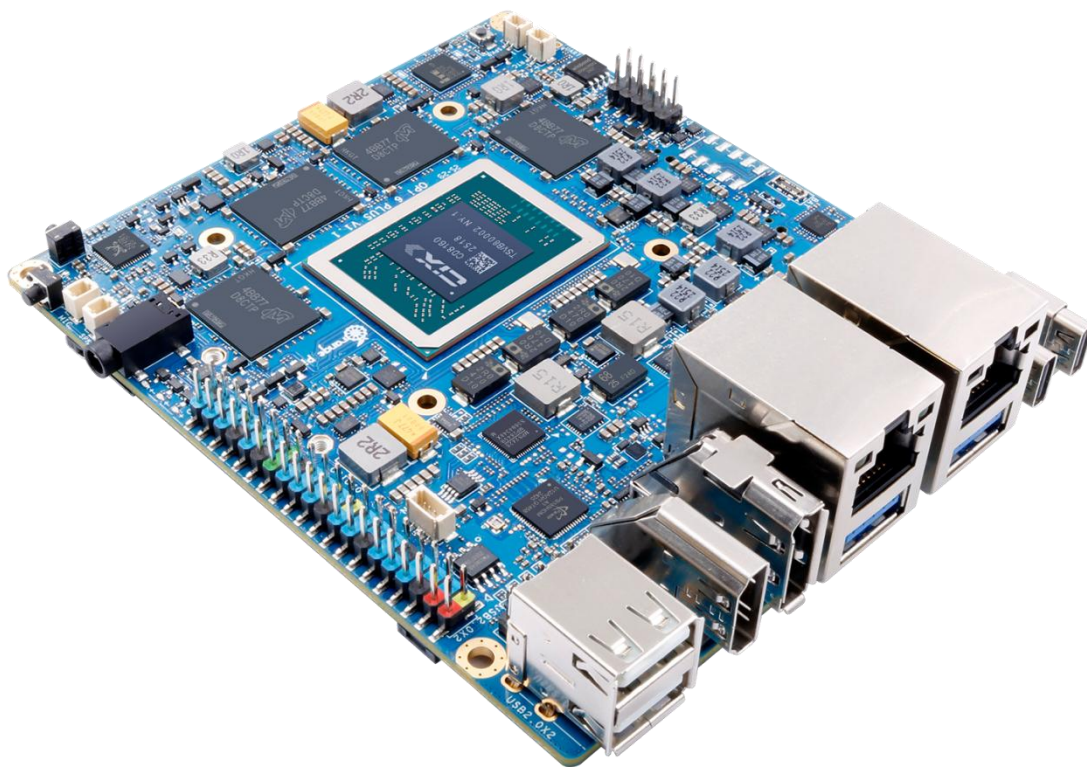




Orange Pi 6 Plus

Linux System User Manual





Catalogue

1. Basic features of Orange Pi 6 Plus	1
1. 1. What is Orange Pi 6 Plus	1
1. 2. Purpose of Orange Pi 6 Plus	1
1. 3. Hardware specifications of Orange Pi 6 Plus	2
1. 4. Top and Bottom Views of Orange Pi 6 Plus	4
1. 5. Interface Details of Range Pi 6 Plus	5
2. Introduction to using the development board	7
2. 1. Prepare the necessary accessories	7
2. 2. Download the image of the development board and related materials	14
2. 3. Method for burning Linux images to storage devices based on Windows PC	14
2. 3. 1. Method of burning Linux images using Win32Diskimager	15
2. 4. Method for burning Linux images to storage devices based on Ubuntu PC	18
2. 5. Method of burning BIOS image to storage device based on Windows PC	25
2. 6. Method for burning BIOS images to storage devices based on Linux system	28
2. 7. Method of burning BIOS firmware to SPI Flash on the development board	35
2. 8. Launch the Orange Pie development board	40
2. 9. 10pin Debugging Serial Port Usage Method	43
2. 9. 1. 10Pin Debugging Serial Port Pin Instructions	43
2. 9. 2. Connection instructions for debugging serial ports corresponding to BIOS and kernel	44
2. 9. 3. Ubuntu Platform Debugging Serial Port Usage Method	45
2. 9. 4. Instructions for Debugging Serial Ports on Windows Platform	49
3. Instructions for using the Debian Gnome desktop system	52
3. 1. Supported Debian image types and kernel versions	52
3. 2. Linux System Adaptation Status	52
3. 2. 1. Linux6.1 System Adaptation Status	52



3. 3.	Explanation of Linux Command Format in This Manual	54
3. 4.	Linux system login instructions	56
3. 4. 1.	Linux system default login account and password	56
3. 4. 2.	Linux Desktop System Automatic Login Instructions	56
3. 5.	Test instructions for onboard LED lights	57
3. 6.	Network connection test	58
3. 6. 1.	5G Ethernet Port Test	58
3. 6. 2.	Wi-Fi connection test	60
3. 6. 3.	How to set a static IP address	69
3. 7.	How to use the M.2 E-Key PCIe Wi-Fi 6+ Bluetooth module	77
3. 8.	SSH remote login to the development board	79
3. 8. 1.	SSH remote login to the development board under Ubuntu	79
3. 8. 2.	SSH remote login to the development board under Windows	80
3. 9.	How to upload files to the Linux system of the development board	82
3. 9. 1.	How to upload files from Ubuntu PC to the Linux system of the development board	82
3. 9. 2.	How to upload files from Windows PC to the Linux system of the development board	86
3. 10.	Display function test	90
3. 10. 1.	HDMI display output test	90
3. 10. 2.	DP display output test	91
3. 10. 3.	Type-C to HDMI display output test	92
3. 10. 4.	How to set the display resolution	92
3. 10. 5.	Instructions for using the EDP screen	94
3. 11.	How to use Bluetooth	95
3. 11. 1.	How to test Bluetooth on the desktop	96
3. 12.	USB interface test	98
3. 12. 1.	9-pin pin header to expand two USB2.0 ports	98
3. 12. 2.	Test method for connecting a USB mouse or keyboard	99
3. 12. 3.	Test method for connecting USB storage devices	99
3. 12. 4.	USB Camera Test Method	100



3. 12. 5.	Type-C USB0 Device Test Method	103
3. 12. 6.	USB Wireless Network Card Test	106
3. 12. 7.	How to use the TF card slot	110
3. 13.	Audio Test	111
3. 13. 1.	Test Method for Audio Playback via Type-C Interface	111
3. 13. 2.	Test method for DP interface audio playback	113
3. 13. 3.	HDIM Interface Audio Playback Testing Methods	115
3. 13. 4.	Test Method for Audio Playback via Headphone Interface	117
3. 13. 5.	Test method for audio playback of speaker interface	118
3. 13. 6.	Test Method for Audio Recording via Headphone Interface	121
3. 13. 7.	Test Method for MIC Interface Audio Recording	122
3. 14.	Temperature sensor	123
3. 15.	40 Pin Extension Interface Pin Usage Instructions	125
3. 16.	40 Pin Interface GPIO, I2C, UART, SPI and PWM Test Methods	128
3. 16. 1.	40 Pin GPIO Test Method	128
3. 16. 2.	40 Pin Interface SPI Test Method	131
3. 16. 3.	40-pin I2C interface test method	132
3. 16. 4.	40 Pin Interface UART Test Method	134
3. 16. 5.	Testing method for 40-pin interface PWM	136
3. 17.	Test of some programming languages supported by Linux system	139
3. 17. 1.	Debian Bookworm System	139
3. 18.	How to install kernel header files	141
3. 19.	Test Methods for OV13850 and OV13855 MIPI Cameras	143
3. 20.	How to use RTC	149
3. 21.	GPU testing methods	150
3. 22.	Testing Methods for Gstreamer Video Hardware Encoding	159
3. 23.	Testing Methods for Video Players and Gstreamer Video Hardware Decoding	162
3. 24.	Testing Methods for FFmpeg Video Hardware Encoding	167
3. 25.	Testing Methods for FFmpeg Video Hardware Decoding	169
3. 26.	Test Method for MPV Video Hardware Decoding	173



3. 27. Testing Methods for Chromium Browser Hardware-Decoded Video Playback	177
3. 28. How to set up the Chinese environment and install the Chinese input method	179
3. 29. How to install the Baota Linux Panel	186
3. 30. QT installation method	190
3. 31. How to install Docker	195
3. 32. How to install and use CasaOS	196
3. 32. 1. How to install CasaOS	196
3. 32. 2. Usage of CasaOS	198
3. 33. Method to view SoC chip serial number	206
3. 34. Usage of ZFS file system	206
3. 34. 1. Method of installing ZFS	206
3. 34. 2. Method for Creating ZFS Pool	209
3. 34. 3. Testing the Data De duplication Function of ZFS	210
3. 34. 4. Testing the Data Compression Function of ZFS	211
3. 35. Method for Setting Swap Memory	213
3. 36. Installation Method for Spark Store	214
3. 37. Usage of CIX AI Model Hub	219
3. 37. 1. Method for downloading CIX AI Model Hub	219
3. 37. 2. Installing Dependency Packages for NPU	222
3. 37. 3. Method for installing onnxruntime_zhouyi	223
3. 37. 4. List of Tested Models	223
3. 37. 5. Testing Method for Speech_Secognition Class Models in Audio	226
3. 37. 6. Testing Methods for BEV Models in ComputeVision	230
3. 37. 7. Testing Methods for Body-Analysis Models in ComputeVision	232
3. 37. 8. Testing Methods for Depth-Estimation Class Models in ComputeVision	234
3. 37. 9. Testing Methods for Face_Setection Class Models in ComputeVision	240
3. 37. 10. Testing Method for Face_Secognition Class Models in ComputeVision	248
3. 37. 11. Testing Method for Image_Classification Class Models in ComputeVision	251



3. 37. 12. Testing Method for Image-Enhancement Class Models in ComputeVision	268
3. 37. 13. Testing Method for LaneDetected Class Models in ComputeVision	275
3. 37. 14. Testing Method for Object Detection Class Models in ComputeVision	285
3. 37. 15. Testing Method for OCR Models in ComputeVision	312
3. 37. 16. Testing Method for Pose_Estimation Class Models in ComputeVision	317
3. 37. 17. Testing Method for Semantic_Segmentation Class Models in ComputeVision	325
3. 37. 18. Testing Method for Super_Resolution Class Models in ComputeVision	345
3. 38. Usage of NOE SDK	351
3. 38. 1. Application Method for NOE SDK	351
3. 38. 2. Installation Method of NOE SDK	352
3. 38. 3. Example of using NOE SDK	357
3. 39. Methods for Installing Anaconda	365
3. 40. How to check the BIOS firmware version	368
3. 41. Methods for shutting down and restarting the development board	368
4. Linux SDK——orange-pi-build usage instructions	370
4. 1. Compilation System Requirements	370
4. 1. 1. Compile on Ubuntu22.04 computer using x64	370
4. 2. Obtain the source code of linux sdk	372
4. 2. 1. Download orange-pi-build	372
4. 2. 2. Download the cross compilation toolchain	374
4. 2. 3. Explanation of the complete directory structure of orange-pi-build	375
4. 3. Compiling Linux Kernel	376
4. 4. Compile rootfs	382
4. 5. Compiling Linux Images	386
5. Appendix	391
5. 1. User Manual Update History	391
5. 2. Image update history	391



1. Basic features of Orange Pi 6 Plus

1.1. What is Orange Pi 6 Plus

The Orange Pi 6 Plus adopts Cix's new generation 12 core 64 bit ARMv9 processor, specifically a quad core large core Cortex®-A720、 Four core Core Cortex®-A720 and quad core small core Cortex®-A520, Main frequency of 2.6GHz, integrated with Arm® Immortals™ G720 MC10 GPU has a total computing power of approximately 45 Tops with CPU+NPU+GPU, and can be equipped with 16GB, 32GB, and 64GB of memory, providing up to 4K display processing capability.

The Orange Pi 6 Plus features a rich array of interfaces, including one HDMI output, one DP output, two USB-C/DP ports, an eDP display interface, two M.2 M-key PCIe4.0x4 slots, an M.2 E-key PCIe4.0x1 slot, two 5G Ethernet ports, USB2.0 and USB3.0 ports, a headphone jack, a microphone jack, a speaker jack, an RTC interface, and a 40-pin expansion header. It is widely applicable in high-end tablets, edge computing, artificial intelligence, cloud computing, AR/VR, smart security, smart home, and other fields, covering all industries of AIoT.

1.2. Purpose of Orange Pi 6 Plus

We can use it to achieve:

- A high-performance Linux desktop computer
- AI development workstation
- Android game console
- OpenHarmony computers, etc

Of course, there are many other features as well. With a powerful ecosystem and various expansion accessories, Orange Pi can help users easily achieve delivery from creativity to prototype to mass production. It is an ideal creative platform for makers, dreamers, and hobbyists.



1. 3. Hardware specifications of Orange Pi 6 Plus

Orange Pi 6 Plus hardware specifications	
CPU	• 12 core 64 bit processor • Four core large core Cortex®-A720 • Four core Core Cortex®-A720 • Four core small core Cortex®-A520
GPU	Arm® Immortals™ G720 MC10 GPU, Support: • Hardware ray tracing • OpenGL® ES3.2 • OpenCL® 3.0 • Vulkan® 1.3
NPU	• Supports INT4/INT8/INT16/FP16/BF16 and TF32 acceleration • With a computing power of up to 28.8TOPs
Video decoding	Supports AV1/H.265/H.264/VP9/VP8/H.263/MPEG-4/MPEG-2 formats
Video encoding	Supports H.265/H.264/VP9/VP8 formats
Memory	• 128-bit LPDDR5 type • 16GB, 32GB, and 64GB optional capacities • 6000MT/s speed
Storage interface	• 1 x SPI Nor FLASH: 64Mbit • 1 x MicroSD card slot • 2 x NVMe SSD (PCIe4.0x4) M.2 M-Key 2280 slots
USB port	• 2 x USB3.0 Host • 2 x USB2.0 Host • 9pin reserved pin: 2 x USB2.0 Host • 1 x Type-C USB3.0 Host • 1 x Type-C USB3.0 OTG, Support Host and Device
Video output	• 1 x HDMI, Up to 4Kp60 • 1 x DisplayPort, Up to 4Kp120 • 2 x USB-C DP, Up to 4Kp60 • 1 x eDP, Up to 4Kp60
Camera	2 x MIPI CSI 4 Lane interfaces
TP interface	6pin 0.5mm spacing FPC socket

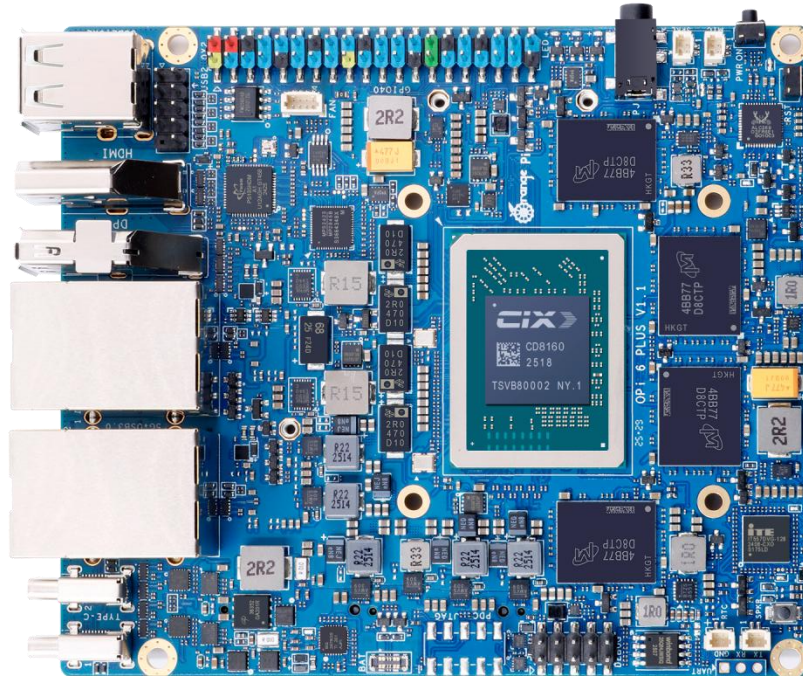


audio	• 1 x 3.5mm headphone jack audio input/output • 1 x HDMI audio output • 1 x DisplayPort audio output • 2 x USB-C DP audio outputs • 1 x MIC audio input interface (2pin 1.0mm specification) • 2 x speaker audio output interface (2pin 1.0mm specification)
Ethernet	2 x PCIe 1G/2.5G/5G Ethernet ports (RTL8126)
40Pin expansion port	Used for expanding UART, I2C, SPI, PWM, GPIO interfaces
PCIe M.2 E-KEY	Includes PCIe 4.0 x 1/USB2.0 interfaces and supports 2230 Wi-Fi 6/BT modules Includes PCIe 4.0 x 1/USB2.0 interfaces and supports 2230 Wi-Fi 6 /BT modules
Button	1 x Fastboot key, 1 x Reset key, 1 x Power on/off key
Power supply	• 2 x Type-C PD 20V 100W power input • Board to board battery interface, 12V, three string
LED lights	• 1 x red LED light, used to indicate battery charging status • 1 x blue LED light, used to indicate the Type-C power supply status • 1 x green LED light, used to indicate the operating status of the system
Fan interface	4pin 1.0mm spacing specification, used to connect 5V fan, supports PWM control switch and speed
RTC battery interface	2pin 1.0mm specification, used to power the RTC module
10pin debugging serial port	10pin debugging of serial port, including: • UART2, BIOS and kernel logs • UART4, PM log • UART5, SE log • UART6, Post code log
Supported operating systems	Debian, Ubuntu, OpenHarmony, Android, and Windows, etc
Introduction to appearance specifications	
Product dimensions	115mm * 100mm
weight	132g
 rangePi™ is a registered trademark of Shenzhen Xunlong Software Co., Ltd	

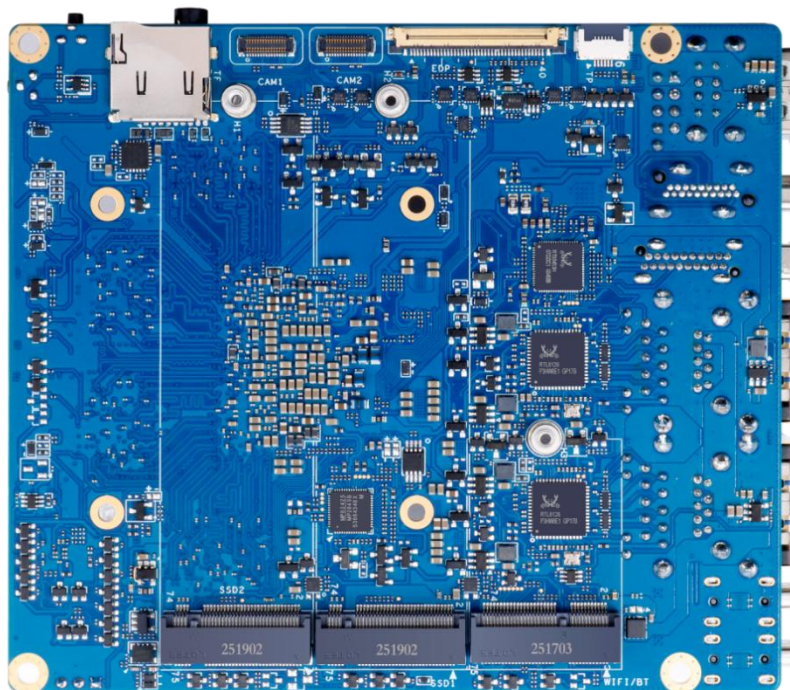


1. 4. Top and Bottom Views of Orange Pi 6 Plus

Top level view:



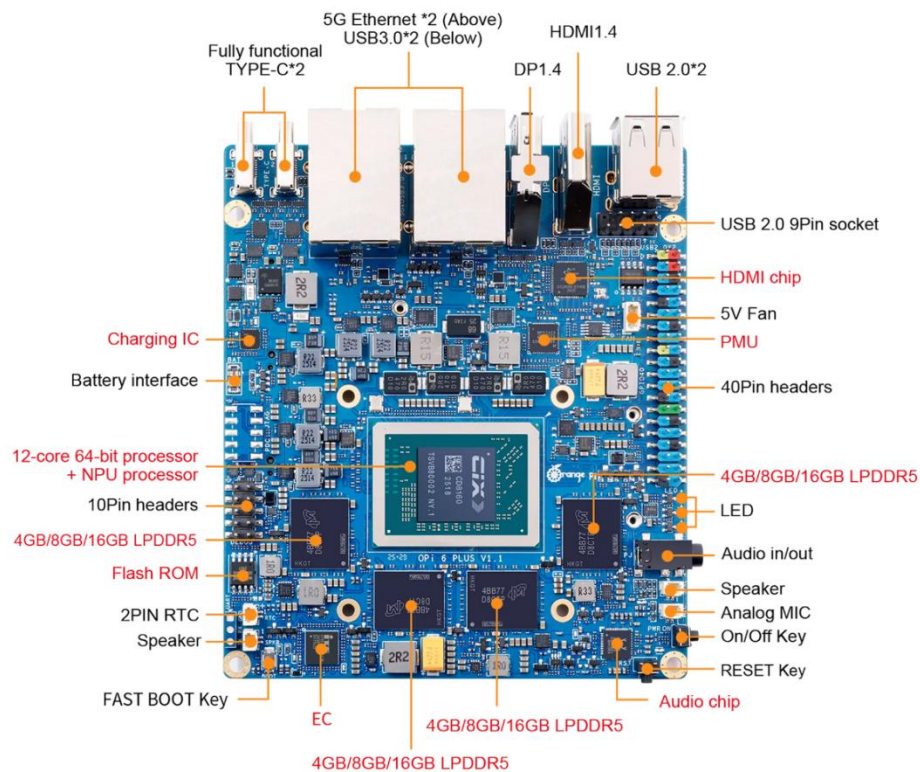
Bottom level view:



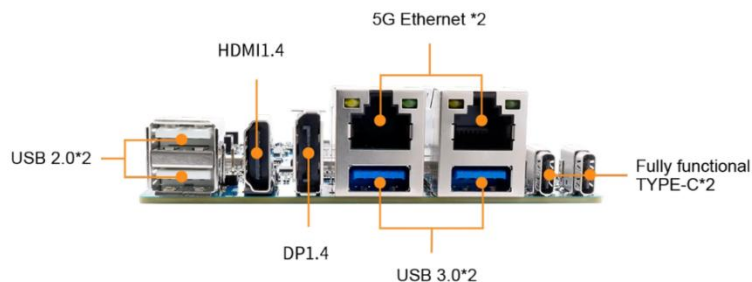


1. 5. Interface Details of Range Pi 6 Plus

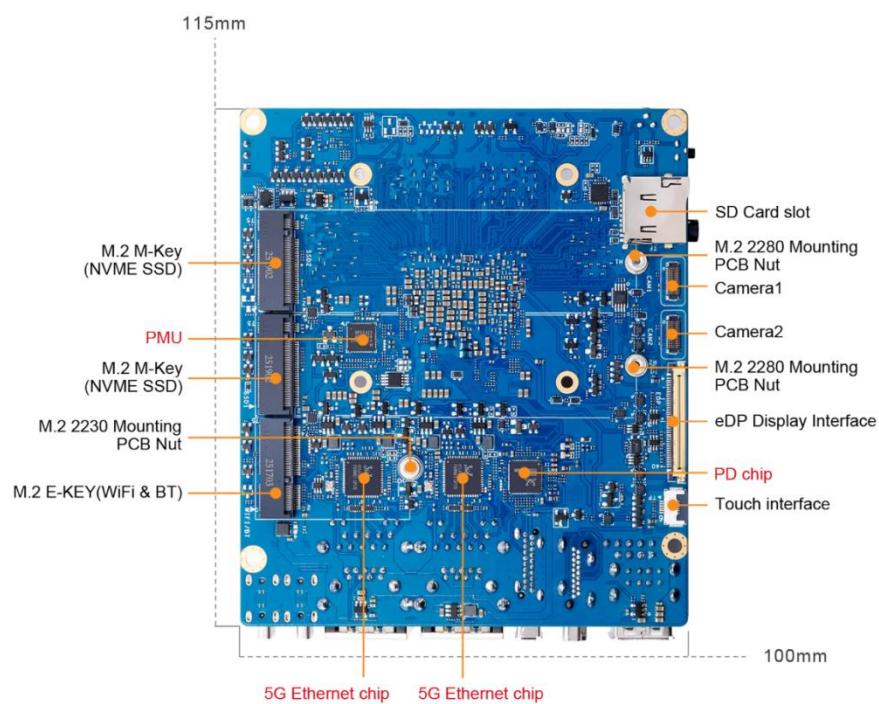
Product display



Top View



Side View



Bottom View



2. Introduction to using the development board

2.1. Prepare the necessary accessories

- 1) TF card, a high-speed flash card with a minimum capacity of 32GB (recommended 64GB or above) and **class10** or above.



- 2) TF card reader, used to burn images onto TF cards.



- 3) HDMI interface display.



- 4) HDMI to HDMI connection cable, connect the development board to an HDMI monitor or TV for display through the HDMI interface.



5) DP to HDMI connection cable, connect the development board to an HDMI monitor or TV for display through the DP interface.



6) Type-C to HDMI cable, connect the development board to an HDMI monitor or TV for display through the Type-C interface.



7) Type-C to USB adapter, used to connect USB storage devices or USB devices such as mice and keyboards through the Type-C interface.



8) eDP screen, used to display the system interface of the development board.



9) 9pin USB2.0 adapter cable, used to expand two USB2.0 Host interfaces.



10) PCIe Wi-Fi Bluetooth module, such as RTL8852BE module.



11) USB camera.

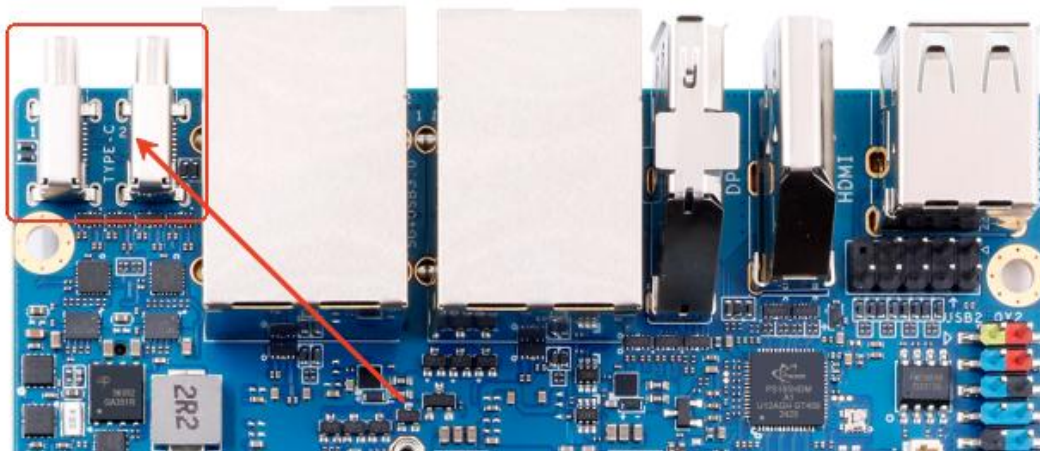




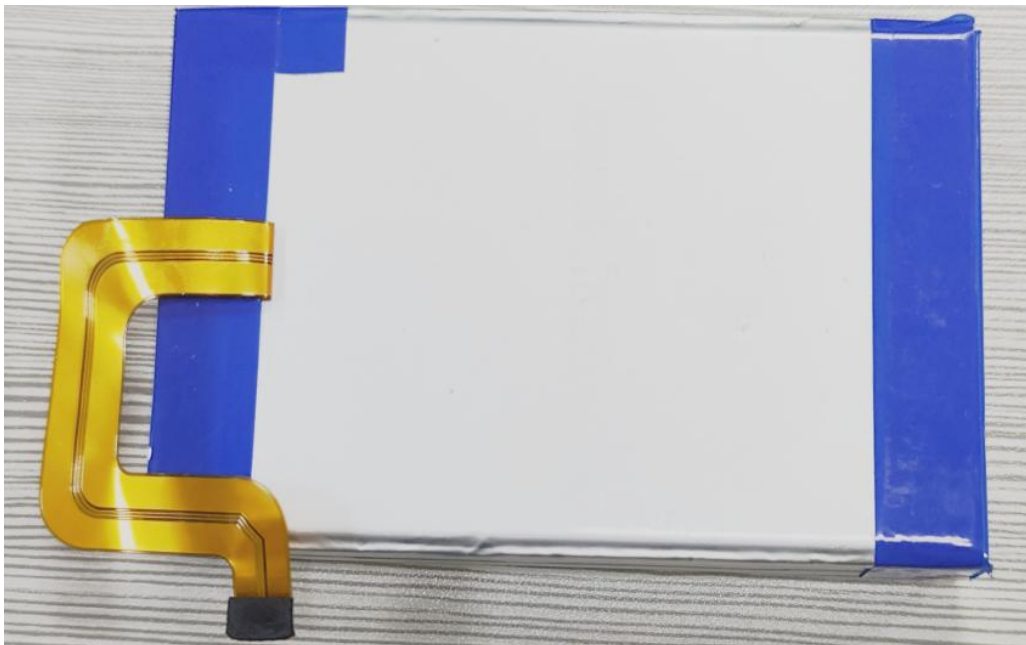
12) The Orange Pi 6 Plus power adapter is recommended to be powered by a 20V 100W Type-C PD power supply.



There are two Type-C interfaces on the development board, both of which support supplying power to the board.



13) Battery used to power the development board

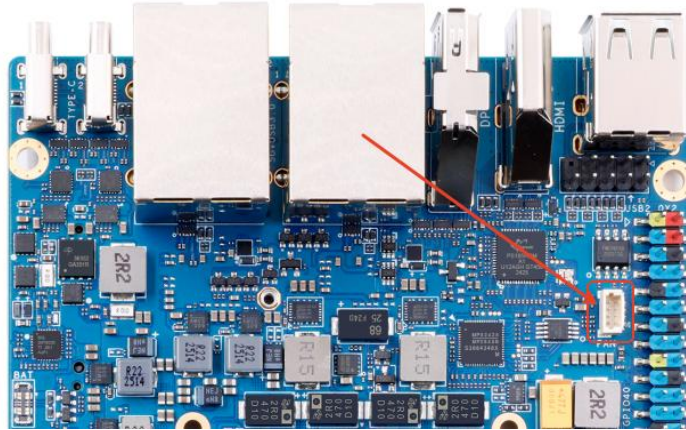


14) A USB interface mouse and keyboard, as long as it is a standard USB interface mouse and keyboard, can be used to control the Orange Pi development board.

15) 5V cooling fan. As shown in the figure below, there is an interface on the development board for connecting PWM cooling fans, with a specification of **4pin** **1.0mm** spacing.

The fan on the development board can be adjusted for speed and on/off through

PWM.



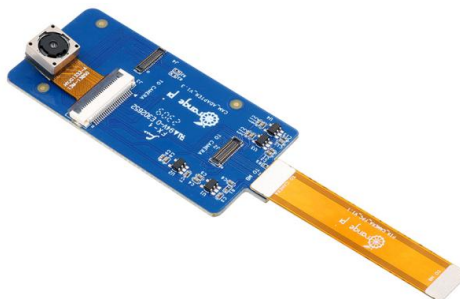
16) Ethernet cable, used to connect the development board to the internet.



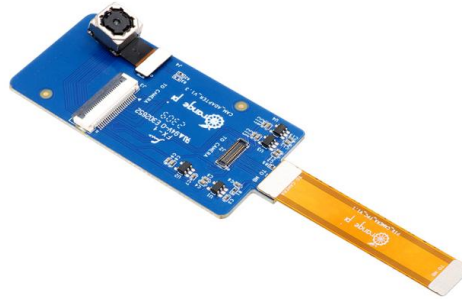
17) The data cable of Type-C interface is used for using the functions of Type-C USB devices.



18) OV13850 camera with 13 million MIPI interface.



19) OV13855 camera with 13 million MIPI interface.



20) The NVMe SSD solid state drive with M.2 M-KEY 2280 specification has a PCIe interface specification of PCIe4.0x4.



21) RTC battery, interface is 2pin, 1.0mm spacing.



The location of the RTC battery interface on the development board is shown in the following figure:



22) When using the serial port debugging function, a 3.3V USB to TTL module and



DuPont cable are required to connect the development board and computer.



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

1) The download link for the English version of the material is:

<http://www.orangepi.org/html/hardWare/computerAndMicrocontrollers/service-and-support/Orange-Pi-6-Plus.html>

2) The information mainly includes:

- a. **User manual and schematic diagram:** Save on Google Cloud Drive.
- b. **Official tools:** This mainly includes the software required during the use of the development board.
- c. **Linux source code:** Save on Github and Gitee.
- d. **Ubuntu image:** Save on Google Cloud Drive.
- e. **Debian image:** Save on Google Cloud Drive.
- f. **BIOS image:** Save on Google Cloud Drive.
- g. **Android image:** Save on Google Cloud Drive.
- h. **Orange Pi OS(OH) image:** Save on Google Cloud Drive.

2.3. Method for burning Linux images to storage devices based on Windows PC

Note that the Linux image referred to here specifically refers to Linux distribution images such as Debian and Ubuntu downloaded from the [Orange Pi data download page](#).

If you are using an image of the Linux6.6 ACPI kernel, please update the BIOS image according to the instructions in the section on [burning BIOS firmware to SPI Flash on the development board](#)



2. 3. 1. Method of burning Linux images using Win32Diskimager

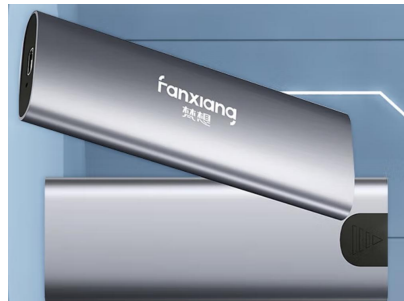
1) The development board supports three boot modes: TF card, USB, and NVMe SSD. We recommend burning the system onto an NVMe SSD for faster startup and better user experience.

2) Before burning the image, the storage device needs to be prepared first.

- a. TF card startup: A 32GB or larger TF card is required, and the transmission speed of the TF card must be at or above **class10**. It is recommended to use TF cards from brands such as SanDisk.
- b. USB boot: A USB flash drive is required, and it is recommended to use a USB3.0 interface USB flash drive for faster speed.
- c. NVMe SSD boot: A M-Key 2280 specification NVMe SSD solid state drive is required, and the PCIe interface specification for the M.2 slot on the development board is PCIe4.0x4. PCIe3.0 SSDs can also be used.

3) Then insert the storage device into the computer.

- a. TF card startup: A card reader is required to connect the TF card to the computer.
- b. USB boot: Usually, simply insert the USB flash drive into the computer.
- c. NVMe SSD startup: A hard drive enclosure similar to the one shown in the figure below is needed to connect the SSD to the computer.



4) Then format the storage device.

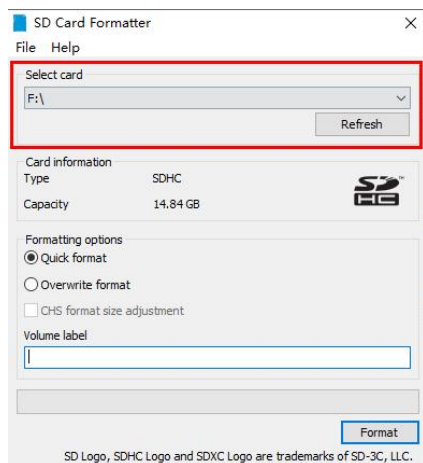
- a. You can use the **SD Card Formatter** software to format storage devices, and its download link is:

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

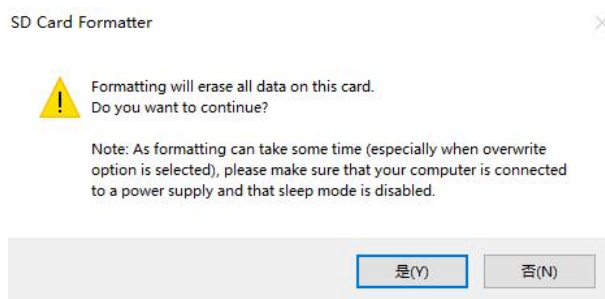
- b. After downloading, simply unzip and install it, and then open the software.
- c. If the computer only has one storage device inserted, the "**Select card**" column



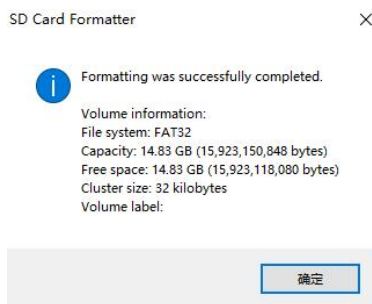
will display the drive letter corresponding to the storage device. If the computer has multiple USB storage devices inserted, you can select the drive letter corresponding to the storage device that needs to be burned through the drop-down menu.



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



- e. After formatting the storage device, the message shown in the figure below will pop up. Click OK to proceed.



- 5) Download the compressed file of the Linux operating system image that you want to



burn from [the Orange Pi download page](#), and then use decompression software to decompress it. In the decompressed file, the file ending with ".img" is the operating system image file, which is usually several GB or larger in size.

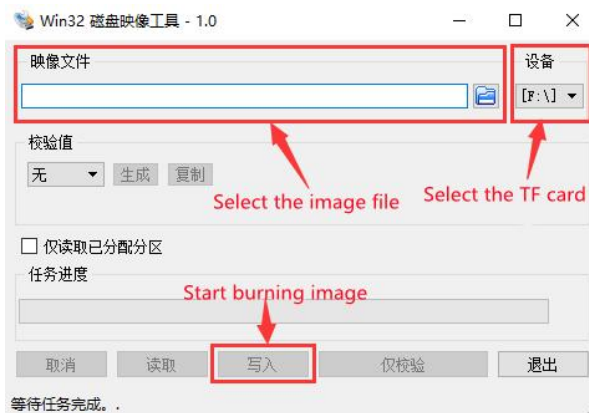
6) The method of burning Linux images to storage devices using **Win32Diskimager** is as follows:

a. The download page for Win32Diskimager is:

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

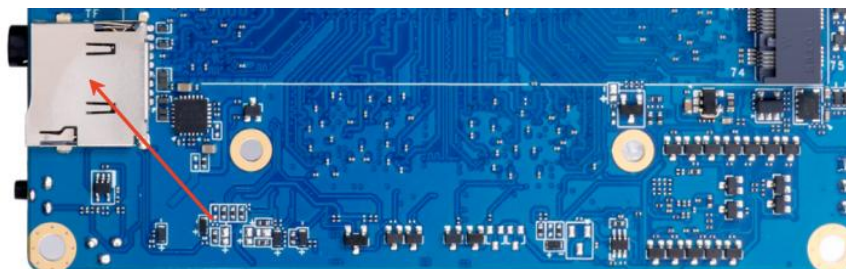
b. After downloading, simply install it directly. The interface of Win32Diskimager is shown below:

- Firstly, select the path of the image file;
- Then confirm that the drive letter of the storage device matches the one displayed in the "**Device**" column;
- Finally, click "**Write**" to start burning.



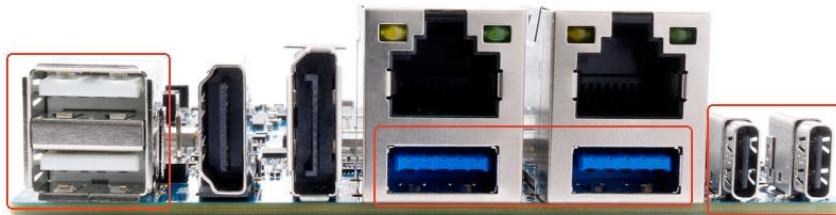
c. After the image writing is completed, click the "**Exit**" button to exit, and then you can unplug the storage device and plug it into the development board to start.

- TF card startup: The TF card with the burned image needs to be inserted into the TF card slot on the development board, located as follows:

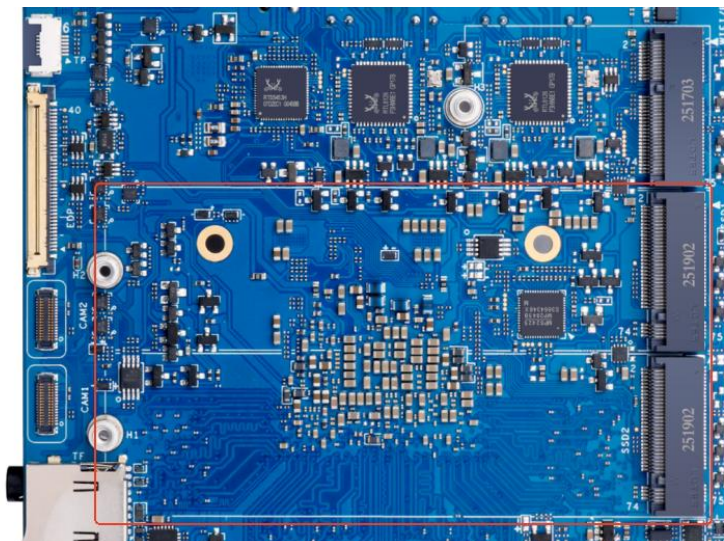




- b) USB boot: You need to insert the USB flash drive with the burned image into the USB 3.0 Host, USB 2.0 Host, or USB-C interface of the development board. Their locations are as follows:



- c) NVMe SSD boot: You need to insert the burned image SSD into the M.2 interface of the M-Key on the development board. The development board has two M.2 interfaces that can be connected to SSDs, both of which support SSD boot system. Therefore, you can choose one of the two, and its location is as follows:



2.4. Method for burning Linux images to storage devices based on Ubuntu PC

Note that the Linux image referred to here specifically refers to Linux distribution images such as Debian and Ubuntu downloaded from the [Orange Pi data download page](#), while Ubuntu PC refers to a personal computer with the Ubuntu system installed.

- 1) The development board supports three boot modes: TF card, USB, and NVMe SSD.



We recommend burning the system onto an NVMe SSD for faster startup and better user experience.

- 2) Before burning the image, the storage device needs to be prepared first.
 - a. TF card startup: A 32GB or larger TF card is required, and the transmission speed of the TF card must be at or above **class10**. It is recommended to use TF cards from brands such as SanDisk.
 - b. USB boot: A USB flash drive is required, and it is recommended to use a USB3.0 interface USB flash drive for faster speed.
 - c. NVMe SSD boot: A M-Key 2280 specification NVMe SSD solid state drive is required, and the PCIe interface specification for the M.2 slot on the development board is PCIe4.0x4. PCIe3.0 SSDs can also be used.
- 3) Then insert the storage device into the computer.
 - a. TF card startup: A card reader is required to connect the TF card to the computer.
 - b. USB boot: Generally, simply insert the USB drive into the computer.
 - c. NVMe SSD boot: A hard drive enclosure similar to the one shown in the figure below is required to connect the SSD to the computer.



- 4) Download the compressed file of the Linux operating system image that you want to burn from the [Orange Pi's information download page](#), as well as the verification and files for the compressed file.

- 5) You can first use the `sha256sum -c *.sha` command to calculate if the checksum is correct. If it prompts **OK**, it means that the downloaded image is correct and can be safely burned to the storage device. If it prompts **FAILED**, it means that the downloaded image has a problem and you should try downloading it again.

```
test@test:~$ sha256sum -c *.sha
```



```
Orangepi6plus_1.0.2_debian_bookworm_desktop_gnome_linux6.1.44.img.xz: OK
```

6) Then use decompression software to decompress the compressed image file. In the decompressed file, the file ending with ".img" is the operating system image file, which is usually several GB or larger in size. **balenaEtcher software burning can skip this step.**

a. The decompression command for the compressed file ending in xz is as follows:

```
test@test:~$ xz -dx Orangepi6plus_1.0.2_debian_bookworm_desktop_gnome_linux6.1.44.xz
test@test:~$ ls Orangepi6plus_1.0.2_debian_bookworm_desktop_gnome_linux6.1.44.*
Orangepi6plus_1.0.2_debian_bookworm_desktop_gnome_linux6.1.44.xz
Orangepi6plus_1.0.2_debian_bookworm_desktop_gnome_linux6.1.44.img    #image file
```

7) Download the balenaEtcher software from the following link:

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

8) After entering the balenaEtcher download page, clicking the green download button will jump to the software download location.



9) Then choose to download the Linux version of the software.

DOWNLOAD

Download Etcher

ASSET	OS	ARCH	
ETCHER FOR WINDOWS (X86 X64) (INSTALLER)	WINDOWS	X86 X64	Download
ETCHER FOR MACOS	MACOS	X64	Download
ETCHER FOR MACOS (ARM64)	MACOS	ARM64	Download
ETCHER FOR LINUX X64 (64-BIT) (ZIP)	LINUX	X64	Download
ETCHER FOR LINUX (LEGACY 32 BIT) (APPIMAGE)	LINUX	X86	Download

10) The downloaded balenaEtcher software package is shown below:

balenaEtcher-linux-x64-2.1.4.zip

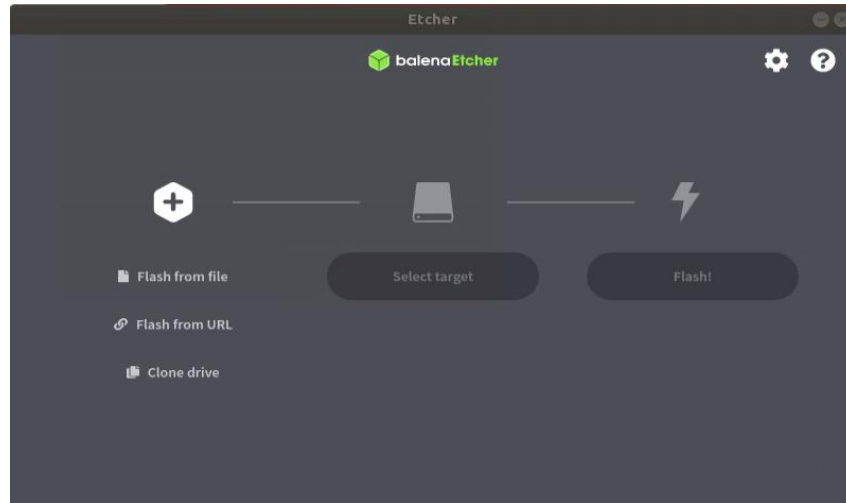
11) Then use the following command to decompress the zip file of balenaEtcher. After decompression, an additional folder named **balenaEtcher-linux-x64** will be created.

```
test@test:~$ unzip balenaEtcher-linux-x64-2.1.4.zip
test@test:~$ ls
balenaEtcher-linux-x64  balenaEtcher-linux-x64-2.1.4.zip
```

12) Then, in the Ubuntu desktop, enter the **balenaEtcher-linux-x64** folder, and double-click balena-etcher to open the balanaEtcher burning program.



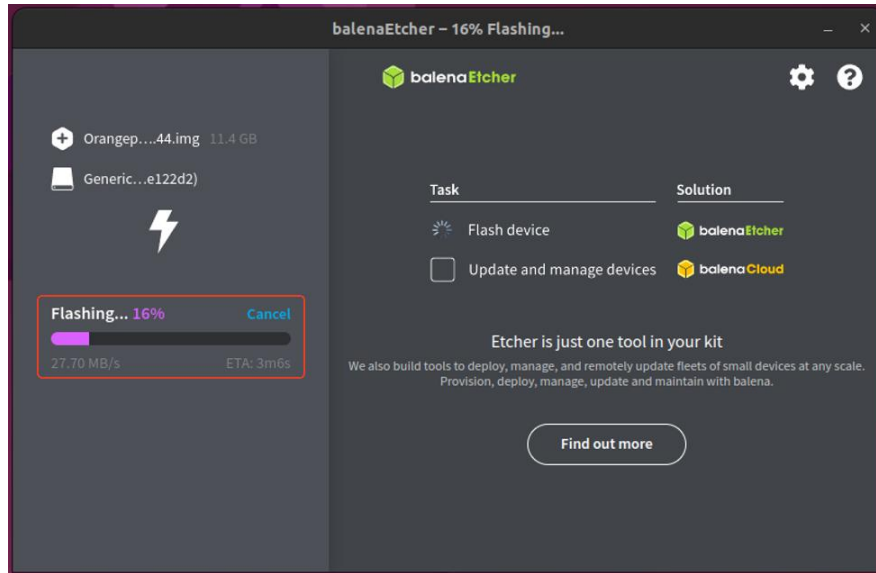
13) The interface displayed after opening balenaEtcher is shown in the following figure:



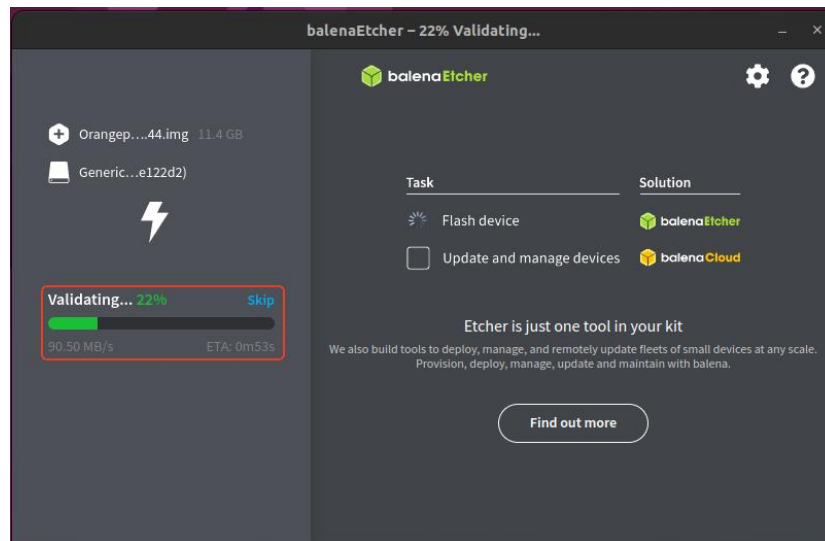
- 14) The specific steps for burning a Linux image using balenaEtcher are as follows:
- Firstly, select the path of the Linux image file to be burned;
 - Then select the drive letter corresponding to the storage device;
 - Finally, clicking Flash will start burning the Linux image to the storage device.



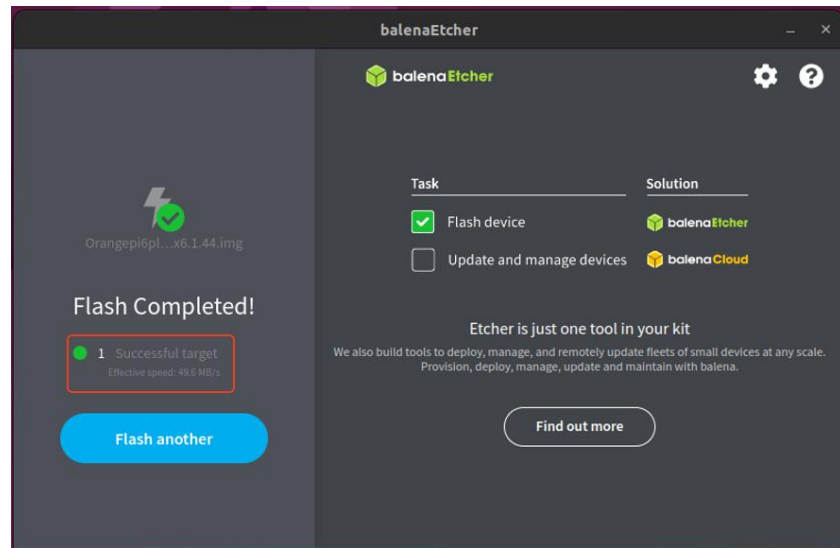
- 15) The interface displayed during the process of burning a Linux image by balenaEtcher is shown in the following figure. In addition, a purple progress bar indicates that the Linux image is being burned to the storage device.



16) After the Linux image is burned, balenaEtcher will also verify the image burned to the storage device by default to ensure that there are no problems during the burning process. As shown in the following figure, a green progress bar indicates that the image has been burned and balenaEtcher is verifying the burned image.

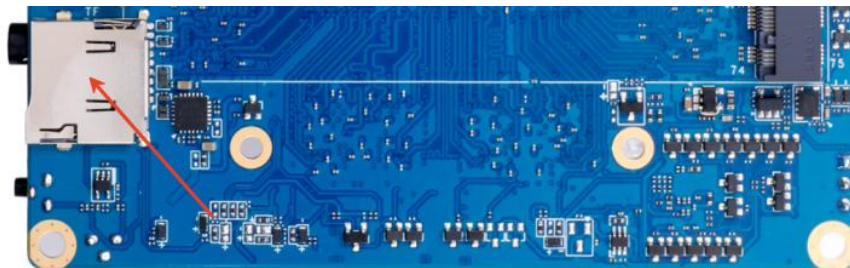


17) After successful burning, the display interface of balenaEtcher is shown in the following figure. If a green indicator icon is displayed, it indicates that the image burning is successful. At this time, you can exit balenaEtcher, then unplug the storage device and insert it into the development board for use.

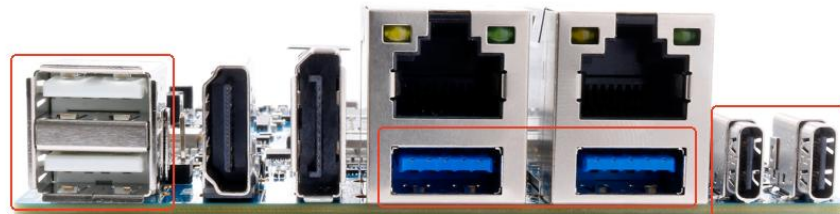


18) The way to insert different types of storage devices into the development board is as follows:

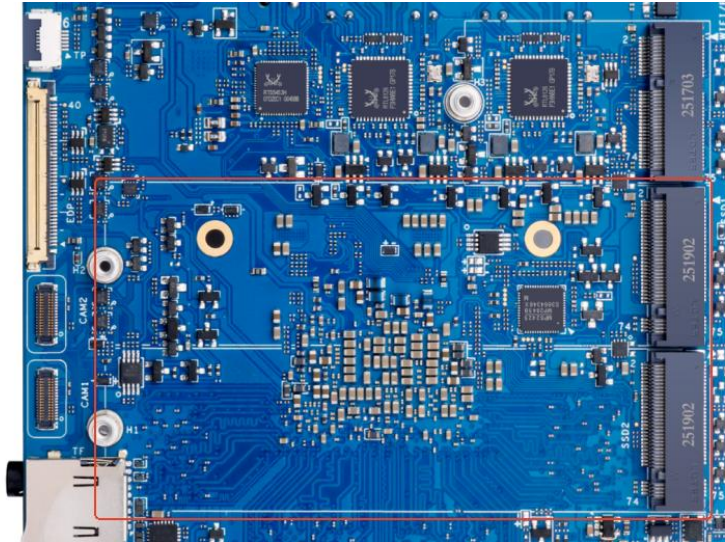
- a. TF card startup: The TF card with the burned image needs to be inserted into the TF card slot on the development board, located as follows:



- b. USB boot: You need to insert the USB flash drive with the burned image into the USB3.0 Host, USB2.0 Host, or USB-C interface of the development board. Their locations are as follows:



- c. NVMe SSD boot: You need to insert the burned image SSD into the M.2 interface of the M-Key on the development board. The development board has two M.2 interfaces that can be connected to SSD, both of which support SSD boot system. Therefore, you can choose one of the two, and its location is as follows:



2.5. Method of burning BIOS image to storage device based on Windows PC

1) Download the BIOS image required for Linux from the [download page of the development board](#). The steps are as follows:

a. Open the data download page of the development board, the link is as follows:

<http://www.orangepi.cn/html/hardWare/computerAndMicrocontrollers/service-and-support/Orange-Pi-6-Plus.html>

b. Then open the download link for BIOS.

Official Resources



User Manual

[Downloads](#)



Schematic

[Downloads](#)



Official Tools

[Downloads](#)



BIOS

[Downloads](#)

c. There are multiple versions of BIOS firmware in the BIOS link. Please download the BIOS image from the **Linux_Bios** folder.



d. The downloaded image file is shown below:

opi6plus_bios_image_for_linux_xxxxxxx.img

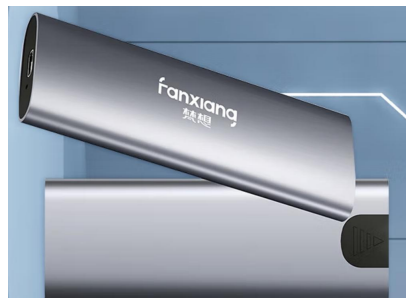
2) The development board supports three boot modes: TF card, USB, and NVMe SSD, and can burn BIOS to any storage device.

3) Before burning the image, the storage device needs to be prepared first.

- a. TF card startup: The BIOS image is very small and there is no requirement for the capacity of the TF card. The recommended transfer speed for the TF card is **class10** or above, and it is recommended to use TF cards from brands such as SanDisk.
- b. USB boot: A USB flash drive is required.
- c. NVMe SSD boot: A M-Key 2280 specification NVMe SSD solid state drive is required.

4) Then insert the storage device into the computer.

- a. TF card startup: A card reader is required to connect the TF card to the computer.
- b. USB boot: Generally, simply insert the USB drive into the computer.
- c. NVMe SSD boot: A hard drive enclosure similar to the one shown in the figure below is required to connect the SSD to the computer.





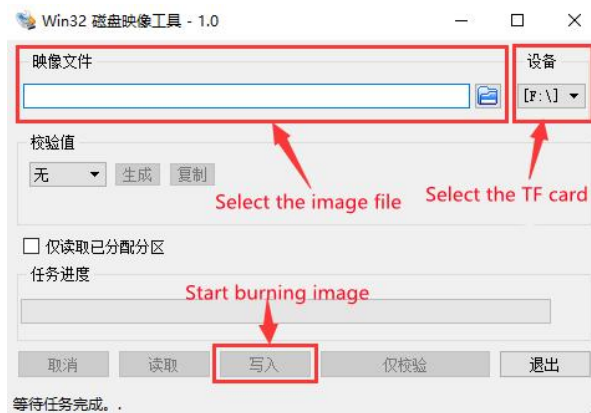
5) The method of burning BIOS images to storage devices using **Win32Diskimager** is as follows:

- a. The download page for Win32Diskimager is:

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

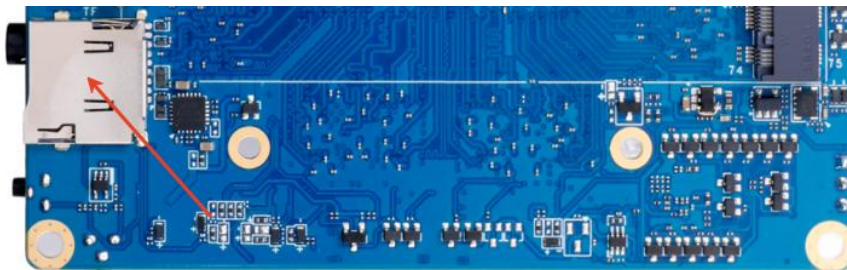
- b. After downloading, simply install it directly. The interface of Win32Diskimager is shown below:

- Firstly, select the path of the BIOS image file;
- Then confirm that the drive letter of the storage device matches the one displayed in the "**Device**" column;
- Finally, click "**Write**" to start burning.

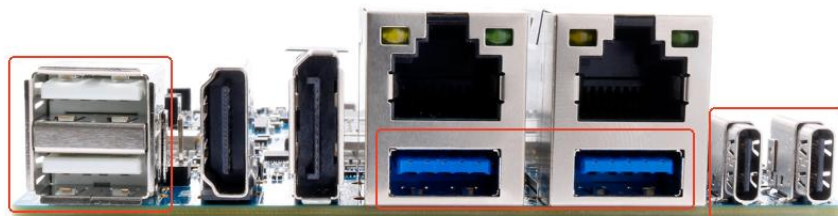


- c. After the image writing is completed, click the "**Exit**" button to exit, and then you can unplug the storage device and plug it into the development board to start.

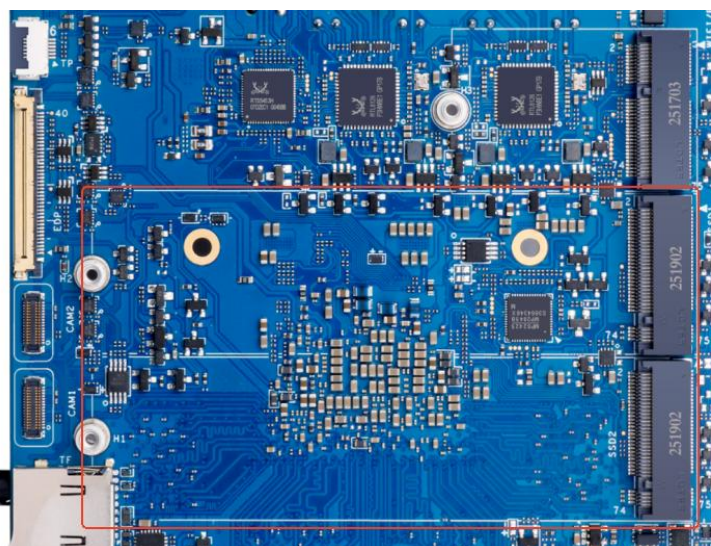
- a) TF card startup: The TF card with the burned image needs to be inserted into the TF card slot on the development board, located as follows:



- b) USB boot: You need to insert the USB flash drive with the burned image into the USB3.0 Host, USB2.0 Host, or USB-C interface of the development board. Their locations are as follows:



- c) NVMe SSD boot: You need to insert the burned image SSD into the M.2 interface of the M-Key on the development board. The development board has two M.2 interfaces that can be connected to SSDs, both of which support SSD boot system. Therefore, you can choose one of the two, and its location is as follows:



- 6) Then please follow the instructions in the section on [burning BIOS firmware to SPI Flash on the development board](#) to burn BIOS firmware to SPI Flash on the development board.

2.6. Method for burning BIOS images to storage devices based on Linux system

Here we use the Linux system of the development board to demonstrate the method of burning BIOS images to storage devices. The advantage of this method is that there is no need to prepare another computer to burn BIOS images.

In addition, if the BIOS image is burned to a TF card, there is no need to prepare a TF card reader. If the BIOS image is burned to an NVMe SSD, there is no



need to prepare an additional SSD hard drive box, as the TF card and SSD can be directly plugged into the development board for use.

1) Ensure that the development board has burned the Linux system and can boot and enter the system normally.

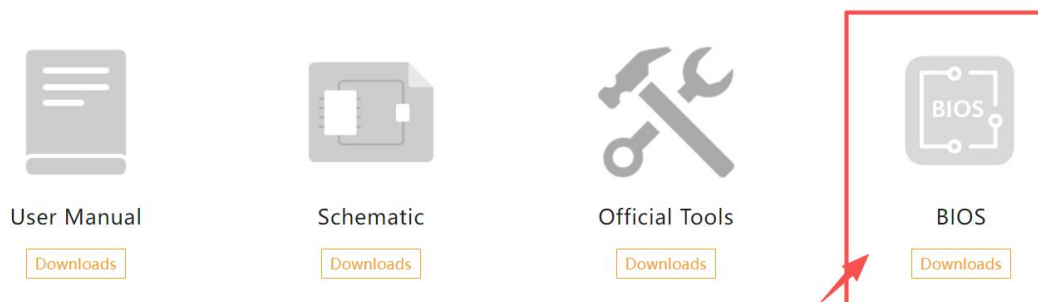
2) Download the BIOS image required for Linux from [the download page of the development board](#). The steps are as follows:

a. Open the data download page of the development board, the link is as follows:

<http://www.orangepi.cn/html/hardWare/computerAndMicrocontrollers/service-and-support/Orange-Pi-6-Plus.html>

b. Then open the download link for BIOS.

Official Resources



c. here are multiple versions of BIOS firmware in the BIOS link. Please download the BIOS image from the **Linux_Bios** folder.

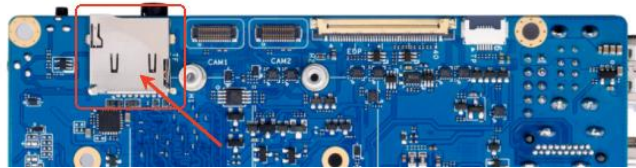


d. The downloaded image file is shown below:

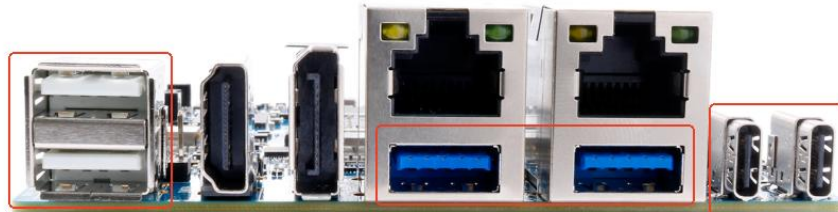
opi6plus_bios_image_for_linux_xxxxxxx.img



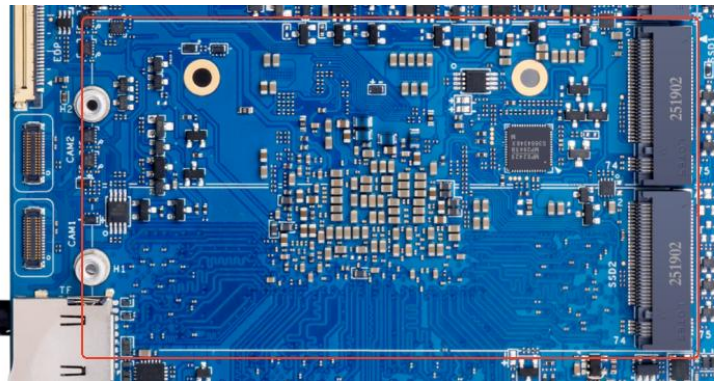
- e. After downloading the BIOS image, please upload it to the **/home/orangepi** directory of the Linux system on the development board.
- 3) The development board supports three boot modes: TF card, USB, and NVMe SSD, and can burn BIOS to any storage device.
 - 4) Before burning the image, the storage device needs to be prepared first.
 - a. Burning BIOS image to TF card: The BIOS image is very small and does not require the capacity of the TF card. The recommended transfer speed for the TF card is **class10** or above, and it is recommended to use TF cards from brands such as SanDisk.
 - b. Burning BIOS image to USB drive: A USB drive needs to be prepared.
 - c. Burning BIOS image to NVMe SSD: A M-Key 2280 specification NVMe SSD solid-state drive is required.
 - 5) Then insert the storage device into the corresponding interface of the development board.
 - a. Burning BIOS image to TF card: The TF card needs to be inserted into the TF card slot on the development board. The location of the TF card slot is as follows:



- b. Burn BIOS image to USB drive: Simply insert the USB drive into the USB interface of the development board.

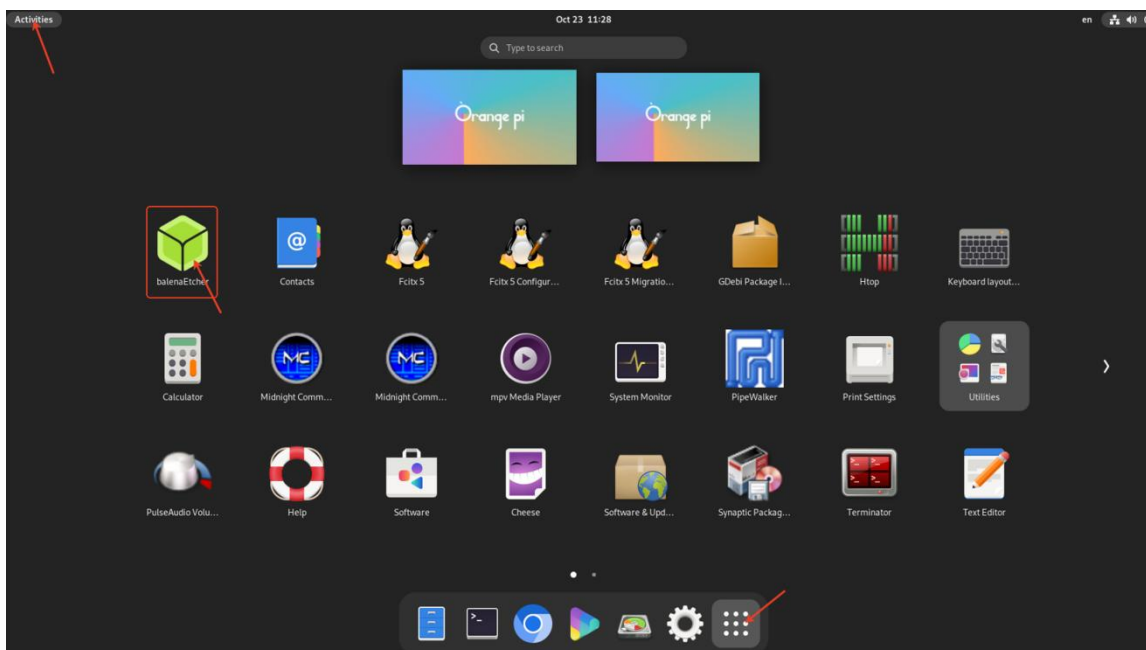


- d. Burning BIOS image to NVMe SSD: The SSD needs to be inserted into the M.2 interface of the M-Key on the development board. The development board has two M.2 interfaces that can be connected to SSDs, so you can choose one of them. Its location is shown below:

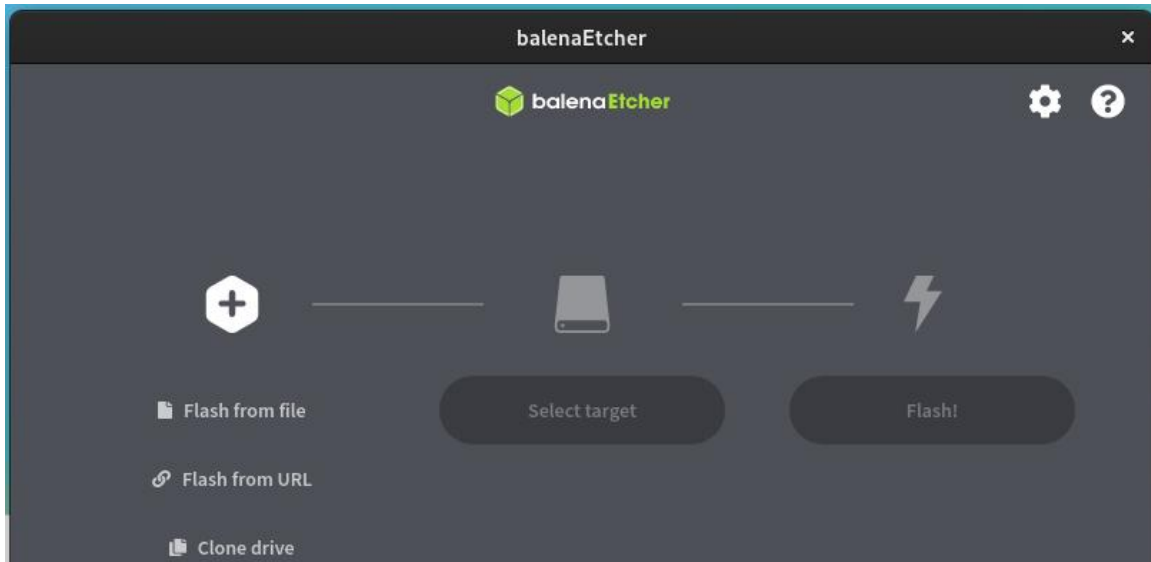


6) The method of burning a BIOS image to a storage device using the balenaEtcher software is as follows:

- a. Open the balenaEtcher software on the Linux system desktop of the development board:



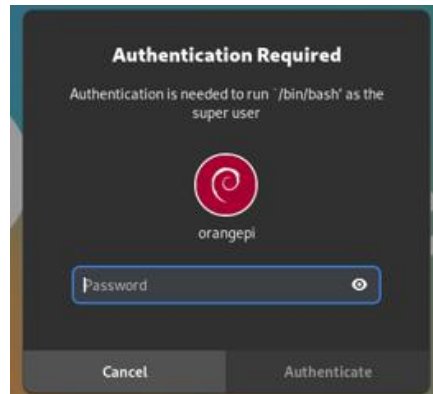
- b. The interface displayed after opening balenaEtcher is shown in the following figure:



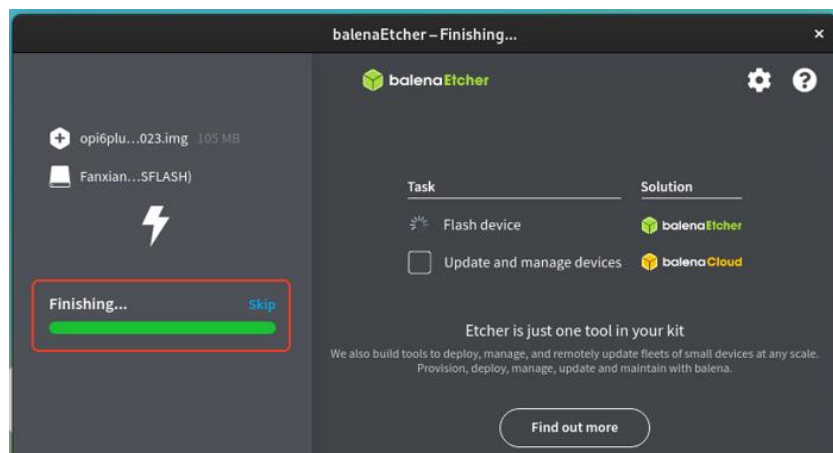
- c. The specific steps for burning BIOS images using balenaEtcher are as follows:
- Select the path of the BIOS image file to burn, which we previously placed in **/home/orangepi**
 - Then select the drive letter corresponding to the TF card, USB flash drive, or SSD storage device;
 - Finally, click Flash to start burning the BIOS image to the storage device.



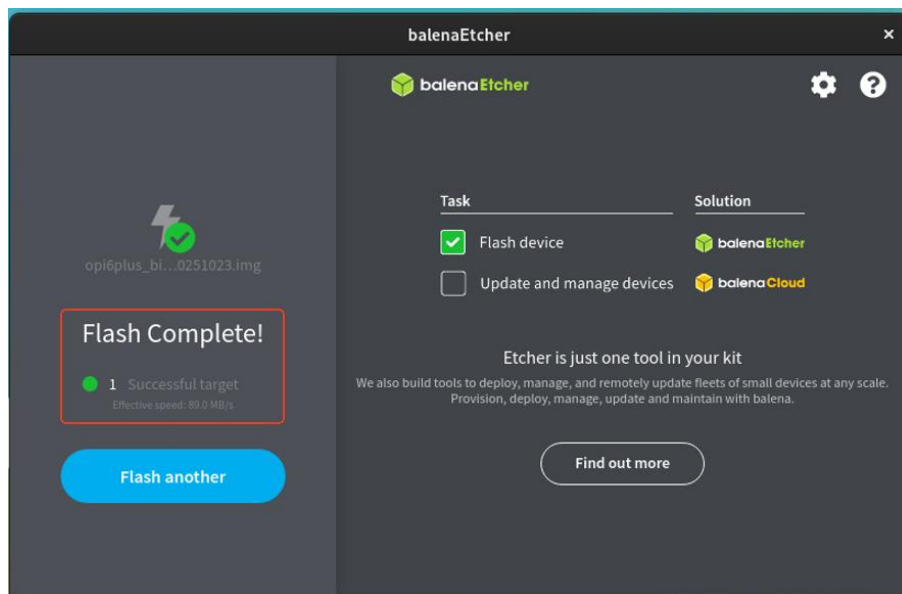
- d) Then enter the password for the Linux system on the development board to officially start burning. If the password is not changed, it defaults to **orangepi**.



- e) After the BIOS image is burned, balenaEtcher defaults to verifying the image burned to the storage device to ensure that there are no issues during the burning process. As shown in the following figure, a green progress bar indicates that the image has been burned and balenaEtcher is verifying the burned image.



- f) After successful burning, the display interface of balenaEtcher is shown in the following figure. If a green indicator icon is displayed, it indicates that the image burning is successful, and you can exit balenaEtcher at this time.



7) The method of burning BIOS images to storage devices using the dd command is as follows:

- a. Use the **fdisk -l** command to view the device file name corresponding to the storage device.
 - a) TF cards and USB drives will print outputs similar to the following, such as /dev/sda or /dev/sdb. Please refer to what you actually see.

```
orange@orange:~$ sudo fdisk -l
```

```
.....
```

```
Disk /dev/sda: 58.59 GiB, 62914560000 bytes, 122880000 sectors
```

```
Disk model: F372
```

```
Units: sectors of 1 * 512 = 512 bytes
```

```
Sector size (logical/physical): 512 bytes / 512 bytes
```

```
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

```
Disklabel type: dos
```

```
Disk identifier: 0x762ca2a7
```

Device	Boot	Start	End	Sectors	Size	Id	Type
/dev/sda1	*	2048	204799	202752	99M	c	W95 FAT32 (LBA)

- b) SSD 会 SSD will print output similar to the following, such as /dev/nvme0n1 or /dev/nvme1n1. Please refer to what you actually see.

```
Disk /dev/nvme0n1: 476.94 GiB, 512110190592 bytes, 1000215216 sectors
```

```
Disk model: KINGSTON OM8SEP4512Q-A0
```



Units: sectors of 1 * 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disklabel type: gpt

Disk identifier: 1E957874-2AE1-4E6B-B6CA-85C187AA3D4C

Device	Start	End	Sectors	Size	Type
/dev/nvme0n1p1	20480	430079	409600	200M	EFI System
/dev/nvme0n1p2	430080	990183423	989753344	472G	Linux filesystem

- b. Confirm that the BIOS image has been uploaded to the **/home/orangepi/** directory on the development board Linux system:

```
orangepi@orangepi:~$ ls *.img
opi6plus_bios_image_for_linux_XXXXXXX.img
```

- c. Then use the following command to burn the BIOS image to the storage device.
- a) The command for TF card or USB drive is as follows. Please note that the **sda** in the following command is only an example. Please replace it with the actual device name corresponding to the storage device.

```
orangepi@orangepi:~$ sudo dd bs=1M if=opi6plus_bios*.img of=/dev/sda
```

- b) The command for SSD is as follows. Please note that **nvme0n1** in the following command is only an example. Please replace it with the actual device name corresponding to the storage device.

```
orangepi@orangepi:~$ sudo dd bs=1M if=opi6plus_bios*.img of=/dev/nvme0n1
```

8) After the burning is completed, the storage device does not need to be unplugged and can be directly placed on the development board. Then, please follow the instructions in the section on **burning BIOS firmware to SPI Flash on the development board** to burn BIOS firmware to SPI Flash on the development board.

2.7. Method of burning BIOS firmware to SPI Flash on the development board

1) Please prepare the storage device for burning the BIOS firmware to the SPI Flash according to the instructions in the section **"Method for Burning BIOS Image to Storage Device Based on Windows PC"** or **"Method for Burning BIOS Image to Storage Device Based on Linux System"**, and then insert the storage device into the



corresponding position of the development board.

- 2) There are several methods to control the BIOS interface, please choose one of them:
 - a. By debugging the serial port: Please ensure that the debugging serial port of the development board can be used normally. For the usage method of debugging the serial port, please refer to the instructions in the "[10pin Debugging Serial Port Usage](#)" section.
 - b. Through the Edp screen: It is necessary to connect the Edp screen and keyboard.
- 3) After powering on the development board, pressing the **F2** or **ESC** shortcut keys on the keyboard can enter the BIOS control interface, as shown in the following figure:

```
CIX Phecda Board
CIX P1 CD8160                               1.90 GHz
v1.0_for_opi6plus_linux-W43M251022191158N-b8be8a190316384 MB RAM

Select Language      <English>
This is the option
one adjusts to change
the language for the
current system

> CIX System Manager
> Advanced
> Boot Manager
> Boot Maintenance Manager

Continue
Reset
```

- 4) Then use the directional keys on the keyboard to select **Boot Manager**.

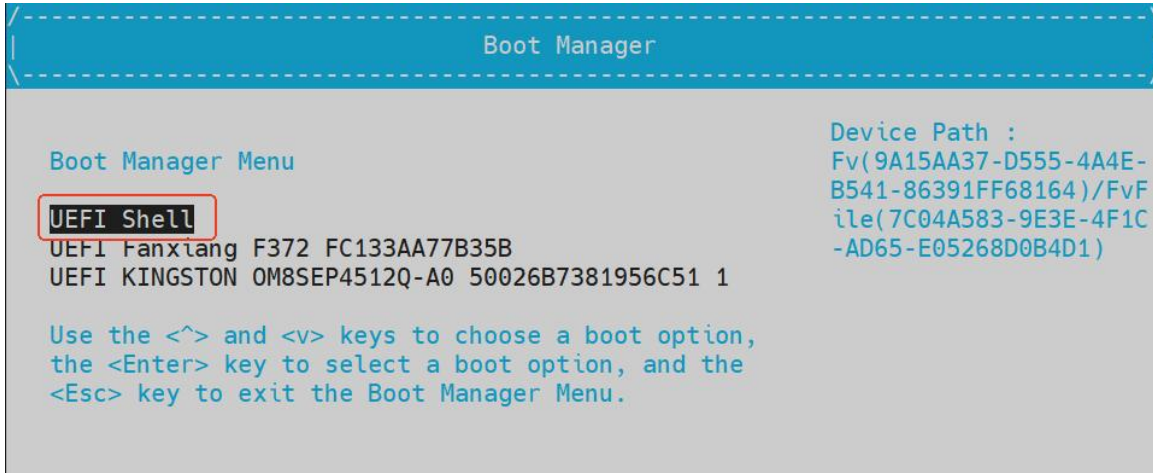
```
CIX Phecda Board
CIX P1 CD8160                               1.90 GHz
v1.0_for_opi6plus_linux-W43M251022191158N-b8be8a190316384 MB RAM

Select Language      <English>
This is the option
one adjusts to change
the language for the
current system

> CIX System Manager
> Advanced
> Boot Manager
> Boot Maintenance Manager

Continue
Reset
```

- 5) After pressing enter, you will enter the interface below, and then please select **UEFI Shell**.



6) After waiting for 5 seconds, it will enter the UEFI Shell interface.

```

UEFI Interactive Shell v2.2
EDK II
UEFI v2.70 (EDK II, 0x00000005)
Mapping table
  FS1: Alias(s):HD1b0b:;BLK4:
        VenHw(0D51905B-B77E-452A-A2C0-ECA0CC8D514A,00801E090000000000)/USB(0x1
,0x0)/HD(1,MBR,0x762CA2A7,0x800,0x31800)
  FS0: Alias(s):HD0b:;BLK1:
        PciRoot(0x1)/Pci(0x0,0x0)/Pci(0x0,0x0)/NVMe(0x1,15-C5-56-19-38-B7-26-0
0)/HD(1,GPT,FEDD3B8A-2116-4BB0-9245-5616EAB83FDD,0x5000,0x64000)
  BLK3: Alias(s):
        VenHw(0D51905B-B77E-452A-A2C0-ECA0CC8D514A,00801E090000000000)/USB(0x1
,0x0)
  BLK0: Alias(s):
        PciRoot(0x1)/Pci(0x0,0x0)/Pci(0x0,0x0)/NVMe(0x1,15-C5-56-19-38-B7-26-0
0)
  BLK2: Alias(s):
        PciRoot(0x1)/Pci(0x0,0x0)/Pci(0x0,0x0)/NVMe(0x1,15-C5-56-19-38-B7-26-0
0)/HD(2,GPT,27BED042-7E2C-49B7-9556-BA049CFAEF32,0x69000,0x1BC3000)
Press ESC in 1 seconds to skip startup.nsh or any other key to continue.
Shell>

```

7) Then enter the file system name after **Shell>** to enter the file system of the storage device where the BIOS image was burned, such as **fs0:** or **fs1**.

- If the development board only inserts a storage device that has burned BIOS images and does not insert any other system disks, then only one **fs0** will be seen.
- If the development board is inserted with a storage device that has burned BIOS images and a Linux system disk, you will see an **fs0** and an **fs1**. If you can't distinguish which fs is the storage device that burned the BIOS image, you can try them all. If you can see the fs in the bios_flash and EFI folders in the file system using the ls command, it is correct.



- a) First, enter **fs0:** or **fs1:** after **Shell>**, be sure to use lowercase.在 **Shell>**

```
Shell>fs1:
```

- b) After entering the file system, if you use the **ls** command to see the **bios_flash** and **EFI** folders, it indicates that the fs entered is correct.

```
FS1:\> ls
Directory of: FS1:\
10/23/2025  01:06 <DIR>          512  bios_flash
10/23/2025  01:06 <DIR>          512  EFI
           0 File(s)          0 bytes
           2 Dir(s)
FS1:\>
```

- 8) Then enter the **bios_flash** directory.

```
FS0:\> cd bios_flash
```

- 9) There are three BIOS firmware stored in **bios_flash**, and their differences are:

```
FS1:\bios_flash> ls
Directory of: FS1:\bios_flash\
10/23/2025  01:06 <DIR>          512  .
10/23/2025  01:06 <DIR>           0  ..
10/23/2025  01:06          6,288,062  bios_40pin_pwm_v1.0.bin
10/23/2025  01:06          6,288,062  bios_40pin_v1.0.bin
10/23/2025  01:06          6,288,062  bios_v1.0.bin
10/23/2025  01:06          434,176  FlashUpdate.efi
10/23/2025  01:06           33  flash_bios.nsh
           5 File(s) 19,298,395 bytes
           2 Dir(s)
FS1:\bios_flash>
```

- a. The three BIOS firmware configurations are the same except for some differences in the configuration of the ACPI 40pin. If the kernel uses a device tree, then there is no difference between the three types of BIOS.
- b. The differences between the three BIOS firmware ACPI configurations are as follows:

BIOS firmware	Differences in ACPI Configuration
bios_v1.0.bin	40pin pins are set to GPIO function by default



bios_40pin_v1.0.bin	Some of the 40pin pins are set to I2C2, I2C4, I2C5, UART1, UART3, SPI1, and PWM5 by default, while other unused pins are set to GPIO function by default
bios_40pin_pwm_v1.0.bin	Some of the 40pin pins are set to I2C2, I2C4, I2C5, UART3, SPI1, PWM5, and PWM3 by default, while other unused pins are set to GPIO function by default

- c. Please refer to the section on [testing methods for GPIO, I2C, UART, SPI, and PWM of the 40pin interface](#) for detailed usage instructions.

10) Then please enter the following command to start burning BIOS firmware into the SPI Flash of the development board:

- a. The command to burn bios_v1.0.bin is as follows:

```
FS1:\bios_flash\> FlashUpdate.efi -f bios_v1.0.bin
```

- b. The command to burn bios_40pin_v1.0.bin is as follows:

```
FS1:\bios_flash\> FlashUpdate.efi -f bios_40pin_v1.0.bin
```

- c. The command to burn bios_40pin_pwm_v1.0.bin is as follows:

```
FS1:\bios_flash\> FlashUpdate.efi -f bios_40pin_pwm_v1.0.bin
```

11) The process of burning BIOS firmware is as follows:

```
FS1:\bios_flash\> FlashUpdate.efi -f bios_40pin_pwm_v1.0.bin
=====
Copyright 2024 Cix Technology Group Co., Ltd. All Rights Reserved.
FlashUpdate Utility v1.04.
Build Date:Jan 23 2025
=====
Old Verison:v1.0_for_opi6plus_linux-W43M251022191158N
New Verison:v1.0_for_opi6plus_linux_40pin_pwm-W43M251022193610N
[00.04] Processing ...
```

12) After burning is completed, the BIOS will automatically reset and start.



```

FS1:\bios_flash\> FlashUpdate.efi -f bios_40pin_pwm_v1.0.bin
=====
Copyright 2024 Cix Technology Group Co., Ltd. All Rights Reserved.
FlashUpdate Utility v1.04.
Build Date:Jan 23 2025
=====
Old Verison:v1.0_for_opi6plus_linux-W43M251022191158N
New Verison:v1.0_for_opi6plus_linux_40pin_pwm-W43M251022193610N
[00.34] Processing ...
Flash update success.
System will be reset...
NOTICE: BL2: v2.7(debug):Beta_2.1.5_release
NOTICE: BL2: Built : 21:53:41, Sep 4 2025
NOTICE: BL2: Booting BL31
NOTICE: BL31: v2.7(debug):Beta_2.1.5_release
NOTICE: BL31: Built : 21:53:41, Sep 4 2025
NOTICE: Send CMD: 0x82000001

```

13) After burning, you can check the BIOS firmware version printed in the BIOS boot log to confirm if the burning is correct:

a. The printing of the bios_v1.0.bin firmware is as follows:

```

Tianocore/EDK2 firmware version v1.0_for_opi6plus_linux-W43M251022191158N-b8be8a
19039c
Press ESCAPE for boot options
PhyDetectDevice: fail to detect ethernet PHY
I2C[5] hardware no response (ISR=0x00000004) during write!
I2C[5] Failed to process I2C message, status No Response
[01.981] [UEFI] E550 BmAfterConsole

```

b. The printing of the bios_40pin_v1.0.bin firmware is as follows:

```

Tianocore/EDK2 firmware version v1.0_for_opi6plus_linux_40pin-W43M251022191320N-
6cab36552d9f
Press ESCAPE for boot options
PhyDetectDevice: fail to detect ethernet PHY
I2C[5] hardware no response (ISR=0x00000004) during write!
I2C[5] Failed to process I2C message, status No Response

```

c. The printing of the bios_40pin_pwm_v1.0.bin firmware is as follows:

```

Tianocore/EDK2 firmware version v1.0_for_opi6plus_linux_40pin_pwm-W43M2510221936
10N-6cab36552d9f
Press ESCAPE for boot options
PhyDetectDevice: fail to detect ethernet PHY
I2C[5] hardware no response (ISR=0x00000004) during write!

```

2. 8. Launch the Orange Pie development board

2) Insert the TF card, USB flash drive, or NVMe SSD that has burned the image into the



Orange Pi development board. Choose one of these three startup methods, and it is recommended to use NVMe SSD for faster speed.

3) The development board has HDMI interface, DP interface, USB-C DP interface, and eDP screen interface, and you can choose one of them to display the desktop of the system.

4) Connect a USB mouse and keyboard to control the Orange Pi development board.

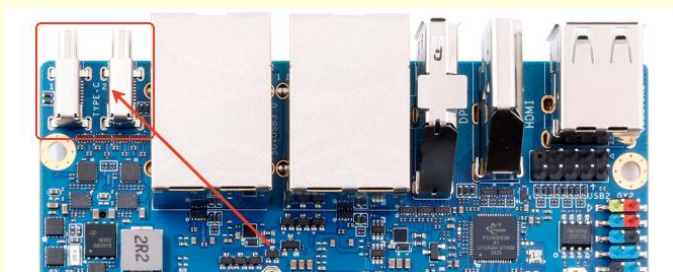
5) The development board has an Ethernet port that can be plugged into a network cable to access the internet. You can also connect an RTL8852BE module to the M.2 E-key interface to access the internet via Wi-Fi.

6) Connect a 20V 100W Type-C PD power supply as shown in the following figure to the Type-C power interface to power the development board.



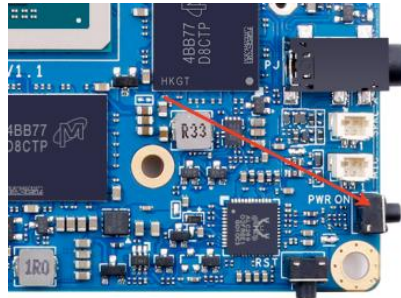
Many unstable phenomena during the system startup process are mostly caused by power supply problems, so a reliable power adapter and cable are important. If you notice continuous restarts during the startup process, please replace the power supply or Type-C data cable and try again.

There are two Type-C interfaces on the development board, both of which support supplying power to the board.





- 7) Then ensure that the power adapter's switch is turned on.
- 8) **Then you need to short press the power button on the development board to officially start it.** The location of the power button is as follows:



- 9) During the system startup process of Linux 6.1 kernel, you can see the following three device trees to choose from:

```

GNU GRUB version 2.11

/-----/
*0 OrangePi 6 Plus (Device Tree)
 1 OrangePi 6 Plus 40pin (Device Tree)
 2 OrangePi 6 Plus 40pin pwm (Device Tree)
/-----/

Use the ^ and v keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the commands
before booting or 'c' for a command-line.
The highlighted entry will be executed automatically in 1s.

```

- Default selection **0 OrangePi 6 Plus (Device Tree)**
- The three device trees have the same configuration except for some differences in the 40pin configuration
- The differences between the three device trees are as follows:

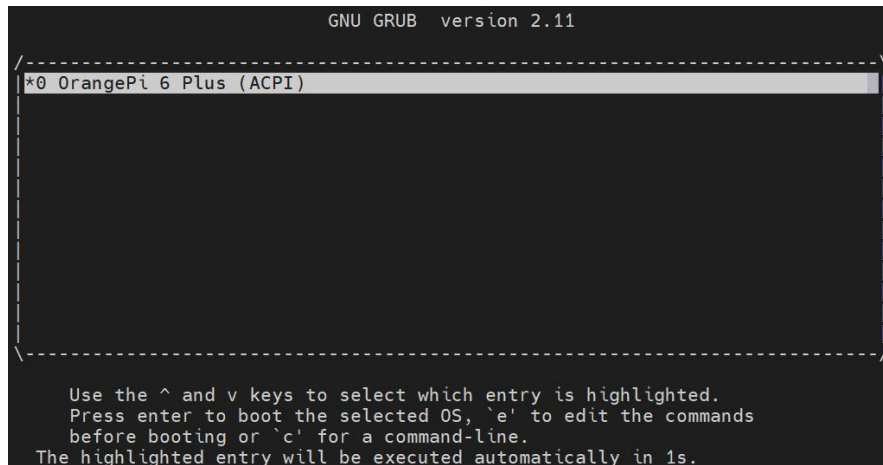
Device Tree Number	Difference
0	40pin pins are set to GPIO function by default
1	Some of the 40pin pins are set to I2C2, I2C4, I2C5, UART1, UART3, SPI1, and PWM5 by default, while other unused pins are set to GPIO function by default



2	Some of the 40pin pins are set to I2C2, I2C4, I2C5, UART3, SPI1, PWM5, and PWM3 by default, while other unused pins are set to GPIO function by default
---	---

- d. Please refer to the section on [testing methods for GPIO, I2C, UART, SPI, and PWM of the 40pin interface](#) for detailed usage instructions.

10) During the system boot process of the Linux 6.6 kernel, only ACPI is the default option.



11) If you want to view the system's output information by debugging the serial port, please connect the development board to the computer using a serial port cable. For the method of connecting the serial port, please refer to the instructions in the section on [using the 10pin debugging serial port](#).

2. 9. 10pin Debugging Serial Port Usage Method

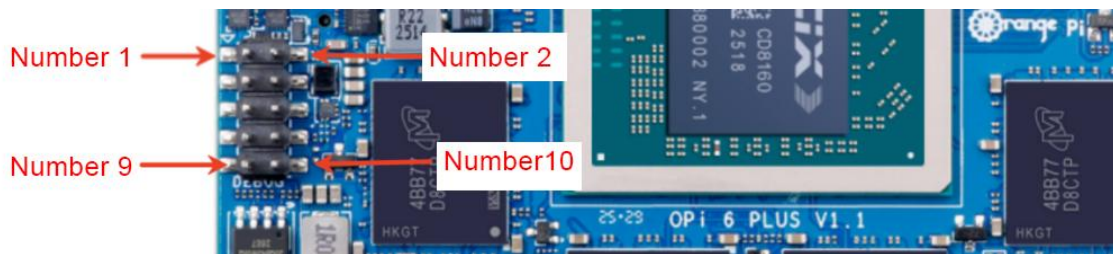
2. 9. 1. 10Pin Debugging Serial Port Pin Instructions

The position of the 10pin debugging serial port pin on the development board is as follows:





The definition of the serial port pin number for 10pin debugging is as follows:



The function of each pin in the 10pin debugging serial port can be found in the following table. **Generally speaking, normal use of the development board only requires the use of the UART2 debugging serial port, and other debugging serial ports do not need to be paid attention to and are basically not needed.**

Default function	Pin function	serial number	serial number	Pin function	Default function
BIOS and kernel logs	UART2_TX	1	2	UART4_TX	PM log
	UART2_RX	3	4	UART4_RX	
	GND	5	6	GND	
Post code log	UART6_TX	7	8	UART5_TX	SE log
	UART6_RX	9	10	UART5_RX	

2. 9. 2. Connection instructions for debugging serial ports corresponding to BIOS and kernel

1) Firstly, it is necessary to prepare a 3.3V USB to TTL module, and then plug one end of the USB interface of the USB to TTL module into the USB interface of the computer.



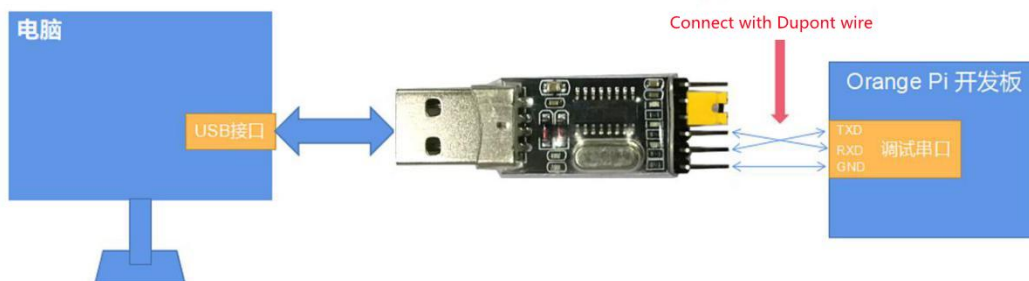
2) The development board defaults to using UART2 to output BIOS and kernel logs. The positions of the UART debugging serial GND, RXD, and TXD pins are shown in the following figure:



3) The GND, TXD, and RXD pins of the USB to TTL module need to be connected to the UART2 debugging serial port on the development board through DuPont wires.

- Connect the GND of the USB to TTL module to the GND of the development board
- Connect the **RX of the USB to TTL module to the UART2-TX pin on the development board**
- Connect the **TX of the USB to TTL module to the UART2-RX pin on the development board**

4) The schematic diagram of connecting a USB to TTL module to a computer and 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 randomly connect the TX and RX of the serial port first. If there is no output during testing, then switch the order of TX and RX. This way, there will always be a correct order.

2. 9. 3. Ubuntu Platform Debugging Serial Port Usage Method

There are many serial debugging software that can be used under Linux, such as Putty, Minicom, etc. Below is a demonstration of how to use Putty.



1) Firstly, insert the USB to TTL module into the USB interface of the Ubuntu computer. If the USB to TTL module is recognized as connected properly, the corresponding device node name can be seen in the **/dev** section of the Ubuntu PC. Remember this node name, which will be used later when setting up the serial port software.

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

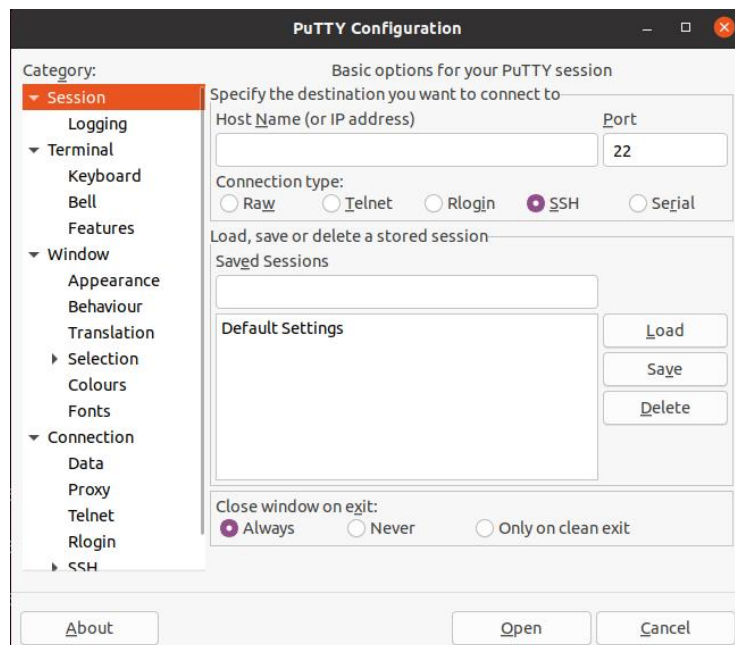
2) Then use the following command to install Putty on Ubuntu PC.

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

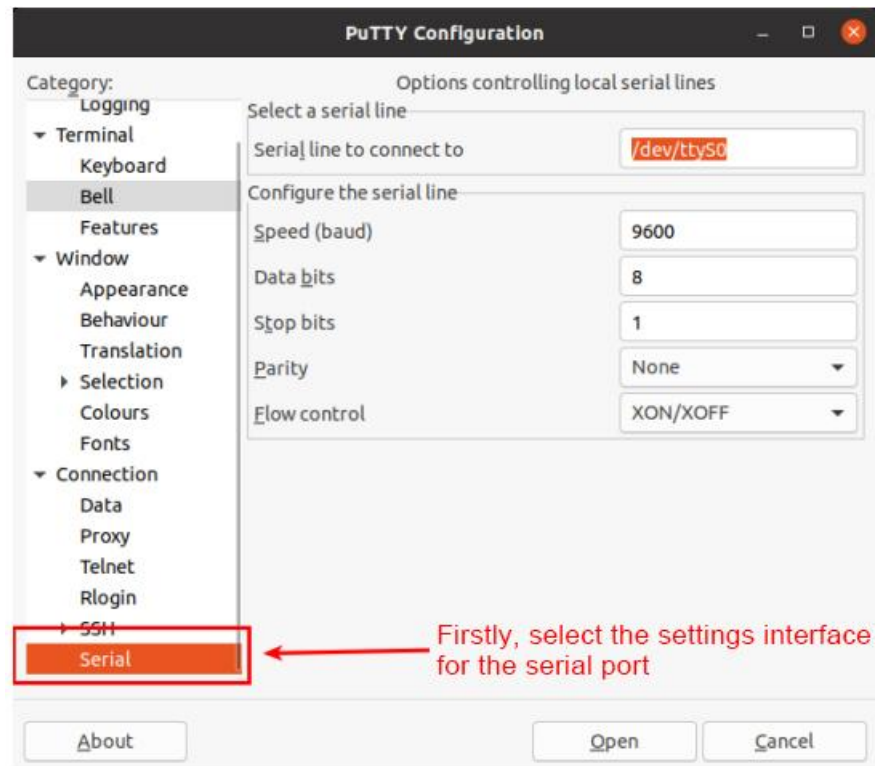
3) Then run Putty, **remember to add sudo privileges.**

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

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

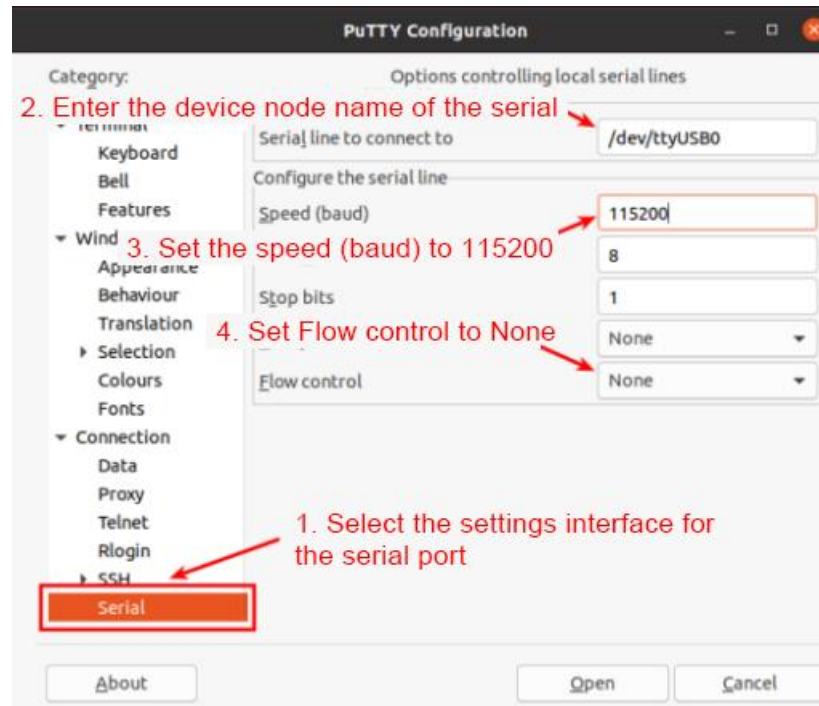


5) Then select the settings interface for the serial port.



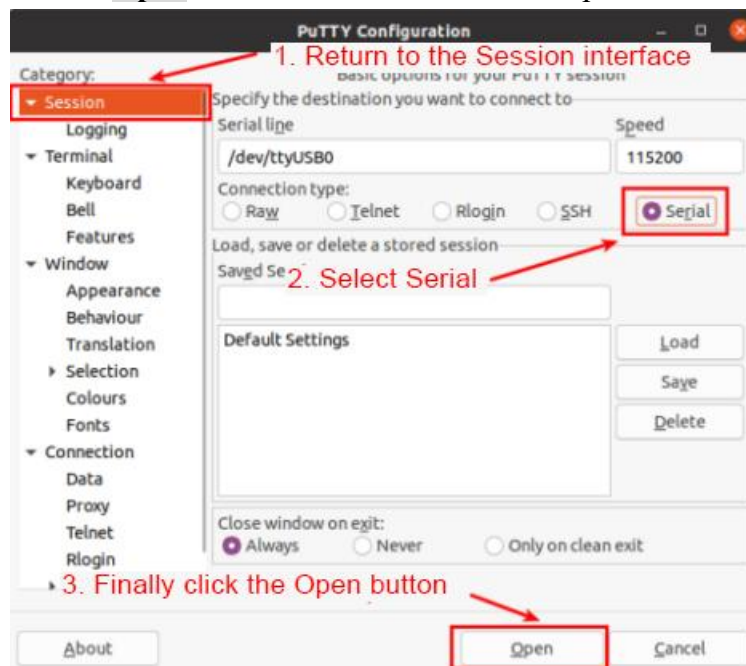
6) Then set the parameters of the serial port.

- Set **Serial line to connect to** to **dev/ttyUSB0** (modify to the corresponding node name, usually **/dev/ttyUSB0**);
- Set **Speed(baud)** to **115200** (baud rate of the serial port);
- Set **Flow control** to **None**.



7) After completing the settings on the serial port interface, return to the Session interface

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



8) After starting the development board, you can see the system output Log information



from the opened serial port 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

```

2.9.4. Instructions for Debugging Serial Ports on Windows Platform

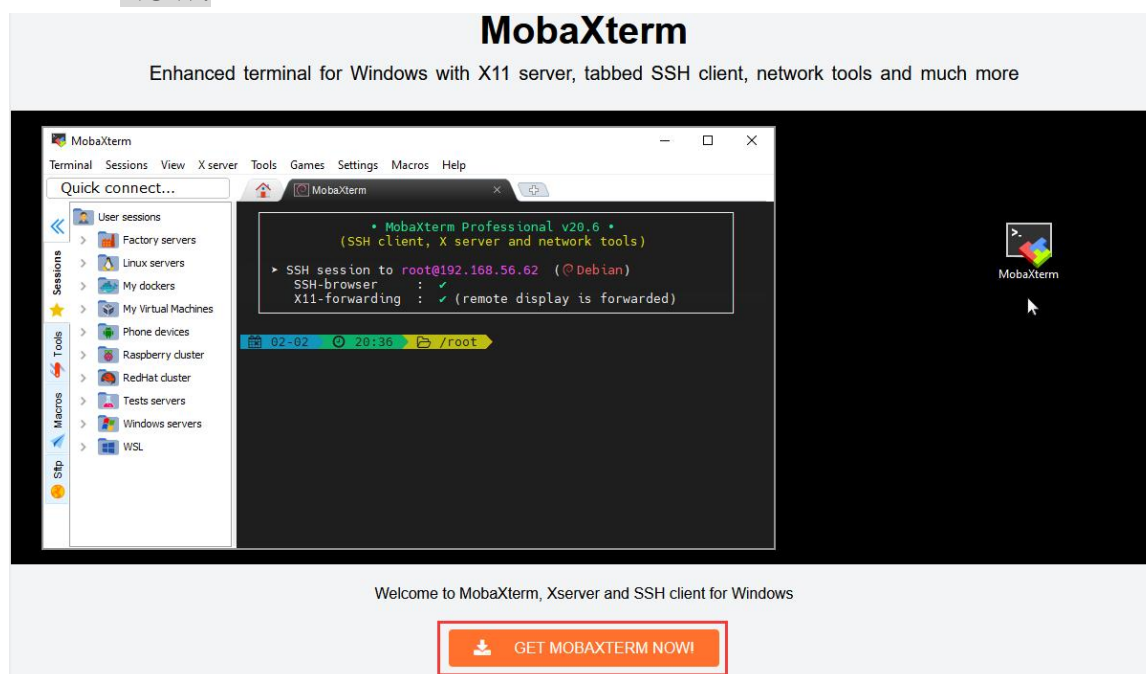
There are many serial debugging software that can be used under Windows, such as SecureCRT, MobaXterm, etc. Below is a demonstration of how to use MobaXterm. This software has a free version and can be used without purchasing a serial number.

1) Download MobaXterm.

- a. The download link for MobaXterm is as follows:

<https://mobaxterm.mobatek.net>

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





c. Then choose to download the Home version.

Home Edition

Free

Full **X server** and **SSH** support
 Remote desktop (RDP, VNC, Xdmcp)
 Remote terminal (SSH, telnet, rlogin, Mosh)
 X11-Forwarding
 Automatic SFTP browser
 Master password protection
 Plugins support
 Portable and installer versions
 Full documentation
 Max. **12** sessions
 Max. **2** SSH tunnels
 Max. **4** macros
 Max. **360** seconds for Tftp, Nfs and Cron

Download now

Professional Edition

\$69 / 49€ per user*

* Excluding tax. Volume discounts [available](#)

Every feature from Home Edition +
 Customize your startup message and logo
 Modify your profile script
 Remove unwanted games, screensaver or tools
 Unlimited number of sessions
 Unlimited number of tunnels and macros
 Unlimited run time for network daemons
 Enhanced security settings
 12-months updates included
 Deployment inside company
 Lifetime right to use

Subscribe online / Get a quote

d. Then select the Portable version, and after downloading, there is no need to install it. Simply open it and you can use it.

MobaXterm Home Edition

Download MobaXterm Home Edition (current version):

MobaXterm Home Edition v22.2
(Portable edition)

MobaXterm Home Edition v22.2
(Installer edition)

Download previous stable version: [MobaXterm Portable v22.1](#) [MobaXterm Installer v22.1](#)

By downloading MobaXterm software, you accept [MobaXterm terms and conditions](#)

You can download the third party plugins and components sources [here](#)

If you use MobaXterm inside your company, you should consider subscribing to [MobaXterm Professional Edition](#): your subscription will give you access to professional support and to the "Customizer" software. This customizer will allow you to generate personalized versions of MobaXterm including your own logo, your default settings and your welcome message. Please [contact us](#) for more information.

2) After downloading, use decompression software to extract the downloaded compressed file to obtain the executable software of MobaXterm, and then double-click to open it.

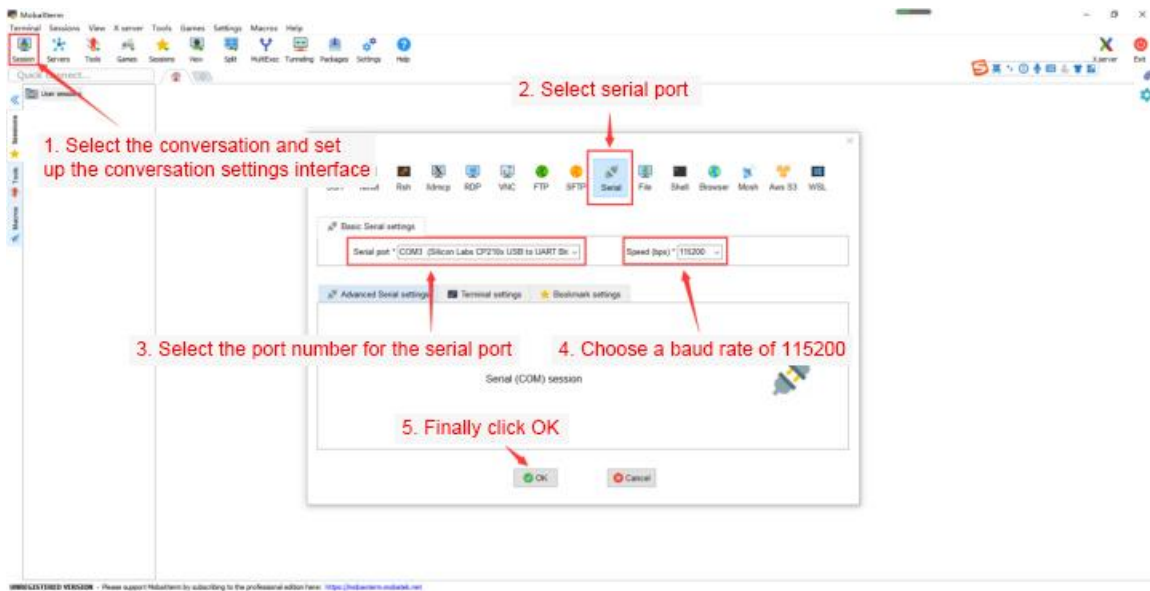
名称	修改日期	类型	大小
CygUtils.plugin	2022/9/24 20:16	PLUGIN 文件	17,484 KB
MobaXterm_Personal_22.2	2022/10/22 16:53	应用程序	16,461 KB

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

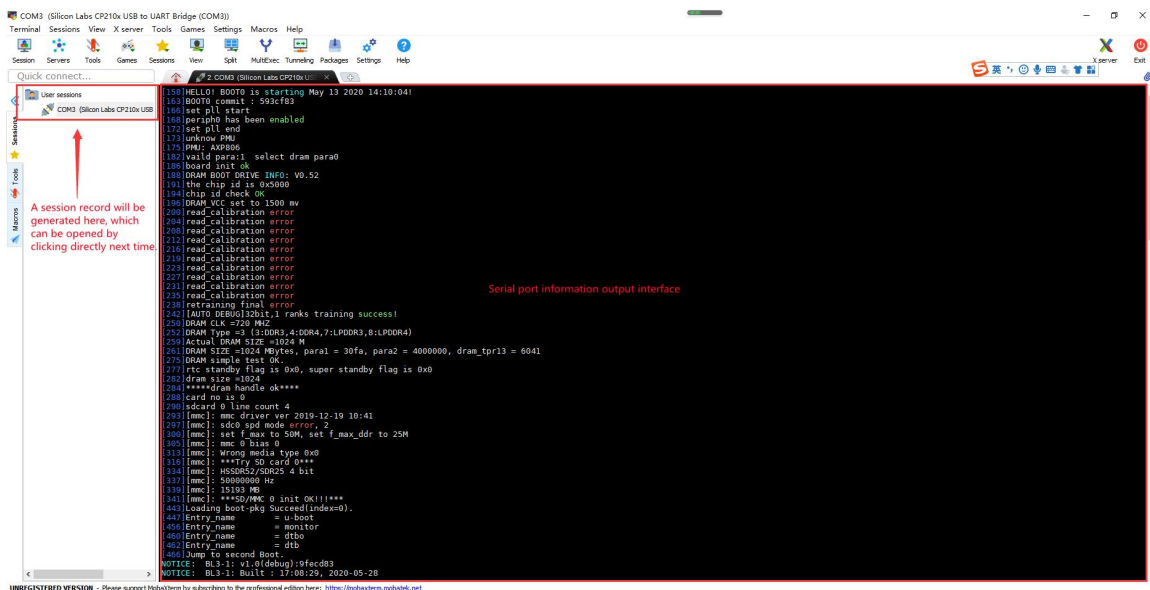
- a. Open the session settings interface;
- b. Select serial port type;



- c. Select the port number for the serial port (choose the corresponding port number according to the actual situation). If you cannot see the port number, please use **360 Driver Master** to scan and install the driver for the USB to TTL serial port chip;
- d. Choose a baud rate of **115200** for the serial port;
- e. Finally, click the **"OK"** button to complete the setup.



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





3. Instructions for using the Debian Gnome desktop system

3.1. Supported Debian image types and kernel versions

Linux image type	kernel version	desktop version
Debian 12 - Bookworm	Linux6.1	support
Debian 12 - Bookworm	Linux6.6	support

3.2. Linux System Adaptation Status

3.2.1. Linux6.1 System Adaptation Status

function	Linux6.1 DT Debian12	Linux6.6 ACPI Debian12
Mali GPU	OK	OK
NPU	OK	OK
Video hardware encoding	OK	OK
Video hardware decoding	OK	OK
memory	OK	OK
TF card startup	OK	OK
NVMe SSD boot recognition TF card	OK	OK
NVMe SSD1 startup	OK	OK
NVMe SSD1 boot recognition SSD2	OK	OK
NVMe SSD2 startup	OK	OK
NVMe SSD2 boot recognition SSD1	OK	OK
SPI Flash - BIOS Startup	OK	OK



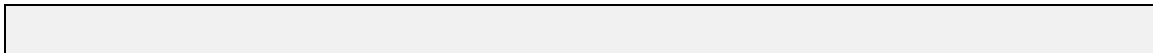
RTL8852BE - Wi-Fi	OK	OK
RTL8852BE - BT	OK	OK
5G Ethernet * 2	OK	OK
5G Ethernet port light * 2	OK	OK
USB3.0 HOST * 2	OK	OK
USB 3.0 HOST boot system * 2	OK	OK
USB2.0 HOST * 2	OK	OK
USB 2.0 HOST boot system * 2	OK	OK
9-pin socket - USB 2.0 HOST * 2	OK	OK
Type-C 3.0 HOST * 2	OK	OK
Type-C 3.0 PD power supply * 2	OK	OK
Type-C 3.0 DP display * 2	OK	OK
Type-C 3.0 DP Audio * 2	OK	OK
Type-C 3.0 Startup System * 2	OK	OK
Type-C 3.0 USB0 Device	OK	OK
Type-C 3.0 Charging the Battery * 2	OK	OK
Reading battery information in Debian	OK	OK
Battery interface power supply	OK	OK
DP display	OK	OK
DP Audio	OK	OK
HDMI display	OK	OK
HDMI Audio	OK	OK
eDP display	OK	OK
eDP backlight	OK	OK
40Pin - GPIO	OK	OK
40Pin - UART	OK	OK
40Pin - I2C	OK	OK
40Pin - SPI	OK	OK
40Pin - PWM	OK	OK
LED lights * 3	OK	OK
PWM fan	OK	OK
RTC	OK	OK



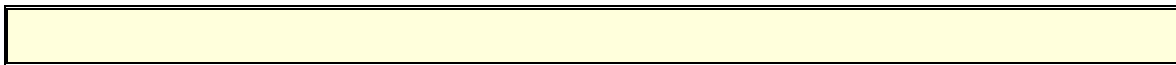
10pin debugging serial port	OK	OK
Poweroff command	OK	OK
Reboot command	OK	OK
watchdog	OK	OK
13 temperature sensors in SoC	OK	OK
Over temperature electricity	OK	OK
USB camera	OK	OK
Reset button	OK	OK
power button	OK	OK
CAM1 - OV13850	OK	OK
CAM1 - OV13855	OK	OK
CAM2 - OV13850	OK	OK
CAM2 - OV13855	OK	OK
sleep wake-up	OK	OK
3.5mm headphone audio playback	OK	OK
3.5mm headphone audio recording	OK	OK
Speaker * 2	OK	OK
Onboard analog MIC recording	OK	OK

3.3. Explanation of Linux Command Format in This Manual

1) All commands that need to be entered in the Linux system in this manual will be enclosed in the boxes below.



As shown below, the content in the yellow box represents the content that needs special attention, except for the commands inside.



2) The prompt type description before the command.

- a. The prompt in front of the command refers to the content in the red part of the box below, which is not part of Linux commands. Therefore, when entering commands in the Linux system, please do not also enter the content in the red font.



```

orangepi@orangepi:~$ sudo apt update
root@orangepi:~# vim /boot/boot.cmd
test@test:~$ ssh root@192.168.1.xxx
root@test:~# ls

```

- b. **root@orangepi:~\$** The prompt indicates that this command was entered in the **Linux system of the development board**, and the **\$** at the end of the prompt indicates that the current user of the system is a regular user. When executing privileged commands, **sudo** needs to be added.
- c. **root@orangepi:~#** The prompt indicates that this command was entered in the **Linux system of the development board**, and the **#** at the end of the prompt indicates that the current user of the system is the root user and can execute any desired command.
- d. **test@test:~\$** The prompt indicates that this command was entered on an Ubuntu PC or Ubuntu virtual machine, not on the Linux system of the development board. The **\$** at the end of the prompt indicates that the current system user is a regular user, and when executing privileged commands, **sudo** needs to be added.
- e. **root@test:~#** The prompt indicates that this command was entered on an Ubuntu PC or Ubuntu virtual machine, not on the Linux system of the development board. The **#** at the end of the prompt indicates that the current system user is the root user and can execute any command they want.

3) What are the commands that need to be entered?

- a. As shown below, **the bold black part represents** the commands that need to be inputted, and the content below the commands is the output (some commands have output, while others may not). This part of the content does not need to be inputted.

```

root@orangepi:~# cat /boot/GRUB/GRUB.CFG
et debug=loader,mm
set term=vt100
set default="0"
set timeout=2
.....

```

- b. As shown below, some commands that cannot be written on one line will be moved to the next line, and any black and bold parts are commands that need to be entered. When these commands are inputted on a single line, the **'\'** at the end of each line needs to be removed, which is not part of the command. Also,



different parts of the command have spaces, please don't miss them.

```
orange@orange:~$ echo \
"deb [arch=$(dpkg --print-architecture) \
signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] \
https://download.docker.com/linux/debian \
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

3. 4. Linux system login instructions

3. 4. 1. Linux system default login account and password

account	password
root	orange
orange	orange

Note that when entering the password, the specific content of the input password will not be displayed on the screen. After entering it, simply press enter.

3. 4. 2. Linux Desktop System Automatic Login Instructions

1) After the desktop version system starts, it will automatically log in to the desktop without entering a password.



2) Running the following command can prevent the desktop version system from



automatically logging into the desktop.

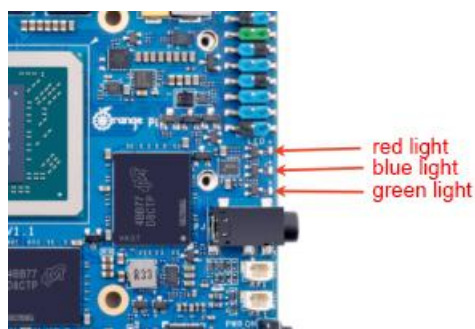
```
orangepi@orangepi:~$ sudo sed -i \  
's/^AutomaticLoginEnable\s*=\s*true$/AutomaticLoginEnable = false/' \  
/etc/gdm3/daemon.conf
```

3) Then restart the system and a login dialog box will appear. At this point, a **password** is required to enter the system.

```
orangepi@orangepi:~$ sudo reboot
```

3. 5. Test instructions for onboard LED lights

1) There are three LED lights on the development board, red, blue, and green, located as shown in the following figure:



2) As long as the development board is connected to the Type-C power supply, the blue LED light will remain on, which is controlled by hardware and cannot be turned off by software. The blue LED light can determine whether the power supply of the development board has been turned on normally.

3) The green LED light will flash continuously after the development board is turned on, which is controlled by software. If the green LED light does not flash after pressing the power button to turn on the development board, it indicates that there is a problem with the board startup (provided that the green LED light is not broken).

4) The red LED light only stays on or flashes when the battery is connected, and does not light up when the battery is not connected.



3. 6. Network connection test

3. 6. 1. 5G Ethernet Port Test

1) The development board has two 5G Ethernet ports. The testing method for both ports is the same. First, plug one end of the network cable into the Ethernet port of the development board and the other end into the router. Make sure the network is open.

2) After the system starts, the IP address will be automatically assigned to the Ethernet card through **DHCP**, and no other configuration is required.

3) The command to check the IP address in the Linux system of the development board is as follows:

```
orangePi@orangePi:~$ sudo ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 10.31.3.166  netmask 255.255.0.0  broadcast 10.31.255.255
    inet6 fe80::2297:b465:28ea:65a8  prefixlen 64  scopeid 0x20<link>
    ether 00:00:4c:68:22:19  txqueuelen 1000  (Ethernet)
    RX packets 17482  bytes 10125819 (9.6 MiB)
    RX errors 0  dropped 2  overruns 0  frame 0
    TX packets 7506  bytes 736594 (719.3 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
    device interrupt 108
```

```
orangePi@orangePi:~$ sudo ifconfig eth1
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 10.31.3.168  netmask 255.255.0.0  broadcast 10.31.255.255
    inet6 fe80::4d2a:20bf:e016:95a6  prefixlen 64  scopeid 0x20<link>
    inet6 fe80::869e:b9c8:686f:dc3d  prefixlen 64  scopeid 0x20<link>
    ether 00:00:4c:68:22:1a  txqueuelen 1000  (Ethernet)
    RX packets 6637  bytes 615215 (600.7 KiB)
    RX errors 0  dropped 1  overruns 0  frame 0
    TX packets 95  bytes 15025 (14.6 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
    device interrupt 110
```



When using ifconfig to check the IP address, if the following message is displayed, it is because sudo is not added. The correct command is: `sudo ifconfig`

```
orangeypi@orangeypi:~$ ifconfig
```

Command 'ifconfig' is available in the following places

- * /sbin/ifconfig

- * /usr/sbin/ifconfig

The command could not be located because '/sbin:/usr/sbin' is not included in the PATH environment variable.

This is most likely caused by the lack of administrative privileges associated with your user account.

```
ifconfig: command not found
```

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

- 1. Connect the monitor, then log in to the system and use the `ifconfig` command to view the IP address**
- 2. Enter the `ifconfig` command in the UART2 debug serial port terminal to view the IP address**
- 3. If you don't have a debug serial port or an HDMI display, you can also check the IP address of the development board's network port through the router's management interface. If you can't see the IP address using this method, the debugging method is as follows:**

A) First check whether the Linux system has started normally. If the green light on the development board is flashing, it usually has started normally. If only the blue light is on, it means the system has not started normally.

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

C) Try another router (I've encountered many router issues, such as the router not being able to assign an IP address properly, or the IP address being assigned properly but not visible in the router)

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

It should also be noted that the development board DHCP automatically assigns an IP address and does not require any settings on the development board.



4) The command to test network connectivity is as follows. The **ping** command can be interrupted by pressing the **Ctrl+C** shortcut key.

```
orangePi@orangePi:~$ ping www.baidu.com -I eth0    #Test command for one of the network
ports
orangePi@orangePi:~$ ping www.baidu.com -I eth1    #Test command for another network port
PING www.a.shifen.com (183.2.172.177) from 10.31.3.166 eth0: 56(84) bytes of data.
64 bytes from 183.2.172.177 (183.2.172.177): icmp_seq=1 ttl=54 time=18.6 ms
64 bytes from 183.2.172.177 (183.2.172.177): icmp_seq=2 ttl=54 time=18.6 ms
^C
--- www.a.shifen.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 15.030/17.607/18.988/1.501 ms
```

3. 6. 2. Wi-Fi connection test

First of all, please note that there is no Wi-Fi module on the development board. An external PCIe network card or USB network card is required to use the Wi-Fi function.

For instructions on using the external PCIe network card, please refer to the section "[How to use the M.2 E-Key PCIe Wi-Fi6+ Bluetooth module.](#)"

For instructions on using the external USB network card, please refer to [the USB wireless network card testing section.](#)

3. 6. 2. 1. How to connect to Wi-Fi via nmcli command

If the development board is not connected to an Ethernet or monitor, but only to a serial port, it's recommended to use the commands demonstrated in this section to connect to a Wi-Fi network. This is because nmtui can only display characters and not a graphical interface in some serial port software (such as minicom). However, if the development board is connected to an Ethernet or monitor, you can still use the commands demonstrated in this section to connect to a Wi-Fi network.

The commands shown below all assume the Wi-Fi network device node name is wlan0. However, different Wi-Fi modules may have different node names. Please replace them with the actual node name you see.



- 1) Log in to the Linux system first. There are three ways:
 - a. If the development board is connected to a network cable, you can log in to [the Linux system remotely via SSH](#).
 - b. If the development board is connected to the debug serial port, you can log in to the Linux system using a serial terminal.
 - c. If the development board is connected to a monitor, you can log in to the Linux system using the desktop displayed on the monitor.

- 2) Then use the **nmcli dev Wi-Fi** command to scan the surrounding Wi-Fi hotspots.

```
orangeipi@orangeipi:~$ nmcli dev wifi
```

```
root@orangeipi:~# nmcli dev wifi
IN-USE  BSSID          SSID          MODE  CHAN  RATE        SIGNAL  BARS  SECURITY
28:6C:07:6E:87:2E  orangeipi      Infra  9     260 Mbit/s  97      ████████ WPA1 WPA2
D8:D8:66:A5:BD:D1  orangeipi      Infra  10    270 Mbit/s  90      ████████ WPA1 WPA2
A0:40:A0:A1:72:20  orangeipi      Infra  4     405 Mbit/s  82      ████████ WPA2
28:6C:07:6E:87:2F  orangeipi_5G   Infra  149   540 Mbit/s  80      ████████ WPA1 WPA2
CA:50:E9:89:E2:44  ChinaNet-TC15  Infra  1     130 Mbit/s  79      ████████ WPA1 WPA2
A0:40:A0:A1:72:31  NETGEAR20      Infra  100   405 Mbit/s  67      ████████ WPA2
D4:EE:07:08:A9:E0  orangeipi      Infra  4     130 Mbit/s  55      ████████ WPA1 WPA2
88:C3:97:49:25:13  orangeipi      Infra  6     130 Mbit/s  52      ████████ WPA1 WPA2
00:BD:82:51:53:C2  orangeipi      Infra  12    130 Mbit/s  49      ████████ WPA1 WPA2
C0:61:18:FA:49:37  orangeipi      Infra  149   270 Mbit/s  47      ████████ WPA1 WPA2
04:79:70:8D:0C:B8  orangeipi      Infra  153   270 Mbit/s  47      ████████ WPA2
04:79:70:FD:0C:B8  orangeipi      Infra  153   270 Mbit/s  47      ████████ WPA2
9C:A6:15:DD:E6:0C  orangeipi      Infra  10    270 Mbit/s  45      ████████ WPA1 WPA2
B4:0F:3B:45:D1:F5  orangeipi      Infra  48    270 Mbit/s  45      ████████ WPA1 WPA2
E8:CC:18:4F:7B:44  orangeipi      Infra  157   135 Mbit/s  45      ████████ WPA1 WPA2
B0:95:8E:D8:2F:ED  orangeipi      Infra  11    405 Mbit/s  39      ████████ WPA1 WPA2
C0:61:18:FA:49:36  orangeipi      Infra  11    270 Mbit/s  24      ████████ WPA1 WPA2
root@orangeipi:~#
```

- 3) Then use the **nmcli** command to connect to the scanned Wi-Fi hotspot, where:
 - a. **Wi-Fi_name** needs to be replaced with the name of the Wi-Fi hotspot you want to connect to;
 - b. **Wi-Fi_passw** needs to be replaced with the password of the Wi-Fi hotspot you want to connect to.

```
orangeipi@orangeipi:~$ sudo nmcli dev wifi connect Wi-Fi_name password
```

```
Wi-Fi_passw
```

```
Device 'wlan0' successfully activated with 'cf937f88-ca1e-4411-bb50-61f402eef293'.
```

- 4) Use the **ip addr show wlan0** command to view the Wi-Fi IP address.

The device node name of Wi-Fi is not always wlan0. Please refer to the actual one you see.

```
orangeipi@orangeipi:~$ sudo ifconfig wlan0
```



```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 10.31.1.91  netmask 255.255.0.0  broadcast 10.31.255.255
    inet6 fe80::ea57:7f73:3c12:bb54  prefixlen 64  scopeid 0x20<link>
    inet6 fe80::6786:c0ae:6658:f25c  prefixlen 64  scopeid 0x20<link>
    ether e0:75:26:4c:ab:dd  txqueuelen 1000  (Ethernet)
    RX packets 1428  bytes 188043 (183.6 KiB)
    RX errors 0  dropped 5  overruns 0  frame 0
    TX packets 114  bytes 24063 (23.4 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

5) Use the **ping** command to test the connectivity of the Wi-Fi network. The **ping** command can be interrupted by pressing the **Ctrl+C** shortcut key.

The device node name of Wi-Fi is not always wlan0. Please refer to the actual one you see.

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

3. 6. 2. 2. How to connect to Wi-Fi via nmtui command

- 1) Log in to the Linux system first. There are three ways:
 - a. If the development board is connected to a network cable, you can log in to [the Linux system remotely via SSH](#).
 - b. If the development board is connected to the debug serial port, you can log in to the Linux system using a serial terminal (use MobaXterm as the serial port software; minicom will not display the graphical interface).

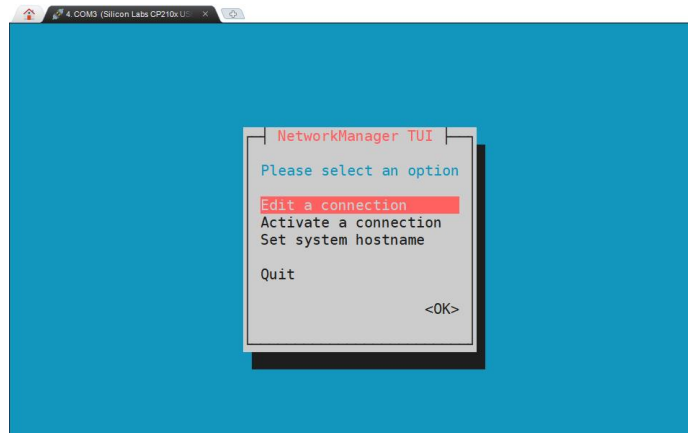


- c. If the development board is connected to a monitor, you can log in to the Linux system using the desktop displayed on the monitor.

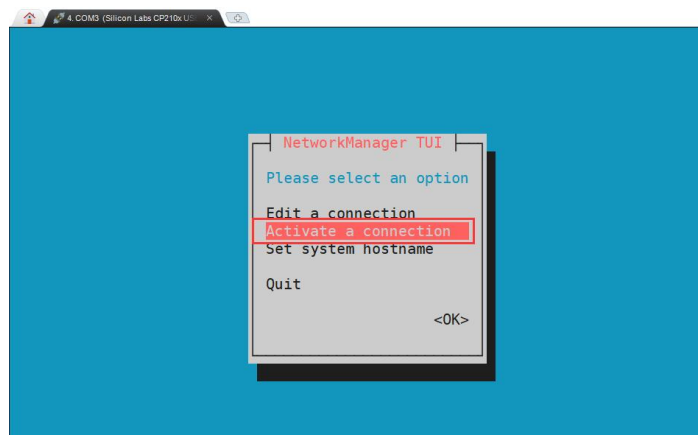
- 2) Then enter the nmtui command in the command line to open the Wi-Fi connection interface.

```
orange@orange:~$ sudo nmtui
```

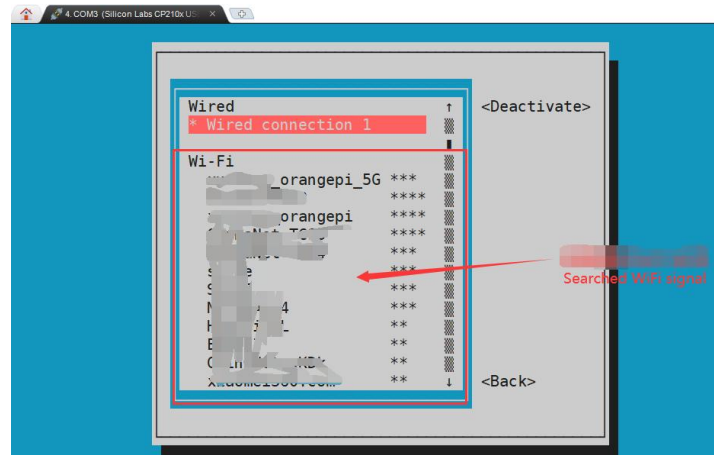
- 3) Enter the nmtui command to open the interface as shown below:



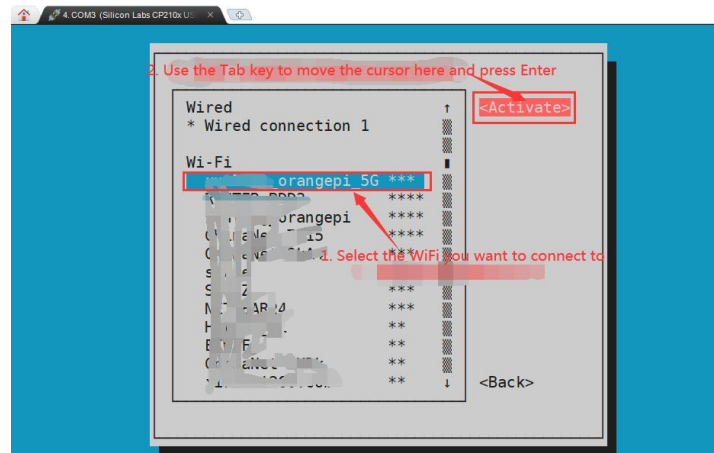
- 4) Select **Activate a connect** and press Enter.



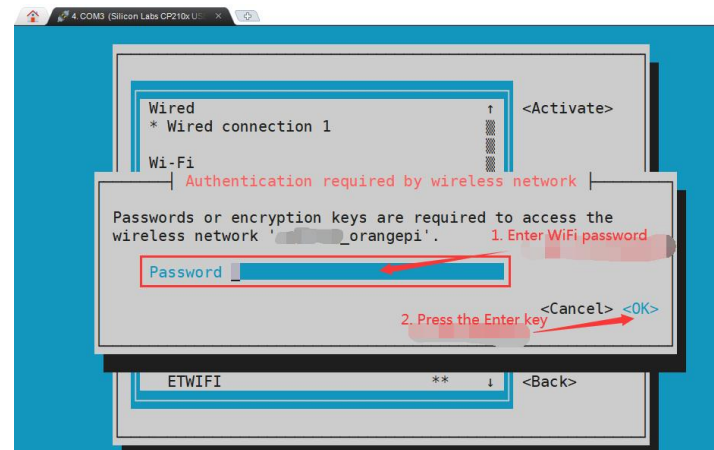
- 5) Then you can see all the searched Wi-Fi hotspots.



6) Select the Wi-Fi hotspot you want to connect to, then use the Tab key to position the cursor on **Activate** and press Enter.

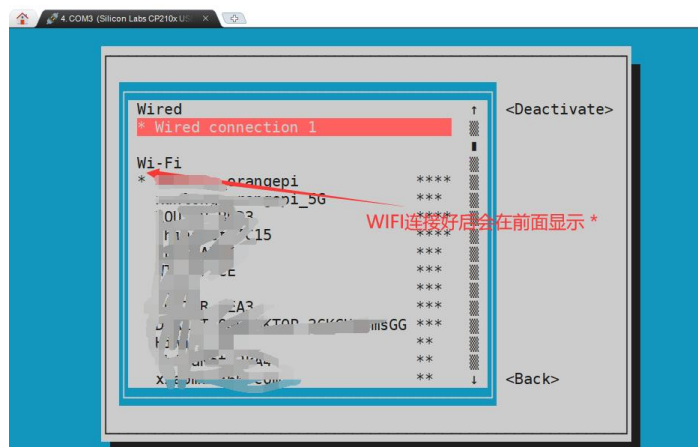


7) Then a dialog box for entering a password will pop up. Enter the corresponding password in the **Password** field and press Enter to start connecting to Wi-Fi.





8) After the Wi-Fi connection is successful, a "*" will be displayed in front of the connected Wi-Fi name.



9) Use the `ip addr show wlan0` command to view the Wi-Fi IP address.

The device node name of Wi-Fi is not always wlan0. Please refer to the actual one you see.

```
orangepi@orangepi:~$ sudo ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 10.31.1.91  netmask 255.255.0.0  broadcast 10.31.255.255
    inet6 fe80::ea57:7f73:3c12:bb54  prefixlen 64  scopeid 0x20<link>
    inet6 fe80::6786:c0ae:6658:f25c  prefixlen 64  scopeid 0x20<link>
    ether e0:75:26:4c:ab:dd  txqueuelen 1000  (Ethernet)
    RX packets 1428  bytes 188043 (183.6 KiB)
    RX errors 0  dropped 5  overruns 0  frame 0
    TX packets 114  bytes 24063 (23.4 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

10) Use the `ping` command to test the connectivity of the Wi-Fi network. The `ping` command can be interrupted by pressing the `Ctrl+C` shortcut key.

The device node name of Wi-Fi is not always wlan0. Please refer to the actual one you see.

```
orangepi@orangepi:~$ ping www.orangepi.org -I wlan0
PING www.orangepi.org (182.92.236.130) from 192.168.1.49 wlan0: 56(84) bytes of data.
```



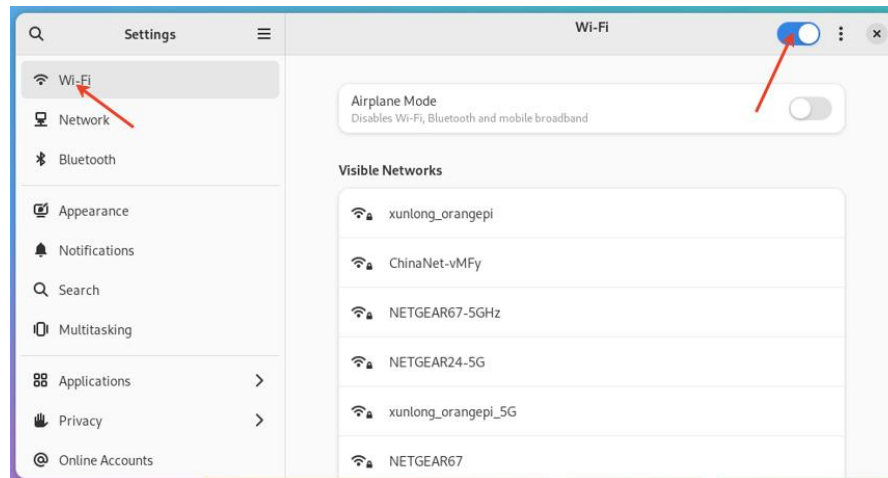

```
64 bytes from 182.92.236.130 (182.92.236.130): icmp_seq=1 ttl=52 time=43.5 ms
64 bytes from 182.92.236.130 (182.92.236.130): icmp_seq=2 ttl=52 time=41.3 ms
64 bytes from 182.92.236.130 (182.92.236.130): icmp_seq=3 ttl=52 time=44.9 ms
64 bytes from 182.92.236.130 (182.92.236.130): icmp_seq=4 ttl=52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

3. 6. 2. 3. How to connect to Wi-Fi on the desktop

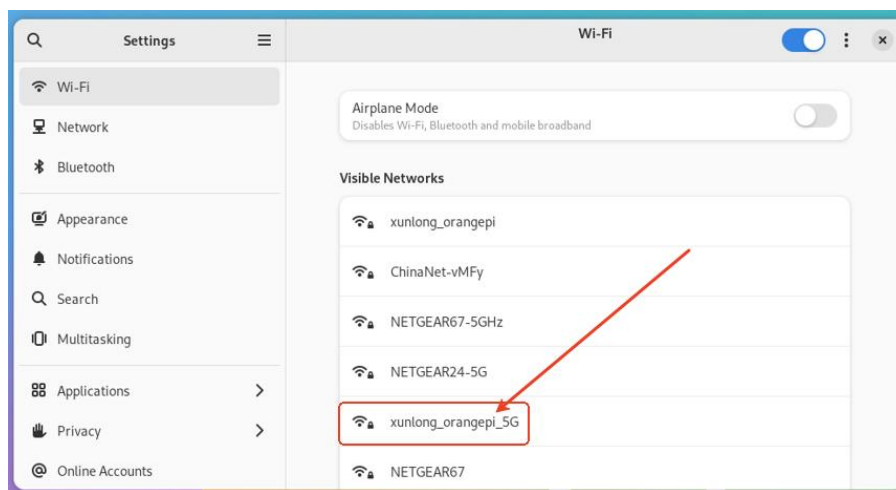
- 1) Click Settings (you don't need to connect an Ethernet cable when testing Wi-Fi).



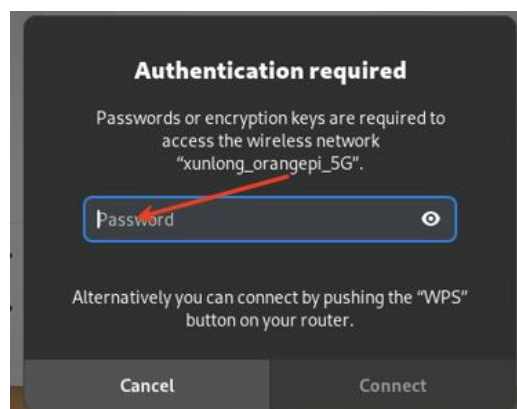
- 2) Click Settings (you don't need to connect an Ethernet cable when testing Wi-Fi).



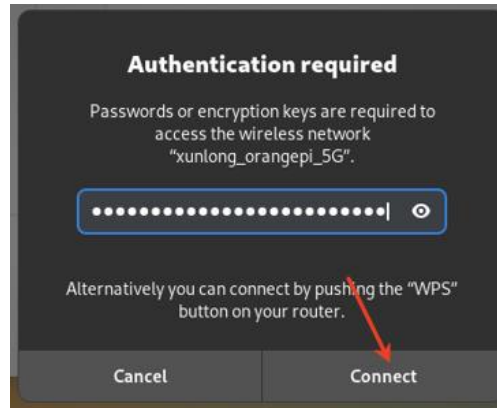
3) Then select the Wi-Fi hotspot you want to connect to.



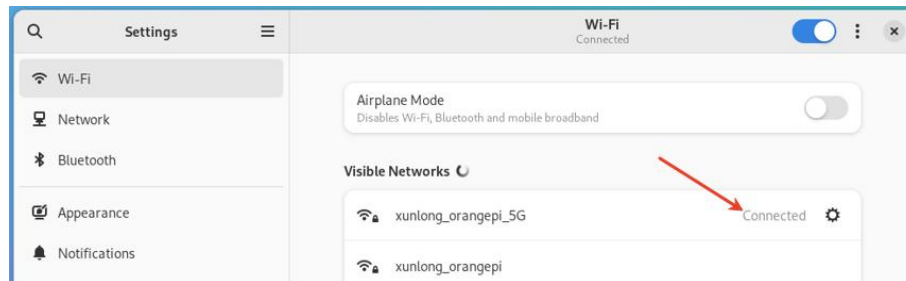
4) Then enter the password for the Wi-Fi hotspot.



5) Then click **Connect** to start the connection.



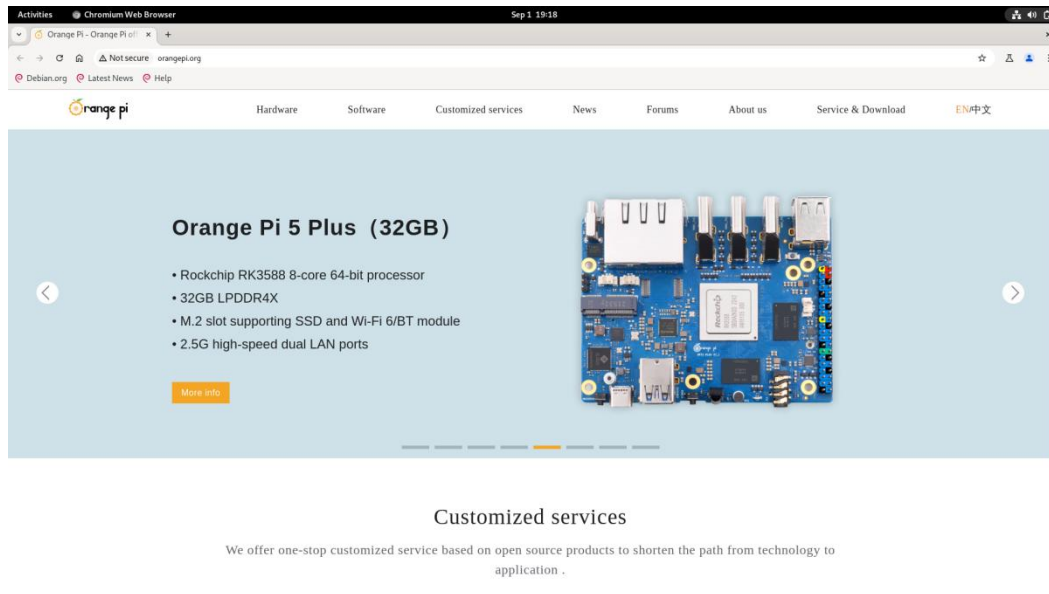
6) The display after Wi-Fi connection is completed is as follows:



7) After connecting to Wi-Fi, you can open the browser to check whether you can access the Internet. The browser entrance is shown in the figure below:



8) If you can access other web pages after opening the browser, it means the Wi-Fi connection is normal.



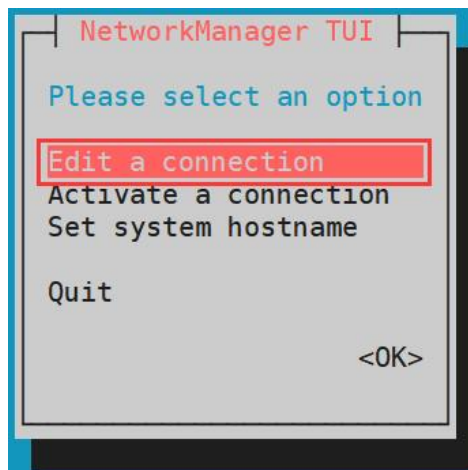
3. 6. 3. How to set a static IP address

3. 6. 3. 1. Using nmtui to set a static IP address

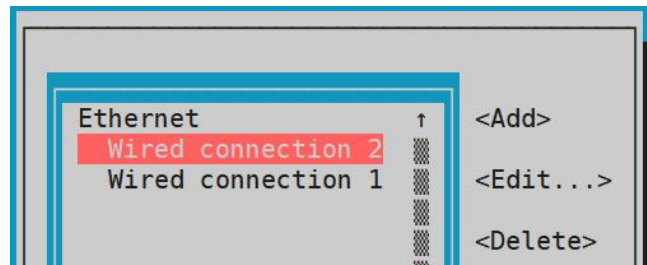
1) First run the **nmtui** command.

```
orangeipi@orangeipi:~$ sudo nmtui
```

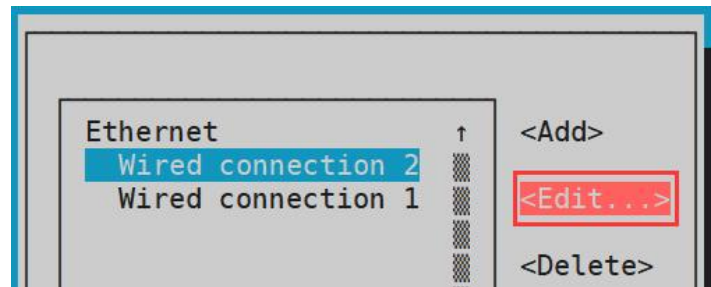
2) Then select **Edit a connection** and press Enter.



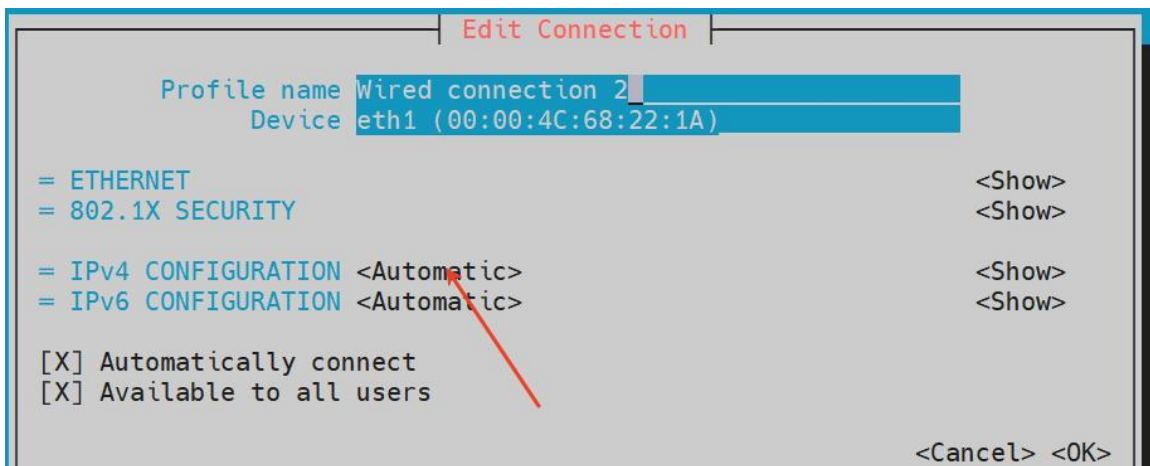
3) Then select the network interface for which you want to set a static IP address. For example, to set a static IP address for an **Ethernet** interface, select **Wired connection 1** or **Wired connection 2**.



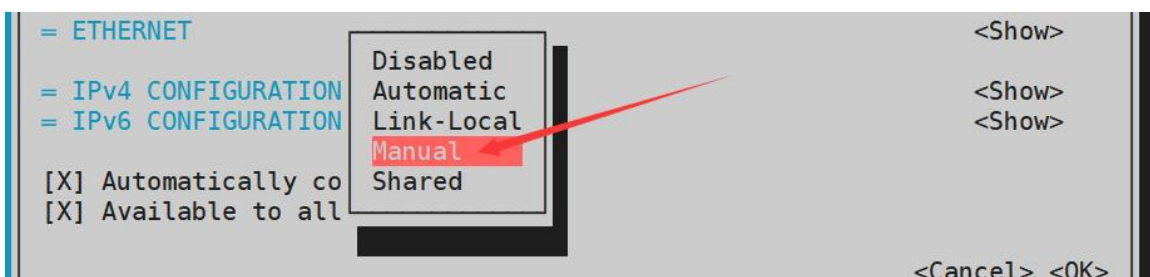
4) Then select **Edit** using the **Tab** key and press Enter.



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



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





7) The display after selection is as shown below:

```
= ETHERNET                                     <Show>
= IPv4 CONFIGURATION <Manual>                 <Show>
= IPv6 CONFIGURATION <Automatic>             <Show>

[X] Automatically connect
[X] Available to all users

<Cancel> <OK>
```

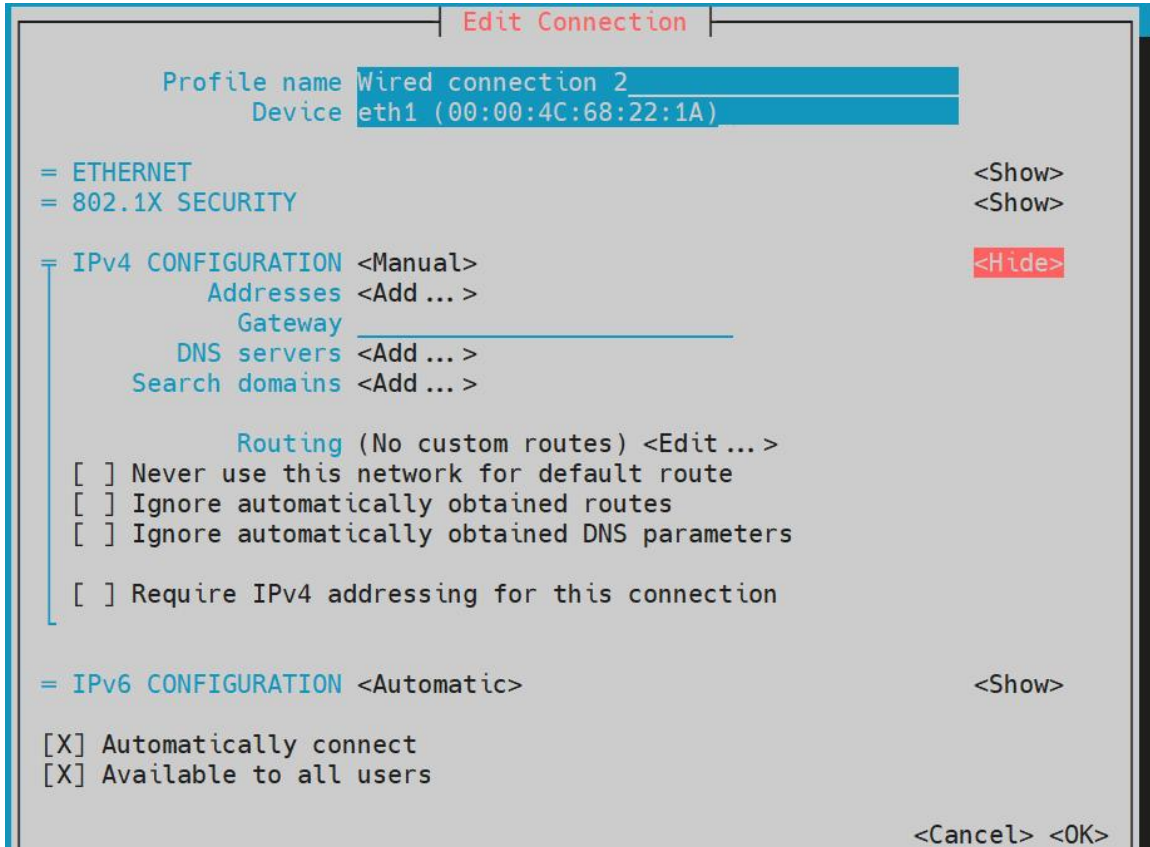
8) Then use the Tab key to move the cursor to **<Show>**.

```
= ETHERNET                                     <Show>
= IPv4 CONFIGURATION <Manual>                 <Show>
= IPv6 CONFIGURATION <Automatic>             <Show>

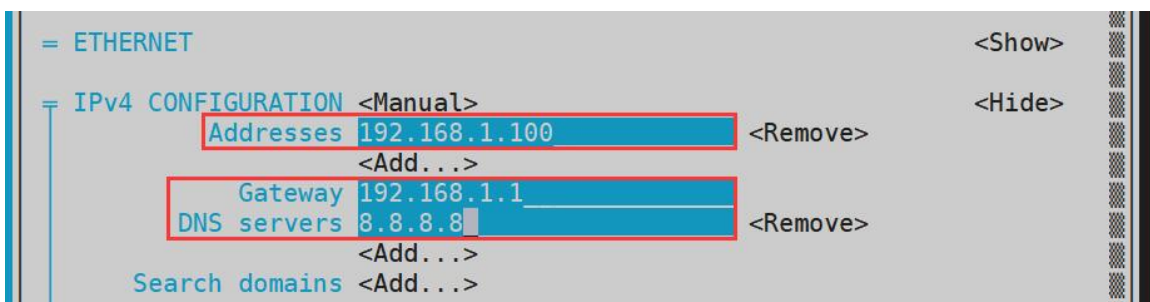
[X] Automatically connect
[X] Available to all users

<Cancel> <OK>
```

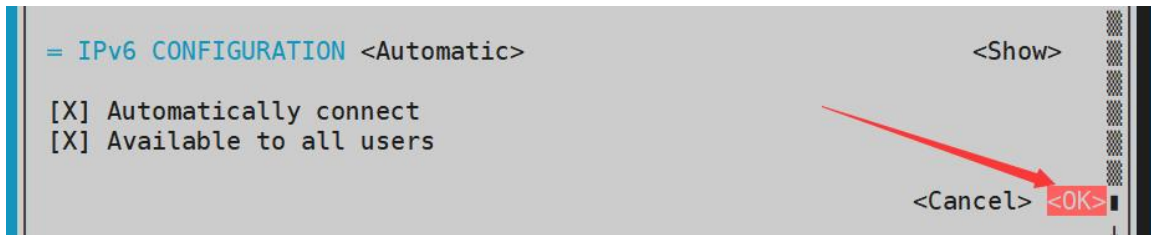
9) Press Enter, and the following setting interface will pop up.



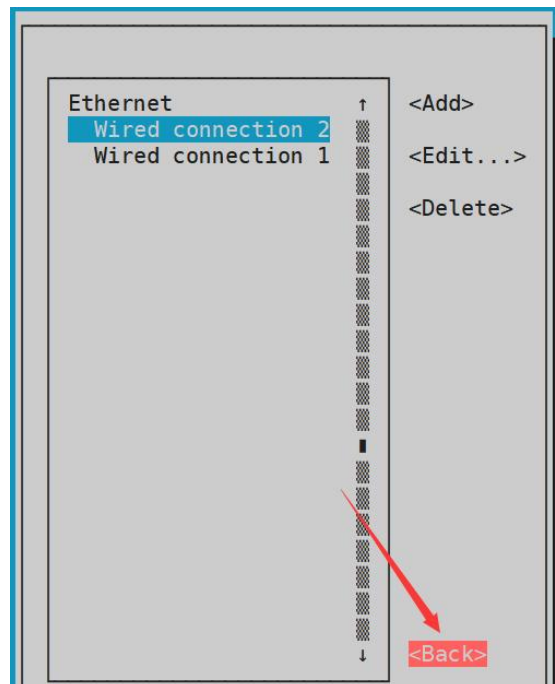
10) Then you can set the IP address (Addresses), gateway (Gateway) and DNS server address as shown in the figure below (there are many other settings options, please explore them yourself). Please set them according to your specific needs. **The values set in the figure below are just an example.**



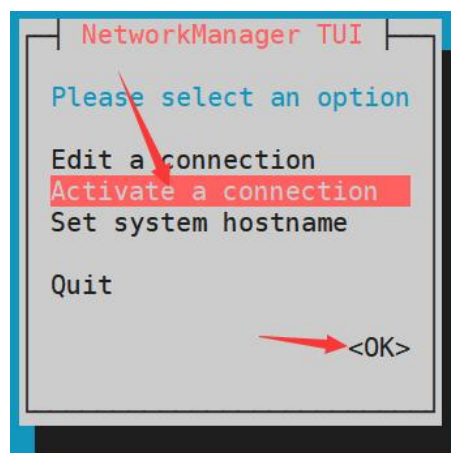
11) After setting, move the cursor to **<OK>** in the lower right corner and press Enter to confirm.



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

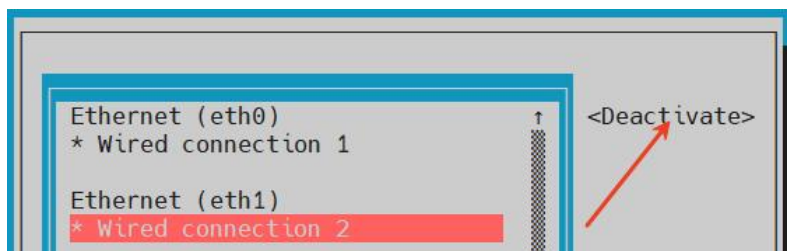


13) Then select **Activate a connection**, move the cursor to **<OK>**, and finally press Enter

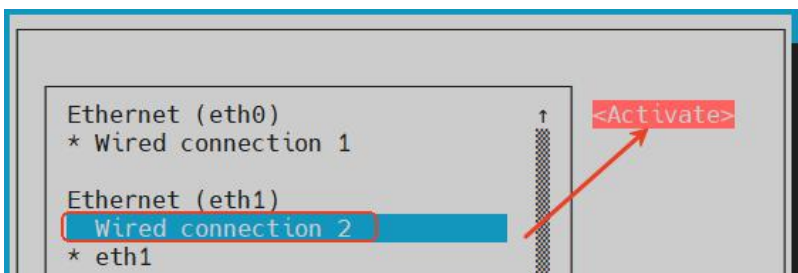




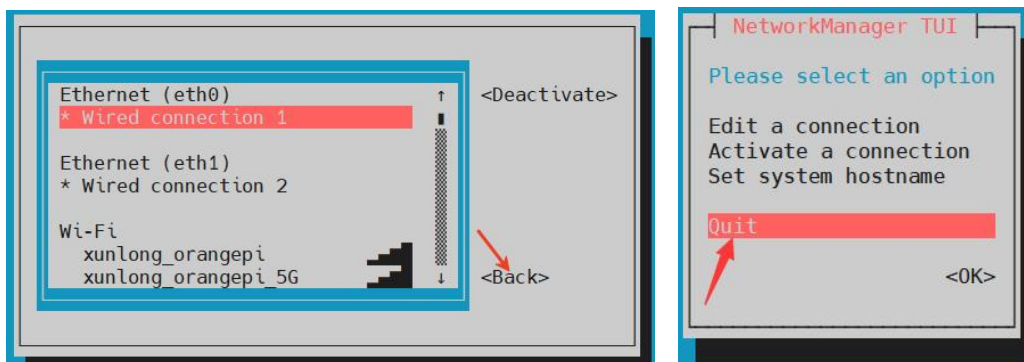
- 14) Then select the network interface you want to configure, such as **Wired connection 2**, move the cursor to **<Deactivate>**, and press Enter to disable **Wired connection 2**.



- 15) Then reselect and enable **Wired connection 2** so that the static IP set previously will take effect.



- 16) Then you can exit nmtui by pressing the **<Back>** and **Quit** buttons.



- 17) You can see through the following command that the IP address of the network port has become the static IP address set previously.

```
orange@orange:~$ sudo ifconfig eth1
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.1.100  netmask 255.255.255.0  broadcast 192.168.1.255
    inet6 fe80::1fb1:5c53:9479:5239  prefixlen 64  scopeid 0x20<link>
    ether 00:00:4c:68:22:1a  txqueuelen 1000  (Ethernet)
    RX packets 21289  bytes 5393604 (5.1 MiB)
```



```

RX errors 0   dropped 5   overruns 0   frame 0
TX packets 320   bytes 37621 (36.7 KiB)
TX errors 0   dropped 0 overruns 0   carrier 0   collisions 0
device interrupt 110

```

18) Then use a network cable to connect the development board's static IP address to another device's network port. Ensure that the other device's IP address is in the same network segment as the development board's static IP address.

19) Then test the network connectivity to check whether the IP address can be used normally. The **ping** command can be interrupted by pressing **Ctrl+C**.

```

orange@orange:~$ ping 192.168.1.47 -I eth1
PING 192.168.1.47 (192.168.1.47) from 192.168.1.100 eth1: 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
^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. 6. 3. 2. Using nmcli to set a static IP address

1) To set a static IP address for the network port, first plug the network cable into the development board. **To set a static IP address for Wi-Fi, first connect to Wi-Fi and then set the static IP address.**

2) Then use the **nmcli con show** command to view the names of the network devices. As shown below, **Wired connection 1** and **Wired connection 2** are the names of the Ethernet interfaces.

```

orange@orange:~$ nmcli con show

```

NAME	UUID	TYPE	DEVICE
Wired connection 1	c043c817-1156-3b72-a559-9a8cd642bf70	ethernet	enP3p49s0
Wired connection 2	6f74598a-ccc6-358b-be05-87eaf34df930	ethernet	enP4p65s0

3) Then enter the following command, where:



- a. **"Wired connection 2"** indicates setting a static IP address for the Ethernet port. If you need to set a static IP address for another network port, change it to the name of the corresponding network interface (which can be obtained using the `nmcli con show` command);
- b. **ipv4.addresses** The following field indicates the static IP address to be set. You can modify it to your desired value;
- c. **ipv4.gateway** indicates the gateway address.

```
orangepi@orangepi:~$ sudo nmcli con mod "Wired connection 2" \
ipv4.addresses "192.168.1.110" \
ipv4.gateway "192.168.1.1" \
ipv4.dns "8.8.8.8" \
ipv4.method "manual"
```

- 4) Then restart the Linux system.

```
orangepi@orangepi:~$ sudo reboot
```

- 5) Through the following command, you can see that the IP address of the network port has become the static IP address set previously.

```
orangepi@orangepi:~$ sudo ifconfig eth1
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.1.100  netmask 255.255.255.0  broadcast 192.168.1.255
    inet6 fe80::1fb1:5c53:9479:5239  prefixlen 64  scopeid 0x20<link>
    ether 00:00:4c:68:22:1a  txqueuelen 1000  (Ethernet)
    RX packets 21289  bytes 5393604 (5.1 MiB)
    RX errors 0  dropped 5  overruns 0  frame 0
    TX packets 320  bytes 37621 (36.7 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
    device interrupt 110
```

- 6) Then use a network cable to connect the development board's static IP address to another device's network port. Make sure the other device's IP address is in the same network segment as the development board's static IP address.

- 7) Then test the network connectivity to check whether the IP address can be used normally. The **ping** command can be interrupted by pressing **Ctrl+C**.



```

orangepi@orangepi:~$ ping 192.168.1.47 -I eth1
PING 192.168.1.47 (192.168.1.47) from 192.168.1.100 eth1: 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
^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

```

3.7. How to use the M.2 E-Key PCIe Wi-Fi 6+ Bluetooth module

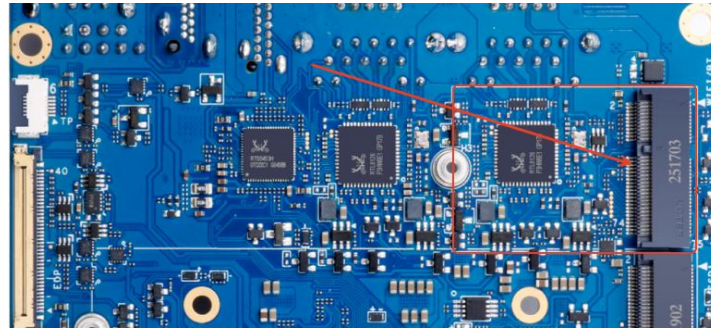
1) First, you need to purchase a PCIe Wi-Fi6+Bluetooth module. The table below shows the modules that have been adapted.

serial number	model	physical image	Supported OS
1	AX210 (PCIe+USB interface)		Debian
2	RTL8852BE (PCIe+USB interface)		Debian

Note: For RTL8852BE, please do not purchase the module shown in the figure below. It has been tested and used to cause problems.



2) Then insert the module into the M.2 E-Key interface of the development board and secure it in the position shown in the figure below:



3) Then use the `lspci` command. If you can see the Wi-Fi module information, it means the module is in good contact.

a. The AX210 display is as follows:

```
orangepi@orangepi:~$ lspci | grep "Network"
0002:21:00.0 Network controller: Intel Corporation Wi-Fi 6 AX210/AX211/AX411
160MHz (rev 1a)
```

b. The RTL8852 display is as follows:

```
orangepi@orangepi:~$ lspci | grep "Network"
0003:01:00.0 Network controller: Realtek Semiconductor Co., Ltd. RTL8852BE PCIe
802.11ax Wireless Network Controller
```

4) Then use the following command to see that there will be an additional Wi-Fi device node.

```
orangepi@orangepi:~$ sudo ifconfig
```



```
.....
wlan0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 64:49:7d:7b:18:9e txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

5) For Wi-Fi connection and testing methods, please refer to the [Wi-Fi connection test](#) section and will not be repeated here.

6) Use the following command to see a Bluetooth device.

```
orangePi@orangePi:~$ hciconfig
hci0: Type: Primary Bus: USB
    BD Address: 64:49:7D:7B:18:A2 ACL MTU: 1021:4 SCO MTU: 96:6
    UP RUNNING PSCAN
    RX bytes:18252 acl:0 sco:0 events:2890 errors:0
    TX bytes:710514 acl:0 sco:0 commands:2888 errors:0
```

7) For the Bluetooth testing method, please refer to [the Bluetooth Usage section](#), which will not be repeated here.

3. 8. SSH remote login to the development board

By default, Linux systems have SSH remote login enabled and allow the root user to log in. Before logging in via SSH, ensure that you are connected to an Ethernet or Wi-Fi network. Then, use the `sudo ifconfig` command or check your router to obtain the IP address of the development board.

3. 8. 1. SSH remote login to the development board under Ubuntu

1) Get the IP address of the development board.

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

```
test@test:~$ ssh root@10.31.3.166 #Need to be replaced with the IP address
of the development board
root@10.31.3.166's password: #Enter the password here. The default
```

password is orangepi

Please note that when entering the password, the specific content of the password will not be displayed on the screen. Just press Enter after entering it.

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

```
orangepi@orangepi:~$ ssh orangepi@10.31.3.166
The authenticity of host '10.31.3.166 (10.31.3.166)' can't be established.
ED25519 key fingerprint is SHA256:dMqsLCAurxLxKSR2Xy+ftTctj6XQrEQU4HREF/NcM.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.31.3.166' (ED25519) to the list of known hosts.
orangepi@10.31.3.166's password:
```

Math 101

```
Welcome to Orange Pi 1.0.0 Bookworm with Linux 6.1.44-cix
```

```
System load: 34%      Up time: 4 min      Local users: 4
Memory usage: 3% of 30.55G
IP: 10.31.3.171
CPU temp: 43°C       Usage of /: 55% of 9.7G
```

```
[ 0 security updates available, 7 updates total: apt upgrade ]
Last check: 2025-08-29 13:45
```

Last login: Fri Aug 29 13:47:42 2025

```
orangePi@orangePi6plus:~$
```

If SSH fails to log in to the Linux system, first check whether the IP address of the development board can be pinged. If the ping is successful, log in to the Linux system through the serial port or the monitor and then enter the following command on the development board to try to connect again:

```
orangepi@orangepi:~$ sudo rm /etc/ssh/ssh_host_*
```

```
orangepi@orangepi:~$ sudo dpkg-reconfigure openssh-server
```

3. 8. 2. SSH remote login to the development board under Windows

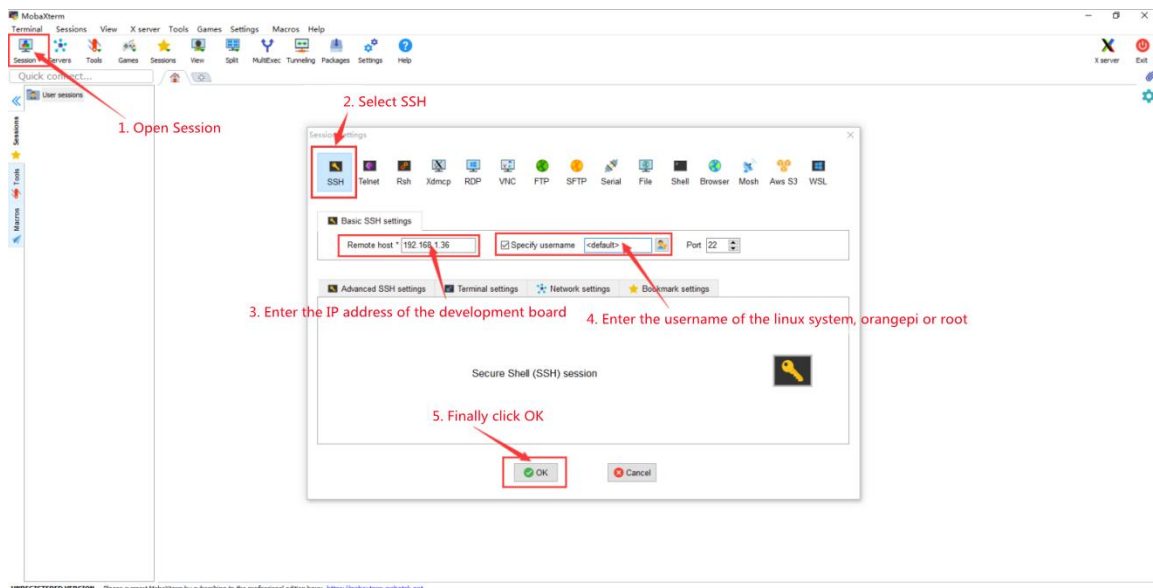
- 1) First, obtain the IP address of the development board.

2) Under Windows, you can use MobaXterm to remotely log in to the development board. First, create a new ssh session.

- Open **Session**.
- Select **SSH** in **Session Settings**.
- Enter the development board's IP address in **Remote host**.
- Enter your Linux username, such as **root** or **orangeipi**, in **Specify username**.

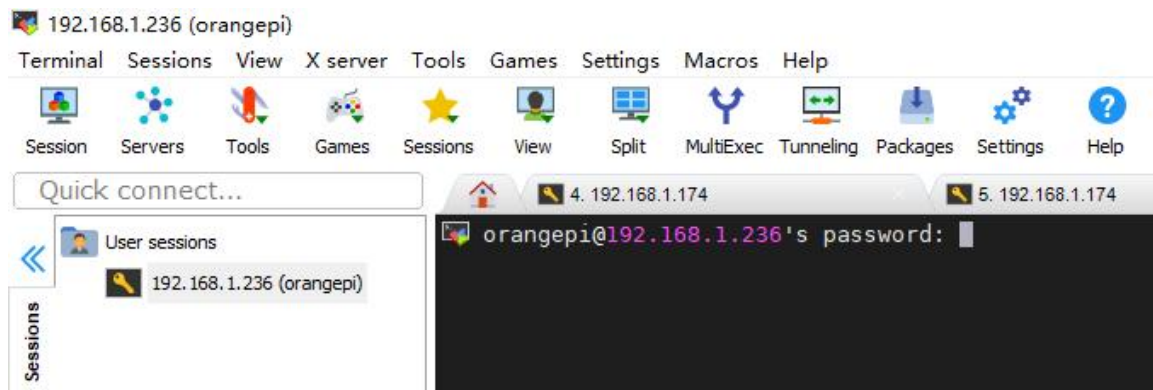


e. Click OK.

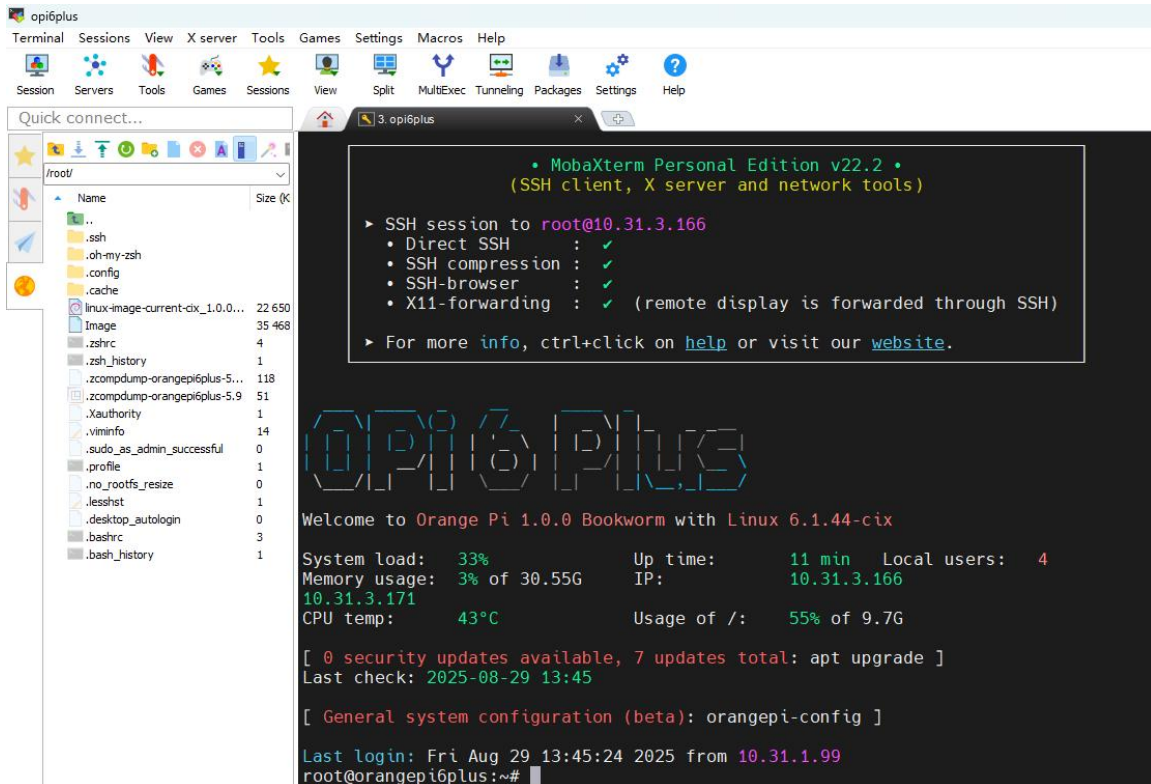


3) You will be prompted to enter a password. The default password for both root and orangepi users is orangepi.

Please note that when entering the password, the specific content of the password will not be displayed on the screen. Just press Enter after entering it.



4) After successfully logging into the system, the display is as shown below:



3.9. How to upload files to the Linux system of the development board

3.9.1. How to upload files from Ubuntu PC to the Linux system of the development board

3.9.1.1. How to upload files using the scp command

1) Use the scp command to upload files from the Ubuntu PC to the Linux system of the development board. The specific commands are as follows

- file_path:** Need to be replaced with the path of the file to be uploaded
- orangeipi:** This is the user name of the development board's Linux system. You can also replace it with other names, such as root.
- 192.168.xx.xx:** The IP address of the development board. Please modify it according to the actual situation.
- /home/orangeipi:** The path in the Linux system of the development board can also be modified to other paths



```
test@test:~$ scp file_path orangepi@192.168.xx.xx:/home/orangepi/
```

2) If you want to upload a folder, you need to add the -r parameter

```
test@test:~$ scp -r dir_path orangepi@192.168.xx.xx:/home/orangepi/
```

3) There are more uses for scp. Please use the following command to view the man page

```
test@test:~$ man scp
```

3. 9. 1. 2. How to upload files using FileZilla

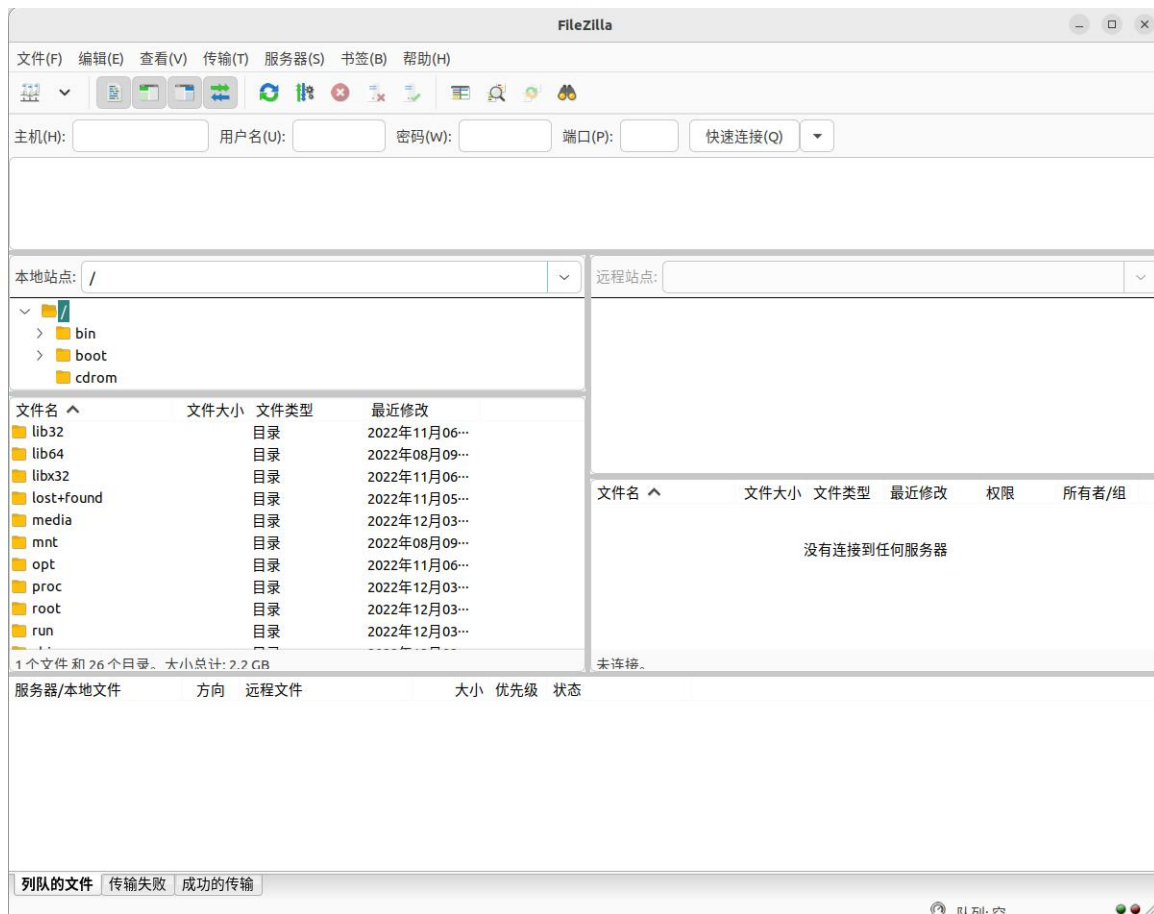
1) First install filezilla in your Ubuntu PC

```
test@test:~$ sudo apt install -y filezilla
```

2) Then open filezilla using the following command

```
test@test:~$ filezilla
```

3) The interface after opening FileZilla is as follows. At this time, the remote site on the right is empty.



4) The method of connecting the development board is shown in the figure below



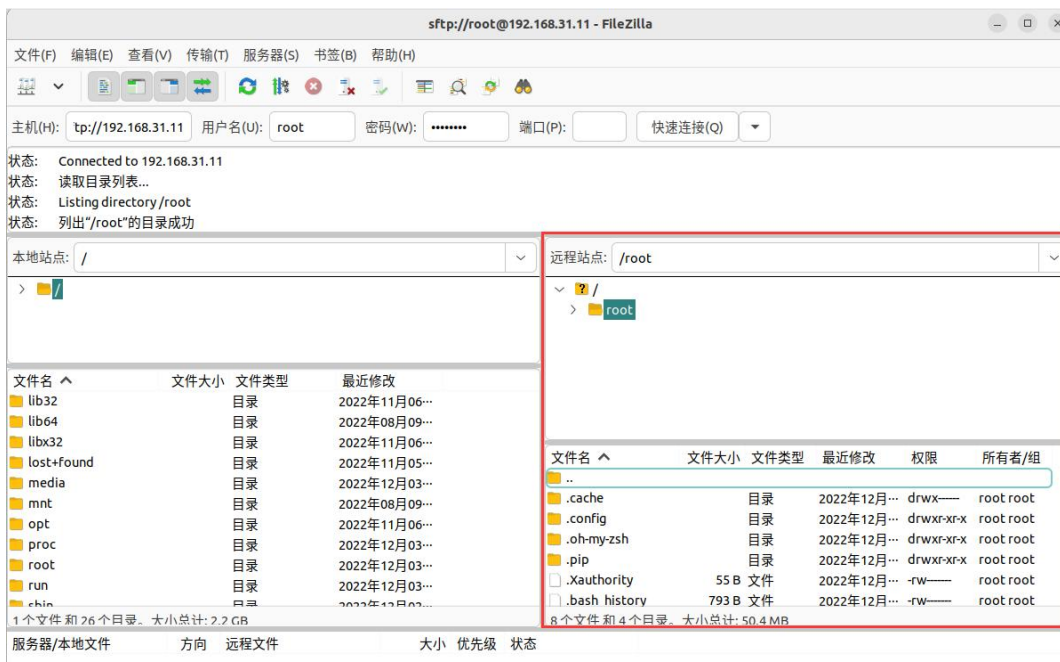
5) Then select **Save Password** and click **OK**



6) Then select **Always trust this host** and click **OK**



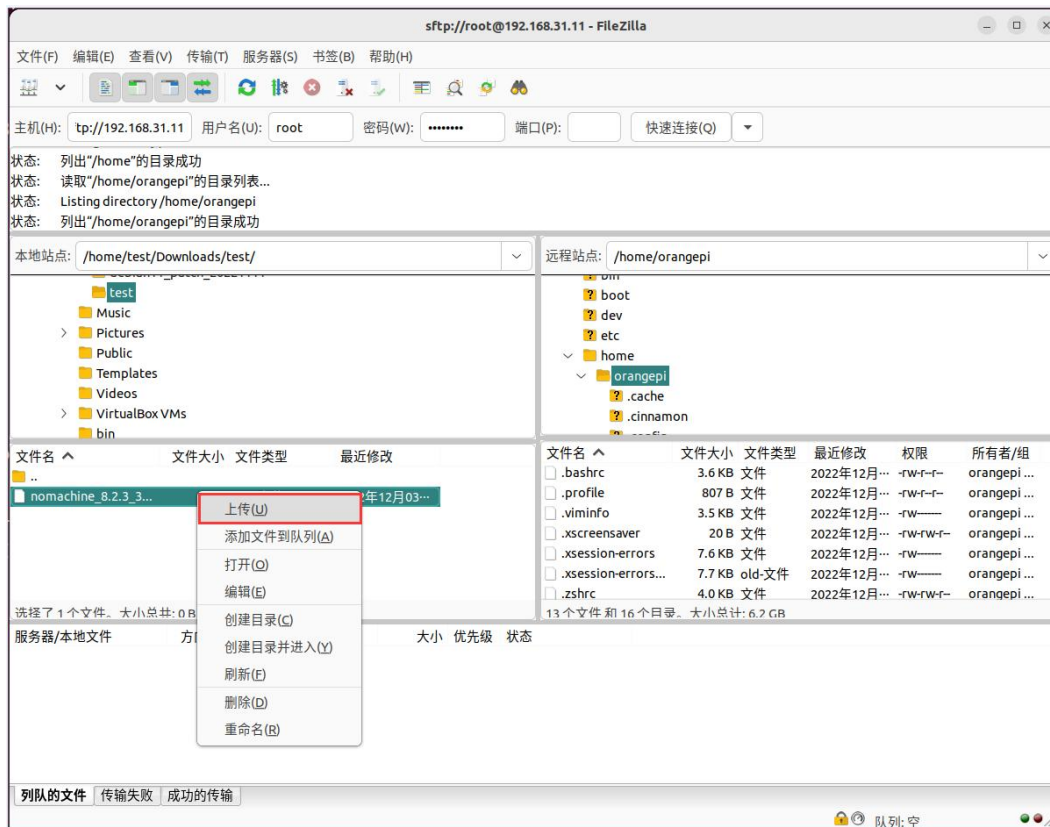
7) After the connection is successful, you can see the directory structure of the development board's Linux file system on the right side of the filezilla software



8) Then select the path to upload to the development board on the right side of the



FileZilla software, then select the file to be uploaded in the Ubuntu PC on the left side of the FileZilla software, right-click the mouse, and then click the Upload option to start uploading the file to the development board.



9) After uploading, you can check the uploaded files in the corresponding path of the development board Linux system.

10) The method of uploading folders is the same as that of uploading files, so I will not go into details here.

3. 9. 2. . How to upload files from Windows PC to the Linux system of the development board

3. 9. 2. 1. How to upload files using FileZilla

1) First download the installation file of the Windows version of the filezilla software. The download link is as follows

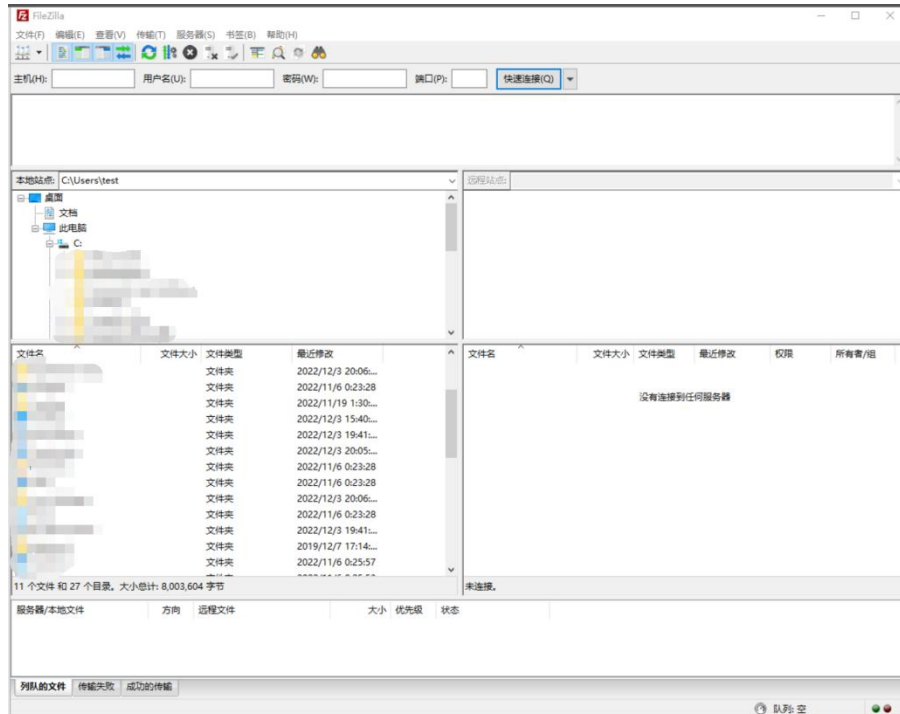
<https://filezilla-project.org/download.php?type=client>



2) The downloaded installation package is as follows, then double-click to install it directly

FileZilla_3.69.3_win64_sponsored2-setup.exe

3) The interface after opening FileZilla is as follows. At this time, the remote site on the right is empty.



4) The method of connecting the development board is shown in the figure below:



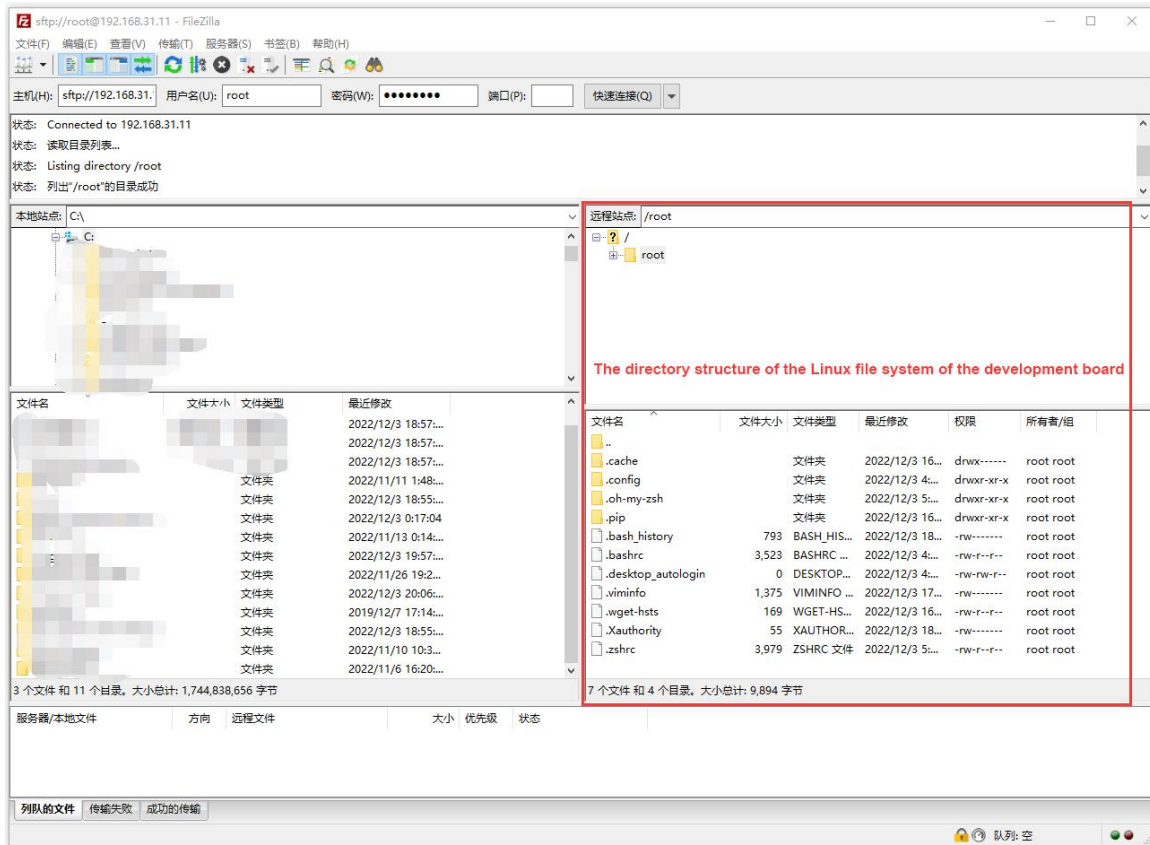
5) Then select **Save Password** and click **OK**



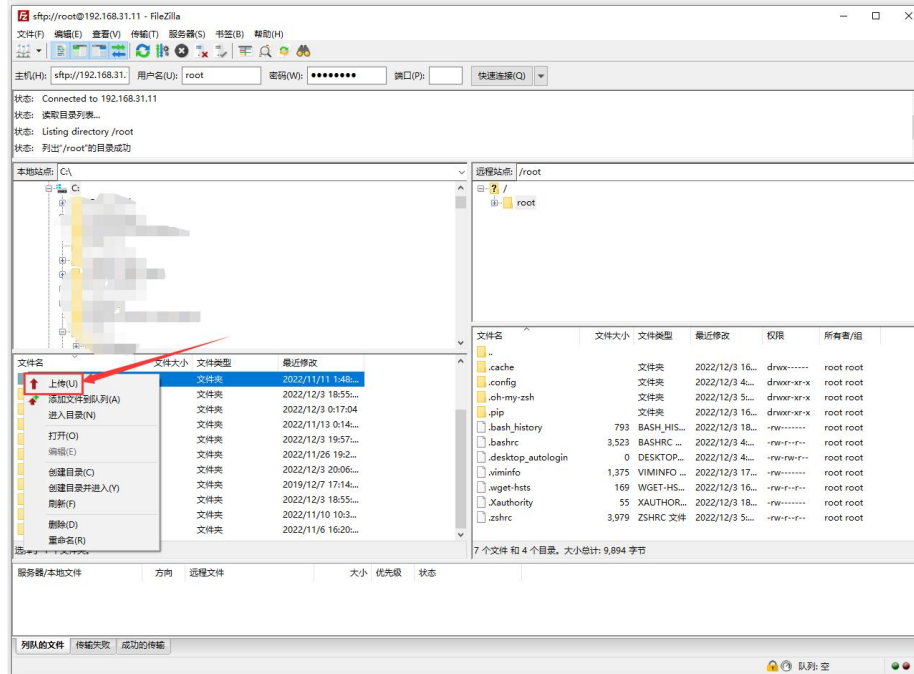
6) Then select **Always trust this host** and click **OK**



7) After the connection is successful, you can see the directory structure of the development board's Linux file system on the right side of the filezilla software



8) Then select the path to upload to the development board on the right side of the filezilla software, then select the file to upload in the Windows PC on the left side of the filezilla software, right-click the mouse, and then click the upload option to start uploading the file to the development board



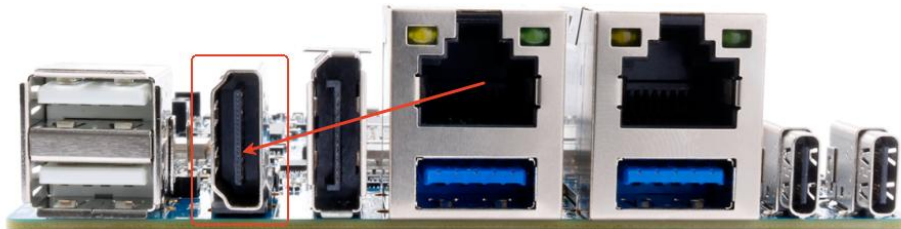
9) After uploading, you can check the uploaded files in the corresponding path of the development board Linux system.

10) The method of uploading folders is the same as that of uploading files, so I will not go into details here.

3. 10. Display function test

3. 10. 1. HDMI display output test

1) The development board has one HDMI port, which is located as shown in the figure below:



2) Then use an HDMI to HDMI cable to connect the development board and the HDMI display.



Note: If you are connecting to a 4K monitor, please make sure the HDMI cable supports 4K display.

3) After starting the Linux system, if the HDMI monitor outputs images, it means that the HDMI interface is working properly.

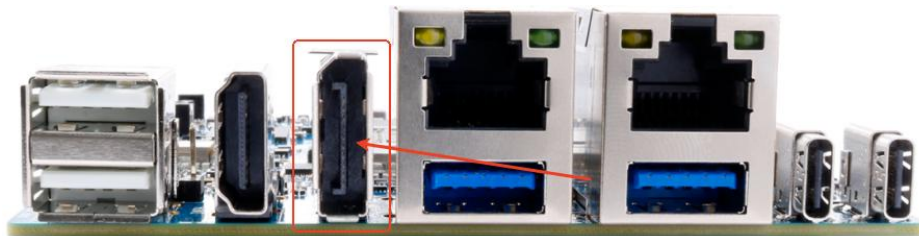
Note that while many laptops have HDMI ports, these ports typically only function as outputs and lack HDMI input functionality. This means that the development board's HDMI output cannot be displayed on the laptop screen.

When connecting the development board's HDMI port to a laptop's HDMI port, please ensure that your laptop supports HDMI input.

When there is no display on HDMI, please first check whether the HDMI cable is plugged in tightly. After confirming that the connection is OK, you can try to use a different screen to see if there is any display.

3. 10. 2. DP display output test

1) The development board has one DP interface, the location of which is shown in the figure below:



2) Then use a DP to HDMI cable to connect the development board and the HDMI monitor.

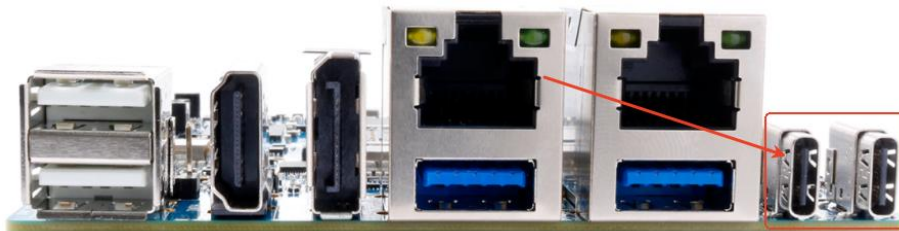


Note: If you are connecting to a 4K monitor, please make sure that the DP to HDMI cable supports 4K display.

3) After starting the Linux system, if the HDMI monitor has image output, it means that the DP interface is working properly.

3. 10. 3. Type-C to HDMI display output test

1) The development board has two Type-C interfaces, both of which support display functions. Their locations are shown in the figure below:



2) Then use a Type-C to HDMI cable to connect the development board and the HDMI display.



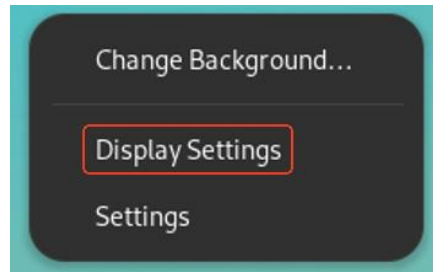
Note: If you are connecting to a 4K monitor, please make sure that the Type-C to HDMI cable supports 4K display.

3) After booting the Linux system, if the HDMI display has image output, it means that the Type-C display function is working properly.

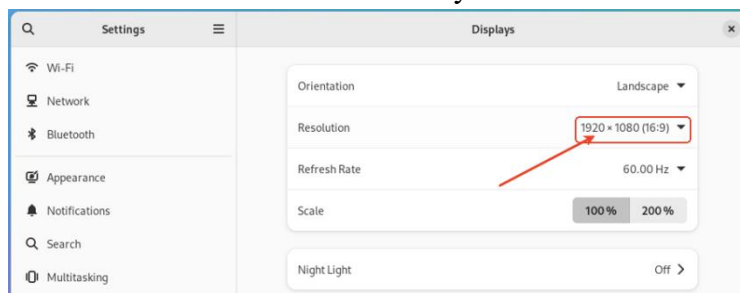
3. 10. 4. How to set the display resolution

1) Right-click on the desktop.

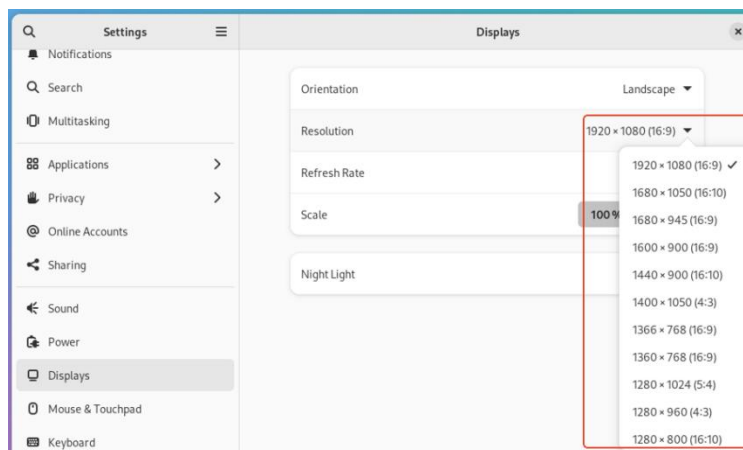
2) Then select **Display Settings**.



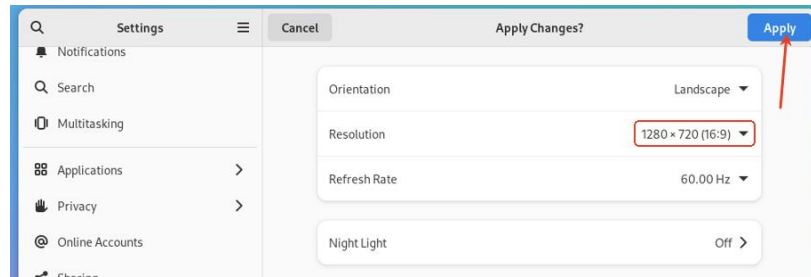
3) Then you can see the current resolution of the system.



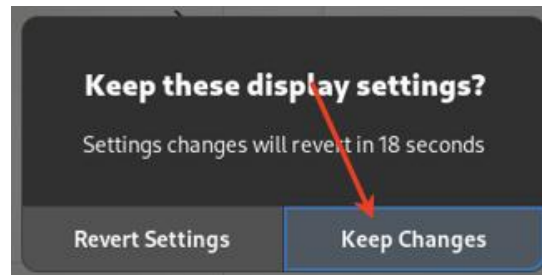
4) Click the drop-down box of Resolution to see all the resolutions currently supported by the monitor.



5) Then select the resolution you want to set and click Apply.



6) After the new resolution is set, select **Keep Changes**.



3. 10. 5. Instructions for using the Edp screen

1) Prepare an edp screen as shown in the following image, which only supports display and does not support touch.



2) The method of connecting the edp screen to the development board is as follows:

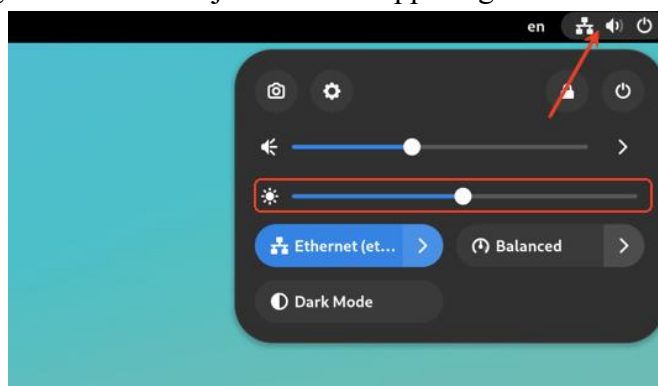




3) After the edp screen is connected, the software does not require any other settings and will display the system interface by default after booting up.



4) The screen brightness can be adjusted in the upper right corner of the desktop system.



3. 11. How to use Bluetooth

Please note that there is no Bluetooth module on the development board. You need to connect an external M.2 E-key network card with Bluetooth or a USB network card with Bluetooth to use the Bluetooth function.

For instructions on using the external M.2 E-key network card with Bluetooth, please refer to the section "[Using the M.2 E-Key PCIe Wi-Fi6+ Bluetooth Module.](#)"

For instructions on using the external USB network card, please refer to [the](#)



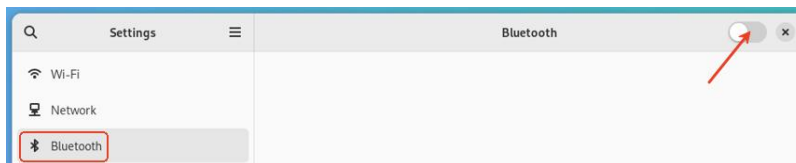
USB wireless network card testing section.

3. 11. 1. How to test Bluetooth on the desktop

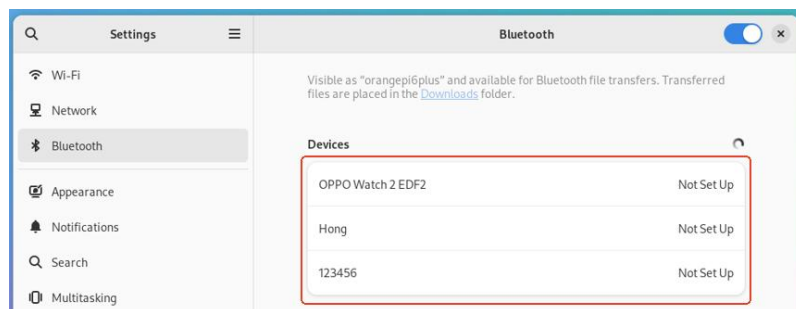
1) Open Settings.



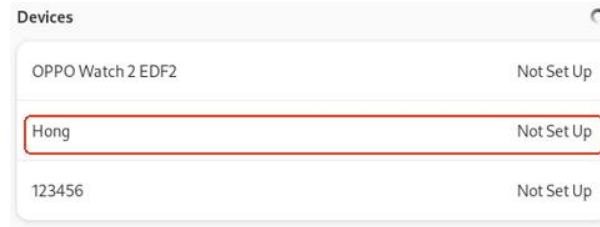
2) Then make sure the Bluetooth switch is turned on.



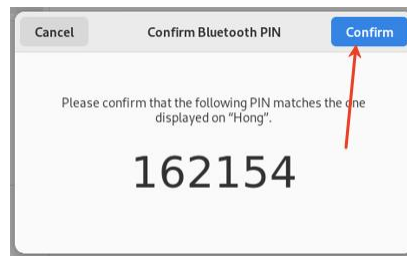
3) After Bluetooth is turned on, you can see the scanned Bluetooth devices in **Devices**.



4) Then select the Bluetooth device you want to connect to, such as the Bluetooth of the Android phone Hong.



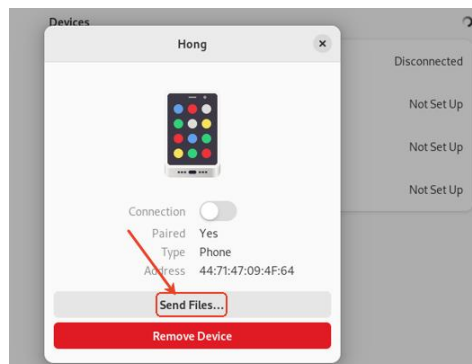
5) During pairing, a confirmation box will pop up as shown below. Select **Confirm**. Confirmation will also be required on your phone.



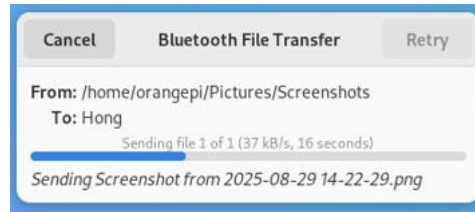
6) After pairing with your phone, you can select the Bluetooth device you just paired with.



7) Then select **Send Files...** to start sending a file, such as a picture, to your phone.



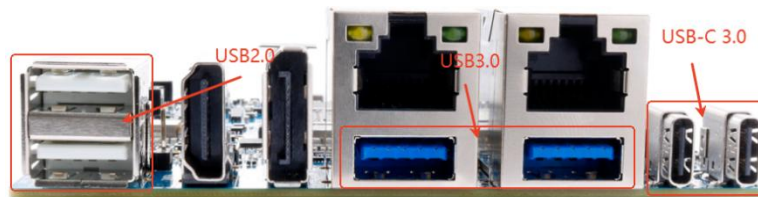
8) The interface for sending files is as follows:



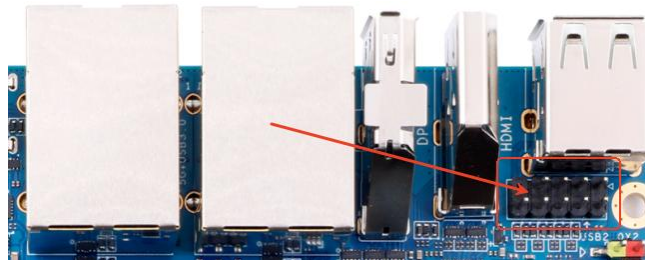
3. 12. USB interface test

3. 12. 1. 9-pin pin header to expand two USB2.0 ports

1) The USB interfaces directly available on the development board are as follows:



2) In addition, there is a 9-pin pin header on the development board that can be used to expand two USB2.0 Host interfaces. Its location is as follows:



3) To expand the USB port via a 9-pin header, you need to prepare a cable as shown in the figure below.



4) The expansion cable connector shown in the figure below needs to be plugged into the



9-pin header on the development board. Note that there is a solid pin that corresponds to the pin without a header on the development board.



5) The other end of the extension cable can then be used to connect two USB 2.0 devices.



3. 12. 2. Test method for connecting a USB mouse or keyboard

- 1) Plug the USB keyboard into the USB port of the development board.
- 2) Connect the development board to the monitor.
- 3) If the mouse or keyboard can operate the system normally, it means that the USB interface is working properly (the mouse can only be used in the desktop version of the system).

3. 12. 3. Test method for connecting USB storage devices

- 1) First, insert the USB flash drive or USB mobile hard drive into the USB port of the development board.
- 2) Execute the following command. If you can see the output of sdX, it means that the USB drive has been successfully recognized.

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



8	1	30043119 sda1
---	---	----------------------

3) If the USB drive does not mount automatically, you can use the mount command to mount the USB drive to **/mnt**, and then you can view the files in the USB drive.

```
orangepi@orangepi:~$ sudo mount /dev/sda1 /mnt/
orangepi@orangepi:~$ ls /mnt/
test.txt
```

4) After mounting, you can use the **df -h** command to view the capacity usage and mount point of the USB drive.

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

3. 12. 4. USB Camera Test Method

1) First, you need to prepare a USB camera that supports the UVC protocol, as shown in the figure below or a similar one, and then insert the USB camera into the USB port of the development board.



2) Use the **v4l2-ctl** command to see that the device node information of the USB camera is **/dev/video2**.

```
orangepi@orangepi:~$ v4l2-ctl --list-devices | grep -A 1 -i "usb"
Q8 HD Webcam: Q8 HD Webcam (usb-xhci-hcd.3.auto-1):
    /dev/video2
```

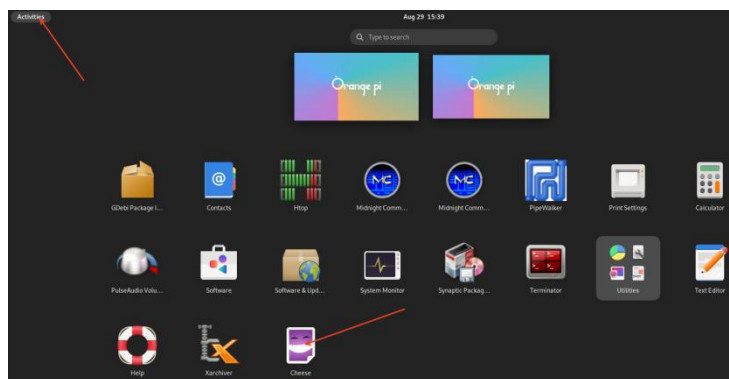
Note that the l in v4l2 is a lowercase letter l, not the number 1.

In addition, the serial number of the video may not always be video2. Please refer to the actual video you see.

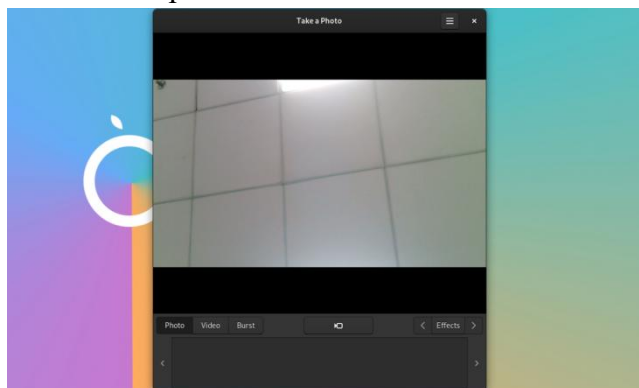
3) You can use Cheese to directly open the USB camera on the desktop system. The



Cheese opening method is shown in the figure below:

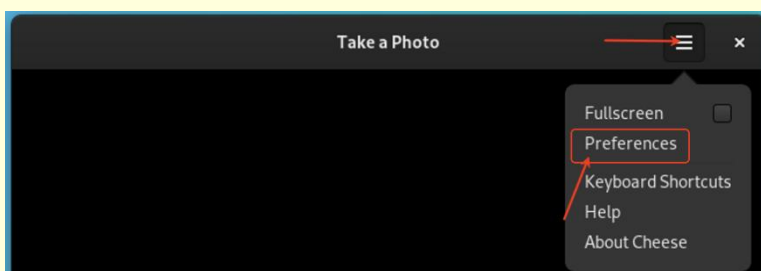


The interface after Cheese opens the USB camera is as shown below:

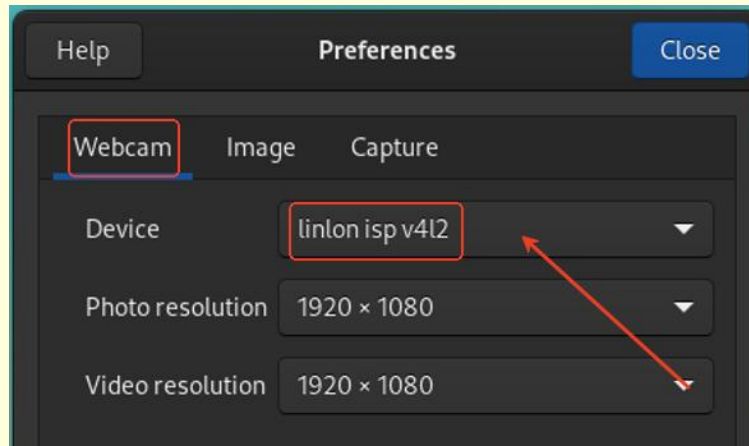


If the camera is not displaying, please select the USB camera in the Device as follows:

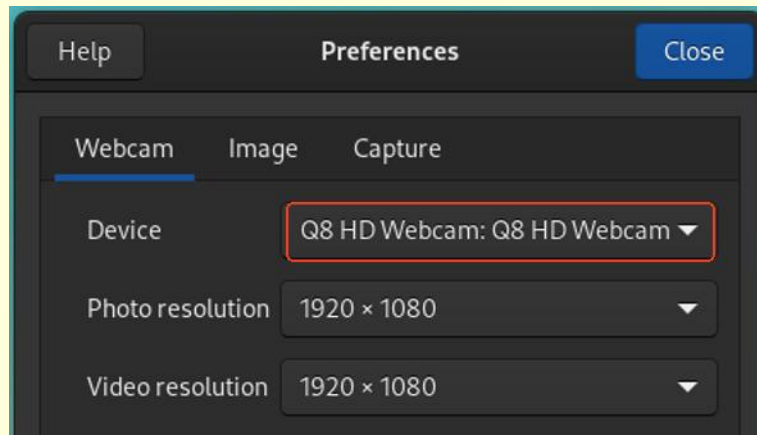
1. Open Preferences.



2. Then click on the dropdown menu of Device.



3. Then select the Device device node corresponding to the USB camera.



4. Then Cheese can see the preview screen of the USB camera.

4) Use fswebcam to test the USB camera.

a. Install fswebcam

```
orange@orange:~$ sudo apt update
```

```
orange@orange:~$ sudo apt-get install -y 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 is used to remove the watermark of the photo

c) -r option is used to specify the resolution of the photo

d) -S option is used to set the number of frames to skip ahead

e) ./image.jpg is used to set the name and path of the generated photo

```
orange@orange:~$ sudo fswebcam -d /dev/video2 \
```

```
--no-banner -r 1280x720 -S 5 ./image.jpg
```



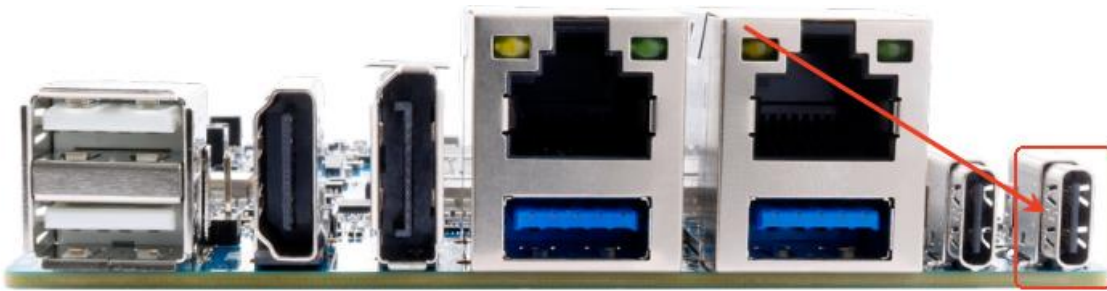
- c. After taking the picture, you can use the **scp** command to transfer the picture to the Ubuntu PC for viewing.

```
orange@orange:~$ scp image.jpg test@192.168.1.55:/home/test ( Please modify the IP address and path according to the actual situation )
```

- d. On the desktop of the development board, you can directly view the captured images through the monitor.

3. 12. 5. Type-C USB0 Device Test Method

- 1) The development board has two Type-C interfaces. Only the Type-C1 interface supports the Device function, while the Type-C2 interface does not. The location of the Type-C1 interface is as follows:



- 2) Use a USB 2.0 or USB 3.0 cable to connect the Type-C1 port to a Windows computer.



- 3) When the Type-C1 port is connected to a computer via a data cable, it will automatically switch to device mode.

- a. Linux6.1:

```
orange@orange:~$ cat /sys/class/usb_role/9010000.usb-controller-role-switch/role  
device
```

- b. Linux6.6 ACPI:

```
orange@orange:~$ cat /sys/class/usb_role/CIXH2031:00-role-switch/role  
device
```

- 4) Use the following command on the development board to virtualize the Type-C port



into a network port.

a. Linux6.1:

```
orange@orange:~$ sudo usb_gadget_setup.sh
Setting up USB NCM Gadget...
Create gadget directory...
Setting the device descriptor...
Set string descriptor...
Create configuration...
Creating NCM Functions...
Linking functionality to configuration...
Enable USB device controller...
Enabled UDC: 9010000.usb-controller
USB NCM Gadget Configuration complete!
The device should now be recognized as a USB network device.
```

b. Linux6.6:

```
orange@orange:~$ sudo usb_gadget_setup.sh
Setting up USB NCM Gadget...
Create gadget directory...
Set device descriptor...
Set string descriptor...
Create configuration...
Creating NCM Capabilities...
Linking functionality to configuration...
Enable USB device controller...
Enabled UDC: CIXH2031:00
USB NCM Gadget Configuration complete!
The device should now be recognized as a USB network device.
```

5) After the settings are completed, you can see an additional USB0 network port in the Linux system of the development board.

```
orange@orange:~$ sudo ifconfig usb0
usb0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 169.254.84.56  netmask 255.255.0.0  broadcast 169.254.255.255
    inet6 fe80::13b4:624b:e730:ca0a  prefixlen 64  scopeid 0x20<link>
    ether 2e:09:41:b9:58:ed  txqueuelen 1000  (Ethernet)
```



```
RX packets 23   bytes 3566 (3.4 KiB)
RX errors 0   dropped 0   overruns 0   frame 0
TX packets 30   bytes 5896 (5.7 KiB)
TX errors 0   dropped 0 overruns 0   carrier 0   collisions 0
```

6) Use the following command to set a static IP address for the USB0 network port of the development board.

```
orangePi@orangePi:~$ sudo ip link set usb0 down
orangePi@orangePi:~$ sudo ip addr flush dev usb0
orangePi@orangePi:~$ sudo ip addr add 192.168.10.1/24 dev usb0
orangePi@orangePi:~$ sudo ip link set usb0 up
```

7) If everything is normal, a virtual USB network port will appear on your Windows computer. If you don't see the USB network port, try unplugging and plugging the data cable. **The network port name shown below will change, so please refer to the actual port name.**



8) Then set a static IP address for the USB network port on your Windows computer. Make sure the set IP address is in the same network segment as the IP address of the USB0 network port on the development board.



9) Then test that both directions can be pinged successfully.

a. Windows computer pings the development board

```
Windows PowerShell
PS C:\Users\xunlong> ping 192.168.10.1

正在 Ping 192.168.10.1 具有 32 字节的数据:
来自 192.168.10.1 的回复: 字节=32 时间=1ms TTL=64
来自 192.168.10.1 的回复: 字节=32 时间=1ms TTL=64
来自 192.168.10.1 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.10.1 的回复: 字节=32 时间=1ms TTL=64

192.168.10.1 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 0ms, 最长 = 1ms, 平均 = 0ms
PS C:\Users\xunlong> |
```

b. The development board pings the Windows computer.

```
orangeypi@orangeypi:~$ ping 192.168.10.2 -I usb0
```

```
root@orangepi6plus:~# ping 192.168.10.2 -I usb0
PING 192.168.10.2 (192.168.10.2) from 192.168.10.1 usb0: 56(84) bytes of data.
64 bytes from 192.168.10.2: icmp_seq=1 ttl=128 time=0.904 ms
64 bytes from 192.168.10.2: icmp_seq=2 ttl=128 time=0.885 ms
64 bytes from 192.168.10.2: icmp_seq=3 ttl=128 time=0.973 ms
64 bytes from 192.168.10.2: icmp_seq=4 ttl=128 time=1.50 ms
64 bytes from 192.168.10.2: icmp_seq=5 ttl=128 time=1.53 ms
^C
--- 192.168.10.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 0.885/1.157/1.527/0.291 ms
root@orangepi6plus:~#
```

3. 12. 6. USB Wireless Network Card Test

The currently **tested** USB wireless network cards are shown below. Please test other models of USB wireless network cards yourself. If they do not work, you need to



transplant the corresponding USB wireless network card driver

Serial number	model	Actual picture
1	RTL8723BU Support 2.4G WIFI+BT4.0	
2	RTL8811 support 2.4G +5G WIFI	

3. 12. 6. 1. RTL8723BU Testing

1) First, insert the RTL8723BU wireless network card module into the USB port of the development board.

2) The Linux system will then automatically load the RTL8723BU Bluetooth and WIFI related kernel modules. You can see through the `lsmod` command that the following kernel modules have been automatically loaded.

```
orangepi@orangepi:~$ lsmod | grep rtl
rtl8xxxu          122880    0
```

3) You can see the loading information of the RTL8723BU module through the `dmesg` command

```
orangepi@orangepi:~$ dmesg
.....
[ 51.228574] [2025:09:26 07:45:24][pid:162,cpu10,kworker/10:1]usb 5-1: new high-speed USB device
number 2 using xhci-hcd
[ 51.382897] [pid:471,cpu0,kworker/u25:1]Bluetooth: hci0: RTL: examining hci_ver=06 hci_rev=000b
lmp_ver=06 lmp_subver=8723
[ 51.383883] [pid:471,cpu0,kworker/u25:1]Bluetooth: hci0: RTL: rom_version status=0 version=1
[ 51.383894] [pid:471,cpu0,kworker/u25:1]Bluetooth: hci0: RTL: loading rtl_bt/rtl8723b_fw.bin
[ 51.430131] [pid:471,cpu0,kworker/u25:1]Bluetooth: hci0: RTL: loading rtl_bt/rtl8723b_config.bin
[ 51.430380] [pid:471,cpu0,kworker/u25:1]Bluetooth: hci0: RTL: cfg_sz 68, total sz 22564
```



```
[ 51.451118] [pid:4031,cpu10,(udev-worker)]usb 5-1: This Realtek USB WiFi dongle (0x0bda:0xb720)
is untested!
[ 51.451130] [pid:4031,cpu10,(udev-worker)]usb 5-1: Please report results to Jes.Sorensen@gmail.com
[ 51.520812] [pid:4031,cpu11,(udev-worker)]usb 5-1: Vendor: Realtek
[ 51.520826] [pid:4031,cpu11,(udev-worker)]usb 5-1: Product: 802.11n WLAN Adapter
[ 51.520832] [pid:4031,cpu11,(udev-worker)]usb 5-1: rtl8723bu_parse_efuse: dumping efuse (0x200
bytes):
[ 51.520838] [pid:4031,cpu11,(udev-worker)]usb 5-1: 00: 29 81 03 7c 01 08 21 00
.....
[ 51.521045] [pid:4031,cpu11,(udev-worker)]usb 5-1: 1f8: ff ff ff ff ff ff ff
[ 51.521049] [pid:4031,cpu11,(udev-worker)]usb 5-1: RTL8723BU rev E (SMIC) 1T1R, TX queues 3,
WiFi=1, BT=1, GPS=0, HI PA=0
[ 51.521056] [pid:4031,cpu11,(udev-worker)]usb 5-1: RTL8723BU MAC: 00:13:ef:f4:58:ae
[ 51.521061] [pid:4031,cpu11,(udev-worker)]usb 5-1: rtl8xxxu: Loading firmware
rtlwifi/rtl8723bu_nic.bin
[ 51.521726] [pid:4031,cpu11,(udev-worker)]usb 5-1: Firmware revision 35.0 (signature 0x5301)
[ 51.865584] [pid:471,cpu0,kworker/u25:1]Bluetooth: hci0: RTL: fw version 0x0e2f9f73
[ 51.981137] [pid:2173,cpu11,bluetoothd]Bluetooth: MGMT ver 1.22
[ 51.992129] [pid:4045,cpu0,modprobe]NET: Registered PF_ALG protocol family
[ 52.413788] [2025:09:26 07:45:25][pid:4031,cpu11,(udev-worker)]usbcore: registered new interface
driver rtl8xxxu
[ 52.413807] [pid:4031,cpu11,(udev-worker)]do_boottime_initcall, initcall
rtl8xxxu_module_init+0x0/0x1000 [rtl8xxxu] returned 0 after 962778 usecs
```

4) Then use the **sudo ifconfig** command to see that an additional network device is added. For the WiFi connection and test methods, please refer to the [WiFi connection test](#) section, which will not be repeated here.

```
orangeypi@orangeypi:~$ sudo ifconfig
.....
wlan0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 00:13:ef:f4:58:ae txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```




5) Then use the **hciconfig** command to see the USB Bluetooth device.

```
orange_pi@orange_pi:~$ sudo apt update && sudo apt install bluez
orange_pi@orange_pi:~$ hciconfig
hci0:  Type: Primary  Bus: USB
        BD Address: 00:13:EF:F4:58:AE  ACL MTU: 820:8  SCO MTU: 255:16
        DOWN
        RX bytes:1252 acl:0 sco:0 events:125 errors:0
        TX bytes:23307 acl:0 sco:0 commands:125 errors:0
```

6) For the Bluetooth testing method, please refer to the instructions in the [Bluetooth Usage section](#), which will not be repeated here.

3. 12. 6. 2. RTL8811 Test

1) First, insert the RTL8811 wireless network card module into the USB port of the development board.

2) Then the Linux system will automatically load the kernel modules related to RTL8811 WIFI. You can see through the **lsmod** command that the following kernel modules have been automatically loaded.

```
orange_pi@orange_pi:~$ lsmod
Module                Size  Used by
8821cu                 1839104  0
```

3) Use the **dmesg** command to view the loading information of the RTL8811 module.

```
orange_pi@orange_pi:~$ dmesg
[ 34.201228] [2025:09:26 07:53:32][pid:142,cpu10,kworker/10:1]usb 5-1: new high-speed USB device number 2
using xhci-hcd
[ 34.376278] [pid:3812,cpu0,(udev-worker)]Driver 'rtl8821cu' needs updating - please use bus_type methods
[ 34.553675] [pid:3812,cpu0,(udev-worker)]usbcore: registered new interface driver rtl8821cu
[ 34.553690] [pid:3812,cpu0,(udev-worker)]do_boottime_initcall, initcall rtw_drv_entry+0x0/0x1000 [8821cu]
returned 0 after 177427 usecs
[ 35.416855] [2025:09:26 07:53:33][pid:73,cpu11,kworker/11:1]IPv6: ADDRCONF(NETDEV_CHANGE): wlan0:
link becomes ready
```



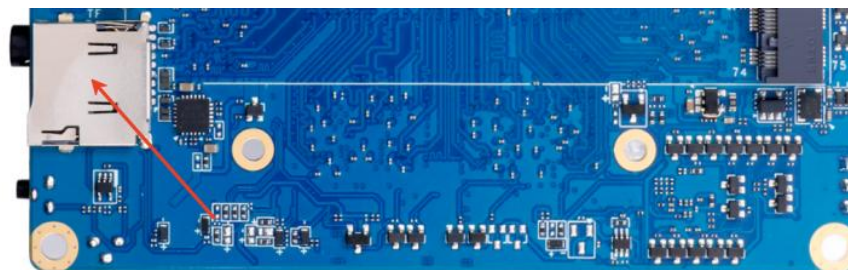
4) Then use the **sudo ifconfig** command to see that an additional network device is added. For the WiFi connection and test methods, please refer to the [WiFi connection test](#) section, which will not be repeated here.

```
orangepi@orangepi:~$ sudo ifconfig
.....
wlan0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 00:13:ef:f4:58:ae  txqueuelen 1000  (Ethernet)
        RX packets 0   bytes 0 (0.0 B)
        RX errors 0   dropped 0   overruns 0   frame 0
        TX packets 0   bytes 0 (0.0 B)
        TX errors 0   dropped 0 overruns 0   carrier 0   collisions 0
```

3. 12. 7. How to use the TF card slot

The TF card slot is transferred through the USB interface, so when the TF card is inserted into the TF card slot, the TF card is equivalent to a USB storage device.

The location of the TF card slot is as follows:



When a TF card is inserted into the TF card slot, if it is recognized normally, you can use the **fdisk -l** command to see a storage device named **sdx**, such as **sda**.

```
orangepi@orangepi:~$ sudo fdisk -l
.....
Disk /dev/sda: 29.73 GiB, 31927042048 bytes, 62357504 sectors
Disk model: MassStorageClass
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 437A156C-902D-4A19-ABB9-0A41287A51C7

Device      Start      End  Sectors  Size Type
```

```
/dev/sda1 2048 62355455 62353408 29.7G Linux filesystem
```

The TF card slot of the development board supports hot plugging. For example, if you insert a TF card after the system starts, it can be recognized normally.

3. 13. Audio Test

3. 13. 1. Test Method for Audio Playback via Type-C Interface

1) The development board has two Type-C interfaces, both of which support audio playback. The one closest to the board is Type-C interface 1, and the one closest to the network port is Type-C interface 2.



2) Before testing audio, please use a Type-C to HDMI cable to connect the development board to an HDMI monitor with audio, and make sure the HDMI monitor displays the desktop normally.



3) You can use the **aplay -l** command to see all the audio devices currently supported by the Linux system, as shown below:

a. Linux6.1:

```
orange@orange:~$ sudo aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: cixsky1 [cix,sky1], device 0: hda hda-audio-codec-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```



```

card 0: cixsky1 [cix,sky1], device 1: dptx0_audio i2s-hifi-1 [dptx0_audio i2s-hifi-1]
Subdevices: 1/1
Subdevice #0: subdevice #0
card 0: cixsky1 [cix,sky1], device 2: dptx1_audio i2s-hifi-2 [dptx1_audio i2s-hifi-2]
Subdevices: 1/1
Subdevice #0: subdevice #0
card 0: cixsky1 [cix,sky1], device 3: dptx3_audio i2s-hifi-3 [dptx3_audio i2s-hifi-3]
Subdevices: 1/1
Subdevice #0: subdevice #0
card 0: cixsky1 [cix,sky1], device 4: dptx4_audio i2s-hifi-4 [dptx4_audio i2s-hifi-4]
Subdevices: 1/1
Subdevice #0: subdevice #0

```

b. Linux6.6 ACPI:

```

orange@orange:~$ sudo aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: HDA [CIX SKY1 ORAPI 6P HDA], device 0: ALC269VC Analog [ALC269VC
Analog]
Subdevices: 1/1
Subdevice #0: subdevice #0
card 1: cixsky1 [cix,sky1], device 0: soc:i2s5-dp0 i2s-hifi-0 [soc:i2s5-dp0 i2s-hifi-0]
Subdevices: 1/1
Subdevice #0: subdevice #0
card 1: cixsky1 [cix,sky1], device 1: soc:i2s6-dp1 i2s-hifi-1 [soc:i2s6-dp1 i2s-hifi-1]
Subdevices: 1/1
Subdevice #0: subdevice #0
card 1: cixsky1 [cix,sky1], device 2: soc:i2s8-dp3 i2s-hifi-2 [soc:i2s8-dp3 i2s-hifi-2]
Subdevices: 1/1
Subdevice #0: subdevice #0
card 1: cixsky1 [cix,sky1], device 3: soc:i2s9-dp4 i2s-hifi-3 [soc:i2s9-dp4 i2s-hifi-3]
Subdevices: 1/1
Subdevice #0: subdevice #0

```

4) The Type-C1 interface in Linux6.1 corresponds to the dptx0_audio device, and the Type-C1 interface in Linux6.6 corresponds to the i2s5-dp0 audio device. The command to play audio is shown below. If the monitor can hear the sound, it indicates that the test is normal.



a. Linux6.1:

```
orange@orange:~$ sudo aplay -D hw:0,1 /usr/share/sounds/alsa/audio.wav
```

b. Linux6.6 ACPI:

```
orange@orange:~$ sudo aplay -D hw:1,0 /usr/share/sounds/alsa/audio.wav
```

5) The Type-C2 interface in Linux6.1 corresponds to the dptx1_audio audio device, and the Type-C1 interface in Linux6.6 corresponds to the i2s6-dp1 audio device. The command to play audio is shown below. If the monitor can hear the sound, it indicates that the test is normal.

a. Linux6.1:

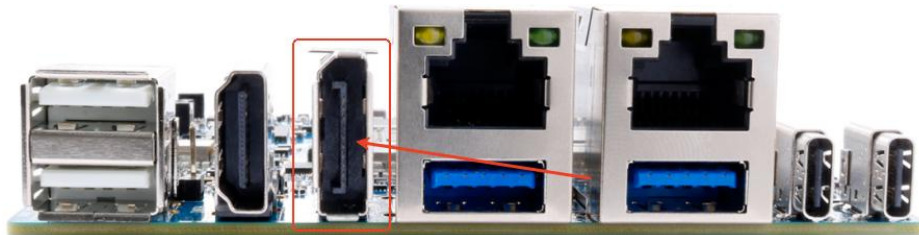
```
orange@orange:~$ sudo aplay -D hw:0,2 /usr/share/sounds/alsa/audio.wav
```

b. Linux6.6:

```
orange@orange:~$ sudo aplay -D hw:1,1 /usr/share/sounds/alsa/audio.wav
```

3. 13. 2. Test method for DP interface audio playback

1) The development board has a DP interface, the location of which is shown in the figure below:



2) Then use a DP to HDMI cable to connect the development board and the HDMI monitor, and make sure the HDMI monitor displays the desktop normally.



3) You can use the **aplay -l** command to see all the audio devices currently supported by the Linux system, as shown below:

a. Linux6.1:

```
orange@orange:~$ sudo aplay -l
```



```
**** List of PLAYBACK Hardware Devices ****
card 0: cixsky1 [cix,sky1], device 0: hda hda-audio-codec-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 0: cixsky1 [cix,sky1], device 1: dptx0_audio i2s-hifi-1 [dptx0_audio i2s-hifi-1]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 0: cixsky1 [cix,sky1], device 2: dptx1_audio i2s-hifi-2 [dptx1_audio i2s-hifi-2]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 0: cixsky1 [cix,sky1], device 3: dptx3_audio i2s-hifi-3 [dptx3_audio i2s-hifi-3]
Subdevices: 1/1
Subdevice #0: subdevice #0
card 0: cixsky1 [cix,sky1], device 4: dptx4_audio i2s-hifi-4 [dptx4_audio i2s-hifi-4]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

b. Linux6.6 ACPI:

```
orangepi@orangepi:~$ sudo aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: HDA [CIX SKY1 ORAPI 6P HDA], device 0: ALC269VC Analog [ALC269VC
Analog]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: cixsky1 [cix,sky1], device 0: soc:i2s5-dp0 i2s-hifi-0 [soc:i2s5-dp0 i2s-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: cixsky1 [cix,sky1], device 1: soc:i2s6-dp1 i2s-hifi-1 [soc:i2s6-dp1 i2s-hifi-1]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: cixsky1 [cix,sky1], device 2: soc:i2s8-dp3 i2s-hifi-2 [soc:i2s8-dp3 i2s-hifi-2]
Subdevices: 1/1
Subdevice #0: subdevice #0
card 1: cixsky1 [cix,sky1], device 3: soc:i2s9-dp4 i2s-hifi-3 [soc:i2s9-dp4 i2s-hifi-3]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```



4) In Linux 6.1, the DP interface corresponds to the dptx3_audio audio device, and in Linux 6.6, the DP interface corresponds to the i2s8-dp3 audio device. The command to play audio is shown below. If the monitor can hear the sound, the test is normal.

a. Linux6.1:

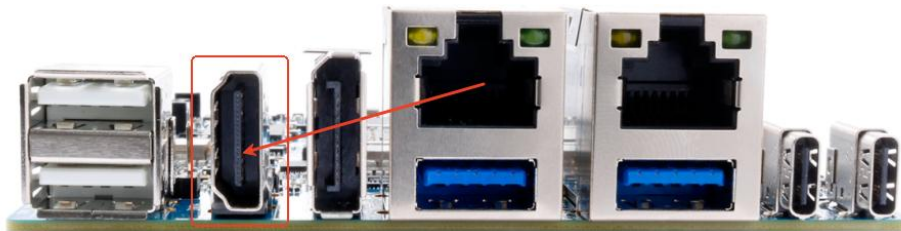
```
orange@orange:~$ sudo aplay -D hw:0,3 /usr/share/sounds/alsa/audio.wav
Playing WAVE 'audio.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
```

b. Linux6.6 ACPI:

```
orange@orange:~$ sudo aplay -D hw:1,2 /usr/share/sounds/alsa/audio.wav
Playing WAVE 'audio.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
```

3. 13. 3. HDIM Interface Audio Playback Testing Methods

1) The development board has an HDMI port, located as shown in the image below.:



2) Connect the development board and the HDMI monitor using an HDMI to HDMI cable, and ensure that the HDMI monitor displays the desktop normally.



3) The **aplay -l** command can be used to view all the audio devices currently supported by the Linux system, as shown below:

a. Linux6.1:

```
orange@orange:~$ sudo aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: cixsky1 [cix,sky1], device 0: hda hda-audio-codec-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 0: cixsky1 [cix,sky1], device 1: dptx0_audio i2s-hifi-1 [dptx0_audio i2s-hifi-1]
```




```

Subdevices: 1/1
Subdevice #0: subdevice #0
card 0: cixsky1 [cix,sky1], device 2: dptx1_audio i2s-hifi-2 [dptx1_audio i2s-hifi-2]
Subdevices: 1/1
Subdevice #0: subdevice #0
card 0: cixsky1 [cix,sky1], device 3: dptx3_audio i2s-hifi-3 [dptx3_audio i2s-hifi-3]
Subdevices: 1/1
Subdevice #0: subdevice #0
card 0: cixsky1 [cix,sky1], device 4: dptx4_audio i2s-hifi-4 [dptx4_audio i2s-hifi-4]
Subdevices: 1/1
Subdevice #0: subdevice #0

```

b. Linux6.6 ACPI:

```

orange@orange:~$ sudo aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: HDA [CIX SKY1 ORAPI 6P HDA], device 0: ALC269VC Analog [ALC269VC
Analog]
Subdevices: 1/1
Subdevice #0: subdevice #0
card 1: cixsky1 [cix,sky1], device 0: soc:i2s5-dp0 i2s-hifi-0 [soc:i2s5-dp0 i2s-hifi-0]
Subdevices: 1/1
Subdevice #0: subdevice #0
card 1: cixsky1 [cix,sky1], device 1: soc:i2s6-dp1 i2s-hifi-1 [soc:i2s6-dp1 i2s-hifi-1]
Subdevices: 1/1
Subdevice #0: subdevice #0
card 1: cixsky1 [cix,sky1], device 2: soc:i2s8-dp3 i2s-hifi-2 [soc:i2s8-dp3 i2s-hifi-2]
Subdevices: 1/1
Subdevice #0: subdevice #0
card 1: cixsky1 [cix,sky1], device 3: soc:i2s9-dp4 i2s-hifi-3 [soc:i2s9-dp4 i2s-hifi-3]
Subdevices: 1/1
Subdevice #0: subdevice #0

```

4) In Linux 6.1, the HDMI interface corresponds to the dptx4_audio audio device, and in Linux 6.6, the HDMI interface corresponds to the i2s9-dp4 audio device. The command to play audio is shown below. If the monitor can hear sound, the test is normal.

a. Linux6.1:



```
orange@orange:~$ sudo aplay -D hw:0,4 /usr/share/sounds/alsa/audio.wav
Playing WAVE 'audio.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
```

b. Linux6.6 ACPI:

```
orange@orange:~$ sudo aplay -D hw:1,3 /usr/share/sounds/alsa/audio.wav
Playing WAVE 'audio.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
```

3. 13. 4. Test Method for Audio Playback via Headphone Interface

a. Linux6.1:

```
orange@orange:~$ sudo aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: cixsky1 [cix,sky1], device 0: hda hda-audio-codec-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 0: cixsky1 [cix,sky1], device 1: dptx0_audio i2s-hifi-1 [dptx0_audio i2s-hifi-1]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 0: cixsky1 [cix,sky1], device 2: dptx1_audio i2s-hifi-2 [dptx1_audio i2s-hifi-2]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 0: cixsky1 [cix,sky1], device 3: dptx3_audio i2s-hifi-3 [dptx3_audio i2s-hifi-3]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 0: cixsky1 [cix,sky1], device 4: dptx4_audio i2s-hifi-4 [dptx4_audio i2s-hifi-4]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

b. Linux6.6 ACPI:

```
orange@orange:~$ sudo aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: HDA [CIX SKY1 ORAPI 6P HDA], device 0: ALC269VC Analog [ALC269VC Analog]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: cixsky1 [cix,sky1], device 0: soc:i2s5-dp0 i2s-hifi-0 [soc:i2s5-dp0 i2s-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: cixsky1 [cix,sky1], device 1: soc:i2s6-dp1 i2s-hifi-1 [soc:i2s6-dp1 i2s-hifi-1]
  Subdevices: 1/1
```



```

Subdevice #0: subdevice #0
card 1: cixskyl [cix,skyl], device 2: soc:i2s8-dp3 i2s-hifi-2 [soc:i2s8-dp3 i2s-hifi-2]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: cixskyl [cix,skyl], device 3: soc:i2s9-dp4 i2s-hifi-3 [soc:i2s9-dp4 i2s-hifi-3]
  Subdevices: 1/1
  Subdevice #0: subdevice #0

```

2) The following settings need to be configured before testing headphone audio playback.

a. Linux6.1:

```

orange@orange:~$ sudo amixer cset iface=MIXER,name='ClassD AMP Mute' 1
orange@orange:~$ sudo amixer cset iface=MIXER,name='HP Out Mute' 0
orange@orange:~$ sudo amixer cset iface=MIXER,name='Headset Switch' 1

```

b. Linux6.6 ACPI:

```

orange@orange:~$ sudo amixer -c 0 cset iface=MIXER,name='Auto-Mute Mode' 0
orange@orange:~$ sudo amixer -c 0 cset iface=MIXER,name='Master Playback Switch' on
orange@orange:~$ sudo amixer -c 0 cset iface=MIXER,name='Master Playback Volume' 100
orange@orange:~$ sudo amixer -c 0 cset iface=MIXER,name='Headphone Playback Switch' on,on
orange@orange:~$ sudo amixer -c 0 cset iface=MIXER,name='Headphone Playback Volume' 100,100
orange@orange:~$ sudo amixer -c 0 cset iface=MIXER,name='Speaker Playback Switch' off,off

```

3) The command to set the volume is shown below. The volume parameter range is from 0 to 100. The following command will set the volume to 50%.

a. Linux6.1:

```

orange@orange:~$ sudo amixer set Master 50

```

b. Linux6.6 ACPI:

```

orange@orange:~$ sudo amixer -c 0 cset iface=MIXER,name='Master Playback Volume' 50

```

4) The command to play audio through the headphone jack is shown below. If you can hear sound through the headphones, the test is normal.

```

orange@orange:~$ sudo aplay -D hw:0,0 /usr/share/sounds/alsa/audio.wav

```

3. 13. 5. Test method for audio playback of speaker interface

1) Prepare two speakers similar to the ones shown below. The speaker socket on the development board has a **2-pin 1.0mm** pitch.



2) The locations of the two speaker ports on the development board are shown below. Once the speakers are prepared, plug them into the speaker ports on the development board.



3) Use the **aplay -l** command to see all the audio devices currently supported by the Linux system, as shown below:

a. Linux6.1:

```

orangepi@orangepi:~$ sudo aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: cixsky1 [cix,sky1], device 0: hda hda-audio-codec-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 0: cixsky1 [cix,sky1], device 1: dptx0_audio i2s-hifi-1 [dptx0_audio i2s-hifi-1]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 0: cixsky1 [cix,sky1], device 2: dptx1_audio i2s-hifi-2 [dptx1_audio i2s-hifi-2]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 0: cixsky1 [cix,sky1], device 3: dptx3_audio i2s-hifi-3 [dptx3_audio i2s-hifi-3]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 0: cixsky1 [cix,sky1], device 4: dptx4_audio i2s-hifi-4 [dptx4_audio i2s-hifi-4]
  Subdevices: 1/1
  
```



Subdevice #0: subdevice #0

b. Linux6.6 ACPI:

```
orange@orange:~$ sudo aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: HDA [CIX SKY1 ORAPI 6P HDA], device 0: ALC269VC Analog
[ALC269VC Analog]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: cixsky1 [cix,sky1], device 0: soc:i2s5-dp0 i2s-hifi-0 [soc:i2s5-dp0 i2s-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: cixsky1 [cix,sky1], device 1: soc:i2s6-dp1 i2s-hifi-1 [soc:i2s6-dp1 i2s-hifi-1]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: cixsky1 [cix,sky1], device 2: soc:i2s8-dp3 i2s-hifi-2 [soc:i2s8-dp3 i2s-hifi-2]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: cixsky1 [cix,sky1], device 3: soc:i2s9-dp4 i2s-hifi-3 [soc:i2s9-dp4 i2s-hifi-3]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

4) The following settings need to be performed before testing the speaker audio playback.

a. Linux6.1:

```
orange@orange:~$ sudo amixer cset iface=MIXER,name='ClassD AMP Mute' 0
orange@orange:~$ sudo amixer cset iface=MIXER,name='HP Out Mute' 1
```

b. Linux6.6 ACPI:

```
orange@orange:~$ sudo amixer -c 0 cset iface=MIXER,name='Auto-Mute Mode' 0
orange@orange:~$ sudo amixer -c 0 cset iface=MIXER,name='Master Playback Switch' on
orange@orange:~$ sudo amixer -c 0 cset iface=MIXER,name='Master Playback Volume' 100
orange@orange:~$ sudo amixer -c 0 cset iface=MIXER,name='Speaker Playback Switch' on,on
orange@orange:~$ sudo amixer -c 0 cset iface=MIXER,name='Speaker Playback Volume' 100,100
orange@orange:~$ sudo amixer -c 0 cset iface=MIXER,name='Headphone Playback Switch' off,off
```

5) The command to set the volume is as follows. The volume parameter value range is 0



to 100. The following command will set the volume to 100%.

- a. Linux6.1:

```
orange@orange:~$ sudo amixer set Master 100
```

- b. Linux6.6 ACPI:

```
orange@orange:~$ sudo amixer -c 0 cset iface=MIXER,name='Master Playback Volume' 100
```

6) The command to play audio through the speaker interface is as follows. If the speaker can hear the sound, the test is normal.

```
orange@orange:~$ sudo aplay -D hw:0,0 /usr/share/sounds/alsa/audio.wav
```

3. 13. 6. Test Method for Audio Recording via Headphone Interface

- 1) Plug the headphones with recording function into the headphone jack of the development board.



- 2) Then set the audio channel for headphone recording.

- a. Linux6.1:

```
orange@orange:~$ sudo amixer cset iface=MIXER,name='Headphone Mic Switch' 1
orange@orange:~$ sudo amixer set 'HP Mic' 2
```

- b. Linux6.6 ACPI:

```
orange@orange:~$ sudo amixer -c 0 cset iface=MIXER,name='Capture Switch' on,on
orange@orange:~$ sudo amixer -c 0 cset iface=MIXER,name='Capture Volume' 39,39
orange@orange:~$ sudo amixer -c 0 cset iface=MIXER,name='Mic Boost Volume' 3,3
orange@orange:~$ sudo amixer -c 0 cset iface=MIXER,name='Internal Mic Boost Volume' 0,0
```

- 3) The command for recording with headphones is as follows:

```
orange@orange:~$ sudo arecord -Dhw:0,0 -c 2 -r 48000 -d 5 -f S16_LE arecord.wav
Playing WAVE 'audio.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
```

- 4) After the recording is completed, a recording file named arecord.wav will be generated in the current directory.



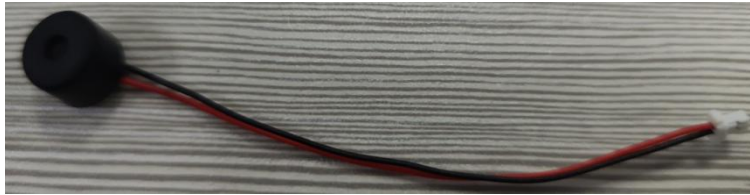
```
orangepi@orangepi:~$ ls arecord.wav
arecord.wav
```

5) Then you can play the recorded arecord.wav audio to headphones or HDMI display.

If you can hear the sound, it means the test is successful.

3. 13. 7. Test Method for MIC Interface Audio Recording

1) Prepare a MIC similar to the one shown below. The MIC socket on the development board has a **2-pin 1.0mm** pitch.



2) The location of the MIC interface on the development board is shown below. After preparing the MIC, please insert it into the MIC interface on the development board.



3) Then set the audio channel for MIC recording.

a. Linux6.1:

```
orangepi@orangepi:~$ sudo amixer cset iface=MIXER,name='Mic2 Switch' 1
orangepi@orangepi:~$ sudo amixer set Mic2 3
```

b. Linux6.6 ACPI:

```
orangepi@orangepi:~$ sudo amixer -c 0 cset iface=MIXER,name='Capture Switch' on,on
orangepi@orangepi:~$ sudo amixer -c 0 cset iface=MIXER,name='Capture Volume' 39,39
orangepi@orangepi:~$ sudo amixer -c 0 cset iface=MIXER,name='Internal Mic Boost Volume' 3,3
orangepi@orangepi:~$ sudo amixer -c 0 cset iface=MIXER,name='Mic Boost Volume' 0,0
```

4) The MIC recording command is as follows:

```
orangepi@orangepi:~$ sudo arecord -Dhw:0,0 -c 2 -r 48000 -d 5 -f S16_LE arecord.wav
Playing WAVE 'audio.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
```




5) After the recording is completed, a recording file named arecord.wav will be generated in the current directory.

```
orangepi@orangepi:~$ ls arecord.wav
arecord.wav
```

6) Then you can play the recorded arecord.wav audio to headphones or HDMI display. If you can hear the sound, it means the test is successful.

3. 14. Temperature sensor

1) There are 13 temperature sensors in the SoC. The command to view the system temperature sensors is:

a. Linux6.1:

```
orangepi@orangepi:~$ sensors
scmi_sensors-virtual-0
Adapter: Virtual device
VPU:          +41.0°C
GPU_btm:      +41.0°C
GPU_top:      +42.0°C
SOC_BRC:      +44.0°C
DDR_btm:      +45.0°C
DDR_top:      +45.0°C
CI:           +43.0°C
NPU:          +43.0°C
CPU_M1:       +42.0°C
CPU_B1:       +43.0°C
CPU_M0:       +44.0°C
CPU_B0:       +44.0°C
SOC_TRC:      +44.0°C
GPU_AVE:      +42.0°C
PCB_HOT:      +41.0°C
PCB_AMB:      +36.0°C
PCB_USER:     +0.0°C

battery-isa-0000
```



Adapter: ISA adapter

in0: 0.00 V

curr1: 0.00 A

b. Linux6.6 ACPI:

```
orangepi@orangepi:~$ sensors
```

```
acpitz-acpi-0
```

Adapter: ACPI interface

temp1: +41.0°C

temp2: +41.0°C

temp3: +43.0°C

temp4: +44.0°C

temp5: +48.0°C

temp6: +43.0°C

temp7: +43.0°C

temp8: +47.0°C

temp9: +44.0°C

temp10: +45.0°C

temp11: +43.0°C

temp12: +44.0°C

temp13: +42.0°C

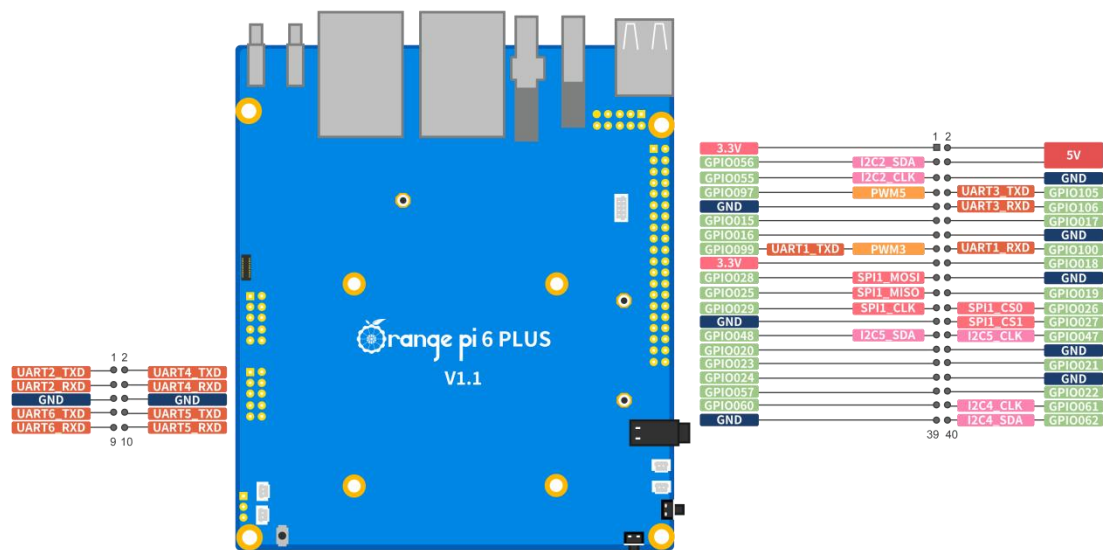
2) The command to check the current temperature of the NVMe SSD is as follows. **Note that nvme0 in the command needs to be replaced with the actual node name of the SSD.**

```
orangepi@orangepi:~$ sudo smartctl -a /dev/nvme0 | grep "Temperature:"
```

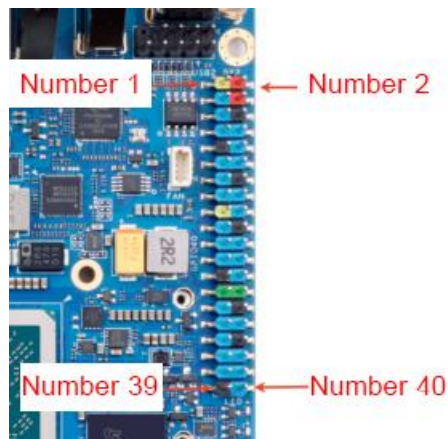
Temperature: **40 Celsius**



3. 15. 40 Pin Extension Interface Pin Usage Instructions



1) Please refer to the figure below for the order of the 40-pin expansion interface pins on the development board.:



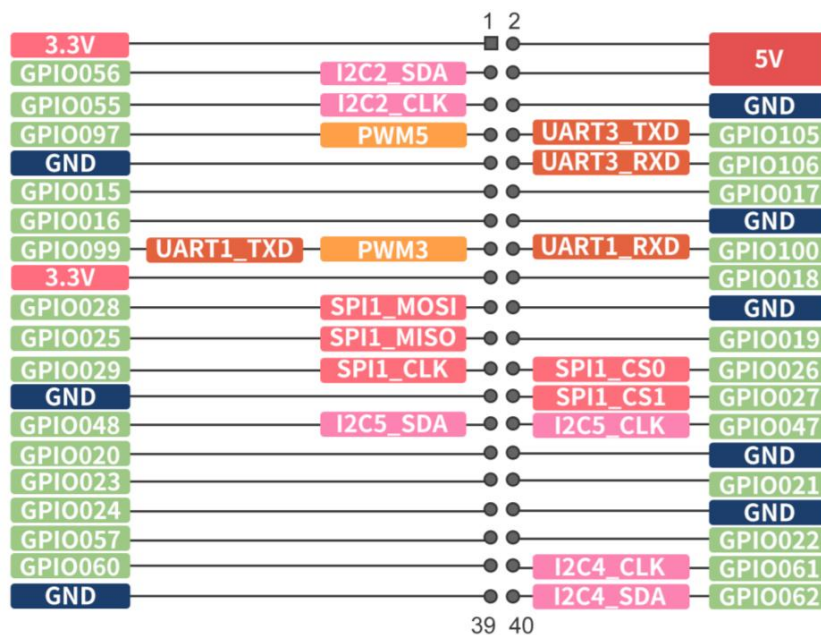
2) The functions of the 40-pin interface pins of the development board are as follows.:

a. Below is the complete pin diagram of the 40-pin interface:



复用功能	复用功能	GPIO	DT GPIO序号	ACPI GPIO序号	引脚序号	引脚序号	ACPI GPIO序号	DT GPIO序号	GPIO	复用功能
		3.3V			1	2			5V	
	I2C2_SDA(0x04030000)	GPIO056	65	13	3	4			5V	
	I2C2_CLK	GPIO055	64	12	5	6			GND	
	PWM5(0x04115000)	GPIO097	106	54	7	8	62	114	GPIO105	UART3_TXD(0x040e0000)
		GND			9	10	63	115	GPIO106	UART3_RXD
		GPIO0015	32	155	11	12	157	34	GPIO1017	
		GPIO0016	33	156	13	14			GND	
UART1_TXD(0x040c0000)	PWM3(0x04113000)	GPIO099	108	56	15	16	57	109	GPIO100	UART1_RXD
		3.3V			17	18	158	35	GPIO1018	
	SPI1_MOSI	GPIO028	17	130	19	20			GND	
	SPI1_MISO	GPIO025	14	127	21	22	159	36	GPIO1019	
	SPI1_CLK	GPIO029	18	131	23	24	128	15	GPIO0026	SPI1_CS0
		GND			25	26	129	16	GPIO0027	SPI1_CS1
	I2C5_SDA(0x04060000)	GPIO048	57	5	27	28	4	56	GPIO047	I2C5_CLK
		GPIO020	37	160	29	30			GND	
		GPIO023	40	163	31	32	161	38	GPIO021	
		GPIO024	41	164	33	34			GND	
		GPIO057	66	14	35	36	162	39	GPIO022	
		GPIO060	69	17	37	38	18	70	GPIO061	I2C4_CLK
		GND			39	40	19	71	GPIO062	I2C4_SDA(0x04050000)

b. The following color picture contains part of the content in the above table:



c. The table below is the content of the left half of the complete table above, which is clearer.



复用功能	复用功能	GPIO	DT GPIO序号	ACPI GPIO序号	引脚序号
		3.3V			1
	I2C2_SDA(0x04030000)	GPIO056	65	13	3
	I2C2_CLK	GPIO055	64	12	5
	PWM5(0x04115000)	GPIO097	106	54	7
		GND			9
		GPIO015	32	155	11
		GPIO016	33	156	13
UART1_TXD(0x040c0000)	PWM3(0x04113000)	GPIO099	108	56	15
		3.3V			17
	SPI1_MOSI	GPIO028	17	130	19
	SPI1_MISO	GPIO025	14	127	21
	SPI1_CLK	GPIO029	18	131	23
		GND			25
	I2C5_SDA(0x04060000)	GPIO048	57	5	27
		GPIO020	37	160	29
		GPIO023	40	163	31
		GPIO024	41	164	33
		GPIO057	66	14	35
		GPIO060	69	17	37
		GND			39

- d. The table below is the content of the right half of the complete table above, which is clearer.

引脚序号	ACPI GPIO序号	DT GPIO序号	GPIO	复用功能
2			5V	
4			5V	
6			GND	
8	62	114	GPIO105	UART3_TXD(0x040e0000)
10	63	115	GPIO106	UART3_RXD
12	157	34	GPIO017	
14			GND	
16	57	109	GPIO100	UART1_RXD
18	158	35	GPIO018	
20			GND	
22	159	36	GPIO019	
24	128	15	GPIO026	SPI1_CS0
26	129	16	GPIO027	SPI1_CS1
28	4	56	GPIO047	I2C5_CLK
30			GND	
32	161	38	GPIO021	
34			GND	
36	162	39	GPIO022	
38	18	70	GPIO061	I2C4_CLK
40	19	71	GPIO062	I2C4_SDA(0x04050000)

The table above shows the base addresses of the corresponding controller registers for PWM, I2C, and UART. This is useful for checking which pin on the



40-pin header a node corresponds to in `/sys/class`.

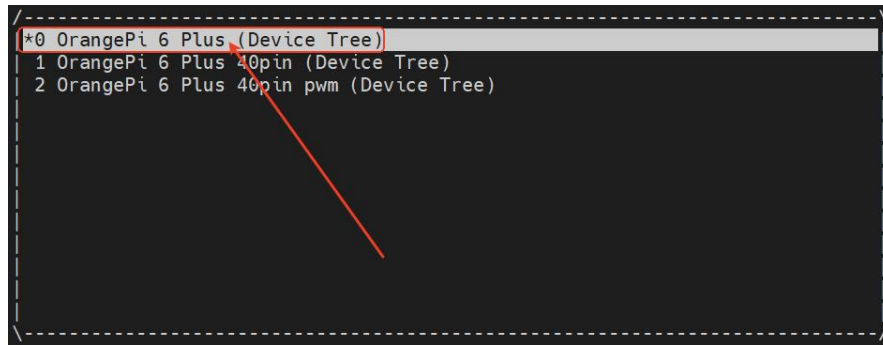
3) There are a total of **28** GPIO ports in the 40-pin interface, and the voltage of all GPIO ports is **3.3v**.

3. 16. 40 Pin Interface GPIO, I2C, UART, SPI and PWM Test Methods

3. 16. 1. 40 Pin GPIO Test Method

1) Before testing GPIO:

- a. Systems with a Linux 6.1 kernel need to ensure that device tree 0 is used during startup.



- b. For Linux 6.6 kernel systems using ACPI, ensure that the BIOS version displayed using the ``dmidecode`` command is **v1.0_for_opi6plus_linux**. If incorrect, please refer to the section on "[How to Flash the BIOS Firmware to the Development Board's SPI Flash](#)" to update the BIOS firmware.

```
orangepi@orangepi:~$ sudo dmidecode -s bios-version
v1.0_for_opi6plus_linux
```

2) The GPIO pin diagram is shown below. If the kernel uses device tree, please refer to the **DT GPIO Number** column for GPIO numbers. If the kernel uses ACPI, please refer to the **ACPI GPIO Number** column for GPIO numbers.



GPIO	DT GPIO序号	ACPI GPIO序号	引脚序号	引脚序号	ACPI GPIO序号	DT GPIO序号	GPIO
3.3V			1	2			5V
GPIO056	65	13	3	4			5V
GPIO055	64	12	5	6			GND
GPIO097	106	54	7	8	62	114	GPIO105
GND			9	10	63	115	GPIO106
GPIO015	32	155	11	12	157	34	GPIO017
GPIO016	33	156	13	14			GND
GPIO099	108	56	15	16	57	109	GPIO100
3.3V			17	18	158	35	GPIO018
GPIO028	17	130	19	20			GND
GPIO025	14	127	21	22	159	36	GPIO019
GPIO029	18	131	23	24	128	15	GPIO026
GND			25	26	129	16	GPIO027
GPIO048	57	5	27	28	4	56	GPIO047
GPIO020	37	160	29	30			GND
GPIO023	40	163	31	32	161	38	GPIO021
GPIO024	41	164	33	34			GND
GPIO057	66	14	35	36	162	39	GPIO022
GPIO060	69	17	37	38	18	70	GPIO061
GND			39	40	19	71	GPIO062

3) The `/sys/class/gpio` interface can be used to control the high and low levels of the GPIO output. For example, the test method for GPIO056 is shown below, and the same applies to other GPIOs.

a. Switch to the root user first.

```
orangePi@orangePi:~$ sudo -i
```

b. As can be seen from the GPIO specification table

a) In the device tree used by the Linux 6.1 kernel, the DT GPIO number corresponding to GPIO056 is 65.

GPIO	DT GPIO序号	ACPI GPIO序号	引脚序号
3.3V			1
GPIO056	65	13	3
GPIO055	64	12	5
GPIO097	106	54	7
GND			9
GPIO015	32	155	11

b) The ACPI used by the Linux 6.6 kernel, GPIO056, corresponds to ACPI GPIO number 13.

GPIO	DT GPIO序号	ACPI GPIO序号	引脚序号
3.3V			1
GPIO056	65	13	3
GPIO055	64	12	5
GPIO097	106	54	7
GND			9
GPIO015	32	155	11

c. Then use the following command to export GPIO056. The command is as follows:

a) Linux6.1:



```
root@orangepi:~# echo 65 > /sys/class/gpio/export
```

b) Linux6.6 ACPI:

```
root@orangepi:~# echo 13 > /sys/class/gpio/export
```

d. Then set GPIO056 to output mode.

a) Linux6.1:

```
root@orangepi:~# echo out > /sys/class/gpio/gpio65/direction
```

b) Linux6.6 ACPI:

```
root@orangepi:~# echo out > /sys/class/gpio/gpio13/direction
```

e. Then use the following command to set the high level of GPIO056.

a) Linux6.1:

```
root@orangepi:~# echo 1 > /sys/class/gpio/gpio65/value
```

b) Linux6.6 ACPI:

```
root@orangepi:~# echo 1 > /sys/class/gpio/gpio13/value
```

f. Then you can use the following command to set the GPIO056 to low level.

a) Linux6.1:

```
root@orangepi:~# echo 0 > /sys/class/gpio/gpio65/value
```

b) Linux6.6 ACPI:

```
root@orangepi:~# echo 0 > /sys/class/gpio/gpio13/value
```

4) The Linux system comes pre-installed with a **blink_gpio.sh** script, which can be used to quickly test the function of GPIO port outputting high and low levels.

a. The path to the **blink_gpio.sh** script is as follows:

```
orangepi@orangepi:~$ which blink_gpio.sh
/usr/bin/blink_gpio.sh
```

b. The usage of the **blink_gpio.sh** script is as follows: simply add the GPIO number corresponding to the pin to be tested after the **blink_gpio.sh** script. Multiple GPIO numbers can be added at the same time, and the script will cycle through turning on and off the GPIO ports corresponding to all the pins that need to be controlled.

```
orangepi@orangepi:~$ sudo blink_gpio.sh
Usage: /usr/bin/blink_gpio.sh gpio_number1 [gpio_number2 ... gpio_numberN]
```

c. For example, the testing method for GPIO056 is as follows:

a) Linux6.1:

```
orangepi@orangepi:~$ sudo blink_gpio.sh 65
```

b) Linux6.6 ACPI:



```
orangeypi@orangeypi:~$ sudo blink_gpio.sh 13
```

- c) After running the above command, use a multimeter to test the pin corresponding to GPIO056. You can see that the voltage value will cycle between 0 and 3.3V.
- d. The following command can cycle all GPIO values between 0 and 3.3V.
 - a) Linux6.1:

```
orangeypi@orangeypi:~$ sudo blink_gpio.sh 65 64 106 32 33 108 17 14 18 57 37 \
40 41 66 69 114 115 34 109 35 36 15 16 56 38 39 70 71
```

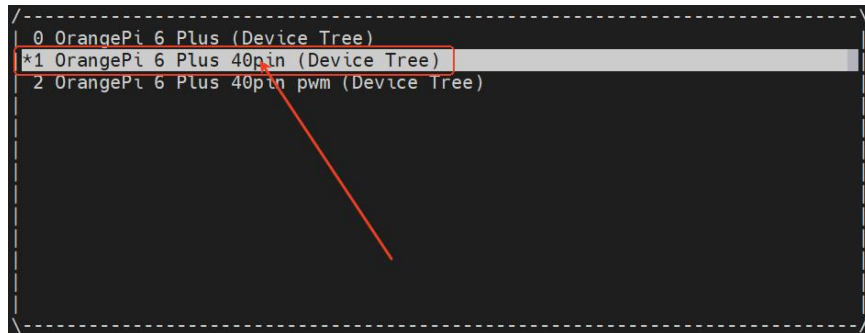
- b) Linux6.6 ACPI:

```
orangeypi@orangeypi:~$ sudo blink_gpio.sh 4 5 12 13 14 17 18 19 54 56 57 62 \
63 127 128 129 130 131 155 156 157 158 159 160 161 162 163 164
```

3. 16. 2. 40 Pin Interface SPI Test Method

- 1) Before testing SPI:

- a. Systems with the Linux 6.1 kernel need to ensure that device tree 1 is used during startup. If the default device tree 0 is being used, please reboot the system to switch to device tree 1 before continuing testing.



- b. For Linux 6.6 kernel systems using ACPI, ensure that the BIOS version displayed using the `dmidecode` command is **v1.0_for_opi6plus_linux_40pin**. If incorrect, please refer to the section on "[How to Flash BIOS Firmware to the Development Board's SPI Flash](#)" to update the BIOS firmware.

```
orangeypi@orangeypi:~$ sudo dmidecode -s bios-version
v1.0_for_opi6plus_linux_40pin
```

- 2) As can be seen from the table below, the available SPI controller in the development board's 40-pin is SPI1.



复用功能	复用功能	GPIO	DT GPIO序号	ACPI GPIO序号	引脚序号	引脚序号	ACPI GPIO序号	DT GPIO序号	GPIO	复用功能
		3.3V			1	2			5V	
	I2C2_SDA(0x04030000)	GPIO056	65	13	3	4			5V	
	I2C2_CLK	GPIO055	64	12	5	6			GND	
	PWM5(0x04115000)	GPIO097	106	54	7	8	62	114	GPIO105	UART3_TXD(0x040e0000)
		GND			9	10	63	115	GPIO106	UART3_RXD
		GPIO015	32	155	11	12	157	34	GPIO017	
		GPIO016	33	156	13	14			GND	
UART1_TXD(0x040c0000)	PWM3(0x04113000)	GPIO099	108	56	15	16	57	109	GPIO100	UART1_RXD
		3.3V			17	18	158	35	GPIO018	
	SPI1_MOSI	GPIO028	17	130	19	20			GND	
	SPI1_MISO	GPIO025	14	127	21	22	159	36	GPIO019	
	SPI1_CLK	GPIO029	18	131	23	24	128	15	GPIO026	SPI1_CS0
		GND			25	26	129	16	GPIO027	SPI1_CS1
	I2C5_SDA(0x04060000)	GPIO048	57	5	27	28	4	56	GPIO047	I2C5_CLK
		GPIO020	37	160	29	30			GND	
		GPIO023	40	163	31	32	161	38	GPIO021	
		GPIO024	41	164	33	34			GND	
		GPIO057	66	14	35	36	162	39	GPIO022	
		GPIO060	69	17	37	38	18	70	GPIO061	I2C4_CLK
		GND			39	40	19	71	GPIO062	I2C4_SDA(0x04050000)

3) Enter the system and check whether there is a device node **spidev0.x** in the Linux system. If it exists, it means that SPI has been set up and can be used directly.

```
orangepi@orangepi:~$ ls /dev/spidev*
/dev/spidev0.0 /dev/spidev0.1
```

4) Without shorting the MOSI and MISO pins of SPI1, the output of running `spidev_test` is shown below. You can see that the TX and RX data are inconsistent.

```
orangepi@orangepi:~$ sudo spidev_test -v -D /dev/spidev0.0

spi mode: 0x0

bits per word: 8

max speed: 500000 Hz (500 KHz)

TX | F1 F2 F3 F4 F5 F6 40 00 00 00 00 95 F7 F8 F9 FA FB FC FD 11 12 13 14 15 16 17 18 19 20 21 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 |.....|
```

5) Then short the MOSI and MISO pins of SPI1 and run `spidev_test` again. The output is as follows. You can see that the sent and received data are the same, indicating that the SPI loopback test is normal.

```
orangepi@orangepi:~$ sudo spidev_test -v -D /dev/spidev0.0

spi mode: 0x0

bits per word: 8

max speed: 500000 Hz (500 KHz)

TX | F1 F2 F3 F4 F5 F6 40 00 00 00 00 95 F7 F8 F9 FA FB FC FD 11 12 13 14 15 16 17 18 19 20 21 F0 0D |.....@..... !..|
RX | F1 F2 F3 F4 F5 F6 40 00 00 00 00 95 F7 F8 F9 FA FB FC FD 11 12 13 14 15 16 17 18 19 20 21 F0 0D |.....@..... !..|
```

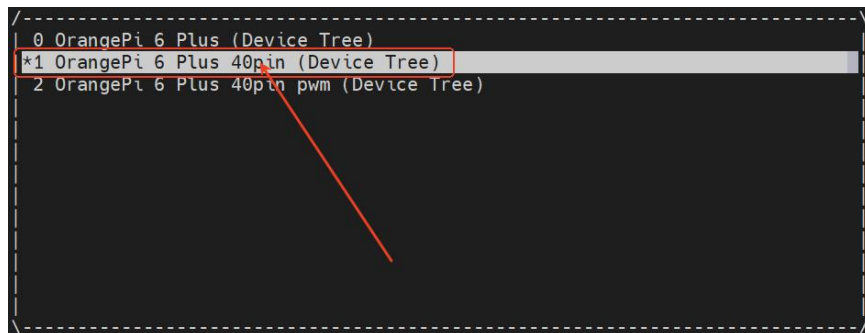
3. 16. 3. 40-pin I2C interface test method

1) Before testing I2C:

- Systems with the Linux 6.1 kernel need to ensure that device tree 1 is used



during startup. If the default device tree 0 is being used, please reboot the system to switch to device tree 1 before continuing testing.



- b. For Linux 6.6 kernel systems using ACPI, ensure that the BIOS version displayed using the `'dmidecode'` command is **v1.0_for_opi6plus_linux_40pin**. If incorrect, please refer to the section on "[How to Flash BIOS Firmware to the Development Board's SPI Flash](#)" to update the BIOS firmware.

```
orange@orange:~$ sudo dmidecode -s bios-version
v1.0_for_opi6plus_linux_40pin
```

2) As can be seen from the table below, the available I2C buses in the 40-pin of the development board are i2c2, i2c4, and i2c5, totaling 3 groups of I2C buses.

复用功能	复用功能	GPIO	DT GPIO序号	ACPI GPIO序号	引脚序号	引脚序号	ACPI GPIO序号	DT GPIO序号	GPIO	复用功能
		3.3V			1	2				
	I2C2_SDA(0x04030000)	GPIO056	65	13	3	4			5V	
	I2C2_CLK	GPIO055	64	12	5	6			5V	
	PWM5(0x04115000)	GPIO097	106	54	7	8	62	114	GND	
		GND			9	10	63	115	GPIO105	UART3_TXD(0x040e0000)
		GPIO015	32	155	11	12	157	34	GPIO106	UART3_RXD
		GPIO016	33	156	13	14			GPIO107	
		GPIO099	108	56	15	16	57	109	GND	
UART1_TXD(0x040c0000)	PWM3(0x04113000)	3.3V			17	18	158	35	GPIO100	UART1_RXD
					19	20			GPIO1018	
	SPI1_MOSI	GPIO028	17	130	21	22	159	36	GND	
	SPI1_MISO	GPIO025	14	127	23	24	128	15	GPIO019	
	SPI1_CLK	GPIO029	18	131	25	26	129	16	GPIO026	SPI1_CS0
		GND			27	28	4	56	GPIO027	SPI1_CS1
	I2C5_SDA(0x04060000)	GPIO048	57	5	29	30			GPIO047	I2C5_CLK
		GPIO020	37	160	31	32	161	38	GND	
		GPIO023	40	163	33	34			GPIO021	
		GPIO024	41	164	35	36	162	39	GND	
		GPIO057	66	14	37	38	18	70	GPIO022	
		GPIO060	69	17	39	40	19	71	GPIO061	I2C4_CLK
		GND							GPIO062	I2C4_SDA(0x04050000)

3) After entering the Linux system, first confirm the correspondence between the i2c controller and the device node name in /dev. The output of the `i2cdetect -l` command shows:

- 0x04030000 represents the I2C2 controller, corresponding to /dev/i2c-2
- 0x04050000 represents the I2C4 controller, corresponding to /dev/i2c-4
- 0x04060000 represents the I2C5 controller, corresponding to /dev/i2c-5

```
orange@orange:~$ sudo i2cdetect -l
i2c-0  i2c          Cadence I2C at 04010000          I2C adapter
i2c-1  i2c          Cadence I2C at 04020000          I2C adapter
```



i2c-2	i2c	Cadence I2C at 04030000	I2C adapter
i2c-3	i2c	Cadence I2C at 04040000	I2C adapter
i2c-4	i2c	Cadence I2C at 04050000	I2C adapter
i2c-5	i2c	Cadence I2C at 04060000	I2C adapter
.....			

4) Then connect an I2C device to the I2C pins corresponding to the 40-pin interface. The corresponding pins of the three I2C buses in the 40-pin interface are shown in the following table:

I2C bus	SDA correspond 40Pin	SCL correspond 40Pin
I2C2	Pin 3	Pin 5
I2C4	Pin 40	Pin 38
I2C5	Pin 27	Pin 28

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

```

orangePi@orangePi:~$ sudo i2cdetect -y -r 2      #i2c2 commands
orangePi@orangePi:~$ sudo i2cdetect -y -r 4      #i2c4commands
orangePi@orangePi:~$ sudo i2cdetect -y -r 5      #i2c5commands

```

```

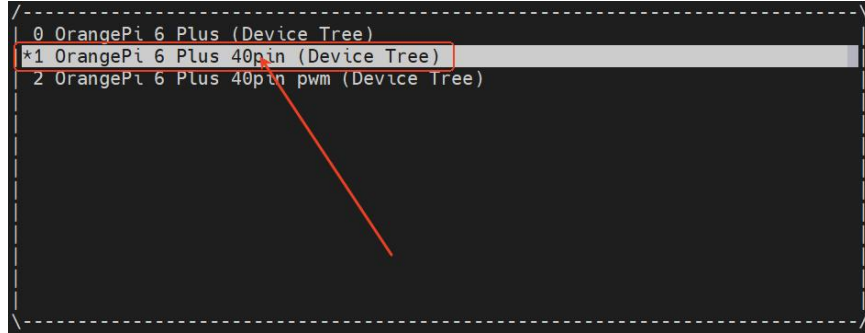
orangePi@orangePi6plus:~$ sudo i2cdetect -y -r 2
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  38  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
orangePi@orangePi6plus:~$

```

3. 16. 4. 40 Pin Interface UART Test Method

1) Before testing UART:

- Systems with the Linux 6.1 kernel need to ensure that device tree 1 is used during startup. If the default device tree 0 is being used, please reboot the system to switch to device tree 1 before continuing testing.



- b. For Linux 6.6 kernel systems using ACPI, ensure that the BIOS version displayed using the `dmidecode` command is `v1.0_for_opi6plus_linux_40pin`. If incorrect, please refer to the section on "[How to Flash BIOS Firmware to the Development Board's SPI Flash](#)" to update the BIOS firmware.

```
orangepi@orangepi:~$ sudo dmidecode -s bios-version
v1.0_for_opi6plus_linux_40pin
```

- 2) As can be seen from the table below, the available UARTs in the development board's 40-pin are UART1 and UART3, a total of two UART buses.

复用功能	复用功能	GPIO	DT GPIO序号	ACPI GPIO序号	引脚序号	引脚序号	ACPI GPIO序号	DT GPIO序号	GPIO	复用功能
	3.3V				1	2				5V
	I2C2_SDA(0x04030000)	GPIO056	65	13	3	4				5V
	I2C2_CLK	GPIO055	64	12	5	6				GND
	PWM5(0x04115000)	GPIO097	106	54	7	8	62	114	GPIO105	UART3_TXD(0x040e0000)
	GND				9	10	63	115	GPIO106	UART3_RXD
		GPIO1015	32	155	11	12	157	34	GPIO1017	
		GPIO1016	33	156	13	14			GND	
UART1_TXD(0x040c0000)	PWM3(0x04113000)	GPIO099	108	56	15	16	57	109	GPIO100	UART1_RXD
	3.3V				17	18	158	35	GPIO1018	
					19	20			GND	
	SPI1_MOSI	GPIO028	17	130	21	22	159	36	GPIO1019	
	SPI1_MISO	GPIO025	14	127	23	24	128	15	GPIO026	SPI1_CS0
	SPI1_CLK	GPIO029	18	131	25	26	129	16	GPIO027	SPI1_CS1
	GND				27	28	4	56	GPIO047	I2C5_CLK
	I2C5_SDA(0x04060000)	GPIO048	57	5	29	30			GND	
		GPIO020	37	160	31	32	161	38	GPIO021	
		GPIO023	40	163	33	34			GND	
		GPIO024	41	164	35	36	162	39	GPIO022	
		GPIO057	66	14	37	38	18	70	GPIO061	I2C4_CLK
		GPIO060	69	17	39	40	19	71	GPIO062	I2C4_SDA(0x04050000)
	GND									

- 3) After entering the Linux system:

- a. The output of `ls /sys/class/tty/ttyAMA* -l` on a Linux 6.1 kernel system can be used to determine the system's behavior.:
- 0x40c0000 correspond uart1, correspond `/dev/ttyAMA1`
 - 0x40e0000 correspond uart3, correspond `/dev/ttyAMA3`

```
orangepi@orangepi:~$ ls /dev/ttyAMA*
/dev/ttyAMA1 /dev/ttyAMA2 /dev/ttyAMA3
orangepi@orangepi:~$ ls /sys/class/tty/ttyAMA* -l
orangepi@orangepi6plus:~$ ls /sys/class/tty/ttyAMA* -lh
lrwxrwxrwx 1 root root 0 Aug 29 19:15 /sys/class/tty/ttyAMA1 -> ../../devices/platform/soc@0/40c0000.uart/tty/ttyAMA1
lrwxrwxrwx 1 root root 0 Aug 29 19:15 /sys/class/tty/ttyAMA2 -> ../../devices/platform/soc@0/40d0000.uart/tty/ttyAMA2
lrwxrwxrwx 1 root root 0 Aug 29 19:15 /sys/class/tty/ttyAMA3 -> ../../devices/platform/soc@0/40e0000.uart/tty/ttyAMA3
orangepi@orangepi6plus:~$
```

- b. The output of `sudo dmesg | grep "ttyAMA. at"` on a Linux 6.6 ACPI kernel



system indicates that...:

- a) 0x40c0000 correspond uart1, correspond **/dev/ttyAMA1**
- b) 0x40e0000 correspond uart3, correspond **/dev/ttyAMA3**

```
orangePi@orangePi:~$ ls /dev/ttyAMA*
```

```
/dev/ttyAMA0 /dev/ttyAMA1 /dev/ttyAMA2 /dev/ttyAMA3
```

```
orangePi@orangePi:~$ sudo dmesg | grep "ttyAMA. at"
```

```
orangePi@orangePi6plus:~$ sudo dmesg | grep "ttyAMA. at"
[sudo] password for orangePi:
[ 1.856768] [pid:1,cpu4,swapper/0]ARMH0011:00: ttyAMA0 at MMIO 0x40b0000 (irq = 21, base_baud = 0) is a SBSA
[ 1.918146] [pid:1,cpu1,swapper/0]ARMH0011:01: ttyAMA1 at MMIO 0x40c0000 (irq = 22, base_baud = 0) is a SBSA
[ 1.919184] [pid:1,cpu1,swapper/0]ARMH0011:02: ttyAMA2 at MMIO 0x40d0000 (irq = 23, base_baud = 0) is a SBSA
[ 1.920538] [pid:1,cpu6,swapper/0]ARMH0011:03: ttyAMA3 at MMIO 0x40e0000 (irq = 24, base_baud = 0) is a SBSA
orangePi@orangePi6plus:~$
```

4) Then start testing the UART interface. First, use a Dupont line to short-circuit the rx and tx pins of the UART interface to be tested. The corresponding pins of the two groups of UART buses in the 40-pin are shown in the following table

UART bus	RX correspond 40Pin	TX correspond 40Pin
UART1	Pin 16	Pin 15
UART3	Pin 10	Pin 8

5) Use the **serial** command to test the loopback function of the serial port as shown below. If you can see that the read and write contents are consistent, it means that the serial port function is normal.

- a. The test of uart1 is as follows:

```
orangePi@orangePi:~$ sudo serial /dev/ttyAMA1
```

```
W: Hello, Serial Port!
```

```
R: Hello, Serial Port!
```

- b. The test of uart3 is as follows:

```
orangePi@orangePi:~$ sudo serial /dev/ttyAMA3
```

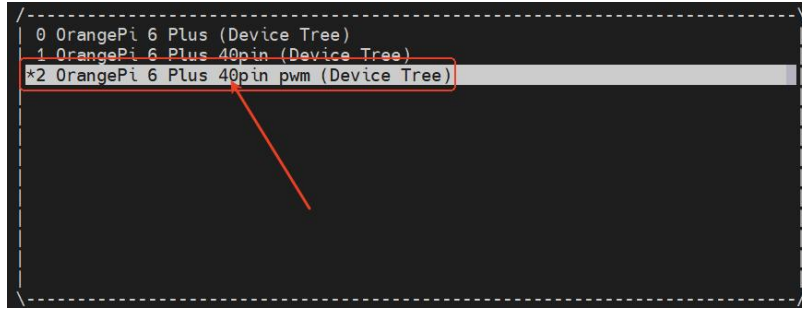
```
W: Hello, Serial Port!
```

```
R: Hello, Serial Port!
```

3. 16. 5. Testing method for 40-pin interface PWM

1) Before testing PWM:

- a. Systems with the Linux 6.1 kernel need to ensure that device tree 2 is used during startup. If the default device tree 0 is being used, please reboot the system to switch to device tree 2 before continuing testing.



- b. On a Linux 6.6 kernel system running ACPI, if the BIOS version shown by the **dmidecode** command is **v1.0_for_opi6plus_linux**, then PWM3 and PWM5 on the 40-pin connector will be unusable. Please refer to the section on "[How to Flash BIOS Firmware to the Development Board's SPI Flash](#)" to update the BIOS firmware.

```
orangepi@orangepi:~$ sudo dmidecode -s bios-version
v1.0_for_opi6plus_linux
```

- c. In a Linux 6.6 kernel system with ACPI, if the BIOS version shown by the **dmidecode** command is **v1.0_for_opi6plus_linux_40pin**, then only PWM5 can be used.

```
orangepi@orangepi:~$ sudo dmidecode -s bios-version
v1.0_for_opi6plus_linux_40pin
```

- d. In a Linux 6.6 kernel system running ACPI, if the BIOS version shown by the **dmidecode** command is **v1.0_for_opi6plus_linux_40pin_pwm**, then both PWM3 and PWM5 can be used.

```
orangepi@orangepi:~$ sudo dmidecode -s bios-version
v1.0_for_opi6plus_linux_40pin_pwm
```

2) As can be seen from the table below, the available PWMs in the 40-pin of the development board are PWM3 and PWM5, a total of 2 PWMs.

复用功能	复用功能	GPIO	DT GPIO序号	ACPI GPIO序号	引脚序号	引脚序号	ACPI GPIO序号	DT GPIO序号	GPIO	复用功能
		3.3V			1	2			5V	
	I2C2_SDA(0x04030000)	GPIO056	65	13	3	4			5V	
	I2C2_CLK	GPIO055	64	12	5	6			GND	
	PWM5(0x04115000)	GPIO097	106	54	7	8	62	114	GPIO105	UART3_TXD(0x040e0000)
		GND			9	10	63	115	GPIO106	UART3_RXD
		GPIO015	32	155	11	12	157	34	GPIO017	
		GPIO016	33	156	13	14			GND	
UART1_TXD(0x040c0000)	PWM3(0x04113000)	GPIO099	108	56	15	16	57	109	GPIO100	UART1_RXD
		3.3V			17	18	158	35	GPIO018	
	SPI1_MOSI	GPIO028	17	130	19	20			GND	
	SPI1_MISO	GPIO025	14	127	21	22	159	36	GPIO019	
	SPI1_CLK	GPIO029	18	131	23	24	128	15	GPIO026	SPI1_CS0
		GND			25	26	129	16	GPIO027	SPI1_CS1
	I2C5_SDA(0x04060000)	GPIO048	57	5	27	28	4	56	GPIO047	I2C5_CLK
		GPIO020	37	160	29	30			GND	
		GPIO023	40	163	31	32	161	38	GPIO021	
		GPIO024	41	164	33	34			GND	
		GPIO057	66	14	35	36	162	39	GPIO022	
		GPIO060	69	17	37	38	18	70	GPIO061	I2C4_CLK
		GND			39	40	19	71	GPIO062	I2C4_SDA(0x04050000)



3) The corresponding pins of PWM in 40Pin are shown in the following table.

PWM bus	correspond 40Pin
PWM3	Pin 15
PWM5	Pin 7

4) After entering the Linux system:

a. The system of Linux 6.1 kernel can be seen from the output of **ls /sys/class/pwm/pwmchip* -l**:

- a) 0x4113000 correspond PWM3, The corresponding PWM chip is PWM chip 1.
- b) 0x4115000 correspond PWM5, The corresponding PWM chip is PWM chip2

```
orangePi@orangePi:~$ ls /sys/class/pwm
pwmchip0  pwmchip1  pwmchip2  pwmchip3
orangePi@orangePi:~$ ls /sys/class/pwm/pwmchip* -l
```

```
orangePi@orangePi:~$ ls /sys/class/pwm/pwmchip* -l
lrwxrwxrwx 1 root root 0 Aug 29 19:48 /sys/class/pwm/pwmchip0 -> ../.. /devices/platform/soc@0/4110000.pwm0/pwm/pwmchip0
lrwxrwxrwx 1 root root 0 Aug 29 19:48 /sys/class/pwm/pwmchip1 -> ../.. /devices/platform/soc@0/4113000.pwm3/pwm/pwmchip1
lrwxrwxrwx 1 root root 0 Aug 29 19:48 /sys/class/pwm/pwmchip2 -> ../.. /devices/platform/soc@0/4115000.pwm5/pwm/pwmchip2
lrwxrwxrwx 1 root root 0 Aug 29 19:48 /sys/class/pwm/pwmchip3 -> ../.. /devices/platform/soc@0/4070000.i2c/i2c-6/6-0076/4070000.i2c-ec@76:ec-pwm/pwm/pwmchip3
```

b. The output of the following command shows that the Linux 6.6 ACPI kernel system:

- a) acpi:CIXH2011:3 correspond PWM3, correspond pwmchip is pwmchip1
- b) acpi:CIXH2011:5 correspond PWM5, correspond pwmchip is pwmchip2

```
orangePi@orangePi:~$ ls /sys/class/pwm
pwmchip0  pwmchip1  pwmchip2
orangePi@orangePi:~$ sudo cat /sys/class/pwm/pwmchip0/device/modalias
acpi:CIXH2011:0
orangePi@orangePi:~$ sudo cat /sys/class/pwm/pwmchip1/device/modalias
acpi:CIXH2011:3
orangePi@orangePi:~$ sudo cat /sys/class/pwm/pwmchip2/device/modalias
acpi:CIXH2011:5
```

5) Then use the following command to make pwm output a 50Hz square wave.

a. The test command for PWM3 is as follows:

```
orangePi@orangePi:~$ sudo -i
root@orangePi:~# echo 0 > /sys/class/pwm/pwmchip1/export
```



```

root@orangepi:~# echo 20000 > /sys/class/pwm/pwmchip1/pwm0/period
root@orangepi:~# echo 10000 > /sys/class/pwm/pwmchip1/pwm0/duty_cycle
root@orangepi:~# echo 1 > /sys/class/pwm/pwmchip1/pwm0/enable

```

b. The test command for PWM5 is as follows:

```

orangepi@orangepi:~$ sudo -i
root@orangepi:~# echo 0 > /sys/class/pwm/pwmchip2/export
root@orangepi:~# echo 20000 > /sys/class/pwm/pwmchip2/pwm0/period
root@orangepi:~# echo 10000 > /sys/class/pwm/pwmchip2/pwm0/duty_cycle
root@orangepi:~# echo 1 > /sys/class/pwm/pwmchip2/pwm0/enable

```



3. 17. Test of some programming languages supported by Linux system

3. 17. 1. Debian Bookworm System

1) Debian Bookworm is installed with the gcc compilation tool chain by default, which can compile C language programs directly in the Linux system of the development board.

a. The version of gcc is as follows:

```

orangepi@orangepi:~$ gcc --version
gcc (Debian 12.2.0-14+deb12u1) 12.2.0
Copyright (C) 2022 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。

```

orangepi@orangepi:~$ vim hello_world.c
#include <stdio.h>

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

```



```
    return 0;
}
```

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

```
orange@orange:~$ gcc -o hello_world hello_world.c
orange@orange:~$ ./hello_world
Hello World!
```

- 2) Debian Bookworm has Python 3 installed by default.

- a. The specific Python versions are as follows:

```
orange@orange:~$ python3
Python 3.11.2 (main, Apr 28 2025, 14:11:48) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Use the Ctrl+D shortcut key to exit Python's interactive mode.

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

```
orange@orange:~$ vim hello_world.py
print('Hello World!')
```

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

```
orange@orange:~$ python3 hello_world.py
Hello World!
```

- 3) Debian Bookworm does not have Java compilation tools and runtime environment installed by default.

- a. You can use the following command to install openjdk. The latest version in Debian Bookworm is openjdk-17.

```
orange@orange:~$ sudo apt install -y openjdk-17-jdk
```

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

```
orange@orange:~$ java --version
openjdk 17.0.16 2025-07-15
OpenJDK Runtime Environment (build 17.0.16+8-Debian-1deb12u1)
OpenJDK 64-Bit Server VM (build 17.0.16+8-Debian-1deb12u1, mixed mode, sharing)
```

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

```
orange@orange:~$ 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**

```
orangeypi@orangeypi:~$ javac hello_world.java
orangeypi@orangeypi:~$ java hello_world
Hello World!
```

3. 18. How to install kernel header files

1) The Linux image released by Orange Pi comes with the deb package of the kernel header file by default, which is stored in **/opt/**.

a. Linux6.1:

```
orangeypi@orangeypi:~$ ls /opt/linux-headers*
/opt/linux-headers-current-cix_x.x.x_arm64.deb
```

b. Linux6.6:

```
orangeypi@orangeypi:~$ ls /opt/linux-headers*
/opt/linux-headers-next-cix_x.x.x_arm64.deb
```

2) Use the following command to install the kernel header file deb package.

a. Linux6.1:

```
orangeypi@orangeypi:~$ sudo dpkg -i /opt/linux-headers-current-cix_*_arm64.deb
```

b. Linux6.6:

```
orangeypi@orangeypi:~$ sudo dpkg -i /opt/linux-headers-next-cix_*_arm64.deb
```

3) After installation, you can see the folder where the kernel header files are located under **/usr/src**.

a. Linux6.1:

```
orangeypi@orangeypi:~$ ls /usr/src
... linux-headers-6.1.44-cix ...
```

b. Linux6.6:

```
orangeypi@orangeypi:~$ ls /usr/src
... linux-headers-6.6.89-cix ...
```



4) Then you can test the kernel header file using the hello kernel module in the **/usr/src** directory.

- a. Enter the **/usr/src/hello** directory.

```
orange@orange:~$ cd /usr/src/hello
```

- b. 然后使用 **make** 命令编译 hello 内核模块，编译过程的输出如下所示：

- a) Linux6.1:

```
orange@orange:/usr/src/hello$ sudo make
make -C /lib/modules/6.1.44-cix/build M=/usr/src/hello modules
make[1]: Entering directory '/usr/src/linux-headers-6.1.44-cix'
CC [M] /usr/src/hello/hello.o
MODPOST /usr/src/hello/Module.symvers
CC [M] /usr/src/hello/hello.mod.o
LD [M] /usr/src/hello/hello.ko
make[1]: Leaving directory '/usr/src/linux-headers-6.1.44-cix'
```

- b) Linux6.6:

```
orange@orange:/usr/src/hello$ sudo make
make -C /lib/modules/6.6.89-cix/build M=/usr/src/hello modules
make[1]: Entering directory '/usr/src/linux-headers-6.6.89-cix'
CC [M] /usr/src/hello/hello.o
MODPOST /usr/src/hello/Module.symvers
CC [M] /usr/src/hello/hello.mod.o
LD [M] /usr/src/hello/hello.ko
make[1]: Leaving directory '/usr/src/linux-headers-6.6.89-cix'
```

- c. After compilation, the **hello.ko** kernel module will be generated

```
orange@orange:/usr/src/hello$ ls *.ko
hello.ko
```

- d. Use the **insmod** command to insert the **hello.ko** kernel module into the kernel

```
orange@orange:/usr/src/hello$ sudo insmod hello.ko
```

- e. Then use the **dmesg** command to view the output of the **hello.ko** kernel module.
If you can see the following output, it means that the **hello.ko** kernel module is loaded correctly.

```
orange@orange:/usr/src/hello$ dmesg | grep "Hello"
[ 2871.893988] Hello Orange Pi -- init
```

- f. Use the **rmmod** command to uninstall the **hello.ko** kernel module

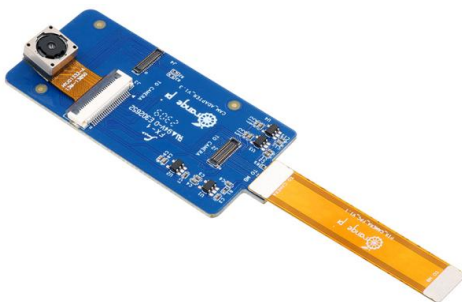


```
orangepi@orangepi:/usr/src/hello$ sudo rmmod hello
orangepi@orangepi:/usr/src/hello$ dmesg | grep "Hello"
[ 2871.893988] Hello Orange Pi -- init
[ 3173.800892] Hello Orange Pi -- exit
```

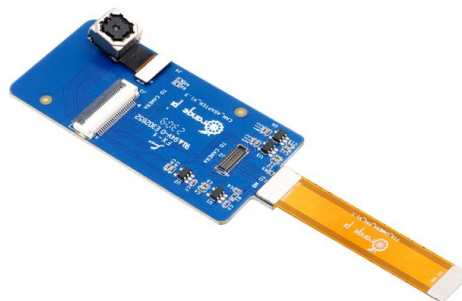
3. 19. Test Methods for OV13850 and OV13855 MIPI Cameras

Currently the development board supports two MIPI cameras, OV13850 and OV13855. The specific pictures are shown below:

- a. 13MP OV13850 camera with MIPI interface



- b. 13MP OV13855 camera with MIPI interface

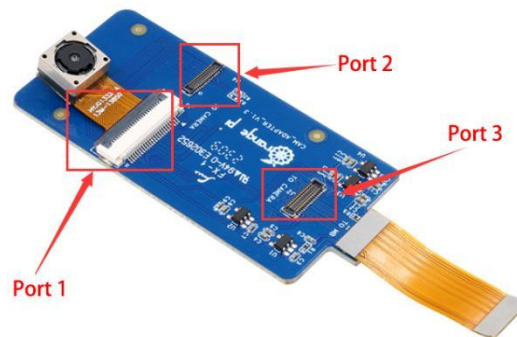


The adapter board and FPC cable used by the OV13850 and OV13855 cameras are the same, but the two cameras connect to the adapter board in different locations. The FPC cable is shown in the figure below. Please note that the FPC cable has a direction: the end marked **"TO MB"** should be plugged into the camera port on the development board, and the end marked **"TO CAMERA"** should be plugged into the camera adapter board.

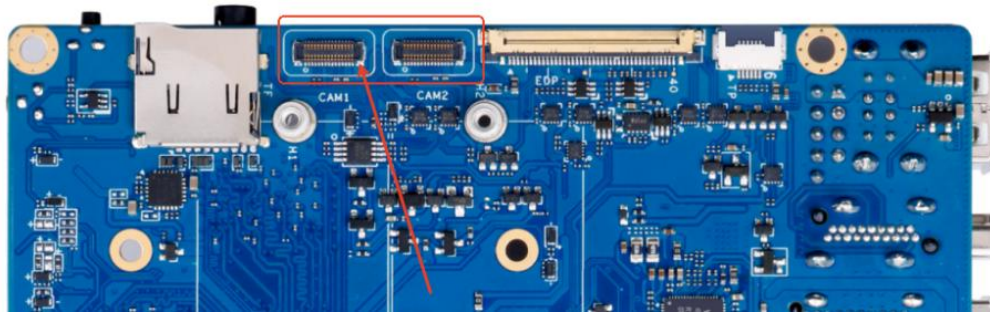


There are three camera interfaces on the camera adapter board. Only one can be used at a time, as shown in the figure below.:

- a. **Interface 1 is connected to the OV13850 camera**
- b. **Interface 2 is connected to the OV13855 camera**
- c. Interface 3 is not used, so just ignore it.

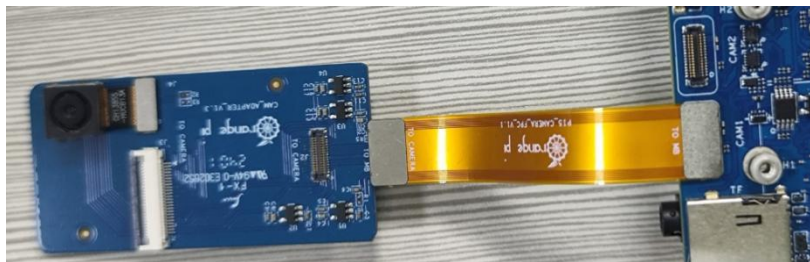


There are two camera interfaces on the development board, numbered CAM1 and CAM2, and their locations are shown below.:



The method of plugging the camera adapter board into the camera interface of the development board is shown below. Note that both CAM1 and CAM2 interfaces can connect to OV13850 and OV13855 cameras, and can also connect two cameras of the same model or different models at the same time.

- a. The method of connecting the CAM1 interface to the camera is as follows:



- b. The method of connecting the CAM2 interface to the camera is as follows:



After connecting the camera to the development board, we can use the following method to test the camera:

- a. Use the following command to ensure that the `armcb_isp_v4l2.ko` kernel module has been loaded normally.

```
orangepi@orangepi:~$ lsmod | grep isp_v4l2
armcb_isp_v4l2      233472  1
```

- b. Then set the parameters of the `isp_app` program:

- a) If testing a **single CAM1 camera**, ensure that the parameter following `isp_app` in `isp-daemon.service` is `-s 0`. If you have modified `isp-daemon.service`, reboot the system for the changes to take effect. If you have not modified anything, rebooting is not necessary.

```
orangepi@orangepi:~$ sudo vim /lib/systemd/system/isp-daemon.service
[Unit]
Description=ISP Daemon
After=network.target load-isp-modules.service

[Service]
Type=simple
Environment=LD_LIBRARY_PATH="/usr/share/cix/lib"
ExecStart=/usr/bin/isp_app -s 0 &
Restart=always
RestartSec=1
StartLimitInterval=10
```



```
StartLimitBurst=5
```

```
[Install]
```

```
WantedBy=multi-user.target
```

b) If testing a **single CAM2 camera**, ensure that the parameter following `isp_app` in **isp-daemon.service** is **-s 1**. If you have modified **isp-daemon.service**, reboot the system for the changes to take effect. If you have not modified anything, rebooting is not necessary.

```
orangepi@orangepi:~$ sudo vim /lib/systemd/system/isp-daemon.service
```

```
[Unit]
```

```
Description=ISP Daemon
```

```
After=network.target load-isp-modules.service
```

```
[Service]
```

```
Type=simple
```

```
Environment=LD_LIBRARY_PATH="/usr/share/cix/lib"
```

```
ExecStart=/usr/bin/isp_app -s 1 &
```

```
Restart=always
```

```
RestartSec=1
```

```
StartLimitInterval=10
```

```
StartLimitBurst=5
```

```
[Install]
```

```
WantedBy=multi-user.target
```

c) If testing the **CAM1 and CAM2 dual cameras**, ensure that the parameter following `isp_app` in **isp-daemon.service** is **-m 2**. If you have modified **isp-daemon.service**, reboot the system for the changes to take effect. If you have not modified anything, reboot is not necessary.

```
orangepi@orangepi:~$ sudo vim /lib/systemd/system/isp-daemon.service
```

```
[Unit]
```

```
Description=ISP Daemon
```

```
After=network.target load-isp-modules.service
```

```
[Service]
```

```
Type=simple
```



```
Environment=LD_LIBRARY_PATH="/usr/share/cix/lib"
ExecStart=/usr/bin/isp_app -m 2 &
Restart=always
RestartSec=1
StartLimitInterval=10
StartLimitBurst=5

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

c. Make sure `isp_app` is running. If you can see the output of `/usr/bin/isp_app`, it is running properly.

```
orange@orange:~$ pgrep -a isp_app
3528 /usr/bin/isp_app -s 0 &
```

d. Run the `v4l2_videos.sh` script to ensure that you can see the two devices `armcb-00-vid-cap` and `armcb-01-vid-cap` in the output.

a) The output of the Linux 6.1 kernel system is shown below, where `armcb-00-vid-cap` corresponds to CAM1, and the video node in the output below is `/dev/video1`. `armcb-01-vid-cap` corresponds to CAM2, and the video node in the output below is `/dev/video3`.

```
orange@orange:~$ sudo v4l2_videos.sh
Video Device and Name Correspondence:
=====
/dev/video0 : armcb-config
/dev/video1 : armcb-00-vid-cap
/dev/video2 : armcb-02-vid-cap
/dev/video3 : armcb-01-vid-cap
/dev/video4 : mvxdec
/dev/video5 : mvxenc
```

b) The output of the Linux 6.6 kernel system is shown below, where `armcb-00-vid-cap` corresponds to CAM1, and the video node in the output below is `/dev/video9`; `armcb-01-vid-cap` corresponds to CAM2, and the video node in the output below is `/dev/video8`.

```
orange@orange:~$ sudo v4l2_videos.sh
Video Device and Name Correspondence:
=====
```

```
/dev/video0 : mvxdec
/dev/video1 : mvxenc
/dev/video2 : armcb-config
/dev/video3 : cix-bridge.0.capture
/dev/video7 : armcb-02-vid-cap
/dev/video8 : armcb-01-vid-cap
/dev/video9 : armcb-00-vid-cap
```

e. Open another terminal on the desktop and run the camera test script.

a) Connect only one camera and test the CAM1 interface with the following command:

```
orange_pi@orange_pi:~$ cam1_test.sh
```

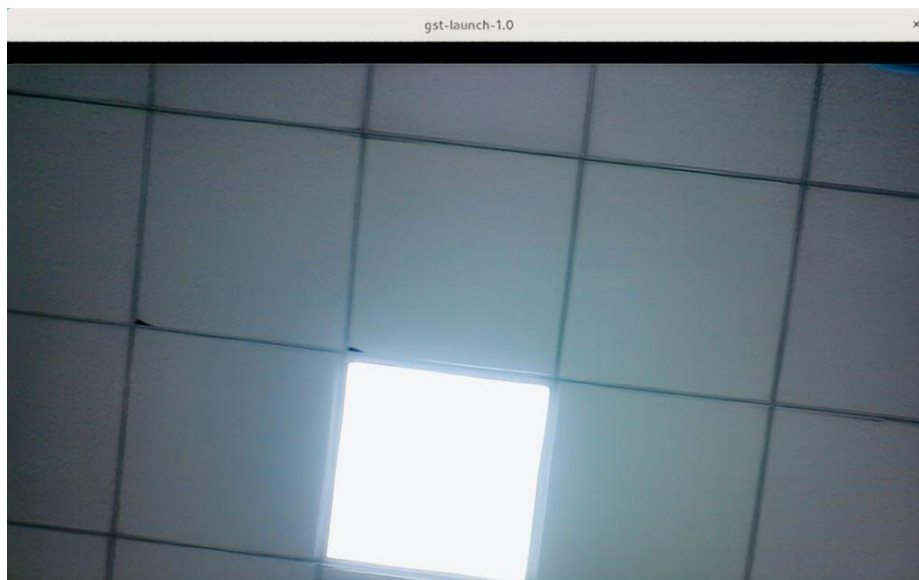
b) Connect only one camera and test the CAM2 interface with the following command:

```
orange_pi@orange_pi:~$ cam2_test.sh
```

c) Connect two cameras and test the CAM1 and CAM2 interfaces using the following commands:

```
orange_pi@orange_pi:~$ dual_cam_test.sh
```

f. Then you can see the camera preview. **Note that if you test two cameras, the preview images will overlap. Use the mouse to drag the preview image of one camera to see the preview image of the other camera**





If the camera does not display any image, please check whether the socket connecting the camera to the development board is loose

3. 20. How to use RTC

1) There is an RTC battery interface reserved on the development board, the location is shown below:



2) The RTC battery you need to buy is as shown below. The interface is 2-pin, 1.0mm pitch.



3) After connecting the RTC battery to the development board, use the following method to test whether the RTC is working properly.:

- a. Boot into the system and use it to record the current time of the system;
- b. Then use the poweroff command to shut down the system normally;
- c. Then unplug the power supply, make sure the development board is not connected to the network cable and wireless network card, and wait a few minutes.;
- d. Then restart the system. If you see the time advance by a few minutes after entering the system, it means that the RTC module and battery are working properly.

4) The command to view RTC information through the procfs interface of the Linux system is:



```
orangePi@orangePi:~$ cat /proc/driver/rtc
```

```
rtc_time      : 13:15:25
rtc_date      : 2025-09-01
alarm_time    : 00:00:00
alarm_date    : 2021-10-16
alarm_IRQ     : no
alarm_pending : no
update IRQ enabled : no
periodic IRQ enabled : no
periodic IRQ frequency : 1
max user IRQ frequency : 1
24hr         : yes
```

3. 21. GPU testing methods

1) Before testing the performance of the GPU, you can set both the CPU and GPU to performance mode. The command is as follows:

- a. The command to set the CPU to performance mode is as follows:

```
orangePi@orangePi:~$ echo performance | sudo tee \
/sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
```

- b. The command to set the GPU to performance mode is as follows:

- a) Linux6.1:

```
orangePi@orangePi:~$ echo performance | sudo tee \
/sys/class/misc/mali0/device/devfreq/15000000.gpu/governor
```

- b) Linux6.6 ACPI:

```
orangePi@orangePi:~$ echo performance | sudo tee \
/sys/class/misc/mali0/device/devfreq/CIXH5000\00/governor
```

- c. After the settings are completed, if the following commands all output "performance", it means that the performance mode of the CPU and GPU has been set successfully.

- a) Linux6.1:

```
orangePi@orangePi:~$ sudo cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
orangePi@orangePi:~$ sudo cat /sys/class/misc/mali0/device/devfreq/15000000.gpu/governor
```




```

orangepi@orangepi6plus:~$ sudo cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
[sudo] password for orangepi:
performance
performance
performance
performance
performance
performance
performance
performance
performance
performance
performance
performance
performance
performance
performance
orangepi@orangepi6plus:~$ sudo cat /sys/class/misc/mali0/device/devfreq/15000000.gpu/governor
performance
orangepi@orangepi6plus:~$

```

b) Linux6.6 ACPI:

```

orangepi@orangepi:~$ sudo cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
orangepi@orangepi:~$ sudo cat /sys/class/misc/mali0/device/devfreq/CIXH5000\:00/governor
orangepi@orangepi6plus:~$ sudo cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
performance
performance
performance
performance
performance
performance
performance
performance
performance
performance
performance
performance
performance
performance
performance
orangepi@orangepi6plus:~$ sudo cat /sys/class/misc/mali0/device/devfreq/CIXH5000\:00/governor
performance
orangepi@orangepi6plus:~$

```

2) The following table shows how to view GPU usage, GPU clock frequency, and the number of GPU cores used:

```

orangepi@orangepi:~$ sudo gpu_utilization_clock_tracing
GPU utilisation: 55.67%, clock: 1000.00MHz, core_number: 10, core_mask: 0x550555

```

3) glmark2-es2-wayland is used to test the performance of GPU OpenGL ES in the Wayland environment. The specific test method is as follows:

- a. Open a terminal on your **desktop** and run the following command.

```

orangepi@orangepi:~$ glmark2-es2-wayland

```

- b. If GL_RENDERER is **Mali-G720-Immortalis**, it means that the hardware GPU is used.



```

orange@orangepi6plus: ~
orangepi@orangepi6plus:~$ glmark2-es2-wayland
=====
glmark2 2023.01
=====
OpenGL Information
GL_VENDOR:      ARM
GL_RENDERER:    Mali-G720-Immortalis
GL_VERSION:     OpenGL ES 3.2 v1.r53p0-00eac0.1b3b3991ed8c4669c37cdca17827392d
Surface Config: buf=32 r=8 g=8 b=8 a=8 depth=24 stencil=0 samples=0
Surface Size:   800x600 windowed

```

c. The test output results are shown below, with the benchmark score at the bottom.

```

orange@orangepi6plus: ~
orangepi@orangepi6plus:~$ glmark2-es2-wayland
=====
glmark2 2023.01
=====
OpenGL Information
GL_VENDOR:      ARM
GL_RENDERER:    Mali-G720-Immortalis
GL_VERSION:     OpenGL ES 3.2 v1.r53p0-00eac0.1b3b3991ed8c4669c37cdca17827392d
Surface Config: buf=32 r=8 g=8 b=8 a=8 depth=24 stencil=0 samples=0
Surface Size:   800x600 windowed
=====
[build] use-vbo=false: FPS: 8223 FrameTime: 0.122 ms
[build] use-vbo=true: FPS: 9138 FrameTime: 0.109 ms
[texture] texture-filter=nearest: FPS: 9336 FrameTime: 0.107 ms
[texture] texture-filter=linear: FPS: 10584 FrameTime: 0.094 ms
[texture] texture-filter=mipmap: FPS: 10583 FrameTime: 0.094 ms
[shading] shading=gouraud: FPS: 8258 FrameTime: 0.121 ms
[shading] shading=blinn-phong-inf: FPS: 8879 FrameTime: 0.113 ms
[shading] shading=phong: FPS: 8195 FrameTime: 0.122 ms
[shading] shading=cel: FPS: 7639 FrameTime: 0.131 ms
[bump] bump-render=high-poly: FPS: 4861 FrameTime: 0.206 ms
[bump] bump-render=normals: FPS: 9529 FrameTime: 0.105 ms
[bump] bump-render=height: FPS: 8890 FrameTime: 0.112 ms
[effect2d] kernel=0,1,0;1,-4,1;0,1,0;: FPS: 8739 FrameTime: 0.114 ms
[effect2d] kernel=1,1,1,1,1;1,1,1,1,1;1,1,1,1,1;: FPS: 7123 FrameTime: 0.140 ms
[pulsar] light=false:quads=5:texture=false: FPS: 9057 FrameTime: 0.110 ms
[desktop] blur-radius=5:effect=blur:passes=1:separable=true:windows=4: FPS: 3665 FrameTime: 0.273 ms
[desktop] effect=shadow:windows=4: FPS: 7857 FrameTime: 0.127 ms
[buffer] columns=200:interleave=false:update-dispersion=0.9:update-fraction=0.5:update-method=map: FPS: 1086 FrameTime: 0.921 ms
[buffer] columns=200:interleave=false:update-dispersion=0.9:update-fraction=0.5:update-method=subdata: FPS: 1131 FrameTime: 0.885 ms
[buffer] columns=200:interleave=true:update-dispersion=0.9:update-fraction=0.5:update-method=map: FPS: 1422 FrameTime: 0.704 ms
[ideas] speed=duration: FPS: 4541 FrameTime: 0.220 ms
[jellyfish] <default>: FPS: 7302 FrameTime: 0.137 ms
[terrain] <default>: FPS: 681 FrameTime: 1.469 ms
[shadow] <default>: FPS: 7125 FrameTime: 0.140 ms
[refract] <default>: FPS: 1653 FrameTime: 0.605 ms
[conditionals] fragment-steps=0:vertex-steps=0: FPS: 8789 FrameTime: 0.114 ms
[conditionals] fragment-steps=5:vertex-steps=0: FPS: 8556 FrameTime: 0.117 ms
[conditionals] fragment-steps=0:vertex-steps=5: FPS: 9373 FrameTime: 0.107 ms
[function] fragment-complexity=low:fragment-steps=5: FPS: 9356 FrameTime: 0.107 ms
[function] fragment-complexity=medium:fragment-steps=5: FPS: 8817 FrameTime: 0.113 ms
[loop] fragment-loop=false:fragment-steps=5:vertex-steps=5: FPS: 9309 FrameTime: 0.107 ms
[loop] fragment-steps=5:fragment-uniform=false:vertex-steps=5: FPS: 9407 FrameTime: 0.106 ms
[loop] fragment-steps=5:fragment-uniform=true:vertex-steps=5: FPS: 8203 FrameTime: 0.122 ms
=====
glmark2 Score: 7190
=====

```

4) glmark2-es2 is used to test the performance of GPU OpenGL ES in X11 environment. The specific test method is as follows:

a. Open a terminal on your **desktop** and run the following command.



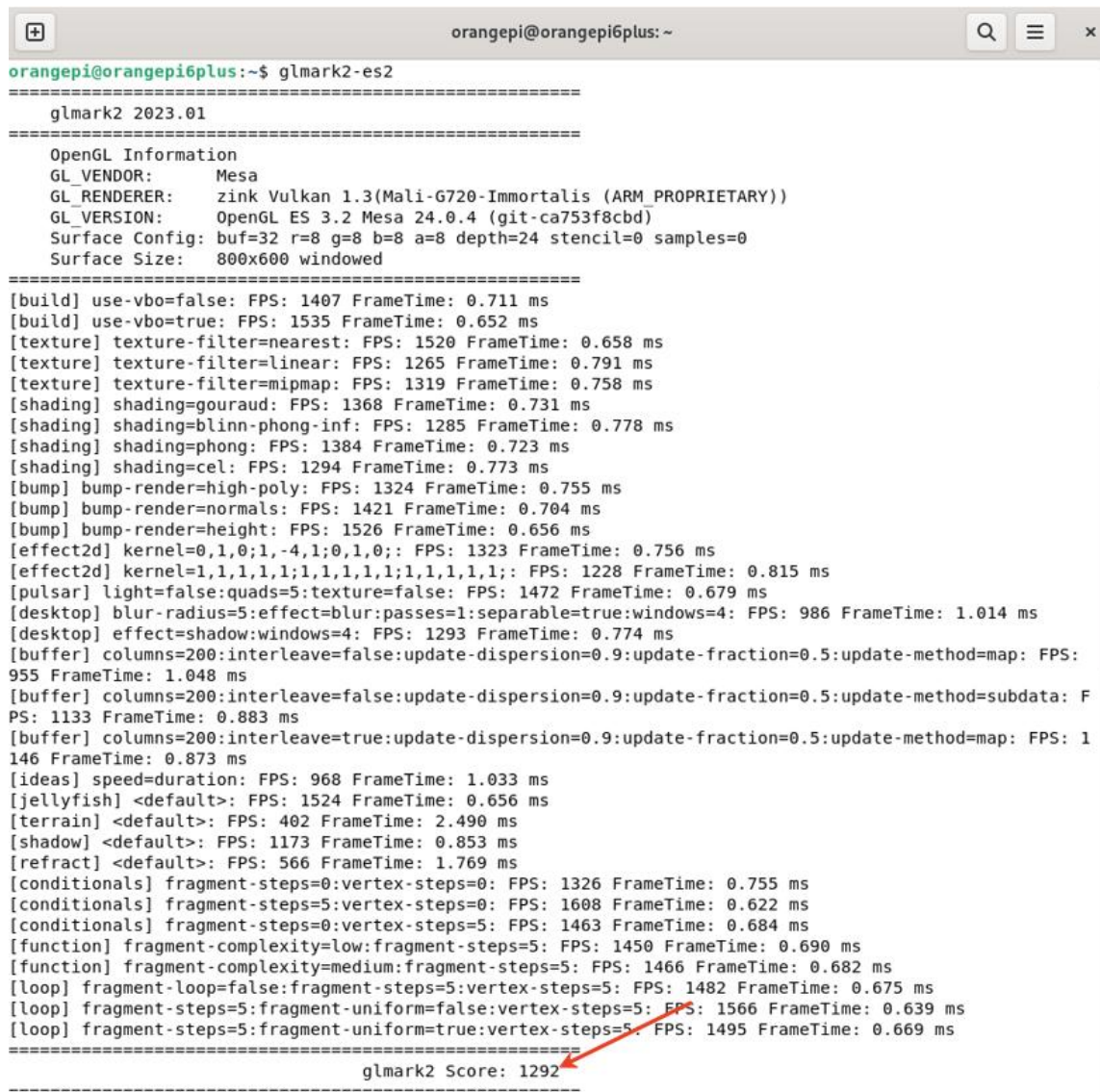
```
orange@orange:~$ glmark2-es2
```

- b. If GL_RENDERER is **Mali-G720-Immortalis**, it means that the hardware GPU is used.

```
orange@orange6plus:~$ glmark2-es2
=====
glmark2 2023.01
=====

OpenGL Information
GL_VENDOR:      Mesa
GL_RENDERER:    zink Vulkan 1.3(Mali-G720-Immortalis (ARM_PROPRIETARY))
GL_VERSION:     OpenGL ES 3.2 Mesa 24.0.4 (git-ca753f8cbd)
Surface Config: buf=32 r=8 g=8 b=8 a=8 depth=24 stencil=0 samples=0
Surface Size:   800x600 windowed
```

- c. The test output results are shown below, with the running score at the bottom.



```
orange@orange6plus:~$ glmark2-es2
=====
glmark2 2023.01
=====

OpenGL Information
GL_VENDOR:      Mesa
GL_RENDERER:    zink Vulkan 1.3(Mali-G720-Immortalis (ARM_PROPRIETARY))
GL_VERSION:     OpenGL ES 3.2 Mesa 24.0.4 (git-ca753f8cbd)
Surface Config: buf=32 r=8 g=8 b=8 a=8 depth=24 stencil=0 samples=0
Surface Size:   800x600 windowed

=====
[build] use-vbo=false: FPS: 1407 FrameTime: 0.711 ms
[build] use-vbo=true: FPS: 1535 FrameTime: 0.652 ms
[texture] texture-filter=nearest: FPS: 1520 FrameTime: 0.658 ms
[texture] texture-filter=linear: FPS: 1265 FrameTime: 0.791 ms
[texture] texture-filter=mipmap: FPS: 1319 FrameTime: 0.758 ms
[shading] shading=gouraud: FPS: 1368 FrameTime: 0.731 ms
[shading] shading=blinn-phong-inf: FPS: 1285 FrameTime: 0.778 ms
[shading] shading=phong: FPS: 1384 FrameTime: 0.723 ms
[shading] shading=cel: FPS: 1294 FrameTime: 0.773 ms
[bump] bump-render=high-poly: FPS: 1324 FrameTime: 0.755 ms
[bump] bump-render=normals: FPS: 1421 FrameTime: 0.704 ms
[bump] bump-render=height: FPS: 1526 FrameTime: 0.656 ms
[effect2d] kernel=0,1,0;1,-4,1;0,1,0,: FPS: 1323 FrameTime: 0.756 ms
[effect2d] kernel=1,1,1,1,1;1,1,1,1,1;1,1,1,1,1,: FPS: 1228 FrameTime: 0.815 ms
[pulsar] light=false:quads=5:texture=false: FPS: 1472 FrameTime: 0.679 ms
[desktop] blur-radius=5:effect=blur:passes=1:separable=true:windows=4: FPS: 986 FrameTime: 1.014 ms
[desktop] effect=shadow:windows=4: FPS: 1293 FrameTime: 0.774 ms
[buffer] columns=200:interleave=false:update-dispersion=0.9:update-fraction=0.5:update-method=map: FPS: 955 FrameTime: 1.048 ms
[buffer] columns=200:interleave=false:update-dispersion=0.9:update-fraction=0.5:update-method=subdata: FPS: 1133 FrameTime: 0.883 ms
[buffer] columns=200:interleave=true:update-dispersion=0.9:update-fraction=0.5:update-method=map: FPS: 1146 FrameTime: 0.873 ms
[ideas] speed=duration: FPS: 968 FrameTime: 1.033 ms
[jellyfish] <default>: FPS: 1524 FrameTime: 0.656 ms
[terrain] <default>: FPS: 402 FrameTime: 2.490 ms
[shadow] <default>: FPS: 1173 FrameTime: 0.853 ms
[refract] <default>: FPS: 566 FrameTime: 1.769 ms
[conditionals] fragment-steps=0:vertex-steps=0: FPS: 1326 FrameTime: 0.755 ms
[conditionals] fragment-steps=5:vertex-steps=0: FPS: 1608 FrameTime: 0.622 ms
[conditionals] fragment-steps=0:vertex-steps=5: FPS: 1463 FrameTime: 0.684 ms
[function] fragment-complexity=low:fragment-steps=5: FPS: 1450 FrameTime: 0.690 ms
[function] fragment-complexity=medium:fragment-steps=5: FPS: 1466 FrameTime: 0.682 ms
[loop] fragment-loop=false:fragment-steps=5:vertex-steps=5: FPS: 1482 FrameTime: 0.675 ms
[loop] fragment-steps=5:fragment-uniform=false:vertex-steps=5: FPS: 1566 FrameTime: 0.639 ms
[loop] fragment-steps=5:fragment-uniform=true:vertex-steps=5: FPS: 1495 FrameTime: 0.669 ms
=====
glmark2 Score: 1292
=====
```

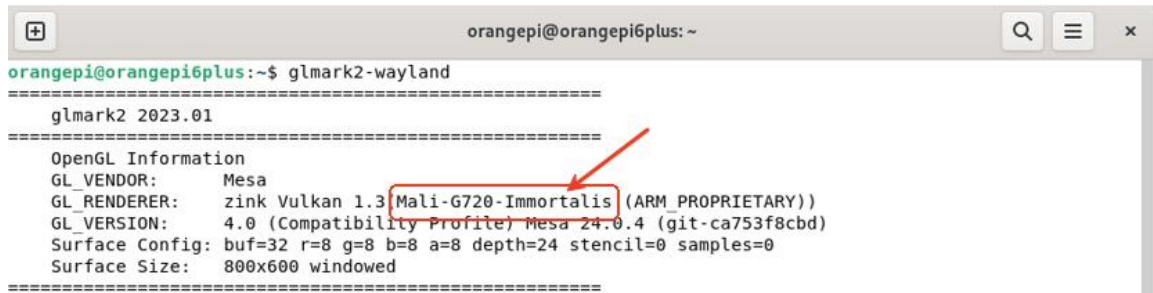


5) glmark2-wayland is used to test the performance of GPU OpenGL in the Wayland environment. The specific test method is as follows:

- a. Open a terminal on your **desktop** and run the following command.

```
orange@orange:~$ glmark2-wayland
```

- b. If GL_RENDERER is **Mali-G720-Immortalis**, it means that the hardware GPU is used.



```
orange@orange6plus: ~
glmark2 2023.01
=====
OpenGL Information
GL_VENDOR:      Mesa
GL_RENDERER:    zink Vulkan 1.3 Mali-G720-Immortalis (ARM_PROPRIETARY)
GL_VERSION:     4.0 (Compatibility Profile) Mesa 24.0.4 (git-ca753f8cbd)
Surface Config: buf=32 r=8 g=8 b=8 a=8 depth=24 stencil=0 samples=0
Surface Size:   800x600 windowed
=====
```

- c. The test output results are shown below, with the benchmark score at the bottom.



```

orangePi@orangePi6plus: ~
=====
glmark2 2023.01
=====
OpenGL Information
GL_VENDOR:      Mesa
GL_RENDERER:    zink Vulkan 1.3(Mali-G720-Immortalis (ARM_PROPRIETARY))
GL_VERSION:     4.0 (Compatibility Profile) Mesa 24.0.4 (git-ca753f8cbd)
Surface Config: buf=32 r=8 g=8 b=8 a=8 depth=24 stencil=0 samples=0
Surface Size:   800x600 windowed
=====
[build] use-vbo=false: FPS: 2837 FrameTime: 0.353 ms
[build] use-vbo=true: FPS: 3711 FrameTime: 0.270 ms
[texture] texture-filter=nearest: FPS: 3541 FrameTime: 0.282 ms
[texture] texture-filter=linear: FPS: 3307 FrameTime: 0.302 ms
[texture] texture-filter=mipmap: FPS: 3129 FrameTime: 0.320 ms
[shading] shading=gouraud: FPS: 2817 FrameTime: 0.355 ms
[shading] shading=blinn-phong-inf: FPS: 2750 FrameTime: 0.364 ms
[shading] shading=phong: FPS: 2537 FrameTime: 0.394 ms
[shading] shading=cel: FPS: 2214 FrameTime: 0.452 ms
[bump] bump-render=high-poly: FPS: 1682 FrameTime: 0.595 ms
[bump] bump-render=normals: FPS: 1941 FrameTime: 0.515 ms
[bump] bump-render=height: FPS: 3533 FrameTime: 0.283 ms
[effect2d] kernel=0,1,0;1,-4,1;0,1,0;: FPS: 4141 FrameTime: 0.241 ms
[effect2d] kernel=1,1,1,1,1;1,1,1,1,1;1,1,1,1,1;: FPS: 3006 FrameTime: 0.333 ms
[pulsar] light=false:quads=5:texture=false: FPS: 3975 FrameTime: 0.252 ms
[desktop] blur-radius=5:effect=blur:passes=1:separable=true:windows=4: FPS: 1561 FrameTime: 0.641 ms
[desktop] effect=shadow:windows=4: FPS: 2038 FrameTime: 0.491 ms
[buffer] columns=200:interleave=false:update-dispersion=0.9:update-fraction=0.5:update-method=map: FPS: 1339 FrameTime: 0.747 ms
[buffer] columns=200:interleave=false:update-dispersion=0.9:update-fraction=0.5:update-method=subdata: FPS: 1740 FrameTime: 0.575 ms
[buffer] columns=200:interleave=true:update-dispersion=0.9:update-fraction=0.5:update-method=map: FPS: 1738 FrameTime: 0.576 ms
[ideas] speed=duration: FPS: 1221 FrameTime: 0.819 ms
[jellyfish] <default>: FPS: 2901 FrameTime: 0.345 ms
[terrain] <default>: FPS: 396 FrameTime: 2.525 ms
[shadow] <default>: FPS: 2464 FrameTime: 0.406 ms
[refract] <default>: FPS: 611 FrameTime: 1.638 ms
[conditionals] fragment-steps=0:vertex-steps=0: FPS: 2937 FrameTime: 0.341 ms
[conditionals] fragment-steps=5:vertex-steps=0: FPS: 3325 FrameTime: 0.301 ms
[conditionals] fragment-steps=0:vertex-steps=5: FPS: 2500 FrameTime: 0.400 ms
[function] fragment-complexity=low:fragment-steps=5: FPS: 2382 FrameTime: 0.420 ms
[function] fragment-complexity=medium:fragment-steps=5: FPS: 2644 FrameTime: 0.378 ms
[loop] fragment-loop=false:fragment-steps=5:vertex-steps=5: FPS: 3995 FrameTime: 0.250 ms
[loop] fragment-steps=5:fragment-uniform=false:vertex-steps=5: FPS: 3664 FrameTime: 0.273 ms
[loop] fragment-steps=5:fragment-uniform=true:vertex-steps=5: FPS: 2599 FrameTime: 0.385 ms
=====
glmark2 Score: 2580
=====

```

6) glmark2 is used to test the OpenGL performance of the GPU in the X11 environment.

The specific test method is as follows:

- a. Open a terminal on your **desktop** and run the following command.

```
orangePi@orangePi:~$ glmark2
```

- b. If GL_RENDERER is **Mali-G720-Immortalis**, it means that the hardware GPU is used.



```

orange_pi@orange_pi6plus: ~
orange_pi@orange_pi6plus:~$ glmark2
=====
glmark2 2023.01
=====
OpenGL Information
GL_VENDOR:      Mesa
GL_RENDERER:    zink Vulkan 1.3(Mali-G720-Immortalis (ARM_PROPRIETARY))
GL_VERSION:     4.0 (Compatibility Profile) Mesa 24.0.4 (git-ca753f8cbd)
Surface Config: buf=32 r=8 g=8 b=8 a=8 depth=24 stencil=0 samples=0
Surface Size:   800x600 windowed
=====

```

c. The test output results are shown below, with the running score at the bottom.

```

orange_pi@orange_pi6plus: ~
orange_pi@orange_pi6plus:~$ glmark2
=====
glmark2 2023.01
=====
OpenGL Information
GL_VENDOR:      Mesa
GL_RENDERER:    zink Vulkan 1.3(Mali-G720-Immortalis (ARM_PROPRIETARY))
GL_VERSION:     4.0 (Compatibility Profile) Mesa 24.0.4 (git-ca753f8cbd)
Surface Config: buf=32 r=8 g=8 b=8 a=8 depth=24 stencil=0 samples=0
Surface Size:   800x600 windowed
=====
[build] use-vbo=false: FPS: 1246 FrameTime: 0.803 ms
[build] use-vbo=true: FPS: 1378 FrameTime: 0.726 ms
[texture] texture-filter=nearest: FPS: 1323 FrameTime: 0.756 ms
[texture] texture-filter=linear: FPS: 1303 FrameTime: 0.768 ms
[texture] texture-filter=mipmap: FPS: 1376 FrameTime: 0.727 ms
[shading] shading=gouraud: FPS: 1254 FrameTime: 0.798 ms
[shading] shading=blinn-phong-inf: FPS: 1262 FrameTime: 0.793 ms
[shading] shading=phong: FPS: 1308 FrameTime: 0.765 ms
[shading] shading=cel: FPS: 1336 FrameTime: 0.749 ms
[bump] bump-render=high-poly: FPS: 1223 FrameTime: 0.818 ms
[bump] bump-render=normals: FPS: 1325 FrameTime: 0.755 ms
[bump] bump-render=height: FPS: 1248 FrameTime: 0.801 ms
[effect2d] kernel=0,1,0;1,-4,1;0,1,0,: FPS: 1376 FrameTime: 0.727 ms
[effect2d] kernel=1,1,1,1,1;1,1,1,1,1;1,1,1,1,1: FPS: 1134 FrameTime: 0.882 ms
[pulsar] light=false:quads=5:texture=false: FPS: 1322 FrameTime: 0.757 ms
[desktop] blur-radius=5:effect=blur:passes=1:separable=true:windows=4: FPS: 833 FrameTime: 1.201 ms
[desktop] effect=shadow:windows=4: FPS: 1061 FrameTime: 0.943 ms
[buffer] columns=200:interleave=false:update-dispersion=0.9:update-fraction=0.5:update-method=map: FPS: 826 FrameTime: 1.212 ms
[buffer] columns=200:interleave=false:update-dispersion=0.9:update-fraction=0.5:update-method=subdata: FPS: 827 FrameTime: 1.210 ms
[buffer] columns=200:interleave=true:update-dispersion=0.9:update-fraction=0.5:update-method=map: FPS: 941 FrameTime: 1.064 ms
[ideas] speed=duration: FPS: 1098 FrameTime: 0.911 ms
[jellyfish] <default>: FPS: 1288 FrameTime: 0.777 ms
[terrain] <default>: FPS: 301 FrameTime: 3.323 ms
[shadow] <default>: FPS: 1172 FrameTime: 0.853 ms
[refract] <default>: FPS: 565 FrameTime: 1.772 ms
[conditionals] fragment-steps=0:vertex-steps=0: FPS: 1391 FrameTime: 0.719 ms
[conditionals] fragment-steps=5:vertex-steps=0: FPS: 1266 FrameTime: 0.790 ms
[conditionals] fragment-steps=0:vertex-steps=5: FPS: 1305 FrameTime: 0.767 ms
[function] fragment-complexity=low:fragment-steps=5: FPS: 1231 FrameTime: 0.813 ms
[function] fragment-complexity=medium:fragment-steps=5: FPS: 1311 FrameTime: 0.763 ms
[loop] fragment-loop=false:fragment-steps=5:vertex-steps=5: FPS: 1244 FrameTime: 0.804 ms
[loop] fragment-steps=5:fragment-uniform=false:vertex-steps=5: FPS: 1258 FrameTime: 0.795 ms
[loop] fragment-steps=5:fragment-uniform=true:vertex-steps=5: FPS: 1241 FrameTime: 0.806 ms
=====
glmark2 Score: 1167
=====

```

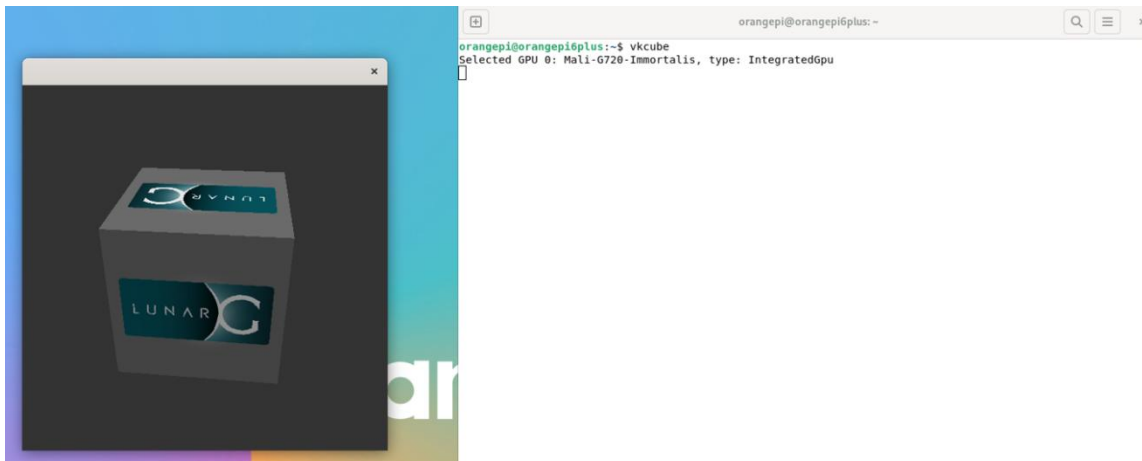
7) vkcube can be used to verify whether the Vulkan API installation, driver and basic functions are normal in the X11 environment. The specific test method is as follows:

a. Open a terminal on your **desktop** and run the following command.



```
orange@orange:~$ vkcube
```

- b. If you see the output of **Mali-G720-Immortalis** and a textured cube rotating smoothly and fluidly in the window without any error messages or crashes, the test was successful.

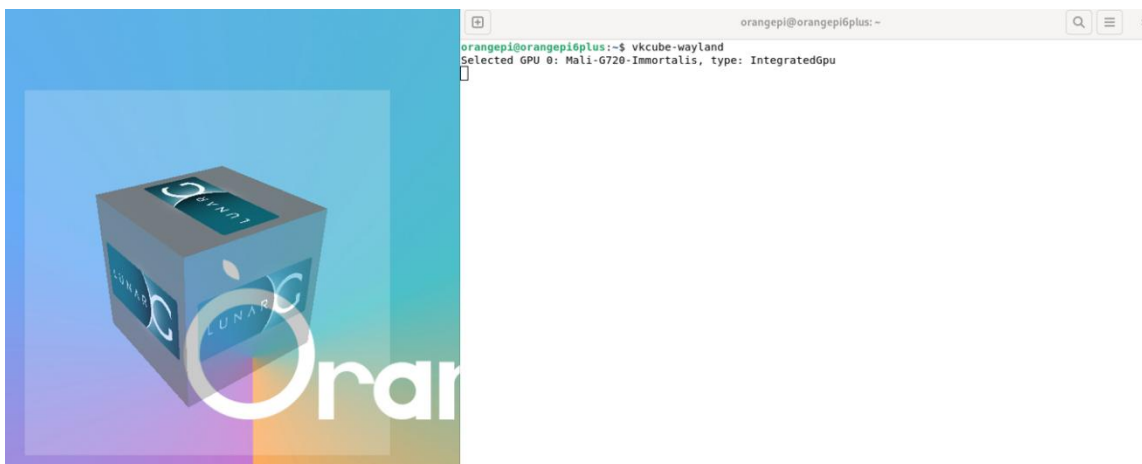


8) vkcube-wayland can be used to verify whether the installation, driver and basic functions of the Vulkan API in the Wayland environment are normal. The specific test method is as follows:

- a. Open a terminal on your **desktop** and run the following command.

```
orange@orange:~$ vkcube-wayland
```

- b. If you see the output of **Mali-G720-Immortalis** and a textured cube rotating smoothly and fluidly in the window without any error messages or crashes, the test was successful.



9) clpeak is a professional benchmark tool designed to measure the "peak" theoretical computing performance and memory bandwidth of OpenCL devices. It tests the general



computing capabilities of the GPU, not its graphics rendering capabilities. The test method is as follows:

- a. Run the following command:

```
orangepi@orangepi:~$ clpeak
```

- b. The test results are as follows. Ensure that the device is **Mali-G720-Immortalis**, indicating that the hardware GPU is being used.

```
orangepi@orangepi:~$ clpeak
```

Platform: ARM Platform

Device: **Mali-G720-Immortalis r0p0**

Driver version : 3.0 (Linux ARM64)

Compute units : 10

Clock frequency : 1000 MHz

Global memory bandwidth (GBPS)

float : 37.66

float2 : 38.64

float4 : 36.93

float8 : 30.33

float16 : 9.83

Single-precision compute (GFLOPS)

float : 1993.14

float2 : 2214.55

float4 : 2270.73

float8 : 2293.11

float16 : 2290.41

Half-precision compute (GFLOPS)

half : 2025.29

half2 : 3957.35

half4 : 4336.04

half8 : 4462.95

half16 : 4467.78

No double precision support! Skipped

**Integer compute (GIOPS)**

int : 317.78
int2 : 317.59
int4 : 317.78
int8 : 317.99
int16 : 317.19

Integer compute Fast 24bit (GIOPS)

int : 317.15
int2 : 317.48
int4 : 316.41
int8 : 317.42
int16 : 316.88

Transfer bandwidth (GBPS)

enqueueWriteBuffer : 8.59
enqueueReadBuffer : 14.26
enqueueWriteBuffer non-blocking : 8.66
enqueueReadBuffer non-blocking : 14.26
enqueueMapBuffer(for read) : 412977.62
memcpy from mapped ptr : 14.25
enqueueUnmap(after write) : 660764.25
memcpy to mapped ptr : 8.65

Kernel launch latency : 22.12 us

3. 22. Testing Methods for Gstreamer Video Hardware

Encoding

- 1) The supported video hardware encoding formats are as follows:
 - a. H.264 video hardware encoding
 - b. HEVC video hardware encoding
 - c. VP8 video hardware encoding
 - d. VP9 video hardware encoding



2) Prepare a video encoding test file. For example, you can decode the big_buck_bunny_1080p_h264.mov file to obtain the NV12 format YUV file big_buck_bunny_1080p.nv12.

a. Download the big_buck_bunny_1080p_h264.mo video.

```
orange@orange:~$ wget \
https://download.blender.org/peach/bigbuckbunny_movies/big_buck_bunny_1080p_h264.mov
```

b. Then obtain the first 60 seconds of the big_buck_bunny_1080p_h264.mov video file to save space.

```
orange@orange:~$ ffmpeg -i big_buck_bunny_1080p_h264.mov -t 60 -c copy \
output_first_1min.mov
```

c. Then decode the output_first_1min.mov file to obtain the NV12-formatted YUV file big_buck_bunny_1080p.nv12.

```
orange@orange:~$ gst-launch-1.0 filesrc \
location=output_first_1min.mov ! \
decodebin ! video/x-raw,format=NV12 ! filesink location=big_buck_bunny_1080p.nv12
```

d. The decoded YUV file is as follows:

```
orange@orange:~$ ls big_buck_bunny_1080p.nv12 -lh
-rw-r--r-- 1 orange orange 4.2G Sep  2 9:50 big_buck_bunny_1080p.nv12
```

3) The test method for H.264 video hardware encoding is as follows::

a. Execute the following command to test the H.264 encoding function.

```
orange@orange:~$ gst-launch-1.0 filesrc \
location=big_buck_bunny_1080p.nv12 ! videoparse width=1920 height=1080 \
framerate=30/1 format=i420 ! video/x-raw,colorimetry=bt709 ! v4l2h264enc \
capture-io-mode=mmap output-io-mode=dmabuf \
extra-controls="encode,fixed_qp=28" ! video/x-h264,profile=main,level=(string)5 ! \
filesink location=big_buck_bunny_1080p.h264
```

b. During the test, open another terminal and enter the following command. If you can see that VPU Utilization is in use and the **H.264 encoder** is called, the H.264 hardware encoding test is successful.

```
orange@orange:~$ sudo vpu_test.sh
VPU Utilization: 100.00%
10:34:20 12:34:18 ~ 12:34:20 [ffff0000beb28138] H.264 encoder 1920x1080 1058
frames, current fps 598.28, average fps 551.07
```



- c. The file generated after encoding is as follows:

```
orange@orange:~$ ls big_buck_bunny_1080p.h264 -lh
-rw-r--r-- 1 orange orange 57M Sep 2 10:23 big_buck_bunny_1080p.h264
```

- 4) The test method for HEVC video hardware encoding is as follows:

- a. Run the following command to test the HEVC encoding function.

```
orange@orange:~$ gst-launch-1.0 filesrc \
location=big_buck_bunny_1080p.nv12 ! \
videoparse width=1920 height=1080 framerate=30/1 format=i420 ! \
video/x-raw,colorimetry=bt709 ! \
v4l2h265enc capture-io-mode=mmap output-io-mode=dmabuf \
extra-controls="encode,fixed_qp=28" ! \
video/x-h265,profile=main,level=(string)5 ! \
filesink location=big_buck_bunny_1080p.hevc
```

- b. During the test, open another terminal and enter the following command. If you can see that VPU Utilization is in use and the **HEVC encoder** is called, the HEVC hardware encoding test is successful.

```
orange@orange:~$ sudo vpu_test.sh
VPU Utilization: 100.00%
10:33:00 12:32:58 ~ 12:33:00 [ffff00009c938138] HEVC encoder 1920x1080 1094
frames, current fps 543.69, average fps 508.90
```

- c. The file generated after encoding is as follows:

```
orange@orange:~$ ls big_buck_bunny_1080p.hevc -lh
-rw-r--r-- 1 orange orange 53M Sep 2 20:34 big_buck_bunny_1080p.hevc
```

- 5) The test method for VP8 video hardware encoding is as follows:

- a. Run the following command to test the VP8 encoding function.

```
orange@orange:~$ gst-launch-1.0 filesrc \
location=big_buck_bunny_1080p.nv12 ! \
videoparse width=1920 height=1080 framerate=30/1 format=i420 ! \
video/x-raw,colorimetry=bt709 ! v4l2vp8enc ! webmmux ! \
filesink location=big_buck_bunny_1080p.vp8
```

- b. During the test, open another terminal and enter the following command. If you can see that VPU Utilization is in use and the **VP8 encoder** is called, the HEVC hardware encoding test is successful.



```

orangepi@orangepi:~$ sudo vpu_test.sh
VPU Utilization: 94.41%
10:38:25 12:38:23 ~ 12:38:25 [ffff0001907dc138] VP8 encoder 1920x1080 1294 frames,
current fps 554.46, average fps 525.38

```

c. The file generated after encoding is as follows:

```

orangepi@orangepi:~$ ls big_buck_bunny_1080p.vp8 -lh
-rw-r--r-- 1 orangepi orangepi 197M Sep  2 10:38 big_buck_bunny_1080p.vp8

```

6) The test method for VP9 video hardware encoding is as follows:

a. Run the following command to test the VP9 encoding function.

```

orangepi@orangepi:~$ gst-launch-1.0 filesrc \
location=big_buck_bunny_1080p.nv12 ! \
videoparse width=1920 height=1080 framerate=30/1 format=i420 ! \
video/x-raw,colorimetry=bt709 ! v4l2vp9enc ! webmmux ! \
filesink location=big_buck_bunny_1080p.vp9

```

b. During the test, open another terminal and enter the following command. If you can see that VPU Utilization is in use and the **VP9 encoder** is called, the HEVC hardware encoding test is successful.

```

orangepi@orangepi:~$ sudo vpu_test.sh
VPU Utilization: 100.00%
20:40:08 12:40:06 ~ 12:40:08 [ffff0001b2ff4138] VP9 encoder 1920x1080 1417 frames,
current fps 582.47, average fps 526.19

```

c. The file generated after encoding is as follows:

```

orangepi@orangepi:~$ ls big_buck_bunny_1080p.vp9 -lh
-rw-r--r-- 1 orangepi orangepi 152M Sep  2 10:40 big_buck_bunny_1080p.vp9

```

3. 23. Testing Methods for Video Players and Gstreamer Video Hardware Decoding

1) The supported hardware decoding formats are as follows:

- a. Video hardware decoding - AV1
- b. Video hardware decoding - H.264
- c. Video hardware decoding - HEVC
- d. Video hardware decoding - VP9
- e. Video hardware decoding - VP8



- f. Video hardware decoding - MPEG-4
- g. Video hardware decoding - MPEG-2
- h. Video hardware decoding - H.263

2) The test video collection is test_videos.tar.gz, which can be downloaded from the **official tool** on the development board's [data download page](#). After downloading, upload it to the HOME directory of the Linux system and then decompress it.

```
orangepi@orangepi:~$ tar xzf test_videos.tar.gz
orangepi@orangepi:~$ ls test_videos
Big_Buck_Bunny_1080_10s_30MB_av1.mp4
Big_Buck_Bunny_h263.3gp
video_1080_10s_30MB_vp8.webm
video_1080p_30fps_MPEG-2.mpg
Big_Buck_Bunny_1080_10s_30MB_h264.mp4
video_1080_10s_30MB_h265.mp4
video_1080_10s_30MB_vp9.webm
video_1080p_30fps_MPEG-4.avi
```

3) Add the test video file to the Videos player.

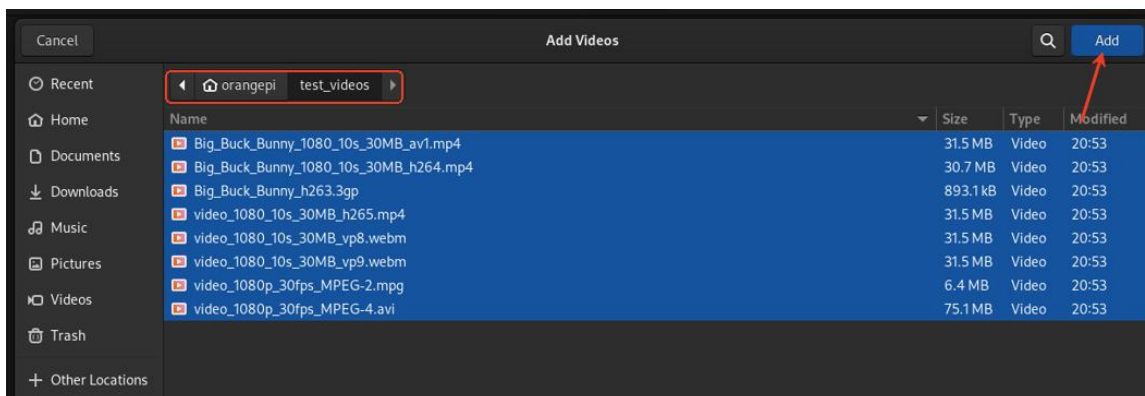
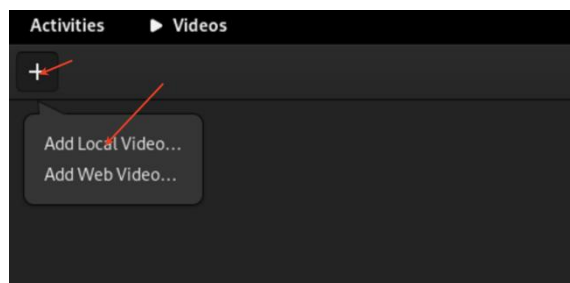
- a. Open the Videos player on the desktop.



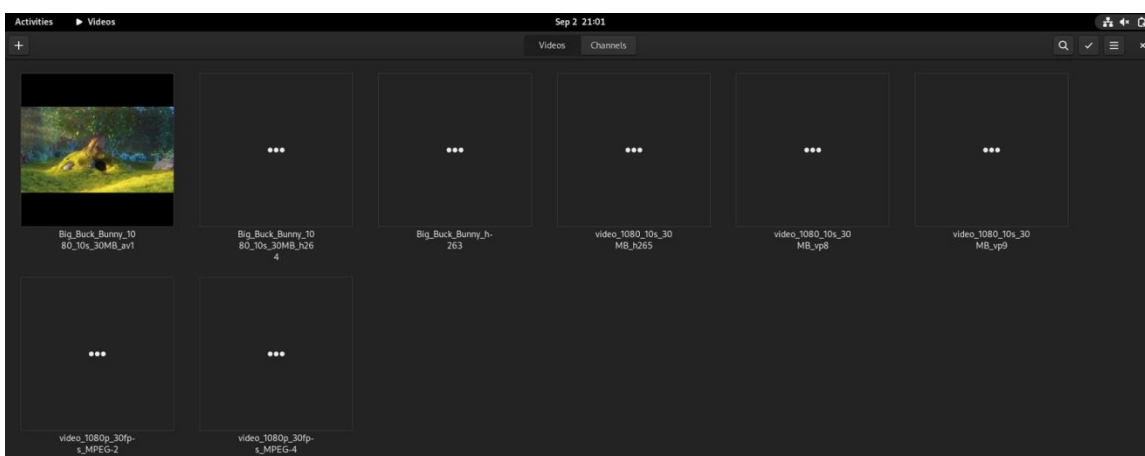
- b. Then click the + sign in the upper left corner of the Videos folder and use the **Add Local Video** button to add all the test video files in the test_videos folder to



the Videos folder.



c. The test video is displayed as follows after it is imported into the player:



4) The test methods for AV1, H.264, HEVC, VP9, VP8, MPEG-4, MPEG-2, and H.263 are similar. AV1 video is used as an example for the test method.

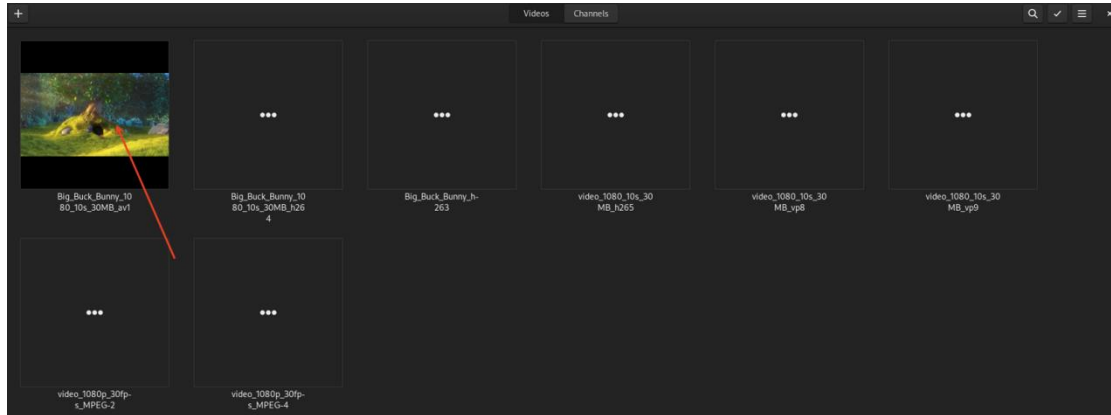
a. The test videos for AV1, H.264, HEVC, VP9, VP8, MPEG-4, MPEG-2, and H.263 are as follows:

Video Format	Corresponding test video file
AV1	Big_Buck_Bunny_1080_10s_30MB_av1.mp4
H.264	Big_Buck_Bunny_1080_10s_30MB_h264.mp4

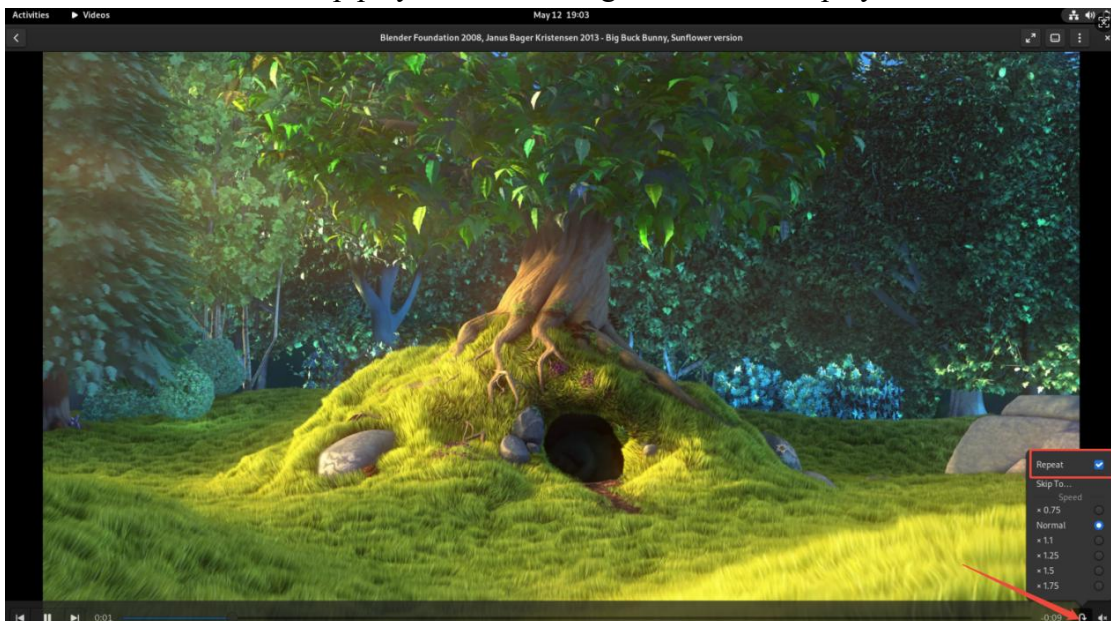


HEVC	video_1080_10s_30MB_h265.mp4
VP9	video_1080_10s_30MB_vp9.webm
VP8	video_1080_10s_30MB_vp8.webm
MPEG-4	video_1080p_30fps_MPEG-4.avi
MPEG-2	video_1080p_30fps_MPEG-2.mpg
H.263	Big_Buck_Bunny_h263.3gp

b. Double-click to open the AV1 video file.



c. You can select loop play in the lower right corner of the player.



d. Then open another terminal and enter the following command. If you can see VPU Utilization is in use and the **AV1 decoder** is called, AV1 hardware decoding is working properly.

```
orange@orange:~$ sudo vpu_test.sh
```

```
VPU Utilization: 7.47%
```



```
11:14:04 13:14:00 ~ 13:14:04 [ffff0000f1ec0138] AV1 decoder 1920x1080 334 frames,
current fps 29.76, average fps 30.19
```

5) The commands for testing hardware-decoded video using Gstreamer are as follows. The test methods for AV1, H.264, HEVC, VP9, VP8, MPEG-4, MPEG-2, and H.263 are similar. AV1 video is used as an example here.

Please perform the following test under the orangepi user, do not switch to the root user.

- a. Use playbin to play audio and video files directly. The specific commands are as follows:

```
orangepi@orangepi:~$ gst-launch-1.0 playbin \
uri=file:///home/orangepi/test_videos/Big_Buck_Bunny_1080_10s_30MB_av1.mp4
```

- b. Use playbin and specify the video sink link (displayed via gtkglsink)

```
orangepi@orangepi:~$ gst-launch-1.0 playbin \
uri=file:///home/orangepi/test_videos/Big_Buck_Bunny_1080_10s_30MB_av1.mp4 \
video-sink="glupload ! glcolorconvert ! glcolorbalance ! gtkglsink"
```

- c. Use decodebin to decode the video and display it.

```
orangepi@orangepi:~$ gst-launch-1.0 filesrc \
location=/home/orangepi/test_videos/Big_Buck_Bunny_1080_10s_30MB_av1.mp4 ! \
decodebin ! glupload ! glcolorconvert ! glcolorbalance ! gtkglsink
```

- d. Use decodebin to only decode the video and output it in NV12 format without displaying it.

```
orangepi@orangepi:~$ gst-launch-1.0 filesrc \
location=/home/orangepi/test_videos/Big_Buck_Bunny_1080_10s_30MB_av1.mp4 ! decodebin ! \
video/x-raw,format=NV12 ! fakesink sync=true
```

- e. When testing, open another terminal and enter the following command. If you can see VPU Utilization in use and the **AV1 decoder** is called, it means that AV1 hardware decoding is working properly.

```
orangepi@orangepi:~$ sudo vpu_test.sh
VPU Utilization: 7.47%
11:14:04 13:14:00 ~ 13:14:04 [ffff0000f1ec0138] AV1 decoder 1920x1080 334 frames,
current fps 29.76, average fps 30.19
```



3. 24. Testing Methods for FFmpeg Video Hardware Encoding

1) The supported video hardware encoding formats are as follows:

- a. H.264 video hardware encoding
- b. HEVC video hardware encoding
- c. VP8 video hardware encoding
- d. MJPEG video hardware encoding

2) Prepare a video encoding test file. For example, you can decode the file `big_buck_bunny_1080p_h264.mov` to obtain an NV12 format YUV file `big_buck_bunny_1080p.nv12`.

- a. Download the video `big_buck_bunny_1080p_h264.mo`.

```
orangepi@orangepi:~$ wget \
https://download.blender.org/peach/bigbuckbunny_movies/big_buck_bunny_1080p_h264.mov
```

- b. Then obtain the first 60 seconds of the video file `big_buck_bunny_1080p_h264.mov` to save space.

```
orangepi@orangepi:~$ ffmpeg -i big_buck_bunny_1080p_h264.mov -t 60 -c copy \
output_first_1min.mov
```

- c. Then decode the `output_first_1min.mov` file to obtain the YUV file `big_buck_bunny_1080p.yuv` in NV12 format.

```
orangepi@orangepi:~$ ffmpeg -c:v h264_v4l2m2m -i output_first_1min.mov -pix_fmt nv12 \
big_buck_bunny_1080p.yuv
```

- d. The decoded YUV file is shown below.:

```
orangepi@orangepi:~$ ls big_buck_bunny_1080p.yuv -lh
-rw-r--r-- 1 orangepi orangepi 4.2G Sep 28 17:56 big_buck_bunny_1080p.yuv
```

3) The test method for H.264 video hardware encoding is as follows:

- a. Execute the following command to test the H.264 encoding function.

```
orangepi@orangepi:~$ ffmpeg -f rawvideo -pix_fmt nv12 -r 60 -video_size \
1920x1080 -i big_buck_bunny_1080p.yuv -c:v h264_v4l2m2m bbb_1080p_h264.mp4
```

- b. During the test, you can open another terminal and enter the following command.
If you can see that VPU Utilization is in use and **H.264 encoder** is being called, it means that the H.264 hardware encoding test is successful.

```
orangepi@orangepi:~$ sudo vpu_test.sh
VPU Utilization: 97.45%
```



```
17:55:41 09:55:38 ~ 09:55:41 [ffff5ff732138138] P2 H.264 encoder 1920x1080 1262
frames, current fps 534.04, average fps 500.48
```

- c. The file generated after encoding is as follows:

```
orangepi@orangepi:~$ ls bbb_1080p_h264.mp4 -lh
-rw-r--r-- 1 orangepi orangepi 1.3M Oct 23 17:57 bbb_1080p_h264.mp4
```

- 4) The test method for HEVC video hardware encoding is as follows:

- a. Run the following command to test the HEVC encoding function.

```
orangepi@orangepi:~$ ffmpeg -f rawvideo -pix_fmt nv12 -r 60 -video_size \
1920x1080 -i big_buck_bunny_1080p.yuv -c:v hevc_v4l2m2m bbb_1080p_hevc.mp4
```

- b. During the test, you can open another terminal and enter the following command.
If you can see that VPU Utilization is in use and **HEVC encoder** is being called,
it means that the HEVC hardware encoding test is successful.

```
orangepi@orangepi:~$ sudo vpu_test.sh
VPU Utilization: 100.00%
17:59:45 09:59:43 ~ 09:59:45 [ffff5ff5e86a4138] P2 HEVC encoder 1920x1080 1398
frames, current fps 508.94, average fps 485.86
```

- c. The generated file after encoding is shown below:

```
orangepi@orangepi:~$ ls bbb_1080p_hevc.mp4 -lh
-rw-r--r-- 1 orangepi orangepi 646K Oct 23 17:59 bbb_1080p_hevc.mp4
```

- 5) The test method for VP8 video hardware encoding is as follows:

- a. Execute the following command to test the VP8 encoding function.

```
orangepi@orangepi:~$ ffmpeg -f rawvideo -pix_fmt nv12 -r 60 -video_size \
1920x1080 -i big_buck_bunny_1080p.yuv -c:v vp8_v4l2m2m \
bbb_1080p_vp8.webm
```

- b. During the test, you can open another terminal and enter the following command.
If you can see that VPU Utilization is in use and **VP8 encoder** is being called, it
means that the VP8 hardware encoding test is successful.

```
orangepi@orangepi:~$ sudo vpu_test.sh
VPU Utilization: 94.41%
18:07:31 10:07:28 ~ 10:07:31 [ffff5ff60f6dc138] P2 VP8 encoder 1920x1080 1414
frames, current fps 475.36, average fps 475.47
```

- c. The generated file after encoding is shown below:

```
orangepi@orangepi:~$ ls bbb_1080p_vp8.webm -lh
```



```
-rw-r--r-- 1 orangepi orangepi 5.8M Oct 23 18:07 bbb_1080p_vp8.webm
```

6) The test method for MJPEG video hardware encoding is as follows:

a. Execute the following command to test the MJPEG encoding function.

```
orangepi@orangepi:~$ ffmpeg -f rawvideo -pix_fmt nv12 -r 60 -video_size \
1920x1080 -i big_buck_bunny_1080p.yuv -c:v mjpeg_v4l2m2m \
bbb_1080p_mjpeg.avi
```

b. During the test, you can open another terminal and enter the following command.
If you can see that VPU Utilization is in use and **MJPEG encoder** is being called, it means that the MJPEG hardware encoding test is successful.

```
orangepi@orangepi:~$ sudo vpu_test.sh
VPU Utilization: 48.70%
18:13:35 10:13:34 ~ 10:13:35 [ffff5ff72cc4c138] P2 MJPEG encoder 1920x1080 1280
frames, current fps 826.51, average fps 786.07
```

c. The generated file after encoding is shown below:

```
orangepi@orangepi:~$ ls bbb_1080p_mjpeg.avi -lh
-rw-r--r-- 1 orangepi orangepi 302M Oct 23 18:13 bbb_1080p_mjpeg.avi
```

3. 25. Testing Methods for Ffmpeg Video Hardware Decoding

1) The supported hardware decoding formats are as follows:

- a. Video hardware decoding - AV1
- b. Video hardware decoding - H.263
- c. Video hardware decoding - H.264
- d. Video hardware decoding - HEVC
- e. Video hardware decoding - MPEG-2
- f. Video hardware decoding - MPEG-4
- g. Video hardware decoding - VP8
- h. Video hardware decoding - VP9

2) The test video collection is named test_videos.tar.gz, which can be downloaded from the **official tool** on the development board's [download page](#). After downloading, upload it to the HOME directory of the Linux system and then decompress it.

```
orangepi@orangepi:~$ tar xzf test_videos.tar.gz
orangepi@orangepi:~$ ls test_videos
```




```
Big_Buck_Bunny_1080_10s_30MB_av1.mp4
Big_Buck_Bunny_h263.3gp
video_1080_10s_30MB_vp8.webm
video_1080p_30fps_MPEG-2.mpg
Big_Buck_Bunny_1080_10s_30MB_h264.mp4
video_1080_10s_30MB_h265.mp4
video_1080_10s_30MB_vp9.webm
video_1080p_30fps_MPEG-4.avi
```

3) The test videos corresponding to AV1, H.264, HEVC, and VP9 are as follows:

Video Format	Corresponding test video file
AV1	Big_Buck_Bunny_1080_10s_30MB_av1.mp4
H.264	Big_Buck_Bunny_1080_10s_30MB_h264.mp4
HEVC	video_1080_10s_30MB_h265.mp4
VP9	video_1080_10s_30MB_vp9.webm
VP8	video_1080_10s_30MB_vp8.webm
MPEG-4	video_1080p_30fps_MPEG-4.avi
MPEG-2	video_1080p_30fps_MPEG-2.mpg
H.263	Big_Buck_Bunny_h263.3gp

4) The FFmpeg method for testing AV1 hard decoding video is as follows:

a. Enter the test video directory

```
orange@orange:~$ cd test_videos
```

b. Use the following command to play AV1 format video.

```
orange@orange:~/test_videos$ ffmpeg -vcodec av1_v4l2m2m \
Big_Buck_Bunny_1080_10s_30MB_av1.mp4
```

c. When testing, open another terminal and enter the following command. If you can see VPU Utilization is in use and the **AV1 decoder** is called, it means that AV1 hardware decoding is working properly.

```
orange@orange:~$ sudo vpu_test.sh
VPU Utilization: 27.02%
17:45:13 09:45:11 ~ 09:45:12 [ffff000142d88138] AV1 decoder 1920x1080 271 frames,
current fps 29.71, average fps 51.35
```

5) The method for testing H264 hardware decoding video using FFmpeg is as follows:



- a. Navigate to the directory containing the test videos.

```
orangepi@orangepi:~$ cd test_videos
```

- b. Use the following command to play H264 format video.

```
orangepi@orangepi:~/test_videos$ ffplay -vcodec h264_v4l2m2m \
Big_Buck_Bunny_1080_10s_30MB_h264.mp4
```

- c. During testing, open another terminal and enter the following command. If you can see VPU Utilization is in use and the **H264 decoder** is called, it means that the H264 hardware decoding is working properly.

```
orangepi@orangepi:~$ sudo vpu_test.sh
VPU Utilization: 27.02%
17:38:33 09:38:29 ~ 09:38:32 [ffff000157ac0138] H.264 decoder 1920x1080 300 frames,
current fps 29.41, average fps 30.19
```

- 6) The FFmpeg method for testing HEVC hard decoding video is as follows:

- a. Enter the directory of the test video.

```
orangepi@orangepi:~$ cd test_videos
```

- b. Use the following command to play HEVC format video.

```
orangepi@orangepi:~/test_videos$ ffplay -vcodec hevc_v4l2m2m \
video_1080_10s_30MB_h265.mp4
```

- c. During testing, open another terminal and enter the following command. If you can see that VPU Utilization is in use and the **HEVC decoder** is called, it means that HEVC hardware decoding is working properly.

```
orangepi@orangepi:~$ sudo vpu_test.sh
VPU Utilization: 14.30%
17:43:19 09:43:19 ~ 09:43:19 [ffff000159e20138] HEVC decoder 1920x1080 201
frames, current fps 25.43, average fps 339.90
```

- 7) The FFmpeg method for testing VP9 hard decoding video is as follows:

- a. Enter the directory of the test video.

```
orangepi@orangepi:~$ cd test_videos
```

- b. Use the following command to play the VP9 format video.

```
orangepi@orangepi:~/test_videos$ ffplay -vcodec vp9_v4l2m2m \
video_1080_10s_30MB_vp9.webm
```

- c. During testing, open another terminal and enter the following command. If you can see VPU Utilization is in use and the **VP9 decoder** is called, it means that



VP9 hardware decoding is working properly.

```
orange@orange:~$ sudo vpu_test.sh
VPU Utilization: 11.23%
17:47:38 09:47:38 ~ 09:47:38 [ffff00011d250138] VP9 decoder 1920x1080 213 frames,
current fps 22.37, average fps 288.94
```

8) The FFmpeg method for testing VP8 hard decoding video is as follows:

- a. Enter the directory of the test video.

```
orange@orange:~$ cd test_videos
```

- b. Use the following command to play the VP8 format video.

```
orange@orange:~/test_videos$ ffmpeg -vcodec vp8_v4l2m2m \
video_1080_10s_30MB_vp8.webm
```

- c. During testing, open another terminal and enter the following command. If you can see VPU Utilization is in use and the **VP8 decoder** is called, it means that VP8 hardware decoding is working properly.

```
orange@orange:~$ sudo vpu_test.sh
VPU Utilization: 11.23%
17:47:38 09:47:38 ~ 09:47:38 [ffff00011d250138] VP8 decoder 1920x1080 213 frames,
current fps 22.37, average fps 288.94
```

9) The FFmpeg method for testing MPEG-2 hard decoding video is as follows:

- a. Enter the directory of the test video.

```
orange@orange:~$ cd test_videos
```

- b. Use the following command to play the video in MPEG-2 format.

```
orange@orange:~/test_videos$ ffmpeg -vcodec mpeg2_v4l2m2m \
video_1080p_30fps_MPEG-2.mpg
```

- c. During testing, open another terminal and enter the following command. If you can see VPU Utilization is in use and the **MPEG-2 decoder** is called, it means that the MPEG-2 hardware decoding is working properly.

```
orange@orange:~$ sudo vpu_test.sh
VPU Utilization: 14.50%
11:57:41 03:57:39 ~ 03:57:41 [ffff0000e17b8138] MPEG-2 ES decoder 1920x1080 252
frames, current fps 29.55, average fps 45.19
```

10) The FFmpeg method for testing MPEG-4 hard decoding video is as follows:



- a. Enter the directory of the test video.

```
orangepi@orangepi:~$ cd test_videos
```

- b. Use the following command to play the video in MPEG-4 format.

```
orangepi@orangepi:~/test_videos$ ffplay -vcodec mpeg4_v4l2m2m \
video_1080p_30fps_MPEG-4.avi
```

- c. During testing, open another terminal and enter the following command. If you can see VPU Utilization is in use and the **MPEG-4 decoder** is called, it means that the MPEG-4 hardware decoding is working properly.

```
orangepi@orangepi:~$ sudo vpu_test.sh
VPU Utilization: 100.00%
11:56:19 03:56:18 ~ 03:56:19 [ffff00018a640138] MPEG-4 part 2 ES decoder
1920x1080 242 frames, current fps 35.98, average fps 36.26
```

- 11) The FFmpeg method for testing H.263 hard decoding video is as follows:

- a. Enter the directory of the test video.

```
orangepi@orangepi:~$ cd test_videos
```

- b. Use the following command to play H.263 format video.

```
orangepi@orangepi:~/test_videos$ ffplay -vcodec h263_v4l2m2m \
Big_Buck_Bunny_h263.3gp
```

- c. During testing, open another terminal and enter the following command. If you can see VPU Utilization is in use and the **H.263 decoder** is called, it means that the H.263 hardware decoding is working properly.

```
orangepi@orangepi:~$ sudo vpu_test.sh
VPU Utilization: 5.02%
11:53:24 03:53:23 ~ 03:53:24 [ffff0001c6ce0138] H.263 decoder 352x288 231 frames,
current fps 27.20, average fps 32.09
```

3. 26. Test Method for MPV Video Hardware Decoding

- 1) If you are using a Linux 6.1 kernel system, you need to install the MPV player. The Linux 6.6 kernel system has it pre-installed by default and does not need to be installed manually. The steps are as follows:

```
orangepi@orangepi:~$ sudo apt-get update
orangepi@orangepi:~$ sudo apt-get install -y mpv
```



2) The hardware decoding formats supported by the MPV player are as follows:

- a. Video hardware decoding - AV1
- b. Video hardware decoding - H.264
- c. Video hardware decoding - HEVC
- d. Video hardware decoding - VP8
- e. Video hardware decoding - VP9

3) The test video collection is test_videos.tar.gz, which can be downloaded from the **official tools** section of the development board's [documentation download page](http://www.orangepi.cn/html/hardWare/computerAndMicrocontrollers/service-and-support/Orange-Pi-6-Plus.html):

<http://www.orangepi.cn/html/hardWare/computerAndMicrocontrollers/service-and-support/Orange-Pi-6-Plus.html>



4) After downloading, please upload **test_videos.tar.gz** to the HOME directory of your Linux system and then extract it.

```
orangepi@orangepi:~$ tar xzf test_videos.tar.gz
orangepi@orangepi:~$ ls test_videos
Big_Buck_Bunny_1080_10s_30MB_av1.mp4
Big_Buck_Bunny_h263.3gp
video_1080_10s_30MB_vp8.webm
video_1080p_30fps_MPEG-2.mpg
Big_Buck_Bunny_1080_10s_30MB_h264.mp4
video_1080_10s_30MB_h265.mp4
video_1080_10s_30MB_vp9.webm
video_1080p_30fps_MPEG-4.avi
```

5) The test videos for AV1, H.264, HEVC, and VP9 are shown below.:

Video format	Corresponding test video file
AV1	Big_Buck_Bunny_1080_10s_30MB_av1.mp4
H.264	Big_Buck_Bunny_1080_10s_30MB_h264.mp4



HEVC	video_1080_10s_30MB_h265.mp4
VP9	video_1080_10s_30MB_vp9.webm
VP8	video_1080_10s_30MB_vp8.webm

6) The MPV method for testing AV1 hardware decoding video is as follows. **Note that the following commands must be executed in a terminal on the Linux system desktop.**

- a. Navigate to the directory containing the test videos.

```
orange@orange:~$ cd test_videos
```

- b. Use the following command to play AV1 format video.

```
orange@orange:~/test_videos$ mpv --hwdec=yes \
Big_Buck_Bunny_1080_10s_30MB_av1.mp4
```

- c. When testing, open another terminal and enter the following command. If you can see VPU Utilization is in use and the **AV1 decoder** is called, it means that AV1 hardware decoding is working properly.

```
orange@orange:~$ sudo vpu_test.sh
VPU Utilization: 14.80%
17:23:46 09:23:46 ~ 09:23:46 [ffff5ff61c2d4138] P2 AV1 decoder 1920x1080 202
frames, current fps 28.49, average fps 31.09
VPU Utilization: 14.80%
```

7) The MPV method for testing H264 hard decoding video is as follows:

- a. Navigate to the directory containing the test videos.

```
orange@orange:~$ cd test_videos
```

- b. Use the following command to play an H264 format video.

```
orange@orange:~/test_videos$ mpv --hwdec=yes \
Big_Buck_Bunny_1080_10s_30MB_h264.mp4
```

- c. During testing, you can open another terminal and enter the following command. If you can see that VPU Utilization is in use and **H264 decoder** is being called, it means that H264 hardware decoding is working properly.

```
orange@orange:~$ sudo vpu_test.sh
VPU Utilization: 33.85%
17:37:25 09:37:22 ~ 09:37:25 [ffff5ff70e218138] P2 H.264 decoder 1920x1080 300
frames, current fps 32.57, average fps 30.86
```



8) The method for testing HEVC hardware decoding video using MPV is as follows::

- a. Navigate to the directory containing the test videos.

```
orangePi@orangePi:~$ cd test_videos
```

- b. Use the following command to play HEVC format video.

```
orangePi@orangePi:~/test_videos$ mpv --hwdec=yes \
video_1080_10s_30MB_h265.mp4
```

- c. During testing, you can open another terminal and enter the following command.
If you can see that VPU Utilization is in use and is calling the **HEVC decoder**, it means that the HEVC hardware decoding is working properly.

```
orangePi@orangePi:~$ sudo vpu_test.sh
VPU Utilization: 21.71%
17:37:33 09:37:32 ~ 09:37:33 [ffff5ff70dd50138] P2 HEVC decoder 1920x1080 215
frames, current fps 30.39, average fps 32.01
```

9) The method for testing VP9 hardware decoding video using MPV is as follows:

- a. Navigate to the directory containing the test videos.

```
orangePi@orangePi:~$ cd test_videos
```

- b. Use the following command to play VP9 format video.

```
orangePi@orangePi:~/test_videos$ mpv --hwdec=yes video_1080_10s_30MB_vp9.webm
```

- c. During testing, you can open another terminal and enter the following command.
If you can see that VPU Utilization is in use and is calling **VP9 decoder**, it means that VP9 hardware decoding can be used normally.

```
orangePi@orangePi:~$ sudo vpu_test.sh
VPU Utilization: 12.00%
17:37:43 09:37:43 ~ 09:37:43 [ffff5ff70d284138] P2 VP9 decoder 1920x1080 213
frames, current fps 30.92, average fps 32.36
```

10) The method for testing VP8 hardware decoding video using MPV is as follows:

- a. Navigate to the directory containing the test videos.

```
orangePi@orangePi:~$ cd test_videos
```

- b. Use the following command to play VP8 format videos.

```
orangePi@orangePi:~/test_videos$ mpv --hwdec=yes video_1080_10s_30MB_vp8.webm
```

- c. During testing, you can open another terminal and enter the following command.
If you can see that VPU Utilization is in use and is calling **VP8 decoder**, it



means that VP8 hardware decoding can be used normally.

```
orangepi@orangepi:~$ sudo vpu_test.sh
VPU Utilization: 11.23%
17:37:53 09:37:53 ~ 09:37:53 [ffff5ff637b08138] P2 VP8 decoder 1920x1080 210
frames, current fps 28.87, average fps 29.65
```

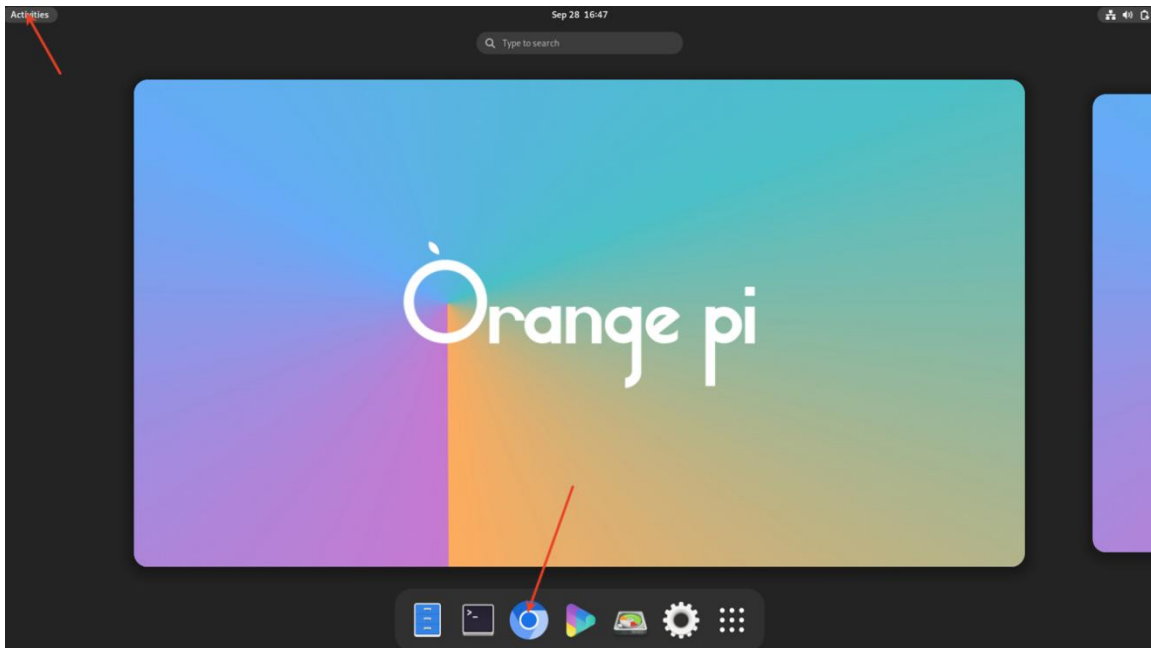
11) After adding the **hwdec=yes** configuration to **~/.config/mpv/mpv.conf**, playing videos using the command line without the **--hwdec=yes** parameter will also use hardware decoding by default. Additionally, playing videos directly using the MPV player on the desktop will also use hardware decoding by default. The specific commands are as follows:

```
orangepi@orangepi:~$ echo "hwdec=yes" > ~/.config/mpv/mpv.conf
```

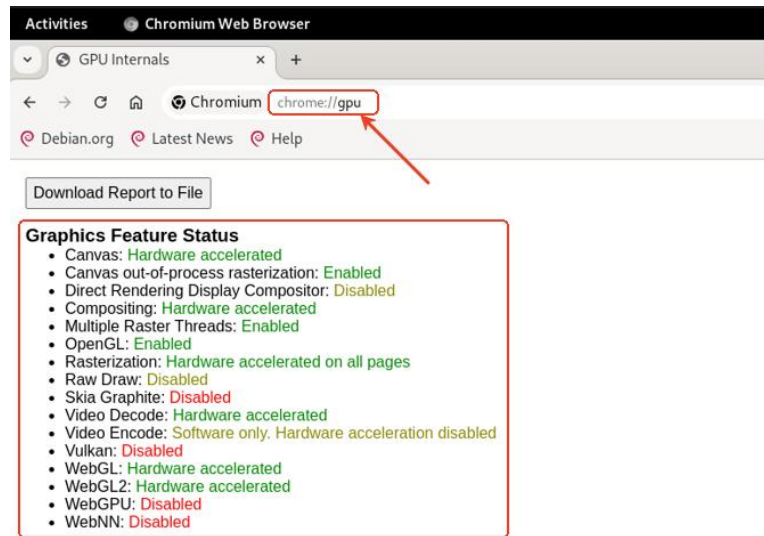
3. 27. Testing Methods for Chromium Browser

Hardware-Decoded Video Playback

1) Open the Chromium browser.

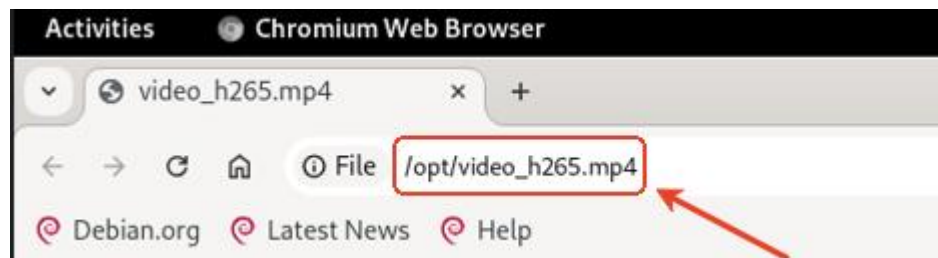


2) Then enter **chrome://gpu** in the Chromium browser to check the support of GPU and video decoding.



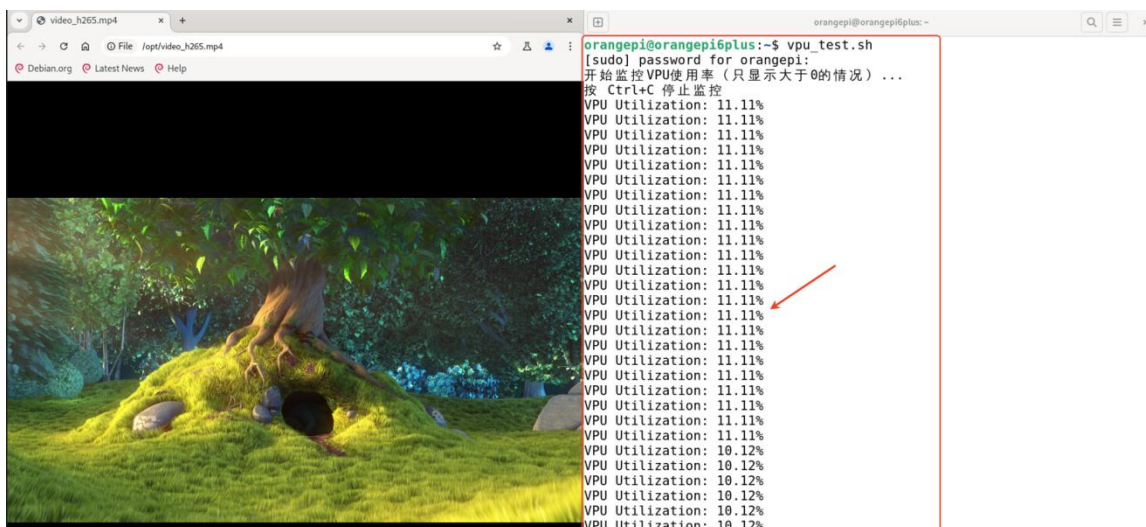
3) Then you can open the video website in the browser to play a video file, or enter the following path name in the browser to play a test video file that comes with the system.

`/opt/video_h265.mp4`



4) When playing a video, you can run the **vpu_test.sh** script in the terminal. If you see the printout on the right side of the figure below, it means that the hardware is being used to decode the video.

`orange@orange:~$ vpu_test.sh`



3. 28. How to set up the Chinese environment and install the Chinese input method

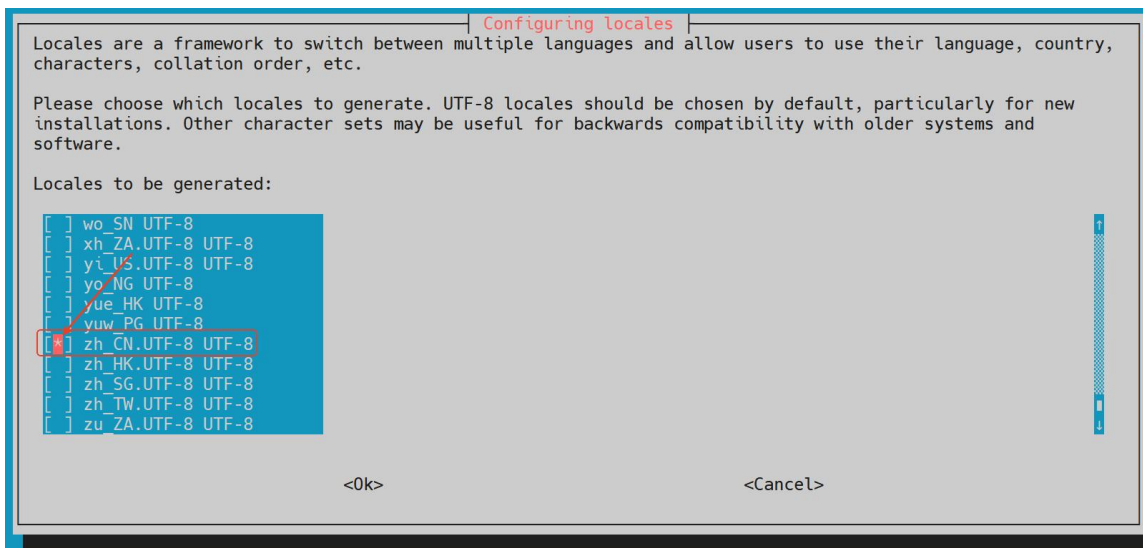
1) Install the following packages.

```
orangePi@orangePi:~$ sudo apt install -y locales fcitx5 fcitx5-rime fcitx5-config-qt  
orangePi@orangePi:~$ sudo apt install -y fcitx5-chinese-addons
```

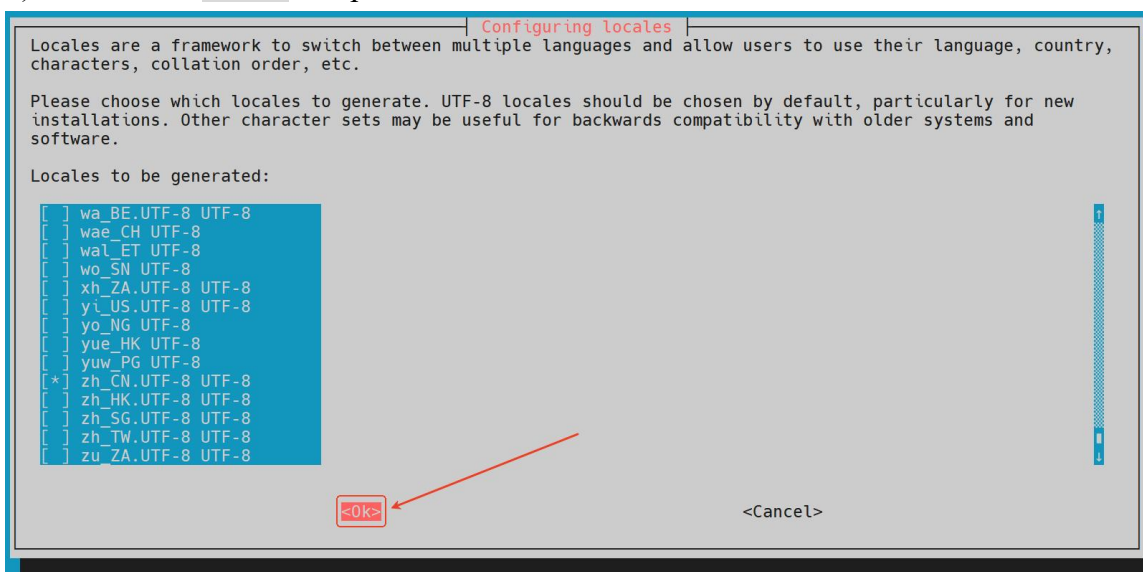
2) Run the following command to generate Chinese locales.

```
orangePi@orangePi:~$ sudo dpkg-reconfigure locales
```

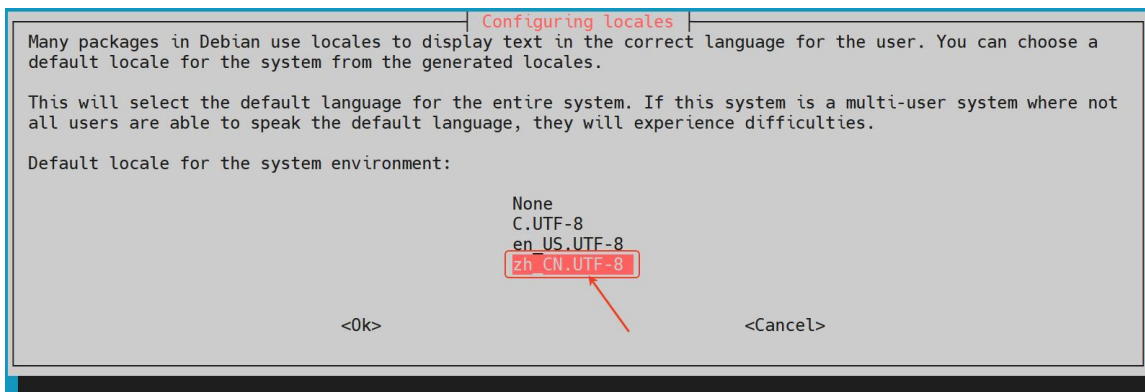
3) Use the down arrow key to navigate to the bottom, then use the space bar to select **zh_CN.UTF-8 UTF-8**



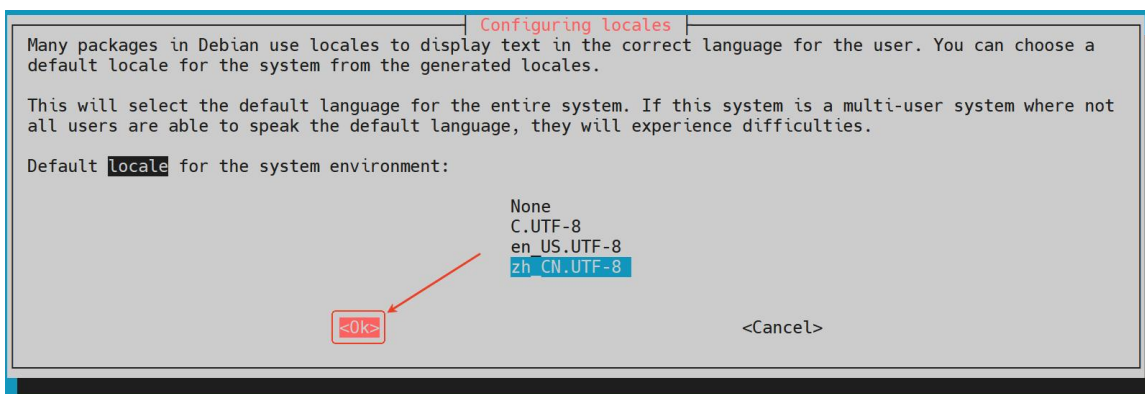
4) Then select **<OK>** and press Enter to confirm.



5) Then select **zh_CN.UTF-8 UTF-8** as the default locale environment.



6) Select OK and press Enter.



7) Then set the system to Chinese.

```
orange@orange:~$ sudo update-locale LANG=zh_CN.UTF-8 \
LANGUAGE=zh_CN:zh LC_ALL=zh_CN.UTF-8 LC_MESSAGES=zh_CN.UTF-8
```

8) Then set fcitx5 to start automatically at boot. Please execute the following command under the **orange** user.

```
orange@orange:~$ mkdir -p ~/.config/autostart
orange@orange:~$ vim ~/.config/autostart/fcitx5.desktop
[Desktop Entry]
Type=Application
Name=Fcitx5
Comment=Fcitx5 Input Method
Exec=fcitx5
TryExec=fcitx5
Icon=fcitx5
```



```
Categories=System;  
StartupNotify=false  
X-GNOME-Autostart-Delay=2  
orangepi@orangepi:~$ chmod +x ~/.config/autostart/fcitx5.desktop
```

9) Then restart the system to make the configuration take effect.

```
orangepi@orangepi:~$ sudo reboot
```

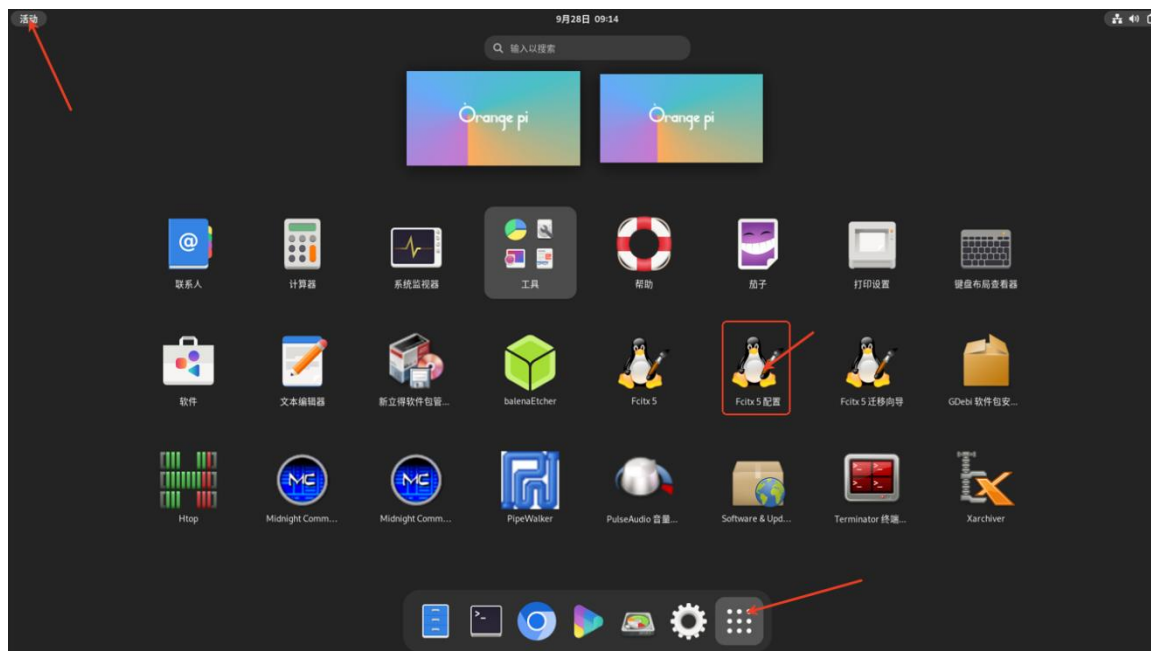
10) After re-entering the system, please select "**Do not ask me again**" in the interface below, and then decide whether to update the standard folder to Chinese according to your preference. **It is recommended to keep the old name.**



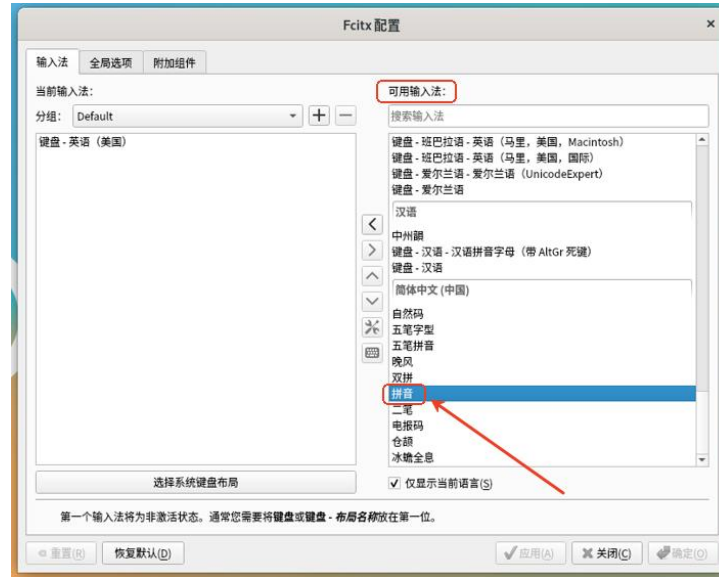
11) Then you can see that the desktop is displayed in Chinese.



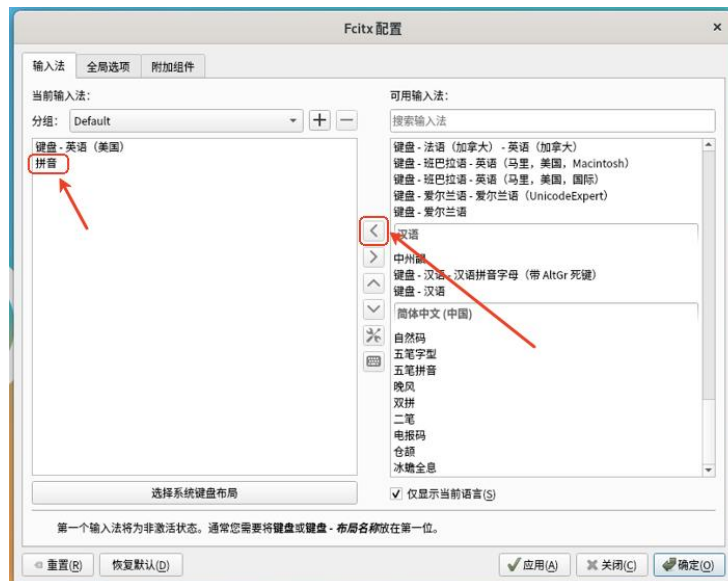
12) Then open the **Fcitx5 configuration** program.



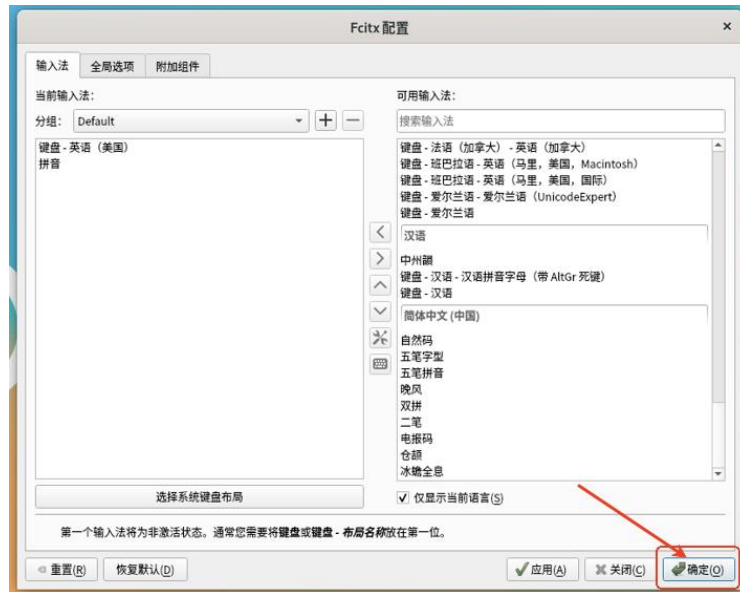
13) Then find **Pinyin** input method **among the available input methods** and select it.



14) Then click the left arrow key in the middle to add the Pinyin input method to the current input method.



15) Then click OK.



16) Then you can open a **text editor** to test the Chinese input method. The opening method is as shown below:



17) After opening the text editor, the default input method is still English. We can switch to the Chinese input method using the **Ctrl+Space** shortcut key, and then we can enter Chinese.



3. 29. How to install the Baota Linux Panel

Baota Linux Panel is a server management software that improves operation and maintenance efficiency. It supports more than 100 server management functions such as LAMP/LNMP/cluster/monitoring/website/FTP/database/JAVA, etc. (Excerpt from [Baota official website](#))

1) Then enter the following command in the Linux system to start the installation of the Baota.

```
orangeipi@orangeipi:~$ sudo install_bt_panel.sh
```

2) The Baota installation program will then prompt you whether to install **Bt-Panel** to the **/www** folder. Just enter y.

```
+-----+
| Bt-WebPanel FOR CentOS/Ubuntu/Debian
+-----+
| Copyright © 2015-2099 BT-SOFT(http://www.bt.cn) All rights reserved.
+-----+
| The WebPanel URL will be http://SERVER_IP:8888 when installed.
+-----+
Do you want to install Bt-Panel to the /www directory now?(y/n): y
```

3) Then all you have to do is wait patiently. When you see the following printed information output by the terminal, it means that the Baota has been installed. The entire installation process takes about a few minutes, which may vary depending on the network speed.



```

=====
Congratulations! Installed successfully!
=====注意：首次打开面板浏览器将提示不安全=====

请选择以下其中一种方式解决不安全提醒
1、下载证书，地址：https://dg2.bt.cn/ssl/baota\_root.pfx，双击安装，密码【www.bt.cn】
2、点击【高级】-【继续访问】或【接受风险并继续】访问
教程：https://www.bt.cn/bbs/thread-117246-1-1.html
mac用户请下载使用此证书：https://dg2.bt.cn/ssl/mac.crt

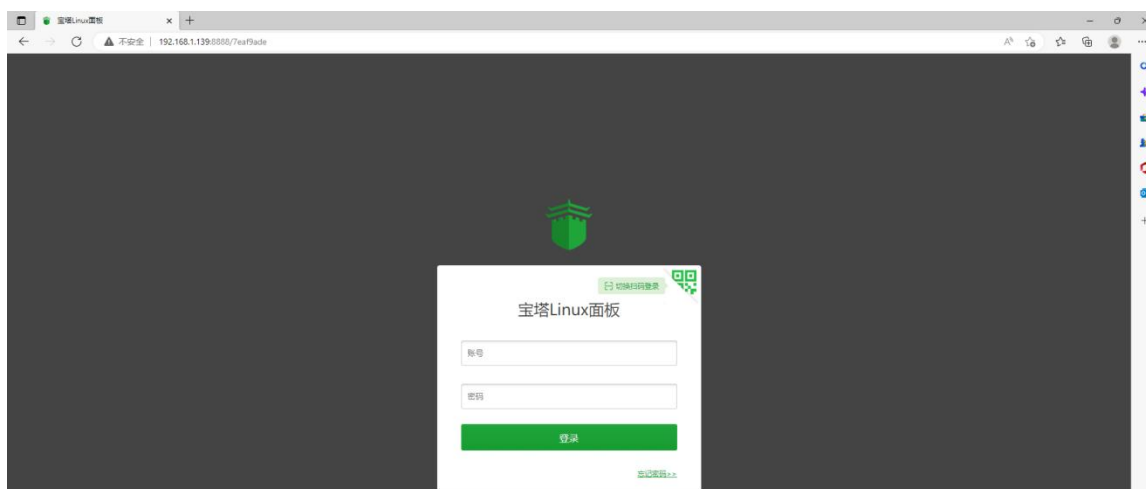
=====面板账户登录信息=====

【云服务器】请在安全组放行 24396 端口
外网面板地址：https://183.17.127.97:24396/a82e02df
内网面板地址：https://169.254.170.169:24396/a82e02df
username: c16zw0jt
password: 25ecfe14

浏览器访问以下链接，添加宝塔客服
https://www.bt.cn/new/wechat\_customer
=====
Time consumed: 8 Minute!

```

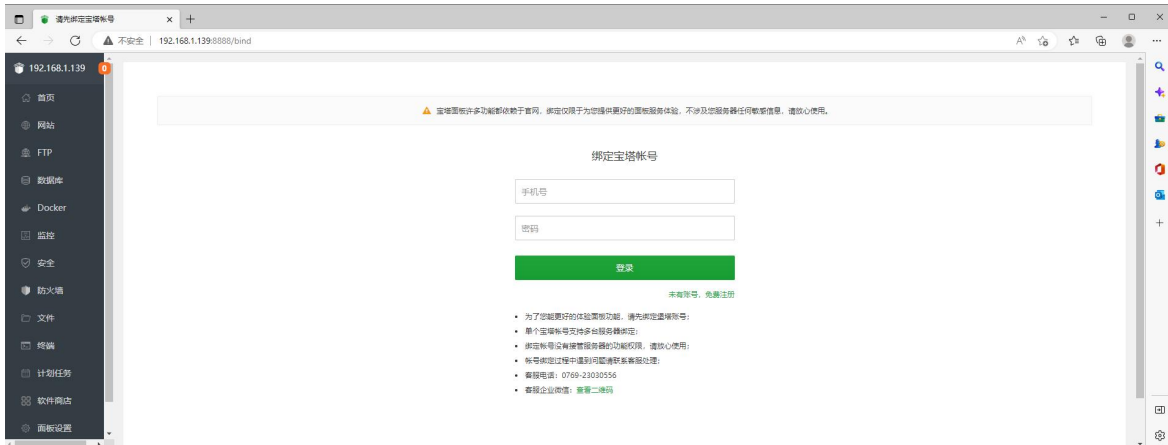
4) At this time, enter the **intranet panel address** shown above in the browser to open the login interface of the Baota Linux panel, and then enter the **username** and **password** shown in the above picture in the corresponding positions to log in to Baota.



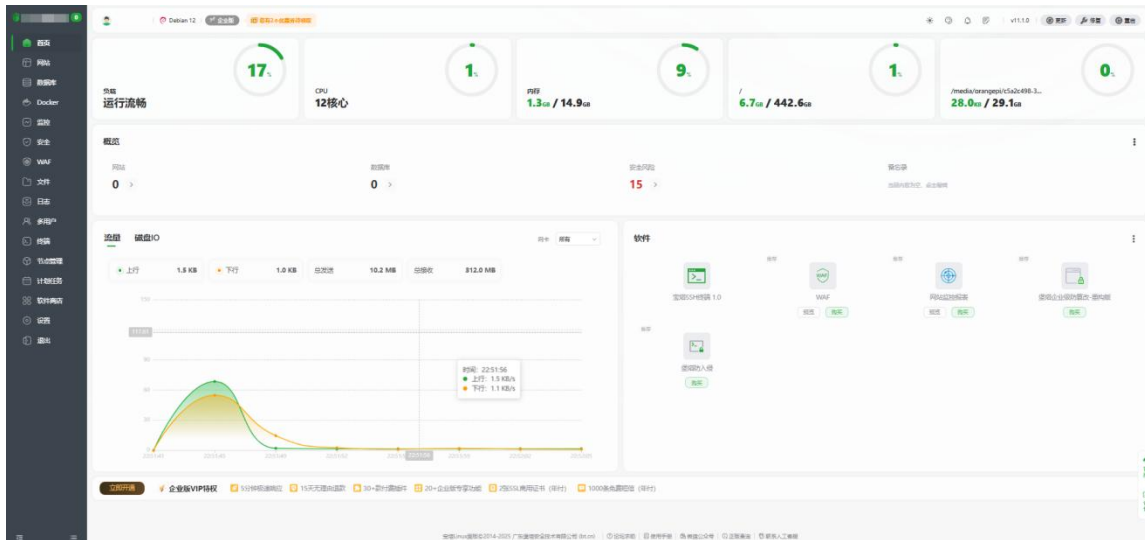
5) After successfully logging into the Baota, the following welcome interface will pop up. First, please read the user instructions in the middle and drag them to the bottom. Then you can select "I have agreed and read the "User Agreement"", and then click "Enter Panel" to enter the Baota.



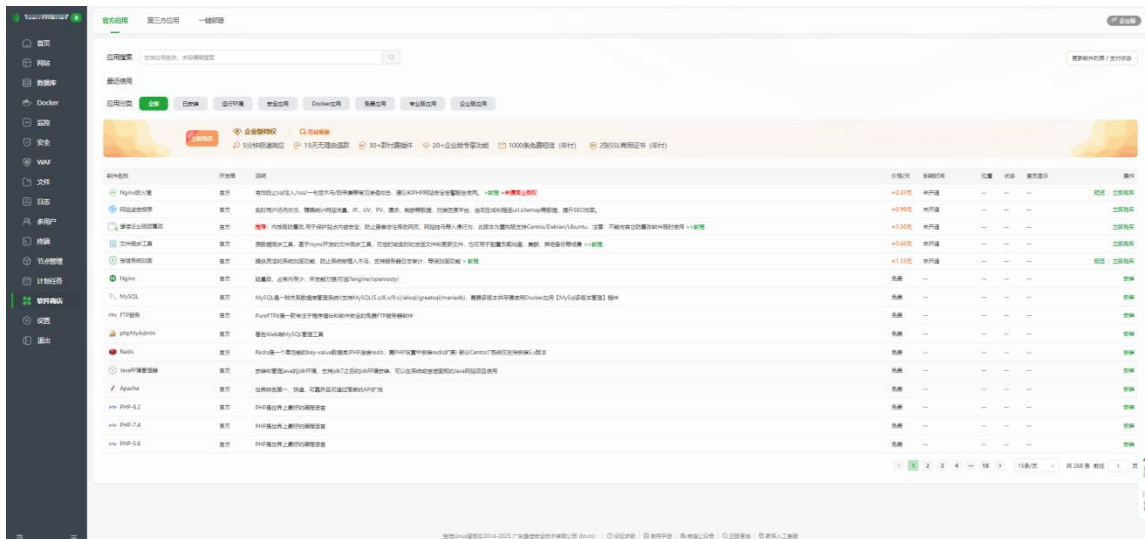
6) After entering the Bt, you will be prompted to bind your account on the Bt official website. If you do not have an account, you can go to the Bt official website (<https://www.bt.cn>) to register one.



7) The final interface is shown in the figure below. You can intuitively see some status information of the development board's Linux system, such as load status, CPU usage, memory usage, and storage space usage.



8) In Baota's software store, you can install software such as Apache, MySQL, and PHP, and you can also deploy various applications with one click. Please explore this function on your own, and I will not demonstrate it one by one here.



9) Baota command line tool test.



```

orangePi@orangePi6plus:~$ sudo bt
[sudo] orangePi 的密码:
=====宝塔面板命令行=====
(1) 重启面板服务          (8) 改面板端口
(2) 停止面板服务          (9) 清除面板缓存
(3) 启动面板服务          (10) 清除登录限制
(4) 重载面板服务          (11) 设置是否开启 IP + User-Agent验证
(5) 修改面板密码          (12) 取消域名绑定限制
(6) 修改面板用户名        (13) 取消 IP访问限制
(7) 强制修改MySQL密码      (14) 查看面板默认信息
(22) 显示面板错误日志      (15) 清理系统垃圾
(23) 关闭BasicAuth认证     (16) 修复面板(检查错误并更新面板文件到最新版)
(24) 关闭动态口令认证      (17) 设置日志切割是否压缩
(25) 设置是否保存文件历史副本 (18) 设置是否自动备份面板
(26) 关闭面板ssl           (19) 关闭面板登录地区限制
(28) 修改面板安全入口      (29) 取消访问设备验证
(30) 取消访问UA验证        (32) 开启/关闭【80、443】端口访问面板
(0) 取消
=====
请输入命令编号: 14
=====
正在执行(14)...
=====
BT-Panel default info!
=====
外网 ipv4面板地址: https://183.17.127.97:24396/a82e02df
内网面板地址: https://169.254.170.169:24396/a82e02df
username: c16zw0jt
password: *****
Warning:
If you cannot access the panel,
release the following port (8888|888|80|443|20|21) in the security group
注意: 初始密码仅在首次登录面板前能正确获取, 其它时间请通过 bt 5 命令修改密码
=====
orangePi@orangePi6plus:~$

```

10) For more functions of the Baota, please refer to the following information to explore it yourself.

User Manual: <http://docs.bt.cn>
 Forum Address: <https://www.bt.cn/bbs>
 GitHub Link: <https://github.com/aaPanel/BaoTa>

3. 30. QT installation method

1) Use the following script to install QT5 and QT Creator.

```
orangePi@orangePi:~$ install_qt.sh
```

2) After installation, the QT version number will be automatically printed.

a. The QT version that comes with Debian12 is **5.15.8**

```
orangePi@orangePi:~$ install_qt.sh
```

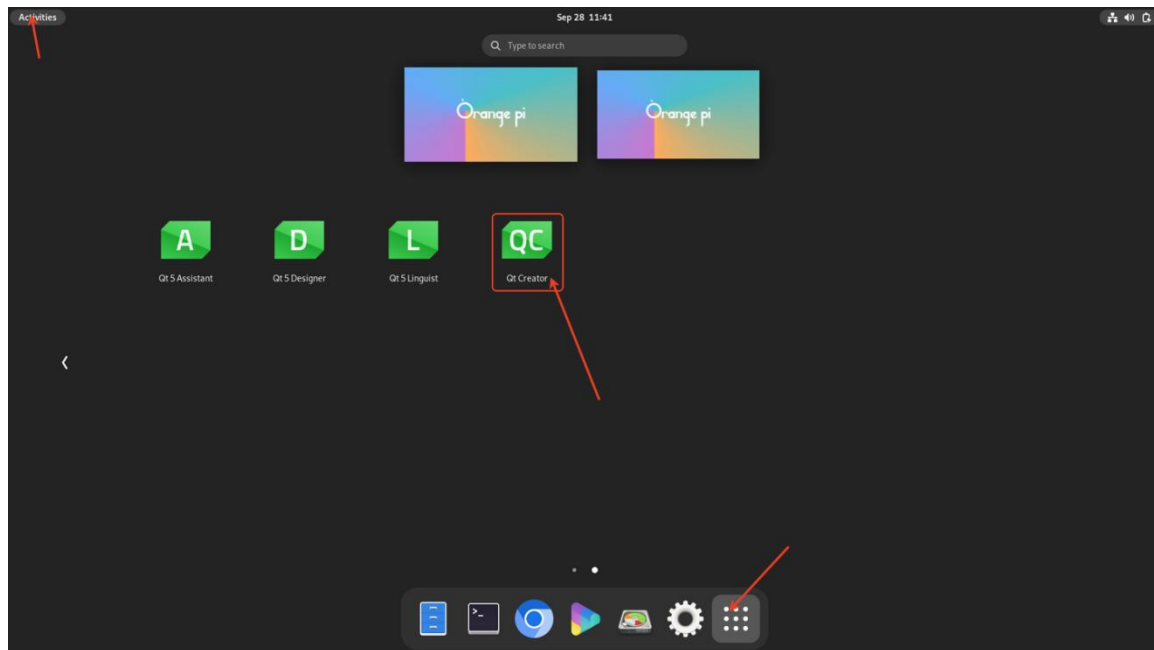



.....

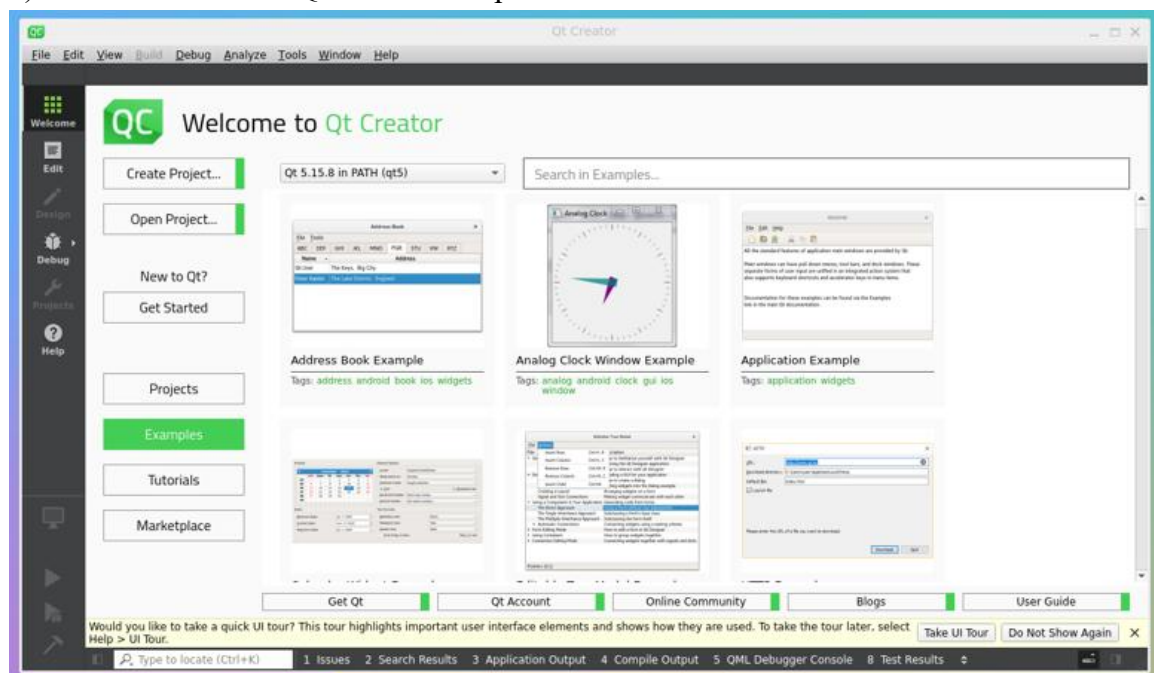
QMake version 3.1

Using Qt version **5.15.8** in /usr/lib/aarch64-linux-gnu

3) Then you can see the QT Creator startup icon in **Applications**.



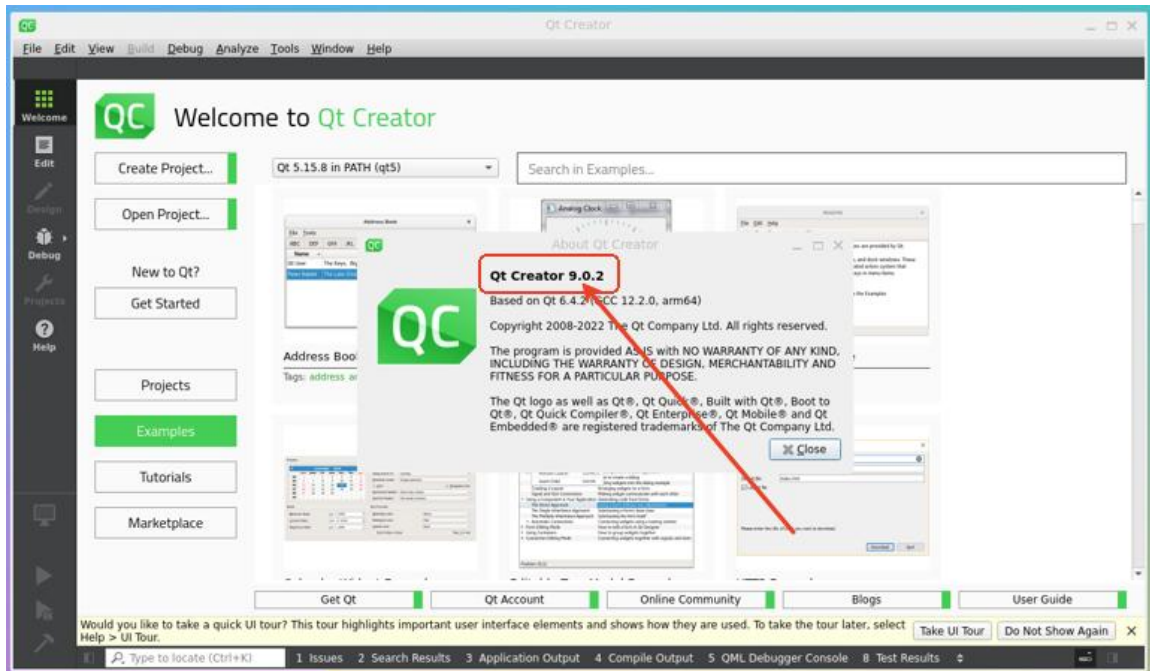
4) The interface after QT Creator is opened is as follows:



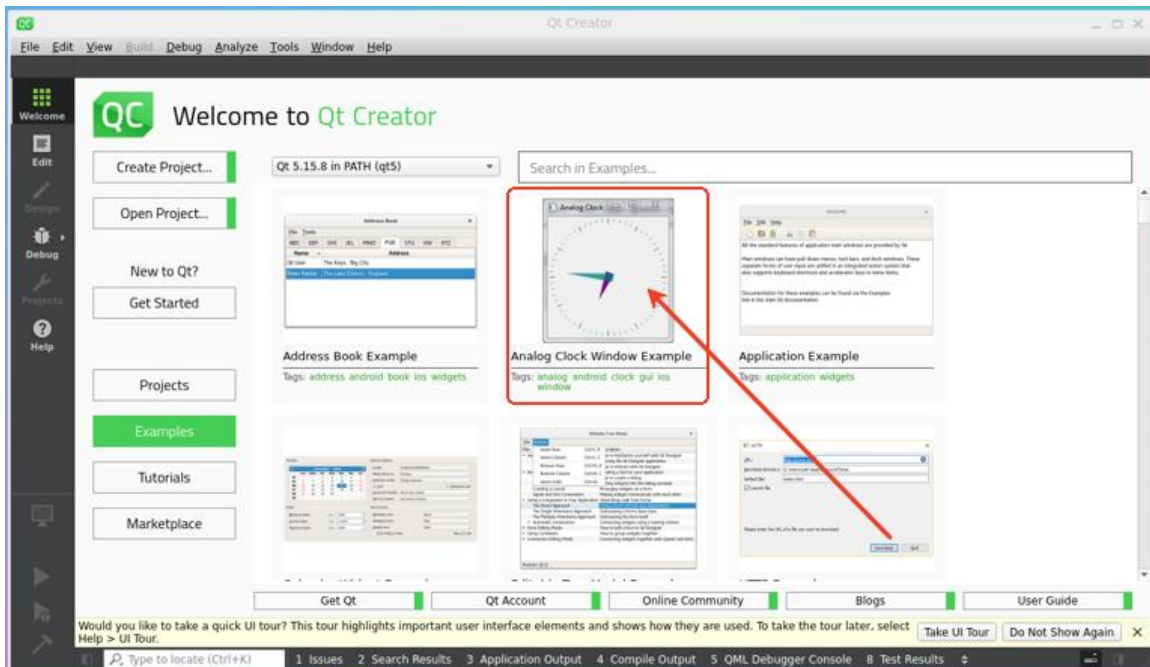


5) The version of QT Creator is as follows:

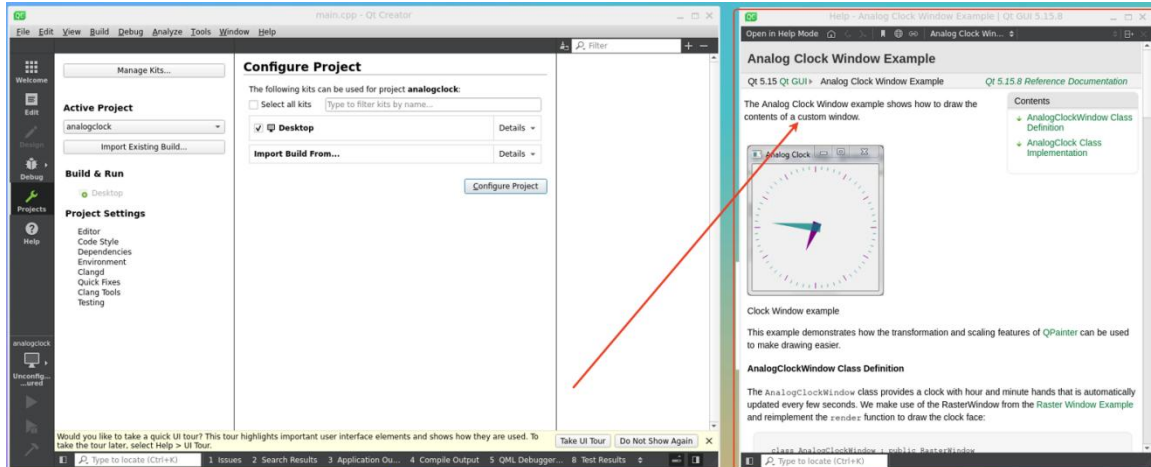
a. The default version of QT Creator in **Debian 12** is as follows:



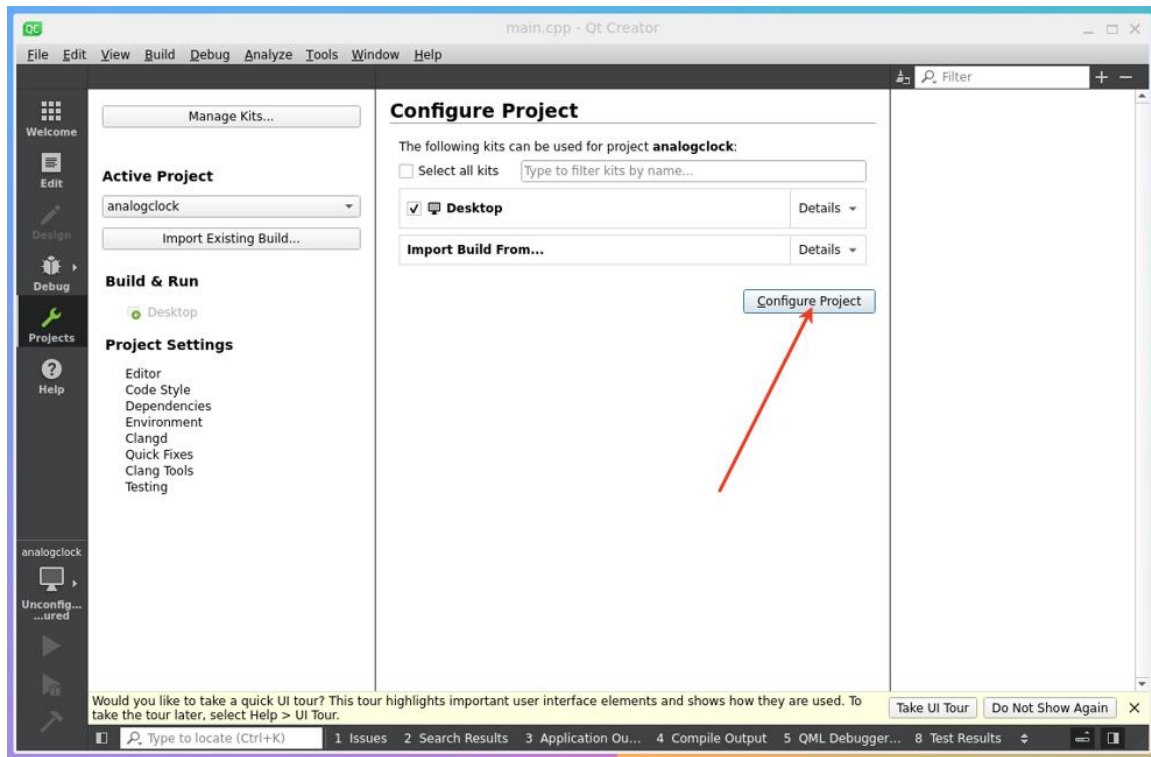
6) Then you can open a sample code.



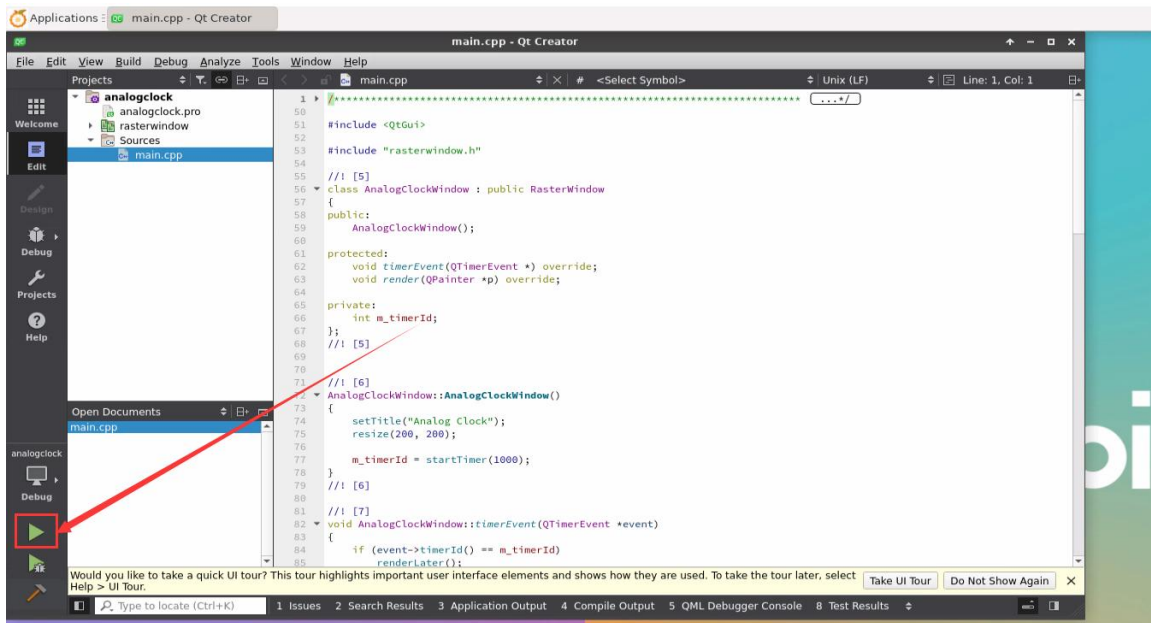
7) Clicking on the sample code will automatically open the corresponding documentation. Please read the instructions carefully.



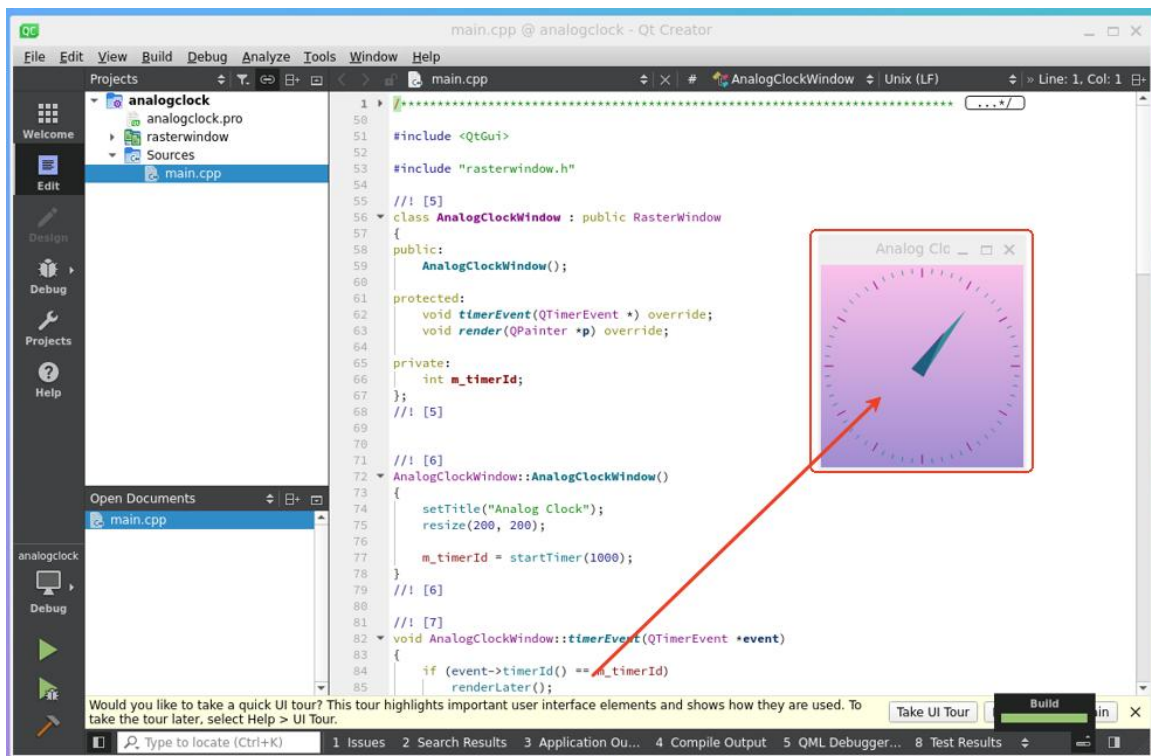
8) Then click **Configure Project**.



9) Then click the green triangle in the lower left corner to compile and run the example code.



10) After waiting for a while, the interface shown in the figure below will pop up, which means that QT can compile and run normally.



11) References:

https://wiki.qt.io/Install_Qt_5_on_Ubuntu



<https://download.qt.io/archive/qtcreator>
<https://download.qt.io/archive/qt>

3. 31. How to install Docker

1) The Linux image provided by Orange Pi has Docker pre-installed, but the Docker service is not enabled by default. The command to enable the Docker service is as follows:

```
orangeapi@orangeapi:~$ sudo systemctl enable docker.service  
orangeapi@orangeapi:~$ sudo systemctl start docker.service
```

2) Then use the following command to test Docker. If hello-world can be run, it means that Docker can be used normally.

```
orangeapi@orangeapi:~$ docker run docker.lms.run/hello-world  
Unable to find image 'docker.lms.run/hello-world:latest' locally  
latest: Pulling from hello-world  
198f93fd5094: Pull complete  
Digest:  
sha256:54e66cc1dd1fcb1c3c58bd8017914dbed8701e2d8c74d9262e26bd9cc1642d31  
Status: Downloaded newer image for docker.lms.run/hello-world:latest
```

Hello from Docker!

This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
(arm64v8)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

```
$ docker run -it ubuntu bash
```



Share images, automate workflows, and more with a free Docker ID:

<https://hub.docker.com/>

For more examples and ideas, visit:

<https://docs.docker.com/get-started/>

3. 32. How to install and use CasaOS

CasaOS is an open source home cloud system based on the Docker ecosystem, which allows you to run a variety of home applications such as NAS, home automation, media server, etc. on your own development board.

3. 32. 1. How to install CasaOS

1) First, you need to install Docker. The OrangePi Pi system has Docker pre-installed, so you can skip this step. You can use the following command to check the installed Docker version.

```
orangepi@orangepi:~$ docker --version
```

2) Download the casaos.sh installation script.

```
orangepi@orangepi:~$ curl -fsSL https://get.casaos.io > casaos.sh
```

3) Then add the following red code to the **casaos.sh** installation script to force the use of domestic links to download and install the required software packages.

```
orangepi@orangepi:~$ vim casaos.sh
.....
# 0 Get download url domain
# To solve the problem that Chinese users cannot access github.
Get_Download_Url_Domain() {
    # Use ipconfig.io/country and https://ipconfig.io/country_code to get the country code
    REGION=$(sudo_cmd curl --connect-timeout 2 -s ipconfig.io/country || echo "")
    REGION=China
    if [ "${REGION}" = "" ]; then
        REGION=$(sudo_cmd curl --connect-timeout 2 -s https://ipconfig.io/country_code || echo "")
    fi
    if [[ "${REGION}" = "China" ]] || [[ "${REGION}" = "CN" ]]; then
        CASA_DOWNLOAD_DOMAIN="https://casaos.oss-cn-shanghai.aliyuncs.com/"
    fi
}
```



```
fi
}
.....
```

4) Run the following command to start installing CasaOS.

```
orangeypi@orangeypi:~$ sudo bash casaos.sh
```

5) The installation process of CasaOS is as follows:

```

CasaOS
--- Made by IceWhale with YOU ---
[ OK ] Your hardware architecture is : aarch64
[ OK ] Your System is : Linux
[ OK ] Your Linux Distribution is : debian
[ OK ] Memory capacity check passed.
[ OK ] Disk capacity check passed.
[ INFO ] Updating package manager ...
W: https://repo.huaweicloud.com/docker-ce/linux/debian/dists/bookworm/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg),
see the DEPRECATION section in apt-key(8) for details.
[ OK ] Update package manager complete.
[ INFO ] Install the necessary dependencies: udev
Selecting previously unselected package udev.
(Reading database ... 140998 files and directories currently installed.)
Preparing to unpack .../udev_0.4.4-3_arm64.deb ...
Unpacking udev (0.4.4-3) ...
Setting up udev (0.4.4-3) ...
Processing triggers for man-db (2.11.2-2) ...
[ INFO ] Install the necessary dependencies: samba
Preconfiguring packages ...
Selecting previously unselected package python3-dnspython.
(Reading database ... 141020 files and directories currently installed.)
Preparing to unpack .../0-python3-dnspython_2.3.0-1_all.deb ...
Unpacking python3-dnspython (2.3.0-1) ...
Selecting previously unselected package python3-ldb.
Preparing to unpack .../1-python3-ldb_2%3a2.6.2+samba4.17.12+dfsg-0+deb12u2_arm64.deb ...
Unpacking python3-ldb (2:2.6.2+samba4.17.12+dfsg-0+deb12u2) ...
Selecting previously unselected package python3-tdb.
Preparing to unpack .../2-python3-tdb_1.4.8-2_arm64.deb ...

```

6) When you see the following information printed on the terminal, it means that CasaOS has been installed.

```

CasaOS v0.4.15 is running at:

- http://10.31.2.166 (eth1)
Open your browser and visit the above address.

CasaOS Project : https://github.com/IceWhaleTech/CasaOS
CasaOS Team   : https://github.com/IceWhaleTech/CasaOS#maintainers
CasaOS Discord : https://discord.gg/knqAbbBbeX
Website       : https://www.casaos.io
Online Demo   : http://demo.casaos.io

Uninstall      : casaos-uninstall

orangeypi@orangeipi6plus:~$
```

7) If you want to uninstall CasaOS, you can use the following command:



```
orange@orange:~$ sudo casaos-uninstall
```

3. 32. 2. Usage of CasaOS

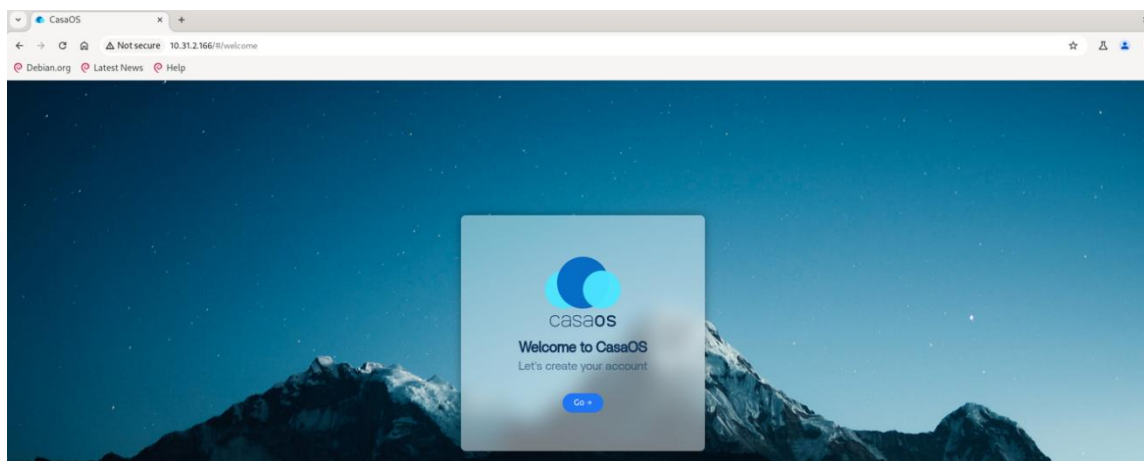
1) After installing CasaOS, enter the IP address of the `http://development` board shown below in the browser to open CasaOS. If the IP address is not displayed, please check if the development board is properly connected to the Ethernet cable or WIFI.

```
CasaOS v0.4.15 is running at:
- http://10.31.2.166 (eth1)
Open your browser and visit the above address.

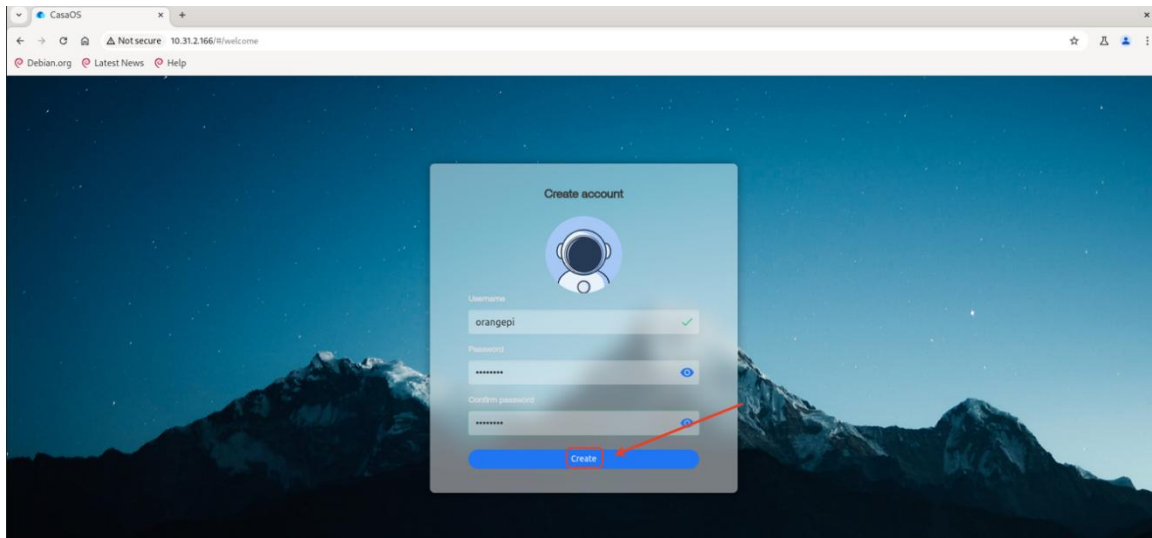
CasaOS Project : https://github.com/IceWhaleTech/CasaOS
CasaOS Team    : https://github.com/IceWhaleTech/CasaOS#maintainers
CasaOS Discord : https://discord.gg/knqAbbBbeX
Website       : https://www.casaos.io
Online Demo    : http://demo.casaos.io

Uninstall      : casaos-uninstall
```

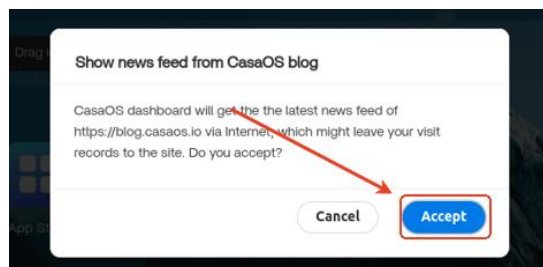
2) After opening CasaOS, a welcome interface will pop up below. Click **Go** to proceed to the next step.



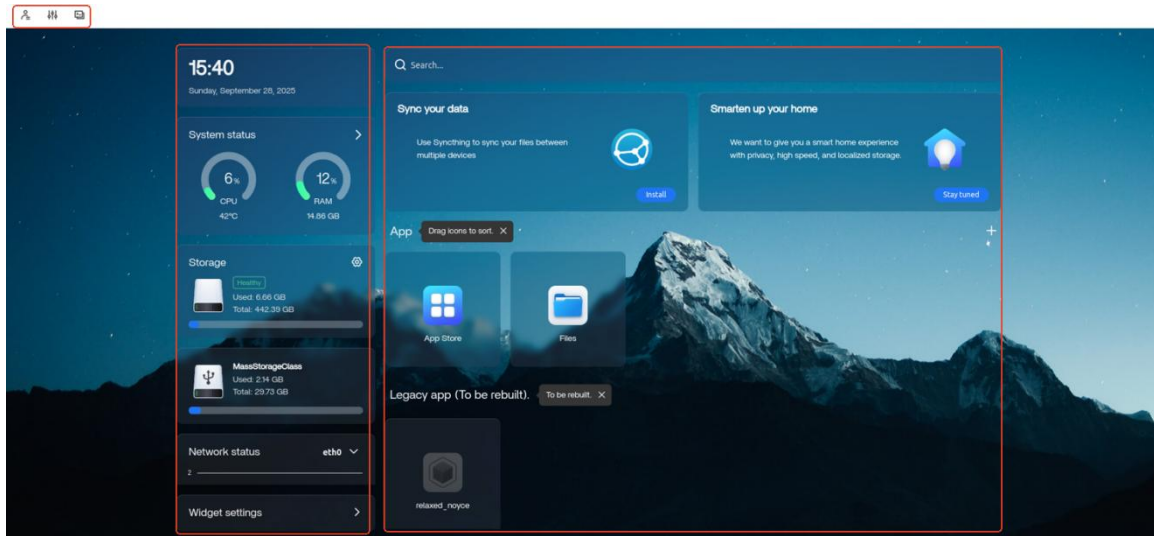
3) When logging into CasaOS for the first time, the login interface will prompt for setting an account password. In the future, when logging in again, only the interface for entering the account password will appear. After setting the account password, click **Create** to proceed to the next step.



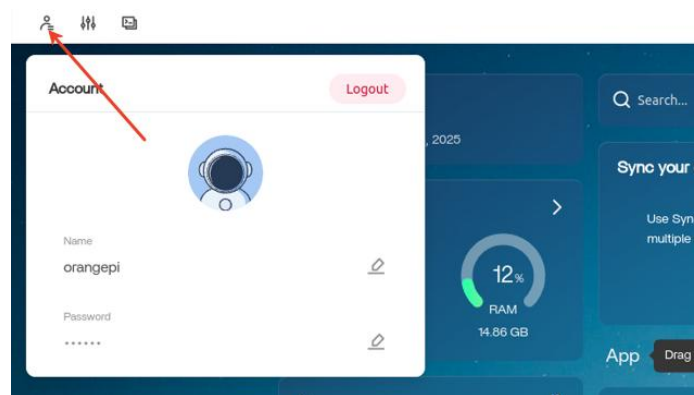
4) Click **Accept** directly on the interface below to proceed to the next step.



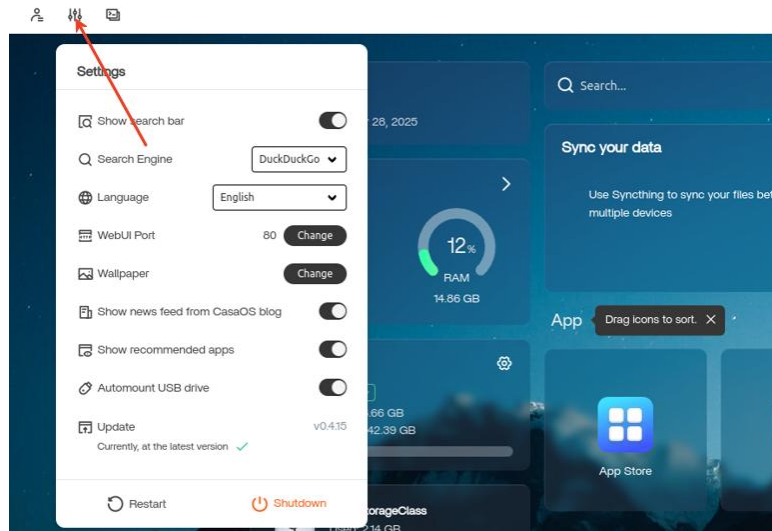
5) Then you will enter the main page of CasaOS. There are three icons in the upper left corner for setting functions. The left side is the performance panel, which displays the current time and status information of CPU, RAM, storage, and network. The right side is the function panel, which has search, application, and other functions.



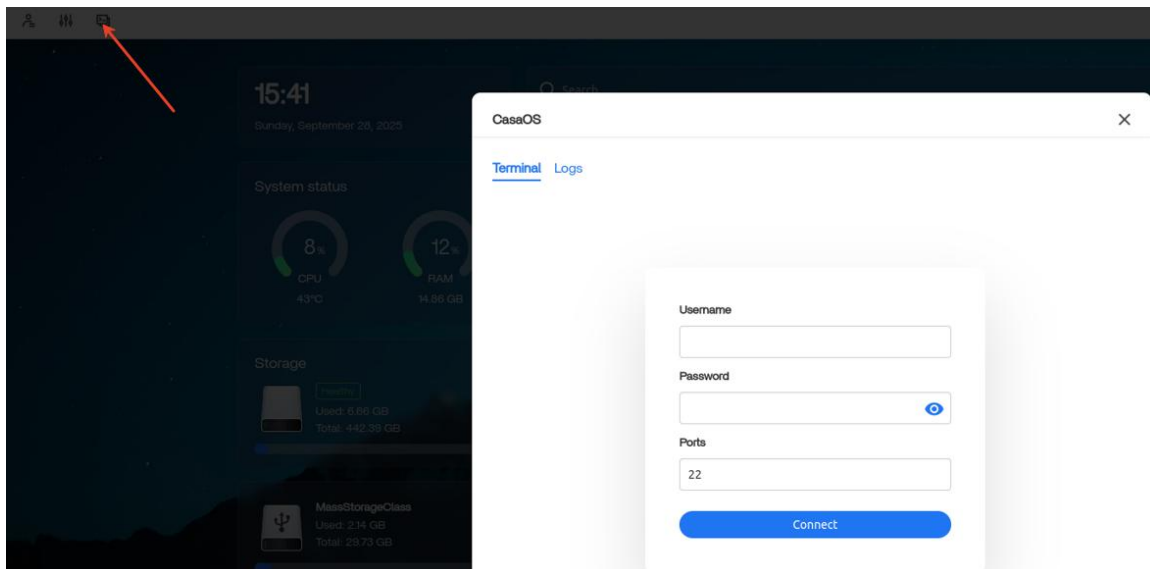
6) Click on the first icon in the upper left corner to change your account and password.



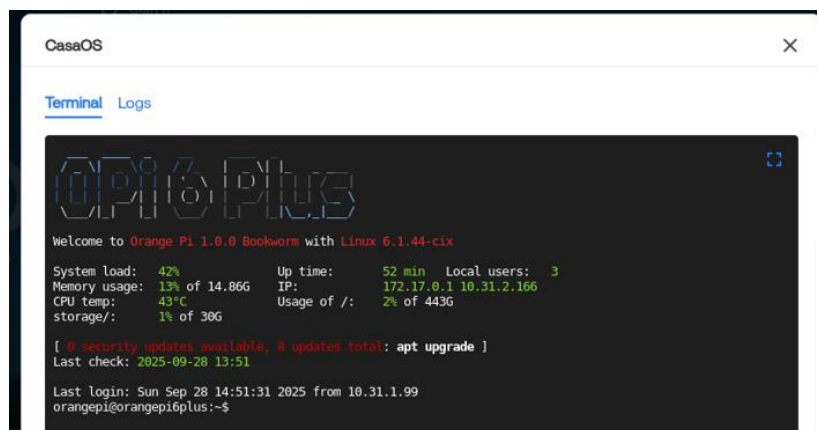
7) Click on the second icon to set up basic functions.



8) The third icon in the upper left corner mainly has two functions, which are to open terminal mode and view log information. When opening the terminal, you need to enter your account and password. Here, the account and password refer to the Linux system account and password of the development board, and the default port number is **22**.



9) Then click **Connect** to enter the command line interface:



```
CasaOS

Terminal Logs

OPi6 PLUS

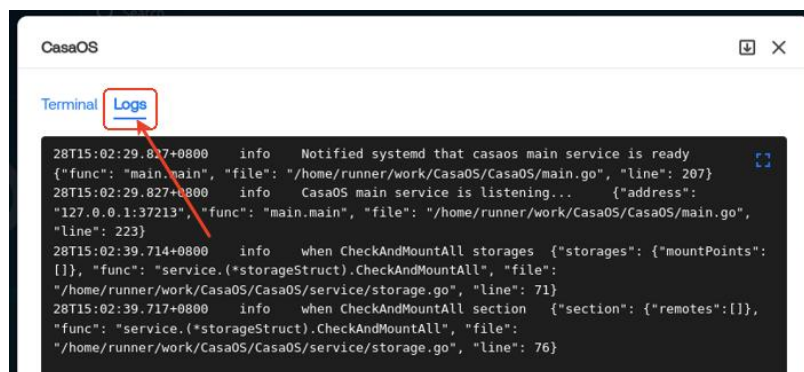
Welcome to Orange Pi 1.0.0 Bookworm with Linux 6.1.44-cix

System load: 42%      Up time: 52 min   Local users: 3
Memory usage: 13% of 14.86G  IP: 172.17.0.1 10.31.2.166
CPU temp: 43°C      Usage of /: 2% of 443G
storage: 1% of 30G

[ 0 security updates available, 0 updates total: apt upgrade ]
Last check: 2025-09-28 13:51

Last login: Sun Sep 28 14:51:31 2025 from 10.31.1.99
orangepi@orangepi6plus:~$
```

10) Another function under the third icon is to view CasaOS logs. Click on **Logs** to enter, and the interface is shown below:

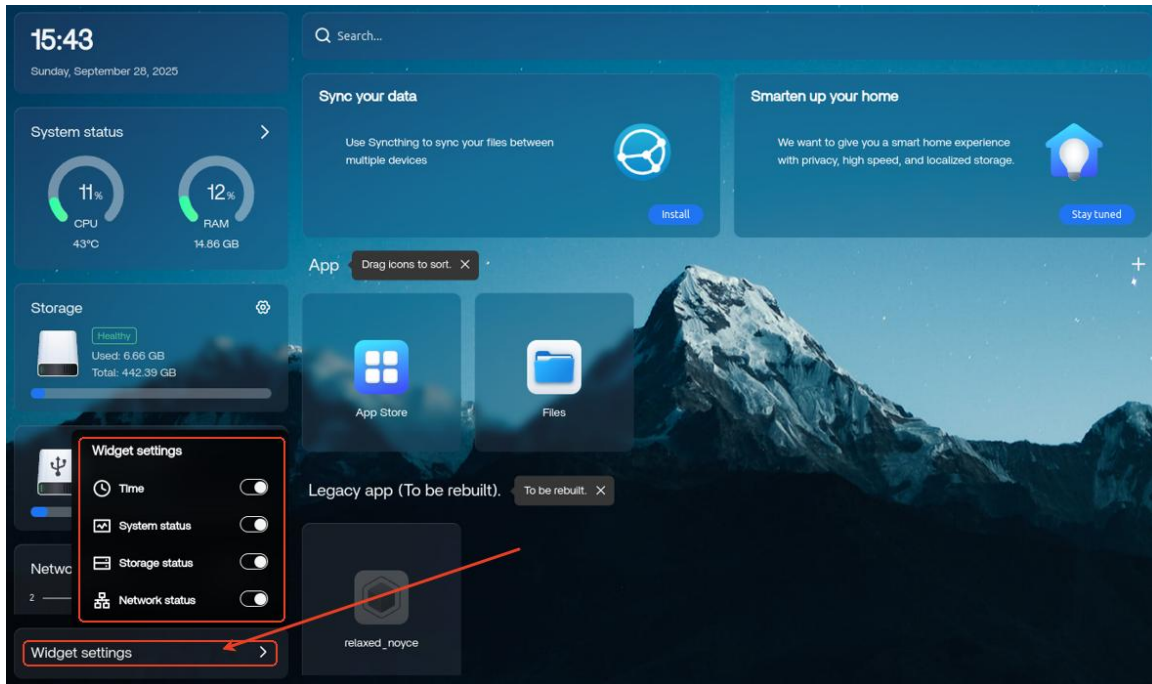


```
CasaOS

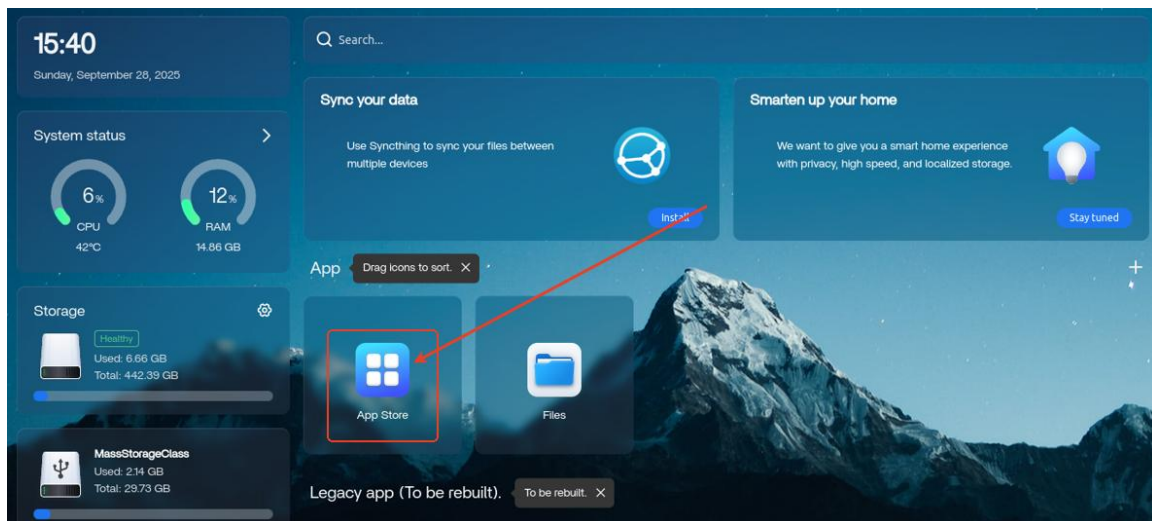
Terminal Logs

28T15:02:29.827+0800 info Notified systemd that casaos main service is ready
{"func": "main.main", "file": "/home/runner/work/CasaOS/CasaOS/main.go", "line": 207}
28T15:02:29.827+0800 info CasaOS main service is listening... {"address":
"127.0.0.1:37213", "func": "main.main", "file": "/home/runner/work/CasaOS/CasaOS/main.go",
"line": 223}
28T15:02:39.714+0800 info when CheckAndMountAll storages {"storages": {"mountPoints":
[]}, "func": "service.(*storageStruct).CheckAndMountAll", "file":
"/home/runner/work/CasaOS/CasaOS/service/storage.go", "line": 71}
28T15:02:39.717+0800 info when CheckAndMountAll section {"section": {"remotes":[]},
"func": "service.(*storageStruct).CheckAndMountAll", "file":
"/home/runner/work/CasaOS/CasaOS/service/storage.go", "line": 76}
```

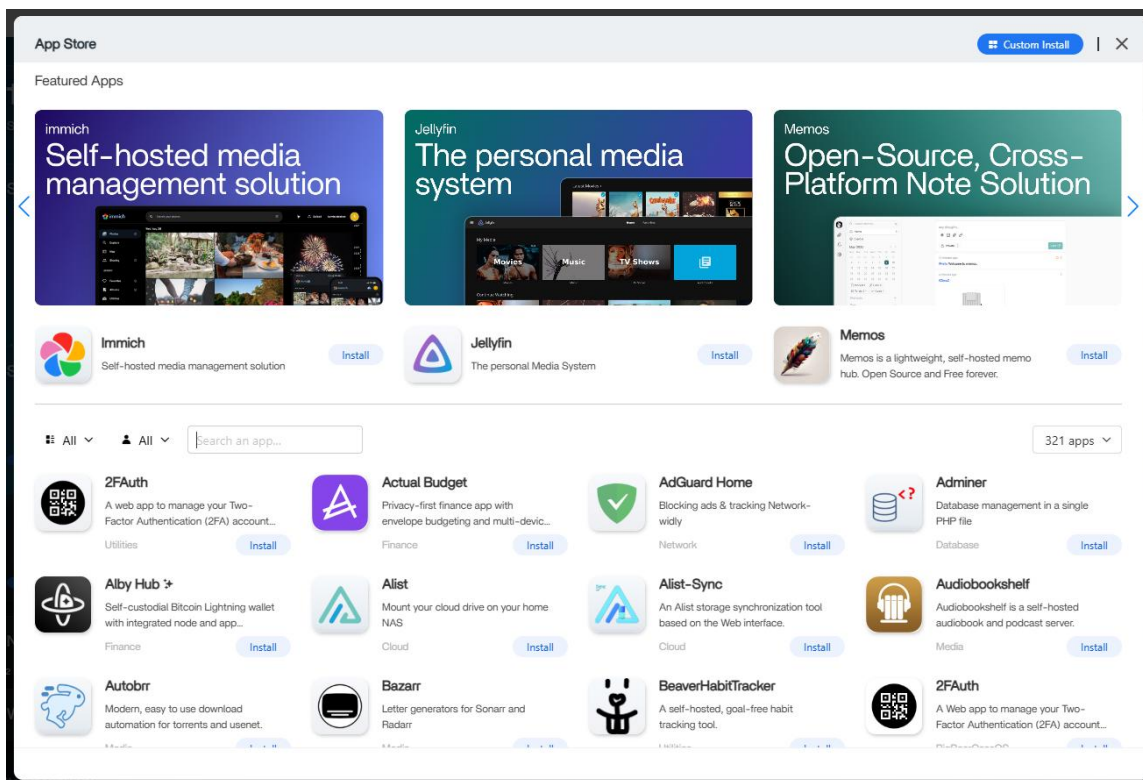
11) Click on the **Widget Settings** in the bottom left corner to open or close widgets on the main page.



12) Click on the **APP Store** in the main interface to open the App Center.



13) The interface of the application center is shown below, where you can select the application you want to use for installation.



If the installation speed of the application is slow, you can try switching the Docker image source. The steps are as follows:

1. Add the following configuration file to set the Docker image source to <https://docker.1ms.run>. If this source is invalid, you can search for other available and fast sources on Baidu.

```
orangepi@orangepi:~$ sudo vim /etc/docker/daemon.json
```

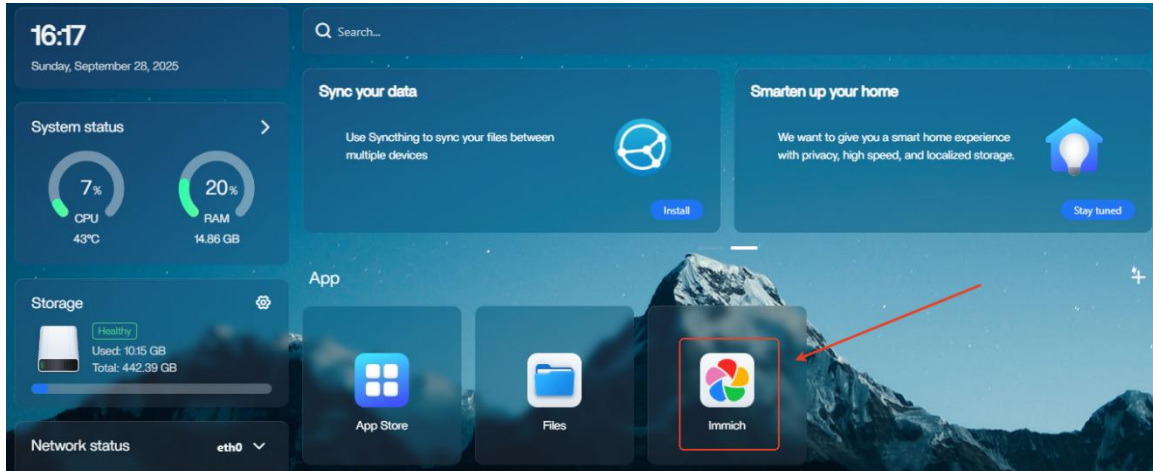
```
{
  "registry-mirrors": ["https://docker.1ms.run"]
}
```

2. Then restart the Docker service.

```
orangepi@orangepi:~$ sudo systemctl daemon-reload
```

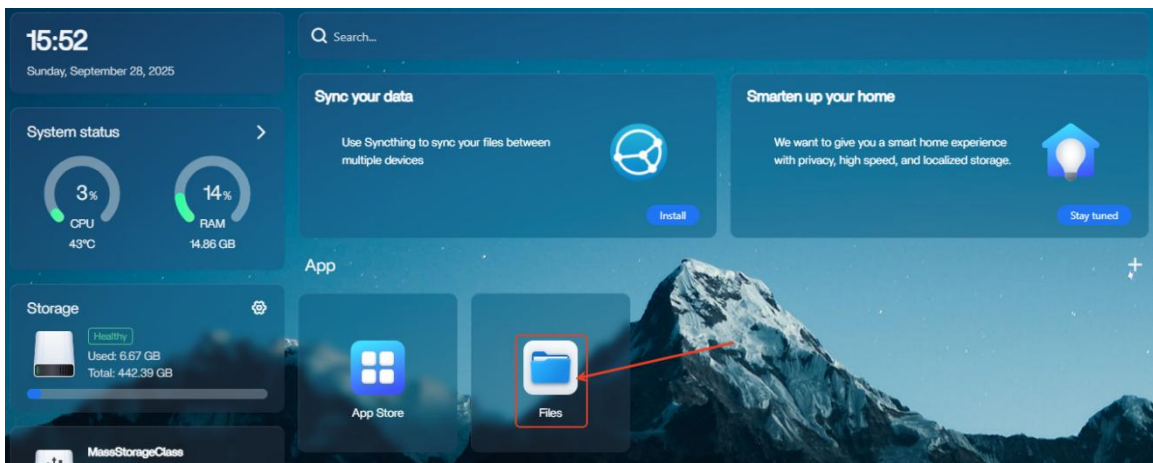
```
orangepi@orangepi:~$ sudo systemctl restart docker
```

14) The installed application will be displayed in the **APP** on the main interface.

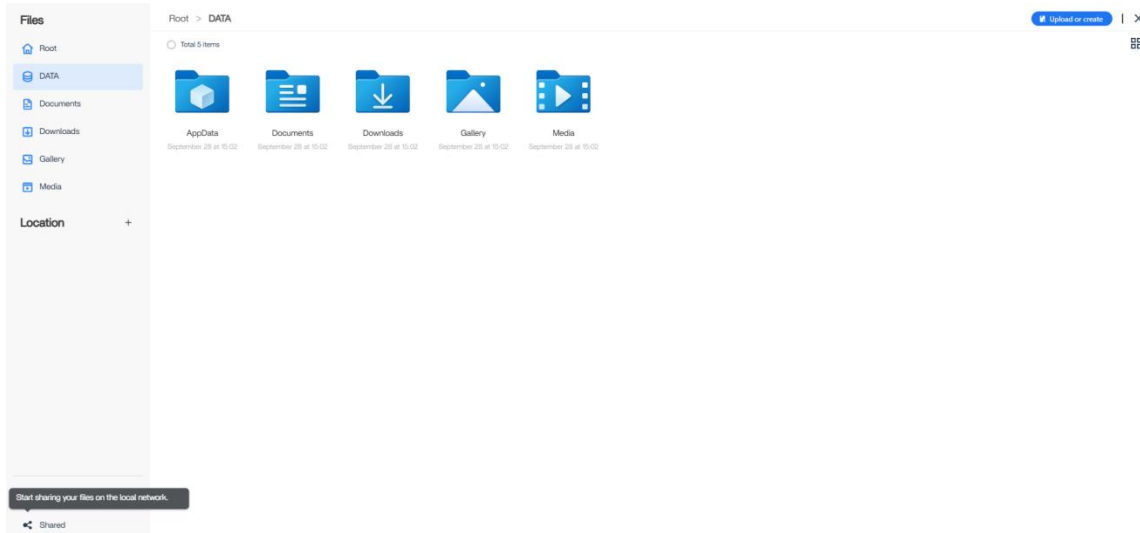


15) Click on **Files** in the main interface to open the built-in file system of CasaOS, and then you can upload and save files.

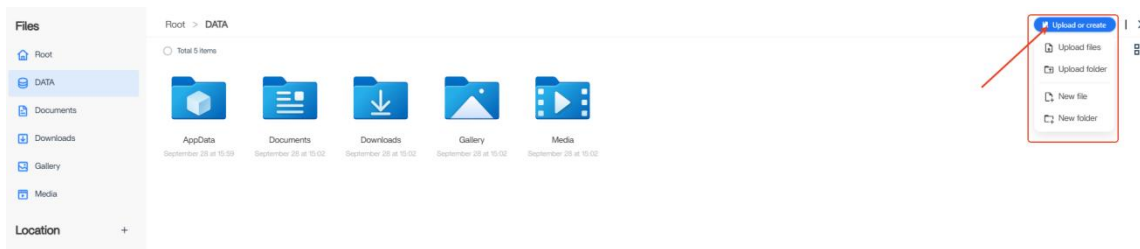
Please ensure that other devices and development boards are on the same LAN.



16) The interface after opening **Files** is shown below:



17) Then click **Upload or Create** to select a file or folder for uploading.



3. 33. Method to view SoC chip serial number

The command to check the chip serial number is as follows. Each chip has a different serial number, so it can be used to distinguish multiple development boards.

```
orangepi@orangepi:~$ sudo dmidecode -t processor | grep Serial
```

```
Serial Number: 00200C030491CD91
```

3. 34. Usage of ZFS file system

Note that the content of this section only applies to systems with the Linux 6.1 kernel; it cannot be used on systems with the Linux 6.6 kernel.

3. 34. 1. Method of installing ZFS

Before installing ZFS, the kernel header file needs to be installed first. Please refer to



the instructions in the section on installing kernel header files for [the method of installing kernel header files](#).

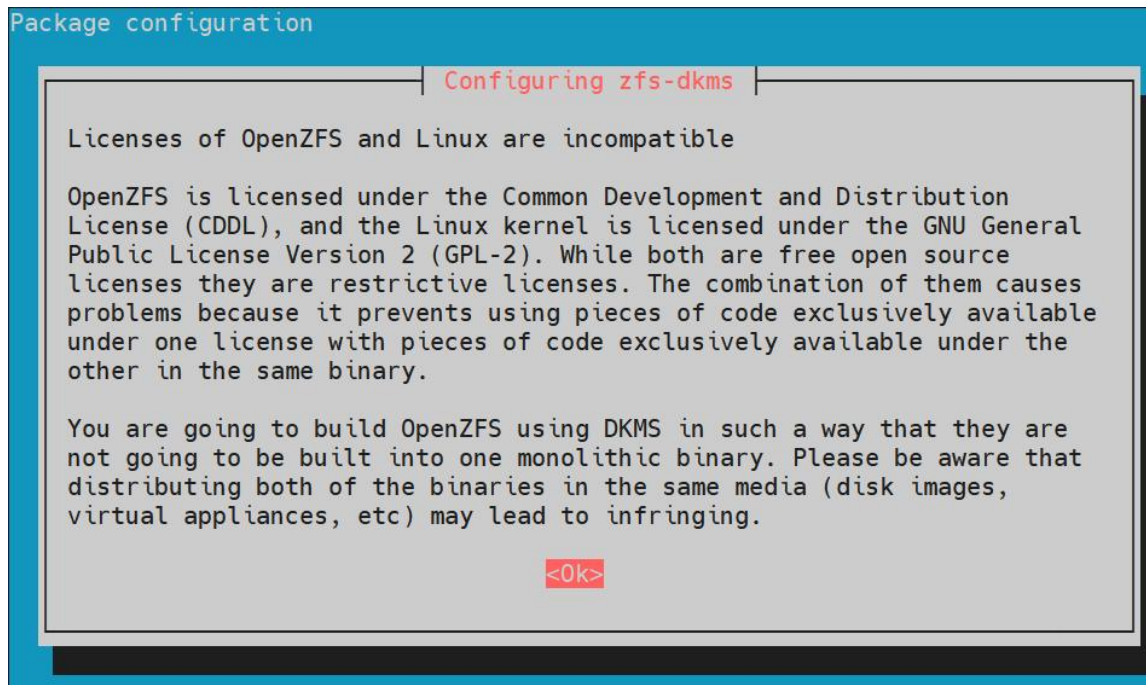
In Debian 12, zfs can be directly installed using the following command. **Once again, it is important to ensure that the deb package of the kernel header files is installed on the system before installation.**

```
orange@orange:~$ sudo apt install -y zfsutils-linux zfs-dkms
```

Note that the content of this section is only applicable to Linux 6.1 kernel systems. Linux 6.6 kernel systems will report the following error:

```
Loading new zfs-2.1.11 DKMS files ...
Building for 6.6.89-cix
Building initial module for 6.6.89-cix
configure: error:
*** None of the expected "bops→release()" interfaces were detected.
*** This may be because your kernel version is newer than what is
*** supported, or you are using a patched custom kernel with
*** incompatible modifications.
***
*** ZFS Version: zfs-2.1.11-1+deb12u1
*** Compatible Kernels: 3.10 - 6.2
```

During the installation process, the following interface will pop up. Simply press Enter.



The printed information after installation is as follows:



```

zzstd.ko:
Running module version sanity check.
- Original module
  - No original module exists within this kernel
- Installation
  - Installing to /lib/modules/6.1.44-cix/updates/dkms/
depmod ...
Setting up libuutil3linux (2.1.11-1+deb12u1) ...
Setting up libzfs4linux (2.1.11-1+deb12u1) ...
Setting up libzpool5linux (2.1.11-1+deb12u1) ...
Setting up zfsutils-linux (2.1.11-1+deb12u1) ...
insmod /lib/modules/6.1.44-cix/updates/dkms/spl.ko
insmod /lib/modules/6.1.44-cix/updates/dkms/icp.ko
insmod /lib/modules/6.1.44-cix/updates/dkms/zavl.ko
insmod /lib/modules/6.1.44-cix/updates/dkms/znvpair.ko
insmod /lib/modules/6.1.44-cix/updates/dkms/zcommon.ko
insmod /lib/modules/6.1.44-cix/updates/dkms/zlua.ko
insmod /lib/modules/6.1.44-cix/updates/dkms/zzstd.ko
insmod /lib/modules/6.1.44-cix/updates/dkms/zunicode.ko
insmod /lib/modules/6.1.44-cix/updates/dkms/zfs.ko
Created symlink /etc/systemd/system/zfs-import.target.wants/zfs-import-cache.service → /lib/systemd/system/zfs-import-cache.service.
Created symlink /etc/systemd/system/zfs.target.wants/zfs-import.target → /lib/systemd/system/zfs-import.target.
Created symlink /etc/systemd/system/zfs-mount.service.wants/zfs-load-module.service → /lib/systemd/system/zfs-load-module.service.
Created symlink /etc/systemd/system/zfs.target.wants/zfs-load-module.service → /lib/systemd/system/zfs-load-module.service.
Created symlink /etc/systemd/system/zfs.target.wants/zfs-mount.service → /lib/systemd/system/zfs-mount.service.
Created symlink /etc/systemd/system/zfs.target.wants/zfs-share.service → /lib/systemd/system/zfs-share.service.
Created symlink /etc/systemd/system/zfs-volumes.target.wants/zfs-volume-wait.service → /lib/systemd/system/zfs-volume-wait.service.
Created symlink /etc/systemd/system/zfs.target.wants/zfs-volumes.target → /lib/systemd/system/zfs-volumes.target.
Created symlink /etc/systemd/system/multi-user.target.wants/zfs.target → /lib/systemd/system/zfs.target.
zfs-import-scan.service is a disabled or a static unit, not starting it.
Processing triggers for libc-bin (2.36-9+deb12u13) ...
Processing triggers for man-db (2.11.2-2) ...
Processing triggers for unitramfs-tools (0.142+deb12u3) ...

```

You can use the following command to see the kernel modules related to zfs:

```

orangeypi@orangeypi:~$ ls /lib/modules/*/updates/dkms
icp.ko  zavl.ko  zfs.ko  znvpair.ko  zzstd.ko
spl.ko  zcommon.ko  zlua.ko  zunicode.ko

```

Using lsmod, you can see that the zfs kernel module will automatically load:

```

orangeypi@orangeypi:~$ lsmod | grep zfs
zfs                3051520  0
zunicode            335872   1 zfs
zzstd               442368   1 zfs
zlua                163840   1 zfs
zcommon             90112    1 zfs
znvpair            106496    2 zfs,zcommon
zavl                20480    1 zfs
icp                 237568    1 zfs
spl                 106496    6 zfs,icp,zzstd,znvpair,zcommon,zavl

```

Use the following command to view the version number of zfs:

```

orangeypi@orangeypi:~$ zfs version
zfs-2.2.99-687_gb3b749161
zfs-kmod-2.2.99-687_gb3b749161

```



3. 34. 2. Method for Creating ZFS Pool

ZFS is based on storage pools, where we can add multiple storage devices to the pool and allocate storage space from it. Here we use files as virtual disks for demonstration.

1) Create two 1GB virtual disk image files.

```
orangepi@orangepi:~$ sudo mkdir /tmp/zfs-test
orangepi@orangepi:~$ cd /tmp/zfs-test
orangepi@orangepi:/tmp/zfs-test$ sudo dd if=/dev/zero of=disk1.img bs=1M count=1024
orangepi@orangepi:/tmp/zfs-test$ sudo dd if=/dev/zero of=disk2.img bs=1M count=1024
```

2) The created file is as follows:

```
orangepi@orangepi:/tmp/zfs-test$ ls
disk1.img  disk2.img
```

3) Associate the virtual disk image file with the loop device.

```
orangepi@orangepi:/tmp/zfs-test$ sudo losetup /dev/loop0 disk1.img
orangepi@orangepi:/tmp/zfs-test$ sudo losetup /dev/loop1 disk2.img
```

4) View associated loop devices.

```
orangepi@orangepi:/tmp/zfs-test$ sudo losetup -a
/dev/loop1: [66306]:2095155 (/tmp/zfs-test/disk2.img)
/dev/loop0: [66306]:2095154 (/tmp/zfs-test/disk1.img)
```

5) Then a ZFS image pool can be created using virtual disks.

```
orangepi@orangepi:~$ sudo zpool create -f vpool mirror /dev/loop0 /dev/loop1
```

6) The command to view the detailed status of the pool is as follows:

```
orangepi@orangepi:~$ sudo zpool status vpool
pool: vpool
state: ONLINE
config:

    NAME      STATE    READ WRITE CKSUM
    vpool     ONLINE      0     0     0
```



mirror-0	ONLINE	0	0	0
loop0	ONLINE	0	0	0
loop1	ONLINE	0	0	0

errors: No known data errors

7) The command to view the pool topology structure is as follows:

```
orangeipi@orangeipi:~$ sudo zpool list -v
```

```
orangeipi@orangeipi:~$ sudo zpool list -v
NAME      SIZE  ALLOC   FREE CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
vpool     960M   108K   960M      -         -         0%    0%   1.00x    ONLINE  -
  mirror-0 960M   108K   960M      -         -         0%   0.01%    -    ONLINE
    loop0   1G      -      -         -         -         -     -     -    ONLINE
    loop1   1G      -      -         -         -         -     -     -    ONLINE
orangeipi@orangeipi:~$
```

8) Executing **df -h** shows that the vpool pool is mounted to the **/vpool** directory.

```
orangeipi@orangeipi:~$ df -h | grep vpool
```

```
vpool          832M  128K  832M    1% /vpool
```

9) The following command shows that the file system type of vpool is zfs.

```
orangeipi@orangeipi:~$ mount | grep vpool
```

```
vpool on /vpool type zfs (rw,xattr,noacl)
```

10) Then we can test copying a file to the ZFS pool.

```
orangeipi@orangeipi:~$ sudo cp -v /opt/video_h265.mp4 /vpool/
'/opt/video_h265.mp4' -> '/vpool/video_h265.mp4'
```

3. 34. 3. Testing the Data De duplication Function of ZFS

1) The following command can be used to view the status of ZFS data deduplication function, which is turned off by default.

```
orangeipi@orangeipi:~$ sudo zfs get dedup vpool
```

NAME	PROPERTY	VALUE	SOURCE
vpool	dedup	off	default

2) The following command can be used to enable the ZFS data deduplication function.

```
orangeipi@orangeipi:~$ sudo zfs set dedup=on vpool
```




3) Then perform a simple test by executing the following command to generate a random file of 1MB size in the vpool folder.

```
orangepi@orangepi:~$ cd /vpool/
orangepi@orangepi:/vpool$ sudo dd if=/dev/urandom of=test.bin bs=1M count=1
```

4) Then use the following command to copy 100 copies of the test.bin file

```
orangepi@orangepi:/vpool$ for ((i=0; i<100; i++)); do sudo cp test.bin $i.test.bin; done
```

```
orangepi@orangepi:/vpool$ for ((i=0; i<100; i++)); do sudo cp test.bin $i.test.bin; done
orangepi@orangepi:/vpool$ ls
0.test.bin 20.test.bin 31.test.bin 42.test.bin 53.test.bin 64.test.bin 75.test.bin 86.test.bin 97.test.bin
10.test.bin 21.test.bin 32.test.bin 43.test.bin 54.test.bin 65.test.bin 76.test.bin 87.test.bin 98.test.bin
11.test.bin 22.test.bin 33.test.bin 44.test.bin 55.test.bin 66.test.bin 77.test.bin 88.test.bin 99.test.bin
12.test.bin 23.test.bin 34.test.bin 45.test.bin 56.test.bin 67.test.bin 78.test.bin 89.test.bin 9.test.bin
13.test.bin 24.test.bin 35.test.bin 46.test.bin 57.test.bin 68.test.bin 79.test.bin 8.test.bin test.bin
14.test.bin 25.test.bin 36.test.bin 47.test.bin 58.test.bin 69.test.bin 7.test.bin 90.test.bin
15.test.bin 26.test.bin 37.test.bin 48.test.bin 59.test.bin 6.test.bin 80.test.bin 91.test.bin
16.test.bin 27.test.bin 38.test.bin 49.test.bin 5.test.bin 70.test.bin 81.test.bin 92.test.bin
17.test.bin 28.test.bin 39.test.bin 4.test.bin 60.test.bin 71.test.bin 82.test.bin 93.test.bin
18.test.bin 29.test.bin 3.test.bin 50.test.bin 61.test.bin 72.test.bin 83.test.bin 94.test.bin
19.test.bin 2.test.bin 40.test.bin 51.test.bin 62.test.bin 73.test.bin 84.test.bin 95.test.bin
1.test.bin 30.test.bin 41.test.bin 52.test.bin 63.test.bin 74.test.bin 85.test.bin 96.test.bin
```

5) Then using **du -h**, it can be seen that there is a total of 102M data in the vpool pool at present.

```
orangepi@orangepi:/vpool$ du -h
102M .
```

6) However, using the **zpool list** command to check that only **1.47M** of space was used in the vpool pool indicates that the data deduplication function is effective. Because these 101 files are all duplicates, ZFS only stores one copy when multiple files contain the same data. Furthermore, from the value of **DEDUP**, it can be seen that the deduplication rate has reached **101.00x**.

```
orangepi@orangepi:/vpool$ zpool list
```

```
orangepi@orangepi:/vpool$ zpool list
NAME      SIZE  ALLOC   FREE CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
vpool    960M  1.47M   959M      -          -         0%    0%  101.00x  ONLINE  -
```

3. 34. 4. Testing the Data Compression Function of ZFS

1) First, turn off the data compression function.

```
orangepi@orangepi:/vpool$ sudo zfs set compression=off vpool
```

2) By using the following command, you can see that the data compression function is in the **off** state.

```
orangepi@orangepi:/vpool$ sudo zfs get compression vpool
NAME      PROPERTY      VALUE      SOURCE
```




vpool	compression	off	local
-------	-------------	-----	-------

3) The amount of disk space saved by compression may vary for different types of data. Here, we choose a plain text file with a relatively large compression ratio for data compression testing. Use the following command to create a plain text file in the vpool pool.

```
orangepi@orangepi:~$ cd /vpool/
orangepi@orangepi:/vpool$ sudo sh -c 'yes "ABCDEF" | head -10000000 > test.txt'
```

4) Before enabling the data compression function, the **test.txt** file that can be seen through the **zpool list** command occupies **67M** of space in the ZFS pool, which is consistent with its actual size.

```
orangepi@orangepi:/vpool$ zpool list
```

```
orangepi@orangepi:/vpool$ ls -lh
```

```
total 67M
```

```
-rw-r--r-- 1 root root 67M Sep 29 10:45 test.txt
```

```
orangepi@orangepi:/vpool$ zpool list
```

NAME	SIZE	ALLOC	FREE	CKPOINT	EXPANDSZ	FRAG	CAP	DEDUP	HEALTH	ALTROOT
vpool	960M	67.1M	893M	-	-	0%	6%	1.00x	ONLINE	-

```
orangepi@orangepi:/vpool$
```

5) The compression ratio obtained by using the following command is 1, which means there is no compression.

```
orangepi@orangepi:/vpool$ sudo zfs get compressratio vpool
```

NAME	PROPERTY	VALUE	SOURCE
vpool	compressratio	1.00x	-

6) Then enable data compression function in the vpool pool.

```
orangepi@orangepi:/vpool$ sudo zfs set compression=lz4 vpool
```

7) At this point, using the following command will show that the data compression function is in the **lz4** state.

```
orangepi@orangepi:/vpool$ sudo zfs get compression vpool
```

NAME	PROPERTY	VALUE	SOURCE
vpool	compression	lz4	local

8) Then execute the following command to regenerate the **test.txt** file.

```
orangepi@orangepi:/vpool$ sudo sh -c 'yes "ABCDEF" | head -10000000 > test.txt'
```



9) At this point, the **test.txt** file visible through the **zpool list** command only occupies a few hundred KB of space in the ZFS pool, indicating that the data has been compressed.

```
orangepi@orangepi:/vpool$ zpool list
orangepi@orangepi:/vpool$ ls -lh
total 547K
-rw-r--r-- 1 root root 67M Sep 29 10:50 test.txt
orangepi@orangepi:/vpool$ zpool list
NAME      SIZE  ALLOC   FREE CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
vpool    960M  806K   959M      -          -         0%    0%   1.00x    ONLINE  -
orangepi@orangepi:/vpool$
```

10) The compression ratio obtained by using the following command is over 111.

```
orangepi@orangepi:/vpool$ sudo zfs get compressratio vpool
NAME      PROPERTY          VALUE        SOURCE
vpool     compressratio     111.12x     -
```

3. 35. Method for Setting Swap Memory

When the memory required by the application is greater than the physical available memory of the development board, Swap memory can be used to expand the maximum memory capacity that the system can use. The method is as follows:

1) Create a swap file, the following command will create a 16GB size swap file, please modify the capacity size according to your own needs.

```
orangepi@orangepi:~$ sudo fallocate -l 16G /swapfile
```

If the Swap partition is already enabled and you run the fallback command, the following error will be reported:

fallocate: fallocate failed: Text file busy

You need to first run the `sudo swapoff /swapfile` command to close the swap partition on the system.

Note that before adding Swap memory, please ensure that the remaining space on the TF card, USB flash drive, or SSD is greater than the Swap memory capacity that needs to be added.



2) Then modify the file permissions to ensure that only the root user can read and write.

```
orange_pi@orange_pi:~$ sudo chmod 600 /swapfile
```

3) Then set this file as a swap space.

```
orange_pi@orange_pi:~$ sudo mkswap /swapfile
```

4) Then enable swap.

```
orange_pi@orange_pi:~$ sudo swapon /swapfile
```

5) After completing the above steps, the following command can be used to check if the swap memory has been successfully added.

```
orange_pi@orange_pi:~$ free -h
```

	total	used	free	shared	buff/cache	available
Mem:	15Gi	1.6Gi	9.5Gi	20Mi	4.4Gi	13Gi
Swap:	15Gi	0B	15Gi			

6) If the swap setting is still valid after restarting, please run the following command to add the corresponding configuration to the `/etc/fstab` file.

```
orange_pi@orange_pi:~$ echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
```

3. 36. Installation Method for Spark Store

The Spark App Store solves the problem of Linux users obtaining applications by integrating dispersed resources and providing massive software download and installation functions.

1) Open a browser on the system desktop of the development board.



2) Open the website below in a browser.

<https://www.spark-app.store/download>

3) Then choose the ARM64 architecture.



4) Then download the latest version of the installation package.



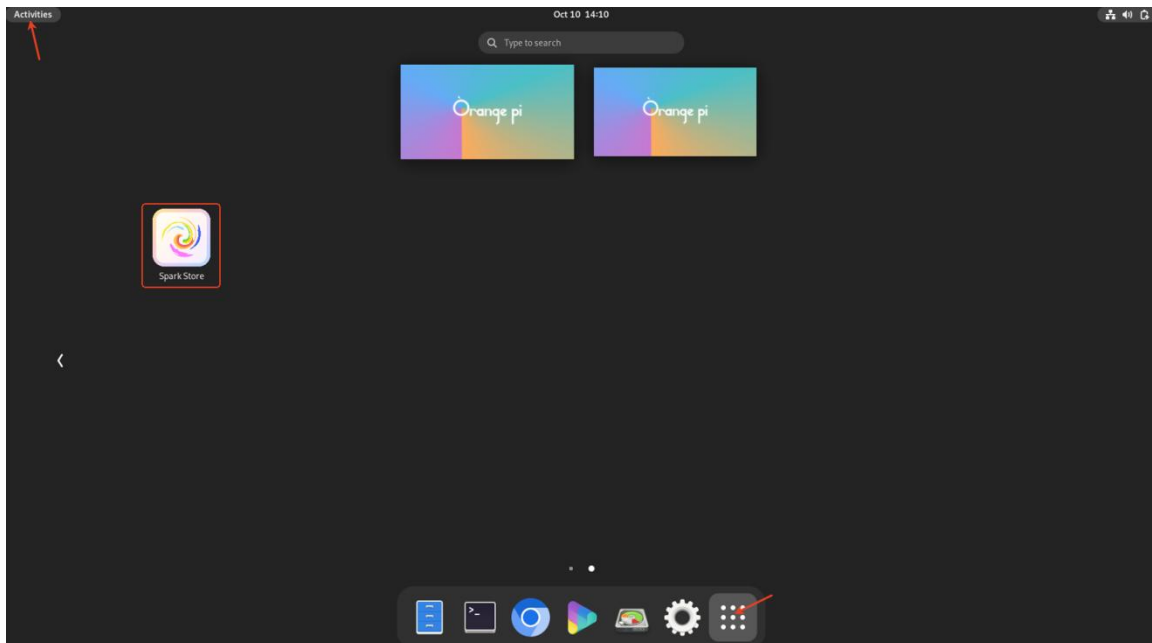
5) The downloaded installation package is shown below:

```
orangepi@orangepi:~$ ls /home/orangepi/Downloads/  
spark-store_4.8.2_arm64.deb
```

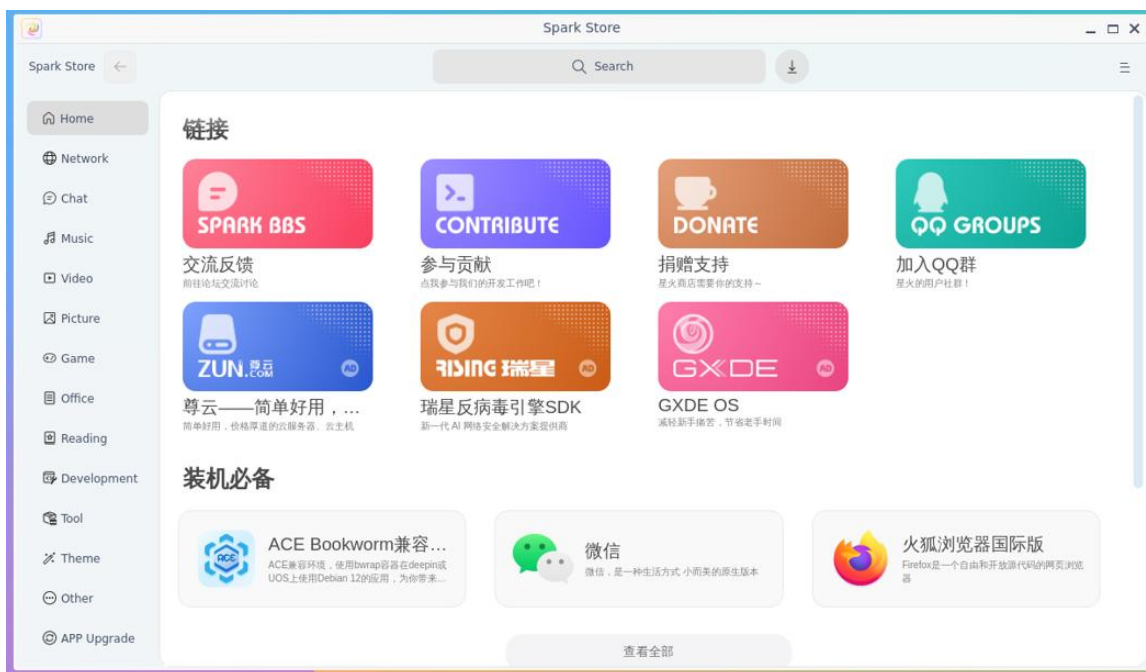
6) Then use the following command to install the Spark App Store.

```
orangepi@orangepi:~$ sudo apt install -y ./Downloads/spark-store_4.8.2_arm64.deb
```

7) The installed Spark app store is shown below:

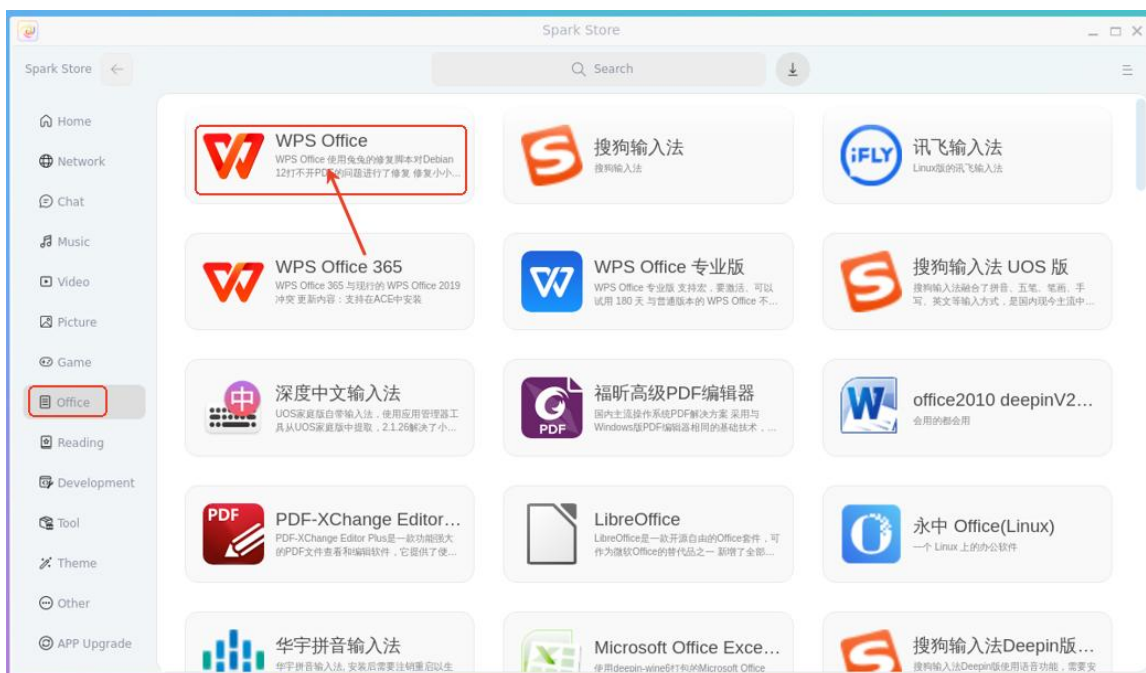


8) The interface of the Spark App Store after opening is as follows:

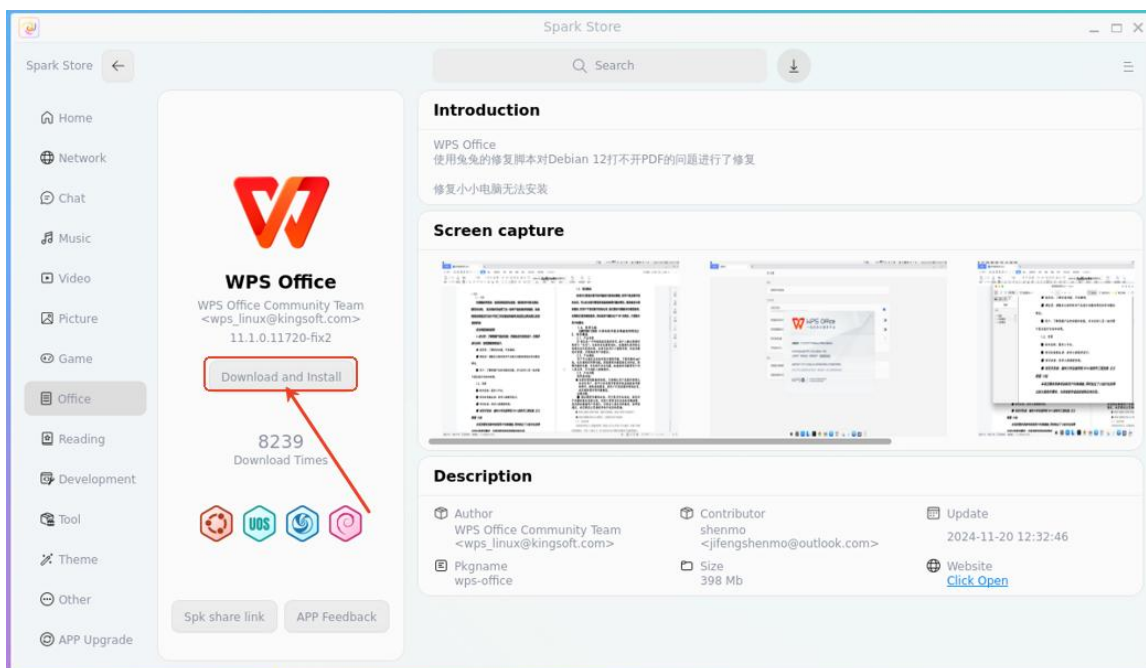


9) Then you can search for the application you want to install, such as using the following steps to install WPS Office:

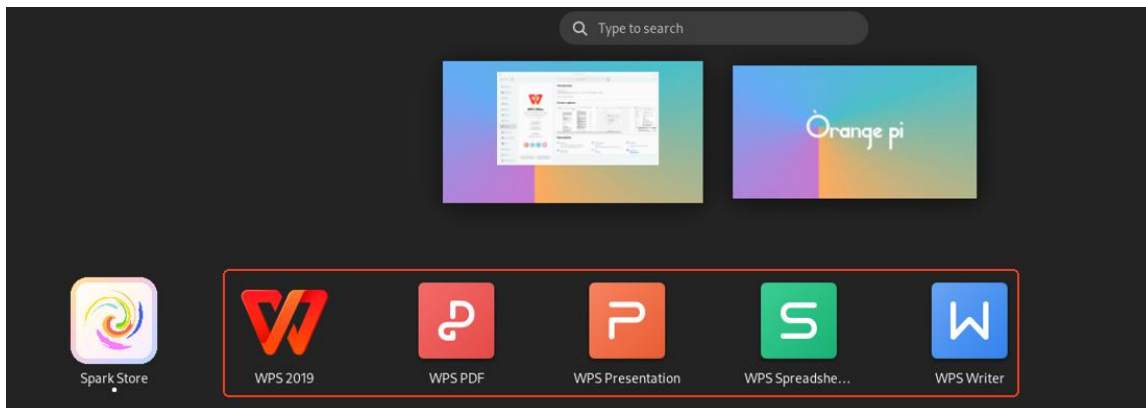
- a. Find **WPS Office** in Office and use the left mouse button to select it.



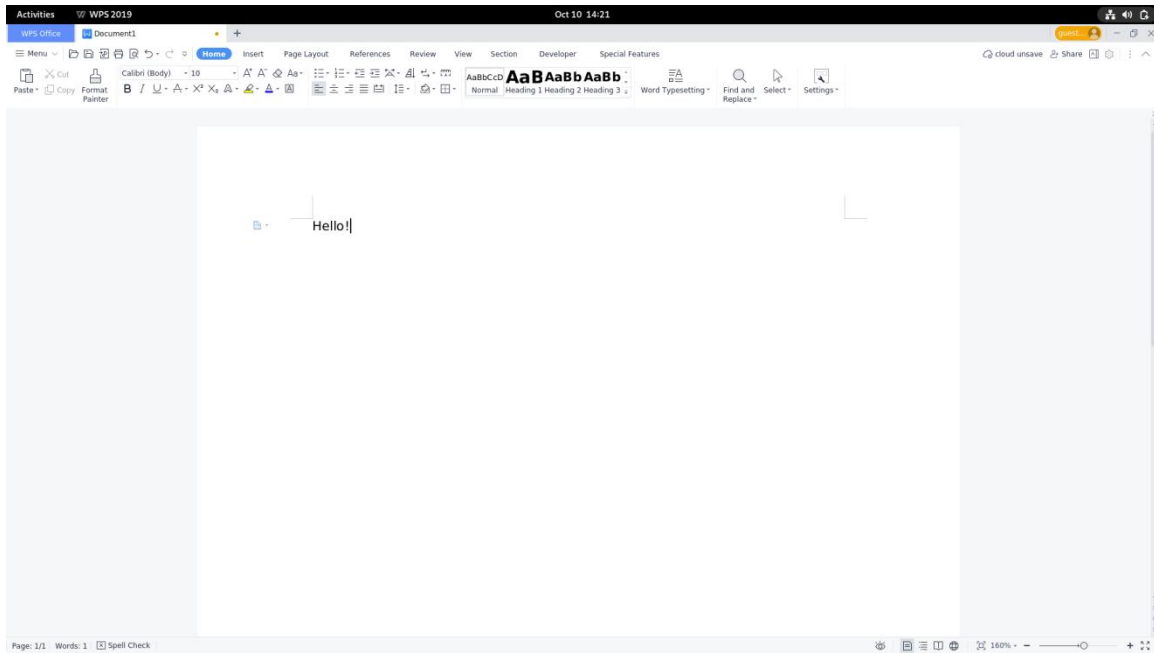
- b. Then click on **Download and Install**.



c. The installed WSP is shown below:



d. The interface of **WPS Writer** after opening is as follows:



10) Note that not all applications in the Spark App Store can be installed and used. You can try more, and for applications that cannot be used, you can only wait for the store's official repair and update.

3. 37. Usage of CIX AI Model Hub

3. 37. 1. Method for downloading CIX AI Model Hub

The CIX AI Model Hub provides a range of cutting-edge machine learning models optimized for device deployment, including numerous pre optimized and out of the box AI models. Through this model library, we can easily deploy AI models to the Orange Pi 6 Plus development board.

3. 37. 1. 1. Method of downloading through the modelscope command line

1) The warehouse address is as follows:

https://www.modelscope.cn/models/cix/ai_model_hub_25_Q3



2) Before downloading, please install ModelScope using the following command.

a. The commands in Debian12 and Ubuntu24.04 are as follows:

```
orangepi@orangepi:~$ sudo apt-get install -y python3-pip
orangepi@orangepi:~$ pip3 install modelscope --break-system-packages \
-i https://repo.huaweicloud.com/repository/pypi/simple \
--trusted-host repo.huaweicloud.com
```

b. The commands in Ubuntu22.04 are as follows:

```
orangepi@orangepi:~$ sudo apt-get install -y python3-pip
orangepi@orangepi:~$ pip3 install modelscope \
-i https://repo.huaweicloud.com/repository/pypi/simple \
--trusted-host repo.huaweicloud.com
```

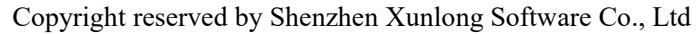
3) Then set the environment variables.

```
orangepi@orangepi:~$ export PATH="$HOME/.local/bin:$PATH"
```

4) The following command can download the complete model library. The `--local_dir` parameter will download the model library to `/home/orangepi/ai_model_hub_25_Q3`, which can be modified according to your own needs.

After downloading ai_comoded_hub_25_Q3, it requires approximately 60 GB of storage space. Please ensure that the system has sufficient available space.

```
orangepi@orangepi:~$ modelscope download --model cix/ai_model_hub_25_Q3 \
--local_dir '/home/orangepi/ai_model_hub_25_Q3'
```



5) The downloaded model library is shown below:

3. 37. 1. 2. Method of downloading through Git

https://www.modelscope.cn/models/cix/ai_model_hub_25_Q3



221



3) Then use the following command to download **ai_model_hub**.

```
orangepi@orangepi:~$ git clone \
https://www.modelscope.cn/cix/ai_model_hub_25_Q3.git
```

After downloading ai_comoded_hub_25_Q3, it requires approximately 120 GB of storage space. Please ensure that the system has sufficient available space.

4) The downloaded model library is shown below:

```
orangepi@orangepi:~$ ls ai_model_hub_25_Q3
configuration.json  datasets  demos  EULA.md  LICENSE  models  README.md
ReleaseNote.md  requirements.txt  scripts  utils
```

3. 37. 2. Installing Dependency Packages for NPU

Before testing the model, please follow the steps below to install the dependency package, otherwise the test will report an error.

1) To install Python dependency packages, the command is as follows:

```
orangepi@orangepi:~$ sudo apt-get install -y cmake
orangepi@orangepi:~$ cd ai_model_hub_25_Q3
orangepi@orangepi:~/ai_model_hub_25_Q3$ pip3 install -r requirements.txt \
--break-system-packages -i https://repo.huaweicloud.com/repository/pypi/simple \
--trusted-host repo.huaweicloud.com
```

2) Ensure that the aipu.ko kernel module has been loaded properly and that there are aipu device nodes under **/dev/**.

```
orangepi@orangepi:~$ lsmod | grep aipu
aipu                98304  0
orangepi@orangepi:~$ ls /dev/aipu
/dev/aipu
```

3) Ensure that the version of cix neo umd is 2.0.2.

```
orangepi@orangepi:~$ dpkg -l | grep cix-noe-umd
ii  cix-noe-umd      2.0.2      arm64      cix-noe-umd package
```

If the cix neo umd version is not 2.0.2, please download and use the latest



version of the Linux image for the next testing.

3. 37. 3. Method for installing onnxruntime_zhouyi

The onnxruntime_zhouyi package is only used by some models. If there is no indication, this package does not need to be installed for now.

1) Ensure that the version of the cix npu onnxruntime package is 1.1.0.

```
orangeipi@orangeipi:~$ dpkg -l | grep cix-npu-onnxruntime
ii  cix-npu-onnxruntime  1.1.0  arm64  cix-npu-onnxruntime package
```

If the cix npu onnx runtime version is not 1.1.0, please download and use the latest version of the Linux image before proceeding with the next testing.

2) Uninstall the onnxruntime package. This package and onnxruntime_zhouyi can only be chosen from two options and cannot be installed simultaneously.

```
orangeipi@orangeipi:~$ pip3 uninstall -y onnxruntime --break-system-packages
```

3) Install onnxruntime_zhouyi with the following command:

```
orangeipi@orangeipi:~$ pip3 install \
/usr/share/cix/pypi/onnxruntime_zhouyi-1.20.0-cp311-cp311-linux_aarch64.whl \
--break-system-packages --force-reinstall \
-i https://repo.huaweicloud.com/repository/pypi/simple \
--trusted-host repo.huaweicloud.com
```

4) After installation, please make sure that the corresponding package version is the same as the one below.

```
orangeipi@orangeipi:~$ pip3 list | grep Zhouyi
ZhouyiOperators-x2  25.4.23
orangeipi@orangeipi:~$ pip3 list | grep onnxruntime-zhouyi
onnxruntime-zhouyi  1.20.0
orangeipi@orangeipi:~$ dpkg -l | grep cix-noe-umd
ii  cix-noe-umd  2.0.2  arm64  cix-noe-umd package
```

3. 37. 4. List of Tested Models

Model Name	CPU inference	NPU reasoning
Audio/Speech_Recognition		



onnx_whisper_medium_multilingual	OK	OK
onnx_whisper_small_multi_language	OK	OK
onnx_whisper_tiny_multi_language	OK	OK
ComputeVision/BEV		
onnx_BEV_RoadSeg	OK	OK
ComputeVision/Body_Analysis		
onnx_age_googlenet	OK	OK
onnx_gender_googlenet	OK	OK
ComputeVision/Depth_Estimation		
onnx_depth_anything_v2	OK	OK
onnx_fsre_depth	OK	OK
onnx_MiDaS_v2	OK	OK
ComputeVision/Face_Detection		
onnx_centerface	OK	OK
onnx_facenet	OK	OK
onnx_rfb_320	OK	OK
ComputeVision/Face_Recognition		
onnx_scrfd_arcface	OK	OK
ComputeVision/Image_Classification		
onnx_inception_v3	OK	OK
onnx_mobilenet_v2	OK	OK
onnx_mobilenet_v2_12_int8	OK	OK
onnx_mobilenet_v2_12_qdq	OK	OK
onnx_resnet_v1_101	OK	OK
onnx_resnet_v1_12_int8	OK	OK
onnx_resnet_v1_12_qdq	OK	OK
onnx_resnet_v1_50	OK	OK
onnx_vgg_16	OK	OK
onnx_vgg16_12_int8	OK	OK
onnx_vgg16_12_qdq	OK	OK
onnx_vit-base-16-224	OK	OK
ComputeVision/Image_Enhancement		
onnx_AGLNet	OK	OK
onnx_nighttime_dehaze	OK	OK
onnx_pairlie	OK	OK



ComputeVision/Lane_Detection		
onnx_lanenet	OK	OK
onnx_LSTR	OK	OK
onnx_Ultra_Fast_Lane_Detection	OK	OK
onnx_Ultra_Fast_Lane_Detection_v2	OK	OK
ComputeVision/Object_Detection		
onnx_resnet50_ssd300	OK	OK
onnx_retinanet	OK	OK
onnx_vehicle_detection	OK	OK
onnx_yolov12_m	OK	OK
onnx_yolov12_n	OK	OK
onnx_yolo_v3	OK	OK
onnx_yolov8_l	OK	OK
onnx_yolov8l_worldv2	OK	OK
onnx_yolov8_n	OK	OK
onnx_yolox_l	OK	OK
onnx_yolox_m	OK	OK
onnx_yolox_s	OK	OK
torch_centernet_resnet50	OK	OK
ComputeVision/OCR		
onnx_crnn	OK	OK
onnx_PP_OCRv4	OK	OK
ComputeVision/Pose_Estimation		
onnx_handpose	OK	OK
onnx_hrnet_pose	OK	OK
onnx_openpose	OK	OK
onnx_yolov8s_pose	OK	OK
ComputeVision/Semantic_Segmentation		
onnx_BiSeNet	OK	OK
onnx_deeplab_v3	OK	OK
onnx_deeplabv3_plus	OK	OK
onnx_duc	OK	OK
onnx_fcn_resnet50	OK	OK
onnx_pphumanseg	OK	OK
onnx_road_segmentation	OK	OK



onnx_sam2_1_tiny	OK	OK
onnx_u2net	OK	OK
onnx_yolov8_segmentation	OK	OK
torch_fast_scnn	OK	OK
ComputeVision/Super_Resolution		
onnx_real_esrgan	OK	OK
onnx_super_resolution	OK	OK
onnx_vdsr	OK	OK

3. 37. 5. Testing Method for Speech_Secognition Class Models in Audio

Before testing the model, please install the dependency packages according to the instructions in the section on [installing NPU dependencies](#), otherwise the test will report errors.

3. 37. 5. 1. onnx_whisper_medium_multilingual Case

The onnx_whisper_medium_multilingual speech recognition case is used to convert audio files into text. The testing steps are as follows:

1) Go to the onnx_hisper_cedium_maultilingual directory.

```
orange@orange:~$ cd \
ai_model_hub_25_Q3/models/Audio/Speech_Recognition/onnx_whisper_medium_multilingual
```

2) The command to use CPU for inference is as follows:

```
orange@orange:~/ai_model_hub_25_Q3/models/Audio/Speech_Recognition/onnx_whisper_medium_multilingual$ python3 inference_onnx.py
```

3) After the CPU completes the inference, a test_audio_onnx.txt file will be generated under the output, which contains the result of converting test_data/test_audio.flac to text.

```
orange@orange:~/ai_model_hub_25_Q3/models/Audio/Speech_Recognition/onnx_whisper_medium_multilingual$ cat output/test_audio_onnx.txt
```

They regain their apartment, apparently without disturbing the household of Gainwell.

4) The command to use NPU for inference is as follows:

```
orange@orange:~/ai_model_hub_25_Q3/models/Audio/Speech_Recognition/onnx_whisper_medium_multilingual$ python3 inference_npu.py --backend npu \
```



```

--encoder_model_path whisper_medium_multilingual_encoder.cix \
--decoder_model_path whisper_medium_multilingual_decoder.cix
.....
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 1.
npu: noe_create_job success
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 5.
Output tensor count is 2.
npu: noe_create_job success
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success

```

5) After the NPU inference is completed, a test_audio_npu.exe file will be generated under the output, which contains the result of converting test_data/test_audio.flac to text.

```

orange@orange:~/ai_model_hub_25_Q3/models/Audio/Speech_Recognition/onnx_whisper_medium_multilingual$ cat output/test_audio_npu.txt
They regain their apartment, apparently without disturbing the household of Gainwell.

```

3. 37. 5. 2. onnx_whisper_small_multi_language Case

The onnx_whisper_small_multi_language speech recognition case is used to convert audio files into text. The testing steps are as follows:

1) Go to the onnx_hisper_stmall_rulti_language directory.

```

orange@orange:~$ cd \
ai_model_hub_25_Q3/models/Audio/Speech_Recognition/onnx_whisper_small_multi_language

```



2) The command to use CPU for inference is as follows, which by default converts test_data/2.wav audio to text.

```
orange@orange:~/ai_model_hub_25_Q3/models/Audio/Speech_Recognition/onnx_whisper_small_multi_language$ python3 inference_onnx.py
.....
[[50258 50260 50359 50363 4200 7248 232 18637 245 32585 8945 32585
 5661 2257 47857 34910 34629 1546 24113 6306 222 4799 117 5916
 11284 4479 118 4479 118 22220 244 23575 29454 11 220 18573
 223 18573 223 18573 223 10928 3581 24113 11016 8350 4511 50257]]
['看羊狗跑前跑後一隻精飛的山雀惹得它旺旺搖幾聲, 翁翁翁地在山間回到']
ok.....
```

3) The command to use NPU for inference is as follows, which by default converts test_data/1.wav audio to text.

```
orange@orange:~/ai_model_hub_25_Q3/models/Audio/Speech_Recognition/onnx_whisper_small_multi_language$ python3 inference_npu.py
.....
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 1.
npu: noe_create_job success
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success
[[50258 50260 50359 50363 3405 26372 5157 17 6939 9356 6890 5708
 44237 19425 27905 2862 223 16206 14696 227 21030 96 16206 47402
 49500 231 21001 39553 7391 255 5155 46563 38291 224 21001 4035
 18214 45277 43362 34719 9890 96 27408 50257]]
['1993年2月23日上午四川省安悅縣安源鄉五村彭家姑嫂五人進城構置衣服']
ok.....
```

3. 37. 5. 3. onnx_whisper_tiny_multi_language Case

The onnx_whisper_tiny_multi_language speech recognition case is used to convert



audio files into text. The testing steps are as follows:

1) Go to the onnx_hisper_tinymulti_language directory.

```
orangePi@orangePi:~$ cd \
ai_model_hub_25_Q3/models/Audio/Speech_Recognition/onnx_whisper_tiny_multi_language
```

2) The command to use CPU for inference is shown below, which by default converts test_data/1.wav audio to text.

```
orangePi@orangePi:~/ai_model_hub_25_Q3/models/Audio/Speech_Recognition/onnx_whisper_tiny_multi_language$ python3 inference_onnx.py --audio test_data/1.wav --onnx_path \
model/whisper_tiny_multilang_encoder.onnx
.....
test_data/1.wav
[[50361 220 3588 4438 1541 29993 19550 21596 1546 34592 7626 1543
50258 50260 50359 50363 3405 26372 5157 17 6939 9356 6890 5708
44237 11 19425 27905 2862 223 16206 6939 16853 123 16206 14812
15106 21001 39553 7391 255 5155 46563 38291 224 21001 4035 36700
45277 1787 253 34719 9890 96 27408 1543 50257]]
['1993年2月23日上午,四川省安月线安元相五村彭家姑嫂五人进城够置衣服。']
```

3) The command to use NPU for inference is as follows, which by default converts test_data/1.wav audio to text.

```
orangePi@orangePi:~/ai_model_hub_25_Q3/models/Audio/Speech_Recognition/onnx_whisper_small_multi_language$ python3 inference_npu.py
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 1.
test_data/1.wav
npu: noe_create_job success
npu: noe_clean_job success
[[50361 220 3588 4438 1541 29993 19550 21596 1546 34592 7626 1543
50258 50260 50359 50363 3405 26372 5157 17 6939 9356 6890 5708
44237 11 19425 27905 2862 223 16206 6939 21030 96 16206 14812
15106 21001 39553 7391 255 5155 46563 38291 224 21001 4035 11957
114 11336 1787 253 30118 9890 96 27408 1543 50257]]
```




```
['1993年2月23日上午,四川省安月縣安元相五村彭家姑嫂五人競成够治衣服。']
```

```
npu: noe_unload_graph success
```

```
npu: noe_deinit_context success
```

3. 37. 6. Testing Methods for BEV Models in ComputeVision

Before testing the model, please install the dependency packages according to the instructions in the section on [installing NPU dependencies](#), otherwise the test will report errors.

3. 37. 6. 1. onnx_BEV_RoadSeg Case

The core function of the BEV_RoadSeg model is to segment the drivable road area from the perspective of the onboard camera and convert the segmentation results into a bird's-eye view, providing vehicles with a top-down, easy to understand "road map".

1) Enter the folder where the case is located.

```
orange@orange:~$ cd \
```

```
ai_model_hub_25_Q3/models/ComputeVision/BEV/onnx_BEV_RoadSeg
```

2) The command to use CPU for inference is as follows, using the image in the test_data folder by default.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/BEV/onnx_BEV_RoadSeg
```

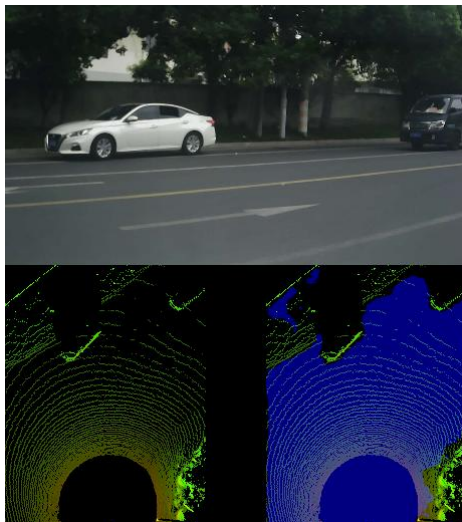
```
$ python3 inference_onnx.py
```

3) The inference result is as follows:

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/BEV/onnx_BEV_RoadSeg
```

```
$ ls output/onnx_1.png
```

```
output/onnx_1.png
```



4) The command to use NPU for inference is as follows, and the default image used is the image in the test_data folder.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/BEV/onnx_BEV_RoadSeg
$ python3 inference_npu.py
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 1.
npu: noe_create_job success
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success
```

5) The inference result is as follows:

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/BEV/onnx_BEV_RoadSeg
$ ls output/npu_1.png
output/npu_1.png
```



3. 37. 7. Testing Methods for Body-Analysis Models in ComputeVision

Before testing the model, please install the dependency packages according to the instructions in the section on [installing NPU dependencies](#), otherwise the test will report errors.

3. 37. 7. 1. onnx_age_googlenet Case

This is an inference case using the GoogleNet model for age segmentation prediction.

1) Enter the folder where the case is located.

```
orangepi@orangepi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Body_Analysis/onnx_age_googlenet
```

2) The command to use CPU for inference is as follows. The default image used is the image in the test_data folder, and the inference result is 25-32 years old.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Body_Analysis/onnx_age_googl
enet$ python3 inference_onnx.py
.....
test_data/test.jpg
```



```
(25-32): 0.99993753
```

3) The command to use NPU for inference is as follows. The default image used is the image in the test_data folder, and the inference result is 25-32 years old.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Body_Analysis/onnx_age_googlenet$ python3 inference_npu.py
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 1.
npu: noe_create_job success
test_data/test.jpg
(25-32): 0.9999454
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success
```

3. 37. 7. 2. onnx_gender_googlenet Case

This case is used to infer the gender of the characters in the picture.

1) Enter the folder where the case is located.

```
orangepi@orangepi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Body_Analysis/onnx_gender_googlenet
```

2) The command to use CPU for inference is shown below. This case is used to infer the gender of the person in the test_data/test.jpg image. From the output, it can be seen that the recognition result is male.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Body_Analysis/onnx_gender_googlenet$ python3 inference_onnx.py
.....
test_data/test.jpg
Male: 0.99999356
```

3) The command to use NPU for inference is shown below. This case is used to infer the



gender of the character in the test_data/test.jpg image. From the output, it can be seen that the recognition result is male.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Body_Analysis/onnx_gender_goglenet$ python3 inference_npu.py
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 1.
npu: noe_create_job success
test_data/test.jpg
Male: 0.9999999
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success
```

3. 37. 8. Testing Methods for Depth-Estimation Class Models in ComputeVision

Before testing the model, please install the dependency packages according to the instructions in the section on [installing NPU dependencies](#), otherwise the test will report errors.

3. 37. 8. 1. onnx_depth_anything_v2 Case

This case involves estimating depth information from a single RGB image - that is, predicting the distance between each pixel in the image and the camera based on a regular 2D image, and generating a corresponding depth map.

1) Enter the folder where the case is located.

```
orangepi@orangepi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Depth_Estimation/onnx_depth_anything_v2
```

2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Depth_Estimation/onnx_depth_anything_v2$ python3 inference_onnx.py
```



3) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Depth_Estimation/onnx_depth_
anything_v2$ ls output/
onnx_2.jpeg  onnx_dog.jpg
```



4) The command to use NPU for inference is as follows, using images from the test_data folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Depth_Estimation/onnx_depth_
anything_v2$ python3 inference_npu.py
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 1.
npu: noe_create_job success
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success
```

5) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Depth_Estimation/onnx_depth_
anything_v2$ ls output
npu_2.jpeg  npu_dog.jpg
```



3. 37. 8. 2. onnx_fsre_depth Case

FSRE Depth is an advanced framework designed specifically for high-precision monocular depth estimation. Its core value lies in the ability to generate accurate depth maps from a single RGB image through multi-scale feature fusion and robust enhancement techniques, without relying on additional hardware such as stereo vision or LiDAR.

1) Enter the folder where the case is located.

```
orange@orange:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Depth_Estimation/onnx_fsre_depth
```

2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Depth_Estimation/onnx_fsre_de
pth$ python3 inference_onnx.py
```



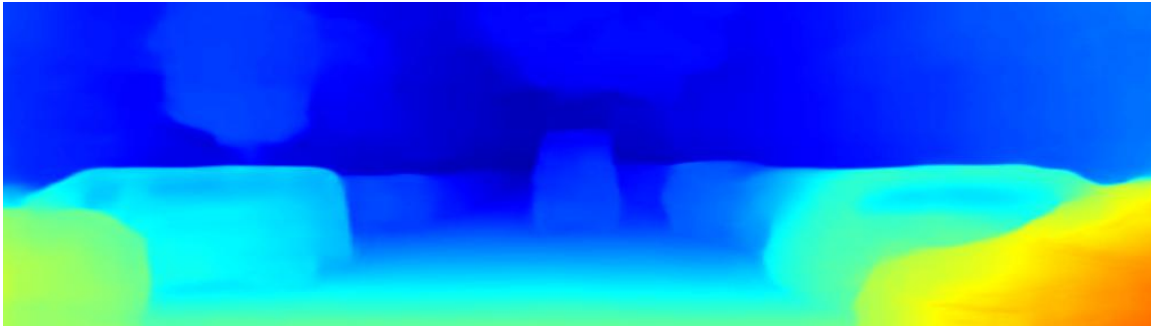
3) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Depth_Estimation/onnx_fsre_de
```




```
pth$ ls output
```

```
onnx_1.png
```



4) The command to use NPU for inference is as follows, using images from the test_data folder for inference.

```
orangeapi@orangeapi:~/ai_model_hub_25_Q3/models/ComputeVision/Depth_Estimation/onnx_fsre_de
```

```
pth$ python3 inference_npu.py
```

```
npu: noe_init_context success
```

```
npu: noe_load_graph success
```

```
Input tensor count is 1.
```

```
Output tensor count is 1.
```

```
npu: noe_create_job success
```

```
npu: noe_clean_job success
```

```
npu: noe_unload_graph success
```

```
npu: noe_deinit_context success
```

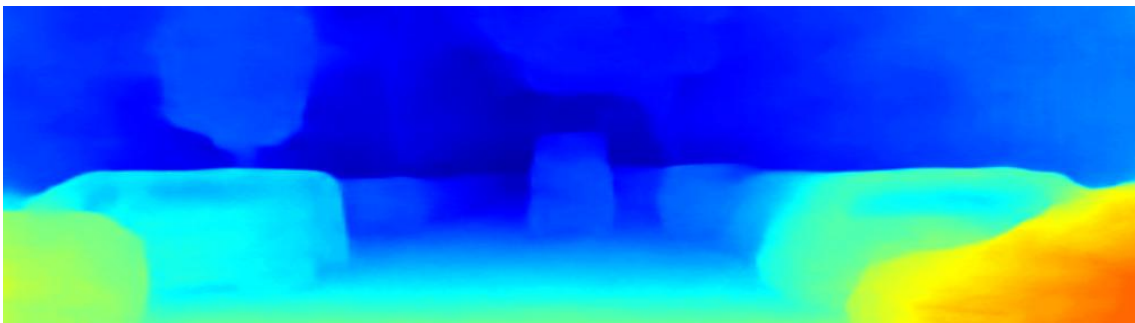


5) The images generated by inference will be saved in the output directory.

```
orangeapi@orangeapi:~/ai_model_hub_25_Q3/models/ComputeVision/Depth_Estimation/onnx_fsre_de
```

```
pth$ ls output
```

```
npu_1.png
```



3. 37. 8. 3. onnx_MiDaS_v2 Case

MiDaS2v2 is a universal monocular depth estimation model that can convert a regular RGB image into a high-quality depth map.

1) Enter the folder where the case is located.

```
orangePi@orangePi:~$ cd \  
ai_model_hub_25_Q3/models/ComputeVision/Depth_Estimation/onnx_MiDaS_v2
```

2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

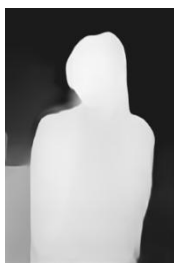
```
orangePi@orangePi:~/ai_model_hub_25_Q3/models/ComputeVision/Depth_Estimation/onnx_MiDaS_v2$ python3 inference_onnx.py  
.....  
initialize  
loading model...  
  processing ./test_data/1.jpg  
Inference time: 202.13 ms  
  processing ./test_data/2.jpg  
Inference time: 196.28 ms  
  processing ./test_data/3.jpg  
Inference time: 189.92 ms  
Finished
```

3) The images generated by inference will be saved in the output directory.

```
orangePi@orangePi:~/ai_model_hub_25_Q3/models/ComputeVision/Depth_Estimation/onnx_MiDaS_v2$ ls output
```



onnx_1.png onnx_2.png onnx_3.png



4) The command to use NPU for inference is as follows, using images from the test_data folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Depth_Estimation/onnx_MiDaS_v2$ python3 inference_npu.py
initialize
loading model...
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 1.
npu: noe_create_job success
  processing ./test_data/1.jpg
Inference time: 4.73 ms
  processing ./test_data/2.jpg
Inference time: 6.33 ms
  processing ./test_data/3.jpg
Inference time: 6.52 ms
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success
finished
```

5) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Depth_Estimation/onnx_MiDaS_v2$ ls output
npu_1.png npu_2.png npu_3.png
```



3. 37. 9. Testing Methods for Face_Selection Class Models in ComputeVision

Before testing the model, please install the dependency packages according to the instructions in the section on [installing NPU dependencies](#), otherwise the test will report errors.

3. 37. 9. 1. onnx_centerface Case

This case is used for real-time face detection and localization - finding the positions of all faces from an image and marking them.

1) Enter the folder where the case is located.

```
orangepi@orangepi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Face_Detection/onnx_centerface
```

2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Face_Detection/onnx_centerface
$ python3 inference_onnx.py
.....
cpu times = 0:00:00.169547
cpu times = 0:00:00.178297
```

3) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Face_Detection/onnx_centerface
$ ls output
onnx_test1.JPEG  onnx_test2.JPEG
```



4) The command to use NPU for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Face_Detection/onnx_centerface
$ python3 inference_npu.py
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 4.
npu: noe_create_job success
npu times = 0:00:00.009426
npu times = 0:00:00.008288
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success
```

5) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Face_Detection/onnx_centerface
$ ls output
npu_test1.JPEG  npu_test2.JPEG
```



3.37.9.2. onnx_facenet Case

This case is used for facial recognition, mainly for facial feature extraction and comparison.

- 1) Enter the folder where the case is located.

```
orangeipi@orangeipi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Face_Detection/onnx_facenet
```

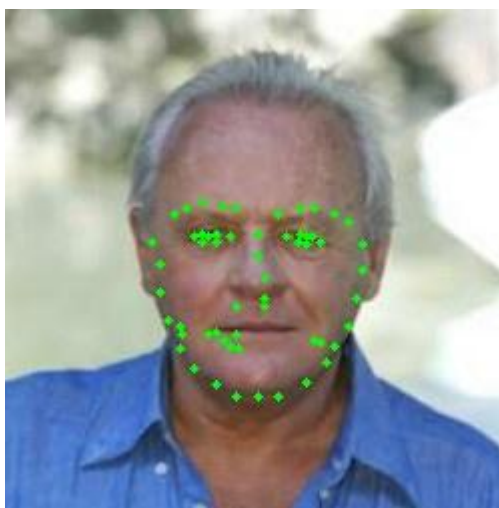
- 2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

```

orangeapi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Face_Detection/onnx_facenet$
python3 inference_onnx.py
.....
['./test_data/Anthony_Hopkins.jpg', './test_data/test2.JPEG', './test_data/test1.JPEG']
./test_data/Anthony_Hopkins.jpg
detect 1 faces
100% ██████████ 1/1 [00:03<00:00, 3.14s/it]
./test_data/test2.JPEG
detect 6 faces
.....
100% ██████████ 1/1 [00:03<00:00, 3.13s/it]
./test_data/test1.JPEG
detect 7 faces
.....
100% ██████████ 1/1 [00:03<00:00, 3.11s/it]
```

- 3) The images generated by inference will be saved in the output directory.

```
orangeipi@orangeipi:~/ai_model_hub_25_Q3/models/ComputeVision/Face_Detection/onnx_facenet$ ls output
onnx_Anthony_Hopkins.jpg  onnx_test1.JPEG  onnx_test2.JPEG
```

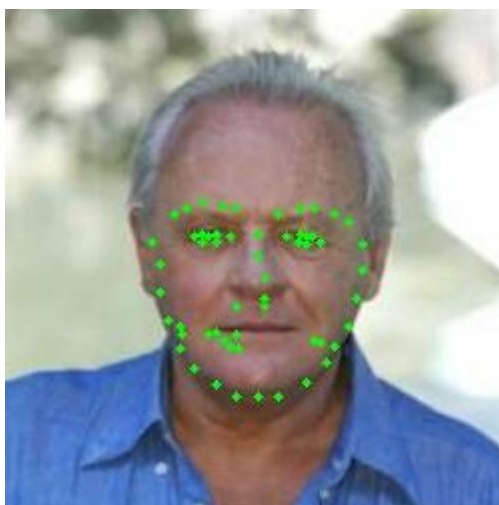



4) The command to use NPU for inference is as follows, using images from the test_data folder for inference.

```
oorange@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Face_Detection/onnx_facenet$
```




```
oorangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Face_Detection/onnx_face
net$ ls output
npu_Anthony_Hopkins.jpg  npu_test1.JPEG  npu_test2.JPEG
```



3. 37. 9. 3. onnx_rfb_320 Case

This case is used to quickly detect the position of faces in images and output bounding boxes.

1) Enter the folder where the case is located.

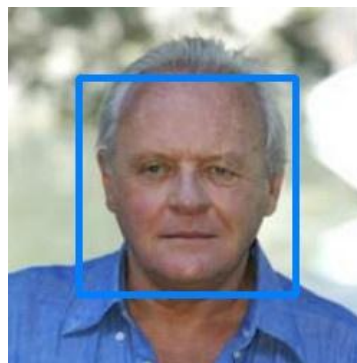
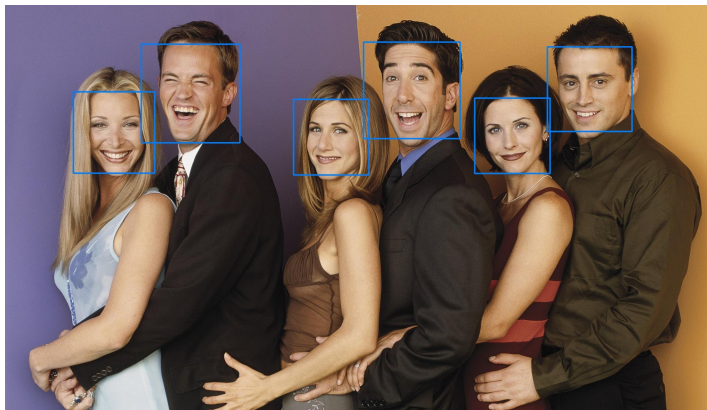
```
orangeapi@orangeapi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Face_Detection/onnx_rfb_320
```

2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

```
orangeapi@orangeapi:~/ai_model_hub_25_Q3/models/ComputeVision/Face_Detection/onnx_rfb_320$ python3 inference_onnx.py
```

3) The images generated by inference will be saved in the output directory.

```
orangeapi@orangeapi:~/ai_model_hub_25_Q3/models/ComputeVision/Face_Detection/onnx_rfb_320$ ls output
onnx_Anthony_Hopkins.jpg  onnx_test1.JPEG  onnx_test2.JPEG
```



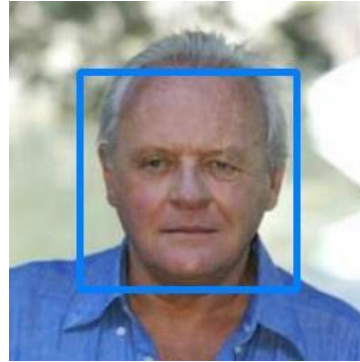
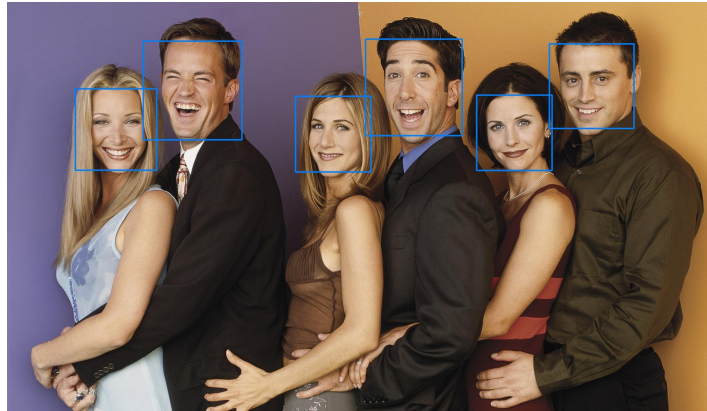


4) The command to use NPU for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Face_Detection/onnx_rfb_320$ python3 inference_npu.py
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 2.
npu: noe_create_job success
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success
```

5) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Face_Detection/onnx_rfb_320$ ls output
npu_Anthony_Hopkins.jpg  npu_test1.JPEG  npu_test2.JPEG
```





3. 37. 10. Testing Method for Face_Secognition Class Models in ComputeVision

Before testing the model, please install the dependency packages according to the instructions in the section on [installing NPU dependencies](#), otherwise the test will report errors.

3. 37. 10. 1. onnx_scrfd_arcface Case

This case combines facial detection and facial recognition, which can detect the position of the face and perform facial identity recognition.

1) Enter the folder where the case is located.

```
orangepi@orangepi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Face_Recognition/onnx_scrfd_arcface
```

2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Face_Recognition/onnx_scrfd_arcface$ python3 inference_onnx.py --det_onnx_path ./model/det_10g.onnx \
--rec_onnx_path ./model/w600k_r50.onnx --faces-dir ./datasets/faces --image_path test_data
```

3) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Face_Recognition/onnx_scrfd_arcface$ ls output
onnx_test1.png  onnx_test2.png
```




4) The command to use NPU for inference is as follows, using images from the test_data folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Face_Recognition/onnx_scrfd_a
rface$ python3 inference_npu.py --det_model_path ./scrfd.cix --rec_model_path ./arcface.cix \
--faces-dir ./datasets/faces --image_path test_data
npu: noe_init_context success
```



```
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 9.
npu: noe_create_job success
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 1.
npu: noe_create_job success
./datasets/faces/Phoebe.png
./datasets/faces/Chandler.png
./datasets/faces/Monica.png
./datasets/faces/Rachel.png
./datasets/faces/Ross.png
./datasets/faces/Joey.png
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success
```

5) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Face_Recognition/onnx_scrfd_a
rcface$ ls output
npu_test1.png  npu_test2.png
```




3.37.11. Testing Method for Image_Classification Class Models in ComputeVision

Before testing the model, please install the dependency packages according to the instructions in the section on [installing NPU dependencies](#), otherwise the test will report errors.



3. 37. 11. 1. onnx_inception_v3 Case

This case is used to classify images.

1) Enter the folder where the case is located.

```
orangePi@orangePi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_inception_v3/
```

2) The command to use CPU for inference is shown below. In this case, the images in the test_data folder will be classified, and the output shows that the model successfully recognized various objects.

```
orangePi@orangePi:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_inception_v3$ python3 inference_onnx.py --images test_data --onnx_path \
model/inception_v3_Opset16_sim.onnx
.....
image path : test_data/ILSVRC2012_val_00037133.JPEG
ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus
image path : test_data/ILSVRC2012_val_00002899.JPEG
rock python, rock snake, Python sebae
image path : test_data/ILSVRC2012_val_00045790.JPEG
Yorkshire terrier
image path : test_data/ILSVRC2012_val_00021564.JPEG
coucal
image path : test_data/ILSVRC2012_val_00024154.JPEG
Ibizan hound, Ibizan Podenco
```

3) The command to use NPU for inference is shown below. In this case, the images in the test_data folder will be classified, and the output shows that the model successfully recognized various objects.

```
orangePi@orangePi:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_inception_v3$ python3 inference_npu.py --images test_data --model_path inception_v3.cix
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 1.
```



```
npu: noe_create_job success
image path : test_data/ILSVRC2012_val_00037133.JPEG
ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus
image path : test_data/ILSVRC2012_val_00002899.JPEG
rock python, rock snake, Python sebae
image path : test_data/ILSVRC2012_val_00045790.JPEG
Yorkshire terrier
image path : test_data/ILSVRC2012_val_00021564.JPEG
coucal
image path : test_data/ILSVRC2012_val_00024154.JPEG
Ibizan hound, Ibizan Podenco
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success
```

3. 37. 11. 2. onnx_mobilenet_v2 Case

This case is used to classify images.

1) Enter the folder where the case is located.

```
orange@orange:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_mobilenet_v2
```

2) The command to use CPU for inference is shown below. In this case, the images in the test_data folder will be classified, and the output shows that the model successfully recognized various objects.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_mobilenet_v2$ python3 inference_onnx.py --images test_data --onnx_path model/mobilenetv2-7.onnx
.....
image path : test_data/ILSVRC2012_val_00037133.JPEG
ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus
image path : test_data/ILSVRC2012_val_00002899.JPEG
rock python, rock snake, Python sebae
image path : test_data/ILSVRC2012_val_00045790.JPEG
Yorkshire terrier
```



```
image path : test_data/ILSVRC2012_val_00021564.JPEG
coucal
image path : test_data/ILSVRC2012_val_00024154.JPEG
Ibizan hound, Ibizan Podenco
```

3) The command to use NPU for inference is shown below. In this case, the images in the test_data folder will be classified, and the output shows that the model successfully recognized various objects.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_mob
ilenet_v2$ python3 inference_npu.py --images test_data --model_path mobilenet_v2.cix
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 1.
npu: noe_create_job success
image path : test_data/ILSVRC2012_val_00037133.JPEG
ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus
image path : test_data/ILSVRC2012_val_00002899.JPEG
boa constrictor, Constrictor constrictor
image path : test_data/ILSVRC2012_val_00045790.JPEG
Yorkshire terrier
image path : test_data/ILSVRC2012_val_00021564.JPEG
coucal
image path : test_data/ILSVRC2012_val_00024154.JPEG
Ibizan hound, Ibizan Podenco
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success
```

3. 37. 11. 3. onnx_mobilenet_v2_12_int8 Case

This case is used to classify images.

1) Enter the folder where the case is located.

```
orangepi@orangepi:~$ cd \
```



```
ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_mobilenet_v2_12_int8
```

2) The command to use CPU for inference is shown below. In this case, the images in the test_data folder will be classified, and the output shows that the model successfully recognized various objects.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_mobilenet_v2_12_int8$ python3 inference_onnx.py --EP CPU
image path : ./test_data/ILSVRC2012_val_00024154.JPEG
Ibizan hound, Ibizan Podenco
image path : ./test_data/ILSVRC2012_val_00045790.JPEG
Yorkshire terrier
image path : ./test_data/ILSVRC2012_val_00002899.JPEG
rock python, rock snake, Python sebae
image path : ./test_data/ILSVRC2012_val_00021564.JPEG
coucal
image path : ./test_data/ILSVRC2012_val_00037133.JPEG
ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus
```

3) The command to use NPU for inference is shown below. In this case, the images in the test_data folder will be classified, and the output shows that the model successfully recognized various objects.

- a. Install the onnxruntime_zhuyi package. Please refer to the instructions in the section on [installing onnxruntime_zhuyi for installation methods](#).
- b. Set the environment variables.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_mobilenet_v2_12_int8$ export \
LD_LIBRARY_PATH=/usr/share/cix/lib/onnxruntime:$LD_LIBRARY_PATH
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_mobilenet_v2_12_int8$ export OPERATOR_PATH=/usr/share/cix/lib/onnxruntime/operator
```

- c. Run the following command for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_mobilenet_v2_12_int8$ python3 inference_onnx.py --EP NPU
image path : ./test_data/ILSVRC2012_val_00024154.JPEG
Ibizan hound, Ibizan Podenco
image path : ./test_data/ILSVRC2012_val_00045790.JPEG
```



```

Yorkshire terrier
image path : ./test_data/ILSVRC2012_val_00002899.JPEG
rock python, rock snake, Python sebae
image path : ./test_data/ILSVRC2012_val_00021564.JPEG
coucal
image path : ./test_data/ILSVRC2012_val_00037133.JPEG
ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus
Total errors: 0,  warnings: 0

```

3. 37. 11. 4. onnx_mobilenet_v2_12_qdq Case

This case is used to classify images.

1) Enter the folder where the case is located.

```

orangeypi@orangeypi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_mobilenet_v2_12_qdq

```

2) The command to use CPU for inference is shown below. In this case, the images in the test_data folder will be classified, and the output shows that the model successfully recognized various objects.

```

orangeypi@orangeypi:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_mobilenet_v2_12_qdq$ python3 inference_onnx.py --EP CPU
image path : ./test_data/ILSVRC2012_val_00024154.JPEG
Ibizan hound, Ibizan Podenco
image path : ./test_data/ILSVRC2012_val_00045790.JPEG
Yorkshire terrier
image path : ./test_data/ILSVRC2012_val_00002899.JPEG
rock python, rock snake, Python sebae
image path : ./test_data/ILSVRC2012_val_00021564.JPEG
coucal
image path : ./test_data/ILSVRC2012_val_00037133.JPEG
ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus

```

3) The command to use NPU for inference is shown below. In this case, the images in the test_data folder will be classified, and the output shows that the model successfully



recognized various objects.

- a. Install the onnxruntime_zhoyi package. Please refer to the instructions in the section on [installing onnxruntime_zhoyi for installation methods](#).
- b. Set the environment variables.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_mob
ilenet_v2_12_qdq$ export \
LD_LIBRARY_PATH=/usr/share/cix/lib/onnxruntime:$LD_LIBRARY_PATH
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_mob
ilenet_v2_12_qdq$ export OPERATOR_PATH=/usr/share/cix/lib/onnxruntime/operator
```

- c. Run the following command for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_mob
ilenet_v2_12_qdq$ python3 inference_onnx.py --EP NPU
image path : ./test_data/ILSVRC2012_val_00024154.JPEG
Ibizan hound, Ibizan Podenco
image path : ./test_data/ILSVRC2012_val_00045790.JPEG
Yorkshire terrier
image path : ./test_data/ILSVRC2012_val_00002899.JPEG
rock python, rock snake, Python sebae
image path : ./test_data/ILSVRC2012_val_00021564.JPEG
coucal
image path : ./test_data/ILSVRC2012_val_00037133.JPEG
ice bear, polar bear, Ursus Maritimus, Thalartos maritimus
Total errors: 0, warnings: 0
```

3. 37. 11. 5. onnx_resnet_v1_101 Case

This case is used to classify images.

- 1) Enter the folder where the case is located.

```
orange@orange:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_101
```

- 2) The command to use CPU for inference is shown below. In this case, the images in the test_data folder will be classified, and the output shows that the model successfully recognized various objects.



```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_101$ python3 inference_onnx.py --images test_data --onnx_path \
model/resnet101-v1-7.onnx
.....
image path : test_data/ILSVRC2012_val_00037133.JPEG
ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus
image path : test_data/ILSVRC2012_val_00002899.JPEG
rock python, rock snake, Python sebae
image path : test_data/ILSVRC2012_val_00045790.JPEG
Yorkshire terrier
image path : test_data/ILSVRC2012_val_00021564.JPEG
coucal
image path : test_data/ILSVRC2012_val_00024154.JPEG
Ibizan hound, Ibizan Podenco
```

3) The command to use NPU for inference is shown below. In this case, the images in the test_data folder will be classified, and the output shows that the model successfully recognized various objects.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_101$ python3 inference_npu.py --images test_data --model_path resnet_v1_101.cix
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 1.
npu: noe_create_job success
image path : test_data/ILSVRC2012_val_00037133.JPEG
ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus
image path : test_data/ILSVRC2012_val_00002899.JPEG
rock python, rock snake, Python sebae
image path : test_data/ILSVRC2012_val_00045790.JPEG
Yorkshire terrier
image path : test_data/ILSVRC2012_val_00021564.JPEG
coucal
image path : test_data/ILSVRC2012_val_00024154.JPEG
Ibizan hound, Ibizan Podenco
npu: noe_clean_job success
```



```
npu: noe_unload_graph success
npu: noe_deinit_context success
```

3. 37. 11. 6. onnx_resnet_v1_12_int8 Case

1) Enter the folder where the case is located.

```
orangeapi@orangeapi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_12_int8
```

2) The command to use CPU for inference is shown below. In this case, the images in the test_data folder will be classified, and the output shows that the model successfully recognized various objects.

```
orangeapi@orangeapi:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_12_int8$ python3 inference_onnx.py --EP CPU
image path : ./test_data/ILSVRC2012_val_00024154.JPEG
Ibizan hound, Ibizan Podenco
image path : ./test_data/ILSVRC2012_val_00045790.JPEG
Yorkshire terrier
image path : ./test_data/ILSVRC2012_val_00002899.JPEG
rock python, rock snake, Python sebae
image path : ./test_data/ILSVRC2012_val_00021564.JPEG
coucal
image path : ./test_data/ILSVRC2012_val_00037133.JPEG
ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus
```

3) The command to use NPU for inference is shown below. In this case, the images in the test_data folder will be classified, and the output shows that the model successfully recognized various objects.

- a. Install the onnxruntime_zhoyi package. Please refer to the instructions in the section on [installing onnxruntime_zhoyi for installation methods](#).
- b. Set the environment variables.

```
orangeapi@orangeapi:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_12_int8$ export \
LD_LIBRARY_PATH=/usr/share/cix/lib/onnxruntime:$LD_LIBRARY_PATH
orangeapi@orangeapi:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_12_int8$
```



```
et_v1_12_int8$ export OPERATOR_PATH=/usr/share/cix/lib/onnxruntime/operator
```

c. Run the following command for inference.

```
orangeypi@orangeypi:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_12_int8$ python3 inference_onnx.py --EP NPU
image path : ./test_data/ILSVRC2012_val_00024154.JPEG
Ibizan hound, Ibizan Podenco
image path : ./test_data/ILSVRC2012_val_00045790.JPEG
Yorkshire terrier
image path : ./test_data/ILSVRC2012_val_00002899.JPEG
rock python, rock snake, Python sebae
image path : ./test_data/ILSVRC2012_val_00021564.JPEG
coucal
image path : ./test_data/ILSVRC2012_val_00037133.JPEG
ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus
Total errors: 0, warnings: 0
```

3. 37. 11. 7. onnx_resnet_v1_12_qdq Case

1) Enter the folder where the case is located.

```
orangeypi@orangeypi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_12_qdq
```

2) The command to use CPU for inference is shown below. In this case, the images in the test_data folder will be classified, and the output shows that the model successfully recognized various objects.

```
orangeypi@orangeypi:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_12_qdq$ python3 inference_onnx.py --EP CPU
image path : ./test_data/ILSVRC2012_val_00024154.JPEG
Ibizan hound, Ibizan Podenco
image path : ./test_data/ILSVRC2012_val_00045790.JPEG
Yorkshire terrier
image path : ./test_data/ILSVRC2012_val_00002899.JPEG
rock python, rock snake, Python sebae
image path : ./test_data/ILSVRC2012_val_00021564.JPEG
coucal
```



```
image path : ./test_data/ILSVRC2012_val_00037133.JPEG
ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus
```

3) The command to use NPU for inference is shown below. In this case, the images in the test_data folder will be classified, and the output shows that the model successfully recognized various objects.

- a. Install the onnxruntime_zhouyi package. Please refer to the instructions in the section on [installing onnxruntime_zhouyi for installation methods](#).
- b. Set the environment variables.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_12_qdq$ export \
LD_LIBRARY_PATH=/usr/share/cix/lib/onnxruntime:$LD_LIBRARY_PATH
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_12_qdq$ export OPERATOR_PATH=/usr/share/cix/lib/onnxruntime/operator
```

- c. Run the following command for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_12_qdq$ python3 inference_onnx.py --EP NPU
image path : ./test_data/ILSVRC2012_val_00024154.JPEG
Ibizan hound, Ibizan Podenco
image path : ./test_data/ILSVRC2012_val_00045790.JPEG
Yorkshire terrier
image path : ./test_data/ILSVRC2012_val_00002899.JPEG
rock python, rock snake, Python sebae
image path : ./test_data/ILSVRC2012_val_00021564.JPEG
coucal
image path : ./test_data/ILSVRC2012_val_00037133.JPEG
ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus
Total errors: 0, warnings: 0
```

3. 37. 11. 8. onnx_resnet_v1_50 Case

This case is used to classify images.

- 1) Enter the folder where the case is located.

```
orange@orange:~$ cd \
```



ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_50

2) The command to use CPU for inference is shown below. In this case, the images in the test_data folder will be classified, and the output shows that the model successfully recognized various objects.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_50$ python3 inference_onnx.py --images test_data --onnx_path \
model/resnet50-v1-12-sim.onnx
.....
image path : test_data/ILSVRC2012_val_00037133.JPEG
ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus
image path : test_data/ILSVRC2012_val_00002899.JPEG
rock python, rock snake, Python sebae
image path : test_data/ILSVRC2012_val_00045790.JPEG
Yorkshire terrier
image path : test_data/ILSVRC2012_val_00021564.JPEG
coucal
image path : test_data/ILSVRC2012_val_00024154.JPEG
Ibizan hound, Ibizan Podenco
```

3) The command to use NPU for inference is shown below. In this case, the images in the test_data folder will be classified, and the output shows that the model successfully recognized various objects.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_50$ python3 inference_npu.py --images test_data --model_path resnet_v1_50.cix
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 1.
npu: noe_create_job success
image path : test_data/ILSVRC2012_val_00037133.JPEG
ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus
image path : test_data/ILSVRC2012_val_00002899.JPEG
rock python, rock snake, Python sebae
image path : test_data/ILSVRC2012_val_00045790.JPEG
Yorkshire terrier
```




```
image path : test_data/ILSVRC2012_val_00021564.JPEG
```

```
coucal
```

```
image path : test_data/ILSVRC2012_val_00024154.JPEG
```

```
Ibizan hound, Ibizan Podenco
```

```
npu: noe_clean_job success
```

```
npu: noe_unload_graph success
```

```
npu: noe_deinit_context success
```

3. 37. 11. 9. onnx_vgg_16 Case

This case is used to classify images.

1) Enter the folder where the case is located.

```
orangeapi@orangeapi:~$ cd \
```

```
ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_vgg_16
```

2) The command to use CPU for inference is shown below. In this case, the images in the test_data folder will be classified, and the output shows that the model successfully recognized the objects in the images.

```
orangeapi@orangeapi:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_vgg_16$ python3 inference_onnx.py --images test_data/ILSVRC2012_val_00002899.JPEG \
--onnx_path model/vgg16-7.onnx
```

```
image path : test_data/ILSVRC2012_val_00002899.JPEG
```

```
rock python, rock snake, Python sebae
```

3) The command to use NPU for inference is shown below. In this case, the images in the test_data folder will be classified, and the output shows that the model successfully recognized the objects in the images.

```
orangeapi@orangeapi:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_vgg_16$ python3 inference_npu.py --images test_data/ILSVRC2012_val_00002899.JPEG \
--model_path vgg_16.cix
```

```
npu: noe_init_context success
```

```
npu: noe_load_graph success
```



```

Input tensor count is 1.
Output tensor count is 1.
npu: noe_create_job success
image path : test_data/ILSVRC2012_val_00002899.JPEG
rock python, rock snake, Python sebae
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success

```

3. 37. 11. 10. onnx_vgg16_12_int8 Case

1) Enter the folder where the case is located.

```

orange@orange:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_vgg16_12_int8

```

2) The command to use CPU for inference is shown below. In this case, the images in the test_data folder will be classified, and the output shows that the model successfully recognized various objects.

```

orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_vgg16_12_int8$ python3 inference_onnx.py --EP CPU
image path : ./test_data/ILSVRC2012_val_00024154.JPEG
Ibizan hound, Ibizan Podenco
image path : ./test_data/ILSVRC2012_val_00045790.JPEG
Yorkshire terrier
image path : ./test_data/ILSVRC2012_val_00002899.JPEG
rock python, rock snake, Python sebae
image path : ./test_data/ILSVRC2012_val_00021564.JPEG
coucal
image path : ./test_data/ILSVRC2012_val_00037133.JPEG
ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus

```

3) The command to use NPU for inference is shown below. In this case, the images in the test_data folder will be classified, and the output shows that the model successfully recognized various objects.

- a. Install the onnxruntime_zhouyi package. Please refer to the instructions in the



section on [installing onnxruntime_zhouyi](#) for installation methods.

b. Set the environment variables.

```
orangeapi@orangeapi:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_vgg16_12_int8$ export \
LD_LIBRARY_PATH=/usr/share/cix/lib/onnxruntime:$LD_LIBRARY_PATH
orangeapi@orangeapi:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_vgg16_12_int8$ export OPERATOR_PATH=/usr/share/cix/lib/onnxruntime/operator
```

c. Run the following command for inference.

```
orangeapi@orangeapi:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_vgg16_12_int8$ python3 inference_onnx.py --EP NPU
image path : ./test_data/ILSVRC2012_val_00024154.JPEG
Ibizan hound, Ibizan Podenco
image path : ./test_data/ILSVRC2012_val_00045790.JPEG
Yorkshire terrier
image path : ./test_data/ILSVRC2012_val_00002899.JPEG
rock python, rock snake, Python sebae
image path : ./test_data/ILSVRC2012_val_00021564.JPEG
coucal
image path : ./test_data/ILSVRC2012_val_00037133.JPEG
ice bear, polar bear, Ursus Maritimus, Thalarcos maritimus
Total errors: 0, warnings: 0
```

3. 37. 11. 11. onnx_vgg16_12_qdq Case

1) Enter the folder where the case is located.

```
orangeapi@orangeapi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_vgg16_12_qdq
```

2) The command to use CPU for inference is shown below. In this case, the images in the test_data folder will be classified, and the output shows that the model successfully recognized various objects.

```
orangeapi@orangeapi:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_vgg16_12_qdq$ python3 inference_onnx.py --EP CPU
image path : ./test_data/ILSVRC2012_val_00024154.JPEG
Ibizan hound, Ibizan Podenco
```



```

image path : ./test_data/ILSVRC2012_val_00045790.JPEG
Yorkshire terrier
image path : ./test_data/ILSVRC2012_val_00002899.JPEG
rock python, rock snake, Python sebae
image path : ./test_data/ILSVRC2012_val_00021564.JPEG
coucal
image path : ./test_data/ILSVRC2012_val_00037133.JPEG
ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus

```

3) The command to use NPU for inference is shown below. In this case, the images in the test_data folder will be classified, and the output shows that the model successfully recognized various objects.

- a. Install the onnxruntime_zhouyi package. Please refer to the instructions in the section on [installing onnxruntime_zhouyi for installation methods](#).
- b. Set the environment variables.

```

orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_vgg
16_12_qdq$ export \
LD_LIBRARY_PATH=/usr/share/cix/lib/onnxruntime:$LD_LIBRARY_PATH
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_vgg
16_12_qdq$ export OPERATOR_PATH=/usr/share/cix/lib/onnxruntime/operator

```

- c. Run the following command for inference.

```

orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_vgg
16_12_qdq$ python3 inference_onnx.py --EP NPU
image path : ./test_data/ILSVRC2012_val_00024154.JPEG
Ibizan hound, Ibizan Podenco
image path : ./test_data/ILSVRC2012_val_00045790.JPEG
Yorkshire terrier
image path : ./test_data/ILSVRC2012_val_00002899.JPEG
rock python, rock snake, Python sebae
image path : ./test_data/ILSVRC2012_val_00021564.JPEG
coucal
image path : ./test_data/ILSVRC2012_val_00037133.JPEG
ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus
Total errors: 0, warnings: 0

```



3. 37. 11. 12. onnx_vit-base-16-224 Case

This case is used to classify images.

1) Enter the folder where the case is located.

```
orange@orange:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_vit-base-16-224
```

2) The command to use CPU for inference is shown below. In this case, the images in the test_data folder will be classified, and the output shows that the model successfully recognized various objects.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_vit-base-16-224$ python3 inference_onnx.py
.....
image path : test_data/ILSVRC2012_val_00037133.JPEG
ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus
image path : test_data/ILSVRC2012_val_00002899.JPEG
rock python, rock snake, Python sebae
image path : test_data/ILSVRC2012_val_00045790.JPEG
Yorkshire terrier
image path : test_data/ILSVRC2012_val_00021564.JPEG
coucal
image path : test_data/ILSVRC2012_val_00024154.JPEG
Ibizan hound, Ibizan Podenco
```

3) The command to use NPU for inference is shown below. In this case, the images in the test_data folder will be classified, and the output shows that the model successfully recognized various objects.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_vit-base-16-224$ python3 inference_npu.py
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 1.
npu: noe_create_job success
```



```
image path : test_data/ILSVRC2012_val_00037133.JPEG
ice bear, polar bear, Ursus Maritimus, Thalarctos maritimus
image path : test_data/ILSVRC2012_val_00002899.JPEG
rock python, rock snake, Python sebae
image path : test_data/ILSVRC2012_val_00045790.JPEG
Yorkshire terrier
image path : test_data/ILSVRC2012_val_00021564.JPEG
coucal
image path : test_data/ILSVRC2012_val_00024154.JPEG
Ibizan hound, Ibizan Podenco
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success
```

3. 37. 12. Testing Method for Image-Enhancement Class Models in ComputeVision

Before testing the model, please install the dependency packages according to the instructions in the section on [installing NPU dependencies](#), otherwise the test will report errors.

3. 37. 12. 1. onnx_AGLNet Case

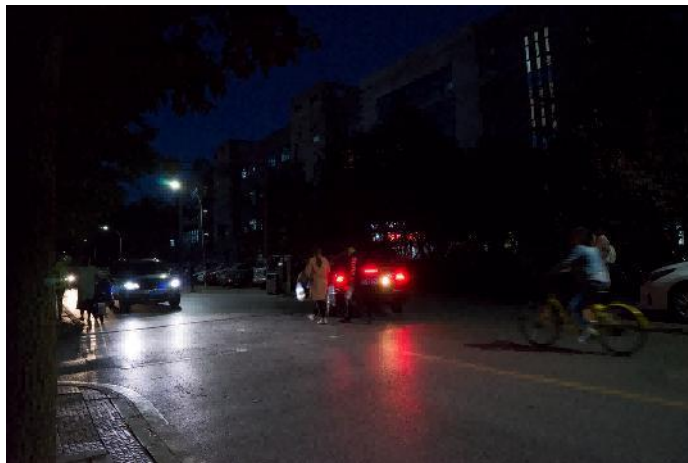
This case is used to enhance blurry and noisy images captured in low light and low light environments into bright and clear images.

1) Enter the folder where the case is located.

```
orangeapi@orangeapi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Image_Enhancement/onnx_AGLNet
```

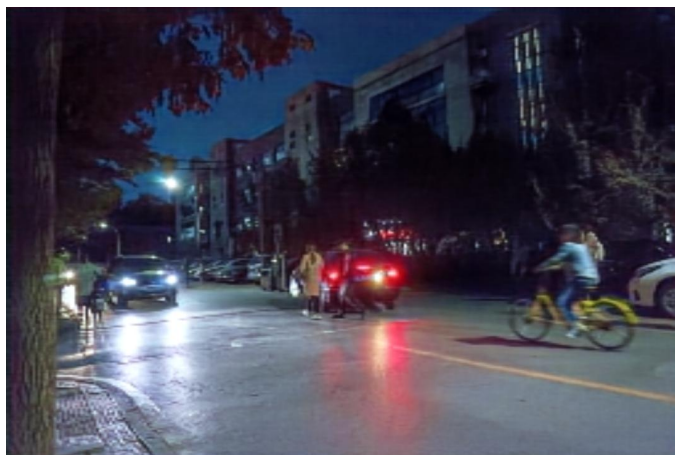
2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

```
orangeapi@orangeapi:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Enhancement/onnx_AGLNet$ python3 inference_onnx.py
```



3) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Enhancement/onnx_AG  
LLNet$ ls output  
onnx_5047.png
```

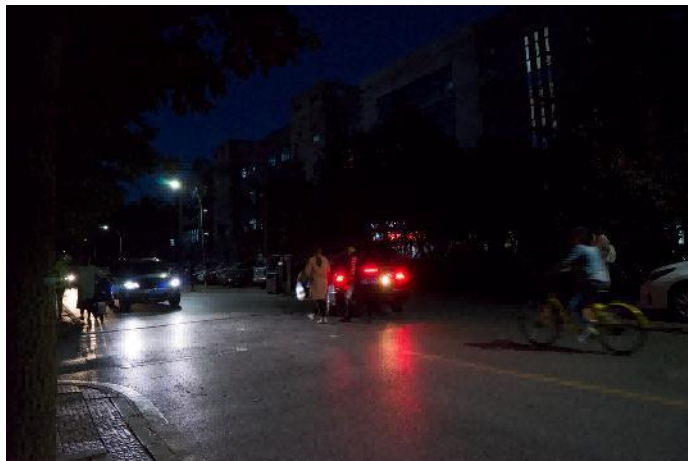


4) The command to use NPU for inference is as follows, using images from the test_data folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Enhancement/onnx_AG  
LLNet$ python3 inference_npu.py  
npu: noe_init_context success  
npu: noe_load_graph success  
Input tensor count is 1.  
Output tensor count is 1.  
npu: noe_create_job success  
npu: noe_clean_job success
```

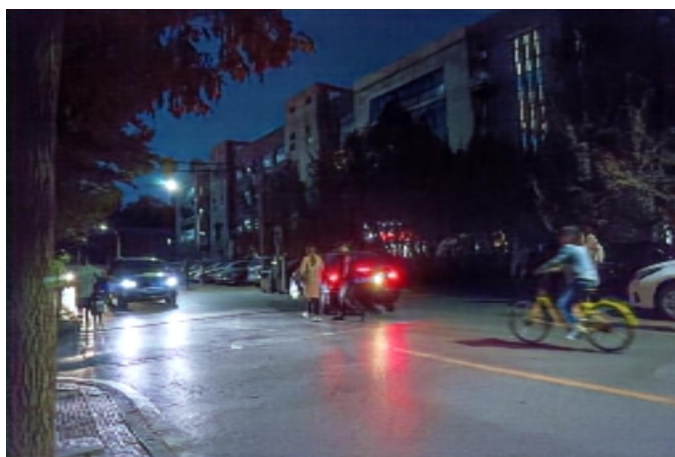



```
npu: noe_unload_graph success  
npu: noe_deinit_context success
```



5) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Enhancement/onnx_AG  
LLNet$ ls output  
npu_5047.png
```



3. 37. 12. 2. onnx_nighttime_dehaze Case

This case is used to improve image quality under foggy, blurry, or low visibility conditions at night, enhancing image clarity and visibility.

1) Enter the folder where the case is located.

```
orangepi@orangepi:~$ cd \
```



```
ai_model_hub_25_Q3/models/ComputeVision/Image_Enhancement/onnx_nighttime_dehaze
```

2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Enhancement/onnx_nighttime_dehaze$ python3 inference_onnx.py
```



3) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Enhancement/onnx_nighttime_dehaze$ ls output
onnx_511.png
```



4) The command to use NPU for inference is as follows, using images from the test_data



folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Enhancement/onnx_nigh
ttime_dehaze$ python3 inference_npu.py
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 1.
npu: noe_create_job success
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success
```



5) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Enhancement/onnx_nigh
ttime_dehaze$ ls output
npu_511.png
```



3. 37. 12. 3. onnx_pairlie Case

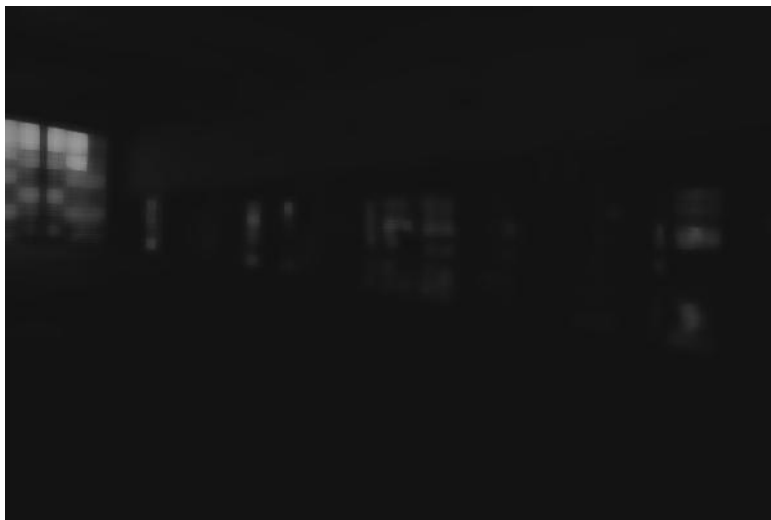
This case significantly improves the quality of low light images by simultaneously optimizing brightness restoration, noise suppression, and color correction.

1) Enter the folder where the case is located.

```
orangepi@orangepi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Image_Enhancement/onnx_pairlie
```

2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Enhancement/onnx_pairlie$ python3 inference_onnx.py
```



3) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Enhancement/onnx_pairlie$ ls output
onnx_780.png
```



4) The command to use NPU for inference is as follows, using images from the test_data folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Enhancement/onnx_pairlie$ python3 inference_npu.py
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 2.
Output tensor count is 1.
```

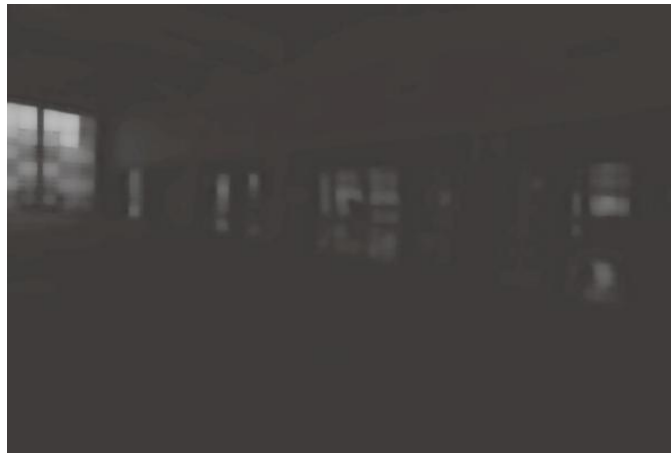


```
npu: noe_create_job success  
npu: noe_clean_job success  
npu: noe_unload_graph success  
npu: noe_deinit_context success
```



5) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Enhancement/onnx_pairl  
ie$ ls output  
npu_780.png
```



3.37.13. Testing Method for LaneDetected Class Models in ComputeVision

Before testing the model, please install the dependency packages according to the instructions in the section on [installing NPU dependencies](#), otherwise the test will report errors.



3. 37. 13. 1. onnx_lanenet Case

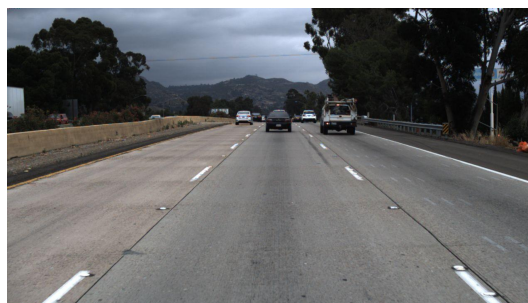
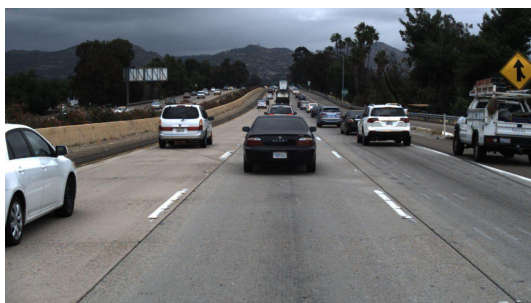
This case is used for lane detection.

1) Enter the folder where the case is located.

```
orangePi@orangePi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Lane_Detection/onnx_lanenet
```

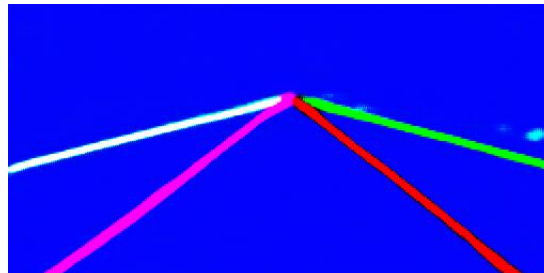
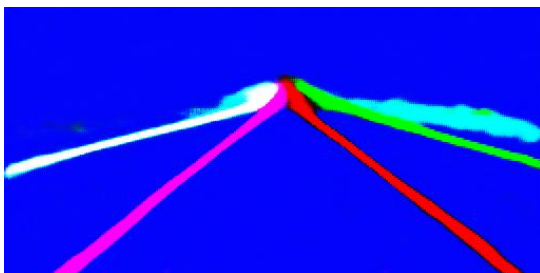
2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

```
orangePi@orangePi:~/ai_model_hub_25_Q3/models/ComputeVision/Lane_Detection/onnx_lanenet$ python3 inference_onnx.py
.....
(3, 256, 512)
[ WARN:0@2.267] global loadsave.cpp:848 imwrite_ Unsupported depth image for selected encoder is fallbacked to CV_8U.
(3, 256, 512)
```



3) The images generated by inference will be saved in the output directory.

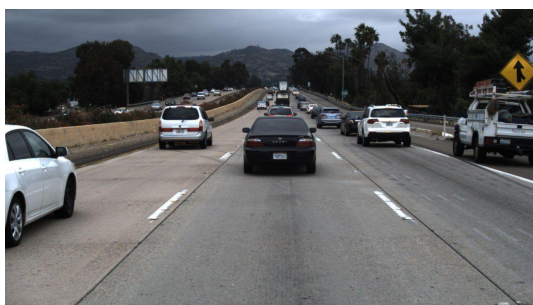
```
orangePi@orangePi:~/ai_model_hub_25_Q3/models/ComputeVision/Lane_Detection/onnx_lanenet$ ls output
onnx_1.png  onnx_2.png
```





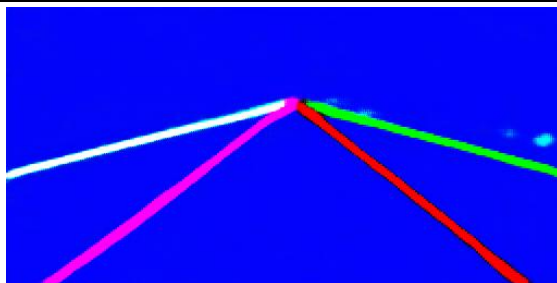
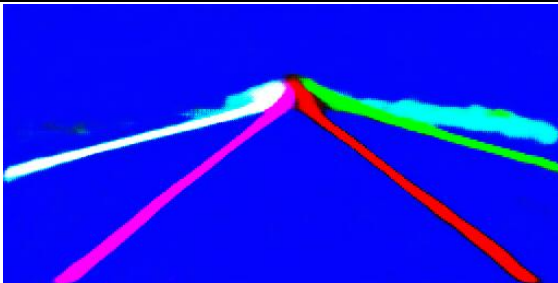
4) The command to use NPU for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Lane_Detection/onnx_lane
net$ python3 inference_npu.py
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 2.
npu: noe_create_job success
(3, 256, 512)
[ WARN:0@1.921] global loadsave.cpp:848 imwrite_ Unsupported depth image for selected
encoder is fallbacked to CV_8U.
(3, 256, 512)
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success
```



5) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Lane_Detection/onnx_lane
et$ ls output
npu_1.png  npu_2.png
```





3. 37. 13. 2. onnx_LSTR Case

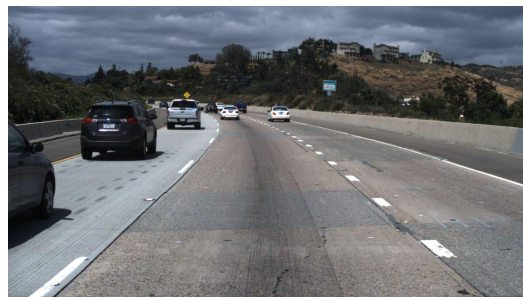
This case is used for lane detection.

1) Enter the folder where the case is located.

```
orangeapi@orangepi:~$ cd \  
ai_model_hub_25_Q3/models/ComputeVision/Lane_Detection/onnx_LSTR
```

2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

```
orangeapi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Lane_Detection/onnx_LSTR$ python3 inference_onnx.py
```



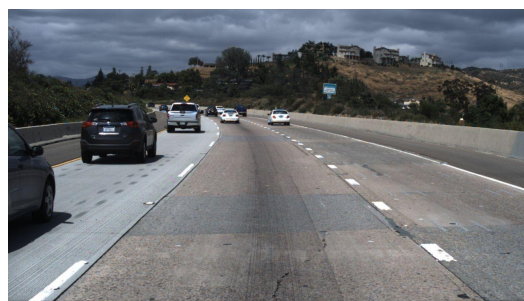
3) The images generated by inference will be saved in the output directory.

```
orangeapi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Lane_Detection/onnx_LSTR$ ls output  
onnx_1.jpg  onnx_2.jpg  onnx_3.jpg  onnx_4.jpg
```



4) The command to use NPU for inference is as follows, using images from the test_data folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Lane_Detection/onnx_LST
R$ python3 inference_npu.py
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 2.
Output tensor count is 2.
npu: noe_create_job success
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success
```



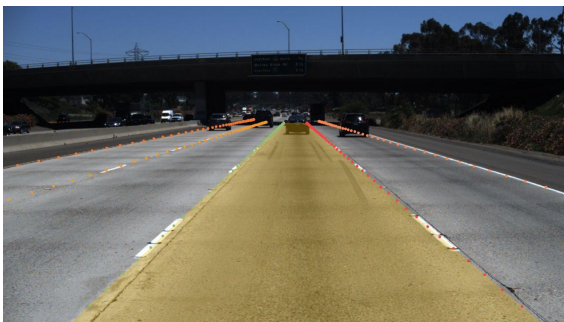
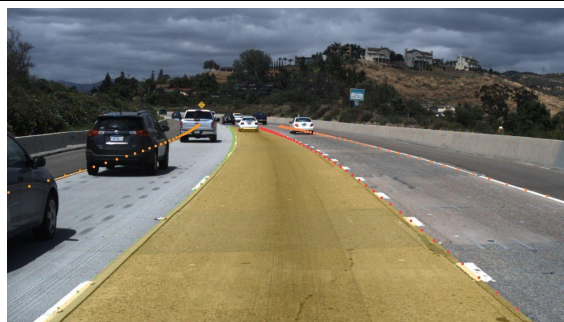
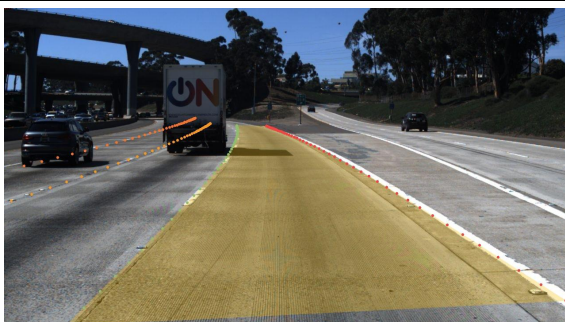


5) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Lane_Detection/onnx_LST
```

RS ls output

```
npu_1.jpg  npu_2.jpg  npu_3.jpg  npu_4.jpg
```



3. 37. 13. 3. onnx_Ultra_Fast_Lane_Detection Case

This case is used for lane detection.

1) Enter the folder where the case is located.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Lane_Detection/onnx_Ultra_Fast_Lane_Detection
```

```
ai_model_hub_25_Q3/models/ComputeVision/Lane_Detection/onnx_Ultra_Fast_Lane_Detection
```



2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Lane_Detection/onnx_Ultra_Fast_Lane_Detection$ python3 inference_onnx.py
```



3) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Lane_Detection/onnx_Ultra_Fast_Lane_Detection$ ls output
onnx_1.jpg
```



4) The command to use NPU for inference is as follows, using images from the test_data folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Lane_Detection/onnx_Ultra_Fast_Lane_Detection$ python3 inference_npu.py
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
```



Output tensor count is 1.

npu: noe_create_job success

npu: noe_clean_job success

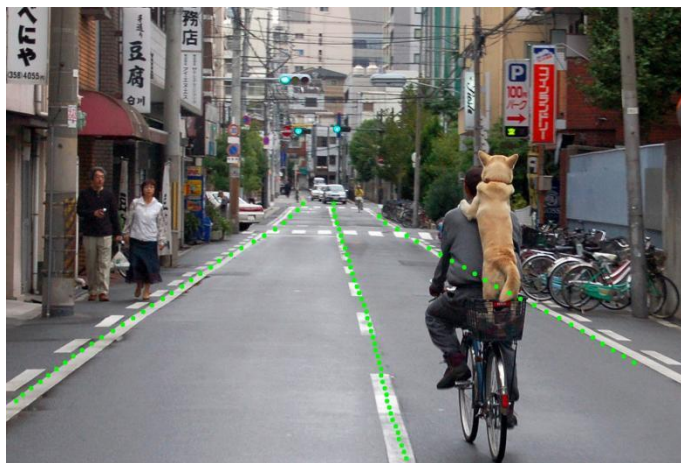
npu: noe_unload_graph success

npu: noe_deinit_context success



5) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Lane_Detection/onnx_Ultra_Fast_Lane_Detection$ ls output
npu_1.jpg
```



3. 37. 13. 4. onnx_Ultra_Fast_Lane_Detection_v2 Case

This case is used for lane detection.



1) Enter the folder where the case is located.

```
orangepi@orangepi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Lane_Detection/onnx_Ultra_Fast_Lane_Detection_v2
```

2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Lane_Detection/onnx_Ultra_Fas
t_Lane_Detection_v2$ python3 inference_onnx.py
```



3) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Lane_Detection/onnx_Ultra
_Fast_Lane_Detection_v2$ ls output
onnx_1.jpg  onnx_2.jpg  onnx_3.jpg
```




4) The command to use NPU for inference is as follows, using images from the test_data folder for inference.

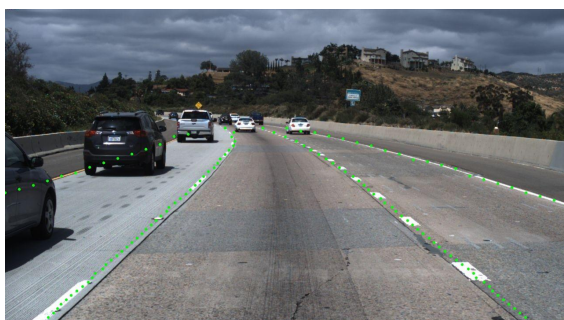
```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Lane_Detection/onnx_Ultra_Fast_Lane_Detection_v2$ python3 inference_npu.py
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 4.
npu: noe_create_job success
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success
```





5) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Lane_Detection/onnx_Ultra_Fast_Lane_Detection_v2$ ls output  
npu_1.jpg  npu_2.jpg  npu_3.jpg
```



3. 37. 14. Testing Method for Object Detection Class Models in ComputeVision

Before testing the model, please install the dependency packages according to the instructions in the section on [installing NPU dependencies](#), otherwise the test will



report errors.

3. 37. 14. 1. onnx_resnet50_ssd300 Case

This case is used for object detection.

1) Enter the folder where the case is located.

```
orangeapi@orangeapi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_resnet50_ssd300
```

2) The command to use CPU for inference is as follows, using images from the test_data folder for inference. At the beginning of the inference process, NVIDIA will download NVIDIA DeepLearning Exemplars_torchhub from GitHub. Please be patient and wait for the download to complete. Any warnings during inference can be ignored.

```
orangeapi@orangeapi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_resnet50_ssd300$ python3 inference_onnx.py --image_path test_data --onnx_path \
model/ssd_resnet50_300_sim.onnx
```

If you cannot download the NVIDIA DeepLearning Exemplars_torchhub compressed file from Github, you can manually set it using the following method, and then the download will be skipped.

1. Download the NVIDIA DeepLearning Exemplars_torchhub. zip compressed file from the official tool on the development board download page.



2. Upload NVIDIA_DeepLearningExemplars_torchhubzip to the Linux system of the development board.

3. Extract NVIDIA_DeepLearningExemplars_torchhubzip to ~/.cache/torch/hub, noting that the following commands are executed under normal user conditions.

```
orangeapi@orangeapi:~$ mkdir -p ~/.cache/torch/hub
orangeapi@orangeapi:~$ unzip NVIDIA_DeepLearningExamples_torchhub.zip -d ~/.cache/torch/hub
```




3) The images generated by inference will be saved in the output directory.

```
orange@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_resnet50_ssd300$ ls output
```

onnx_000000037777.jpg onnx_000000252219.jpg onnx_000000559842.jpg





4) The command to use NPU for inference is as follows, using images from the test_data folder for inference. At the beginning of the inference process, NVIDIA will download NVIDIA DeepLearning Exemplars_torchhub from GitHub. Please be patient and wait for the download to complete. Any warnings during inference can be ignored.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_resnet50_ssd_300$ python3 inference_npu.py --image_path test_data --model_path ssd_300.cix
```

If you cannot download the NVIDIA DeepLearning Exemplars_torchhub compressed file from Github, you can manually set it using the following method, and then the download will be skipped.

1. Download the NVIDIA DeepLearning Exemplars_torchhub. zip compressed file from the official tool on the development board download page.



2. Upload NVIDIA_DeepLearningExamples_torchhubzip to the Linux system of the development board.

3. Extract NVIDIA_DeepLearningExamples_torchhubzip to ~/.cache/torch/hub, noting that the following commands are executed under normal user conditions.

```
orange@orange:~$ mkdir -p ~/.cache/torch/hub
```



```
orange@orange:~$ unzip NVIDIA_DeepLearningExamples_torchhub.zip -d ~/.cache/torch/hub
```

5) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_resnet50_ssd300$ ls output
npu_000000037777.jpg  npu_000000252219.jpg  npu_000000559842.jpg
```





3. 37. 14. 2. **onnx_retinanet Case**

This case is used for object detection.

1) Enter the folder where the case is located.

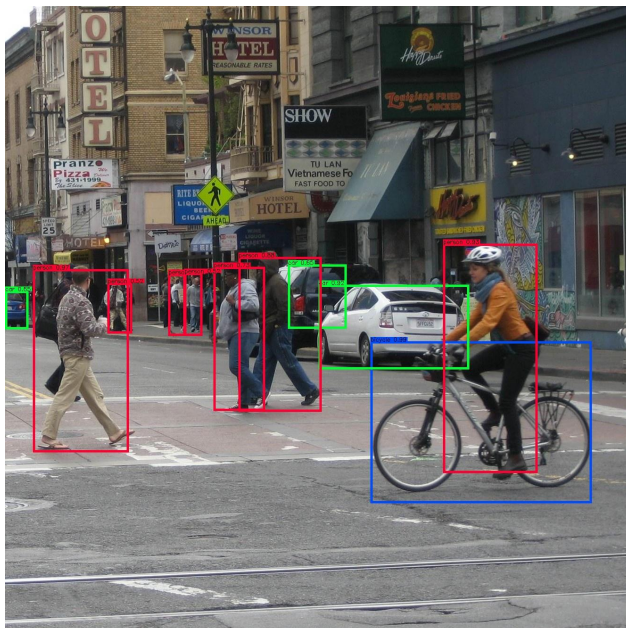
```
orangepi@orangepi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_retinanet
```

2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_retinanet$ python3 inference_onnx.py
```

3) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_retinanet$ ls output
onnx_street.jpg
```

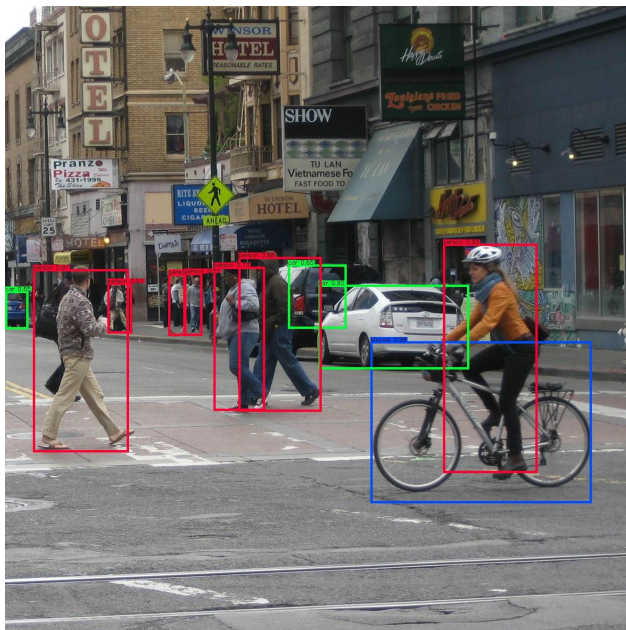


4) The command to use NPU for inference is as follows, using images from the test_data folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_reti
nanet$ python3 inference_npu.py
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 2.
npu: noe_create_job success
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success
```

5) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_reti
nanet$ ls output
npu_street.jpg
```



3. 37. 14. 3. onnx_vehicle_detection Case

This case is used for object detection.

1) Enter the folder where the case is located.

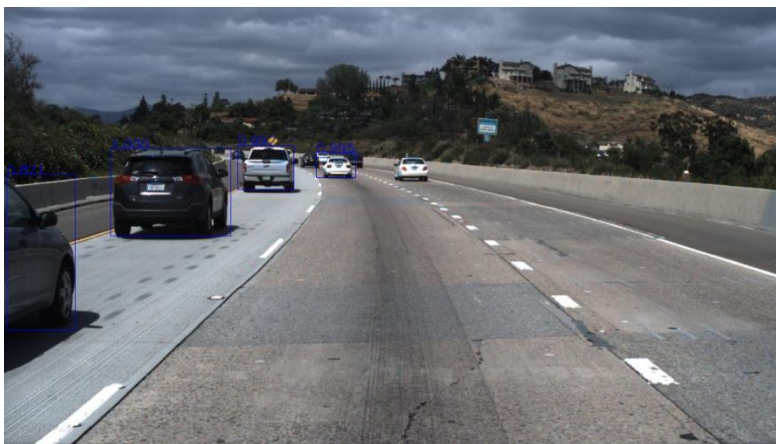
```
orange_pi@orange_pi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_vehicle_detection
```

2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

```
orange_pi@orange_pi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_vehicle_detection$ python3 inference_onnx.py
```

3) The images generated by inference will be saved in the output directory.

```
orange_pi@orange_pi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_vehicle_detection$ ls output
onnx_3.jpg  onnx_4.jpg
```



4) The command to use NPU for inference is as follows, using images from the test_data folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detectiononnx_veh  
cle_detection$ python3 inference_npu.py  
npu: noe_init_context success  
npu: noe_load_graph success  
Input tensor count is 1.  
Output tensor count is 2.  
npu: noe_create_job success  
npu: noe_clean_job success  
npu: noe_unload_graph success  
npu: noe_deinit_context success
```

5) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_veh
```

```
icle_detection$ ls output
npu_3.jpg  npu_4.jpg
```



3.37.14.4. onnx_yolov12_m Case

This case is used for object detection.

1) Enter the folder where the case is located.

```
orangeipi@orangeipi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolov12_m
```

2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

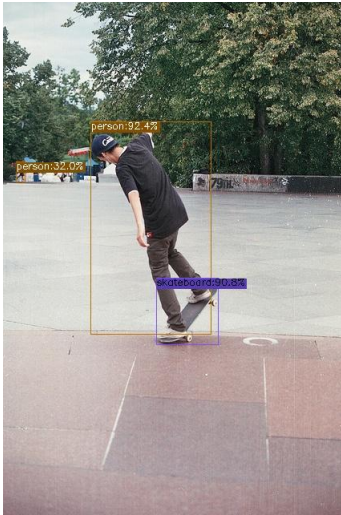
```
orangeipi@orangeipi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolov12_m$ python3 inference_onnx.py
```

.....

100% | 2/2 [00:02<00:00, 1.28s/it]

3) The images generated by inference will be saved in the output directory.

```
orangepepi@orangepepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolov12_m$ ls output
onnx_000000007977.jpg  onnx_1.jpeg
```

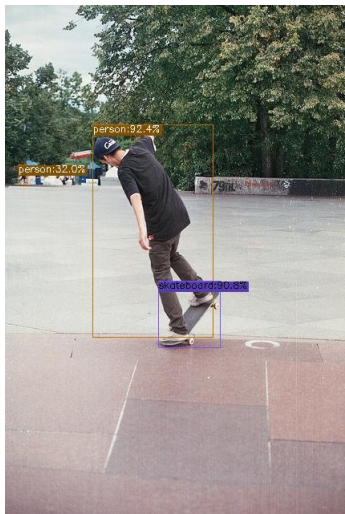


4) The command to use NPU for inference is as follows, using images from the test_data folder for inference.

```
orangeipi@orangeipi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolov12_m$ python3 inference_npu.py
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 1.
npu: noe_create_job success
100% ██████████ 2/2 [00:00<00:00, 9.08it/s]
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success
```

5) The images generated by inference will be saved in the output directory.

```
orangepepi@orangepepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolov12_m$ ls output
npu 0000000007977.jpg  npu 1.jpeg
```

3. 37. 14. 5. onnx_yolov12_n Case

This case is used for object detection.

1) Enter the folder where the case is located.

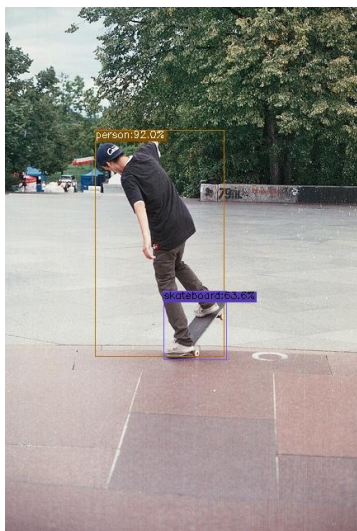
```
orangepi@orangepi:~$ cd \  
ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolov12_n
```

2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolov12_n$ python3 inference_onnx.py
```

3) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolov12_n$ ls output  
onnx_000000007977.jpg  onnx_1.jpeg
```

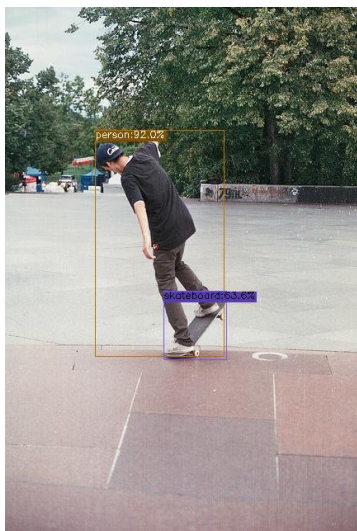


4) The command to use NPU for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolov12_n$ python3 inference_npu.py
```

5) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolov12_n$ ls output
npu_000000007977.jpg  npu_1.jpeg
```





3. 37. 14. 6. onnx_yolo_v3 Case

This case is used for object detection.

1) Enter the folder where the case is located.

```
orangeapi@orangepi:~$ cd \  
ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolo_v3
```

2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

```
orangeapi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolo_v3$ python3 inference_onnx.py --image_path test_data --onnx_path \  
model/yolov3_dynamic_batch.onnx
```

3) The images generated by inference will be saved in the output directory.

```
orangeapi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolo_v3$ ls output  
onnx_bus.jpg
```





4) The command to use NPU for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolo_v3$ python3 inference_npu.py --image_path test_data --model_path yolo_v3.cix
```

5) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolo_v3$ ls output
npu_bus.jpg
```



3. 37. 14. 7. onnx_yolov8_l Case

This case is used for object detection.

1) Enter the folder where the case is located.

```
orangepi@orangepi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolov8_l
```



2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolov8_1$ python3 inference_onnx.py --image_path test_data --onnx_path model/yolov8l.onnx
```

3) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolov8_1$ ls output
onnx_1.jpeg  onnx_ILSVRC2012_val_00004704.JPEG
```



4) The command to use NPU for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolov8_1$ python3 inference_npu.py --image_path test_data --model_path yolov8_l.cix
```

5) The images generated by inference will be saved in the output directory.

```
oorangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolov8_1$ ls output
npu_1.jpeg  npu_ILSVRC2012_val_00004704.JPEG
```



3. 37. 14. 8. onnx_yolov8l_worldv2 Case

This case is used for object detection.

When testing this case, it is necessary to download something from the network. Before testing, please ensure that the network of the development board is unobstructed.

1) Enter the folder where the case is located.

```
orange_pi@orange_pi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolov8l_worldv2
```

2) The command to use CPU for inference is as follows, using images from the test_data folder for inference. This case is used for object detection.

```
orange_pi@orange_pi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolov8l_worldv2$ python3 inference_onnx.py
```

3) The images generated by inference will be saved in the output directory.

```
orange_pi@orange_pi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolov8l_worldv2$ ls output
onnx_000000011264.jpg  onnx_000000141416.jpg  onnx_000000170127.jpg
onnx_000000176799.jpg  onnx_000000229659.jpg
```

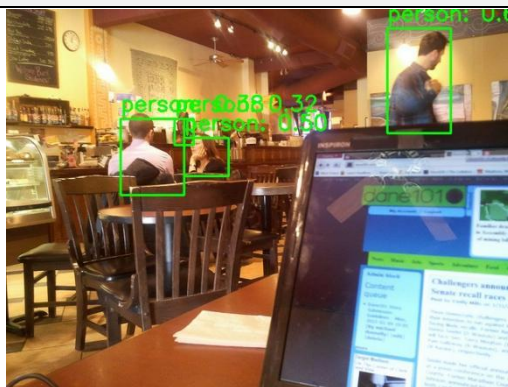



4) The command to use NPU for inference is as follows, using images from the test_data folder for inference. This case is used for object detection.

```
oorangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detectiononnx_yolov8l_worldv2$ python3 inference_npu.py
```

5) The images generated by inference will be saved in the output directory.

```
oorangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detectiononnx_yolov8l_worldv2$ ls output
npu_000000011264.jpg  npu_000000141416.jpg  npu_000000170127.jpg
npu_000000176799.jpg  npu_000000229659.jpg
```



3. 37. 14. 9. onnx_yolov8_n Case

This case is used for object detection.

1) Enter the folder where the case is located.

```
orangepi@orangepi:~$ cd \
```



```
ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolov8_n
```

2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolov8_n$ python3 inference_onnx.py
```

3) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolov8_n$ ls output
onnx_000000007977.jpg  onnx_000000008762.jpg
```

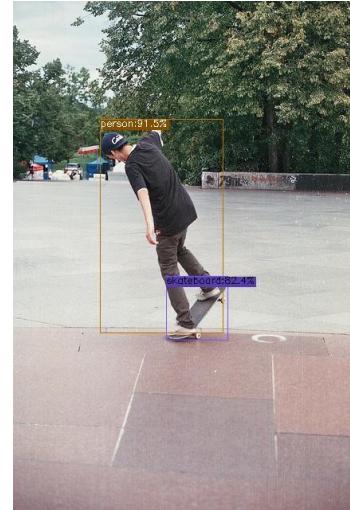


4) The command to use NPU for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolov8_n$ python3 inference_npu.py
```

5) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolov8_n$ ls output
npu_000000007977.jpg  npu_000000008762.jpg
```



3. 37. 14. 10. onnx_yolox_l Case

This case is used for object detection.

1) Enter the folder where the case is located.

```
orangepi@orangepi:~$ cd \  
ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolox_l
```

2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolox_l$ python3 inference_onnx.py --image_path \  
test_data/ILSVRC2012_val_00004704.JPEG --onnx_path model/yolox_l.onnx
```

3) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolox_l$ ls output  
onnx_ILSVRC2012_val_00004704.JPEG
```



4) The command to use NPU for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolox_1$ python3 inference_npu.py --image_path \
test_data/ILSVRC2012_val_00004704.JPEG --model_path yolox_1.cix
```

5) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolox_1$ ls output
npu_ILSVRC2012_val_00004704.JPEG
```





3. 37. 14. 11. onnx_yolox_m Case

This case is used for object detection.

1) Enter the folder where the case is located.

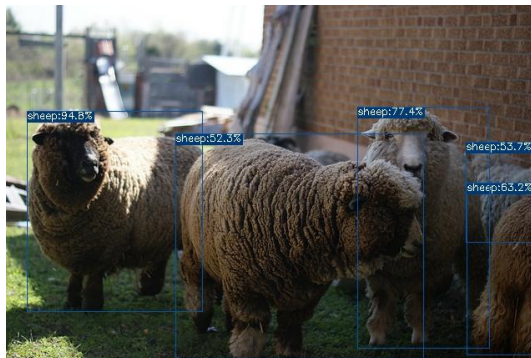
```
orangeapi@orangepi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolox_m
```

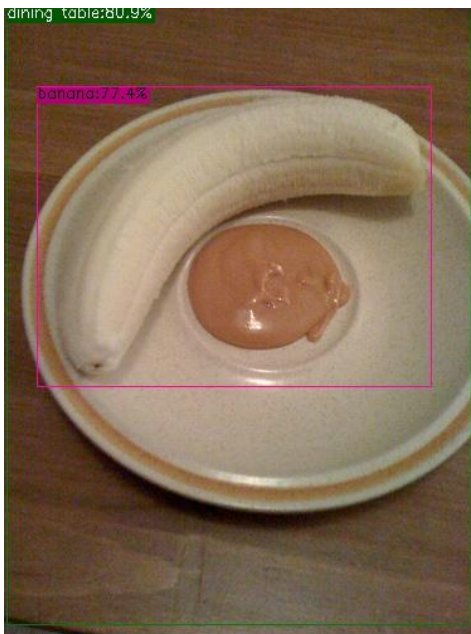
2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

```
orangeapi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolox_m$ python3 inference_onnx.py
```

3) The images generated by inference will be saved in the output directory.

```
orangeapi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolox_m$ ls output
onnx_000000005992.jpg  onnx_000000013004.jpg  onnx_000000014439.jpg
```



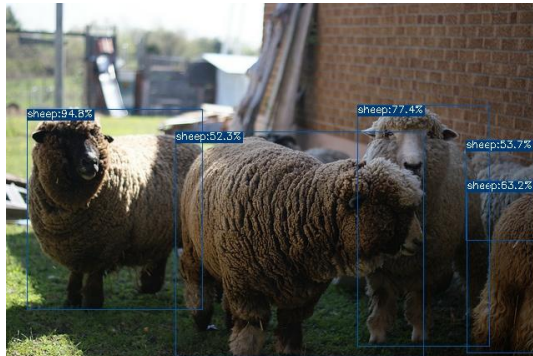


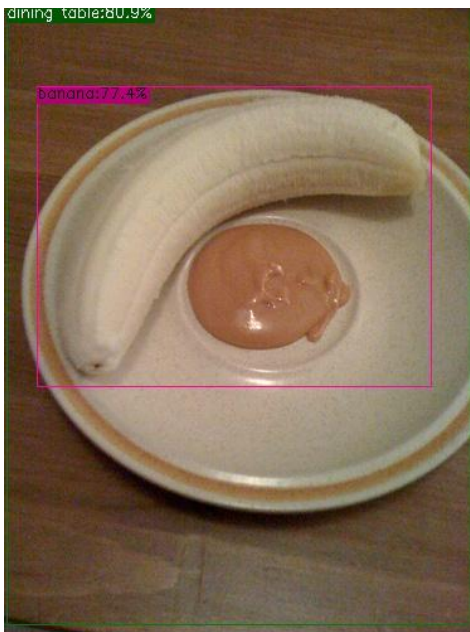
4) The command to use NPU for inference is as follows, using images from the test_data folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolox_m$ python3 inference_npu.py
```

5) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolox_m$ ls output/  
npu_000000005992.jpg  npu_000000014439.jpg  npu_000000013004.jpg
```





3. 37. 14. 12. onnx_yolox_s Case

This case is used for object detection.

1) Enter the folder where the case is located.

```
orange_pi@orange_pi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolox_s
```

2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

```
orange_pi@orange_pi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolox_s$ python3 inference_onnx.py
```

3) The images generated by inference will be saved in the output directory.

```
orange_pi@orange_pi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolox_s$ ls output
onnx_ILSVRC2012_val_00004704.JPEG
```



4) The command to use NPU for inference is as follows, using images from the test_data folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolo_x$ python3 inference_npu.py
```

5) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/onnx_yolo_x$ ls output
npu_ILSVRC2012_val_00004704.JPEG
```



3. 37. 14. 13. torch_centernet_resnet50 Case

This case is used for object detection.

1) Enter the folder where the case is located.

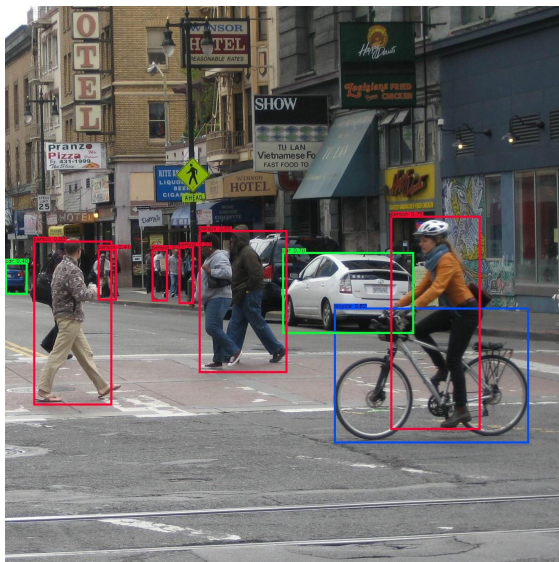
```
orangeapi@orangepi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/torch_centernet_resnet50
```

2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

```
orangeapi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/torch_centernet_resnet50$ python3 inference_pt.py
```

3) The images generated by inference will be saved in the output directory.

```
orangeapi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/torch_centernet_resnet50$ ls output
pt_street.jpg
```

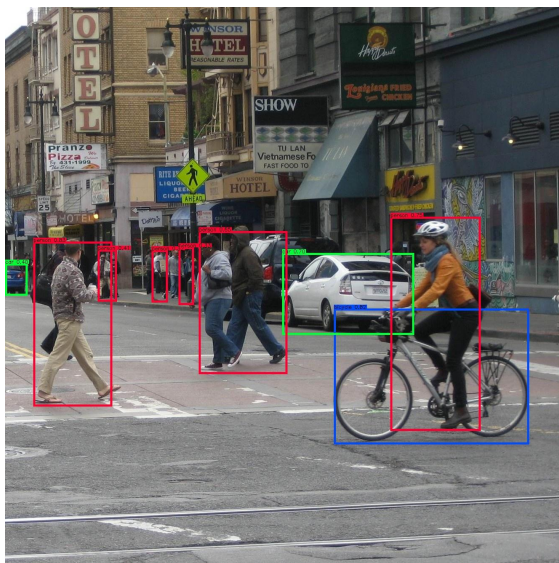


4) The command to use NPU for inference is as follows, using images from the test_data folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/torch_cen  
ternet_resnet50$ python3 inference_npu.py
```

5) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Object_Detection/torch_cen  
ternet_resnet50$ ls output  
npu_street.jpg
```





3. 37. 15. Testing Method for OCR Models in ComputeVision

Before testing the model, please install the dependency packages according to the instructions in the section on [installing NPU dependencies](#), otherwise the test will report errors.

3. 37. 15. 1. onnx_cnn Case

This case is used to recognize text in an image and convert it into a computer-readable string.

1) Enter the folder where the case is located.

```
orangePi@orangePi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/OCR/onnx_crnn
```

2) The command to use cpu for inference is as follows, using images from the test_data folder for inference.

```
orangePi@orangePi:~/ai_model_hub_25_Q3/models/ComputeVision/OCR/o
nnx_crnn$ python3 inference_onnx.py
.....
image_path: test_data/7.jpg
h---iig-hh--f-lly-i-nn-g-- => highflying
image_path: test_data/9.jpg
ss--c--u---t-u---l-a--r--- => scutular
image_path: test_data/1.jpg
t--iir-e--d---n--e--s--s-- => tiredness
image_path: test_data/8.jpg
g----rr-o---s--s--if-yy--- => grossify
image_path: test_data/2.jpg
p----a--y---rr-o-l-l-e-r-- => payroller
image_path: test_data/10.jpg
c---r-i-s-p--e--n--e--d--- => crispened
image_path: test_data/3.jpg
rr-h--iz-oo-p--o--d-i-st-- => rhizopodist
image_path: test_data/4.jpg
tt--hh--e-r--i-a-c--a--s-- => theriacas
```




```
image_path: test_data/6.jpg
d-----u---a-l-iis-t-iic--- => dualistic
image_path: test_data/5.jpg
s--e--m--a-n-tticaal-lyy- => semantically
```

3) The command to use npu for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/OCR/onnx_crnn$ python3 inference_npu.py
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 1.
npu: noe_create_job success
image_path: test_data/7.jpg
h---i-g-hh--f-lly-i-n--g-- => highflying
image_path: test_data/9.jpg
sss-c-u---t-u---l-a-r--- => scutular
image_path: test_data/1.jpg
t--iir-e-d---n--e--s--s-- => tiredness
image_path: test_data/8.jpg
gg----r-o---s--s--iff-y--- => grossify
image_path: test_data/2.jpg
p----a-y---r--o-l-l-e-r-- => payroller
image_path: test_data/10.jpg
c---r-i-s-p--e--n--e--d--- => crispened
image_path: test_data/3.jpg
rr-h--iz-oo-p--o--d-i-stt- => rhizopodist
image_path: test_data/4.jpg
tt-hh--e-r--i-a-c--a--s-- => theriacas
image_path: test_data/6.jpg
dd----u---a-l-i-s-t-iic--- => dualistic
image_path: test_data/5.jpg
ss-e--m--a-n-tticaal-lyy- => semantically
npu: noe_clean_job success
```




```
npu: noe_unload_graph success
npu: noe_deinit_context success
```

3. 37. 15. 2. onnx_PP_OCRv4 Case Study

This case is used to recognize text in an image and convert it into a computer-readable string.

1) Enter the folder where the case is located.

```
orangePi@orangePi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/OCR/onnx_PP_OCRv4
```

2) The command to use cpu for inference is as follows, using images from the test_data folder for inference.

```
orangePi@orangePi:~/ai_model_hub_25_Q3/models/ComputeVision/OCR/onnx_PP_OC
Rv4$ python3 inference_onnx.py
2025-10-17 11:00:11.338772178 [W:onnxruntime:Default, device_discovery.cc:164
DiscoverDevicesForPlatform] GPU device discovery failed: device_discovery.cc:89 ReadFileContents
Failed to open file: "/sys/class/drm/card3/device/vendor"
[[[292.0, 295.0], [335.0, 294.0], [350.0, 852.0], [307.0, 853.0]], ('土地整治与土壤修复研究中心',
0.957057535648346)]
[[[343.0, 298.0], [380.0, 297.0], [389.0, 665.0], [352.0, 666.0]], ('华南农业大学-东图',
0.9861801862716675)]
[[[36.0, 409.0], [486.0, 386.0], [489.0, 434.0], [38.0, 457.0]], ('上海斯格威铂尔大酒店',
0.9942368268966675)]
[[[183.0, 453.0], [401.0, 444.0], [403.0, 485.0], [185.0, 494.0]], ('打浦路 15 号',
0.9480958580970764)]
[[[14.0, 501.0], [519.0, 483.0], [521.0, 537.0], [15.0, 555.0]], ('绿洲仕格维花园公寓',
0.9961634874343872)]
[[[73.0, 550.0], [451.0, 539.0], [452.0, 576.0], [74.0, 587.0]], ('打浦路 2529 35 号→',
0.975419282913208)]
[[[34.0, 79.0], [440.0, 82.0], [439.0, 174.0], [33.0, 171.0]], ('纯臻营养护发素',
0.9949556589126587)]
[[[31.0, 183.0], [253.0, 183.0], [253.0, 243.0], [31.0, 243.0]], ('产品信息/参数',
0.9938035607337952)]
```



```
[[[39.0, 258.0], [469.0, 258.0], [469.0, 309.0], [39.0, 309.0]], (' (45 元 / 每公斤, 100 公斤起订',
0.981097936630249)]
[[[35.0, 325.0], [410.0, 327.0], [409.0, 382.0], [34.0, 380.0]], ('每瓶 22 元, 1000 瓶起订') ,
0.999458909034729)]
[[[34.0, 406.0], [435.0, 406.0], [435.0, 454.0], [34.0, 454.0]], (' 【品牌】 : 代加工方式/OEMODM',
0.9994497895240784)]
[[[32.0, 477.0], [341.0, 474.0], [341.0, 526.0], [32.0, 528.0]], (' 【品名】 : 纯臻营养护发素',
0.9984856247901917)]
[[[32.0, 549.0], [353.0, 549.0], [353.0, 600.0], [32.0, 600.0]], (' 【产品编号】 : YM-X-3011',
0.9997683763504028)]
[[[30.0, 621.0], [263.0, 617.0], [264.0, 668.0], [31.0, 672.0]], (' 【净含量】 : 220ml',
0.9565288424491882)]
[[[33.0, 692.0], [365.0, 695.0], [364.0, 743.0], [33.0, 740.0]], (' 【适用人群】 : 适合所有肤质',
0.9993979334831238)]
[[[32.0, 763.0], [499.0, 766.0], [498.0, 816.0], [32.0, 813.0]], (' 【主要成分】 : 鲸蜡硬脂醇、燕麦β-
葡聚', 0.9533689618110657)]
[[[38.0, 840.0], [407.0, 840.0], [407.0, 884.0], [38.0, 884.0]], ('糖、椰油酰胺丙基甜菜碱、泛醒',
0.9451623558998108)]
[[[525.0, 842.0], [690.0, 842.0], [690.0, 898.0], [525.0, 898.0]], (' (成品包材) ',
0.9980880618095398)]
[[[34.0, 910.0], [522.0, 910.0], [522.0, 957.0], [34.0, 957.0]], (' 【主要功能】 : 可紧致头发磷层, 从
而达到', 0.9985362887382507)]
[[[39.0, 983.0], [536.0, 983.0], [536.0, 1027.0], [39.0, 1027.0]], ('即时持久改善头发光泽的效果, 给
干燥的头', 0.9993788003921509)]
[[[32.0, 1051.0], [201.0, 1048.0], [202.0, 1104.0], [33.0, 1107.0]], ('发足够的滋养',
0.9753411412239075)]
```

3) The command to use npu for inference is as follows, using images from the test_data folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/OCR/onnx_PP_OC
Rv4$ python3 inference_npu.py
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 1.
npu: noe_create_job success
```



```
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 1.
npu: noe_create_job success
npu: noe_init_context success
npu: noe_load_graph success
Input tensor count is 1.
Output tensor count is 1.
npu: noe_create_job success
[[[292.0, 297.0], [335.0, 295.0], [350.0, 850.0], [307.0, 851.0]], ('土地整治与土壤修复研究中心',
0.976573646068573)]
[[[344.0, 299.0], [381.0, 298.0], [387.0, 662.0], [351.0, 663.0]], ('华南农业大学-东图',
0.9912211298942566)]
[[[36.0, 409.0], [486.0, 386.0], [489.0, 434.0], [38.0, 457.0]], ('上海斯格威铂尔大酒店',
0.9929969906806946)]
[[[141.0, 456.0], [403.0, 444.0], [404.0, 483.0], [143.0, 495.0]], ('←打浦路 15 号',
0.862202525138855)]
[[[17.0, 505.0], [519.0, 484.0], [521.0, 535.0], [19.0, 555.0]], ('绿洲仕格维花园公寓',
0.9960622787475586)]
[[[67.0, 550.0], [418.0, 539.0], [420.0, 578.0], [68.0, 590.0]], ('打浦路 25 29 35 号',
0.9729113578796387)]
[[[34.0, 78.0], [442.0, 80.0], [441.0, 174.0], [33.0, 171.0]], ('纯臻营养护发素',
0.9860424399375916)]
[[[30.0, 181.0], [255.0, 181.0], [255.0, 244.0], [30.0, 244.0]], ('产品信息/参数',
0.949313759803772)]
[[[39.0, 258.0], [478.0, 258.0], [478.0, 309.0], [39.0, 309.0]], ('（45 元 / 每公斤、100 公斤起订）',
0.9828777313232422)]
[[[36.0, 321.0], [411.0, 325.0], [411.0, 384.0], [35.0, 380.0]], ('每瓶 22 元、1000 瓶起订）',
0.9913207292556763)]
[[[37.0, 406.0], [432.0, 406.0], [432.0, 450.0], [37.0, 450.0]], ('【品牌】：代加工方式/OEM ODM',
0.9849441051483154)]
[[[31.0, 475.0], [342.0, 472.0], [342.0, 527.0], [31.0, 530.0]], ('【品名】：纯臻营养护发素',
0.9962107539176941)]
[[[593.0, 539.0], [623.0, 539.0], [623.0, 700.0], [593.0, 700.0]], ('ODM OEM',
0.9357462525367737)]
```



```

[[[31.0, 549.0], [353.0, 546.0], [353.0, 599.0], [31.0, 601.0]], ('【产品编号】：YM-X-3011',
0.9970366358757019)]
[[[29.0, 620.0], [264.0, 617.0], [264.0, 668.0], [30.0, 671.0]], ('【净含量】：220ml',
0.9971547722816467)]
[[[33.0, 691.0], [367.0, 694.0], [367.0, 742.0], [33.0, 739.0]], ('【适用人君】：适合所有肤质',
0.9611490964889526)]
[[[33.0, 764.0], [497.0, 767.0], [497.0, 813.0], [33.0, 811.0]], ('【主要成分】：鲸蜡硬脂醇、燕麦-
葡聚', 0.9434943795204163)]
[[[37.0, 839.0], [409.0, 839.0], [409.0, 886.0], [37.0, 886.0]], ('糖、椰油酰胺丙基甜菜碱、泛醒',
0.9171066880226135)]
[[[526.0, 843.0], [689.0, 843.0], [689.0, 896.0], [526.0, 896.0]], ('（成品色材）',
0.8261211514472961)]
[[[33.0, 908.0], [522.0, 910.0], [522.0, 957.0], [33.0, 955.0]], ('【主要功能】：可紧致头发磷层，从
而达到', 0.9950319528579712)]
[[[39.0, 983.0], [536.0, 983.0], [536.0, 1027.0], [39.0, 1027.0]], ('即时持久改善头发光泽的效果，给
干燥的头', 0.9946616291999817)]
[[[34.0, 1051.0], [201.0, 1051.0], [201.0, 1103.0], [34.0, 1103.0]], ('发定够的滋养',
0.9353836178779602)]
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success
npu: noe_clean_job success
npu: noe_unload_graph success
npu: noe_deinit_context success

```

3. 37. 16. Testing Method for Pose_Estimation Class Models in ComputeVision

Before testing the model, please install the dependency packages according to the instructions in the section on [installing NPU dependencies](#), otherwise the test will report errors.



3. 37. 16. 1. onnx_handpose Case

The purpose of this case is to accurately identify and locate the human hand from images or videos, and infer the spatial position and posture of key joints in the hand.

1) Enter the folder where the case is located.

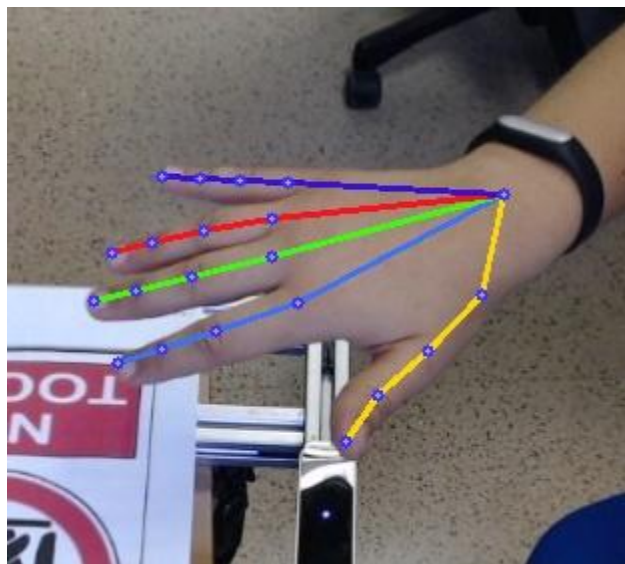
```
orangepi@orangepi:~$ cd \  
ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/onnx_handpose
```

2) The command to use CPU for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/onnx_handpos  
e$ python3 inference_onnx.py
```

3) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/onnx_handpos  
e$ ls output  
onnx_image_2021-02-04_20-06-04.jpg
```



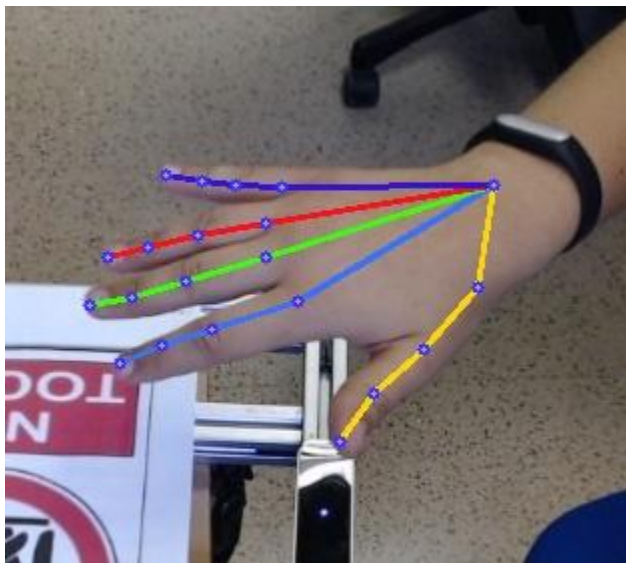
4) The command to use npu for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/onnx_handpos  
e$ python3 inference_npu.py
```



5) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/onnx_handpos  
e$ ls output  
npu_image_2021-02-04_20-06-04.jpg
```



3. 37. 16. 2. onnx_rnet_pose Case

The purpose of this case is to perform human pose estimation.

1) Enter the folder where the case is located.

```
orangepi@orangepi:~$ cd \  
ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/onnx_hrnet_pose
```

2) The command to use cpu for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/on  
nx_hrnet_pose$ python3 inference_onnx.py
```




```

orangepi@orangepi6plus:~/ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/onnx_hrnet_pose$ python3 inference_onnx.py
Inference time: 851.45 ms
Traceback (most recent call last):
  File "/home/orangepi/ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/onnx_hrnet_pose/inference_onnx.py", line 49, in <module>
    total_heatmap, peaks = hrnet(img)
    ^^^^^^^^^^^^^
  File "/home/orangepi/ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/onnx_hrnet_pose/HRNET_onnx.py", line 29, in __call__
    return self.update(image)
    ^^^^^^^^^^^^^
  File "/home/orangepi/ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/onnx_hrnet_pose/HRNET_onnx.py", line 87, in update
    self.total_heatmap, self.poses = self.process_output(outputs)
    ^^^^^^^^^^^^^
  File "/home/orangepi/ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/onnx_hrnet_pose/HRNET_onnx.py", line 123, in process_output
    peaks[max_vals < self.conf_threshold] = np.array([np.NaN, np.NaN])
    ^^^^^^^^^
  File "/home/orangepi/.local/lib/python3.11/site-packages/numpy/_init_.py", line 781, in __getattr__
    raise AttributeError(
AttributeError: 'np.NaN' was removed in the NumPy 2.0 release. Use 'np.nan' instead.
orangepi@orangepi6plus:~/ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/onnx_hrnet_pose$

```

If the above error occurs during testing, please reinstall the dependency package according to the instructions in the section on [installing NPU dependencies](#).

3) The images generated by inference will be saved in the output directory.

```

orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/onnx_hrnet_pose$ ls output
onnx_1.jpg

```



4) The command to use npu for inference is as follows, using images from the test_data folder for inference.

```

orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/onnx_hrnet_pose$ python3 inference_npu.py

```

5) The images generated by inference will be saved in the output directory.

```

orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/onnx_hrnet_pose$ ls output

```



npu_1.jpg



3. 37. 16. 3. onnx_pose Case

The purpose of this case is to detect real-time key points of multiple people's bodies, hands, and faces from a single image or video, and construct a complete posture skeleton for each person.

1) Enter the folder where the case is located.

```
orangepi@orangepi:~$ cd \  
ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/onnx_openpose
```

2) The command to use cpu for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/on  
nx_openpose$ python3 inference_onnx.py --image_path test_data
```

3) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/onnx_o  
npose$ ls output  
onnx_1.jpeg  onnx_2.jpeg
```

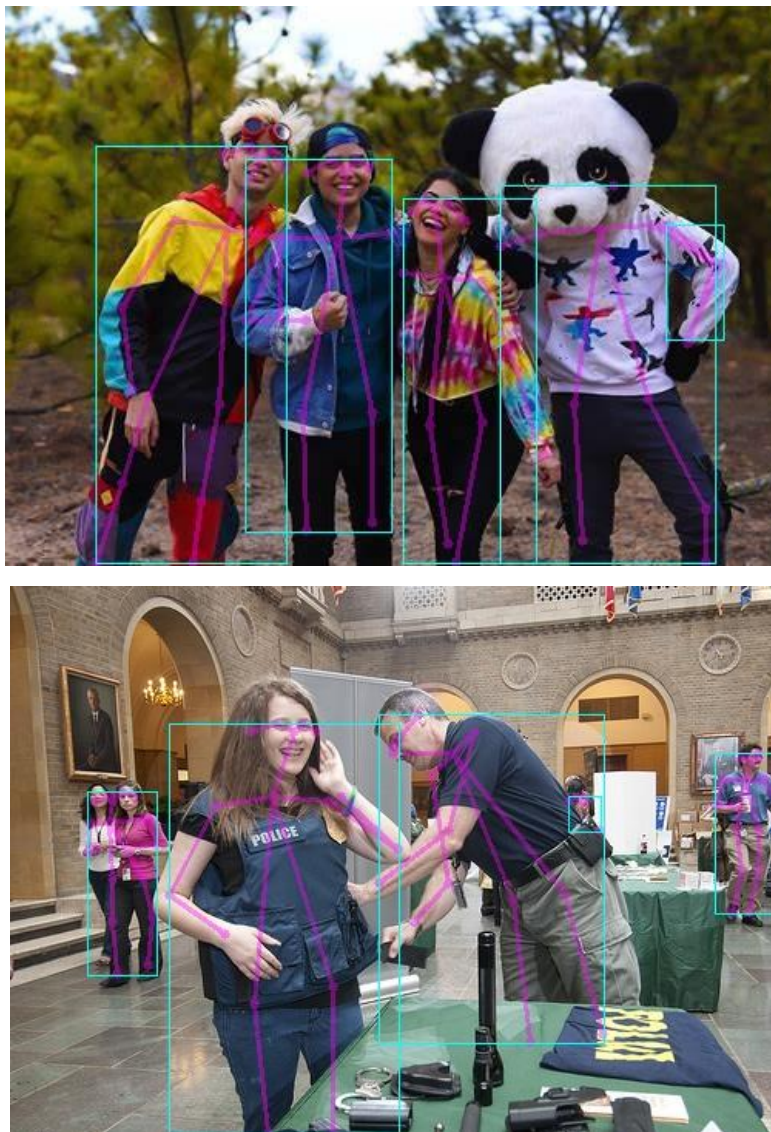


4) The command to use npu for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/on  
nx_openpose$ python3 inference_npu.py --image_path test_data
```

5) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/on  
nx_openpose$ ls output  
npu_1.jpeg  npu_2.jpeg
```

3. 37. 16. 4. onnx_yolov8s_pose Case

This case achieves real-time human pose estimation, which can quickly locate the human body in the image and mark the joint points.

1) Enter the folder where the case is located.

```
orange_pi@orange_pi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/onnx_yolov8s_pose
```

2) The command to use cpu for inference is as follows, using images from the test_data

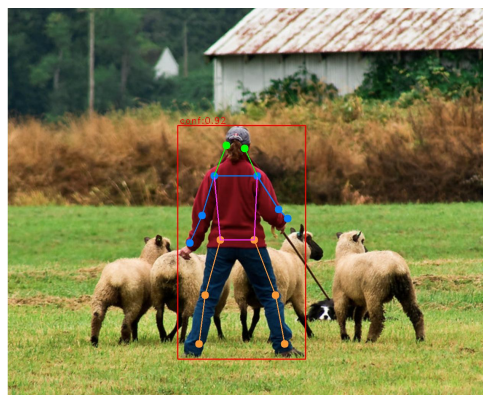
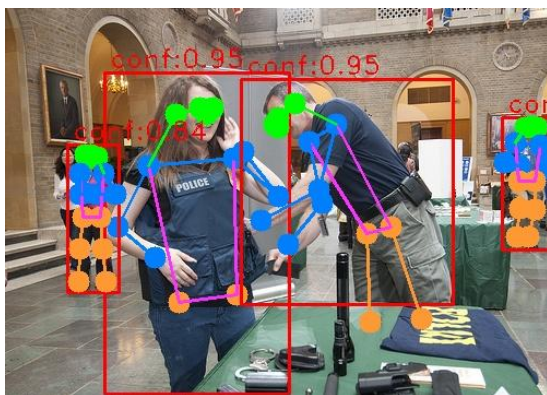


folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/onnx_yolov8s_pose$ python3 inference_onnx.py
```

3) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/onnx_yolov8s_pose$ ls output
onnx_1.png  onnx_2.jpeg
```

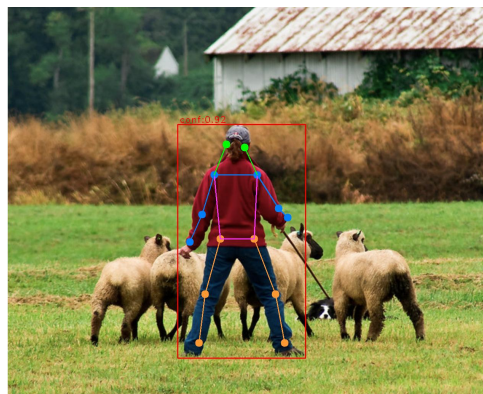
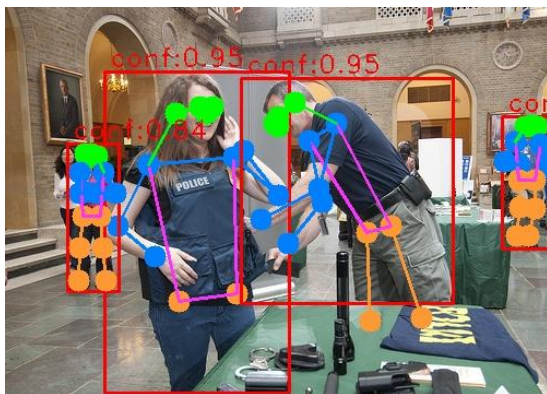


4) The command to use npu for inference is as follows, using images from the test_data folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/onnx_yolov8s_pose$ python3 inference_npu.py
```

5) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Pose_Estimation/onnx_yolov8s_pose$ ls output
npu_1.png  npu_2.jpeg
```





3. 37. 17. Testing Method for Semantic_Segmentation Class Models in ComputeVision

Before testing the model, please install the dependency packages according to the instructions in the section on [installing NPU dependencies](#), otherwise the test will report errors.

3. 37. 17. 1. onnx_BiSeNet Case

This case achieves pixel level precise segmentation of facial components such as eyes, nose, mouth, and contours.

1) Enter the folder where the case is located.

```
orangepi@orangepi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_BiSeNet
```

2) The command to use cpu for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_BiSeNet$ python3 inference_onnx.py
```

3) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_BiSeNet$ ls output
onnx_1319.jpg  onnx_1333.jpg  onnx_1.jpg  onnx_2095.jpg  onnx_2125.jpg
```



4) The command to use npu for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_BiSeNet$ python3 inference_npu.py
```




5) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_BiSeNet$ ls output
npu_1319.jpg  npu_1.jpg  npu_2125.jpg  npu_1333.jpg  npu_2095.jpg
```



3. 37. 17. 2. onnx_deeplab_v3

DeepLab V3 is a semantic segmentation model based on deep learning, mainly used to solve pixel level classification problems of multi-scale objects in images. Its core function is to achieve high-precision pixel level semantic segmentation through innovative technology.

1) Enter the folder where the case is located.

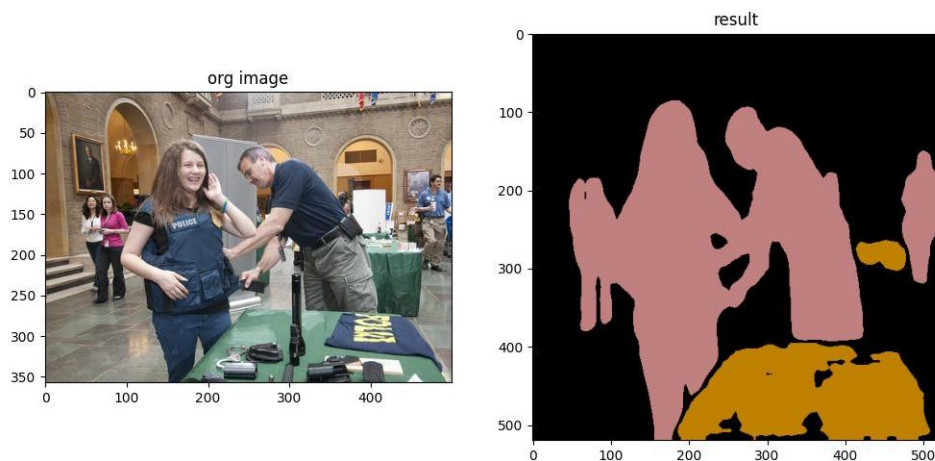
```
orangepi@orangepi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_deeplab_v3
```

2) The command to use cpu for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_deeplab_v3$ python3 inference_onnx.py --image_path test_data \
--onnx_path model/deeplabv3_resnet50.onnx
```

3) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_deeplab_v3$ ls output
onnx_ILSVRC2012_val_00004704.JPEG
```

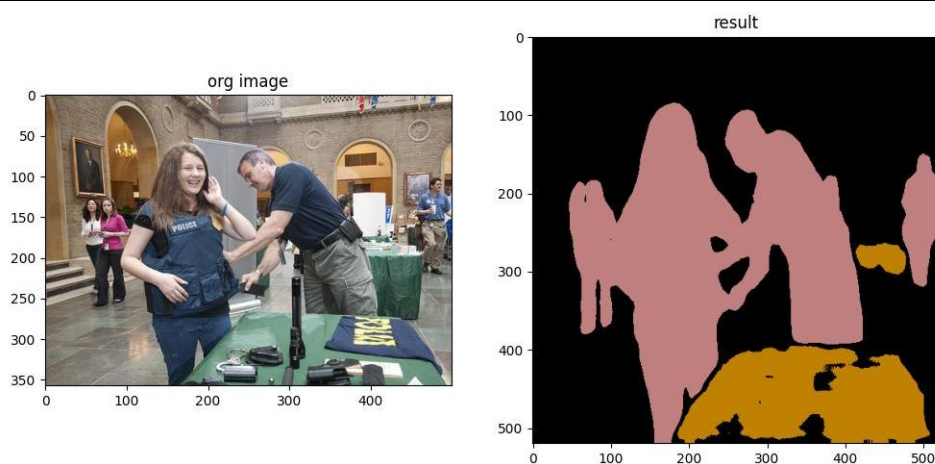


4) The command to use npu for inference is as follows, using images from the test_data folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_deeplab_v3$ python3 inference_npu.py --image_path test_data \
--model_path deeplab_v3.cix
```

5) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_deeplab_v3$ ls output
npu_ILSVRC2012_val_00004704.JPEG
```



3. 37. 17. 3. onnx_deeplabv3_plus Case

DeepLabV3+ is a semantic segmentation model based on encoder decoder



architecture, which achieves pixel level accurate segmentation through dilated convolution and multi-scale feature fusion technology, especially in complex scenes such as autonomous driving and medical imaging.

1) Enter the folder where the case is located.

```
orange@orange:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_deeplabv3_plus
```

2) The command to use cpu for inference is as follows, using images from the test_data folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_deeplabv3_plus$ python3 inference_onnx.py --image_path test_data \
--onnx_path model/deeplabv3plus.onnx
```

3) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_deeplabv3_plus$ ls output
onnx_bus.jpg  onnx_ILSVRC2012_val_00004704.JPEG
onnx_ILSVRC2012_val_00037133.JPEG  onnx_ILSVRC2012_val_00045790.JPEG
```





4) The command to use npu for inference is as follows, using images from the test_data folder for inference.



```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_deeplabv3_plus$ python3 inference_npu.py --image_path test_data \
--model_path deeplabv3plus.cix
```

5) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_deeplabv3_plus$ ls output
npu_bus.jpg      npu_ILSVRC2012_val_00045790.JPEG
npu_ILSVRC2012_val_00004704.JPEG  npu_ILSVRC2012_val_00037133.JPEG
```





3. 37. 17. 4. onnx_duc Case

The DUC model achieves finer image semantic segmentation through learnable dense upsampling and mixed dilated convolution techniques, particularly adept at restoring details and segmenting multi-scale targets.

1) Enter the folder where the case is located.

```
orange@orange:~$ cd \
```

```
ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_duc
```

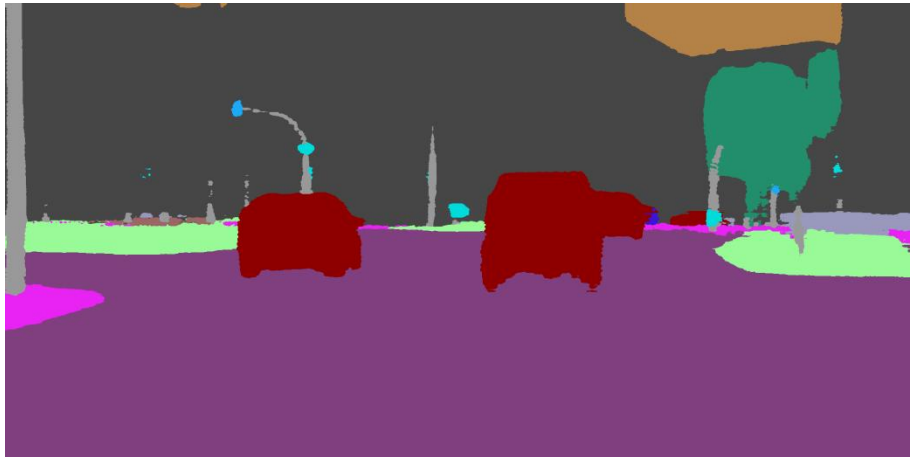



2) The command to use cpu for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_duc$ python3 inference_onnx.py --image_path test_data \  
--onnx_path model/duc.onnx
```

3) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_duc$ ls output  
onnx_city.png
```

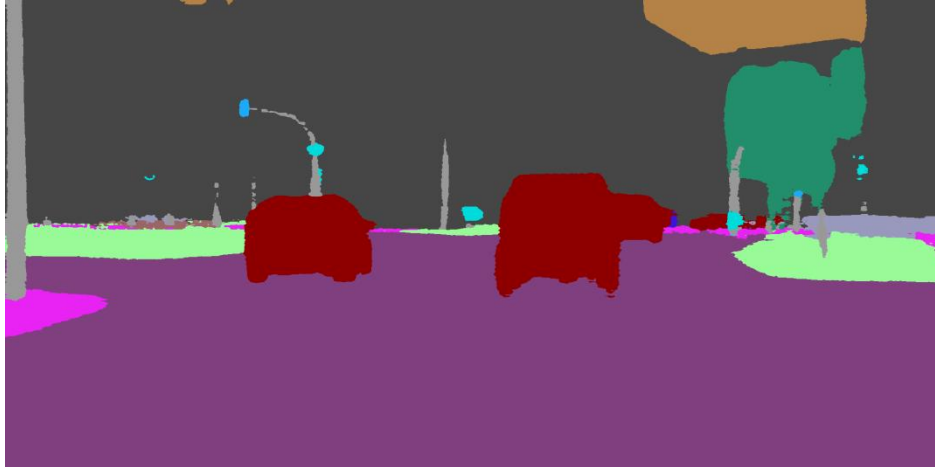


4) The command to use npu for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_duc$ python3 inference_npu.py --image_path test_data \  
--model_path duc.cix
```

5) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_duc$ ls output  
npu_city.png
```



3. 37. 17. 5. **onnx_fcn_resnet50 case study**

The core function of this case is to achieve efficient and accurate pixel level semantic segmentation through the combination of fully convolutional network (FCN) and ResNet50 backbone network.

1) Enter the folder where the case is located.

```
orangePi@orangePi:~$ cd \  
ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_fcn_resnet50
```

2) The command to use cpu for inference is as follows, using images from the test_data folder for inference.

```
orangePi@orangePi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_fcn_resnet50$ python3 inference_onnx.py --image_path test_data \  
--onnx_path model/fcn_resnet50.onnx
```

3) The images generated by inference will be saved in the output directory.

```
orangePi@orangePi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_fcn_resnet50$ ls output  
onnx_1.png
```

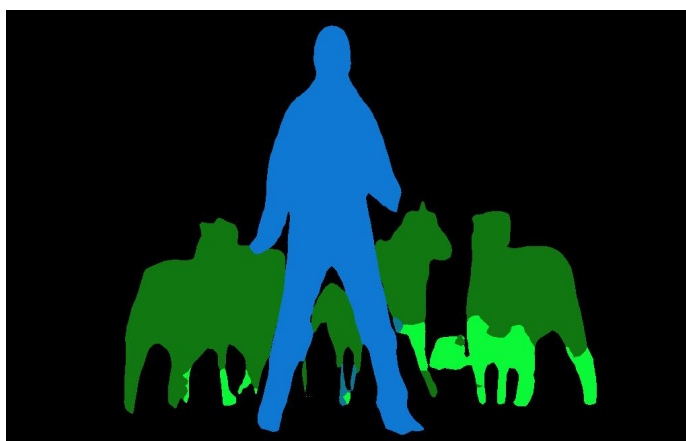


4) The command to use npu for inference is as follows, using images from the test_data folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_fcn_resnet50$ python3 inference_npu.py --image_path test_data \
--model_path fcn_resnet50.cix
```

5) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_fcn_resnet50$ ls output
npu_1.png
```



3. 37. 17. 6. onnx_pphumanseg Case

The PPHumanSeg model is specifically designed for high-precision real-time portrait



segmentation, capable of accurately separating people from backgrounds in images or videos.

1) Enter the folder where the case is located.

```
orangePi@orangePi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_pphumanseg
```

2) The command to use cpu for inference is as follows, using images from the test_data folder for inference.

```
orangePi@orangePi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_pphumanseg$ python3 inference_onnx.py
```

3) The images generated by inference will be saved in the output directory.

```
orangePi@orangePi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_pphumanseg$ ls output
onnx_000000000785.jpg  onnx_0000000009448.jpg  onnx_000000017905.jpg
onnx_000000021604.jpg  onnx_000000000872.jpg  onnx_000000011051.jpg
onnx_000000020333.jpg  onnx_000000021879.jpg  onnx_000000001296.jpg
onnx_000000016598.jpg  onnx_000000021167.jpg
```



4) The command to use npu for inference is as follows, using images from the test_data folder for inference.

```
orangePi@orangePi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation
```



```
ation/onnx_pphumanseg$ python3 inference_npu.py
```

5) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_pphumanseg$ ls output
npu_000000000785.jpg      npu_000000001296.jpg      npu_000000011051.jpg
npu_000000017905.jpg      npu_000000021167.jpg      npu_000000021879.jpg
npu_000000000872.jpg      npu_000000009448.jpg      npu_000000016598.jpg
npu_000000020333.jpg      npu_000000021604.jpg
```



3. 37. 17. 7. onnx_load_degmentation Case

In this case, the pixel level semantic segmentation technology is used to identify the driving areas (such as lanes and sidewalks) and obstacles in images or videos, so as to provide the auto drive system with an environment awareness basis.

1) Enter the folder where the case is located.

```
orange@orange:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_road_segmentation
```

2) The command to use cpu for inference is as follows, using images from the test_data folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_road_segmentation$ python3 inference_onnx.py
```

3) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_road_segmentation$ ls output
onnx_2.png
```



4) The command to use npu for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_road_segmentation$ python3 inference_npu.py
```

5) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_road_segmentation$ ls output
npu_2.png
```




3. 37. 17. 8. onnx_sam2_1_tiny Case

The role of SAM2 (Segment Anything Model 2) model is to achieve universal zero sample image segmentation, which can quickly segment any object in the image based on various prompts.

1) Enter the folder where the case is located.

```
orangepi@orangepi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_sam2_1_tiny
```

2) The command to use cpu for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_sam2_1_tiny$ python3 inference_onnx.py
```

3) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_sam2_1_tiny$ ls output
onnx_0truck.jpg
```



4) The command to use npu for inference is as follows, using images from the test_data folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_sam2_1_tiny$ python3 inference_npu.py
```

5) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_sam2_1_tiny$ ls output
npu_0truck.jpg
```



3. 37. 17. 9. onnx_u2net Case

This case can accurately separate the foreground and background in an image, and is suitable for tasks such as document photo extraction and portrait cutout.



1) Enter the folder where the case is located.

```
orangepi@orangepi:~$ cd \  
ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_u2net
```

2) The command to use cpu for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_u2net$ python3 inference_onnx.py
```

3) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_u2net$ ls output  
onnx_000000000785.jpg  onnx_000000000872.jpg  onnx_000000011051.jpg  
onnx_000000020333.jpg
```



4) The command to use npu for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_u2net$ python3 inference_npu.py
```

5) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_u2net$ ls output  
npu_000000000785.jpg      npu_000000000872.jpg      npu_000000011051.jpg  
npu_000000020333.jpg
```



3. 37. 17. 10. **nnx_yolov8_segmentation case**

The role of the YOLOv8-segmentation model is to perform instance segmentation while performing real-time object detection. It can simultaneously recognize multiple objects in an image, locate their positions, and generate pixel level accurate masks.

1) Enter the folder where the case is located.

```
orangepi@orangepi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/nnx_yolov8_segmentation
```

2) The command to use cpu for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/nnx_yolov8_segmentation$ python3 inference_onnx.py --image_path \
test_data --onnx_path model/yolov8l_seg.onnx
```

3) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/nnx_yolov8_segmentation$ ls output
onnx_1.jpeg  onnx_2.jpeg
```



4) The command to use npu for inference is as follows, using images from the test_data folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_yolov8_segmentation$ python3 inference_npu.py --image_path \
test_data --model_path yolov8l_seg.cix
```

5) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/onnx_yolov8_segmentation$ ls output
npu_1.jpeg  npu_2.jpeg
```




3. 37. 17. 11. torch_fast_scnn case study

This case is mainly applied in scenarios that require fast processing of high-resolution images, such as real-time recognition of roads, pedestrians, and obstacles by vehicles.

1) Enter the folder where the case is located.

```
orangepi@orangepi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/torch_fast_scnn
```

2) The command to use cpu for inference is as follows, using images from the test_data folder for inference.



```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/torch_fast_scnn$ python3 inference_pt.py
```

3) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/torch_fast_scnn$ ls output  
torch_frankfurt_000001_058914_leftImg8bit.png
```

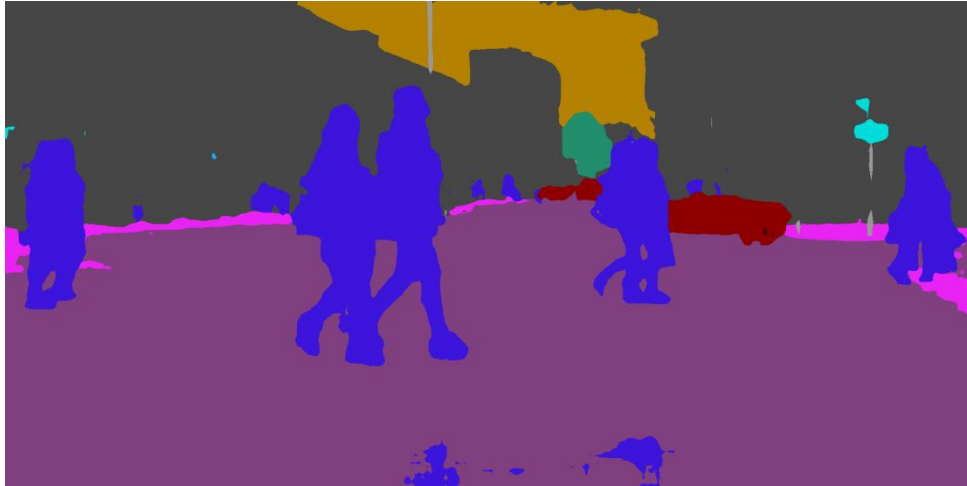


4) The command to use npu for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/torch_fast_scnn$ python3 inference_npu.py
```

5) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Semantic_Segmentation/torch_fast_scnn$ ls output  
npu_frankfurt_000001_058914_leftImg8bit.png
```



3. 37. 18. Testing Method for Super_Resolution Class Models in ComputeVision

Before testing the model, please install the dependency packages according to the instructions in the section on [installing NPU dependencies](#), otherwise the test will report errors.

3. 37. 18. 1. onnx_real_eskan Case

The core function of the Real ESRGAN model is to efficiently enlarge and restore low resolution, blurry, or noisy images into high-resolution, detail rich super-resolution images.

1) Enter the folder where the case is located.

```
orangepi@orangepi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Super_Resolution/onnx_real_esrgan
```

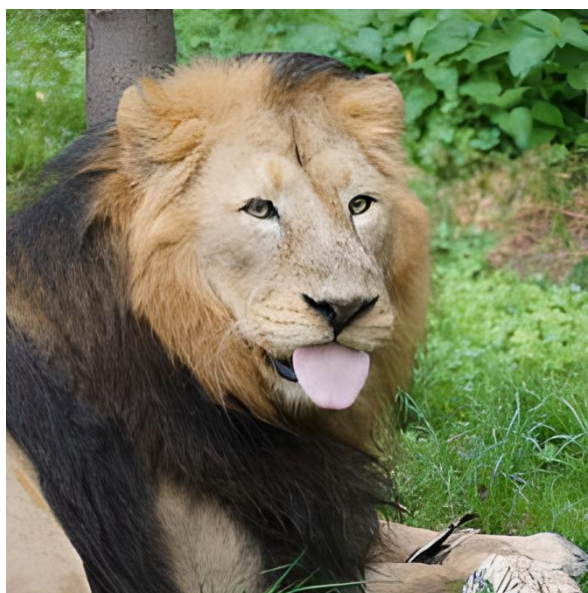
2) The command to use cpu for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Super_Resolution/o
nnx_real_esrgan$ python3 inference_onnx.py
```



3) The images generated by inference will be saved in the output directory.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Super_Resolution/onnx_real_esrgan$ ls output
onnx_1.png
```



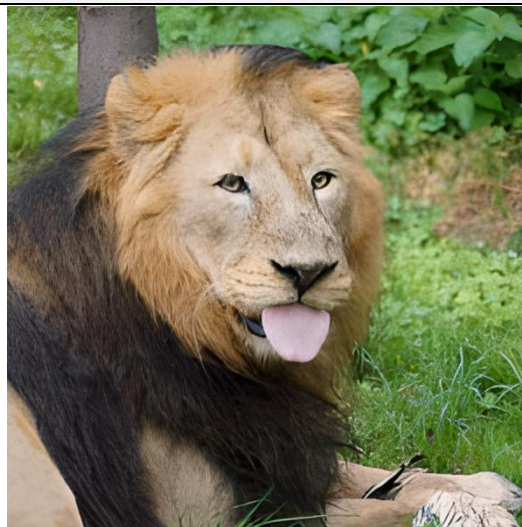
4) The command to use npu for inference is as follows, using images from the test_data folder for inference.

```
orange@orange:~/ai_model_hub_25_Q3/models/ComputeVision/Super_Resolution/onnx_real_esrgan$ python3 inference_npu.py
```



5) The images generated by inference will be saved in the output directory.

```
orange_pi@orange_pi:~/ai_model_hub_25_Q3/models/ComputeVision/Super_Resolution/onnx_real_esrgan$ ls output
npu_1.png
```



3. 37. 18. 2. onnx_super_resolution Case

The core function of the Super Resolution model is to enlarge and reconstruct low resolution images into high-resolution images, while intelligently supplementing details to enhance their clarity and visual quality.

1) Enter the folder where the case is located.

```
orange_pi@orange_pi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Super_Resolution/onnx_super_resolution
```



2) The command to use cpu for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Super_Resolution/o  
nnx_super_resolution$ python3 inference_onnx.py
```



3) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Super_Resolution/o  
nnx_super_resolution$ ls output  
onnx_1.png
```



4) The command to use npu for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Super_Resolution/o  
nnx_super_resolution$ python3 inference_npu.py
```




5) The images generated by inference will be saved in the output directory.

```
orange_pi@orange_pi:~/ai_model_hub_25_Q3/models/ComputeVision/Super_Resolution/onnx_super_resolution$ ls output
npu_1.png
```



3.37.18.3. onnx_vdsr Case

The VDSR model is a super-resolution tool based on deep residual learning, which achieves precise image enlargement and clarity by specifically learning and completing high-frequency details missing from the image.

1) Enter the folder where the case is located.

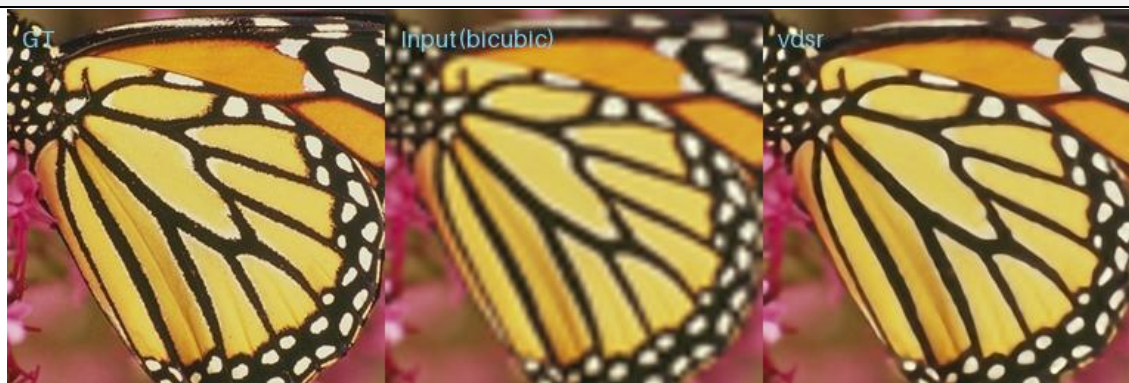
```
orange_pi@orange_pi:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Super_Resolution/onnx_vdsr
```


2) The command to use cpu for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Super_Resolution/onnx_vdsr$ python3 inference_onnx.py --image_path test_data --onnx_path \
model/vdsr.onnx
```

3) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Super_Resolution/onnx_vdsr$ ls output
onnx_butterfly.png
```



4) The command to use npu for inference is as follows, using images from the test_data folder for inference.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Super_Resolution/onnx_vdsr$ python3 inference_npu.py --image_path test_data --model_path vdsr.cix
```

5) The images generated by inference will be saved in the output directory.

```
orangepi@orangepi:~/ai_model_hub_25_Q3/models/ComputeVision/Super_Resolution/onnx_vdsr$ ls output
npu_butterfly.png
```



3. 38. Usage of NOE SDK

3. 38. 1. Application Method for NOE SDK

The NOE SDK is an NPU software toolkit provided by this chip for developers. Here, we mainly use the NOE compiler inside to compile models. At present, the NOE SDK for this chip has not been publicly released and needs to be obtained through the **Early Bird** Program. The specific steps are as follows:

- 1) Open the following website:

<https://earlybird.cixtech.com>

- 2) Then fill in the relevant information according to the prompts.

CIX

此丽科技

English

此芯早鸟计划注册登记单

· 用户名称

请输入您的用户名

· 公司名称

请输入您的公司名称

· 公司邮箱

请输入您的邮箱

手机号码

请输入您的手机号

· 下载类型

NeuralOne AI SDK ☒

魔搭账号

请提供魔搭账号！

验证码

28021

☐ 勾选同意 《终端用户许可协议》

下载类型说明

同意并创建帐户

这里需要选择NeuralOne AI SDK



3) Then click **agree and create an account**.

4) After the review is approved, you will receive an email containing a download link, which roughly reads as follows:

感谢您参与此芯早鸟计划! 以下为下载须知:

Thank you for participating in Cix early bird program! The following are download instructions:

NeuralOne AI SDK download URL is <https://earlybird.cixtech.com/download?>



5) Click on the download link to download the NOE SDK. The downloaded compressed file is shown below. When writing the manual, the latest version of the NOE SDK is **25_q3**.

cix_noe_sdk_25_q3_release.tar.gz

3. 38. 2. Installation Method of NOE SDK

Note that all the following operations are run on an X86_64 computer, not on the development board.

1) Please download the compressed file of NOE SDK first. For specific instructions, please refer to the section **on application methods of NOE SDK**.

2) The NOE compiler provided in the NOE SDK——CixBuilder, has the following runtime environment requirements:

- Linux operating system with X86_64 architecture;
- GNU libc version 2.17 or above;
- Only supports Python3.10 runtime environment.

3) Here we choose the Ubuntu22.04.5 operating system with X86_64 for demonstration. Please ensure that other versions of Linux systems meet the above requirements.

- The installation image download address for Ubuntu 22.04 **amd64** version is:

<https://mirrors.tuna.tsinghua.edu.cn/ubuntu-releases/22.04/ubuntu-22.04.5-desktop-amd64.iso>

或者

<https://repo.huaweicloud.com/ubuntu-releases/22.04/ubuntu-22.04.5-desktop-amd64.iso>



- b. The version of Ubuntu 22.04 GNU libc is as follows, meeting the requirements of GNU libc version 2.17 or above.

```
test@test:~$ ldd --version | grep GLIBC
ldd (Ubuntu GLIBC 2.35-0ubuntu3.11) 2.35
```

- c. The default Python 3 version for Ubuntu 22.04 is **3.10.12**, which meets the above requirements. If you are using a different version of Linux system and the default Python version is not Python3.10, you can refer to the instructions in the section on [installing Anaconda](#) to create an isolated Python3.10 virtual environment using conda.

```
test@test:~$ python3 -V
Python 3.10.12
```

- 4) Upload the downloaded **cix_noe_sdk_25_q3_release.tar.gz** to the Ubuntu system.

- 5) Use the following command to decompress **cix_noe_sdk_25_q3_release.tar.gz**.

```
test@test:~$ tar -zxvf cix_noe_sdk_25_q3_release.tar.gz
```

- 6) The files included after decompression are shown below. The two deb packages, **cix-npu-driver** and **cix-noe-umd**, are pre installed by default in the Linux image of the development board and do not need to be reinstalled. Here we mainly use the whl package CixBuilder, which is only available in x86_64 version.

```
test@test:~$ ls cix_noe_sdk_25_q3_release
CixBuilder-6.1.3407.2-cp310-none-linux_x86_64.whl
cix-npu-driver_2.0.1_arm64.deb
npu_sdk_last_manifest_list.xml
cix-noe-umd_2.0.2_arm64.deb
env_setup.sh
requirements.txt
```

- 7) Install Python dependency packages.

- a. The requirements.txt file needs to be modified before it can be installed properly. The modified requirements. txt file is shown below. Please save the following content to the requirements. txt file.

```
test@test:~/cix_noe_sdk_25_q3_release$ vim requirements.txt
absl-py==2.2.2
```



```
astunparse==1.6.3
attrs==25.3.0
certifi==2025.4.26
charset-normalizer==3.4.2
clang==20.1.0
cloudpickle==3.1.1
contourpy==1.3.2
cyclers==0.12.1
decorator==5.2.1
editdistance==0.8.1
filelock==3.18.0
flatbuffers==25.2.10
fonttools==4.58.0
fsspec==2025.3.2
future==1.0.0
google-pasta==0.2.0
grpcio==1.72.0
h5py==3.13.0
idna==3.10
Jinja2==3.1.6
keras==2.14.0
kiwisolver==1.4.8
libclang==18.1.1
Markdown==3.8
markdown-it-py==3.0.0
MarkupSafe==3.0.2
matplotlib==3.10.3
mdurl==0.1.2
ml-dtypes==0.2.0
mpmath==1.3.0
namex==0.0.9
networkx==3.4.2
numpy==1.24.3
nvidia-cublas-cu12==12.6.4.1
nvidia-cuda-cupti-cu12==12.6.80
nvidia-cuda-nvrtc-cu12==12.6.77
```



```
nvidia-cuda-runtime-cu12==12.6.77
nvidia-cudnn-cu12==9.5.1.17
nvidia-cufft-cu12==11.3.0.4
nvidia-cufile-cu12==1.11.1.6
nvidia-curand-cu12==10.3.7.77
nvidia-cusolver-cu12==11.7.1.2
nvidia-cusparselt-cu12==0.6.3
nvidia-nccl-cu12==2.26.2
nvidia-nvjitlink-cu12==12.6.85
nvidia-nvtx-cu12==12.6.77
onnx==1.17.0
onnxoptimizer==0.3.13
onnxsim==0.4.36
opencv-python==4.11.0.86
opt_einsum==3.4.0
packaging==25.0
pillow==11.2.1
protobuf==3.20.3
psutil==7.0.0
pyfma==0.1.6
Pygments==2.19.1
pyparsing==3.2.3
python-dateutil==2.9.0.post0
PyYAML==6.0.2
requests==2.32.3
scipy==1.15.3
setuptools==80.4.0
six==1.17.0
sympy==1.14.0
synr==0.6.0
tensorboard==2.14.0
tensorboard-data-server==0.7.2
tensorflow==2.14.0
tensorflow-io-gcs-filesystem==0.37.1
termcolor==3.1.0
```




```
torch==2.7.0
torchvision==0.22.0
tornado==6.4.2
tqdm==4.67.1
triton==3.3.0
urllib3==2.4.0
Werkzeug==3.1.3
wheel==0.46.1
wrapt==1.14
pandas
```

b. The command to install dependency packages is as follows:

```
test@test:~/cix_noe_sdk_25_q3_release$ pip3 install -r requirements.txt
```

If the download speed of Python packages is slow, you can use the following method to replace the Python source:

1. Create a pip configuration directory

```
test@test:~/cix_noe_sdk_25_q3_release$ mkdir -p ~/.pip
```

2. Edit configuration file

```
test@test:~/cix_noe_sdk_25_q3_release$ cat > ~/.pip/pip.conf << EOF
```

```
[global]
```

```
index-url = https://pypi.tuna.tsinghua.edu.cn/simple
```

```
trusted-host = pypi.tuna.tsinghua.edu.cn
```

```
timeout = 6000
```

```
EOF
```

3. Then run the installation command again.

8) Execute the following command to install the NOE compiler.

```
test@test:~/cix_noe_sdk_25_q3_release$ ./env_setup.sh
```

9) After installation, use the following command. If you can see the version number, it indicates that the NOE compiler has been successfully installed.

```
test@test:~/cix_noe_sdk_25_q3_release$ cixbuild -v
```



```
/home/test/.local/bin/cixbuild version: 6.1.3407
```

3. 38. 3. Example of using NOE SDK

This section describes how to compile and run a neural network model using the CIX NOE SDK. Using ResNet50 as an example, demonstrate how to process pre trained models step by step and run them on the NPU of the development board. The complete case (including code and script) can be obtained from CIX AI Model Hub.

1) Please install the NOE compiler in the NOE SDK first. For specific instructions, please refer to the [installation method section of the NOE SDK](#).

2) Then download the CIX AI Model Hub on the X86_64 Linux system. Please refer to the instructions in the section on [downloading the CIX AI Model Hub](#) for the download method. The downloaded code repository is shown below:

```
test@test:~$ ls ai_model_hub_25_Q3
configuration.json  demos  LICENSE  README.md  requirements.txt
utils  datasets  EULA.md  models  ReleaseNote.md  scripts
```

3) The following demonstrates the method of compiling and deploying the model using the **onnx_resnet_v1_50** case as an example. Firstly, enter the directory where the onnx_resnet_v1_50 case is located in the CIX AI Model Hub.

```
test@test:~$ cd \
ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_50
```

4) The files included in the onnx_resnet_v1_50 case are shown below. By default, the converted model files resnet50-v1-12-sim.onnx and resnet_v1_50.cix can be directly used. We can delete them first and then manually generate them. If we do not delete them, we can directly generate an overwrite afterwards.

```
test@test:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_50$ tree
.
├── cfg
│   └── onnx_resnet_v1_50build.cfg
├── datasets
│   └── calib_data.npy
└── inference_npu.py
```



```
|—— inference_onnx.py
|—— label.txt
|—— main.cpp
|—— makefile
|—— model
|   |—— LICENSE.txt
|   |—— resnet50-v1-12.onnx
|   |—— resnet50-v1-12-sim.onnx
|—— noe_utils
|   |—— cix_noe_standard_api.h
|—— ReadMe.md
|—— resnet_v1_50.cix
|—— test_data
|   |—— ILSVRC2012_val_00002899.JPEG
|   |—— ILSVRC2012_val_00021564.JPEG
|   |—— ILSVRC2012_val_00024154.JPEG
|   |—— ILSVRC2012_val_00037133.JPEG
|   |—— ILSVRC2012_val_00045790.JPEG
|—— Tutorials.ipynb
```

5 directories, 19 files

5) Using the ONNX Simplifier tool to simplify and optimize the ONNX model is optional. If you skip this step, please note that the **input_model** parameter in the cfg configuration file needs to be modified accordingly.

```
test@test:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_50$ onnxsim model/resnet50-v1-12.onnx model/resnet50-v1-12-sim.onnx
```



Simplifying ...
Finish! Here is the difference:

	Original Model	Simplified Model
Add	16	16
BatchNormalization	53	0
Constant	299	108
Conv	53	53
Flatten	1	1
Gemm	1	1
GlobalAveragePool	1	1
MaxPool	1	1
Relu	49	49
Model Size	97.8MiB	97.4MiB

- a. Command parsing:
 - a) **onnxsim**: Command line tool for ONNX Simplifier
 - b) **model/resnet50-v1-12.onnx**: Original ONNX model file input
 - c) **model/resnet50-v1-12-sim.onnx**: Simplified ONNX model file output
- b. onnxsim mainly performs the following optimizations:
 - a) Operator fusion: Merge BatchNormalization with adjacent Conv layers
 - b) Constant folding: precomputing static constant expressions
 - c) Redundant node elimination: Remove nodes that do not affect output
 - d) Graph structure optimization: simplifying the calculation of graph topology

6) A calibration data has been prepared for the onnx_resnet_v1_50 case, and new calibration data can be generated if necessary. Please note that the **calibration_data** parameter in the cfg configuration file after using the self generated calibration dataset needs to be modified accordingly.

- a. The default calibration data is as follows:

```
test@test:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_50$ ls datasets
calib_data.npy
```

- b. The method of generating calibration data by oneself is as follows:
 - a) Prepare the calibration dataset. The **test_data** directory already contains multiple image files of the calibration dataset, and more can be added.

```
test@test:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_50$ tree test_data
test_data/
```



```
|—— ILSVRC2012_val_00002899.JPEG  
|—— ILSVRC2012_val_00021564.JPEG  
|—— ILSVRC2012_val_00024154.JPEG  
|—— ILSVRC2012_val_00037133.JPEG  
|—— ILSVRC2012_val_00045790.JPEG
```

0 directories, 5 files

b) Install dependency packages:

```
test@test:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_re  
snet_v1_50$ pip3 install imageio
```

c) Then use the following python script to generate a new calibration file:

```
test@test:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_re  
snet_v1_50$ vim gen_calib_data.py  
import sys  
import os  
import numpy as np  
_abs_path = os.path.join(os.getcwd(), ".././.././")  
sys.path.append(_abs_path)  
from utils.image_process import imagenet_preprocess_method1  
  
from utils.tools import get_file_list  
# Get a list of images from the provided path  
images_path = "test_data"  
images_list = get_file_list(images_path)  
data = []  
for image_path in images_list:  
    input = imagenet_preprocess_method1(image_path)  
    data.append(input)  
# concat the data and save calib dataset  
data = np.concatenate(data, axis=0)  
print(data.shape)  
np.save("datasets/calib_data_new.npy", data)  
print("Generate calib dataset success.")
```



```
test@test:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_50$ python3 gen_calib_data.py
(5, 3, 224, 224)
Generate calib dataset success.
```

d) The newly generated calibration dataset is **datasets/calib_data_new.npy**.

```
test@test:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_50$ ls datasets
calib_data_new.npy  calib_data.npy
```

7) The default content of the **cfg** configuration file used for quantization and compilation is as follows:

```
test@test:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_50$ cat cfg/onnx_resnet_v1_50build.cfg
[Common]
mode = build

[Parser]
model_type = onnx
model_name = resnet_v1_50
detection_postprocess =
model_domain = image_classification
input_model = model/resnet50-v1-12-sim.onnx
output_dir = ./out
input_shape = [1, 3, 224, 224]
input = data

[Optimizer]
output_dir = ./out
calibration_data = datasets/calib_data.npy
calibration_batch_size = 16
dataset = numpydataset
save_statistic_info = True
cast_dtypes_for_lib = True
global_calibration = adaround[10, 10, 32, 0.01]
```




```
[GBuilder]
target = X2_1204MP3
outputs = resnet_v1_50.cix
```

8) The command to compile the model using the **cixbuild** tool through calibration data is as follows:

a. Set environment variables:

```
test@test:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_50$ export \
LD_LIBRARY_PATH=~/.local/lib/python3.10/site-packages/AIPUBuilder/simulator-lib:$LD_LIBRARY_PATH
```

b. The command to compile the model is as follows:

```
test@test:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_50$ cixbuild cfg/onnx_resnet_v1_50build.cfg
[I] Build with version 6.1.3407
[I] Parsing model....
[I] [Parser]: Begin to parse onnx model resnet_v1_50...
.....
[I] [Parser]: The input tensor(s) is/are: data_0
[I] [Parser]: Input data from cfg is shown as tensor data_0 in IR!
[I] [Parser]: 0 error(s), 0 warning(s) generated.
[I] [Parser]: Parser done!
[I] [Parser]: Parser cost 5.21 seconds.
[I] Parse model complete
[I] Simplifying float model.
[I] [IRChecker] Start to check IR:
/home/test/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_50/out/internal_2025_10_24_22_9_34_rth1u/resnet_v1_50.txt
[I] [IRChecker] model_name: resnet_v1_50
[I] [IRChecker] IRChecker: All IR pass (Checker Plugin disabled)
[I] [graph.cpp :1605] loading graph weight:
/home/test/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_50/./out/internal_2025_10_24_22_9_34_rth1u/resnet_v1_50.bin size: 0x61641a0
[I] Start to simplify the graph...
[I] Using fixed-point full optimization, it may take long long time ....
[I] Simplify Done.
[I] Simplify float model Done.
```



```
[I] Optimizing model...
[I] [OPT] [22:09:39]: [arg_parser] is running.
[I] [OPT] [22:09:39]: tool name: Compass-Optimizer, version: 1.3.3407, use cuda: False,
running device: cpu
[I] [OPT] [22:09:39]: [quantization config Info][model name]: resnet_v1_50,
[quantization method for weight]: per_tensor_symmetric_restricted_range, [quantization
method for activation]: per_tensor_symmetric_full_range, [calibration strategy for weight]:
extrema, [calibration strategy for activation]: mean, [quantization precision]:
activation_bits=8, weight_bits=8, bias_bits=32, lut_items_in_bits=8

[I] [OPT] [22:09:39]: Suggest using "aipuchecker" to validate the IR firstly if you are not
sure about its validity.
[I] [OPT] [22:09:39]: IR loaded.
Building graph: 100%|██████████| 75/75 [00:00<00:00, 679.34it/s]
[I] [OPT] [22:09:39]: Begin to load weights.
[I] [OPT] [22:09:39]: Weights loaded.
.....
[I] [aipu_plugin.cpp: 344] Concat(resnetv17_stage1__plus1/tile/concat) uses
performance-lib
[I] [aipu_plugin.cpp: 344] Concat(resnetv17_stage2__plus3/tile/concat) uses
performance-lib
[I] [actg.cpp : 471] new sgnode with actg: 0
[I] [datalayout_schedule.cpp:2507] Layout loss: 136
[I] [datalayout_schedule.cpp:2508] Layout scheduling ...
[I] [datalayout_schedule.cpp:2511] The layout loss for graph resnet_v1_50: 0
[I] [datalayout_schedule.cpp:1415] The graph resnet_v1_50 post optimized score:0
[I] [datalayout_schedule.cpp:1419] layout schedule costs: 2.27663ms
[I] [IRChecker] Start to check IR:
[I] [IRChecker] model_name: cost_model
[I] [IRChecker] IRChecker: All IR pass
[I] [load_balancer.cpp:3051] enable multicore schedule optimization for load balance
strategy 0 it may degrade performance on single core targets.
[I] [soms_scheduler.cpp: 195] [EVAL] init time t1: 674 ms
[I] [soms_scheduler.cpp: 201] [EVAL] unsafe check time t2: 241 ms
[I] [soms_scheduler.cpp: 922] not found!
[I] [soms_scheduler.cpp: 947] get max in loop 0
```



```

[I] [soms_scheduler.cpp: 214] [EVAL] mem assignment time t3: 107 ms
[I] [builder.cpp:1808] [EVAL] duration time of mem allocation: 1038 ms
[I] [builder.cpp:1927] The graph DDR Footprint requirement(estimation) of feature maps:
[I] [builder.cpp:1928]      Read and Write:30.41MB
[I] [builder.cpp:1109] Reduce constants memory size: 1.957MB
[I] [builder2.cpp:1694] memory statistics for this graph (resnet_v1_50)
[I] [builder.cpp: 614] Text          section: 0x0000c2b0 Bytes ( 0.048MB)
[I] [builder.cpp: 614] Weight       section: 0x010e26e0 Bytes ( 16.884MB)
[I] [builder.cpp: 614] Ro          section: 0x00000b00 Bytes ( 0.003MB)
[I] [builder.cpp: 614] Desc        section: 0x00026500 Bytes ( 0.150MB)
[I] [builder.cpp: 614] Stack       section: 0x000c0c00 Bytes ( 0.753MB)
[I] [builder.cpp: 614] Workspace   section: 0x00000000 Bytes ( 0.000MB)
[I] [builder.cpp: 614] Activations section: 0x00304804 Bytes ( 3.018MB)
[I] [builder.cpp: 614] Total:      0x014dad94 Bytes ( 20.855MB)
[I] [tools.cpp :1480] - compile time: 1.905 s
[I] [tools.cpp :1382] With GM optimization, DDR Footprint stastic(estimation):
[I] [tools.cpp :1389]      Read and Write:39.89MB
[I] [tools.cpp :1436] - draw graph time: 0.001 s
[I] [tools.cpp :1991] remove global cwd:
/tmp/2598acfa425076f3c1a612b420875c94fdb999b17d3b72f2641e827afe3c8
build success.....
Total errors: 0,  warnings: 7

```

9) The model file generated by **cixbuild** compilation is as follows:

```

test@test:~/ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_re
snet_v1_50$ ls *.cix
resnet_v1_50.cix

```

4) Then upload the compiled **resnet_v1_50.cix** model file to the Linux system of the development board. The steps for testing the model on the Linux system of the development board are as follows:

- a. Download CIX AI Model Hub on the Linux system of the development board according to the instructions in the section on [downloading CIX AI Model Hub](#).
- b. Install dependency packages on the Linux system of the development board according to the instructions in the section on [installing NPU dependency](#)



packages.

- c. Then replace the CIX AI Model Hub with the recently uploaded **resnet_v1_50.cix** model file **ai_model_hub_25_Q3/models/ComputeVision/Image_Classification/onnx_resnet_v1_50/resnet_v1_50.cix** Model.
- d. Then, follow the instructions in the section on **onnx_resnet_v1_50 case** to test the newly compiled model file.

3. 39. Methods for Installing Anaconda

1) To download the installation script for Anaconda3, please open the link below in your browser and download it directly.

- a. The download link for version X86_64 is shown below. This version needs to be installed on X86_64 computers.

https://mirror.tuna.tsinghua.edu.cn/anaconda/archive/Anaconda3-2025.06-1-Linux-x86_64.sh

- b. The download link for AARCH64 version is shown below. This version needs to be installed on the Linux system of the development board.

<https://mirror.tuna.tsinghua.edu.cn/anaconda/archive/Anaconda3-2025.06-1-Linux-aarch64.sh>

2) Add execution permissions to the Anaconda3 installation script.

```
test@test:~$ chmod +x Anaconda3*Linux*.sh
```

3) Run the installation script for Anaconda3.

```
test@test:~$ ./Anaconda3*Linux*.sh
```

4) Then press **Enter** to view the license agreement, and enter **yes** to accept the agreement.

```
Welcome to Anaconda3 2025.06-1
```

```
In order to continue the installation process, please review the license agreement.
```

```
Please, press ENTER to continue
```

```
>>>
```

```
By continuing installation, you hereby consent to the Anaconda Terms of Service available at https://anaconda.com/legal.
```



```
Do you accept the license terms? [yes|no]
```

```
>>> yes
```

5) Continuing to press enter will install Anaconda in the **anaconda3** folder of the home directory by default. You can also specify different installation locations here.

```
Anaconda3 will now be installed into this location:
```

```
/home/test/anaconda3
```

- Press ENTER to confirm the location
- Press CTRL-C to abort the installation
- Or specify a different location below

```
[/home/test/anaconda3] >>>
```

6) Enter **yes** to have the installation program initialize Anaconda3.

```
Do you wish to update your shell profile to automatically initialize conda?
```

```
This will activate conda on startup and change the command prompt when activated.
```

```
If you'd prefer that conda's base environment not be activated on startup,  
run the following command when conda is activated:
```

```
conda config --set auto_activate_base false
```

```
You can undo this by running `conda init --reverse $SHELL`? [yes|no]
```

```
[no] >>>yes
```

7) Then reopen a terminal and if you can see the **(base)** prompt, it means Conda has been successfully installed and the **base** environment has been activated.

```
(base) test@test:~$
```

8) The command to view the conda version is as follows:

```
(base) test@test:~$ conda --version
```

```
conda 25.5.1
```



9) The command to exit the current environment and return to the default system environment is as follows:

```
(base) test@test:~$ conda deactivate
```

10) The steps to create different Python environments are as follows:

a. The basic syntax of the command is as follows:

```
conda create -n environment name python=version number
```

b. For example, the command to create a new environment for Python3.10 is as follows:

```
(base) test@test:~$ conda create -n py310 python=3.10
```

```
.....
```

```
Proceed ([y]/n)? y
```

```
Downloading and Extracting Packages:
```

```
Preparing transaction: done
```

```
Verifying transaction: done
```

```
Executing transaction: done
```

```
#
```

```
# To activate this environment, use
```

```
#
```

```
# $ conda activate py310
```

```
#
```

```
# To deactivate an active environment, use
```

```
#
```

```
# $ conda deactivate
```

c. The command to activate the py310 environment is as follows:

```
(base) test@test:~$ conda activate py310
```

d. After activation, you can see the prompt for **(py310)**.

```
(py310) test@test:~$
```

e. Verify Python version.

```
(py310) test@test:~$ python3 --version
```

```
Python 3.10.19
```

f. Run the following command to set the system to default to the py310 environment.



```
(py310) orangepi@orangepi:~$ echo "conda activate py310" >> ~/.bashrc
```

3. 40. How to check the BIOS firmware version

The command to check the BIOS firmware version in a Linux system is as follows:

```
orangepi@orangepi:~$ sudo dmidecode -s bios-version  
v1.0_for_opi6plus_linux
```

Additionally, the BIOS firmware version can also be seen in the logs printed during the BIOS boot process.

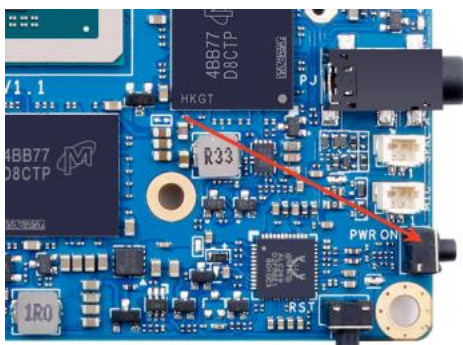
```
Tianocore/EDK2 firmware version v1.0_for_opi6plus_linux-W43M251022191158N-b8be8a  
19039c  
Press ESCAPE for boot options  
PhyDetectDevice: fail to detect ethernet PHY  
I2C[5] hardware no response (ISR=0x00000004) during write!  
I2C[5] Failed to process I2C message, status No Response  
[01.981] [UEFI] E550 BmAfterConsole
```

3. 41. Methods for shutting down and restarting the development board

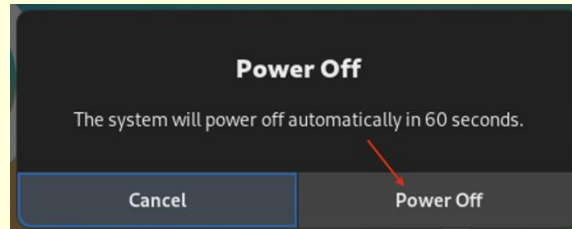
1) During the operation of the Linux system, if the Type-C power is directly unplugged and the power is cut off, it may cause the file system to lose some data or be damaged. Therefore, please use the **poweroff** command to shut down the Linux system of the development board before unplugging the power.

```
orangepi@orangepi:~$ sudo poweroff
```

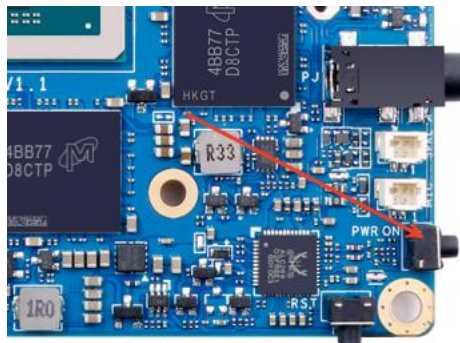
2) In addition, the development board is equipped with a power button, which can be **briefly pressed** to shut down the device. The position of the power button is as follows:



Note that when the power button is pressed on the Linux desktop system, a confirmation box as shown in the figure below will pop up by default. You need to click the **Power Off option before shutting down, otherwise you will need to wait for 60 seconds before shutting down.**



3) After shutting down, press the power button on the development board briefly to turn it on. The location of the power button is as follows:



4) The command to restart the linux system is:

```
orange@orange:~$ sudo reboot
```

5) In addition, there is a reset button on the development board that can be used to reset the board. The reset button is located as follows:



4. Linux SDK——orange-pi-build usage instructions

4. 1. Compilation System Requirements

4. 1. 1. Compile on Ubuntu22.04 computer using x64

1) Linux SDK, **orange-pi-build** supports running on computers with **Ubuntu 22.04** installed. Therefore, before downloading orange-pi-build, please ensure that the Ubuntu version installed on the computer is Ubuntu 22.04. The command to check the installed Ubuntu version on the computer is as follows. If the Release field does not display **22.04**, it means that the current Ubuntu version used does not meet the requirements. Please replace the system before performing the following operations.

```
test@test:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description: Ubuntu 22.04.5 LTS
Release: 22.04
Codename: jammy
```

2) If the computer is installed with a Windows system and does not have Ubuntu 22.04 installed, you can consider using **VirtualBox** or **VMware** to install an Ubuntu 22.04 virtual machine on the Windows system.

3) The installation image download address for Ubuntu 22.04 **amd64** version is:

```
https://mirrors.tuna.tsinghua.edu.cn/ubuntu-releases/22.04/ubuntu-22.04.5-desktop-amd64.iso
或者
https://repo.huaweicloud.com/ubuntu-releases/22.04/ubuntu-22.04.5-desktop-amd64.iso
```

4) After installing Ubuntu 22.04 on your computer or virtual machine, please first set the



software source of Ubuntu 22.04 to Qinghua Source, otherwise errors may occur during software installation due to network issues.

- a. The method of replacing Tsinghua Source can refer to the instructions on this webpage.

<https://mirrors.tuna.tsinghua.edu.cn/help/ubuntu/>

- b. Note that Ubuntu version needs to be switched to 22.04.

Ubuntu 镜像使用帮助

Ubuntu 的软件源配置文件是 `/etc/apt/sources.list`。将系统自带的该文件做个备份，将该文件替换为下面内容，即可使用 TUNA 的软件源镜像。

选择你的ubuntu版本: **22.04 LTS**

```
# 默认注释了源码镜像以提高 apt update 速度，如有需要可自行取消注释
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-updates main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-updates main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-backports main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-backports main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-security main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-security main restricted universe multiverse

# 预发布软件源，不建议启用
# deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-proposed main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-proposed main restricted universe multiverse
```

- c. The contents of the `/etc/apt/sources.list` file that needs to be replaced are:

```
test@test:~$ sudo mv /etc/apt/sources.list /etc/apt/sources.list.bak
test@test:~$ sudo vim /etc/apt/sources.list

#The source code mirror is commented out by default to improve the speed of apt update. You can uncomment it if
needed.

deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-updates main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-updates main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-backports main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-backports main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-security main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-security main restricted universe multiverse

# Pre-release software repositories are not recommended for use.
# deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-proposed main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-proposed main restricted universe multiverse
```

- d. After replacement, it is necessary to update the package information and ensure that there are no errors.



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

4. 2. Obtain the source code of linux sdk

4. 2. 1. Download orangepi-build

1) The linux sdk actually refers to the orangepi-build code, which is modified based on the armbian build compilation system. Using orangepi-build, multiple versions of linux images can be compiled. First, download the code for orangepi-build. The command is as follows:

a. The command downloaded from github is as follows:

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

b. The command downloaded from gitee is as follows:

```
test@test:~$ sudo apt-get update && sudo apt-get install -y git
test@test:~$ git clone https://gitee.com/orangepi-xunlong/orangepi-build.git -b next
```

Note that the Orange Pi 6 Plus development board requires downloading the **next branch source code of orangepi-build. The git clone command above needs to specify the branch of orangepi-build source code as next.**

orangepi-xunlong / orangepi-build Public

<> Code Issues 6 Pull requests 1 Discussions Actions Projects Wiki Security Insights Settings

next 2 branches 0 tags

Go to file Add file >> Code

Switch branches/tags

Find or create a branch...

Branches Tags

main default

✓ next View all branches

Need to switch to next

69dd359 4 days ago 222 commits

Update for Orange Pi 5 v1.0.2	4 days ago
Update for Orange Pi 5 v1.0.2	4 days ago
Update for Orange Pi 5 v1.0.2	4 days ago
First Commit	2 years ago
Support orangepi3 next branch	8 months ago
Bump to next branch	9 months ago



Downloading the code for orangepi-build through the git clone command does not require entering the username and password of a github or gitee account. If you are prompted to enter the username and password of a github or gitee account after entering the git clone command, it is usually because the address of the orangepi-build repository after git clone is entered incorrectly. Please carefully check the spelling of the command for errors.

2) The current linux kernel version used by the development board is as follows:

branch	Linux kernel version
current	Linux6.1
next	Linux6.6

The branch mentioned here and the branch of orangepi-build source code are not the same thing, please don't confuse them. This branch is mainly used to distinguish between different versions of kernel source code.

At present, the linux6.1 bsp kernel provided by this chip is defined as the current branch.

At present, we define the Linux 6.6 BSP kernel provided by this chip as the next branch.

3) After downloading orangepi-build, it will include the following files and folders

- build.sh**: Compile startup script
- external**: Contains configuration files required for compiling images, specific scripts, and source code for some programs, etc
- LICENSE**: GPL 2 License File



- d. **README.md**: orangepi-build documentation
- e. **scripts**: General script for compiling linux images

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

If you download the code for orangepi-build from github you may find that the orangepi-build does not contain the source code for the linux kernel, nor does it require a cross compilation toolchain to compile the linux kernel. This is normal because these things are stored in other separate repositories or on certain servers (the address will be detailed below). orangepi-build specifies the download addresses for the linux kernel and cross compilation toolchain in the script and configuration files. When running orangepi-build, if it finds that these things are not available locally, it will automatically go to the corresponding location to download them.

4. 2. 2. Download the cross compilation toolchain

1) When orangepi-build runs for the first time, it automatically downloads the cross compilation **toolchains** and places it in the toolchains folder. After running the build.sh script of orangepi-build, it checks whether all the cross compilation toolchains in **toolchains** exist. If they do not exist, it will restart the download. If they do exist, it will be used directly without repeated downloads.

```
orangepi@orangepi:~/orangepi-build$ sudo ./build.sh
[ o.k. ] Using config file [ /home/orangepi/orangepi-build/userpatches/config-example.conf ]
[ o.k. ] Extension manager [ processed 1 Extension Methods calls and 1 Extension Method implementations ]
[ o.k. ] Preparing [ host ]
[ o.k. ] Build host OS release [ jammy ]
[ .... ] Installing build dependencies
[ o.k. ] Syncing clock [ cn.pool.ntp.org ]
[ o.k. ] Checking for external GCC compilers
[ .... ] downloading using http(s) network [ arm-gnu-toolchain-12.3.rel1-x86_64-aarch64-none-linux-gnu.tar.xz ]
#998971 124MiB/125MiB(99%) CN:1 DL:2.6MiB
[ o.k. ] Verified [ MD5 ]
[ .... ] decompressing
[ .... ] arm-gnu-toolchain-12.3.rel1-x86_64-aarch64-none-linux-gnu.tar.xz: 125MiB [19.6MiB/s] [=====] 100%
```

2) The cross compilation toolchain used to compile linux kernel source code is:

```
test@test:~/orangepi-build$ ls toolchains/
arm-gnu-toolchain-12.3.rel1-x86_64-aarch64-none-linux-gnu
```

If there is a problem with downloading the cross compilation toolchain, you can download toolchains.tar.gz from the official tool and extract it to the top-level directory of orangepi-build.



The decompression command for toolchains.tar.gz is as follows:

```
test@test:~/orange-pi-build$ tar xzf toolchains.tar.gz
```

```
test@test:~/orange-pi-build$ ls toolchains
```

[arm-gnu-toolchain-12.3.rel1-x86_64-aarch64-none-linux-gnu](#)

4. 2. 3. Explanation of the complete directory structure of orange-pi-build

1) After downloading the orange-pi-build repository, it does not include the linux kernel or cross compilation toolchain. The source code of the linux kernel is stored in a separate git repository:

a. The git repository where the Linux6.1 kernel source code is stored is as follows:

a) Github:

<https://github.com/orangepi-xunlong/linux-orangepi/tree/orange-pi-6.1-cix>

b) Gitee:

<https://gitee.com/orangepi-xunlong/orange-pi-6.1-cix>

b. The git repository where the Linux6.6 kernel source code is stored is as follows:

a) Github:

<https://github.com/orangepi-xunlong/linux-orangepi/tree/orange-pi-6.6-cix>

b) Gitee:

<https://gitee.com/orangepi-xunlong/orange-pi-6.6-cix>

2) When orange-pi-build is first run, it will download the cross compilation toolchain and linux kernel source code. After successfully compiling the linux image once, the files and folders that can be seen in orange-pi-build are:

a. **build.sh**: Compile startup script

b. **external**: Contains configuration files required for compiling the image, scripts



for specific functions, and source code for some programs. The rootfs compressed file cached during the image compilation process is also stored in the external file

- c. **kernel**: This folder contains the Linux kernel source code. The folder named **orange-pi-6.1-cix** contains the kernel source code for the `current` branch of this chip series development board, and the folder named **orange-pi-6.6-cix** contains the kernel source code for the `next` branch. Please do not manually change the names of these kernel source code folders. If you do, the compilation system will download the kernel source code again during runtime.
- d. **LICENSE**: GPL 2 License File
- e. **README.md**: orangepi-build documentation
- f. **output**: Files containing compiled Linux (e.g., .deb packages), compilation logs, and compiled images.
- g. **scripts**: General script for compiling linux images
- h. **toolchains**: Store cross compilation toolchain
- i. **userpatches**: Store the configuration files required for compiling scripts

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

4. 3. Compiling Linux Kernel

- 1) Run the build.sh script and remember to grant sudo privileges.

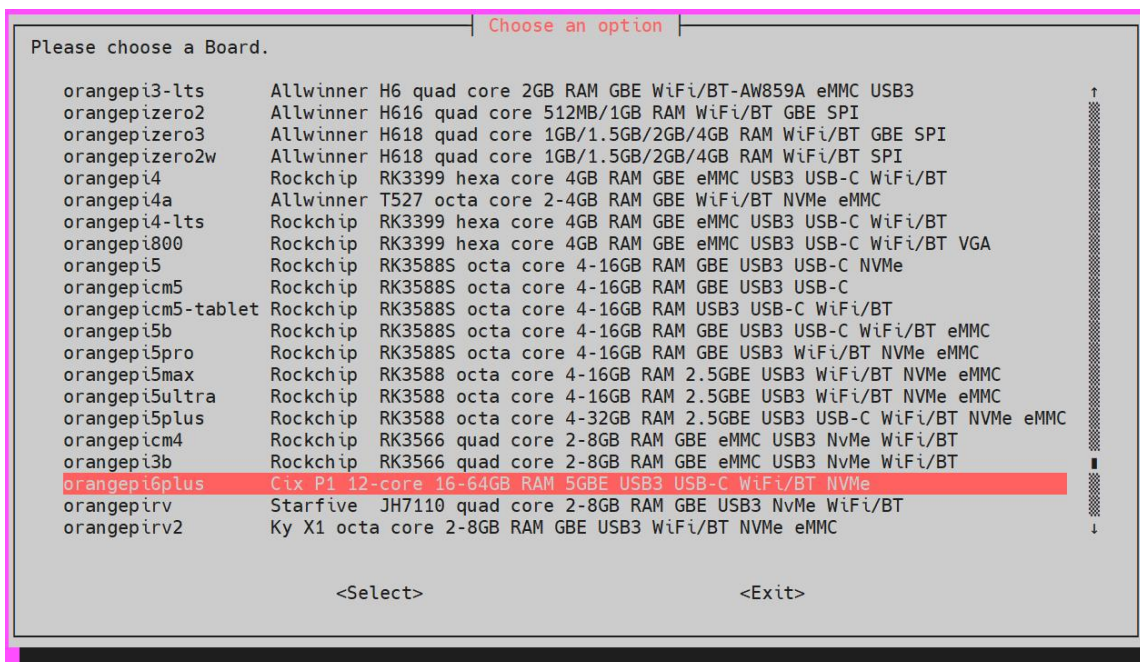
```
test@test:~/orangepi-build$ sudo ./build.sh
```

If downloading code from github is slow, you can do it at /Add the parameter GITEE_SERVER=yes after build.sh, so that during compilation, all repositories except component_cix will be downloaded from the gitee website.

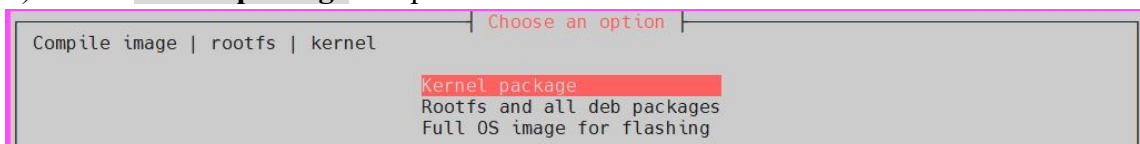
The compilation command for adding the GITEE_SERVER=yes parameter is as follows:

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

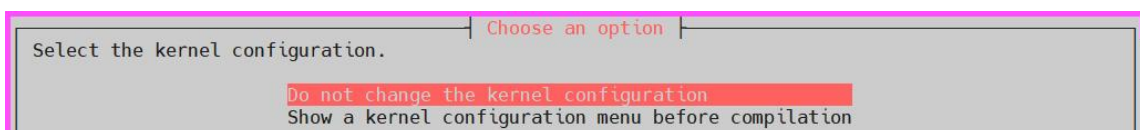
- 2) Select the model of the development board and press Enter.



3) Select **Kernel package** and press enter.

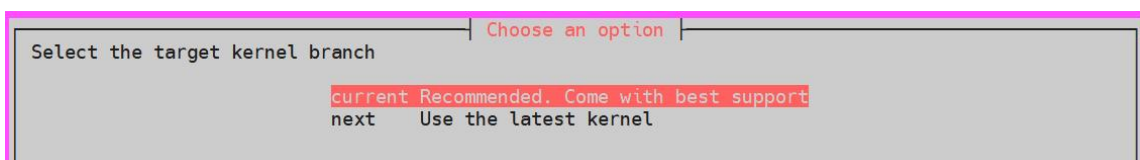


4) Then it will prompt whether the kernel configuration interface needs to be displayed. If the kernel configuration does not need to be modified, select the first one. If the kernel configuration needs to be modified, select the second one.



5) Then select the kernel branch.:

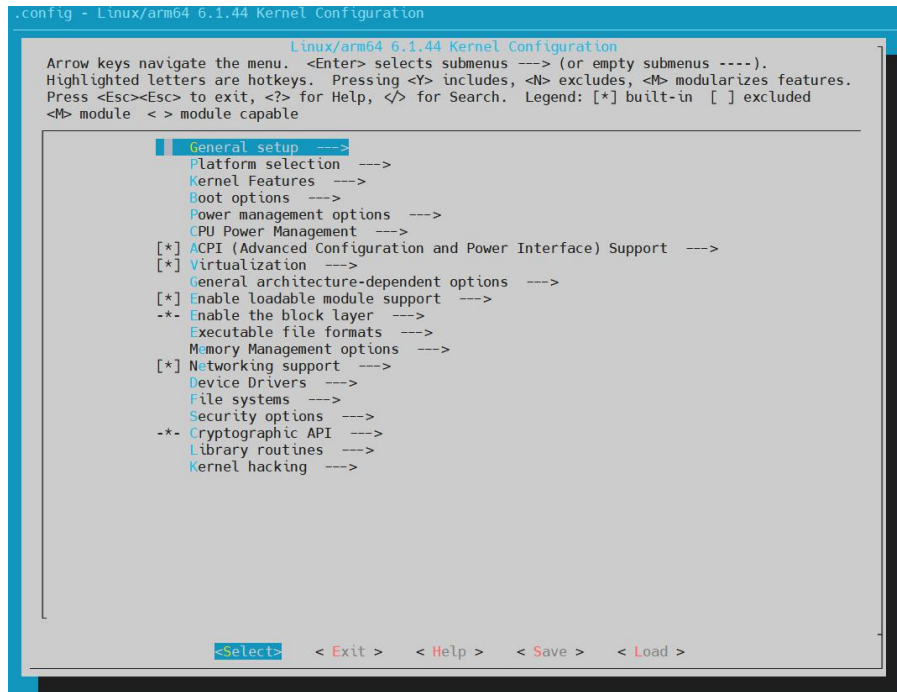
- current** branch will compile the Linux 6.1 kernel.
- next** branch will compile the Linux 6.6 kernel.



6) If step 4) selects the option to display the kernel configuration menu (second option),



a kernel configuration interface opened through **make menuconfig** will pop up. At this time, you can directly modify the kernel configuration, save and exit after modification, and the kernel source code will be compiled after exit.



- a. If there is no need to modify the configuration options of the kernel, passing **KERNEL_CONFIGURE=no** when running the build.sh script can temporarily block the pop-up kernel configuration interface.

```
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 appears when compiling the kernel, it is due to the small terminal interface of Ubuntu PC, which causes the **make menuconfig** interface to not display. Please set the Ubuntu PC terminal to its maximum size and then run the build.sh script again.



```

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

```

Please note that the following demonstration uses the Linux 6.1 kernel, specifically the current branch. If you select the next branch, the displayed kernel version and branch name will be slightly different. Also, the version number 1.0.2 may be updated. If you see a newer version number, that's normal.

7) The following is a partial explanation of the information prompted when compiling kernel source code:

a. Version of linux kernel source code

```
[ o.k. ] Compiling current kernel [ 6.1.44 ]
```

b. The version of the cross compilation toolchain used

```
[ o.k. ] Compiler version [ aarch64-none-linux-gnu-gcc 12.3.1 ]
```

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

```
[ o.k. ] Using kernel config file [ config/kernel/linux-cix-sky1-current.config ]
```

d. The path of the compiled kernel related deb package

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

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

```
[ o.k. ] File name [ linux-image-current-cix_1.0.2_arm64.deb ]
```

f. Compilation time used

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

g. Finally, the compilation command for the kernel selected last time will be displayed. The following command can be used to start compiling the kernel source code without selecting it through the graphical interface

```
[ o.k. ] Repeat Build Options [ sudo ./build.sh BOARD=orangepi6plus
BRANCH=current BUILD_OPT=kernel KERNEL_CONFIGURE=yes ]
```




8) Check the kernel-related deb packages generated during compilation.

a. Linux6.1:

- a) **linux-headers-current-cix_1.0.2_arm64.deb** Include kernel header files
- b) **linux-image-current-cix_1.0.2_arm64.deb** Currently primarily used for updating kernel modules

```
test@test:~/orangepi-build$ ls output/debs/linux-*
output/debs/linux-image-current-cix_1.0.2_arm64.deb
output/debs/linux-headers-current-cix_1.0.2_arm64.deb
```

b. Linux6.6:

- a) **linux-headers-next-cix_1.0.2_arm64.deb** Include kernel header files
- b) **linux-image-next-cix_1.0.2_arm64.deb** Currently primarily used for updating kernel modules

```
test@test:~/orangepi-build$ ls output/debs/linux-*
output/debs/linux-image-next-cix_1.0.2_arm64.deb
output/debs/linux-headers-next-cix_1.0.2_arm64.deb
```

- c. The paths to the compiled GPU, NPU, VPU, and ISP kernel driver .deb packages are shown below.:

a) Linux6.1:

```
test@test:~/orangepi-build$ ls output/debs/cix_debs-current
cix-gpu-driver_1.0.0+2503.orangepi_arm64.deb
cix-npu-driver_1.0.0+2503.orangepi_arm64.deb
cix-isp-driver-v4l2_1.0.0+2503.orangepi_arm64.deb
cix-vpu-driver_1.0.0+2503.orangepi_arm64.deb
```

b) Linux6.6:

```
test@test:~/orangepi-build$ ls output/debs/cix_debs-next
cix-gpu-driver_1.0.0+2503.orangepi_arm64.deb
cix-npu-driver_1.0.0+2503.orangepi_arm64.deb
cix-isp-driver-v4l2_1.0.0+2503.orangepi_arm64.deb
cix-vpu-driver_1.0.0+2503.orangepi_arm64.deb
```

- d. The path to the compiled kernel image file is as follows:

```
test@test:~/orangepi-build$ ls output/cix/Image
output/cix/Image
```

- e. The path to the compiled dtb file is shown below. **Note that the Linux 6.6 kernel currently uses ACPI and does not use a device tree to configure the kernel.**



```
test@test:~/orange-pi-build$ ls output/cix/sky1-orangepi*.dtb
output/cix/sky1-orangepi-6-plus.dtb
output/cix/sky1-orangepi-6-plus-40pin.dtb
output/cix/sky1-orangepi-6-plus-40pin-pwm.dtb
```

9) When the orange-pi-build compilation system compiles the Linux kernel source code, it first synchronizes the Linux kernel source code with the GitHub server's Linux kernel source code. Therefore, if you want to modify the Linux kernel source code, you first need to turn off the source code update function (**you need to compile the Linux kernel source code completely before turning off this function, otherwise it will prompt that the Linux kernel source code cannot be found. If it is a source code compressed package downloaded from Baidu Cloud Drive, there is no problem because the Linux source code is already cached**). Otherwise, the modifications made will be restored. The method is as follows:

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

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

10) If modifications have been made to the kernel, the following method can be used to update the kernel and kernel modules of the Linux system on the development board.

- a. Upload the compiled Linux kernel module's .deb package, image file, and .dtb file to the development board's Linux system.

- a) Linux6.1:

```
test@test:~/orange-pi-build$ scp output/cix/Image root@192.168.1.xxx:/root/
test@test:~/orange-pi-build$ scp output/cix/sky1-orangepi*.dtb \
root@192.168.1.xxx:/root/
test@test:~/orange-pi-build$ scp \
output/debs/linux-image-current-cix_1.0.2_arm64.deb root@192.168.1.xxx:/root/
```

- b) Linux6.6 ACPI:

```
test@test:~/orange-pi-build$ scp output/cix/Image root@192.168.1.xxx:/root/
test@test:~/orange-pi-build$ scp \
output/debs/linux-image-next-cix_1.0.2_arm64.deb root@192.168.1.xxx:/root/
```

- b. Then log in to the development board and uninstall the installed Linux kernel deb package.

- a) Linux6.1:



```
root@orangepi:~# apt purge -y linux-image-current-cix
```

b) Linux6.6:

```
root@orangepi:~# apt purge -y linux-image-next-cix
```

c. Then install the new Linux kernel deb package that I just uploaded.

a) Linux6.1:

```
root@orangepi:~# dpkg -i linux-image-current-cix_1.0.2_arm64.deb
```

b) Linux6.6:

```
root@orangepi:~# dpkg -i linux-image-next-cix_1.0.2_arm64.deb
```

d. Then use the following command to update the kernel image.

If it is a TF card or USB drive startup, the nvme0n1p1 in the following command needs to be replaced with the node name of the corresponding storage device, such as sda1.

```
root@orangepi:~# mount /dev/nvme0n1p1 /mnt
```

```
root@orangepi:~# mv Image /mnt/IMAGE
```

```
root@orangepi:~# umount /mnt
```

e. Then use the following command to update the dtb, which is only required for Linux 6.1 kernel systems that use the device tree.

If it is a TF card or USB drive startup, the nvme0n1p1 in the following command needs to be replaced with the node name of the corresponding storage device, such as sda1.

```
root@orangepi:~# mount /dev/nvme0n1p1 /mnt
```

```
root@orangepi:~# mv sky1-orangepi-6-plus.dtb \
/mnt/SKY1-ORANGEPI-6-PLUS.DTB
```

```
root@orangepi:~# mv sky1-orangepi-6-plus-40pin.dtb \
/mnt/SKY1-ORANGEPI-6-PLUS-40PIN.DTB
```

```
root@orangepi:~# mv sky1-orangepi-6-plus-40pin-pwm.dtb \
/mnt/SKY1-ORANGEPI-6-PLUS-40PIN-PWM.DTB
```

```
root@orangepi:~# umount /mnt
```

f. Then restart the development board and check if the kernel related modifications have taken effect

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

4. 4. Compile rootfs

1) Run the build.sh script and remember to grant sudo privileges.



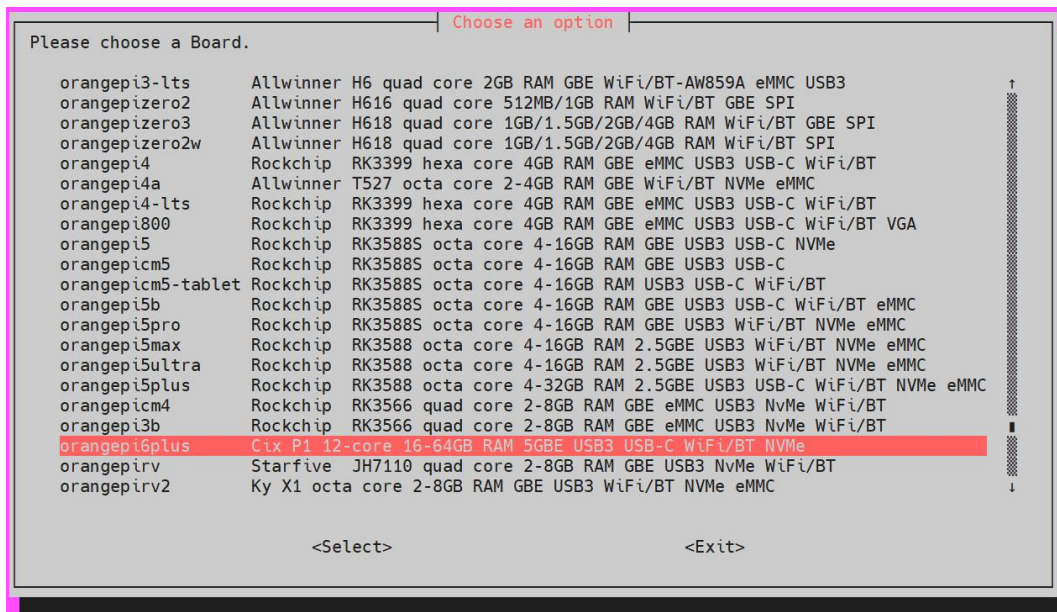
```
test@test:~/orange-pi-build$ sudo ./build.sh
```

If downloading code from github is slow, you can do it at /Add the parameter **GITEE_SERVER=yes** after **./build.sh**, so that during compilation, all repositories except **component_cix** will be downloaded from the gitee website.

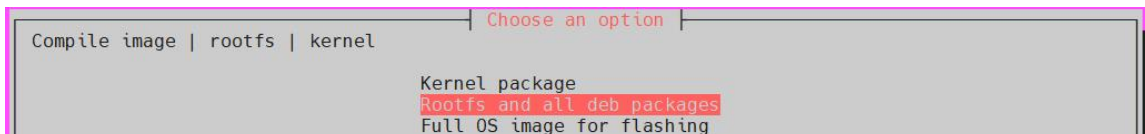
The compilation command for adding the **GITEE_SERVER=yes** parameter is as follows:

```
test@test:~/orange-pi-build$ sudo ./build.sh GITEE_SERVER=yes
```

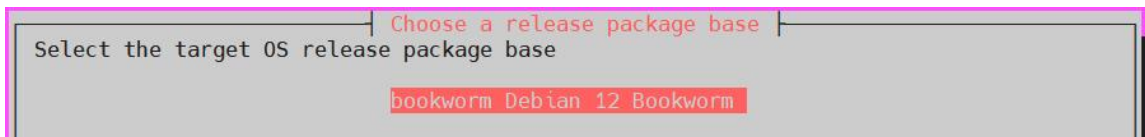
2) Next, select the model of the development board.



a. Select **Rootfs and all deb packages**, then press enter.



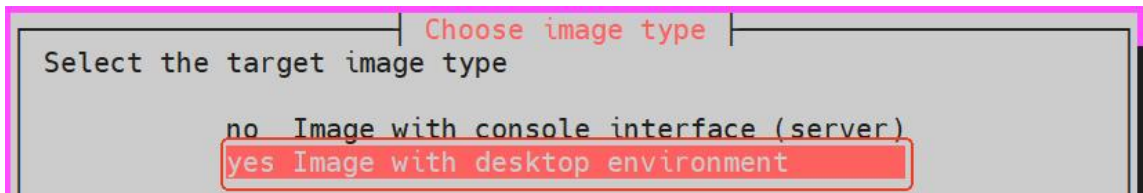
3) Then select the type of rootfs (currently mainly maintained as **bookworm**).



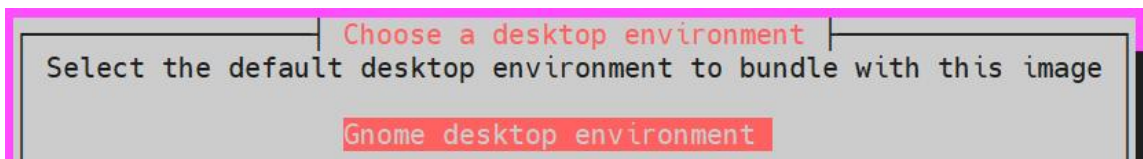
4) Then select the kernel branch.:



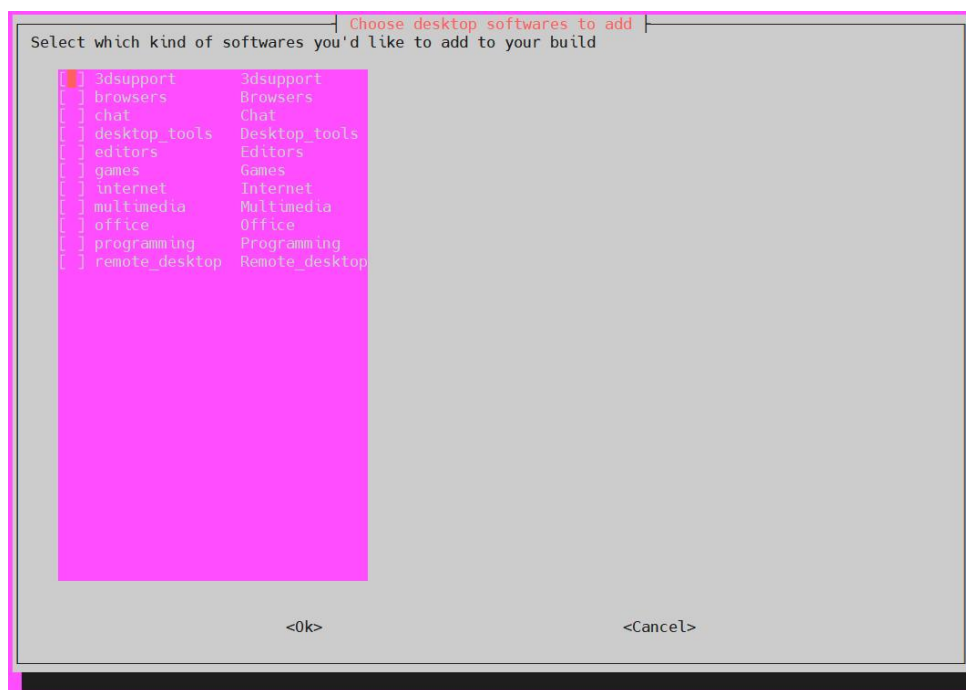
- a. **current** branch will compile the Linux 6.1 kernel.
- b. **next** branch will compile the Linux 6.6 kernel.



5) Compiling the desktop version of the image also requires selecting the type of desktop environment. Currently, Debian Bookworm mainly maintains the Gnome desktop.



Then you can select the additional software packages that need to be installed, please press Enter to skip directly.



6) Then rootfs will start compiling, and some of the information prompted during compilation is as follows:

- a. Types of rootfs



```
[ o.k. ] local not found [ Creating new rootfs cache for bookworm ]
```

- b. The storage path of the rootfs compressed file generated by compilation

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

- c. The name of the rootfs compressed file generated by compilation

```
[ o.k. ] File name
```

```
[ bookworm-gnome-arm64.f930ff6ebbac1a72108a2e100762b18f.tar.lz4 ]
```

- d. Compilation time used

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

7) View the compiled rootfs compressed file

- a. **bookworm-gnome-arm64.f930ff6ebbac1a72108a2e100762b18f.tar.lz4** is a compressed file of rootfs, and the meaning of each field in the name is
- bookworm** Indicate the type of Linux distribution representing rootfs
 - gnome** Indicates that rootfs is the desktop version type, and if it is **cli**, it indicates the server version type
 - arm64** represents the architecture type of rootfs
 - f930ff6ebbac1a72108a2e100762b18f** is an MD5 hash value generated from the package names of all software packages installed by rootfs. As long as the list of software packages installed by rootfs is not modified, this value will not change. The compilation script will use this MD5 hash value to determine whether to recompile rootfs
- b. **bookworm-gnome-arm64.f930ff6ebbac1a72108a2e100762b18f.tar.lz4.list** lists the package names of all rootfs installed software packages

```
test@test:~/orange-pi-build$ ls external/cache/rootfs/
```

```
bookworm-gnome-arm64.f930ff6ebbac1a72108a2e100762b18f.tar.lz4
```

```
bookworm-gnome-arm64.f930ff6ebbac1a72108a2e100762b18f.tar.lz4.current
```

```
bookworm-gnome-arm64.f930ff6ebbac1a72108a2e100762b18f.tar.lz4.list
```

8) If the required rootfs already exist in **external/cache/rootfs**, compiling rootfs again will skip the compilation process and will not restart. When compiling the image, it will also search for available rootfs in **external/cache/rootfs**, and if so, use them directly, which can save a lot of download and compilation time.



4. 5. Compiling Linux Images

1) Run the build.sh script, remember to grant sudo privileges

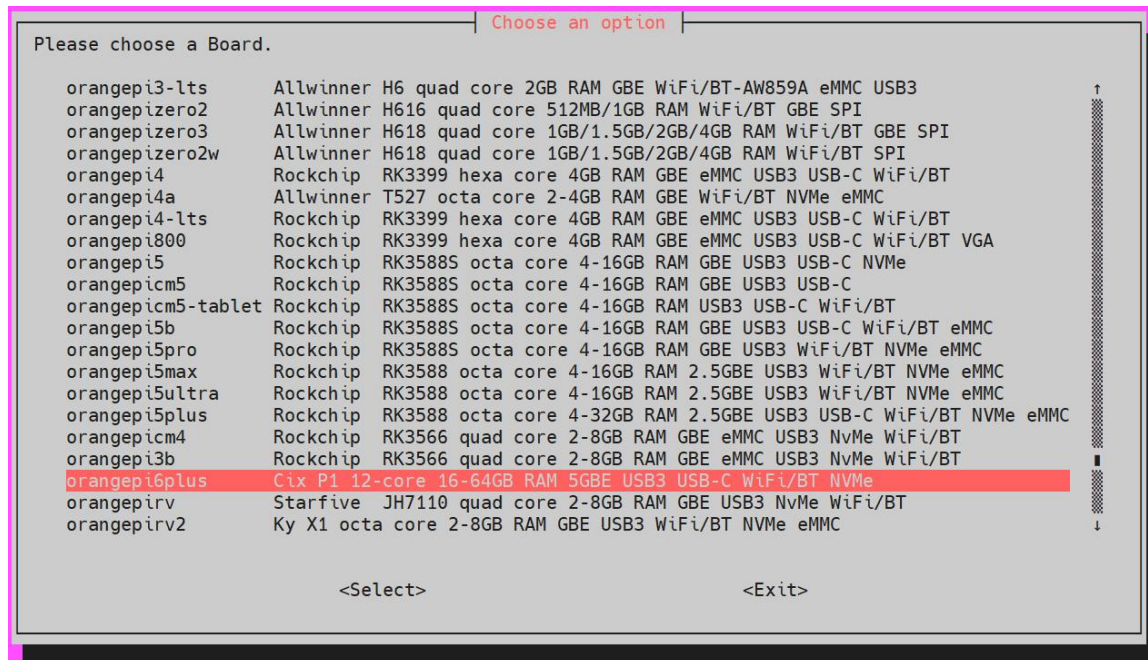
```
test@test:~/orange-pi-build$ sudo ./build.sh
```

If downloading code from github is slow, you can do it at /Add the parameter **GITEE_SERVER=yes** after build.sh, so that during compilation, all repositories except component_cix will be downloaded from the gitee website.从 github

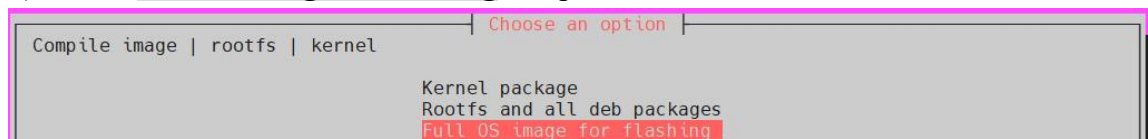
The compilation command for adding the **GITEE_SERVER=yes** parameter is as follows:

```
test@test:~/orange-pi-build$ sudo ./build.sh GITEE_SERVER=yes
```

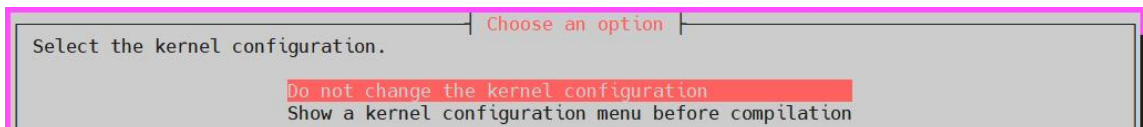
2) Then select the model of the development board



3) Select **Full OS image for flashing** and press Enter

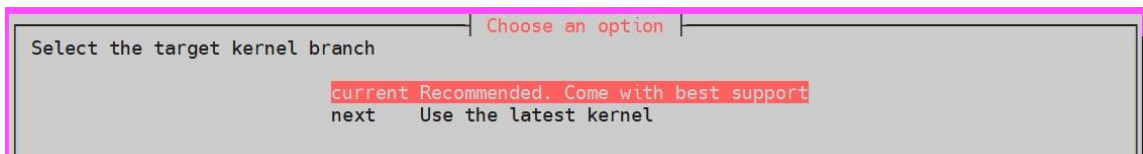


4) Then you will be prompted whether you want to display the kernel configuration interface. If you do not need to modify the kernel configuration, select the first option. If you need to modify the kernel configuration, select the second option.



5) Then select the kernel branch:

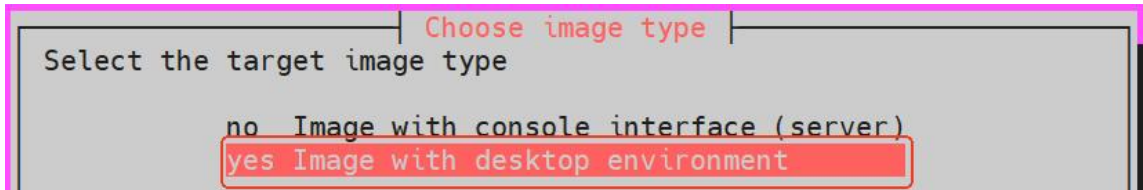
- a. **current** branch will compile the Linux 6.1 kernel.
- b. The **next** branch will compile the Linux 6.6 kernel.



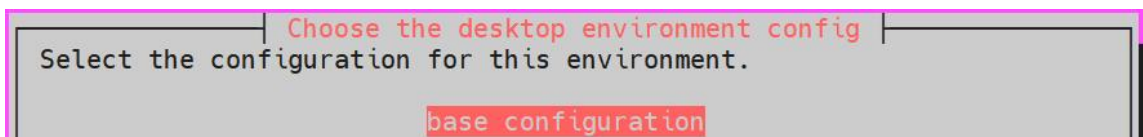
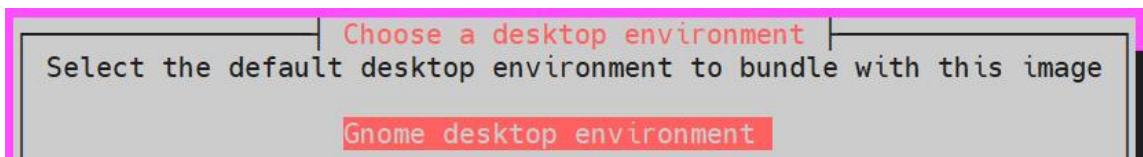
6) Then select the type of rootfs (currently mainly maintaining **bookworm**).



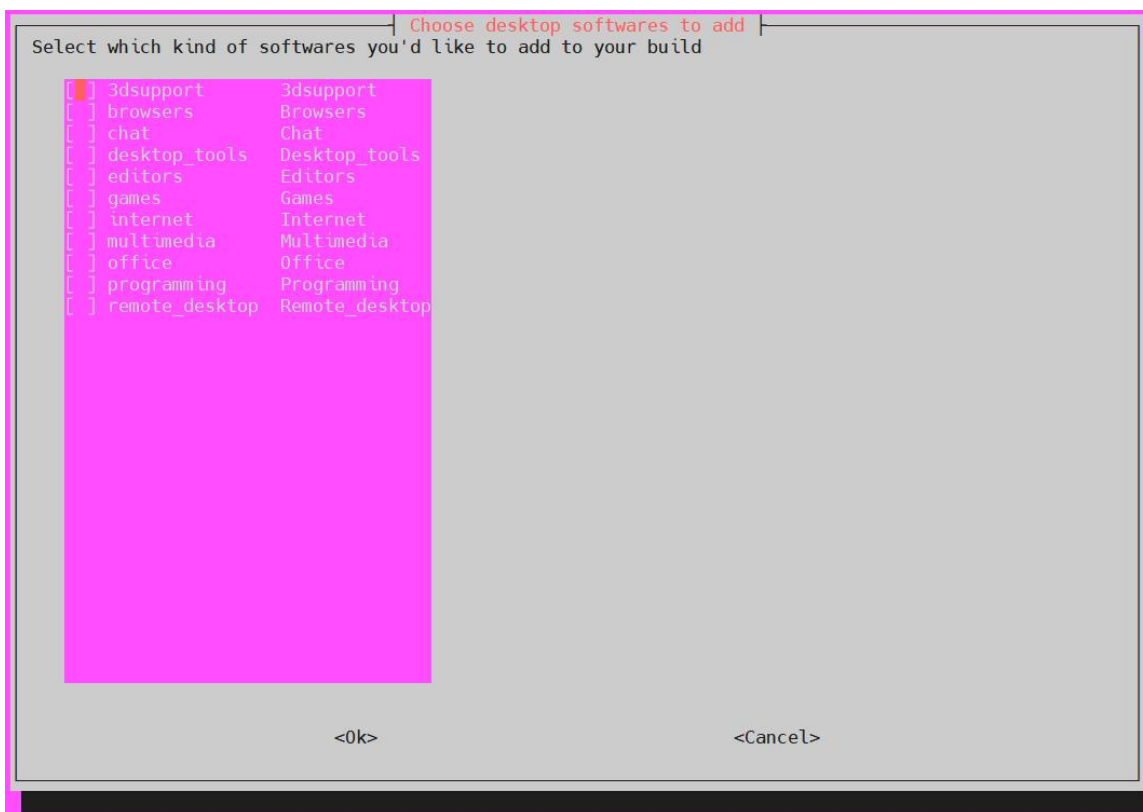
7) Then select the desktop version image.



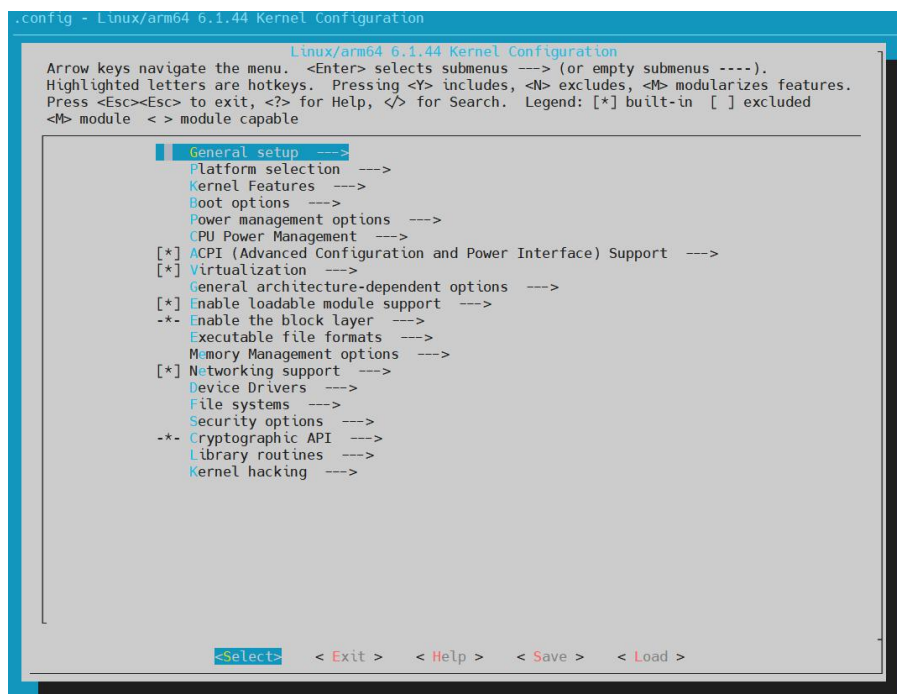
8) Compiling the desktop version of the image also requires selecting the type of desktop environment. Currently, Debian Bookworm mainly maintains the Gnome desktop.



Then you can select any additional packages you want to install. Please press Enter to skip this step.



9) If you selected to display the kernel configuration menu (the second option) in step 4), the kernel configuration interface opened via **make menuconfig** will pop up. You can directly modify the kernel configuration at this time. After modification, save and exit. After exiting, the kernel source code will be compiled.



- d. If you do not need to modify the kernel configuration options, you can temporarily disable the kernel configuration interface by passing **KERNEL_CONFIGURE=no** when running the build.sh script.

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

- e. Alternatively, you can set **KERNEL_CONFIGURE=no** in the **orange-pi-build/userpatches/config-default.conf** configuration file to permanently disable this feature.
- f. If you encounter the following error while compiling the kernel, it's because the terminal interface on your Ubuntu PC is too small, preventing the **make menuconfig** interface from displaying. Please maximize the size of your Ubuntu PC's terminal and then rerun the build.sh script.

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



10) Then it will start compiling the linux image, and the general process of compilation is as follows:

- a. Initialize the compilation environment of Ubuntu PC and install the necessary software packages for the compilation process
- b. Download the source code of the linux kernel (if cached, only update the code)
- c. Compile Linux source code and generate deb packages related to Linux
- d. Creating a deb package for Linux firmware
- e. Create a deb package for the orangepi config tool
- f. Create deb packages that support board level support
- g. If compiling the desktop version image, desktop related deb packages will also be created
- h. Check if rootfs have been cached. If not, create a new rootfs. If cached, decompress and use it directly
- i. Install the deb package generated earlier into rootfs
- j. Make specific settings for different development boards and types of images, such as pre installing additional software packages, modifying system configurations, etc
- k. Then create boot image and rootfs image files
- l. Finally, write the boot image and rootfs image to the image using the dd command

11) After compiling the image, the following message will be prompted

- a. The storage path of the compiled image

```
[ o.k. ] Done building
[ output/images/orangepi6plus_1.0.2_debian_bookworm_desktop_gnome_linux6.1.4
4/orangepi6plus_1.0.2_debian_bookworm_desktop_gnome_linux6.1.44.img ]
```

- b. Compilation time used

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

- a. The command to repeatedly compile the image can be used to start compiling the image without selecting through the graphical interface

```
[ o.k. ] Repeat Build Options [ sudo ./build.sh BOARD=orangepi6plus
BRANCH=current BUILD_OPT=image RELEASE=bookworm
BUILD_MINIMAL=no BUILD_DESKTOP=no KERNEL_CONFIGURE=yes
COMPRESS_OUTPUTIMAGE=sha,xz ]
```



5. Appendix

5.1. User Manual Update History

Version	Date	Update Notes
v0.1	2025-10-15	Initial version
v0.2	2025-10-21	Add instructions for using CIX AI Model Hub
v0.3	2025-10-24	1. Method for burning BIOS image to storage device based on Windows PC 2. Method for burning a BIOS image to a storage device based on a Linux system 3. Method for burning BIOS firmware into the development board's SPI Flash 4. How to use an EDP screen 5. Added user manual for Linux 6.6 Debian 12 system. 6. How to check the BIOS firmware version 7. Testing methods for FFmpeg video hardware encoding 8. Test methods for MPV video hardware decoding
v0.4	2025-10-27	1. Instructions for using the NOE SDK 2. How to add and install Anaconda

5.2. Image update history

Date	Update Notes
2025-10-15	Orangepi6plus_1.0.0_debian_bookworm_desktop_gnome_linux6.1.44.img.xz



	* Initial version
2025-10-21	Orangepi6plus_1.0.2_debian_bookworm_desktop_gnome_linux6.1.44.img.xz * Update NPU kernel drivers and user space dependency packages
2025-10-24	Orangepi6plus_1.0.2_debian_bookworm_desktop_gnome_linux6.6.89.img.xz *Initial version