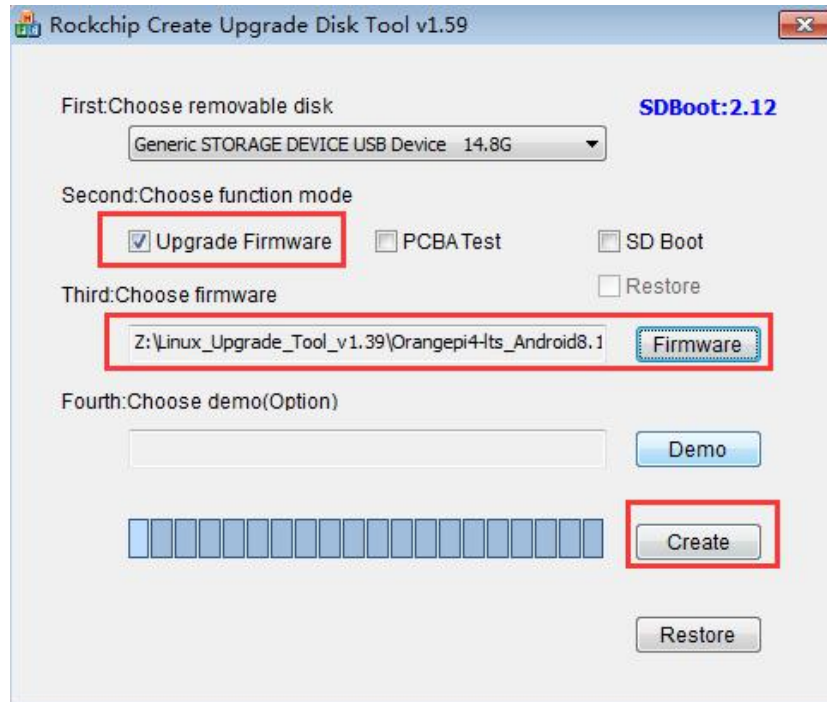


Language	2019/12/2 20:09	文件夹	
Log	2021/2/2 15:22	文件夹	
config.ini	2017/3/24 15:35	配置设置	2 KB
sd_boot_config.config	2014/9/3 9:52	CONFIG 文件	1 KB
SD_Firmware_Tool.exe	2019/9/5 18:08	应用程序	694 KB
SDBoot.bin	2015/9/29 17:13	BIN 文件	149 KB

6) After opening SDDiskTool, if the TF card is recognized normally, the inserted disk device will be displayed in "Select Removable Disk Device". Please make sure that the displayed disk device is consistent with the drive letter of the TF card you want to burn. If there is no display, you can try to unplug the TF card

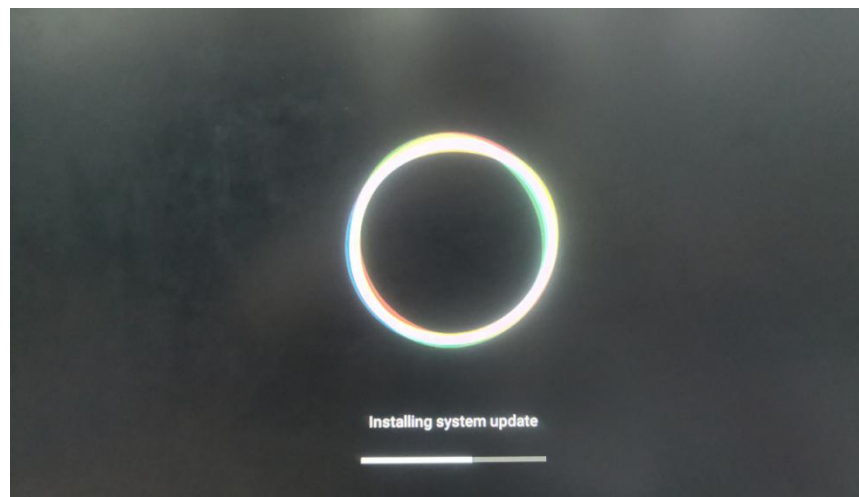


- 7) Then start writing Android firmware to TF card
 - a. First select the path of the Android firmware in the "Choose firmware" column
 - b. Then select "Firmware Upgrade" in "Choose function mode"
 - c. Finally click the "Create" button to start burning



8) After burning, you can exit SDDiskTool, and then you can unplug the TF card from the computer and insert it into the development board. After the development board is powered on, it will start to burn the Android firmware in the TF to the eMMC.

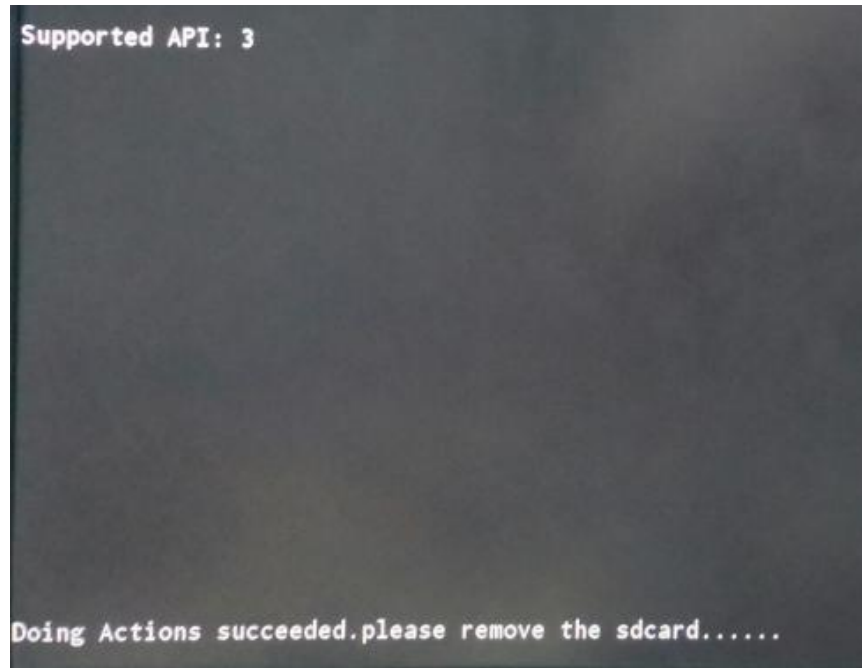
9) If the development board is connected to an HDMI display, you can also see the progress bar of burning Android firmware to eMMC from the HDMI display



10) When the following information is displayed, it means that the Android firmware has been burned into the eMMC. At this time, the TF card can be pulled out, and then the



Android system in the eMMC will start to start.



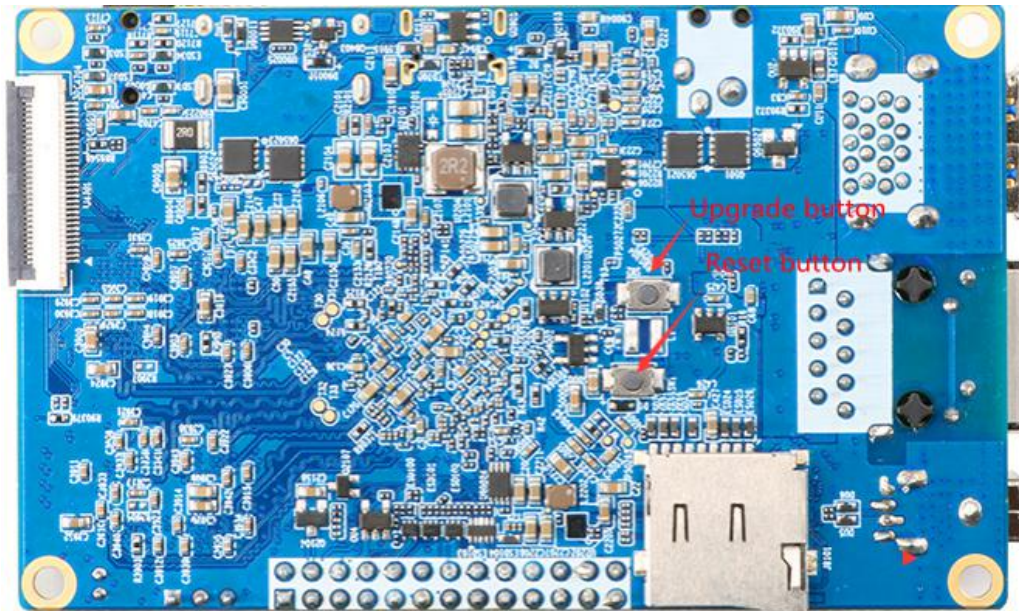
2.9. Method of Burning Android Image to eMMC Based on Ubuntu PC

- 1) First prepare a good quality Type C data cable
- 2) Then download the upgrade_tool tool and Android 8.1 firmware from the data download page of Orange Pi. The firmware on the official website has two versions: eMMC boot and TF card boot. Here you need to select the eMMC boot version. Note that the firmware that supports eMMC boot does not contain "SD", please make sure the upgrade_tool version is v1.39, please do not use the upgrade_tool software lower than v1.39 to burn the Android 8.1 firmware of Orange Pi 4 LTS, the upgrade_tool tool lower than this version Burning Android 8.1 may be problematic
- 3) Then execute the command in the terminal to decompress upgrade_tool and add executable permissions

```
test@test:~$ unzip Linux_Upgrade_Tool_v1.39.zip  
test@test:~$ cd Linux_Upgrade_Tool_v1.39  
test@test:~/Linux_Upgrade_Tool_v1.39$ sudo chmod +x ./upgrade_tool
```



- 4) Then start the burning of Android firmware
 - a. First connect the DC power adapter to the OrangePi 4 LTS development board, and make sure to unplug the TF card
 - b. Then connect OrangePi 4 LTS with Ubuntu PC via Type-C data cable
 - c. First press and hold the upgrade button of Orange Pi 4 LTS, then lightly press the reset button and release it immediately, wait for 3~5 seconds and then release the upgrade button. The position of the button on the development board is shown in the figure below.



- a. If the previous steps are successful, the development board has entered the Loader mode at this time, execute the following command and you will see Mode=Loader, indicating that the Loader device has been recognized

```
test@test:~/Linux_Upgrade_Tool_v1.39$ ./upgrade_tool LD  
Program Data in /home/csy/.config/upgrade_tool  
List of rockusb connected(1)  
DevNo=1 Vid=0x2207,Pid=0x330c,LocationID=2010201 Mode=Loader
```

If no system is burned in eMMC, then the Maskrom device will be recognized, and the value of Mode will be Maskrom

- b. Then copy the downloaded Android image to the Linux_Upgrade_Tool_v1.39 directory



- c. Then enter the following command in the terminal of the Ubuntu PC to erase the eMMC

```
test@test:~$ sudo ./upgrade_tool ef OrangePi4-lts_Android8.1_v1.0.img
```

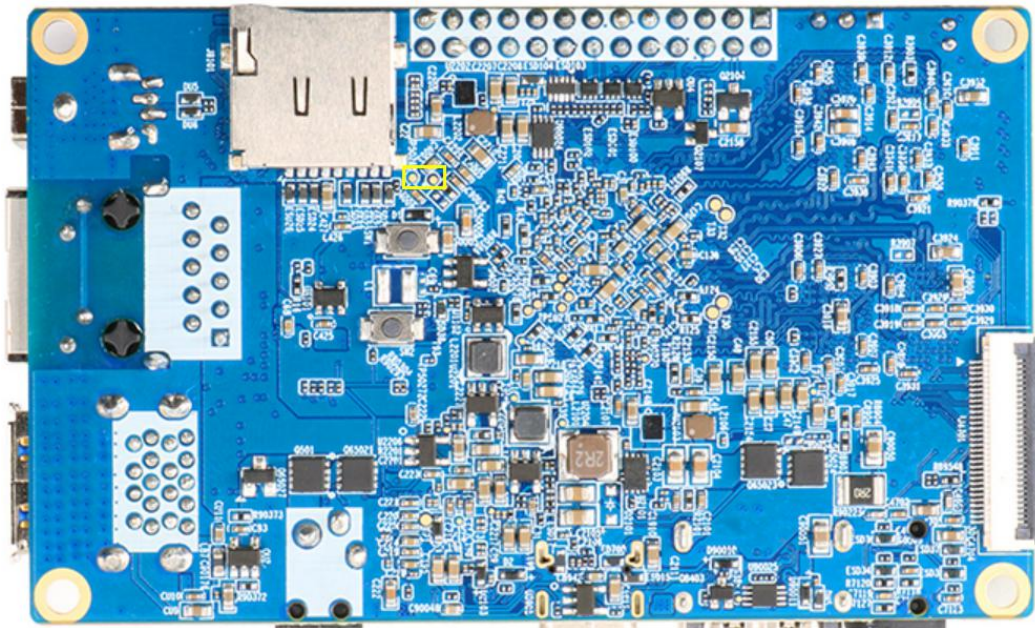
- d. Finally execute the following command to start burning Android firmware to eMMC

```
test@test:~$ sudo ./upgrade_tool uf OrangePi4-lts_Android8.1_v1.0.img
```

2. 10. How to enter MaskRom mode

Under normal circumstances, it is not necessary to enter MaskRom mode. Only when the bootloader is damaged and the system cannot be started, it is necessary to enter Maskrom mode for burning.

- 1) First make sure that the OrangePi 4 LTS development board is disconnected from all power sources, and the SD card is unplugged
- 2) Then use metal tweezers to connect the two test points in the yellow box in the picture below reserved by the OrangePi 4 LTS development board, and keep it still (**make sure that the two test points are short-circuited**)

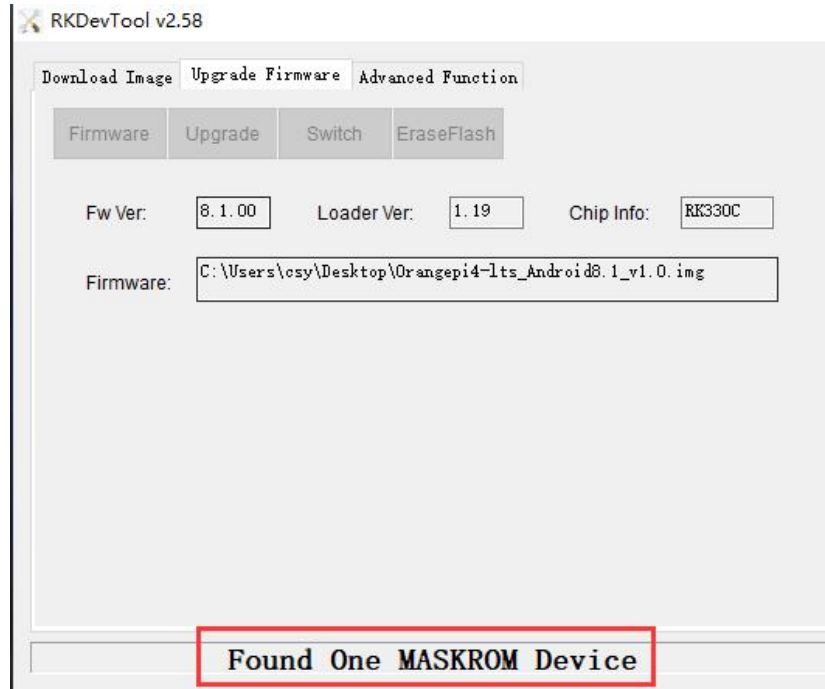


- 3) Then plug in the DC power supply to the Orange Pi 4 LTS development board, wait



for 2~3 seconds and then release the metal tweezers. At this point the OrangePi 4 LTS development board will enter maskrom mode

4) Then use the Type C cable to connect the OrangePi 4 LTS development board to the Windows PC, and then open the **AndroidTool** tool, if all goes well, you can see the AndroidTool interface prompts "**Found One MASKROM Device**"



5) At this point, you can burn the Android firmware through the AndroidTool tool under Windows

2. 11. Start the orange pi development board

- 1) The development board has an onboard eMMC, and the Android 8.1 image is burned by default in it. When you get the development board, you can directly use the image in the eMMC for startup and full-function testing.
- 2) If you need to use the linux image, you can insert the TF card with the linux image burned into the TF card slot of the Orange Pi development board
- 3) The development board has an HDMI interface, and the development board can be connected to a TV or HDMI display through an HDMI to HDMI cable



- 4) Connect the USB mouse and keyboard to control the orange pi development board
- 5) The development board has an Ethernet port, which can be plugged into a network cable for Internet access
- 6) Connect a 5V/3A (5V/4A can also) high-quality power adapter

a. Orange Pi 4 LTS cannot be powered through the 5v pin on the 26pin interface

b. Remember not to plug in the 12V power adapter, if the 12V power adapter is plugged in, it will burn out the development board

c. Many unstable phenomena during system startup are basically caused by power supply problems, so a reliable power adapter is very important

- 1) Then turn on the switch of the power adapter. If everything is normal, the HDMI display can see the startup screen of the system.
- 2) If you want to view the output information of the system through the debugging serial port, please use the USB to TTL module and DuPont cable to connect the development board to the computer. For the connection method of the serial port, please refer to the section on how to use the debugging serial port.

How to judge whether the system has started normally:

- 1) **If the HDMI display is connected, the judgment method is very simple. As long as the HDMI display normally displays the interface of the system, it means that the system has been started normally;**
- 2) **If there is no HDMI display, you can connect the development board to the computer through the serial cable, and check the startup status of the system by debugging the log information output by the serial port. If the serial port output stops at the login interface of the terminal, it means that the system has been started normally;**
- 3) **If there is no HDMI display and serial port cable, you can judge the startup status of the system through the two LED lights on the development board. If the red LED light is on and the green LED light is flashing, the system has generally started normally. , If after power-on and waiting for a period of time, only the red LED light is on, or the red and green LED lights are not on, it means that the system has not started normally.**

If the system fails to start or cannot enter the login interface normally, please check the following first:



- 1) Check whether the downloaded image is damaged, which can be judged by calculating the checksum attached to the image;
- 2) If there is any problem in the process of burning the image to the TF card, you can re-burn the image and test it again;
- 3) Make sure there is no problem with the power adapter, you can try another one;
- 4) Make sure that the TF card meets the requirements of the Orange Pi development board. If there is an extra TF card, you can try to change the TF card and then burn the image and test it again;
- 5) If all of the above are OK, please save the output log of the debugging serial port during the system startup process (preferably in the form of txt text instead of taking pictures), and then report the problem to the customer service.

2. 12. How to debug the serial port

2. 12. 1. Connection instructions for debugging serial port

1) First, you need to prepare a 3.3V USB to TTL module. For better platform compatibility, it is recommended to use the CH340 USB to TTL module. Then insert one end of the USB interface of the USB to TTL module into the USB interface of the computer



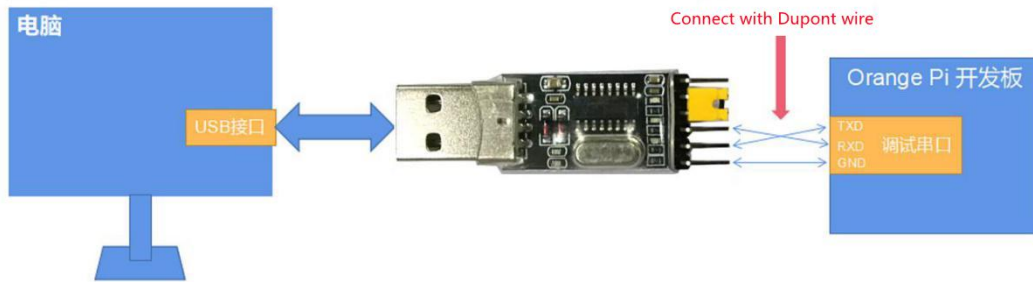
2) The corresponding relationship between the debug serial port GND, RXD and TXD pins of the development board is shown in the figure below



3) The GND, TXD and RXD 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 development board
 - b. The RX of the USB to TTL module is connected to the TX of the development board
 - c. The TX of the USB to TTL module is connected to the RX of the development 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 the USB to TTL module to the computer and the Orange Pi development board

The TX and RX of the serial port need to be cross-connected. If you don't want to carefully distinguish the order of TX and RX, you can connect the TX and RX of the serial port casually. If there is no output in the test, then exchange the order of TX and RX, so there is always a the order is correct

5) If the CP2102 USB to TTL module is used, in the case of a baud rate of 1500000, some systems may encounter garbled or unusable problems. The specific test situation is as follows

USB to TTL module model	Host system	Support situation
CH340	win7	OK
	win10	OK
	ubuntu14.04	OK
	ubuntu18.04	OK
	ubuntu20.04	OK
CP2102	win7	OK
	win10	NO
	ubuntu14.04	OK
	ubuntu18.04	NO



	ubuntu20.04	NO
--	-------------	----

2. 12. 2. How to use the debugging serial port on Ubuntu platform

There are many serial debugging software that can be used under Linux, such as putty, minicom, etc. The following demonstrates how to use putty

1) First, insert the USB to TTL module into the USB interface of the Ubuntu computer. If the connection and recognition of the USB to TTL module is normal, you can see the corresponding device node name under /dev of the Ubuntu PC, remember this node name, and set the serial port later software will be used

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

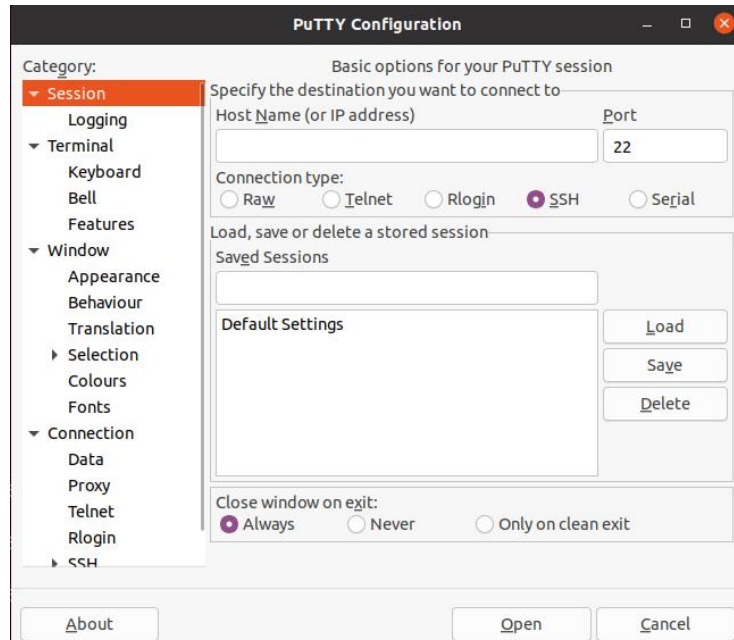
2) Then use the following command to install putty on Ubuntu PC

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

3) Then run putty, remember to add sudo permissions

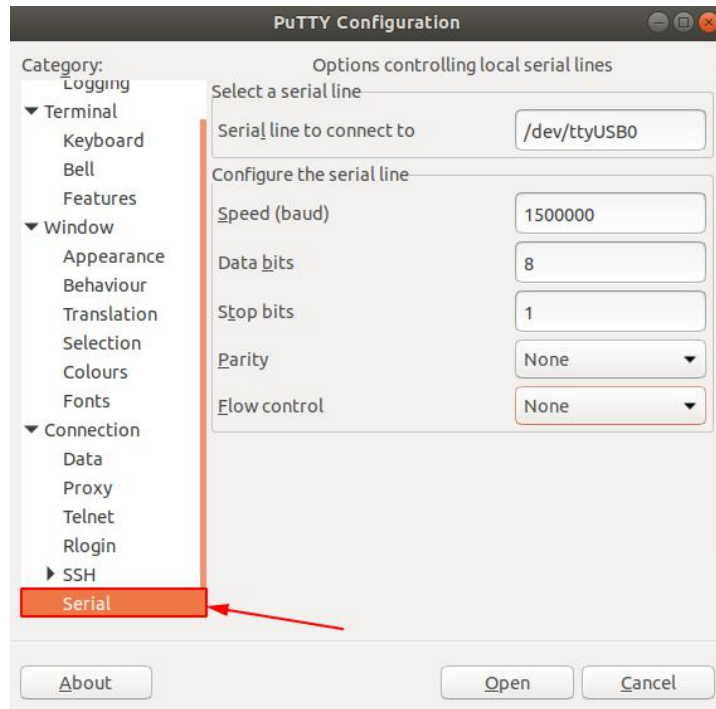
```
test@test:~$ sudo putty
```

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



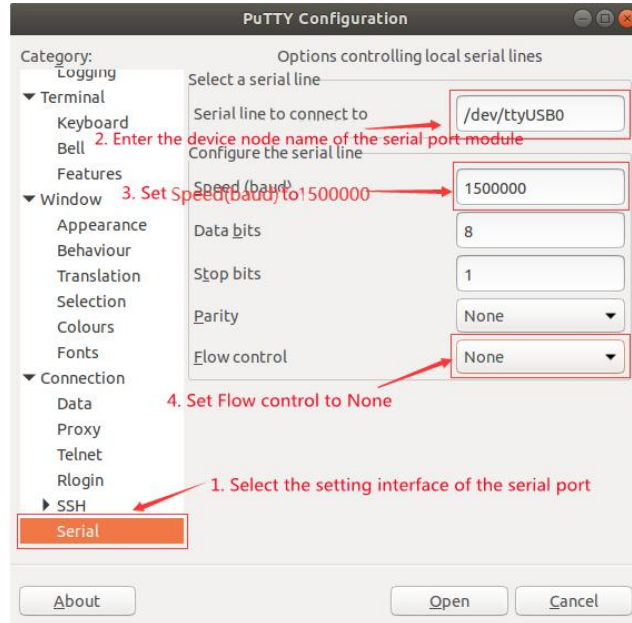


5) First select the setting interface of the serial port



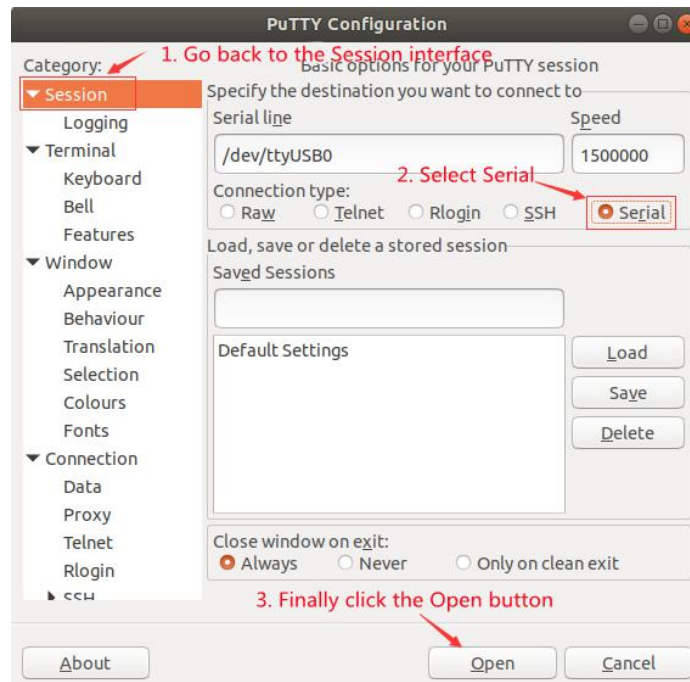
6) Then set the parameters of the serial port

- a. Set Serial line to connect to to /dev/ttyUSB0 (modify to the corresponding node name, usually /dev/ttyUSB0)
- b. Set Speed(baud) to 1500000 (the baud rate of the serial port)
- c. Set Flow control to None



7) After setting the serial port setting interface, go back to the Session interface

- a. First select the Connection type as Serial
- b. Then click the Open button to connect the serial port



8) After starting the development board, you can see the Log information output by the system from the open serial terminal



```

/dev/ttyUSB0 - PuTTY
R0=0x18
MR4=0x1
MR5=0x1
MR8=0x8
MR12=0x72
MR14=0x72
MR18=0x0
MR19=0x0
MR24=0x8
MR25=0x0
R0=0x18
MR4=0x1
MR5=0x1
MR8=0x8
MR12=0x72
MR14=0x72
MR18=0x0
MR19=0x0
MR24=0x8
MR25=0x0
channel 0 training pass!
channel 1 training pass!
change freq to 416MHz 0,1
Channel 0: LPDDR4,416MHz
Bus Width=32 Col=10 Bank=8 Row=15/15 CS=2 Die Bus-Width=16 Size=2048MB
Channel 1: LPDDR4,416MHz
Bus Width=32 Col=10 Bank=8 Row=15/15 CS=2 Die Bus-Width=16 Size=2048MB
256B stride
R0=0x18

```

2. 12. 3. How to use the debugging serial port on Windows platform

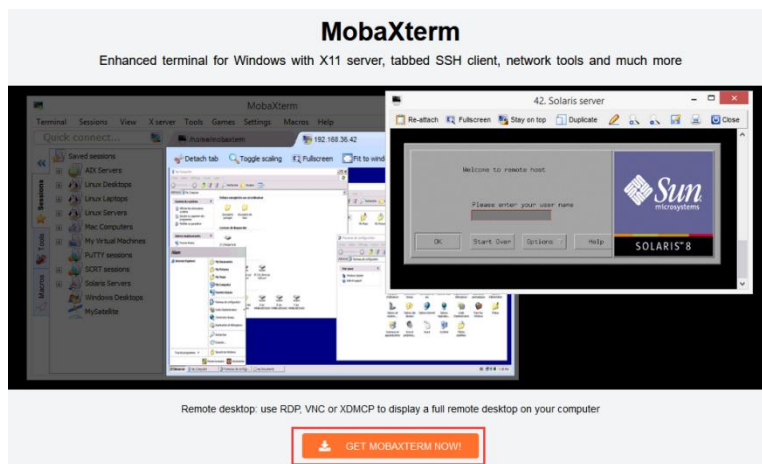
There are many serial port debugging software that can be used under Windows, such as SecureCRT, MobaXterm, etc. The following shows how to use MobaXterm. This software has a free version and can be used without purchasing a serial number.

1) Download MobaXterm

- a. Download MobaXterm URL as follows

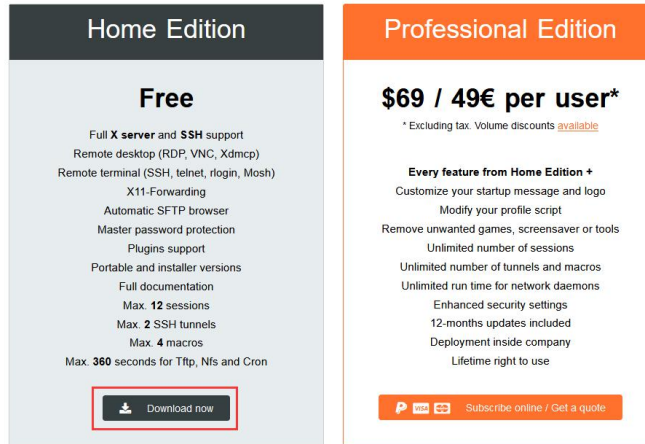
<https://mobaxterm.mobatek.net/>

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

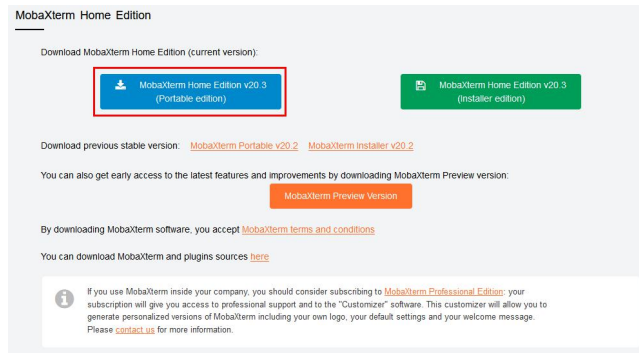




c. Then choose to download the Home version



d. Then select the Portable portable version. After downloading, there is no need to install it, just open it and use it



2) 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

名称	修改日期	类型	大小
CygUtils.plugin	2020/5/21 4:06	PLUGIN 文件	15,570 KB
MobaXterm_Personal_20.3	2020/6/5 4:30	应用程序	14,104 KB

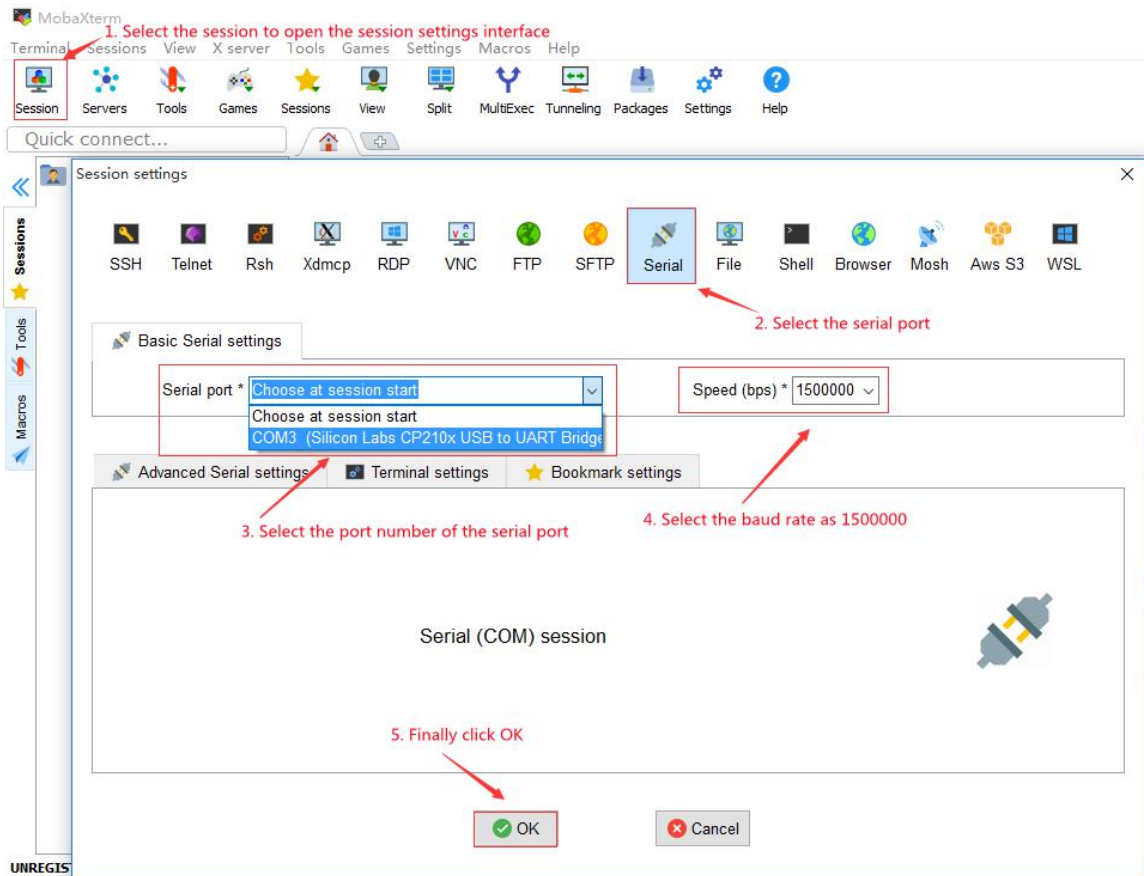
3) After opening the software, the steps to set the serial port connection are as follows

- Open the session settings interface
- Select serial port type
- Select the port number of the serial port (select 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 port



chip.

- d. Select the baud rate of the serial port to be 1500000
- e. Finally, click the "OK" button to complete the setting



4) After clicking the "OK" button, it will enter the following interface. At this time, you can see the output information of the serial port when you start the development board.



COM3 (Silicon Labs CP210x USB to UART Bridge (COM3))

Terminal Sessions View X server Tools Games Settings Macros Help

Session Servers Tools Games Sessions View Solt MUXExec Turning Packages Settings Help

Quick connect...

User sessions

COM3 (Silicon Labs CP210x USB)

A session record will be generated here, which can be opened by clicking directly next time

Serial port information output interface

```

158 HELLO! BOOT0 is starting May 13 2020 14:10:04!
163 BOOT0 commit : 993cf83
166 set_pll_start
168 peripio has been enabled
172 set_pll_end
173 unknow PMU
175 DMAC: ADDR06
182 vvalid para:1 select dram para0
185 board init ok
188 DRAM BOOT DRIVE INFO: V0.52
191 the chip id is 0x5000
194 Chip_id check OK
196 DRAM VCC set to 1500 mv
200 read_calibration error
204 read_calibration error
208 read_calibration error
212 read_calibration error
216 read_calibration error
220 read_calibration error
224 read_calibration error
228 read_calibration error
232 read_calibration error
236 read_calibration error
240 [AUTO DEBIO]32bit,1 ranks training success!
242 DRAM CLK =720 MHz
252 DRAM Type =3 (3DDR3,4:DDR4,7:LPDDR3,8:LPDDR4)
259 Actual DRAM SIZE =8024 M
261 DRAM SIZE =1024 Mbytes, para1 = 30fa, para2 = 4000000, dram_tpr13 = 6041
275 DRAM simple test OK
277 rts standby flag is 0x0, super standby flag is 0x0
282 dram size =1024
284 *****dram handle ok****
288 card no is 0
290 sdcard 0 line count 4
291 [mmc]: mmc driver ver 2019-12-19 10:41
297 [mmc]: sdc0 spd mode error, 2
305 [mmc]: set f_max to 500, set f_max_ddr to 25M
305 [mmc]: mmc 0 bias 0
313 [mmc]: Wrong media type 0x0
316 [mmc]: ***Try SD card 0***
324 [mmc]: HSODP02/SOD25 4 bit
337 [mmc]: 5000000 Hz
339 [mmc]: 15193 MB
341 [mmc]: ***SD MMC 0 init OK!****
343 Loading boot-pkg Succeed(index=0).
447 Entry_name = u-boot
456 Entry_name = monitor
460 Entry_name = dtbo
462 Entry_name = dtb
466 Jump to second Boot.
NOTICE: BL3-1: v1.0(040a)09fec083
NOTICE: BL3-1: Built : 17:08:29, 2020-05-20

```

UNREGISTERED VERSION - Please support Mobaxterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>



3. Instructions for use of Linux system

3.1. Supported linux distribution types and kernel versions

Release type	Kernel version	Server version	Desktop version
Debian 10	Linux 4.4	support	support
Ubuntu 20.04	Linux5.10	support	support
Debian 11	Linux5.10	support	support

3.2. Linux kernel driver adaptation

Function	Linux4.4	Linux5.10
USB2.0x2	OK	OK
USB3.0x1	OK	OK
USB Type-C 3.0	OK	NO
Type-C (Power supply)	OK	OK
MiniPCIE	OK	OK
gpio (26pin)	OK	OK
spi/uart4 (26pin)	OK	OK
i2c8 (26pin)	OK	OK
i2c3 (26pin)	OK	OK
pwm (26pin)	OK	OK
Debug serial port	OK	OK
EMMC	OK	OK
TF card boot	OK	OK
HDMI video	OK	OK
HDMI audio	OK	OK
MIPI camera 1	OK	NO
MIPI Camera2/Lcd2	OK	NO
Lcd1	OK	OK
Gigabit Ethernet port YT8531C	OK	OK
Network port status light	OK	OK
MIC	OK	OK



Headphone playback	OK	OK
Headphone recording	OK	OK
WIFI	OK	OK
Bluetooth	OK	OK
LED lights	OK	OK
Type-C to HDMI display	OK	OK
GPU	OK	OK
Hardware codec	OK	OK
Reset button	OK	OK
webGL hardware acceleration	OK	OK
Dual camera display at the same time	OK	NO
MIPI screen dual screen	OK	OK
simultaneous display	OK	OK
watchdog test	OK	OK

3.3. Onboard LED light display description

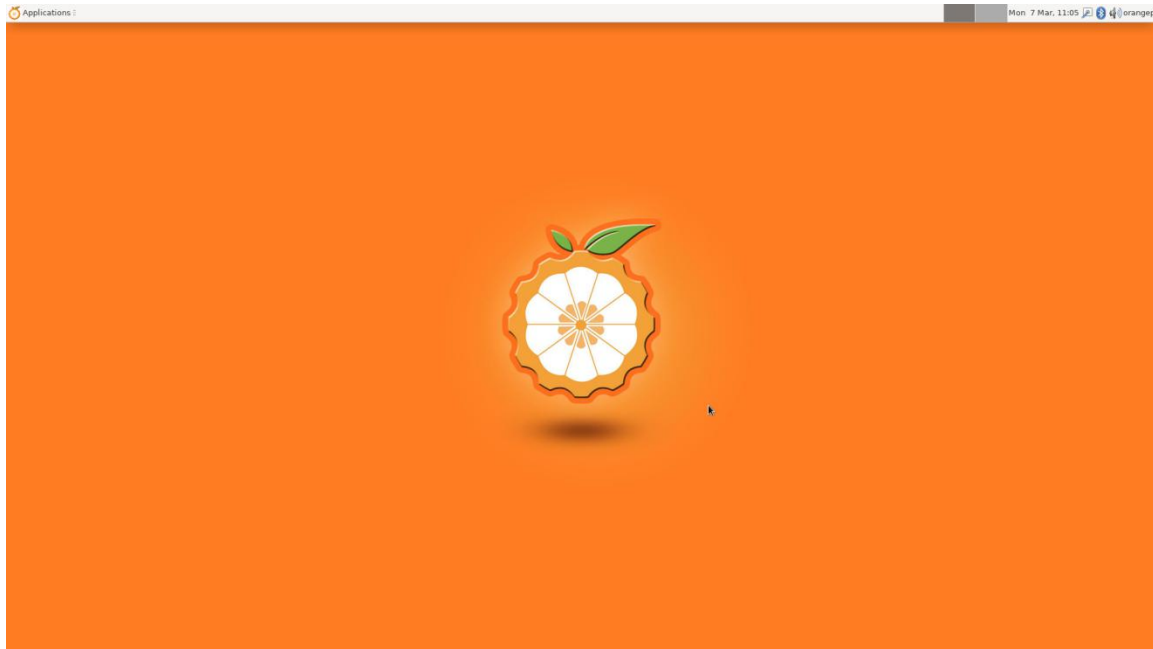
	green light	red light
u-boot startup phase	off	on
The kernel boots into the system	flashing	bright

3.4. Linux system default login account and password

Account	Password
root	orangeypi
orangeypi	orangeypi

3.5. Instructions for automatic login of Linux desktop system

1) The desktop version of the system will automatically log in to the desktop after starting by default, no need to enter a password



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

```
root@orangepi4-lts:~# sed -i "s/autologin-user=orangepi/#autologin-user=orangepi/"  
/etc/lightdm/lightdm.conf.d/22-orangepi-autologin.conf
```

3) After modification, the configuration of `/etc/lightdm/lightdm.conf.d/22-orangepi-autologin.conf` is as follows

```
root@orangepi4-lts:~# 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 system, the login dialog box will appear, at this time, you need to enter the password to enter the system



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

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

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

```

root@orangepi4-lts:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            1.9G   0    1.9G   0% /dev
tmpfs           382M  5.6M  376M   2% /run
/dev/mmcblk0p1  15G   2.5G  12G   18% /
tmpfs           1.9G  140K  1.9G   1% /dev/shm
tmpfs           5.0M   4.0K  5.0M   1% /run/lock
tmpfs           1.9G   0    1.9G   0% /sys/fs/cgroup

```



tmpfs	1.9G	12K	1.9G	1% /tmp
/dev/zram0	49M	3.3M	42M	8% /var/log
cgmfs	100K	0	100K	0% /run/cgmanager/fs
tmpfs	382M	8.0K	382M	1% /run/user/1000
tmpfs	382M	0	382M	0% /run/user/0

3) It should be noted that the Linux system has only one partition in ext4 format, and a separate BOOT partition is not used to store files such as kernel images, so there is no problem of BOOT partition expansion.

4) In addition, if you do not need to automatically expand rootfs, you can use the following method to prohibit

- a. First burn the linux image to the TF card
- b. Then insert the TF card into the Ubuntu PC (Windows does not work), the Ubuntu PC will generally automatically mount the partition of the TF card. If the automatic mounting is normal, you can use the ls command to see the following output. The partition name of the TF card is shown in the following command. The name is not necessarily the same, please modify it according to the actual situation

```
test@test:~$ ls /media/test/27e62f92-8250-4ef1-83db-3d8f0c2e23db/
bin boot dev etc home lib lost+found media mnt opt proc root run
sbin selinux srv sys tmp usr var
```

- c. Then switch the current user to the root user in the Ubuntu PC

```
test@test:~$ sudo -i
[sudo] test 的密码:
root@test:~#
```

- d. Then enter the root directory of the Linux system in the TF card and create a new file named `.no_rootfs_resize`

```
root@test:~# cd /media/test/27e62f92-8250-4ef1-83db-3d8f0c2e23db
root@test:/media/test/27e62f92-8250-4ef1-83db-3d8f0c2e23db# cd root
root@test:/media/test/27e62f92-8250-4ef1-83db-3d8f0c2e23db/root# touch .no_rootfs_resize
root@test:/media/test/27e62f92-8250-4ef1-83db-3d8f0c2e23db/root# ls .no_rootfs*
.no_rootfs_resize
```

- e. Then you can uninstall the TF card, then pull out the TF card and insert it into the development board to start. When the linux system starts, when it is detected



that there is a `.no_rootfs_resize` file in the `/root` directory, the rootfs will no longer be automatically expanded.

- f. After prohibiting the automatic expansion of rootfs, you can see that the available capacity of the TF card is only about 200M

```
root@orangepi4-lts:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            1.9G   0  1.9G   0% /dev
tmpfs           382M  5.6M  376M   2% /run
/dev/mmcblk0p1  2.9G  2.3G  487M  83% /
tmpfs           1.9G  140K  1.9G   1% /dev/shm
tmpfs           5.0M   4.0K  5.0M   1% /run/lock
tmpfs           1.9G   0  1.9G   0% /sys/fs/cgroup
tmpfs           1.9G  12K  1.9G   1% /tmp
/dev/zram0      49M   2.6M   43M   6% /var/log
cgmanagerfs    100K   0  100K   0% /run/cgmanager/fs
tmpfs           382M   0  382M   0% /run/user/0
tmpfs           382M  12K  382M   1% /run/user/1000
```

3. 7. How to modify the linux log level (loglevel)

- 1) The loglevel of the linux system 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 2.2.0 Bionic ttyFIQ0

orangepi login:
```

- 2) When there is a problem with the startup of the Linux system, you can use the following method to modify the value of loglevel, so as to print more log information to the serial port display, which is convenient for debugging. If the Linux system fails to start and cannot enter the system, you can insert the TF card into the Ubuntu PC through the card reader, and then directly modify the configuration of the Linux system in the TF



card after mounting the TF card in the Ubuntu PC. Insert the TF card into the development board to start

```
root@orangepi4-lts:~# sed -i "s/verbosity=1/verbosity=7/" /boot/orangepiEnv.txt
root@orangepi4-lts:~# sed -i "s/console=both/console=serial/" /boot/orangepiEnv.txt
```

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

```
root@orangepi4-lts:~# cat /boot/orangepiEnv.txt
verbosity=7
console=serial
```

4) Then restart the development board, the output information of the kernel will be printed to the serial port 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 system 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@orangepi4-lts:~# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.1.122  netmask 255.255.255.0  broadcast 192.168.1.255
    inet6 fe80::4443:cee2:9b38:ff21  prefixlen 64  scopeid 0x20<link>
    ether 0a:b4:bd:10:02:90  txqueuelen 1000  (Ethernet)
    RX packets 14616  bytes 14141207 (14.1 MB)
    RX errors 0  dropped 1  overruns 0  frame 0
    TX packets 6845  bytes 725742 (725.7 KB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
    device interrupt 24
```

There are three ways to check the IP address after the development board is



started:

1. Connect the HDMI display, then log in to the system and use the ifconfig command to check the IP address

2. Enter the ifconfig command in the debug serial terminal to view the IP address

3. If there is no debugging serial port and no HDMI display, you can also view the IP address of the network port of the development board through the management interface of the router. However, this method often fails to see the IP address of the development board. If you can't see it, the debug method looks like this:

A) First check whether the Linux system has been started normally. If the green light of the development board is flashing, it is generally started normally. If only the red light is on, it means that the system has not started normally.

B) Check whether the network cable is plugged in tightly, or try another network cable

C) Try another router (I have encountered many problems with the router, such as the router cannot assign an IP address normally, or the IP address has been assigned normally but cannot be seen in the router)

D) If there is no router to replace, you can only connect the HDMI display or use the debug serial port to view the IP address

In addition, it should be noted that the development board DHCP automatically assigns an IP address without any settings.

4) The command to test network connectivity is as follows

```
root@orangePi4-lts:~# ping www.baidu.com -I eth0
PING www.a.shifen.com (14.215.177.39) from 192.168.1.122 eth0: 56(84) bytes of data.
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=1 ttl=56 time=8.56 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=2 ttl=56 time=7.66 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=3 ttl=56 time=9.28 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=4 ttl=56 time=8.29 ms
^C
--- www.a.shifen.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 7.666/8.454/9.286/0.584 ms
```

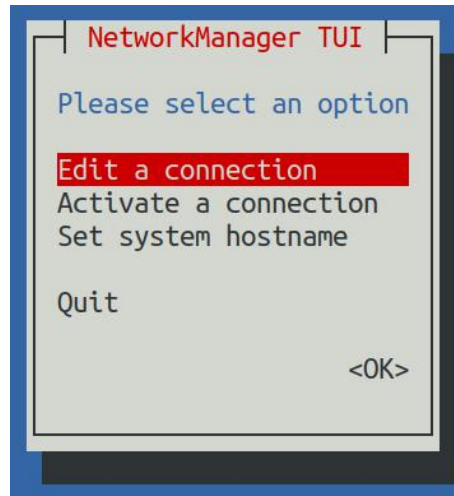


3.9. How to set a static IP address

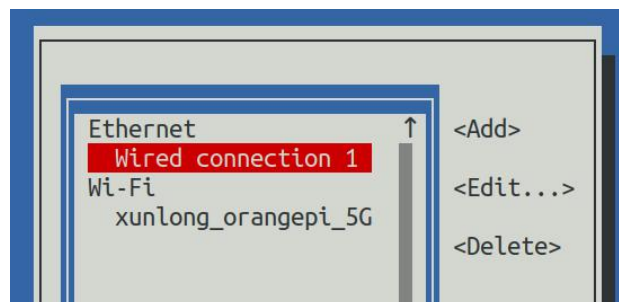
1) First run the nmtui command

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

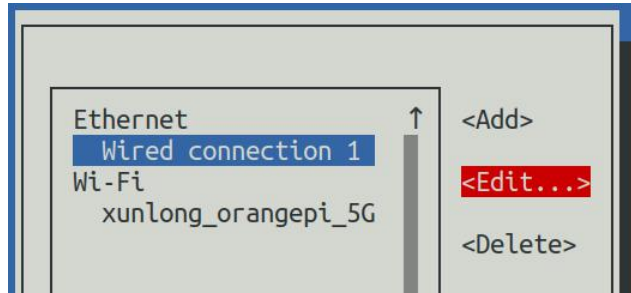
2) Then select Edit a connection and press enter



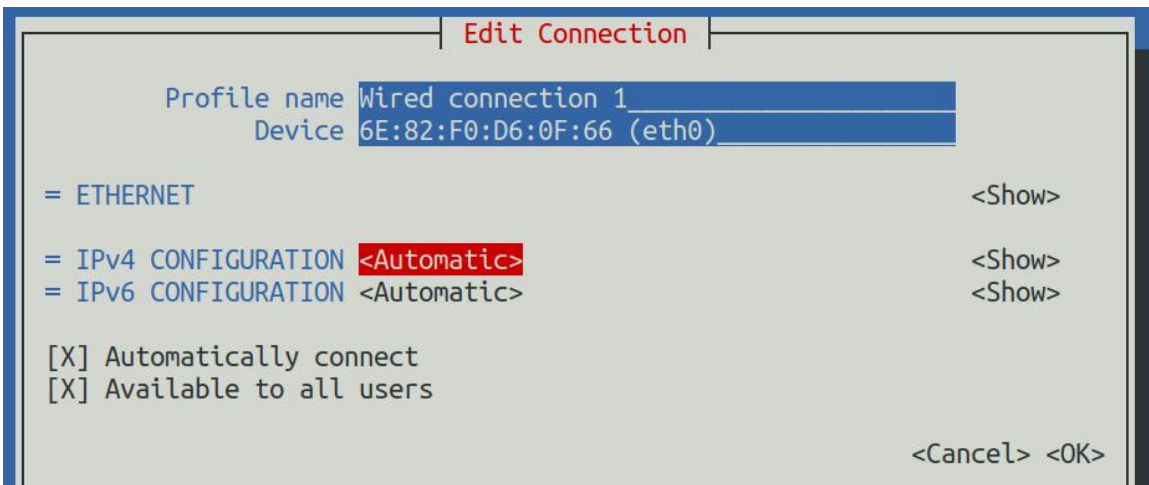
3) Then select the network interface that needs to set a static IP address, such as setting the static IP address of the Ethernet interface and select Wired connection 1.



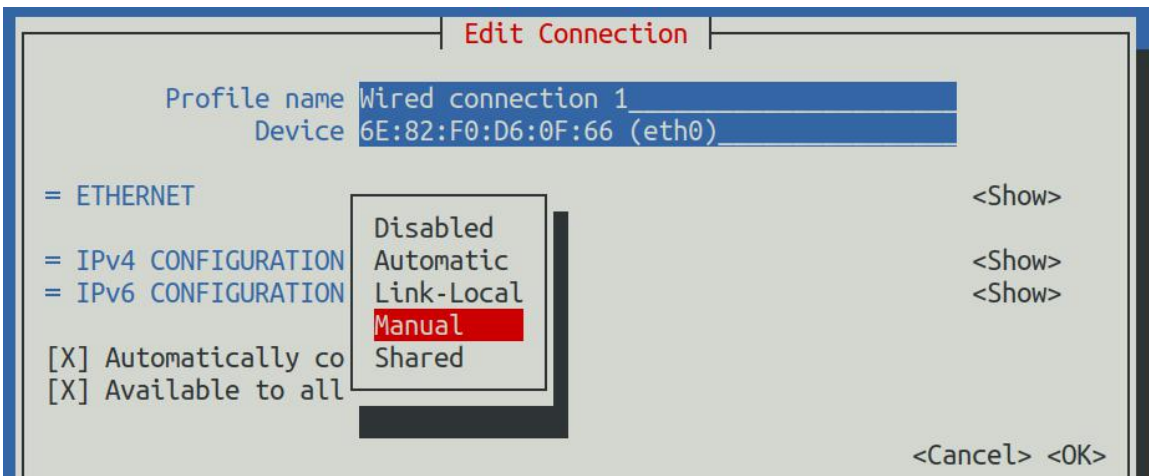
4) Then select Edit by Tab key and press Enter



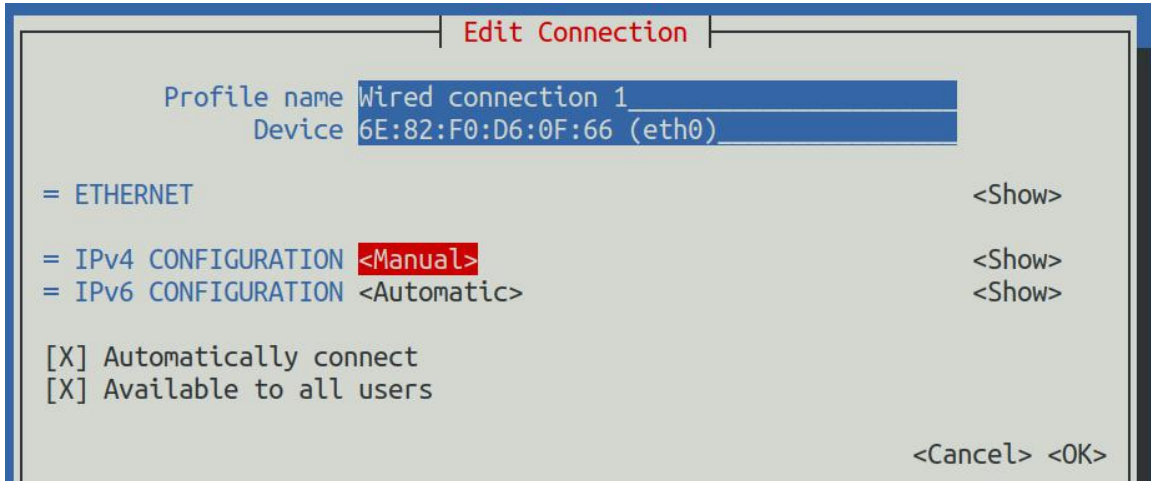
5) Then use the Tab key to move the cursor to the <Automatic> position shown in the figure below to configure IPv4



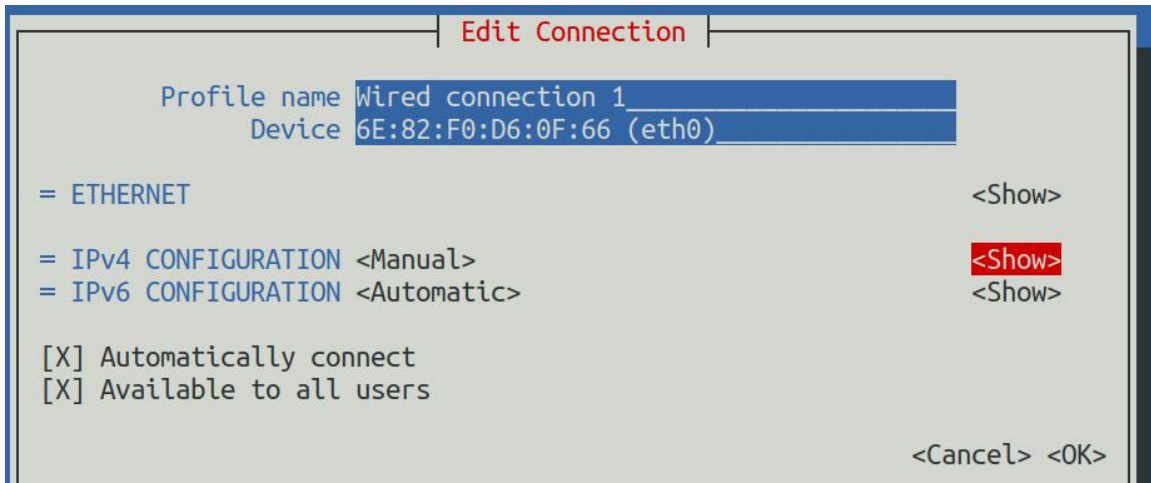
6) Then press Enter, use the up and down arrow keys to select Manual, then press Enter to confirm



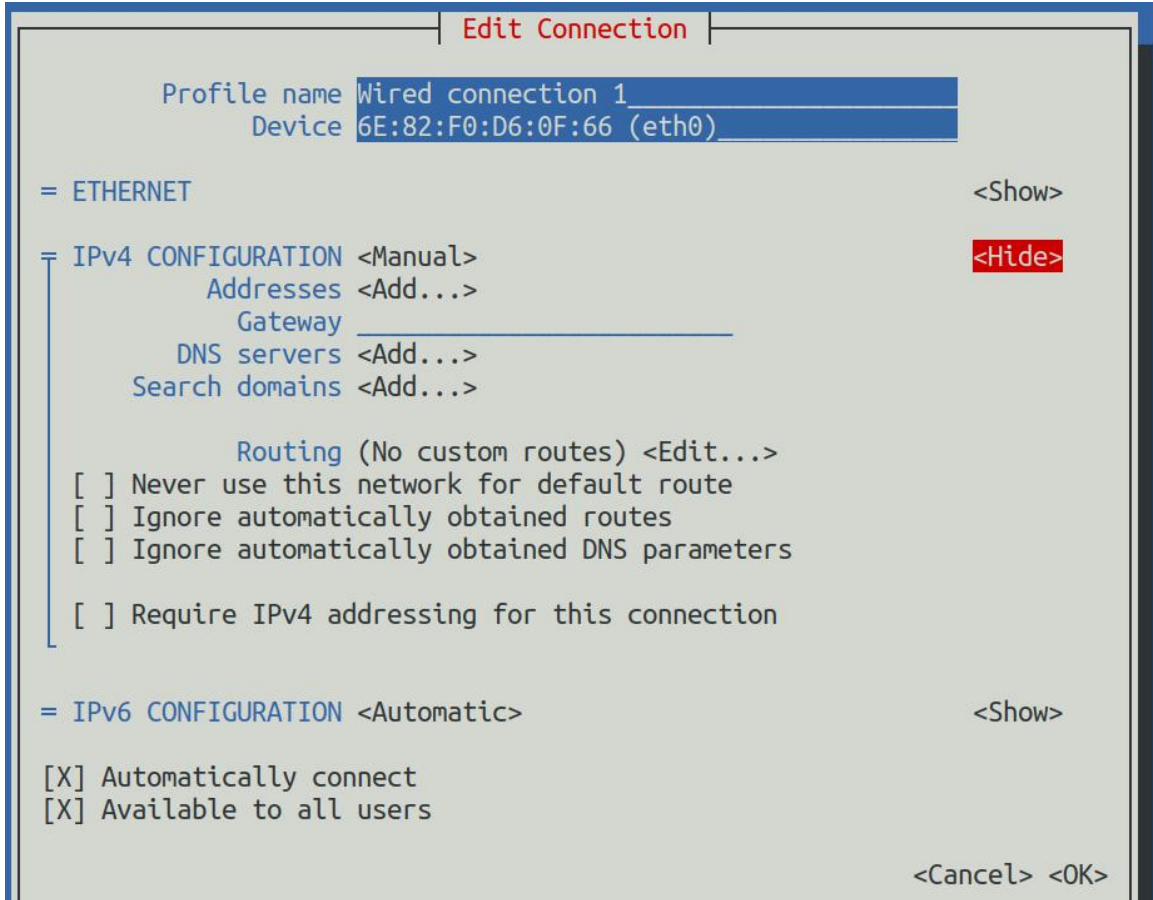
7) The display after selection is as shown below



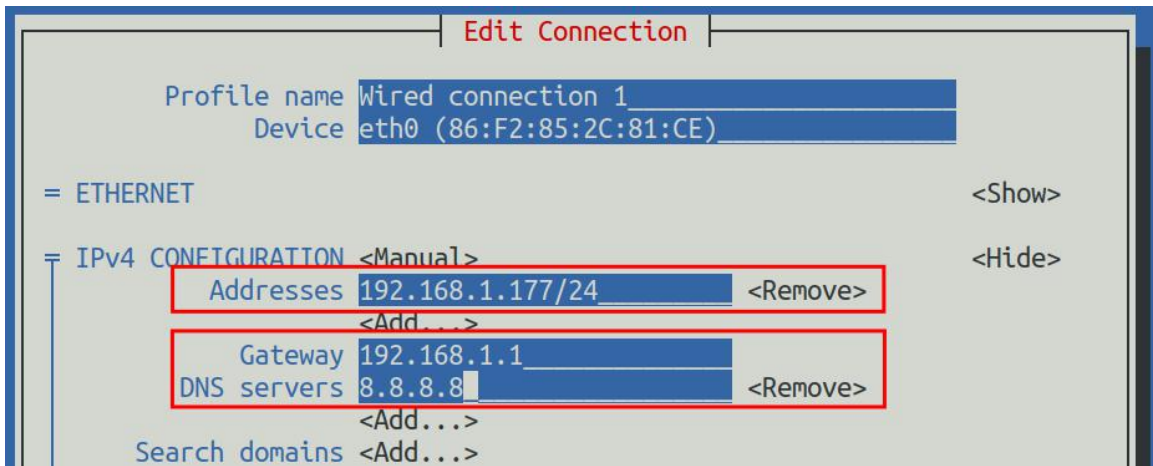
8) Then move the cursor to <Show> by Tab key



9) Then press Enter, the following setting interface will pop up after pressing Enter



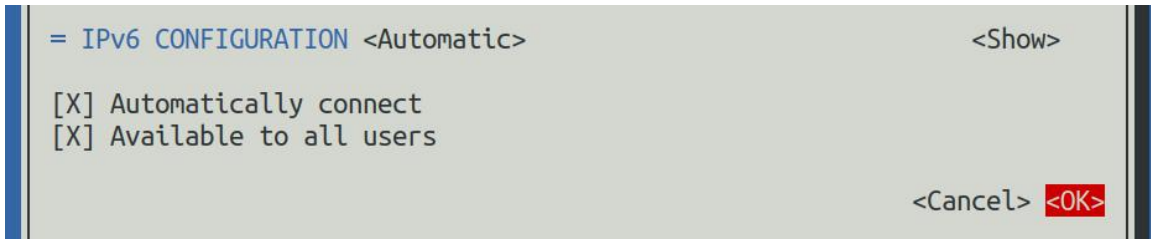
10) Then you can set the IP address (Addresses), gateway (Gateway) and DNS server address in the location shown in the figure below (there are many other setting options, please explore by yourself), please set according to your specific needs, The value set in the image below is just an example



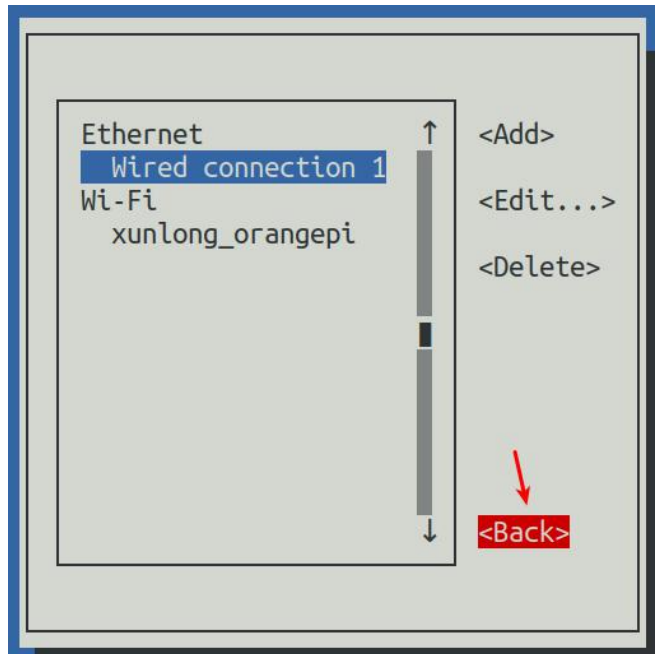
11) After setting, move the cursor to <OK> in the lower right corner, then press Enter to



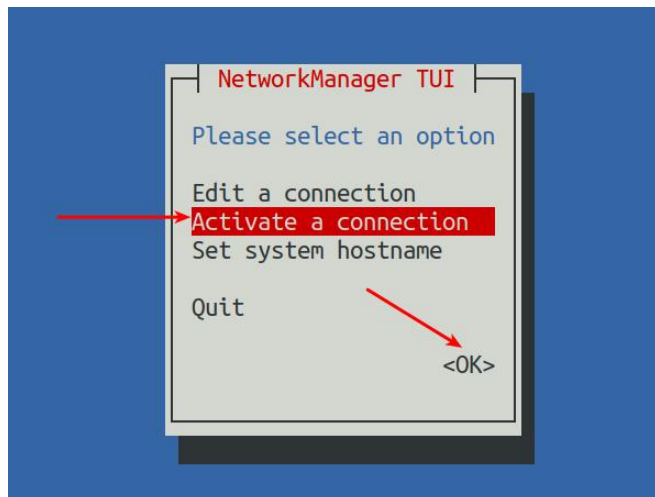
confirm



12) Then click <Back> to return to the previous selection interface

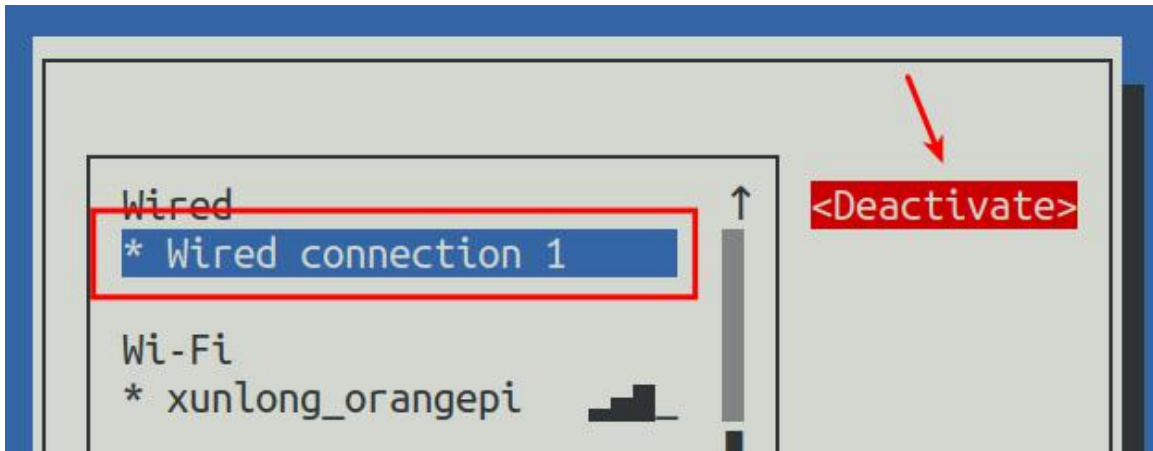


13) Then select Activate a connection, move the cursor to <OK>, and finally click Enter

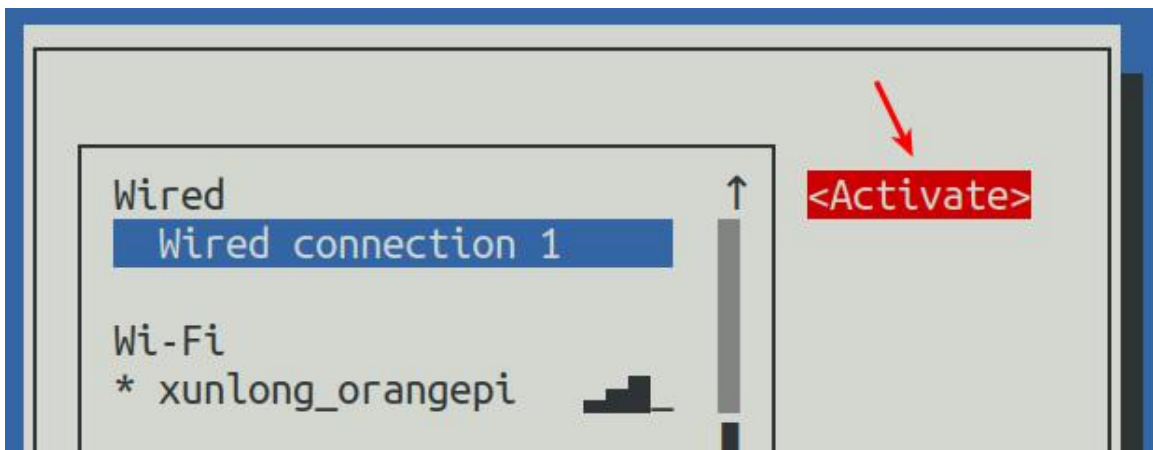




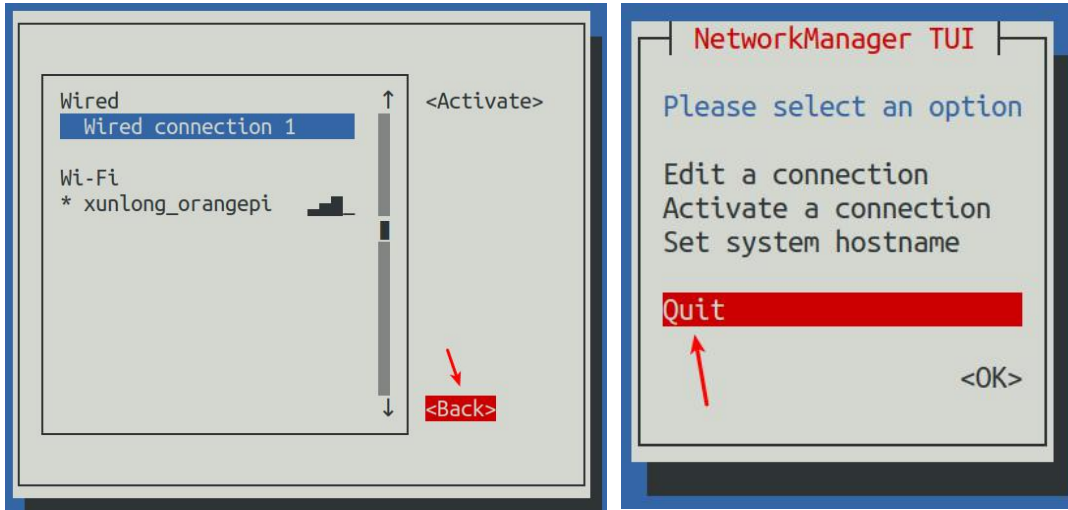
14) Then select the network interface to be set, such as Wired connection 1, then move the cursor to <Deactivate>, and press Enter to disable Wired connection 1



15) Then please do not move the cursor, and then press the Enter key to re-enable Wired connection 1, so that the static IP address set earlier will take effect



16) Then exit nmtui through the <Back> and Quit buttons



17) Then through ifconfig, you can see that the IP address of the network port has become the static IP address set earlier

```
root@orangepi:~# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.1.177  netmask 255.255.255.0  broadcast 192.168.1.255
    inet6 fe80::69f6:23d3:a3f3:9772  prefixlen 64  scopeid 0x20<link>
    ether 86:f2:85:2c:81:ce  txqueuelen 1000  (Ethernet)
    RX packets 47817  bytes 4047732 (4.0 MB)
    RX errors 0  dropped 1563  overruns 0  frame 0
    TX packets 1815  bytes 295954 (295.9 KB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
    device interrupt 64
```

18) Then you can test the connectivity of the network to check whether the IP address is configured OK

```
root@orangepi:~# ping 192.168.1.47 -I eth0
PING 192.168.1.47 (192.168.1.47) from 192.168.1.188 eth0: 56(84) bytes of data.
64 bytes from 192.168.1.47: icmp_seq=1 ttl=64 time=0.233 ms
64 bytes from 192.168.1.47: icmp_seq=2 ttl=64 time=0.263 ms
64 bytes from 192.168.1.47: icmp_seq=3 ttl=64 time=0.273 ms
64 bytes from 192.168.1.47: icmp_seq=4 ttl=64 time=0.269 ms
64 bytes from 192.168.1.47: icmp_seq=5 ttl=64 time=0.275 ms
^C
--- 192.168.1.47 ping statistics ---
```



```
5 packets transmitted, 5 received, 0% packet loss, time 4042ms
rtt min/avg/max/mdev = 0.233/0.262/0.275/0.015 ms
```

3. 10. SSH remote login development board

By default, Linux systems enable ssh remote login and allow root users to log in to the system. Before ssh login, you need to ensure that the Ethernet or wifi network is connected, and then use the ifconfig command or obtain the IP address of the development board by viewing the router

3. 10. 1. SSH remote login development board under Ubuntu

1) Obtain the IP address of the development board

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

```
test@test:~$ ssh root@192.168.1.57 (It needs to be replaced with the IP address of the
development board)
root@192.168.1.57's password: (Enter the password here, the default password is
orangeypi)
```

3) After successfully logging in to the system, the display is as shown below

```
csy@ubuntu:~$ ssh root@192.168.1.57
root@192.168.1.57's password:
Welcome to Orange Pi 2.2.0 Bullseye with Linux 5.10.43

System load: 25%          Up time: 5 min    Local users: 2
Memory usage: 16% of 3.78G  IP: 192.168.1.57
CPU temp: 41°C           Usage of /: 22% of 15G

[ 0 security updates available, 26 updates total: apt upgrade ]
Last check: 2022-02-22 05:59

[ General system configuration (beta): orangepi-config ]

Last login: Tue Feb 22 06:03:37 2022 from 192.168.1.78
```

If ssh cannot log in to the Linux system normally, you can enter the following

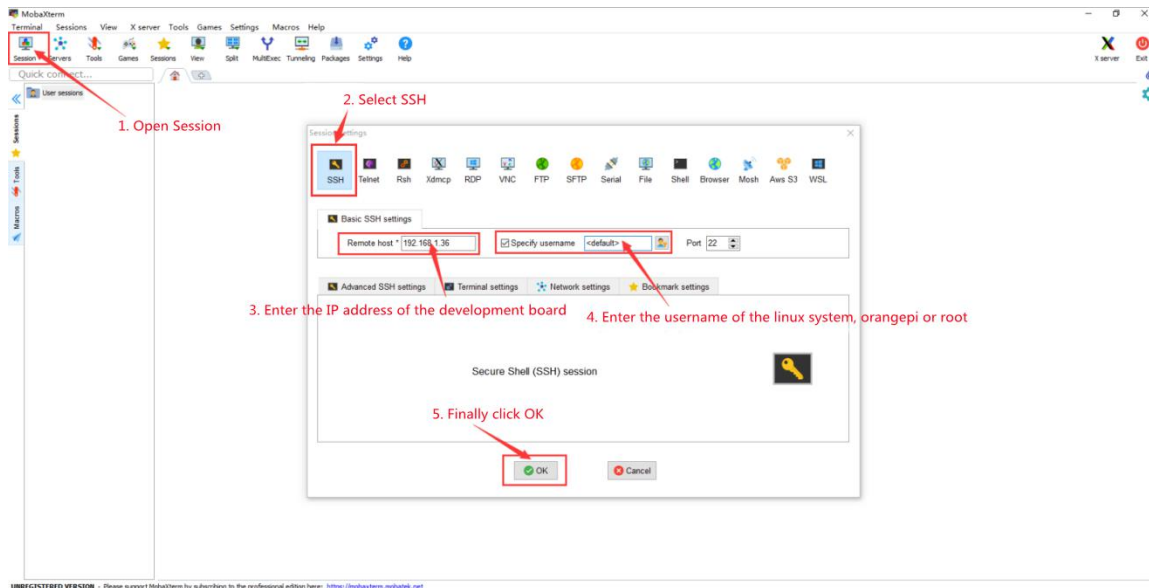


command on the development board and then try to connect:

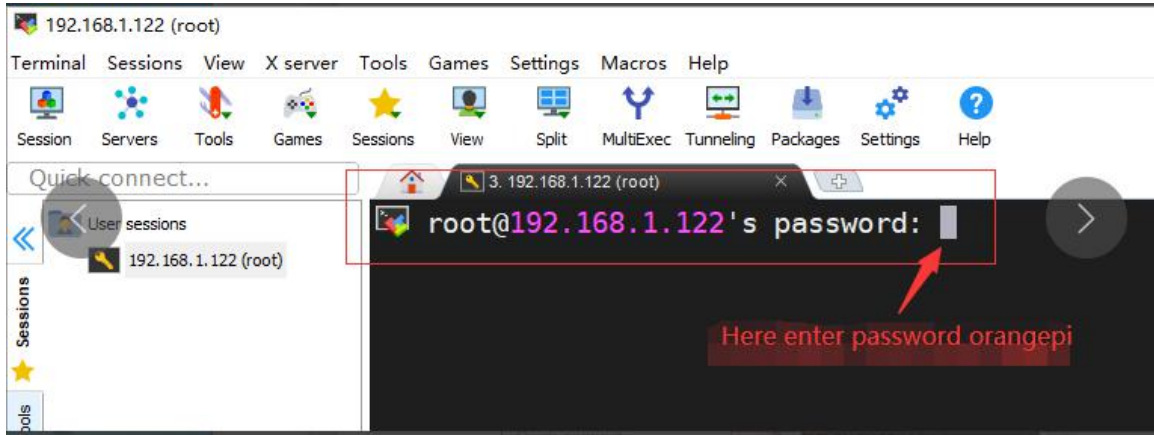
```
root@orangeypi:~# rm /etc/ssh/ssh_host_*  
root@orangeypi:~# dpkg-reconfigure openssh-server
```

3. 10. 2. SSH remote login development board under Windows

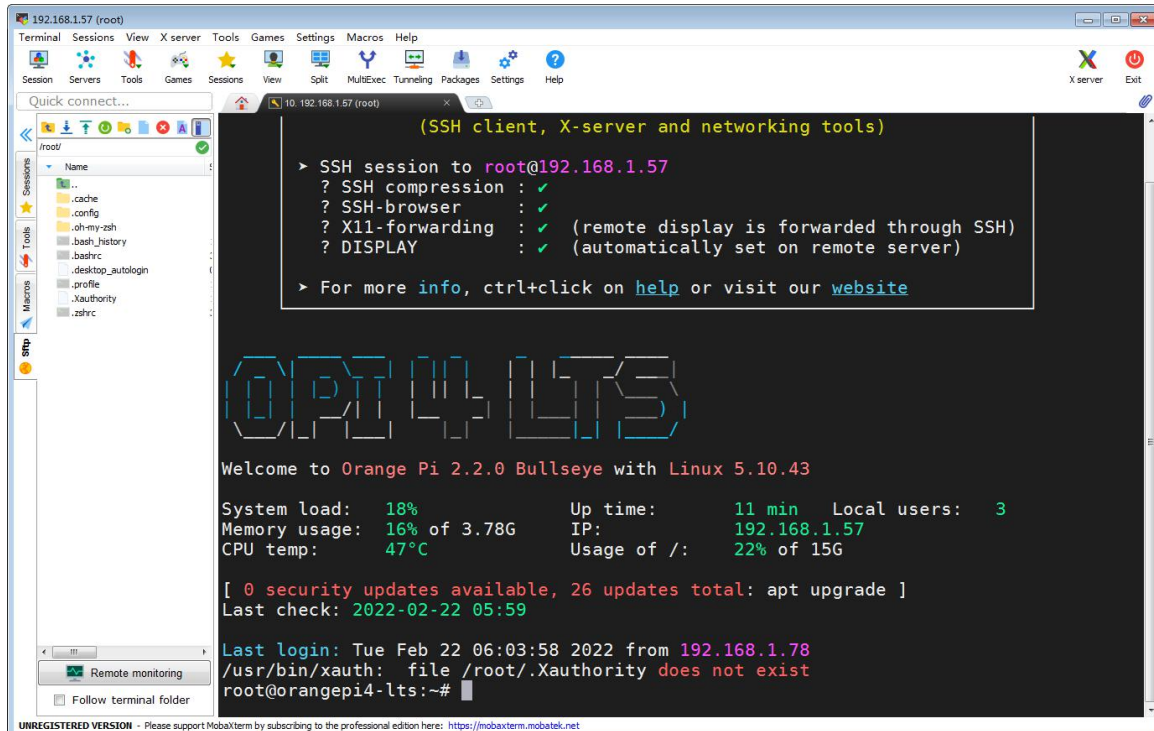
- 1) First get the IP address of the development board
- 2) Under Windows, you can use MobaXterm to remotely log in to the development board, first create a new ssh session
 - a. Open Session
 - b. Then select SSH in Session Setting
 - c. Then enter the IP address of the development board in Remote host
 - d. Then enter the username root or orangeypi of the linux system in Specify username
 - e. Finally click OK



3) Then you will be prompted to enter a password. The default passwords for both root and orangeypi users are orangeypi



4) After successfully logging in to the system, the display is as shown below



3. 11. HDMI display test

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



2) After starting the linux system, if the HDMI display has image output, it means that the HDMI interface is working normally

3. 12. Type C to HDMI display test

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



2) After starting the linux system, if the HDMI display has image output, it means that the Type C to HDMI display function is normal

3. 13. HDMI to VGA display test

1) First you need to prepare the following accessories

- a. HDMI to VGA converter

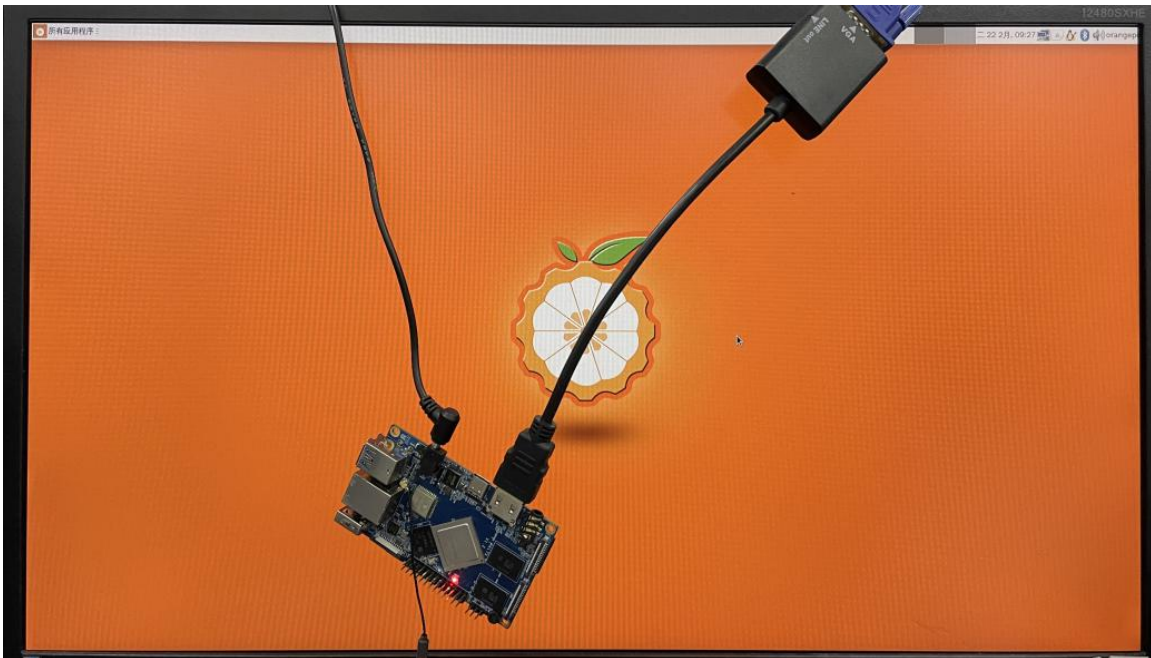


- b. A VGA cable



c. A monitor or TV that supports VGA interface

2) HDMI to VGA display test as shown below



When using HDMI to VGA display, the development board and the Linux system of the development board do not need to do any settings, as long as the HDMI interface of the development board can display normally. So if there is a problem with the test, please check the HDMI to VGA converter, VGA cable and monitor if there is any problem

3. 14. WIFI connection test

Please do not connect to WIFI by modifying the `/etc/network/interfaces` configuration file. There will be problems with connecting to the WIFI network in this way.



3.14.1. The server version image is connected to WIFI through the command

When the development board is not connected to the Ethernet, not connected to the HDMI display, and only connected to the serial port, it is recommended to use the commands demonstrated in this section to connect to the WIFI network. Because nmtui can only display characters in some serial port software (such as minicom), it cannot display the graphical interface normally. Of course, if the development board is connected to an Ethernet or HDMI display, you can also use the commands demonstrated in this section to connect to the WIFI network

- 1) First log in to the linux system, there are the following three ways
 - a. If the development board is connected to the network cable, you can log in to the Linux system remotely through ssh
 - b. If the development board is connected to the debugging serial port, you can use the serial port terminal to log in to the linux system
 - c. If the development board is connected to the HDMI display, you can log in to the linux system through the HDMI display terminal

- 1) First use the nmcli dev wifi command to scan the surrounding WIFI hotspots

```
root@orangepi4-lts:~# nmcli dev wifi
```

```
root@orangepi4-lts:~# nmcli dev wifi
IN-USE BSSID SSID MODE CHAN RATE
CA:50:E9:89:E2:44 Infra 1 130 Mbit/s
EE:9F:80:DF:4F:3E -- Infra 12 130 Mbit/s
E8:9F:80:DF:4F:3E orangepi Infra 12 130 Mbit/s
06:0E:3C:84:85:E9 Infra 1 65 Mbit/s
E8:9F:80:DF:4F:40 orangepi_5G Infra 157 540 Mbit/s
34:79:16:01:BA:AC Infra 13 270 Mbit/s
E8:9F:80:DF:4F:3F Infra 48 270 Mbit/s
A0:40:A0:A1:72:1D Infra 153 405 Mbit/s
A0:40:A0:A1:72:1E Infra 4 195 Mbit/s
00:BD:82:51:53:C2 Infra 11 130 Mbit/s
04:79:70:8D:0C:B4 Infra 6 270 Mbit/s
34:79:16:01:BA:B1 -- Infra 13 270 Mbit/s
```

- 2) Then use the nmcli command to connect to the scanned WIFI hotspot, where:
 - a. wifi_name needs to be replaced with the name of the WIFI hotspot you want to connect to
 - b. wifi_passwd needs to be replaced with the password of the WIFI hotspot you want to connect to

```
root@orangepi4-lts:~# nmcli dev wifi connect wifi_name password wifi_passwd
```



```
Device 'wlan0' successfully activated with '36387224-f4ff-4021-85a1-eda7825ce09e'.
```

3) You can view the IP address of the wifi through the ifconfig command

```
root@orangePi4-lts:~# ifconfig wlan0
wlan0      Link encap:Ethernet  HWaddr b0:02:47:cf:28:77
           inet addr:192.168.1.187  Bcast:192.168.1.255  Mask:255.255.255.0
           inet6 addr: fe80::8008:703d:8f92:41c7/64 Scope:Link
           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
           RX packets:17 errors:0 dropped:0 overruns:0 frame:0
           TX packets:42 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:1000
           RX bytes:1837 (1.8 KB)  TX bytes:8320 (8.3 KB)
```

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

```
root@orangePi4-lts:~# ping www.baidu.com -I wlan0
PING www.a.shifen.com (14.215.177.39) from 192.168.1.187 wlan0: 56(84) bytes of
data.
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=3 ttl=56 time=15.0 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=4 ttl=56 time=12.8 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=5 ttl=56 time=13.7 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=6 ttl=56 time=14.9 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=7 ttl=56 time=13.6 ms
^C
```

3. 14. 2. Connect WIFI graphically in the command line

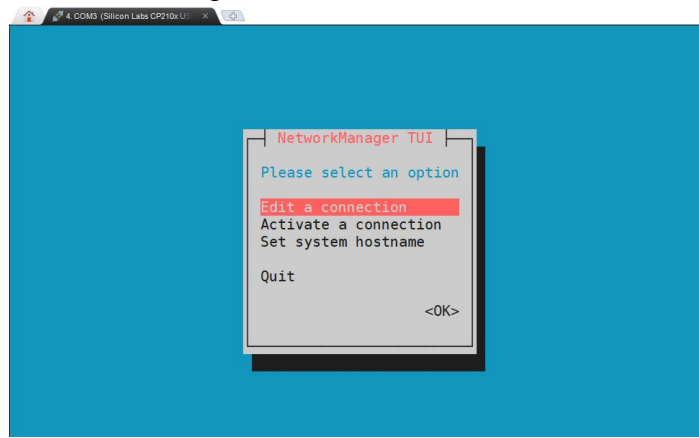
- 1) First log in to the linux system, there are the following three ways
 - a. If the development board is connected to the network cable, you can log in to the Linux system remotely through ssh
 - b. If the development board is connected to the debugging serial port, you can use the serial port terminal to log in to the linux system (please use MobaXterm for serial port software, and the graphical interface cannot be displayed using minicom)
 - c. If the development board is connected to the HDMI display, you can log in to the linux system through the HDMI display terminal



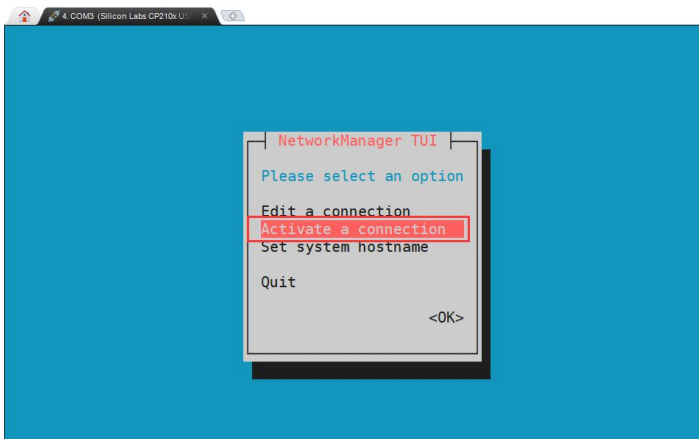
2) Then enter the nmtui command in the command line to open the wifi connection interface

```
root@orangePi4-lts:~# nmtui
```

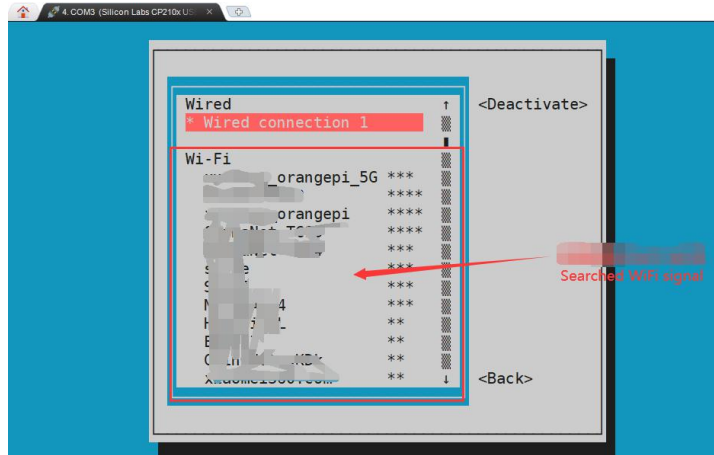
3) Enter the nmtui command to open the interface as shown below



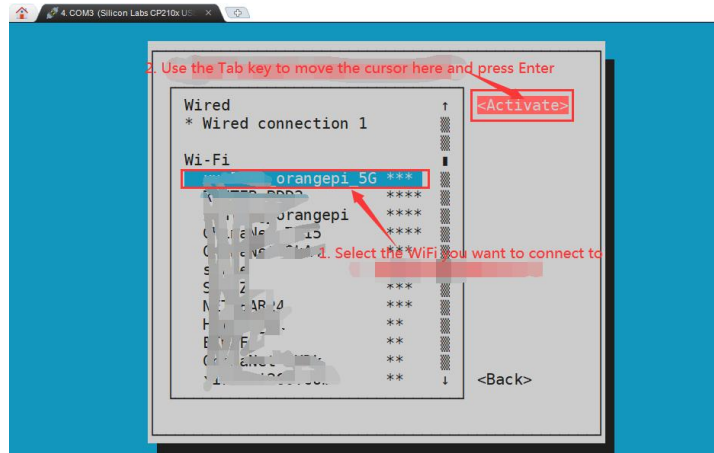
4) Select Activate a connect and press Enter



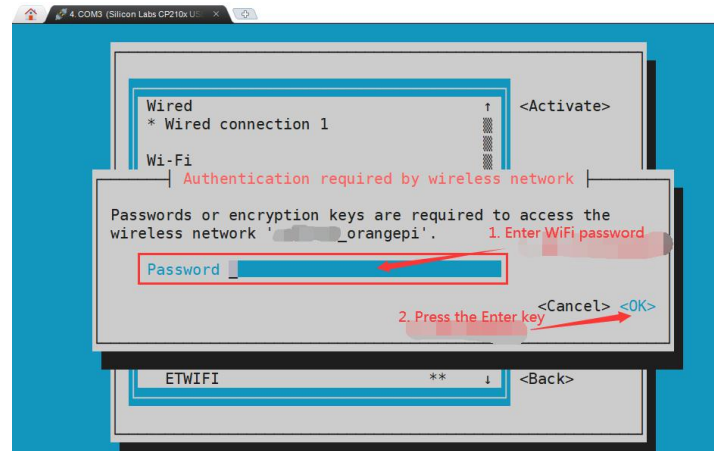
5) Then you can see all the searched WIFI hotspots



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

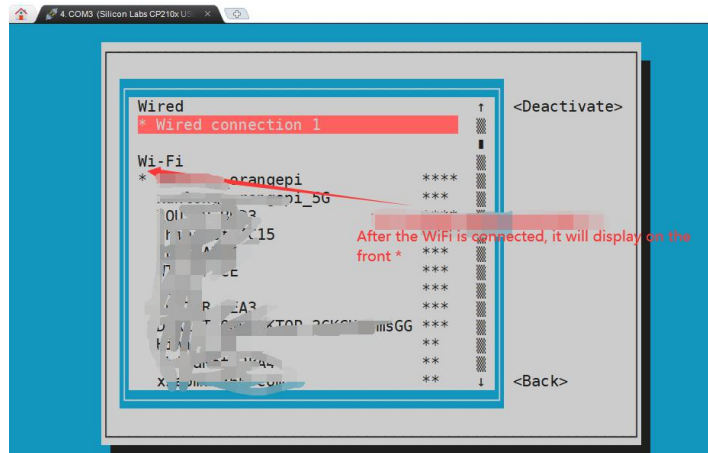


7) Then a dialog box for entering a password will pop up, enter the corresponding password in Password and press Enter to start connecting to WIFI





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



9) You can view the IP address of the wifi through the ifconfig command

```

root@orangepi4-lts:~# ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.1.187  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@orangepi4-lts:~# ping www.baidu.com -I wlan0
PING www.a.shifen.com (14.215.177.39) from 192.168.1.187 wlan0: 56(84) bytes of
data.
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=3 ttl=56 time=15.0 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=4 ttl=56 time=12.8 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=5 ttl=56 time=13.7 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=6 ttl=56 time=14.9 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=7 ttl=56 time=13.6 ms
^C
    
```

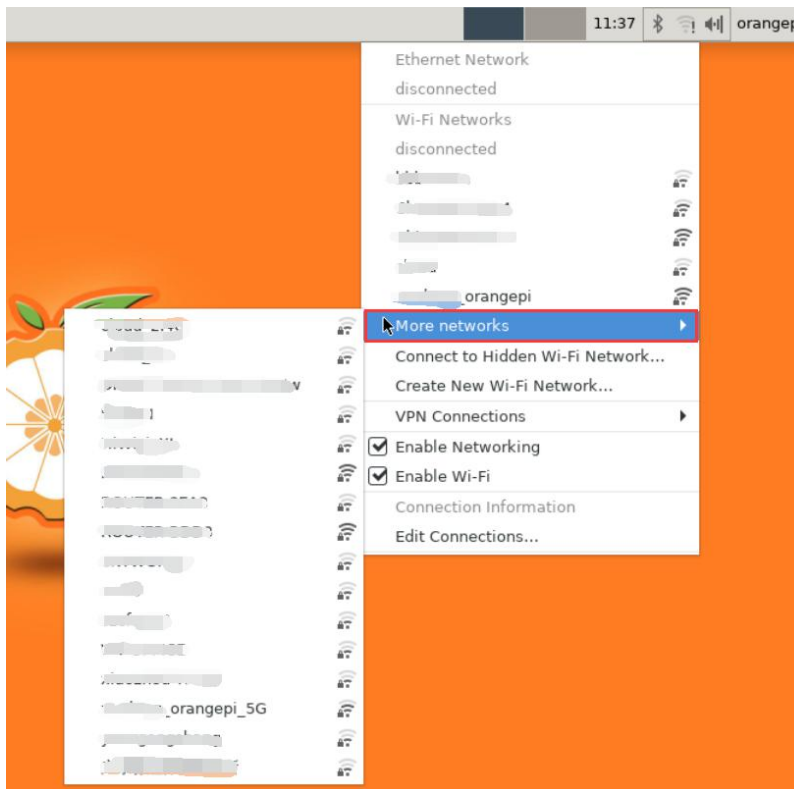


3. 14. 3. How to use the Linux desktop to connect to WIFI

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



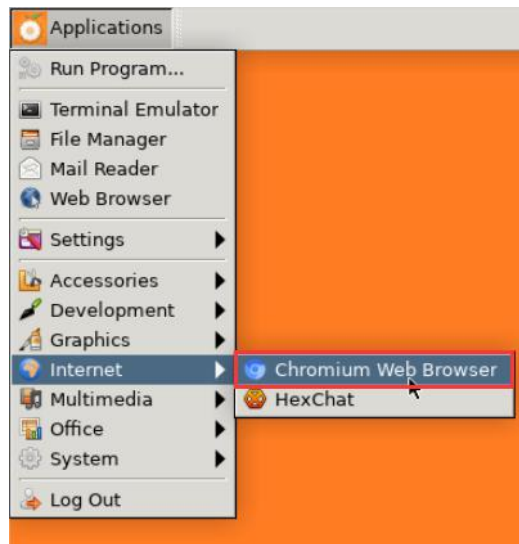
2) Click More networks in the pop-up drop-down box to see all scanned WIFI hotspots, and then select the WIFI hotspot you want to connect to



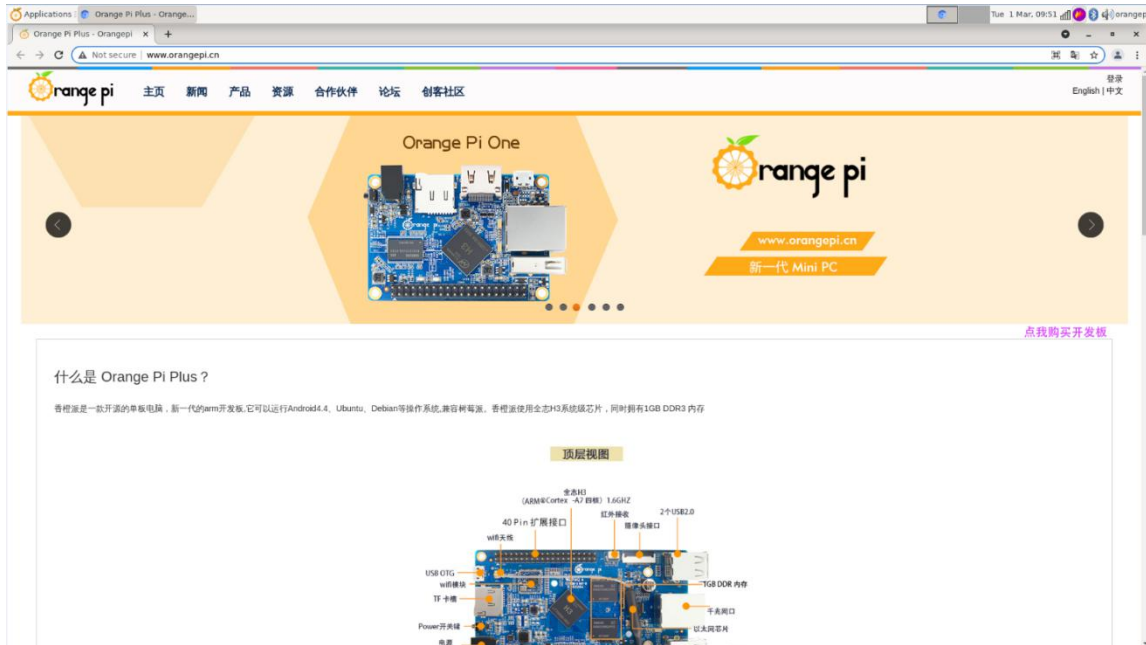
3) Then enter the password of the WIFI hotspot, and then click Connect to start connecting to the WIFI



4) After connecting to WIFI, you can open the browser to check whether you can access 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 Orange Pi website, or you can open other pages, it means the WIFI connection is normal



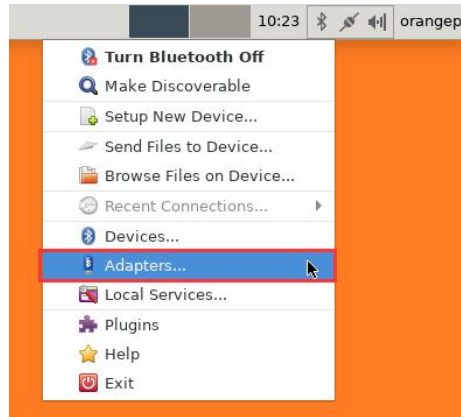
3. 15. How to use Bluetooth

3. 15. 1. Test method of desktop version image

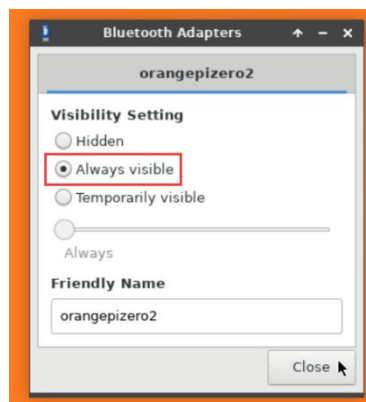
1) Click the Bluetooth icon in the upper right corner of the desktop



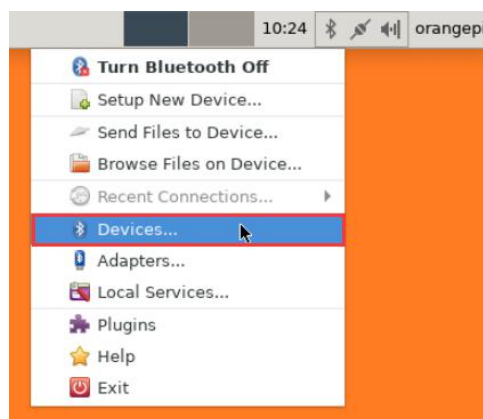
2) Then select adapter



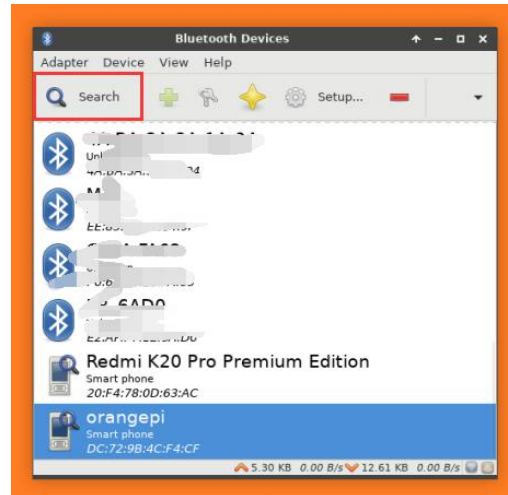
3) In the Bluetooth adapter setting interface, set the Visibility Setting to Always visible, and then click the upper right corner to close the window



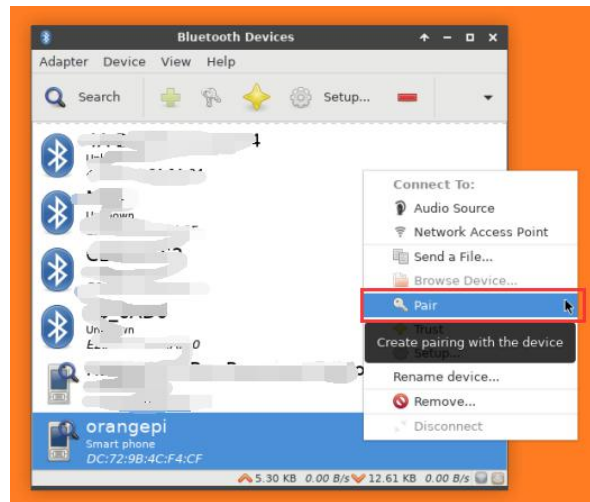
4) Then open the configuration interface of the Bluetooth device



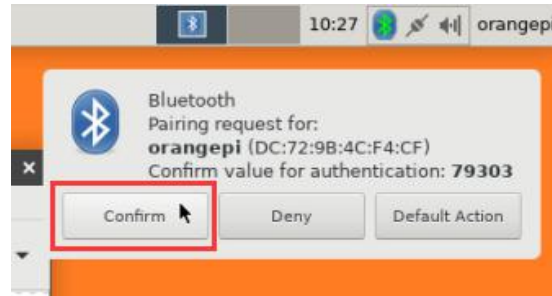
5) Click Search to start scanning for surrounding Bluetooth devices



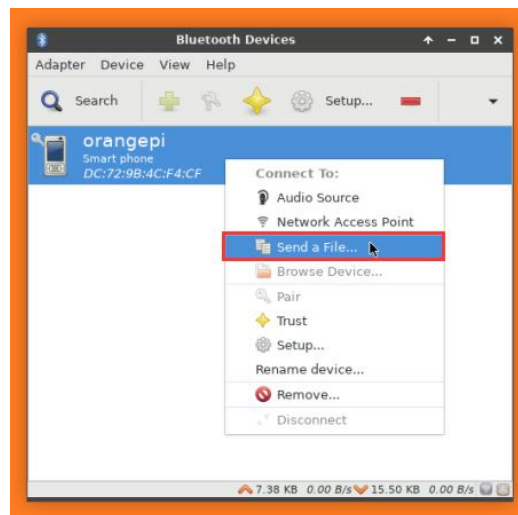
6) Then select the Bluetooth device you want to connect, and then click the right mouse button to pop up the operation interface of the Bluetooth device. Select Pair to start pairing. The demonstration here is pairing with an Android phone.



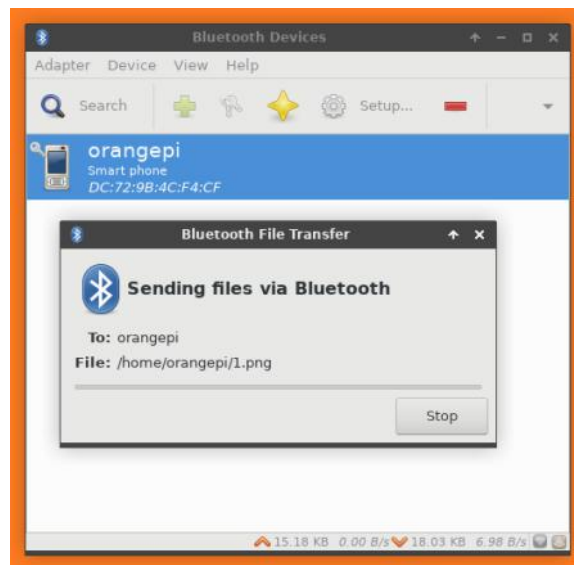
7) When pairing, a pairing confirmation box will pop up in the upper right corner of the desktop. Select Confirm to confirm. At this time, the mobile phone also needs to be confirmed.



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



9) The interface for sending pictures is as follows





3. 15. 2. How to use the server version image

1) After entering the system, you can first check whether there is a Bluetooth device node through the hciconfig command. If there is, it means that the Bluetooth initialization is normal.

```
root@orangepi4-lts:~# hciconfig -a
hci0:  Type: BR/EDR  Bus: UART
      BD Address: 43:45:C5:00:1F:AC  ACL MTU: 1021:8  SCO MTU: 64:1
      UP RUNNING PSCAN ISCAN
      RX bytes:897 acl:0 sco:0 events:65 errors:0
      TX bytes:4355 acl:0 sco:0 commands:65 errors:0
      Features: 0xbf 0xfe 0xcf 0xfe 0xdb 0xff 0x7b 0x87
      Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
      Link policy: RSWITCH SNIFF
      Link mode: SLAVE ACCEPT
      Name: 'orangepi4-lts'
      Class: 0x1c0000
      Service Classes: Rendering, Capturing, Object Transfer
      Device Class: Miscellaneous,
      HCI Version: (0x9)  Revision: 0x26
      LMP Version: (0x9)  Subversion: 0x6606
      Manufacturer: Broadcom Corporation (15)
```

2) Scan for bluetooth devices using bluetoothctl

```
root@orangepi4-lts:~# bluetoothctl
[NEW] Controller 10:11:12:13:14:15 orangepi3 [default]
Agent registered
[bluetooth]# power on      #enable controller
Changing power on succeeded
[bluetooth]# discoverable on  #Make the controller 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
```

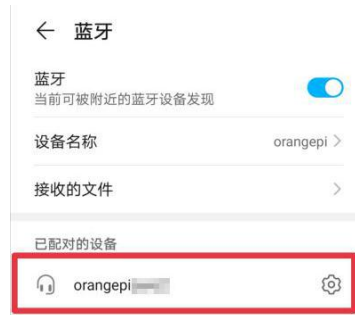


```
[NEW] Device 76:60:79:29:B9:31 76-60-79-29-B9-31
[NEW] Device 9C:2E:A1:42:71:11 Xiaomi phone
[NEW] Device DC:72:9B:4C:F4:CF orangepi
[bluetooth]# scan off      #After scanning the Bluetooth device you want to connect,
you can close the scan, and then write down the MAC address of the Bluetooth
device. 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 needs to use 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 pairing is successful
[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 display of the Bluetooth interface of the mobile phone is as follows



3. 16. USB interface test

3. 16. 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 in normal use (the mouse can only be used in the desktop version of the system)

3. 16. 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@orangepi4-lts:~# 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@orangepi4-lts:~# mount /dev/sda1 /mnt/
root@orangepi4-lts:~# 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@orangepi4-lts:~# df -h | grep "sd"  
/dev/sda1          29G  208K  29G   1% /mnt
```

3. 17. USB camera test

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

2) Through the `v4l2-ctl` (note that the `l` in `v4l2` is a lowercase letter `l`, not the number `1`) command, you can see that the device node information of the USB camera is `/dev/video10`

```
root@orangepi4-lts:~# apt update  
root@orangepi4-lts:~# apt install v4l-utils  
root@orangepi4-lts:~# v4l2-ctl --list-devices  
USB2.0 Camera RGB (usb-xhci-hcd.11.auto-1):  
  /dev/video10
```

Note that the `l` in `v4l2` is a lowercase `l`, not the number `1`.

In addition, the serial number of the video is not necessarily `video10`, please modify it according to the actual situation.

3) Use `fswebcam` to test the USB camera

a. Install `fswebcam`

```
root@orangepi4-lts:~# apt update  
root@orangepi4-lts:~# apt-get -y install fswebcam
```

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

- a) The `-d` option is used to specify the device node of the USB camera
- b) `--no-banner` to remove watermark from photos
- c) The `-r` option is used to specify the resolution of the photo
- d) The `-S` option is set to skip previous frames
- e) `/image.jpg` is used to set the name and path of the generated photo

```
root@orangepi4-lts:~# fswebcam -d /dev/video10 --no-banner -r 1280x720 -S  
5 ./image.jpg
```



- c. If there is no HDMI display or LCD screen connected, after taking the picture, you can use the scp command to transfer the taken picture to the Ubuntu PC for image viewing

```
root@orangePi4-lts:~# scp image.jpg test@192.168.1.55:/home/test (Modify the IP address and path according to the actual situation)
```

- d. If an HDMI display or LCD screen is connected, you can view the captured pictures directly through the HDMI display or LCD screen

4) Use motion to test the USB camera

- a. Install the camera test software motion

```
root@orangePi4-lts:~# apt update
root@orangePi4-lts:~# apt -y install motion
```

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

```
root@orangePi4-lts:~# 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@orangePi4-lts:~# sed -i "s/stream_localhost on/stream_localhost off/" \
/etc/motion/motion.conf (this is a command)
```

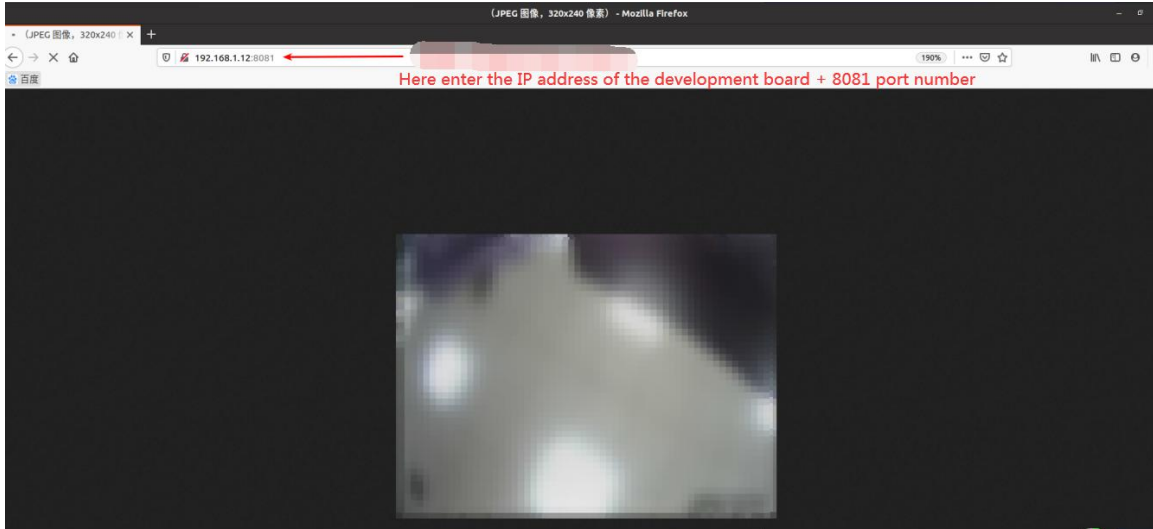
- d. Modify the configuration of /etc/motion/motion.conf and change videodevice /dev/video0 to videodevice /dev/video10

```
root@orangePi4-lts:~# sed -i "s/videodevice \dev\video0/videodevice \dev\video10/"
/etc/motion/motion.conf
```

- e. Then restart the motion service

```
root@orangePi4-lts:~# /etc/init.d/motion restart
[ ok ] Restarting motion (via systemctl): motion.service.
```

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



5) Use mjpg-streamer to test the USB camera

- a. Download mjpg-streamer

```
root@orangepi4-lts:~# git clone https://github.com/jacksonliam/mjpg-streamer
```

- b. b. Install the dependent packages (under debian10, you need to replace libjpeg8-dev with libjpeg62-turbo-dev)

```
root@orangepi4-lts:~# apt-get -y install cmake libjpeg8-dev
```

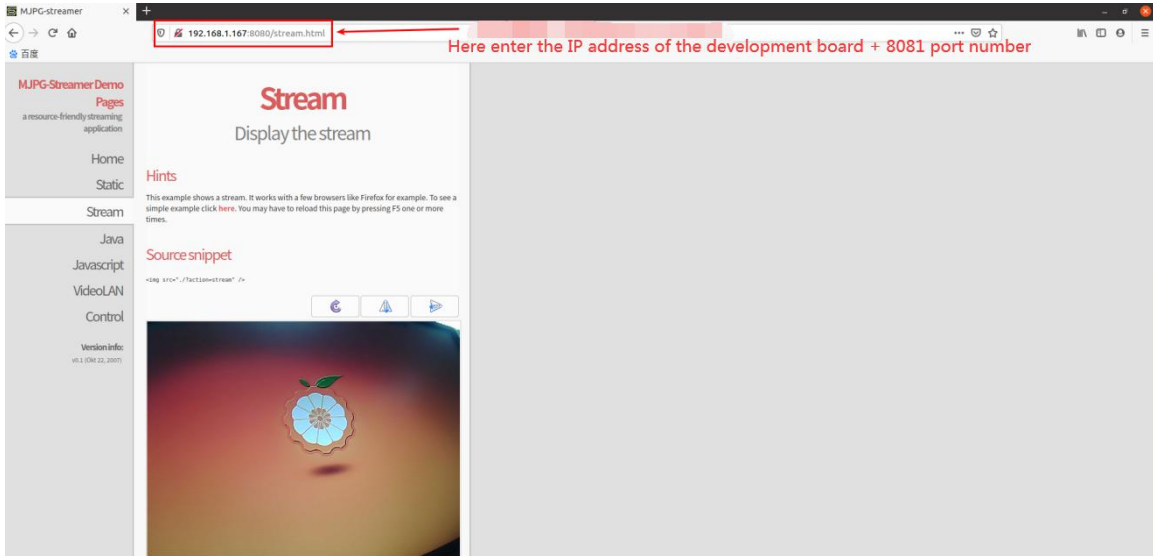
- c. Compile and install mjpg-streamer

```
root@orangepi4-lts:~# cd mjpg-streamer/mjpg-streamer-experimental
root@orangepi4-lts:~/mjpg-streamer/mjpg-streamer-experimental# make
root@orangepi4-lts:~/mjpg-streamer/mjpg-streamer-experimental# make install
```

- d. Then enter the following command to start mjpg_streamer

```
root@orangepi4-lts:~/mjpg-streamer/mjpg-streamer-experimental# export
LD_LIBRARY_PATH=. (this is a command)
root@orangepi4-lts:~/mjpg-streamer/mjpg-streamer-experimental# ./mjpg_streamer -i
"./input_uvc.so -d /dev/video0 -u -f 30" -o "./output_http.so -w ./www"
```

- e. Then enter [IP address of the development board: 8080] in the browser of the Ubuntu PC or Windows PC or mobile phone on the same LAN as the development board to see the video output by the camera



- f. It is recommended to use mjpg-streamer to test the USB camera, which is much smoother than motion, and you will not feel any lag when using mjpg-streamer

3. 18. Audio Test

3. 18. 1. Headphone jack audio playback test

- 1) First insert headphones into the audio interface
- 2) Through the aplay -l command, you can view the sound card devices supported by the linux system, among which rockchipes8316c is the sound card device required for headphone playback

```

root@orangepi4-lts:~# aplay -l
**** List of PLAYBACK Hardware Devices ****
xcb_connection_has_error() returned true
card 0: rockchipes8316c [rockchip-es8316c], device 0: ff890000.i2s-ES8316 HiFi
ES8316 HiFi-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: rkhdmidpsound [rk-hdmi-dp-sound], device 0: HDMI-DP multicodec-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0

```

- 3) Then use the aplay command to play the audio, and the headset can hear the sound



```
root@orangepi4-lts:~# aplay -D hw:0,0 /usr/share/sounds/alsa/audio.wav  
Playing WAVE '/usr/share/sounds/alsa/audio.wav' : Signed 16 bit Little Endian, Rate  
44100 Hz, Stereo
```

3. 18. 2. Onboard MIC recording test

1) First make sure the system has opened the following recording channel

```
root@orangepi4-lts:~# amixer cset name='Differential Mux' lin2-rin2
```

2) The recording command is as follows

```
root@orangepi4-lts:~# arecord -D hw:0,0 -d 5 -f cd -t wav test.wav  
Recording WAVE 'test.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
```

3) After the recording is completed, a recording file named test.wav will be generated in the current path. Use the aplay command to play test.wav to check whether there is sound output. If there is sound output, the recording is normal.

```
root@orangepi4-lts:~# ls -lh  
total 862K  
-rw-r--r-- 1 root root 862K Feb  5 04:24 test.wav  
root@orangepi4-lts:~# aplay -D hw:0,0 test.wav
```

3. 18. 3. Headphone recording

1) First plug in the headphones with the recording function

2) Then use amixer to open the recording channel below

```
root@orangepi4-lts:~# amixer cset name='Differential Mux' lin1-rin1
```

3) The recording command is as follows

```
root@orangepi4-lts:~# arecord -D hw:0,0 -d 5 -f cd -t wav test.wav  
Recording WAVE 'test.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
```

4) After the recording is completed, a recording file named test.wav will be generated in the current path. Use the aplay command to play test.wav to check whether there is sound output. If there is sound output, the recording is normal.

```
root@orangepi4-lts:~# ls -lh  
total 862K
```



```
-rw-r--r-- 1 root root 862K Feb  5 04:24 test.wav
root@orangepi4-lts:~# aplay -D hw:0,0 test.wav
```

3. 18. 4. HDMI audio playback test

1) First connect the Orange Pi development board to the TV with an HDMI cable (other HDMI monitors need to ensure that they can play audio)

2) Upload the audio files to be played to the /root folder of the linux system. In Ubuntu PC, you can use the scp command to upload (the IP address in the command is the IP address of the Orange Pi development board), or use a U disk to copy

```
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

```
root@orangepi4-lts:~# aplay -D hw:1,0 audio.wav
```

3. 19. Temperature sensor

1) RK3399 has a total of 2 temperature sensors, the command to check the temperature is as follows

a. sensor0: CPU

```
root@orangepi4-lts:~# cat /sys/class/thermal/thermal_zone0/type
soc-thermal
root@orangepi4-lts:~# cat /sys/class/thermal/thermal_zone0/temp
48125
```

b. sensor1: GPU

```
root@orangepi4-lts:~# cat /sys/class/thermal/thermal_zone1/type
gpu-thermal
root@orangepi4-lts:~# cat /sys/class/thermal/thermal_zone1/temp
49375
```



3. 20. How to use Mini-PCIE

At present, the Mini-PCIE interface only supports the function of connecting the hard disk through the ASM1062 mini-PCIE to SATA module, and other functions are temporarily not supported.

1) For the method of hardware wiring, please refer to the instructions for connecting the mini-PCIE to the hard disk.

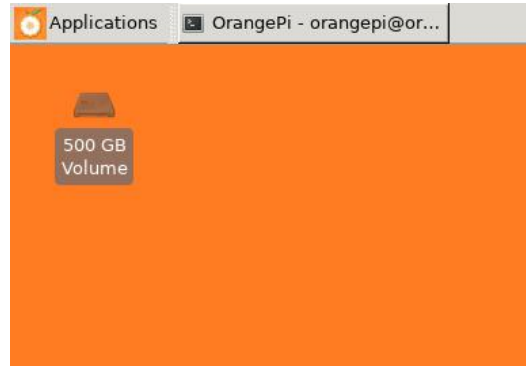
2) After completing the wiring, connect the DC power supply to the development board. After the system starts, execute the `lspci` command in the terminal to see the recognized pci device

```
root@orangepi4-lts:~# lspci
00:00.0 PCI bridge: Fuzhou Rockchip Electronics Co., Ltd RK3399 PCI Express Root Port
01:00.0 SATA controller: ASMedia Technology Inc. ASM1062 Serial ATA Controller (rev 02)
```

And the information about the hard disk can also be seen in the output information of the `dmesg` command

```
[ 14.535120] scsi 0:0:0:0: Direct-Access      ATA          WDC WD5000LPCX-2
1A01 PQ: 0 ANSI: 5
[ 14.536785] sd 0:0:0:0: [sda] 976773168 512-byte logical blocks: (500 GB/466 GiB)
[ 14.536808] sd 0:0:0:0: [sda] 4096-byte physical blocks
[ 14.537238] sd 0:0:0:0: [sda] Write Protect is off
[ 14.537262] sd 0:0:0:0: [sda] Mode Sense: 00 3a 00 00
[ 14.537411] sd 0:0:0:0: [sda] Write cache: enabled, read cache: enabled, doesn't
support DPO or FUA
[ 14.590267]  sda: sda1
[ 14.592221] sd 0:0:0:0: [sda] Attached SCSI disk
```

3) On the desktop of the Linux system, you can see that the hard disk has been mounted



3. 21. GPU test method

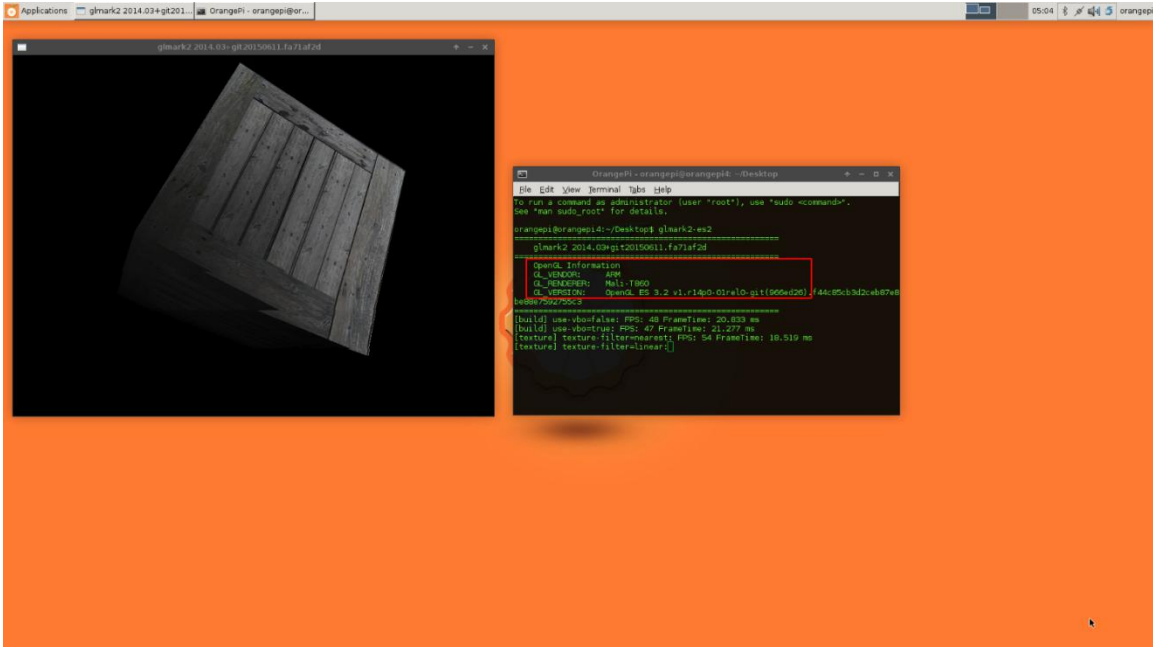
Note that only Debian10 and Debian11 desktop systems support this feature, and Ubuntu20.04 does not currently support this feature

1) the GPU driver has been adapted to Debian10 and Debian11. Gmark2-es2 can be used for testing. **Gmark2-es2 is preinstalled on Debian10 and Debian11 by default**

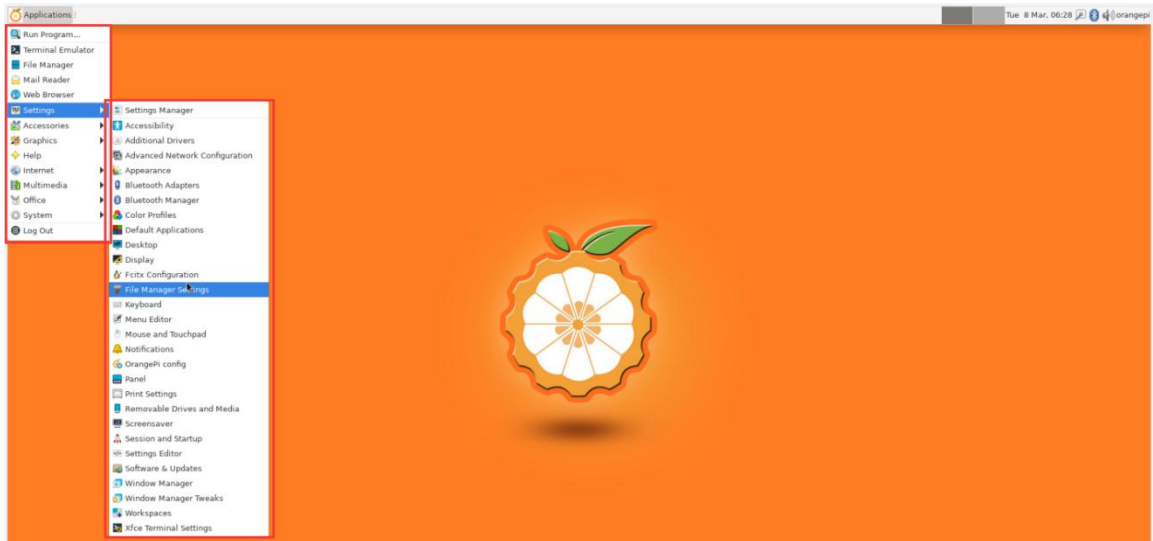
2) Run gmark2-es2 in the terminal of the Ubuntu desktop

```
root@orangepi4-lts:~# gmark2-es2
```

3) The effect of running gmark2-es2 is shown in the figure below. From the output log, you can see that the GPU used is Mali-T860



4) After Debian11 turns on GPU hardware acceleration, as shown in the figure below, when dragging with the mouse in the application list in the upper left corner, the speed will be slower than when GPU hardware acceleration is not turned on. This is a known issue



5) If you do not need GPU hardware acceleration, you can modify the following configuration file to turn off GPU hardware acceleration (after turning off GPU and VPU, both GPU and VPU cannot be used, including MPV hardware decoding and Chromium



hardware acceleration), and then restart the system, it will not work I have the above carton problem.

```
root@orangePi4-lts:~# cat /etc/X11/xorg.conf.d/02-modesetting.conf
Section "Device"
    Identifier "Rockchip Graphics"
    Driver      "modesetting"

### Use Rockchip RGA 2D HW accel
#   Option      "AccelMethod"    "exa"

### Use GPU HW accel
    Option      "AccelMethod"    "none"    #Set to none to turn off GPU
                                           acceleration
```

3. 22. MPV hardware decoding test

1) The desktop version system has integrated MPV player, the player can call the rkmpv decoding plug-in, the currently tested and supported decoding formats are H264, H265, VP9

2) The test video file can be downloaded by opening the link below

a. The video download link in H264 format is

```
https://test-videos.co.uk/vids/jellyfish/mp4/h264/1080/Jellyfish\_1080\_10s\_30MB.mp4  
http://bbb3d.renderfarming.net/download.html
```

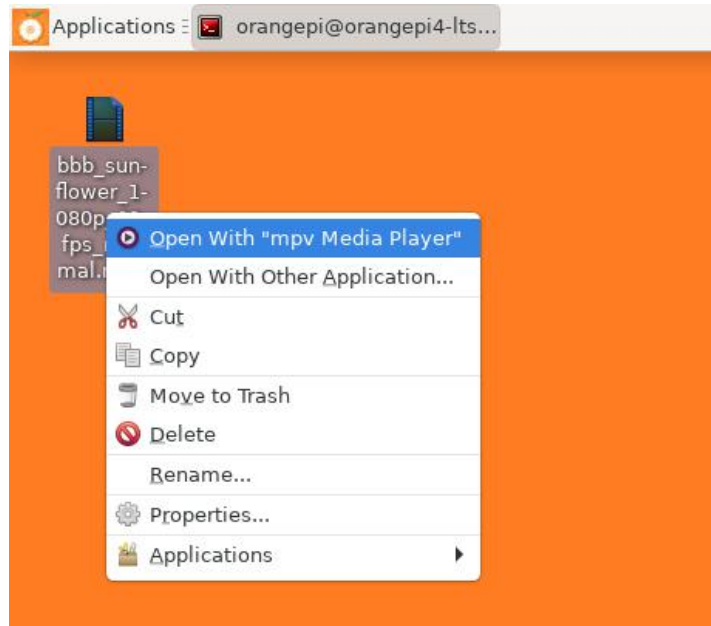
b. The video download link in H265 format is

```
https://test-videos.co.uk/vids/jellyfish/mp4/h265/1080/Jellyfish\_1080\_10s\_30MB.mp4
```

c. The video download link in VP9 format is

```
https://test-videos.co.uk/vids/jellyfish/webm/vp9/1080/Jellyfish\_1080\_10s\_30MB.webm
```

3) Copy the video file to the desktop, right click and open it with MPV player



4) The video will start to play, then open the terminal and enter the top command, you can see the CPU usage



5) Then enter the following command in the terminal to turn on the debugging switch of the VPU driver, and you can see the output information of the VPU driver

a. The command for Linux4.4 is as follows

```
root@orangepi4-lts:~# echo 0x100 > \
```



```
/sys/module/rk_vcodec/parameters/debug
```

b. The command for Linux5.10 is as follows

```
root@orangepi4-lts:~# echo 0x100 > \
```

```
/sys/module/rk_vcodec/parameters/mpp_dev_debug
```

```
root@orangepi4-lts:~# dmesg -c
root@orangepi4-lts:~# dmesg
[ 1229.295635] rk_vcodec: ff660000.rkvdec: pid: 4561, session: 00000000c9clef4e, time: 5262 us
[ 1229.309028] rk_vcodec: ff660000.rkvdec: pid: 4561, session: 00000000c9clef4e, time: 3210 us
[ 1229.320295] rk_vcodec: ff660000.rkvdec: pid: 4561, session: 00000000c9clef4e, time: 5366 us
[ 1229.326177] rk_vcodec: ff660000.rkvdec: pid: 4561, session: 00000000c9clef4e, time: 3438 us
[ 1229.332428] rk_vcodec: ff660000.rkvdec: pid: 4561, session: 00000000c9clef4e, time: 3708 us
[ 1229.342385] rk_vcodec: ff660000.rkvdec: pid: 4561, session: 00000000c9clef4e, time: 7921 us
[ 1229.354728] rk_vcodec: ff660000.rkvdec: pid: 4561, session: 00000000c9clef4e, time: 6564 us
[ 1229.364151] rk_vcodec: ff660000.rkvdec: pid: 4561, session: 00000000c9clef4e, time: 7135 us
[ 1229.375270] rk_vcodec: ff660000.rkvdec: pid: 4561, session: 00000000c9clef4e, time: 7242 us
```

6) Use the left and right arrow keys to control the video playback progress

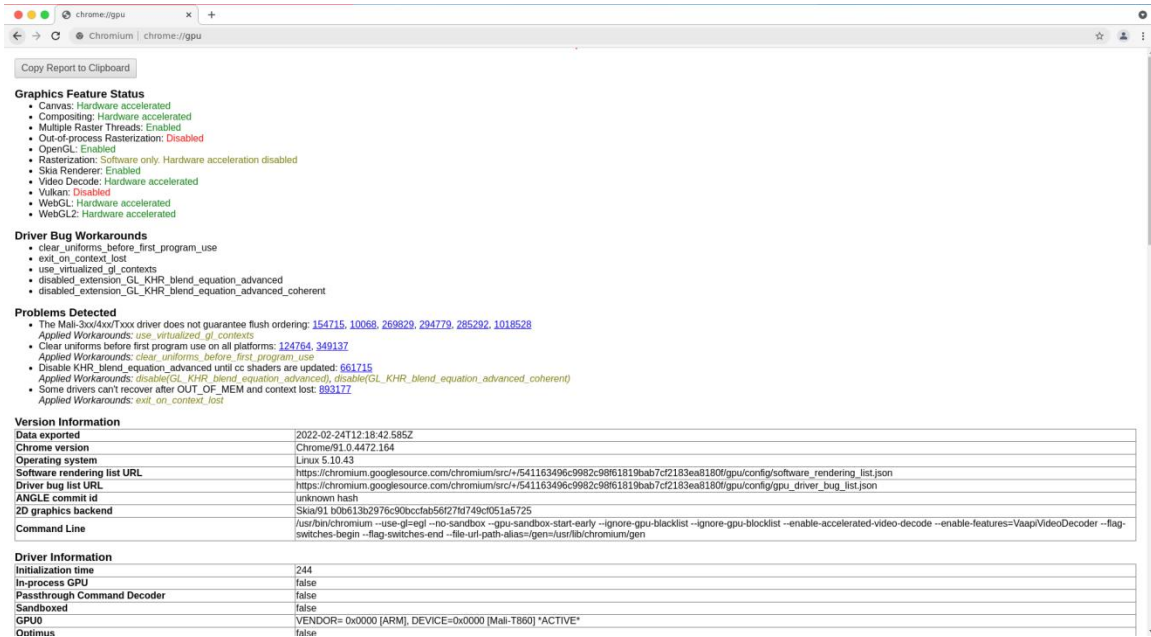
3. 23. Chromium hardware acceleration test

Note that the Ubuntu system does not support this function. Currently, the Debian11 desktop version system supports WebGL hardware acceleration and video decoding hardware acceleration. Debian10 only supports WebGL hardware acceleration, not video decoding hardware acceleration. This needs to be noted.

1) First open the chromium browser,



2) Then enter `chrome://gpu` in the address bar to view the hardware acceleration



3) Use chromium to play local video or play video online

4) Then enter the following command in the terminal to turn on the debugging switch of the VPU driver, and you can see the printout of the VPU driver

a. The command for Linux4.4 is as follows

```
root@orangepi4-lts:~# echo 0x100 > \
/sys/module/rk_vcodec/parameters/debug
```

b. The command for Linux5.10 is as follows

```
root@orangepi4-lts:~# echo 0x100 > \
/sys/module/rk_vcodec/parameters/mpp_dev_debug
```

```
root@orangepi4-lts:~# dmesg -c
root@orangepi4-lts:~# dmesg
[ 1229.295635] rk_vcodec: ff660000.rkvdec: pid: 4561, session: 00000000c9c1ef4e, time: 5262 us
[ 1229.309028] rk_vcodec: ff660000.rkvdec: pid: 4561, session: 00000000c9c1ef4e, time: 3210 us
[ 1229.320295] rk_vcodec: ff660000.rkvdec: pid: 4561, session: 00000000c9c1ef4e, time: 5366 us
[ 1229.326177] rk_vcodec: ff660000.rkvdec: pid: 4561, session: 00000000c9c1ef4e, time: 3438 us
[ 1229.332428] rk_vcodec: ff660000.rkvdec: pid: 4561, session: 00000000c9c1ef4e, time: 3708 us
[ 1229.342385] rk_vcodec: ff660000.rkvdec: pid: 4561, session: 00000000c9c1ef4e, time: 7921 us
[ 1229.354728] rk_vcodec: ff660000.rkvdec: pid: 4561, session: 00000000c9c1ef4e, time: 6564 us
[ 1229.364151] rk_vcodec: ff660000.rkvdec: pid: 4561, session: 00000000c9c1ef4e, time: 7135 us
[ 1229.375270] rk_vcodec: ff660000.rkvdec: pid: 4561, session: 00000000c9c1ef4e, time: 7242 us
```

3. 24. How to install Docker

1) First install the following packages



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

2) Add the key of Alibaba Cloud docker

a. Ubuntu OS

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

b. Debian OS

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

3) Add the corresponding docker source to the system source

a. Ubuntu OS

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

b. Debian OS

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

4) Install the latest version of docker-ce

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

5) 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
```



6) Test docker

```

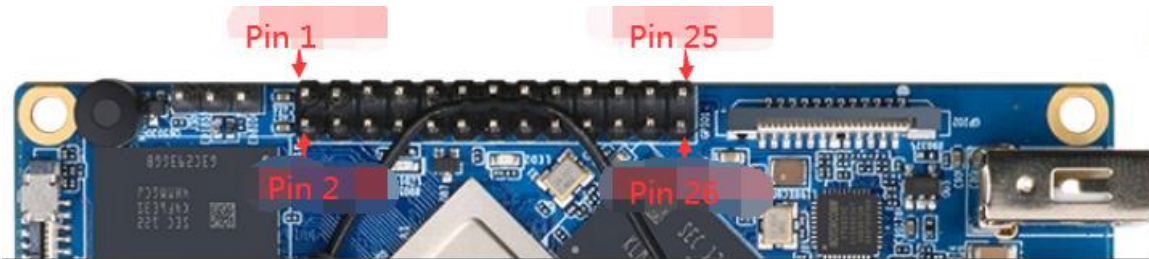
root@orangepi:~# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
256ab8fe8778: Pull complete
Digest:
sha256:7f0a9f93b4aa3022c3a4c147a449ef11e0941a1fd0bf4a8e6c9408b2600777c5
Status: Downloaded newer image for hello-world:latest

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

```

3. 25. 26pin interface pin description

1) Please refer to the figure below for the order of the 26 pin interface pins of the Orange Pi 4 LTS development board



2) The functions of the 40 pin interface pins of the Orange Pi 4 LTS development board are shown in the table below

GPIO	GPIO	Function	Pin
		3.3V	1
52	GPIO1_C4	I2C8_SDA	3
53	GPIO1_C5	I2C8_SCL	5
150	GPIO4_C6	PWM1	7
		GND	9
33	GPIO1_A1	GPIO1_A1	11
35	GPIO1_A3	GPIO1_A3	13

Pin	Function	GPIO	GPIO
2	5V		
4	5V		
6	GND		
8	I2C3_SCL	GPIO4_C1	145
10	I2C3_SDA	GPIO4_C0	144
12	GPIO1_C2	GPIO1_C2	50
14	GND		



Use gpio readall to see that the mode of pin 11 has changed to out

```
root@orangepi4-lts:~# gpio readall
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| GPIO | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | GPIO |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 64 | 0 | 3.3V | | | 1 | 2 | | | 5V | | |
| I2C8_SDA | ALT2 | 1 | 3 | 4 | | | 5V | | |
| 65 | 1 | I2C8_SCL | ALT2 | 1 | 5 | 6 | | | GND | | |
| 150 | 2 | PWM1 | IN | 0 | 7 | 8 | 1 | ALT2 | I2C3_SCL | 3 | 145 |
| GND | | | | | 9 | 10 | 1 | ALT2 | I2C3_SDA | 4 | 144 |
| 33 | 5 | GPIO1_A1 | OUT | 0 | 11 | 12 | 1 | IN | GPIO1_C2 | 6 | 50 |
| 35 | 7 | GPIO1_A3 | OUT | 1 | 13 | 14 | | | GND | | |
| 92 | 8 | GPIO2_D4 | IN | 0 | 15 | 16 | 0 | IN | GPIO1_C6 | 9 | 54 |
| 3.3V | | | | | 17 | 18 | 0 | IN | GPIO1_C7 | 10 | 55 |
```

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

```
root@orangepi4-lts:~/wiringOP# gpio write 5 0
```

Use gpio readall to see that the value of pin 11 (V) has become 0

```
root@orangepi4-lts:~# gpio readall
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| GPIO | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | GPIO |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 64 | 0 | 3.3V | | | 1 | 2 | | | 5V | | |
| I2C8_SDA | ALT2 | 1 | 3 | 4 | | | 5V | | |
| 65 | 1 | I2C8_SCL | ALT2 | 1 | 5 | 6 | | | GND | | |
| 150 | 2 | PWM1 | IN | 0 | 7 | 8 | 1 | ALT2 | I2C3_SCL | 3 | 145 |
| GND | | | | | 9 | 10 | 1 | ALT2 | I2C3_SDA | 4 | 144 |
| 33 | 5 | GPIO1_A1 | OUT | 0 | 11 | 12 | 1 | IN | GPIO1_C2 | 6 | 50 |
| 35 | 7 | GPIO1_A3 | OUT | 1 | 13 | 14 | | | GND | | |
| 92 | 8 | GPIO2_D4 | IN | 0 | 15 | 16 | 0 | IN | GPIO1_C6 | 9 | 54 |
| 3.3V | | | | | 17 | 18 | 0 | IN | GPIO1_C7 | 10 | 55 |
```

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

```
root@orangepi4-lts:~/wiringOP# gpio write 5 1
```

Use gpio readall to see that the value (V) of pin 11 has changed to 1



```

root@orangepi4-lts:~/wiringOP/examples# ./spidev_test -v -D /dev/spidev1.0
spi mode: 0x0
bits per word: 8
max speed: 500000 Hz (500 KHz)
TX | FF 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 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 FF | .....

```

5) Then short-circuit the two pins of SPI1's txd (pin 19 in the 40pin interface) and rxd (pin 21 in the 40pin interface) and then run the output of spidev_test as follows, you can see the sent and received the same data

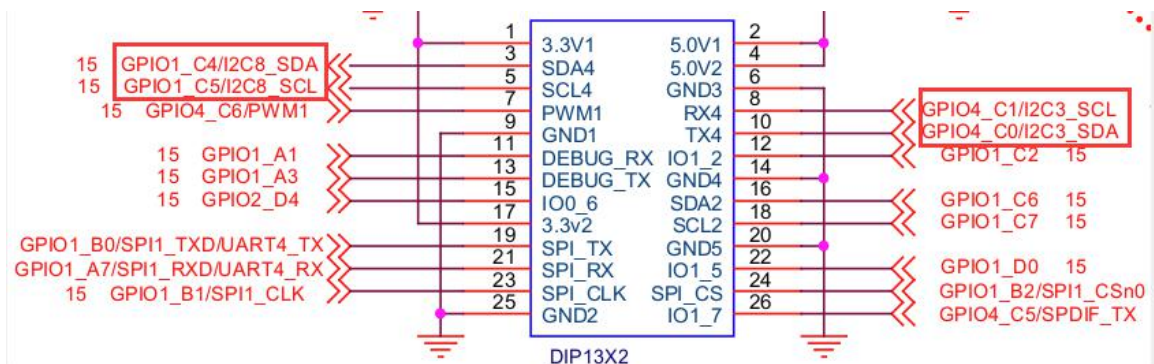
```

root@orangepi4-lts:~/wiringOP/examples# ./spidev_test -v -D /dev/spidev1.0
spi mode: 0x0
bits per word: 8
max speed: 500000 Hz (500 KHz)
TX | FF 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 FF F0 0D | .....@.....
RX | FF 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 FF F0 0D | .....@.....

```

3. 27. 3. 26pin I2C test

1) It can be seen from the schematic diagram of 26pin that the available i2c for the development board are i2c3 and i2c8



2) After the system starts, you can see the following multiple i2c device nodes under /dev

- a. i2c3 in 26pin corresponds to /dev/i2c-3



b. i2c8 in 26pin corresponds to /dev/i2c-8

```
root@orangepi4-lts:~# ls /dev/i2c*
/dev/i2c-0  /dev/i2c-1  /dev/i2c-2  /dev/i2c-3  /dev/i2c-4  /dev/i2c-7  /dev/i2c-8
/dev/i2c-9
```

3) Then start testing i2c, first install i2c-tools

```
root@orangepi4-lts:~# apt update
root@orangepi4-lts:~# apt -y install i2c-tools
```

4) Then connect an i2c device to the i2c8 pin of the 26pin connector (the i2c3 test is the same as the i2c8, just connect the device to the i2c3 pin, the following takes i2c8 as an example)

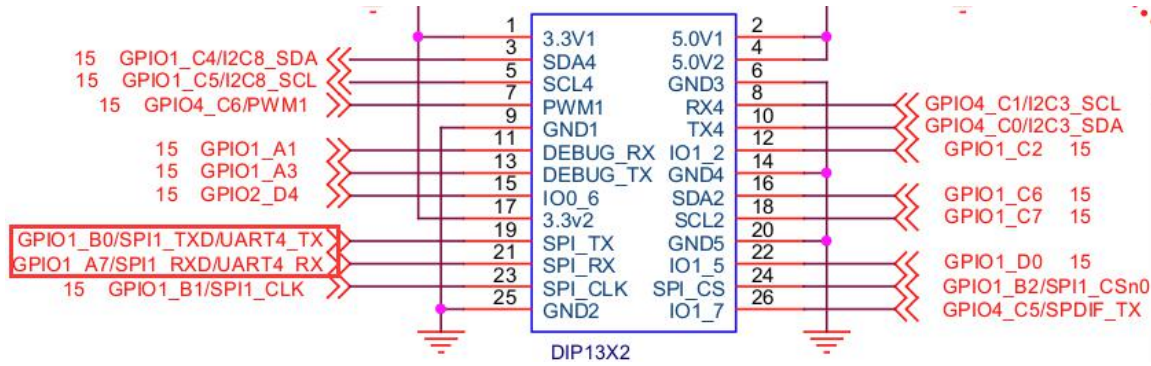
	i2c8	i2c3
sda pin	Corresponds to pin 3	Corresponds to pin 10
sck pin	Corresponds to pin 5	Corresponds to pin 8
vcc pin	Corresponds to pin 1	Corresponds to pin 1
gnd pin	Corresponds to pin 6	Corresponds to pin 6

5) Then use the `i2cdetect -y 8` command. If the address of the connected i2c device can be detected, it means that the i2c can be used normally

```
root@orangepi4-lts:~# i2cdetect -y 8
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
10:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30:  -- -- -- -- -- -- -- 38 -- -- -- -- -- -- --
40:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

3. 27. 4. 26pin UART test

1) It can be seen from the schematic diagram of 26pin that the uart available for the development board is uart4



2) The SPI and UART4 of OrangePi 4 LTS multiplex the same pins. In dts, UART4 is turned off and SPI1 is turned on by default. In Linux system, DT overlay can be used to open the configuration of uart4

a. For Linux4.4 system, the method is as follows

- a) A script named **orangepi-add-overlay** is pre-installed in the linux system. Through this script, we can use DT overlay to dynamically modify the configuration in dts. First write the rockchip-uart4.dts file, the content is as follows

```

/dts-v1/;
/plugin/;

/ {
    compatible = "rockchip,rk3399";

    fragment@0 {
        target = &spi1;
        __overlay__ {
            status = "disabled";
        };
    };

    fragment@1 {
        target = &uart4;
        __overlay__ {
            status = "okay";
        };
    };
};
    
```



```
};
```

- b) Then you can use **orangepi-add-overlay** to compile rockchip-uart4.dts into rockchip-uart4.dtbo

```
root@orangepi:~# orangepi-add-overlay rockchip-uart4.dts
```

```
Compiling the overlay
```

```
Copying the compiled overlay file to /boot/overlay-user/
```

```
Reboot is required to apply the changes
```

- c) Then restart the Linux system, you can see the following printing information in the log output from the serial port, indicating that rockchip-uart4.dtbo is loaded successfully

```
384 bytes read in 6 ms (62.5 KiB/s)
```

```
Applying user provided DT overlay rockchip-uart4.dtbo
```

- d) After entering the Linux system, you can see the device node ttyS4 under /dev

```
root@orangepi4-lts:~# ls /dev/ttyS*
```

```
/dev/ttyS0 /dev/ttyS4
```

- b. For Linux5.10 system, the method is as follows

- a) First set the overlays variable in /boot/orangepiEnv.txt to open the uart4 configuration

```
root@orangepi4-lts:~# cat /boot/orangepiEnv.txt
```

```
verbosity=1
```

```
bootlogo=true
```

```
overlay_prefix=rockchip
```

```
fdtfile=rockchip/rk3399-orangepi-4-lts.dtb
```

```
rootdev=UUID=c51e6614-42cf-473c-9134-46a72667eb9c
```

```
rootfstype=ext4
```

```
overlays=uart4
```

- b) Then restart the system. When starting, you can see the following print information in the startup log of u-boot, indicating that the configuration of uart4 is loaded successfully

```
Applying kernel provided DT overlay rockchip-uart4.dtbo
```

```
2698 bytes read in 8 ms (329.1 KiB/s)
```

- c) After entering the Linux system, you can see the device node ttyS4 under /dev



```
root@orangepi4-lts:~# ls /dev/ttyS*
/dev/ttyS2  /dev/ttyS4
```

3) Then start to test the uart interface, first use the DuPont line to short-circuit the rx and tx of the uart4 interface to be tested

	Uart4
tx pin	Corresponds to pin 19
rx pin	Corresponds to pin 21

4) Then modify the serial device node name opened by the serial test program serialTest in wiringOP to **/dev/ttyS4**

```
root@orangepi4-lts:~/wiringOP/examples# vim serialTest.c
```

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

    if ((fd = serialOpen ("/dev/ttyS4", 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@orangepi4-lts:~/wiringOP/examples# make serialTest
[CC] serialTest.c
[link]
root@orangepi4-lts:~/wiringOP/examples#
```

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

```
root@orangepi4-lts:~/wiringOP/examples# ./serialTest
Out:  0:  ->  0
Out:  1:  ->  1
Out:  2:  ->  2
Out:  3:  ->  3
Out:  4:  ->  4
```



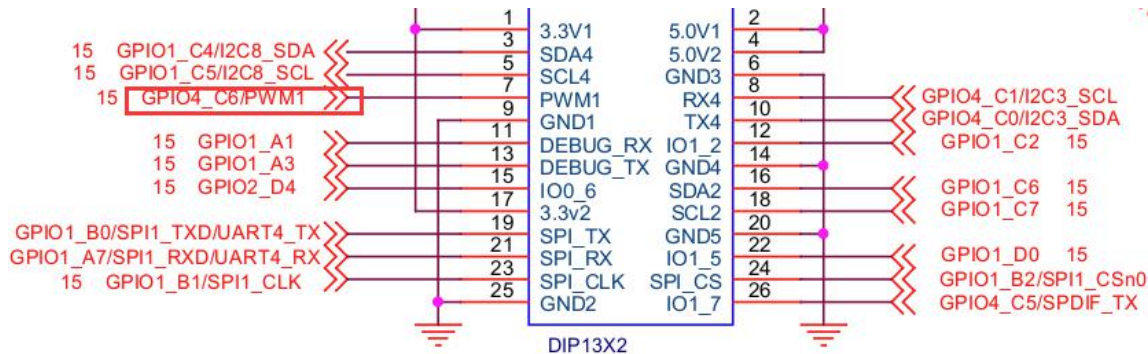
```

Out: 5: -> 5
Out: 6: -> 6
Out: 7: -> 7
Out: 8: -> 8^C

```

3.27.5. 26pin PWM test

1) The 7th pin of 26pin is PWM1, the official image has turned on PWM1 by default, and PWM1 can be used without other configuration



2) After the pwm driver is loaded successfully, the pwmchip1 directory will be generated under /sys/class/pwm/, write 0 to the export file, the pwm timer will be opened, and a pwm0 directory will be generated. On the contrary, writing 0 to the unexport file will turn off the pwm timer, and the pwm0 directory will be deleted. This directory has the following files:

```

enable :    pwmWrite 1 to enable pwm, write 0 to disable pwm
polarity :  There are two parameter options, normal and inversed, indicating that the
output pin level is inverted.
duty_cycle : The unit is nanoseconds. In normal mode, it means the duration of high level.
In inversed mode, it means the duration of low level.
period :    The unit is nanoseconds, indicating the duration of the pwm wave

```

3) Example of use: let pwm1 output a square wave with a duty cycle of 50% and a period of 50 microseconds

```

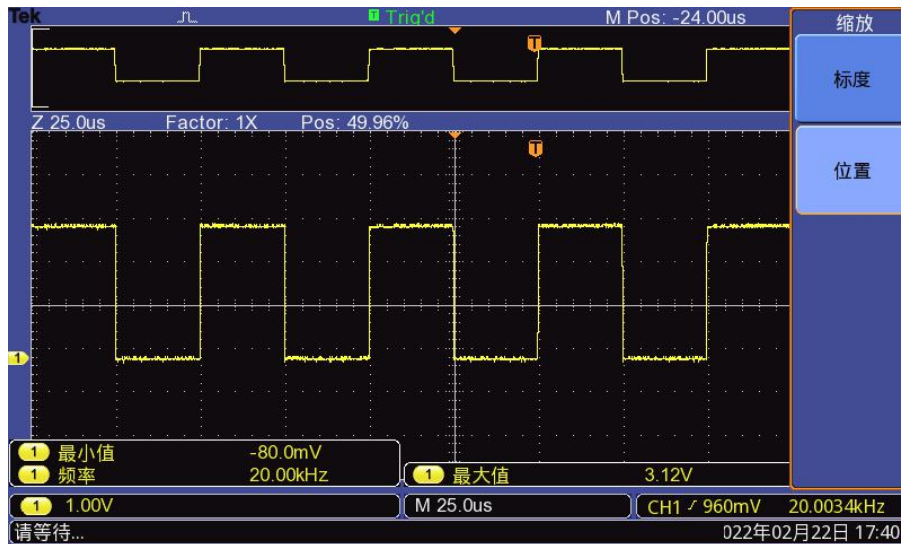
root@orangepi4-lts:~#cd /sys/class/pwm/pwmchip1
root@orangepi4-lts:/sys/class/pwm/pwmchip1#echo 0 > export
root@orangepi4-lts:/sys/class/pwm/pwmchip1#
echo 50000 > pwm0/period

```



```
root@orangepi4-lts:/sys/class/pwm/pwmchip1#  
echo 25000 > pwm0/duty_cycle  
root@orangepi4-lts:/sys/class/pwm/pwmchip1#  
echo 1 > pwm0/enable
```

4) On the oscilloscope, you can see that pwm1 outputs the following waveforms



3. 28. How to use 0.96-inch OLED module with I2C interface

1) The 0.96-inch OLED module of Orange Pi is shown in the figure below, and its 7-bit i2c slave address is 0x3c

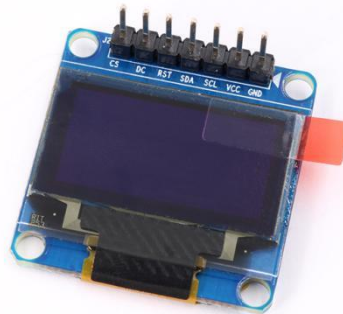


2) First, connect the 0.96-inch OLED module to the 26pin interface of the Orange Pi



development board through the DuPont cable. The wiring method is as follows (the following takes i2c8 as an example, i2c3 only needs to change the scl to pin 8, and the sda to the 8th pin. to pin 10)

OLED module pins	description	Development board 26pin interface i2c8 pin	Development board 26pin interface i2c3 pin
GND	power ground	pin 6	pin 6
VCC	5V	pin 4	pin 4
SCL	I2C clock line	pin 5	pin 8
SDA	I2C data line	pin 3	pin 10
RST	Connect to 3.3V	pin 1	pin 1
DC	Connect to GND	pin 9	pin 9
CS	Connect to GND	pin 25	pin 25



3) After connecting the OLED module to the development board, first use the i2c-tools tool to check whether the address of the OLED module can be scanned

```
root@orangepi4-lts:~# apt update
root@orangepi4-lts:~# apt install i2c-tools
root@orangepi4-lts:~# i2cdetect -y 8
```



```

root@orangepi4-lts:~# i2cdetect -y 8
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  3c  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --

```

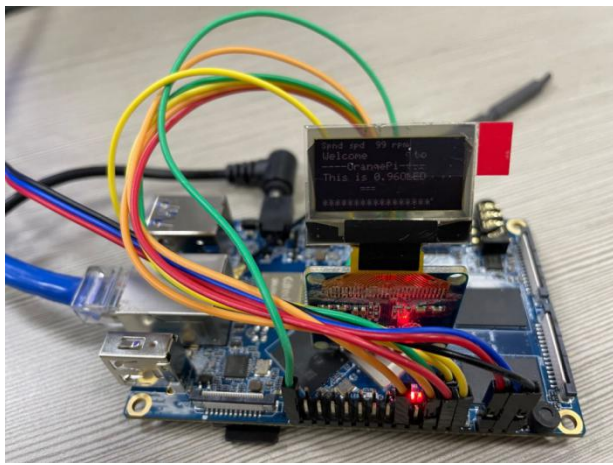
4) Then you can use the oled_demo in wiringOP to test the OLED module. The test steps are as follows

```

root@orangepi4-lts:~# git clone https://github.com/orangepi-xunlong/wiringOP
root@orangepi4-lts:~# cd wiringOP
root@orangepi4-lts:~/wiringOP# ./build clean && ./build
root@orangepi4-lts:~/wiringOP# cd examples
root@orangepi4-lts:~/wiringOP/examples# make oled_demo
root@orangepi4-lts:~/wiringOP/examples# ./oled_demo /dev/i2c-8
-----start-----
-----end-----

```

5) After running oled_demo, you can see the following output on the OLED screen





3. 29. Hardware watchdog test

1) Download the code of wiringOP

```
root@orangePi4-lts:~# apt update
root@orangePi4-lts:~# apt -y install git
root@orangePi4-lts:~# git clone https://github.com/orangepi-xunlong/wiringOP
```

2) Compile the watchdog test program

```
root@orangePi4-lts:~# cd wiringOP/examples/
root@orangePi4-lts:~/wiringOP/examples# gcc watchdog.c -o watchdog
```

3) Run the watchdog test program

- a. The second parameter 10 represents the counting time of the watchdog. If the dog is not fed within this time, the system 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 was fed successfully

```
root@orangePi4-lts:~/wiringOP/examples# ./watchdog 10
open success
options is 33152,identity is Synopsys DesignWare Watchdog
put_usr return,if 0,success:0
The old reset time is: 21
return ENOTTY,if -1,success:-1
return ENOTTY,if -1,success:-1
put_user return,if 0,success:0
put_usr return,if 0,success:0
keep alive
keep alive
```

3. 30. How to program linux image to eMMC

Note that only the Orange Pi 4 LTS development board with the eMMC chip model can burn the image into the eMMC. If the Orange Pi 4 LTS development board without the eMMC chip is purchased, the image cannot be burned into the eMMC.

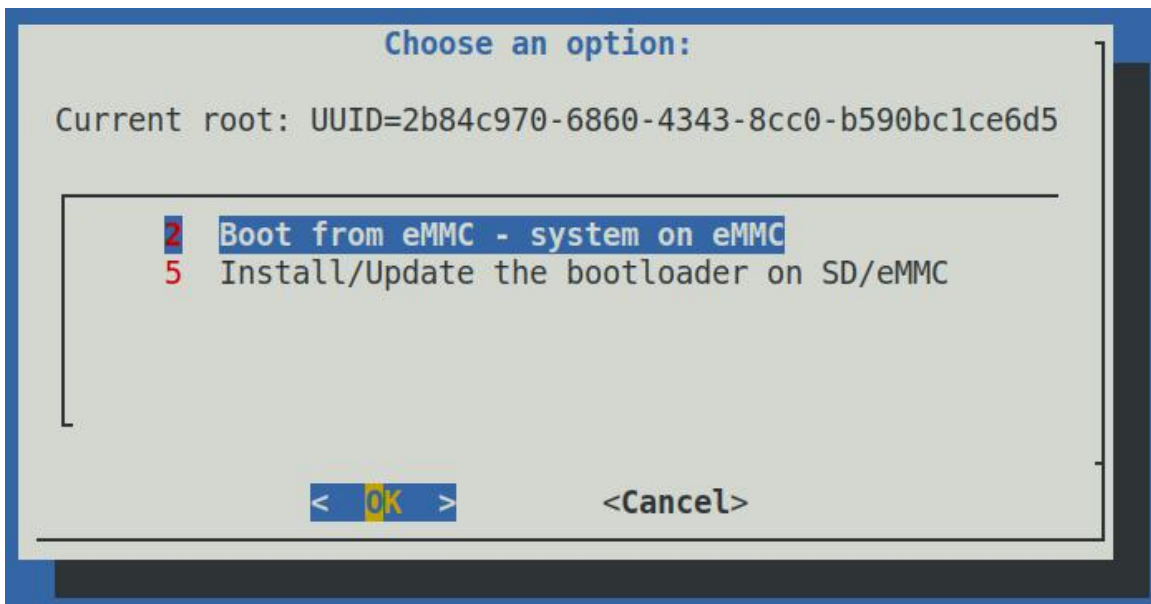


1) Burning the linux image to eMMC needs to be done with the help of a TF card, first burn the linux image to the TF card, and then start the development board to enter the linux system

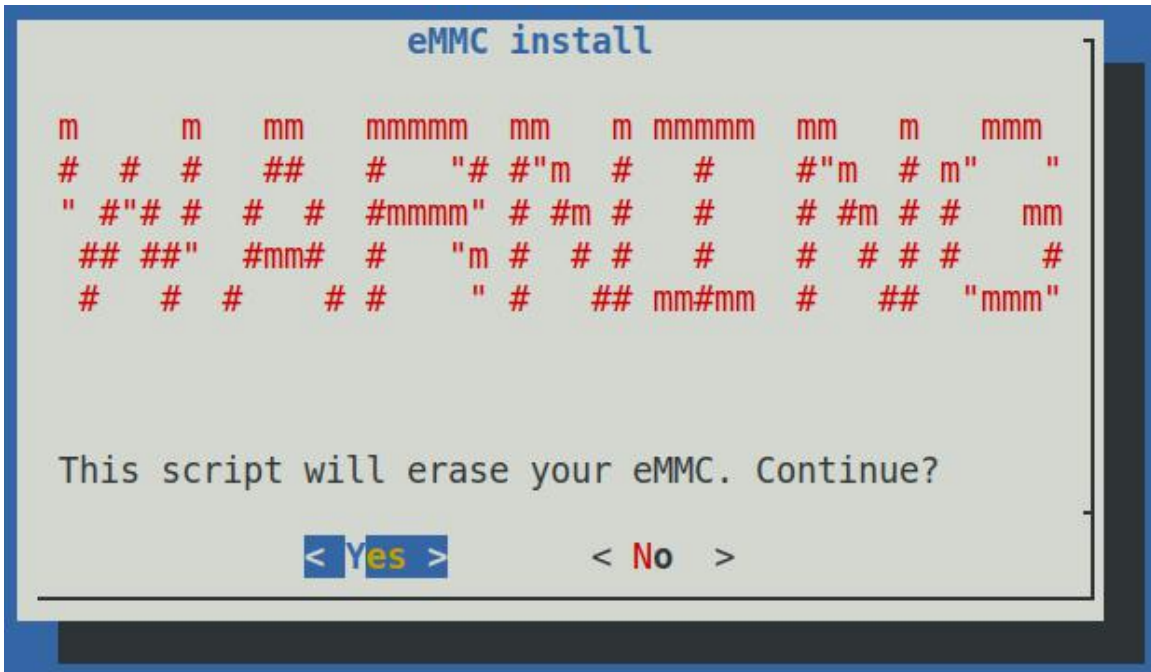
2) Then run the nand-sata-install script

```
root@orangePi4-lts:~# nand-sata-install
```

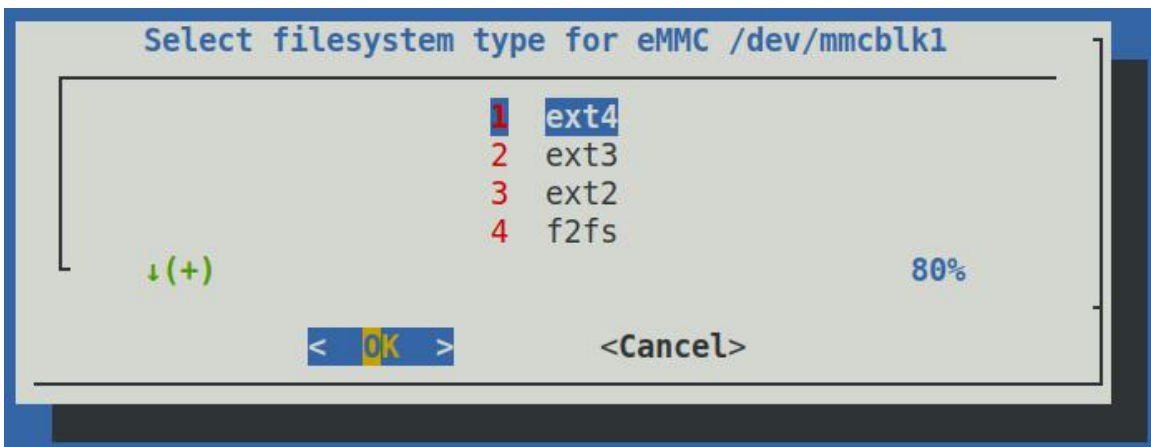
3) then select **2 Boot from eMMC - system on eMMC**

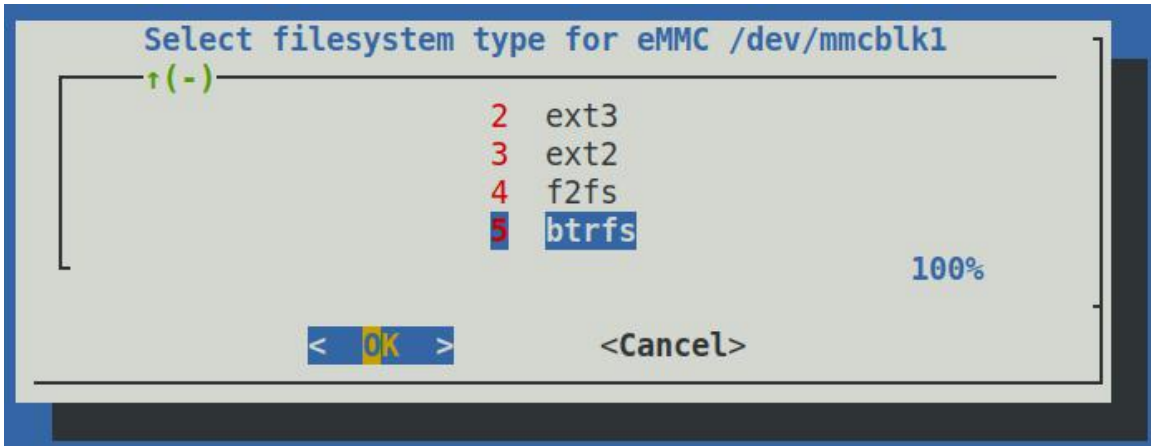


4) Then a warning will pop up, the script will erase all data on the eMMC, select **<Yes>** to continue

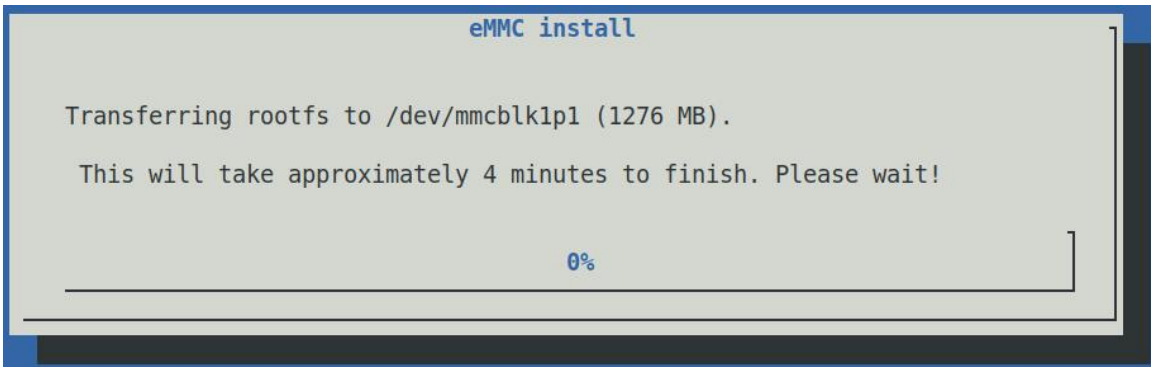


5) Then you will be prompted to select the type of file system, which supports ext2/3/4, f2fs and btrfs five file systems

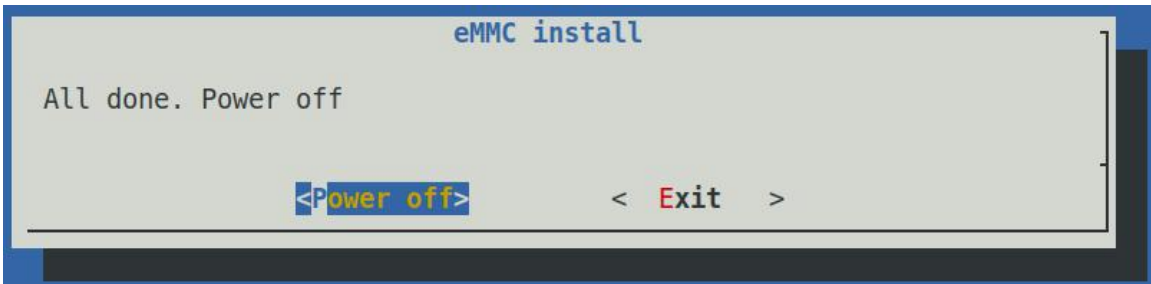




6) Then it will start to format eMMC. After formatting eMMC, it will start to burn the linux image into eMMC.



7) After burning, the following options will be prompted, you can select <Power off> to directly shut down



8) Then pull out the TF card, and then power on again, it will start the linux system in eMMC

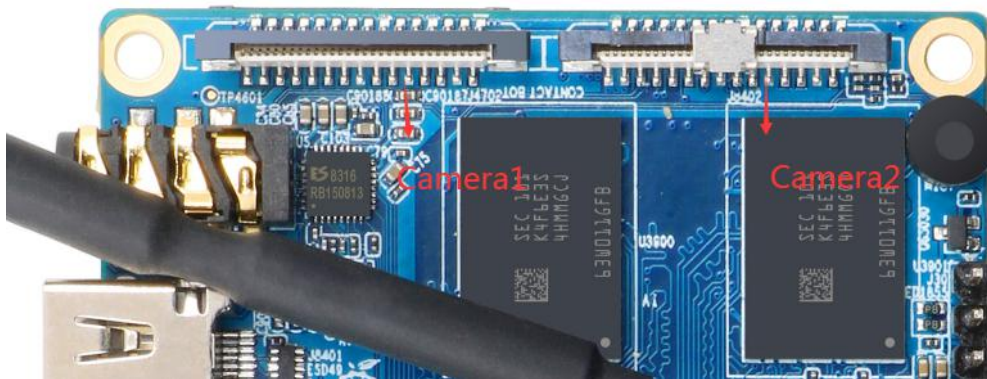


3. 31. How to use Linux 4.4 OV13850 camera

Note that only Debian10 supports this function, other systems do not support it temporarily

3. 31. 1. Camera connection instructions

1) OrangePi 4 LTS has two Camera ports, both ports only support OV13850 camera by default. The two Camera interfaces can use one of the interfaces alone, or you can use the two Camera interfaces to connect two cameras at the same time



2) The OV13850 camera kit includes an OV13850 camera, an adapter board and a cable

Orange Pi RK3399



3) First insert the OV13850 camera into the adapter board, and then insert the cable into another card slot of the adapter board

Orange Pi RK3399





4) Then insert the other end of the cable into the Camera camera interface of the development board. The interface can be connected to two cameras at the same time, or one camera can be connected separately. After connecting the camera, start the Linux system (do not plug in the camera after powering on)



5) After starting the system, execute the following commands. If the following information appears, the camera is working normally. If there is no such information, please check whether the camera is connected correctly.

```
root@orangepi4-lts:~# dmesg | grep Async
[ 1.623685] rkisp1: Async subdev notifier completed
```

3. 31. 2. Using a single camera

1) The system calls the camera through the test_camera-gst.sh script, the parameters are as follows

Parameter	Function
--index or -i	Select the serial number of the camera to be used. The optional values are 0 and 1. When connecting two cameras at the same time, specify 1 to operate the second camera.
--action or -a	Specifies the action to be executed by the command. The optional



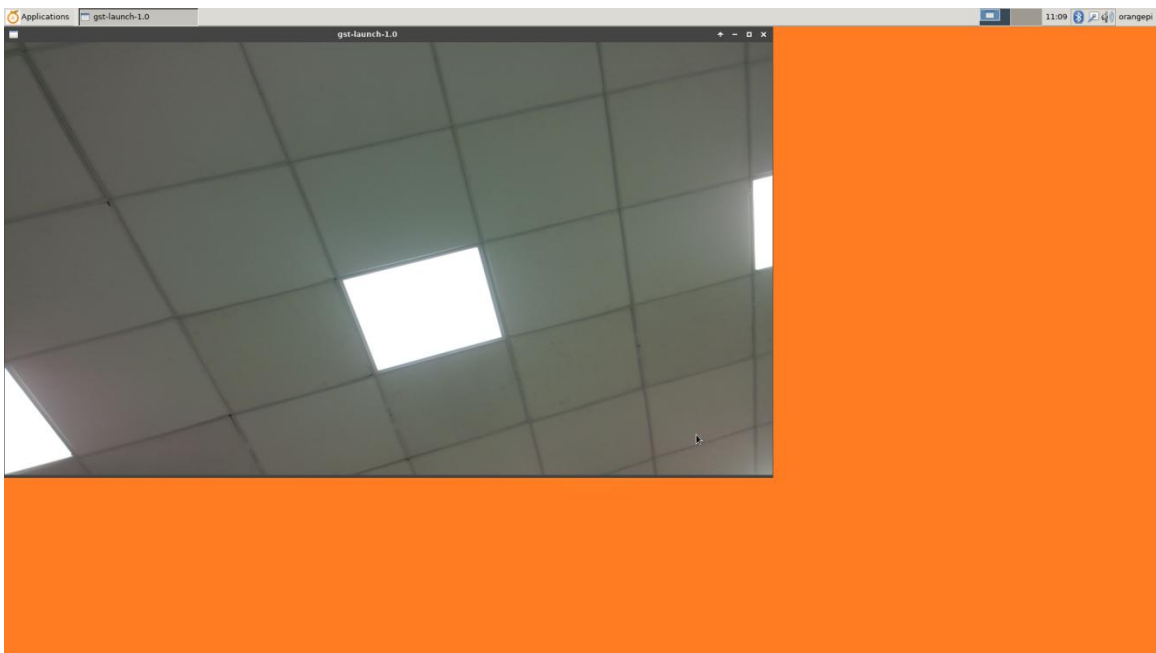
	parameters are: preview, photo and video, corresponding to preview, photo and video respectively
--output or -o	Specify the output file name, which is used to specify the output file name when taking pictures and videos
--verbose or -v	When specified as yes, the full command line that invokes the <code>gst-launch-1.0</code> command will be output

2) Then the way to preview the image is as follows

a. First run the `test_camera-gst.sh` script

```
root@orangepi4-lts:~# test_camera-gst.sh
Setting pipeline to PAUSED ...
media get entity by name: lens is null
Pipeline is live and does not need PREROLL ...
```

b. The effect is as shown in the figure below, a real-time camera window will be opened on the desktop



c. If no parameters are specified for this command, the default action is to preview, which will open the camera recognized by the system. If two OV13850 cameras are connected, camera1 (the camera interface near the headphone holder) will be opened first.

d. Take a picture with the camera, take a picture with the command, and save it as



the file test.jpg

```
root@orangePi4-lts-lts:~# test_camera-gst.sh -a photo -o test.jpg
gst-launch-1.0: no process found
Setting pipeline to PAUSED ...
media get entity by name: lens is null
Pipeline is live and does not need PREROLL ...

root@orangePi4-lts:~# ls
test.jpg
```

3) Use the camera to record, after running the script, it will start to record a 17s video.

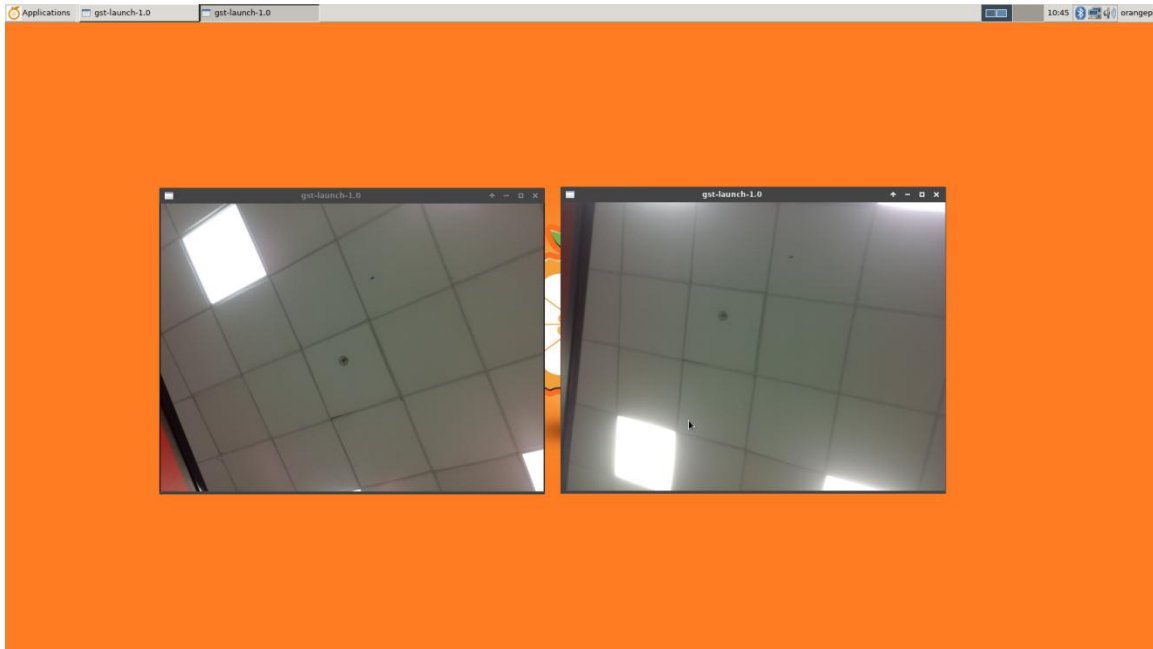
```
root@orangePi4-lts-lts:~# test_camera-gst.sh --action video --output test.ts
gst-launch-1.0: no process found
Setting pipeline to PAUSED ...
mpi: mpp version: Without VCS info
mpp_rt: NOT found ion allocator
...
root@orangePi4-lts:~# ls
test.ts
```

3.31.3. Dual camera usage

1) Run test_camera-dual.sh to open the dual camera

```
root@orangePi4-lts:~# test_camera-dual.sh
Start MIPI CSI Camera Preview!
Setting pipeline to PAUSED ...
Setting pipeline to PAUSED ...
```

2) The effect is as shown in the figure below, the real-time windows of the two cameras will be opened on the desktop respectively



3. 32. 10.1 inch MIPI screen usage

The Linux image on the official website supports two video outputs by default, namely HDMI video output and TypeC-DP video output. MIPI LCD output is disabled by default. Orange Pi 4 LTS has two MIPI interfaces, and MIPI LCD1 only has the function of video output. MIPI LCD2 and Camera 2 are multiplexed

3. 32. 1. LCD1 interface instructions

1) In the dts of the Linux system, LCD1 is turned off by default. First, you need to open the configuration of LCD1 in the dts.

a. For Linux 4.4 kernel, the method is as follows

a) A script named `orange-pi-add-overlay` is pre-installed in the linux system. Through this script, we can use DT overlay to dynamically modify the configuration in dts. First write the `rockchip-lcd1.dts` file, the content is as follows

```
root@orange-pi4-lts:~# cat rockchip-lcd1.dts
/dts-v1/;
/plugin/;
```



```

/ {
    compatible = "rockchip,rk3399";

    fragment@0 {
        target = <&dsi>;
        __overlay__ {
            status = "okay";
        };
    };

    fragment@1 {
        target = <&gt9xx>;
        __overlay__ {
            status = "okay";
        };
    };
};

```

- b) Then you can use `orangepi-add-overlay` to compile `rockchip-lcd1.dts` into `rockchip-lcd1.dtbo`

```

root@orangepi:~# orangepi-add-overlay rockchip-lcd1.dts
Compiling the overlay
Copying the compiled overlay file to /boot/overlay-user/
Reboot is required to apply the changes

```

- c) Then restart the Linux system, you can see the following printing information in the log output from the serial port, indicating that `rockchip-lcd1.dtbo` is loaded successfully

```

379 bytes read in 6 ms (61.5 KiB/s)
Applying user provided DT overlay rockchip-lcd1.dtbo

```

- d) After entering the Linux system, enter the following command to see the relevant information of the kernel output

```

root@orangepi4-lts:~# dmesg |grep mipi
[ 1.478148] dw-mipi-dsi ff960000.dsi: final DSI-Link bandwidth: 444 x 4 Mbps

```

- b. For Linux 5.10 kernel, the method is as follows
- a) First set the `overlays` variable in `/boot/orangepiEnv.txt` to open the LCD1



interface configuration

```
root@orangepi4-lts:~# cat /boot/orangepiEnv.txt
verbosity=1
bootlogo=true
overlay_prefix=rockchip
fdtfile=rockchip/rk3399-orangepi-4-lts.dtb
rootdev=UUID=c51e6614-42cf-473c-9134-46a72667eb9c
rootfstype=ext4
overlays=lcd1
```

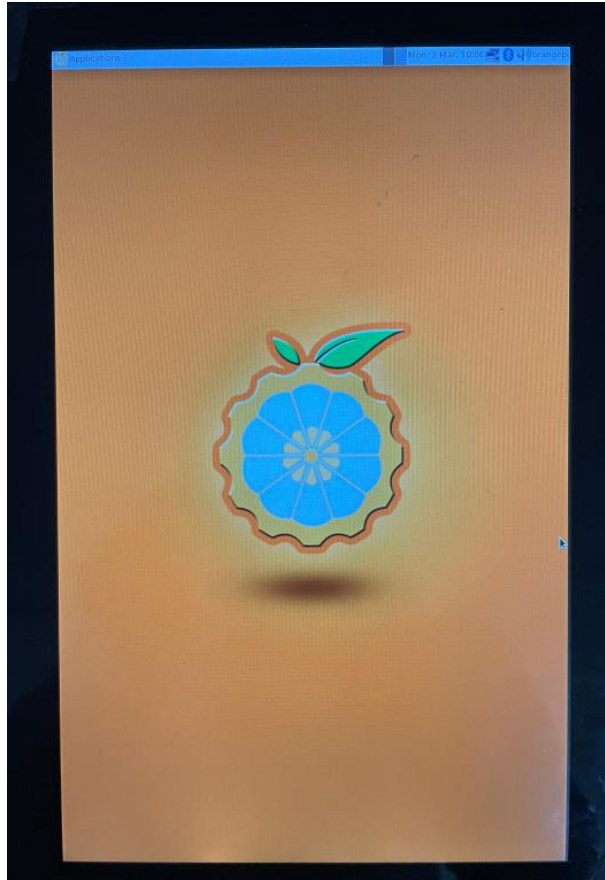
- b) Then restart the system. When starting, you can see the following printing information in the startup log of u-boot, indicating that the configuration of lcd1 is loaded successfully

```
379 bytes read in 9 ms (41 KiB/s)
Applying kernel provided DT overlay rockchip-lcd1.dtbo
```

- c) After entering the Linux system, enter the following command to see the relevant information of the kernel output

```
root@orangepi4-lts:~# dmesg |grep mipi
[          8.622148]      dw-mipi-dsi-rockchip      ff960000.dsi:
[drm:dw_mipi_dsi_bridge_mode_set] final DSI-Link bandwidth: 444 x 4 Mbps
```

- 2) Then disconnect the power supply of the development board, and then connect the MIPI screen by referring to [the usage of the 10.1-inch MIPI screen](#), plug in the power supply to start the system, and you can see the system interface on the screen



3. 32. 2. LCD2 interface instructions

1) In the dts of the Linux system, LCD2 is turned off by default. First, you need to open the configuration of LCD2 in dts.

a. For Linux 4.4 kernel, the method is as follows

a) A script named **orangepi-add-overlay** is pre-installed in the linux system. Through this script, we can use DT overlay to dynamically modify the configuration in dts. First write the rockchip-lcd2.dts file, the content is as follows

```
root@orangepi4-lts:~# cat rockchip-lcd2.dts
/dts-v1/;
/plugin/;

/ {
    compatible = "rockchip,rk3399";
    fragment@0 {
        target = <&dsi1>;
```



```
        __overlay__ {
            status = "okay";
        };
};
fragment@1 {
    target = <&gt9xx_1>;
    __overlay__ {
        status = "okay";
    };
};
fragment@2 {
    target = <&rkisp1_1>;
    __overlay__ {
        status = "disabled";
    };
};
fragment@3 {
    target = <&isp1_mmu>;
    __overlay__ {
        status = "disabled";
    };
};
fragment@4 {
    target = <&mipi_dphy_tx1rx1>;
    __overlay__ {
        status = "disabled";
    };
};
fragment@5 {
    target = <&ov13850_1>;
    __overlay__ {
        status = "disabled";
    };
};
fragment@6 {
    target = <&cdn_dp>;
```



```

        __overlay__ {
            status = "disabled";
        };
    };
    fragment@7 {
        target = <&hdmi>;
        __overlay__ {
            status = "disabled";
        };
    };
};

```

- b) Then you can use **orangepi-add-overlay** to compile rockchip-lcd2.dts into rockchip-lcd2.dtbo

```

root@orangepi:~# orangepi-add-overlay rockchip-lcd2.dts
Compiling the overlay
Copying the compiled overlay file to /boot/overlay-user/
Reboot is required to apply the changes

```

- c) Then restart the Linux system, you can see the following printing information in the log output by the serial port, indicating that rockchip-lcd2.dtbo is loaded successfully

```

1135 bytes read in 6 ms (184.6 KiB/s)
Applying user provided DT overlay rockchip-lcd2.dtbo

```

- d) After entering the Linux system, enter the following command to see the relevant information of the kernel output

```

root@orangepi4-lts:~# dmesg |grep mipi
root@orangepi4-lts:~# dmesg |grep mipi
[ 2.714240] dw-mipi-dsi ff960000.dsi: final DSI-Link bandwidth: 444 x 4 Mbps
[ 3.104433] dw-mipi-dsi ff968000.dsi: final DSI-Link bandwidth: 444 x 4 Mbps

```

- b. For Linux5.10 system, the method is as follows
- a) You can set overlays variable in /boot/orangepiEnv.txt to open LCD2 configuration

```

root@orangepi4-lts:~# cat /boot/orangepiEnv.txt
verbosity=1
bootlogo=true

```



```
overlay_prefix=rockchip
fdtfile=rockchip/rk3399-orangepi-4-lts.dtb
rootdev=UUID=c51e6614-42cf-473c-9134-46a72667eb9c
rootfstype=ext4
overlays=lcd1 lcd2
```

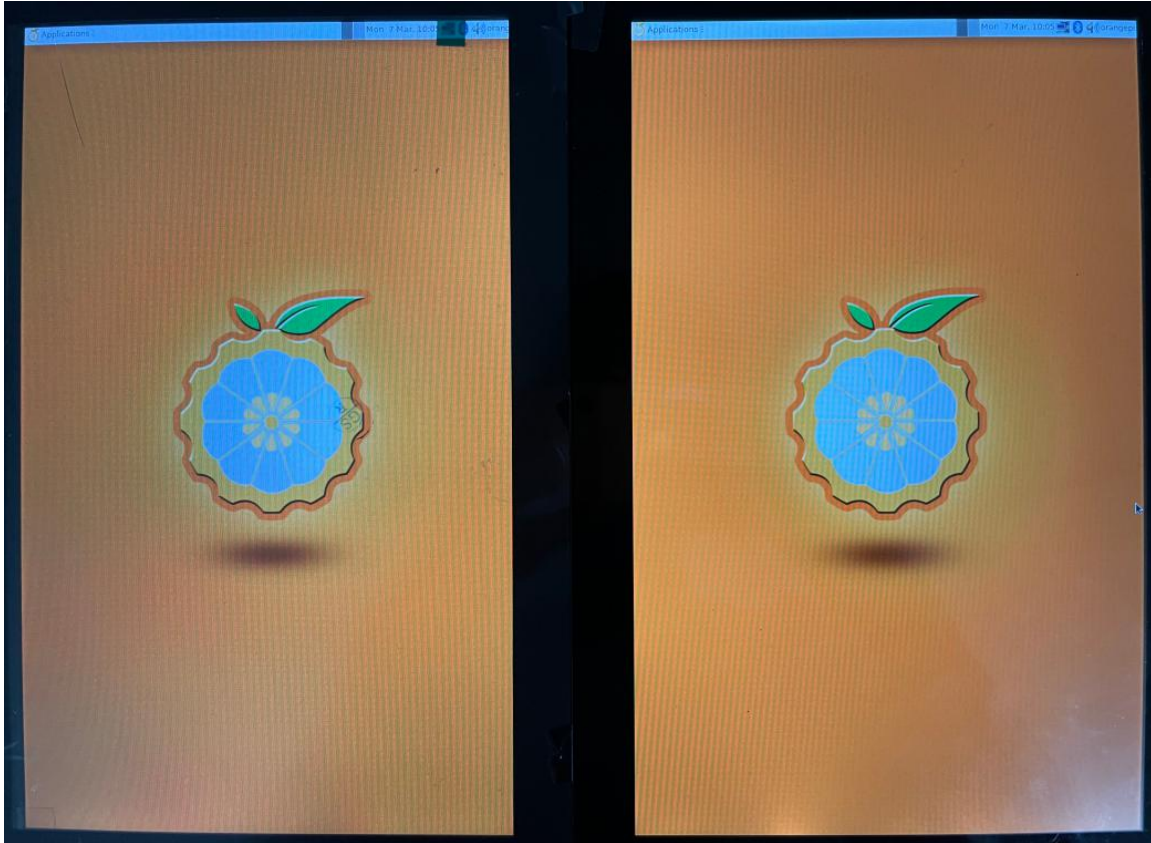
- b) Then restart the system. When starting up, you can see the following printing information in the startup log of u-boot, indicating that the configurations of lcd1 and lcd2 are loaded successfully

```
379 bytes read in 9 ms (41 KiB/s)
Applying kernel provided DT overlay rockchip-lcd1.dtbo
626 bytes read in 9 ms (67.4 KiB/s)
Applying kernel provided DT overlay rockchip-lcd2.dtbo
```

- c) After entering the Linux system, enter the following command to see the relevant information of the kernel output

```
root@orangepi4-lts:~# dmesg |grep mipi
[          8.124926]      dw-mipi-dsi-rockchip      ff960000.dsi:
[drm:dw_mipi_dsi_bridge_mode_set] final DSI-Link bandwidth: 444 x 4 Mbps
[          8.168243]      dw-mipi-dsi-rockchip      ff968000.dsi:
[drm:dw_mipi_dsi_bridge_mode_set] final DSI-Link bandwidth: 444 x 4 Mbps
```

- 2) Then disconnect the power supply of the development board, and then connect the MIPI screen by referring to [the usage of the 10.1-inch MIPI screen](#), plug in the power supply to start the system, if two 10.1-inch screens are connected at the same time, you can see the system on the two 10.1-inch screens interface



3. 33. How to set up dual screen display

1) The Linux image on the official website supports HDMI output and TypeC-DP output by default. You can use the following commands to set dual-screen simultaneous display

```
root@orangepi4-lts:~#  
su orangepi -c "DISPLAY=:0 xrandr --output HDMI-1 --same-as DP-1"
```

2) If you use MIPI LCD output and connect two 10.1-inch MIPI screens, you can use the following commands to set up, **if the system restarts after setting, please confirm whether the power supply is sufficient**

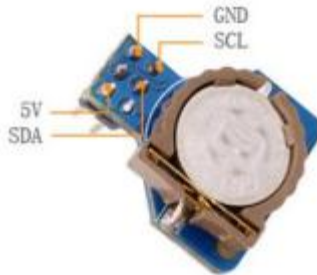
```
root@orangepi4-lts:~#  
su orangepi -c "DISPLAY=:0 xrandr --output DSI-1 --same-as DSI-2"
```

3. 34. How to use the orange pi DS1307 RTC clock module

1) The orange pi DS1307 RTC clock module is shown in the figure below. It uses the



i2c interface to communicate with the development board, and the i2c device address is 0x68. The RTC module is not equipped with a battery by default, and a button battery needs to be prepared before use



2) First, connect the RTC module to the 40pin of the development board. The wiring method is as follows

Pins of the RTC module	The corresponding pin of the development board 26pin
5V	pin 2
GND	Pin 6
SDA	Pin 3
SCL	Pin 5

3) After connecting the RTC module, first use the i2cdetect command to check whether the device address of the RTC module can be detected

```

root@orangepi:~# apt update
root@orangepi:~# apt -y install i2c-tools
root@orangepi:~# i2cdetect -y 8
root@orangepi4-lts:~# i2cdetect -y 8
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
10:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60:  -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

```

4) Because the kernel has opened the ds1037 driver by default, the function can be



tested directly. Execute the following command to add an rtc device and view the generated rtc device, where rtc0 is the onboard rtc and rtc1 is the newly added external rtc

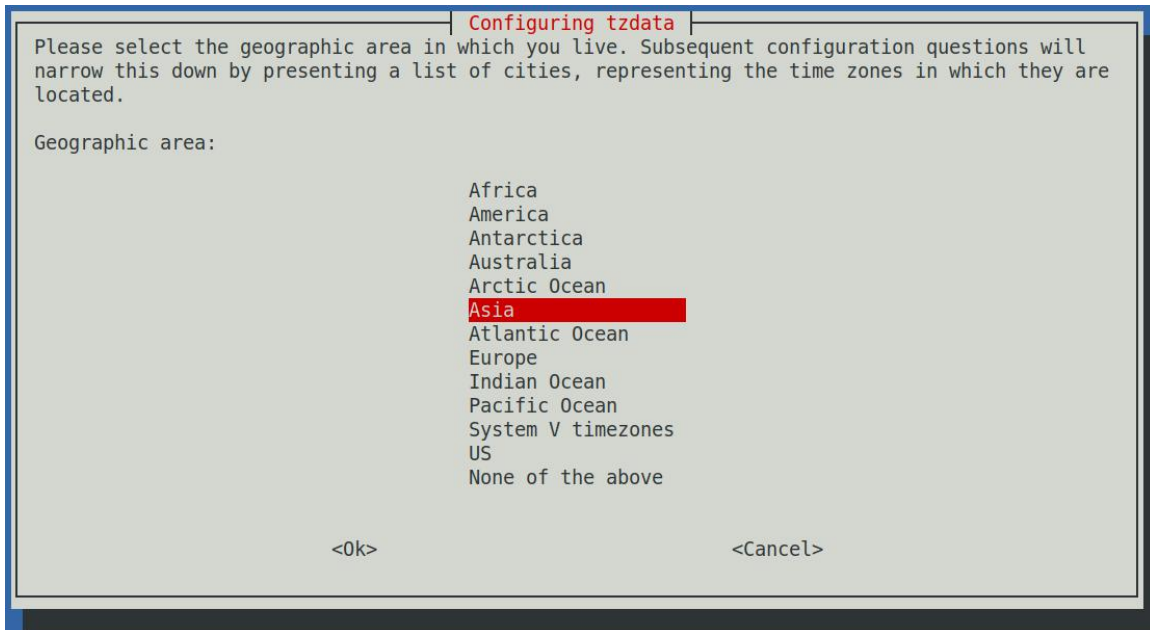
```
root@orangepi:~# echo "ds1307 0x68" > /sys/class/i2c-adapter/i2c-8/new_device
root@orangepi:~# ls /dev/rtc*
/dev/rtc /dev/rtc0 /dev/rtc1
```

5) When the linux system starts, if the development board is connected to the network, the linux system will automatically synchronize the system time to the correct time through the network. The default time of the linux system is UTC. In China, the time zone needs to be changed to **Asia/Shanghai**. The time obtained by using the data command is correct, the method is as follows

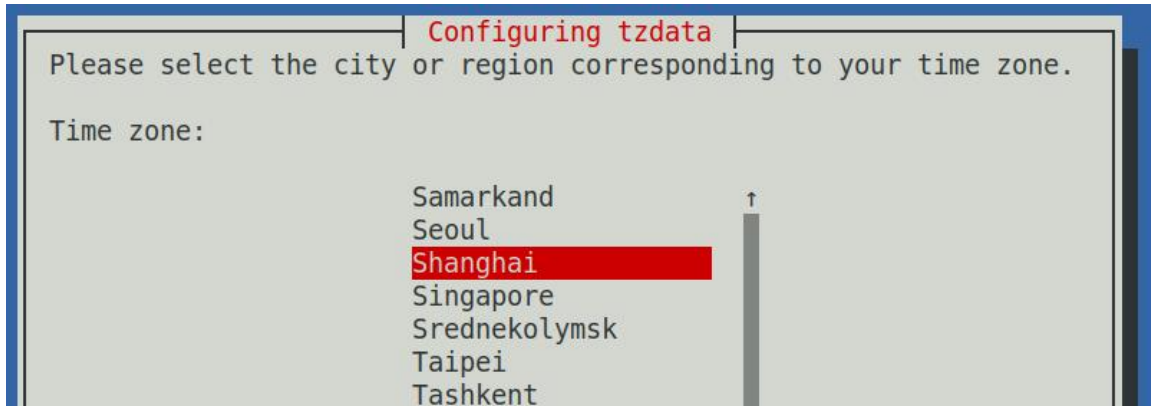
- a. Execute the following command

```
root@orangepi:~# dpkg-reconfigure tzdata
```

- b. Then select the geographic area as **Asia**



- c. Then select the time zone as **Shanghai**



d. After the configuration is completed, use the date command to view the time and it will be normal

```
root@orangePi:~# date
```

6) If the current time of the system is incorrect, please connect to the network first, and then use the following command to synchronize the time. The reason why the system time is set correctly is to prepare for the synchronization of the time of the RTC module later.

```
root@orangePi:~# apt -y update
root@orangePi:~# apt install ntpdate
root@orangePi:~# ntpdate 0.cn.pool.ntp.org
```

7) The command to view the current time of the RTC module is as follows, because the rtc specified by hwclock by default is the rtc0 device, so you need to use the -f option to specify the rtc1 device

```
root@orangePi:~# hwclock -r -f /dev/rtc1
```

8) The time read by the RTC module for the first time is definitely wrong. The current time of the system can be synchronized to the RTC module through the following command. Before synchronization, it is necessary to ensure that the current time of the system is correct

```
root@orangePi:~# date #First make sure the current system time is
correct
root@orangePi:~# hwclock -w -f /dev/rtc1 #Then write the system time to the RTC
module
root@orangePi:~# hwclock -r -f /dev/rtc1 #Finally read the time of the RTC module to
```



confirm that the settings are correct

9) If it is confirmed that the time in the RTC module is correct, then you can unplug the power supply, then power it on, and execute the following command to synchronize the time in the RTC module to the system

```
root@orangepi:~#
echo "ds1307 0x68" > /sys/class/i2c-adapter/i2c-8/new_device #add rtc device
root@orangepi:~# hwclock -s -f /dev/rtc1 #Then synchronize the RTC module
time to the system
root@orangepi:~# date #Finally, use the date command to check
whether the system time is correct
```

10) The above operation is to manually synchronize the time of the RTC module to the system. If you need to automatically synchronize the system time at startup, you need to set the startup script as follows to automatically synchronize the system time

a. Create the rc-local.service file

```
root@orangepi:~# sudo vi /etc/systemd/system/rc-local.service
```

Copy the following content into the rc-local.service file

```
[Unit]
Description=/etc/rc.local Compatibility
ConditionPathExists=/etc/rc.local

[Service]
Type=forking
ExecStart=/etc/rc.local start
TimeoutSec=0
StandardOutput=tty
RemainAfterExit=yes
SysVStartPriority=99

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

b. Create the rc.local file

```
root@orangepi:~# sudo vi /etc/rc.local
```

Copy the following content into the rc-local.service file



```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

echo "ds1307 0x68" > /sys/class/i2c-adapter/i2c-8/new_device
hwclock -s -f /dev/rtc1

exit 0
```

c. Add permissions to rc.local

```
root@orangepi:~# sudo chmod +x /etc/rc.local
```

d. Enable service

```
root@orangepi:~# sudo systemctl enable rc-local
```

e. Start the service and check the status

```
root@orangepi:~# sudo systemctl start rc-local.service
root@orangepi:~# sudo systemctl status rc-local.service
```

11) At this point, you can disconnect all network connections of the development board, wait for a few minutes, restart the system, and then check the system time to find that even if there is no network, the system time is correct

3. 35. Reset and shutdown methods

1) During the operation of the Linux system, if the power is directly unplugged, some data may be lost in the file system. It is recommended to use the poweroff command to shut down the Linux system of the development board before power off, and then unplug the power.



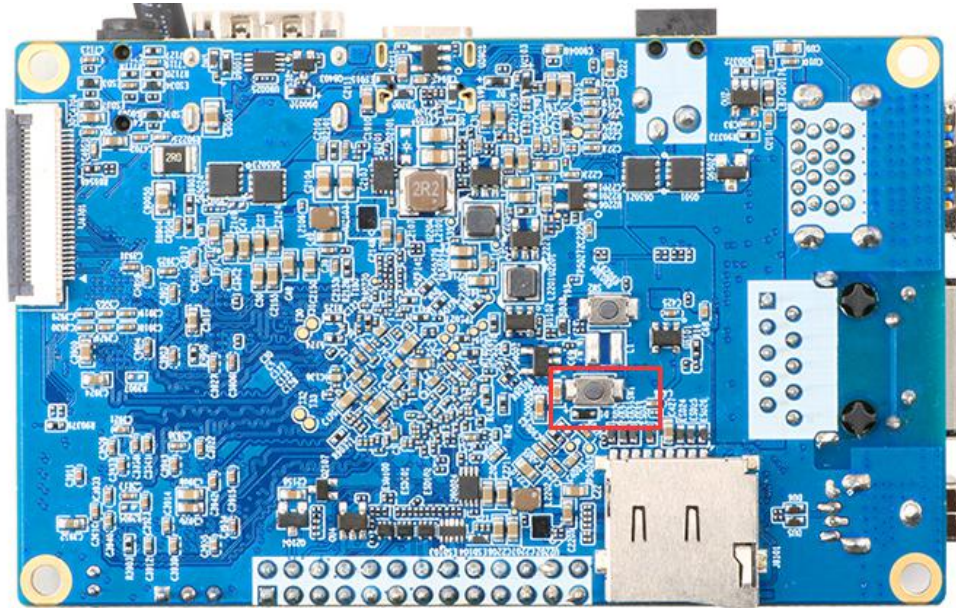
```
root@orangepi:~# poweroff
```

2) After turning off the development board, you need to re-plug the power supply to turn it on

3) Use the reboot command to restart the Linux system on the development board

```
root@orangepi:~# reboot
```

4) You can also short press the power button on the development board to reset



3. 36. Partial programming language test supported by Linux system

1) The full name of the image tested is as follows

```
Orangepi4-lts_2.2.0_debian_bullseye_linux5.10.43_xfce_desktop..7z
```

2) Debian Bullseye is installed with gcc compilation toolchain by default, which can directly compile C language programs in the Linux system of the development board

a. The version of a gcc is shown below

```
root@orangepi:~# gcc --version
```

```
gcc (Debian 10.2.1-6) 10.2.1 20210110
```

```
Copyright (C) 2020 Free Software Foundation, Inc.
```



This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE..

- b. Write the **hello_world.c** program in C language

```
root@orangepi:~# cat hello_world.c
#include <stdio.h>

int main(void)
{
    printf("Hello World!\n");

    return 0;
}
```

- c. Then compile and run **hello_world.c**

```
root@orangepi:~# gcc -o hello_world hello_world.c
root@orangepi:~# ./hello_world
Hello World!
```

3) Debian Bullseye has Python3 installed by default

- a. The specific version of Python is as follows

```
root@orangepi4-lts:~# python3
Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

- b. Write the **hello_world.py** program in the Python language

```
root@orangepi:~# cat hello_world.py
print('Hello World!')
```

- c. The result of running **hello_world.py** is as follows

```
root@orangepi:~# python3 hello_world.py
Hello World!
```

4) Debian Bullseye does not install Java compilation tools and runtime environment by default

- a. You can install openjdk with the following command, the default version in



Debian Bullseye is openjdk-11

```
root@orangeypi:~# apt -y install openjdk-11-jdk
```

b. After installation, you can check the version of Java

```
root@orangeypi:~# java --version
```

```
openjdk 11.0.12 2021-07-20
```

```
OpenJDK Runtime Environment (build 11.0.12+7-post-Debian-2)
```

```
OpenJDK 64-Bit Server VM (build 11.0.12+7-post-Debian-2, mixed mode)
```

c. Write the Java version of **hello_world.java**

```
root@orangeypi:~# vim hello_world.java
```

```
public class hello_world
```

```
{
```

```
    public static void main(String[] args)
```

```
    {
```

```
        System.out.println("Hello World!");
```

```
    }
```

```
}
```

d. Then compile and run **hello_world.java**

```
root@orangeypi:~# javac hello_world.java
```

```
root@orangeypi:~# java hello_world
```

```
Hello World!
```



4. Instructions for use of Android system

4.1. Supported Android Versions

Android version	kernel version
Android 8.1	linux4.4

4.2. Android 8.1 function adaptation

Function	status
HDMI video	OK
HDMI audio	OK
USB2.0 x 2	OK
USB3.0 x 1	OK
TypeC USB3.0	OK
TF card boot	OK
EMMC start	OK
network card	OK
WIFI	OK
Bluetooth	OK
Bluetooth earphone	OK
headphone recording	OK
Headphone playback	OK
microphone recording	OK
LED lights	OK
Temperature Sensor	OK
USB camera	OK
GPU	OK
Video codec	OK
Reset button	OK
upgrade key	OK
ADB debugging	OK
OV13850 camera	OK



10.1 inch MIPI screen	OK
mini-PCIE	OK
TypeC to HDMI	OK

4. 3. On-board LED light display description

	green light	red light
u-boot startup phase	Light off	light on
The kernel boots into the system	flicker	light on

4. 4. How to use ADB

4. 4. 1. Use data cable to connect adb debugging

1) First prepare a good quality Type-C data cable. It is recommended to use the white port Type-C 2.0 data cable shown on the left. The purple port quick charging cable shown on the right does not work properly on the OrangePi 4 LTS development board



2) First, you need to use the Type-C data cable to connect the development board to the USB interface of the computer (please use the DC power supply to power the development board at the same time)

3) Install adb tool on Ubuntu PC

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



4) Check to identify the ADB device

```
test@test:~$ adb devices
List of devices attached
S63QCF54CJ    device
test@test:~$ lsusb
Bus 003 Device 006: ID 2207:0006
```

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

```
test@test:~$ adb shell
rk3399_mid:/ $
```

6) If you need to change the Android system files, you need to close the security verification, execute the following command

```
test@test:~$ adb root
test@test:~$ adb disable-verity
```

7) Then execute the command to restart the system

```
test@test:~$ adb reboot
```

8) Remount the Android system

```
test@test:~$ adb root
test@test:~$ adb remount
```

9) Then you can transfer files to the Android system

```
test@test:~$ adb push example.txt /system/
```

4. 4. 2. Using network connection adb debugging

Using network adb does not require the USB Type 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 has been connected, and then obtain the IP address of the development board, and then to use

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



```
rk3399_mid:/ # getprop | grep "adb.tcp"  
[service.adb.tcp.port]: [5555]
```

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

```
rk3399_mid:/ # setprop service.adb.tcp.port 5555  
rk3399_mid:/ # stop adbd  
rk3399_mid:/ # start adbd
```

3) Install adb tool on Ubuntu PC

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

4) Then connect network adb on Ubuntu PC

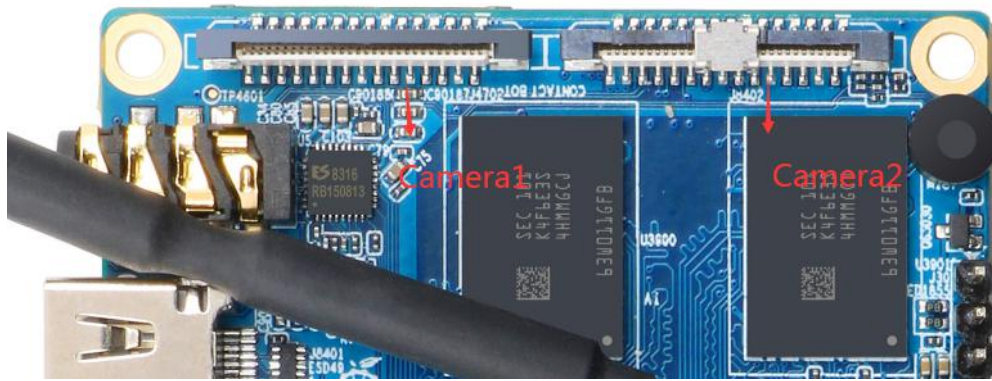
```
test@test:~$ adb connect 192.168.1.xxx (The IP address needs to be changed to  
the IP address of the development board)  
* daemon not running; starting now at tcp:5037  
* daemon started successfully  
connected to 192.168.1.xxx:5555  
  
test@test:~$ adb devices  
List of devices attached  
192.168.1.xxx:5555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

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

```
test@test:~$ adb shell  
rk3399_mid:/ #
```

4. 5. How to use OV13850 camera

1) OrangePi 4 LTS has two Camera ports and both support OV13850 camera. The two Camera interfaces can use one of the interfaces alone, or you can use the two Camera interfaces to connect two cameras at the same time. After connecting two cameras, one is front and one is rear



2) The OV13850 camera kit includes an OV13850 camera, an adapter board, and a ribbon cable



3) First insert the OV13850 camera into the adapter board, and then insert the cable into another slot on the adapter board

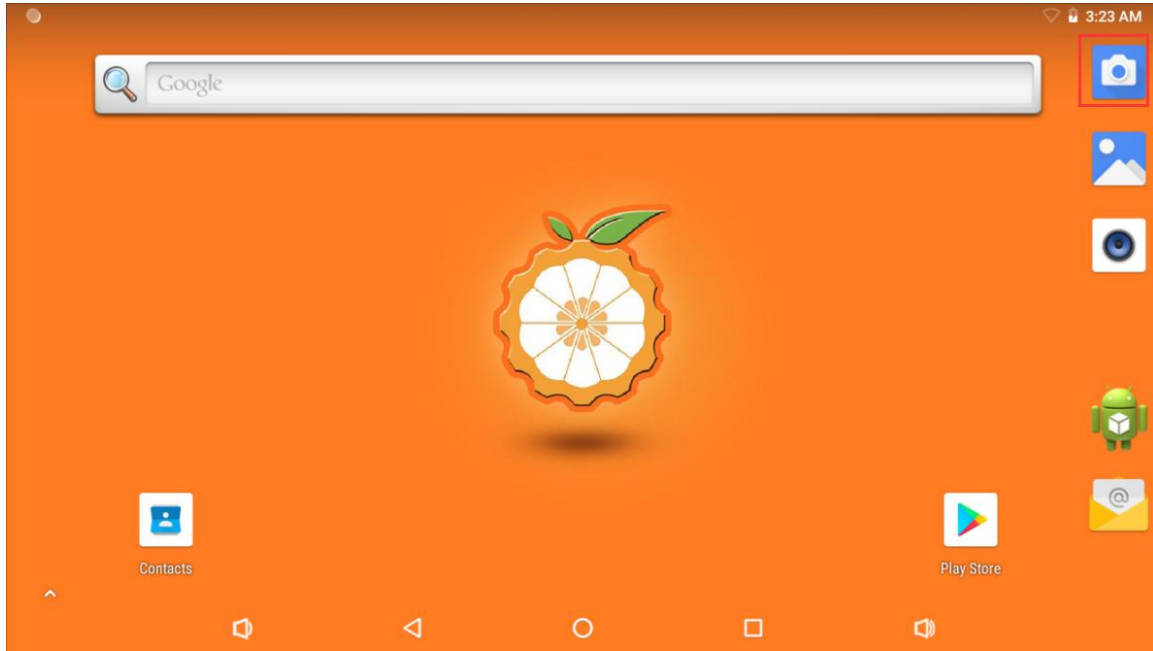




4) Then insert the other end of the cable into the Camera camera interface of the development board. The interface can be connected to two cameras at the same time, or one camera can be connected separately. After connecting the camera, start the Android system (**do not plug in the camera after power on**)

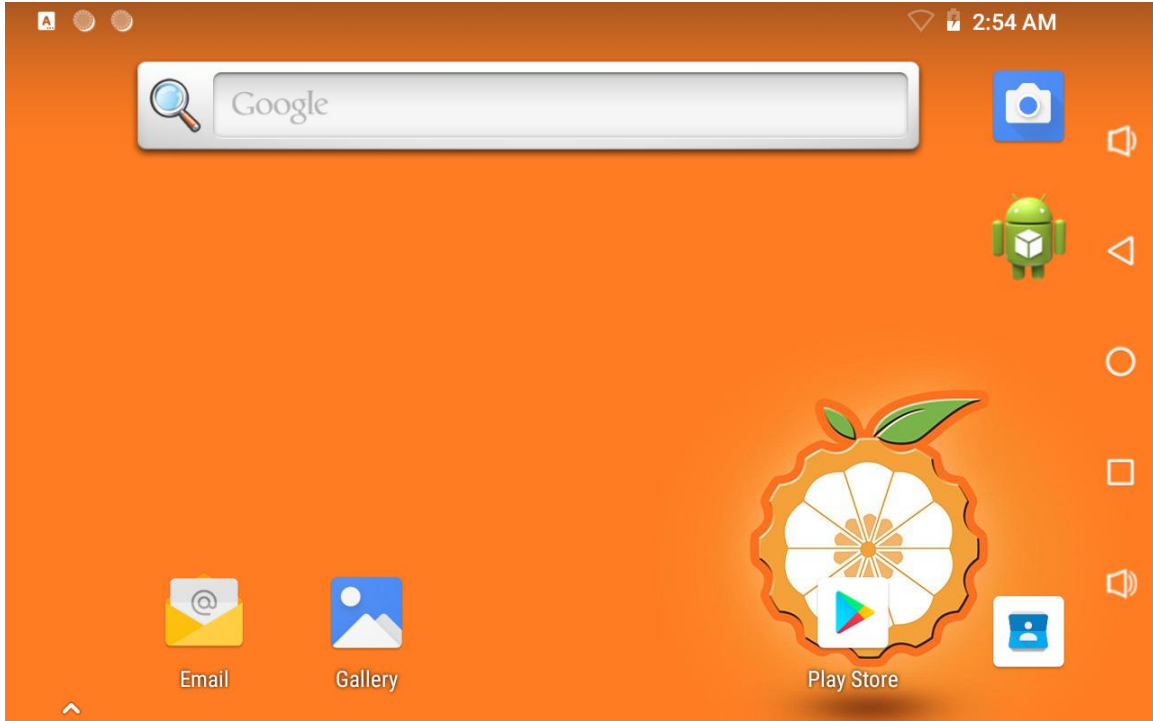


5) After the Android system is started, open the camera APP and you can see the output of the OV13850 camera. The location of the camera APP is shown in the following figure:

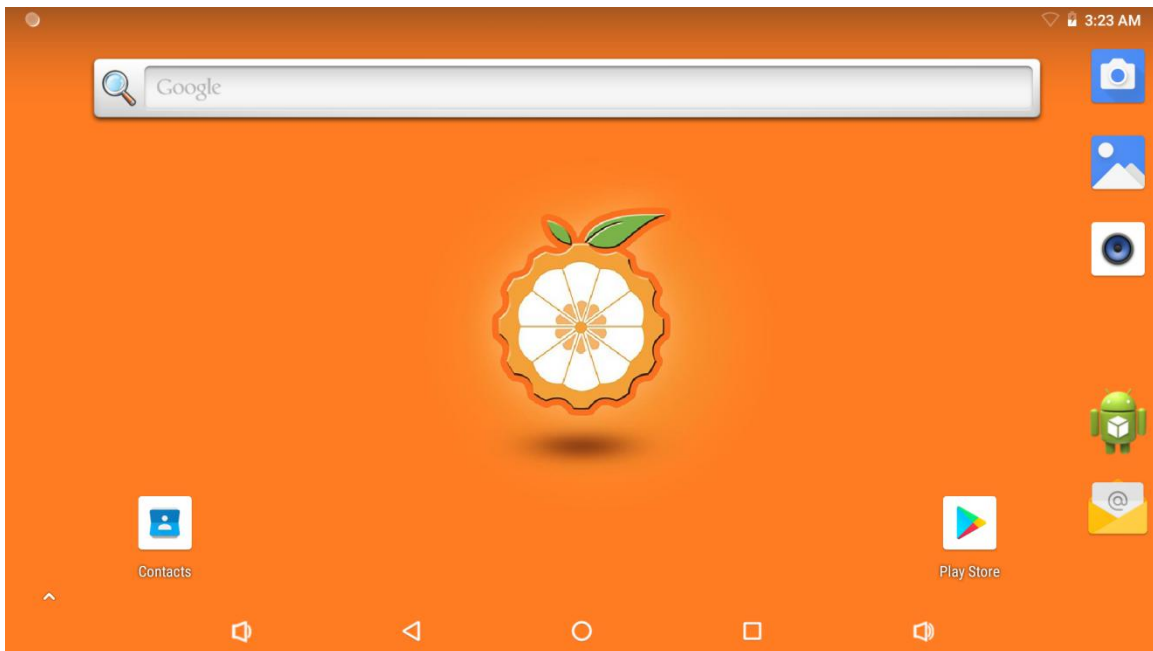


4.6. How to use the HDMI interface

1) The factory-installed Android of the Orange Pi 4 LTS development board supports HDMI display. Since it also supports a 10.1-inch MIPI screen, its UI resolution is only 1280x800. After connecting the development board to an HDMI display or TV through an HDMI to HDMI cable, the interface after the system starts is as shown in the figure below



2) After testing all functions with the pre-installed system and confirming that there is no problem, if the 10.1-inch MIPI screen is not connected, in order to obtain a better display effect, it is recommended to re-burn the image without the word "LCD" and wait for the system to start up. After that, the display interface is as shown in the figure below, and the UI resolution is 1920*1080





4.7. The method of displaying the system interface through the TypeC interface

1) The factory-installed Android of the Orange Pi 4 LTS development board supports Type-C to HDMI display. You need to prepare a Type-C to HDMI cable, and connect the development board to an HDMI display or TV through the Type-C interface for display

4.8. 10.1 inch MIPI screen usage

The pre-installed Android system of the Orange Pi 4 LTS development board already supports MIPI screen. If it is not the pre-installed Android system, please re-burn the Android firmware with the word "LCD" on the official website that supports 10.1-inch MIPI screen.

1) 10.1 inch MIPI screen delivery list, including a touch screen, a MIPI LCD screen, a 31pin to 40pin cable, a 12pin touch screen cable, a 30pin MIPI cable, and a transfer board



2) Connect the 12pin touch screen cable and 30pin MIPI cable to the adapter board as shown in the figure below. Note that the 12pin touch screen cable direction is the blue bar facing down.



3) With the touch screen facing down, stack the MIPI LCD screen on the touch screen as shown below



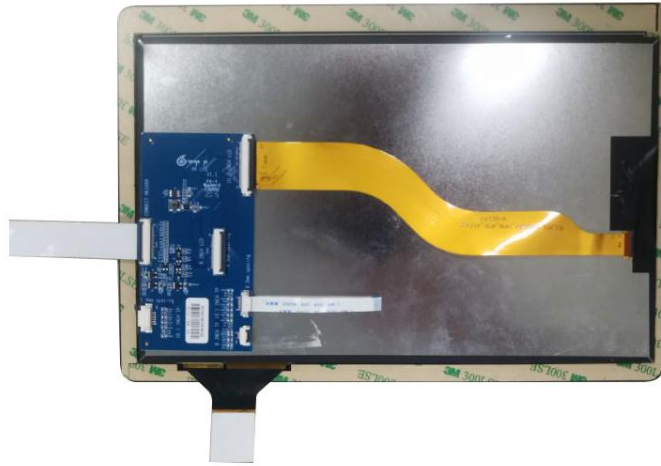
4) Place the connected adapter board on the MIPI LCD screen as shown in the figure below



5) Then connect the MIPI LCD screen and the adapter board through the 31pin to 40pin



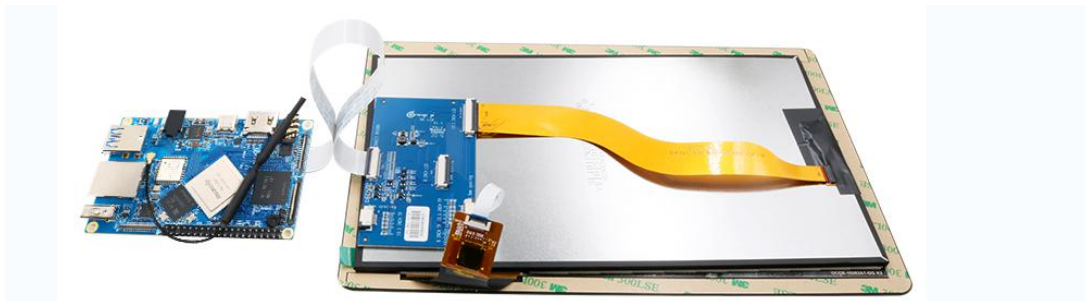
cable



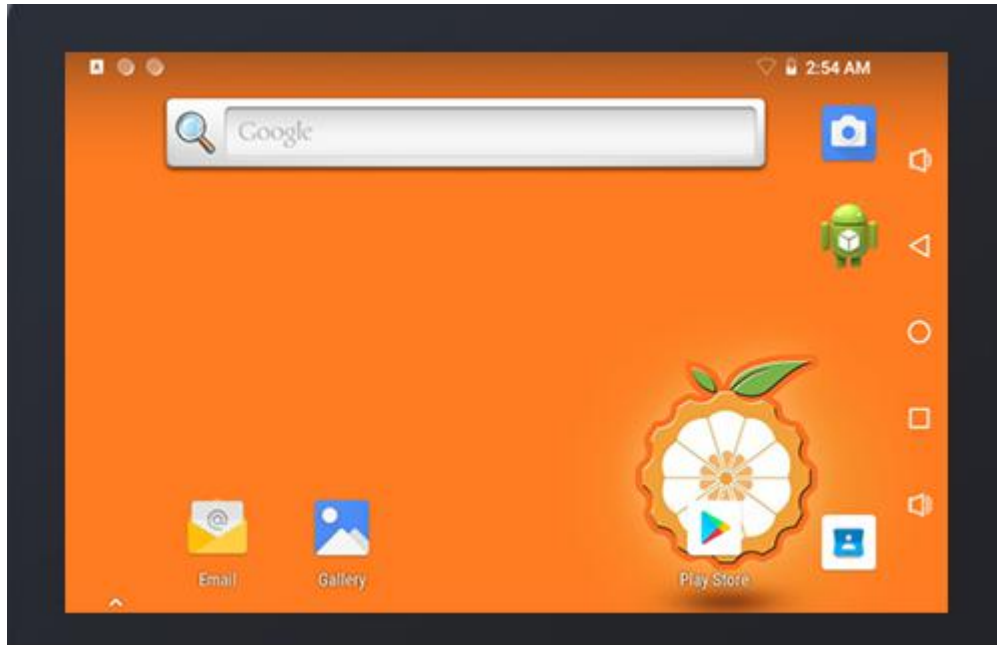
6) Then connect the touch screen and the adapter board through the 12pin touch screen cable



7) Connect the adapter board to the LCD1 interface of the Orange Pi 4 LTS through a 30pin MIPI cable



8) Then insert the DC power supply into the development board. After the system is started, the interface is as shown in the figure below.



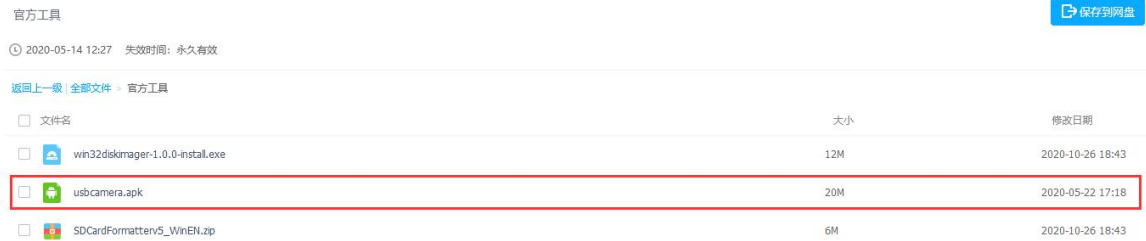
4.9. How to use the USB camera

1) First insert the USB camera into the USB interface of the development board. If the USB camera is recognized normally, the corresponding video device node will be generated under /dev

```
rk3399_mid:/ $ ls /dev/video*  
/dev/video0  
rk3399_mid:/ $ ls /sys/class/video4linux/ -lh  
total 0  
lrwxrwxrwx 1 root root 0 2020-09-30 03:29 video0 \  
-> ../../devices/platform/usb@fe900000/fe900000.d0
```

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

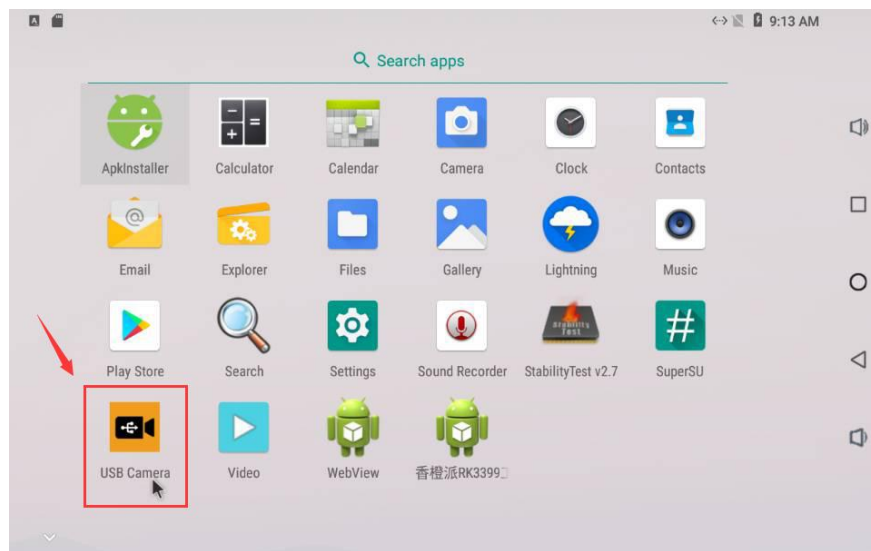
3) Download the USB camera test APP in the official tool on the data download page of Orange Pi 4 LTS



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

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

5) After installation, you can see the startup icon of the USB camera in the Android App list

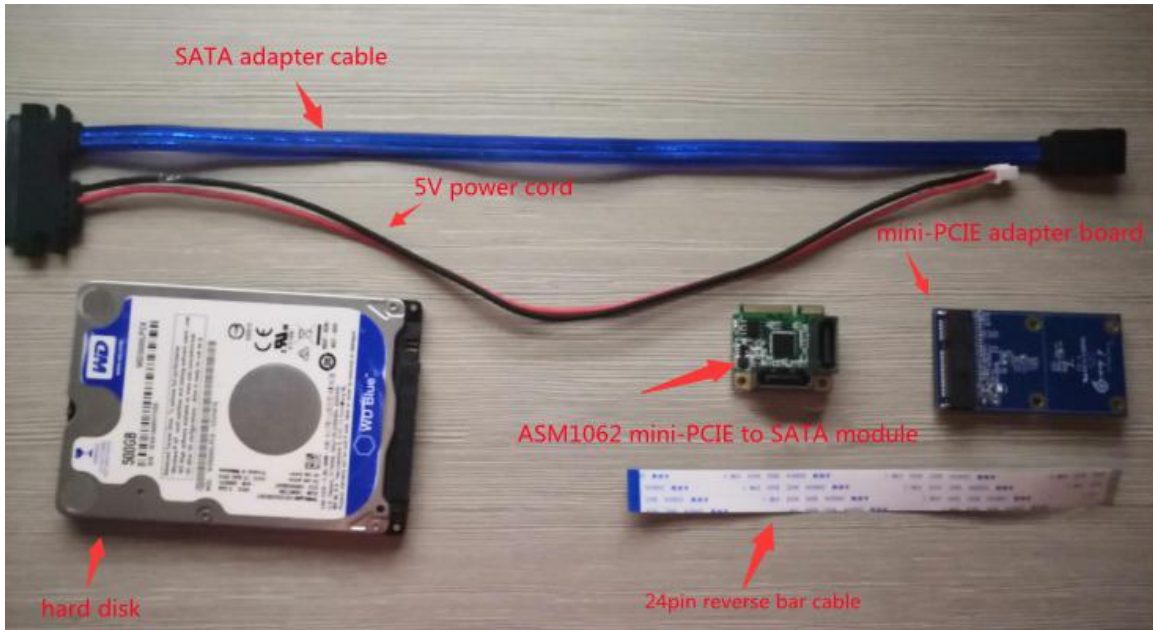


6) Then double-click to open the USB camera APP to see the output video of the USB camera

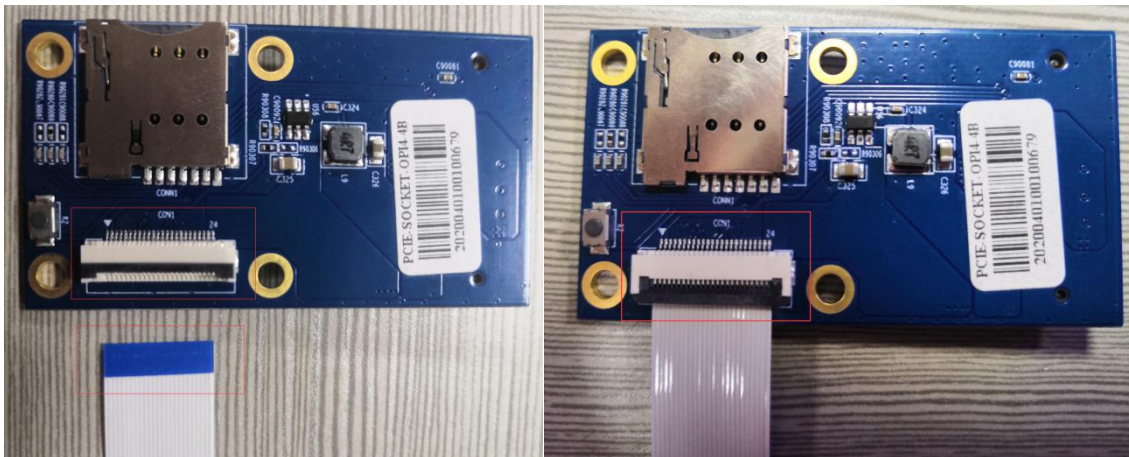
4. 10. Instructions for use of Mini-PCIE

4. 10. 1. Instructions for connecting mini-PCIE to hard disk

1) Prepare the required accessories, 24pin reverse cable, mini-PCIE adapter board, ASM1062 mini-PCIE to SATA module, hard disk, SATA adapter cable, 5V power cable, accessories picture as shown below



2) Connect the 24pin reverse cable to the mini-PCIE adapter board as shown in the figure below. Note that the blue bar of the cable is facing outward.



3) Connect the mini-PCIE adapter board to the 24pin interface of the Orange Pi 4 LTS development board



4) Connect the ASM1062 mini-PCIE to SATA module to the mini-PCIE adapter board



5) Connect the hard disk to the interface of the mini-PCIE to SATA module through a SATA cable

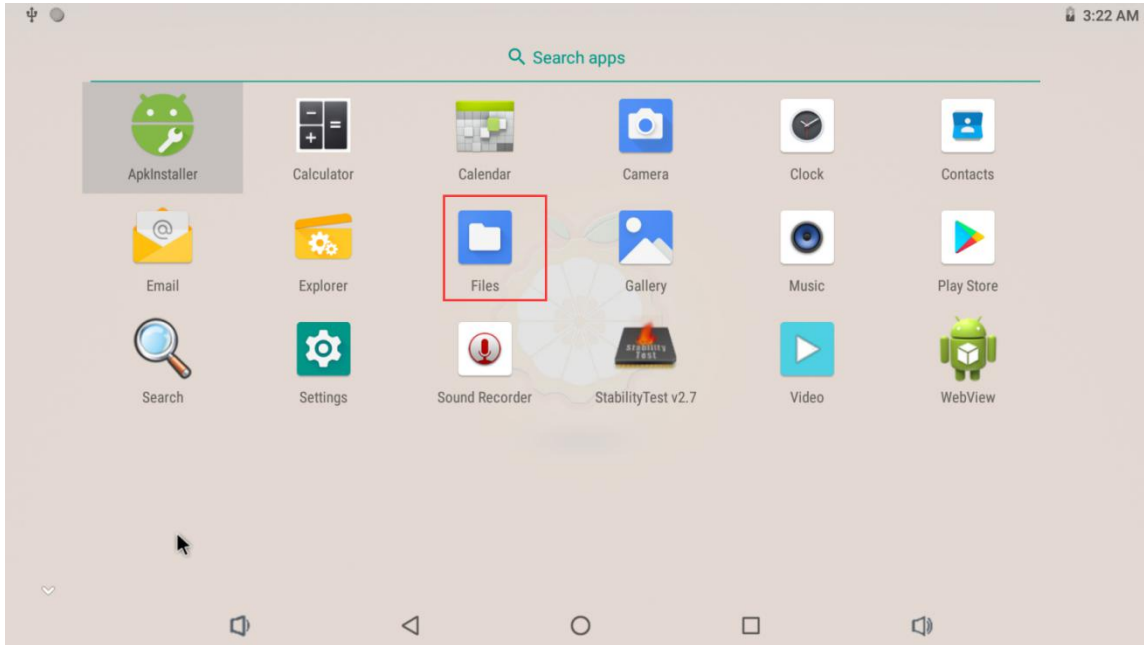


6) Connect the power cable of the SATA adapter cable to the 5V power supply. After the connection is completed, the development board is connected to the DC power supply and powered on. ASM1062 mini-PCIe to SATA module LED light flashes, indicating that the connection is successful

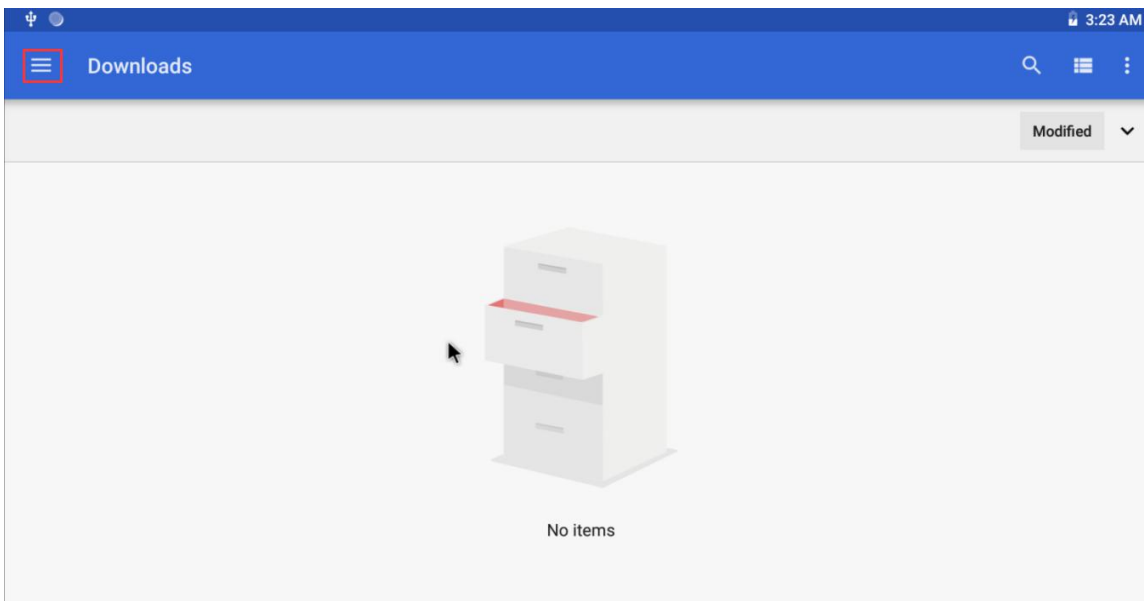


This function does not support hot swapping and must be connected before powering on

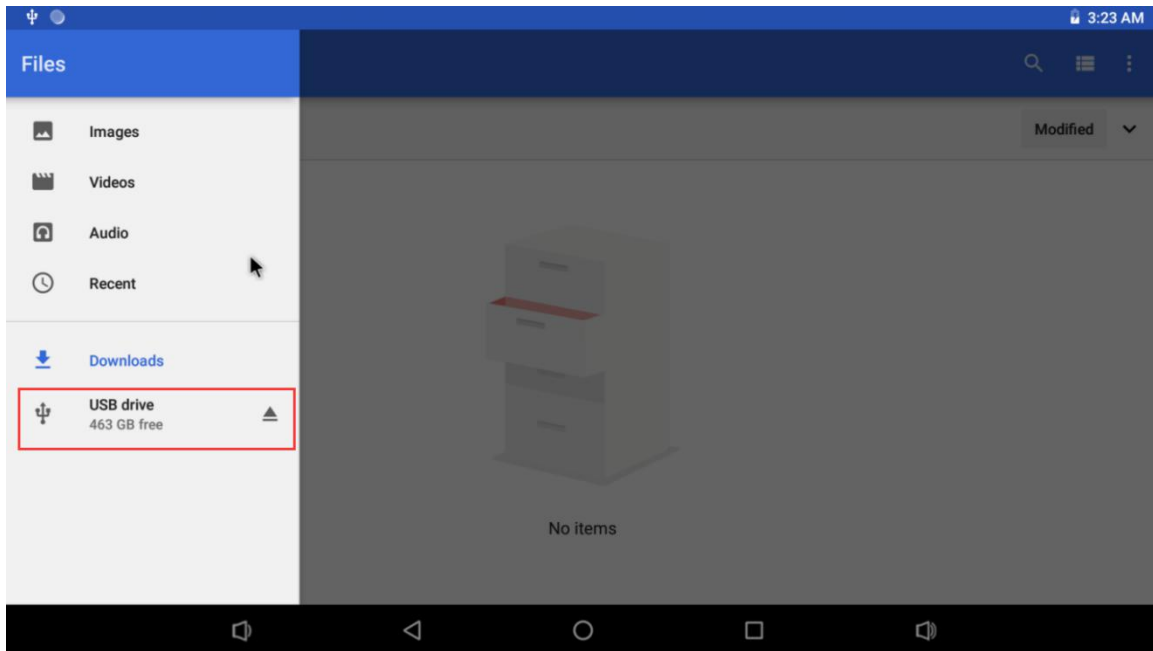
7) Power on and start the development board. After the system is started, open the file manager app



8) After opening, the file manager interface is as shown below, and then click the position of the red box in the upper left corner



9) At this point, you can see that the 500G hard disk is recognized





5. Instructions for using the Linux SDK

5.1. Compilation system requirements

1) The Linux SDK, the Ubuntu_21.04 branch of orangepi-build, only supports running on a computer with **Ubuntu21.04** installed, so before downloading orangepi-build, please make sure that the Ubuntu version installed on your computer is Ubuntu21.04. The command to check the Ubuntu version installed on the computer is as follows. If the Release field shows other than **21.04**, it means that the current Ubuntu version does not meet the requirements. Please change the system before performing the following operations.

```
test@test:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 21.04
Release:        21.04
Codename:       hirsute
test@test:~$
```

2) If the Windows system is installed on the computer and there is no computer with Ubuntu 21.04 installed, you can consider using VirtualBox or VMware to install an Ubuntu21.04 virtual machine in the Windows system. But please note, do not compile orangepi-build on the WSL virtual machine, because orangepi-build has not been tested in the WSL virtual machine, so it cannot be guaranteed that orangepi-build can be used normally in WSL, and please do not use the Linux system of the development board. using orangepi-build in

5.2. Get the source code of linux sdk

Note that Orange Pi 4 LTS must use the source code of the next branch of orangepi-build, and the source code of the main branch is not adapted to Orange Pi 4 LTS.

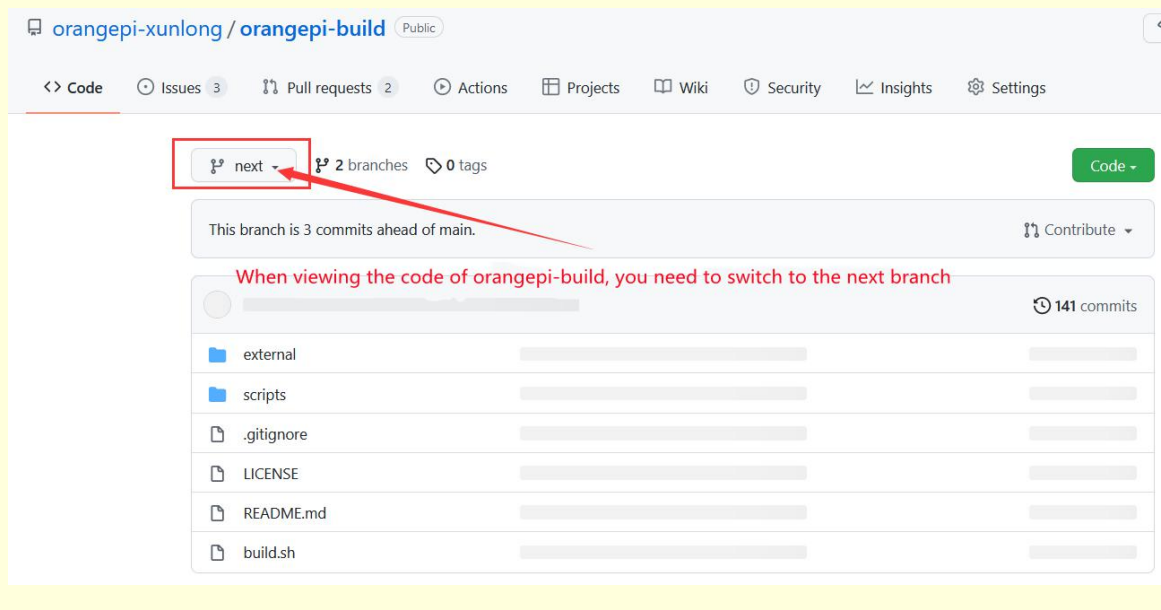


5. 2. 1. Download orangepi-build from github

1) The linux sdk actually refers to the code of orangepi-build. Orangepi-build is modified based on the armbian build compilation system. Using orangepi-build, multiple versions of linux images can be compiled. First download the code of orangepi-build. Currently, the RK3399 series development boards in the Linux SDK already support the legacy branch and the current branch.

```
test@test:~$ sudo apt update
test@test:~$ sudo apt install git
test@test:~$ git clone https://github.com/orangepi-xunlong/orangepi-build.git -b
Ubuntu21.04
```

Note that Orange Pi 4 LTS must use the source code of the next branch of orangepi-build. The above git clone command specifies that the branch of the orangepi-build source code is next. The source code of the main branch is not adapted to Orange Pi 4 LTS.



Downloading the code of orangepi-build through the git clone command does not require entering the username and password of the github account (the same is true for downloading other codes in this manual). If the Ubuntu PC prompts the user who needs to enter the github account after entering the git clone command The name and password are usually the wrong address of the orangepi-build



repository behind git clone. Please check the spelling of the command carefully, instead of thinking that we forgot to provide the username and password of the github account here.

2) The u-boot and linux kernel versions currently used by the RK3399 series development boards are as follows

branch	u-boot version	linux kernel version
legacy	u-boot 2020.10	linux4.4
current	u-boot 2020.10	linux5.10

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

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

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

If you downloaded the code of orangepi-build from github, after downloading, you may find that orangepi-build does not contain the source code of u-boot and linux kernel, nor does it require cross-compilation to compile u-boot and linux kernel Toolchain, this is normal, because these things are stored in other separate github repositories or some servers (the addresses will be detailed below). orangepi-build will specify the address of u-boot, linux kernel and cross-compilation toolchain in the script and configuration file. When running orangepi-build, when it finds that these things are not available locally, it will automatically go to the corresponding place to download.

5. 2. 2. Download the cross-compilation toolchain

1) When orangepi-build runs for the first time, it will automatically download the cross-compilation toolchain and put it in the toolchains folder. After running the build.sh script of orangepi-build, 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

```
[ o.k. ] Checking for external GCC compilers
[ ..... ] downloading using http(s) network [ gcc-linaro-aarch64-none-elf-4.8-2013.11_linux.tar.xz ]
#8d7029 16MiB/24MiB (65%) CN:1 DL:7.9MiB ETA:1s
[ o.k. ] Verified [ PGP ]
[ ..... ] decompressing
[ ..... ] gcc-linaro-aarch64-none-elf-4.8-2013.11_linux.tar.xz: 24.9MiB [14.4MiB/s] [=====] 100%
[ ..... ] downloading using http(s) network [ gcc-linaro-arm-none-eabi-4.8-2014.04_linux.tar.xz ]
#e38eec 17MiB/33MiB (99%) CN:1 DL:10MiB ETA:1s
[ o.k. ] Verified [ PGP ]
[ ..... ] decompressing
[ ..... ] gcc-linaro-arm-none-eabi-4.8-2014.04_linux.tar.xz: 33.9MiB [9.66MiB/s] [=====] 100%
[ ..... ] downloading using http(s) network [ gcc-linaro-arm-linux-gnueabi-4.8-2014.04_linux.tar.xz ]
#041c24 48MiB/48MiB (99%) CN:1 DL:2.7MiB
[ o.k. ] Verified [ PGP ]
[ ..... ] decompressing
[ ..... ] gcc-linaro-arm-linux-gnueabi-4.8-2014.04_linux.tar.xz: 48.8MiB [13.0MiB/s] [=====] 100%
[ ..... ] downloading using http(s) network [ gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabi.tar.xz ]
#3dee3e 72MiB/76MiB (93%) CN:1 DL:3.7MiB ETA:1s
[ o.k. ] Verified [ MD5 ]
[ ..... ] decompressing
[ ..... ] gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabi.tar.xz: 77.0MiB [14.2MiB/s] [=====] 100%
[ ..... ] downloading using http(s) network [ gcc-linaro-7.4.1-2019.02-x86_64_arm-linux-gnueabi.tar.xz ]
#42e728 104MiB/104MiB (99%) CN:1 DL:2.0MiB
[ o.k. ] Verified [ MD5 ]
[ ..... ] decompressing
[ ..... ] gcc-linaro-7.4.1-2019.02-x86_64_arm-linux-gnueabi.tar.xz: 104MiB [13.9MiB/s] [=====] 100%
[ ..... ] downloading using http(s) network [ gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu.tar.xz ]
#2c065e 108MiB/111MiB (97%) CN:1 DL:3.9MiB
[ o.k. ] Verified [ MD5 ]
[ ..... ] decompressing
[ ..... ] gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu.tar.xz: 111MiB [13.4MiB/s] [=====] 100%
[ ..... ] downloading using http(s) network [ gcc-arm-9.2-2019.12-x86_64-arm-none-linux-gnueabi.tar.xz ]
#d232ee 250MiB/251MiB (99%) CN:1 DL:2.0MiB
[ o.k. ] Verified [ MD5 ]
[ ..... ] decompressing
[ ..... ] gcc-arm-9.2-2019.12-x86_64-arm-none-linux-gnueabi.tar.xz: 251MiB [13.7MiB/s] [=====] 100%
[ ..... ] downloading using http(s) network [ gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu.tar.xz ]
#88b441 268MiB/269MiB (99%) CN:1 DL:0.9MiB
[ o.k. ] Verified [ MD5 ]
[ ..... ] decompressing
```

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

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

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

```
test@test:~/orange-pi-build$ ls toolchains/
gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu
gcc-arm-9.2-2019.12-x86_64-arm-none-linux-gnueabi
gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabi
gcc-linaro-5.5.0-2017.10-x86_64_arm-linux-gnueabi
gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu
gcc-linaro-7.4.1-2019.02-x86_64_arm-linux-gnueabi
gcc-linaro-aarch64-none-elf-4.8-2013.11_linux
gcc-linaro-arm-linux-gnueabi-4.8-2014.04_linux
gcc-linaro-arm-none-eabi-4.8-2014.04_linux
```

4) The cross-compilation toolchain used to compile the RK3399 Linux kernel source code is

a. linux4.4

[gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu](#)

b. linux5.10



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

- 5) The cross-compilation toolchain used to compile RK3399 u-boot source code is
- a. v2020.10

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

5.2.3. Orangepi-build complete directory structure description

1) After the orangepi-build repository is downloaded, it does not contain the source code of the linux kernel, u-boot and the cross-compilation tool chain. The source code of the linux kernel and u-boot are stored in a separate git repository

- a. The git repository where the linux kernel source code is stored is as follows, pay attention to switch the branch of the linux-orangepi repository to
 - a) linux4.4

```
https://github.com/orangepi-xunlong/linux-orangepi/tree/orange-pi-4.4-rockchip64
```

- b) Linx5.10

```
https://github.com/orangepi-xunlong/linux-orangepi/tree/orange-pi-5.10-rk3399
```

- b. The git repository where the u-boot source code is stored is as follows. Note that the branch of the u-boot-orangepi repository is switched to

v2020.10-rockchip64

```
https://github.com/orangepi-xunlong/u-boot-orangepi/tree/v2020.10-rockchip64
```

If you are not familiar with orangepi-build and do not know the detailed process of compiling the linux kernel and u-boot, please do not download and use the above linux kernel and u-boot source code for compilation operation, because the compilation script and configuration file of orangepi-build Some adjustments and optimizations will be made to u-boot and linux. If you do not use orangepi-build to compile u-boot and linux, you may encounter problems of compilation failure or failure to start.

2) When orangepi-build runs for the first time, it will download the cross-compilation toolchain, u-boot and linux kernel source code. After successfully compiling a linux image, the files and folders that can be seen in orangepi-build are:

- c. **build.sh**: Compile startup script
- d. **external**: Contains configuration files, scripts for specific functions and source code of some programs needed to compile the image. The rootfs compressed package cached in the process of compiling the image is also stored in external



- e. **kernel**: Store the source code of the linux kernel, the folder named orange-pi-4.4-rockchip64 stores the kernel source code of the legacy branch of the RK3399 series development board, and the folder named orange-pi-5.10-rk3399 stores the The kernel source code of the current branch of the RK3399 series development board (if only the linux image of the legacy branch is compiled, then only the kernel source code of the legacy branch can be seen; if only the linux image of the current branch is compiled, then only the current branch can be seen. Kernel source code), please do not manually modify the name of the folder of the kernel source code. If modified, the compilation system will re-download the kernel source code when running.
- f. **LICENSE**: GPL 2 license file
- g. **README.md**: orangepi-build documentation
- h. **output**: Store the compiled u-boot, linux and other deb packages, compilation logs, and compiled images and other files
- i. **scripts**: Generic script for compiling linux images
- j. **toolchains**: Store the cross-compilation toolchain
- k. **u-boot**: The source code of u-boot is stored. The folder named v2020.10-rockchip64 stores the u-boot source code of the current branch and legacy branch of the RK3399 series development board. Please do not manually modify the name of the folder of the u-boot source code. If it is modified, the u-boot source code will be re-downloaded when the compilation system is running.
- l. **userpatches**: Store the configuration files needed to compile the script

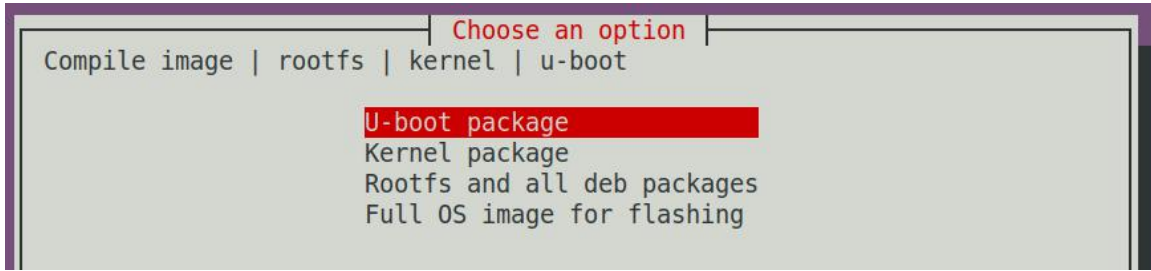
```
test@test:~/orangepi-build$ ls
build.sh  external  kernel    LICENSE   output    README.md  scripts
toolchains  u-boot   userpatches
```

5.3. Compile u-boot

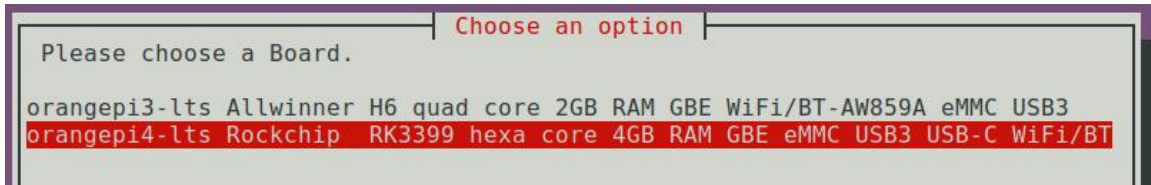
- 1) Run the build.sh script, remember to add sudo permissions

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

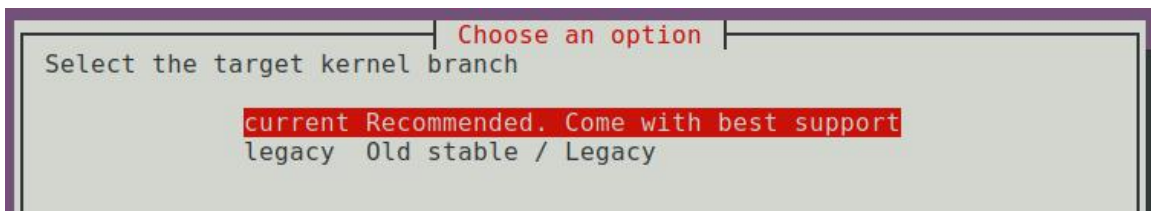
- 2) Select **U-boot package** and press Enter



3) Then select the model of the development board



4) Then select branch



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

a. u-boot source code version

```
[ o.k. ] Compiling u-boot [ v2020.10 ]
```

b. The version of the cross-compile toolchain

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

c. The path to the generated u-boot deb package

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

d. The package name of the u-boot deb package generated by compilation

```
[ o.k. ] File name [ linux-u-boot-current-orangepi4-lts_2.2.0_arm64.deb ]
```

e. Compilation time used

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

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

```
[ o.k. ] Repeat Build Options [ sudo ./build.sh BOARD=orangepi4-lts
```



```
BRANCH=current BUILD_OPT=u-boot KERNEL_CONFIGURE=yes ]
```

6) View the compiled u-boot deb package

```
test@test:~/orangepi-build$ ls output/debs/u-boot/  
linux-u-boot-current-orangepi4-lts_2.2.0_arm64.deb
```

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

a. Use the following command to decompress 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-current-orangepi4-lts_2.2.0_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-orangepi4-lts_2.2.0_arm64.deb  usr
```

b. The decompressed file is as follows

```
test@test:~/orangepi-build/output/debs/u-boot$ tree usr  
usr  
├── lib  
│   ├── linux-u-boot-current-orangepi4-lts_2.2.0_arm64  
│   │   ├── idbloader.bin  
│   │   ├── trust.bin  
│   │   └── uboot.img  
│   └── u-boot  
│       ├── LICENSE  
│       ├── orangepi-4-rk3399_defconfig  
│       └── platform_install.sh  
3 directories, 6 files
```

8) When the orangepi-bulid compilation system 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 need to turn off the download and update function of the source code (**required This function can only be turned off after u-boot is fully compiled once, otherwise it will prompt that the source code of u-boot cannot be found**), otherwise the modifications will be restored, the method is as follows



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

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

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

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

```
test@test:~/orange-pi-build$ cd output/debs/u-boot
test@test:~/orange-pi_build/output/debs/u-boot$ scp \
linux-u-boot-current-orangepi4-lts_2.2.0_arm64.deb root@192.168.1.xxx:/root
```

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

```
root@orangepi:~# apt purge -y linux-u-boot-orangepi4-lts-current
```

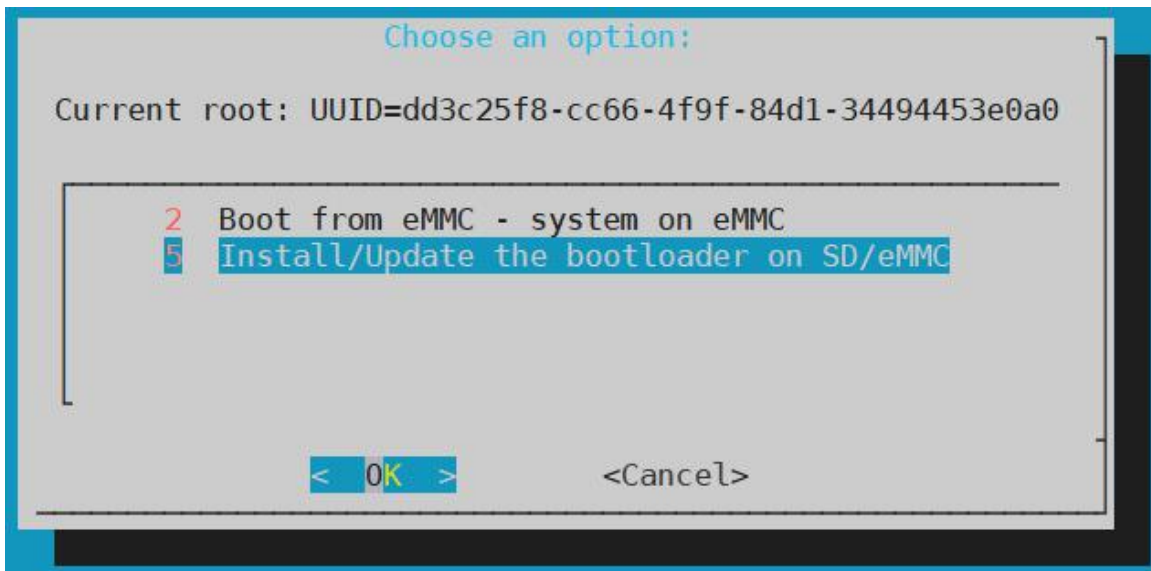
- c. Install the new u-boot deb package just uploaded

```
root@orangepi:~# dpkg -i linux-u-boot-current-orangepi4-lts_2.2.0_arm64.deb
```

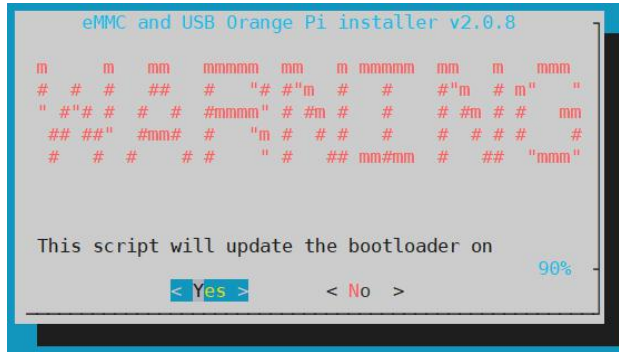
- d. Then run the nand-sata-install script

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

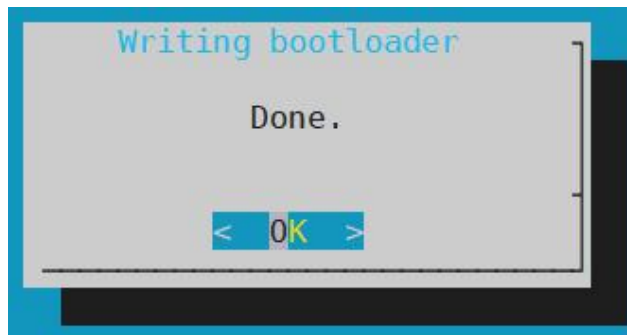
- e. then select **5 Install/Update the bootloader on SD/eMMC**



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



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



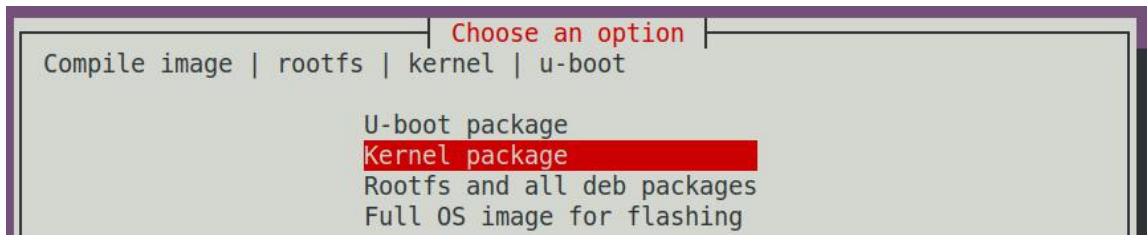
- h. Then you can restart the development board to test whether the modification of u-boot takes effect

5. 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



```
Choose an option
Please choose a Board.
orangepi3-lts Allwinner H6 quad core 2GB RAM GBE WiFi/BT-AW859A eMMC USB3
orangepi4-lts Rockchip RK3399 hexa core 4GB RAM GBE eMMC USB3 USB-C WiFi/BT
```

- 4) Then select branch
 - a. current will compile linux 5.10
 - b. legacy will compile linux4.4

```
Choose an option
Select the target kernel branch
current Recommended. Come with best support
legacy Old stable / Legacy
```

- 5) Then the kernel configuration interface opened by **make menuconfig** will pop up. At this time, you can directly modify the kernel configuration. If you do not need to modify the kernel configuration, you can simply exit. After exiting, the kernel source code will be compiled.



```

Linux/arm64 5.10.43 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: [*]

| General setup --->
[*] Support DMA zone
[*] Support DMA32 zone
    Platform selection --->
    Kernel Features --->
    Boot options --->
    Power management options --->
    CPU Power Management --->
    Firmware Drivers --->
[ ] Virtualization ----
[*] ARM64 Accelerated Cryptographic Algorithms --->
    General architecture-dependent options --->
[*] Enable loadable module support --->
[*] Enable the block layer --->
    IO Schedulers --->
    Executable file formats --->
    Memory Management options --->
[*] Networking support --->
    Device Drivers --->
    File systems --->
    Security options --->
--* Cryptographic API --->
    Library routines --->
    Kernel hacking --->

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

```

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

```
test@test:~/orange-pi-build$ sudo ./build.sh KERNEL_CONFIGURE=no
```

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

c. If the following error is displayed when compiling the kernel, this is because the terminal interface of the Ubuntu PC is too small, so the interface of `make menuconfig` cannot be displayed. Please adjust the terminal of the Ubuntu PC to the maximum, and then re-run 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

```

6) Part of the information prompted when compiling the kernel source code is explained as follows

a. The version of the linux kernel source code

```
[ o.k. ] Compiling current kernel [ 5.10.43 ]
```

b. The version of the cross-compilation toolchain used

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

c. The configuration file used by the kernel by default and the path where it is stored

```
[ o.k. ] Using kernel config file [ config/kernel/linux-5.10-rk3399.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 the same as the default configuration file

```
[ o.k. ] Exporting new kernel config [ output/config/linux-5.10-rk3399.config ]
```

e. The path to the generated kernel-related deb package

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

f. The package name of the kernel image deb package generated by compilation

```
[ o.k. ] File name [ linux-image-current-rk3399_2.2.0_arm64.deb ]
```

g. Compilation time used

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

h. Finally, the compilation command to repeat the compilation of the last selected kernel will be displayed. Use the following command to directly start compiling the kernel source code without selecting through the graphical interface.

```
[ o.k. ] Repeat Build Options [ sudo ./build.sh BOARD=orange_pi4-lts  
BRANCH=current BUILD_OPT=kernel KERNEL_CONFIGURE=yes ]
```



- 7) View the compiled and generated kernel-related deb packages
- linux-dtb-current-rk3399_2.2.0_arm64.deb** Not used yet, don't worry about it
 - linux-headers-current-rk3399_2.2.0_arm64.deb** Include kernel header files
 - linux-image-current-rk3399_2.2.0_arm64.deb** Contains kernel images and kernel modules

```
test@test:~/orangepi-build$ ls output/debs/linux-*
output/debs/linux-dtb-current-rk3399_2.2.0_arm64.deb
output/debs/linux-image-current-rk3399_2.2.0_arm64.deb
output/debs/linux-headers-current-rk3399_2.2.0_arm64.deb
```

- 8) The files contained in the generated linux-image deb package are as follows
- Use the following command to decompress 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-current-rk3399_2.2.0_arm64.deb test/
test@test:~/orangepi_build/output/debs$ cd test
test@test:~/orangepi_build/output/debs/test$ dpkg -x \
linux-image-current-rk3399_2.2.0_arm64.deb .
test@test:~/orangepi_build/output/debs/test$ ls
boot  etc  lib  linux-image-current-rk3399_2.2.0_arm64.deb  usr
```

- The decompressed file is as follows

```
test@test:~/orangepi-build/output/debs/test$ tree -L 2
.
├── boot
│   ├── config-5.10.43                //The configuration file used to
│                                   compile the kernel source code
│   ├── System.map-5.10.43
│   └── vmlinuz-5.10.43              //Compile the generated kernel image file
├── etc
│   └── kernel
├── lib
│   └── modules                      //Compile the generated kernel module
└── linux-image-current-rk3399_2.2.0_arm64.deb
```



```
├── usr
│   ├── lib
│   └── share
└──
```

8 directories, 4 files

9) The orangepi-bulid compilation system will first synchronize the linux kernel source code with the linux kernel source code of the github server when compiling the linux kernel source code, so if you want to modify the linux kernel source code, you first need to turn off the update function of the source code (**you need to compile it once This function can only be turned off after the linux kernel source code, otherwise it will prompt that the source code of the linux kernel cannot be found**), otherwise the modification 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"
```

10) If the kernel is modified, the following methods can be used to update the kernel and kernel modules of the development board linux system

- a. Upload the compiled deb package of the linux kernel to the linux system of the development board

```
test@test:~/orangepi-build$ cd output/debs
test@test:~/orangepi-build/output/debs$ scp \
linux-image-current-rk3399_2.2.0_arm64.deb root@192.168.1.207:/root
```

- b. Then log in to the development board and uninstall the deb package of the installed linux kernel

```
root@orangepi:~# apt purge -y linux-image-current-rk3399
```

- c. Install the deb package of the new linux kernel just uploaded

```
root@orangepi:~# dpkg -i linux-image-current-rk3399_2.2.0_arm64.deb
```

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

5.5. Compile rootfs

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



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

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

```
Choose an option
Compile image | rootfs | kernel | u-boot

U-boot package
Kernel package
Rootfs and all deb packages
Full OS image for flashing
```

3) Then select the model of the development board

```
Choose an option
Please choose a Board.

orange-pi3-lts Allwinner H6 quad core 2GB RAM GBE WiFi/BT-AW859A eMMC USB3
orange-pi4-lts Rockchip RK3399 hexa core 4GB RAM GBE eMMC USB3 USB-C WiFi/BT
```

4) Then select branch

```
Choose an option
Select the target kernel branch

current Recommended. Come with best support
legacy Old stable / Legacy
```

5) Then select the type of rootfs

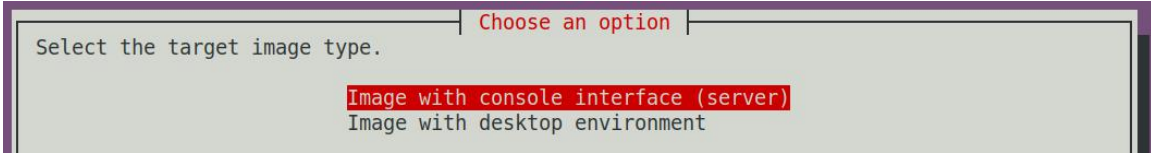
- a. bullseye means Debian 11
- b. **Focal** means Ubuntu 20.04

```
Choose a release package base
Select the target OS release package base

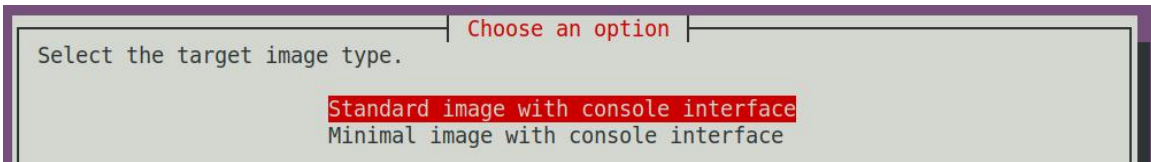
bullseye Debian 11 Bullseye
focal Ubuntu Focal 20.04 LTS
```

6) Then select the type of image

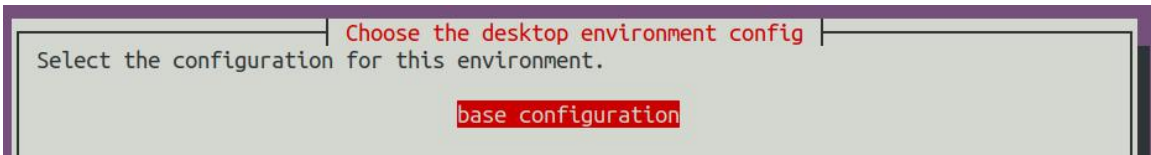
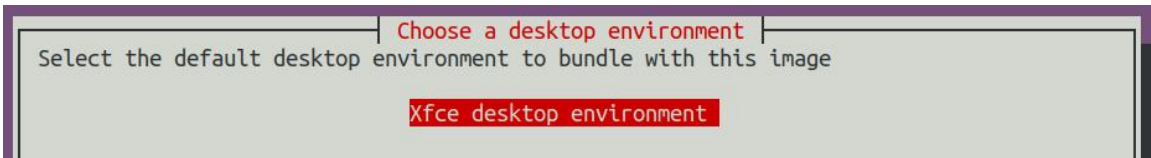
- a. **Image with console interface (server)** Indicates the image of the server version, which is relatively small in size
- b. **Image with desktop environment** Indicates a image with a desktop, which is relatively large in size



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



8) If you are compiling the desktop version of the image, you also need to select the type of desktop environment, but only XFCE is currently supported, so just press Enter.



9) You can then select additional packages that need to be installed. For example, if you need to install a browser, you can choose browsers. Which packages are included in each selection can be seen in the code of orangepi-build, you can also modify these configurations to add the packages you want to install

```

1. First enter external/config/desktop to see the desktop configuration folders of
different linux distributions. Note that not all the Orange Pi development boards that can
be seen in the code are supported and tested.
test@test:~/orangepi-build$ cd external/config/desktop
test@test:~/orangepi-build/external/config/desktop$ ls
bionic bookworm bullseye buster focal jammy README.md sid

2. Then select the type of distribution you want to view or modify, and enter the
    
```



corresponding directory, such as **bullseys**

```
test@test:~/orange-pi-build/external/config/desktop$ cd bullseye
```

```
test@test:~/orange-pi-build/external/config/desktop/bullseye$ ls
```

```
appgroups environments
```

3. Then enter the **appgroups** directory to see all app groups

```
test@test:~/orange-pi-build/external/config/desktop/bullseye$ cd appgroups
```

```
test@test:~/orange-pi-build/external/config/desktop/bullseye/appgroups$ ls
```

```
browsers chat desktop_tools editors games internet multimedia office
programming remote_desktop
```

4. Open the **packages** file under different groups to view the software contained in the group

```
test@test:~/orange-pi-build/external/config/desktop/bullseye/appgroups$ cat
```

```
programming/packages
```

```
geany
```

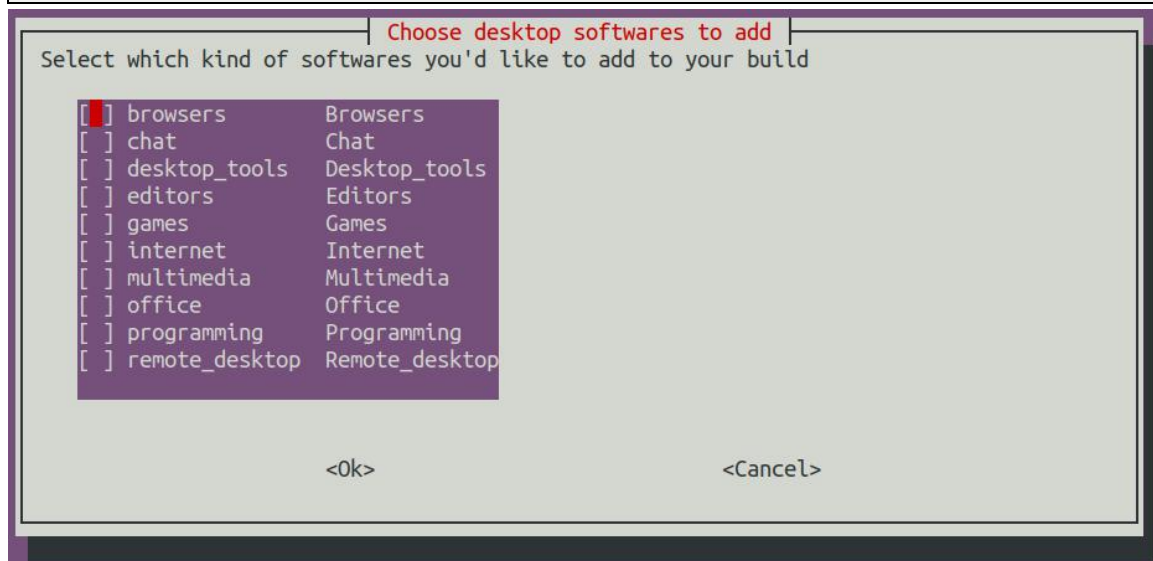
```
thonny
```

```
test@test:~/orange-pi-build/external/config/desktop/bullseye/appgroups$ cat
```

```
office/packages
```

```
ibreoffice
```

5. Except for the browser, the Linux image released by Orange Pi does not choose to install other software packages here. This is mainly to reduce the size of the Linux image.





10) Then the rootfs will be compiled, and some of the information prompted during compilation are as follows

a. type of rootfs

```
[ o.k. ] local not found [ Creating new rootfs cache for bullseye ]
```

b. The storage path of the rootfs compressed package generated by compilation

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

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

```
[ o.k. ] File name
```

```
[ bullseye-xfce-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4 ]
```

d. Compilation time

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

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

```
[ o.k. ] Repeat Build Options [ sudo ./build.sh BOARD=orangepi4-lts  
BRANCH=current BUILD_OPT=rootfs RELEASE=bullseye  
BUILD_MINIMAL=no BUILD_DESKTOP=no  
KERNEL_CONFIGURE=yes ]
```

11) View the rootfs compressed package generated by compilation

a. `bullseye-xfce-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4`

is the compressed package of rootfs, the meaning of each field of the name is

- a) **bullseye** indicates the type of linux distribution of rootfs
- b) **xfce** indicates that the rootfs is of the desktop version, and if it is cli, it indicates the server version
- c) **arm64** represents the architecture type of rootfs
- d) **5250ec7002de9e81a41de169f1f89721** is the MD5 hash value generated by the package names of all packages installed by rootfs. As long as the list of packages installed by rootfs is not modified, this value will not change, and the compilation script will use this MD5 hash value to Determine if you need to recompile rootfs

b. `bullseye-xfce-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4.list` lists the package names of all packages installed by rootfs

```
test@test:~/orangepi-build$ ls external/cache/rootfs/  
bullseye-xfce-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4
```



```
bullseye-xfce-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4.current
bullseye-xfce-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4.list
```

12) 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 find out whether it has There is a rootfs available for cache, if there is one, use it directly, which can save a lot of download and compilation time

13) Since it takes a long time to compile rootfs, if you do not want to compile rootfs from scratch, or there is a problem with the process of compiling rootfs, you can directly download the rootfs compressed package cached by Orange Pi. The download link of the rootfs compressed package Baidu cloud disk is as follows, download A good rootfs compressed package needs to be placed in the **external/cache/rootfs** directory of orangepi-build to be used normally by the compiled script

```
Link: https://pan.baidu.com/s/1vWQmCmSYdH7iCDFyKpJtVw
password: zero
```

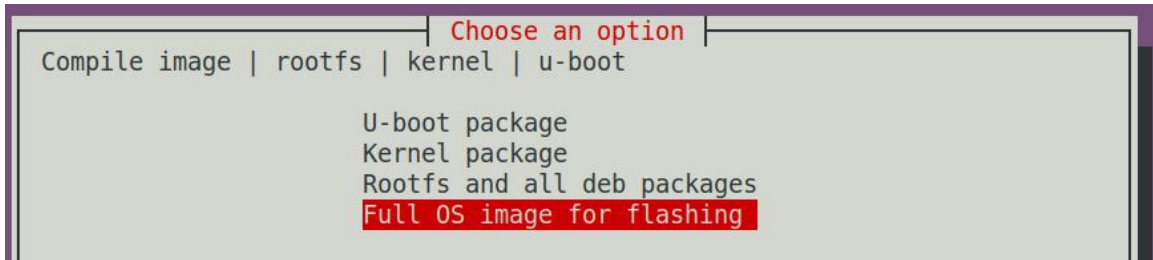


5.6. Compile the linux image

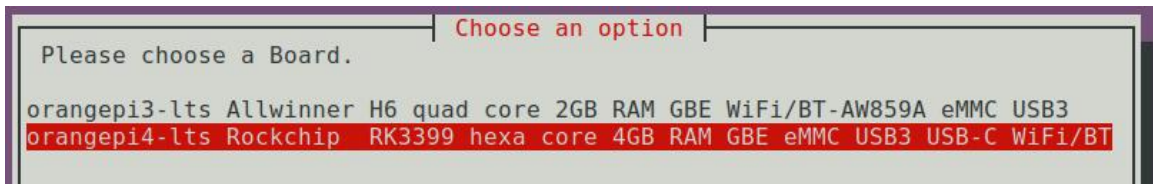
1) Run the build.sh script, remember to add sudo permissions

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

2) Select **Full OS image for flashing** and press Enter

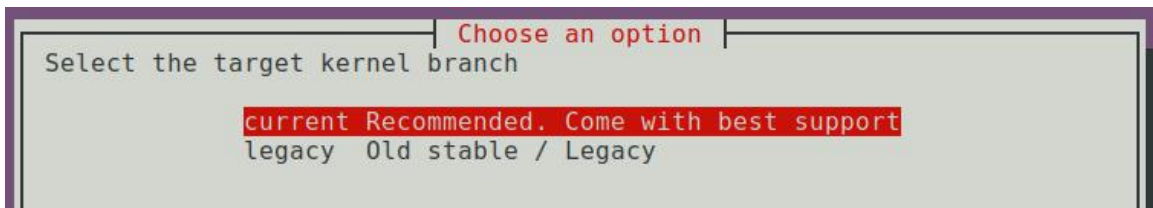


3) Then select the model of the development board



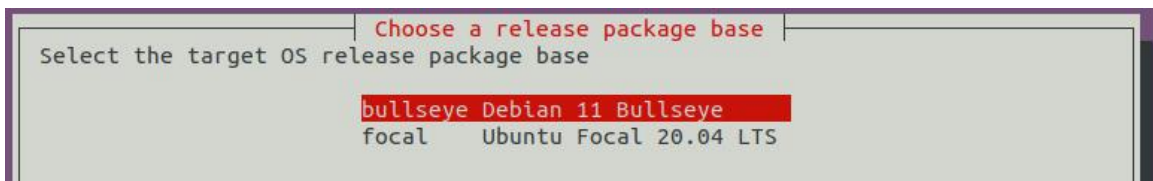
4) Then select branch

- a. current will compile linux 5.10
- b. legacy will compile linux4.4



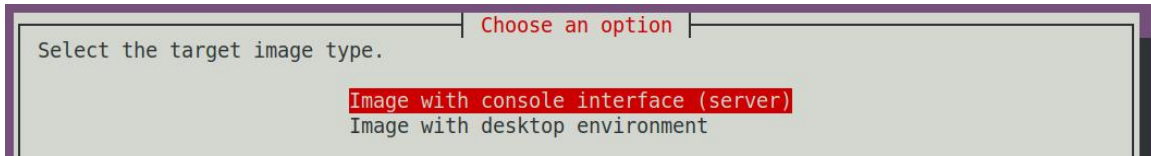
5) Then select the type of rootfs

- a. **Bullseye** means Debian 11
- b. **Focal** means Ubuntu 20.04

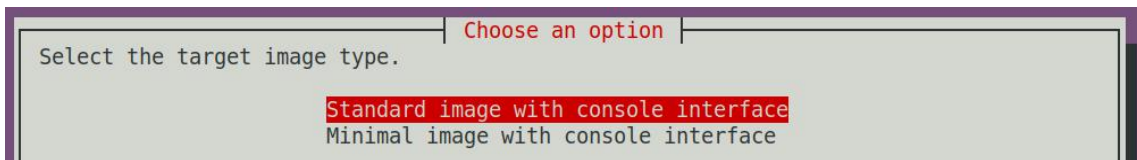


6) Then select the type of image

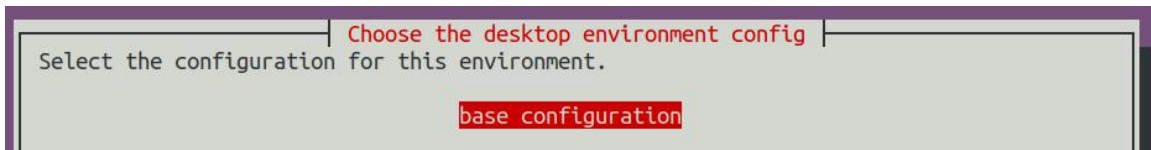
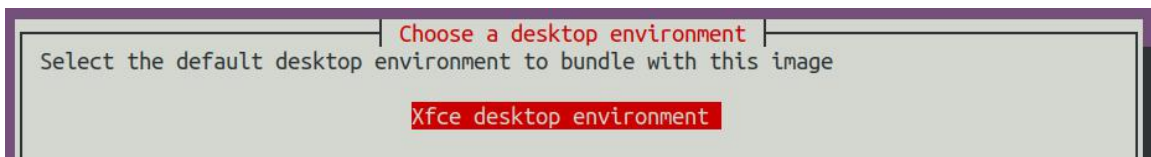
- a. **Image with console interface (server)** Indicates the image of the server version, which is relatively small in size
- b. **Image with desktop environment** Indicates a image with a desktop, which is relatively large in size



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



8) If you are compiling the desktop version of the image, you also need to select the type of desktop environment, but only XFCE is currently supported, so just press Enter.



9) You can then select additional packages that need to be installed. For example, if you need to install a browser, you can choose browsers. Which packages are included in each selection can be seen in the code of orangepi-build, you can also modify these configurations to add the packages you want to install

```

1. First enter external/config/desktop to see the desktop configuration folders of
different linux distributions. Note that not all the Orange Pi development boards that can
be seen in the code are supported and tested.
test@test:~/orangepi-build$ cd external/config/desktop
test@test:~/orangepi-build/external/config/desktop$ ls
bionic bookworm bullseye buster focal jammy README.md sid

2. Then select the type of distribution you want to view or modify, and enter the
corresponding directory, such as bullseys
    
```



```
test@test:~/orange-pi-build/external/config/desktop$ cd bullseye
```

```
test@test:~/orange-pi-build/external/config/desktop/bullseye$ ls
```

```
appgroups  environments
```

3. Then enter the **appgroups** directory to see all app groups

```
test@test:~/orange-pi-build/external/config/desktop/bullseye$ cd appgroups
```

```
test@test:~/orange-pi-build/external/config/desktop/bullseye/appgroups$ ls
```

```
browsers  chat  desktop_tools  editors  games  internet  multimedia  office
programming  remote_desktop
```

4. Open the **packages** file under different groups to view the software contained in the group

```
test@test:~/orange-pi-build/external/config/desktop/bullseye/appgroups$ cat
```

```
programming/packages
```

```
geany
```

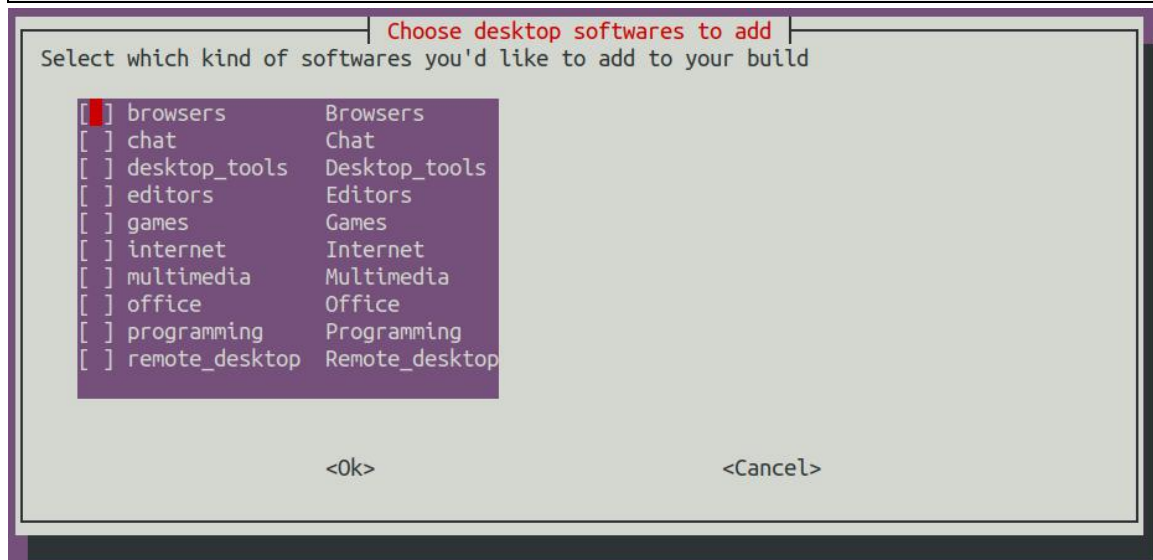
```
thonny
```

```
test@test:~/orange-pi-build/external/config/desktop/bullseye/appgroups$ cat
```

```
office/packages
```

```
ibreoffice
```

5. Except for the browser, the Linux image released by Orange Pi does not choose to install other software packages here. This is mainly to reduce the size of the Linux image.





10) Then the linux image will be compiled. The general process of compilation is as follows

- a. Initialize the compilation environment of the Ubuntu PC and install the software packages required for the compilation process
- b. Download the source code of u-boot and linux kernel (if cached, only update the code)
- c. Compile u-boot source code and generate u-boot deb package
- d. Compile the linux source code to generate linux-related deb packages
- e. Make a deb package of linux firmware
- f. Make the deb package of the orangepi-config tool
- g. Make board-level supported deb packages
- h. If you are compiling the desktop version of the image, you will also create a desktop-related deb package
- i. Check whether the rootfs has been cached, if there is no cache, then recreate the rootfs, if it has been cached, directly decompress and use
- j. Install the deb package generated earlier into rootfs
- k. Make some specific settings for different development boards and different types of images, such as pre-installing additional software packages, modifying system configuration, etc.
- l. Then make an image file and format the partition, the default type is ext4
- m. Copy the configured rootfs to the mirrored partition
- n. then update initramfs
- o. Finally, write the bin file of u-boot into the image through the dd command

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

- a. The storage path of the compiled image

```
[ o.k. ] Done building
[ output/images/Orangepi4-lts_2.2.0_debian_bullseye_linux5.10.43_xfce_desktop/Orangepi4-lts_2.2.0_debian_bullseye_linux5.10.43_xfce_desktop.img ]
```

- b. Compilation time used

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

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

```
[ o.k. ] Repeat Build Options [ sudo ./build.sh BOARD=orangepi4-lts BRANCH=current BUILD_OPT=image RELEASE=bullseye
```



```
BUILD_MINIMAL=no BUILD_DESKTOP=no KERNEL_CONFIGURE=yes ]
```



6. Instructions for using Android SDK

The compilation of Android SDK is carried out on a PC with Ubuntu 14.04 installed. Other versions of Ubuntu systems may have some differences. The image download address of Ubuntu14.04 amd64 version is as follows

<https://repo.huaweicloud.com/ubuntu-releases/14.04/ubuntu-14.04.6-desktop-amd64.iso>

6. 1. Download the source code of Android SDK

1) First download the Android SDK's sub-volume compressed package from the Google disk

Name	Owner	Last modified	File size
RK3399-Android8.1.tar.gz.md5sum	me	8 Mar 2022 me	244 bytes
RK3399-Android8.1.tar.gz00	me	8 Mar 2022 me	4 GB
RK3399-Android8.1.tar.gz03	me	8 Mar 2022 me	3.91 GB
RK3399-Android8.1.tar.gz02	me	9 Mar 2022 me	4 GB
RK3399-Android8.1.tar.gz01	me	9 Mar 2022 me	4 GB

2) After downloading the compressed package of Android SDK, please check whether the MD5 checksum is correct. If it is not correct, please download the source code again.

```
test@test:~$ md5sum -c RK3399-Android8.1.tar.gz.md5sum
```

```
RK3399-Android8.1.tar.gz00: OK
```

```
RK3399-Android8.1.tar.gz01: OK
```

```
RK3399-Android8.1.tar.gz02: OK
```

```
RK3399-Android8.1.tar.gz03: OK
```

3) Then you need to combine multiple compressed files into one, and then decompress

```
test@test:~$ cat RK3399-Android8.1.tar.gz* > RK3399-Android8.1.tar.gz
```

```
test@test:~$ tar xvf RK3399-Android8.1.tar.gz
```



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) 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. Compiling u-boot

- 1) The compilation method of u-boot is as follows

```
test@test:~$ cd RK3399-Android8.1
test@test:~/RK3399-Android8.1$ ./make.sh -B
```

- 2) After the compilation is successful, the output content is as follows



```
out:trust.img
merge success(trust.img)
load addr is 0x200000!
pack input u-boot.bin
pack file size: 682652
crc = 0xc21153c6
pack uboot.img success!
```

6. 3. 2. Compile the kernel

1) The compilation method of the kernel is as follows

```
test@test:~/RK3399-Android8.1$ ./make.sh -K
```

2) After the compilation is successful, the output content is as follows

```
scripts/kconfig/conf --silentoldconfig Kconfig
CHK      include/config/kernel.release
CHK      include/generated/uapi/linux/version.h
.....
make[2]: "include/generated/vdso-offsets.h" is the latest.
Building modules, stage 2.
MODPOST 13 modules
Pack to resource.img succeeded!
Image:  resource.img (with rk3399-orangepi-4-lts.dtb logo.bmp logo_kernel.bmp) is
ready
Image:  boot.img (with Image resource.img) is ready
```

6. 3. 3. Compile android

1) The compilation method of android is as follows

```
test@test:~/RK3399-Android8.1$ ./make.sh -A
```

2) After the compilation is successful, the output content is as follows

```
Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
.....
```



```
[100% 129/129] Install system fs image: out/target/product/rk3399_mid/system.img
out/target/product/rk3399_mid/system.img+out/target/product/rk3399_mid/obj/PACKAG
ING/recovery_patch_intermediates/recovery_from_boot.p          maxsize=2740531200
blocksize=135168 total=1198447603 reserve=27709440

##### build completed successfully (03:53 (mm:ss)) #####
```

6.3.4. Packaging the full image

1) The android image packaging method is as follows

```
test@test:~/RK3399-Android8.1$ ./make.sh -M -u
```

2) After the compilation is successful, the output content is as follows

```
create uboot.img...done.
create trust.img...done.
create loader...done.
.....
Make firmware OK!
----- OK -----
*****RKImageMaker ver 1.63*****
Generating new image, please wait...
Writing head info...
Writing boot file...
Writing firmware...
Generating MD5 data...
MD5 data generated successfully!
New image generated successfully!
Making update.img OK.
```

3) After the compilation is completed, the generated image file will be placed under **rockdev/Image-rk3399_mid/**. Where update.img is the Android firmware that can be burned and run

```
test@test:~/RK3399-Android8.1$ cd rockdev/Image-rk3399_mid/
test@test:~/RK3399-Android8.1$ ls update*
update.img
```

