

*Wish to be the cornerstone of your
success!*

Baijie Technology

HelperBoard A733 development board

User-manual

V1.3

2026-06

Shenzhen Baijie Technology Co. Ltd

Address: No. 602, Building 1A, Feiyada Watch Building, Zhonghua Base, Ma Tian Street,
Guangming District, Shenzhen City

www.szbaijie.comwww.szbaijie.cn

Revision record

Data	Version	Revised Notes	Author
2025.08.11	V1.0	Creat	Duan Jingcheng
2026.06.01	V1.1	Add screen connection diagram	Duan Jingcheng
2026.06.12	V1.2	Add instructions for rotating the screen in the Ubuntu system	Duan Jingcheng
2026.06.15	V1.3	Add EDP screen	Duan Jingcheng
2026.06.18	V1.3	Add test instructions for PCIe (NVMe) interface	Duan Jingcheng

Directory

1 Overview Introduction to the HelperBoard A733 Development Board	9
1.1 Introduction	9
1.2 System Features	9
1.3 Hardware Resource Description	10
1.4 Hardware Dimensions diagram	12
1.5 Screen Connection diagram	13
2 Terminal Debugging	18
2.1 Downloading Web Disk Resources	18
2.2 Installation and Use of Xshell	18
2.2.1 Installation of Xshell	18
2.2.2 Serial port connection of the development board	23
2.2.3 Connect to the development board using Xshell	25
2.3 adb Debugging	28
2.3.1 Basic Use of adb	28
2.3.2 Install APK using adb	30
3 Hardware Function Test	32
3.1 Function testing of the Ubuntu system on the development board	32
3.1.1 Development board Ubuntu account and password	32
3.1.2 485 Test	32
3.1.3 UART serial port test	35
3.1.4 Network port test	36

3.1.5 USB-HOST port test.....	37
3.1.6 Camera test.....	37
3.1.7 TF card interface test.....	38
3.1.8 Microphone, headphones, speaker test.....	38
3.1.9 Buzzer test.....	39
3.1.10 Touchscreen test.....	39
3.1.11 WIFI test.....	40
3.1.12 4G module test.....	40
3.1.13 Bluetooth test.....	41
3.1.14 Backlight adjustment.....	45
3.1.15 PCIE port test.....	45
3.2 Function testing of the Android system on the development board.....	47
3.2.1 TF card interface test.....	47
3.2.2 Microphone, headphones, speaker test.....	48
3.2.3 WIFI test.....	49
3.2.4 Bluetooth test.....	49
3.2.5 USB-OTG port test.....	50
3.2.6 Camera test.....	52
4 Use of software tools.....	53
4.1 VMware development environment Setup.....	53
4.1.1 Hardware Requirements.....	53
4.1.2 Software Requirements.....	53

4.2 Install PhoenixSuit	56
4.3 Flash the firmware of the development board	58
4.4 PhoenixCard produces SD card burning discs	62
4.5 If the card write fails, use SDFormatter to format the memory card	65
4.6 The use of Dragonface	69
4.6.1 Change the Android startup logo	70
4.6.2 Change the opening animation	72
4.6.3 Add built-in APK	73
5 Creating the firmware for the Ubuntu system	75
5.1 Compile the Linux kernel	75
5.2 Compile u-boot	79
5.3 Compile and package the Ubuntu + Qt file system	80
5.4 Compile and package the Ubuntu + Xfce file system	80
5.5 Notes for Attention	80
6 Remote running and debugging of Qt programs	82
6.1 Copy the compiler to the virtual machine	82
6.2 Configuration of the QtCreator environment in the virtual machine	82
6.2.1 Add remote device	82
6.2.2 Set up the compiler	88
6.2.3 Set the Qt version	89
6.2.4 Set kits	90
6.2.5 Create a new Qt project	91

6.2.6 Set the path for the running program.....	92
6.2.7 Operation and Debugging.....	93
6.3 Using Cmake for cross-compilation in Qtcreator.....	94
6.4 Set the development board to start up automatically with Qt.....	98
6.4.1 Start up automatically on the already burned board.....	98
6.4.2 Set up self-starting in the firmware.....	99
7 Creating the firmware for the Android system.....	100
7.1 Compile the Linux kernel.....	100
7.2 Compile u-boot.....	102
7.3 Compilation and packaging of Android.....	102
8 Other operations.....	104
8.1 Modify the startup logo of Android.....	104
8.2 Modify the startup animation and startup music of Android.....	105
8.2.1 Directory Structure Explanation.....	105
8.2.2 Boot-up music.....	106
8.2.3 Packaging and Compilation.....	107
8.3 Modify screen rotation in Android.....	108
8.4 Set the static IP address for the network card in Ubuntu.....	109
8.5 Ubuntu controls Wi-Fi.....	110
8.6 Ubuntu root file system backup.....	112
8.7 Modify the boot logo of Ubuntu.....	114
8.8 Modify screen rotation in Ubuntu.....	115

8.8.1 Qt system modification	115
8.8.2 Xfce System Modification	115
8.9 Patch the defect	115
8.10 Kernel configuration	117
9 Regarding customized services	119

1 Overview Introduction to the HelperBoard A733 Development Board

1.1 Introduction

The HelperBoard A733 development board (**hereinafter referred to as HelperA733**) is the seventh model in the HelperBoard series of development boards produced by Baijie Technology. It inherits all the excellent features of HelperA523, being simple, elegant, stable and efficient, and is suitable for more advanced application requirements.

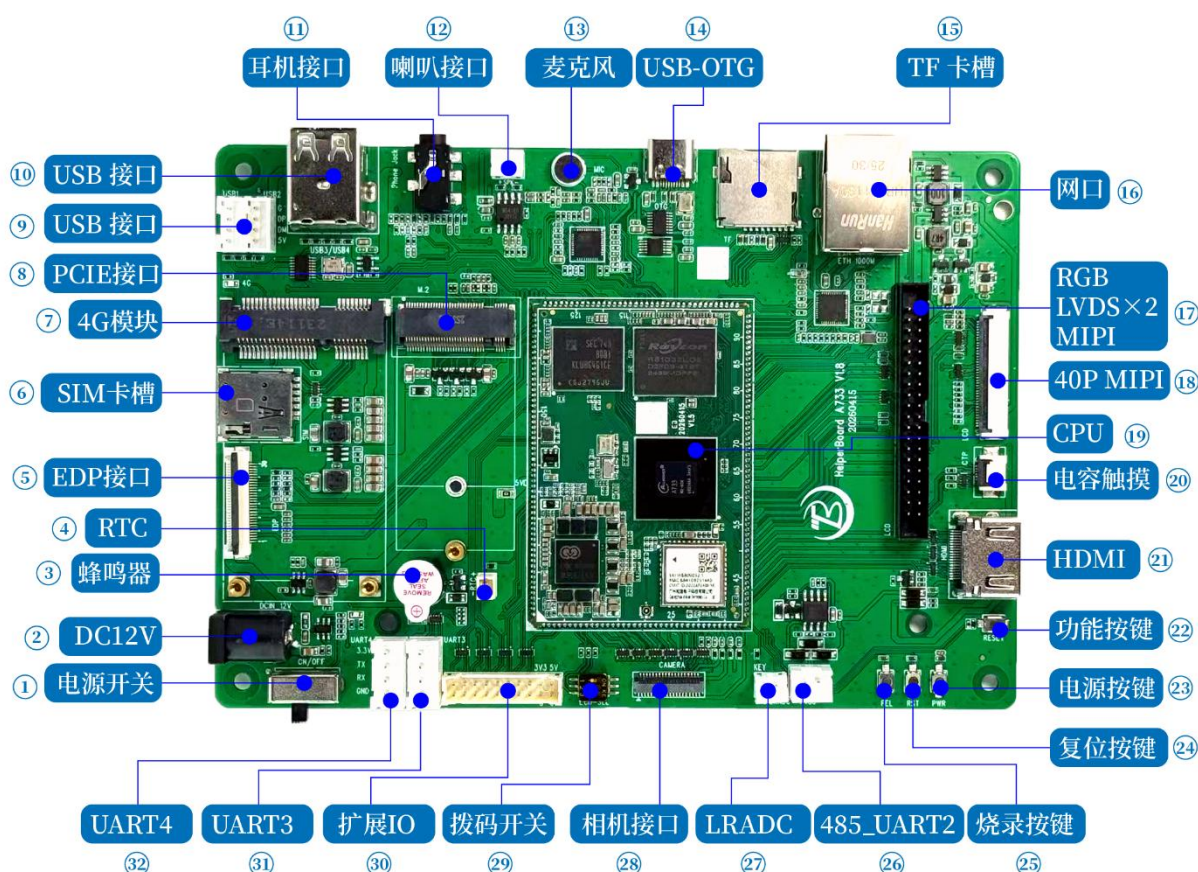
1.2 System Features

- ◆Processor: Allwinner A733
- ◆CPU: 64-bit Dual-Core Cortex-A76 + Hexa-Core Cortex-A55, up to 2.0 GHz
- ◆NPU: 3TOPS
- ◆Memory: 4 GB / 6 GB LPDDR5 (up to 16 GB)
- ◆Storage: 64 GB UFS 2.2 (up to 1 TB, optional UFS 3.0)
- ◆GPU: IMG BXM-4-64 MC1
- ◆Video Engine: H.265 MP/VP9/AVS2 decoder 8K@24fps、H.264 BL/MP/HP decoder 4K@30fps、JPEG decoder 1080P@60fps、H.264/H.265 encoder 4K@30fps、MJPEG/JPEG encoder 4K@15fps、JPEG encoder 4K@15fps
- ◆Video Decoding: 8K@24fps H.265、4K@30fps H.264
- ◆Display: 40-pin multiplexed MIPI DSI / Dual LVDS / RGB interface, HDMI, eDP
- ◆Ethernet: 10/100/1000 Mbps Gigabit Ethernet
- ◆WIFI/BT: Dual-band 5 GHz Wi-Fi and Bluetooth 5.0
- ◆Cellular Module: 4G LTE (EC20 module)
- ◆USB: Four High-Speed USB 2.0 Host ports and one High-Speed USB OTG port
- ◆USB Type-C: One USB Type-C port with integrated ADB and UART0 debug console
- ◆TF Card Slot: One TF (microSD) card slot
- ◆Microphone: One MIC input
- ◆Audio Output: One 3.5 mm headphone jack
- ◆Speaker: One speaker output
- ◆MIPI Display Interface: One MIPI DSI display interface
- ◆Capacitive Touch Interface: One capacitive touch interface
- ◆eDP Interface: One eDP interface
- ◆PCIe Interface: One PCIe 3.0 interface
- ◆Camera Interface: One dual-camera MIPI CSI interface, supporting image sensors up to 13 MP
- ◆Buzzer: One onboard buzzer

页

- ◆Expansion Interfaces: 2 × UART, 1 × RS-485, 1 × 3.3 V power output, 1 × 5 V power output, and 1 × LRADC interface
- ◆Buttons: One Power button (PWR), one Reset button (RST), one Firmware Flashing button (FEL), and one User Function button
- ◆RTC Connector: One external RTC battery connector
- ◆Power Input: 9–12 V DC input
- ◆Operating Systems: Android 15, Ubuntu + Qt, and Ubuntu + XFCE

1.3 Hardware Resource Description



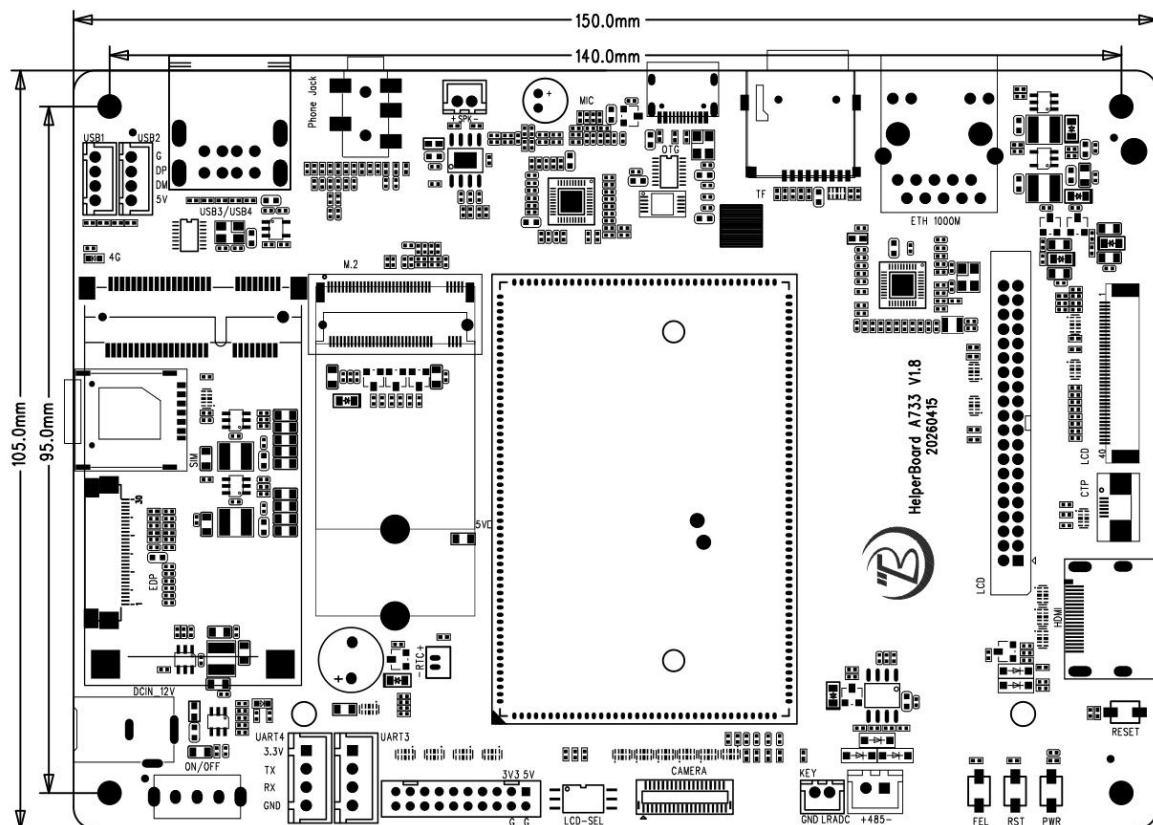
Note: This document applies to the HelperBoard A733 V1.8 baseboard hardware version.

Location	Description
①	Power Switch
②	DC 12 V Power Input
③	Buzzer
④	RTC Battery Connector
⑤	30-pin eDP Connector
⑥	SIM Card Slot
⑦	4G LTE Module
⑧	PCIe 3.0 Connector
⑨	USB 2.0 Expansion Header
⑩	USB 2.0 Host Port
⑪	Headphone
⑫	Speaker
⑬	Microphone
⑭	USB OTG Port (Supports Debug Console and Firmware Flashing)
⑮	TF (microSD) Card Slot
⑯	Gigabit Ethernet Port
⑰	40-pin LCD Connector (Multiplexed Dual LVDS / MIPI DSI / RGB Interface, Refer to the Schematic for Pin Definitions)
⑱	40-pin MIPI CSI Connector
⑲	Allwinner A733 Processor
⑳	Capacitive Touch Connector
㉑	HDMI Output

页

②②	User Function Button
②③	Power Button
②④	Reset Button
②⑤	FEL Button (Firmware Flashing)
②⑥	RS-485 Interface (UART2)
②⑦	LRADC Interface
②⑧	MIPI CSI Camera Connector
②⑨	DIP Switch
③⑩	Expansion GPIO Header
③①	UART3
③②	UART4

1.4 Hardware Dimensions diagram



1.5 Screen Connection diagram

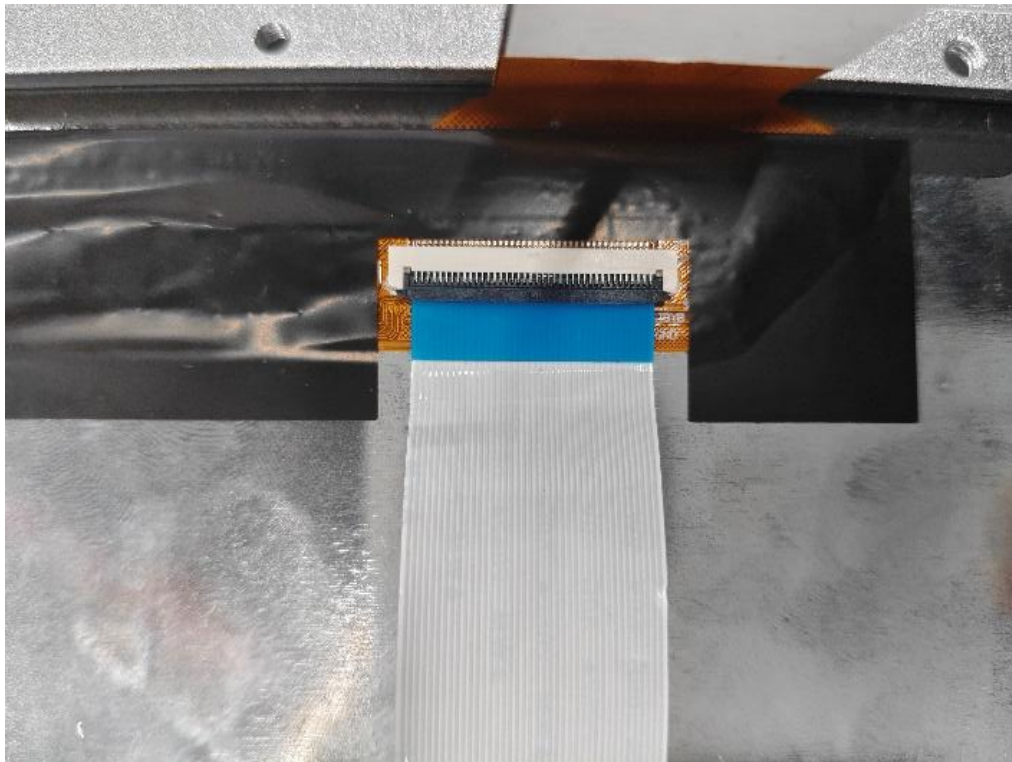
(1) LVDS 7.0 1024x600 Screen Connection Diagram



(2) Connection Diagrams for MIPI 10.1 800x1280 and MIPI 10.1 1200x1920 Displays



(3) MIPI 10-inch Screen Rear Connection Diagram



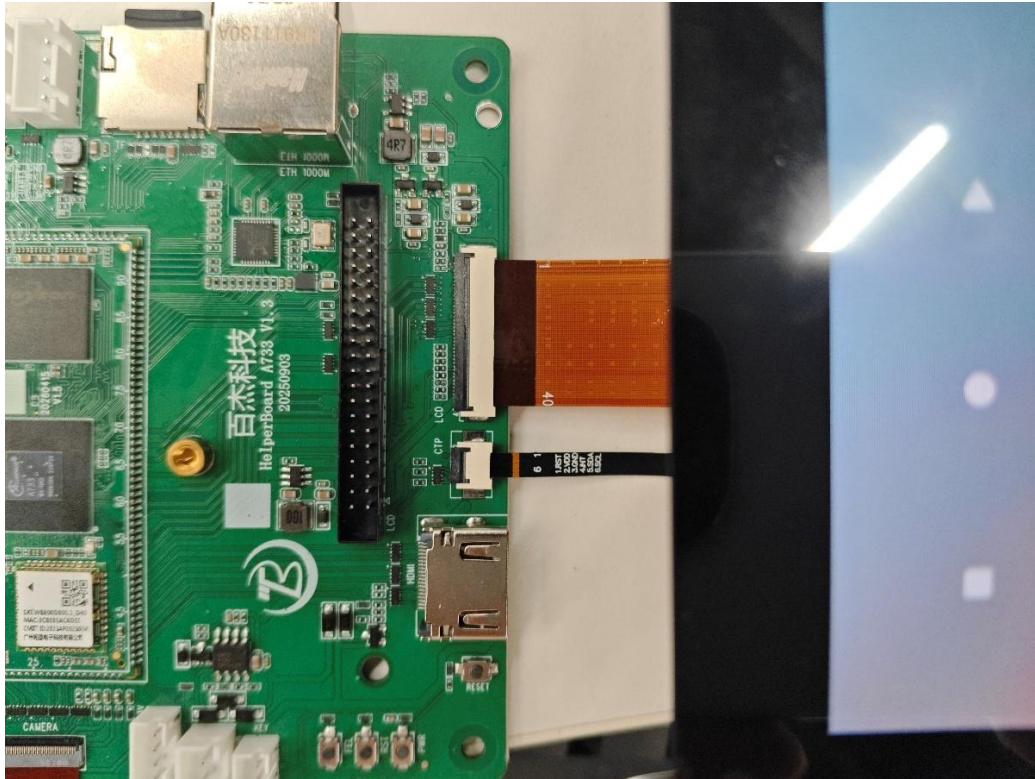
(4) MIPI 4.3 480x800 Display Connection Diagram



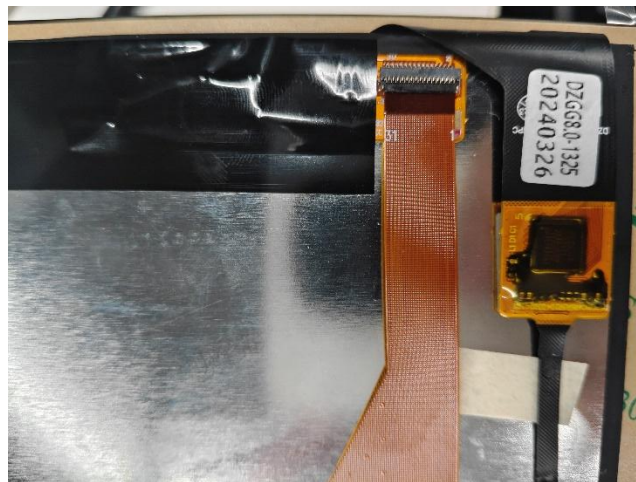
(5) MIPI 5.0 720x1280 Screen Connection Diagram



(6) MIPI8.0_800x1280 Screen Connection Diagram



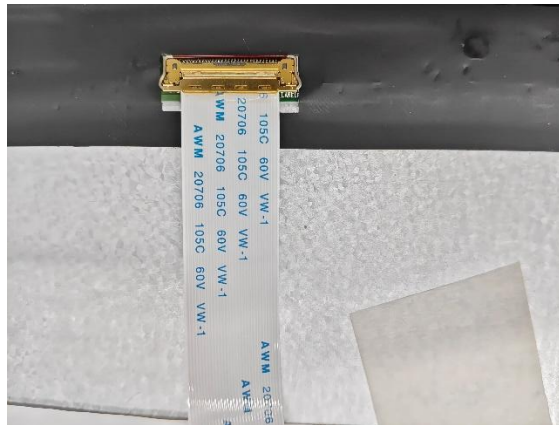
(7) MIPI8.0_800x1280 Screen Rear Connection Diagram



(8) EDP Screen Connection Diagram



(9) EDP Screen Rear Connection Diagram

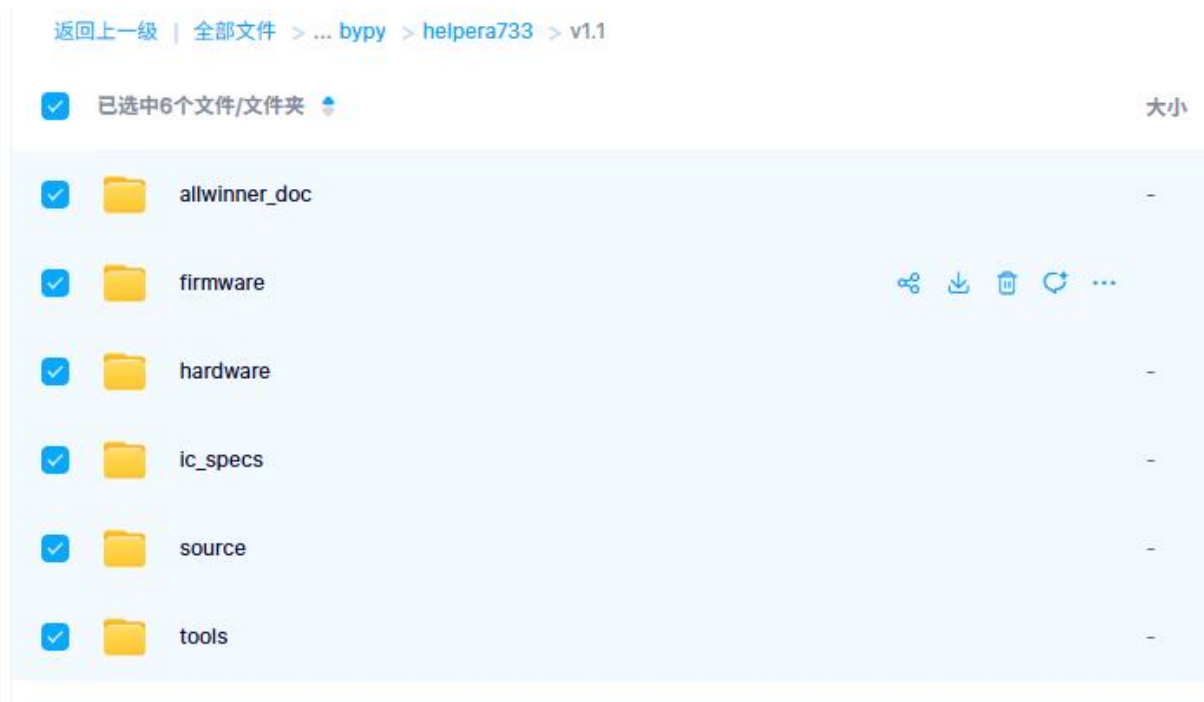


2 Terminal Debugging

2.1 Downloading Web Disk Resources

All software, hardware design files, source code, and firmware packages referenced in this user manual are available from the Baidu Netdisk. Please contact our customer service team to obtain the download link.

The directory structure of the Baidu Netdisk is shown below:



2.2 Installation and Use of Xshell

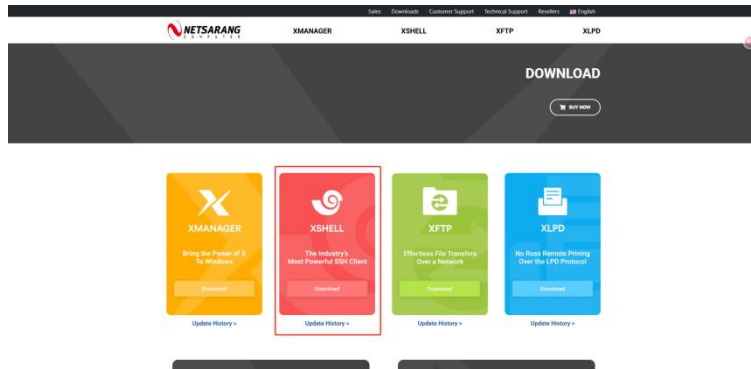
2.2.1 Installation of Xshell

Before performing any tests, install **Xshell** to access the development board through a serial terminal. The installation procedure is as follows:

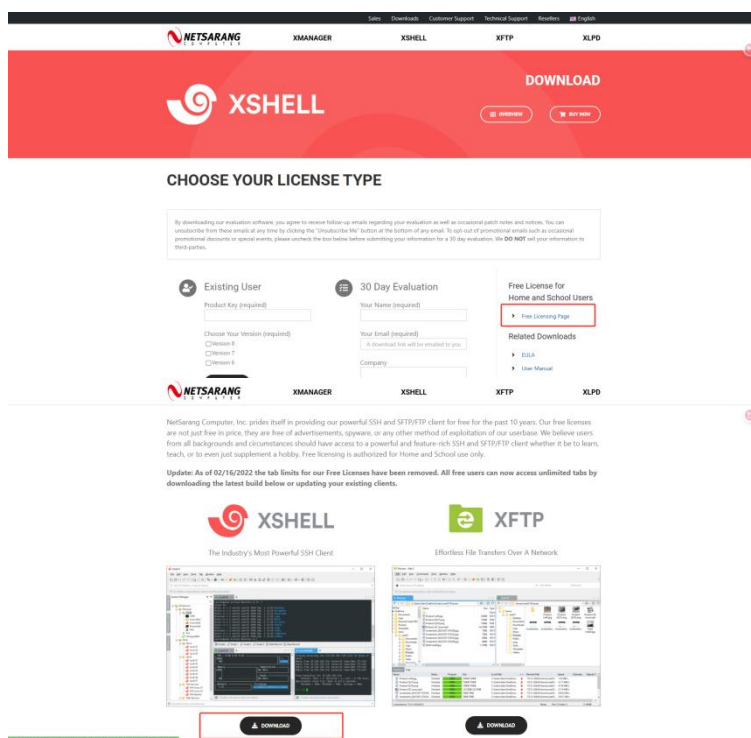
- 1、Visit the Xshell official website: <https://www.xshell.com/>

2、Click **DOWNLOAD** to open the download page.

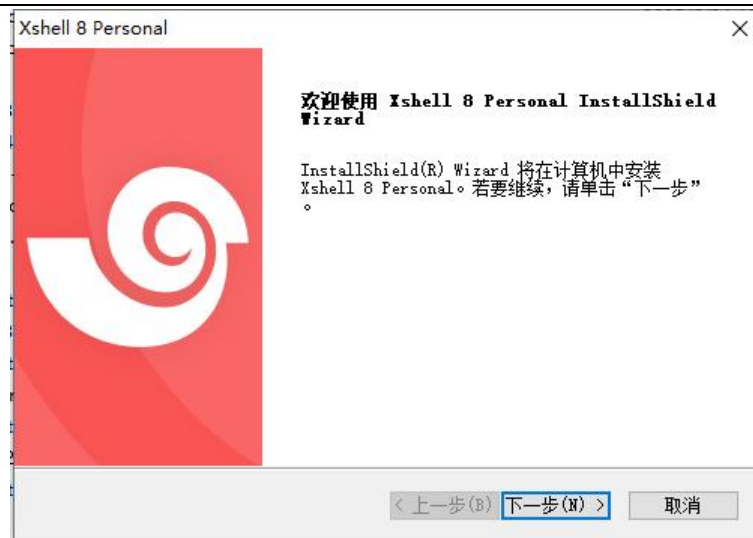
3、Select **Xshell** and download.



4、Select the **Free License** (or **Free for Home/School** if available) and download the installation package.



5、After the download is complete, double-click the **.exe** installation file to launch the installation wizard, and then click **Next**.



Select **I accept the terms of the License Agreement**, and then click **Next** to continue.



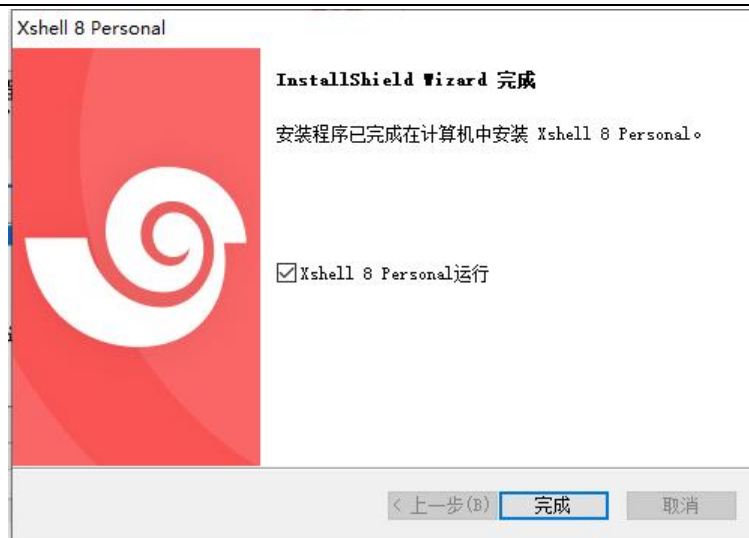
Choose the installation directory, and then click **Next**. The default installation path is shown below.



Click **Install** to begin the installation.



Click **Finish** to complete the installation. Xshell will launch automatically.



When launching Xshell for the first time, you will be prompted to activate the **Free License**. Enter a valid email address and submit the request to complete the registration.



After completing the registration through the link in the confirmation email, restart Xshell. Xshell is now ready for use.



感谢您对 NetSarang 免费许可的关注。

单击下面的链接以验证您的电子邮件地址并完成注册。

<https://api.netsarangapi.com/f/c/ZM7g06fjKwvlev0ceJEt5qlvl1RIqZKSDeL1UwH4BudCOHjmaRau89CfjP0LiHAKJVC26ox6x1RWgdfngtnGxilmQHSWm1Qb6bVpVHz7S1tqZtnY3LE5nxHv2Cr7aD%2Bgr8or%2B6qNppTQGP7QytRCW%2Bxl4IC5kPn9pDmiN%2BwzCyOCeh6P39A6ddeSyM3G0x24bdTUTNmn%2BnvwlFZrXsjQs2WxbTBKVapSd0kLFA%3D%3D>

如果您无法单击该链接，请将链接复制并粘贴到您的网络浏览器中。

如果您有任何问题，请通过 support@xshell.com 与我们联系或直接回复此电子邮件。

感谢您使用 NetSarang 软件！

6、Extension Software

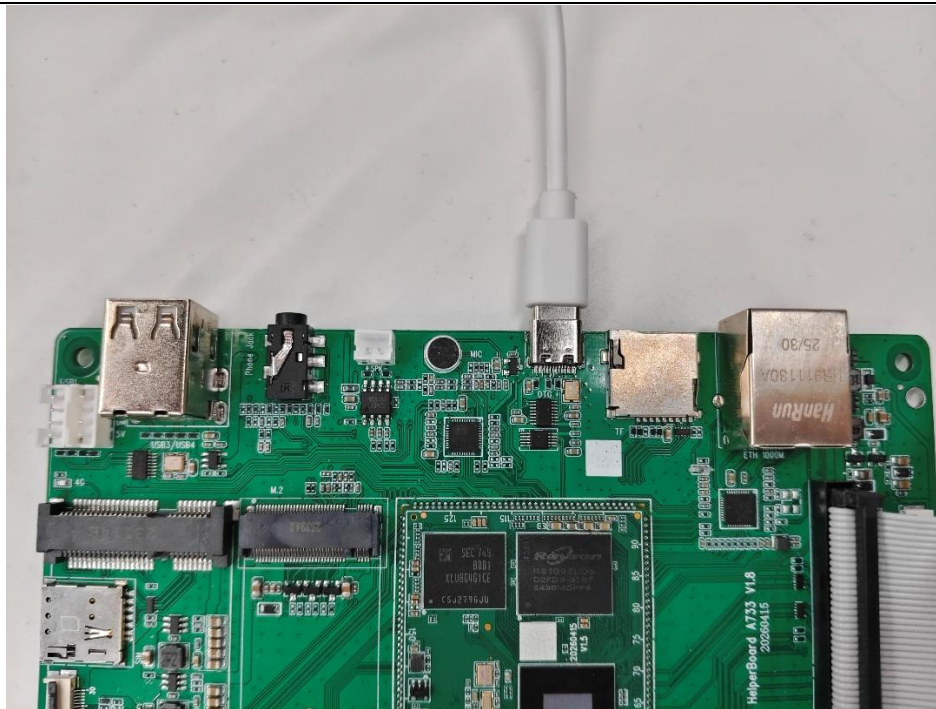
Xshell is only one of the supported serial terminal tools. Users may choose any serial terminal application according to their preference, such as **MobaXterm**, **PuTTY**, or **Tera Term**.

Due to space limitations, the installation procedures for these tools are not covered in this manual. Please refer to their official documentation or online installation guides for detailed instructions.

2.2.2 Serial port connection of the development board

1、Preparation

Connect the USB cable to the ⑭ USB OTG port described in Section 1.3, as shown in the figure below.

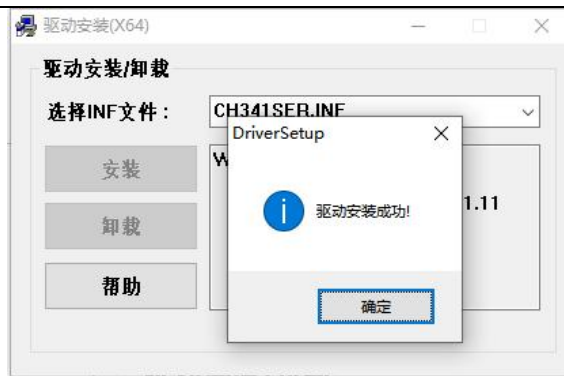


2、Install the USB-to-Serial Driver

Locate the **CH340 Driver.exe** installer in the following directory of the Baidu Netdisk resources, the download and install the driver.

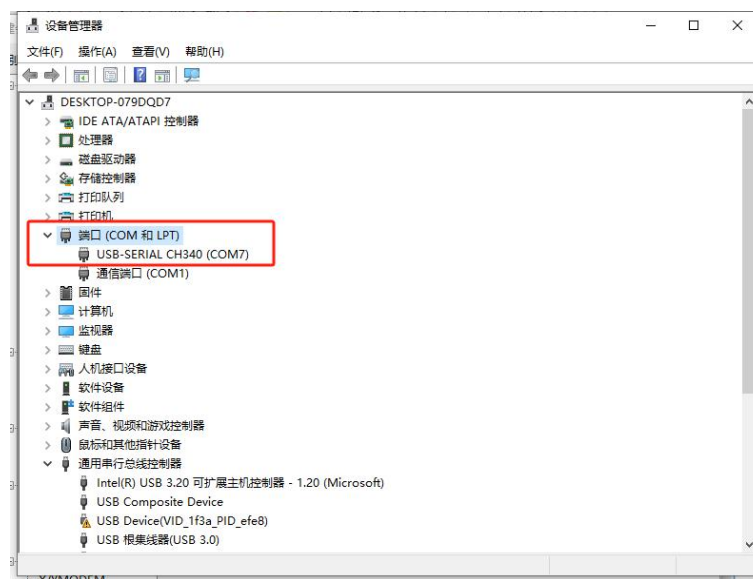
`tools/usb2 serial_driver/CH340 Driver.exe`





2.2.3 Connect to the development board using Xshell

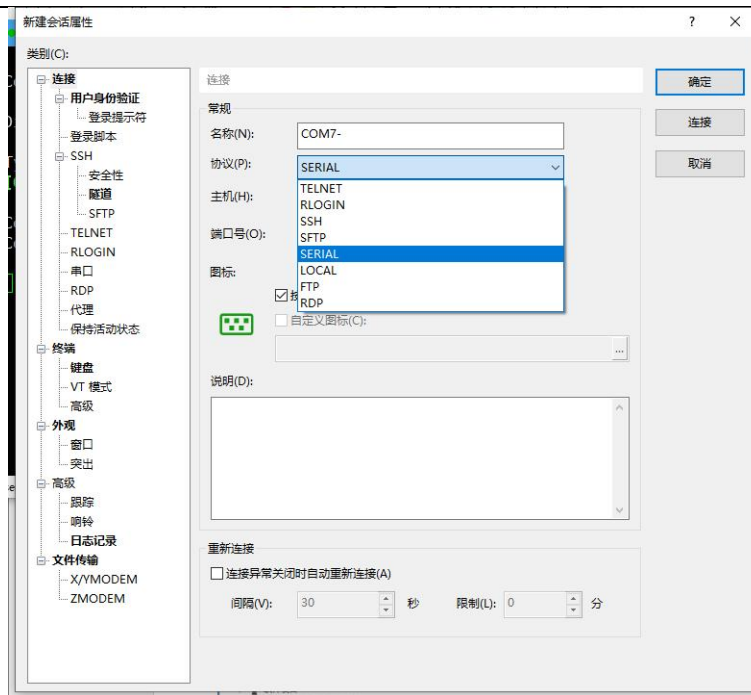
Connect the development board's **USB OTG** port to a USB port on the host PC using a USB cable. Then open **Device Manager** and verify that a new **COM** port has been detected. In this manual, the communication port assigned to the development board is **COM7**, and all examples use **COM7** for illustration. If your system assigns a different COM port number, replace **COM7** with the corresponding port number in the following steps.



After installing **Xshell**, launch the application and select **File** → **New**.

On the **Connection** page, set the **Name** field to **COM7**.

Under the **Protocol** category, select **SERIAL**.



After selecting the protocol, click **Serial** under the **Category** pane.

Configure the serial port parameters as follows:

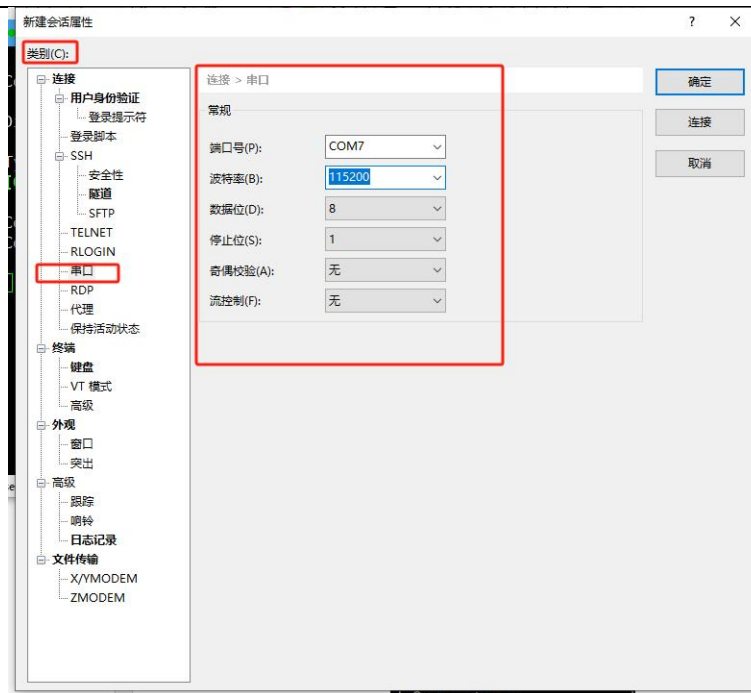
Port: Select the COM port assigned to the development board (for example, **COM7**).

Baud Rate: 115200

Data Bits: 8

Stop Bits: 1

The last two items are "No".



After configuring the serial port settings, click **Connect**. Then restart the development board. If the connection is successful, the boot log will be displayed in the Xshell terminal, as shown in the figure below. You can now access and control the development board through the **serial console**.

```
COM7- - Xshell 8 (Free for Home/School)
文件(F) 编辑(E) 查看(V) 工具(T) 选项卡(B) 窗口(W) 帮助(H)
COM7- x
Starting Modem Manager...
Starting OpenBSD Secure Shell server...
Starting Hostname Service...
Starting Permit User Sessions...
[ OK ] Started Dispatcher daemon for systemd-networkd.
[ OK ] Finished Permit User Sessions.
[ OK ] Started Hostname Service.
[ OK ] Started Modem Manager.
[ OK ] Started OpenBSD Secure Shell server.
Starting Network Manager Script Dispatcher Service...
[ OK ] Started Serial Getty on ttyS0.
[ OK ] Created slice Slice /system/getty.
[ OK ] Reached target Login Prompts.
Starting Load/Save RF Kill Switch Status...
[ OK ] Started Load/Save RF Kill Switch Status.
[ OK ] Started Network Manager Script Dispatcher Service.
[ OK ] Finished Set console font and keymap.
Starting Save/Restore Sound Card State...
[ OK ] Finished Save/Restore Sound Card State.
[ OK ] Reached target Sound Card.
[ OK ] Started rcS-Service.
[ OK ] Reached target Multi-User System.
[ OK ] Reached target Graphical Interface.
Starting Record Runlevel Change in UTMP...
[ OK ] Finished Record Runlevel Change in UTMP.

Ubuntu 22.04.4 LTS HelperA733 ttyS0

HelperA733 login: root
Password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.6.77 aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Wed Nov 22 04:57:35 CST 2023 on ttyS0
COLUMNS=189;LINES=43;export COLUMNS LINES;
root@HelperA733:~#
```

2.3 adb Debugging

Connect the development board to the host PC using a USB data cable through the ⑭ USB OTG port shown in Section 1.3

2.3.1 Basic Use of adb

On the host PC, press **Win + R** to open the **Run** dialog box. Enter `cmd`, and then press **Enter** to open the Command Prompt.

Run the following command to check the installed ADB version:

```
$ adb version
```

```
C:\Windows\system32\cmd.exe
C:\Users\BJ42>adb version
Android Debug Bridge version 1.0.41
Version 35.0.1-11580240
Installed as D:\platform-tools\adb.exe
Running on Windows 10.0.19045

C:\Users\BJ42>
```

If ADB is not installed on your PC, download [adb_V1.0.41.zip](#) from the following directory in the Baidu Netdisk resources:

[tools/adb/adb_V1.0.41.zip](#)

Extract the archive. In the extracted directory (or in any directory if the ADB executable has been added to the system **PATH**), hold **Shift** and right-click in the blank area of the folder, then select **Open PowerShell window here**.

On the A733 system, run the following command to enable the ADB service (enabled by default in the factory firmware):

```
$ cat /sys/devices/platform/soc@30000000/10.usbc0/usb_device
```

If the following output is displayed, the ADB service has been enabled successfully.

```
root@HelperA733:/# cat /sys/devices/platform/soc@30000000/10.usbc0/usb_device
device_chose finished, otg disabled!
root@HelperA733:/#
```

Next, run the following command in the **Command Prompt** to list the connected ADB devices:

```
$ adb devices
```

```
C:\Users\BJ42>adb devices
List of devices attached
7c0016117200c7c1bcc    device

C:\Users\BJ42>
```

Run the following command to connect to the development board. If the connection is successful, the device status will be displayed as shown below.

```
$ adb shell

C:\Users\BJ42>adb shell
# ls
bin    dev    init    media  proc    run    sys    usr
boot  etc    lib     mnt    qt_env.sh sbin  system var
data  home  lost+found opt     root    srv    tmp    vendor
#
```

For more information about ADB commands, refer to the official Android Developer documentation or other online ADB tutorials.

2.3.2 Install APK using adb

Once the device is connected, you can install an APK. Ensure that the directory containing the APK does **not** include any folders with Chinese characters; otherwise, the installation may fail.

Run the following command to install the APK:

```
$ adb install apk 路径和文件名
```

If the APK is installed successfully, the output should be similar to the following:

```
C:\Users\BJ42>adb install E:\tools\antutu-benchmark-v10.apk
Performing Streamed Install
Success
```

If the `adb devices` command lists multiple devices, use the `-s` option to specify the target device by its serial number.

For example:

页

```
C:\Users\BJ42>adb devices
List of devices attached
6c000c71b504c891e1d    device
7c0016117200c7c1bcc    device

C:\Users\BJ42>adb -s 7c0016117200c7c1bcc install E:\tools\antutu-benchmark-v10.apk
Performing Streamed Install
Success

C:\Users\BJ42>
```

3 Hardware Function Test

The **HelperA733** development board is preloaded with the **Ubuntu** operating system at the factory. The board can be powered by a DC power supply. After power-on, you can perform hardware function tests to verify that all onboard peripherals are operating correctly.

Powering the Board via DC Power Supply:

The ② **DC power input** is shown in **Section 1.3**. Connect the supplied **12 V power adapter** to the DC power input, and then turn on the ① **Power** switch shown in **Section 1.3** to start the system.

3.1 Function testing of the Ubuntu system on the development board

3.1.1 Development board Ubuntu account and password

The default user accounts for the HelperA733 development board are as follows:

Root (superuser) account: **root**

Standard user account: **szbaijie**

Default password (for both accounts): **szbaijie**

3.1.2 485 Test

The ②⑥ interface shown in Figure 1.3 is the **485_UART2 interface**. To test the RS485 function, two RS485 interfaces are required to be connected together. A single interface cannot be tested independently. This test is performed with **Development Board A and Development Board B connected to each other**. Both Development Board A and Development Board B need to be connected to **Xshell**. **Note:** For Xshell usage instructions, please refer to **Section 2.2**.

The RS485 communication test uses the **minicom** software. Both development boards need to have **minicom** installed. The installation command is as follows:

```
$ apt-get install minicom
```

After the installation is completed, enter the Minicom configuration interface using the following command:

```
$ minicom -s
```

```
+-----[configuration]-----+
| Filenames and paths          |
| File transfer protocols      |
| Serial port setup            |
| Modem and dialing            |
| Screen and keyboard          |
| Save setup as dfl             |
| Save setup as..              |
| Exit                         |
| Exit from Minicom            |
+-----+-----+

```

Select **Serial port setup** and press **Enter** to enter the serial port configuration interface. Press **"A"** to configure the communication port and set the port to **/dev/ttyS2**. Then press **"F"** to disable **Hardware Flow Control** by setting it to **"No"**. The final configuration is shown in the figure below:

```
+-----+-----+
| A - Serial Device           : /dev/ttyS2
| B - Lockfile Location       : /var/lock
| C - Callin Program          :
| D - Callout Program         :
| E - Bps/Par/Bits            : 115200 8N1
| F - Hardware Flow Control   : No
| G - Software Flow Control   : No
| H - RS485 Enable            : No
| I - RS485 Rts On Send       : No
| J - RS485 Rts After Send    : No
| K - RS485 Rx During Tx      : No
| L - RS485 Terminate Bus     : No
| M - RS485 Delay Rts Before: 0
| N - RS485 Delay Rts After  : 0
|
| Change which setting?
+-----+-----+

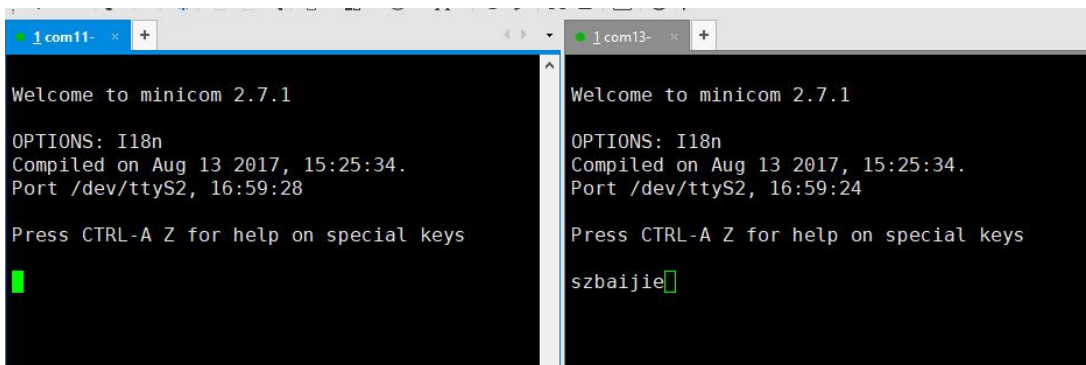
```

After completing the configuration, press **Enter** to exit the serial port configuration interface. Select **Save setup as dfl** to save the current settings as the default configuration, as shown below:

```
+-----[configuration]-----+
| Filenames and paths          |
| File transfer protocols      |
| Serial port setup           |
| Modem and dialing            |
| Screen and keyboard         |
| Save setup as dfl            |
| Save setup as ..             |
| Exit                         |
| Exit from Minicom           |
+-----+-----+-----+-----+
```

Select **Exit** to leave the configuration mode and enter the serial communication interface. Enter **"szbaijie"** on Development Board A. The same message should be displayed on Development Board B. Similarly, enter **"szbaijie"** on Development Board B, and the message should be displayed on Development Board A.

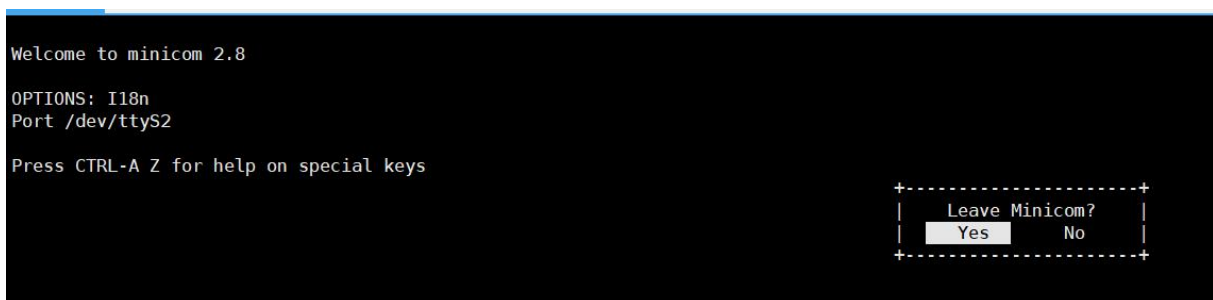
If the data can be transmitted and received successfully between the two development boards, the RS485 communication test is completed successfully.



```
1 com11- x +
Welcome to minicom 2.7.1
OPTIONS: I18n
Compiled on Aug 13 2017, 15:25:34.
Port /dev/ttyS2, 16:59:28
Press CTRL-A Z for help on special keys
█

1 com13- x +
Welcome to minicom 2.7.1
OPTIONS: I18n
Compiled on Aug 13 2017, 15:25:34.
Port /dev/ttyS2, 16:59:24
Press CTRL-A Z for help on special keys
szbaijie█
```

To exit Minicom, press **"Ctrl+A"** first, then press **"X"**. Select **"Yes"** in the confirmation dialog to exit Minicom.



```
Welcome to minicom 2.8
OPTIONS: I18n
Port /dev/ttyS2
Press CTRL-A Z for help on special keys

+-----+-----+-----+-----+
| Leave Minicom? |
| Yes No |
+-----+-----+-----+-----+
```

3.1.3 UART serial port test

The interfaces shown in Figure 1.3 are ③① **UART3** and ③② **UART4**. The UART test requires two development boards to be connected together to verify data transmission and reception. The UART connection is as follows: GND — GND RX — TX TX — RX. The **RX** and **TX** lines must be cross-connected.

Taking **UART3** as an example, the test procedure is as follows:

1. Test Development Board 1 for data reception and Development Board 2 for data transmission. The device node generated by UART3 is **ttyS3**, so both data transmission and reception use **ttyS3**.

Run the following command on **Development Board 1** to wait for incoming data:

```
$ cat /dev/ttyS3
```

```
root@HelperA733:~# cat /dev/ttyS3
```

Run the following command on **Development Board 2** to send data:

```
$ echo szbaijie >/dev/ttyS3
```

```
root@HelperA733:~# echo szbaijie > /dev/ttyS3
root@HelperA733:~#
```

After executing the command, Development Board 1 will receive the data sent from Development Board 2.

```
root@HelperA733:~# cat /dev/ttyS3
szbaijie
```

2. Test Development Board 1 for data transmission and Development Board 2 for data reception.

Run the first command mentioned above on **Development Board 2** to wait for data reception, and then run the second command mentioned above on **Development Board 1** to send data.

3.1.4 Network port test

The interface shown in Figure 1.3 is the ⑩ **Gigabit Ethernet interface**. Use a standard Ethernet cable to connect the Ethernet ports of the development boards. First, check whether the development board has obtained an IP address. Run the following command:

```
$ ifconfig
```

```
root@HelperA733:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.140 netmask 255.255.254.0 broadcast 192.168.1.255
    inet6 fe80::3672:c004:88d5:bc82 prefixlen 64 scopeid 0x20<link>
    ether d4:bf:48:ac:9a:57 txqueuelen 1000 (Ethernet)
    RX packets 46496 bytes 3746531 (3.7 MB)
    RX errors 0 dropped 541 overruns 0 frame 0
    TX packets 746 bytes 84023 (84.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 7 base 0x8000
```

Run the **ping** command on the development board. If data packets are successfully transmitted and received, the Ethernet interface and network communication test are completed successfully.

```
$ ping www.baidu.com
```

```
root@HelperA733:~# ping www.baidu.com
PING www.a.shifen.com (183.2.172.17) 56(84) bytes of data:
64 bytes from 183.2.172.17 (183.2.172.17): icmp_seq=1 ttl=54 time=8.27 ms
64 bytes from 183.2.172.17 (183.2.172.17): icmp_seq=2 ttl=54 time=8.29 ms
64 bytes from 183.2.172.17 (183.2.172.17): icmp_seq=3 ttl=54 time=8.36 ms
64 bytes from 183.2.172.17 (183.2.172.17): icmp_seq=4 ttl=54 time=8.24 ms
64 bytes from 183.2.172.17 (183.2.172.17): icmp_seq=5 ttl=54 time=8.43 ms
64 bytes from 183.2.172.17 (183.2.172.17): icmp_seq=6 ttl=54 time=8.20 ms
64 bytes from 183.2.172.17 (183.2.172.17): icmp_seq=7 ttl=54 time=8.36 ms
64 bytes from 183.2.172.17 (183.2.172.17): icmp_seq=8 ttl=54 time=8.16 ms
^C
--- www.a.shifen.com ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7009ms
rtt min/avg/max/mdev = 8.155/8.287/8.428/0.084 ms
```

Note: The ping test in this example uses a domestic website. You can ping a different address according to your network environment.

3.1.5 USB-HOST port test

The two interfaces shown in Figure 1.3 are the ⑩ **USB HOST interfaces** of the development board. Connect a mouse or keyboard to the USB HOST port. Then run the following command in **Xshell**. When moving the mouse or pressing any key on the keyboard, corresponding data will be generated.

The command is as follows:

```
$ getevent
```

```
root@HelperA733:~# getevent
could not open /dev/input/by-id, Is a directory
add device 1: /dev/input/event5
  name:      "USB OPTICAL MOUSE "
add device 2: /dev/input/event4
  name:      "sndi2s0 Headphones"
add device 3: /dev/input/event3
  name:      "gt9xxnew_ts"
could not open /dev/input/by-path, Is a directory
add device 4: /dev/input/event2
  name:      "sunxi-keyboard"
add device 5: /dev/input/event1
  name:      "axp8191-pek"
add device 6: /dev/input/event0
  name:      "sunxi-keyboard"
/dev/input/event5: 0004 0004 00090001
/dev/input/event5: 0001 0110 00000001
/dev/input/event5: 0000 0000 00000000
/dev/input/event5: 0004 0004 00090001
/dev/input/event5: 0001 0110 00000000
/dev/input/event5: 0000 0000 00000000
/dev/input/event5: 0002 0001 00000001
/dev/input/event5: 0000 0000 00000000
```

3.1.6 Camera test

The interface shown in Figure 1.3 is the ⑪ **camera interface**. Connect the camera to the development board, and then run the script command to perform the camera test. After capturing

an image, the script will display the captured image on the screen and save the image file "yuv.jpg" in the current directory.

The command is as follows:

```
$ photo_csi.sh
```

3.1.7 TF card interface test

The interface shown in Figure 1.3 is the ⑮ **TF card interface**. Insert the TF card into the development board, and then run the following command in **Xshell**. The TF card device information and storage capacity will be displayed in the terminal.

The command is as follows:

```
$ fdisk -l
```

Device	Boot	Start	End	Sectors	Size	Id	Type
/dev/mmcblk0p1		2048	31116254	31114207	14.8G	c	W95 FAT32 (LBA)

3.1.8 Microphone, headphones, speaker test

The interfaces shown in Figure 1.3 are the ⑪ **headphone interface**, ⑫ **speaker interface**, and ⑬ **microphone interface**. Run the following command in **Xshell** to perform the audio test. After executing the command, the system will record audio for **3 seconds** and then automatically play it back. If sound is output successfully, the audio function is working properly.

The command is as follows:

```
$ sound_aplay.sh
```

```
root@HelperA733:~# sound_aplay.sh
Recording WAVE '/data/luyin.wav' : Signed 16 bit Little Endian, Rate 16000 Hz, Mono
Playing WAVE '/data/luyin.wav' : Signed 16 bit Little Endian, Rate 16000 Hz, Mono
root@HelperA733:~#
```

Note: Under the Ubuntu system, sound output is available when both the headphone and speaker are connected simultaneously. It is recommended to connect only one audio output device at a time during testing.

3.1.9 Buzzer test

The interface shown in Figure 1.3 is the ③ **buzzer interface**. Run the following commands in the **Xshell** serial terminal to control the buzzer **on and off**.

```
$ gpio-test.64 w m 5 1  
$ gpio-test.64 w m 5 0
```

3.1.10 Touchscreen test

The interface shown in Figure 1.3 supports ⑪ **RGB/LVDS/MIPI multiplexed LCD displays**. After powering on the development board, tap applications on the screen or swipe across the display to perform the touch screen test. When touch data is generated during screen interaction, the touch screen test is completed successfully.

The command is as follows:

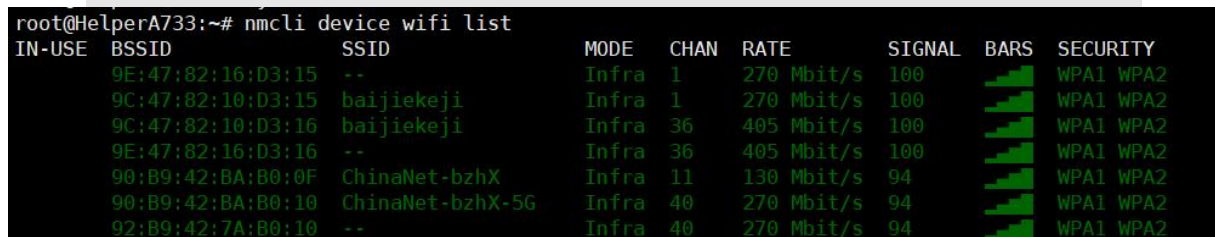
```
$ getevent  
  
root@HelperA733:~# getevent  
add device 1: /dev/input/event4  
  name:      "sndi2s0 Headphones"  
add device 2: /dev/input/event3  
  name:      "gt9xxnew_ts"  
could not open /dev/input/by-path, Is a directory  
add device 3: /dev/input/event2  
  name:      "sunxi-keyboard"  
add device 4: /dev/input/event1  
  name:      "axp8191-pek"  
add device 5: /dev/input/event0  
  name:      "sunxi-keyboard"  
/dev/input/event3: 0001 014a 00000001  
/dev/input/event3: 0003 0035 000000f2  
/dev/input/event3: 0003 0036 00000123  
/dev/input/event3: 0003 0030 00000018  
/dev/input/event3: 0003 0032 00000018  
/dev/input/event3: 0003 0039 00000000  
/dev/input/event3: 0000 0000 00000000
```

3.1.11 WIFI test

After the development board is powered on, enter the command shown below in the **Xshell serial terminal** to scan for available WiFi networks. If WiFi networks are detected, the test is completed successfully.

The command is as follows:

```
$ nmcli device wifi list
```



IN-USE	BSSID	SSID	MODE	CHAN	RATE	SIGNAL	BARS	SECURITY
	9E:47:82:16:D3:15	--	Infra	1	270 Mbit/s	100		WPA1 WPA2
	9C:47:82:10:D3:15	baijiekeji	Infra	1	270 Mbit/s	100		WPA1 WPA2
	9C:47:82:10:D3:16	baijiekeji	Infra	36	405 Mbit/s	100		WPA1 WPA2
	9E:47:82:16:D3:16	--	Infra	36	405 Mbit/s	100		WPA1 WPA2
	90:B9:42:BA:80:0F	ChinaNet-bzhX	Infra	11	130 Mbit/s	94		WPA1 WPA2
	90:B9:42:BA:80:10	ChinaNet-bzhX-5G	Infra	40	270 Mbit/s	94		WPA1 WPA2
	92:B9:42:7A:80:10	--	Infra	40	270 Mbit/s	94		WPA1 WPA2

3.1.12 4G module test

The interface shown in Figure 1.3 is the ⑥ **SIM card slot**. The 4G module used on the development board is the ⑦ **EC20 module** (Note: The module needs to be purchased separately). Insert the 4G module and SIM card into the development board, and secure the 4G module properly to prevent it from becoming loose. Power on the development board and perform the following steps after the system is ready.

There are two scripts related to the 4G function: one is used to **enable 4G**, and the other is used to **remove 4G**. After enabling 4G, the **ifconfig** command should display the **ppp0** interface and its assigned IP address. If the development board can successfully ping www.baidu.com, the 4G test is completed successfully.

1. First, ensure that the required software packages have been installed. If they are not installed, run the following commands to install them and then reboot the system.

```
$ apt update && apt install ppp modemmanager -y
$ reboot
```

2. After rebooting the system, run the following command to enable the 4G function. The first time the 4G connection is established, it may take some time to obtain an **IP** address.

```
$ 4g_set.sh

root@HelperA733:~# 4g_set.sh
Connection 'ppp0' (87005a90-fb12-4e20-b95d-814202968f45) successfully added.
root@HelperA733:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.140 netmask 255.255.254.0 broadcast 192.168.1.255
    inet6 fe80::5b97:eddc:4dc0:5855 prefixlen 64 scopeid 0x20<link>
    ether d4:bf:48:ac:9a:57 txqueuelen 1000 (Ethernet)
    RX packets 1948 bytes 161095 (161.0 KB)
    RX errors 0 dropped 13 overruns 0 frame 0
    TX packets 28 bytes 2156 (2.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 7

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 84 bytes 6688 (6.6 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 84 bytes 6688 (6.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

p2p0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 18:84:c1:01:bb:f8 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ppp0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
    inet 10.3.231.150 netmask 255.255.255.255 destination 10.64.64.64
    ppp txqueuelen 3 (Point-to-Point Protocol)
    RX packets 10 bytes 122 (122.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 11 bytes 191 (191.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

3. Disable 4G

```
$ 4g_del.sh
```

After the 4G function is removed successfully, run the **ifconfig** command to check the network interfaces. The **ppp0** interface should no longer be displayed.

3.1.13 Bluetooth test

Use a mobile phone to communicate with the Bluetooth module on the development board for testing. There are two Bluetooth-related scripts on the development board. One script is used to **start Bluetooth and configure the connection**, and the other is used to **start the communication port**. Run the Bluetooth configuration script in the **Xshell terminal**.

The command is as follows:

```
$ bluetooth_set.sh
```

```
[CHG] Controller 10:11:12:13:14:15 UUIDs: 0000110e-0000-1000-8000-00805f9b34fb
[CHG] Controller 10:11:12:13:14:15 UUIDs: 00001200-0000-1000-8000-00805f9b34fb
[CHG] Controller 10:11:12:13:14:15 UUIDs: 0000110c-0000-1000-8000-00805f9b34fb
[CHG] Controller 10:11:12:13:14:15 UUIDs: 00001800-0000-1000-8000-00805f9b34fb
[CHG] Controller 10:11:12:13:14:15 Alias: BlueZ 5.48
[CHG] Controller 10:11:12:13:14:15 Pairable: yes
[CHG] Controller 10:11:12:13:14:15 Alias: localhost.localdomain
[bluetooth]#
```

Note: After entering the Bluetooth interface, you can enter "**help**" to view the available command information.

1. Scan for Bluetooth devices

```
$ scan on
```

```
[CHG] Controller 10:11:12:13:14:15 Alias: localhost.localdomain
[bluetooth]# scan on
Discovery started
[CHG] Controller 10:11:12:13:14:15 Discovering: yes
[NEW] Device 69:60:1B:95:9A:89 69-60-1B-95-9A-89
```

After the scan is completed, the Bluetooth device "**szbaijie_bluetooth**" is detected.

The device ID (MAC address) of the mobile phone Bluetooth device is obtained as

F0:C4:2F:39:6E:55, as shown below:

```
[NEW] Controller 10:11:12:13:14:15 localhost.localdomain [default]
[NEW] Device F0:C4:2F:39:6E:55 szbaijie bluetooth
Agent registered
[CHG] Controller 10:11:12:13:14:15 Class: 0x00000000
[CHG] Controller 10:11:12:13:14:15 Powered: yes
[CHG] Controller 10:11:12:13:14:15 UUIDs: 00001801-0000-1000-8000-00805f9b34fb
```

After finding the corresponding device ID, stop the Bluetooth scan using the following command:

```
$ scan off
```

If an error occurs during the Bluetooth scan, check whether the Bluetooth device **hci0** is enabled. If it is not enabled, run the **bluetooth_set.sh** script again.

Use the following command to check the status:

```
$ hciconfig -a
```

When **hci0** is enabled properly, the output is shown as follows:

页

```
root@HelperA733:~# hciconfig -a
hci0: Type: BR/EDR Bus: UART
      BD Address: 18:84:C1:01:BB:FA ACL MTU: 1021:9 SCO MTU: 255:4
      UP RUNNING PSCAN ISCAN
      RX bytes:3000 acl:0 sco:0 events:106 errors:0
      TX bytes:2335 acl:0 sco:0 commands:63 errors:0
      Features: 0xbf 0xee 0xcd 0xfe 0xd8 0x3f 0x7b 0x87
      Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV3
      Link policy: RSWITCH SNIFF
      Link mode: SLAVE ACCEPT
      Name: 'HelperA733'
      Class: 0x000000
      Service Classes: Unspecified
      Device Class: Miscellaneous,
      HCI Version: (0xb) Revision: 0x14
      LMP Version: (0xb) Subversion: 0x14
      Manufacturer: RivieraWaves S.A.S (96)
```

2. Pair with the Bluetooth device

After obtaining the device ID of the mobile phone Bluetooth device, enter the pairing command. A confirmation prompt will appear after a short delay. Enter **"yes"** in the development board command line to confirm the pairing.

```
$ pair F0:C4:2F:39:6E:55
```

Note: **F0:C4:2F:39:6E:55** is the device ID (MAC address) of the Bluetooth device, which is displayed after executing the **scan on** command.

```
[bluetooth]# pair F0:C4:2F:39:6E:55
Attempting to pair with F0:C4:2F:39:6E:55
[CHG] Device F0:C4:2F:39:6E:55 Connected: yes
Request confirmation
[szba1m[agent] Confirm passkey 623495 (yes/no):
```

At this point, a pairing request will also appear on the mobile phone. Tap **"Pair"** to confirm the connection. The interface after successful pairing is shown below:

```
[bluetooth]# pair F0:C4:2F:39:6E:55
Attempting to pair with F0:C4:2F:39:6E:55
[CHG] Device F0:C4:2F:39:6E:55 Connected: yes
Request confirmation
[szba1m[agent] Confirm passkey 839765 (yes/no): yes
[CHG] Device F0:C4:2F:39:6E:55 Modalias: bluetooth:v010Fp107Ed1436
[CHG] Device F0:C4:2F:39:6E:55 UUIDs: 0000046a-0000-1000-8000-00805f9b34fb
[CHG] Device F0:C4:2F:39:6E:55 UUIDs: 00001105-0000-1000-8000-00805f9b34fb
[CHG] Device F0:C4:2F:39:6E:55 UUIDs: 0000110a-0000-1000-8000-00805f9b34fb
[CHG] Device F0:C4:2F:39:6E:55 UUIDs: 0000110c-0000-1000-8000-00805f9b34fb
[CHG] Device F0:C4:2F:39:6E:55 UUIDs: 00001112-0000-1000-8000-00805f9b34fb
```

After successful pairing, enter **"exit"** to exit the Bluetooth connection interface. The command is as follows:

```
$ exit
```

3. Connect to the Bluetooth device

After exiting the pairing interface, run the Bluetooth port registration command on the development board.

The command is as follows:

```
$ bluetooth_rfcomm.sh
```

After executing the above command, the system will enter the waiting state. At this point, install a **Bluetooth serial port application** on the mobile phone. Open the application and connect to the paired Bluetooth device on the development board. After the connection is established, the development board will display a successful connection message and exit the waiting state. The Bluetooth serial port application on the mobile phone and the development board can now transmit and receive data.

The successful connection status is shown below:

```
root@HelperA733:~# Waiting for connection on channel 1
Connection from 78:34:FD:E3:F8:2E to /dev/rfcomm0
Press CTRL-C for hangup
```

Note: If the Bluetooth serial port application is not installed or no connection is established, the system will remain in the waiting state.

4. Bluetooth Communication

(1) Receive Data

Run the following command in the **Xshell terminal**. Send data through the Bluetooth serial port application on the mobile phone, and the corresponding data will be displayed in the terminal.

```
$ cat /dev/rfcomm0
```

(2) Send Data

Run the following **echo** command in the **Xshell terminal** to send data. The sent data will also be displayed in the Bluetooth serial port application on the mobile phone.

The command is as follows:

```
$ echo szbaijie >/dev/rfcomm0
```

3.1.14 Backlight adjustment

The backlight brightness can be adjusted using the following command. The **value** parameter is a number ranging from **0 to 255**. In general, **0** represents the lowest backlight brightness, and **255** represents the highest backlight brightness.

```
$ set_brightness.64 value
```

Use the above command to set the backlight brightness to **150**. The change in backlight brightness can be observed on the display.

```
root@HelperA733:~# set_brightness.64 150
Brightness set to 150 (max: 255)
root@HelperA733:~#
```

3.1.15 PCIE port test

The interface shown in Figure 1.3 is the **⑧ PCIE M.2 interface**. Insert a solid-state drive (SSD) that supports this interface, and then power on the development board.



After entering the system, run the following commands to verify that the system has detected the NVMe SSD, and format the corresponding partition with the **ext4** file system.

```
$ ls /dev/nvme0*
$ mkfs.ext4 /dev/nvme0n1
```

```
root@HelperA733:~# ls /dev/nvme0*
/dev/nvme0 /dev/nvme0n1
root@HelperA733:~# mkfs.ext4 /dev/nvme0n1
mke2fs 1.47.0 (5-Feb-2023)
/dev/nvme0n1 contains a ext4 file system
    last mounted on /mnt/nvme on Sat Jun 27 16:33:04 2026
Proceed anyway? (y,N) y
Discarding device blocks: done
Creating filesystem with 31258710 4k blocks and 7815168 inodes
Filesystem UUID: 11dedd36-a0b4-4e07-a801-2798f8552245
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
    4096000, 7962624, 11239424, 20480000, 23887872

Allocating group tables: done
Writing inode tables: done
Creating journal (131072 blocks): done
Writing superblocks and filesystem accounting information: done

root@HelperA733:~#
```

After formatting is completed, create a mount directory and manually mount the SSD. After the mount operation is successful, use the **df -h** command to check the disk capacity and mount status.

```
$ mkdir -p /mnt/nvme
$ mount /dev/nvme0n1 /mnt/nvme
$ df -h
```

```
root@HelperA733:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda4        4.9G  1.3G  3.6G  27% /
tmpfs            2.0G   0    2.0G   0% /dev/shm
tmpfs            782M  1.6M  781M   1% /run
tmpfs            5.0M   8.0K  5.0M   1% /run/lock
/dev/sda5        25G   24K   23G   1% /data
tmpfs            391M   8.0K  391M   1% /run/user/0
/dev/nvme0n1     117G   24K  111G   1% /mnt/nvme
```

3.2 Function testing of the Android system on the development board

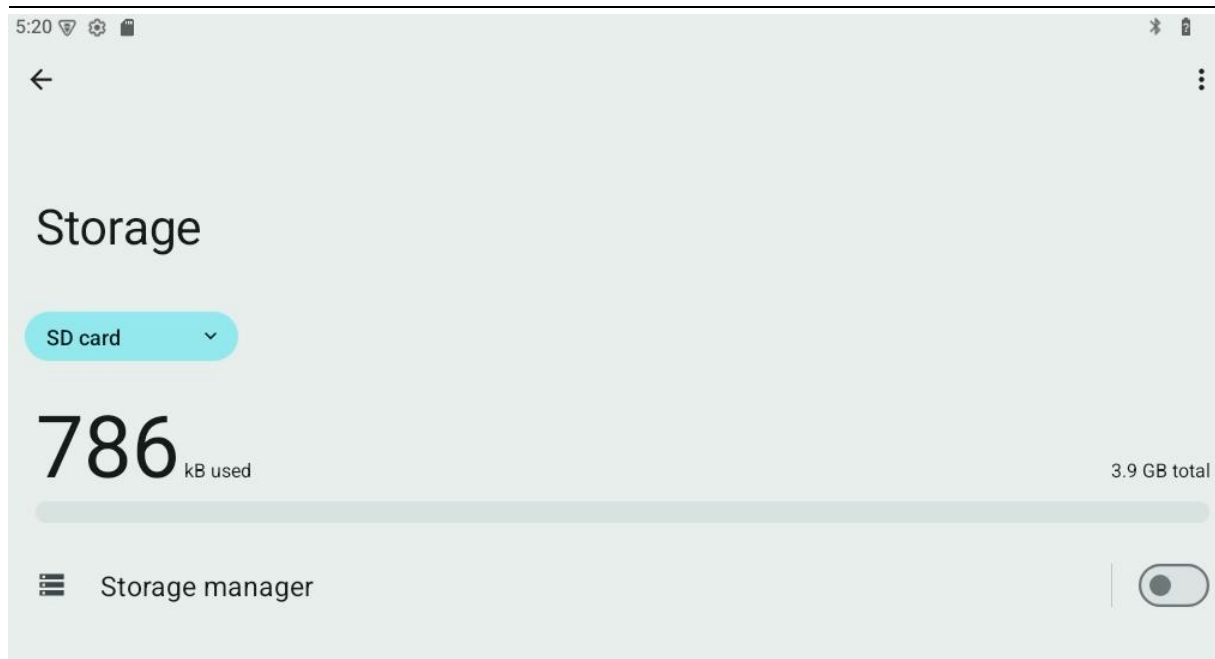
The Android system and Ubuntu system share many common test items. The test methods for the common items are as follows:

- 485 communication test
- Ethernet port test
- USB HOST port test
- Touch screen test

3.2.1 TF card interface test

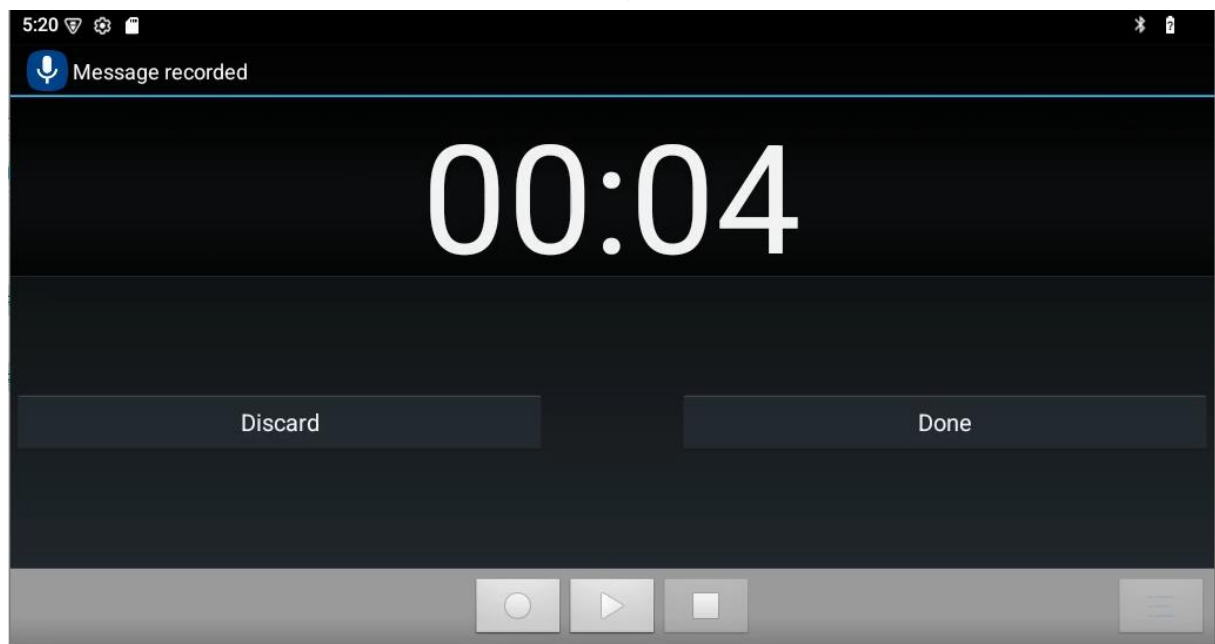
The interface shown in Figure 1.3 is the ⑮ **TF card interface**. Insert an SD card, then open **Settings** → **Storage**. The TF card storage capacity will be displayed in the storage information.

页



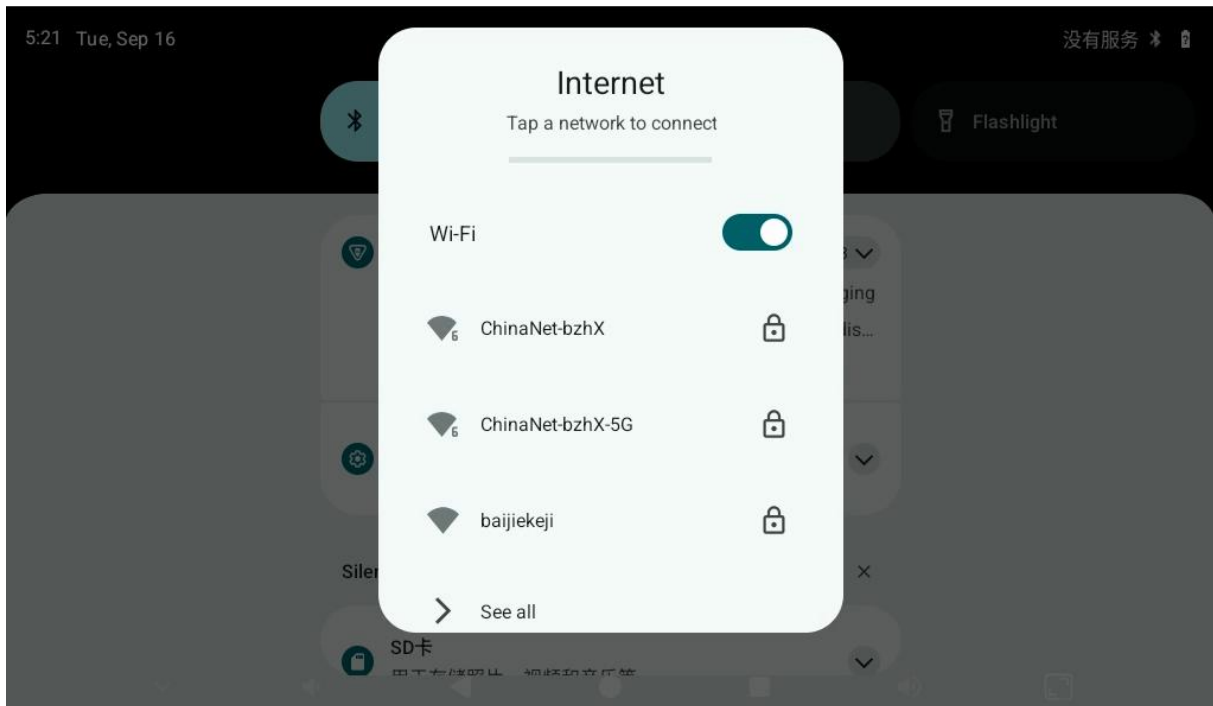
3.2.2 Microphone, headphones, speaker test

The interfaces shown in Figure 1.3 are the ⑪ **headphone interface**, ⑫ **speaker interface**, and ⑬ **microphone interface**. The **Recorder** application can be used to test the microphone, speaker, and other audio interfaces simultaneously, as shown below:



3.2.3 WIFI test

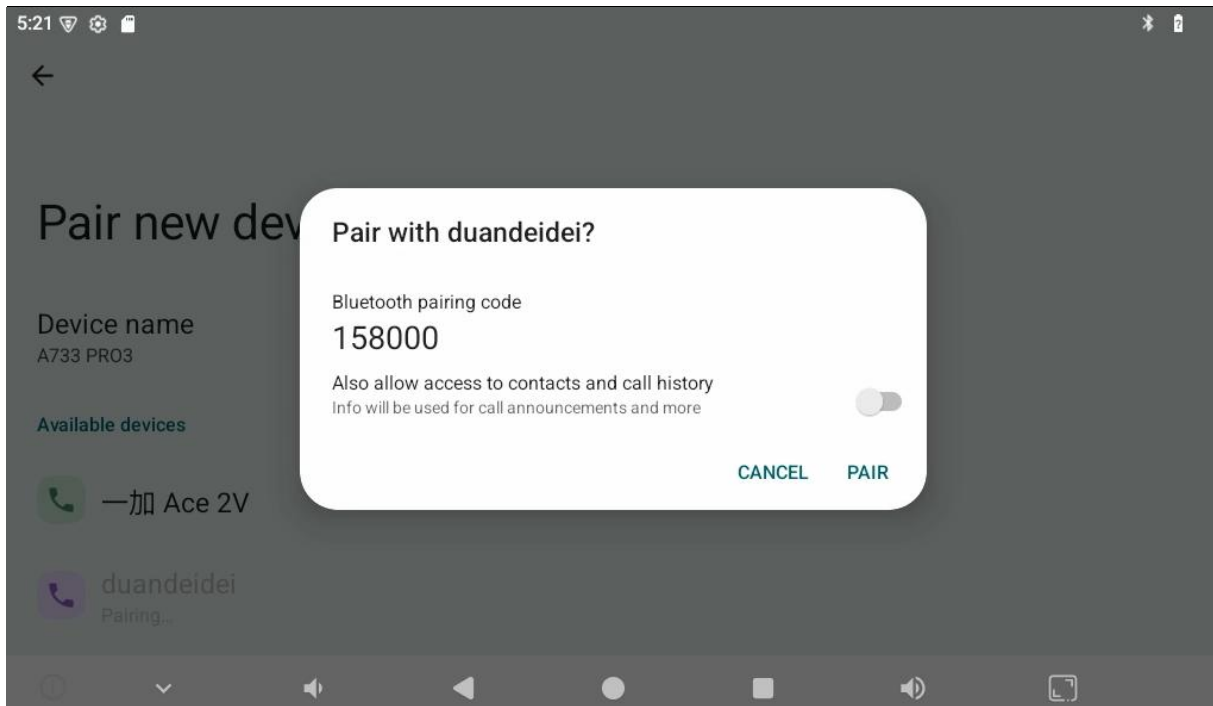
After powering on the development board, pull down the notification panel on the interface to enable or disable Wi-Fi.



3.2.4 Bluetooth test

After powering on the development board, pull down the notification panel and tap **"Pair with new device"** to search for available Bluetooth devices and complete the pairing process.

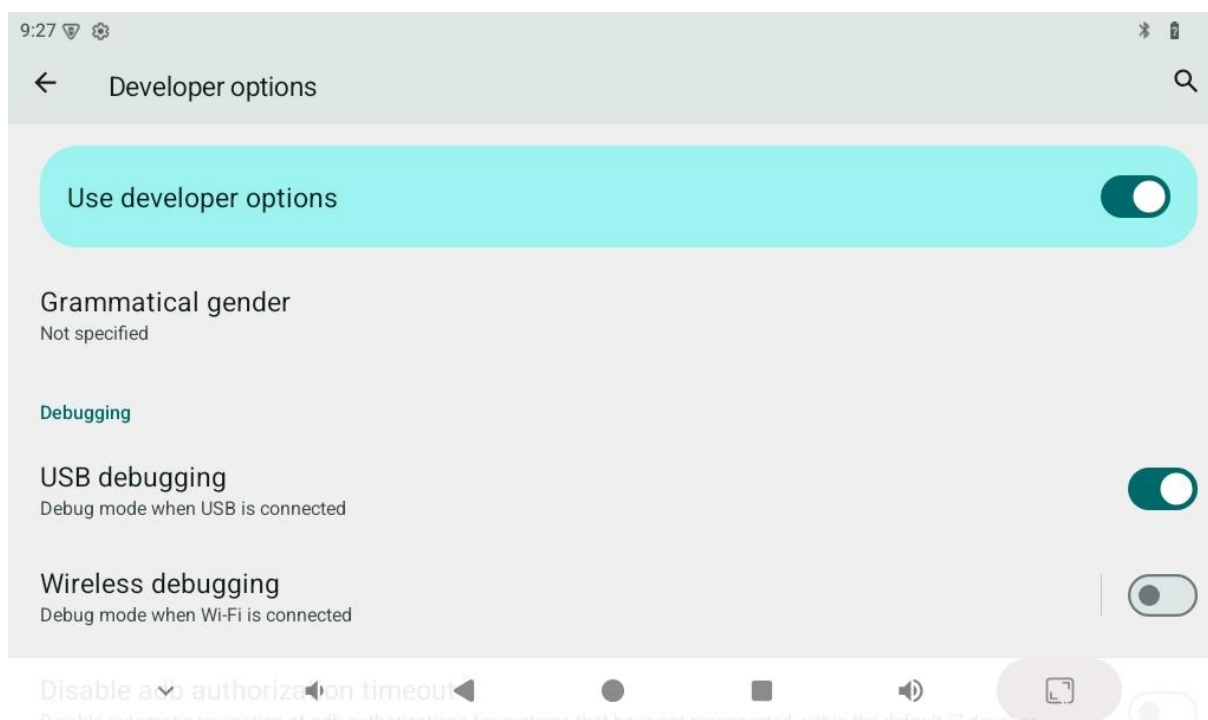
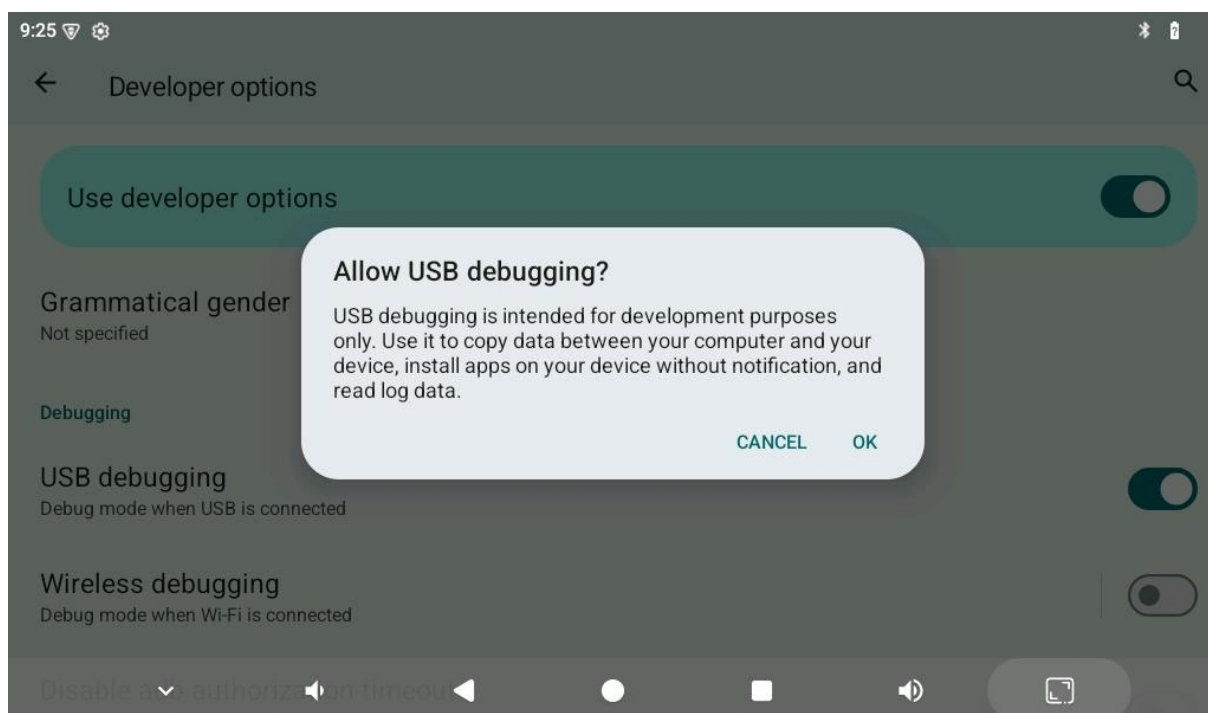
The pairing interface is shown below:



3.2.5 USB-OTG port test

The interface shown in Figure 1.3 is the ⑭ **USB OTG interface**. After the system boots up, open the following path on the interface: **Settings** → **System** → **Advanced** → **Developer options** → **USB debugging** (**USB debugging is enabled by default**). When enabling USB debugging, a confirmation prompt will appear as shown below. Select "**Yes**" to enable Android **ADB debugging**. If the ADB connection is successful, the test is passed. This interface can also be used as the firmware flashing port. For detailed flashing instructions, please refer to **Section 4.3**.

Note: For more information about Android **ADB debugging**, please refer to **Section 2.3**.



If the development board cannot be connected to Android using **adb shell** (skip this step if the connection works directly), and no device ID is displayed when running **adb devices**, the OTG function must first be enabled through the serial terminal of the development board.

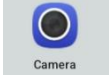
The operation steps are as follows:

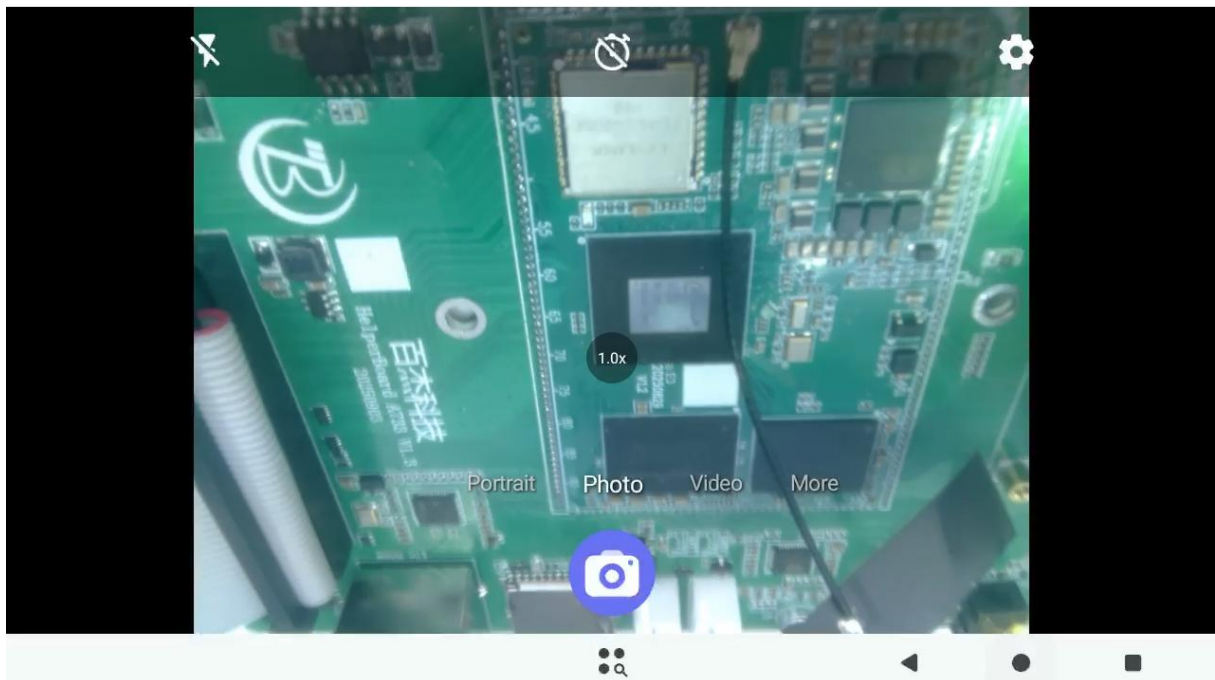
```
$ su
$ cat /sys/devices/platform/soc@30000000/10.usbc0/usb_device
```

```
console:/ # cat /sys/devices/platform/soc@30000000/10.usbc0/usb_device
device_chose finished, otg disabled!
console:/ #
```

After the message **"device chose finished"** is displayed, the Android system can be connected using **ADB**. At this point, run the following command in the computer terminal:
adb devices. The device ID will be displayed in the terminal.

3.2.6 Camera test

Open the application menu and tap the **"Camera"** application  to enter the camera interface. You can then take photos or record videos, as shown below:



Note: The **Camera** application will only appear after a camera device is connected.

4 Use of software tools

4.1 VMware development environment Setup

4.1.1 Hardware Requirements

- **Memory (RAM):** ≥ 12 GB (16 GB or above is recommended for Android compilation)
- **Storage:** ≥ 500 GB

4.1.2 Software Requirements

To simplify the development process, we have prepared an **Ubuntu 20.04 virtual machine image** with all the required software pre-installed.

If you do not use the virtual machine image provided by our company, please download and install the following software:

Ubuntu16.04 downloading

```
$ sudo apt-get install libncurses5 bison \
lsuf busybox build-essential subversion \
git-core libncurses5-dev \
zlib1g-dev gawk flex quilt libssl-dev \
xsltproc libxml-parser-perl mercurial \
bzip2 ecj cvs unzip pigz lib32z1 \
lib32z1-dev lib32stdc++6 libstdc++6 \
openssl linux-tools-common gperf \
android-tools-fsutils -y
```

Unubtu18.04 downloading

```
$ sudo apt-get install libncurses5 bison lsuf \
busybox build-essential \
subversion git-core libncurses5-dev \
zlib1g-dev gawk flex quilt libssl-dev \
xsltproc libxml-parser-perl mercurial \
bzip2 ecj cvs unzip pigz lib32z1 \
lib32z1-dev lib32stdc++6 libstdc++6 \
openssl linux-tools-common gperf \
python-minimal android-tools-fsutils -y
```

Ubuntu20.04 downloading

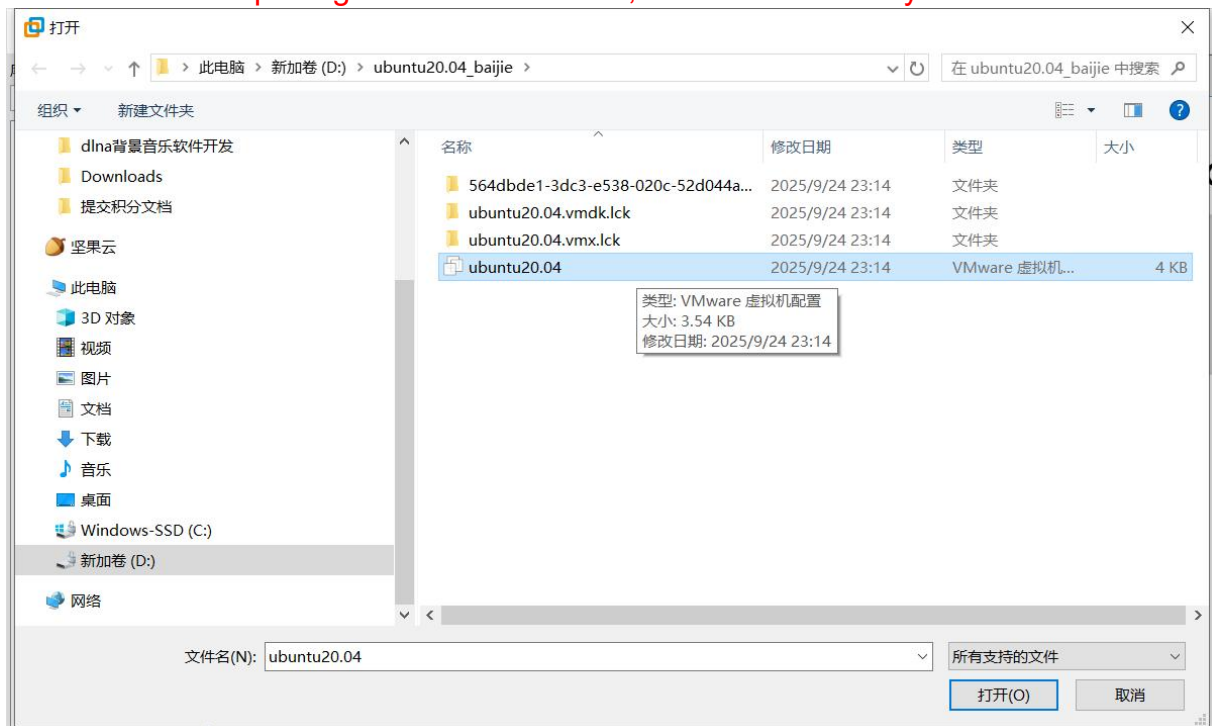
```
$ sudo apt-get install libncurses5 bison lsof \
busybox build-essential \
subversion git-core libncurses5-dev \
zlib1g-dev gawk flex quilt libssl-dev \
xsltproc libxml-parser-perl mercurial bison \
bzip2 ecj cvs unzip pigz lib32z1 libelf-dev \
lib32z1-dev lib32stdc++6 libstdc++6 \
openssl linux-tools-common gperf \
android-sdk-libparse-utils android-sdk-ext4-utils python2 -y
```

Installing the **VMware 17** virtual machine software is straightforward. The installation package is available in the shared network drive. Please install it by yourself. The virtual machine image is located in the following shared directory: **tools/vmware17/helperboard_ubuntu20.04.zip**. After extracting the file, the virtual machine can be used directly. The following section describes how to configure the development environment in **VMware 17**.

Open **VMware 17**, then select the menu: **File** → **Open**, Select the extracted virtual machine image file: **ubuntu20.04.vmx**

as shown below:

Note: After opening the virtual machine, the default memory allocation is **4 GB**.



The default virtual machine configuration is as follows:

- **CPU:** 2 processors, 3 cores per processor
- **Memory (RAM):** 12 GB
- **Storage:** 500 GB hard disk, including a 24 GB swap partition

Users can adjust the configuration according to their requirements, such as increasing the memory size or the number of CPU cores.

The CPU used for creating this virtual machine image is **Intel Core i7-6700K**. When running the virtual machine, some users may encounter compatibility issues. In this case, you can copy the virtual machine and select the option to automatically configure it for your own CPU. The virtual machine should then run normally.

Note:

1. The username and password of the **Ubuntu 20.04 virtual machine** are both: **szbaijie**.
2. Do not update the system after installing the virtual machine. Otherwise, the Python version may be changed, which could cause compilation failures.

Then click the **Power on** button on the **VMware 17**  toolbar to start the **HelperBoard virtual machine**.

If the following prompt appears, click **"Take Ownership"** to continue.



Then click **"I Moved It"** to enter the system.



4.2 Install PhoenixSuit

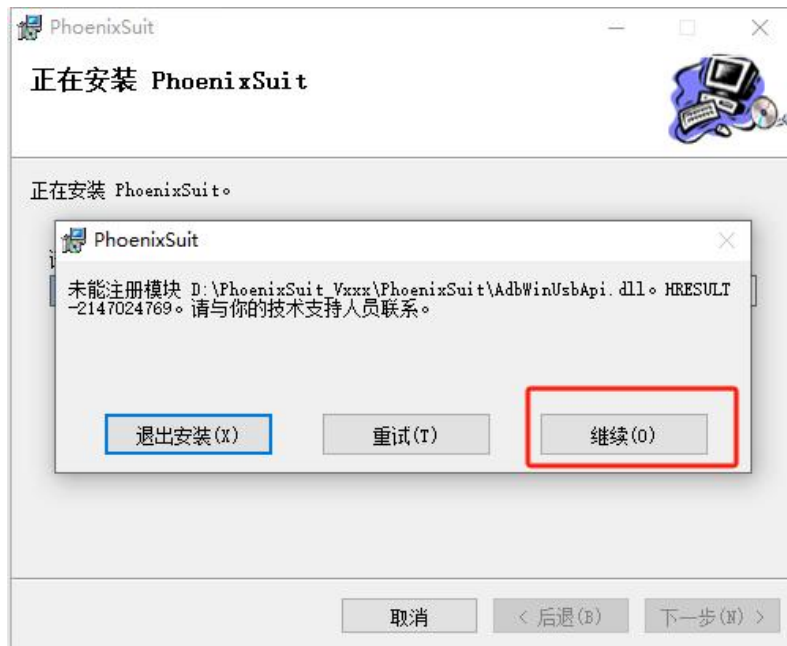
Download and install the software from the following shared directory:

tools/windows_burning_tool/PhoenixSuit_CN.msi. During the installation process, a driver installation window will appear. Follow the steps below to complete the driver installation:



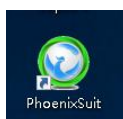


If the following prompt appears, click **"Continue"** to proceed.





After the installation is completed, a **PhoenixSuit** shortcut icon will be created on the desktop.



4.3 Flash the firmware of the development board

Install **PhoenixSuit** according to the instructions in **Section 4.2**. After opening the software, the following interface will be displayed:

Note: The firmware flashing procedure may vary slightly depending on the PhoenixSuit version. This section describes the flashing process based on **version 2.0.1**.



Extract the downloaded firmware package, click **"One-Key Flash"**, and then click **"Browse"** to select the firmware file to be flashed, for example: **a733_dragonboard_pro3_uart0.img**. After selecting the firmware file, set the flashing mode to **"Partition Erase Upgrade"**.



Firmware flashing steps:

1. Connect the USB OTG port of the development board to the computer (the port ⑭ is shown in **Figure 1.3**).
2. Power on the development board.
3. After the system startup is completed, press and hold the **FEL** flashing button (⑮ shown in **Figure 1.3**) without releasing it, then press the **RST** reset button (⑯ shown in **Figure 1.3**). When releasing the buttons, release the **RST** button first, and then release the **FEL** button. If the development board successfully enters **firmware flashing mode**, the software interface will display the status shown below.



Note: Do not disconnect the power supply of the development board during the flashing process. Otherwise, the firmware flashing may fail. Wait for a few minutes until the flashing process is completed successfully, as shown below. After flashing is completed, restart the development board to enter the system. The first boot may take a little longer than usual.



For **PhoenixSuit versions earlier than 1.19**, a confirmation dialog will appear when entering the flashing mode. Click "**Yes**" to continue. A prompt will also be displayed after the flashing process is completed, as shown below:





Note: Due to the limitations of the Allwinner download tool, the progress bar may occasionally stop moving during the flashing process. If this issue occurs, restart the development board and repeat the flashing procedure described above.

After the firmware flashing is completed successfully and the development board boots up normally, the system interface will be displayed.

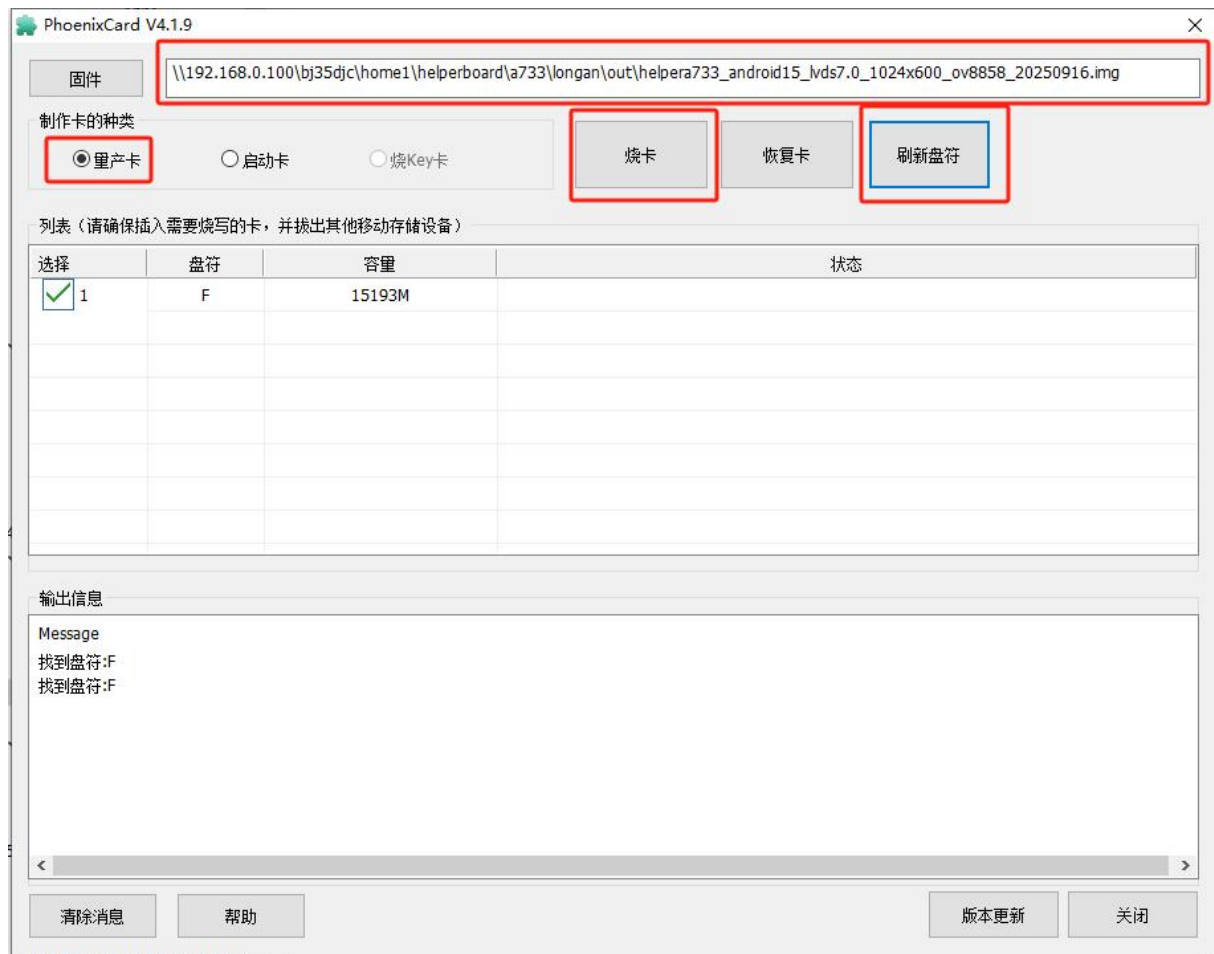
4.4 PhoenixCard produces SD card burning discs

Download the following file from the shared directory:

tools/windows_burning_tool/PhoenixCard_V4.1.9_20190227.zip. After downloading and

extracting the package, open the installation directory  **PhoenixCard.exe** and run the executable file directly.

The software interface is shown below:

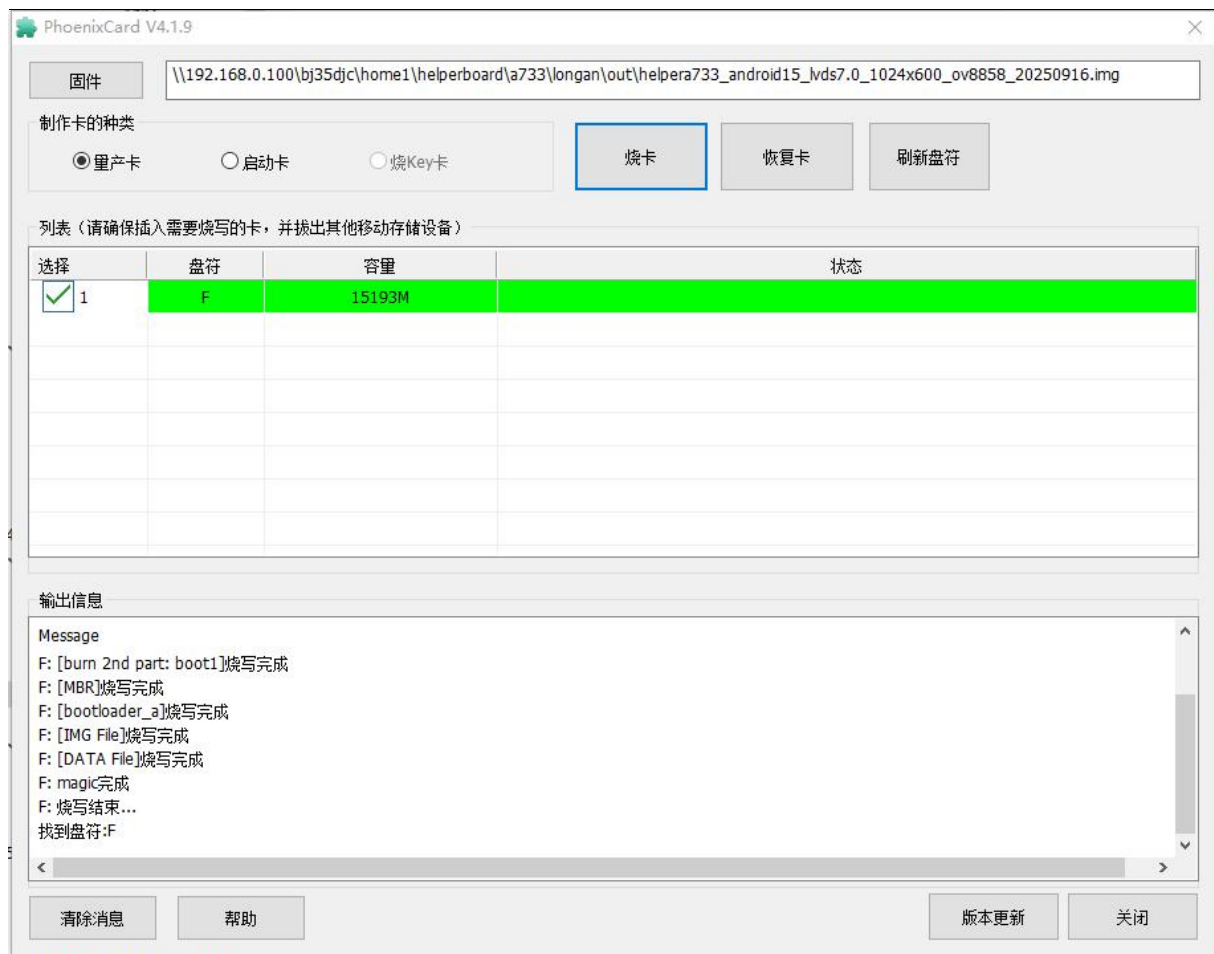


Click "**Refresh**" to automatically search for the available drive letters of the card reader, and then select the SD card to be used for creating the flashing card.

After the drive letter is detected, click "**Firmware**". A dialog box for selecting the **.img firmware file** will appear. Select the required firmware image file.

After selecting the firmware image, change the flashing mode to "**Production Card**", and then click the "**Burn**" button to start creating the SD card. After the process is completed successfully, the selected SD card will become a system flashing card. The successful flashing status is shown below. The SD card can then be inserted into the development board to flash the firmware onto the board.

页



Note: When the SD card is no longer required as a bootable flashing card, click "**Restore Card**" to format the SD card flashing card back to a normal storage card.

```
partdata hi 0x0
partdata lo 0x1729000
sparse: bad magic
[07.077]succeeded in writting part boot
origin_verify value = 10830c90, active_verify value = 10830c90
[07.339]succeeded in verify part boot
[07.342]succeeded in download part boot
[07.346]begin to download part rootfs
partdata hi 0x0
partdata lo 0x390000000
sparse: bad magic
```

Note: If the SD card flashing process fails, use SDFormatter to format the SD card and try again.If the issue still persists after formatting, replace the SD card with another one and retry the flashing process.

4.5 If the card write fails, use SDFormatter to format the memory card

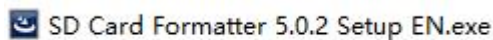
If the development board cannot boot normally after flashing the firmware using an SD card and the error shown below appears, use **SDFormatter** to format the SD card first. Then use **PhoenixCard** to rewrite the firmware image to the SD card, and use the SD card to flash the firmware onto the development board again.

```
partdata lo 0x00000000
download rootfs 100M
[ 140.452]succeeded in writting part rootfs
origin_verify value = 6c33df22, active_verify value = d4aba917
origin checksum=6c33df22 active checksum=d4aba917
sunxi sprite part ROOTFS_FEX0000000VROOTFS_FEX000000 verify error
sunxi sprite err: sunxi_sprite_deal_part, download normal failed
sunxi sprite error : download part error
sprite_test do a sprite test

Usage:
sprite_test NULL
sunxi
```

Download the following file from the shared directory:

tools/SD_card_formatter/SDCardFormatterv5_WinEN.zip.After downloading and extracting

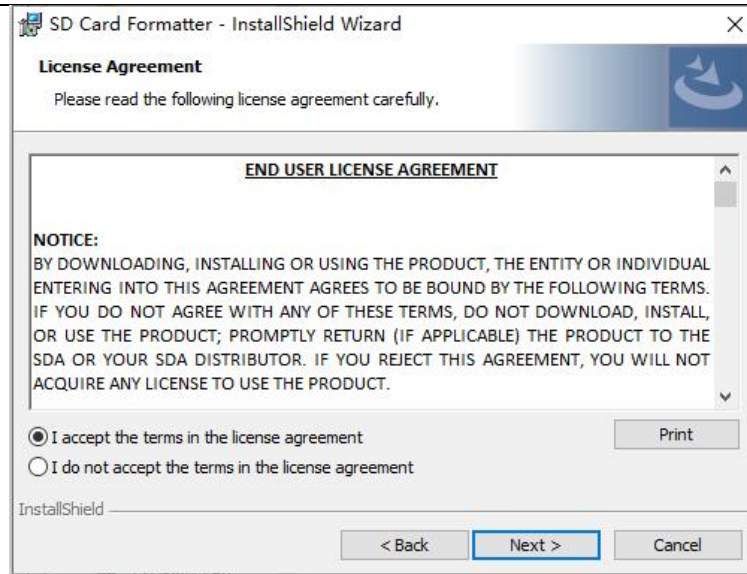


the package, open the directory and double-click the installer to start the installation.

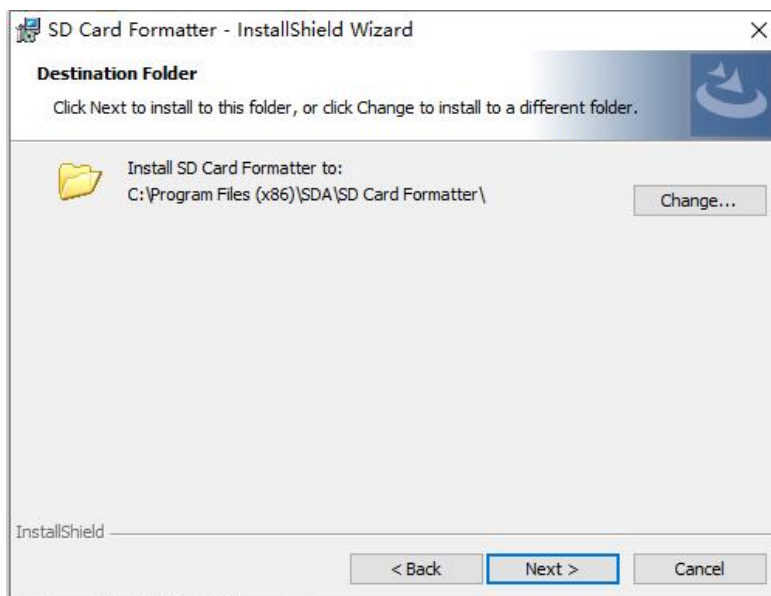
The installation process is shown below:



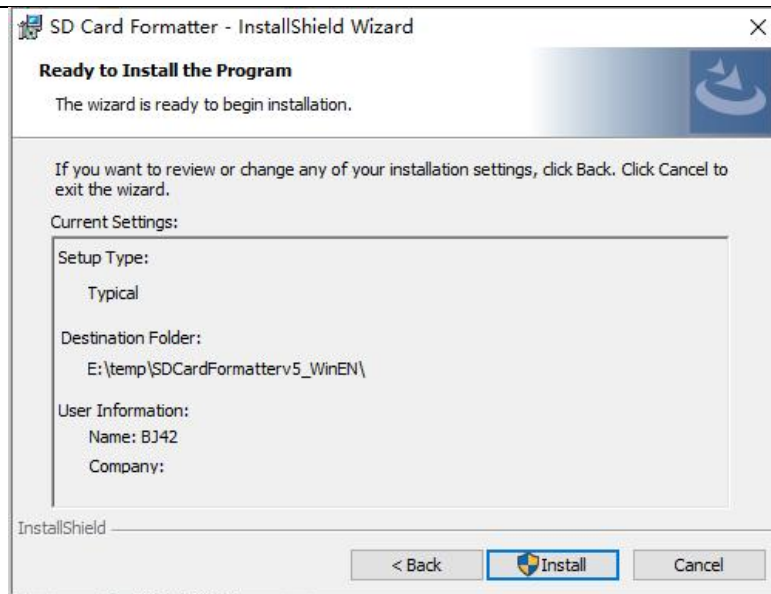
Click "**Next**" to continue.



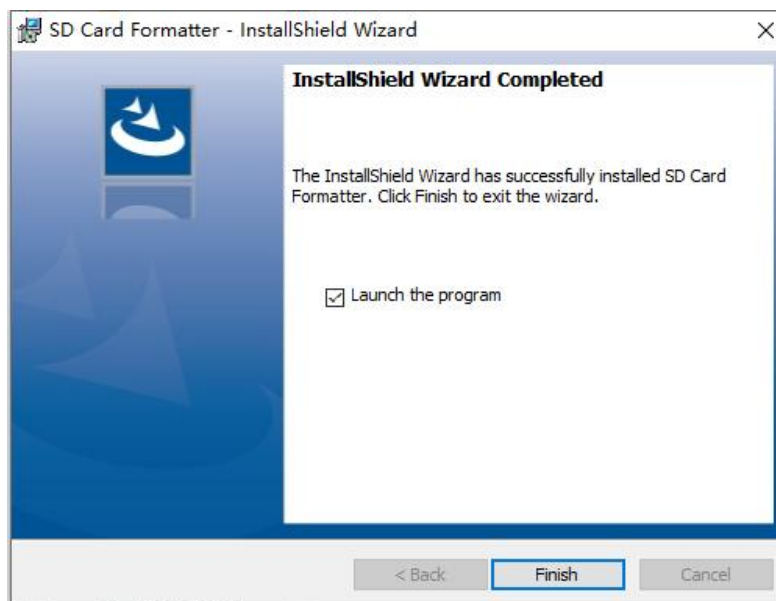
Select "**I accept**", and then click "**Next**" to continue.



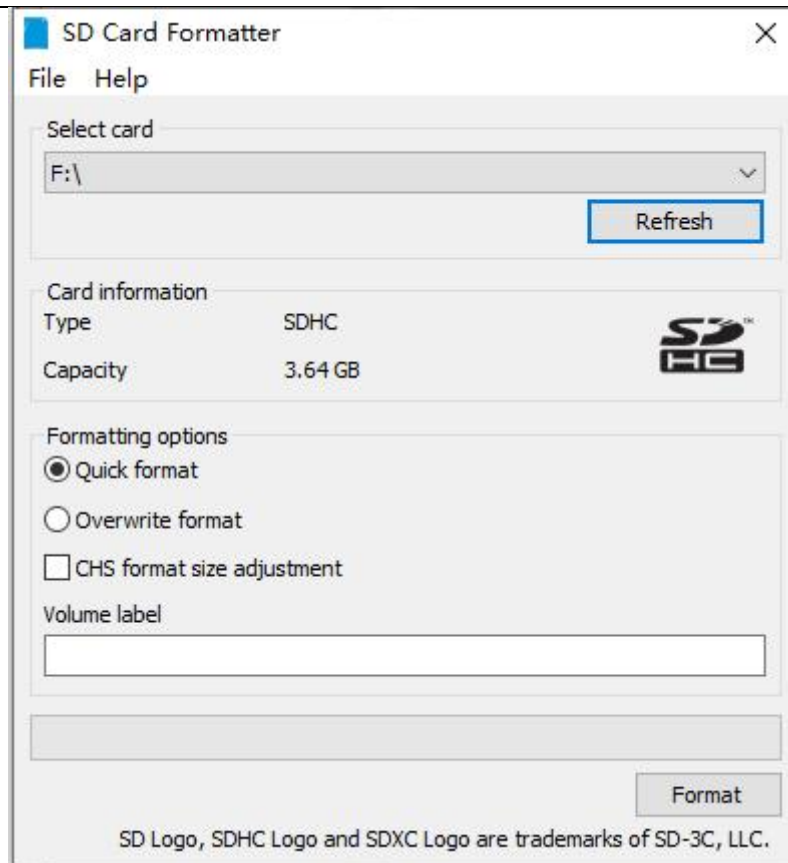
Select the installation path.



Click "Install" to start the installation.



After the installation is completed, click "Finish" to launch the software.

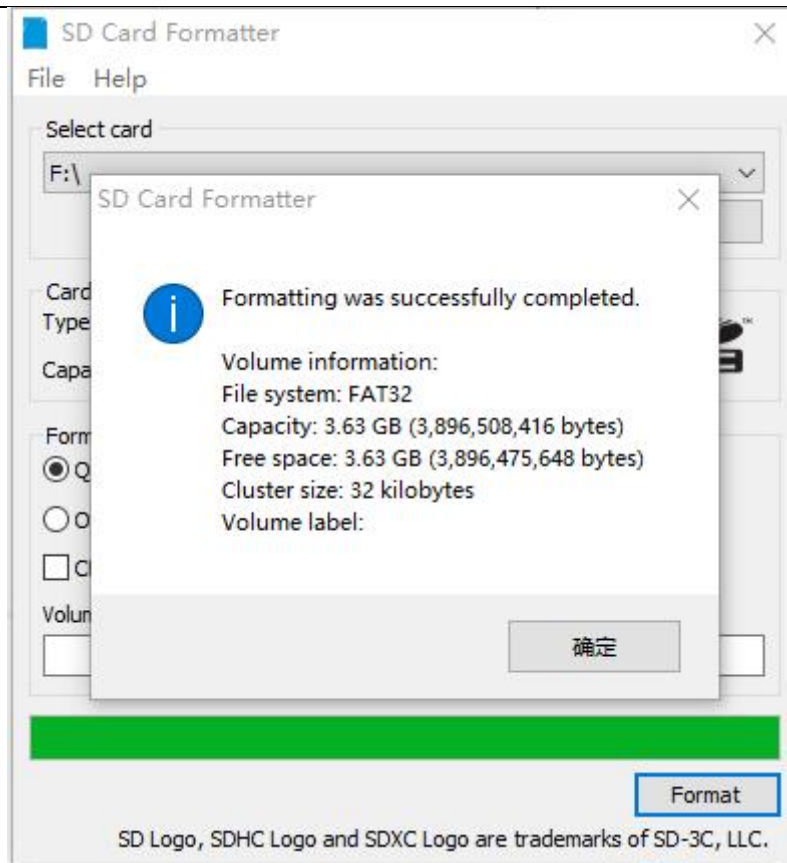


After inserting the SD card, the software will automatically detect and read the card. If the SD card is not detected, click **"Refresh"** to scan again.

After the SD card is detected successfully, click the **"Format"** button and use the default formatting options.

If an error occurs during the formatting process, check whether the card reader interface is damaged, whether the SD card is inserted correctly, and whether the connection is loose.

The following interface indicates that the formatting process has been completed successfully.



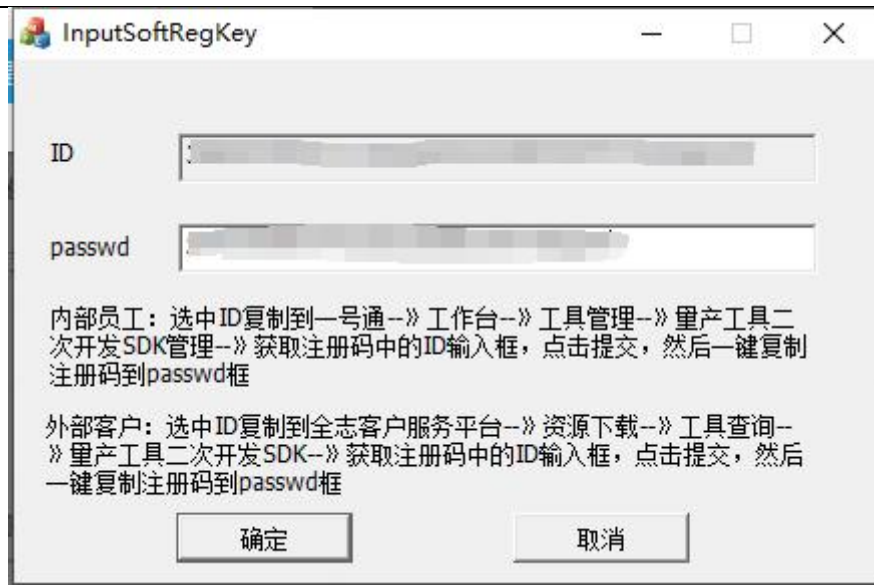
Note: If the issue still cannot be resolved after formatting, try using another SD card.

4.6 The use of Dragonface

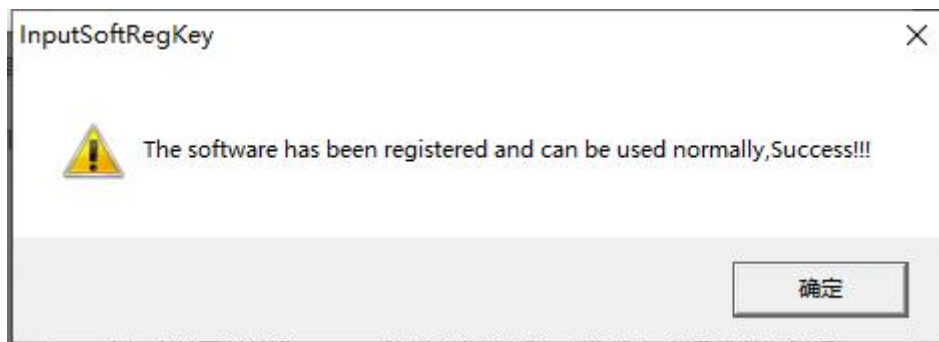
Currently, **DragonFace** only supports modifications for the **Android system**. Download and extract the following file from the shared directory:

tools/firmware_modify_tool/DragonFace_V4.1.6.zip

After extraction, double-click  **DragonFace.exe** the executable file to open the registration window. Copy the displayed **ID** and send it to the technical support team to obtain the registration code.



The registration success message is shown below:

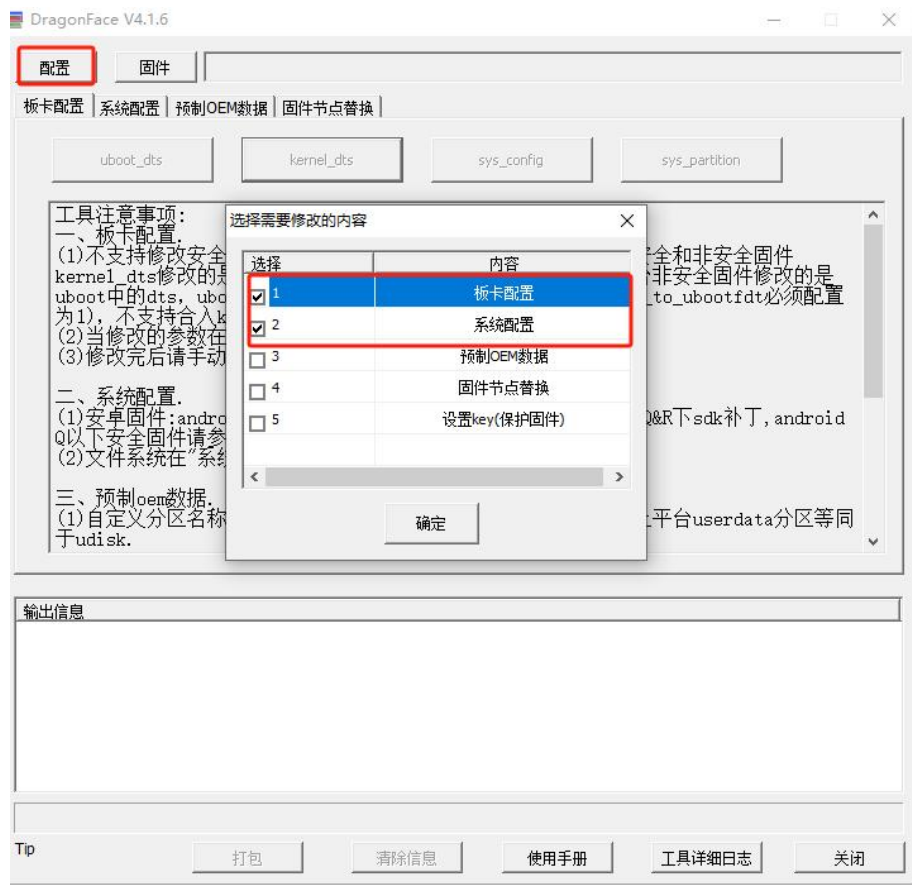


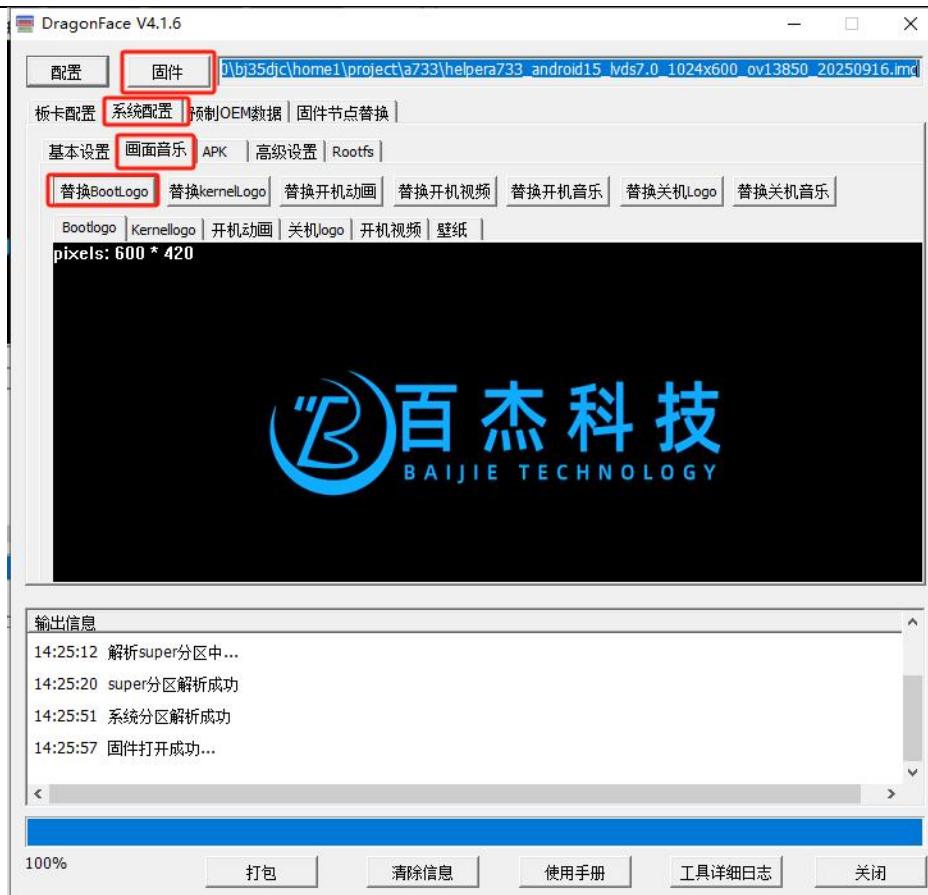
4.6.1 Change the Android startup logo

Open **DragonFace**:

1. Click **"Configuration"** and select the corresponding configuration file.
2. Click **"Firmware"** and select the firmware image that needs to be modified for the boot logo. Wait until the firmware is loaded.
3. After the firmware is loaded successfully, select **"System Configuration"** and **"Picture and Music"**.
4. Click **"Replace Bootlogo"** and select the boot logo image to be replaced.
5. After replacing the boot logo, click **"Pack"** in the lower-left corner and select the save path.

Note: When selecting an image, the file dialog displays **BMP**, **PNG**, and **JPG** formats by default. If you need to select images in other formats, choose **"All files"** in the lower-right corner of the dialog box.



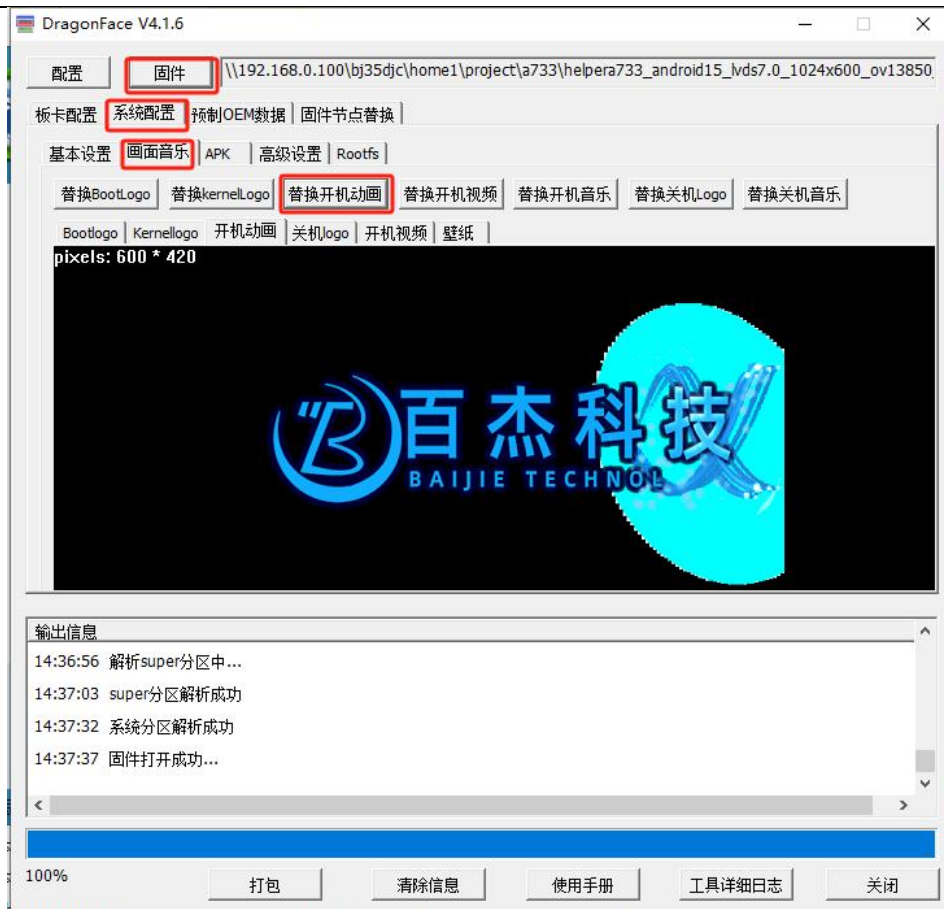


4.6.2 Change the opening animation

Open **DragonFace**:

1. Click "**Firmware**" and select the firmware image that needs to be modified for the boot animation. Wait until the firmware is loaded.
2. After the firmware is loaded successfully, select "**System Configuration**" and "**Picture and Music**".
3. Click "**Replace Boot Animation**" and select the boot animation package to be replaced (ZIP format).
4. After replacing the boot animation, click "**Pack**" in the lower-left corner and select the save path.

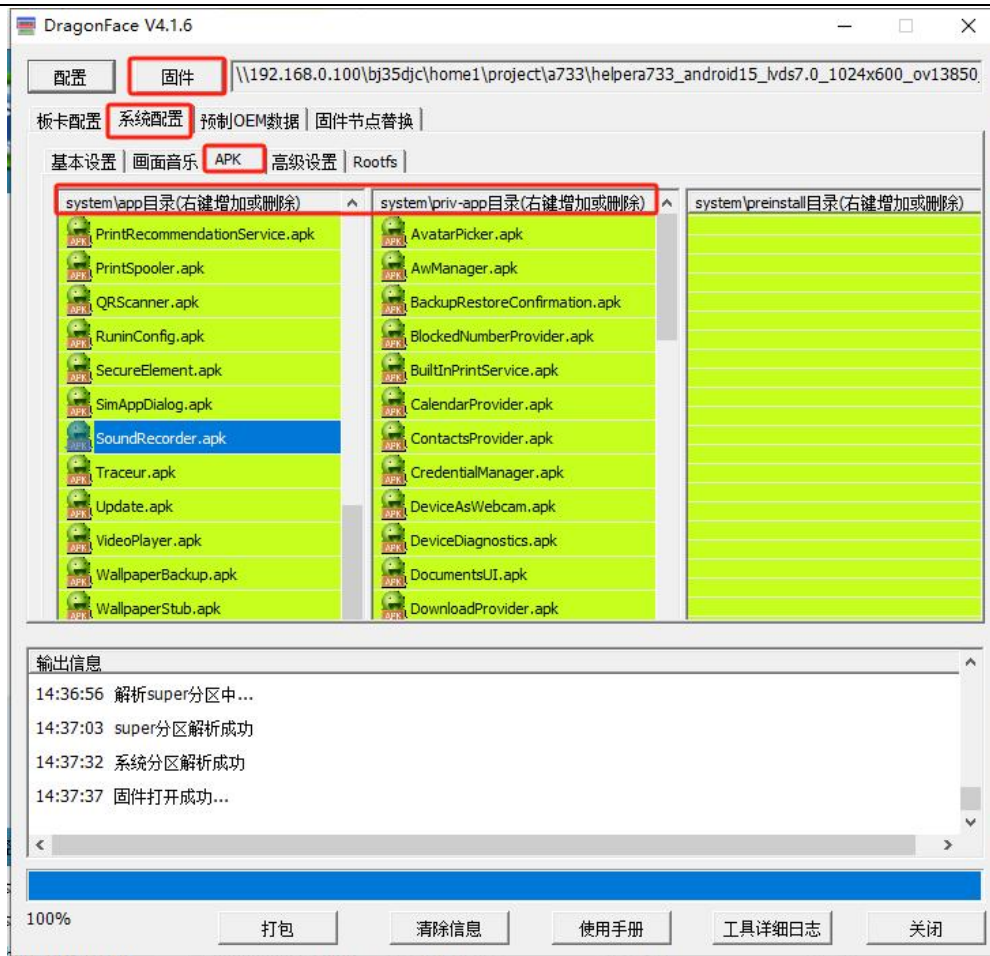
Note: The creation process of the boot animation package **bootanimation.zip** is described in **Section 8.2**.



4.6.3 Add built-in APK

Open **DragonFace**:

1. Click **"Firmware"** and select the firmware image that needs to be modified for built-in APKs. Wait until the firmware is loaded.
2. After the firmware is loaded successfully, select **"System Configuration"** and **"APK"**.
3. Right-click the APK file in the preview area to add or delete applications.
4. After completing the modifications, click **"Pack"** in the lower-left corner and select the save path.



5 Creating the firmware for the Ubuntu system

The pre-compiled **Ubuntu + Qt** system image is stored in the following directory:

firmware/gz, The file name is:

helpera733_ubuntu24.04_qt6.8.3_lvds7.0_1024x600_bj_20260627.tar.gz

After extracting the package, the image can be flashed to the development board using **PhoenixSuit**. If you need to run your own Qt applications on the development board, you can directly refer to **Chapter 6: "Remote running and debugging of Qt programs"**.

5.1 Compile the Linux kernel

Download the following files from the **sources** directory:

helpera733_android15_repo_v1.3_20260627_1758.tar.gz and **repo.tar.gz**, save them to the local machine. These files contain the **HelperA733 U-Boot source code**, **kernel source code**, **board-level configuration files**, and other related development resources.

Copy the downloaded files to the directory of the **non-root user** in the virtual machine system, and extract the files. For example, extract them to:

~/workspace/a733/

Then execute the following commands:

```
$ mkdir -p ~/workspace/a733/
$ cp helpera733_android15_repo_v1.3_20260627_1758.tar.gz repo.tar.gz ~/workspace/a733/
$ cd ~/workspace/a733/
$ tar -xvf helpera733_android15_repo_v1.3_20260627_1758.tar.gz
$ tar -xvf repo.tar.gz -C .repo/
$ .repo/repo/repo sync -l
```

After the synchronization is completed, execute the following commands in sequence to select the corresponding development board configuration.

Note: All compilation processes must not be performed using the **root** user. Compilation can only be performed using a **non-root user**.

```
$ cd longan
```

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

After entering the above command, configure the Ubuntu build information as follows:

1. Enter **"1"** for **platform** to select **Linux**.
2. Enter **"2"** for **linux_dev** to select **dragonboard**.
3. Enter **"0"** for **ic** to select **A733**.
4. Enter **"10"** for **board** to select **pro3**, which corresponds to the hardware configuration path of the **HelperA733 development board**.
5. Enter **"0"** for **flash** to select **default**.
6. Enter **"2"** for **devicetree** to select **lvds7.0_1024x600_bj**. This option can be changed according to requirements.
7. Enter **"0"** for **baijie_extra** to select **extra-qt6.8.3**. This option can be changed according to requirements.
8. Enter **"0"** for **baijie_rootfs** to select **ubuntu24.04_base.tar.gz**. This option can be changed according to requirements.

The detailed configuration is shown below:

```
All available platform:
  0. android
  1. linux
Choice [linux]: 1
All available linux_dev:
  0. bsp
  1. dragonabts
  2. dragonboard
Choice [dragonboard]: 2
All available ic:
  0. a733
Choice [a733]: 0
All available board:
  0. ag863109vcb
  1. ag863109vcb_axp517
  2. evb1
  3. evb1_acin
  4. evb1_qc_double
  5. fpga
  6. ft
  7. ft_lp5
  8. perf1
  9. pro2
 10. pro3
 11. pro3_ac101b
 12. qa
 13. qa2
 14. s6
 15. ver1
Choice [pro3]: 10
All available flash:
  0. default
  1. nor
Choice [default]: 0
All available devicetree:
  0. edp_1920x1080_bj
  1. hdm1_bj
  2. lvds7.0_1024x600_bj
  3. mipi10.1_1200x1920_bj
  4. mipi10.1_800x1280_bj
  5. mipi4.3_480x800_bj
  6. mipi5.0_720x1280_bj
  7. mipi8_800x1280_bj
Choice [lvds7.0_1024x600_bj]: 2
```

```
All available baijie_extra:
  0. extra-qt6.8.3
  1. extra-xfce
  2. no_extra
Choice [extra-qt6.8.3]: 0
Creating symlink /home1/bj35djc/helperboard/a733/longan/test/dragonboard/extra
All available baijie_rootfs:
  0. ubuntu24.04_base.tar.gz
  1. ubuntu24.04_xfce.tar.gz
Choice [ubuntu24.04_base.tar.gz]: 0
rootfs decompression done!
```

Note:

The available **devicetree** options correspond to different display configurations. By default, Baijie provides the following display configurations:

1. **edp_1920x1080_bj**: Select **0**, corresponding to the Baijie eDP display.
2. **hdmi_bj**: Select **1**, corresponding to the Baijie HDMI display.
(A development board version with HDMI support is required.)
3. **lvds7.0_1024x600_bj**: Select **2**, corresponding to the Baijie 7-inch LVDS display.
4. **mipi10.1_1200x1920_bj**: Select **3**, corresponding to the Baijie 10.1-inch high-resolution MIPI display.
5. **mipi10.1_800x1280_bj**: Select **4**, corresponding to the Baijie 10.1-inch MIPI display.
6. **mipi4.3_480x800_bj**: Select **5**, corresponding to the Baijie 4.3-inch MIPI display.
7. **mipi5.0_720x1280_bj**: Select **6**, corresponding to the Baijie 5-inch MIPI display.
8. **mipi8_800x1280_bj**: Select **7**, corresponding to the Baijie 8-inch MIPI display.

The available **baijie_extra** options correspond to different filesystem supplements. By default, Baijie provides the following options:

1. **extra-qt6.8.3**: Select **0**, corresponding to the Qt 6.8.3 system supplement.
2. **extra-xfce**: Select **1**, corresponding to the XFCE desktop system supplement.
3. **no_extra**: Select **2**, corresponding to no additional filesystem supplement.

The available **baijie_rootfs** options correspond to different root filesystem configurations. By default, Baijie provides the following options:

1. **ubuntu24.04_base.tar.gz**: Select **0**, corresponding to the Ubuntu 24.04 system.
(Must be used together with *extra-qt6.8.3*.)
2. **ubuntu24.04_xfce.tar.gz**: Select **1**, corresponding to the Ubuntu 24.04 + XFCE desktop system.
(Must be used together with *extra-xfce*.)

This configuration step is only required during the **first compilation**. Afterwards, if you modify the **devicetree** or drivers, there is no need to configure it again. You can directly start the compilation process.

After completing the configuration selection, execute the following command to start compiling:

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

Build Notes:

1. Please use the virtual machine image **helperboard_ubuntu20.04** provided by us.
Otherwise, the compilation may fail.
2. After synchronizing the source code, the first kernel compilation must be completed first.
Then enter the **bsp** directory and execute the following command. After that, compile the kernel again.

```
$ cd ~/workspace/a733/kernel/linux-6.6/bsp  
$ git checkout .
```

3. After selecting the display configuration, the corresponding configuration files will overwrite **board.dts** in: **device/config/chips/a733/configs/pro3/linux-6.6** ; **uboot-board.dts** in:

device/config/chips/a733/configs/pro3. If the platform (**Android or Linux**) is not changed, you can directly compile after modifying the source code. There is no need to select the configuration again.

4. If you need to change the platform, please back up **board.dts** and **uboot-board.dts** first.
Otherwise, these files will be overwritten, and all previous modifications will be lost.

5.2 Compile u-boot

Enter the **U-Boot** directory and compile directly.

```
$ cd ~/workspace/a733/longan  
$ cd brandy/brandy-2.0/u-boot-2018/  
$ make sun60iw2p1_a733_defconfig && make -j
```

5.3 Compile and package the Ubuntu + Qt file system

When executing `./build.sh` config in **Section 5.1** for configuration, select the following options:

- **baijie_extra**: select **extra-qt6.8.3**
- **baijie_rootfs**: select **ubuntu24.04_base.tar.gz**

Then execute the following command to compile and package the firmware:

```
$ ./build.sh && ./build.sh pack
```

5.4 Compile and package the Ubuntu + Xfce file system

When executing `./build.sh` config in **Section 5.1** for configuration, select the following options:

- **baijie_extra**: select **extra-xfce**
- **baijie_rootfs**: select **ubuntu24.04_xfce.tar.gz**

```
$ ./build.sh && ./build.sh pack
```

5.5 Notes for Attention

1. The account information of the flashed development board is as follows:

- Root user account: **root**
- Normal user account: **szbaijie**
- Password for both root and normal users: **szbaijie**

2. If the code under the **longan** directory has been modified, the entire project must be recompiled and repackaged. Directly packaging the firmware without recompilation is not supported, as shown below:

```
$ cd ~/workspace/a733/longan
```

```
$ ./build.sh && ./build.sh pack
```

6 Remote running and debugging of Qt programs

6.1 Copy the compiler to the virtual machine

Copy **helper733_ubuntu24.04_aarch64_glibc2.39_gcc13.3_qt6.8.3.tar.gz** from the following directory in the shared network drive:**tools/toolchain_for_linux** to the /opt/toolchain directory of the **helperboard_ubuntu20.04** virtual machine.

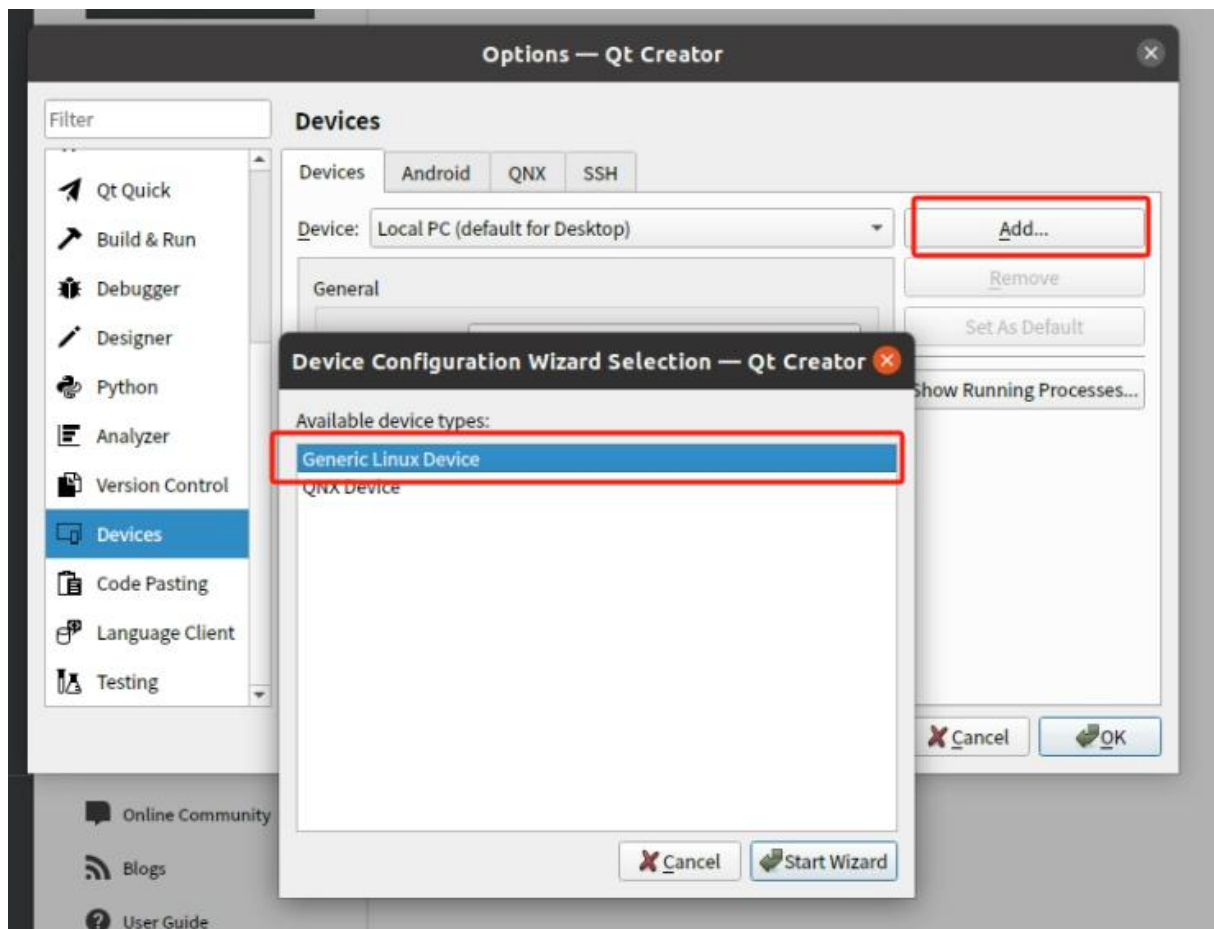
Extract the archive and configure the environment variables by executing the following commands:

```
$ sudo mkdir -p /opt/toolchain
$ cd /opt/toolchain
$ sudo tar -xf helper733_ubuntu24.04_aarch64_glibc2.39_gcc13.3_qt6.8.3.tar.gz
```

6.2 Configuration of the QtCreator environment in the virtual machine

6.2.1 Add remote device

To enable remote application deployment and execution, you need to add a remote device in **Qt Creator**. Open the **Qt Creator** application pre-installed in the virtual machine, then navigate to: **Tools** → **Options** → **Devices** → **Devices** → **Add** to add a new remote device, as shown below:

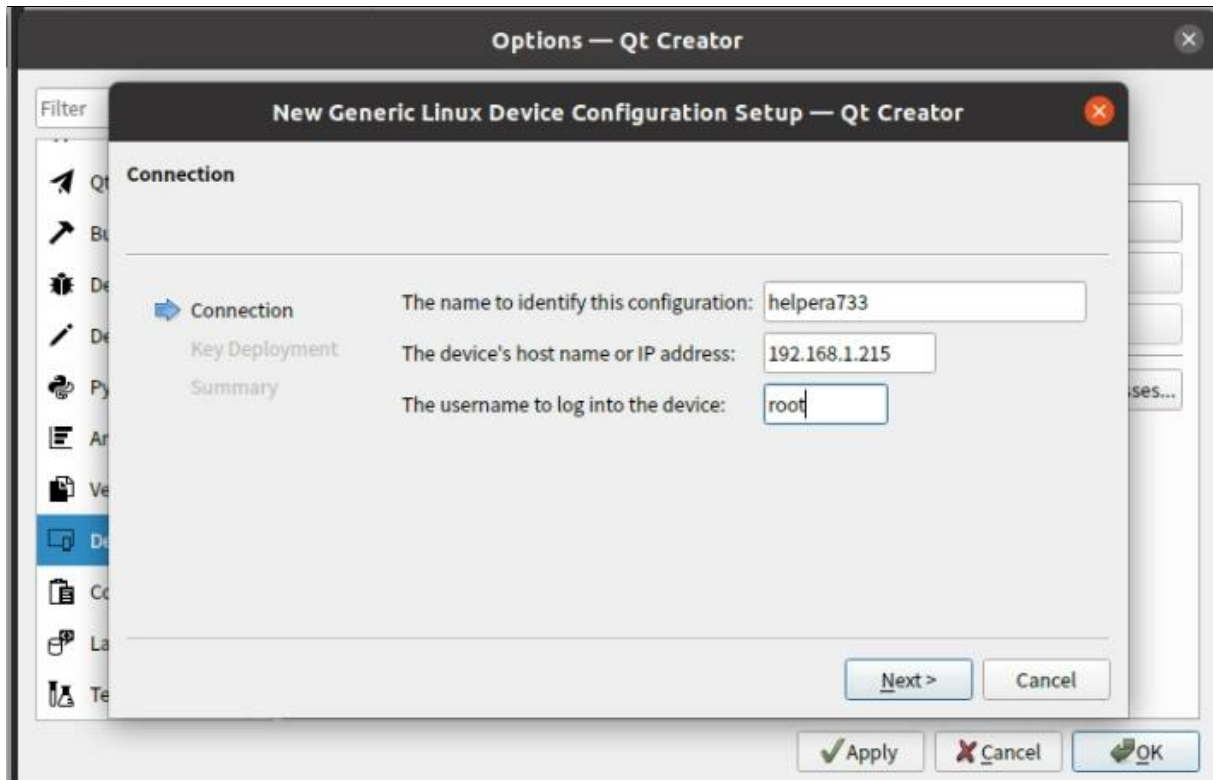


Enter the following information:

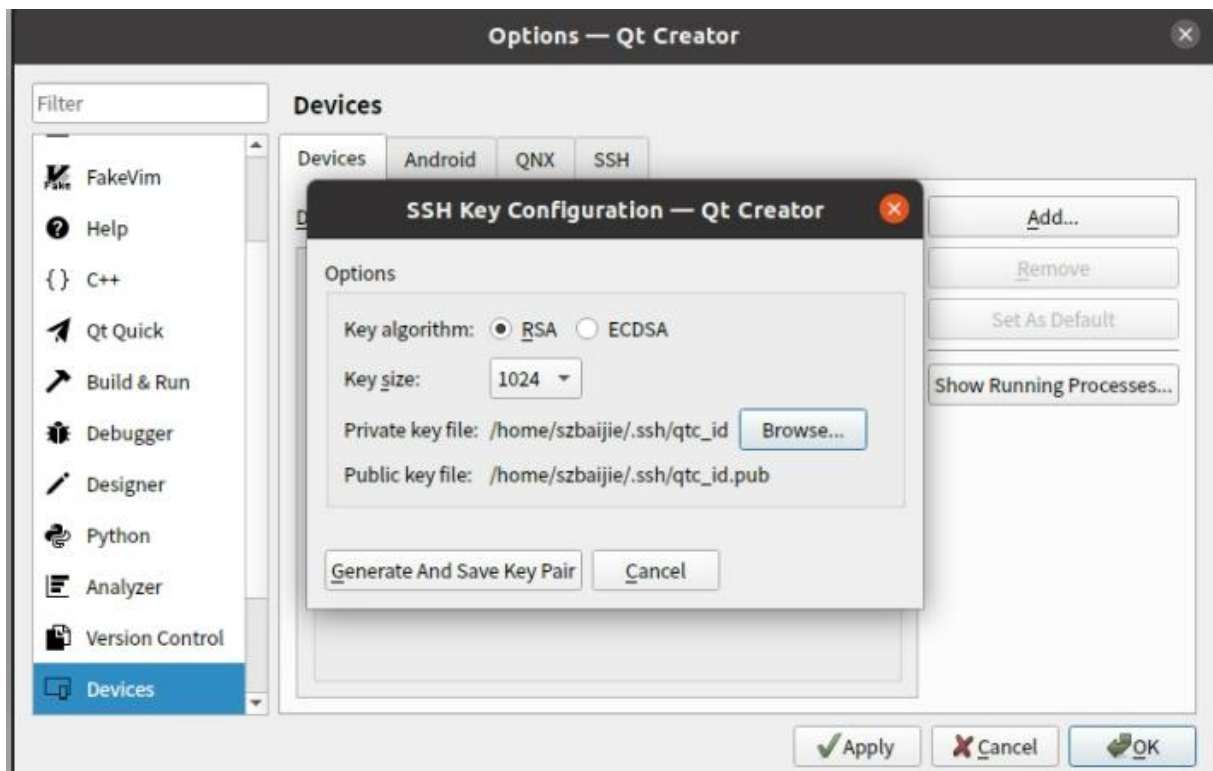
- **Username** of the development board
- **Password** of the development board
- **IP address** of the development board

The configuration is shown below:

页



Create a new SSH key.

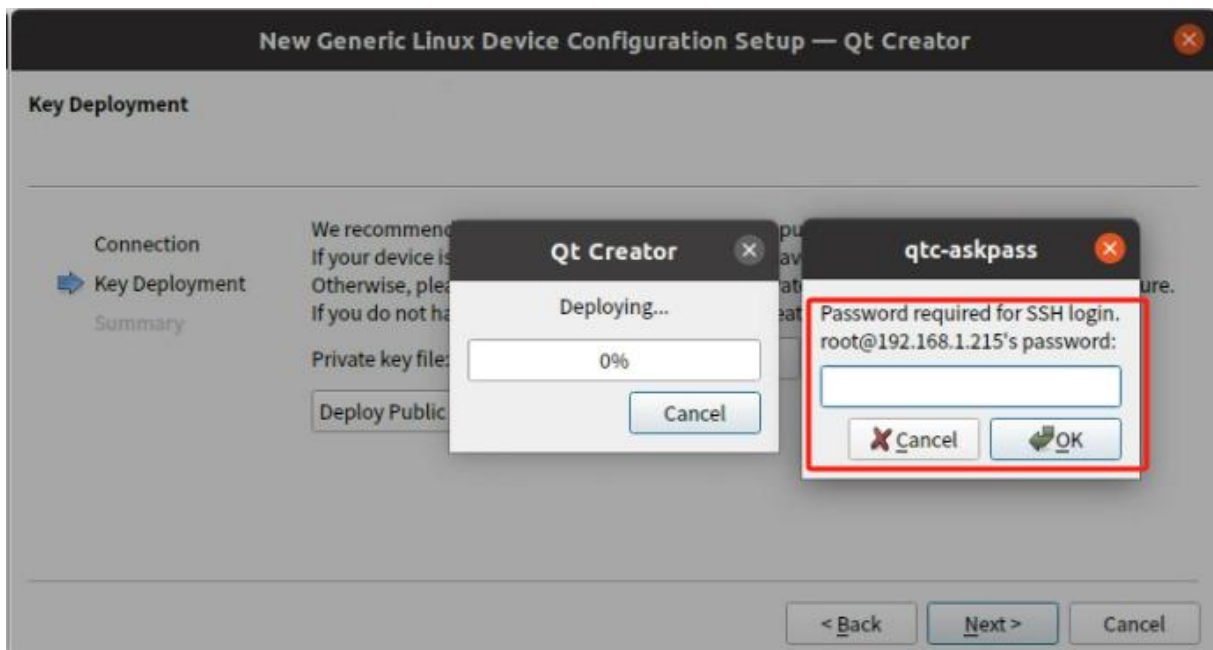


页

After the SSH key has been created, click **"Deploy Public Key"**.



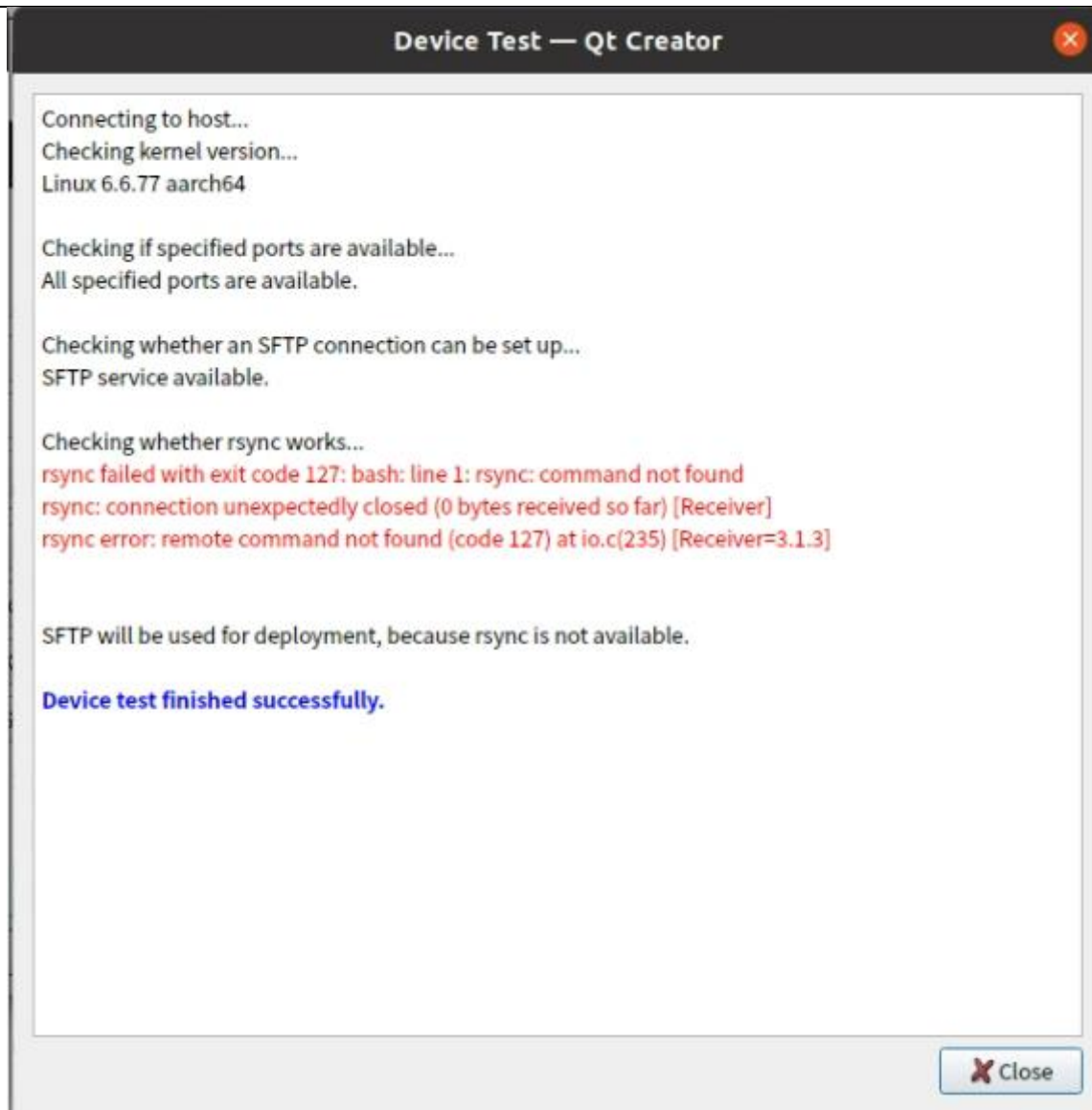
Enter the password for the development board login account.



After the public key is deployed successfully, click **"Next"**.



Click **"Finish"**. After the configuration is completed, the remote device will be connected successfully, as shown below.



Note: The IP address of the development board must be obtained by logging in to the board through the serial console (refer to **Section 2.2, Using Xshell**). Execute the following command to view the IP address:

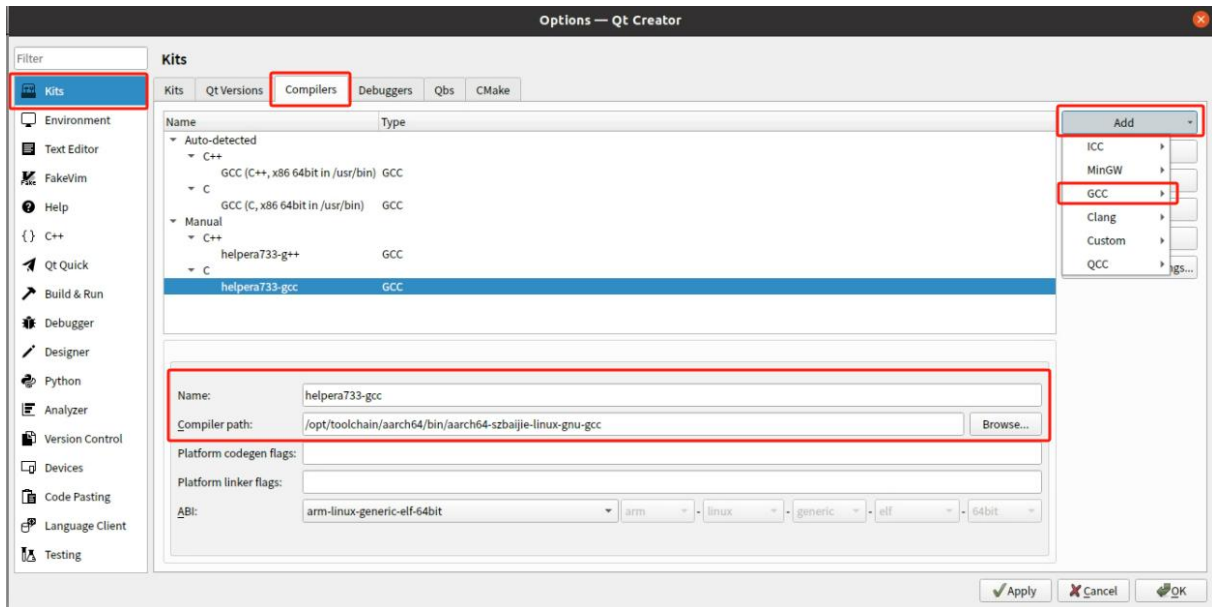
```
$ ifconfig
```

页

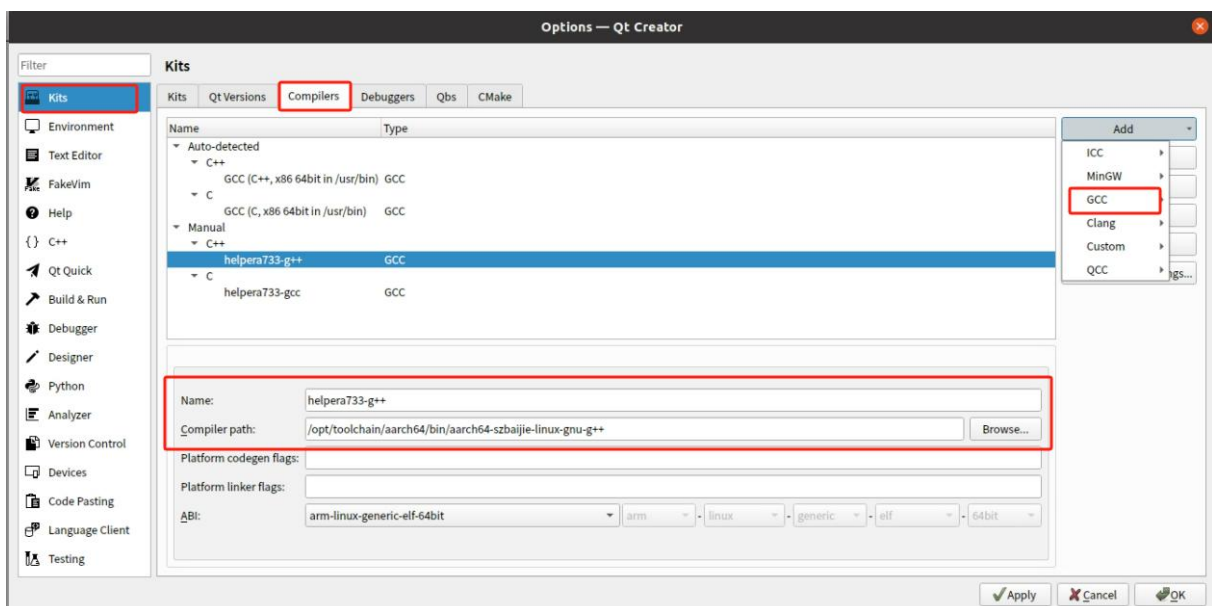
6.2.2 Set up the compiler

After copying the toolchain in **Section 6.1**, you also need to configure the toolchain in **Qt Creator**. Open the **Qt Creator** application pre-installed in the virtual machine, then navigate to: **Tools** → **Options** → **Kits** → **Compilers**. Configure the **C compiler** as shown below:

Note: When configuring the compiler, set the **ABI** option to **arm-linux-generic-elf-64bit**.

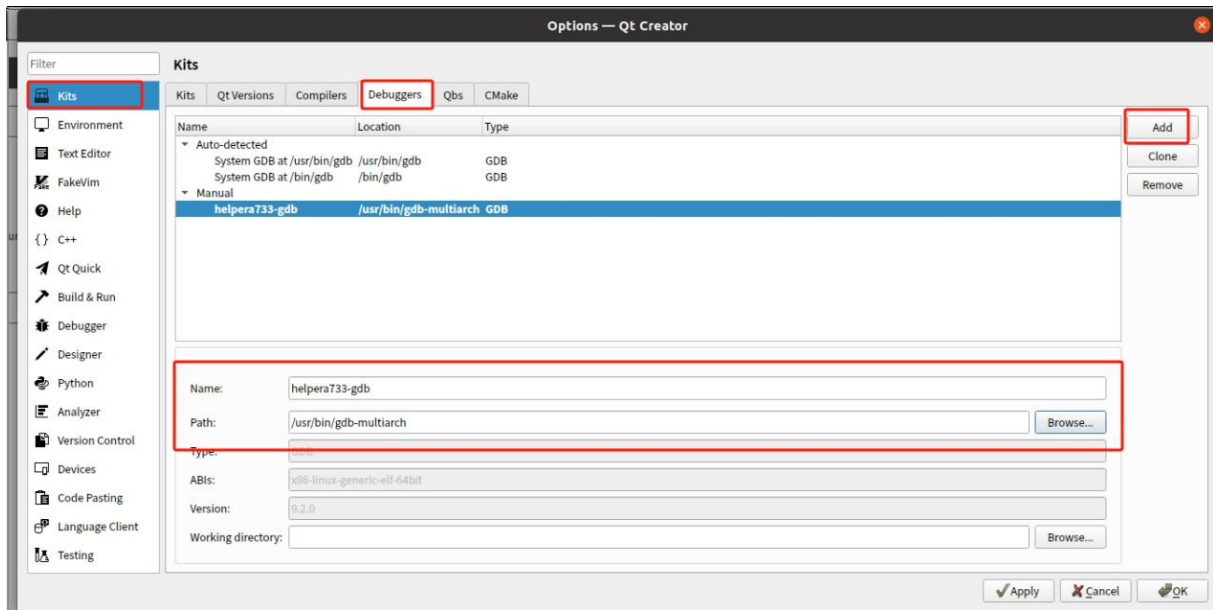


Then configure the **g++ compiler** as follows:



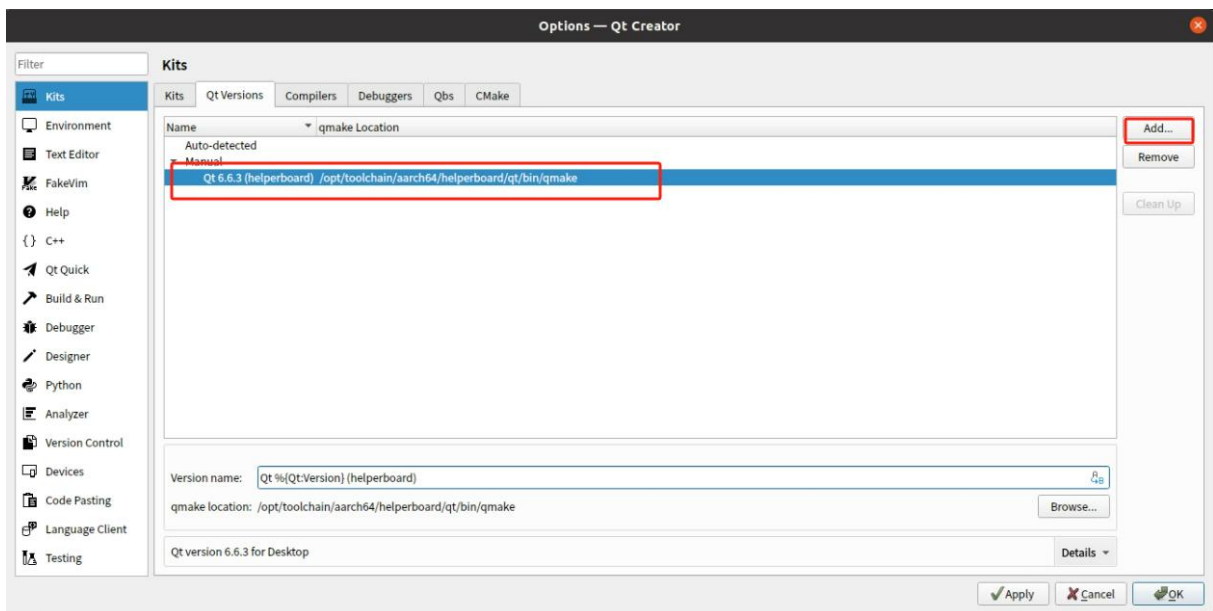
Then configure the **GDB debugger** as follows:

页



6.2.3 Set the Qt version

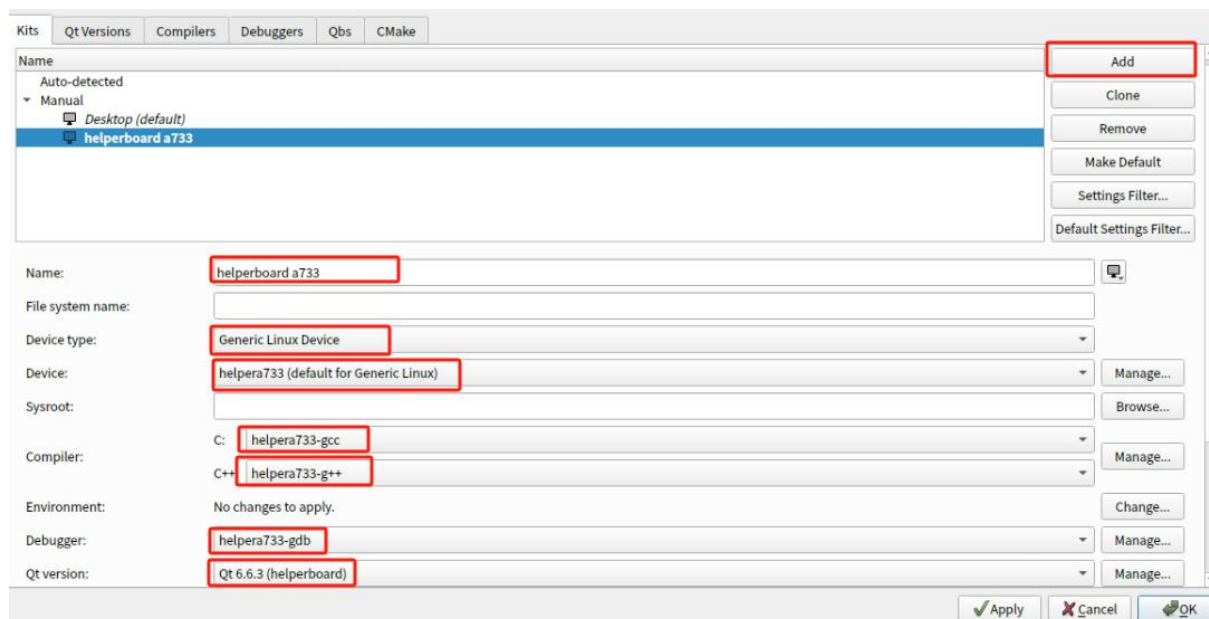
If an exclamation mark or other warning symbols appear when configuring the Qt version, it indicates that there may be a problem with the related libraries. Please check whether the correct compiler has been selected.



6.2.4 Set kits

After completing the other configurations, configure the **Kits** settings as follows:

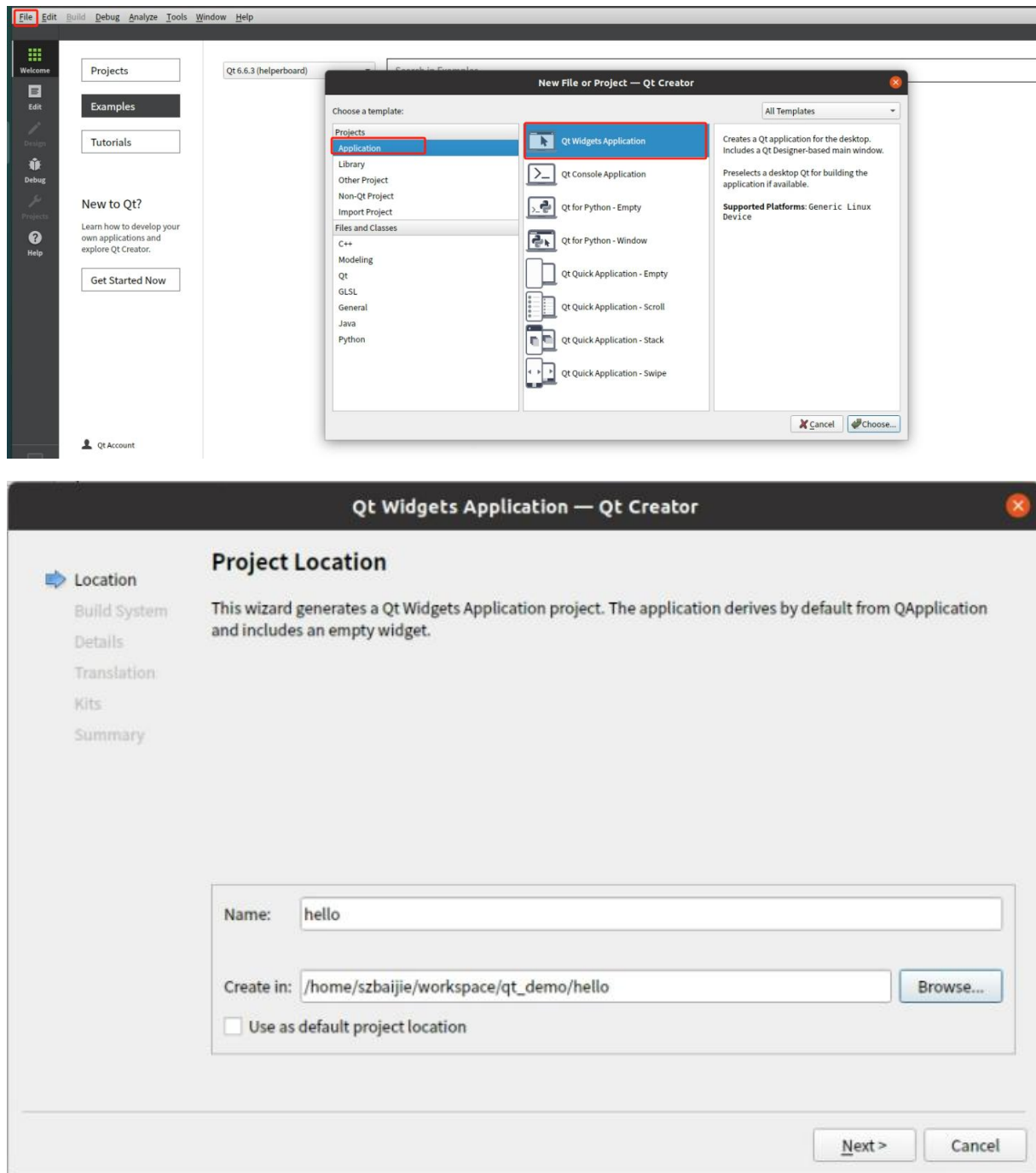
Note: If the configuration is not successful, an exclamation mark will appear. In this case, please check whether the provided compiler has been selected correctly.



Remote Debugging and Running Applications with Qt Creator in the Virtual Machine

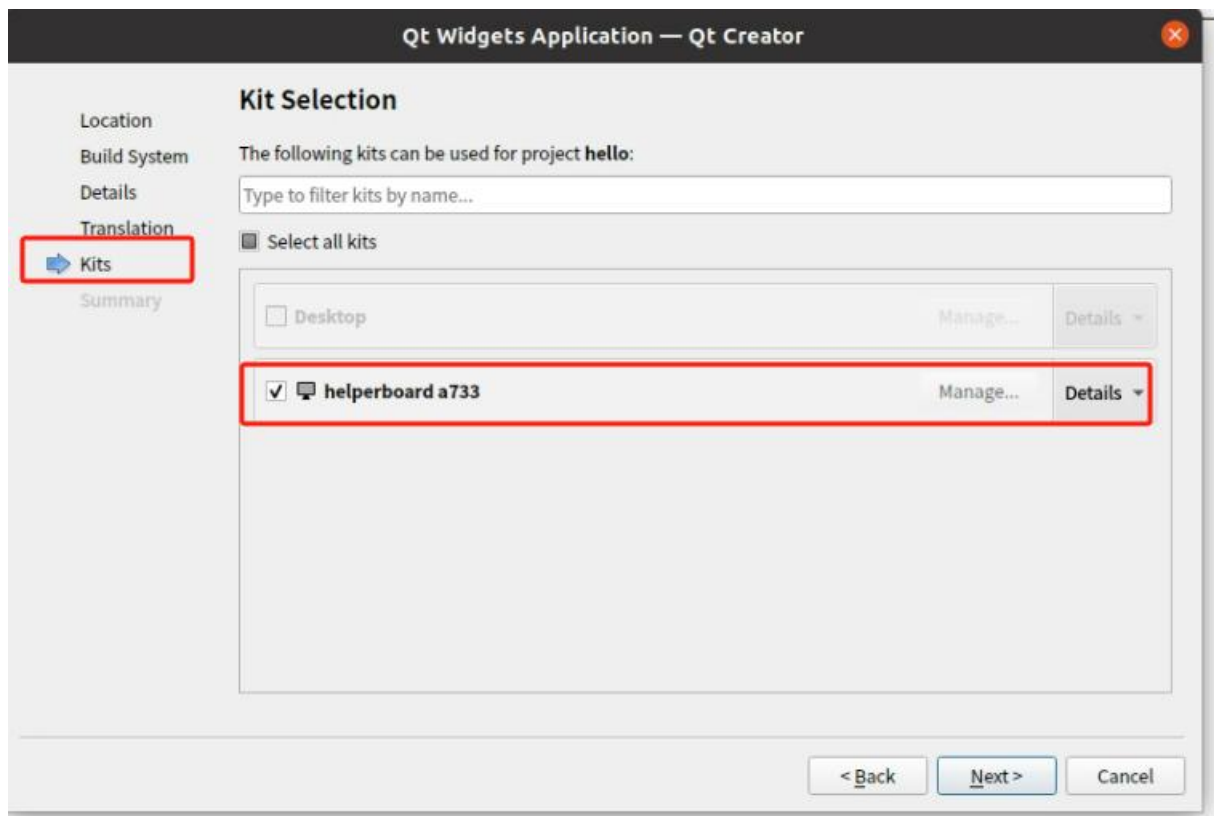
6.2.5 Create a new Qt project

Open **Qt Creator**, then navigate to: **File** → **New File or Project** .to create a new project, as shown below.



After clicking **"Next"** in the previous step, continue until the following page is displayed.

At this point, select the **Kit** that we configured previously. In this example, the selected Kit is **HelperBoard A733**.



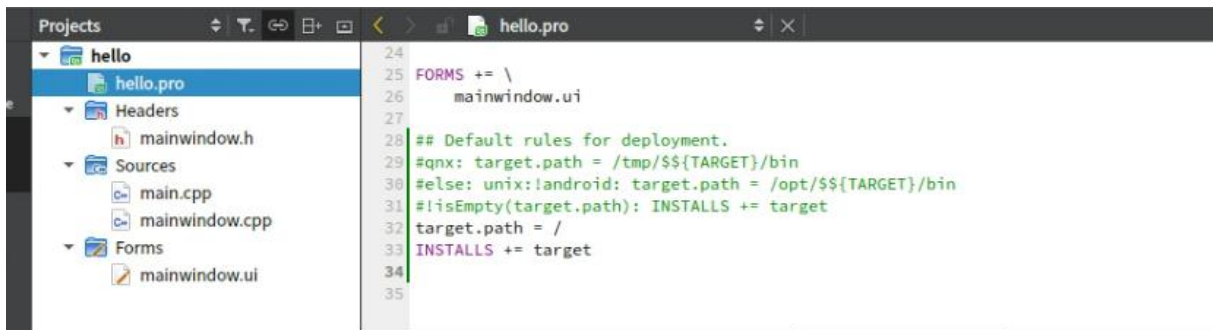
After selecting the Kit, click **"Next"** → **"Finish"** to complete the project creation.

6.2.6 Set the path for the running program

At this point, modify the corresponding code in the project file **hello.pro**. The required modifications are shown in the following command:

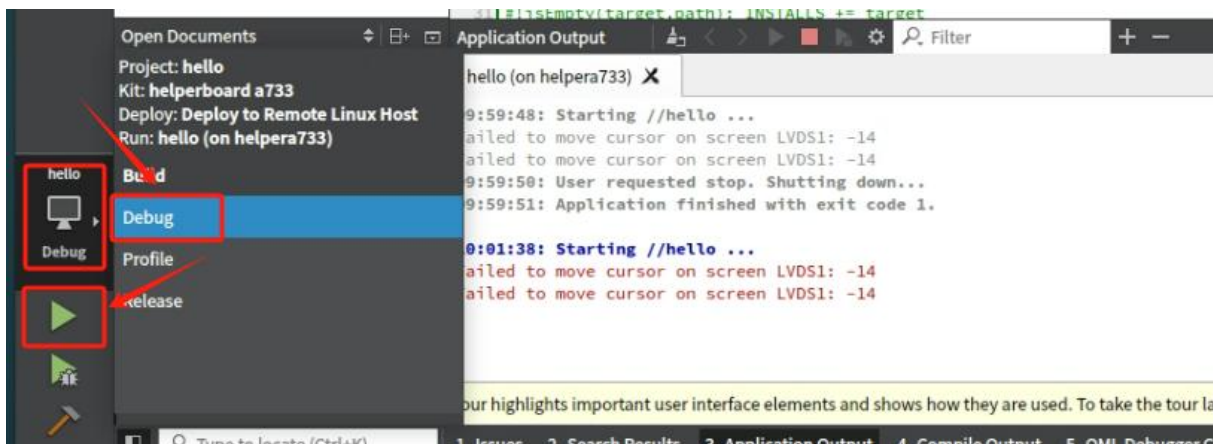
```
target.path = /  
INSTALLS += target
```

These two commands indicate that the application can be remotely run in the **root directory** of the development board.

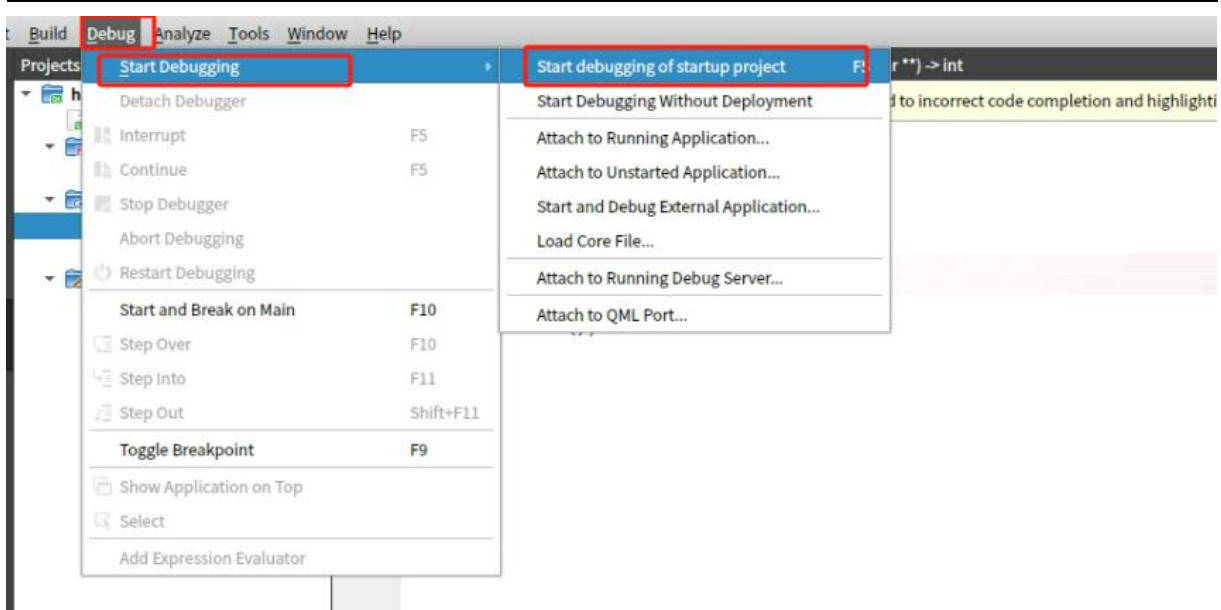


6.2.7 Operation and Debugging

Navigate to **Debug** → **Debug** and select the configured Kit. After confirming the selection, click the **Run** button (triangle icon) to execute the application. As shown below, the application has been successfully executed.



If debugging is required, navigate to: **Debug** → **Start Debugging** → **Start Debugging of Startup Project** to start debugging. The detailed debugging procedures can be learned independently.

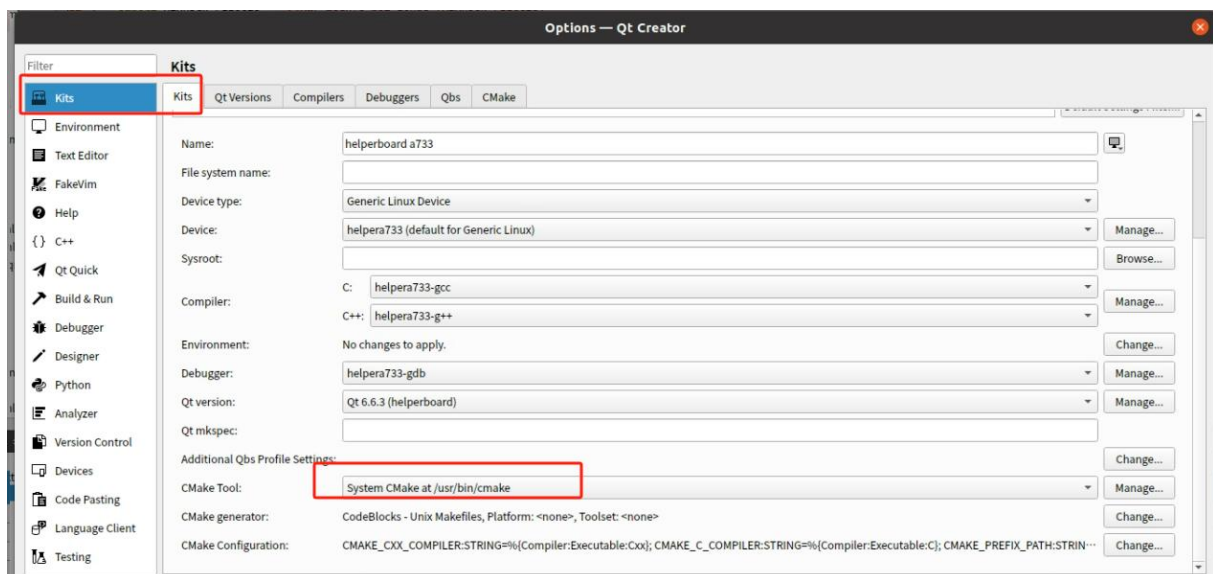


6.3 Using Cmake for cross-compilation in Qtcreator

If you use **CMake** to configure and build the project, the following configuration needs to be added. First, make sure that the **CMake** tool is installed in the virtual machine system.

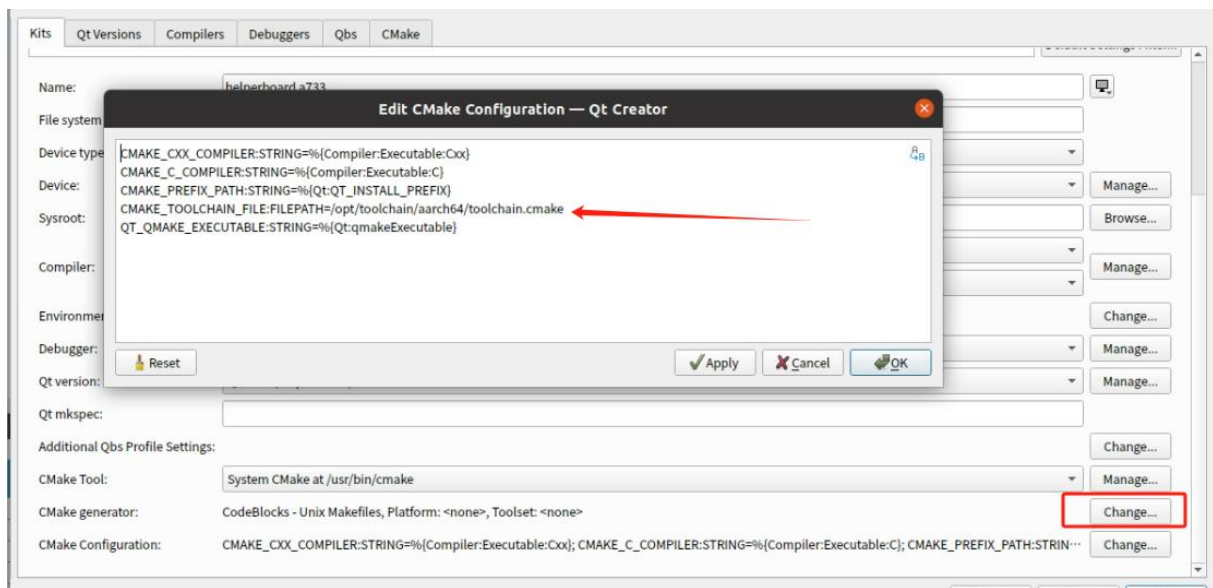
```
$ sudo apt update && apt install cmake -y
```

Open **Qt Creator**, navigate to the following configuration page, and select the system **CMake** tool.

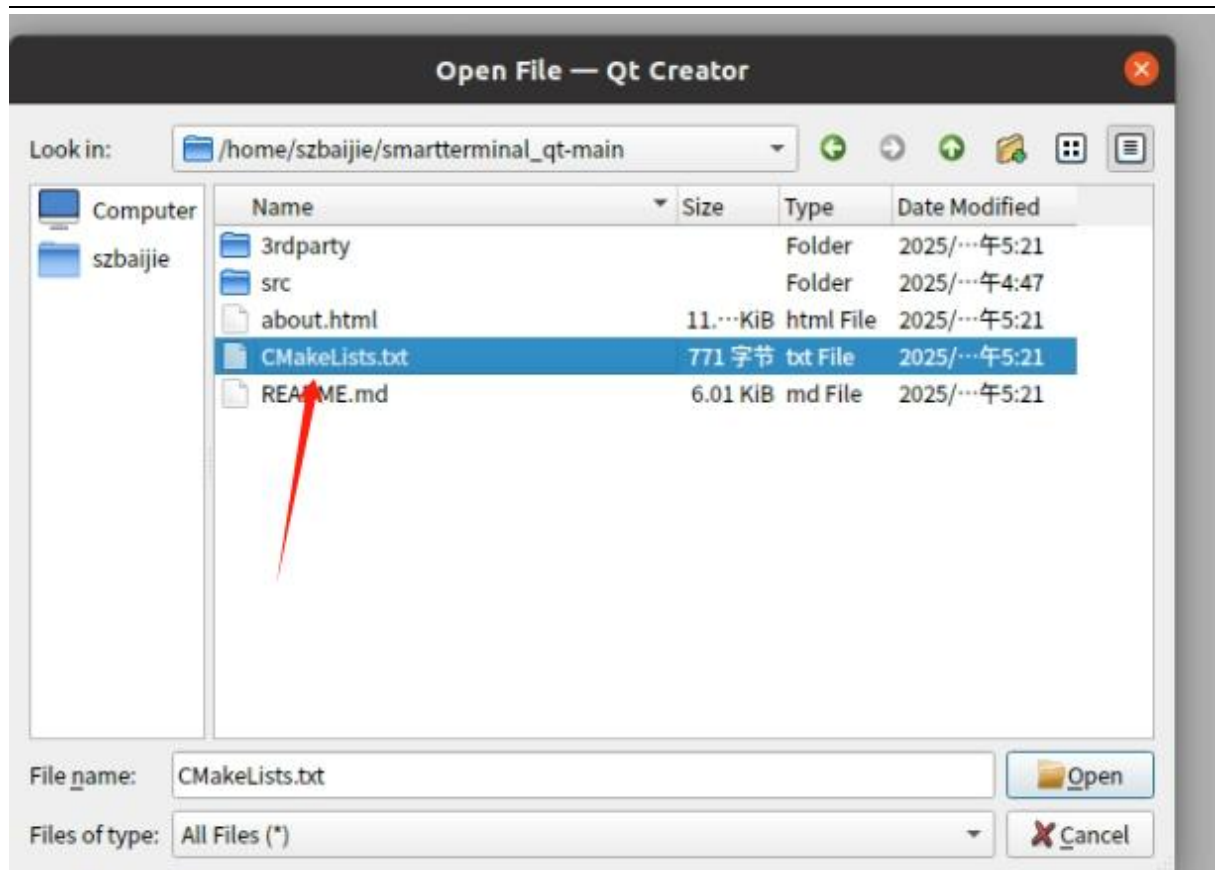


Modify the **CMake Configuration** settings as follows. Add one line to specify the **TOOLCHAIN_FILE** configuration file. The content of this file is the same as described above. It is recommended to first compile the project successfully using the command line before configuring Qt Creator to use it.

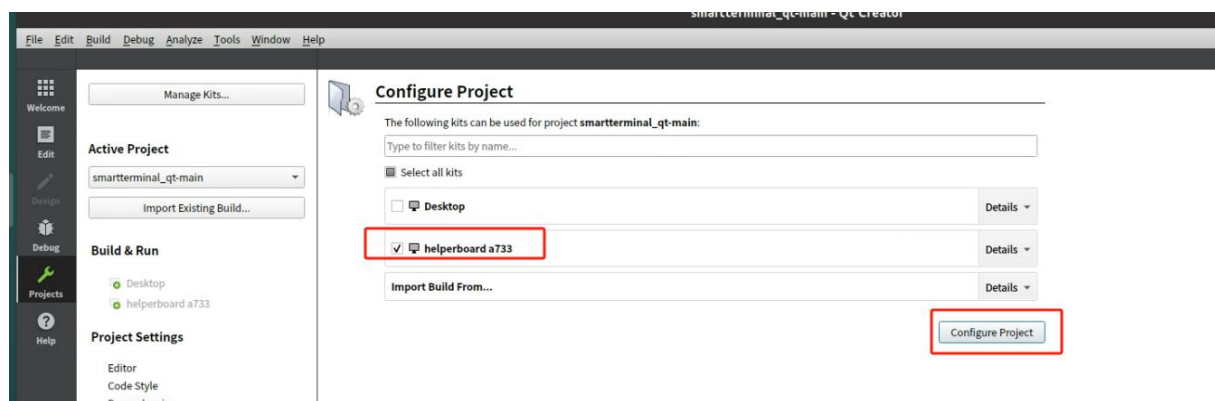
```
$ CMAKE_TOOLCHAIN_FILE:FILEPATH=/opt/toolchain/aarch64/toolchain.cmake
```



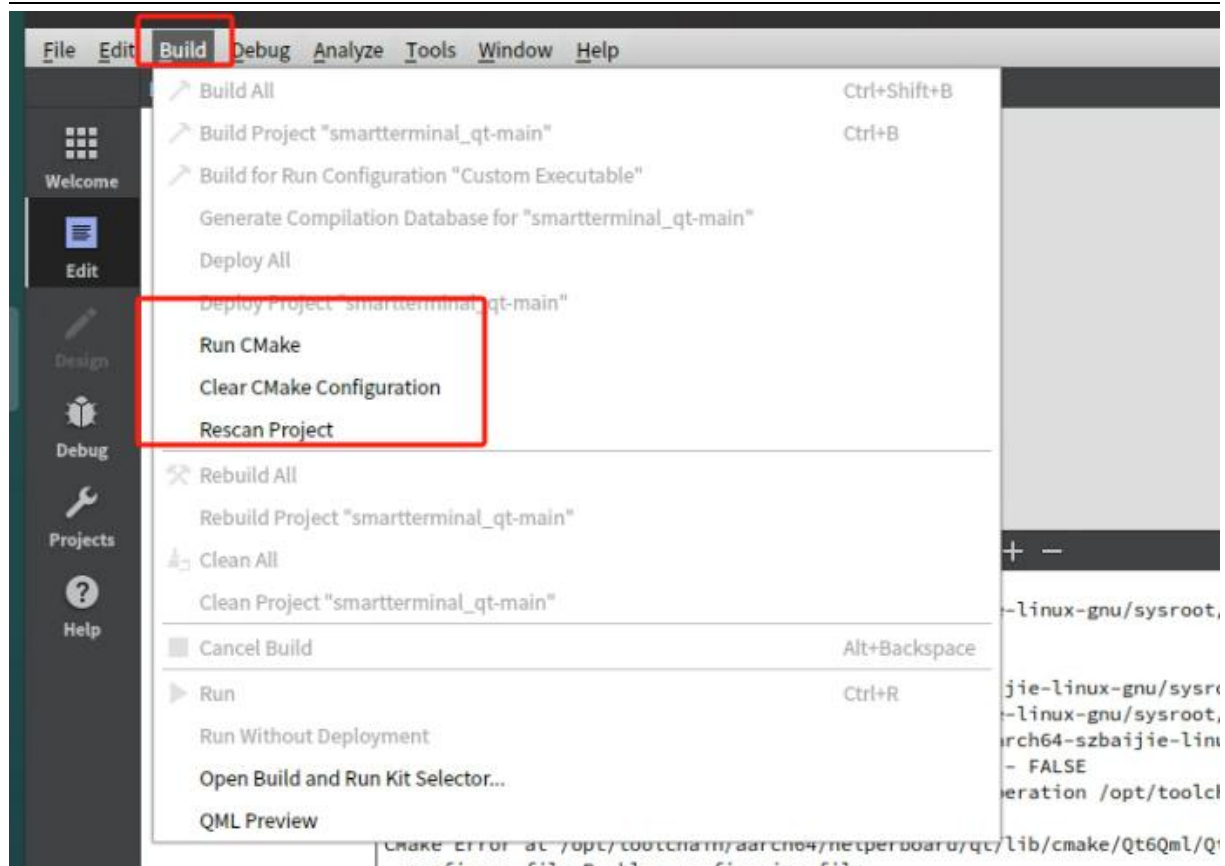
After modifying the configuration, open the Qt project.



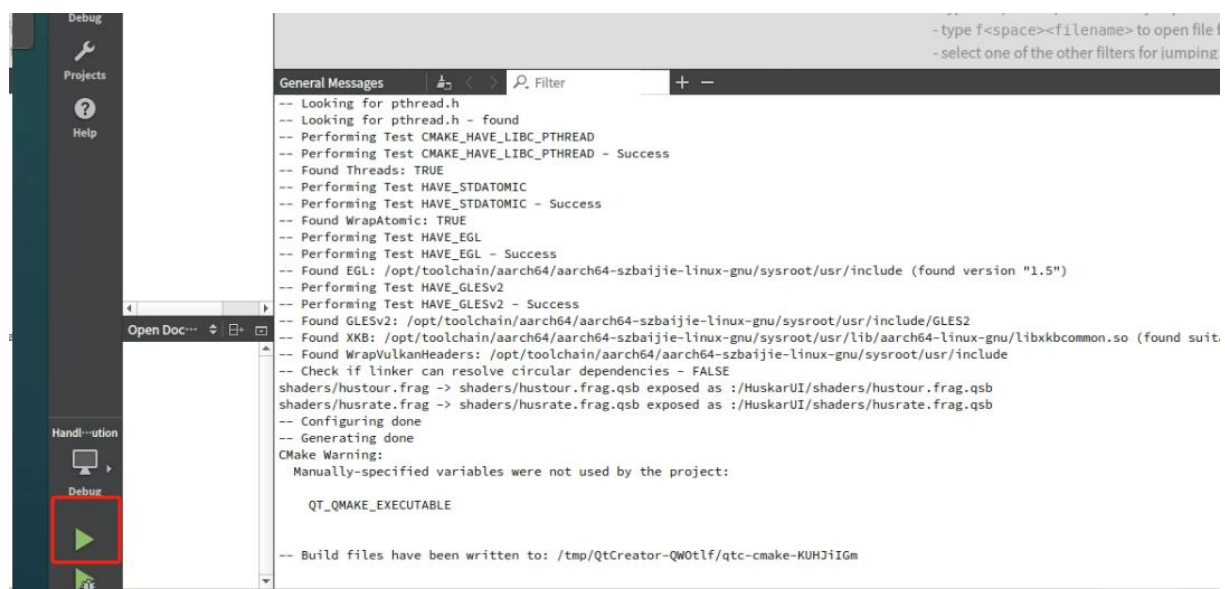
Select the target development board and click **"Configure Project"**.



After modifying the configuration, you can clean the project and rebuild it.



After the build is completed successfully, click **"Build"** to compile the project.



6.4 Set the development board to start up automatically with Qt

6.4.1 Start up automatically on the already burned board

The default **qt_env.sh** file is located in the **/usr/helperboard/** directory. If you need to modify the environment variables, you can edit this file accordingly.

The default Qt application started by the development board is configured in **/etc/init.d/rcS**. If you need to replace it with your own Qt application, first use the **scp** command to copy the Qt application to the development board.

Here, **BaijieDemo** is used as an example.

```
$ scp BaijieDemo root@192.168.1.187:~
```

Note: Replace **BaijieDemo** with your own Qt application name. In the command, **root** is the username used to log in to the development board. The part after **"@"** is the IP address of the development board, and the path after the colon **":"** is the destination directory on the development board. Modify these parameters according to your actual situation.

The successful file transfer result is shown below.

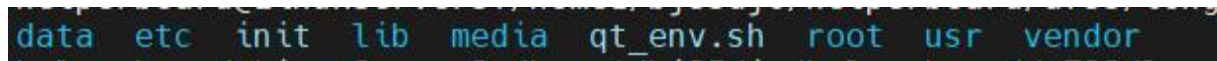
```
bj35dic@LinuxServer3:~/A733$ scp BaijieDemo root@192.168.1.187:~
BaijieDemo
100% 1543KB 11.0MB/s 00:00
bj35dic@LinuxServer3:~/A733$
```

After transferring the Qt application to the development board, replace the Qt application startup command in **/etc/init.d/rcS** with your own command, as shown in the red box below:

```
. /usr/helperboard/qt_env.sh
#/usr/helperboard/examples/opengl/helloqles3/helloqles3 &
/usr/helperboard/examples/quick3d/helloqtquick3d/helloqtquick3d &
```

6.4.2 Set up self-starting in the firmware

After selecting the corresponding **extra** option according to Section 5.3, enter the directory:**longan/test/dragonboard/extra**.During the build process, the files in the **extra** directory will replace the corresponding files in the **rootfs**.



Therefore, we only need to modify **/etc/init.d/rcS** in the **extra** directory to integrate the Qt application into the firmware.After rebuilding and flashing the firmware to the development board, the Qt application will start automatically after boot.

7 Creating the firmware for the Android system

7.1 Compile the Linux kernel

Before compiling **Android**, the **Linux kernel** must be compiled first. The Linux kernel compilation process is the same as described in **Section 5.1**, except that the Android configuration options are different.

The configuration procedure is as follows:

```
$ cd ~/workspace/a733/longan  
$ ./build.sh config
```

After entering the command above, configure the Ubuntu information as follows:

1. Enter **"0"** for **platform** to select **Android**.
2. Enter **"0"** for **ic** to select **A733**.
3. Enter **"10"** for **board** to select **pro3**, which corresponds to the hardware configuration path of the **HelperA733 development board**.
4. Enter **"0"** for **flash** to select **default**.
5. Enter **"2"** for **devicetree** to select **lvds7.0_1024x600_bj**. This option can be changed according to your requirements.

The configuration details are shown below:

```
All available platform:
  0. android
  1. linux
Choice [android]: 0
All available ic:
  0. a733
Choice [a733]: 0
All available board:
  0. ag863109vcb
  1. ag863109vcb_axp517
  2. evb1
  3. evb1_acin
  4. evb1_qc_double
  5. fpga
  6. ft
  7. ft_lp5
  8. perf1
  9. pro2
 10. pro3
 11. pro3_ac101b
 12. qa
 13. qa2
 14. s6
 15. ver1
Choice [pro3]: 10
All available flash:
  0. default
  1. nor
Choice [default]: 0
All available devicetree:
  0. edp_1920x1080_bj
  1. hdm_i_bj
  2. lvds7.0_1024x600_bj
  3. mipi10.1_1200x1920_bj
  4. mipi10.1_800x1280_bj
  5. mipi4.3_480x800_bj
  6. mipi5.0_720x1280_bj
  7. mipi8_800x1280_bj
Choice [lvds7.0_1024x600_bj]: 2
```

After completing the configuration selection, execute the following command to start the compilation:

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

Note: Please use the provided **helperboard_ubuntu20.04** virtual machine. Otherwise, the compilation may fail.

7.2 Compile u-boot

Enter the **U-Boot** directory and compile directly.

```
$ cd ~/workspace/a733/longan
$ cd brandy/brandy-2.0/u-boot-2018/
$ make sun60iw2pl_a733_defconfig && make -j
```

7.3 Compilation and packaging of Android

Return to the main directory and configure the development environment:(**The environment needs to be configured again after exiting the terminal or opening a new terminal.**)

```
$ cd ~/workspace/a733/
$ source build/envsetup.sh
```

Run the **lunch** command:

```
$ lunch
```

Note: Enter "**20**" to select the configuration corresponding to the development board: **a733_pro3_arm64-ap3a-userdebug** You can also directly enter:**lunch 20** to select the configuration.

Compile Android:

```
$ make -j8
```

The compilation will then start, and you need to wait until the process is completed. Depending on the computer configuration, the compilation may take anywhere from one hour to several hours.

Note: If the compilation fails, remove the **-j** option and the following number from the build command. The number after **-j** indicates the number of parallel compilation threads. In this example, the maximum number of threads used is **8**, which can generally be set to twice the number of CPU cores. Increasing the number of parallel tasks will require more memory.

If the compilation still fails after removing the **-j** option, run the following command and compile again until the compilation succeeds:

```
$ make installclean  
$ make
```

Note: If the compilation still fails, try compiling several more times.

After the compilation is completed, use the following command to package and generate the Android flashing image:

```
$ pack
```

The final generated image is located at: **longan/out/a733_android15_pro3_uart0.img**
Then, you can use the flashing tool "**PhoenixSuit**" (refer to [Section 4.3](#)) to flash the image to the development board.

8 Other operations

8.1 Modify the startup logo of Android

The boot logo image file must be named **bootlogo.bmp**. Replace the existing **bootlogo.bmp** files in the following directories: `longan/device/config/chips/a733/configs/pro3/dragonboard` and `longan/device/config/chips/a733/configs/pro3/android` with your customized boot logo image.

Note:

1. When replacing the **bootlogo.bmp** image, the bitmap color depth must be set to **32-bit** during the BMP conversion process.
2. If the boot logo image file name is incorrect, an error will occur during compilation.

The image format can be converted using image conversion software. In this example, **Adobe Photoshop** is used for the conversion process, as shown below:



After the conversion is completed, replace the **bootlogo.bmp** file and return to the **longan** directory to compile the project. The procedure is shown below:

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

After the compilation is completed, package the firmware according to the Android or Ubuntu firmware packaging process. The generated image will be located in the **longan/out** directory.

Note: For details, refer to **Chapter 5: Ubuntu Firmware Creation** and **Chapter 7: Android Firmware Creation**.

8.2 Modify the startup animation and startup music of Android

The **bootanimation.zip** package file for the boot animation can be created using the following steps.

8.2.1 Directory Structure Explanation

The **bootanimation.zip** file contains the **part0** and **part1** folders, as well as the **desc.txt** file. The **part0** and **part1** folders contain the animation frames. The images in **part0** are displayed only once, while the images in **part1** are displayed repeatedly in a loop. The image format can be **PNG** or **JPG**. The directory structure of the boot animation is shown below:

名称	修改日期	类型	大小
part0	2020/11/25 16:50	文件夹	
part1	2020/11/25 16:49	文件夹	
desc.txt	2015/7/28 19:55	文本文档	1 KB

Example of the part0 directory:

名称	修改日期	类型	大小
Animation_00000.png	2015/7/28 11:53	PNG 文件	2 KB
Animation_00001.png	2015/7/28 11:52	PNG 文件	2 KB
Animation_00002.png	2015/7/28 11:52	PNG 文件	2 KB
Animation_00003.png	2015/7/28 11:52	PNG 文件	3 KB

The content of the **desc.txt** file is as follows:

```
500 350 20
p 1 0 part0
p 0 0 part1
```

The first line defines the image resolution and frame rate. The values **500** and **350** in the first line must match the image resolution. **500** represents the image width, and **350** represents the image height. (**Note: The image resolution should not exceed the screen resolution.**)

The value **20** represents the frame rate (FPS). Starting from the second line, **p** is the identifier. The second parameter indicates the number of playback loops (**0** means infinite looping). The third parameter indicates the number of frames to pause between two loops. The fourth parameter indicates the corresponding directory name. During animation playback, the images will be played automatically in the order of their file names.

8.2.2 Boot-up music

If boot-up sound is required, place the boot sound file in the **part0** directory and rename it to **audio.wav**.

Add the **audio_conf.txt** file to the root directory. The content is as follows:

```
card=0
device=0
period_size=4096
period_count=8
#Input_Event or Switch_State
Headset detection=Switch_State
switch_path=/sys/module/snd_soc_sunxi_common/parameters/jack_state
```

```
single_volume_control=1
speaker_headphones_out=0
mixer "SPK Volume"=20
mixer "OutputR Mixer DACR Switch"=1
mixer "OutputL Mixer DACL Switch"=1
mixer "LINEOUTL Switch"=1
mixer "LINEOUTR Switch"=1
mixer "SPK Switch"=1
headset mixer "OutputR Mixer DACR Switch"=1
headset mixer "OutputL Mixer DACL Switch"=1
headset mixer "HPOUTL Switch"=1
headset mixer "HPOUTR Switch"=1
headset mixer "SPK Switch"=0
```

8.2.3 Packaging and Compilation

When using a packaging tool on **Windows**, select the **ZIP** format and set the compression method to **"Store"**.

On **Linux**, enter the **bootanimation** directory and package it using the following command:

```
$ zip -0 -r ../bootanimation.zip ./*
```

Note: The Linux command uses the **-0** option to set the compression level to the lowest level (**Stored**), which only archives the files without compression. Otherwise, the animation may be displayed as a black screen due to an incompatible package format.

Then replace the packaged file with **bootanimation.zip** in the following directory:
device/softwinner/jupiter/common/media/bootanimation and modify the file permissions:

```
$ chmod 775 bootanimation.zip
```

Return to the **android** directory, clear the previous build configuration, and then package the firmware:

```
$ make installclean
$ make -j8 && pack
```

The final generated image is located at: `longan/out/a733_android15_pro3_uart0.img`

8.3 Modify screen rotation in Android

For screen rotation on the **A733**, modify the rotation direction in the following file:

`device/softwinner/jupiter/a733-pro3/device-common.mk`

Modify the touch direction in the following file:

`device/softwinner/jupiter/a733-pro3/input/gt9xxnew_ts.idc`

The boot logo uses a fixed image orientation, so the boot logo direction will not be changed with the screen rotation.

Open the **device-common.mk** file and find the **PRODUCT_PROPERTY_OVERRIDES** property. The rotation angle is defined as a clockwise rotation angle.

For example, to rotate the screen **270° clockwise**, add the following configuration:

```
PRODUCT_PROPERTY_OVERRIDES += \  
    ro.surface_flinger.primary_display_orientation=ORIENTATION_270
```

```
# set primary display orientation to 0  
PRODUCT_PROPERTY_OVERRIDES += \  
    ro.surface_flinger.primary_display_orientation=ORIENTATION_270 \  
    ro.vendor.gsi_gsen_rotation=0
```

Open the **gt9xxnew_ts.idc** file. There is only one touch configuration property in this file. The rotation angle is defined as a **clockwise rotation angle**. For example, to rotate the touch screen **270° clockwise**, add the following configuration:

```
touch.orientation=ORIENTATION_270
```

```
touch.orientation=ORIENTATION_270  
~  
~  
~
```

After the modification is completed, recompile the Android firmware according to **Chapter 7** and flash it to the development board.

8.4 Set the static IP address for the network card in Ubuntu

There are three scripts related to the static IP configuration of the network interface:

1. Set static IP
2. Modify static IP
3. Delete static IP

1、Set static IP

After running the script, enter the **IP address/24**, **subnet mask**, and **DNS** information according to the prompts. The command is as follows:

```
$ eth0_set.sh
```

```
root@HelperA733:~# eth0_set.sh
Enter your static IP (e.g., 192.168.0.110/24): 192.168.1.134/24
Enter your static route (gateway, e.g., 192.168.0.1): 192.168.0.1
Enter DNS server (e.g., 192.168.0.1): 192.168.0.1
Connection 'static-eth0' (5820a12c-5874-4ad5-856a-eccf8b4a10bc) successfully added.
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/12)
root@HelperA733:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.134 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::a6c7:e56f:546e:a114 prefixlen 64 scopeid 0x20<link>
    ether 78:58:c8:99:31:60 txqueuelen 1000 (Ethernet)
    RX packets 96490 bytes 6580320 (6.5 MB)
    RX errors 0 dropped 44 overruns 0 frame 0
    TX packets 807596 bytes 1222472214 (1.2 GB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 217
```

2、Modify static IP

When modifying the static IP, only the IP address needs to be reconfigured. There is no need to configure the subnet mask again. The command is as follows:

```
$ eth0_mod.sh
```

页

```
root@HelperA733:~# eth0_mod.sh
Enter your static ip: 192.168.1.252/24
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/6)
root@HelperA733:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.252 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::e361:49f4:5763:59d7 prefixlen 64 scopeid 0x20<link>
    ether d4:bf:48:ac:9a:57 txqueuelen 1000 (Ethernet)
    RX packets 19442 bytes 1665622 (1.6 MB)
    RX errors 0 dropped 234 overruns 0 frame 0
    TX packets 189 bytes 20406 (20.4 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 7
```

3、Delete static IP

```
$ eth0_del.sh
```

```
root@HelperA733:~# eth0_del.sh
Connection 'static-eth0' (79951ab3-7b63-454b-bf48-2437461fd406) successfully deleted.
root@HelperA733:~# ifconfig
```

8.5 Ubuntu controls Wi-Fi

There is only one script related to WiFi configuration: `/usr/bin/wifi.sh`

1、View available WiFi networks

```
$ nmcli device wifi list
```

2、Configure WiFi

```
$ wifi.sh -f c -n xxxx(wifi 名字) -p xxxxx(wifi 密码)
```

```
root@HelperA733:~# wifi.sh -f c -n baijiekeji -p [REDACTED]
Device 'wlan0' successfully activated with 'b508f158-c245-4393-a0f3-9bcd10f4d71'.
root@HelperA733:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.140 netmask 255.255.254.0 broadcast 192.168.1.255
    inet6 fe80::5b97:eddc:4dc0:5855 prefixlen 64 scopeid 0x20<link>
    ether d4:bf:48:ac:9a:57 txqueuelen 1000 (Ethernet)
    RX packets 20465 bytes 1755880 (1.7 MB)
    RX errors 0 dropped 240 overruns 0 frame 0
    TX packets 213 bytes 22260 (22.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 7

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 84 bytes 6688 (6.6 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 84 bytes 6688 (6.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

p2p0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 18:84:c1:01:bb:f8 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.75 netmask 255.255.254.0 broadcast 192.168.1.255
    inet6 fe80::b0b8:e401:70b8:be7 prefixlen 64 scopeid 0x20<link>
    ether 18:84:c1:01:bb:f9 txqueuelen 1000 (Ethernet)
    RX packets 492 bytes 75630 (75.6 KB)
    RX errors 0 dropped 4 overruns 0 frame 0
    TX packets 30 bytes 3080 (3.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

After the WiFi configuration is completed, the device will automatically connect to the configured WiFi network after every boot.

3、Modify WiFi configuration

If the password of the currently connected WiFi network is changed, the corresponding WiFi password on the development board must also be updated accordingly. Run the following command:

```
$ wifi.sh -f m -n xxxx(wifi 名字) -p xxxx(新密码)
```

```
root@HelperA733:~# wifi.sh -f m -n baijiekeji -p bjkj0818
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/16)
```

4、Delete WiFi configuration

```
$ wifi.sh -f d -n baijiekeji
```

```
root@HelperA733:~# wifi.sh -f d -n baijiekeji
Connection 'baijiekeji' (b508f158-c245-4393-a0f3-9bcd10f4d71) successfully deleted.
root@HelperA733:~#
```

5、Set static IP

```
$ wifi.sh -f s -i 192.168.1.253 -n xxxx(wifi 名字) -p xxxx(wifi 密码) -g
192.168.1.1
```

```
root@HelperA733:~# wifi.sh -f s -i 192.168.1.253 -n baijiekeji -p [REDACTED] -g 192.168.1.1
Connection 'baijiekeji' (7de53b40-915a-4d8b-affe-581772dc6102) successfully deleted.
Device 'wlan0' successfully activated with '4e545168-b504-4e25-ad4c-7d39cfff9b1b'.
Connection 'baijiekeji' successfully deactivated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/19)
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/20)
root@HelperA733:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.140 netmask 255.255.254.0 broadcast 192.168.1.255
    inet6 fe80::5b97:eddc:4dc0:5855 prefixlen 64 scopeid 0x20<link>
    ether d4:bf:48:ac:9a:57 txqueuelen 1000 (Ethernet)
    RX packets 29850 bytes 2532832 (2.5 MB)
    RX errors 0 dropped 387 overruns 0 frame 0
    TX packets 263 bytes 25760 (25.7 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 7

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 98 bytes 9172 (9.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 98 bytes 9172 (9.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

p2p0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 18:84:c1:01:bb:f8 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.253 netmask 255.255.254.0 broadcast 192.168.1.255
    inet6 fe80::79ee:e893:9b13:4782 prefixlen 64 scopeid 0x20<link>
    ether 18:84:c1:01:bb:f9 txqueuelen 1000 (Ethernet)
    RX packets 6657 bytes 984310 (984.3 KB)
    RX errors 0 dropped 118 overruns 0 frame 0
    TX packets 205 bytes 18679 (18.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

8.6 Ubuntu root file system backup

When modifications have been made to the Ubuntu system on the development board and need to be preserved, the entire root filesystem can be backed up.

During firmware compilation, replace the original **rootfs** with the backed-up filesystem. The newly generated firmware will then include all previous modifications, eliminating the need to repeat the configuration process.

This provides an efficient way to migrate customized features and system configurations.

1、File Packaging

Run the **szbaijie_rootfs_backup** command. The **rootfs.tar.gz** file will be generated in the root directory. The details are as follows:

```
$ szbaijie_rootfs_backup /  
$ ls -l /rootfs.tar.gz  
-rw-r--r-- 1 root root 428518374 Sep 22 09:32 /rootfs.tar.gz
```

Note: If important files are stored in other mounted directories, they will not be included during the packaging process. For example, if **/dev/sda5** is mounted to the **/data** directory, the files under the **/data** directory will not be included in the generated root filesystem package.

2、Build

After packaging is completed, use a USB drive or the **scp** command to copy the **rootfs.tar.gz** file to the following directory in the Linux source code: **longan/test/dragonboard/baijie_system** (It is recommended to rename the file after copying.) Then return to the **longan** directory and run the **./build.sh config** command again.

In the configuration menu:

- Select **no_extra** for **baijie_extra**.
- Select the newly added root filesystem package for **baijie_rootfs**.

```
$ cd ~/workspace/a733/longan  
$ ./build.sh config
```

```
All available baijie_extra:  
0. extra-24.04-core  
1. extra-a733-mate  
2. extra-gt5.15.14  
3. no extra  
Choice [extra-24.04-core]: 3  
All available baijie_rootfs:  
0. rootfs_new.tar.gz  
1. rootfs_ubuntu22.04_core.tar.gz  
2. rootfs_ubuntu22.04_mate.tar.gz  
3. ubuntu24.04_core.tar.gz  
Choice [rootfs_new.tar.gz]: 0  
rootfs decompression done!
```

At this point, you can follow the **User Development Manual** to compile and generate the new firmware. The command is as follows:

```
$ ./build.sh && ./build.sh pack
```

Flash the newly compiled firmware image to the development board.

8.7 Modify the boot logo of Ubuntu

When the boot logo needs to be changed in the Ubuntu system, use the following method to update it. The new logo will take effect after rebooting the system.

Requirements:

- The **bootlogo.bmp** image must be a **32-bit BMP image**.
- The image resolution must be smaller than the display resolution.

Usage:

```
szbaijie_logo_update_64 ./bootlogo.jpg 1024 600 0
```

Parameter description:

- The first parameter specifies the image path.
- The second parameter specifies the screen width.
- The third parameter specifies the screen height.
- The fourth parameter specifies the image rotation angle (**counterclockwise rotation**).

```
root@HelperA733:~# szbaijie_logo_update_64 ./bootlogo.jpg 1024 600 0
*****
Company       : BaiJie Technology
Website      : www.szbaijie.cn
Product       : bootlogo update
Update time   : 2025.11.10
Version       : v1.3
*****

The current resolution is 1920 x 1080
bootlogo blk name: /dev/sda1
Use roatation, rotate angle is 0.000000
Conversion successful: /run/bootlogo.bmp
mount: /mnt: can't read superblock on /dev/sda1.
       dmesg(1) may have more information after failed mount system call.
/dev/loop0
Change bootlogo success!
```

After the command is executed successfully, the result is shown as above. Reboot the system, and the updated **boot logo** will be displayed.

8.8 Modify screen rotation in Ubuntu

8.8.1 Qt system modification

1、 To change the **display orientation**, set the following environment variable and restart the Qt application. The value **90** indicates the rotation angle. Modify it according to your requirements.

```
$ export QT_QPA_EGLFS_ROTATION=90
```

2、 To change the **touch orientation**, run the following script. The value **90** indicates the rotation angle. Modify it according to your requirements.

```
$ /usr/bin/touch_rotate.sh 90
```

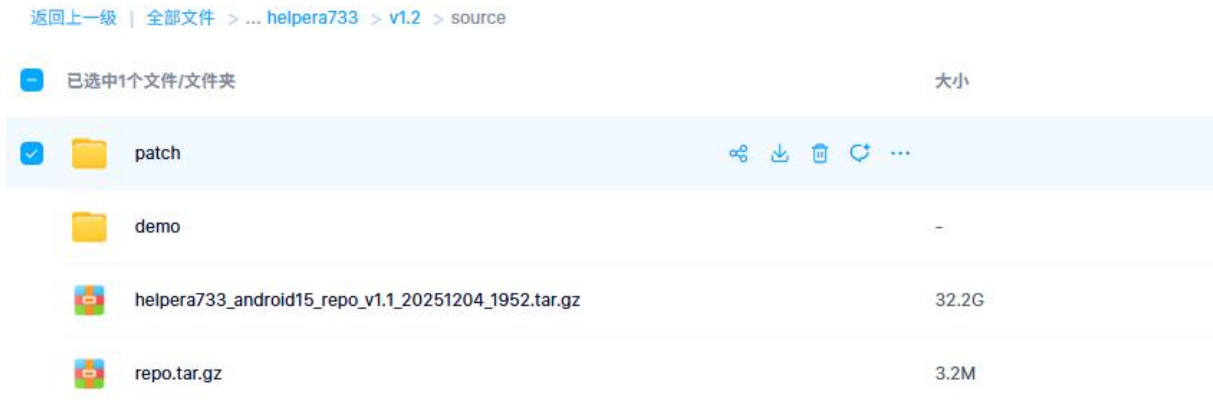
8.8.2 Xfce System Modification

Execute the following script command. The parameter after the command indicates the clockwise rotation angle.

```
$ /usr/bin/rotate-screen.sh 0  
$ /usr/bin/rotate-screen.sh 90  
$ /usr/bin/rotate-screen.sh 180  
$ /usr/bin/rotate-screen.sh 270
```

8.9 Patch the defect

The released system version may require users to apply additional patches in some cases. The patch files are located in the following directory: `source/patch`



For example, if we need to apply the **example.patch** patch, the procedure is generally as follows:

```
diff --git a/drivers/vin/modules/sensor/ov8858_r2a_4lane.c b/drivers/vin/modules/sensor/ov8858_r2a_4lane.c
index 9350b346..1915e6f1 100644
--- a/drivers/vin/modules/sensor/ov8858_r2a_4lane.c
+++ b/drivers/vin/modules/sensor/ov8858_r2a_4lane.c
@@ -45,7 +45,7 @@ int ov8858_sensor_vts;
 * The ov8858 sits on i2c with ID 0x6c
 */
#define I2C_ADDR 0x6c
-#define SENSOR_NAME "ov8858_mipi"
+#define SENSOR_NAME "ov8858_r2a"

struct cfg_array { /* coming later */
    struct regval_list *regs;
diff --git a/drivers/vin/vin-cci/sunxi_cci.c b/drivers/vin/vin-cci/sunxi_cci.c
index 24cebf78..756bbaf6 100644
--- a/drivers/vin/vin-cci/sunxi_cci.c
+++ b/drivers/vin/vin-cci/sunxi_cci.c
@@ -198,6 +198,7 @@ sunxi_i2c_xfer(struct i2c_adapter *adap, struct i2c_msg *msgs, int num)
    kfree(buf);
    return i;
}
+ return 0;
}

static unsigned int sunxi_i2c_functionality(struct i2c_adapter *adap)
~
~
```

Before applying a patch, navigate to the corresponding directory. In the example above, the patch was generated from the longan/bsp directory.

The patch modifies the following files:

`drivers/vin/modules/sensor/ov8858_r2a_4lane.c`

`drivers/vin/vin-cci/sunxi_cci.c`

Therefore, you only need to change to the longan/bsp directory before applying the patch.

Note: Applying the patch from any other directory will result in an error because the file paths in the patch cannot be resolved correctly.

The command is shown below:

```
$ cd ~/workspace/a733/longan/bsp
$ patch -p1 < example.patch
```

8.10 Kernel configuration

The required kernel configuration for the A733 platform has already been completed. The following example is provided only to help users become familiar with the kernel configuration process. Before configuring the kernel, load the default kernel configuration file provided by HelperBoard (refer to Sections 5.1 and 7.1).

Default kernel configuration for Linux:

longan/device/config/chips/a733/configs/default/linux-6.6/dragonboard_defconfig

Default kernel configuration for Android:

longan/device/config/chips/a733/configs/default/linux-6.6/android15_arm64_defconfig

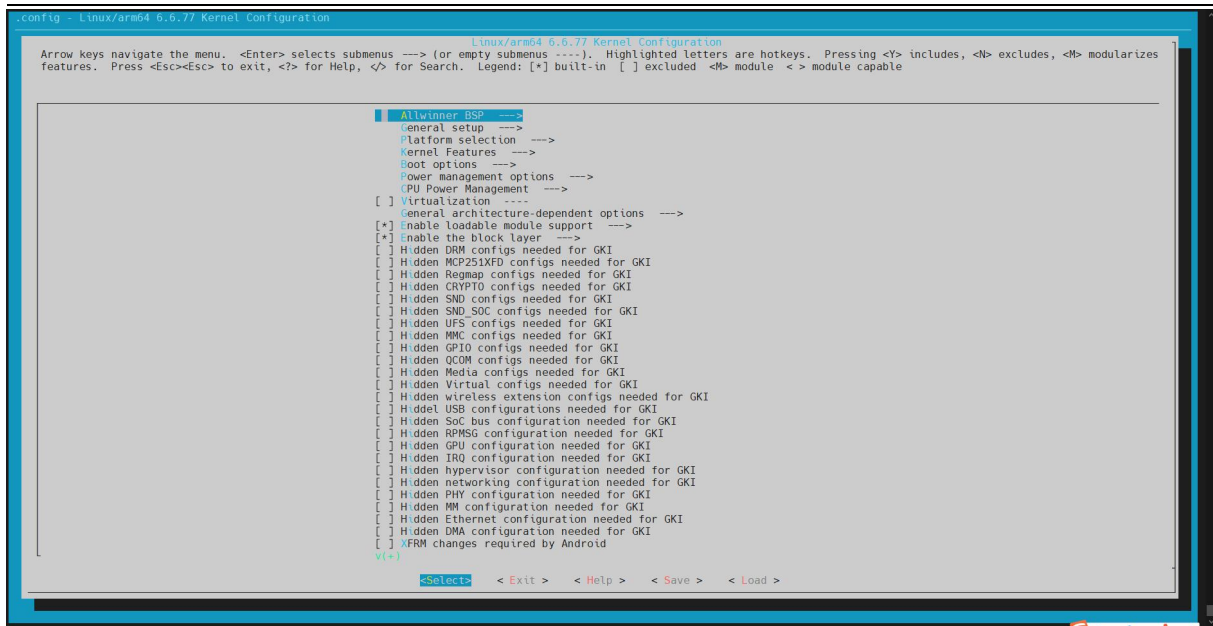
Make any required modifications based on the default configuration. The updated configuration will be saved to the following file in the kernel build directory:

longan/out/kernel/build/.config

The kernel configuration commands are shown below.

```
$ cd ~/workspace/a733/longan/
$ ./build.sh menuconfig
```

Note: As shown in the first few lines of text in the image above, “Y” indicates selection, ‘N’ indicates deselection, and “M” indicates compilation as a module, among other information.



9 Regarding customized services

Our company offers customization services for projects based on the Allwinner A733 platform. If you are interested, please contact Mr. Chen at [13510178868](tel:13510178868).