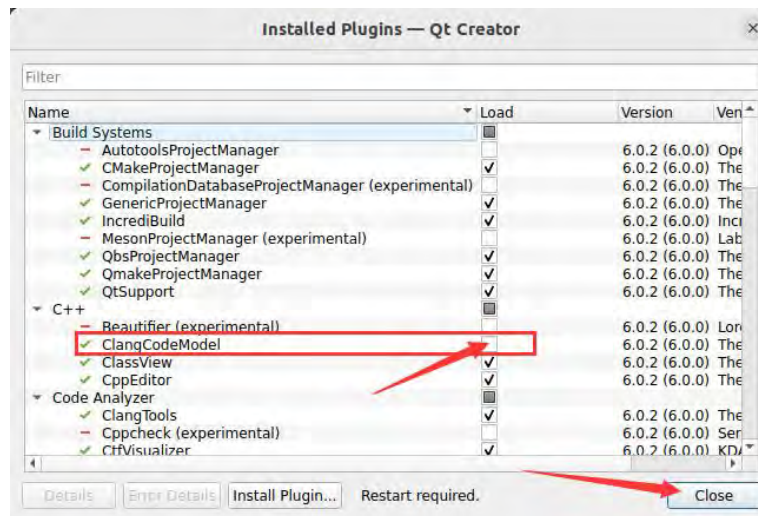


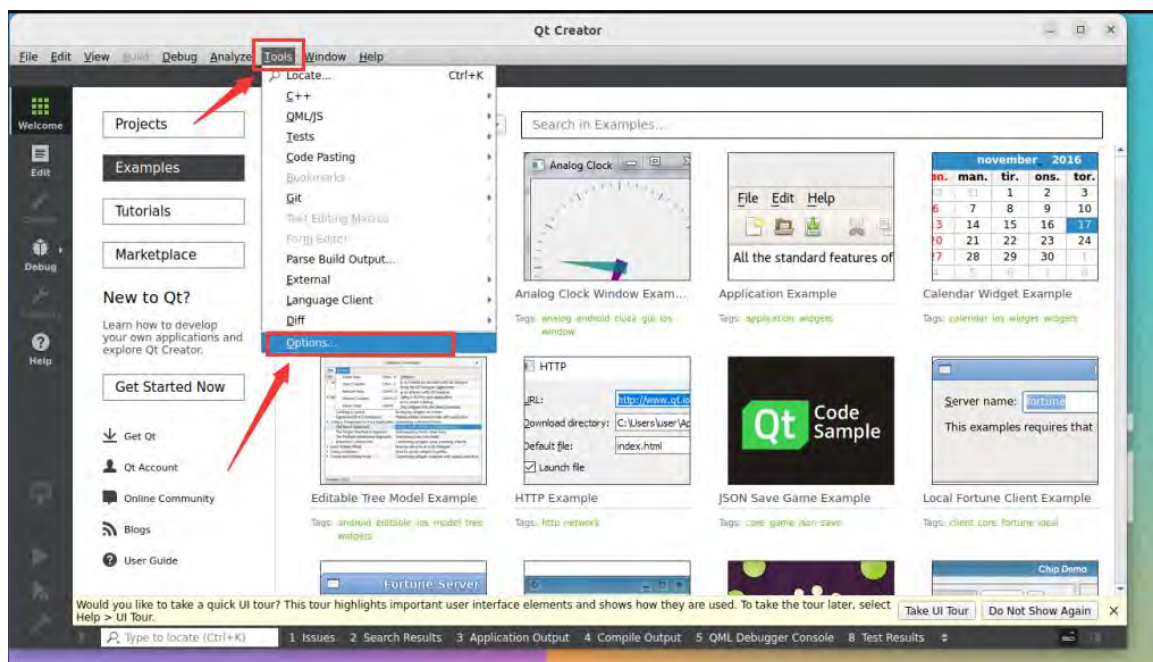


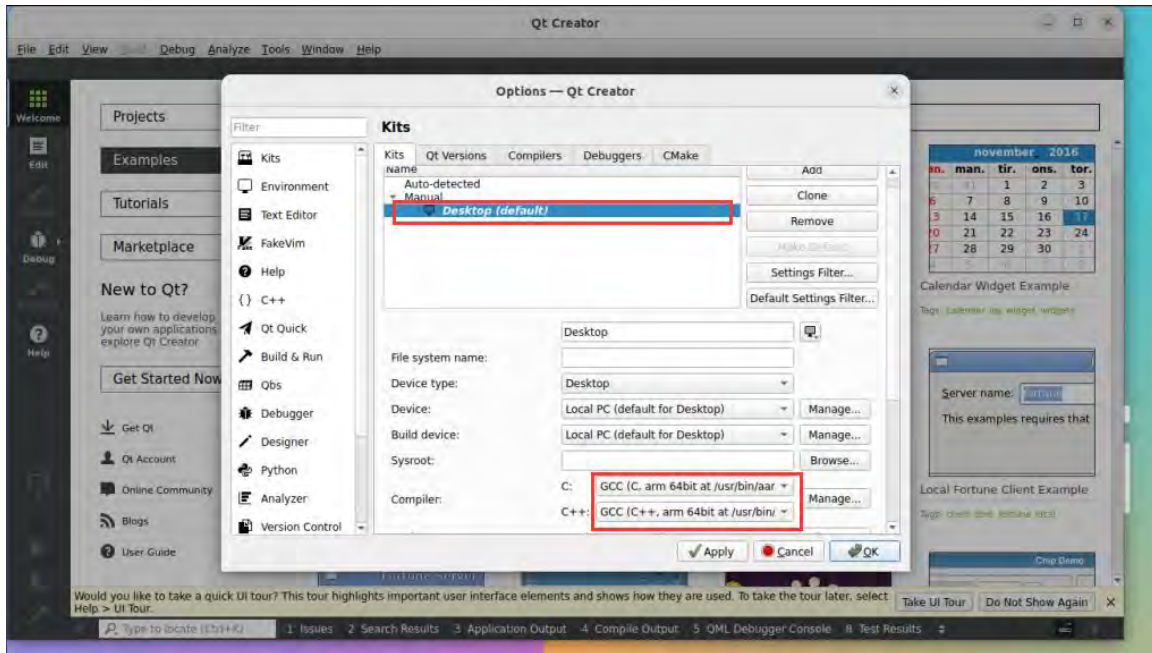
- b. Then remove the check mark of **ClangCodeModel**



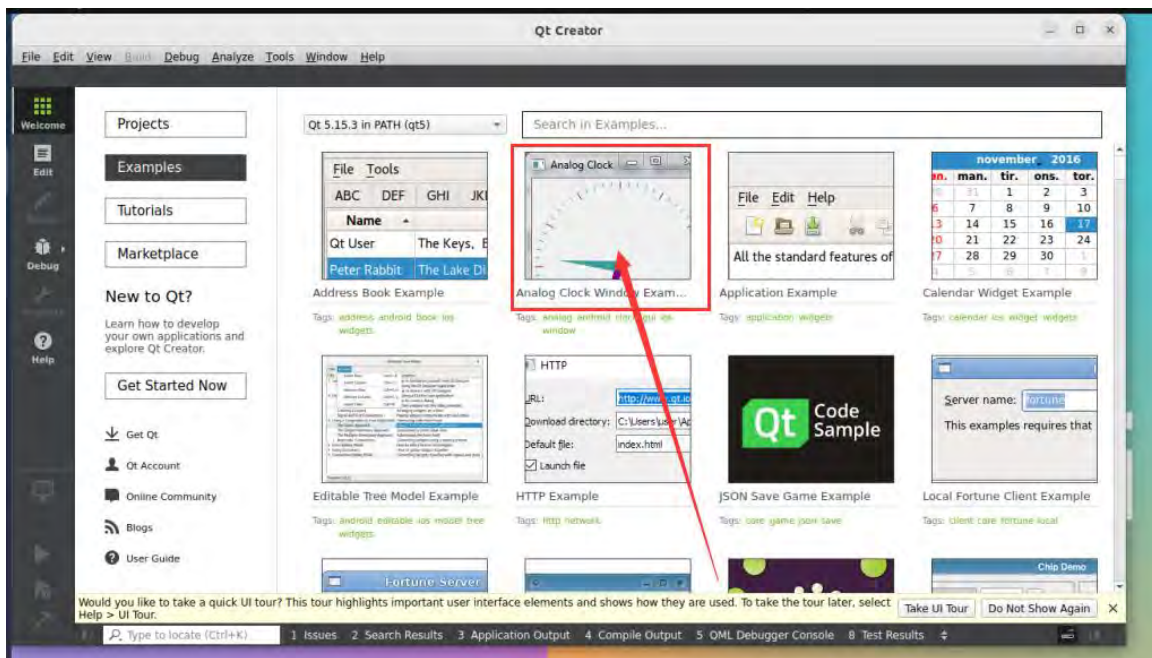
- c. **After setting, you need to restart QT Creator**
- d. Then make sure that QT Creator uses the GCC compiler. If it defaults to Clang, change it to GCC

For Debian 12, please skip this step.

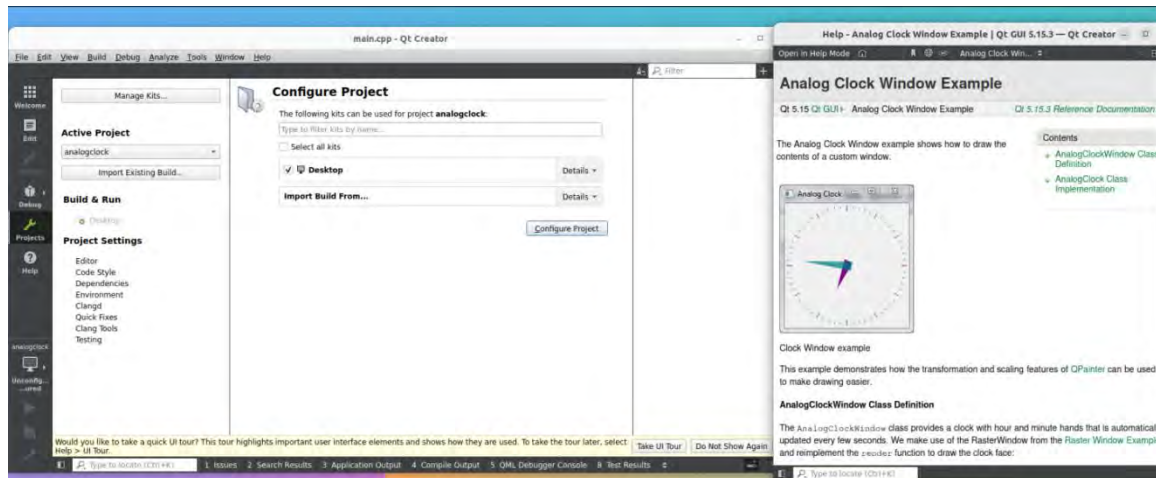




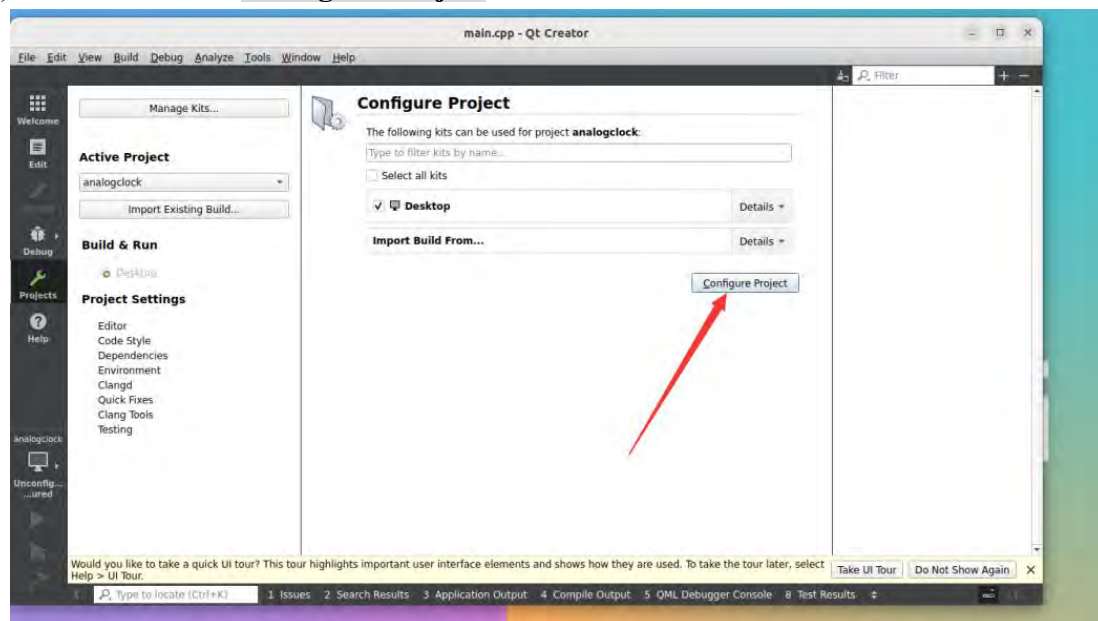
7) Then you can open a sample code



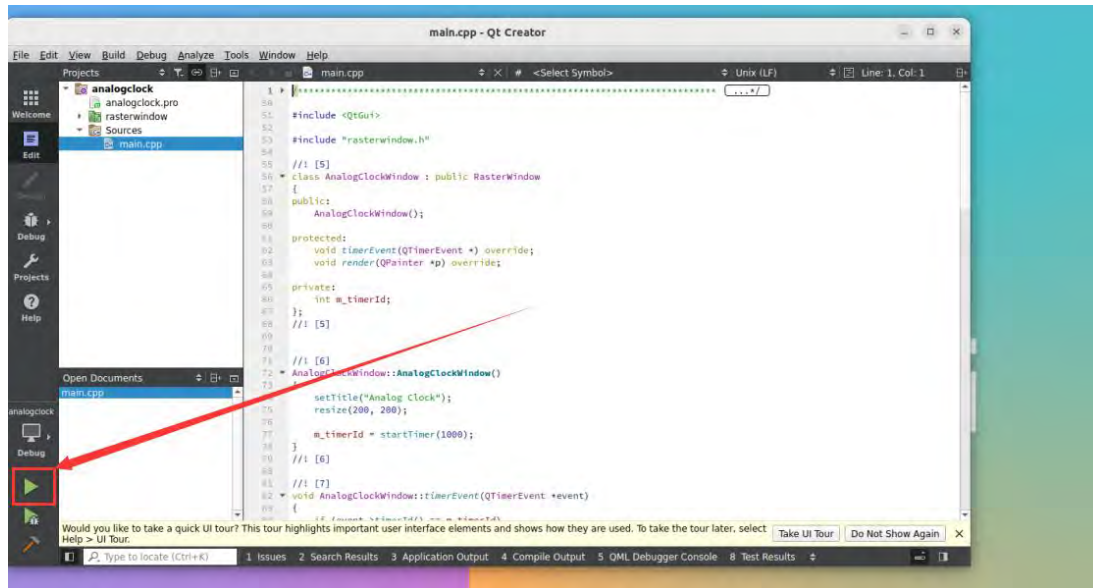
8) Clicking on the sample code will automatically open the corresponding documentation. Please read the instructions carefully.



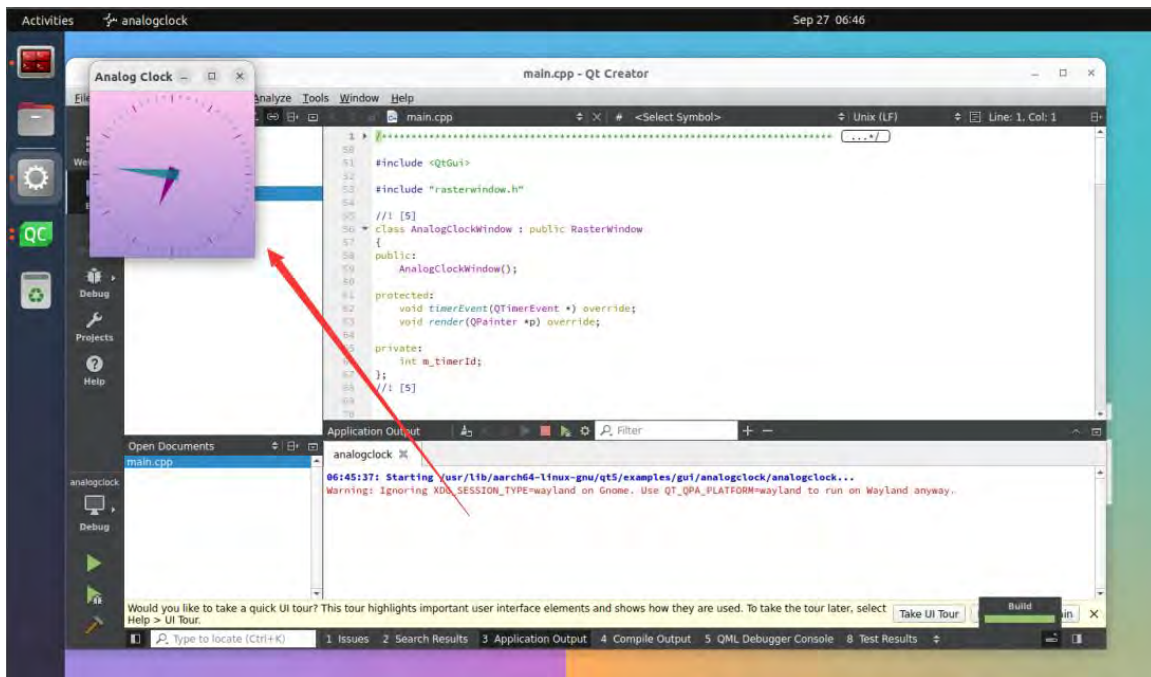
9) Then click Next **Configure Project**



10) Then click the green triangle in the lower left corner to compile and run the sample code



11) After waiting for a while, the interface shown in the figure below will pop up, which means that QT can compile and run normally



12) References

https://wiki.qt.io/Install_Qt_5_on_Ubuntu

<https://download.qt.io/archive/qtcreator>



<https://download.qt.io/archive/qt>

3. 26. ROS installation method

3. 26. 1. How to install ROS 2 Humble on Ubuntu 22.04

1) Use the `install_ros.sh` script to install ros2

```
orangeypi@orangeypi:~$ install_ros.sh ros2
```

2) After the `install_ros.sh` script installs ros2, it will automatically run the `ros2 -h` command. If you can see the following print, it means that ros2 is installed successfully.

```
usage: ros2 [-h] Call `ros2 <command> -h` for more detailed usage. ...
```

ros2 is an extensible command-line tool for ROS 2.

optional arguments:

`-h, --help` show this help message and exit

Commands:

action	Various action related sub-commands
bag	Various rosbag related sub-commands
component	Various component related sub-commands
daemon	Various daemon related sub-commands
doctor	Check ROS setup and other potential issues
interface	Show information about ROS interfaces
launch	Run a launch file
lifecycle	Various lifecycle related sub-commands
multicast	Various multicast related sub-commands
node	Various node related sub-commands
param	Various param related sub-commands
pkg	Various package related sub-commands
run	Run a package specific executable
security	Various security related sub-commands
service	Various service related sub-commands
topic	Various topic related sub-commands
wtf	Use `wtf` as alias to `doctor`



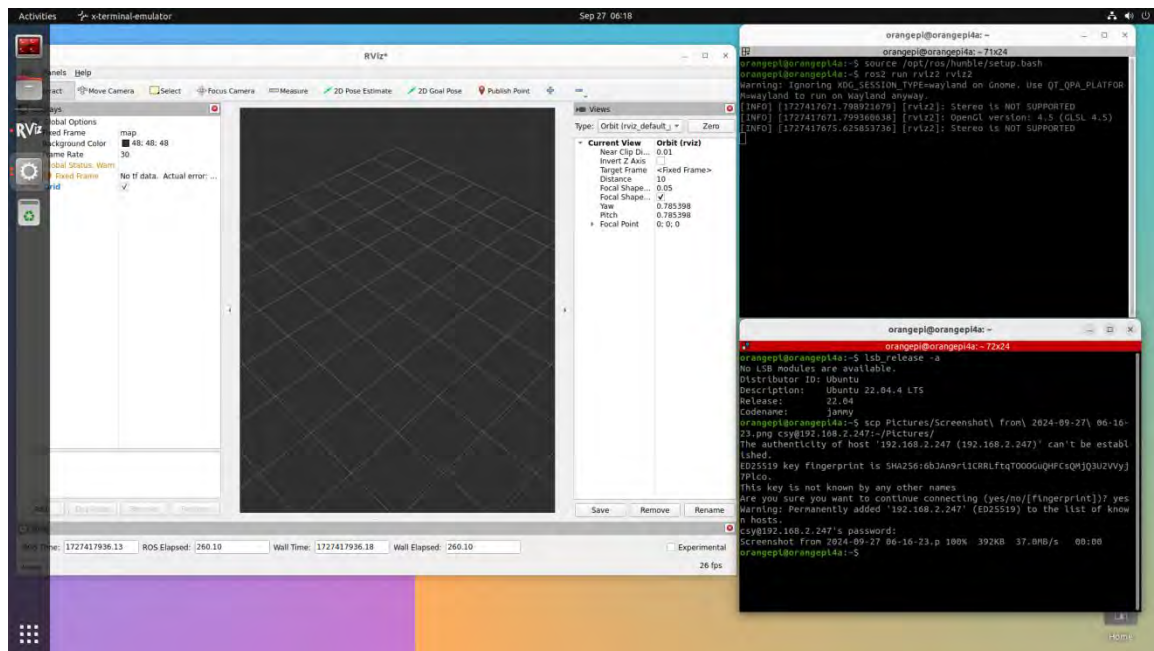
Call `ros2 <command> -h` for more detailed usage.

- 3) Then you can use the **test_ros.sh** script to test whether ROS 2 is installed successfully. If you can see the following print, it means that ROS 2 can run normally.

```
orangepi@orangepi:~$ test_ros.sh
[INFO] [1671174101.200091527] [talker]: Publishing: 'Hello World: 1'
[INFO] [1671174101.235661048] [listener]: I heard: [Hello World: 1]
[INFO] [1671174102.199572327] [talker]: Publishing: 'Hello World: 2'
[INFO] [1671174102.204196299] [listener]: I heard: [Hello World: 2]
[INFO] [1671174103.199580322] [talker]: Publishing: 'Hello World: 3'
[INFO] [1671174103.204019965] [listener]: I heard: [Hello World: 3]
```

- 4) Run the following command to open rviz2

```
orangepi@orangepi:~$ source /opt/ros/humble/setup.bash
orangepi@orangepi:~$ ros2 run rviz2 rviz2
```



- 5) Reference Documents

<http://docs.ros.org/en/humble/index.html>

<http://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debians.html>



3. 27. How to install kernel header files

1) The Linux image released by OPi comes with a deb package of kernel header files by default, which is stored in **/opt/**

```
orangePi@orangePi:~$ ls /opt/linux-headers*  
/opt/linux-headers-xxx-sun55iw3_x.x.x_arm64.deb
```

2) Use the following command to install the kernel header file deb package

```
orangePi@orangePi:~$ sudo dpkg -i /opt/linux-headers*.deb
```

3) After installation, you can see the folder where the kernel header files are located under **/usr/src**

```
orangePi@orangePi:~$ ls /usr/src  
linux-headers-x.x.x
```

4) Then you can compile the source code of the hello kernel module that comes with the Linux image. The source code of the hello module is in **/usr/src/hello**. After entering this directory, use the make command to compile it.

```
orangePi@orangePi:~$ cd /usr/src/hello/  
orangePi@orangePi:/usr/src/hello$ sudo make  
make -C /lib/modules/5.15.147-sun55iw3/build M=/usr/src/hello modules  
make[1]: Entering directory '/usr/src/linux-headers-5.15.147-sun55iw3'  
  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-5.15.147-sun55iw3'
```

5) After compilation, the **hello.ko** kernel module will be generated

```
orangePi@orangePi:/usr/src/hello$ ls *.ko  
hello.ko
```

6) Use the **insmod** command to insert the **hello.ko** kernel module into the kernel

```
orangePi@orangePi:/usr/src/hello$ sudo insmod hello.ko
```



7) 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.

```
orangepi@orangepi:/usr/src/hello$ dmesg | grep "Hello"  
[ 2871.893988] Hello Orange Pi -- init
```

8) 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. 28. How to use the 10.1 inch MIPI LCD screen

3. 28. 1. 10.1 inch MIPI screen assembly method

1) First prepare the necessary accessories

- a. 10.1 inch MIPI LCD display + touch screen



- b. Screen adapter board + 31pin to 40pin cable



c. 30pin MIPI cable



d. 12pin touch screen cable



2) Connect the 12-pin touch screen cable, 31-pin to 40-pin cable, and 30-pin MIPI cable to the screen adapter board as shown below. Note **that the blue insulation side of the touch screen cable should face down**, and the insulation sides of the other two cables should face up. If connected incorrectly, it will cause no display or inability to touch.



3) Place the adapter board with the connected cable on the MIPI LCD screen as shown below, and connect the MIPI LCD screen and the adapter board via a 31pin to 40pin cable.



4) Then connect the touch screen and the adapter board through the 12-pin touch screen cable, paying attention to the direction of the insulating surface



5) Finally, connect it to the LCD interface of the development board through the 30pin MIPI cable



3. 28. 2. How to open the 10.1-inch MIPI LCD screen configuration

1) The Linux image does not have the mipi lcd screen configuration turned on by default. If you need to use the mipi lcd screen, you need to turn it on manually.



2) The interface of the mipi lcd screen on the development board is shown in the figure below



3) The method to open mipi lcd configuration is as follows:

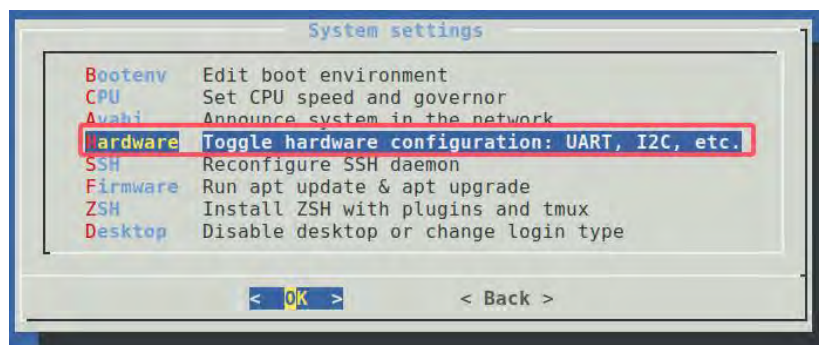
- a. First, run **orangepi-config**. Regular users should remember to add **sudo** privileges

```
orangepi@orangepi:~$ sudo orangepi-config
```

- b. Then select **System**



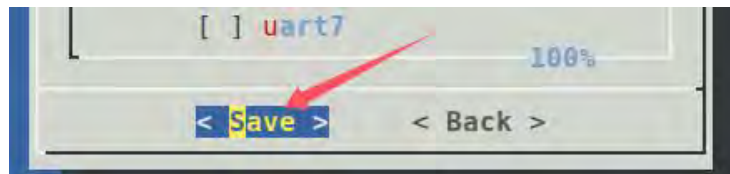
- c. Then select **Hardware**



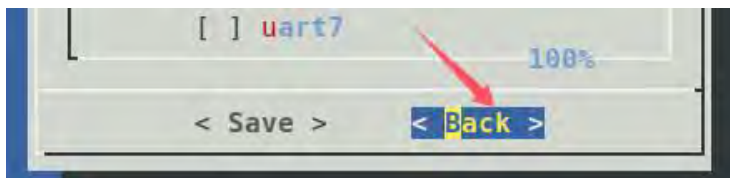
- d. Then use the directional keys on the keyboard to locate the **lcd**, and use a **space** to select it



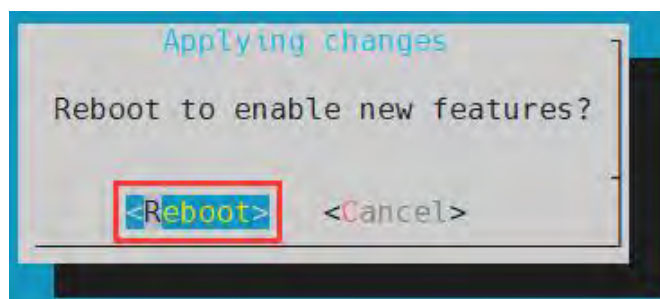
- e. Then select **<Save>** to save



- f. Then select **<Back>**



- g. Then select **<Reboot>** to restart the system for the configuration to take effect



The above settings will eventually add **overlays=lcd** in `/boot/orangepiEnv.txt`. After setting it up, you can check it first. If this line of configuration does not exist, then there is a problem with the settings.

If you find using `orangepi-config` troublesome, you can also use the Vim editor to open `/boot/orangepiEnv.txt` and add the "**overlays=lcd**" configuration line.

```
orangepi@orangepi:~$ cat /boot/orangepiEnv.txt | grep "lcd"
```

overlays=lcd **#Example configuration**

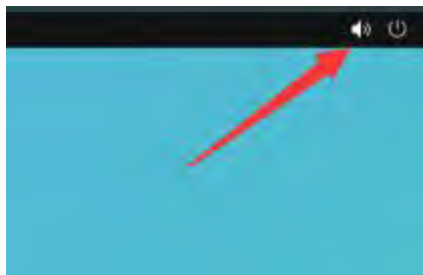
4) Then restart the Linux system

5) After restarting, you can see the display of the LCD screen as shown below (the default is vertical screen):

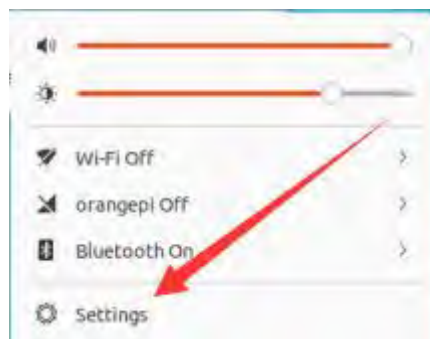


3. 28. 3. Methods for rotating display and touch directions

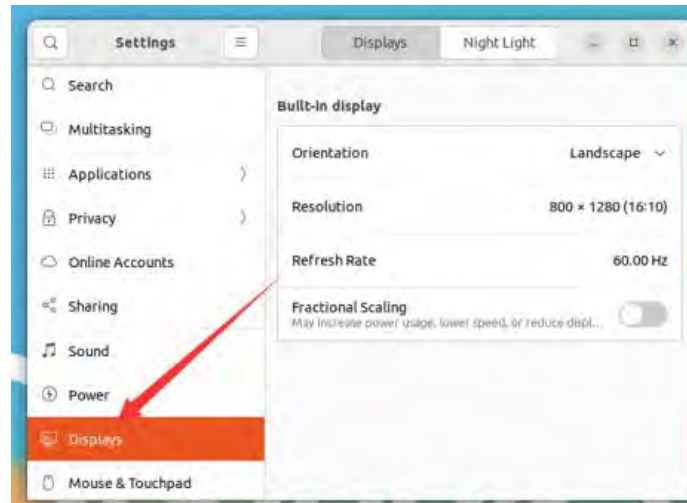
- 1) First click on the area in the upper right corner of the desktop



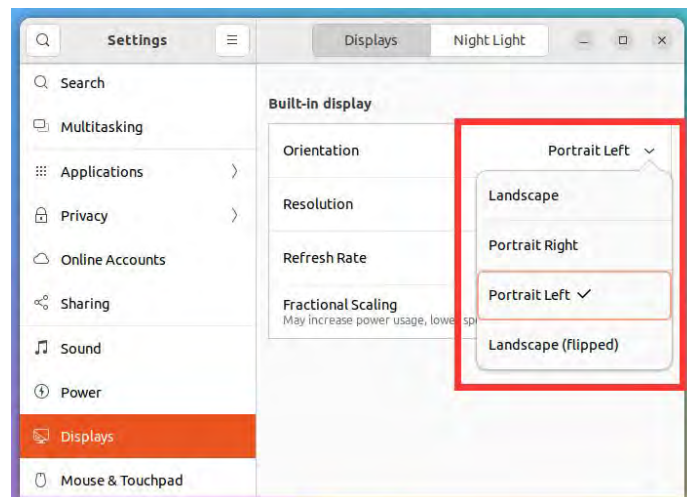
- 2) Then open Settings



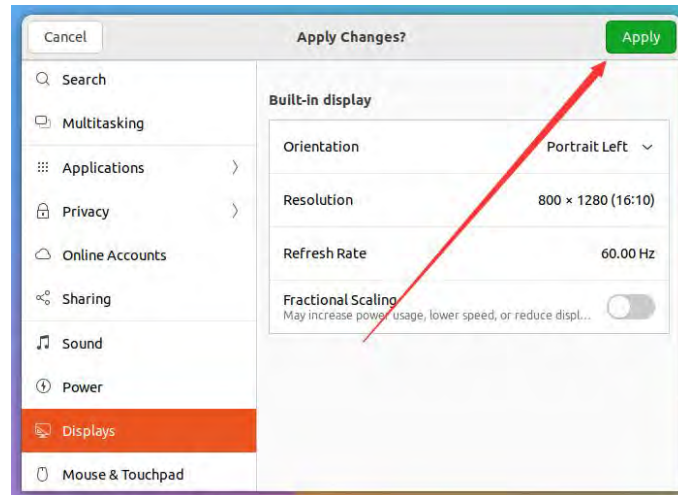
- 3) Then select **Displays**



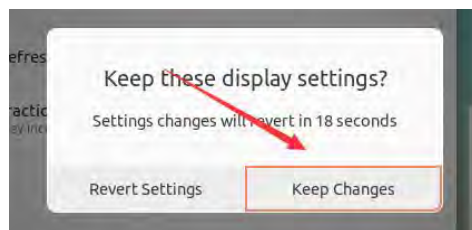
4) Then select the direction you want to rotate in **Orientation** of **Displays**



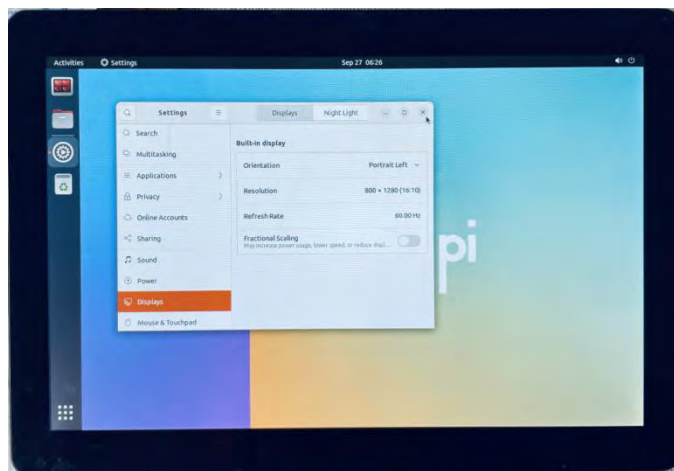
5) Then select **Apply**



6) Then you can see that the screen has been rotated. At this time, you need to select **Keep Changes** to finalize the rotation.



7) The LCD screen will display the following after rotating 90 degrees:



8) The touch function of the Linux system LCD screen will rotate with the rotation of the display direction, without any other settings



3. 29. How to use the eDP screen

3. 29. 1. Assembly method of eDP screen

1) Currently only a 15.6-inch eDP screen is compatible, and the accessories included are as follows:

- a. 0.5 pitch 30pin single head same direction cable



- b. 15.6-inch eDP display, resolution 1920x1080.



2) Connect the FPC end of the 30-pin single-head unidirectional cable to the eDP interface of the development board, and the other end to the eDP interface of the screen.



3. 29. 2. How to open eDP screen configuration

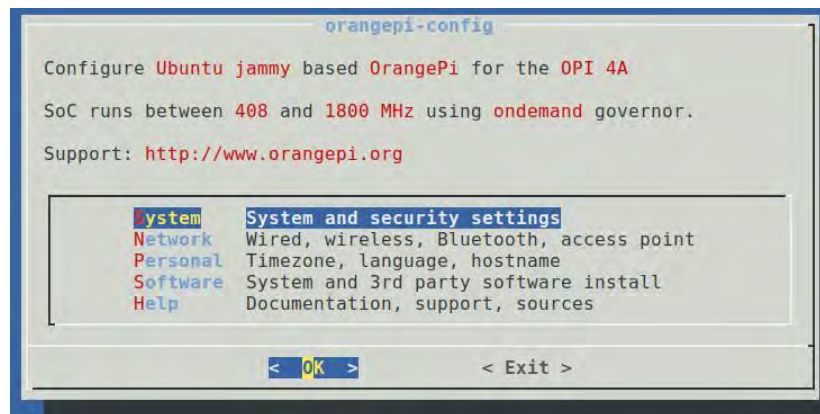
Please note that the method described below is only applicable to adapted eDP screens. If the customer is using an unadapted screen, it will not light up according to the method below.

1) Linux images are not configured with eDP screens enabled by default. If you need to use eDP screens, you need to manually enable them.

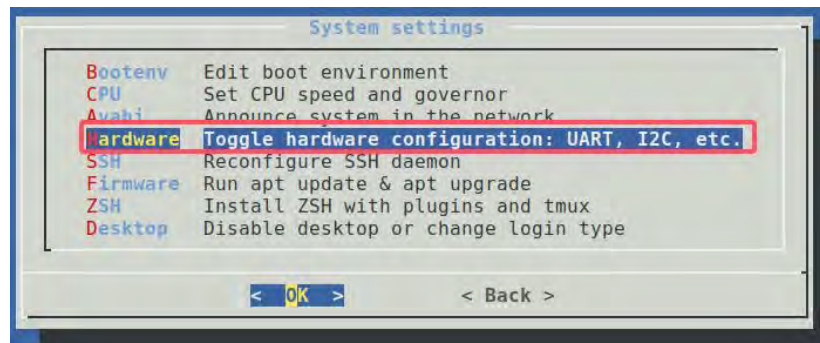
- a. First, run **orangeypi-config**. Regular users should remember to add **sudo** privileges

```
orangeypi@orangeypi:~$ sudo orangeypi-config
```

- b. Then select **System**



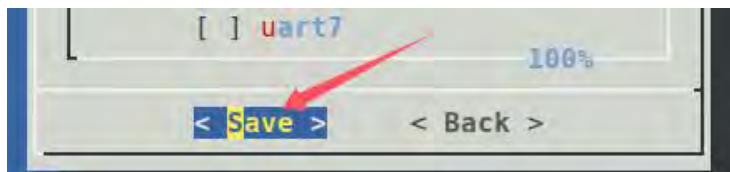
- c. Then select **Hardware**



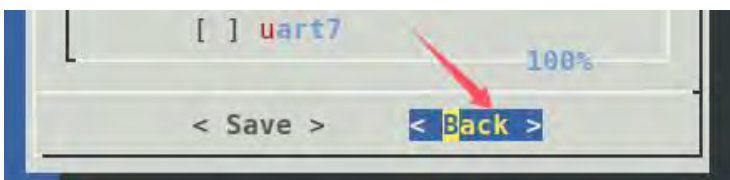
- d. Then use the directional keys on the keyboard to locate **edp**, and use a **space** to select it



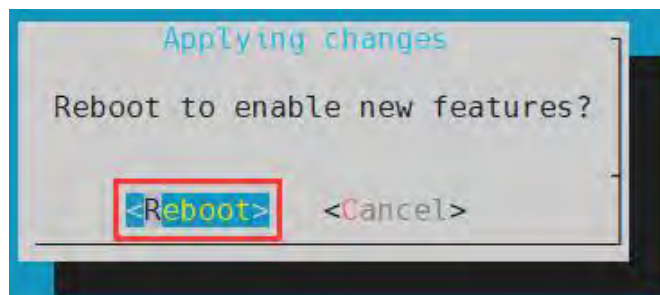
- e. Then select **<Save>** to save



- f. Then select **<Back>**



- g. Then select **<Reboot>** to restart the system for the configuration to take effect



The above settings will eventually add **overlays=edp** in **/boot/orangepiEnv.txt**. After setting it up, you can check it first. If this line of configuration does not exist, then there is a problem with the settings.

If you find using **orangepi-config** troublesome, you can also use the Vim editor to open **/boot/orangepiEnv.txt** and add the "**overlays=edp**" configuration line.

```
orangepi@orangepi:~$ cat /boot/orangepiEnv.txt | grep "edp"
```

overlays=edp **#Example configuration**

2) After startup, you can see the eDP screen display as shown below:



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

3. 30. 1. Debian Bookworm System

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

a. gcc version is as follows

```
orangepi@orangepi:~$ gcc --version
gcc (Debian 12.2.0-14) 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**

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

2) Debian Bookworm has Python 3 installed by default

- a. The specific version of Python is as follows

```
orangepi@orangepi:~$ python3
Python 3.11.2 (main, Mar 13 2023, 12:18:29) [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

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

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

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

3) Debian Bookworm does not install Java compilation tools and runtime environment by default

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

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

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

```
orangepi@orangepi:~$ java --version
```

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

```
orangepi@orangepi:~$ 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**

```
orange@orange:~$ javac hello_world.java
orange@orange:~$ java hello_world
Hello World!
```

3.30.2. Ubuntu Jammy System

1) Ubuntu Jammy 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. gcc version is as follows

```
orange@orange:~$ gcc --version
gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0
Copyright (C) 2021 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

```
orange@orange:~$ 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) Ubuntu Jammy has Python 3 installed by default

- a. The specific version of Python3 is as follows

```
orange@orange:~$ python3
Python 3.10.12 (main, Jul 29 2024, 16:56:48) [GCC 11.4.0] 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

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

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

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

3) Ubuntu Jammy does not install Java compilation tools and runtime environment by default

- a. You can use the following command to install openjdk-18

```
orangepi@orangepi:~$ sudo apt install -y openjdk-18-jdk
```

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

```
orangepi@orangepi:~$ java --version
openjdk 18.0.2-ea 2022-07-19
OpenJDK Runtime Environment (build 18.0.2-ea+9-Ubuntu-222.04)
OpenJDK 64-Bit Server VM (build 18.0.2-ea+9-Ubuntu-222.04, mixed mode, sharing)
```

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

```
orangepi@orangepi:~$ 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**

```
orangepi@orangepi:~$ javac hello_world.java
orangepi@orangepi:~$ java hello_world
Hello World!
```



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

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

3.31.1.1. 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

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

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

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

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

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

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

3.31.1.2. How to upload files using FileZilla

1) First install filezilla in your Ubuntu PC

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

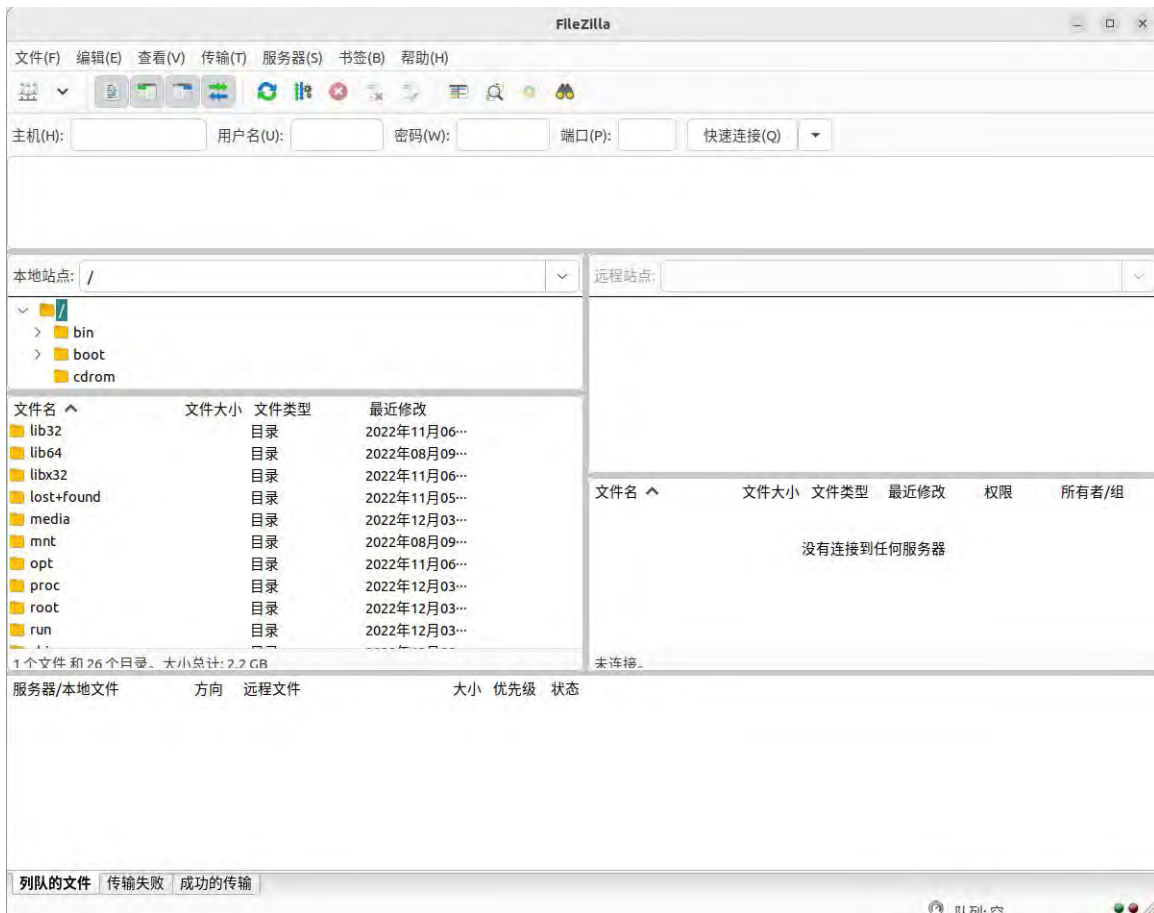
2) Then open filezilla using the following command

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

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



right is empty.



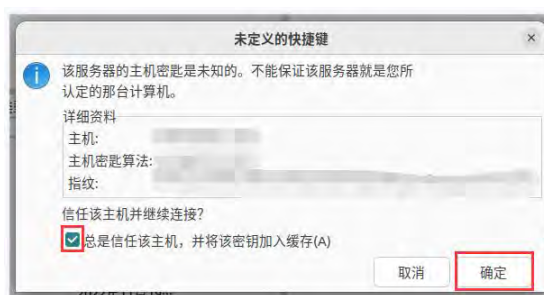
4) The method of connecting the development board is shown in the figure below



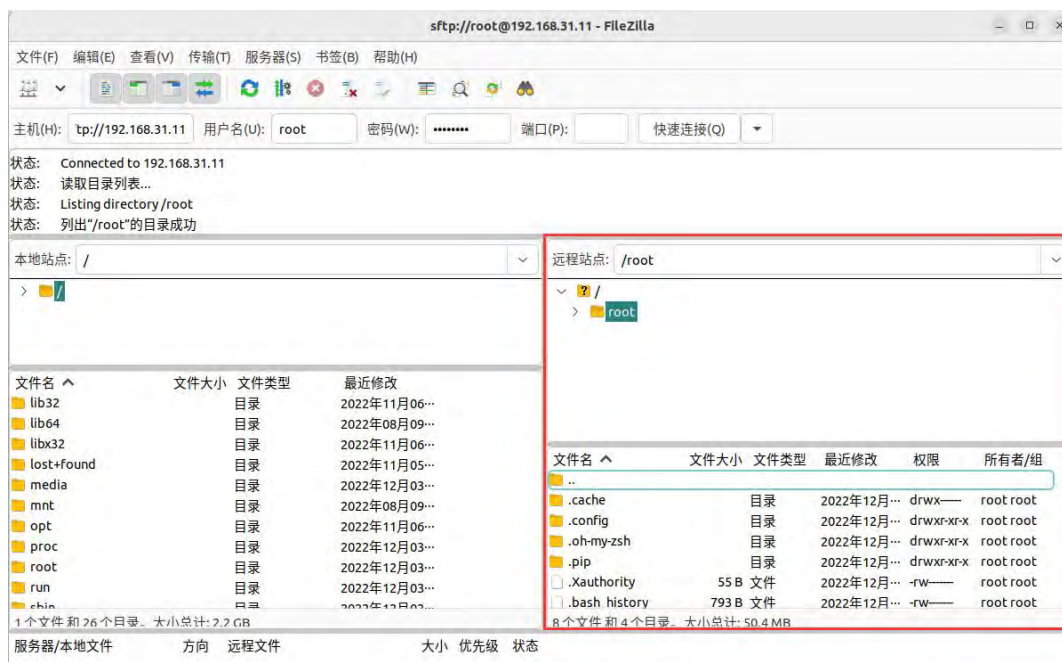
5) Then select **Save Password** and click **OK**



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



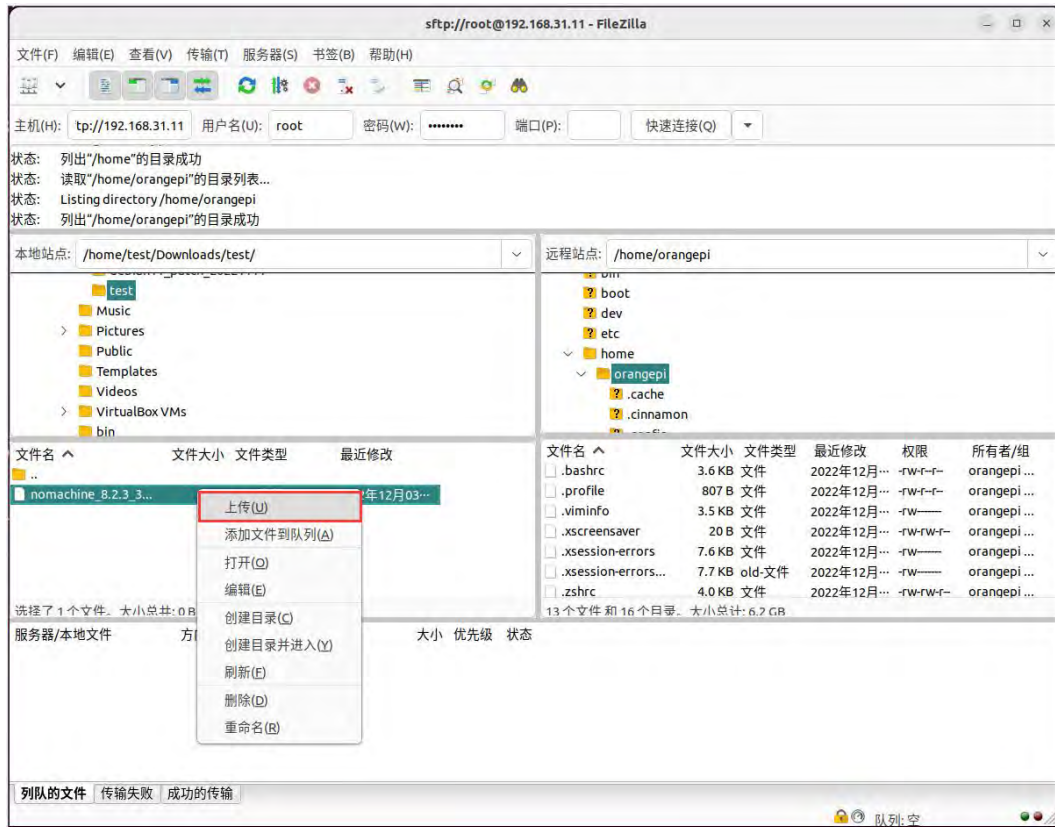
7) After the connection is successful, you can see the directory structure of the development board's Linux file system on the right side of the filezilla software



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



filezilla software, 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 a folder is the same as the method of uploading a file, so I will not go into details here.

3. 31. 2. How to upload files from Windows PC to the Linux system of the development board

3. 31. 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>



FileZilla The free FTP solution

Home
FileZilla
Features
Screenshots
Download
Documentation
FileZilla Pro
FileZilla Server
Download
Community
Forum
Wiki
General
FAQ
Support
Contact
License
Privacy Policy
Trademark Policy
Development
Source code
Nightly builds
Translations
Version history
Changelog
Issue tracker
Other projects

Download FileZilla Client for Windows (64bit x86)
The latest stable version of FileZilla Client is 3.62.2
Please select the file appropriate for your platform below.
Windows (64bit x86)
Download FileZilla Client Click here to download
This installer may include bundled offers. Check below for more options.
The 64bit versions of Windows 8.1, 10 and 11 are supported.
More download options
Other platforms:
Not what you are looking for?
Show additional download options

Please select your edition of FileZilla Client

	FileZilla	FileZilla with manual	FileZilla Pro	FileZilla Pro + CLI
Standard FTP	Yes	Yes	Yes	Yes
FTP over TLS	Yes	Yes	Yes	Yes
SFTP	Yes	Yes	Yes	Yes
Comprehensive PDF manual	-	Yes	Yes	Yes
Amazon S3	-	-	Yes	Yes
Backblaze B2	-	-	Yes	Yes
Dropbox	-	-	Yes	Yes
Microsoft OneDrive	-	-	Yes	Yes
Google Drive	-	-	Yes	Yes
Google Cloud Storage	-	-	Yes	Yes
Microsoft Azure Blob + File Storage	-	-	Yes	Yes
WebDAV	-	-	Yes	Yes
OpenStack Swift	-	-	Yes	Yes
Box	-	-	Yes	Yes
Site Manager synchronization	-	-	Yes	Yes
Command-line interface	-	-	-	Yes
Batch transfers	-	-	-	Yes

Then select here to download

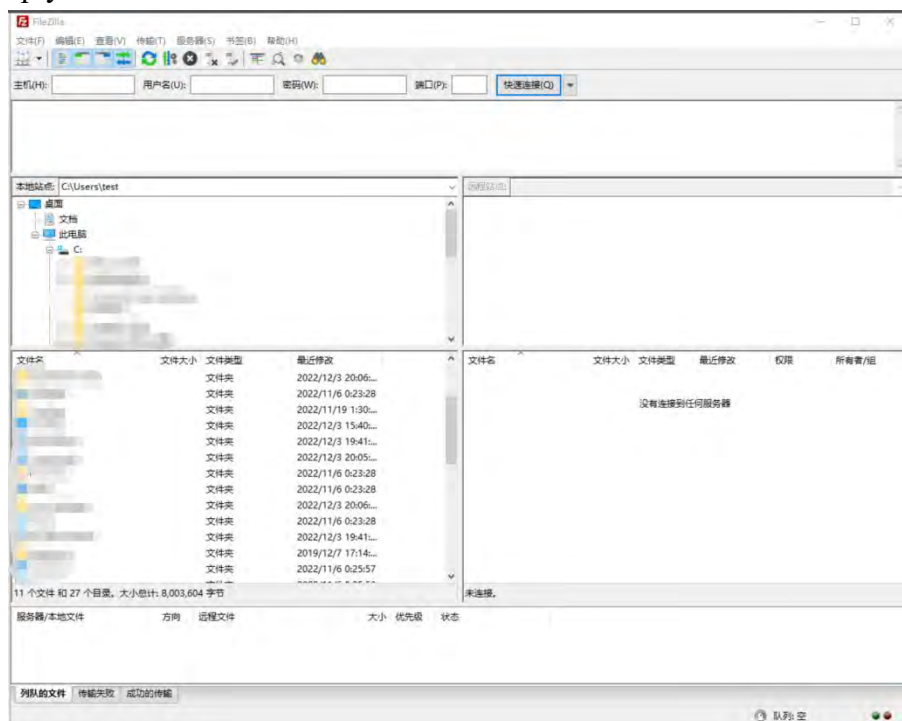
Download **Select** **Select** **Select**

2) The downloaded installation package is as follows, then double-click to install directly **FileZilla_Server_1.5.1_win64-setup.exe**

During the installation process, select **Decline** on the following installation interface, then select **Next>**



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:



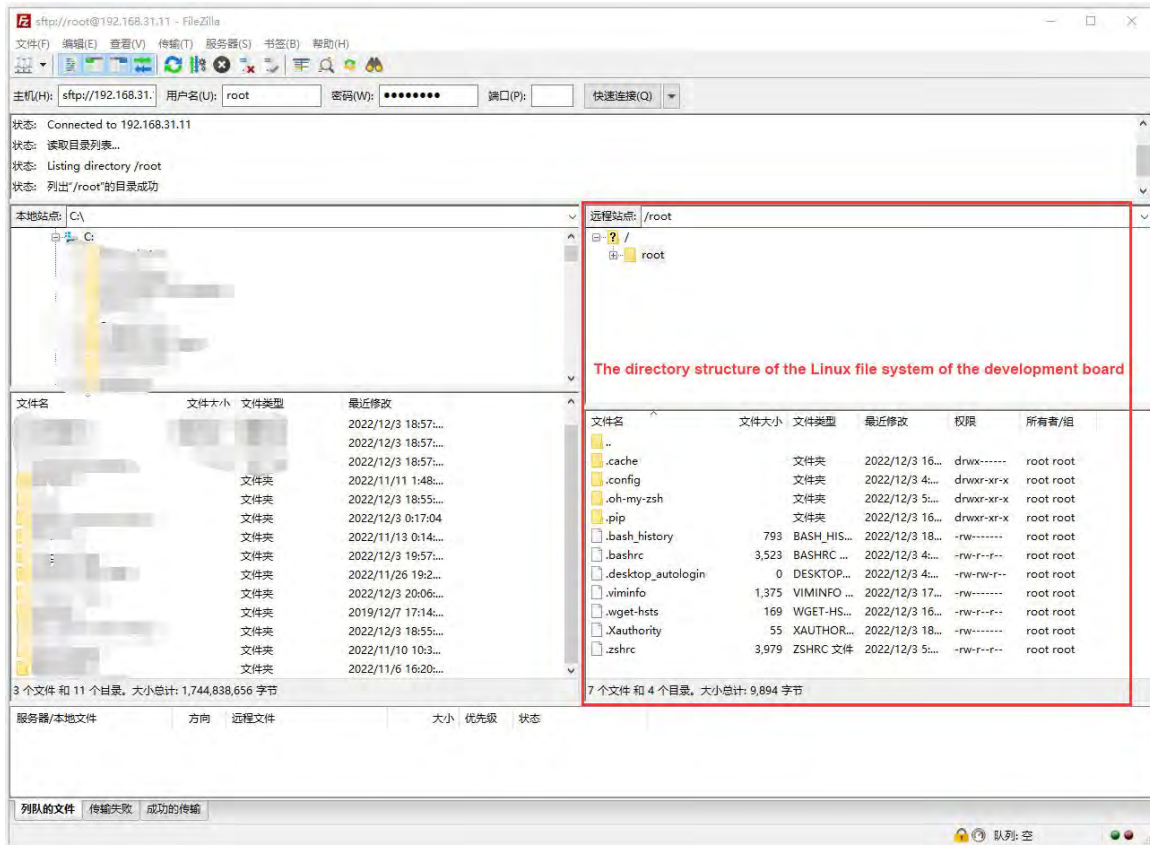
5) Then select **Save Password** and click **OK**



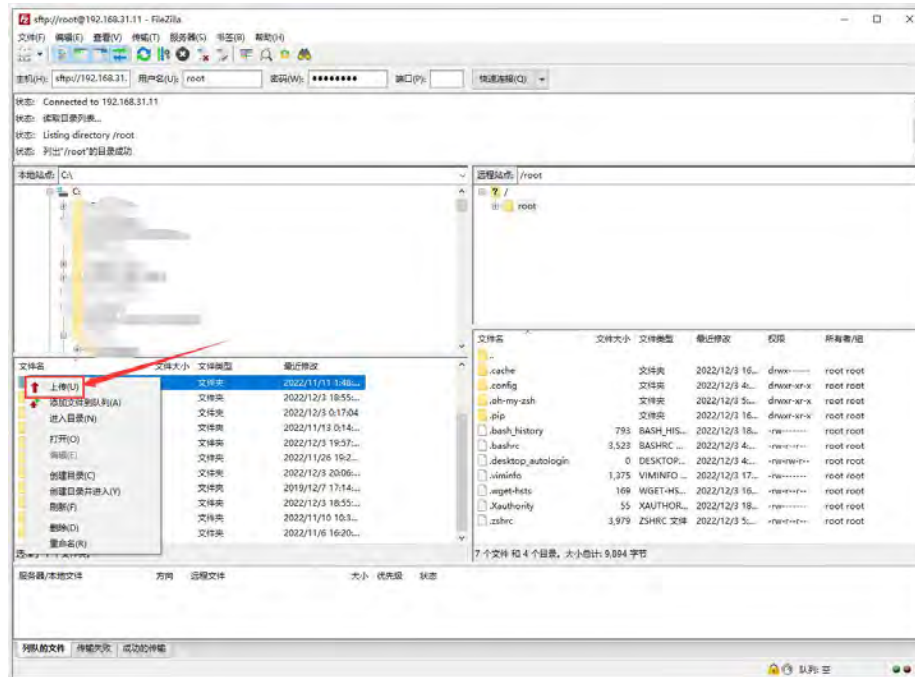
6) Then select **Always trust this host** and click **OK**



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 be uploaded to the development board on the right side of the filezilla software, then select the file to be uploaded in the Windows PC on the left side of the filezilla software, then 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 a folder is the same as the method of uploading a file, so I will not go into details here.

3. 32. Instructions for use of NPU

3. 32. 1. Board environment preparation

1) First, you need to install opencv and cmake on the development board.

```
orangePi@orangePi:~$ sudo apt update
orangePi@orangePi:~$ sudo apt install libopencv-dev cmake
```

2) Then download the compressed package of the NPU sample program from the official Baidu Cloud.

3) Then upload the NPU sample program compressed package to the development board and decompress it.



```
orangeypi@orangeypi:~$ tar -xvf board-demo.tar.gz
```

4) After the compressed package is decompressed, the directory structure is as follows.

```
orangeypi@orangeypi:~$ cd board-demo/
orangeypi@orangeypi:~/board-demo$ ls
chineseocr  common  head_pose  lenet  libawnn_viplite  libawutils
mobilenet_v2_ssd_demo  struct2depth  struct2depth  yolov5
```

5) Then you need to copy the NPU library to the system's /usr/lib directory, which is needed to run the sample program below.

```
orangeypi@orangeypi:~/board-demo$ sudo cp ./common/lib_linux_aarch64/T527/*.so
/usr/lib
```

3. 32. 2. Board Example Run

3. 32. 2. 1. Run the mobilenet_v2_ssd target detection example

1) First compile the mobilenet_v2_ssd target detection example

```
orangeypi@orangeypi:~/board-demo$ cd mobilenet_v2_ssd_demo
orangeypi@orangeypi:~/board-demo/mobilenet_v2_ssd_demo$ mkdir build
orangeypi@orangeypi:~/board-demo/mobilenet_v2_ssd_demo$ cd build
orangeypi@orangeypi:~/board-demo/mobilenet_v2_ssd_demo/build$ cmake ..
orangeypi@orangeypi:~/board-demo/mobilenet_v2_ssd_demo/build$ make
```

2) Then execute the following command to run the example.

```
orangeypi@orangeypi:/root/board-demo/mobilenet_v2_ssd_demo/build$ ./mbv2-ssd-demo
-b ../model/mbv2_ssd_x527.nb -i ../000012.jpg
```

3) The output information of normal program operation is as follows.

```
orangeypi@orangeypi:/root/board-demo/mobilenet_v2_ssd_demo/build$ ./mbv2-ssd-demo
-b ../model/mbv2_ssd_x527.nb -i ../000012.jpg
.....
get output finished.
mbv2_ssd_postprocess.cpp run.
car = 0.97705 at 162 94 179 x 170
```



```
postprocess time : 0.025 Sec
destory npu finished.
~NpuUint.
```

4) The output will be saved as `ssd_out.png` in the current directory

```
orange@orange:~/board-demo/mobilenet_v2_ssd_demo/build$ ls ssd_out.png
ssd_out.png
```



3.32.2.2. Run the yolov5 object detection example

1) First compile the yolov5 example.

```
orange@orange:~/board-demo$ cd yolov5
orange@orange:~/board-demo/yolov5$ mkdir build
orange@orange:~/board-demo/yolov5$ cd build
orange@orange:~/board-demo/yolov5/build$ cmake ..
orange@orange:~/board-demo/yolov5/build$ make
```

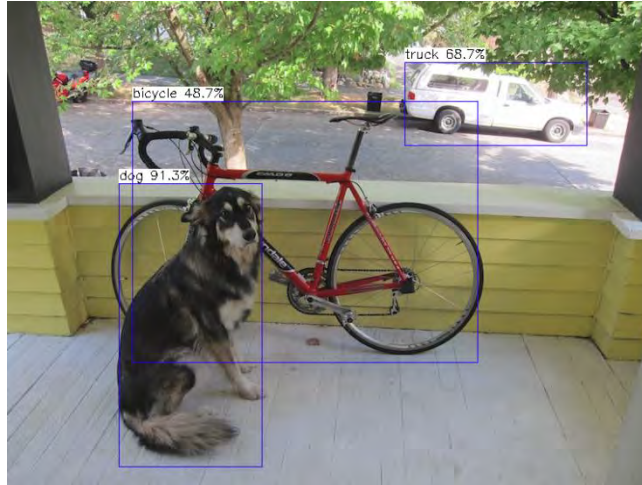
2) Then execute the following command to run the example.

```
orange@orange:~/board-demo/yolov5/build$ ./yolov5 ../model/v2/yolov5.nb ../input_data/dog.jpg
./yolov5 nbg input
VIPLite driver software version 1.13.0.0-AW-2023-10-19
yolov5_preprocess.cpp run.
yolov5_postprocess.cpp run.
detection num: 3
16: 91%, [ 135, 218, 305, 553], dog
7: 69%, [ 473, 75, 689, 174], truck
1: 49%, [ 151, 121, 561, 431], bicycle
```



3) The output will be saved as result.png in the current directory

```
orangepi@orangepi4a:~/board-demo/yolov5/build$ ls result.png
result.png
```



3.32.2.3. Run the head_pose human posture recognition example

1) First compile the head_pose example.

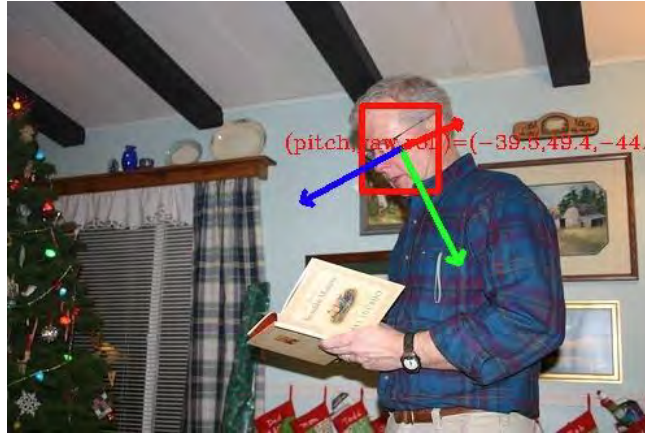
```
orangepi@orangepi:~/board-demo$ cd head_pose
orangepi@orangepi:~/board-demo/head_pose$ mkdir build
orangepi@orangepi:~/board-demo/head_pose$ cd build
orangepi@orangepi:~/board-demo/head_pose/build$ cmake ..
orangepi@orangepi:~/board-demo/head_pose/build$ make
```

2) Then execute the following command to run the example.

```
orangepi@orangepi:~/board-demo/head_pose/build$ ./head_pose -b1 ../model/rfb_landm_face_320_320_sim_x527.nb
-b2 ../model/head_pose_x527.nb -i ../input_data/000438.jpg
```

3) The output result will be saved to head_pose_result.jpg in the current directory.

```
orangepi@orangepi:~/board-demo/head_pose/build$ ls head_pose_result.jpg
head_pose_result.jpg
```



3. 32. 2. 4. Run the resnet50 image classification example

1) First compile the resnet50 example.

```
orangepi@orangepi:~/board-demo$ cd resnet50
orangepi@orangepi:~/board-demo/resnet50$ mkdir build
orangepi@orangepi:~/board-demo/resnet50$ cd build
orangepi@orangepi:~/board-demo/resnet50/build$ cmake ..
orangepi@orangepi:~/board-demo/resnet50/build$ make
```

2) Then execute the following command to run the example.

```
orangepi@orangepi:~/board-demo/resnet50/build$ ./resnet50 ../model/v2/resnet50.nb ../
input_data/dog_224_224.jpg
```

3) The output result information is as follows, which outputs the top 5 predicted by the model, among which the most likely category is collie

```
orangepi@orangepi:~/board-demo/resnet50/build$ ./resnet50 ../model/v2/resnet50.nb ../
input_data/dog_224_224.jpg
...
===== top5 =====
class id: 231, prob: 15.432617, label: collie
class id: 230, prob: 13.103271, label: Shetland sheepdog, Shetland sheep dog, Shetland
class id: 169, prob: 12.617920, label: borzoi, Russian wolfhound
class id: 224, prob: 12.423828, label: groenendael
class id: 160, prob: 10.191406, label: Afghan hound, Afghan
class_postprocess success.
```



3. 32. 2. 5. Run struct2depth depth detection example

1) First compile the struct2depth example.

```
orangepi@orangepi:~/board-demo$ cd struct2depth
orangepi@orangepi:~/board-demo/struct2depth$ mkdir build
orangepi@orangepi:~/board-demo/struct2depth$ cd build
orangepi@orangepi:~/board-demo/struct2depth/build$ cmake ..
orangepi@orangepi:~/board-demo/struct2depth/build$ make
```

2) Then execute the following command to run the example

```
orangepi@orangepi:~/board-demo/struct2depth/build$ ./struct2depth
-b ../model/v2/struct2depth.nb -i ../input_data/0015.jpg
```

3) The depth information of the model inference will be saved in jpg and txt files.

```
orangepi@orangepi:~/board-demo/struct2depth/build$ ls disp_* output_*
disp_color.jpg  disp_show.jpg  output_1.txt  output_3.txt
disp_gray.jpg   output_0.txt   output_2.txt
```

3. 32. 2. 6. Run ChineseOCR text recognition example

1) First compile the ChineseOCR example.

```
orangepi@orangepi:~/board-demo$ cd chineseocr
orangepi@orangepi:~/board-demo/chineseocr$ mkdir build
orangepi@orangepi:~/board-demo/chineseocr$ cd build
orangepi@orangepi:~/board-demo/chineseocr/build$ cmake ..
orangepi@orangepi:~/board-demo/chineseocr/build$ make
```

2) Then execute the following command to run the example.

```
orangepi@orangepi:~/board-demo/chineseocr/build$ ./chineseocr -d ../model/v2/ -1 dbnet_
1024 -2 angle_net -3 crnn_lite_lstm_256 -4 keys.txt -i ../input_data/1.jpg
```

3) The output result is as follows, and you can see that the text in the image has been recognized.

```
orangepi@orangepi:~/board-demo/chineseocr/build$ ./chineseocr -d ../model/v2/ -1 dbnet_
```



```
1024 -2 angle_net -3 crnn_lite_lstm_256 -4 keys.txt -i ../input_data/1.jpg
...
=====End detect=====
FullDetectTime(903.447417ms)
We at Allwinner Technology
AI
chip
take off12345666666 !

run finished.
~CrnnNet.
~AngleNet.
~DbNet.
~NpuUint.
```

3. 33. How to burn Linux image to eMMC

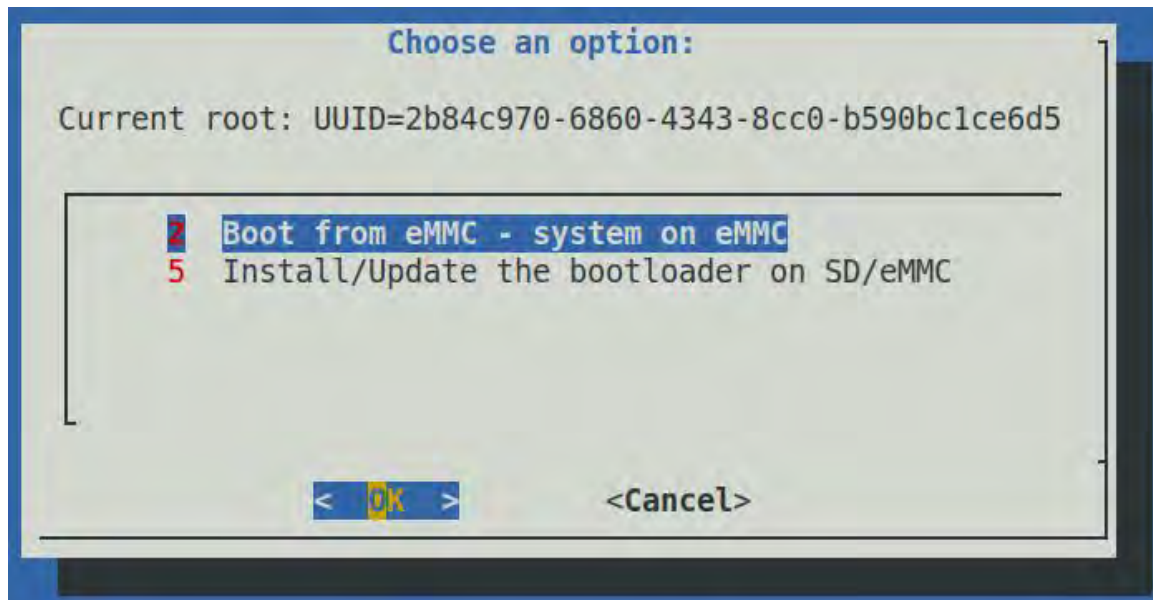
Note that the development board can be started through a TF card or eMMC, and the priority of the TF card is higher than that of the eMMC. In other words, if a TF card is inserted into the development board and there is a system in the TF card, the system in the TF card will be started by default, and the system in the eMMC will not be started.

1) Burning the Linux image to the eMMC requires the use of a TF card. First, burn the Linux image to the TF card, then start the development board and enter the Linux system

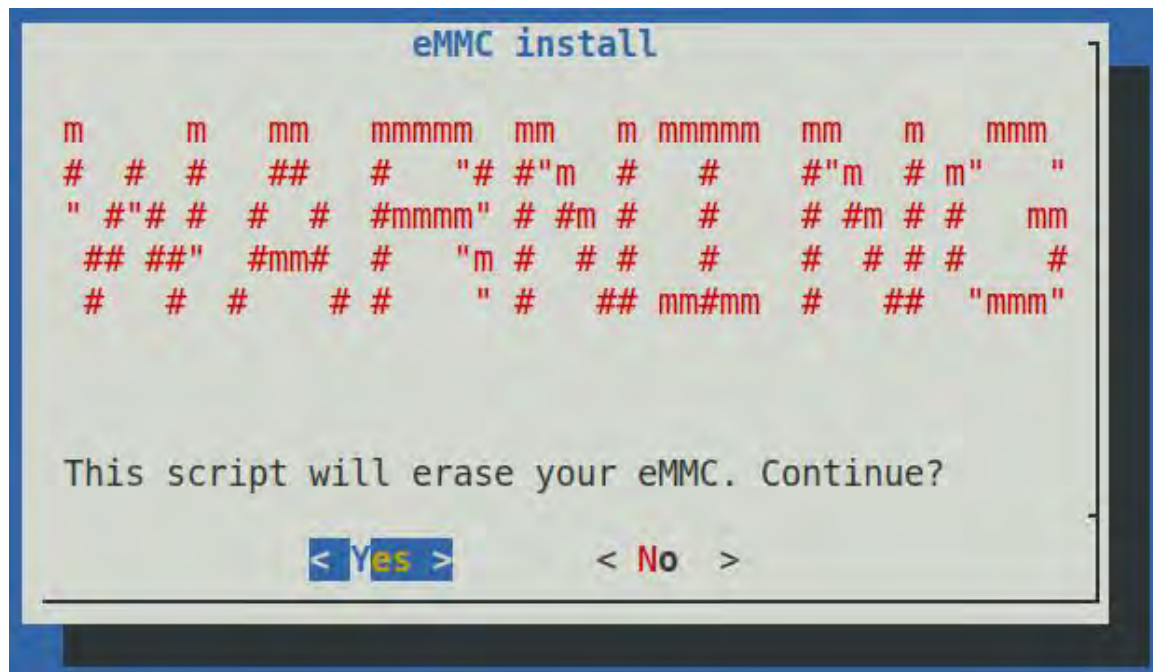
2) Then run the **nand-sata-install** script, remember to **add sudo permissions**.

```
orangeypi@orangeypi:~$ sudo nand-sata-install
```

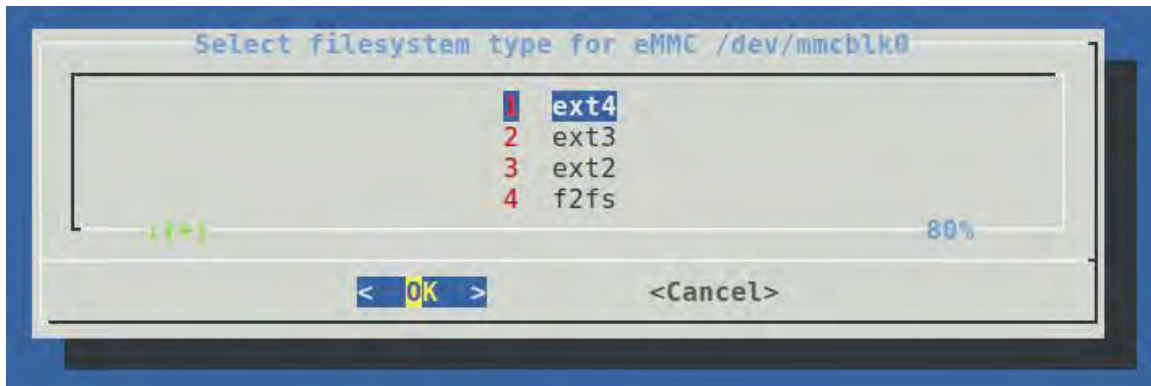
3) Then select **2 Boot from eMMC - system on eMMC**



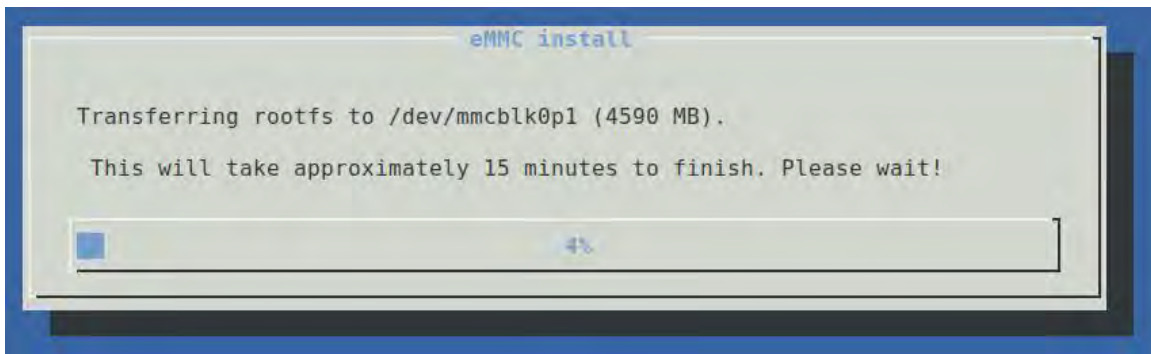
4) Then a warning will pop up, the script will erase all data on the eMMC, select **<Yes>** to continue



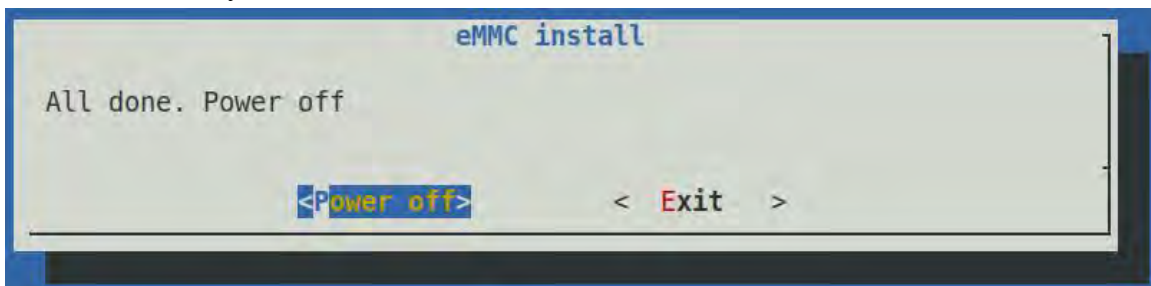
5) You will then be prompted to select the type of file system. Five file systems are supported: ext2/3/4, f2fs, and btrfs.



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



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



8) Then remove the TF card and power on again, the Linux system in the eMMC will start.

3. 34. Method for burning Linux images to SPIFlash+NVMe SSD

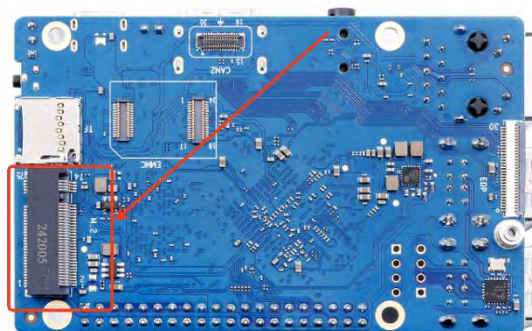
1) Firstly, it is necessary to prepare an NVMe SSD solid state drive with a PCIe interface



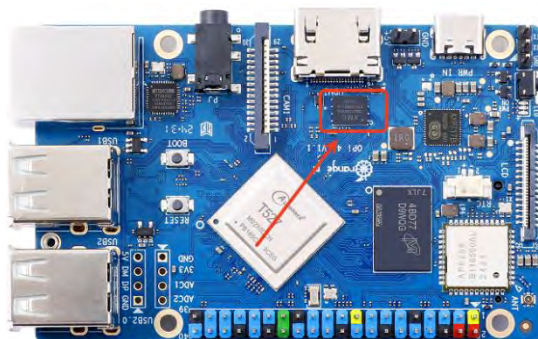
specification of PCIe2.0x2 for the M.2 slot on the development board.



2) Then insert the NVMe SSD into the M.2 PCIe interface of the development board and secure it in place.



3) The position of SPI Flash on the development board is shown in the following figure, and no other settings are required before starting to burn.



4) Burning the image to SPIFlash+NVMe SSD requires the use of a TF card, so the first step is to burn the Linux image onto the TF card, and then use the TF card to boot the development board into the Linux system. The method of burning a Linux image to a TF card can be found in the two sections: [the method of burning a Linux image to a TF card based on Windows PC](#) and [the method of burning a Linux image to a TF card based on Ubuntu PC](#).

5) After starting the linux system with a TF card, you can burn the image to SPI



Flash+NVMe SSD.

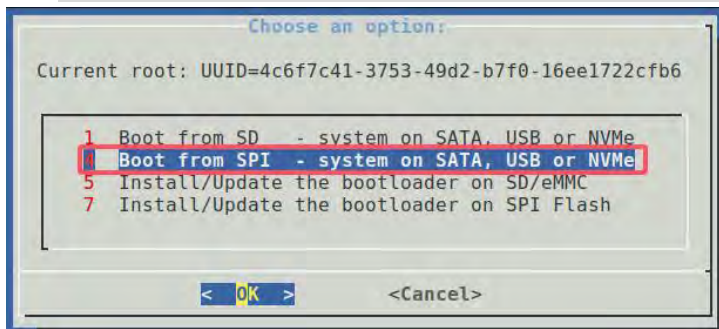
- a. First, create a partition for NVMe SSD.

```
orange@orange:~$ sudo parted --script /dev/nvme0n1 mklabel gpt mkpart primary ext4 8192s 100%
```

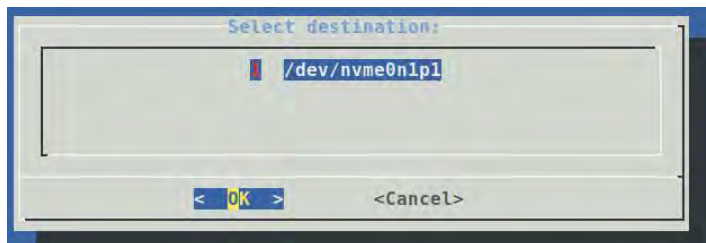
- b. Then run **nand-sata-install**, **remember to add sudo privileges for regular users**.

```
orange@orange:~$ sudo nand-sata-install
```

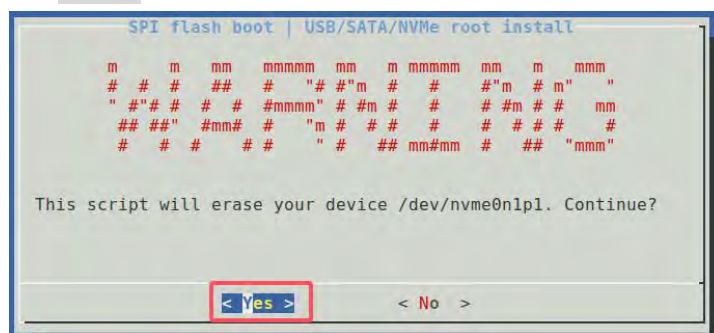
- c. Then select **4 Boot from SPI - system on SATA, USB or NVMe**.



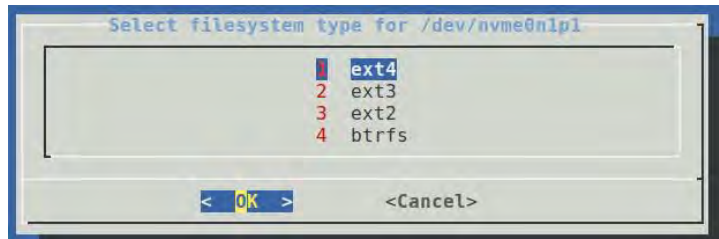
- d. Then press enter to confirm



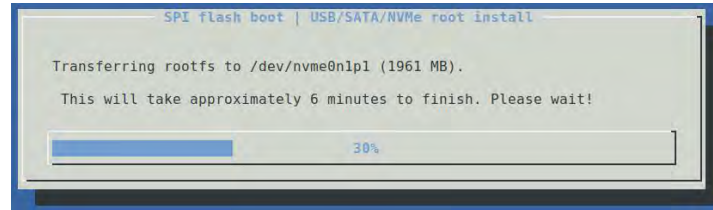
- e. Then select **<Yes>**.



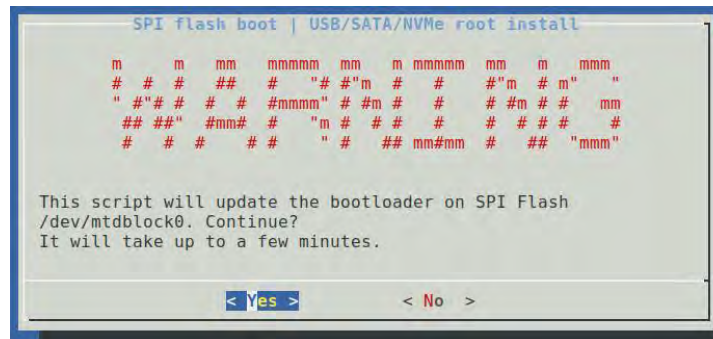
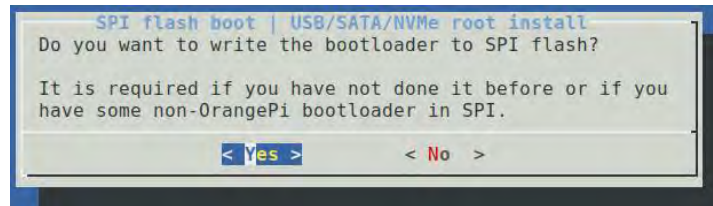
- f. Then it will prompt to select the type of file system.



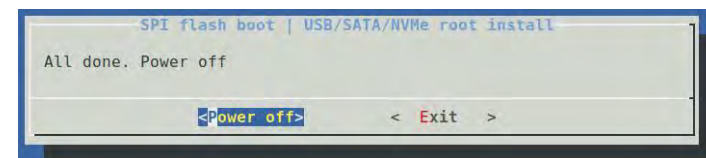
- g. Then it will start formatting the NVMe SSD, and after formatting is complete, it will start burning the system into the NVMe SSD.



- h. Then please be patient and wait for the burning to complete. After burning, you will be prompted whether to burn the bootloader to SPI Flash, and then select **<Yes>**.



- i. After burning, the following options will be prompted, you can choose **<Power off>** to shut down directly



3.35. Instructions for using the system backup script opi-bking

opi-bking is a system backup script designed for the Orange Pi series development board, which can help users fully backup their currently running Linux system to an image file for subsequent recovery or migration to other devices.

Before starting the backup, please ensure that there is enough space left in the



system, otherwise the backup may fail.

1) Run the **opi-bking** script to start backing up the system.

```
orange@orange:~$ sudo opi-bking
```

2) If there is no **opi-bking** script in the system, you can download it from the following repository by yourself.

<https://gitee.com/orangepi-xunlong/opi-bking>

3) The script will check the root file system size and create a properly sized image file, then use the **rsync** command to copy system data to the image file. The specific process is as follows.

```
orange@orange:~$ sudo opi-bking
[ o.k. ] Starting backup to image [ /mnt/backup.img ]
[ o.k. ] Current rootfs size [ 2561 MiB ]
[ o.k. ] Creating blank image for rootfs [ 3500 MiB ]
[ .... ] dd: 3.42GiB [ 126MiB/s] [=====] 100%
[ o.k. ] Creating partitions [ root: ext4 ]
[ .... ] Creating rootfs [ ext4 on /dev/loop1p1 ]
[ .... ] Copying file to [ / ] [#####] 100%
[ .... ] Re-enabling [ orangepi-resize-filesystem ]
[ o.k. ] Writing U-boot bootloader [ /dev/loop1 ]
[ o.k. ] Unmounting [ /mnt/opi-bking.iaioMi/rootfs ]
[ o.k. ] Backup completed [ /mnt/backup.img ]
orange@orange:~$
```

4) When the script displays the message '**Backup completed**', it indicates that the backup has ended. The image generation path is shown below,

```
orange@orange:~$ ls /mnt/
backup.img
```

5) **backup.img** is the image file that can be burned.

3. 36. Usage of Linux Overlayoot

In some user scenarios, the system often shuts down by directly cutting off power instead of executing the **poweroff** command, which may cause file system damage and prevent normal startup. To prevent such issues, it is recommended to enable the overlay root feature and mount the root file system as read-only. This ensures that even in the event of an abnormal power outage, the original file system



is not affected, thereby improving the stability and reliability of the system.

This feature has been tested on Ubuntu Jammy, Ubuntu Focal, Ubuntu Noble, and Debian Bookworm, but it is currently not supported on Debian Bullseye.

Overlayboot is a technology based on Overlayfs that can mount the Linux root file system (/) as read-only and provide writable storage through an overlay. All user modifications (such as file writes, configuration changes, etc.) will be redirected to the writable layer, while the original root file system remains unchanged, making it suitable for scenarios such as embedded devices and servers that require high system stability.

Overlayboot consists of two layers: a read-only layer (lowerdir) and a writable layer (upperdir). The read-only layer is the original root file system (/), mounted in read-only mode; The writable layer is used to store user modified data, which is merged with the read-only layer to form the final root file system view, making users unaware of the hierarchical structure of the underlying layer during operation.

The writable layer can be stored in tmpfs or block devices. When using tmpfs, all modifications are stored in memory and lost after restart, and the system returns to its original state; When using block devices such as USB drives and TF card partitions, data can be stored persistently and will not be lost after restarting.

3. 36. 1. Configure Overlayboot to tmpfs mode

In this mode, all modifications made by the system will be stored in memory (tmpfs). After restarting, all changes will be lost and the system will return to its initial state.

1) Overlayboot is turned off by default in Linux systems and needs to be manually turned on to use. The opening steps are as follows:

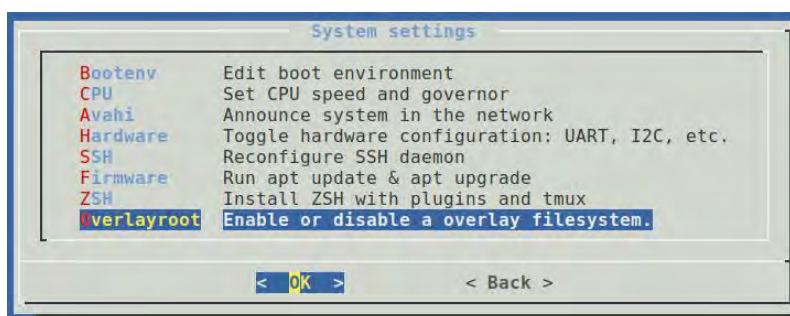
- a. Firstly, please ensure that the development board is connected to a wired or wireless network. For specific methods, please refer to the **Network Connection** Testing section
- b. Then run **orangeapi-config**, remember to add **sudo** privileges for regular users

```
orangeapi@orangepi:~$ sudo orangeapi-config
```

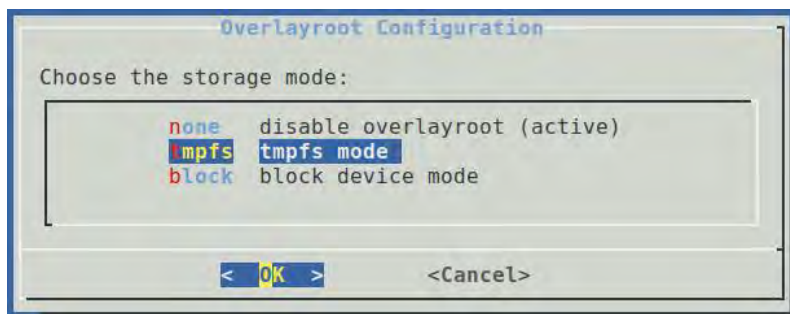
- c. Then select **System**



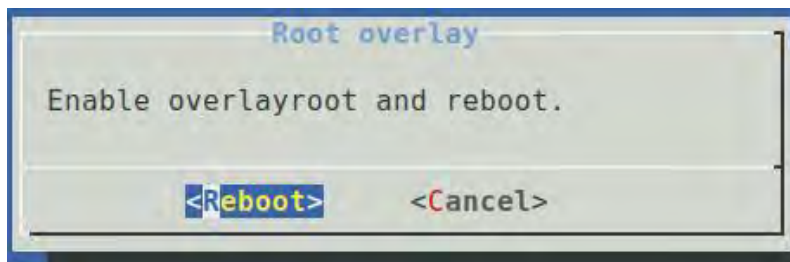
- d. Then select **Overlayroot**



- e. Then select **tmpfs** and press enter



- f. Then select **<Reboot>** to restart the system for the configuration to take effect



- 2) Then enter the **mount | grep root** command to see the mounting information of the file system.



```

orange@orange:~$ mount |grep root
/dev/mmcblk0p1 on /media/root-ro type ext4 (ro,relatime)
tmpfs-root on /media/root-rw type tmpfs (rw,relatime)
overlayroot on / type overlay (rw,relatime,lowerdir=/media/root-ro,upperdir=/media/root-rw/overlay,workdir=/media/root-rw/overlay-workdir/. )
overlayroot on /var/log.hdd type overlay (rw,relatime,lowerdir=/media/root-ro,upperdir=/media/root-rw/overlay,workdir=/media/root-rw/overlay-workdir/. )
orange@orange:~$

```

From the mounting information, it can be seen that the file system partition `/dev/mmcblk0p1` is mounted in read-only mode to `/media/root-ro`, `tmpfs` is mounted as a writable layer to `/media/root-rw`, indicating that the overlay root configuration has taken effect.

Due to `tmpfs` being a memory file system, all modifications will be lost after restarting.

3. 36. 2. Configuring Overlayboot to block device storage mode

In this mode, the writable layer uses specific storage devices (such as TF card partition `/dev/mmcblk0p2`), and data will not be lost after restarting.

Please note that after opening Overlayboot, modifications to the `/boot` partition (such as updating the kernel or modifying `/boot/orangepiEnv.txt`) will not take effect. The reason is that in block device mode, modifications are actually made to `/boot` in the overlay layer, while Uboot reads files from the real partition at startup, so the modifications will not take effect. To make the modifications effective, it is necessary to disable Overlayboot before performing these operations.

1) Overlayboot is turned off by default in Linux systems and needs to be manually turned on to use. The opening steps are as follows:

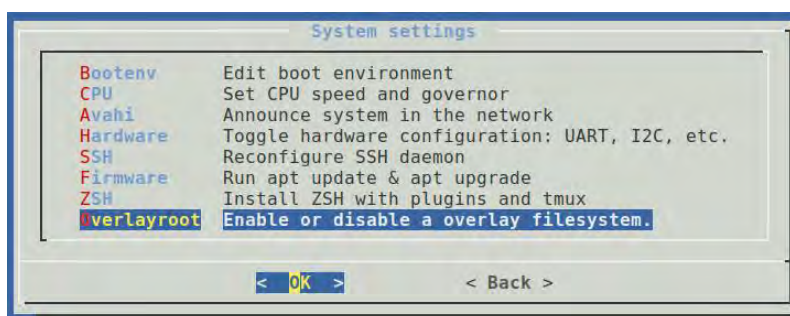
- a. Firstly, please ensure that the development board is connected to a wired or wireless network. For specific methods, please refer to the [Network Connection Testing](#) section
- a. Then run **orangepi-config**, remember to add **sudo** privileges for regular users

```
orange@orange:~$ sudo orangepi-config
```

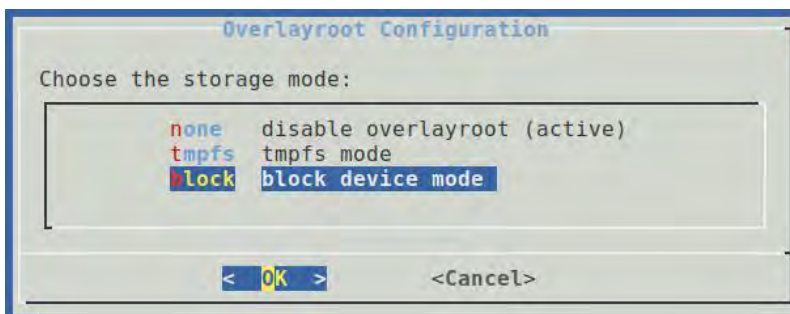
- b. Then select **System**



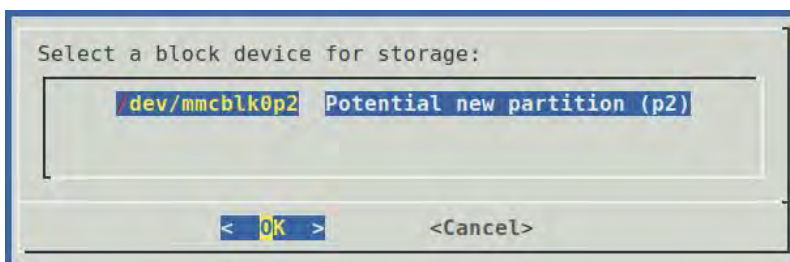
- c. Then select **Overlayroot**



- d. Then select **block** and press enter

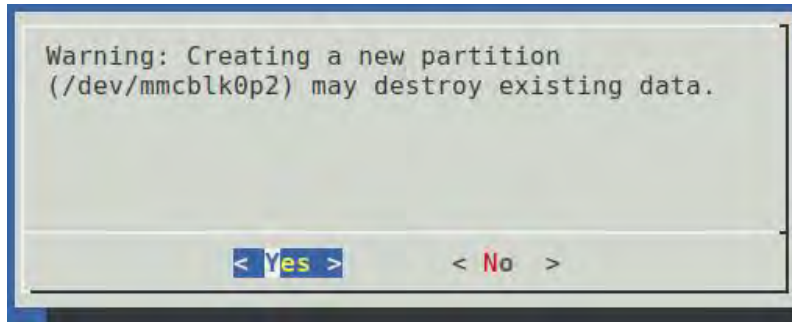


- e. Then an Overlay writable layer partition will be created. You need to select the storage device where the **Overlay writable layer** partition is located. If no other USB devices are inserted into the development board, there is only one option by default, which means creating a partition in the TF card. The size of the partition is: the total capacity of TF minus twice the capacity occupied by rootfs.

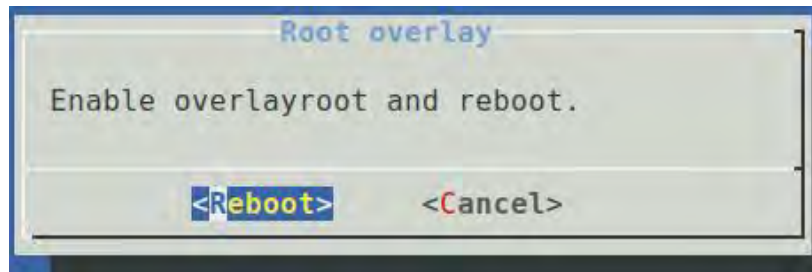




- f. Then select **<Yes>** to automatically create and format partitions



- g. Then select **<Reboot>** to restart the system for the configuration to take effect



- 2) Then enter the **mount | grep root** command to see the mounting information of the file system.

```
orangepi@orangepi:~$ mount | grep root
/dev/mmcblk0p1 on /media/root-ro type ext4 (ro,relatime)
/dev/mmcblk0p2 on /media/root-rw type ext4 (rw,relatime)
overlayroot on / type overlay (rw,relatime,lowerdir=/media/root-ro,upperdir=/media/root-rw/overlay,workdir=/media/root-rw/overlay-workdir/_ )
overlayroot on /var/log.hdd type overlay (rw,relatime,lowerdir=/media/root-ro,upperdir=/media/root-rw/overlay,workdir=/media/root-rw/overlay-workdir/_ )
orangepi@orangepi:~$
```

From the mounting information, it can be seen that the file system partition **/dev/mmcblk0p1** is mounted in read-only mode to **/media/root-ro**, **/dev/mmcblk0p2** is mounted as a writable layer to **/media/root-rw**, indicating that the overlay root configuration has taken effect.

Since **/dev/mmcblk0p2** is a partition on the TF card, data will not be lost after restarting.

- 3) Because all user modifications are redirected to the **/media/root-rw** directory, it is possible to restore factory settings by deleting all content under **/media/root-rw**/(**which requires a restart to take effect**).

```
orangepi@orangepi:~$ rm -rf /media/root-rw/*
orangepi@orangepi:~$ reboot
```

3. 36. 3. Method for disabling Overlayboot

- 1) The steps to disable Overlayboot are as follows:

- a. First, run **orangepi-config**. Regular users should remember to add **sudo**



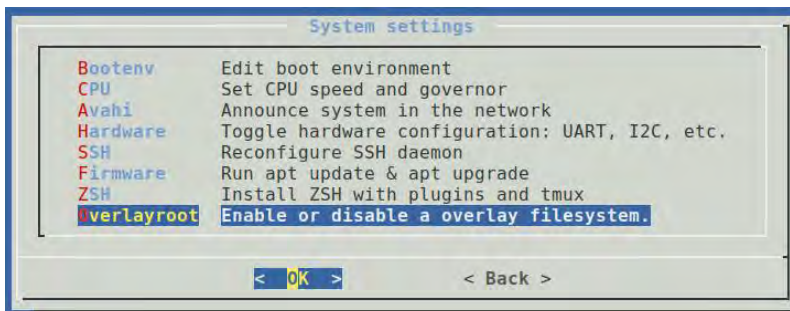
privileges

```
orangepi@orangepi:~$ sudo orangepi-config
```

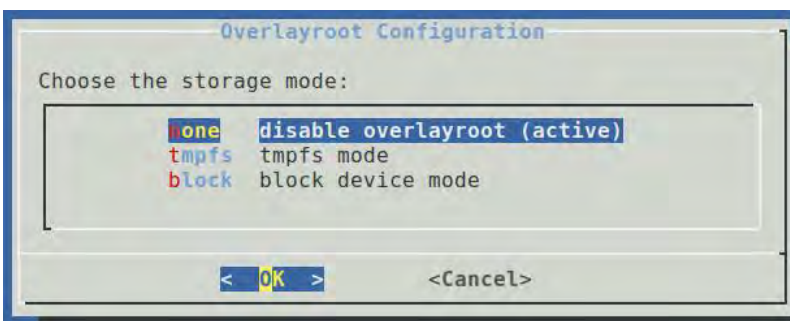
- b. Then select **System**



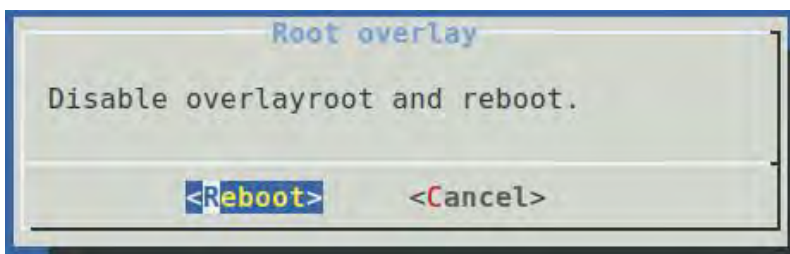
- c. Then select **Overlayroot**



- d. Then select **none**, and press enter



- e. Then select **<Reboot>** to restart the system for the configuration to take effect





2) If you find using orangepi-config cumbersome, you can also follow the steps below.

- a. Firstly, it is necessary to re mount the **root-ro** directory in writable mode.

```
orangepi@orangepi:~$ sudo mount -o remount,rw /media/root-ro
```

- b. Then modify the **/etc/overlayroot.conf** file under the read-only partition, set the overlayroot variable to empty, and restart the system.

```
orangepi@orangepi:~$ sudo vim /media/root-ro/etc/overlayroot.conf  
overlayroot=""
```

3. 37. How to shut down and restart the development board

1) If you unplug the power supply directly while the Linux system is running, some data may be lost in the file system. It is recommended to use the **poweroff** command to shut down the Linux system of the development board before unplugging the power supply.

```
orangepi@orangepi:~$ sudo poweroff
```

2) In addition, the development board is equipped with a power button, and you can also **short press** the power button on the development board to shut down.



Note that when you press the power button on the Linux desktop system, a confirmation box as shown in the figure below will pop up. You need to click the **Power Off** option before the system will shut down.



3) After shutting down, long press the power button on the development board to turn it



on.



4) Use the reboot command to restart the Linux system in the development board

```
orange@orange:~$ sudo reboot
```

4. Linux SDK——orange-build usage instructions

4.1. Compilation system requirements

Linux SDK, **orange-build**, only supports running on X64 computers with Ubuntu 22.04 installed, so before downloading orange-build, please first make sure that the Ubuntu version installed on your computer is **Ubuntu 22.04**. The command to check the Ubuntu version installed on the computer is as follows. If the Release field does not display **22.04**, it means that the current Ubuntu version does not meet the requirements. Please change the system before performing the following operations.

```
test@test:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04 LTS
Release:        22.04
Codename:       jammy
```



If your computer is running Windows and does not have Ubuntu 22.04 installed, you can consider using **VirtualBox** or **VMware** to install an Ubuntu 22.04 virtual machine in Windows. But please note that you should not compile orangepi-build on a WSL virtual machine, because orangepi-build has not been tested in a WSL virtual machine, so you cannot ensure that orangepi-build can be used normally in WSL. In addition, please do not use orangepi-build in the Linux system of the development board. The installation image download address of Ubuntu 22.04 **amd64** version is:

<https://mirrors.tuna.tsinghua.edu.cn/ubuntu-releases/22.04/ubuntu-22.04-desktop-amd64.iso>

After installing Ubuntu 22.04 on your computer or in a virtual machine, please first set the software source of Ubuntu 22.04 to Tsinghua source (or other domestic sources that you think are fast), otherwise it is easy to make mistakes when installing software later due to network reasons. The steps to replace Tsinghua source are as follows:

- a. To replace Tsinghua source, please refer to the instructions on this page

<https://mirrors.tuna.tsinghua.edu.cn/help/ubuntu/>

- b. Note that the 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 content of the `/etc/apt/sources.list` file that needs to be replaced is:

```
test@test:~$ sudo mv /etc/apt/sources.list cat /etc/apt/sources.list.bak
test@test:~$ sudo vim /etc/apt/sources.list

#The source mirror is commented out by default to increase the speed of apt update. You can uncomment it if
necessary.

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 source, not recommended to enable
# 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 the replacement, you need to update the package information and ensure that there is no error

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

- e. **In addition, since the source code of the kernel and U-boot are stored on GitHub, it is very important to make sure that the computer can download the code from GitHub normally when compiling the image.**

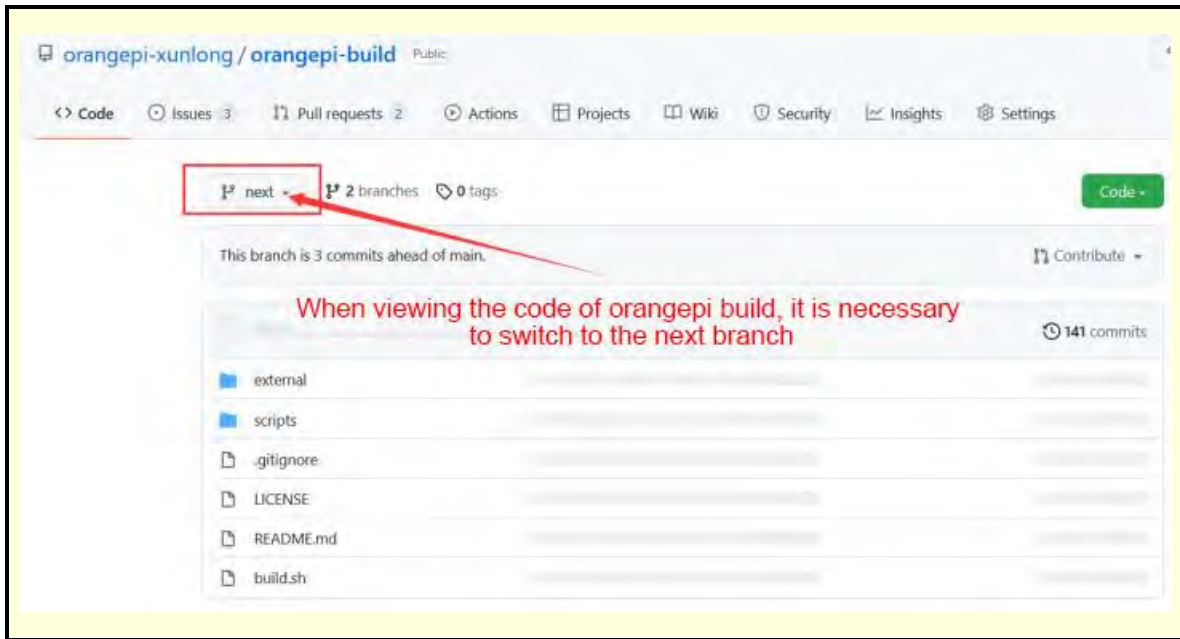
4. 2. Get the source code of Linux SDK

4. 2. 1. Download orangepi-build from github

Linux SDK refers to the orangepi-build code. Orangepi-build is modified based on the armbian build compilation system. Orangepi-build can be used to compile multiple versions of Linux images. Use the following command to download the orangepi-build code:

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

Note that to use the T527 Soc development board, you need to download the **next branch source code of orangepi-build. The above git clone command needs to specify the branch of orangepi-build source code as next.**



You do not need to enter the username and password of the GitHub account when downloading the orangepi-build code through the git clone command (the same applies to downloading other codes in this manual). If the Ubuntu PC prompts you to enter the username and password of the GitHub account after entering the git clone command, it is usually because the address of the orangepi-build warehouse after git clone is entered incorrectly. Please check the command spelling carefully for any errors, instead of thinking that we forgot to provide the username and password of the GitHub account here.

The u-boot and linux kernel versions currently used by the T527 series development board are as follows:

Branches	u-boot Version	Linux Kernel version
current	u-boot v2018.05	linux5.15

The branch mentioned here is not the same as the branch of orangepi-build source code, please do not confuse them. This branch is mainly used to distinguish different kernel source code versions.

The Linux 5.15 BSP kernel currently provided by Allwinner is defined as the current branch.

orangepi-build After downloading, the following files and folders will be included:



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

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

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

4. 2. 2. Download the cross-compilation toolchain

When orangepi-build is run for the first time, it will automatically download the cross-compilation toolchain and put it in the **toolchains** folder. Each time you run the build.sh script of orangepi-build, it will check whether the cross-compilation toolchain in **toolchains** exists. If not, it will restart the download. If it exists, it will be used directly without repeated download.



```

[ o.k. ] Checking for external GCC compilers
[ ... ] downloading using http(s) network [ gcc-linaro-aarch64-none-elf-4.8-2013.11_linux.tar.xz ]
#8d7029 16MiB/24MiB(65%) CN:1 DL:7.9MiB ETA:1s
[ o.k. ] Verified [ PGP ]
[ ... ] decompressing
[ ... ] gcc-linaro-aarch64-none-elf-4.8-2013.11_linux.tar.xz: 24.9MiB [14.4MiB/s] [=====] 100%
[ ... ] downloading using http(s) network [ gcc-linaro-arm-none-eabi-4.8-2014.04_linux.tar.xz ]
#e38eec 17MiB/33MiB(50%) CN:1 DL:16MiB ETA:1s
[ o.k. ] Verified [ PGP ]
[ ... ] decompressing
[ ... ] gcc-linaro-arm-none-eabi-4.8-2014.04_linux.tar.xz: 33.9MiB [9.66MiB/s] [=====] 100%
[ ... ] downloading using http(s) network [ gcc-linaro-arm-linux-gnueabi-4.8-2014.04_linux.tar.xz ]
#041c24 48MiB/48MiB(99%) CN:1 DL:2.7MiB
[ o.k. ] Verified [ PGP ]
[ ... ] decompressing
[ ... ] gcc-linaro-arm-linux-gnueabi-4.8-2014.04_linux.tar.xz: 48.8MiB [13.0MiB/s] [=====] 100%
[ ... ] downloading using http(s) network [ gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabi.tar.xz ]
#3dec3e 72MiB/76MiB(93%) CN:1 DL:3.7MiB ETA:1s
[ o.k. ] Verified [ MD5 ]
[ ... ] decompressing
[ ... ] gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabi.tar.xz: 77.0MiB [14.2MiB/s] [=====] 100%
[ ... ] downloading using http(s) network [ gcc-linaro-7.4.1-2019.02-x86_64_arm-linux-gnueabi.tar.xz ]
#42e728 104MiB/104MiB(99%) CN:1 DL:2.0MiB
[ o.k. ] Verified [ MD5 ]
[ ... ] decompressing
[ ... ] gcc-linaro-7.4.1-2019.02-x86_64_arm-linux-gnueabi.tar.xz: 104MiB [13.9MiB/s] [=====] 100%
[ ... ] downloading using http(s) network [ gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu.tar.xz ]
#2c065e 108MiB/111MiB(97%) CN:1 DL:3.9MiB
[ o.k. ] Verified [ MD5 ]
[ ... ] decompressing
[ ... ] gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu.tar.xz: 111MiB [13.4MiB/s] [=====] 100%
[ ... ] downloading using http(s) network [ gcc-arm-9.2-2019.12-x86_64-arm-none-linux-gnueabi.tar.xz ]
#d232ee 259MiB/251MiB(99%) CN:1 DL:2.0MiB
[ o.k. ] Verified [ MD5 ]
[ ... ] decompressing
[ ... ] gcc-arm-9.2-2019.12-x86_64-arm-none-linux-gnueabi.tar.xz: 251MiB [13.7MiB/s] [=====] 100%
[ ... ] downloading using http(s) network [ gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu.tar.xz ]
#88b441 268MiB/269MiB(99%) CN:1 DL:0.9MiB
[ o.k. ] Verified [ MD5 ]
[ ... ] decompressing

```

The mirror website of the cross-compilation tool chain in China is the open source software mirror website of Tsinghua University:

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

Toolchains After downloading, it will contain multiple versions of cross-compilation tool chains:

```

test@test:~/orange-pi-build$ ls toolchains/
gcc-arm-11.2-2022.02-x86_64-aarch64-none-linux-gnu
gcc-linaro-4.9.4-2017.01-x86_64_aarch64-linux-gnu
gcc-linaro-7.4.1-2019.02-x86_64_arm-linux-gnueabi
gcc-arm-11.2-2022.02-x86_64-arm-none-linux-gnueabi
gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabi
gcc-linaro-aarch64-none-elf-4.8-2013.11_linux
gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu
gcc-linaro-5.5.0-2017.10-x86_64_arm-linux-gnueabi
gcc-linaro-arm-linux-gnueabi-4.8-2014.04_linux
gcc-arm-9.2-2019.12-x86_64-arm-none-linux-gnueabi
gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu
gcc-linaro-arm-none-eabi-4.8-2014.04_linux

```

The cross-compilation tool chain used to compile the T527 Linux kernel source code is:

- a. linux5.15



```
gcc-arm-11.2-2022.02-x86_64-aarch64-none-linux-gnu
```

The cross-compilation toolchain used to compile the T527 u-boot source code is:

- a. v2018.05

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

4. 2. 3. Explanation of the complete directory structure of orangepi build

1) After downloading the orangepi build repository, it does not include the Linux kernel, U-boot source code, or cross compilation toolchain. The Linux kernel and U-boot source code are stored in separate Git repositories

- a. The git repository where the Linux kernel source code is stored is as follows. Please note to switch the branch of the Linux orangepi repository to

```
https://github.com/orangepi-xunlong/linux-orangepi/tree/orange-pi-5.15-sun55iw3
```

- b. The git repository where the u-boot source code is stored is as follows. Please note to switch the branch of the u-boot orangepi repository to

```
https://github.com/orangepi-xunlong/u-boot-orangepi/tree/v2018.05-t527
```

2) When Orangepi build is first run, it will download the cross compilation toolchain, u-boot, and Linux kernel source code. After successfully compiling the Linux image once, the files and folders that can be seen in Orangepi 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**: Store the source code of the Linux kernel
- d. **LICENSE**: GPL 2 License File
- e. **README.md**: Orangepi build documentation
- f. **output**: Store compiled deb packages such as u-boot and Linux, compilation logs, and compiled images
- g. **scripts**: General script for compiling Linux images
- h. **toolchains**: Store cross compilation toolchain
- i. **u-boot**: Store the source code of u-boot
- j. **userpatches**: Store the configuration files required for compiling scripts

```
test@test:~/orangepi-build$ ls
```



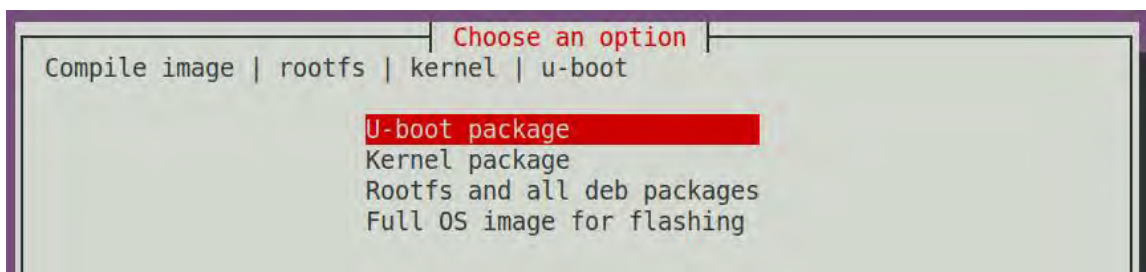
```
build.sh  external  kernel  LICENSE  output  README.md  scripts  toolchains
u-boot   userpatches
```

4. 3. Compiling u-boot

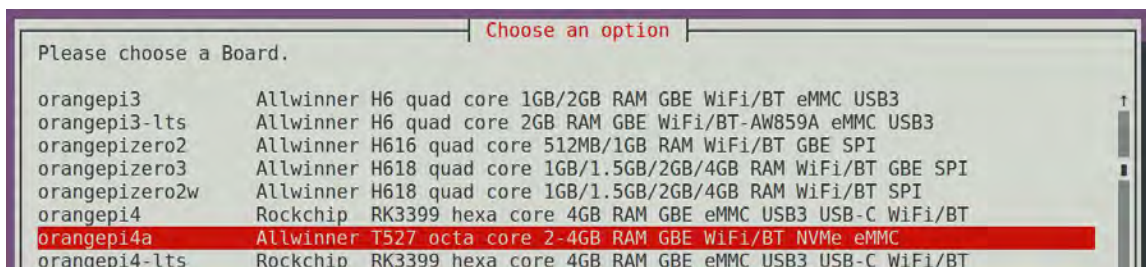
1) Run the build.sh script, remember to grant sudo privileges

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

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



3) Next, select the model of the development board



4) Then it will start compiling u-boot, and some of the information prompted when compiling the current branch is as follows:

a. Version of u-boot source code

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

b. Version of cross compilation toolchain

```
[ o.k. ] Compiler version [ aarch64-linux-gnu-gcc 11 ]
```

c. The path of the compiled u-boot deb package

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

d. The package name of the compiled u-boot deb package

```
[ o.k. ] File name [ linux-u-boot-current-orange-pi4a_x.x.x_arm64.deb ]
```

e. Compilation time used



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

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

```
[ o.k. ] Repeat Build Options [ sudo ./build.sh BOARD=orangepi4a  
BRANCH=current BUILD_OPT=u-boot ]
```

5) View the compiled u-boot deb package

```
test@test:~/orange-pi-build$ ls output/debs/u-boot/  
linux-u-boot-current-orangepi4a_x.x.x_arm64.deb
```

6) When the orange-pi build compilation system compiles the U-boot source code, it first synchronizes the U-boot source code with the GitHub server's U-boot source code. Therefore, if you want to modify the U-boot source code, you first need to turn off the download and update function of the source code (**you need to compile the U-boot completely before turning off this function, otherwise you will prompt that the U-boot source code cannot be found**). Otherwise, the modifications made will be restored. The method is as follows:

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

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

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

- a. Firstly, upload the compiled deb package of u-boot to the Linux system on the development board

```
test@test:~/orange-pi-build$ cd output/debs/u-boot  
test@test:~/orange-pi-build/output/debs/u-boot$ scp \  
linux-u-boot-current-orangepi4a_x.x.x_arm64.deb root@192.168.1.xxx:/root
```

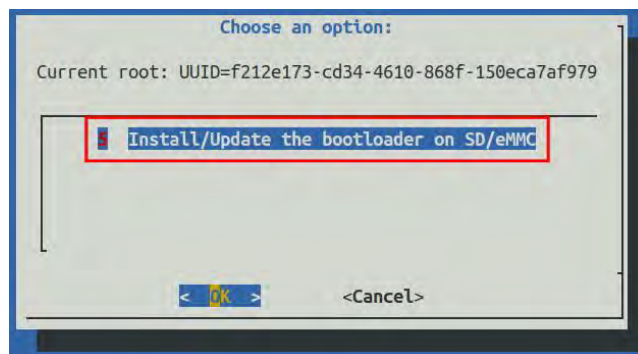
- b. Reinstall the newly uploaded deb package for u-boot

```
orange-pi@orange-pi:~$ sudo dpkg -i linux-u-boot-current-orangepi4a_x.x.x_arm64.deb
```

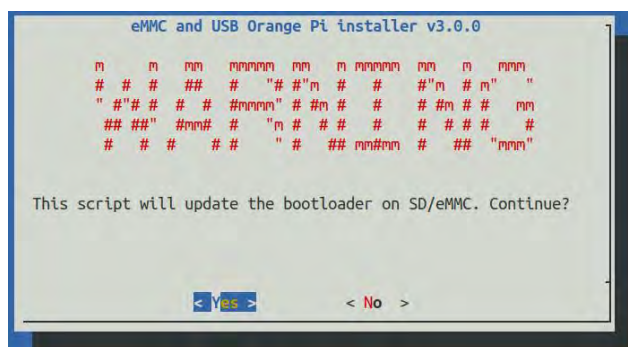
- c. Then run the nand sata install script

```
orange-pi@orange-pi:~$ sudo nand-sata-install
```

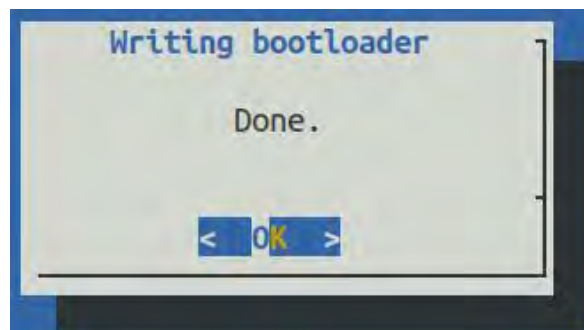
- d. Then select **5 Install/Update the bootloader on SD/eMMC**



- e. After pressing the enter key, a warning will first pop up



- f. Pressing the enter key again will start updating u-boot, and after the update is complete, the following information will be displayed



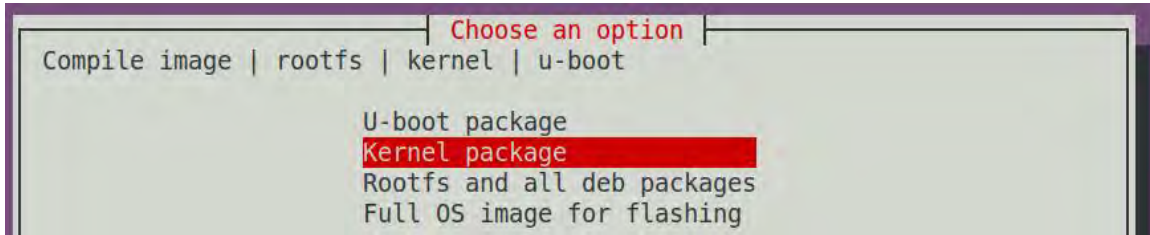
- g. Then you can restart the development board to test whether the u-boot modifications have taken effect

4.4. Compiling Linux Kernel

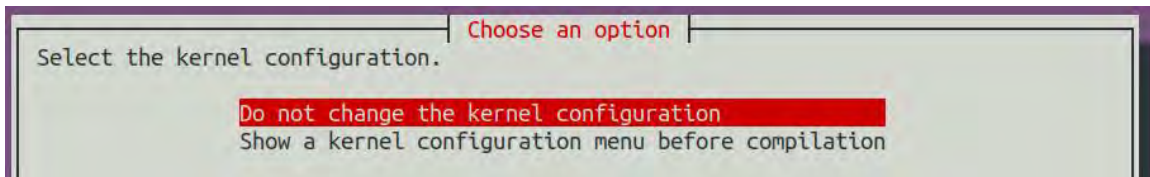
- 1) Run the **build.sh** script, remember to grant sudo privileges

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

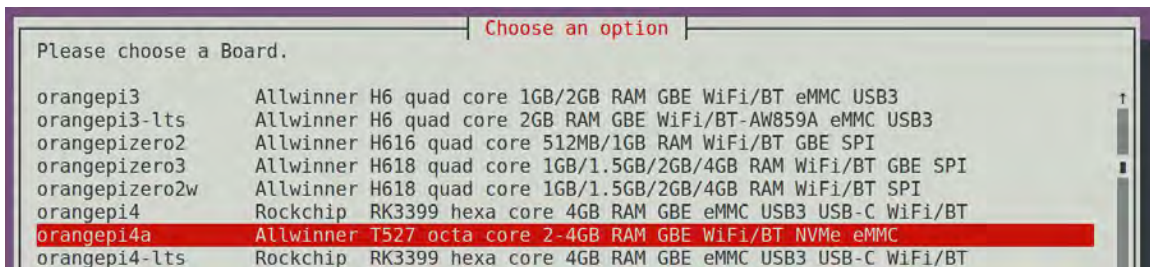
- 2) Select **Kernel package** and press Enter



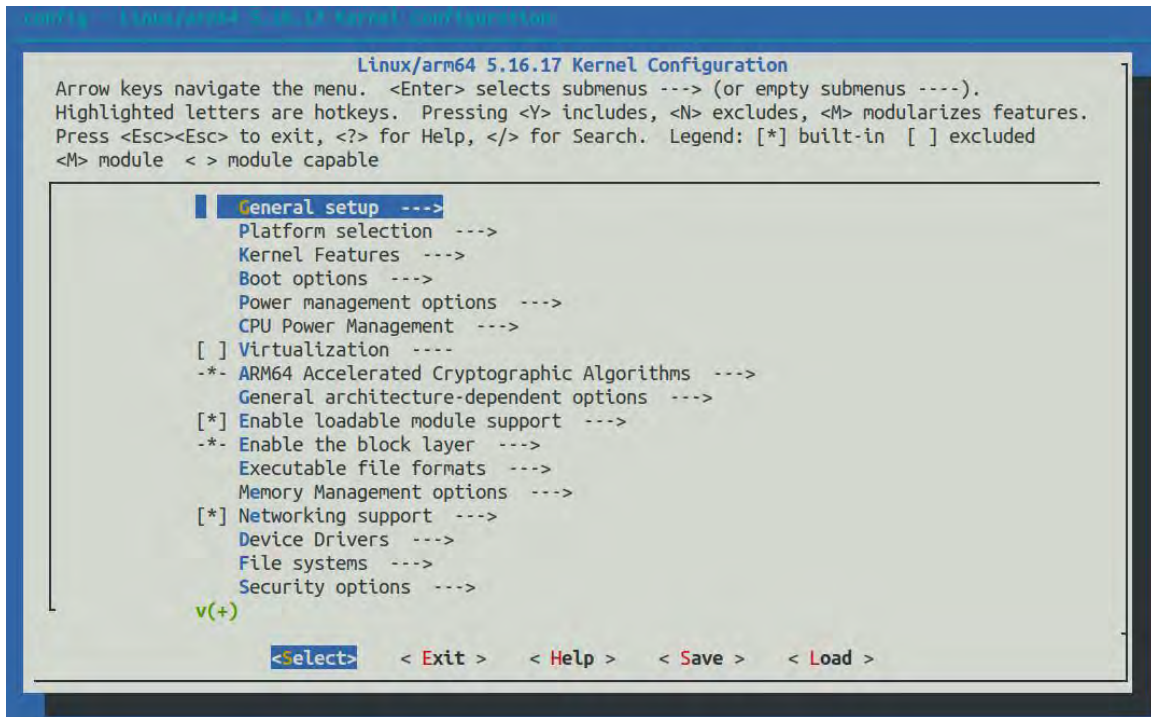
3) Then it will prompt whether the kernel configuration interface needs to be displayed. If the kernel configuration does not need to be modified, select the first one. If the kernel configuration needs to be modified, select the second one



4) Next, select the model of the development board



5) If step 3) 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` default.exe 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 terminal of Ubuntu PC to its maximum size and 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
```



6) The following is a partial explanation of the information prompted when compiling the current branch kernel source code:

a. Version of Linux kernel source code

```
[ o.k. ] Compiling current kernel [ 5.15.147 ]
```

b. The version of the cross compilation toolchain used

```
[ o.k. ] Compiler version [ aarch64-linux-gnu-gcc 11 ]
```

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

```
[ o.k. ] Using kernel config file
```

```
[ orangepi-build/external/config/kernel/linux-5.15-sun55iw3-current.config ]
```

d. The path of the compiled kernel related deb package

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

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

```
[ o.k. ] File name [ linux-image-current-sun55iw3_x.x.x_arm64.deb ]
```

f. Compilation time used

```
[ o.k. ] Runtime [ 10 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=orangepi4a  
BRANCH=current BUILD_OPT=kernel KERNEL_CONFIGURE=no ]
```

7) View the compiled kernel related deb packages

a. **linux-dtb-current-sun55iw3_x.x.x_arm64.deb** Contains dtb files used by the kernel

b. **linux-headers-current-sun55iw3_x.x.x_arm64.deb** Contains kernel header files

c. **linux-image-current-sun55iw3_x.x.x_arm64.deb** Contains kernel images and kernel modules

```
test@test:~/orangepi-build$ ls output/debs/linux-*  
output/debs/linux-dtb-current-sun55iw3_x.x.x_arm64.deb  
output/debs/linux-headers-current-sun55iw3_x.x.x_arm64.deb  
output/debs/linux-image-current-sun55iw3_x.x.x_arm64.deb
```



8) When the OrangePi 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 (**which requires a complete compilation of the Linux kernel source code before it can be turned off, otherwise it will prompt that the Linux kernel source code cannot be found**). Otherwise, the modifications made will be restored. The method is as follows:

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

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

9) If the kernel has been modified, you can use the following methods to update the kernel and kernel modules of the development board linux system

- a. Upload the compiled deb package of the linux kernel to the linux system on the development board

```
test@test:~/orange-pi-build$ cd output/debs
test@test:~/orange-pi-build/output/debs$ scp \
linux-image-current-sun55iw3_x.x.x_arm64.deb root@192.168.1.xxx:/root
```

- b. Install the new linux kernel deb package that you just uploaded

```
orange-pi@orange-pi:~$ sudo dpkg -i linux-image-current-sun55iw3_x.x.x_arm64.deb
```

- c. Restart the development board and check whether the kernel changes have taken effect

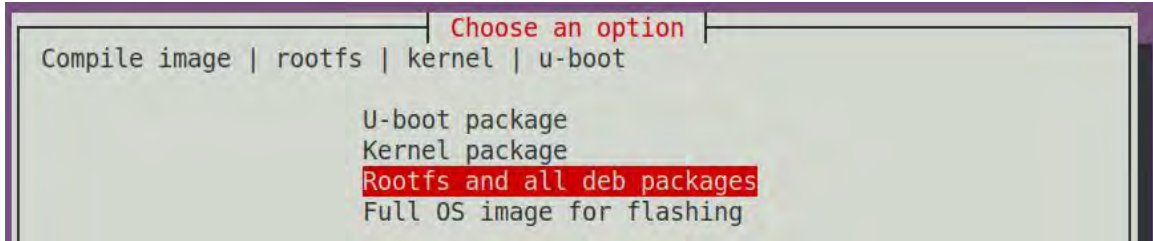
```
orange-pi@orange-pi:~$ sudo reboot
```

4. 5. Compile rootfs

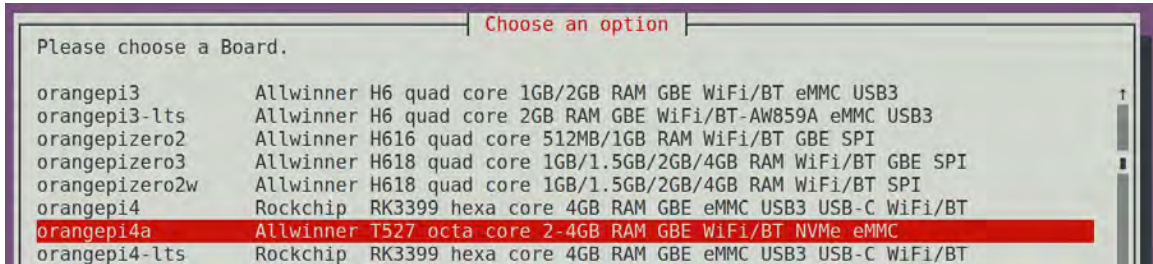
- 1) Run the build.sh script and remember to add sudo permissions

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

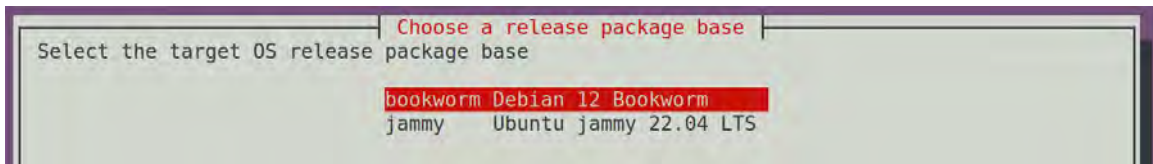
- 2) Select **Rootfs and all deb packages**, then press enter



3) Then select the model of the development board

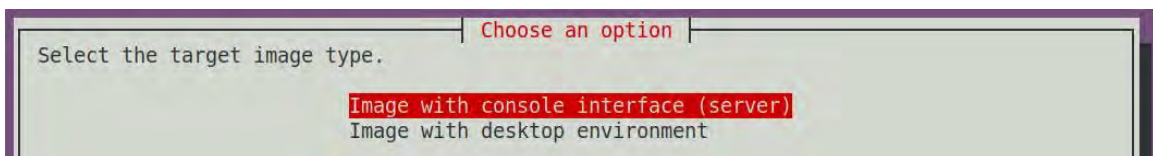


4) Then select the type of rootfs

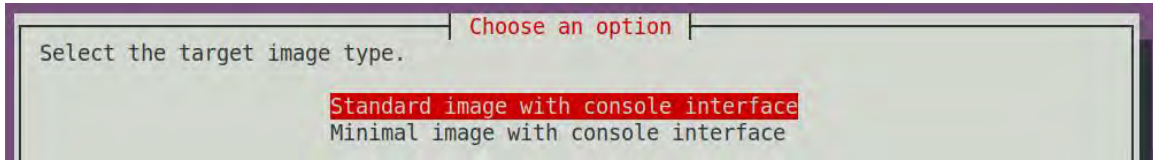


5) Then select the type of image

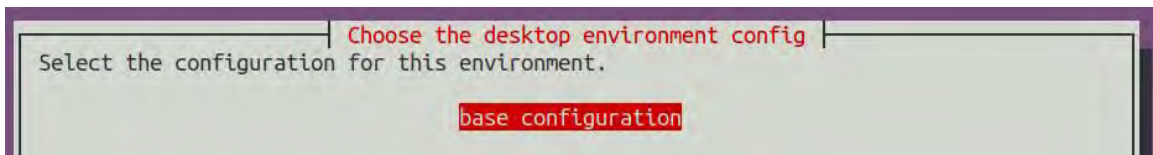
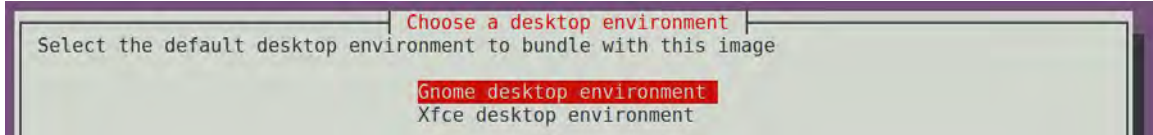
- a. **Image with console interface (server)** Indicates the server version of the image, the volume is relatively small
- b. **Image with desktop environment** It is a large image with a desktop



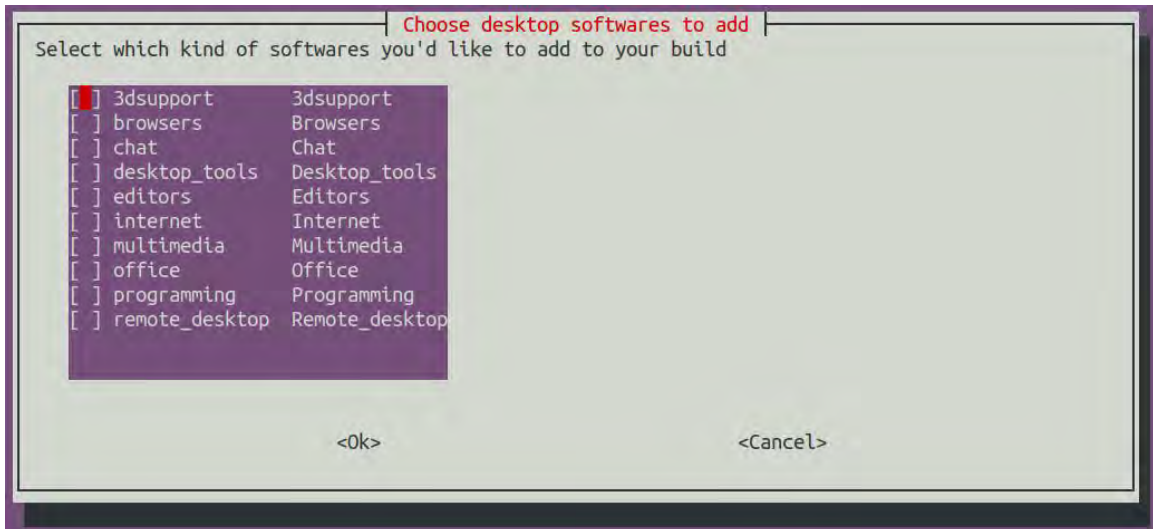
6) If it is the image of the compile server version, you can also choose to compile the Standard version or Minimal version, the Minimal version of the pre-installed software will be much less than the Standard version (**no special requirements, please do not choose the Minimal version, because many things are not pre-installed by default, some functions may not be used**)



7) If you compile the desktop version of the image also need to select the type of desktop environment, currently only maintain GNOME, so please choose GNOME type desktop



You can then select additional packages that need to be installed. Please press Enter to skip this.



8) You will then start compiling the rootfs. Some of the information prompted during compilation is described below

a. The type of rootfs

```
[ o.k. ] local not found [ Creating new rootfs cache for jammy ]
```

b. Directory for storing the generated rootfs package



[o.k.] Target directory [**orange-pi-build/external/cache/rootfs**]

c. Name of the generated rootfs package

[o.k.] File name [**jammy-gnome-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4**]

9) View the compiled rootfs compressed package

- a. **jammy-gnome-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4** is rootfs zip, the meaning of each field name is
 - a) **jammy** indicates the type of linux distribution for rootfs
 - b) **gnome** indicates that rootfs is the type of the desktop version. If it is **cli**, it indicates the type of the server version
 - c) **arm64** indicates the schema type of rootfs
 - d) **5250ec7002de9e81a41de169f1f89721** is generated by the package name rootfs to install all of the packages of the MD5 hash value, as long as no modification rootfs installation package list, then this value will not change, The compilation script uses this MD5 hash to determine if the rootfs needs to be recompiled
- b. **jammy-gnome-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4.list** lists the package names of all packages installed by rootfs

```
test@test:~/orange-pi-build$ ls external/cache/rootfs/
jammy-gnome-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4
jammy-gnome-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4.current
jammy-gnome-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4.list
```

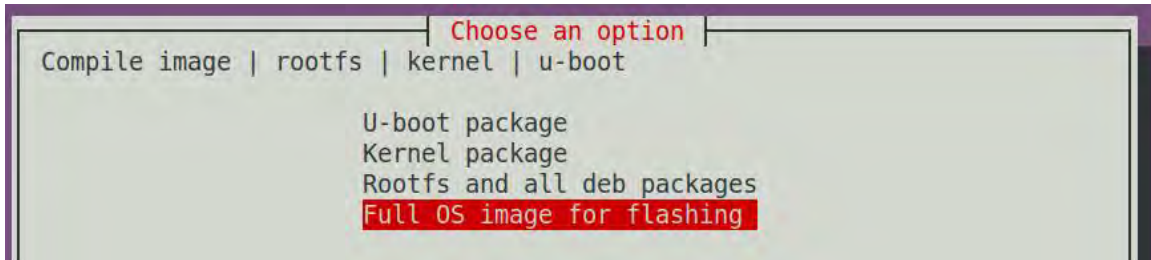
10) If the required rootfs already exists under **external/cache/rootfs**, compiling the rootfs again will directly skip the compilation process and will not restart the compilation. When compiling the image, it will also go to the **external/cache/rootfs** to check whether there is a cached rootfs. If there is one, it will be used directly, which can save a lot of download and compilation time

4. 6. Compile the linux image

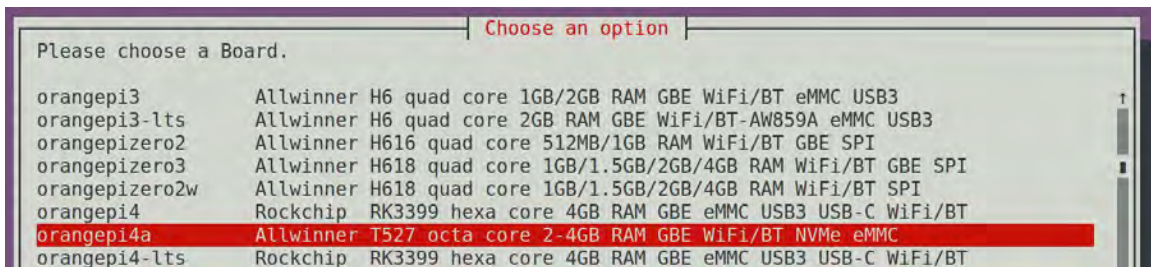
- 1) Run the **build.sh** script and remember to add sudo permissions

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

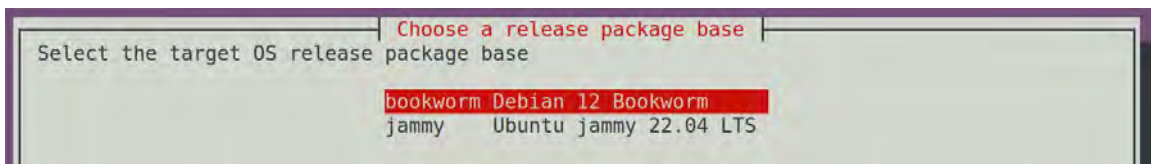
- 2) Select **Full OS image for flashing** and press Enter



3) Then select the model of the development board

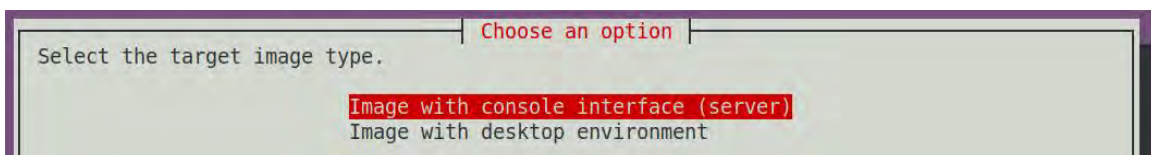


4) Then select the type of rootfs

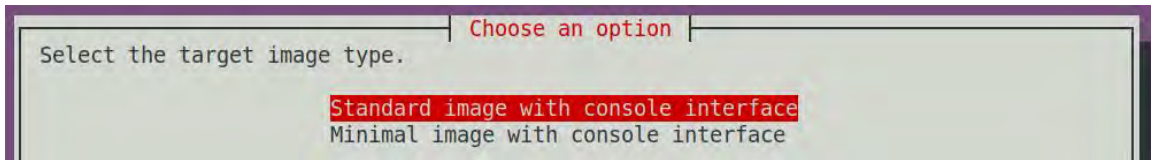


5) Then select the type of image

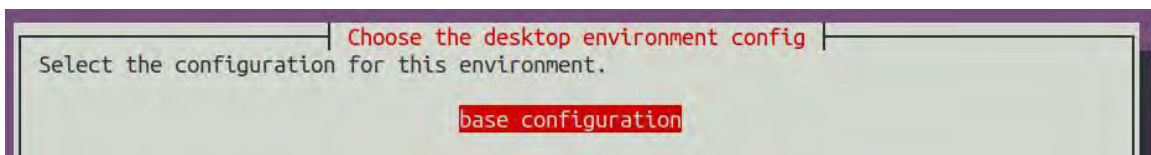
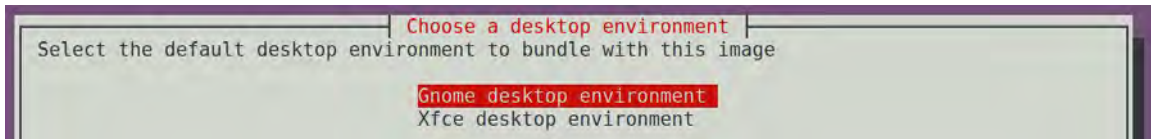
- a. **Image with console interface (server)** Indicates the server version of the image, the volume is relatively small
- b. **Image with desktop environment** It is a large image with a desktop



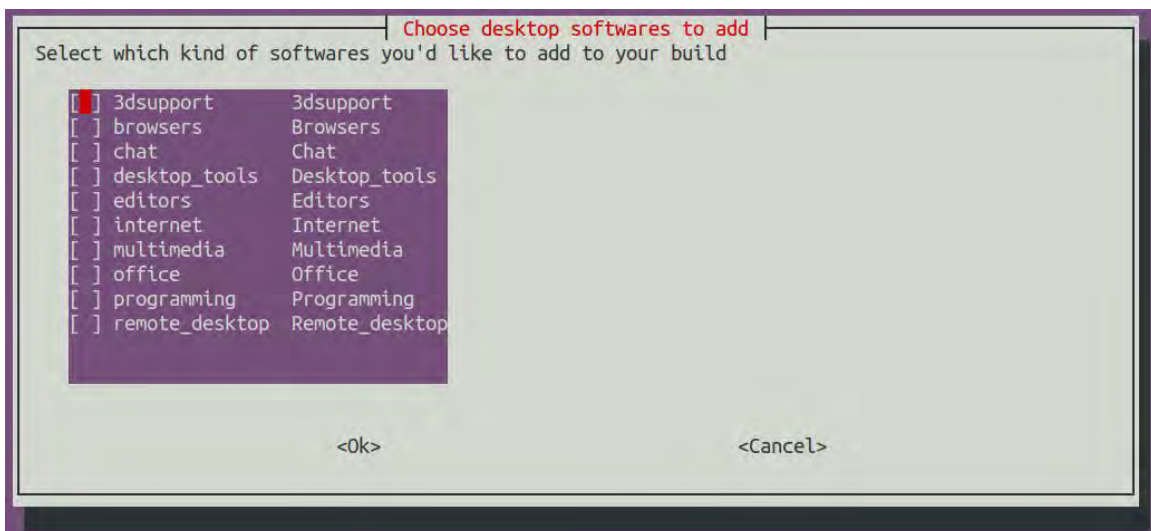
6) If it is the image of the compile server version, you can also choose to compile the Standard version or Minimal version, the Minimal version of the pre-installed software will be much less than the Standard version (**no special requirements, please do not choose the Minimal version, because many things are not pre-installed by default, some functions may not be used**)



7) If you compile the desktop version of the image also need to select the type of desktop environment, currently only maintain GNOME, so please choose GNOME type desktop



You can then select additional packages that need to be installed. Please press Enter to skip this.



8) The linux image will then be compiled, the general process of compilation is as follows

- a. Initialize the Ubuntu PC compilation environment and install the software packages required for the compilation process
- b. Download the u-boot and linux kernel source code (if cached, update the code only)



- c. Compile u-boot source code to generate the u-boot deb package
- d. Compile linux source code and generate Linux-related deb packages
- e. Create the deb package of the linux firmware
- f. Create the deb package of the orangepi-config tool
- g. Create a board-level deb package
- h. If you are compiling the desktop version image, you will also make desktop related deb packages
- i. Check whether the rootfs is cached. If no, create a new rootfs. If the rootFS is cached, decompress it and use it
- j. Install the previously generated deb package into rootfs
- k. Perform some specific Settings for different development boards and different types of images, such as preinstalling additional software packages and modifying system configurations
- l. Then create an image file and format the partition. The default type is ext4
- m. Copy the configured rootfs to the image partition
- n. Then update initramfs
- o. Run the dd command to write the bin file of the u-boot to the image

9) The following information is displayed after the image is compiled

- a. Path for storing the generated image

```
[ o.k. ] Done building
[ output/images/orangepi4a_x.x.x_debian_jammy_linux5.15.xx_gnome_desktop/orangepi4a_x.x.x_debian_jammy_linux5.15.xx_gnome_desktop.img ]
```

- b. Compilation time

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

- c. Repeat the image compilation command. Run the following command to compile the image without using the GUI

```
[ o.k. ] Repeat Build Options [ sudo ./build.sh BOARD=orangepi4a
BRANCH=current BUILD_OPT=image RELEASE=jammy BUILD_MINIMAL=no
BUILD_DESKTOP=no KERNEL_CONFIGURE=yes ]
```



5. Android 13 operating system instructions

5.1. Supported Android versions

Android version	Kernel version
Android 13	linux5.15

5.2. Android 13 Function Adaptation

Function	Android 13
HDMI video	OK
HDMI Audio	OK
USB2.0 x 4	OK
TF card start	OK
eMMC	OK
Identify NVME SSDS	OK
Gigabit network card	OK
WIFI	OK
Bluetooth	OK
RTC chip	OK
Headphone audio	OK
LCD screen	OK
EDP	OK
CAM1	NO
CAM2	NO
LED light	OK
40 pin GPIO	OK
40 pin I2C	OK
40 pin SPI	OK
40 pin UART	OK



40 pin PWM	OK
Temperature sensor	OK
Mali GPU	OK
Video codec	OK

5.3. Usage of ADB

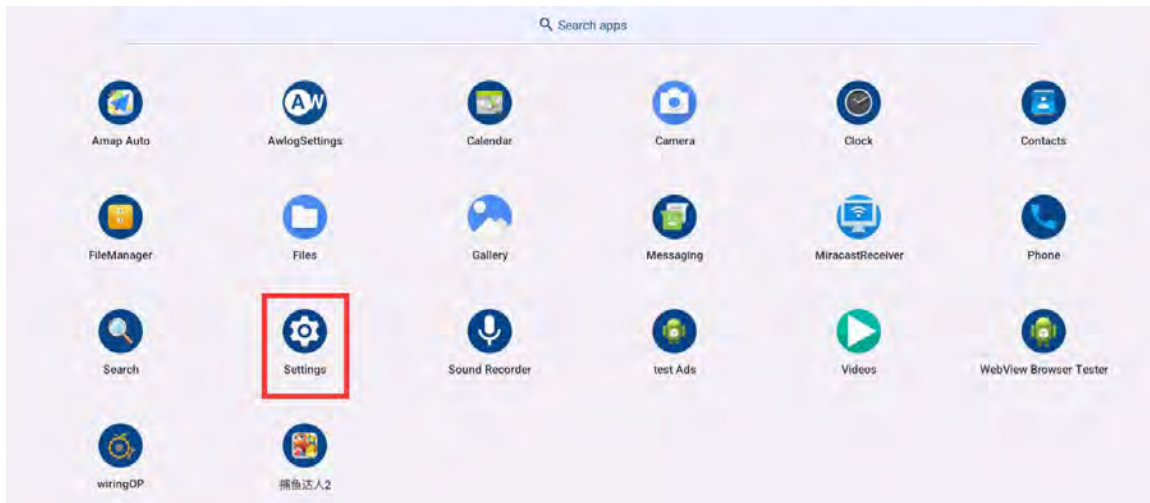
5.3.1. USB OTG mode switching method

The development board has four USB ports, of which the USB port marked in the red box below can support both Host mode and Device mode, and the other three USB ports only support Host mode.

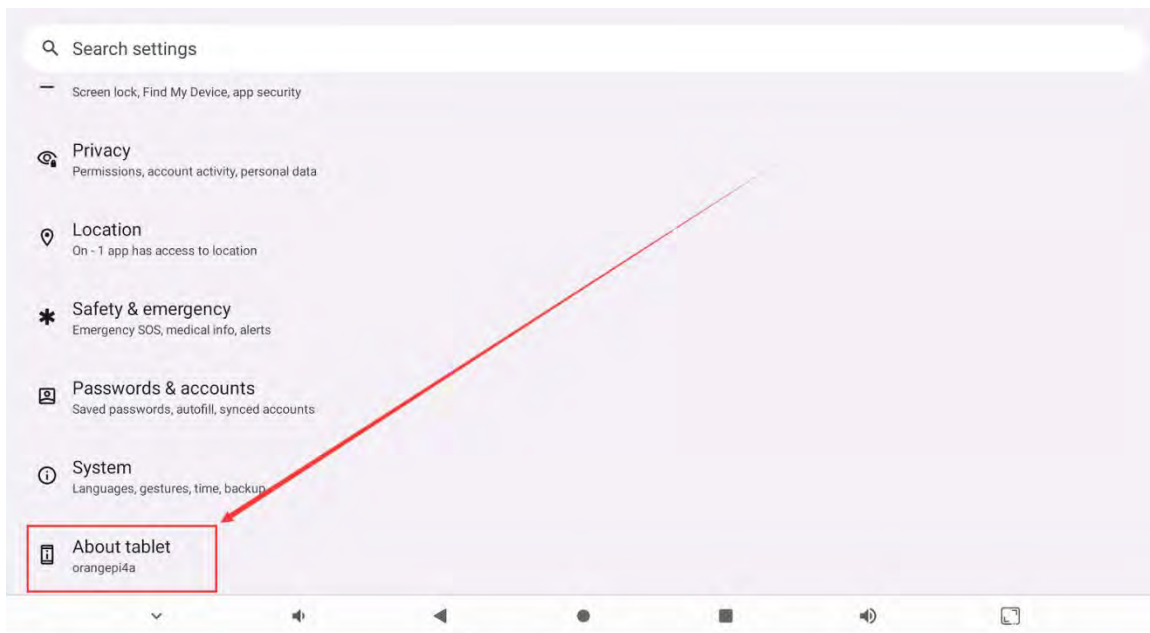


The USB OTG interface is in Host mode by default and can be used to connect USB devices such as mouse and keyboard. If you want to use ADB, you need to **manually** switch to Device mode.

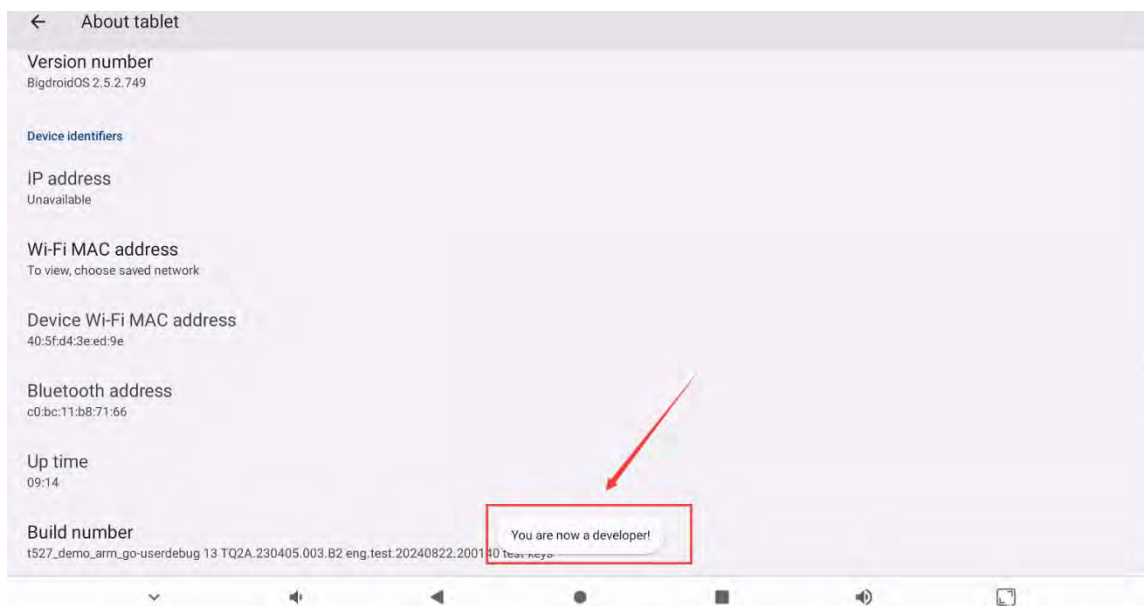
- 1) First open Settings



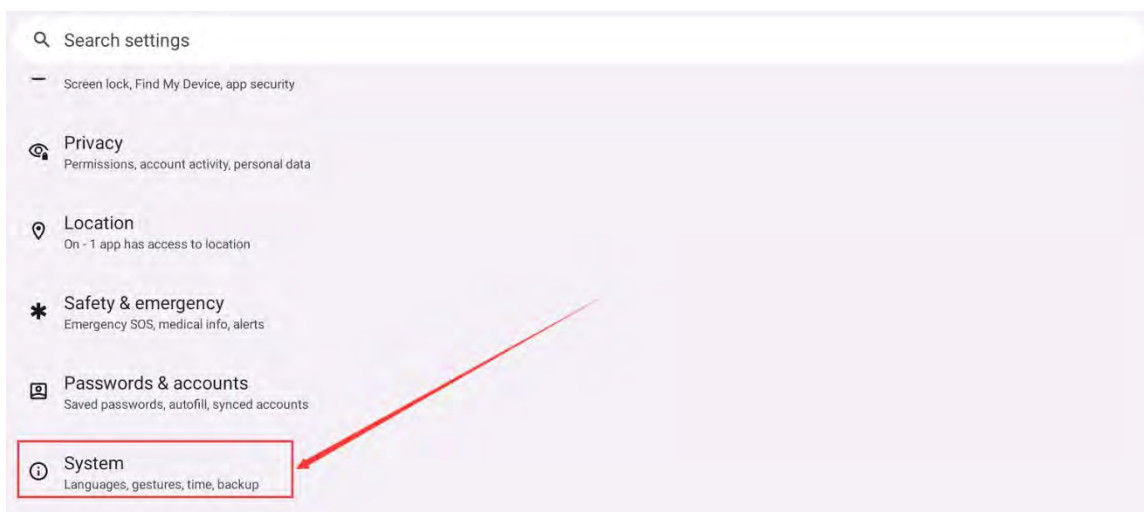
2) Then find **About tablet**



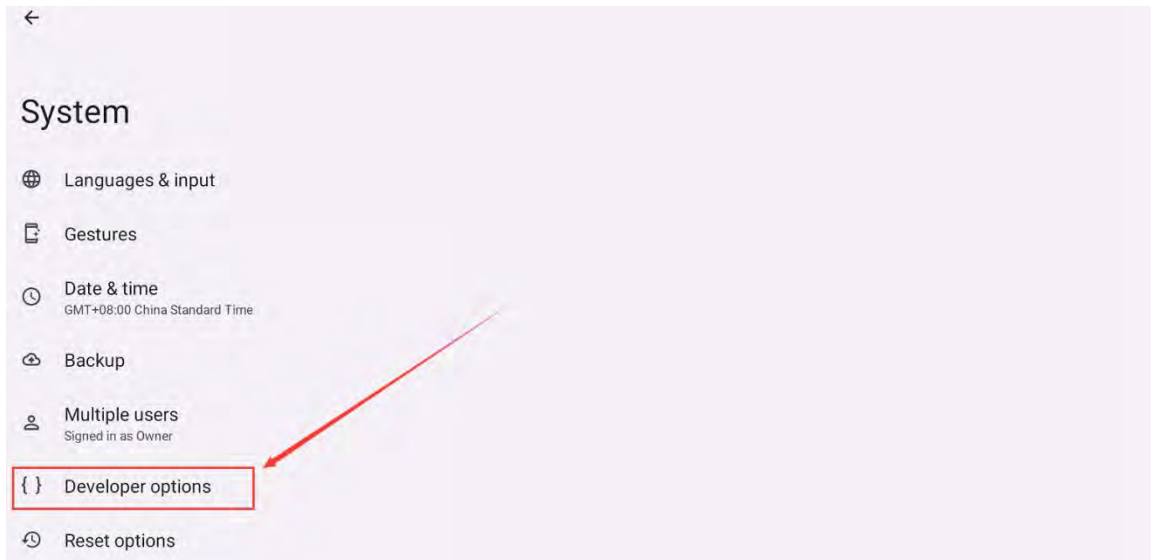
3) Then click the **Build number** option several times with your mouse until it appears. **You are now a developer!** Tips for



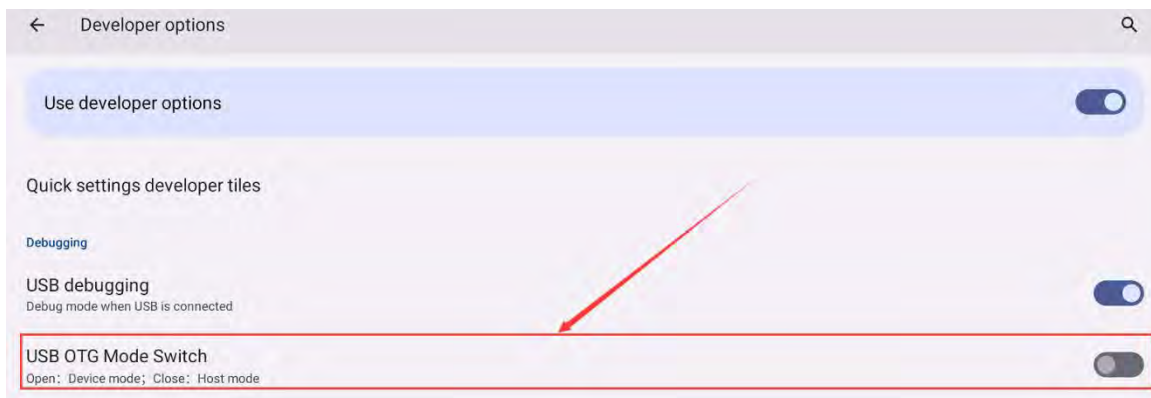
4) Then return to the upper-level menu and select **System**



5) Then select **Developer options**



6) Finally, find the **USB OTG Mode Switch**, **turn on the switch to switch to Device mode, turn off the switch to switch to Host mode**



5. 3. 2. Use a data cable to connect adb to debug

1) First prepare a good quality USB 2.0 public to public data cable



2) Install adb tool on Ubuntu PC

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



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

3) View the identified ADB device

```
test@test:~$ adb devices
```

List of devices attached

```
4c00146473c28651dd0    device
```

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

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

```
t527-demo:/ #
```

5. 3. 3. adb debugging using a network connection

Using network adb does not require a USB2.0 peer-to-peer data cable to connect the computer to the development board, but communicates over the network, so first make sure that the wired or wireless network of the development board is connected, and then obtain the IP address of the development board, which will be used later.

1) Ensure that the **service.adb.tcp.port** of the Android operating system is set to 5555

```
console:/ # getprop | grep "adb.tcp"
```

```
[service.adb.tcp.port]: [5555]
```

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

```
console:/ # setprop service.adb.tcp.port 5555
```

```
console:/ # stop adbd
```

```
console:/ # start adbd
```

3) Install adb tool on Ubuntu PC

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

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

4) Then connect network adb on Ubuntu PC

```
test@test:~$ adb connect 192.168.1.xxx:5555    (Change it to the IP address of the development board)
```



```
* daemon not running; starting now at tcp:5037
```

```
* daemon started successfully
```

```
connected to 192.168.1.xxx:5555
```

```
test@test:~$ adb devices
```

```
List of devices attached
```

```
192.168.1.xxx:5555      device
```

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

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

```
t527-demo:/ #
```

5. 4. HDMI to VGA display test

1) The following accessories need to be prepared first

a. HDMI to VGA converter

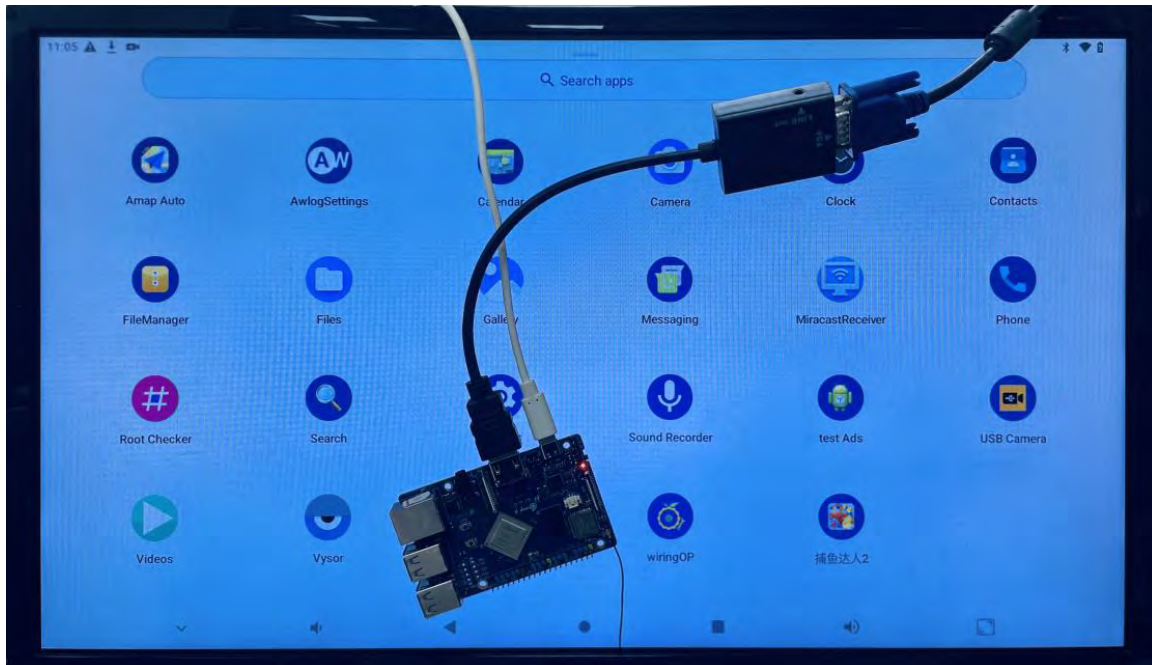


b. One VGA cable



c. A monitor or TV that supports VGA port

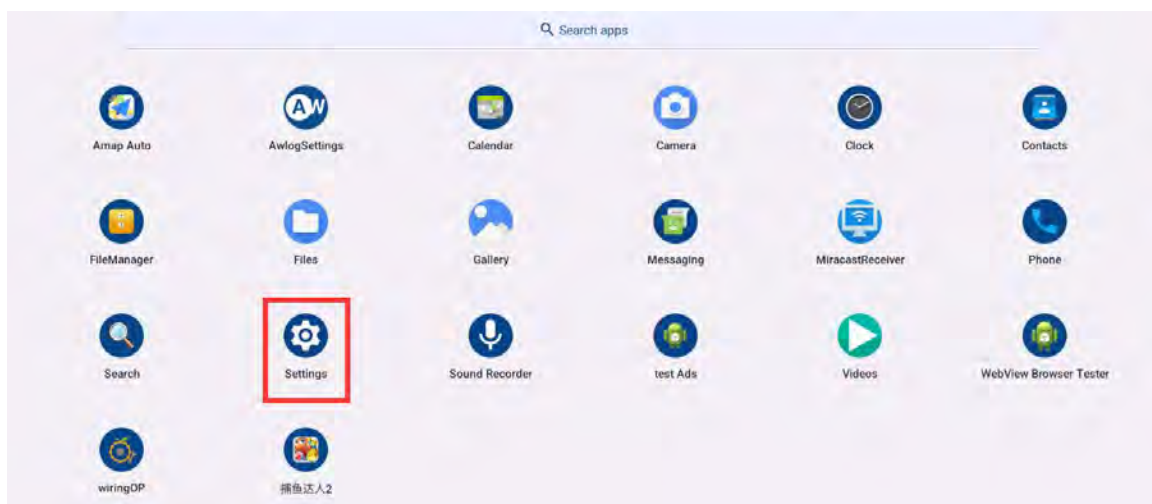
2) HDMI to VGA display test is shown below



When using HDMI to VGA display, the development board and the Android system of the development board do not need to do any Settings, only the HDMI interface of the development board can be displayed normally. So if the test has problems, check the HDMI to VGA converter, VGA cable, and monitor for problems.

5.5. WI-FI connection method

1) First select **Settings**

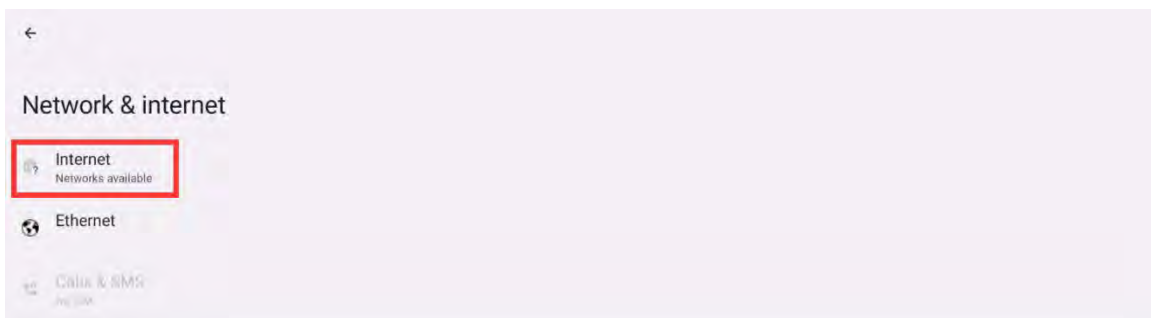




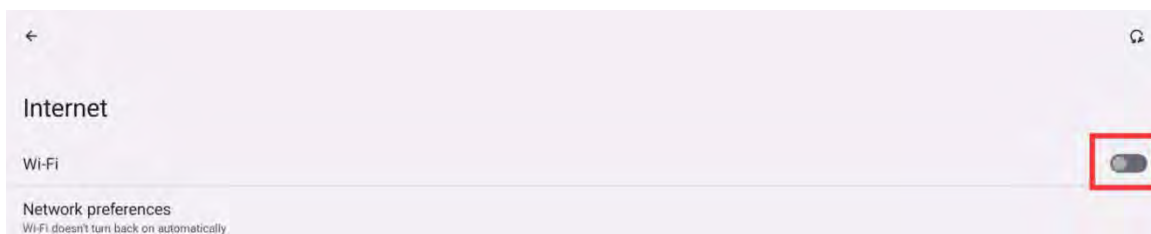
2) Then select **Network & Internet**



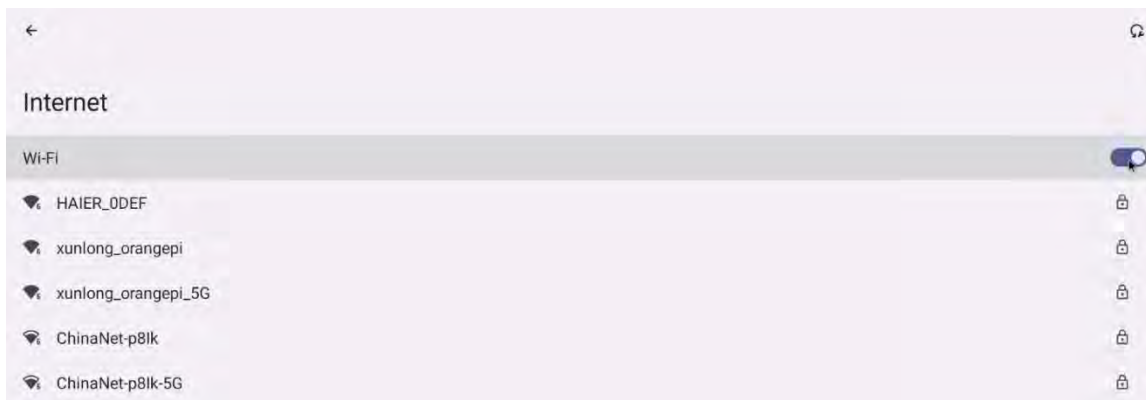
3) Then select **Internet**



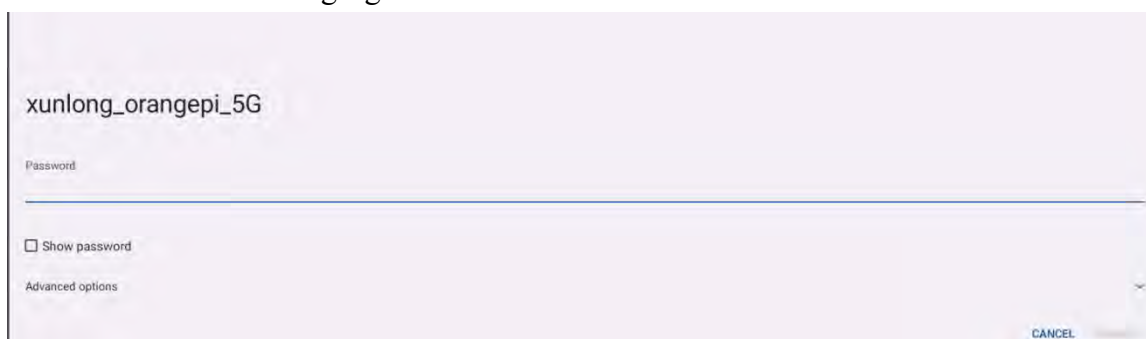
4) Then open WI-FI



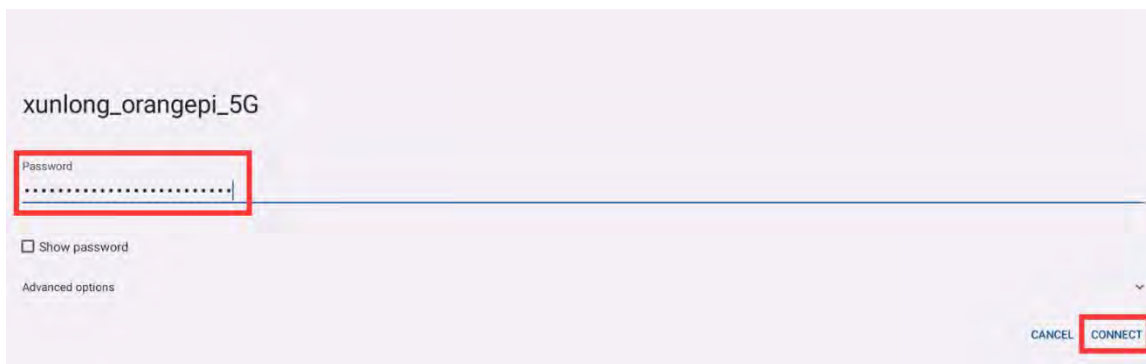
5) After turning on WI-FI, you can see the search signal under **Available networks**



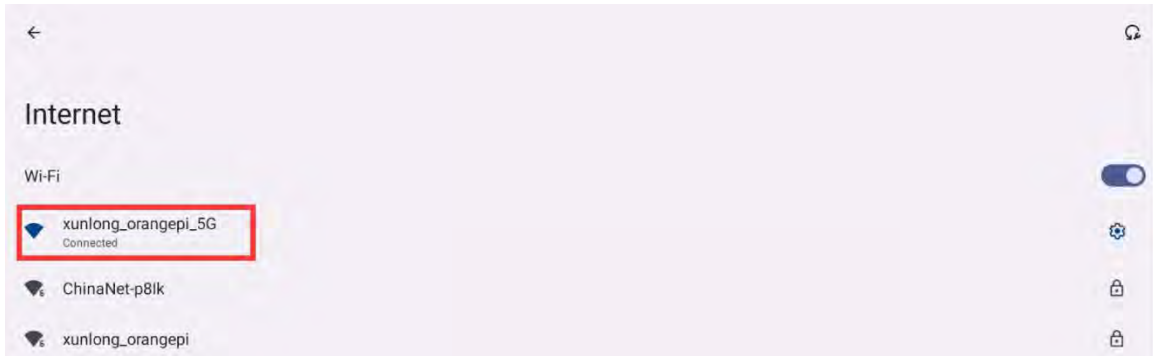
6) Select the WI-FI you want to connect to, and the password input interface will pop up as shown in the following figure



7) Then use the keyboard to enter the WI-FI password, and then use the **mouse** to click the Enter button in the virtual keyboard to start the WI-FI connection

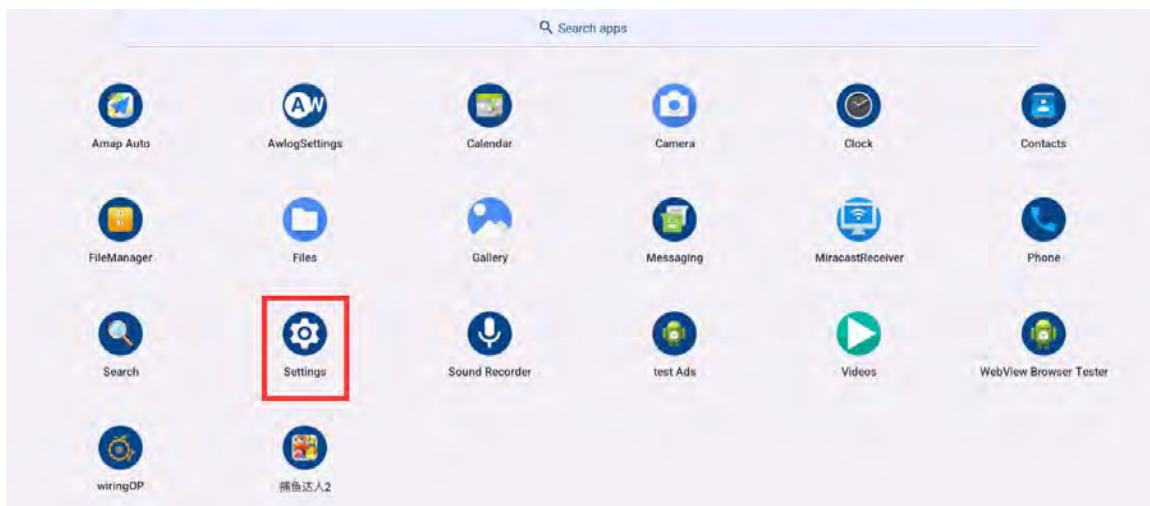


8) The display after the WI-FI connection is successful is as shown in the following figure

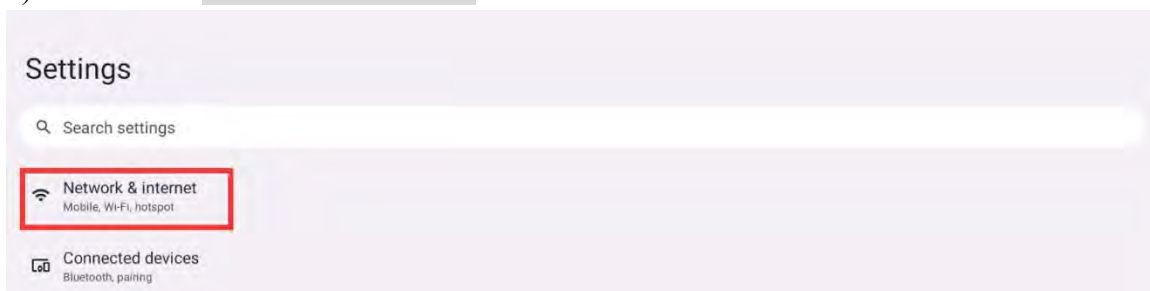


5.6. How to use WI-FI hotspot

- 1) Ensure that the Ethernet port is connected to a network cable and can access the Internet properly
- 2) Then select **Settings**

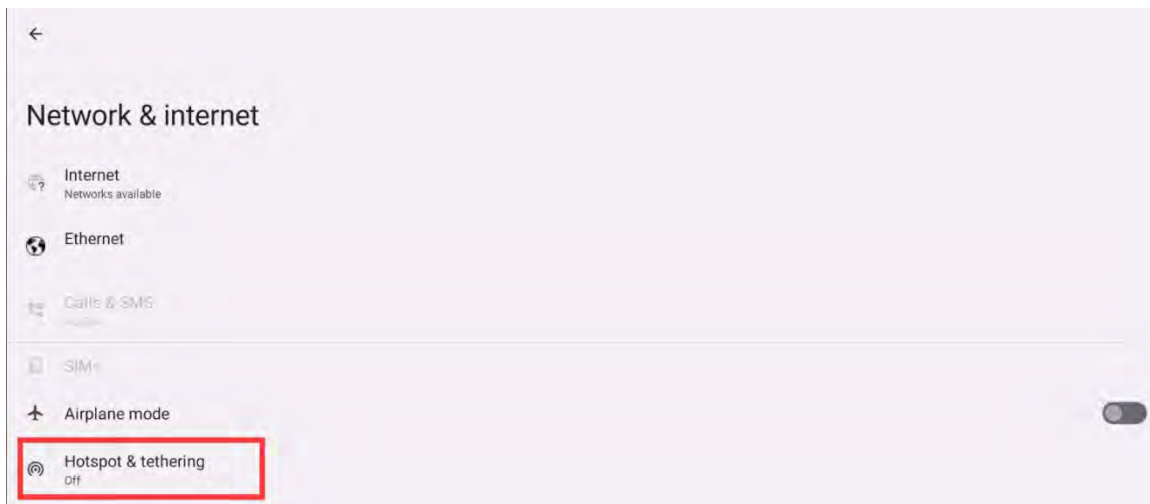


- 3) Then select **Network & Internet**





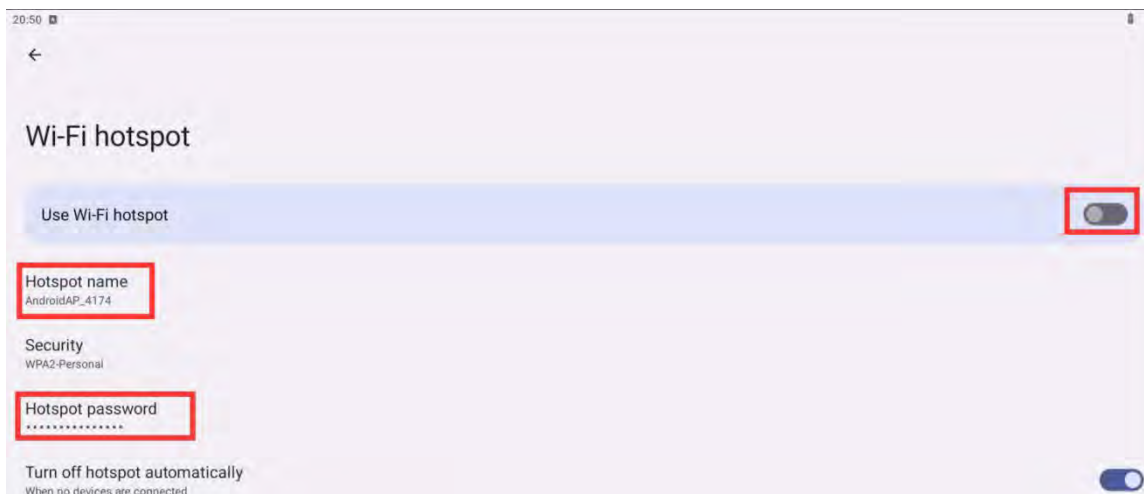
4) Then select **Hotspot & tethering**



5) Then select **Wi-Fi hotspot**



6) Then open the **Wi-Fi Hotspot**, you can also see the name and password of the generated Hotspot in the following figure, remember them, and need to use them when connecting to the hotspot (if you need to change the name and password of the Hotspot, you need to close the **Wi-Fi Hotspot** first, and then modify it)



7) At this time, you can take out your mobile phone, if everything is normal, you can find the WIFI Hotspot with the same name (**AndroidAP_4174**) shown below **Hotspot name** in the above picture in the WI-FI search list. Then you can tap AndroidAP_4174 to connect to the Hotspot, the password can be seen under **Hotspot password** in the image above



8) After successful connection, it will be displayed as shown in the following figure (different mobile phone interfaces will be different, the specific interface is subject to the display of your mobile phone). At this time, you can open a web page on the phone to see whether the Internet can be accessed. If the web page can be opened normally, it indicates that the **WI-FI Hotspot** of the development board can be used normally

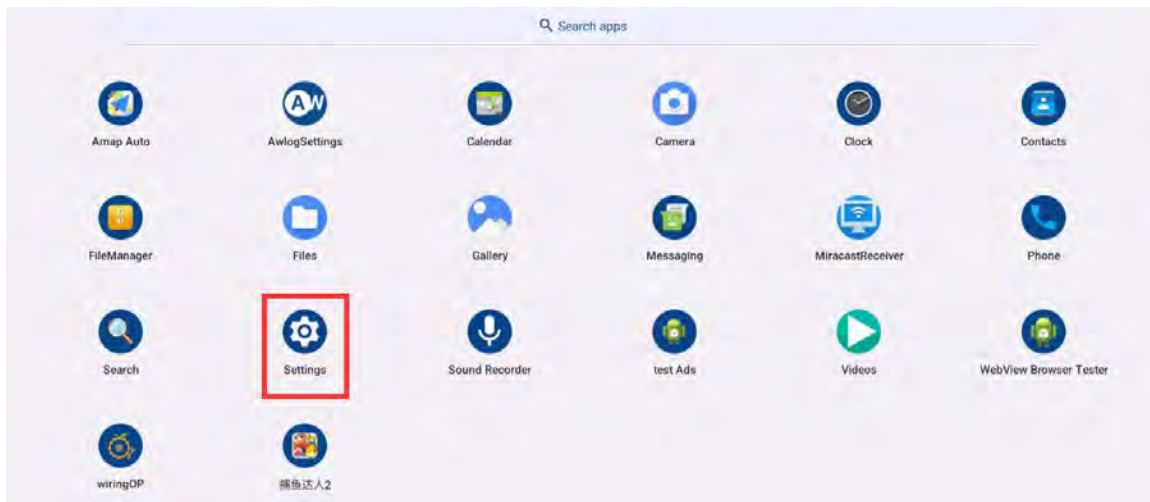




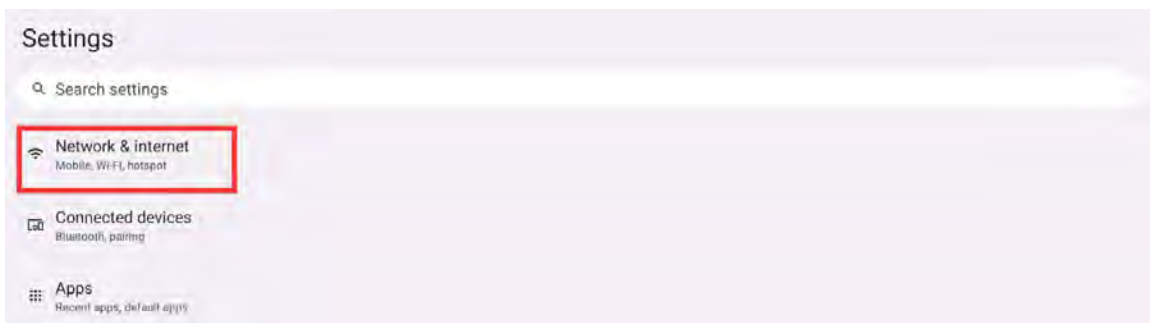
5.7. Method to view Ethernet port IP address

1) First, ensure that the gigabit network port of the development board is connected to a router or switch

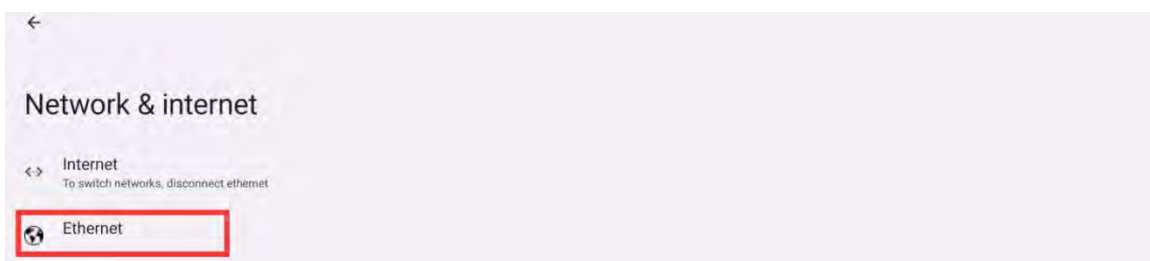
2) First open **Settings**



3) Then select **Network & Internet**

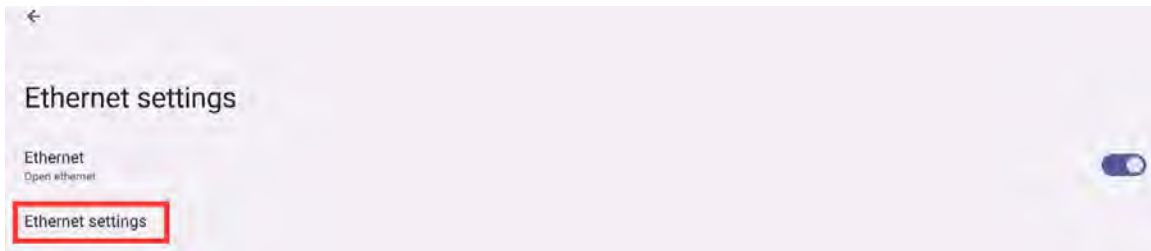


4) Then select **Ethernet**

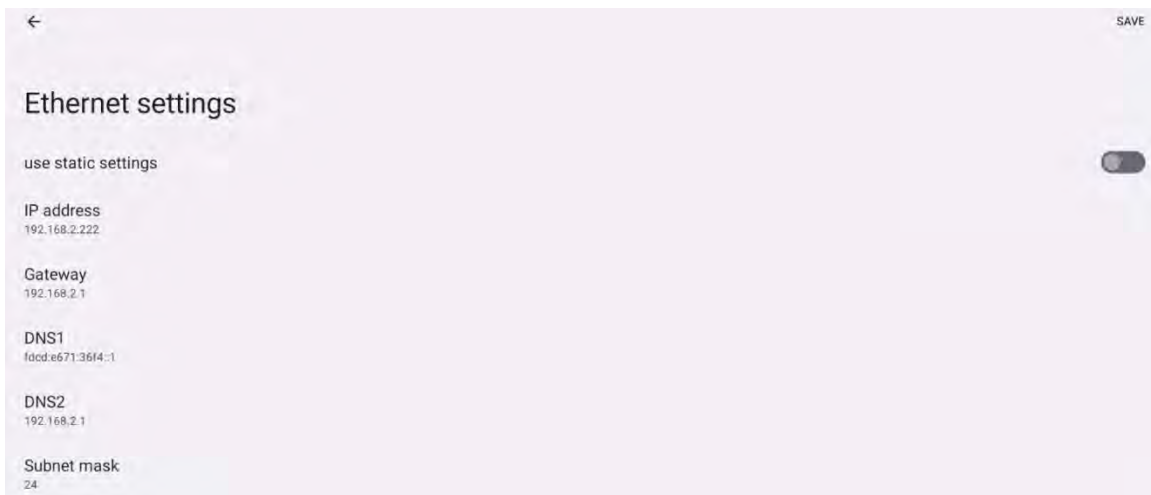




5) Then select **Ethernet settings**

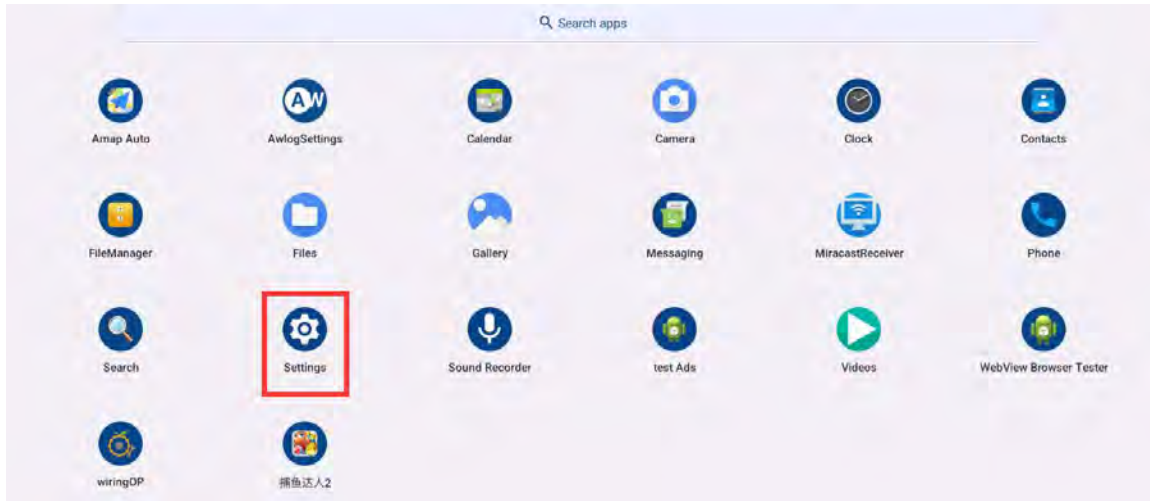


6) Then you can see the IP address information of the wired network port of the development board

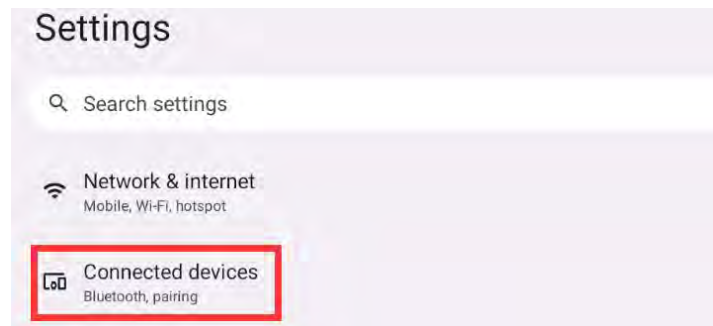


5. 8. Bluetooth connection method

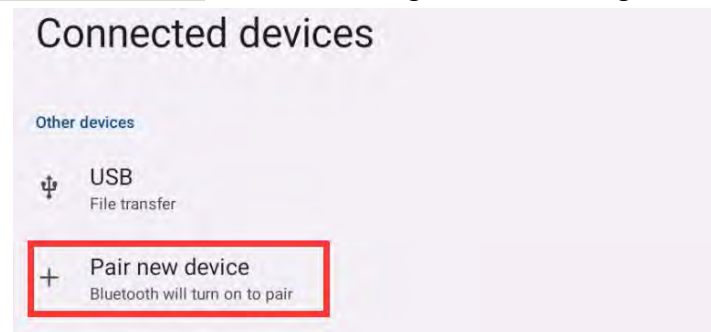
1) First select **Settings**



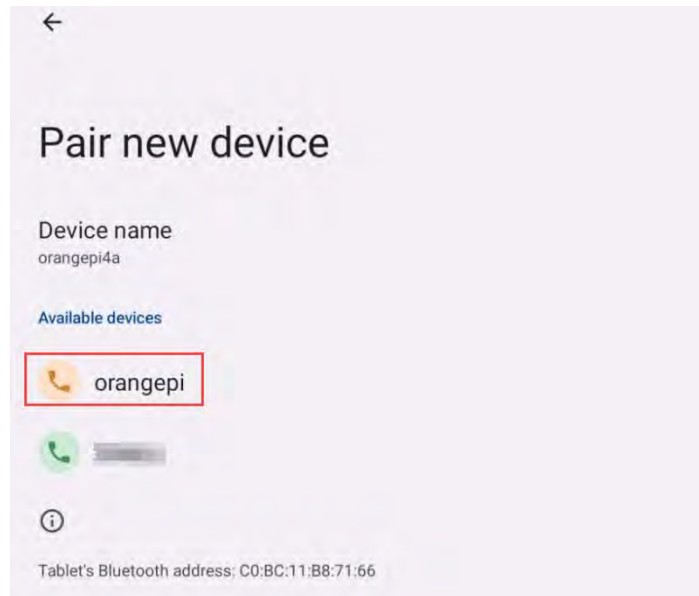
2) Then select **Connected devices**



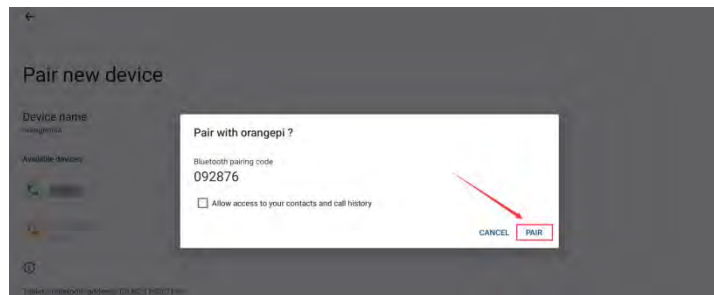
3) Then select **Pair new device** to start scanning the surrounding Bluetooth devices



4) The searched Bluetooth device will be displayed under **Available devices**

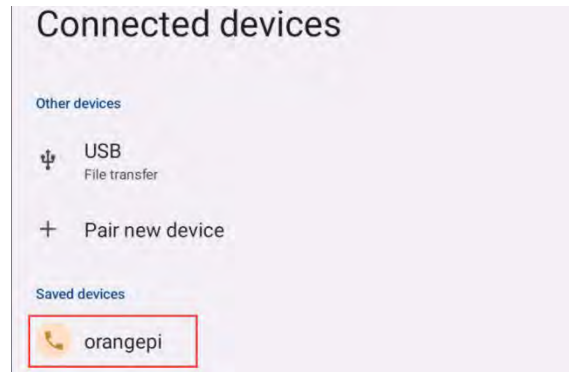


5) Then click on the Bluetooth device you want to connect to start pairing. When the following interface pops up, please use the mouse to select the **Pair** option

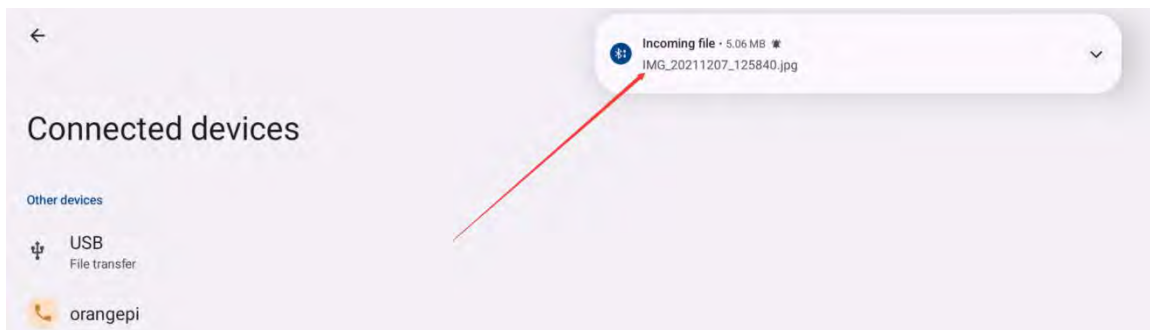


6) The test here is the configuration process of the development board and Bluetooth of **Android phones**. At this time, the confirmation interface will pop up on the phone, and the pairing process will start after clicking the pairing button on the phone

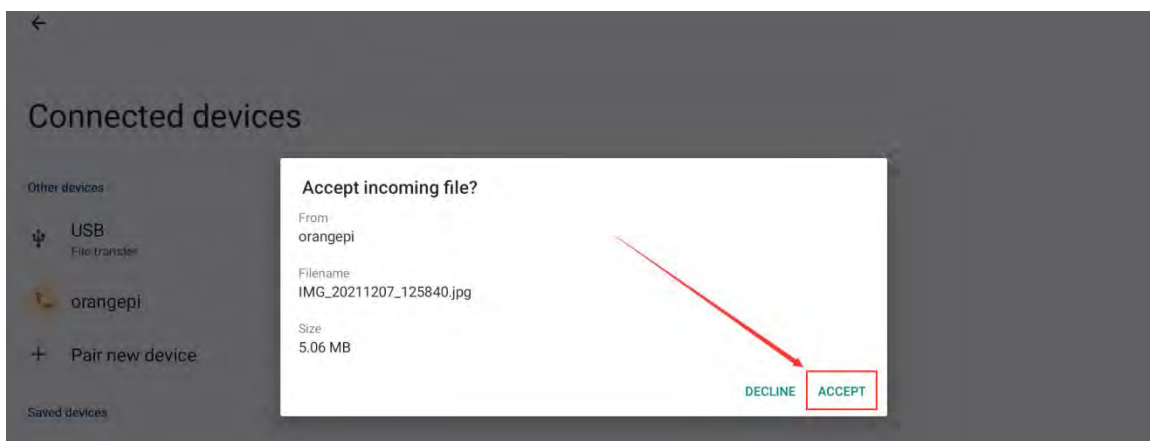
7) After pairing, open **Paired devices** and you will see the paired Bluetooth devices



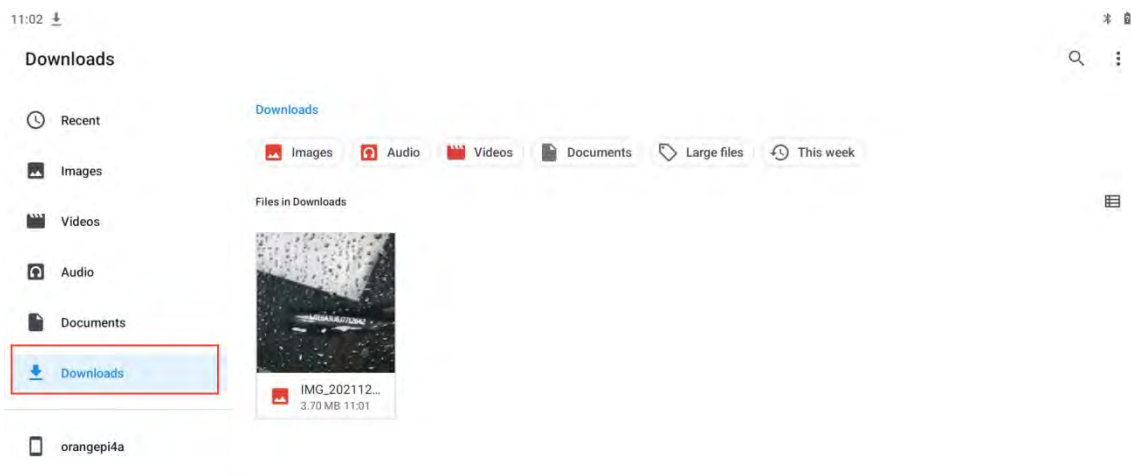
8) At this time, you can send a picture to the development board using the Bluetooth of your phone. After sending, you can see the following prompt in the Android system of the development board, and then click **Incoming file**



9) Then click **Accept** in the pop-up window to start receiving pictures sent by the phone



10) The picture received by the Bluetooth of the Android system can be viewed by opening the **Download** directory of the file manager



5.9. 10.1 inch MIPI screen usage

Make sure that the Android image you use is one of the following versions:
OrangePi4A_T527_Android13_lcd_v1.x.x.img

- 1) The screen needs to be assembled first, please refer to [the assembly method of 10.1 inch MIPI screen](#)
- 2) The position of the interface of the mipi lcd screen on the development board is shown as follows:



- 3) Connect the assembled screen to the LCD interface of the development board, pay **attention to unplug the HDMI interface**, connect the Type-C power supply to the board, and power on, after the system starts, you can see the screen displayed as follows (default is portrait screen)



5. 10. How to use eDP screen

eDP screens are touch-free.

Make sure that the Android image you use is one of the following versions:

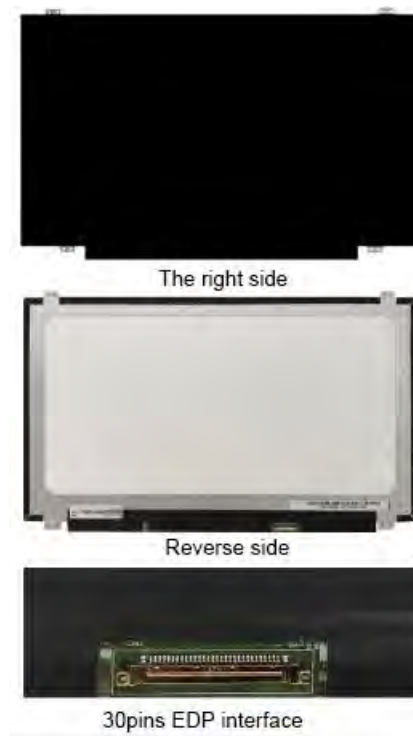
OrangePi4A_T527_Android13_v1.x.x.img

3) Currently only one 15.6-inch eDP screen is available, including the following accessories:

c. 0.5 Spaced 30-pin single-head coaxial cables



d. 15.6-inch eDP display, resolution is 1920x1080.



- 4) Connect the FPC end of the 30pin single-head codirectional cable to the eDP interface of the development board, and connect the other end to the eDP interface of the screen



- 5) Then connect the Type-C power supply to the board and power it on. After the system is started, you can see the screen displayed as shown in the following figure





5. 11. How to use USB camera

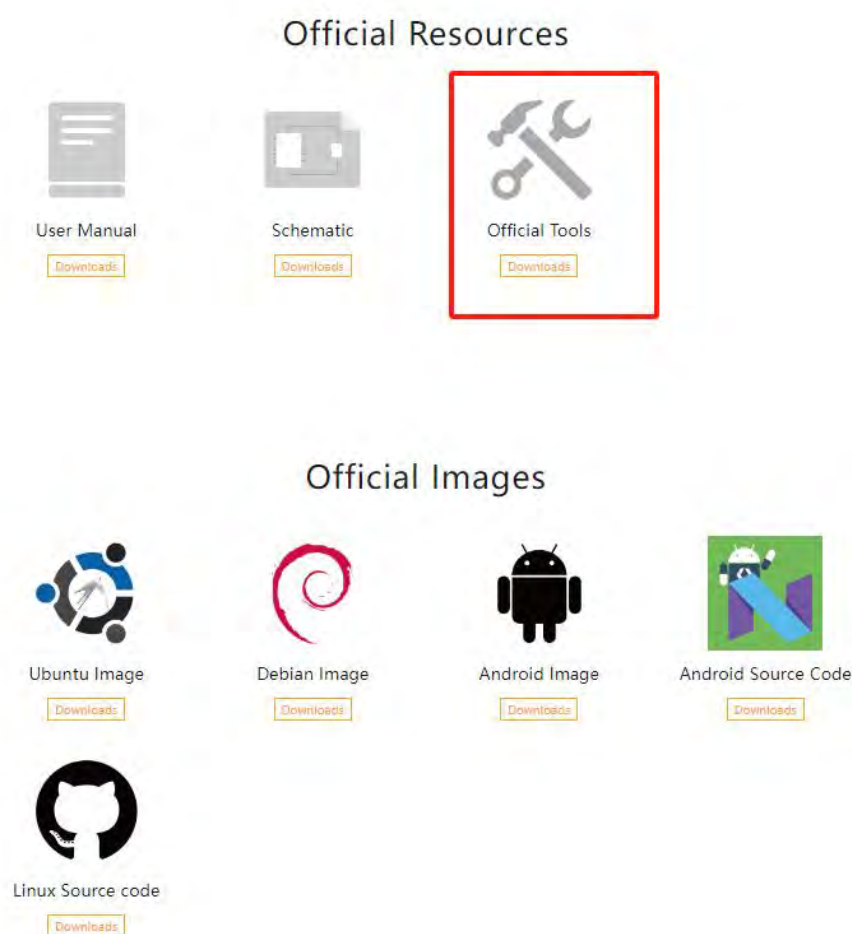
1) First, insert a USB (UVC protocol) camera into the USB interface of the development board

2) If the USB camera is properly identified, a video device node is generated under `/dev`

```
console:/ # ls /dev/video0  
/dev/video0
```

3) Then ensure that the ADB connection between the Ubuntu PC and the development board is normal. Please refer to the instructions in the section on [adb Usage](#)

4) Download the USB camera test APP from the **official tool** on the development board data download page

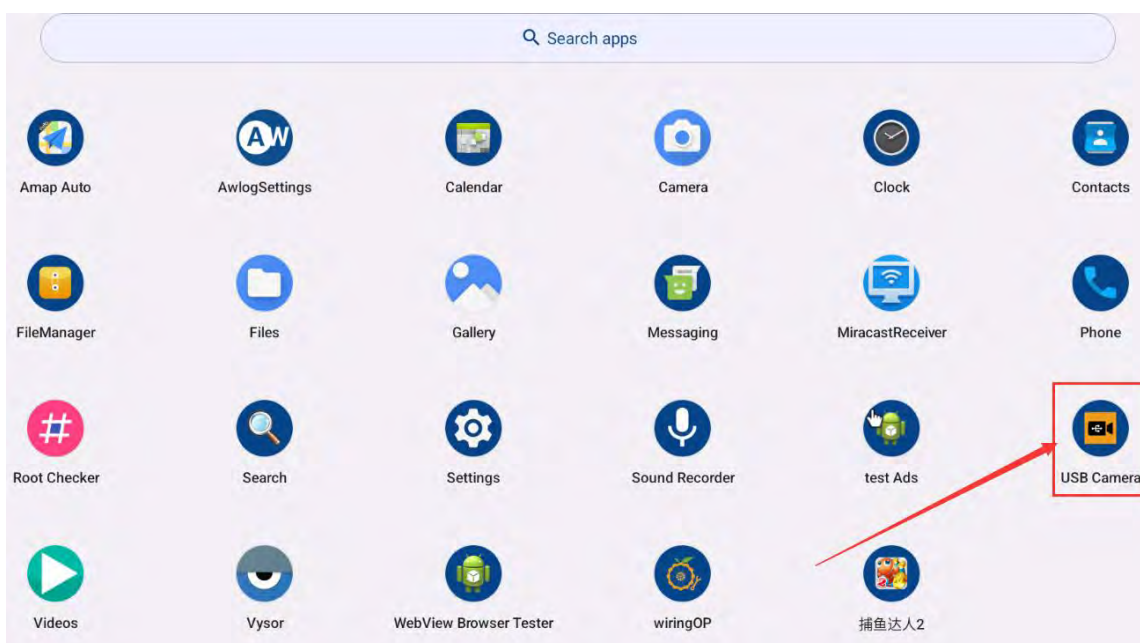




5) Then use adb command to install the USB camera test APP into the Android system, of course, you can also use the U disk copy for installation

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

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



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

5. 12. Android system ROOT Description

The Android system released by Orange Pi has been ROOT, you can use the

**following method to test.**

1) Download **rootcheck.apk** from the **official tool** on the development board data download page

Official Resources



User Manual
[Downloads](#)



Schematic
[Downloads](#)



Official Tools
[Downloads](#)

Official Images



Ubuntu Image
[Downloads](#)



Debian Image
[Downloads](#)



Android Image
[Downloads](#)



Android Source Code
[Downloads](#)



Linux Source code
[Downloads](#)

官方工具

2020-11-03 14:09 失效时间: 永久有效

[返回上一级](#) [全部文件](#) [官方工具](#) [Android测试APP](#)

文件名称	大小	修改日期
 usbcamera.apk	20M	2020-11-04 13:56
 rootcheck.apk	2M	2020-11-04 13:48
 REFFB.apk	4.4M	2020-11-04 13:48

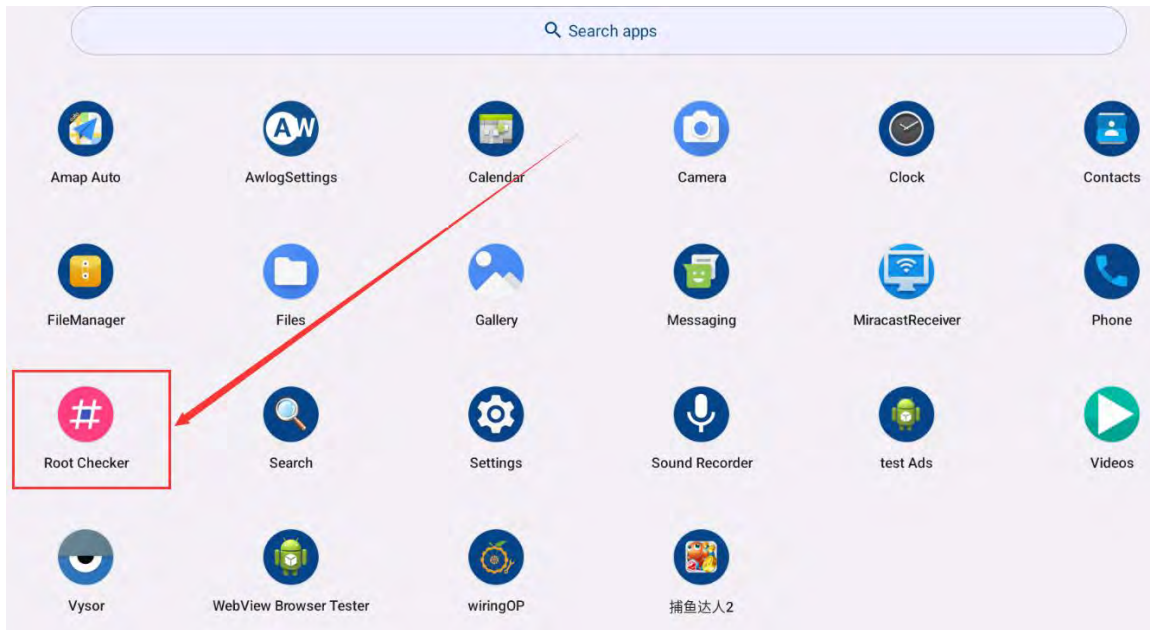
2) Then ensure that the ADB connection between the Ubuntu PC and the development board is normal. Please refer to the instructions in the section on **adb Usage**



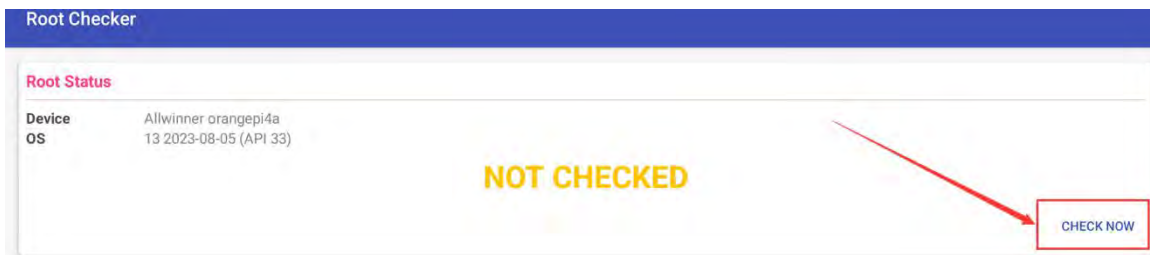
3) Then use adb command to install rootcheck.apk into the Android system, of course, you can also use the U disk copy for installation

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

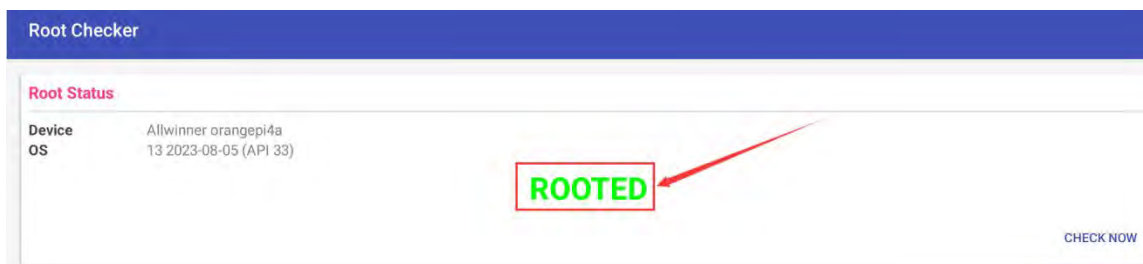
4) After installation, you can see the startup icon of the ROOT test tool on the Android APP interface



5) After opening the **ROOT test tool** for the first time, the display interface is as shown in the following figure



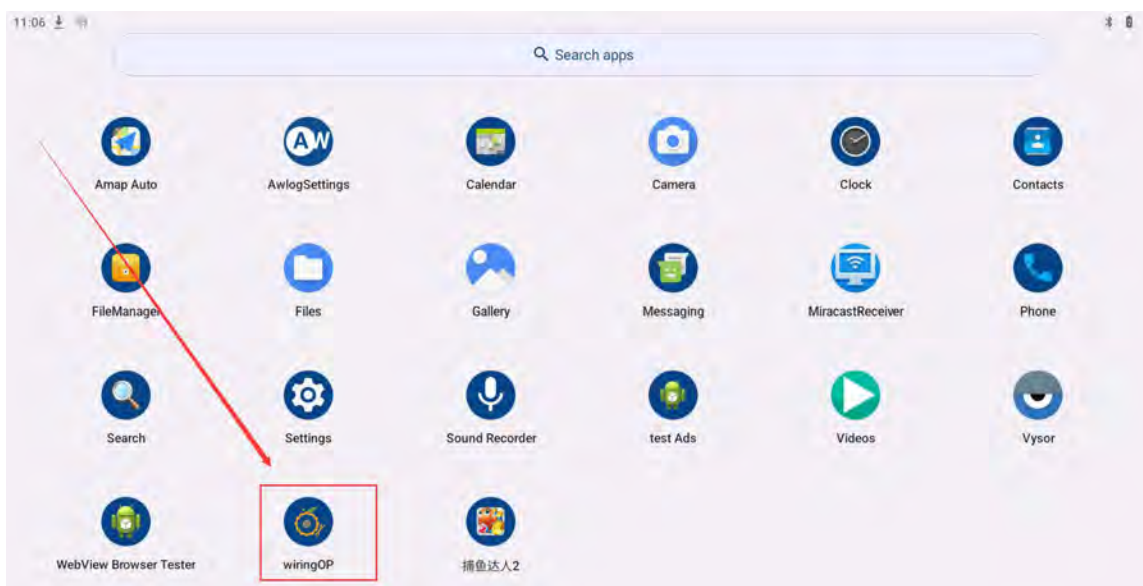
6) Then you can click **CHECK NOW** to start the ROOT status check of the Android system. After the check, the display is as follows, you can see that the Android system has obtained the ROOT permission



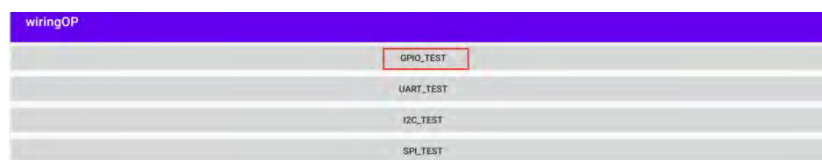
5. 13. 40 pin interface GPIO, UART, SPI test

5. 13. 1. 40 pin GPIO port test method

1) First open the wiringOP APP on your desktop



2) Then click **GPIO_TEST** button to open the GPIO test interface



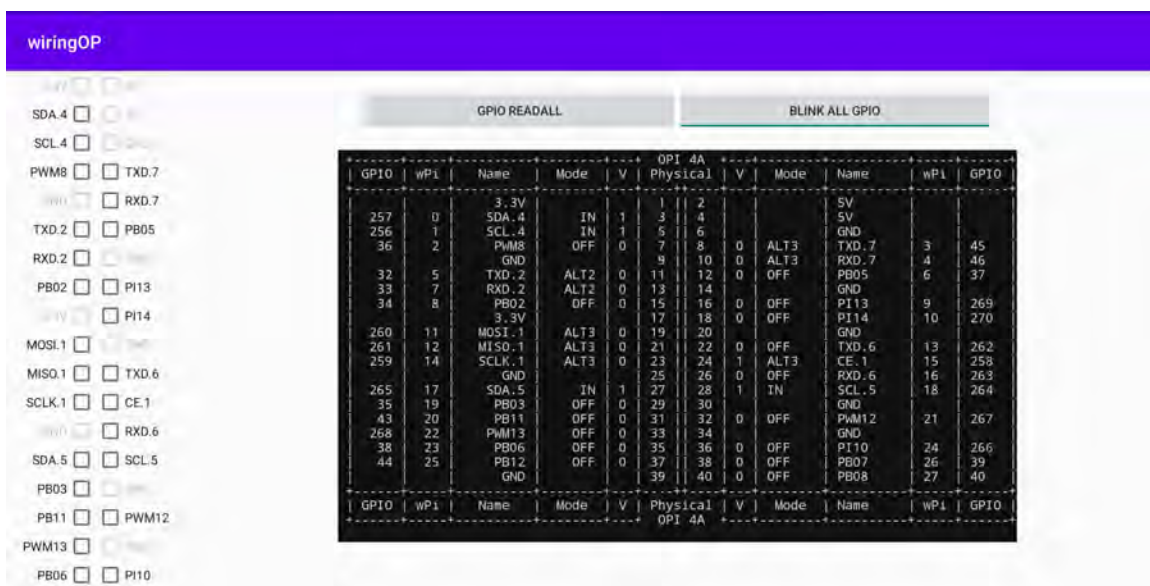
3) The GPIO test interface is shown in the figure below. The two rows of **CheckBox** buttons on the left are one-to-one corresponding to the 40-pin pin. When **CheckBox** button is checked, the corresponding GPIO pin will be set to **OUT** mode and the pin level will be set to high. When unchecked, the GPIO pin level is set to low; When you click the



GPIO READALL button on the right, you can get the wPi number, GPIO mode, and pin parity information. When the **BLINK ALL GPIO** button is clicked, all pins continuously switch between high and low levels



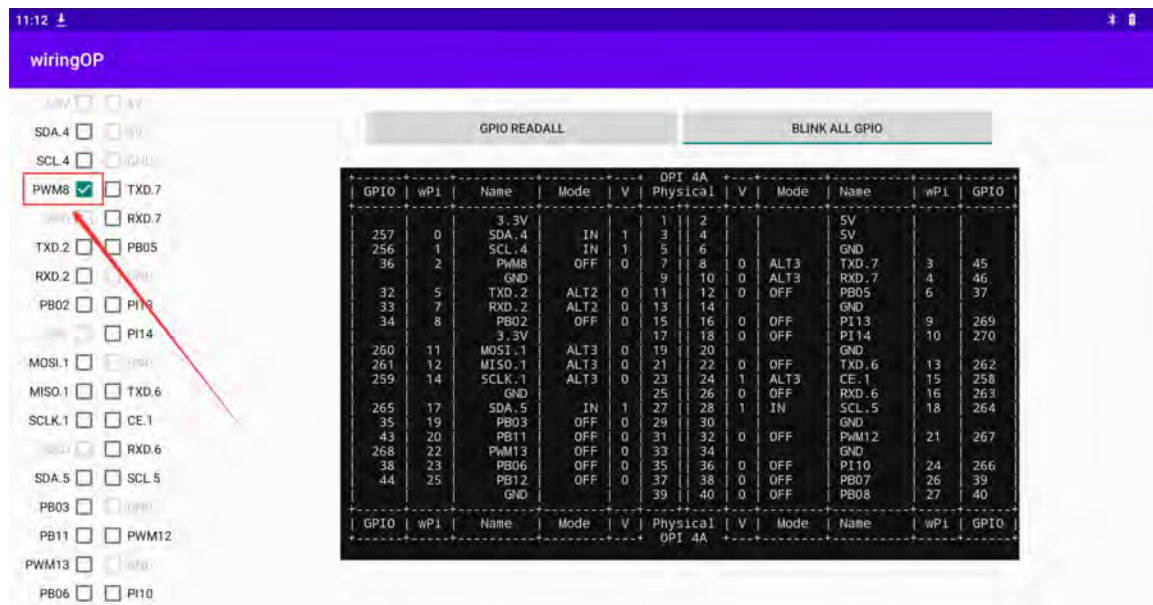
4) Then click the **GPIO READALL** button, and the output information is as shown below:



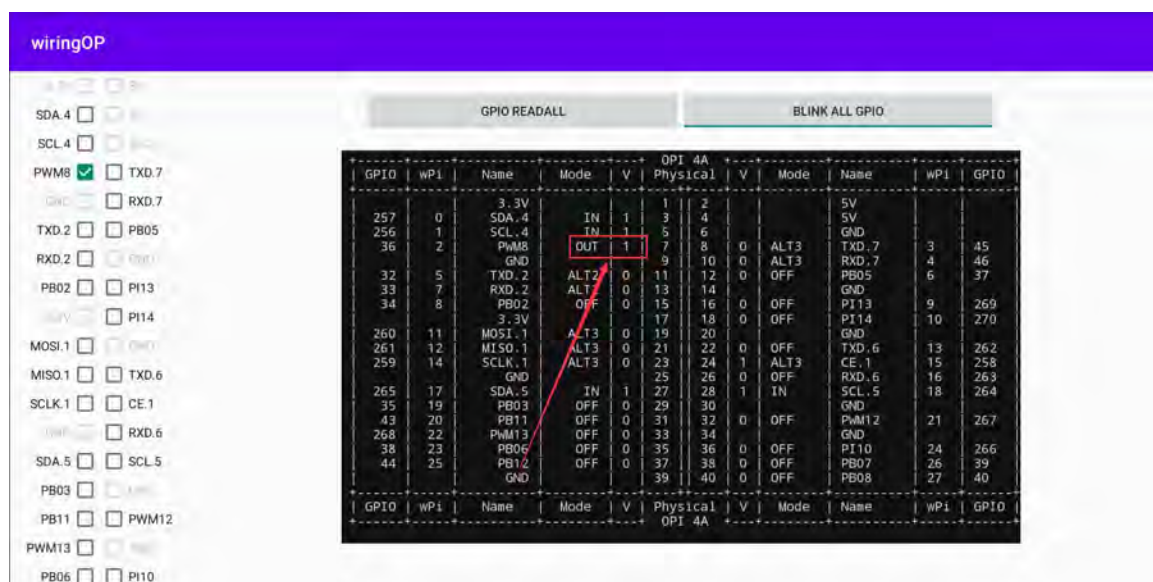
5) There are a total of 28 GPIO ports available in the 40 pin development board. Taking pin 7- corresponding to GPIO PB4- corresponding to wPi serial number 2- as an example, we will demonstrate how to set the high and low levels of GPIO ports. Firstly, click on



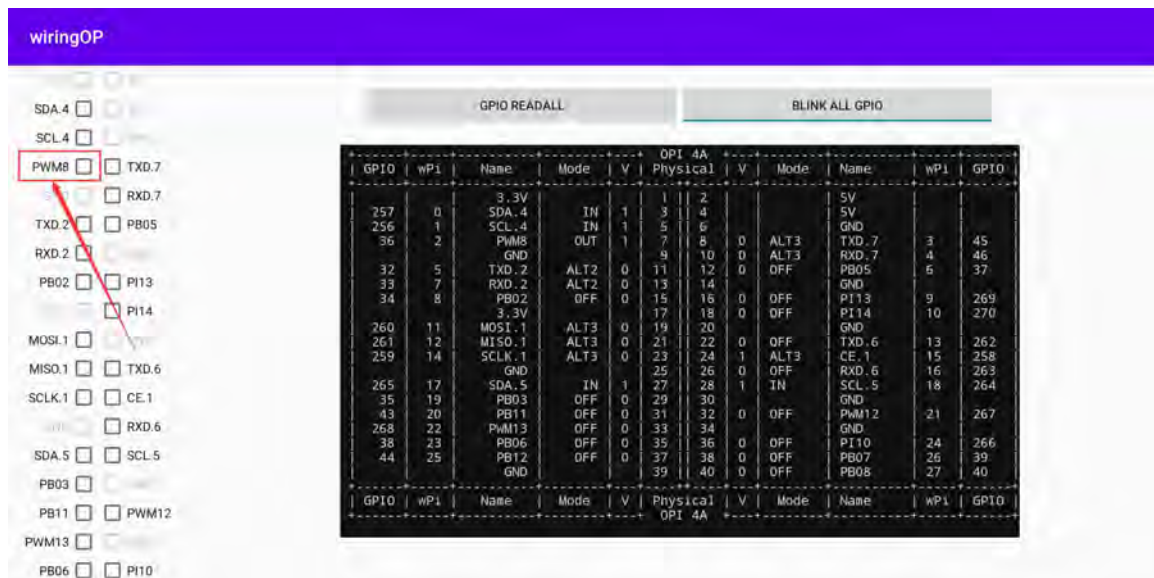
the **CheckBox** button corresponding to pin 7. When the button is selected, pin 7 will be set to a high level. After setting, you can use a multimeter to measure the voltage value of the pin. If it is **3.3v**, it means that the high level has been successfully set



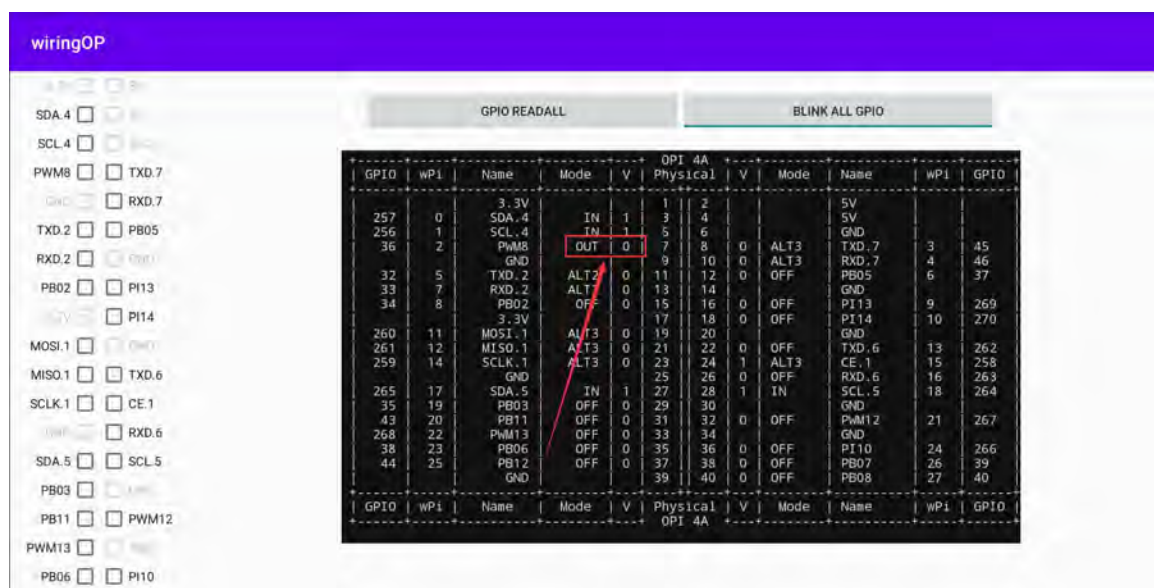
6) Then click the **GPIO READALL** button, and you can see that the current mode of pin 7 is **OUT** and the pin level is high



7) Click the **CheckBox** button in the following image again to uncheck the status. Pin 7 will be set to a low level. After setting, you can use a multimeter to measure the voltage value of the pin. If it is **0v**, it means that the low level has been successfully set



8) Then click the **GPIO READALL** button, and you can see that the current mode of pin 7 is OUT and the pin level is low



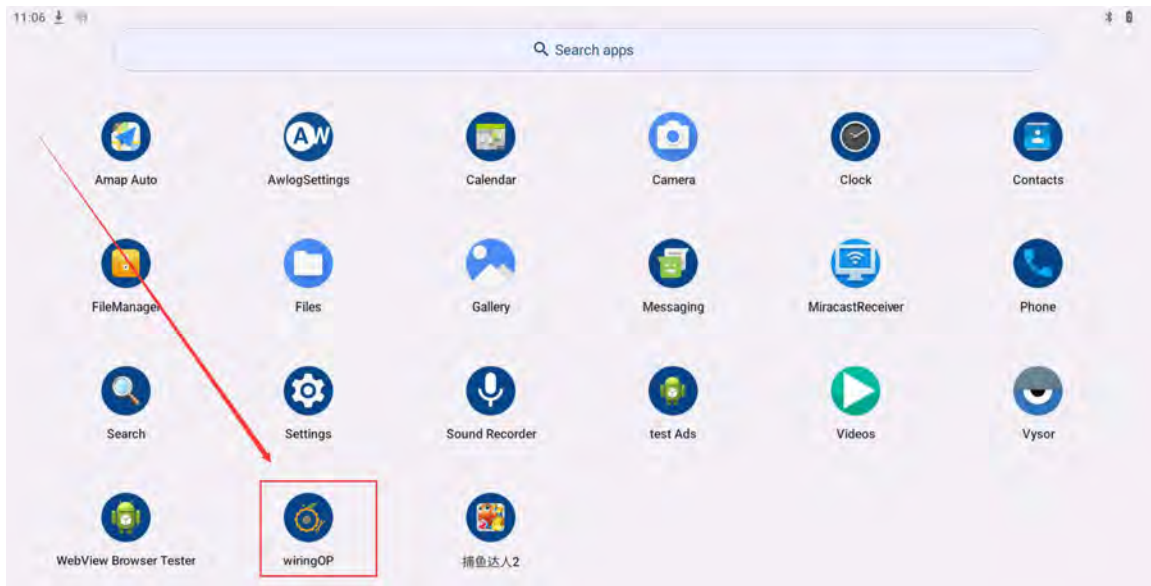
5. 13. 2. 40 pin UART testing method

1) By default, **UART2** and **UART7** are enabled in Android, corresponding to device nodes **/dev/ttyAS2** and **/dev/ttyAS7**

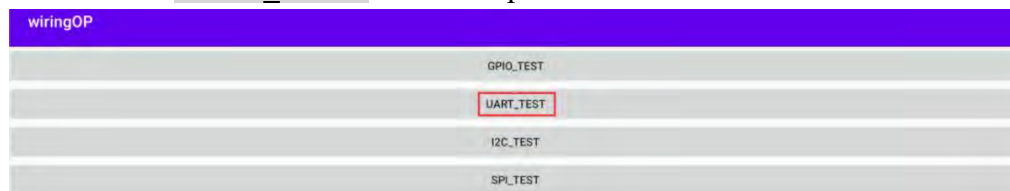
```
t527-demo:/ $ ls /dev/ttyAS*
```

```
ttyAS0  ttyAS1  ttyAS2  ttyAS7
```

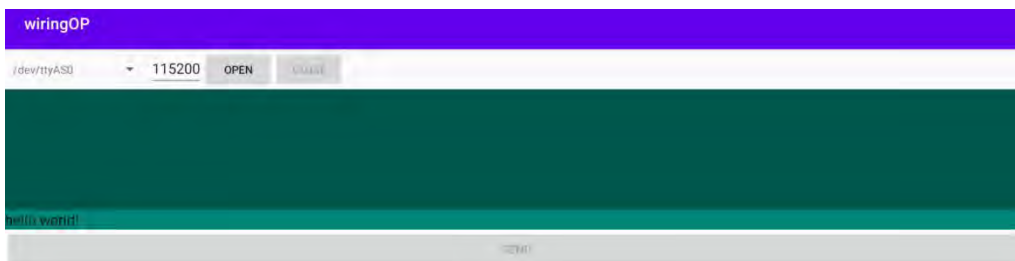
2) First, open the WiringOP app on the desktop



3) Then click the **UART_TEST** button to open the UART test interface



4) The serial port test interface of WiringOP is shown in the following figure



5) Taking testing UART2 as an example, select the **/dev/ttyAS2** node in the selection box





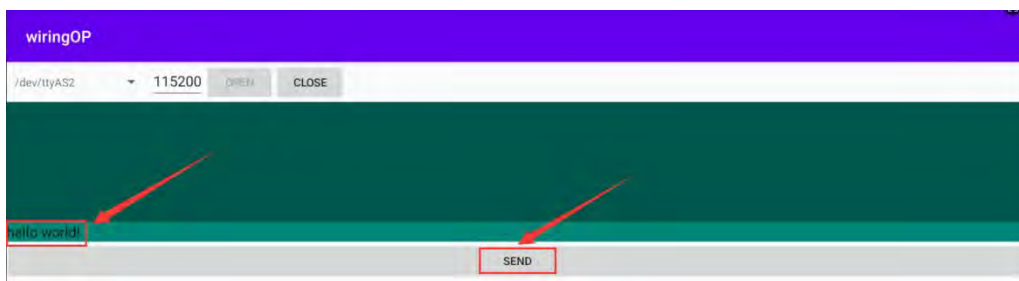
- 6) Enter the desired baud rate in the editing box, then click the **OPEN** button to open the **/dev/ttyAS2** node. After successful opening, the **OPEN** button becomes unselectable, and the **CLOSE** and **SEND** buttons become selectable



- 7) Then use DuPont wire to short-circuit the rx and tx pins of UART 2

	uart2
tx pin	Corresponding to pin 11 of pin 40
rx pin	Corresponding to pin 13 of pin 40

- 8) Then you can enter a character in the send edit box below and click the **SEND** button to start sending

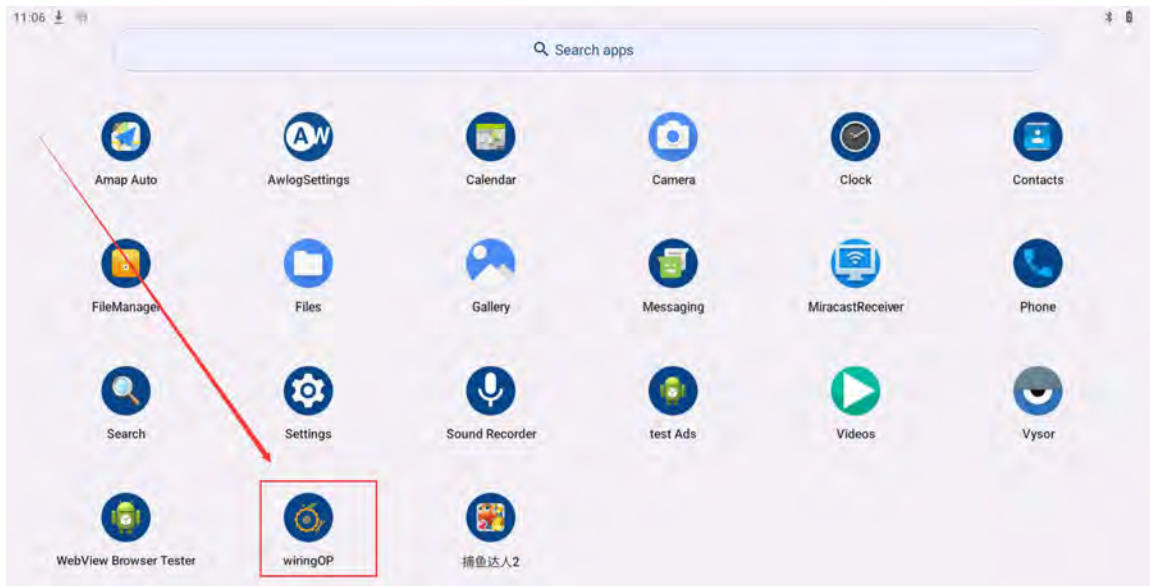


- 9) If everything is normal, the received string will be displayed in the receiving box



5. 13. 3. 40 pin SPI testing method

- 1) The SPI that can be used in 40 pins is SPI1, and the corresponding device node is **/dev/spidev1.0**

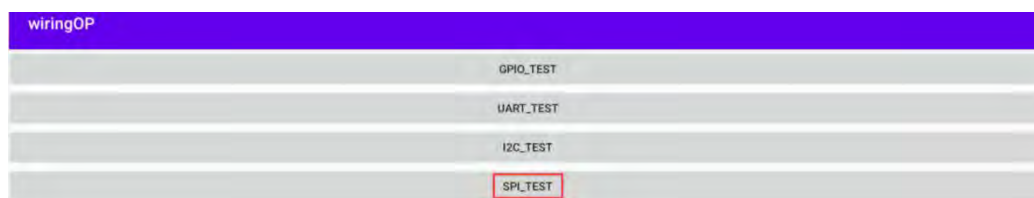


2) Here is a demonstration of testing the SPI1 interface using the **w25q64** module. First, connect the w25q64 module to the SPI1 interface

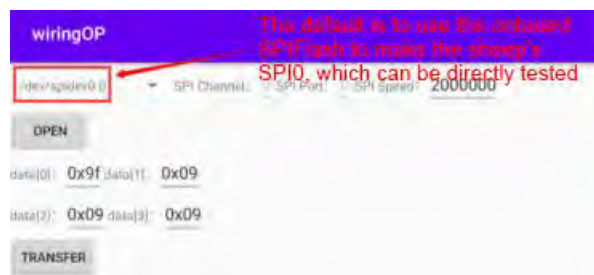
If there is no w25q64 module, it doesn't matter, because there is a SPIFlash connected to SPI0 on the development board, and the SPI0 configuration is also enabled by default in Android, so we can directly use the onboard SPIFlash for testing.

3) Then open the WiringoP app on the desktop

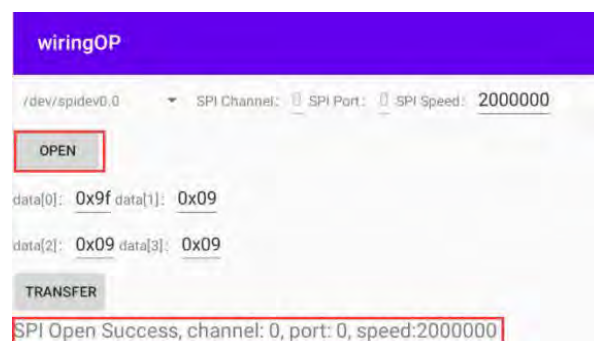
4) Then click the **SPI_TEST** button to open the SPI testing interface



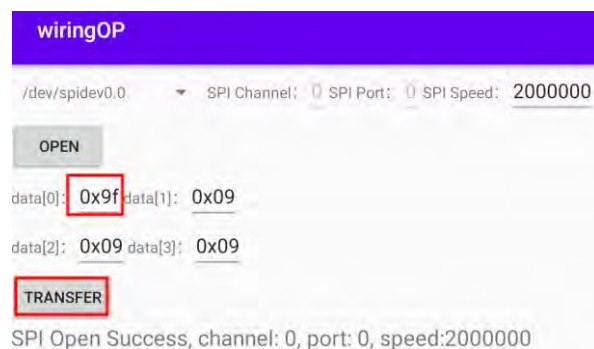
5) Then select the device node of SPI in the upper left corner. If testing the onboard SPIFlash directly, keep the default **/dev/spidev0.0**. If the **w25q64** module is connected to the 40 pin SPI1, select **/dev/spidev1.0**



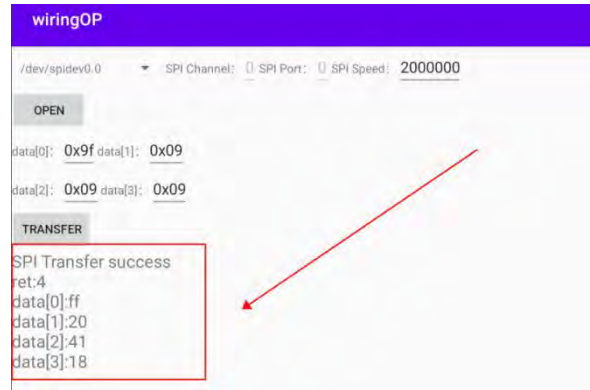
6) Then click the **OPEN** button to initialize SPI



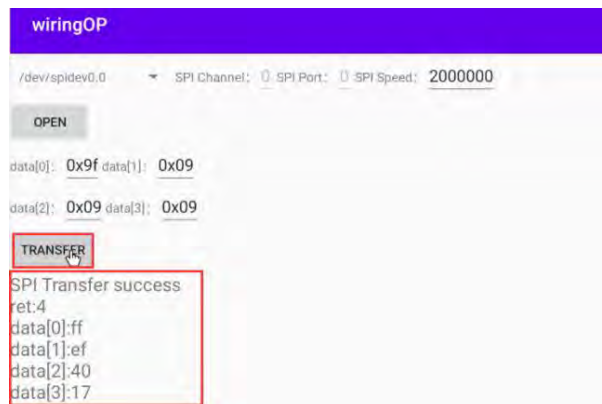
7) Then fill in the bytes that need to be sent, such as reading the ID information of the onboard SPIFlash, filling in the address 0x9f in data [0], and then clicking the **TRANSFER** button



8) Finally, the APP will display the ID information of the read onboard SPI Flash



9) If reading the w25q64 module connected to the 40 pin SPI1, the read ID information is shown in the following figure



10) The MANUFACTURER ID of the w25q64 module is EFh, and the Device ID is 4017h, which corresponds to the values read above (h represents hexadecimal)

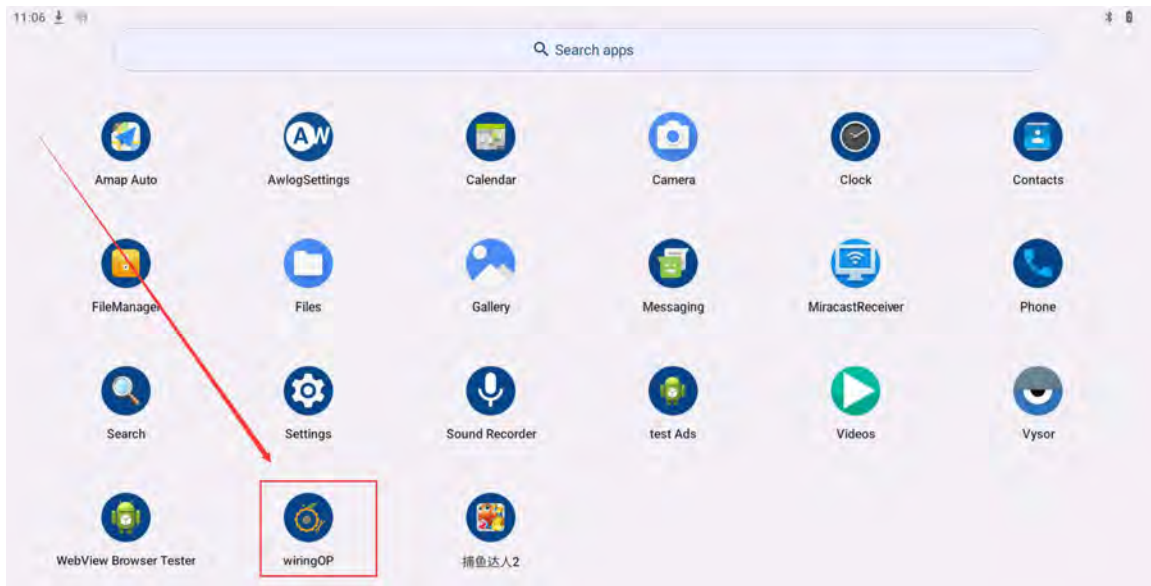
MANUFACTURER ID	(MF7 - MF0)	
Winbond Serial Flash	EFh	
Device ID	(ID7 - ID0)	(ID15 - ID0)
Instruction	ABh, 90h, 92h, 94h	9Fh
W25Q64FV (SPI)	16h	4017h
W25Q64FV (QPI)	16h	6017h

5. 13. 4. 40 pin I2C testing method

1) In Android, i2c4 and i2c5 in pin 40 are enabled by default, corresponding to device nodes `/dev/i2c-4` and `/dev/i2c-5`, respectively

```
console:/ # ls /dev/i2c-4 /dev/i2c-5
/dev/i2c-4 /dev/i2c-5
```

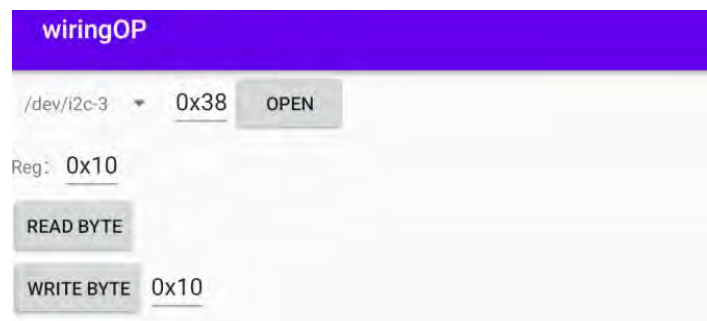
2) First, open the WiringOP app on the desktop



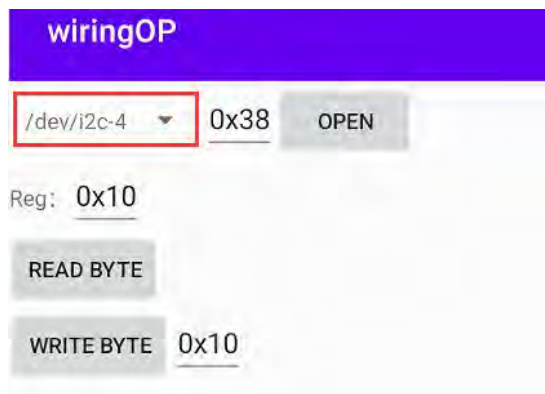
3) Then click the **I2C_TEST** button to open the testing interface of i2c



4) The i2c testing interface of WiringOP is shown in the following figure



5) Taking testing i2C4 as an example, select the **/dev/i2c-4 node** in the selection box



6) Then connect an I2C device to the 40 pin I2C4 pin, using the DS1307 RTC module as an example



Pin of RTC module	Development board 40 pin corresponding pins
5V	Pin 2
GND	Pin 6
SDA	Pin 3
SCL	Pin 5

7) The i2c address of the ds1307 RTC module is 0x68. After connecting the cable, we can use the **i2cdetect -y -r 4** command in the serial port command line to check if we can scan the i2c address of the ds1307 RTC module. As shown in the figure below, if the address 0x68 can be seen, it indicates that the wiring of the ds1307 RTC module is correct.

```
console:/ # i2cdetect -y 4
```

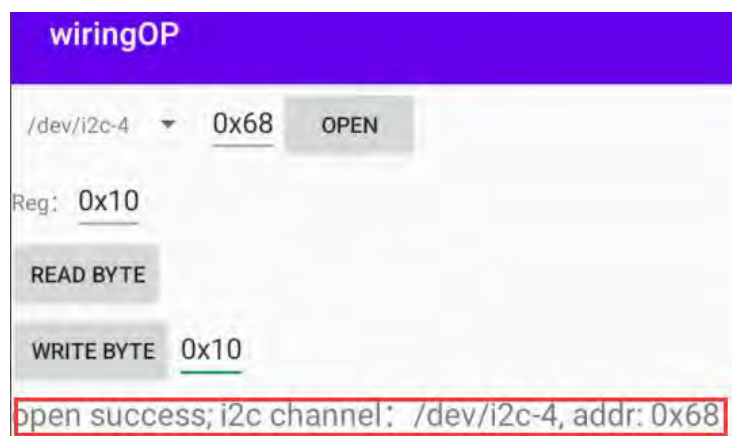


```
t527-demo:/ # i2cdetect -y -r 4
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  68  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
t527-demo:/ #
```

8) Then set the address of i2c to 0x68 in WirgOP, and click the **OPEN** button to open i2c4



9) After clicking the **OPEN** button to open i2c4, the display is as follows:



10) Then we will test writing a value to the register of the RTC module, such as writing 0x55 to address 0x1c



- a. We first set the address of the register that needs to be written to 0x1c



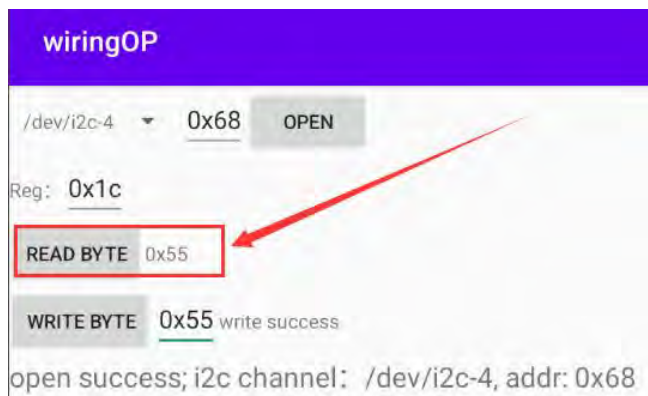
- b. Then set the value to be written as 0x55



- c. Then click the **WRITE BYTE** button to perform the write action



11) Then click the **READ BYTE** button to read the value of the 0x1c register. If it displays 0x55, it means that the i2c read-write test has passed



6. Compilation method of Android 13 source code

6.1. Download the source code of Android 13

- 1) Firstly, download the compressed Android 13 source code from Baidu or Google Drive
- 2) After downloading the compressed Android 13 source code, please check if the MD5 checksum is correct. If it is not correct, please download the source code again. The method for checking MD5 checksum is as follows:

```
test@test:~$ md5sum -c md5sum
t527_android13.tar.gz00: OK
t527_android13.tar.gz01: OK
.....
```

- 3) Execute the following command to extract the Android source code

```
test@test:~$ cat t527_android13.tar.gz0* | tar -xvzf -
```



6.2. Compile the source code for Android 13

The compilation of Android 13 is performed on an x86_64 computer with **Ubuntu 22.04** installed. There may be some differences in the dependencies of other versions of Ubuntu system packages. The download address for the Ubuntu 22.04 **amd64** version image is as follows:

<https://repo.huaweicloud.com/ubuntu-releases/22.04/ubuntu-22.04.2-desktop-amd64.iso>

The recommended hardware configuration for compiling Android 13 source code on an x86_64 computer is 16GB or more of memory, 200GB or more of hard disk space, and as many CPU cores as possible.

1) First, install and compile the software packages required for Android 13 source code

```
test@test:~$ sudo apt-get update
test@test:~$ sudo apt-get install -y git gnupg flex bison gperf build-essential \
zip curl zlib1g-dev gcc-multilib g++-multilib libc6-dev-i386 \
lib32ncurses5-dev x11proto-core-dev libx11-dev lib32z1-dev ccache \
libgl1-mesa-dev libxml2-utils xsltproc unzip u-boot-tools python-is-python3 \
libssl-dev libncurses5 clang gawk
```

2) Then compile the code in the **longan** folder, which mainly contains u-boot and Linux kernel

a. First run **./build.sh config** sets compilation options

```
test@test:~$ cd t527_android13/longan
test@test:~/t527_android13/longan$ ./build.sh config
=====ACTION List: mk_config ;=====
options :
All available platform:
    0. android
    1. linux
Choice [android]: 0
All available ic:
    0. a523
    1. a527
```



```
2. t527
Choice [t527]: 2
All available board:
0. demo
1. demo_car
2. demo_fastboot
3. demo_linux_aiot
4. demo_linux_car
Choice [demo]: 0
All available flash:
0. default
1. nor
Choice [default]: 0
Setup BSP files
INFO: Prepare toolchain ...
INFO: longanforandroid project, clang_top_path: /home/test/a527_android13
INFO: kernel defconfig: generate /home/test/a527_android13/longan/out/t527/kernel/build/.config by
/home/test/a527_android13/longan/device/config/chips/t527/configs/default/linux-5.15/android13_arm64_defconfig
INFO: Prepare toolchain ...
make: Entering directory '/home/test/a527_android13/longan/kernel/linux-5.15'
make[1]: 进入目录 “/home/test/a527_android13/longan/out/t527/kernel/build”
GEN      Makefile
HOSTCC   scripts/basic/fixdep
HOSTLD   scripts/kconfig/conf
*** Default configuration is based on
'../../../../device/config/chips/t527/configs/default/linux-5.15/android13_arm64_defconfig'
#
# configuration written to .config
#
make[1]: 离开目录 “/home/test/a527_android13/longan/out/t527/kernel/build”
make: Leaving directory '/home/test/a527_android13/longan/kernel/linux-5.15'
INFO: clean buildserver
INFO: prepare_buildserver
```

b. Then run **./build.sh** script can start compiling now

```
test@test:~/t527_android13/longan$ ./build.sh
```

c. After compilation, you will see the following output



```
sun55iw3p1 compile all(Kernel+modules+boot.img) successful
```

```
INFO: build dts ...
```

```
INFO: Prepare toolchain ...
```

```
Setup BSP files
```

```
'/home/test/a527_android13/longan/out/t527/kernel/staging/sunxi.dtb' ->
```

```
'/home/test/a527_android13/longan/out/t527/demo/android/sunxi.dtb'
```

```
INFO: build rootfs ...
```

```
INFO: skip make rootfs for android
```

```
INFO: -----
```

```
INFO: build Tina OK.
```

```
INFO: -----
```

3) Then use the following command to compile the Android source code and generate the final Android image

```
test@test:~$ cd t527_android13
```

```
test@test:~/t527_android13$ source build/envsetup.sh
```

```
test@test:~/t527_android13$ lunch t527_demo_arm_go-userdebug
```

```
test@test:~/t527_android13$ make -j8
```

```
test@test:~/t527_android13$ pack
```

4) The storage path of the compiled Android image is:

```
longan/out/t527_android13_demo_uart0.img
```



7. Appendix

7.1. User Manual Update History

edition	Date	Update Explanation
v1.0	2024-11-14	Initial version
v1.1	2025-03-26	1. Compilation method of Android 13 source code 2. Method for burning Linux images to SPIFlash+NVMe SSD 3. Instructions for using the system backup script opi-bking 4. How to use Linux Overlayboot

7.2. Image update history

Date	Update Explanation
2024-11-14	OrangePi4A_T527_Android13_v1.0.0.tar.gz OrangePi4A_T527_Android13_lcd_v1.0.0.tar.gz Orangepi4a_1.0.0_ubuntu_jammy_server_linux5.15.147.7z Orangepi4a_1.0.0_debian_bookworm_server_linux5.15.147.7z Orangepi4a_1.0.0_ubuntu_jammy_desktop_gnome_linux5.15.147.7z Orangepi4a_1.0.0_debian_bookworm_desktop_gnome_linux5.15.147.7z *Initial version
2024-11-26	Orangepi4a_1.0.2_ubuntu_jammy_desktop_xfce_linux5.15.147.7z Orangepi4a_1.0.2_ubuntu_jammy_desktop_gnome_linux5.15.147.7z Orangepi4a_1.0.2_debian_bookworm_desktop_gnome_linux5.15.147.7z * Supports GPU acceleration and Gstreamer hard decoding, and can use the gst-play-1.0 command to hard decode and play videos



2025-03-26	OrangePi4a_1.0.4_ubuntu_jammy_server_linux5.15.147.7z OrangePi4a_1.0.4_debian_bookworm_server_linux5.15.147.7z OrangePi4a_1.0.4_ubuntu_jammy_desktop_xfce_linux5.15.147.7z OrangePi4a_1.0.4_ubuntu_jammy_desktop_gnome_linux5.15.147.7z OrangePi4a_1.0.4_debian_bookworm_desktop_gnome_linux5.15.147.7z * Kernel updated to Allwinner bsp v1.4 version * Switch boot mode from extLinux to boot.scr * Support SPIFlash+NVMe SSD startup * Add system backup script opi-bking * Support configuring overlay read-only file system through orangepi config
------------	---

FCC Caution: Any changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate this equipment.

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

This device and its antenna(s) must not be co-located or operating in conjunction with any other antenna or transmitter.

NOTE: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

To maintain compliance with FCC's RF Exposure guidelines, This equipment should be installed and operated with minimum distance between 20cm the radiator your body: Use only the supplied antenna.