Proculus Technologie Limited

Android LCMs Development Guide

V5.3

Terms and Conditions

Accessing Specifications and Development Guide

Before purchasing or using an Android display module, you can first understand the characteristics and interface of the Android display module. Visit the technical documents section (https://www.proculustech.com/technical-documents/) to access the Android product datasheet and Android display development guide for the specific model by clicking on the corresponding model.

1. Android Product Datasheet

The document introduces the performance and functional features of the Android module, and provides explanations on after-sales and technical support.

2. Android Display Development Guide

The document provides a detailed guide on setting up the development environment, along with examples of serial port development and buzzer usage, to help you quickly get started with the Android module. It also includes instructions for flashing the Android module system and customizing the system.

Note:

For obtaining the image files, please contact your sales representative. You can download the image tools and flashing tools from proculustech-download (https://www.proculustech.com/download/). Please refer to the table in the development guide for the corresponding tools. If you encounter any unresolved issues while using the Android display module, please refer to Chapter 4 of the Android Display Development Guide for important notes. If you are unable to resolve the issue, please seek technical support from your designated sales representative.

Retrieve the accompanying software and source code

To help you quickly get started with the development of the Android display module software, we have prepared multiple demo programs for you. You can download the examples from proculustech-download (https://www.proculustech.com/download/).

1. **Serial Port Example Program Reference**

   The serial port example program helps you quickly get started with serial port development by providing code reference examples for using serial ports on Android.

2. **Hidden Navigation Bar Example**

   Using system-level interfaces to hide the navigation bar and display the app in fullscreen.

3. **Buzzer Example**

Calling the buzzer, providing code reference examples for using and disabling the buzzer.

How we can help

After you purchase the Android screen module, we will provide you with limited assistance on the software/hardware aspects of the Android screen module.

- **Hardware**

We provide support for customers in answering hardware-related questions regarding their Android products.

We can also assist in providing answers to inquiries about the interface of the Android screen module.

- **Software**

We offer assistance and provide answers to questions regarding software development environments for Android screen modules.

We can also help with inquiries and provide documentation support for the Android screen module's system APIs and software development.

⚠️ Note:

After purchasing an Android screen module, you may need to run your project on the module. The Android system supports executable files with the .apk extension, and only projects with the .apk extension can run on the Android screen module. The recommended development tools for this purpose are Android Studio or Eclipse, with Android Studio being the mainstream integrated development environment (IDE) for Android.

📌 Hint:

The programming languages supported for developing Android applications include Java, Kotlin, Flutter, etc. You can develop an Android application project on your own if you have knowledge of any of these programming languages. If you are unable to develop an Android application on your own, we can offer Android application development services, and the pricing would be discussed with our sales team. You can also seek out a reliable and stable Android application development team, and we will provide limited technical documentation support.

Frequently Asked Questions (FAQs)

1. **How can I choose an Android screen module that suits my needs?**

   You can choose an Android screen module through the product selection interface on our official website or communicate with our customer service team via our official Taobao store to select the appropriate product.

2. **Are your Android screen modules stable?**

   Our Android screen modules are designed for industry use and can operate continuously for 24 hours without crashing or power interruption. The software backend supports stable operation for long periods using 4G or Wi-Fi connectivity.

3. **What tests have you performed on your Android screen module?**

   We have conducted power-off tests, temperature tests, power loss tests, interface pressure tests, high and low temperature tests, and aging tests before leaving the factory to ensure product stability.

4. **What versions can the Android screen module be equipped with?**

   In addition to the standard version, the Android screen module can be equipped with the following options: WIFI module (without Bluetooth), WIFI module (with Bluetooth), 4G module (without GPS), and 4G module (with GPS).

5. **Will continuous power-off of the Android screen module cause any problems?**

   We have tested the product for over 100,000 power-off cycles, and the product functions perfectly without any issues.

6. **Can the Android screen module be waterproof?**

   For screen waterproof requirements: Infrared touch screens are waterproof, and you can choose an infrared touch screen. We also offer customized capacitive touch waterproof screens (by default, capacitive touch screens are not waterproof). For machine waterproof requirements: Customers need to create their own waterproof structure. Our K series and R series casings can only achieve partial waterproofing, and the interfaces are not treated for waterproofing.

7. **Can I have neutral packaging?**

   Yes, we can provide neutral packaging. There will be no logo on the boot interface or on the machine itself. Only the packaging box, repair card, and certificate will have logos. If you require neutral packaging, please inform our sales personnel, and we can remove the logo before shipping.

8. **Can the Android screen module be powered by a battery?**

   Our devices primarily operate on a 12V 2A DC power supply. You can use any power

source that can provide this voltage. However, we do not have a recommended portable battery, which would need to be developed or purchased at your discretion.

9. **If I purchase your product, will you provide technical support?**

   What is covered in the support? From the date of purchase, for a period of 12 months, our frontline engineers will provide timely support during working days and hours.

   Support scope:

   - Support for running Android system and related interface testing programs.
   - Support for common configurations of Android system.
   - Support for hardware aspects of customer's Android products.

10. **Can you provide warranty for the products I purchase?**

    We offer a 7-day no-reason return service, a one-year warranty, and lifetime paid maintenance services. CPUs and accessories are not covered by the warranty. Damages caused by incorrect use or force majeure are not covered by the warranty.

11. **Do you have technical support documents?**

    You can find the documents you need in the "Technical Support - Program Download" section on our official website.

12. **What is flashing image, and why do we need to flash system image?**

    Flashing image refers to installing the supported system on the module, such as Android or Ubuntu. If you have other specific settings, system function updates, or adding new external modules, you need to flash the Android system again.

13. **How do I flash the system?**

    Can I flash the system image on a MAC computer? Please refer to section 3.6 "Firmware Flashing Program" in the user manual for the Android screen, which provides detailed instructions for flashing the system. Before flashing the image, you need to prepare the corresponding Android system image for the Android module. Currently, system image flashing is not supported on MAC computers. Please use a Windows system for flashing.

14. **Now that I know how to flash, where can I download your system images?**

    We do not currently provide download links for the system image. You can contact your sales representative to obtain the latest system image.

15. **What modifications can be made to your image system?**

    The image system supports replacing the boot logo, changing the boot animation, modifying the default wallpaper, and adding Android application APKs to the image. For detailed operation steps, please refer to section 3.5 "Boot Configuration" in the Android screen user manual.

16. **I need to develop an Android project on the Android module. Can you help? What do I need to provide?**

    Yes, we support software development and customization. You need to provide software development requirements and existing information. We need to evaluate the project timeline and quotation, and once the evaluation is completed, we will coordinate with you.

17. **I need to develop an Android project on the Android module, and I have found another team. What do I need to provide them?**

    You need to provide the Android product datasheet and the Android screen user manual. The product datasheet provides detailed information about the interfaces and usage specifications of the Android module. If the above materials cannot solve the problem, you can also communicate with our technical team.

Version History

Document Version	Last Update	Description
V5.3	20230919	Optimization Content

Table of Contents

Chapter 1: Hardware and Interface Introduction

1.1 Product Interfaces

The Android module is a complete LCD display module solution based on Rockchip CPU core board and equipped with different baseboard interfaces. With support from various specifications of CPU chips, the module can be connected to the host computer through interfaces such as serial port, WiFi, and USB. Among them, serial port I/O is the main communication method for the product.

Take a 7-inch Android module with RK3288 as an example. The Android module supports 3 channels of RS232 or TTL level serial ports and 1 channel of RS485/RS232/TTL level serial port. Among them, ttyCOM0, ttyS1, and ttyS3 are ordinary serial ports that can be used for serial communication, with a maximum baud rate support of 115200. ttyS4 is the default RS232 but can be modified to RS485.

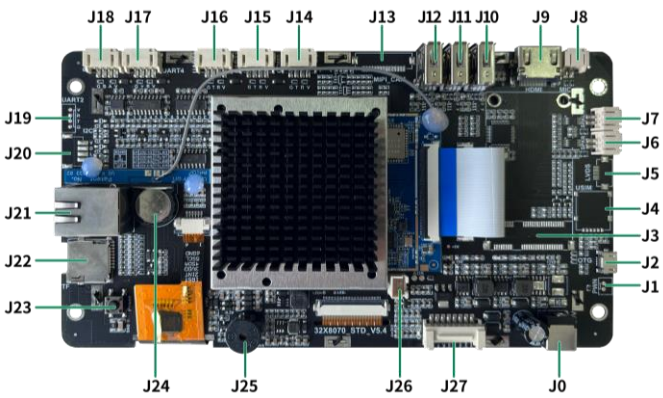
Hint：

In products with other specifications, the ttyCOM0 port can also support USB to serial conversion for expansion.

- USB interface

Taking a 7-inch product with RK3288 as an example, the product is equipped with three USB HOST ports and one USB OTG port, which supports peripheral devices such as USB mice, USB flash drives, and cameras. The USB OTG is used by Android developers for Android application development.

The product interfaces of the RK3288 Android display are shown as depicted in Figure 1-1-1：



1-1-1 7-inch RK3288 standard interface description

Num.	Interface Name	Description
J0	Power	12V/2A power supply (maximum voltage ranges from 6V to 16V DC)
J1	Wake-up	Control screen system on/off
J2	USB_Micro	OTG /App debugging/ Firmware upgrade interface
J3	MINI PCIE 4G	4G LTE Module/GPS (Optional)
J4	SIM card	Nano-SIM supported (Optional)
J5	LVDS interface	Reserved
J6	SPK_R	Right channel audio output interface
J7	SPK_L	Left channel audio output interface
J8	MIC	Audio input interface
J9	HDMI	HDMI output interface
J10	USB_HOST3	Support USB Peripherals
J11	USB_HOST2	Support USB Peripherals
J12	USB_HOST1	Support USB Peripherals/Support dual USB parallel use
J13	MIPI camera interface	Reserved
J14	COM0	Device name: COM0. Pin definition: 5V, RXD, TXD, GND
J15	UART1	Device name: ttyS1. Pin definition: 5V, RXD, TXD, GND
J16	UART3	Device name: ttyS3. Pin definition: 5V, RXD, TXD, GND
J17	UART4	Device name: ttyS4. Pin definition: 5V, RXD, TXD, GND/Same as J18
J18	RS485	Device name: ttyS4. Pin definition: 5V, RXD, TXD, GND/Same as J17
J19	Debug interface	not open temporarily
J20	IIC communication	Pin definition: 5V, SDA, SCL, GND (Reserved)
J21	RJ45 interface	Support 10M/100M/1000M network
J22	TF card	Can do memory expansion

Num.	Interface Name	Description
J23	Wake-up	Control screen system on/off
J24	RTC	Supply system RTC
J25	Buzzer	Buzzer
J26	Radiator interface	Reserved
J27	User interface	Pin Definition: 12V, 12V, NC, TXD, RXD, RXD, GND, GND

1.2 Hardware modification of RS232/RS485/TTL

Android module, the serial port is set to RS232 mode by default. If you need to use RS485 or TTL functionality, please refer to the following method for modification, taking RK3288 Android display as an example.

1. Check if the Android module supports RS485/TTL.

Check the serial port interface parameters in the product specification document of the Android display. If it supports RS485/TTL, it means that the product can be modified for the serial port mode. As shown in figure 1-2-1 below.

Interface					
Item	**Condition**	**Min.**	**Typ.**	**Max.**	**Unit**
Baud Rate	Standard	1200	9600	115200	bps
	User Defin	1200	—	115200	bps
Serial Mode	Serial Port*4 (3*RS232/TTL, 1*RS232/TTL/RS485).				
User Interface	Standard serial communication protocol. 8Pin_2.0mm socket.				
USB	USB DEBUG*1. USB HOST*3.				
Ethernet	Support 10m/100m/1000m Ethernet.				
Wi-Fi/Bluetooth	Support 802.11b/g/n Wi-Fi wireless network. Bluetooth is Optional.				

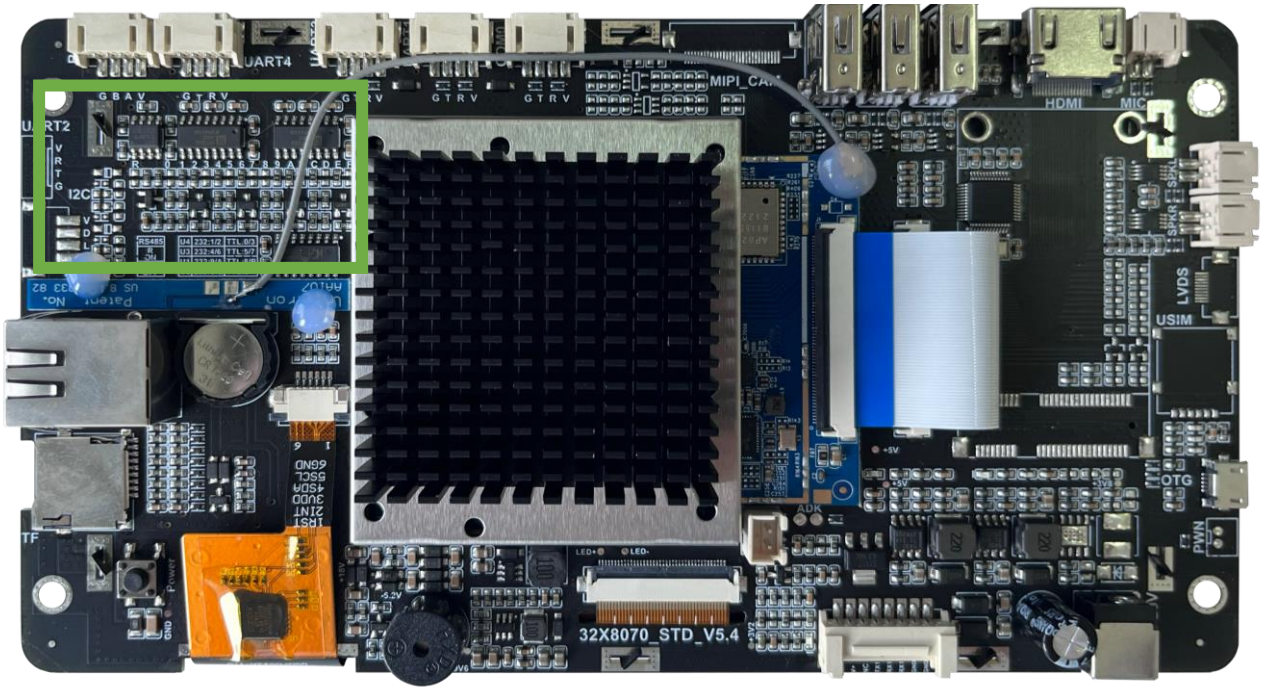
1-2-1 Serial port mode

Hint:

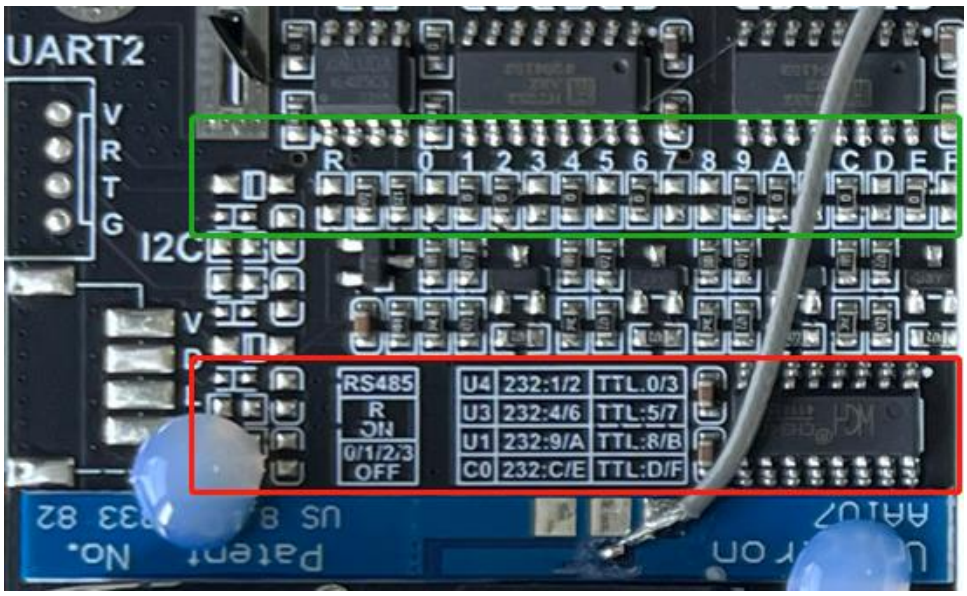
You can visit the official website of ProculusTech (https://www.proculustech.com/) to download the product specification document for the Android display. If you are unable to resolve the issue you are facing, please seek technical support from the salesperson you are in contact with.

2. Check the module silk screen and modify the serial port mode

Check the silk screen on the upper left corner of the back of the Android module, as shown in figures 1-2-2 and 1-2-3 below.



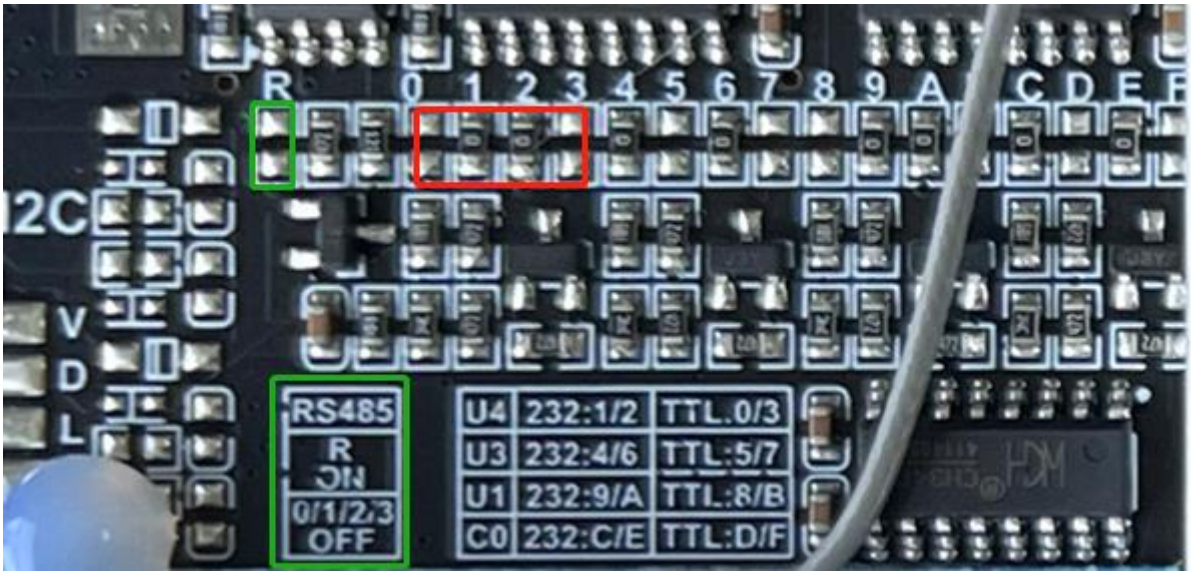
1-2-2 Overall diagram



1-2-3 Modify the area/region

The green and red boxes are labeled in the above figure 1-2-3. The green box indicates the modified area, while the red box indicates the reference area.

Taking changing RS485 as an example, please refer to the following figure 1-2-4.



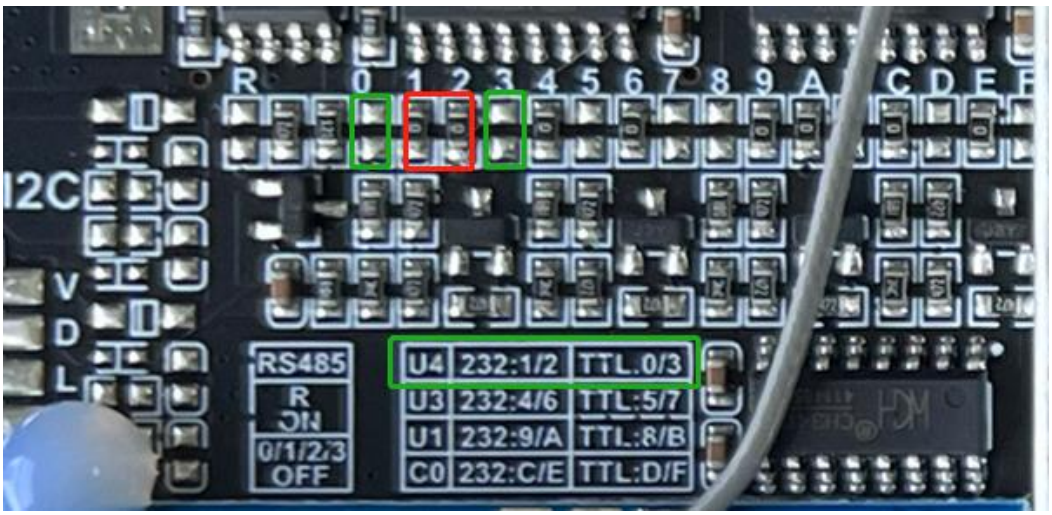
1-2-4 Modify to use RS485.

The above figure 1-2-4 indicates the green and red boxes. The green box represents the modification area, where a 0Ω resistor is added. The red box represents the removal area, where four resistors are removed. After the modification is complete, the RS485 interface can communicate properly.

Note：

If you need to use other serial ports, you only need to add a 0Ω resistor in the R area. There is no need to make any changes in the 0/1/2/3 areas. If you are unable to resolve the issue you are facing, please seek technical support from the salesperson you are in contact with. We will assign a hardware engineer to assist you in resolving the problem.

Taking changing UART4 to TTL as an example, please refer to the following figure 1-2-5.



1-2-5 Modify to use UART4 as TTL

The above figure 1-2-5 indicates the green and red boxes. The green box represents the modification area, where a 0Ω resistor is added. The red box represents the removal area, where two resistors are removed. After the modification is complete, the UART4 interface can be used for TTL communication.

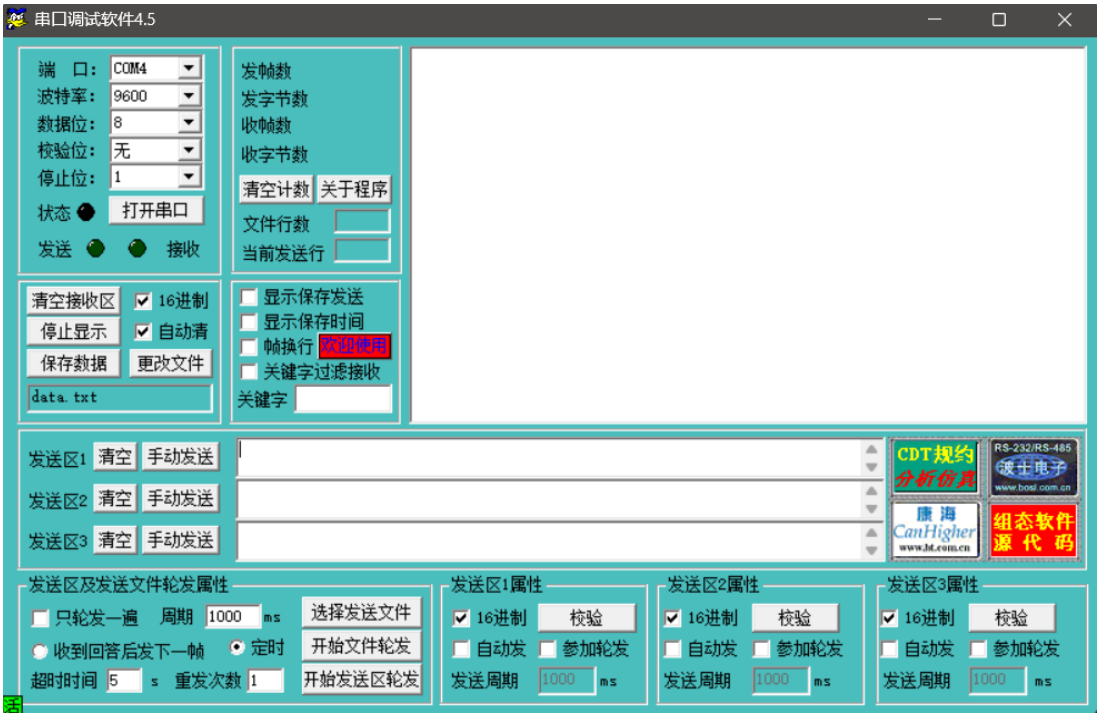
Note:

If you are unable to resolve the issue you are facing, please reach out to the salesperson you are in contact with for technical support. We will allocate a hardware engineer to assist you in resolving the problem.

3. Check the serial communication functionality

Install ComAssistant serial debugging software on the Android module, as shown in the following figure 1-2-6. Open any serial debugging software on the computer, as shown in the following figure 1-2-7.

1-2-6 ComAssistant



1-2-7 Windowsserial debugging software

Set the baud rate, parity bit, and stop bit to be consistent, then open the serial port for sending and receiving data. Testing the stability of the connection is recommended by running it for a period of time.

Note:

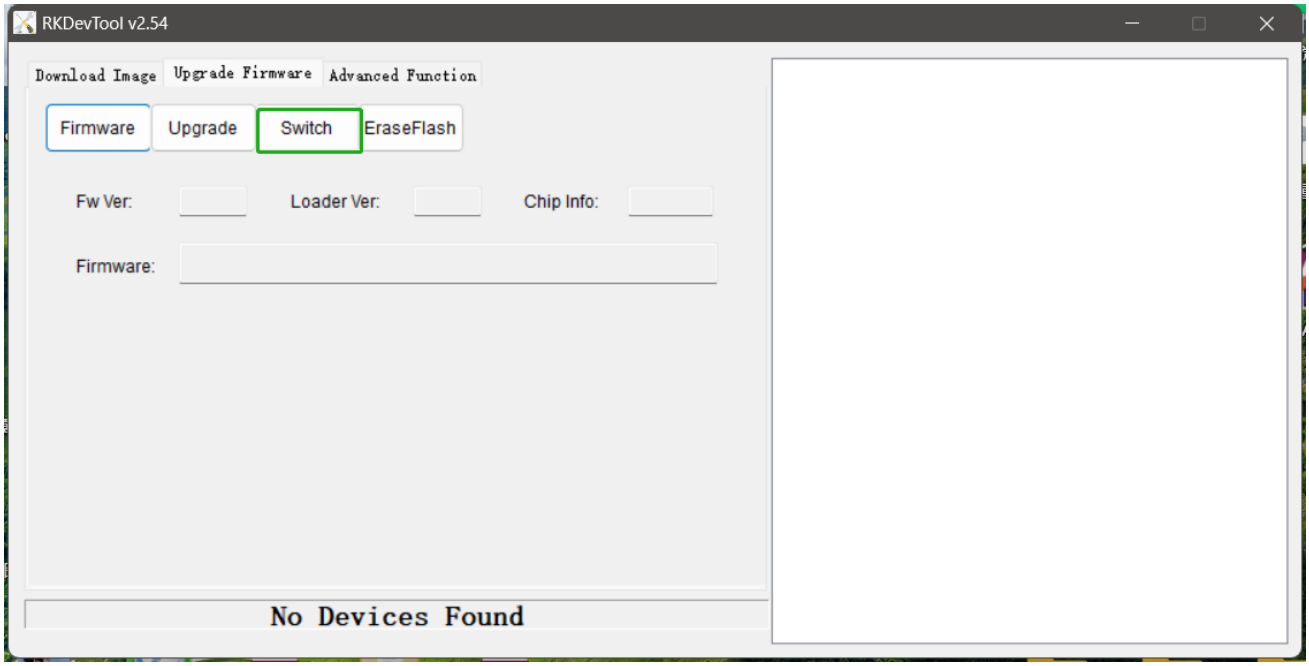
Download link for ComAssistant serial debugging software: Click here to download. Currently, For Windows systems, the option exists to procure SuperCom or similar software for serial port debugging. If you are unable to resolve the issue you are facing, please seek technical support from the salesperson you are in contact with. We will allocate a hardware engineer to assist you in resolving the problem.

1.3 Enter the system burning mode

Android modules will be pre-installed with the corresponding version of Android system before delivery. If you have other special Settings, such as modifying startup animation, modifying startup logo, and customizing system functions, you need to burn the system, please enter the system burning mode by following the methods below to demonstrate on the RK3288 Android screen.

Initiate system firmware programming mode directly via the tool

Install the Rockchip driver and enable the USB debugging functionality of the module. Power on the Android module and connect it to your computer. Once the Android Tool utility displays a notification at the bottom indicating the discovery of an ADB device, proceed by clicking the 'Switch' button. The Android module shall seamlessly transition into the system's programming mode, eliminating the need for ADK short-circuiting.



Access the system firmware programming mode through ADB

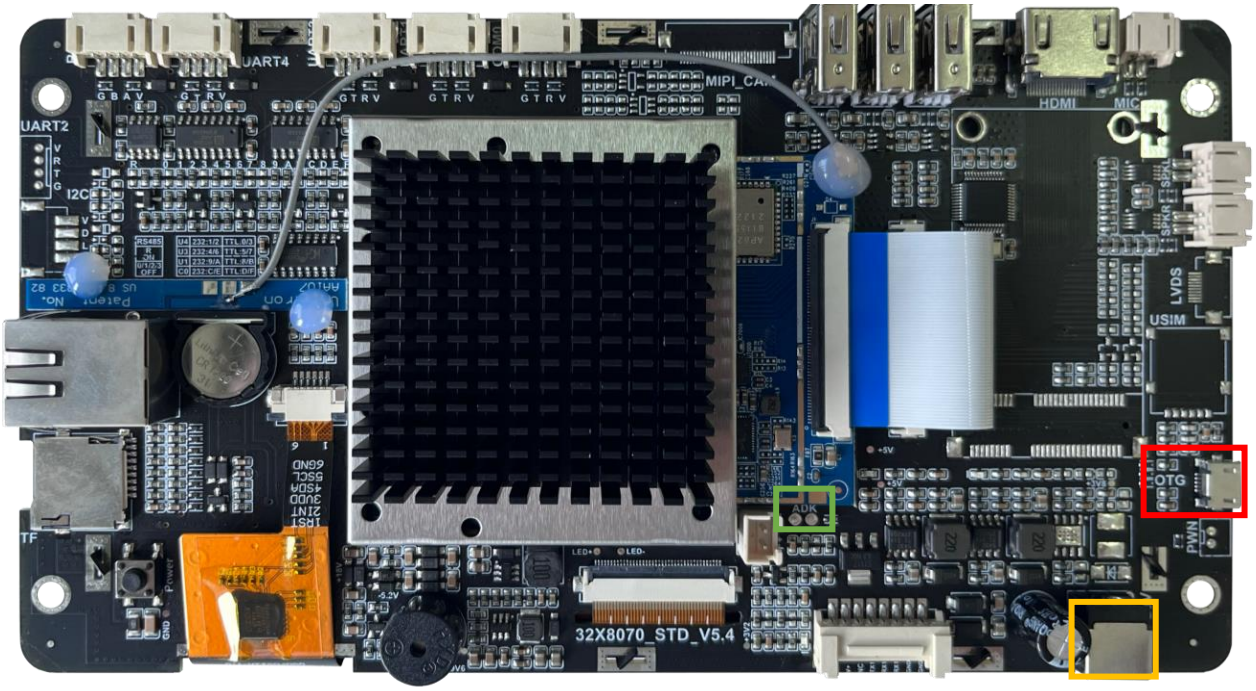
In the state of powering up the module, access the system and execute ADB commands to transition the module into firmware programming mode.

```
adb reboot loader
```

Note:

If neither of the aforementioned approaches facilitates entry into the programming mode, one may resort to the method of short-circuiting hardware contact points below to access the system's programming mode.

Rockchip Driver Download Link: https://www.proculustech.com/uploads/file/androidtool-v2.35-rk31x8-4.4-5.1.zip



1-3-1 ADK

The green box in figures 1-3-2 marks the short-circuit point (ADK) for entering system programming mode, and includes two contact points.

Please strictly follow the steps below in the correct order

1. Connect the USB-MicroUSB cable to the OTG port inside the red box and connect it to the computer.
2. Use tweezers to short-circuit the ADK. Insert the power cable into the power port inside the yellow box, power it on, and do not release the short-circuit of the ADK at this time.
3. When the Rockchip Android Tool prompts that a LOADER device is detected, disconnect the short-circuit with tweezers. The system enters the burning mode successfully.

Warning:

The short-circuit point of ADK for some older products is located above the product's core board without any markings. Do not randomly short-circuit unmarked contact points. If you need to modify the hardware, please perform the modification and short-circuit under the guidance of a hardware engineer. For more details, please contact your salesperson for the corresponding technical support.

Note:

The USB-MicroUSB cable must be a data transmission capable cable, typically with 4 cores. Please

note that a 2-core cable cannot transmit data and cannot communicate with modules or computers. If the module is connected to the computer but does not respond, please try replacing the data cable.

If you have confirmed that the data cable can communicate, please install the Rockchip Android Driver and then restart the computer.

Rockchip Driver Download Link: https://www.proculustech.com/uploads/file/androidtool-v2.35-rk31x8-4.4-5.1.zip

Hint：

For detailed hardware specifications, please refer to the Android screen product specification sheet. The specification sheet can be downloaded from the official website of ProculusTech (). If you are unable to resolve the issue you are experiencing, please seek technical support from the salesperson you are in contact with.

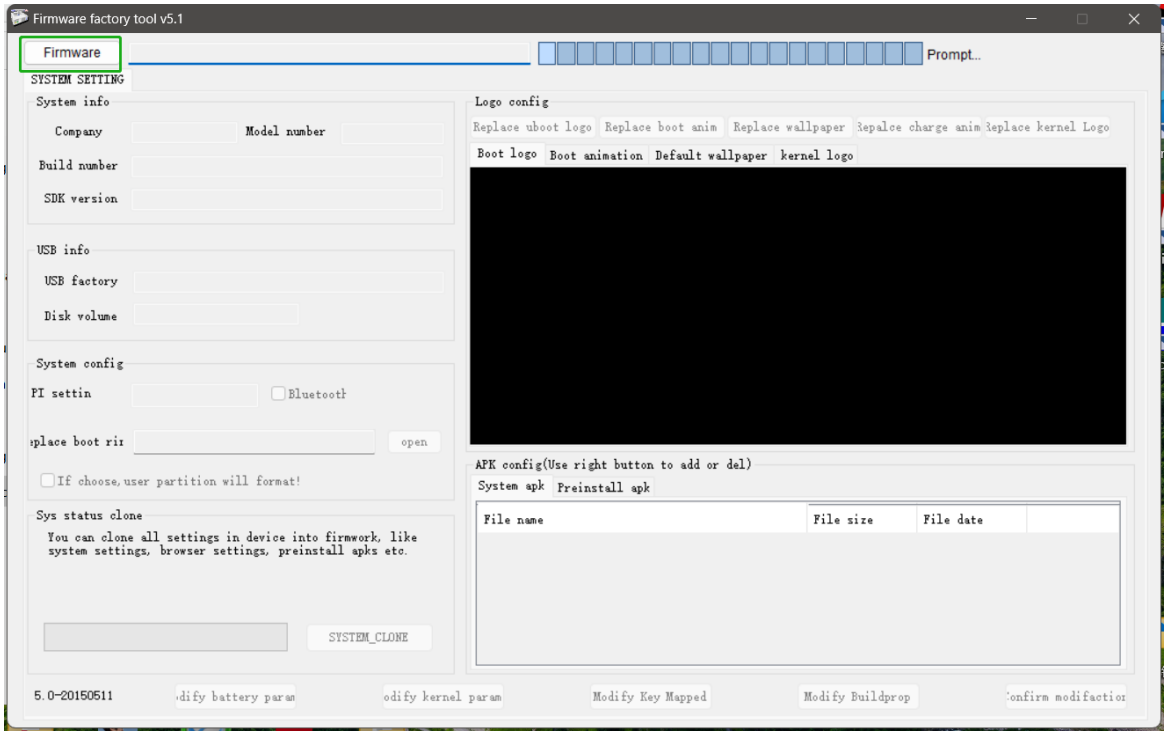
Chapter 2: System Customization and Flashing

2.1 System Customization

The Android open-source system loaded in the Android module has open-root access, allowing customization of settings such as boot animation, default wallpaper, time zone, and screen orientation. These settings include but are not limited to boot images or animations, wallpapers, and startup items. System customization will enhance the aesthetics of your project.

If you need to configure the system's boot settings, please select the firmware factory tool corresponding to the version of the development board on the official website. Please refer to the table below for the corresponding firmware factory tools.

Firmware Factory Tools	Applicable Products
FWFactoryTools_v5.1_RK3128_5.1	Customization of Android 5.1 system for RK3128 development board.
FWFactoryTools_v5.3_RK3288_5.1	Customization of Android 5.1 system for RK3288 development board.
FWFactoryTools_v5.51_RK3128_RK3288_7.1	Customization of Android 7.1 system for RK3128 and RK3288 development boards.

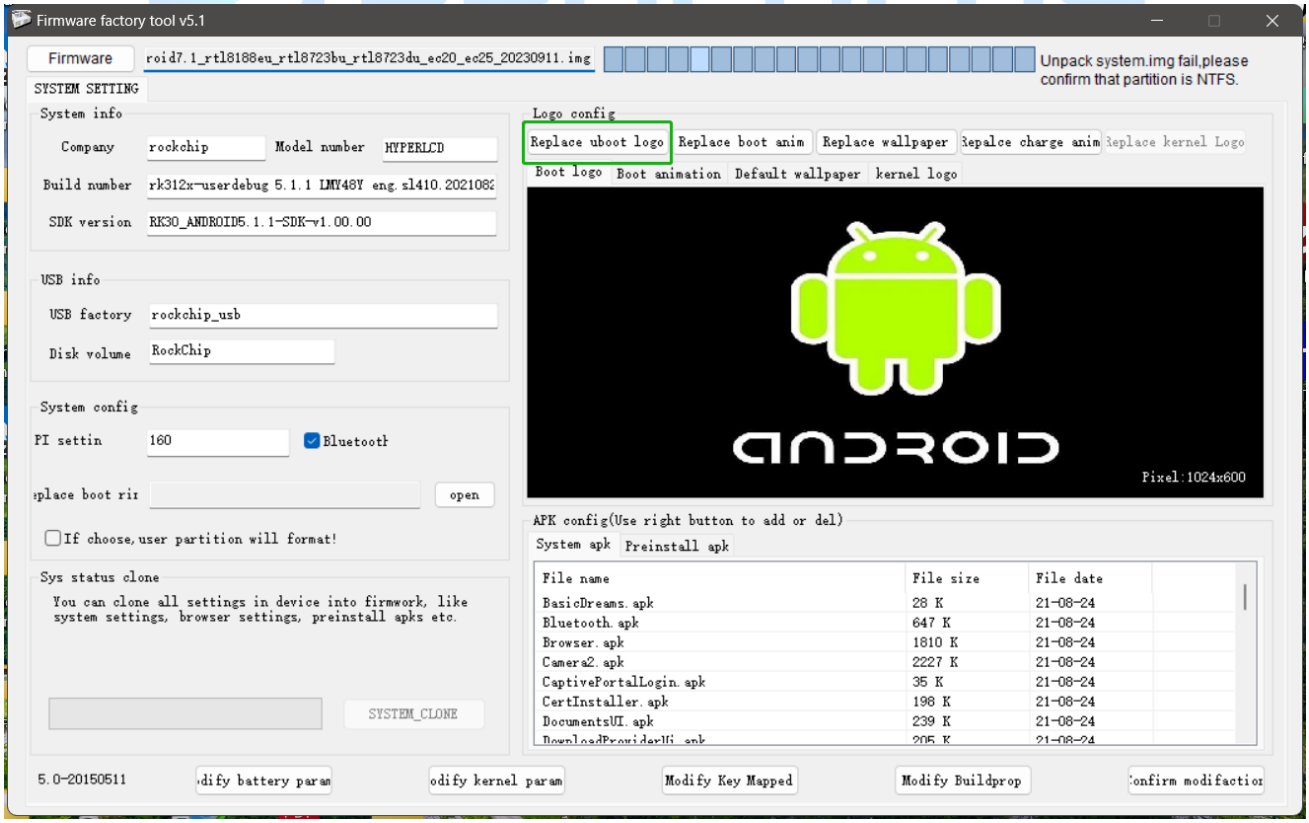


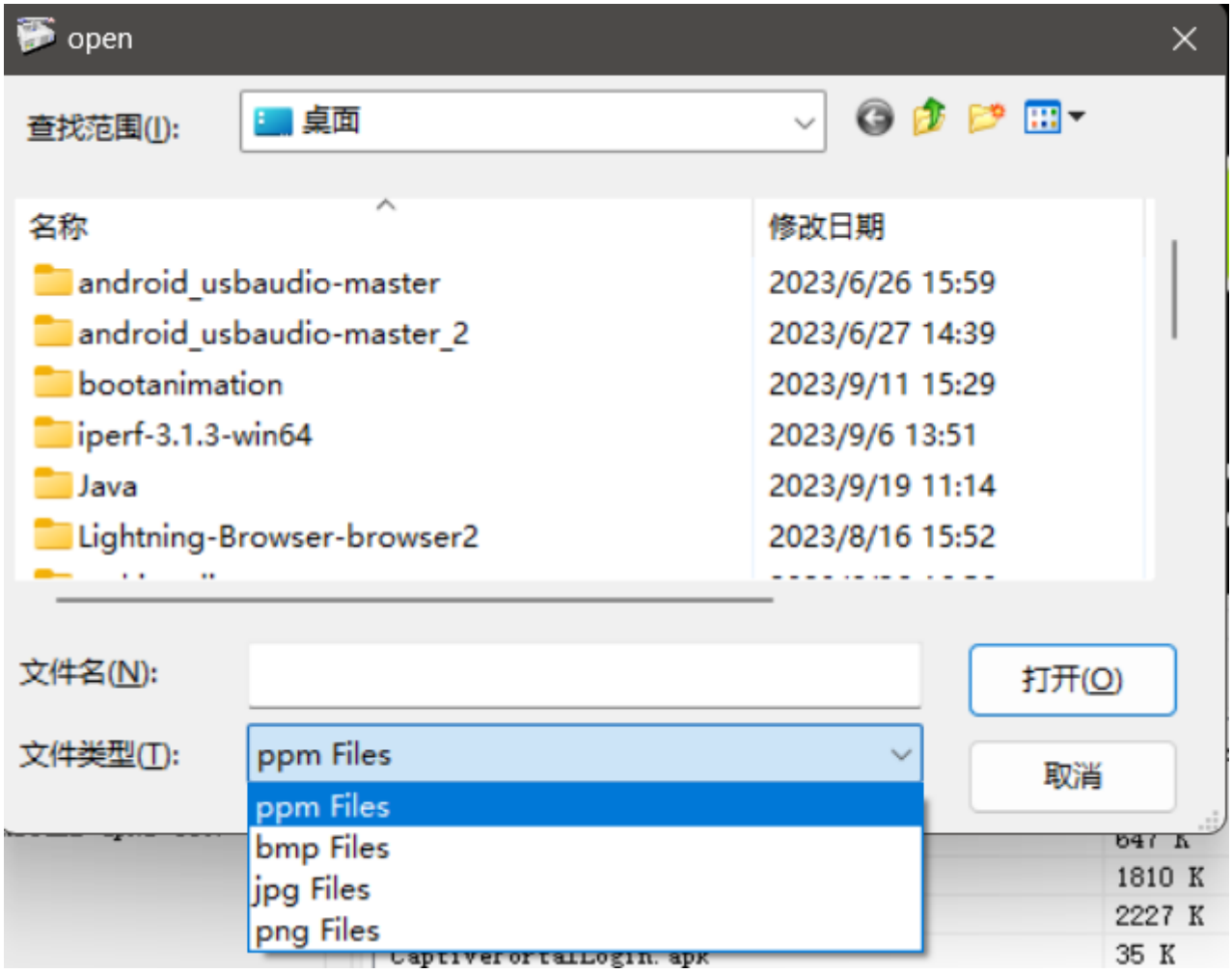
Note：Firmware factory tools have different versions based on the Android version and are not interchangeable between versions. Please select the firmware factory tool corresponding to the development board in the link below and download the corresponding version from the official website.

Hint：Firmware factory tool download address: https://www.proculustech.com/tools/

2.1.1 Modify Boot Logo

This feature is only supported on Android 5.1 system. For modifying the boot logo on Android 7.1, please refer to the attention prompt box below. In the firmware factory tool interface, select the corresponding firmware (please consult your sales representative for firmware information), then click the 'Replace Boot Logo' button and choose the desired boot logo to replace. You can select an image file that meets the requirements based on the file types supported, including ppm, bmp, jpg, and png formats.





Note:

1. Firmware with the kernel configuration set to Bmp Logo supports replacing multiple resolutions and various image formats. The resolution is not specifically set, but it is recommended to use the resolution of the development board as a reference. Otherwise, the image may be compressed due to resolution differences.
2. For Android 7.1, as the boot logo file is located in the kernel, self-defined modifications are currently not supported. Please consult your sales representative who is in contact with you, and we will make the necessary modifications in the source code.

2.1.2 Modify startup animation

The default boot animation for the Android system displays the word "ANDROID" on the screen. If you want to replace it with something else, you will need to modify the boot

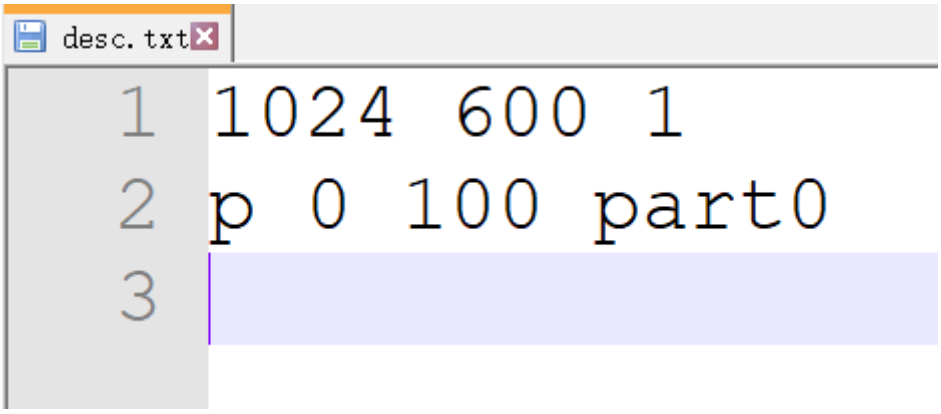
animation. Please refer to the method provided below.

1. Create startup animation

The default boot animation for Android displays an image with the word "ANDROID". To replace "ANDROID" with a different animation, you will need to use the bootanimation.zip file. Please rename your desired boot animation file to "bootanimation.zip". The file should contain two parts: the "part0" folder and the "desc.txt" file.

The "part0" folder: This folder contains the animation images that will be displayed during boot. The resolution of these images must match the device's screen resolution. The file names should follow either the format "0001.png" to "0099.png" or "001.png" to "099.png". The supported image formats are PNG and JPG. Please ensure that the file names do not include special characters such as *(), -_, etc.

"desc.txt": This file should be generated on a Windows or Linux system. The parameters for the file are as follows (using a screen resolution of 1024x600 as an example)：



Open desc.txt on Windows

```
1024 600 1
p 0 100 part0
```

Here are the parameters that you can copy (Note: there is a space between each parameter, please do not remove the spaces, and remember to include an empty line after the third line):

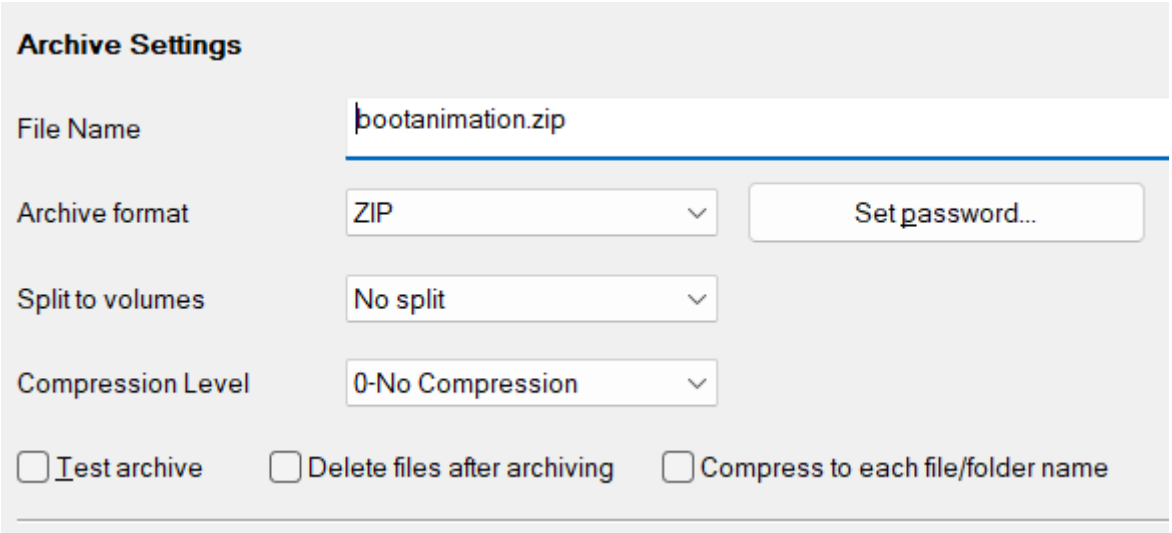
Parameter	Descriptions
1024 600 1	1024 600 - Screen resolution (width * height); 1 - Number of images played per second.
p 0 100 part0	p - Indicates play; 0 - Indicates loop playback; 1 - Indicates single playback; Here, p can be replaced with c. 100 - Delay time between image playback; part0 - Folder that stores the animation image files.

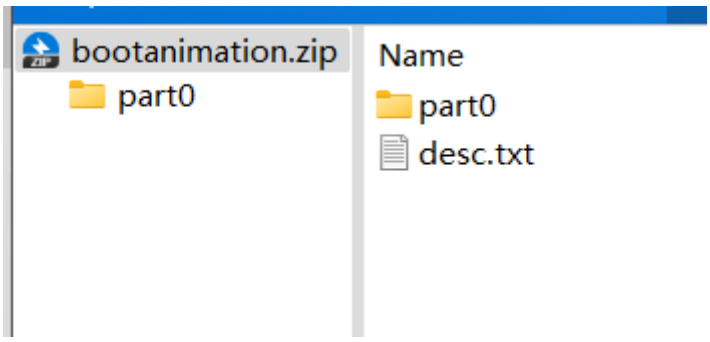
boot animation can also have multiple folders to achieve the best display effect by setting parameters. For example:

```
1024 600 5
p 1 0 part0
p 0 0 part1
```

The above example demonstrates playing a boot animation on an Android device with a resolution of 1024*600, at a speed of 5 images per second. It starts by playing the images in the "part0" folder, with no delay between each image. After the "part0" animation completes, it proceeds to play the images in the "part1" folder in an infinite loop until the system finishes booting.

⚠ Note："c" and "p" are two different animation playback modes. "p" stands for "play once," which means the animation may be interrupted. "c" stands for "continue," which means even if the Android boot process is complete, it will still wait for the animation to finish before entering the interface.





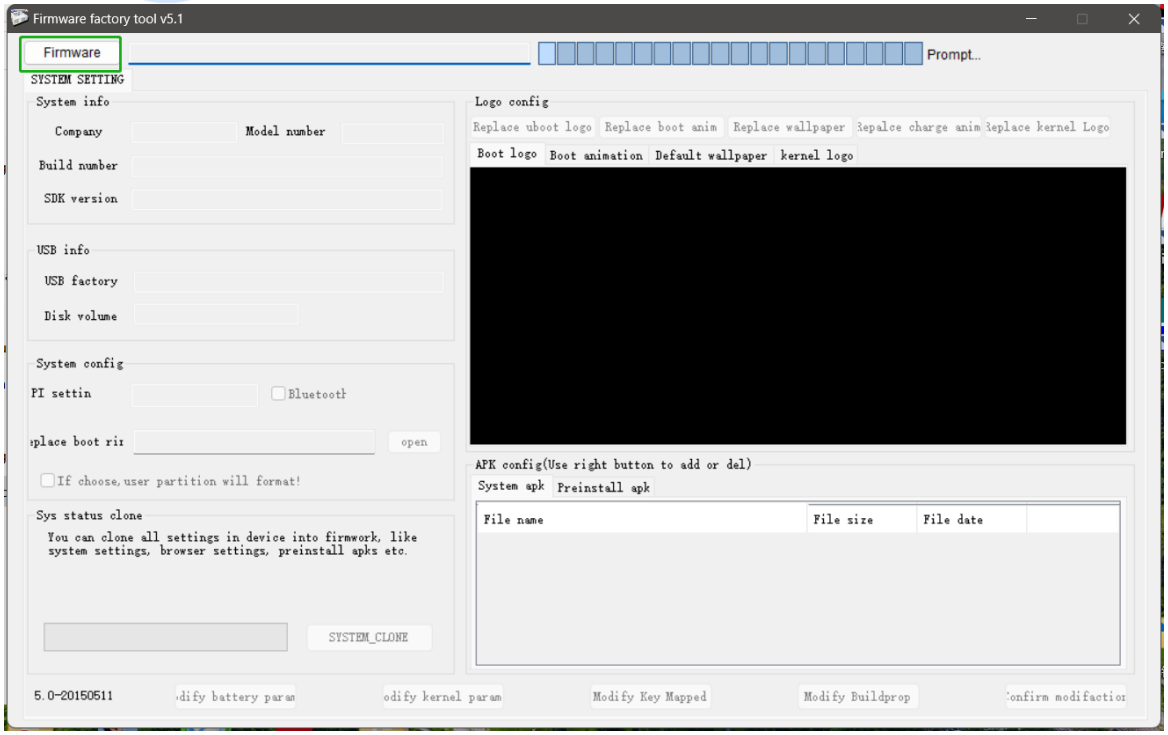
After setting up the "part0" folder and the "desc.txt" file, please compress the "bootanimation" folder using the following steps:

- Select the folder and compress it using the ZIP format.
- Choose the compression method as "Store only."

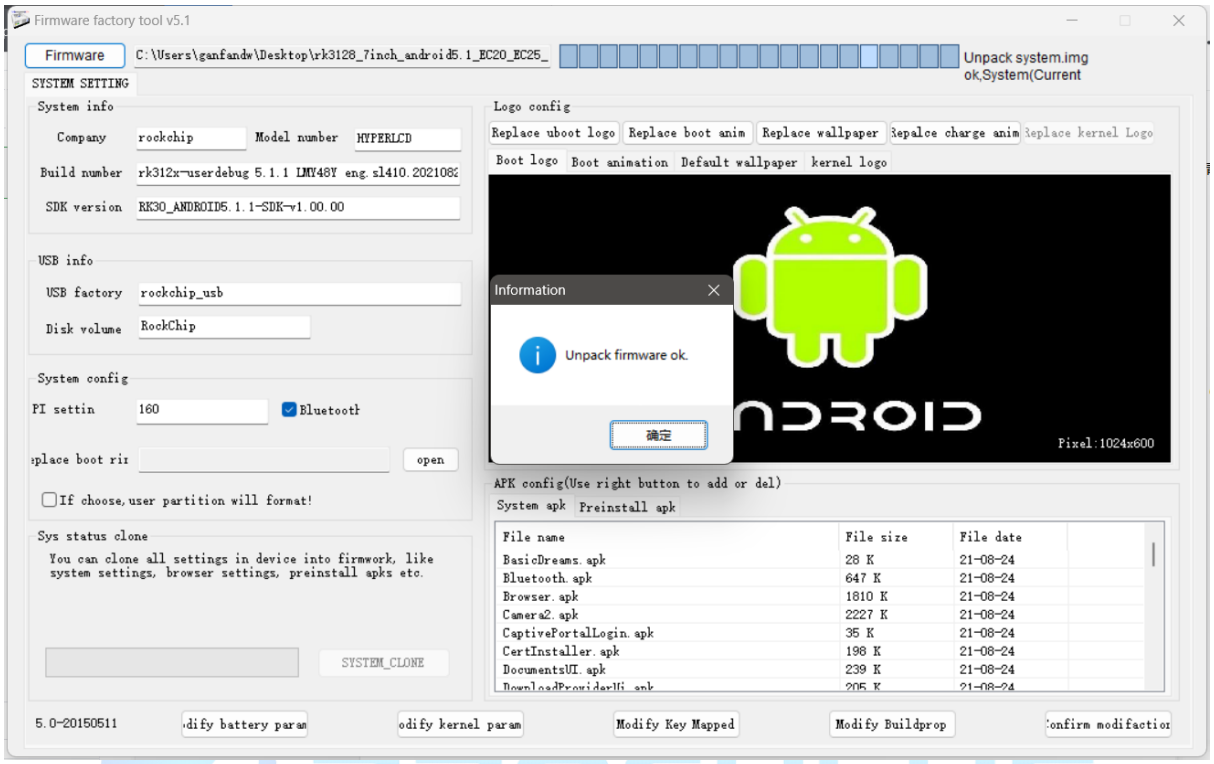
Once the compression is complete, double-click to open the compressed file and check for any extra files. If there are any, delete them. Otherwise, the boot animation may not work. The compressed package should contain only the "desc.txt" file and the "part0" folder, as shown in the example above.

2. Loading the boot animation

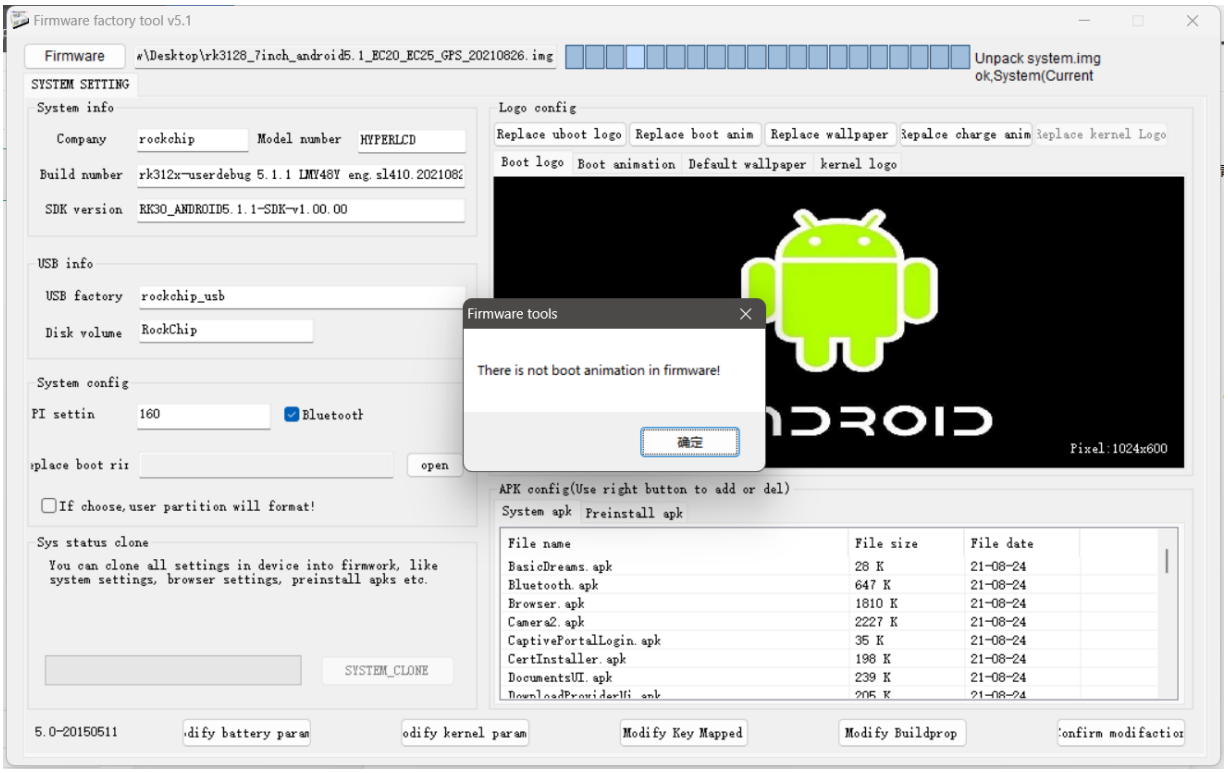
Click on the firmware to open the firmware that needs to be modified. Once selected, the software will automatically unpack the firmware.

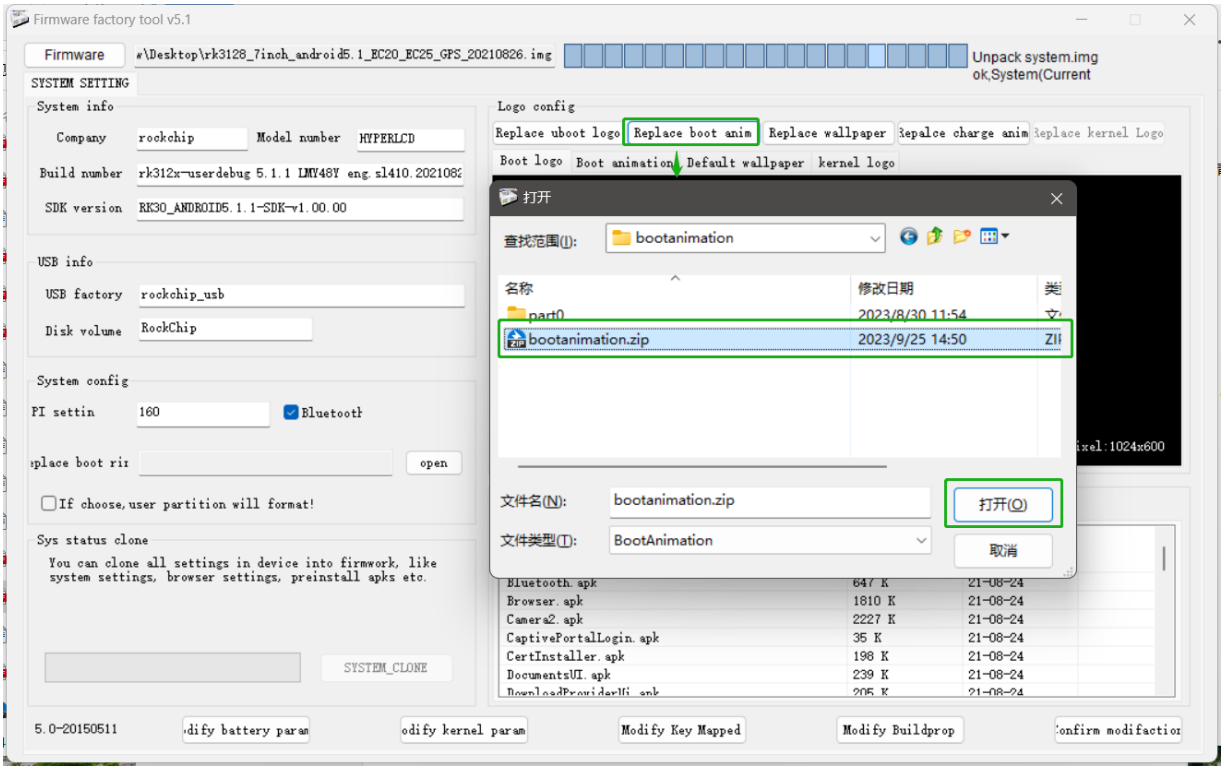


Hint：For default IMG firmware, please inquire with your sales representative.



If the firmware does not contain a boot animation, please click "OK" to proceed.

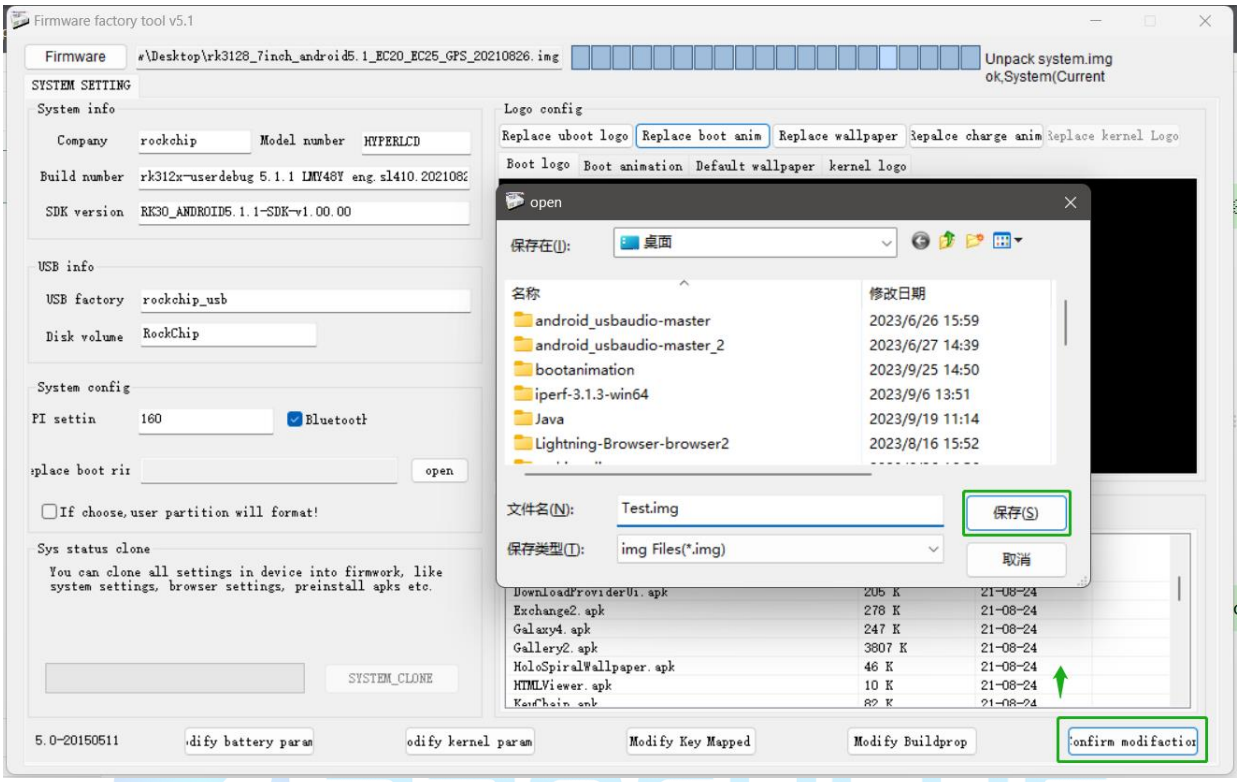




Replace Boot Animation" button to import the prepared boot animation.

Preview the effect of the boot animation.

After completing the modification, click on "Confirm modification" and save the IMG file.



2.1.3 Add startup sound

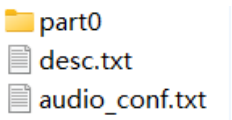
If you want to play startup sound when booting up, please follow the instructions below.

- **add a startup sound in Android 5.1**

Please follow section 2.1.2 to create a boot animation and create a new file named "audio_conf.txt". The contents of the txt document are as follows:

```
card=0
device=0
period_size=1024
period_count=4
```

Total 5 lines (including one blank line)



To add a WAV audio file named "audio.wav" to part0,To package the system according to section 2.1.2

- **add a startup sound in Android 7.1**

To create a boot animation in accordance with Section 2.1.2, add a WAV audio file named "audio.wav" to part0.

2.1.4 Flash the system firmware program

Android modules, before leaving the factory, will be pre-installed with the corresponding version of the Android system. If you have other special settings, you need to flash the Android system files (IMG files) for a second time. Please follow the steps below to flash the firmware program.

1. Open Android firmware flashing tool

Please refer to the following table for the corresponding Android firmware flashing tool for development boards.

Android firmware flashing tool	Applicable product
AndroidTool_V2.54	For RK3128/3188/3288 development boards, the img file flashing tool can be used to flash Android and Linux-based systems.

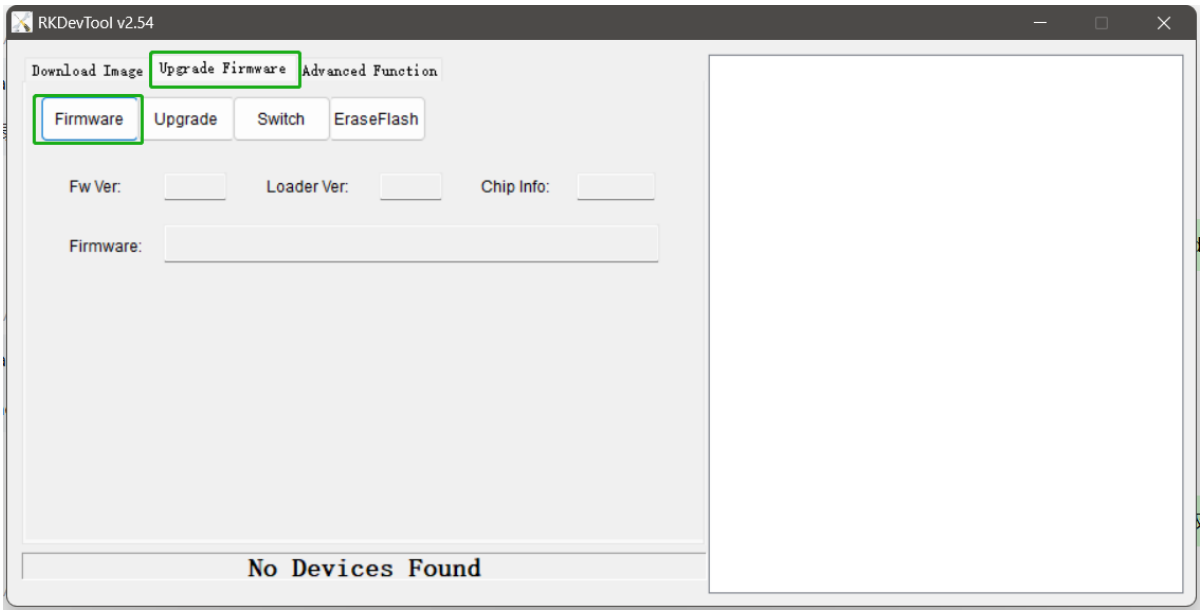
Note：Before flashing the firmware, you must first install the driver.

The driver can be downloaded from the following address:

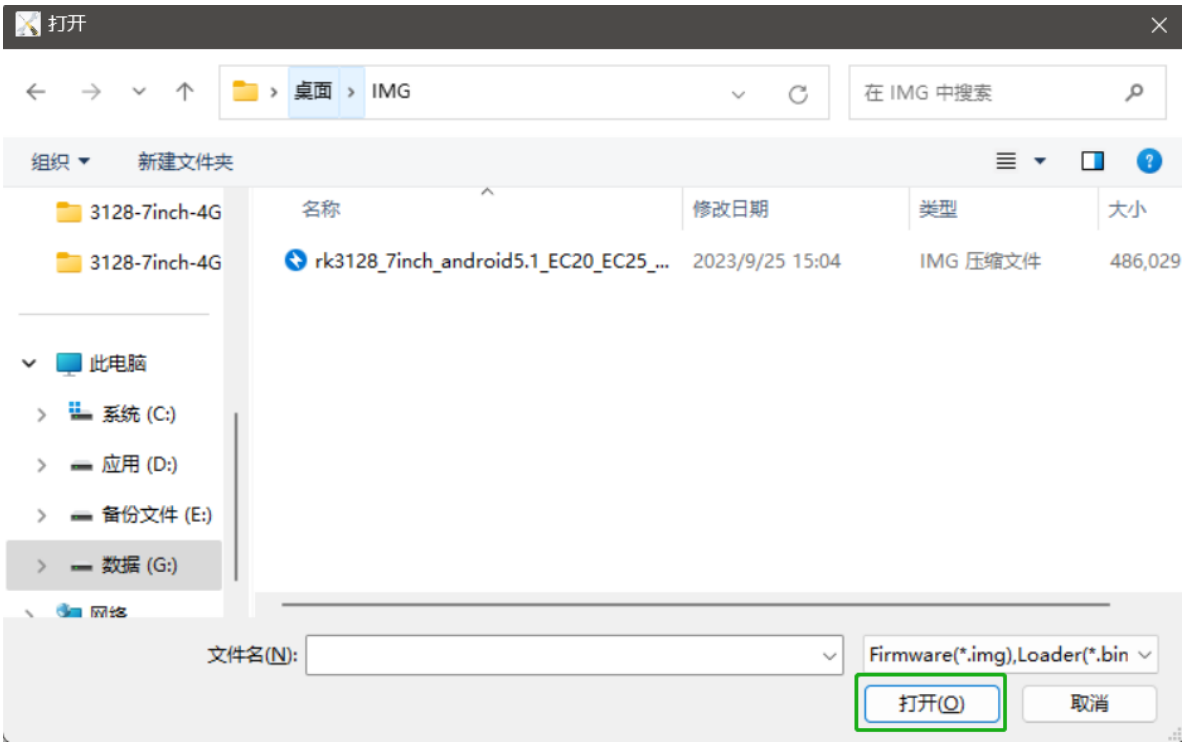
Android Tool tool download address:

https://www.proculustech.com/tools/

2. **Click "Upgrade Firmware" to enter the firmware selection interface.**



3. **Connect the module and select the firmware that needs to be burned.**

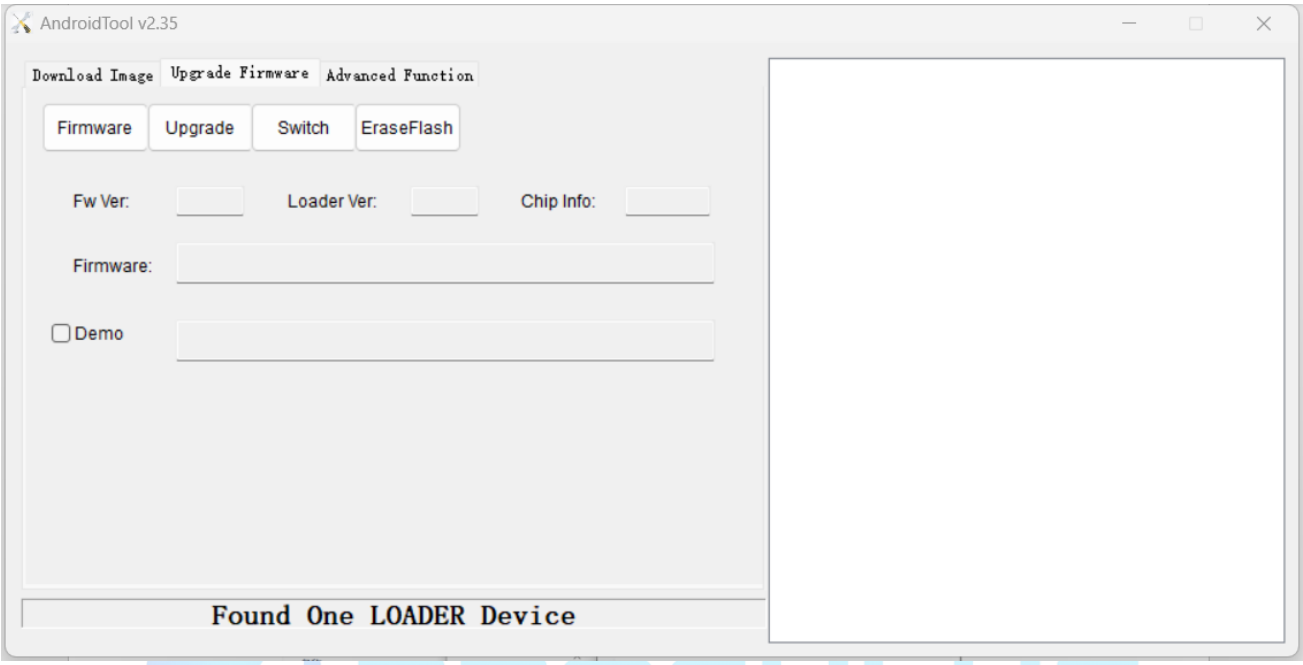


4. **Enter system flashing mode**

Note:

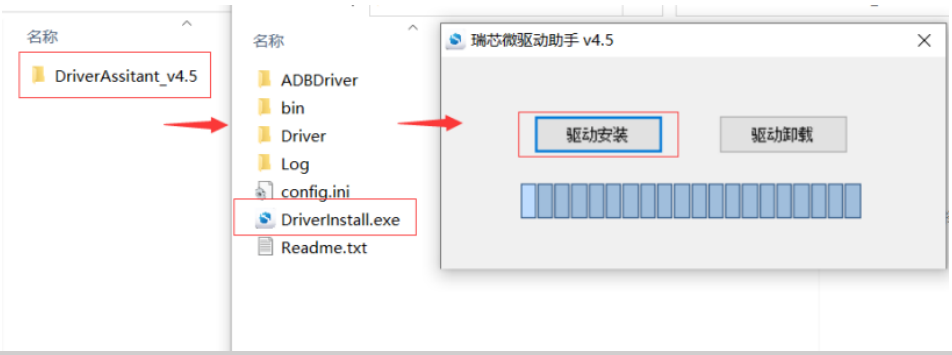
Please refer to section 1.3 of this document for entering the system flashing mode. If you cannot solve the problem you encountered, please contact your sales representative for technical support.

5. **When the software displays "Found a LOADER device," it indicates that the connection has been recognized. At this point, you can release the tweezers' short circuit.**

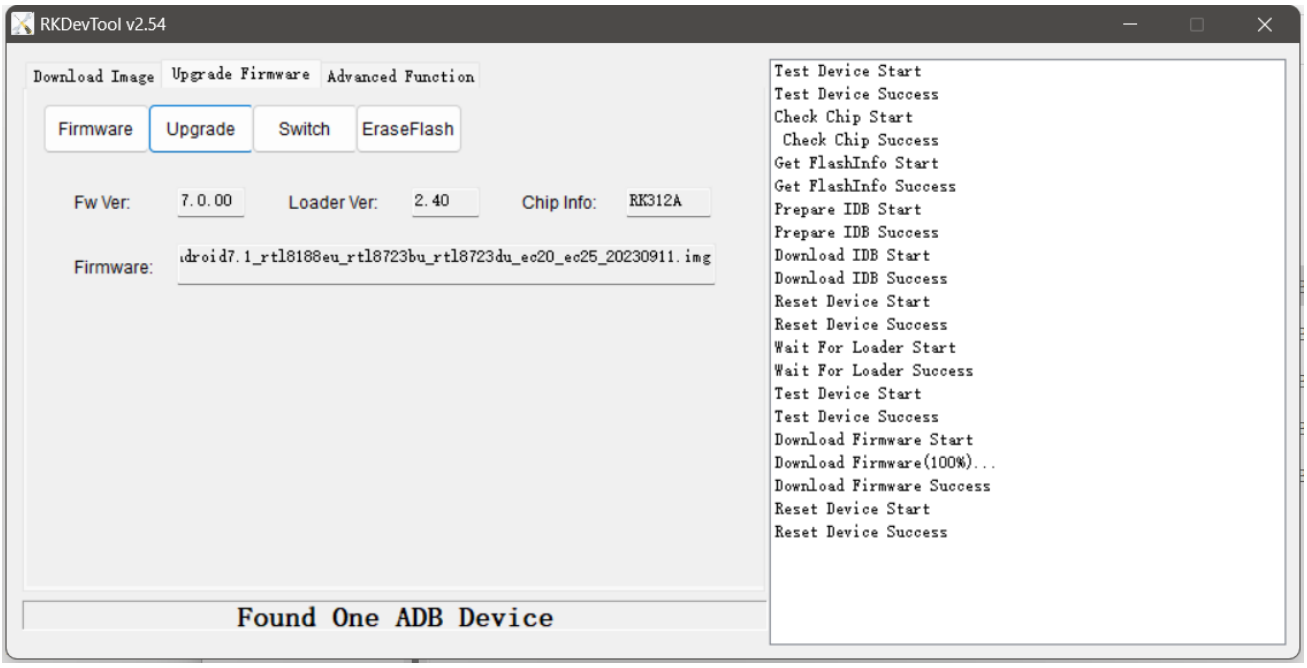


Note:

If unable to load the firmware normally, install the driver first. Driver download address: [Click here to download](#).



6. **Click on Upgrade, and the software will start loading and flashing the img automatically.**



The right side indicates that the firmware download was successful, and the device has been successfully restarted. At this point, the img system file of your device has been successfully updated.

Note:

If you receive a flashing failure message, please check if the firmware factory tool, flashing tool, and directory names contain special characters such as letters and no spaces. Additionally, having special characters or Chinese characters in folder names may cause flashing failures.

Chapter 3: Software Development Environment Setup

We provide complete hardware configuration and an Android open-source system environment. Based on different interface definitions, we offer customized hardware interface design services (please contact our sales team for more details). In terms of software development, you can provide us with your development requirements and technical specifications. Please consult our sales team to negotiate the cost of software development. Alternatively, you have the option to develop the software on your own or outsource it to a third party.

This chapter provides basic instructions for setting up the Android development environment. These instructions will help you quickly get started with using and testing Android screens. We provide support for the source code of user-side example software APKs and assistance with related technical issues. For more details, please contact your technical personnel.

⚠ Note:

During the product development process, if your software development setup involves related system-level settings, please contact our technical personnel to modify the Android system IMG file.

📌 Tip:

Configuring an IDE is an important part of Android application development, as a good IDE can greatly improve development efficiency.

3.1 Installation and Setup of Java Environment

3.1.1 Tool List

- JDK toolkit

JDK (Java Development Kit) is a software development toolkit for the Java programming language. It includes the Java runtime environment, a collection of tools, and foundational libraries for Java development.

- Android SDK

Android SDK (Software Development Kit) is a development toolkit provided by Google for Android. When developing Android applications, it is necessary to import this toolkit in order to utilize the Android-specific APIs.

- Android Studio

Android Studio is an official IDE (Integrated Development Environment) tool introduced by Google. Unlike Eclipse, which used to be in the form of a plugin, Android Studio is a standalone software that offers even greater power and convenience for Android application development.。

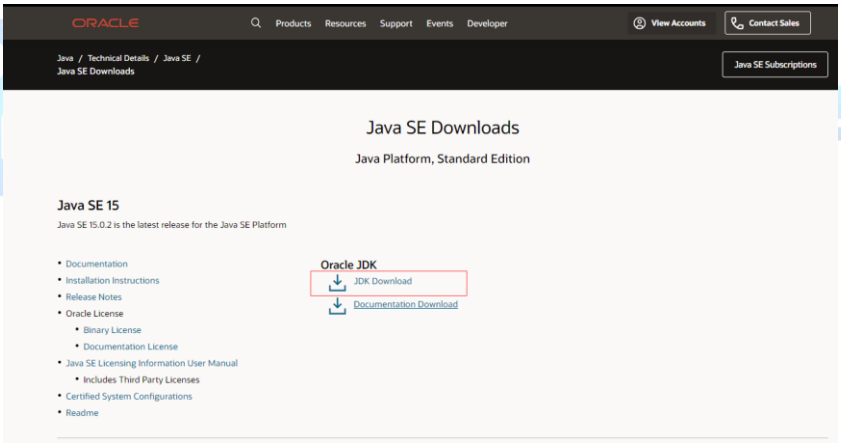
Tip:

You can choose different IDE environments, but Android Studio is recommended by our company as the Android development environment. If you are using other Android development environments, you may skip this section.

3.1.2 Download Java SDK

Please download the Java SE Development Kit (JDK). If the network speed is slow, please consider using a proxy server or VPN support.

Download link:http://www.oracle.com/technetwork/java/javase/downloads/index.html



Note:

Please choose the corresponding software version based on your computer's operating system. In this document, we will use Windows 64-bit system as an example.

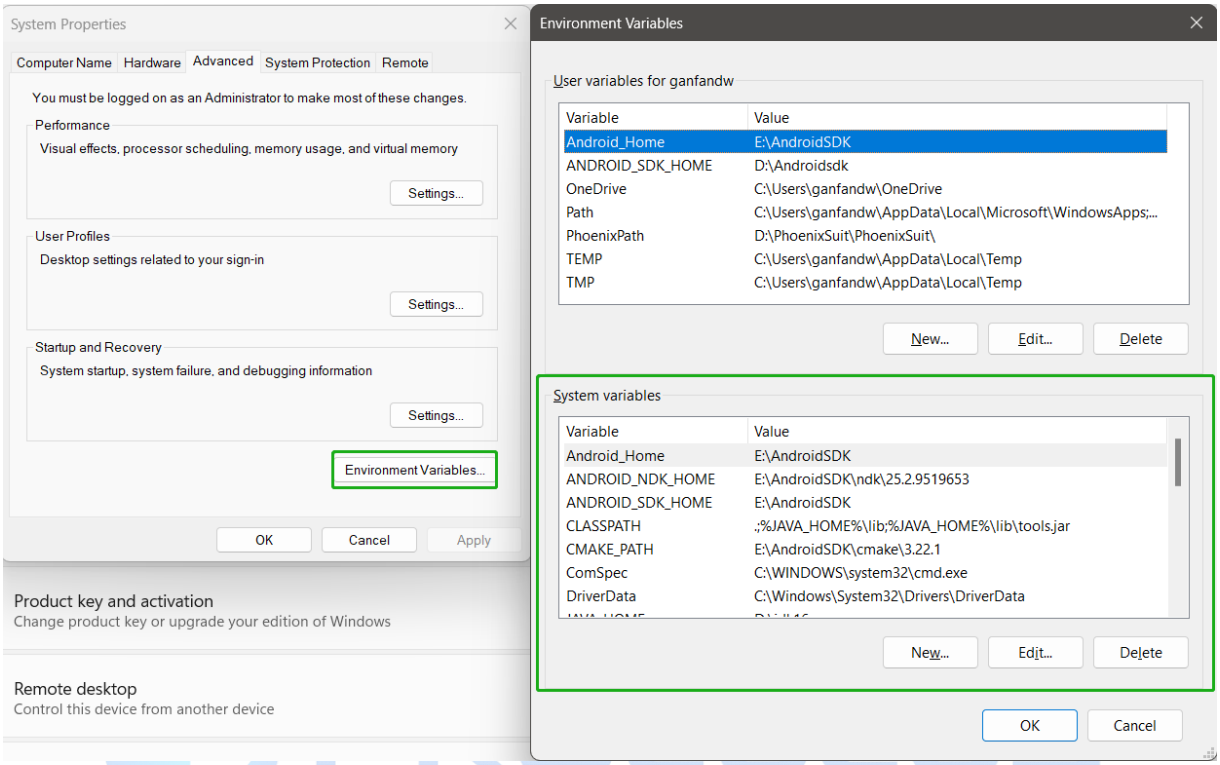
Please follow the JDK installation prompts to complete your download. During the download, the JRE (Java Runtime Environment) will also be installed. You can customize the installation directory and other information during the installation process. For example, let's choose the installation directory as C:\Program Files\Java\jdk-14\

3.1.3 Configure Environment Variables

1. After the installation is complete, right-click on the Windows start menu, select "System" then scroll down and choose "System Information" and click on "Advanced system settings" in the left column.

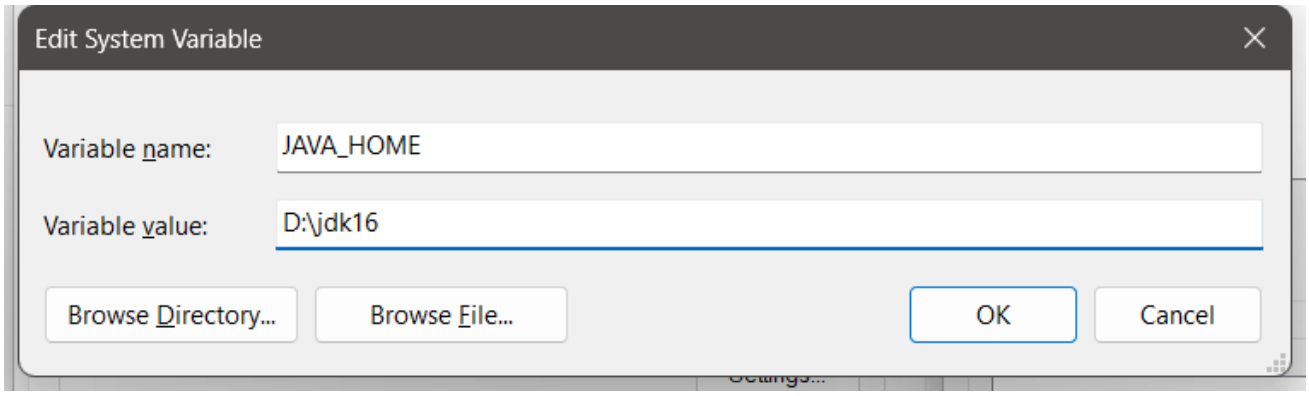
2. Select the "Advanced" tab, and click on "Environment Variables."

3. Configure 3 values in “System Variables”: “JAVA_HOME”, “PATH” and “CLASSPATH”. Click “Edit” if they exist.Otherwise click “New”.

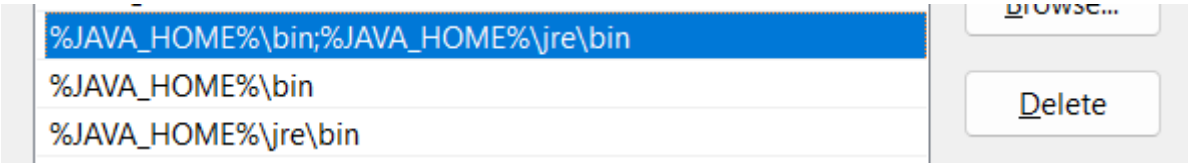


Name	Value
JAVA_HOME	C:\Program Files \Java\jdk-14
PATH	%Java_Home%\bin;%Java_Home%\jre\bin;
CLASSPATH	.;%Java_Home%\bin;%Java_Home%\lib\dt.jar;%Java_Home%\lib\tools.jar

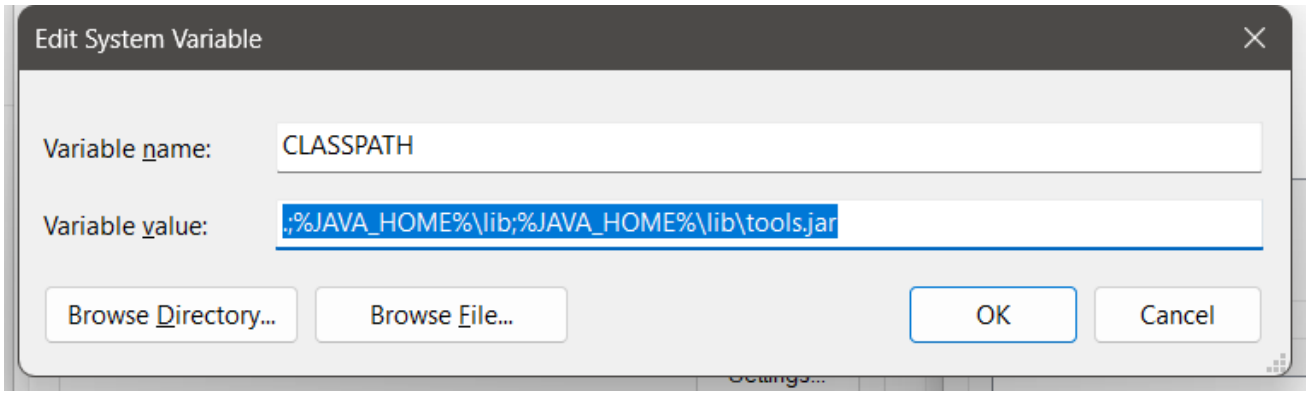
- JAVA_HOME SETTINGS



- PATH SETTINGS



- CLASSPATH SETTINGS



3.1.4 Test the installation status of JDK

1. Start > Run, type cmd;
2. Type the command "java -version" and if you see similar information displayed, it indicates that the environment variable configuration is successful.

```
java version "14.0.1" 2020-04-14
Java(TM) SE Runtime Environment (build 14.0.1+7)
Java HotSpot(TM) 64-Bit Server VM (build 14.0.1+7, mixed mode, sharing)
```

3.2 Installation and Environment Setup of Android Studio

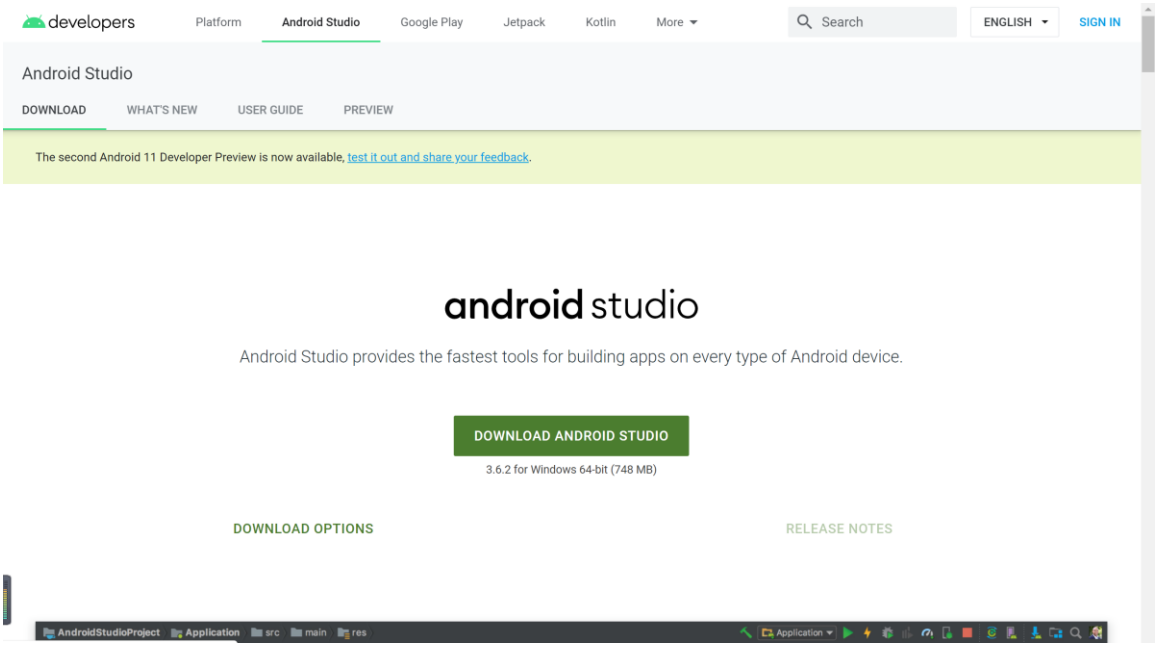
3.2.1 Install Android Studio

Download Android Studio from the following website to get the latest version:
https://developer.android.google.cn/studio/

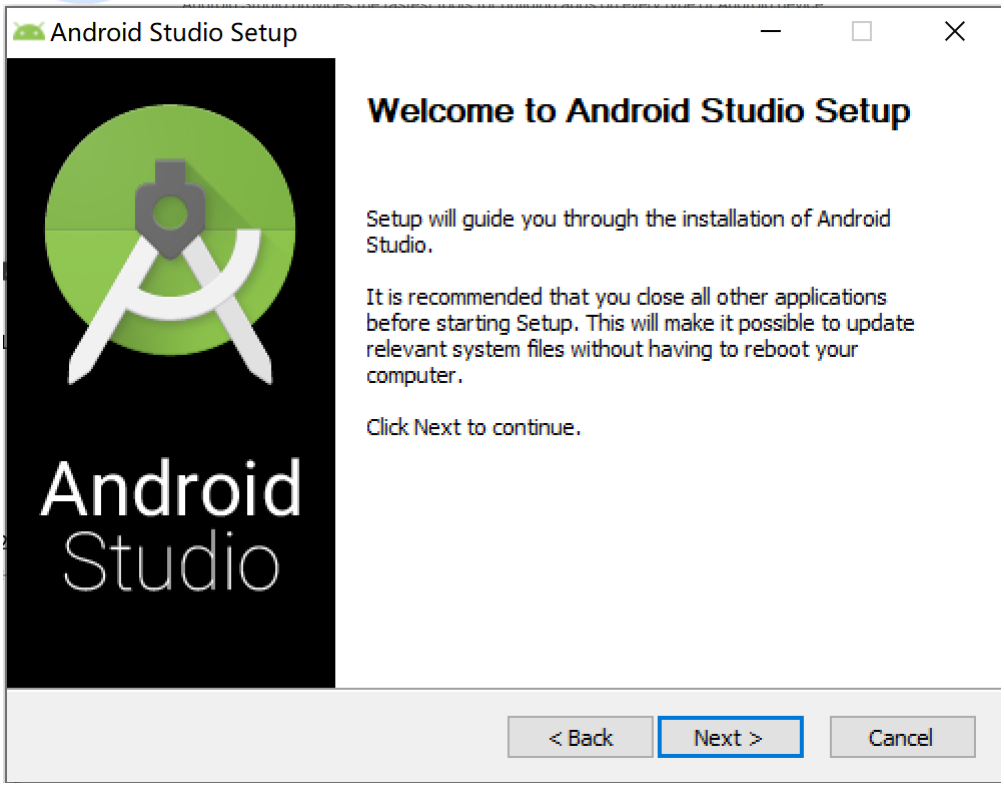
⚠ Note:

The release of Android Studio versions depends on the updates from Google Developers. You do not have to update the product version constantly; just ensure it meets your development needs. The

updates and releases of Android Studio may not be notified separately in this document.



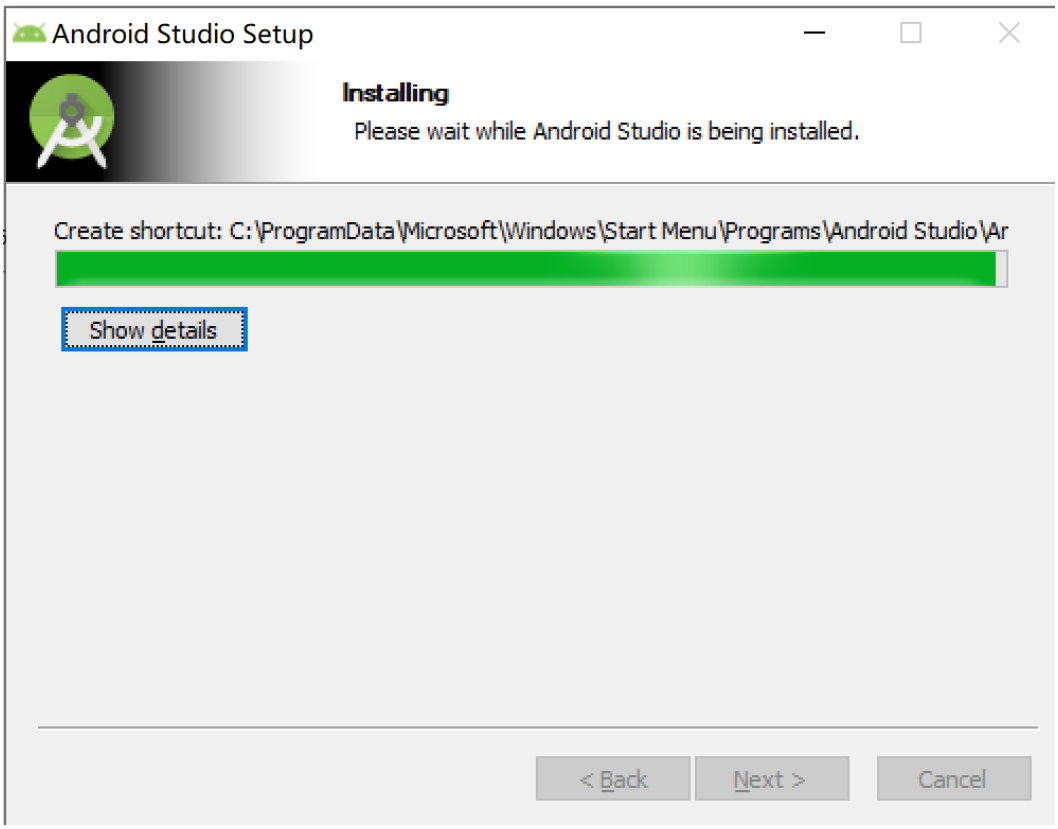
After the download is complete, double-click the SETUP installer to start the installation. Then click "Next", as shown in the following image



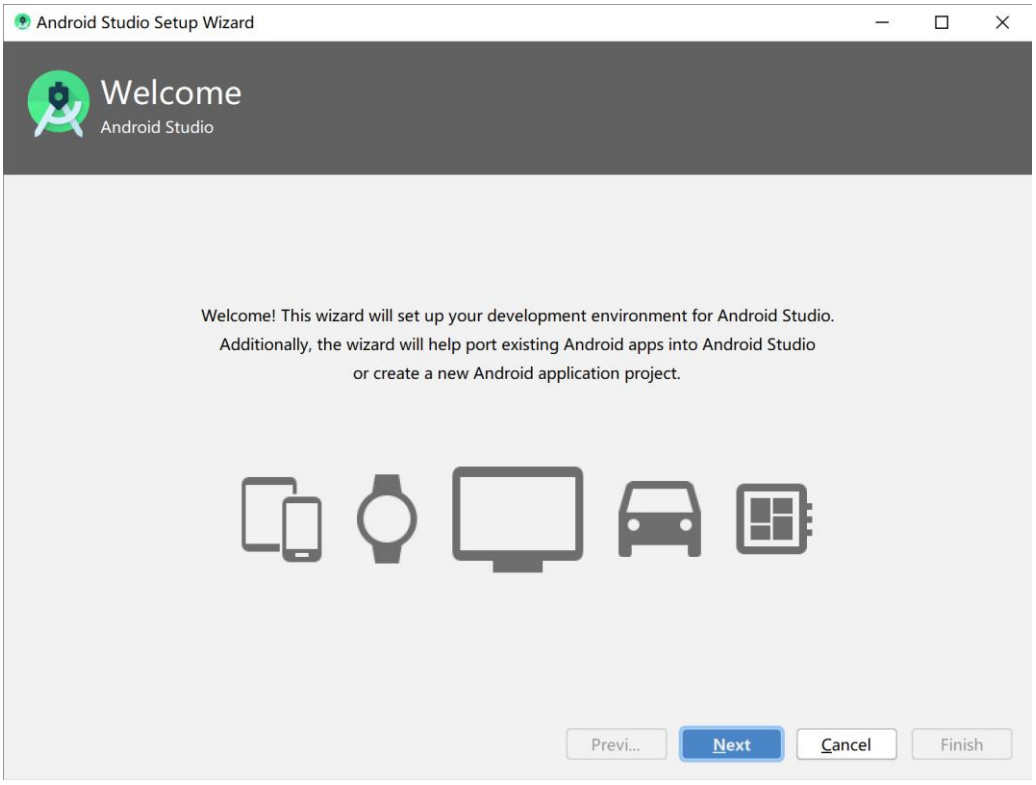
Click on the "I Agree" button, as shown in the following image.

Choose the installation location for Android Studio and Android SDK, then click "Next".

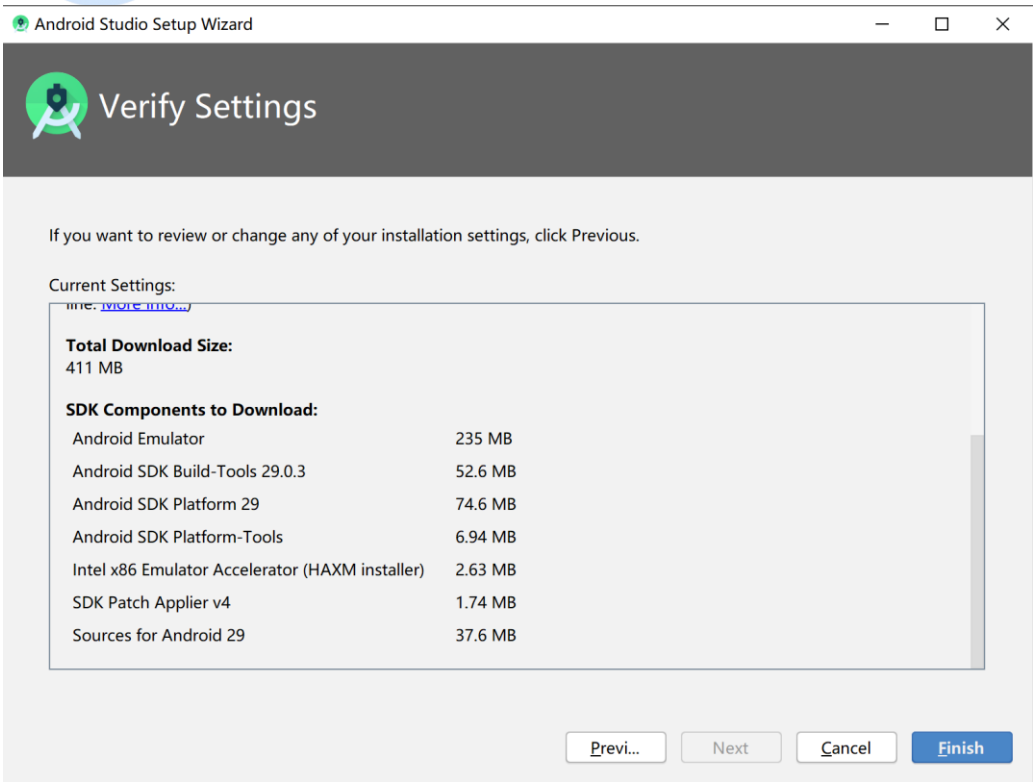
Click the "Install" button to start the installation process, as shown in the following image:

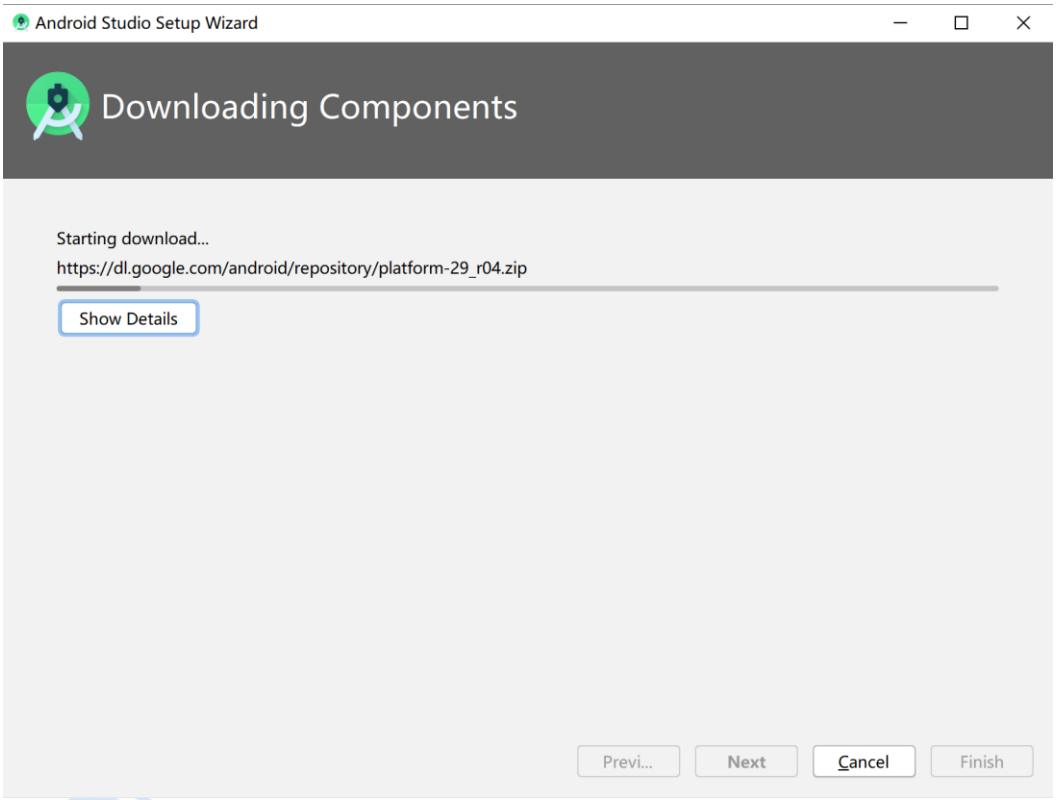


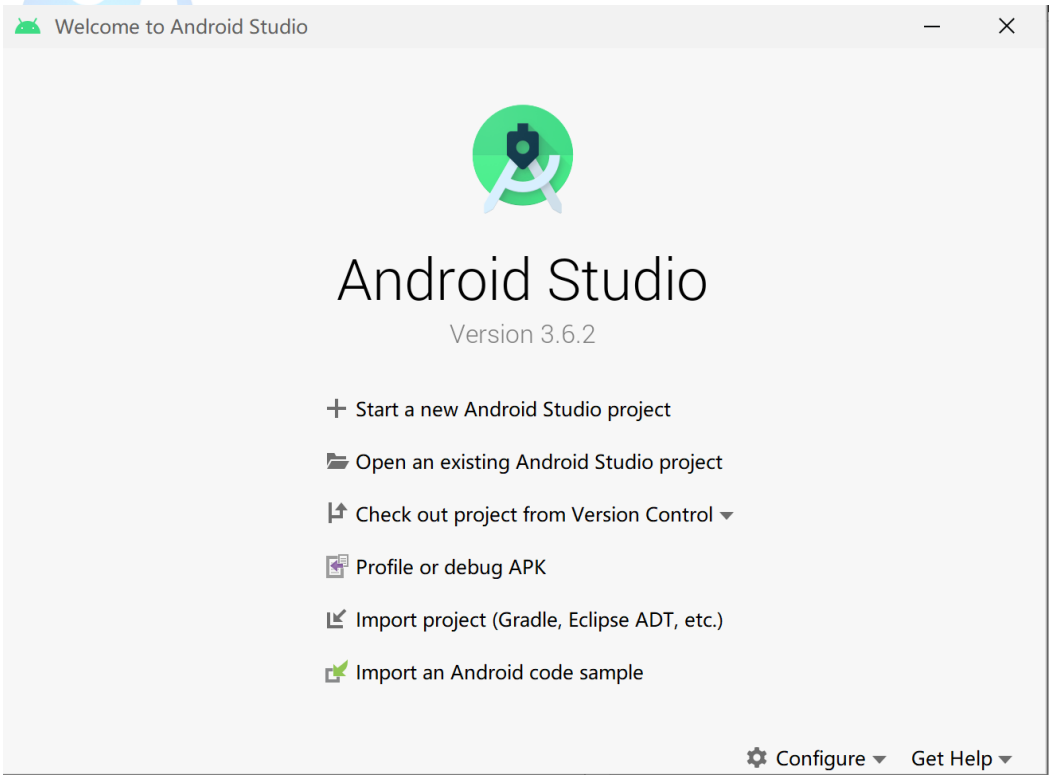
Wait until the installation is completed, as shown in the following image:



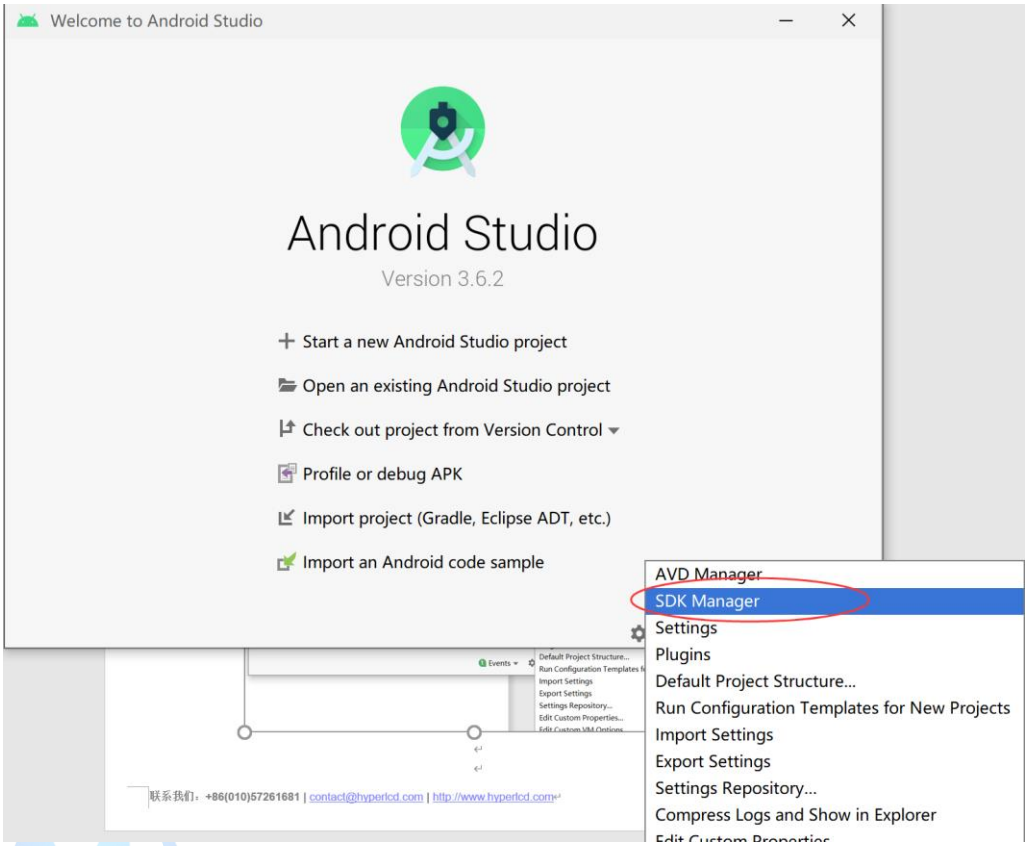
After entering the Android Studio welcome screen, you can proceed with the configuration and download of the related SDK tools. These configurations include but are not limited to the download and usage of Android SDK-related resources.



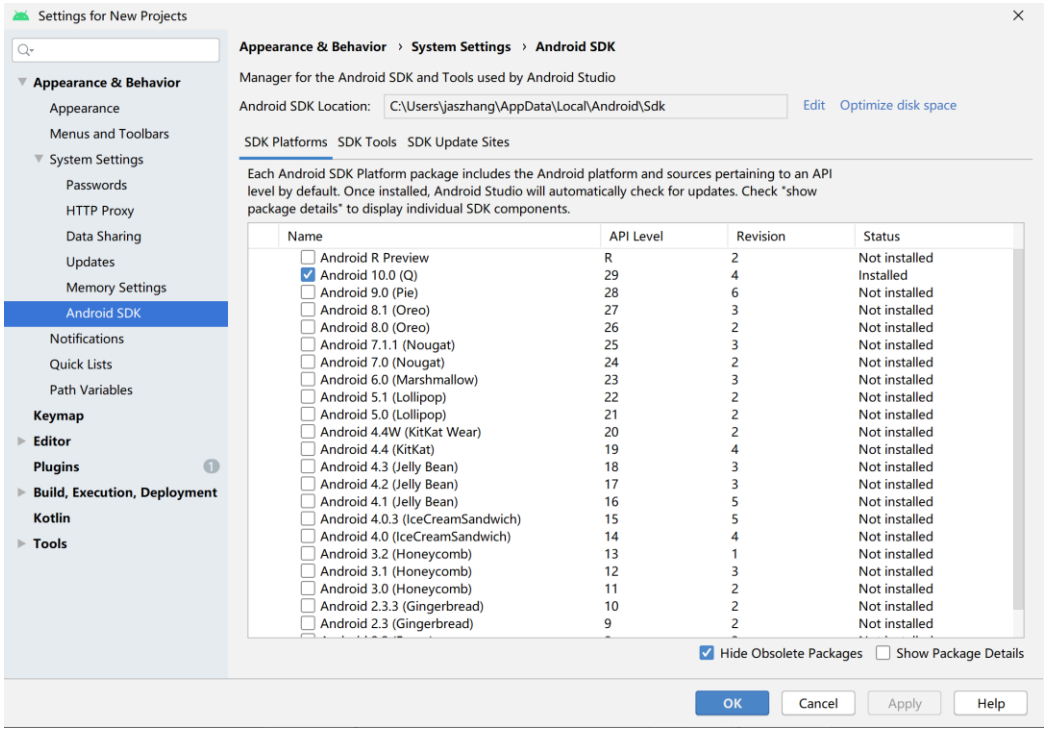


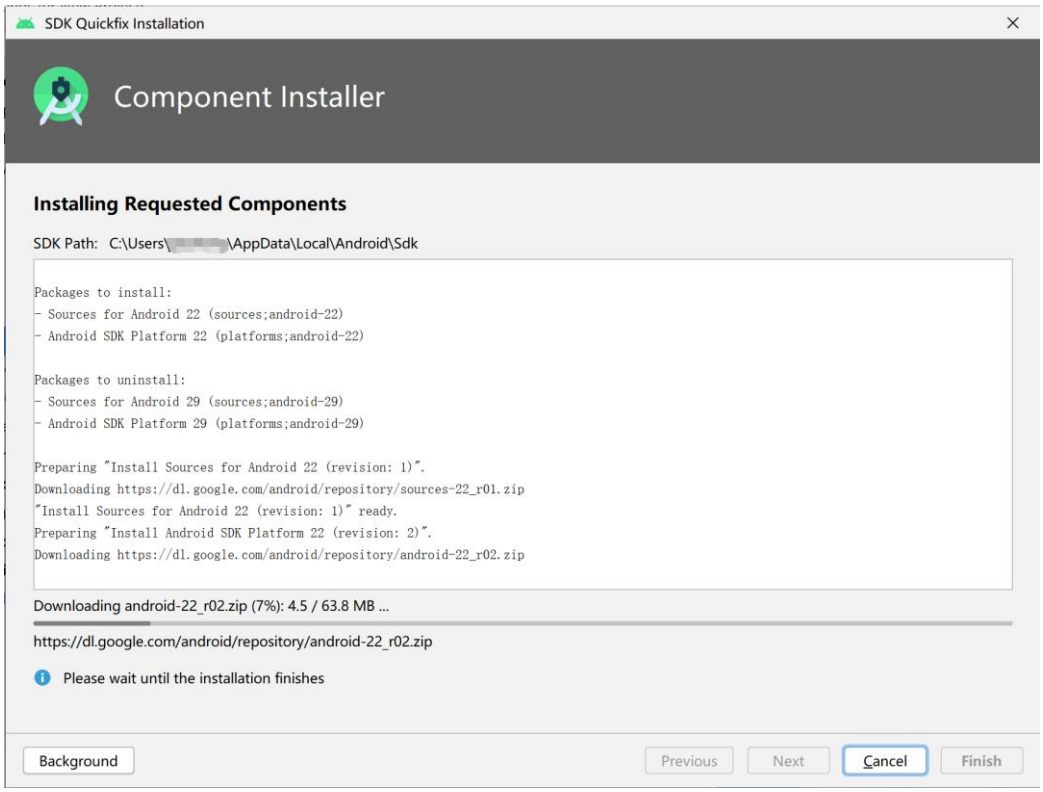


Select "SDK Manager", as shown in the following image:



In the "System Settings" section, click on "Android SDK", as shown in the following image:





3.2.2 Configure the NDK development environment

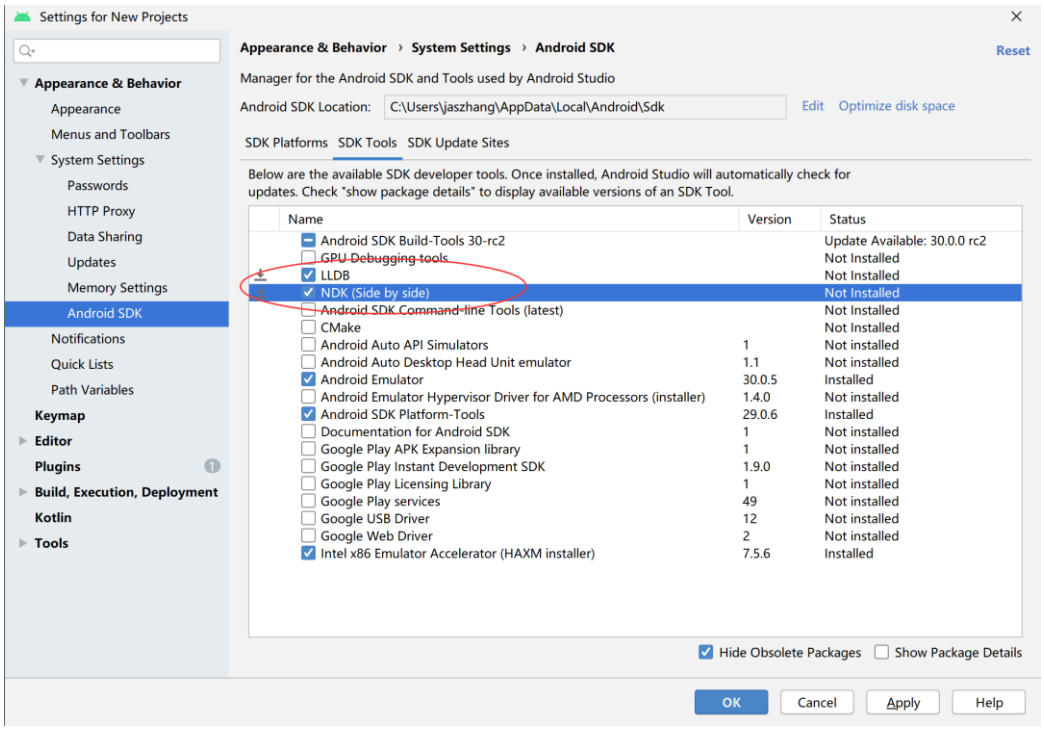
NDK (Native Development Kit) is a set of tools that allows you to utilize C and C++ code in your Android applications. It can be used to build from your own source code or make use of existing pre-built libraries. With NDK, you can achieve native-level performance and functionality in your Android applications by leveraging C and C++ capabilities.

1. Installing NDK

Open Tools->Android->SDK Manager->SDK Tools, select LLDB and NDK, and click OK. The software will automatically install NDK.

2. Import the .so library into the "libs" directory (in the project structure)

.so file is a program function library in Linux, which contains compiled code and data that can be used by other programs. The supported ABI (Application Binary Interface) for an Android application depends on the .so files located in the lib/ABI directory within the APK file.


Settings for New Projects
Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
Passwords
HTTP Proxy
Data Sharing
Updates
Memory Settings
Android SDK
Notifications
Quick Lists
Path Variables
Keymap
Editor
Plugins
Build, Execution, Deployment
Kotlin
Tools
Appearance & Behavior › System Settings › Android SDK
Reset
Manager for the Android SDK and Tools used by Android Studio
Android SDK Location: C:\Users\jaszhang\AppData\Local\Android\Sdk
Edit
Optimize disk space
SDK Platforms
SDK Tools
SDK Update Sites
Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.
Name
Version
Status
Android SDK Build-Tools 30-rc2 Update Available: 30.0.0 rc2
GPU Debugging tools Not Installed
LLDB Not Installed
NDK (Side by side) Not Installed
Android SDK Command-line Tools (latest) Not Installed
CMake Not Installed
Android Auto API Simulators 1 Not installed
Android Auto Desktop Head Unit emulator 1.1 Not installed
Android Emulator 30.0.5 Installed
Android Emulator Hypervisor Driver for AMD Processors (installer) 1.4.0 Not installed
Android SDK Platform-Tools 29.0.6 Installed
Documentation for Android SDK 1 Not installed
Google Play APK Expansion library 1 Not installed
Google Play Instant Development SDK 1.9.0 Not installed
Google Play Licensing Library 1 Not installed
Google Play services 49 Not installed
Google USB Driver 12 Not installed
Google Web Driver 2 Not installed
Intel x86 Emulator Accelerator (HAXM installer) 7.5.6 Installed
Hide Obsolete Packages
Show Package Details
OK
Cancel
Apply
Help

Chapter 4：Examples of Android Development.

4.1 Serial Port Development Example

Serial communication refers to a communication method where data is transmitted between an external device and a computer through data signal lines, ground wires, control lines, etc. on a bit-by-bit basis. Serial ports send and receive bytes on a bit level. Although slower than parallel communication on a byte-by-byte basis, serial ports can send and receive data using different lines simultaneously. It is simple and can achieve long-distance communication.

The most important parameters for serial communication are baud rate, data bits, stop bits, and parity. These parameters must match for the communicating ports.

We provide a serial port development sample program. Before starting your project development, you can refer to our related sample program source code to successfully complete the communication design of the serial part of your application software.
Download link：https://www.proculustech.com/tools/

⚠ Note:
The source code design for serial communication may vary depending on the diversity of the product, and the program provided as a reference example is for learning and understanding the principle of the code. The usability of the serial port for your product may need to be adjusted during actual use depending on the performance of your downstream machine, please consult with the development personnel for related questions.

4.1.1 Serial communication steps

Implementation steps:

1. Serial port initialization - Create a serial port instance and set the serial port parameters.
2. Get the input stream - Read serial port data.
3. Get the output stream - Send data to the serial port.
4. Data processing and display.
5. Close the serial port.

4.1.2 Serial Port Demo Code Explanation

Taking the serial port example program 2_v5.1 as an example, our company provides a code reference for a serial port sample program. Please download the relevant source files from the following address:https://www.proculustech.com/tools/

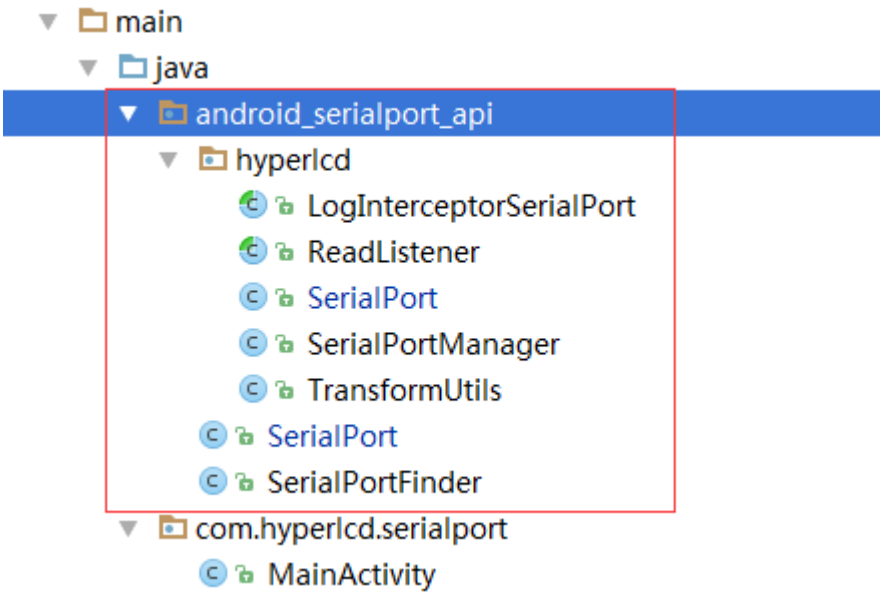
Step 1: Importing SO library

1. Copy the corresponding SO file under the architecture to the "jniLibs" folder of the project. (The operation steps are the same for Android 5.1-Android 7.0 systems.)
2. Add configuration under the Android tag in "build.gradle".

```
sourceSets {
    main {
        jni.srcDirs = []
        jniLibs.srcDirs = ['src/main/jniLibs', 'libs']
    }
}
```

Step 2: Copying Files

Copy all files from the "android_serialport_api" to the "java" folder of the current project, and please note that the path should not be changed.



Step 3: Using Serial Port

1. Declare global variables.

```
private BaseReader baseReader; // Read callback
private SerialPortManager spManager; // Serial port control

private String checkPort = "dev/ttyS1"; //Modify the port number here according to the needs of testing
private boolean isAscii = true; //When true, send in ASCII format
```

2. Initialize SerialPortManager

```
spManager = SerialPortManager.getInstances().setLogInterceptor(new LogInterceptorSerialPort() {
    @Override
    public void log(@SerialPortManager.Type final String type, final String port, final boolean isAscii, final
String log) {
        Log.d("SerialPortLog", new StringBuffer()
                .append("Serial port number: ").append(port)
                .append("\nData format: ").append(isAscii ? "ascii" : "hexString")
                .append("\nOperation type: ").append(type)
                .append("\nOperation message: ").append(log).toString());
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                logTV.append(new StringBuffer()
                        .append(" ").append(port)
                        .append(" ").append(isAscii ? "ascii" : "hexString")
                        .append(" ").append(type)
                        .append(": ").append(log)
                        .append("\n").toString());
            }
        });
    }
});
baseReader = new BaseReader() {
    @Override
    protected void onParse(final String port, final boolean isAscii, final String read) {
        Log.d("SerialPortRead", new StringBuffer()
                .append(port).append("/").append(isAscii ? "ascii" : "hex")
                .append(" read: ").append(read).append("\n").toString());
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                readTV.append(new StringBuffer()
                        .append(port).append("/").append(isAscii ? "ascii" : "hex")
                        .append(" read: ").append(read).append("\n").toString());
            }
        });
```

```
    }
};
```

3. Open the serial port

```
//checkport serial port number
//isAscii data format
//baseReader read callback
spManager.startSerialPort(checkPort, isAscii, baseReader);
```

4. Modify the data format for reading

```
// checkport serial port number
//isAscii data format
spManager.setReadCode(checkPort, isAscii);
```

5. Send data (The serial port needs to be opened first)

```
spManager.send(checkPort,"send");
// Write data to serial port currentPort, with the data content being: send
```

6. Close the serial port

```
spManager.stopSerialPort(checkPort);
// colse currentPort
```

7. Destroy the SerialPortManager. After destruction, if you need to use the serial port again, you will need to perform the initialization operation again

```
spManager.destroy();
```

Step 4: Precautions for serial port operation

1. When running, if you encounter 'java.lang.UnsatisfiedLinkError: … couldn't find "libserial_port.so"' error, it means that 'libserial_port.so' cannot be found. Please check if the jniLibs folder name is correct. Also, verify that the jniLibs folder is located correctly. In the project directory structure, the jniLibs folder should be inside the main folder, consistent with the location of the java folder.

2. If accessing certain serial ports causes freezing on an Android screen, please check if the software is accessing '/dev/ttyS2' port. This serial port is dedicated for system debugging and cannot be used by normal users. Please block this serial port in your code. For RK3288 development board, ttyS0 is occupied by a wireless Bluetooth combo module and is not available for users.

3. If the data sent through the serial port is not received successfully, when you send data via the serial port and receive a success message but the receiving end does not receive the information, please first check if your connecting cables are capable of transmitting and receiving data properly. You can try using a different cable to confirm its correctness. Check your code and ensure that the serial ports and baud rates are consistent between the receiving and sending ends. Please check the Logcat for any exceptions in the code.

4. ANR (Application Not Responding) Exception: There are two common cases that can cause ANR: 1) No response to input events (such as key presses or screen touches) within 5 seconds, and 2) BroadcastReceiver not completing execution within 10 seconds. When encountering an ANR exception, locate the code where this situation occurs and make modifications accordingly.

4.2 Introduction to Autostart on Boot

There are generally two ways to achieve auto-start on boot:

1. Start after receiving the boot broadcast: The APK will start after the system enters the desktop after booting.
2. Set a custom APK as the launcher: Skip the system desktop and directly enter the APK upon boot.

You can choose either method, but the second method is recommended.

4.2.1 Start after receiving the boot broadcast

When Android starts up, it sends out a system broadcast with the content ACTION_BOOT_COMPLETED.

Its string constant representation is android.intent.action.BOOT_COMPLETED. As long as this message is 'captured' in the program, you can start the desired action. Therefore, the implementation method is to create a Broadcast Receiver.

Step 1: Custom Broadcast Class BootReceiver

```
public class BootReceiver extends BroadcastReceiver {
    private SharedPreferences pref;
    private boolean autoStarts = false;
    @Override
    public void onReceive(Context context, Intent intent) {
        if(intent.getAction().equals("android.intent.action.BOOT_COMPLETED")) {
            // boot
           pref = context.getSharedPreferences("BOOT_COMPLETED", Context.MODE_PRIVATE);
            autoStarts = pref.getBoolean("Autostarts",false);
            if (autoStarts){
                Log.e("Autostarts"," Automatic Startup on Boot ");
                Intent intent2 = new Intent(context, MainActivity.class);
                intent2.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                context.startActivity(intent2);
            }else {
                Log.e("Autostarts"," Do not automatically start on boot ");
            }
```

```
            }
            if (intent.getAction().equals("android.intent.action.ACTION_SHUTDOWN")){
                Log.e("shutdown","shutting down);
            }
        }
}
```

Step 2: Manifest File Configuration

Inside the <application> element of the AndroidManifest.xml file, add the custom receiver.

```
<receiver
    android:name=".BootReceiver"
    tools:ignore="Instantiatable">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
        <action android:name="android.intent.action.ACTION_SHUTDOWN"/>
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</receiver>
```

Step 3: Add Permissions

Inside the <manifest> element of the AndroidManifest.xml file, add the boot startup permission.

```
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
```

Restart the Android module and test if the app starts automatically.

Install the app on the module and then launch the program once (starting the program once is required on Android 4.0 and later to receive the boot completed broadcast, which is to prevent malicious programs).

Check if security assistants like 360 are installed. If yes, please allow the app in the auto-start software management of such software. Install the app in internal storage instead of the SD card.

4.2.2 Set APK as system desktop

Replace the <intent-filter> of the first launched activity in the configuration file.

```
<activity android:name=".MainActivity">
    <intent-filter>
```

```
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
            <category android:name="android.intent.category.HOME" />
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
</activity>
```

After the program is running, click on 'Home'. A Launcher selection box will appear. Select your developed APK and click on 'Always'.

4.3 APK encryption

4.3.1 Prevent repackaging or secondary packaging

1. Add signature verification in Java code (directly modify smali files);
2. Add signature verification in NDK (use IDA to view and directly modify hexadecimal values);
3. Utilize the flaws of the repackaging tools themselves to prevent repackaging (e.g. deceiving the manifest, modifying image extensions, etc.)

4.3.2 Using third-party encryption tools

If you need to encrypt an application, export the APK resources and use a third-party encryption platform to complete the process.

4.4 How to use a buzzer

A buzzer is an integrated electronic sound device, which uses DC voltage as power supply and acts as a sound emitter in electronic products. The buzzer is controlled by GPIO to produce sound. The code for controlling a buzzer is as follows:

Step 1: Create the execShell method

```
public void execShell(String cmd) {
    try {
        // Permission setup
        Process p = Runtime.getRuntime().exec("su");
        //Get output stream
        OutputStream outputStream = p.getOutputStream();
         DataOutputStream dataOutputStream = new DataOutputStream(outputStream);
        //Write the command
        dataOutputStream.writeBytes(cmd);
```

```
            //Submit the command
            dataOutputStream.flush();
            //Close stream operations
            dataOutputStream.close();
            outputStream.close();
        } catch (Throwable t) {
            t.printStackTrace();
        }
}
```

Step 2: Choose the corresponding method based on the module version

```
//3128, 3288 - Use this method for 10-inch products.
public void newBuzzer(boolean flag,int port) throws IOException,
            InterruptedException {
        execShell("echo chmod 0777 /sys/class/gpio/export");
        execShell("echo "+port+"> /sys/class/gpio/export");
        execShell("echo out > /sys/class/gpio/gpio"+port+"/direction");
        if (flag){
            execShell("echo 1 > /sys/class/gpio/gpio"+port+"/value");
        }else {
            execShell("echo 0 > /sys/class/gpio/gpio"+port+"/value");
        }
}

//3288 - Use this method for 7-inch products.
public void newBuzzer(boolean flag) throws IOException, InterruptedException {
        if (flag){
            execShell("echo 1 > /sys/class/leds/beep/brightness");
        }else {
            execShell("echo 0 > /sys/class/leds/beep/brightness");
        }
}

//3128 - Use this method for 7-inch products.
public void newBuzzer(boolean flag) throws IOException, InterruptedException {
        if (flag){
            execShell("echo 0 > /sys/class/leds/beep/brightness");
        }else {
            execShell("echo 1 > /sys/class/leds/beep/brightness");
        }
}
```

4.5 GPIO Operations

Note:

GPIO functionality is only supported by some screens. Please consult with your sales representative who is working with you to confirm whether the module supports it.

Create a new GPIO class

```
public class Gpio {
    public Gpio() {
    }
    /**
     * Create a GPIO operation instance
     * @param gpio GPIO pin number
     * **/
    public String getGpio(String gpio) {
        String path = "echo " + gpio + "> /sys/class/gpio/export";
        return execRootCmd(path);
    }
    /**
     * Set GPIO as output
     * @param gpio GPIO pin number
     * **/
    public void setGpioOut(String gpio) {
        execRootCmd("echo out > /sys/class/gpio/gpio" + gpio + "/direction");
    }
    /**
     * Set GPIO as input
     * @param gpio GPIO pin number
     * **/
    public void setGpioIn(String gpio) {
        execRootCmd("echo in > /sys/class/gpio/gpio" + gpio + "/direction");
    }
    /**
     * Set GPIO to high level
     * @param gpio GPIO pin number
     * **/
    public void setGpioHigh(String gpio) {
        execRootCmd("echo 1 > /sys/class/gpio/gpio" + gpio + "/value");
    }
    /**
     *  Set GPIO to low level
```

```
 * @param gpio GPIO pin number
 * **/
public void setGpioLow(String gpio) {
        execRootCmd("echo 0 > /sys/class/gpio/gpio" + gpio + "/value");
}
/**
 * Get current GPIO level
 * @param gpio GPIO pin number
 * **/
public String getGpioValue(String gpio) {
        return this.getGpioString("/sys/class/gpio/gpio" + gpio + "/value");
}
public static String execRootCmd(String cmd) {
        String result = "";
        DataOutputStream dos = null;
        DataInputStream dis = null;
        try {
                Process p = Runtime.getRuntime().exec("su");
                dos = new DataOutputStream(p.getOutputStream());
                dis = new DataInputStream(p.getInputStream());
                dos.writeBytes(cmd + "\n");
                dos.flush();
                dos.writeBytes("exit\n");
                dos.flush();
                for(String line = null; (line = dis.readLine()) != null; result =
                        result + line) {
                }
                p.waitFor();
        } catch (Exception var18) {
                var18.printStackTrace();
        } finally {
                if (dos != null) {
                        try {
                                dos.close();
                        } catch (IOException var17) {
                                var17.printStackTrace();
                        }
                }
                if (dis != null) {
                        try {
                                dis.close();
                        } catch (IOException var16) {
                                var16.printStackTrace();
                        }
```

```
            }
        }
        return result;
    }
    private String getGpioString(String path) {
        String defString = "0";
        try {
            BufferedReader reader = new BufferedReader(new FileReader(path));
            defString = reader.readLine();
        } catch (IOException var4) {
            var4.printStackTrace();
        }
        return defString;
    }
}
```

Usage:

```
// Get GPIO
Gpio gpio = new Gpio();
// GPIO pin number - please consult your sales representative for the correct GPIO number
gpio.getGpio("165");
// Set GPIO 165 to input mode
gpio.setGpioIn("165");
// Get the value of GPIO 165 (returns 0 or 1)
gpio.getGpioValue("165");
// Set GPIO 165 to output mode
gpio.setGpioOut("165");
// Set GPIO 165 to high level
gpio.setGpioHigh("165");
// Set GPIO 165 to low level
gpio.setGpioLow("165");
```

Note:

GPIO pin numbers need to be calculated. Please consult your sales representative for the correct GPIO pin numbers. Here is the calculation method:

The port value for each GPIO is calculated using a lookup table and formula.

We know that GPIOs are typically divided into groups, such as GPIO1, GPIO2, GPIO3...

```
For the RK3128 series, the calculation formula is: num = 32 * GPIO_X + PORT

GPIO_X corresponds to the GPIO group number, and the relationship is as follows:

GPIO0 --> 0
```

GPIO1 --> 1

GPIO2 --> 2

...

The values for PORT are as shown in the table below:

[Table with GPIO_PORT values]

For example, if we want to control GPIO2_D4 pin, its port number is:

2 * 32 + 28 = 92

If we want to control GPIO0_A1 pin, its port number is:

0 * 32 + 1 = 1

For the RK3288 series with Android 5.1, calculate as follows:

Convert the pin definition into the following form: GPIOX_YZ

We have 3 parameters X, Y, Z; Y exists in the pin definition in the form of a letter and should be replaced with a number as follows:

A = 0

B = 1

C = 2

D = 3

Formula: (X32) + (Y8) + Z

Examples:

GPIO1_A7 (132) + (08) + 7 = 39

GPIO1_B1 (132) + (18) + 1 = 41

GPIO2_C4 (232) + (28) + 4 = 84

GPIO3_D0 (332) + (38) + 0 = 120

GPIO3_D6 (332) + (38) + 6 = 126

For the RK3288 series with Android 7.1, calculate as follows:

When X=0 (pin definitions start from GPIO0):

Formula: (Y8) + Z

Examples:

GPIO0_A7 (08) + 7 = 7

GPIO0_B4 (18) + 4 = 12

GPIO0_C2 (28) + 2 = 18

When X>0 (GPIOs start in a non-GPIO0 form):

Formula: 24 + ((X-1)32) + (Y8) + Z

Examples:

GPIO1_A7 24 + ((1-1)32) + (08) + 7 = 31

GPIO1_B1 24 + ((1-1)32) + (18) + 1 = 33

GPIO2_C4 24 + ((2-1)32) + (28) + 4 = 76

GPIO3_D0 24 + ((3-1)32) + (38) + 0 = 112

GPIO3_D6 24 + ((3-1)32) + (38) + 6 = 118

4.6 Functional Code Examples

4.6.1 4G Issue Troubleshooting

Issue: No SIM Card Detected

1. Go to Settings > About Device, and check the baseband version. If there is no baseband version, please try the following steps after powering off: Remove the 4G module, wipe the gold contacts, and then reinsert it into the MINI-PCIE slot. If there is still no baseband version, please take a photo of the system version and the module's back and send it to your sales representative.

2. Go to Settings > About Device, and check the baseband version. If there is a baseband version, please follow these steps after powering off: Gently wipe the SIM card, lightly clean the gold contacts on the SIM card, place the SIM card into the slot, and gently press it a few times. Power on the device and check if the SIM card is detected. If there is still no SIM card detected, please take a photo of the system version and the module's back and send it to your sales representative.

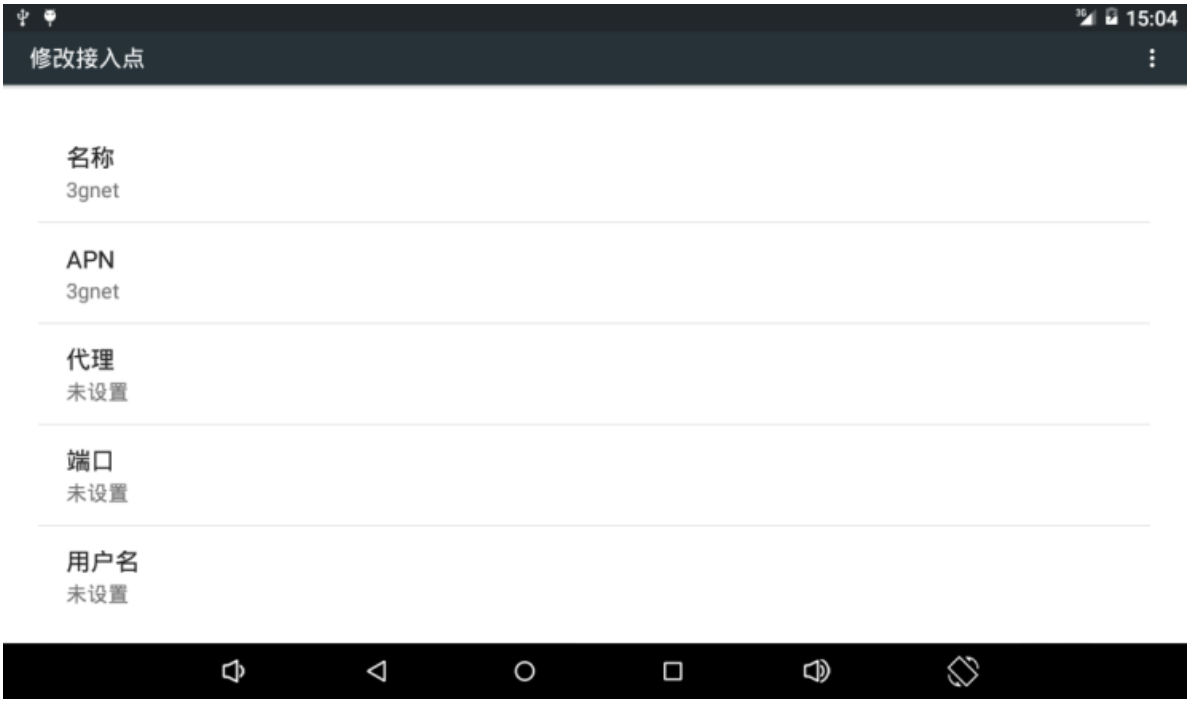
Issue: Signal Detected, but Unable to Access the Internet

Go to Settings > More > Mobile Networks > Access Point Names (APN). Click the plus (+)

button in the top right corner to add an APN.

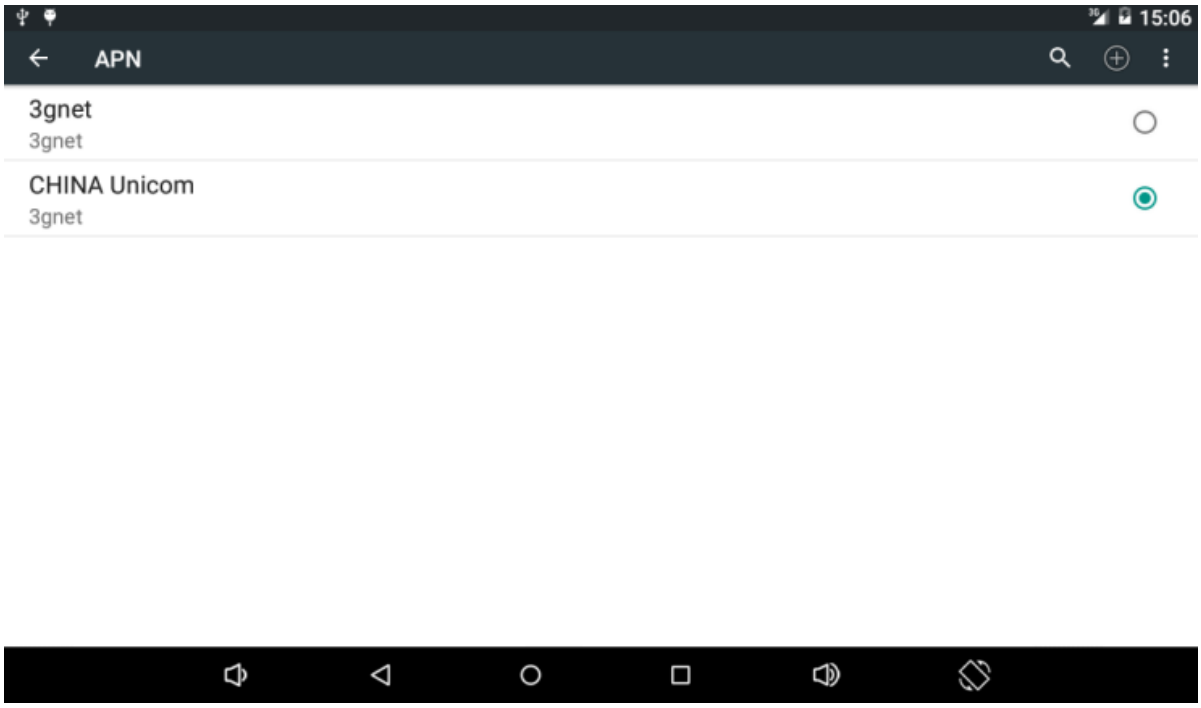
When dialing with an APN, you need to include the APN of your mobile network operator. Different operators have different APNs. Here are the APNs for various operators:

- China Mobile: cmnet
- China Unicom: 3gnet
- China Telecom: ctnet



For the China Unicom SIM card as an example, enter the following settings: Name should be set to "3gnet," and APN should also be set to "3gnet." Click the button in the upper right corner and save the settings.

Select the newly created APN you just added. Wait for a moment, and the signal in the status bar will display "4G" or "3G" based on the signal strength.



4.6.2 Invoke a shell command

```
public void execShell(String cmd) {
    try {
        //Permission setup
        Process p = Runtime.getRuntime().exec("su");
         //Get output stream
        OutputStream outputStream = p.getOutputStream();
         DataOutputStream dataOutputStream = new DataOutputStream(outputStream);
         //Write the command
        dataOutputStream.writeBytes(cmd);
         //Submit the command
        dataOutputStream.flush();
         //Close stream operations
        dataOutputStream.close();
         outputStream.close();

    } catch (Throwable t) {
        t.printStackTrace();
    }
}
```

Usage Example

```
Add the execShell method to the required JAVA class, and call the following methods to turn
Ethernet on and off:

execShell("busybox ifconfig eth0 down");// Turn off Ethernet

execShell("busybox ifconfig eth0 up");// Turn on Ethernet

Modify I2C0 device permissions:

execShell("chmod 777 /dev/i2c-0");
```

4.6.3 HDMI dual-screen display

To configure HDMI dual-screen display, you can use the execShell method from section 4.6.2 to control HDMI output as follows:

```
execShell("on > /sys/class/drm/card0-HDMI-A-1/status");// Turn on HDMI output:
execShell("off > /sys/class/drm/card0-HDMI-A-1/status");// Turn off HDMI output:
execShell("cat /sys/class/drm/card0-HDMI-A-1/status");// Check HDMI status (returns "connected" when
open and "disconnected" when closed):

/**

 * Android 7.1 HDMI Dual Screen Display

 * If dual-screen display is not enabled, the HDMI display may appear in a small area in the top left corner
instead of full-screen.

 * Set HDMI Dual Screen Mode

 **/
public void setHDMIDualScreenMode() {
    if (Build.VERSION.SDK_INT >= 25) {
        boolean state = Settings.System.getInt(this.getContentResolver(), "dual_screen_mode", 0) == 1;
        if (state) {
            Log.d("HDMI", " Android 7.1 Dual Screen Display is already enabled");
        } else {
            Settings.System.putInt(this.getContentResolver(), "dual_screen_mode", 1);
            Log.d("HDMI", " Enabling Android 7.1 Dual Screen Display");
            Toast.makeText(this, " Dual-screen display was detected as not enabled and has been
enabled for you. Please restart the application. ", Toast.LENGTH_SHORT).show();
            finish();
        }
    }
}
```

4.6.4 Configuring and Installing ADB (Android Debug Bridge)

ADB, which stands for Android Debug Bridge, serves as a debugging bridge and is a client-server program. The client is your computer used for operations, and the server is the Android device itself. ADB is also a tool within the Android SDK, allowing you to directly manage Android emulators or real Android devices.

1. Download ADB

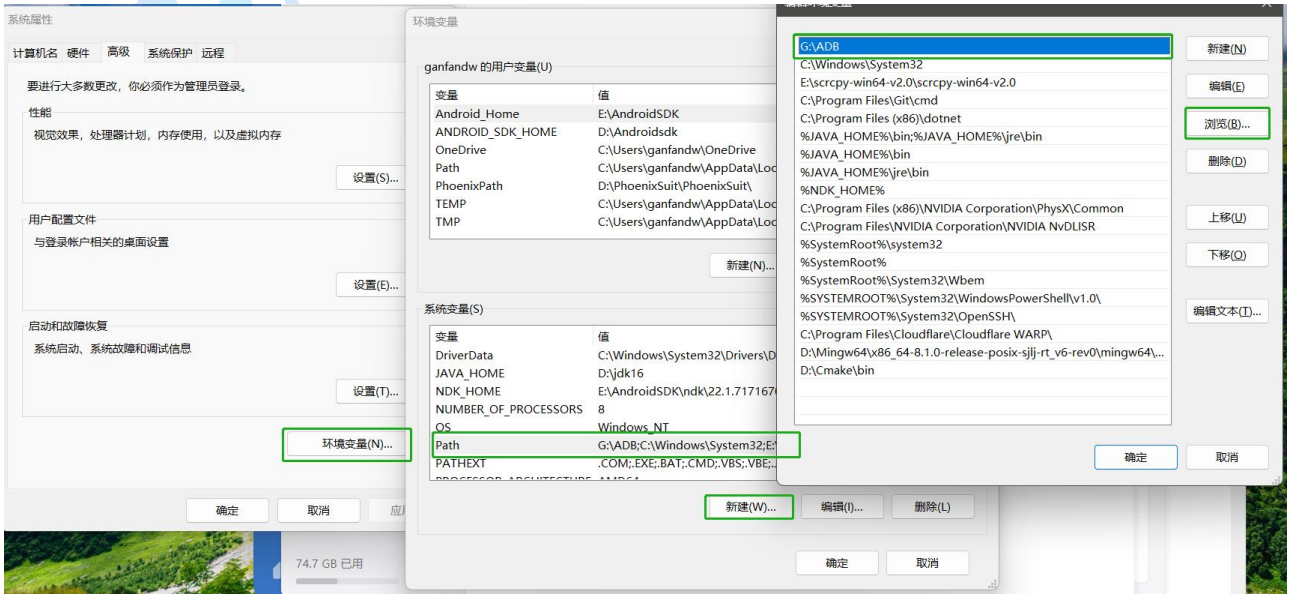
For Windows：https://dl.google.com/android/repository/platform-tools-latest-windows.zip

For Mac：https://dl.google.com/android/repository/platform-tools-latest-windows.zip

For Linux：https://dl.google.com/android/repository/platform-tools-latest-linux.zip

2. Configure Environment Variables

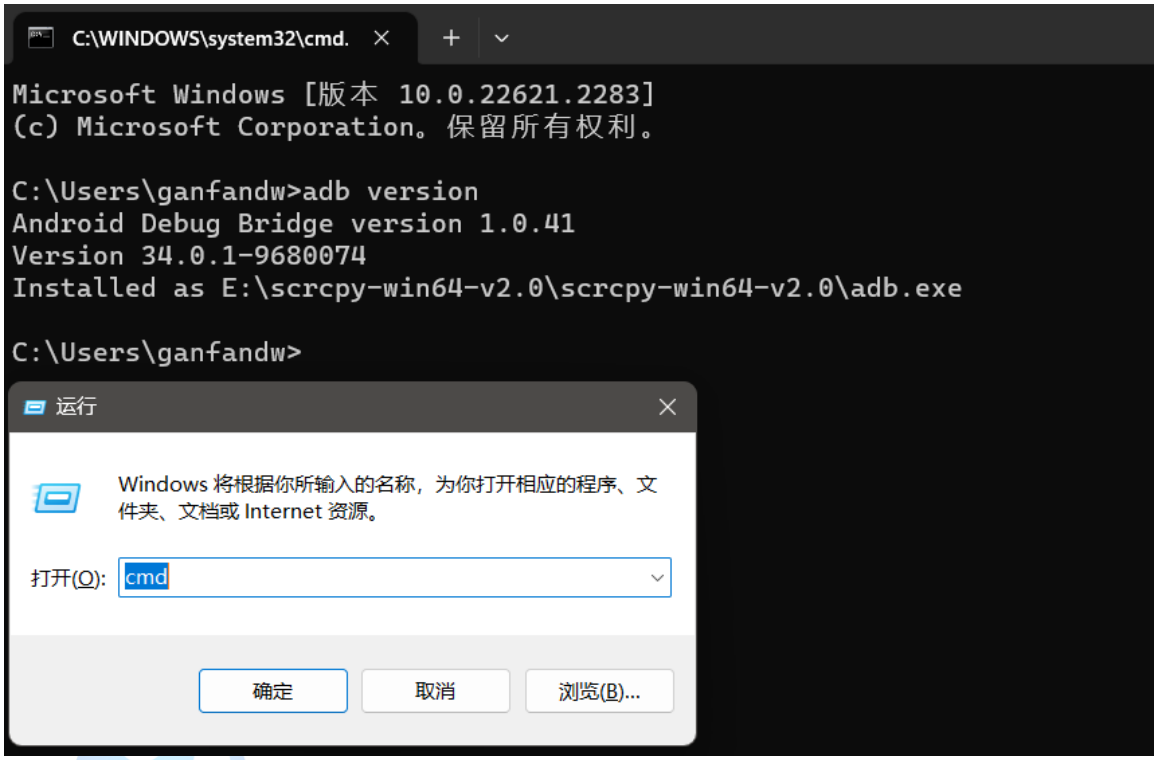
- Extract the downloaded files and add the path to the extracted folder to your system's PATH variable.
- Navigate to System Properties -> Environment Variables -> Select the "Path" variable -> Edit -> Add or browse to select the path where ADB was extracted.



3. Check ADB Installation

- On Windows, use the shortcut Win+R to open the Run dialog, then type "cmd" to open the command prompt.

- Type "adb version" to verify the ADB installation. If it displays the ADB version, it means the installation was successful.。



4. Connect Your Device

Connect your Android device to your computer using a USB data cable, open the developer options, and enable USB debugging. Then, open a CMD command prompt and type "adb devices" to check if any devices are recognized.

Note:

Here's how to enable developer options: Go to "Settings" -> "About phone" -> Tap the "Build number" multiple times (usually about ten times) until you see a message indicating that developer options have

been enabled. Then, go back to the main settings menu. Now, you should be able to see a new menu item called "Developer options." Click on it. In the "Developer options" menu, enable the "USB debugging" option.

4.6.5 Capturing logs through ADB (Android Debug Bridge)

1. Capture general logs

```
adb logcat
```

```
C:\Users\ganfandw>adb logcat
--------- beginning of main
I/installd(    0): installd firing up
W/auditd  (  119): type=2000 audit(0.0:1): initialized
I/auditd  (  119): type=1403 audit(0.0:2): policy loaded
--------- beginning of system
E/DrmService(    0): ----------------running drmservice-
E/DrmService(    0): ----------serianno =2V8QOJAZ3B
E/DrmService(    0): auto generate serialno,serialno = 2
E/DrmService(    0): rknand_sys_storage open fail
```

2. Capture kernel logs (Dmesg/kernel logs)

```
adb shell dmesg or adb shell " cat /proc/kmsg "
```

```
C:\Users\ganfandw>adb shell dmesg
<6>[    0.000000] Booting Linux on physical CPU 0xf00
<6>[    0.000000] Initializing cgroup subsys cpu
<6>[    0.000000] Initializing cgroup subsys cpuacct
<5>[    0.000000] Linux version 3.10.0 (sl410@sl410-OEM) (gcc version 4.6.x-goog
PREEMPT Wed Dec 22 13:40:46 CST 2021
<4>[    0.000000] CPU: ARMv7 Processor [410fc075] revision 5 (ARMv7), cr=10c5387
<4>[    0.000000] CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruc
<6>[    0.000000] Machine: Rockchip RK3128, model: rockchip,rk3128
<6>[    0.000000] rockchip_uboot_logo_setup: mem: 2000000@9dc00000, offset:0
<6>[    0.000000] rockchip_uboot_mem_reserve: reserve 2000000@9dc00000 for uboot
<4>[    0.000000] rockchip_ion_reserve
<6>[    0.000000] ion heap(cma): base(0) size(800000) align(0)
<6>[    0.000000] ion heap(vmalloc): base(0) size(0) align(0)
<6>[    0.000000] cma: CMA: reserved 8 MiB at 9d400000
<6>[    0.000000] ion_reserve: cma reserved base 9d400000 size 8388608
```

3. Capture ANR (Application Not Responding) logs

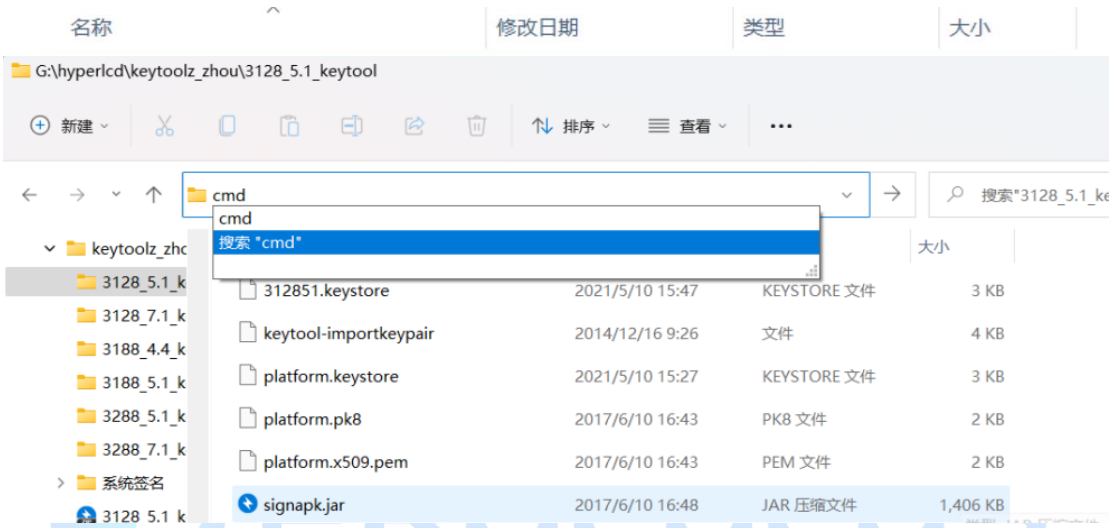
```
adb shell cat /data/anr/traces.txt
```

4.6.6 Adding System Signature to an Application

Sometimes, our application requires a system-level signature to access certain methods, and our team has the necessary files ready for signing. Please contact your sales representative

for the files required. Manually adding a system signature is different from re-signing an Android APK. Re-signing involves modifying the signature file of a previously signed APK, and certain algorithms may be tied to the package name and signature, making it unusable if changed. Manually adding a system signature is the process of adding a system signature to an unsigned APK generated through Android Studio's build process. Here's the step-by-step process:

1. Download the pre-packaged signature tool and security files.



2. Perform the system signature operation. Navigate to the folder containing the files, then type CMD in the address bar and press Enter to open the Command Prompt.

3. Enter the following command in the Command Prompt

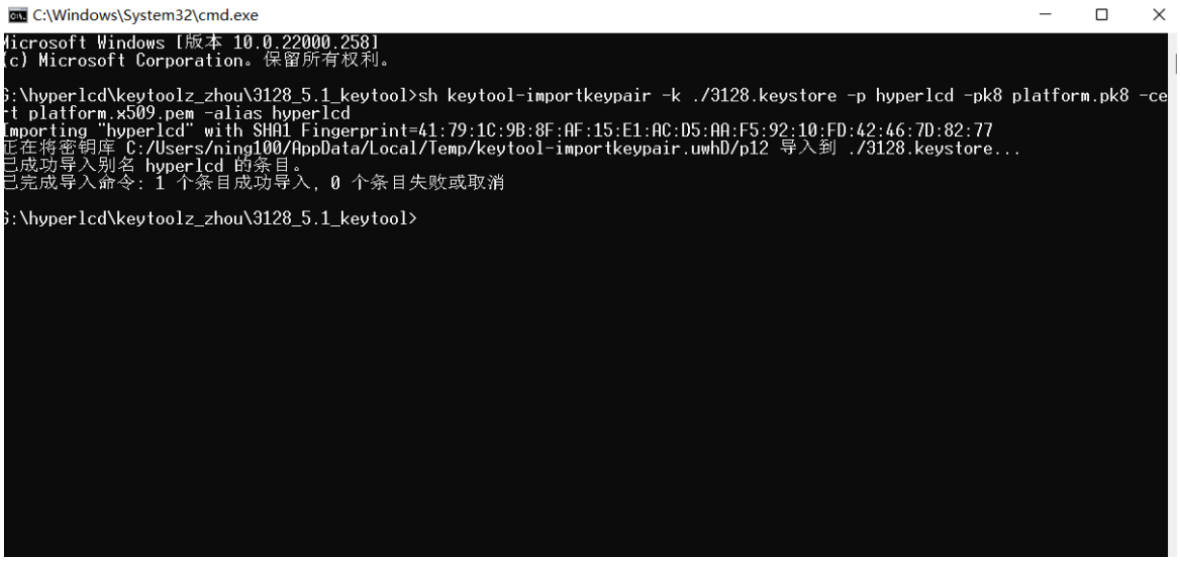
```
java -jar signapk.jar platform.x509.pem platform.pk8 in.apk out.apk
```

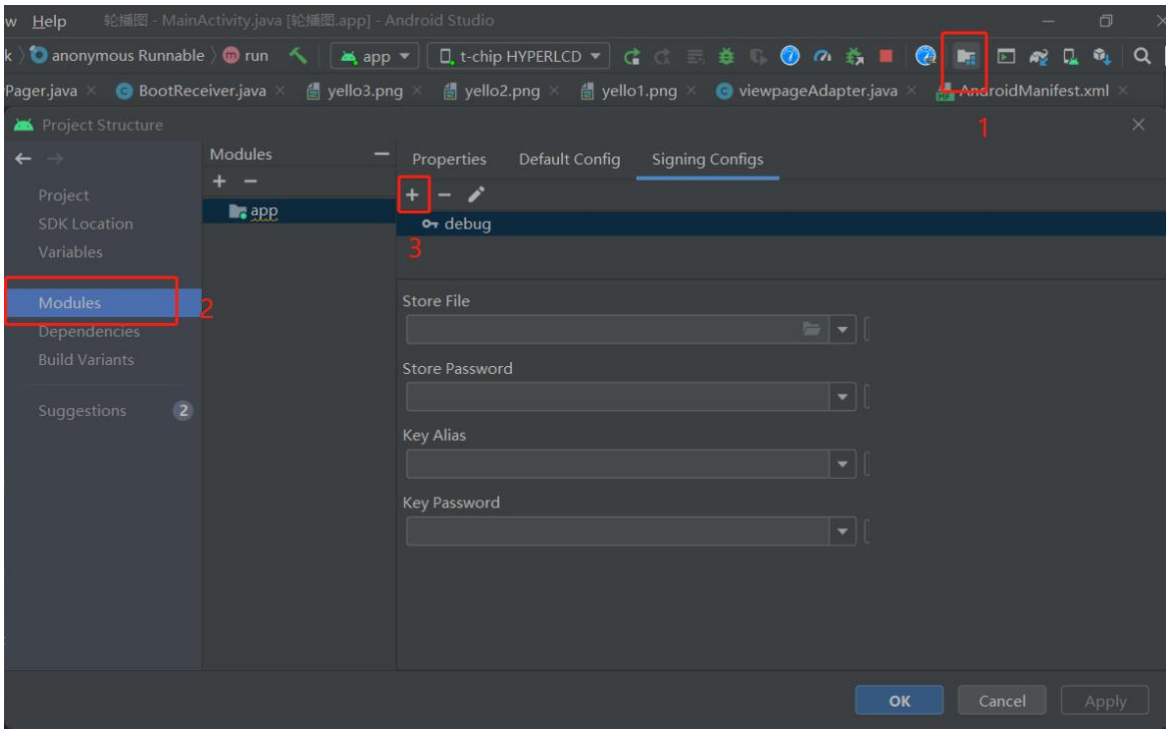
Where in.apk is the unsigned application generated by Android Studio, which should be placed in the signing folder, and out.apk is the application after signing.

A more recommended alternative method is as follows, enter the following command:

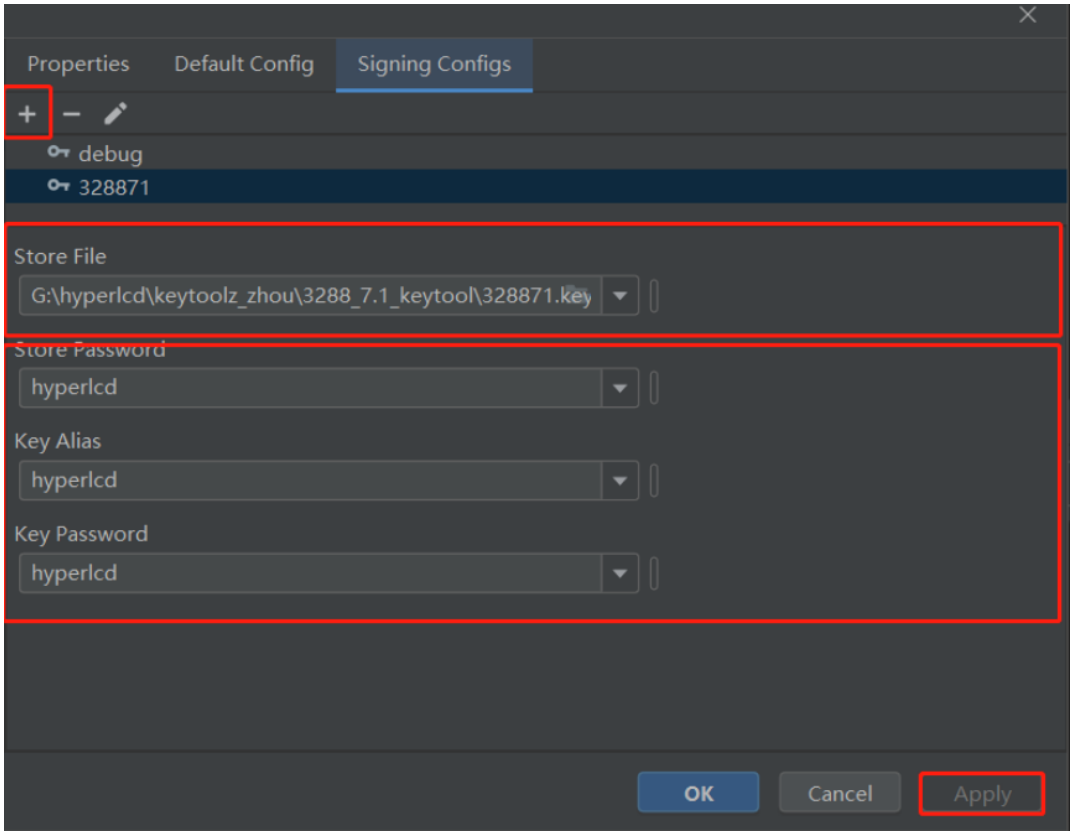
```
sh keytool-importkeypair -k ./3128.keystore -p hyperlcd -pk8 platform.pk8 -cert

platform.x509.pem -alias hyperlcd

_____________________________________________________________________________

k: Generated keystore file

pk8: The platform.pk8 file to import

cert: The platform.x509.pem file to import

alias Specifies the alias of the generated keystore
```

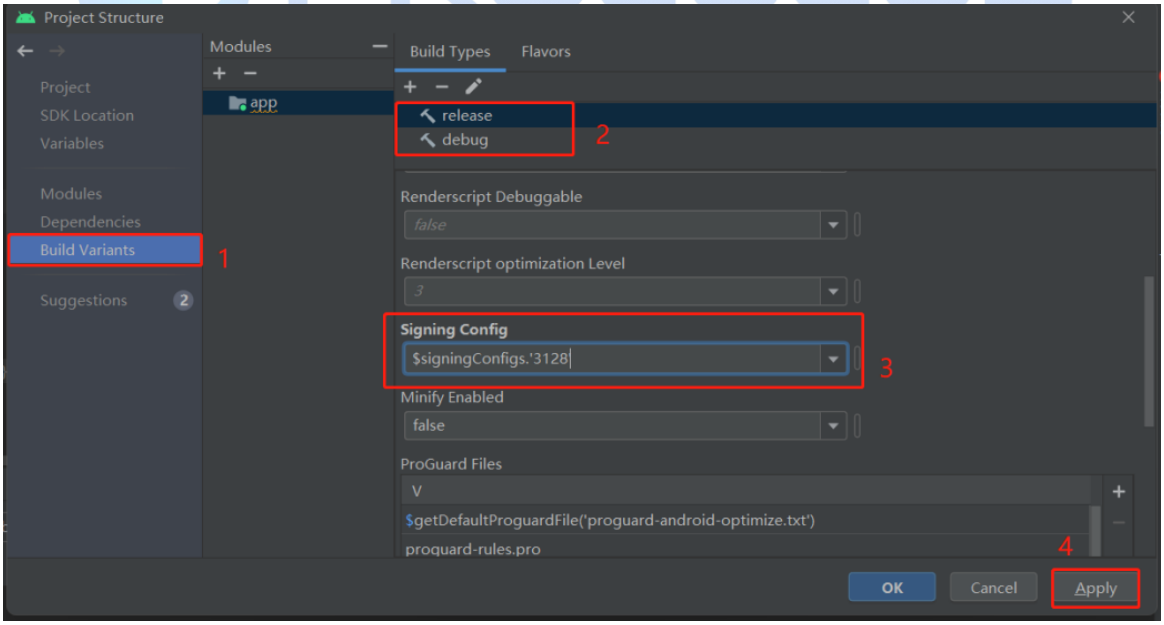
p: password of the generated keystore file





After adding, select the default signature for debug and release





To add the system signature to the AndroidManifest.xml of your app

```
android:sharedUserId="android.uid.system"
```

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:sharedUserId="android.uid.system"
    package="com.hyperlcd.usertest">
    <meta-data android:name="android.webkit.WebView.EnableSafeBrowsing"
        android:value="true"/>
```

⚠ Note: This method can resolve the issue of not being able to install the APK compiled in Android Studio, which results in the error "INSTALL_FAILED_SHARED_USER_INCOMPATIBLE." If the issue persists, consider downloading the system signature corresponding to the system version again.

4.6.7 Hide the system navigation bar

Create a HideMenu class, use the HideMenus method, pass true to hide, pass false to show. Can be used with the lifecycle to exit and display the navigation bar, hide the navigation bar when entering the application.

```
public class HideMenu {
    //安卓7.1 隐藏导航栏的广播
    private final static String show = "am broadcast -a action.SHOW_STATUSBAR";
    //安卓7.1 显示导航栏的广播
    private final static String hide = "am broadcast -a action.HIDE_STATUSBAR";
    //Execute shell commands
    public void execShell(String cmd){
        try{
            // Permission setup
            Process p = Runtime.getRuntime().exec("su");
            //Get output stream
            OutputStream outputStream = p.getOutputStream();
            DataOutputStream dataOutputStream = new DataOutputStream(outputStream);
            //Write the command
            dataOutputStream.writeBytes(cmd);
            //Submit the command
            dataOutputStream.flush();
            //Close stream operations
            dataOutputStream.close();
            outputStream.close();
        }
```

```
            catch(Throwable t)
            {
                t.printStackTrace();
            }
        }

    // Method to hide or show the navigation bar
        public void HideMenus(boolean flag, Context context){
            //Android 7.0 and above
            if (Build.VERSION.SDK_INT > 11 && Build.VERSION.SDK_INT <= 23) {
                if (flag){
                    Intent intent = new Intent();
                    intent.setAction("marvsmart_bar");
                    intent.putExtra("marvsmart_swich", false);
                    intent.putExtra("marvsmart_toast", "");
                    context.sendBroadcast(intent);
                }else {
                    Intent intent = new Intent();
                    intent.setAction("marvsmart_bar");
                    intent.putExtra("marvsmart_swich", true);
                    intent.putExtra("marvsmart_toast", "");
                    context.sendBroadcast(intent);
                }
//For Android versions below 7.0 (API level 24)
            } else if (Build.VERSION.SDK_INT > 23) {
                Log.d("menu",">21");
                if(flag){
                    execShell(hide);
                }else {
                    execShell(show);
                }
            }
        }
}
```

Usage:

```
// Hide the status bar
HideMenus(true,context);
// Show the status bar
HideMenus(false,context);
```

You can also use the following method after signing the application with system privileges:

```
Settings.System.putInt(this.getContentResolver(),"always_hide_bar",0);// 0 for showing, 1 for hiding
sendBroadcast(new Intent("action.ALWAYS_HIDE_STATUSBAR_CHENAGE"));
```

4.6.8 Modify system time

To modify the system time, a system signature is required. Set the android:sharedUserId="android.uid.system" attribute in the root node of AndroidManifest.xml. After compilation, it grants permission to modify system functions.

```
android:sharedUserId="android.uid.system"

<uses-permission android:name="android.permission.SET_TIME"
    tools:ignore="ProtectedPermissions" />
<uses-permission android:name="android.permission.WRITE_SETTINGS"
    tools:ignore="ProtectedPermissions" />
<uses-permission android:name="android.permission.WRITE_SECURE_SETTINGS"
    tools:ignore="ProtectedPermissions" />
<uses-permission android:name="android.permission.SET_TIME_ZONE"
    tools:ignore="ProtectedPermissions" />
```

Please elucidate the means to alter the system time through code modification:

```
/**

Establish the system's temporal parameters

@param year The year, exemplified as '2023'

@param month The month, demonstrated as '3'

@param day The day, indicated as '1'

@param hour The hour, represented as '1'

@param minute The minute, delineated as '1'

@param second The second, typified as '1'

@throws Settings.SettingNotFoundException */
public void setNowTime(int year ,int month ,int day,int hour ,int minute,int second) throws
        Settings.SettingNotFoundException {
    //Cease the automatic determination of dates and times.
    int autoTime = Settings.Global.getInt(mContext.getContentResolver(),
            Settings.Global.AUTO_TIME);
    if (autoTime == 1) {
        Settings.Global.putInt(mContext.getContentResolver(),
                Settings.Global.AUTO_TIME, 0);
    }
    // Discontinue the automatic determination of time zones.
    int autoZoneEnable = Settings.Global.getInt(mContext.getContentResolver(),
            Settings.Global.AUTO_TIME_ZONE);
    if (autoZoneEnable == 1) {
```

```
            Settings.Global.putInt(mContext.getContentResolver(),
                    Settings.Global.AUTO_TIME_ZONE, 0);
        }
        // Establish the employment of the 24-hour format.
        if (!DateFormat.is24HourFormat(mContext)) {
            Settings.System.putString(mContext.getContentResolver(),
                    Settings.System.TIME_12_24, "24");
        }
        Calendar c = Calendar.getInstance();
        c.set(Calendar.YEAR, year);
        c.set(Calendar.MONTH, month);
        c.set(Calendar.DAY_OF_MONTH, day);
        c.set(Calendar.HOUR_OF_DAY, hour);
        c.set(Calendar.MINUTE, minute);
        c.set(Calendar.SECOND, second);
        long when = c.getTimeInMillis();
        ((AlarmManager) mContext.getSystemService(Context.ALARM_SERVICE)).setTime(when);
}

使用方法：

setNowTime(Year,Month,Day,Hour,Minute,0)
```

4.6.9 Alter the screen orientation

The following section will elucidate two approaches to alter screen orientation: manipulating screen orientation through code and adjusting system configurations to define the screen orientation.

1. Altering Screen Orientation Through Code

```
//Execute shell commands
    public void execShell(String cmd){
        try{
            // Permission setup
              Process p = Runtime.getRuntime().exec("su");
             //Get output stream
              OutputStream outputStream = p.getOutputStream();
              DataOutputStream dataOutputStream = new DataOutputStream(outputStream);
              //Write the command
              dataOutputStream.writeBytes(cmd);
              //Submit the command
               dataOutputStream.flush();
               //Close stream operations
               dataOutputStream.close();
```

```
                outputStream.close();
            }
            catch(Throwable t)
            {
                t.printStackTrace();
            }
        }
```

Integrate four buttons and establish their listeners.

Define four buttons for click-testing: Button 0 for 'Normal', Button 1 for 'Landscape', Button 2 for 'Reverse Portrait', and Button 3 for 'Reverse Landscape'.

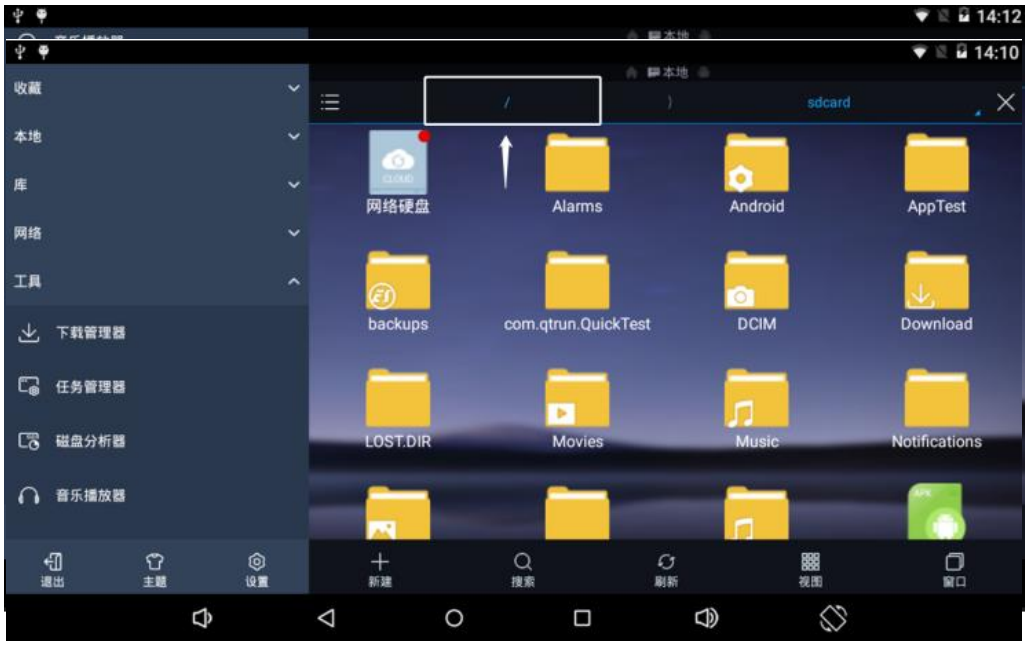
```
@Override
public void onClick(View v) {
    switch (v.getId()){
        case R.id.bt_set_0:
            execShell("settings put system accelerometer_rotation 0");
            execShell("settings put system user_rotation "+0);
            break;
        case R.id.bt_set_1:
            execShell("settings put system accelerometer_rotation 0");
            execShell("settings put system user_rotation "+1);
            break;
        case R.id.bt_set_2:
            execShell("settings put system accelerometer_rotation 0");
            execShell("settings put system user_rotation "+2);
            break;
        case R.id.bt_set_3:
            execShell("settings put system accelerometer_rotation 0");
            execShell("settings put system user_rotation "+3);
            break;
    }
}
```

2. Screen Orientation Modification via ES File Manager

⚠ Note:

This method carries risks! Please refrain from altering other data, as it may result in system instability.

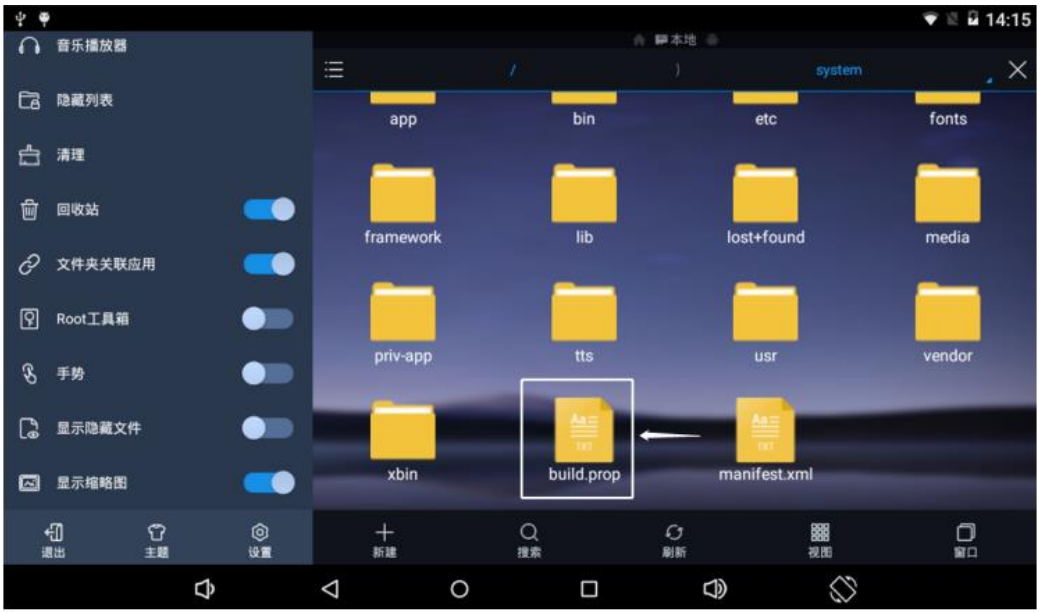
Download ES File Manager,Access the ROOT Toolbox

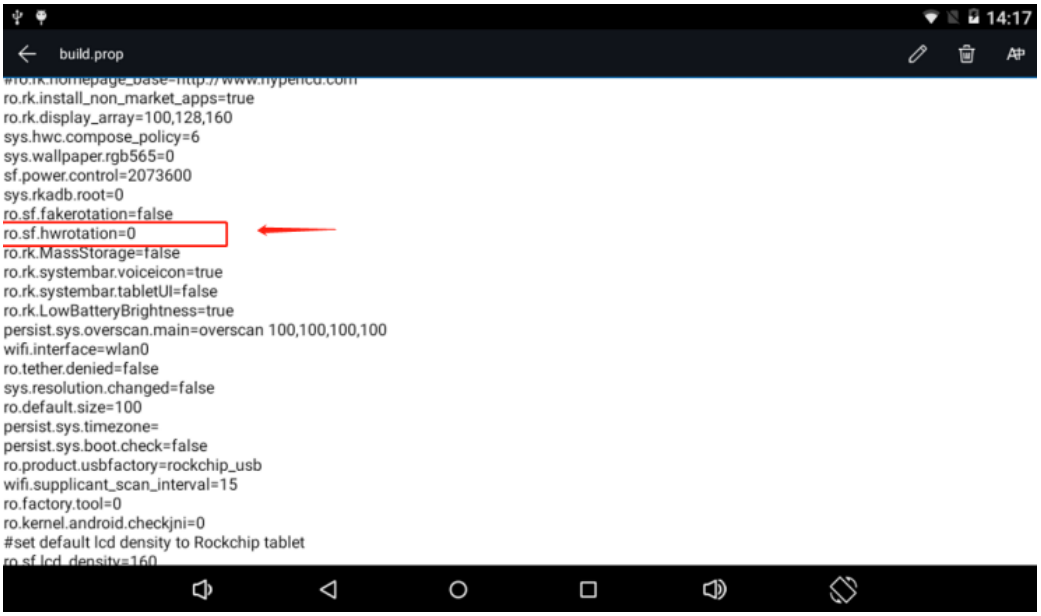


Click to access the root directory and locate the "System" folder.

Find the build.prop file, open it, select 'edit,'

locate 'ro.sf.hwrotation=0,' and change it to 'ro.sf.hwrotation=90.' After saving, please restart.





4.6.10 Set the application as the desktop

Set the application as the main desktop by adding the following parameters to the intent filter of the app's home page in the AndroidManifest.xml file. Afterward, restart the app and select it as the launcher.

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
    <category android:name="android.intent.category.HOME" />
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.MONKEY" />
</intent-filter>
```

```
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
        <category android:name="android.intent.category.HOME" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.MONKEY" />
    </intent-filter>
</activity>
```

4.6.11 Set up static Ethernet connection

This method is applicable for setting up static Ethernet on Android 5.1 to Android 7.1. First, import the EthernetManager.jar file from the folder, then create the NetUtils class.

```
public class NetUtils {
    /*
    * convert subMask string to prefix length
    */
    public static int maskStr2InetMask(String maskStr) {
        StringBuffer sb ;
        String str;
        int inetmask = 0;
        int count = 0;
       /*
        * check the subMask format
        */
        Pattern pattern = Pattern.compile("(^((\\d|[01]?\\d\\d|2[0-4]\\d|25[0-5])\\.){3}(\\d|[01]?\\d\\d|2[0-
4]\\d|25[0-5])$)|^(\\d|[1-2]\\d|3[0-2])$");
        if (pattern.matcher(maskStr).matches() == false) {
            Log.e("log","subMask is error");
            return 0;
        }
```

```
        String[] ipSegment = maskStr.split("\\.");
        for(int n =0; n<ipSegment.length;n++) {
            sb = new StringBuffer(Integer.toBinaryString(Integer.parseInt(ipSegment[n])));
            str = sb.reverse().toString();
            count=0;
            for(int i=0; i<str.length();i++) {
                i=str.indexOf("1",i);
                if(i==-1)
                    break;
                count++;
            }
            inetmask+=count;
        }
        return inetmask;
    }


    public static Inet4Address getIPv4Address(String text) {
        try {
            return (Inet4Address) NetworkUtils.numericToInetAddress(text);
        } catch (IllegalArgumentException|ClassCastException e) {
            return null;
        }
    }

}
```

以下是操作静态以太网的变量

```
private StaticIpConfiguration mStaticIpConfiguration;
private IpConfiguration mIpConfiguration;
private EthernetManager mEthManager;
private static String mEthIpAddress = "192.168.88.154"; //IP
private static String mEthNetmask = "255.255.255.0"; // NETMASK
private static String mEthGateway = "192.168.88.1"; //GATEWAY
private static String mEthdns1 = "8.8.8.8"; // DNS1
private static String mEthdns2 = "8.8.4.4"; // DNS2

/**



Set up static Ethernet

@param ip IP address

@param gateway Gateway
```

```
@param netmask Subnet mask

@param dns1 DNS

@param dns2 DNS

*/
public void setEthernetStaticIp(String ip, String gateway, String netmask, String dns1, String dns2) {
    mStaticIpConfiguration = new StaticIpConfiguration();
    /*
     * get ip address, netmask,dns ,gw etc.
     */
    Inet4Address inetAddr = getIPv4Address(ip);
    int prefixLength = maskStr2InetMask(netmask);
    InetAddress gatewayAddr = getIPv4Address(gateway);
    InetAddress dnsAddr1 = getIPv4Address(dns1);
    InetAddress dnsAddr2 = getIPv4Address(dns2);
    if (inetAddr.getAddress().toString().isEmpty() || prefixLength == 0 ||
            gatewayAddr.toString().isEmpty()
            || dnsAddr1.toString().isEmpty() || dnsAddr2.toString().isEmpty()) {
        return;
    }
    Class<?> clazz = null;
    try {
        clazz = Class.forName("android.net.LinkAddress");
        Class[] cl = new Class[]{InetAddress.class, int.class};
        Constructor cons = null;
        //取得所有构造函数
        cons = clazz.getConstructor(cl);
        //给传入参数赋初值
        Object[] x = {inetAddr, prefixLength};
        mStaticIpConfiguration.ipAddress = (LinkAddress) cons.newInstance(x);
        mStaticIpConfiguration.gateway = gatewayAddr;
        mStaticIpConfiguration.dnsServers.add(dnsAddr1);
        mStaticIpConfiguration.dnsServers.add(dnsAddr2);
        mIpConfiguration = new
                IpConfiguration(IpConfiguration.IpAssignment.STATIC,
                IpConfiguration.ProxySettings.NONE, mStaticIpConfiguration,
                null);
        mEthManager.setConfiguration(mIpConfiguration);
    } catch (Exception e) {
        // TODO: handle exception
        ILogUtils.e(e.toString());
    }

}
```

```
/**
 * Set Ethernet to DHCP mode

 */
public void setEthernetDhcp() {
    mIpConfiguration = new
            IpConfiguration(IpConfiguration.IpAssignment.DHCP,
            IpConfiguration.ProxySettings.NONE, null,
            null);
    mEthManager.setConfiguration(mIpConfiguration);
}
```

Add permissions in the AndroidManifest.xml file:

```
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
<uses-permission android:name="android.permission.CONNECTIVITY_INTERNAL" />
```

Finally, sign the application with a system signature, please refer to section 4.6.6 of this document.

Install the software, connect via Ethernet, and check if the Ethernet has a static IP address with the following data:

```
"192.168.88.154"; //IP

"255.255.255.0"; // NETMASK

"192.168.88.1"; //GATEWAY

"8.8.8.8"; // DNS1

"8.8.4.4"; // DNS2
```

4.6.12 Set up static WiFi

This method is applicable for setting up static WiFi on Android 5.1 to Android 7.1. First, import the EthernetManager.jar file from the folder, and then create the NetUtils class.

```
public class NetUtils {
    /*
    * convert subMask string to prefix length
    */
    public static int maskStr2InetMask(String maskStr) {
        StringBuffer sb ;
        String str;
        int inetmask = 0;
        int count = 0;
```

```
        /*
         * check the subMask format
         */
         Pattern pattern = Pattern.compile("(^((\\d|[01]?\\d\\d|2[0-4]\\d|25[0-5])\\.){3}(\\d|[01]?\\d\\d|2[0-
4]\\d|25[0-5])$)|^(\\d|[1-2]\\d|3[0-2])$");
         if (pattern.matcher(maskStr).matches() == false) {
             Log.e("log","subMask is error");
             return 0;
         }

         String[] ipSegment = maskStr.split("\\.");
         for(int n =0; n<ipSegment.length;n++) {
             sb = new StringBuffer(Integer.toBinaryString(Integer.parseInt(ipSegment[n])));
             str = sb.reverse().toString();
             count=0;
             for(int i=0; i<str.length();i++) {
                 i=str.indexOf("1",i);
                 if(i==-1)
                     break;
                 count++;
             }
             inetmask+=count;
         }
         return inetmask;
     }


     public static Inet4Address getIPv4Address(String text) {
         try {
             return (Inet4Address) NetworkUtils.numericToInetAddress(text);
         } catch (IllegalArgumentException|ClassCastException e) {
             return null;
         }
     }

}
```

Here are the variables for configuring static WiFi:

```
//Define the variables you need and modify them
public String STATIC_IP = "192.8.8.8";
public String STATIC_NETMASK = "255.0.0.0";
public String STATIC_GATEWAY = "255.255.255.225";
public String STATIC_DNS1 = "192.8.8.8";
public String STATIC_DNS2 = "192.8.8.8";
```

```
private static android.net.wifi.WifiManager mWifiManager;

private static Context mContext;
```

Here is the method for configuring static WiFi:

```
/**
 * WiFi control constructor, no callback listener needed
 */
public IWifiManager(Context context) {
    mContext = context;
    mWifiManager = (android.net.wifi.WifiManager) mContext.getSystemService(Context.WIFI_SERVICE);
}


/**
Connected to Wi-Fi and configured STATIC Wifi settings via IP address, gateway, and DNS.

@param ip Device IP address

@param gateway Gateway

@param netmask Subnet mask

@param dns1 DNS1

@param dns2 DNS2

@return Whether the configuration was successful
 */
public boolean setStaticWifi(String ip, String gateway, String netmask, String dns1, String dns2) {
    List<WifiConfiguration> configuredNetworks = getConfiguredNetworks();
    WifiConfiguration wifiConfig = null;
    WifiInfo connectionInfo = mWifiManager.getConnectionInfo();    //得到连接的 wifi 网络
    for (WifiConfiguration conf : configuredNetworks) {
        if (conf.networkId == connectionInfo.getNetworkId()) {
            wifiConfig = conf;
            break;
        }
    }
    return setStaticWifi(wifiConfig, ip, gateway, netmask, dns1, dns2);
}


private boolean setStaticWifi(WifiConfiguration config, String ip, String gateway, String netmask, String dns1,
String dns2) {
    try {
        WifiConfiguration wifiConfig = config;
        Inet4Address inetAddr = getIPv4Address(ip);
```

```
        int prefixLength = maskStr2InetMask(netmask);
        InetAddress gatewayAddr = getIPv4Address(gateway);
        InetAddress dnsAddr1 = getIPv4Address(dns1);
        InetAddress dnsAddr2 = getIPv4Address(dns2);
        Class[] cl = new Class[]{InetAddress.class, int.class};
        Constructor cons = null;

        Class<?> clazz = Class.forName("android.net.LinkAddress");

        try {
            cons = clazz.getConstructor(cl);
        } catch (NoSuchMethodException e) {
            e.printStackTrace();
        }

        if (cons == null) {
            return false;
        }
        Object[] x = {inetAddr, prefixLength};

        Class<?> staticIpConfigurationCls = Class.forName("android.net.StaticIpConfiguration");

        Object staticIpConfiguration = null;

        staticIpConfiguration = staticIpConfigurationCls.newInstance();
        Field ipAddress = staticIpConfigurationCls.getField("ipAddress");
        Field gateWay = staticIpConfigurationCls.getField("gateway");
        Field dnsServers = staticIpConfigurationCls.getField("dnsServers");
        //set ipAddress
        ipAddress.set(staticIpConfiguration, (LinkAddress) cons.newInstance(x));
        //set gateway
        gateWay.set(staticIpConfiguration, gatewayAddr);
        //set dns
        ArrayList<InetAddress> dnsList = (ArrayList<InetAddress>) dnsServers.get(staticIpConfiguration);
        dnsList.add(dnsAddr1);
        dnsList.add(dnsAddr2);
        @SuppressLint("PrivateApi") Class ipAssignmentCls =
Class.forName("android.net.IpConfiguration$IpAssignment");
        Object ipAssignment = null;
        ipAssignment = Enum.valueOf(ipAssignmentCls, "STATIC");
        Method setIpAssignmentMethod = wifiConfig.getClass().getDeclaredMethod("setIpAssignment",
ipAssignmentCls);
        setIpAssignmentMethod.invoke(wifiConfig, ipAssignment);
        Method setStaticIpConfigurationMethod =
```

```
wifiConfig.getClass().getDeclaredMethod("setStaticIpConfiguration", staticIpConfiguration.getClass());
            // Set static IP by assigning StaticIpConfiguration to WifiConfiguration.
            setStaticIpConfigurationMethod.invoke(wifiConfig, staticIpConfiguration);
            int netId = mWifiManager.addNetwork(wifiConfig);
            mWifiManager.disableNetwork(netId);
            return mWifiManager.enableNetwork(netId, true);
        } catch (NoSuchFieldException | IllegalAccessException | InstantiationException |
InvocationTargetException | ClassNotFoundException | NoSuchMethodException e) {
            e.printStackTrace();
        }
        return false;
}

/**
 * Connected to Wi-Fi and configured DHCP Wi-Fi settings.
 * @return return Whether the configuration was successful
 */
public boolean setDhcpWifi() throws Exception{
        List<WifiConfiguration> configuredNetworks = getConfiguredNetworks();
        WifiConfiguration wifiConfig = null;
        WifiInfo connectionInfo = mWifiManager.getConnectionInfo();
        for (WifiConfiguration conf : configuredNetworks) {
            if (conf.networkId == connectionInfo.getNetworkId()) {
                wifiConfig = conf;
                break;
            }
        }
        Class ipAssignmentCls = Class.forName("android.net.IpConfiguration$IpAssignment");
        Object ipAssignment = null;
        ipAssignment = Enum.valueOf(ipAssignmentCls, "DHCP");
        Method setIpAssignmentMethod = wifiConfig.getClass().getDeclaredMethod("setIpAssignment",
ipAssignmentCls);
        setIpAssignmentMethod.invoke(wifiConfig, ipAssignment);
        int netId = mWifiManager.addNetwork(wifiConfig);
        mWifiManager.disableNetwork(netId);
        return mWifiManager.enableNetwork(netId, true);
}
```

4.6.13 Set up the screensaver mask

In the actual usage of the product, if the LCD screen runs in a bright state for a long time, it will reduce the lifespan of the LCD screen and sometimes even cause image retention. It is recommended to add a black protective mask to the application and reduce the screen

brightness when the application is idle. This can extend the lifespan of the LCD screen. The following are variables for displaying operation masks.

```
private WindowManager mWindowManager = null;
private WindowManager.LayoutParams mNightViewParam;
private View mNightView = null;
private boolean mIsAddedView;
```

Add the following methods

```
/**
* Set the black mask
*/
    private void changeToNight() {
        if (mIsAddedView == true)
            return;
        mNightViewParam = new WindowManager.LayoutParams(
                WindowManager.LayoutParams.TYPE_APPLICATION,
                WindowManager.LayoutParams.FLAG_NOT_TOUCHABLE |
                        WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE,
                PixelFormat.TRANSPARENT);
        mWindowManager = getWindowManager();
        mNightView = new View(this);
        mNightView.setBackgroundResource(R.color.black);
        mWindowManager.addView(mNightView, mNightViewParam);
        mIsAddedView = true;
    }
    /**
     * Unmask screen black
     */
    public void changeToDay() {
        if (mIsAddedView && mNightView != null) {
            mWindowManager.removeViewImmediate(mNightView);
            mWindowManager = null;
            mNightView = null;
            mIsAddedView = false;
        }

    }
```

Usage:

```
// Switch to black mask
changeToNight();
// Switch to normal display
changeToDay();
```

4.6.14 To enable auto-start on boot for the application

Enabling auto-start on boot for an Android application can indeed enhance user experience satisfaction. Here are the steps you've provided for configuring auto-start on boot:

1. Configure Permissions and Create Broadcast Receiver

In the `AndroidManifest.xml` file, set up permissions to listen for boot events and create a broadcast receiver named `BootReceiver`.

2. Add the following permissions

Create a broadcast receiver for listening to boot completion.

```
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
```

Add the Receiver

```
<receiver
    android:name=".BootReceiver"
    tools:ignore="Instantiatable">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
        <action android:name="android.intent.action.ACTION_SHUTDOWN"/>
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</receiver>
```

Create a broadcast receiver for listening to boot completion

```
public class BootReceiver extends BroadcastReceiver {

    private SharedPreferences pref;
    private boolean autoStarts = false;
    @Override
    public void onReceive(Context context, Intent intent) {
        if(intent.getAction().equals("android.intent.action.BOOT_COMPLETED")) {
            // boot
            pref = context.getSharedPreferences("BOOT_COMPLETED", Context.MODE_PRIVATE);
            autoStarts = pref.getBoolean("Autostarts",false);
            if (autoStarts){
                Log.e("Autostarts","--");
                Intent intent2 = new Intent(context, MainActivity.class);
```

```
                intent2.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                context.startActivity(intent2);
            }else {
                Log.e("Autostarts","--");
            }

        }
        if (intent.getAction().equals("android.intent.action.ACTION_SHUTDOWN")){
            Log.e("shutdown","--");
        }
    }
}
```

3. Reboot the Android Device

Restart the Android device to test if the app automatically starts after booting.

Note：

Initial App Launch Required: Starting with Android 4.0 and later versions, it's necessary to launch the app at least once manually. This is a security measure to prevent malicious apps from automatically starting on boot. Once the app has been manually launched, it can then receive the broadcast indicating that the device has completed booting.

Check for Security Apps: Make sure to check if any security or task management apps like 360 Security are installed on the device. If they are, you may need to go into their settings and allow your app to auto-start.

Install the App to Internal Storage: Installing the app to the internal storage is recommended. Some Android devices may not allow apps installed on the SD card to auto-start.

4.6.15 Set up WIFI ADB debugging

Execute via command line

```
//Set port to 5555.

setprop service.adb.tcp.port 5555

//Enable Wi-Fi debugging.

start adbd

//Disable Wi-Fi debugging.

stop adbd

//Connect to the device using ADB.
```

```
adb connect ip:5555
```

Execute in code form

```
public void execShell(String cmd) {
    try {
        //Permission setup
        Process p = Runtime.getRuntime().exec("su");
        //Get output stream
        OutputStream outputStream = p.getOutputStream();
        DataOutputStream dataOutputStream = new DataOutputStream(outputStream);
        //Write the command
        dataOutputStream.writeBytes(cmd);
        //Submit the command
        dataOutputStream.flush();
        //Close stream operations
        dataOutputStream.close();
        outputStream.close();

    } catch (Throwable t) {
        t.printStackTrace();
    }
}

    ____________________________________________________________________________

execShell("setprop service.adb.tcp.port 5555");

execShell("start adbd");
```

Hint:

1. Navigate to "Settings" > "About Phone" > "Status Information" to access the IP address.
2. Proceed to "Settings" > "Wi-Fi," then tap the currently connected Wi-Fi network to reveal the IP address. For instance, navigate through "Settings" > "Wireless & Network" > "Wi-Fi Settings" to inspect the IP address of the presently connected Wi-Fi network.
3. To obtain the device's IP address through the command line, employ the following adb command: adb shell netcfg. This command shall unveil the desired information.

4.6.16 Disable the USB root node

Inquire about the nomenclature of the USB root node.

```
adb shell

su

cd /sys/bus/usb/devices

ls
```

```
C:\Users\ning100>adb shell
rk3288:/ $ su
su
rk3288:/ # cd /sys/bus/usb/devices
cd /sys/bus/usb/devices
rk3288:/sys/bus/usb/devices # ls
ls
1-0:1.0 1-1 1-1.3 1-1.3:1.0 1-1:1.0 2-0:1.0 3-0:1.0 usb1 usb2 usb3
rk3288:/sys/bus/usb/devices #
```

Deactivate the corresponding USB root node and restart if ineffectual.

```
adb shell

su

// Disabled usb1

echo 'usb1' > /sys/bus/usb/drivers/usb/unbind

//Open usb1

echo 'usb1' > /sys/bus/usb/drivers/usb/bind
```

4.6.17 Silent installation with automatic initialization

Silent installation occurs imperceptibly to the user, requiring no manual interaction. The underlying principle involves invoking "pm install -r" for installation, where "-r" preserves the app's existing data.

Adding the methodology:

```
protected void executeInstall(String path) {
    Process process = null;
    OutputStream out = null;
    InputStream in = null;
    try {
```

```
            process = Runtime.getRuntime().exec("su");
            out = process.getOutputStream();
            out.write(("pm install -r " + path + "\n").getBytes());
            in = process.getInputStream();
            int len = 0;
            byte[] bs = new byte[256];
            while (-1 != (len = in.read(bs))) {
                String state = new String(bs, 0, len);
                if (state.equals("success\n")) {
                    Log.d("SUC","success");
                      Intent intent=new Intent();

                    intent.setAction("android.intent.action.PACKAGE_REPLACED");

                    sendBroadcast(intent);

                }
            }
        }catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                if (out != null) {
                    out.flush();
                    out.close();
                }
                if (in != null) {
                    in.close();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
}
```

Add Receiver

```
<receiver android:name=".Utils.UpdateRestartReceiver"
    >
    <intent-filter>
        <action android:name="android.intent.action.PACKAGE_REPLACED"/>
        <data android:scheme="package"/>
    </intent-filter>

</receiver>
```

Establishing a broadcast receiver to monitor the completion of startup.

```
public class UpdateRestartReceiver extends BroadcastReceiver
{
    @Override
    public void onReceive(Context context, Intent intent) {
        if (intent.getAction().equals("android.intent.action.PACKAGE_REPLACED")){
            Intent intent2 = new Intent(context, LoginActivity.class);
            intent2.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            context.startActivity(intent2);

        }
    }
}
```

Employing the approach:

```
executeInstall(Path);
```

4.6.18 Acquiring module system information

```
/*

</br>

<p>To utilize the interface methods in this class, you must first invoke {@link
ISystemInfo#getInstance(Context)} to obtain an instance.</p>

<p>Upon conclusion of usage, invoke {@link ISystemInfo#destroyInstance()} to dismantle the singleton.</p>

Interface functionalities:

</br>

<ol>

<li>{@link ISystemInfo#getBrand()} - Retrieve the manufacturer's appellation for the mobile device.</li>

<li>{@link ISystemInfo#getProduct()} - Acquire a mobile manufacturer denomination that holds significance
for users.</li>

<li>{@link ISystemInfo#getBoard()} - Obtain the motherboard model.</li>

<li>{@link ISystemInfo#getCpuABI()} - Through this field, glean the device's instruction set nomenclature
(CPU type).</li>

<li>{@link ISystemInfo#getModel()} - Utilize this field to ascertain the model.</li>

<li>{@link ISystemInfo#getSerial()} - Utilize this field to access the serial number.</li>
```

```
<li>{@link ISystemInfo#getTelephonyState()} - Retrieve the mobile data connection status.</li>

<li>{@link ISystemInfo#getTelephonyDeviceId()} - Retrieve the mobile device ID.</li>

<li>{@link ISystemInfo#getTelephonyNumber()} - Acquire the phone number.</li>

<li>{@link ISystemInfo#getTelephonyNetworkOperatorName()} - Retrieve the network operator's name.</li>

<li>{@link ISystemInfo#getTelephonyNetworkType()} - Acquire the network type.</li>

<li>{@link ISystemInfo#getLocalIpAddress()} - Access the local IP address.</li>

<li>{@link ISystemInfo#getAvailMemory(Context)} - Retrieve the current available Android system memory
size.</li>

<li>{@link ISystemInfo#getTotalMemory(Context)} - Obtain the total current Android system memory
size.</li>

<li>{@link ISystemInfo#getInternalStorageTotal()} - Access the total internal storage capacity.</li>

<li>{@link ISystemInfo#getInternalStorageRemain()} - Retrieve the remaining internal storage capacity.</li>

<li>{@link ISystemInfo#getExternalStorageTotal()} - Obtain the total external storage capacity.</li>

<li>{@link ISystemInfo#getExternalStorageRemain()} - Acquire the available external storage capacity.</li>

</ol>

*/
public class ISystemInfo {
    private static Context mContext;
    private static ISystemInfo mInstance;
    private static TelephonyManager telephonyManager;

    public ISystemInfo(Context context) {
        mContext = context;
        telephonyManager = (TelephonyManager)
mContext.getSystemService(Context.TELEPHONY_SERVICE);
    }

/**

Retrieve a singleton instance for system-related information.

@param context - The contextual object.

@return ISystemInfo - The singleton instance.

*/
    public static ISystemInfo getInstance(Context context) {
        if (mInstance == null) {
            synchronized (ISystemInfo.class) {
                if (mInstance == null) {
```

```
                    mInstance = new ISystemInfo(context);
                }
            }
        }
        return mInstance;
    }
/**

Terminate and control the singleton instance.

*/
    public static void destroyInstance() {
        if (mInstance != null) {
            mInstance = null;
        }
    }
    /**
     * 通过这个字段可以获取到对用户有意义的手机厂商名称，例如 Xiaomi，Meizu，Huawei 等。
     *
     * @return brand
     */
    public String getBrand() {
        return Build.BRAND;
    }

/**


This field enables the retrieval of the meaningful mobile manufacturer name to the user, such as Xiaomi,
Meizu, Huawei, and others.

@return brand

*/
    public String getProduct() {
        return Build.PRODUCT;
    }

/**


This field allows for the retrieval of the mainboard model.

@return motherboard model

*/
    public String getBoard() {
```

```
            return Build.BOARD;
        }

/**

This field facilitates the acquisition of the device instruction set name, denoting the CPU type.

@return instruction set name

*/
        public String getCpuABI() {
            return Build.CPU_ABI;
        }

/**

This field enables the acquisition of the model information.

@return model

*/
        public String getModel() {
            return Build.MODEL;
        }

/**

This field grants access to the device's serial number.

@return serial number

*/
        public String getSerial() {
            return Build.SERIAL;
        }

/**

Obtain the status of mobile data connectivity through this field.

<p>

DATA_CONNECTED = 2: Data connectivity status - Established.

DATA_CONNECTING = 1: Data connectivity status - Currently establishing.
```

```
DATA_DISCONNECTED = 0: Data connectivity status - Disconnected.

DATA_SUSPENDED = 3: Data connectivity status - Temporarily paused.

*/
    public int getTelephonyState() {
        if (telephonyManager != null) {
            telephonyManager.getDataState();
        }
        return -1;
    }

/**

Retrieve the unique device identifier.

If operating on a GSM network, it will yield the IMEI; in the case of a CDMA network, it shall provide the
MEID. Should the device identifier be unavailable, it shall return null.
  */
    public String getTelephonyDeviceId() {
        if (telephonyManager != null) {
            telephonyManager.getDeviceId();
        }
        return "";
    }

/**



Return the mobile number.

Referred to as MSISDN for GSM networks. If unavailable, it shall return null.

*/
    @SuppressLint("MissingPermission")
    public String getTelephonyNumber() {
        if (telephonyManager != null) {
            telephonyManager.getLine1Number();
        }
        return "";
    }

/**

Retrieve the appellation of the mobile network operator (SPN).

*/
    public String getTelephonyNetworkOperatorName() {
```

```
        if (telephonyManager != null) {
            telephonyManager.getNetworkOperatorName();
        }
        return "";
    }

/**


Acquire the network type.

<p>

NETWORK_TYPE_CDMA = 4, denoting CDMA network type.

NETWORK_TYPE_EDGE = 2, denoting EDGE network type.

NETWORK_TYPE_EVDO_0 = 5, denoting EVDO0 network type.

NETWORK_TYPE_EVDO_A = 6, denoting EVDOA network type.

NETWORK_TYPE_GPRS = 1, denoting GPRS network type.

NETWORK_TYPE_HSDPA = 8, denoting HSDPA network type.

NETWORK_TYPE_HSPA = 10, denoting HSPA network type.

NETWORK_TYPE_HSUPA = 9, denoting HSUPA network type.

NETWORK_TYPE_UMTS = 3, denoting UMTS network type.

<p>

*/
    @SuppressLint("MissingPermission")
    public int getTelephonyNetworkType() {
        if (telephonyManager != null) {
            telephonyManager.getNetworkType();
        }
        return -1;
    }

/**


Retrieve the system version.

@return The version of the system.

*/
    public String getSystemVersion() {
```

```
        return Build.VERSION.RELEASE;
    }

/**

Obtain the current system language and region.

@return The system language and region, such as "zh-CN".

*/
    public String getSystemLanguageAndCountry() {
        return Locale.getDefault().toString();
    }

/**

Acquire the current system language.

@return The system language, such as "zh".

*/
    public static String getSystemLanguage() {
        return Locale.getDefault().getLanguage();
    }

/**

Retrieve the current system language locale.

@return The system language locale, such as "CN".

*/
    public String getSystemCountry() {
        return Locale.getDefault().getCountry();
    }

/**

Retrieve the current CPU temperature.

@return The temperature as a floating-point value, e.g., "52.1" for 52.1°C.

*/
    public float getCurrentCPUTemperature() {
```

```
        String file = readFile("/sys/devices/virtual/thermal/thermal_zone0/temp", '\n');
        if (file != null) {
            if(Build.VERSION.SDK_INT > Build.VERSION_CODES.M) {
                return Long.parseLong(file)/1000;
            }
            return Long.parseLong(file)/100;
        } else {
            return Long.parseLong("0");

        }
    }

    private byte[] mBuffer = new byte[4096];
    @SuppressLint("NewApi")
    private String readFile(String file, char endChar) {
        StrictMode.ThreadPolicy savedPolicy = StrictMode.allowThreadDiskReads();
        FileInputStream is = null;
        try {
            is = new FileInputStream(file);
            int len = is.read(mBuffer);
            is.close();

            if (len > 0) {
                int i;
                for (i = 0; i < len; i++) {
                    if (mBuffer[i] == endChar) {
                        break;
                    }
                }
                return new String(mBuffer, 0, i);
            }
        } catch (java.io.FileNotFoundException e) {
        } catch (java.io.IOException e) {
        } finally {
            if (is != null) {
                try {
                    is.close();
                } catch (java.io.IOException e) {
                }
            }
            StrictMode.setThreadPolicy(savedPolicy);
        }
        return null;
    }
```

```
/**

Retrieve the total external storage capacity.

@return The capacity as a floating-point value, where "1024" represents 1024MB.

*/
    public float getExternalStorageTotal() {
        String state = Environment.getExternalStorageState();
        if(Environment.MEDIA_MOUNTED.equals(state)) {
            File sdcardDir = Environment.getExternalStorageDirectory();
            StatFs sf = new StatFs(sdcardDir.getPath());
            long blockSize = sf.getBlockSize();
            long blockCount = sf.getBlockCount();
            return blockSize*blockCount/1024/1024;
        }
        return 0;
    }

/**

Retrieve the remaining external storage capacity.

@return The remaining capacity as a floating-point value, where "1024" signifies 1024MB remaining.

*/
    public float getExternalStorageRemain() {
        String state = Environment.getExternalStorageState();
        if(Environment.MEDIA_MOUNTED.equals(state)) {
            File sdcardDir = Environment.getExternalStorageDirectory();
            StatFs sf = new StatFs(sdcardDir.getPath());
            long blockSize = sf.getBlockSize();
            long blockCount = sf.getBlockCount();
            long availCount = sf.getAvailableBlocks();
            return   availCount*blockSize/1024/1024;
        }
        return 0;
    }

   /**

Obtain the cumulative internal storage capacity.
```

```
@return A floating-point value denoting an internal memory capacity of 1024MB.

*/      public float getInternalStorageTotal() {
            File root = Environment.getRootDirectory();
            StatFs sf = new StatFs(root.getPath());
            long blockSize = sf.getBlockSize();
            long blockCount = sf.getBlockCount();
            long availCount = sf.getAvailableBlocks();
            return blockSize*blockCount/1024/1024;
        }

/**



Acquire the remaining internal storage capacity.

@return A floating-point value representing 1024MB of available memory.

*/
        public float getInternalStorageRemain() {
            File root = Environment.getRootDirectory();
            StatFs sf = new StatFs(root.getPath());
            long blockSize = sf.getBlockSize();
            long blockCount = sf.getBlockCount();
            long availCount = sf.getAvailableBlocks();
            return availCount*blockSize/1024/1024;
        }

/**



Retrieve the local IP address.

@return An illustration of the local IP address, e.g., "192.168.1.135".

*/
        public String getLocalIpAddress() {
            String ip = "null";
            ConnectivityManager conMann = (ConnectivityManager)
                        mContext.getSystemService(Context.CONNECTIVITY_SERVICE);
            NetworkInfo mobileNetworkInfo = conMann.getNetworkInfo(ConnectivityManager.TYPE_MOBILE);
            NetworkInfo wifiNetworkInfo = conMann.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
            if (mobileNetworkInfo.isConnected()) {
                ip = getIpAddress();
            }else if(wifiNetworkInfo.isConnected())
            {
                WifiManager wifiManager = (WifiManager)
```

```
mContext.getApplicationContext().getSystemService(Context.WIFI_SERVICE);
                WifiInfo wifiInfo = wifiManager.getConnectionInfo();
                int ipAddress = wifiInfo.getIpAddress();
                ip = intToIp(ipAddress);
            }
            return ip;
        }
        private static String intToIp(int ipInt) {
            StringBuilder sb = new StringBuilder();
            sb.append(ipInt & 0xFF).append(".");
            sb.append((ipInt >> 8) & 0xFF).append(".");
            sb.append((ipInt >> 16) & 0xFF).append(".");
            sb.append((ipInt >> 24) & 0xFF);
            return sb.toString();
        }
/**


Attain the IP address.

@return

*/
        private String getIpAddress() {
            try {
                for (Enumeration<NetworkInterface> en = NetworkInterface
                        .getNetworkInterfaces(); en.hasMoreElements();) {
                    NetworkInterface intf = en.nextElement();
                    for (Enumeration<InetAddress> enumIpAddr = intf
                            .getInetAddresses(); enumIpAddr.hasMoreElements();) {
                        InetAddress inetAddress = enumIpAddr.nextElement();
                        if (!inetAddress.isLoopbackAddress()
                                && inetAddress instanceof Inet4Address) {
                            return inetAddress.getHostAddress().toString();
                        }
                    }
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
            return null;
        }

/**
```

```
Retrieve the current available operational memory size for Android.

@param context The context.

*/
    public String getAvailMemory(Context context) {
        ActivityManager am = (ActivityManager) context.getSystemService(Context.ACTIVITY_SERVICE);
        ActivityManager.MemoryInfo mi = new ActivityManager.MemoryInfo();
        am.getMemoryInfo(mi);
        // mi.availMem;
        return String.format("%.2f", (float) mi.availMem/1024/1024);
    }

/**



Obtain the total operational memory size for Android.

@param context The context.

*/
    public String getTotalMemory(Context context) {
        String str1 = "/proc/meminfo";
        String str2;
        String[] arrayOfString;
        float initial_memory = 0;
        try {
            FileReader localFileReader = new FileReader(str1);
            BufferedReader localBufferedReader = new BufferedReader(localFileReader, 8192);
            str2 = localBufferedReader.readLine();
            arrayOfString = str2.split("\\s+");
            // Acquire the total system memory for Android, measured in kilobytes.
            initial_memory = Integer.valueOf(arrayOfString[1]).intValue();
            localBufferedReader.close();
        } catch (IOException e) {
        }
        return String.format("%.2f", initial_memory/1024);
    }
/**



Retrieve the serial number of the mobile device.

@return The mobile device's serial number.

*/
    @SuppressLint({"NewApi", "MissingPermission"})
```

```
    public String getSerialNumber() {
        String serial = "";
        try {
            if (Build.VERSION.SDK_INT > Build.VERSION_CODES.N) {//8.0+
                serial = Build.SERIAL;
            } else {//8.0-
                Class<?> c = Class.forName("android.os.SystemProperties");
                Method get = c.getMethod("get", String.class);
                serial = (String) get.invoke(c, "ro.serialno");
            }
        } catch (Exception e) {
            e.printStackTrace();
            Log.e("e", " Anomalies encountered while retrieving the device's serial number: " +
e.toString());
        }
        return serial;
    }
}
```

Chapter 5:Peripheral selection and functionality support

5.1 Peripherals and accessories

The module can accommodate a diverse array of peripheral accessories, allowing you to select those that align with your specific requirements.

Functionality	Configuration	Detailed diagram
GPS	To avail of the GPS positioning feature, it is imperative to engage in a discourse with the vendor during the module acquisition, with a view to procuring the 4G functionality. Once the 4G feature augmentation is secured, the purchase of a GPS module and the subsequent connection of its terminal via an IPEX to SMA female connector shall suffice to facilitate GPS positioning.	
Audio Output	For the 7-inch module, the audio output interfaces consist of dual 4-ohm 3-watt connectors. The terminal interface adopts the PH2.0 MM connector. As for the 10.1-inch module, the audio output interfaces also feature dual 4-ohm 3-watt connectors. The terminal interface, in this case, utilizes the XH2.54 MM connector.	
HDMI	The module offers compatibility with HDMI 2.0. If HDMI functionality is desired, it is necessary to procure an HDMI 2.0 adapter cable.	
Power	It is advised to employ a 12V 2A power adapter for operational power supply (for detailed maximum voltage support, please refer to the Android product datasheet).	

External Storage Information	Prolonged utilization of software, particularly with irregular memory cards, may precipitate data loss or corruption. It is advisable to opt for memory cards from reputable brands to mitigate such risks.	
Camera	USB 2.0 Plug-and-Play, featuring a high-definition camera ideal for the advancement of facial recognition technology.	
Camera	USB 2.0 Plug-and-Play, boasting a high-definition infrared camera, perfectly tailored for the progression of facial recognition technology.	
4G	Equipped with 4G connectivity, the following module options are at your disposal: 1. N58: Featuring a directional communication module with GPS support, offering a cost-effective CAT1 solution. 2. EC20: Embracing a MobileTek communication module, renowned for its enhanced signal stability and operating under the CAT4 standard. 3. EC25: The MobileTek communication module, supporting a wide range of global frequency bands, also operates under the CAT4 standard.	
USBOTG	The modules are compatible with the Micro USB interface, facilitating configuration through a Micro USB data cable. In the event that the module remains unrecognized by the computer via the data cable, consider installing the Android driver.	

Printer	The Android system integrated into the modules is compatible with a variety of mainstream printers available on the market. Users have the option to install printing applications developed by printer manufacturers from the Android marketplace or employ the PrinterShare application for their printing needs.	No pictures

Chapter 6: Precautions for Module Usage

6.1 Considerations

1. Refrain from inserting or removing the core board and peripheral modules while they are powered.
2. Kindly adhere to all the warnings and guidance provided on the product.
3. Maintain the product's dryness diligently. In the unfortunate event of any liquid spillage or immersion, promptly power it down and allow for thorough drying.
4. During operation, pay heed to the ventilation and heat dissipation of this product, preventing high temperatures from causing component damage.
5. Avoid using or storing this product in dusty or untidy environments.
6. Do not subject this product to environments with fluctuating temperature extremes to prevent component damage.
7. Handle this product with care; impacts, drops, or excessive vibrations may result in damage to its circuits and components.
8. Do not utilize organic solvents or corrosive liquids for cleaning this product.
9. Do not attempt self-repairs or disassembly of the company's products. In the event of product malfunction, please promptly contact the company for repairs.
10. Unauthorized modification or the use of unapproved accessories may lead to damage to the product, and such damages will not be covered under warranty.
11. If the LCD screen operates continuously at maximum brightness, the backlight's lifespan will be halved. Prolonged, high-contrast static displays for 30 minutes or more may lead to LCD screen retention. It is advisable to implement a screensaver to mitigate this issue.

6.2 Serial Port Usage Guidelines

Guideline One: Path

Due to the presence of multiple serial ports on the Android screen, it is imperative to ascertain the path of the connected serial port.

Guideline Two: Baud Rate

Baud rate plays a pivotal role as a transmission frequency, and any inconsistency can lead to garbled data. In cases where messages can be received but proper functionality remains elusive during testing, the primary culprit is often an incorrect serial port baud rate.

Guideline Three: Garbled Data

Garbled data typically arises from inconsistencies between the transmitter and receiver, necessitating a meticulous check to ensure that baud rates, parity bits, stop bits, and data formats (HEX or ASCII) match.

Guideline Four: Abnormal Data Transmission

1. It is recommended to first eliminate issues with the serial port connections and address any deficiencies in product wiring.
2. Upon resolving the first scenario, assess the working performance of the serial port through serial port testing software.
3. After addressing the first scenario, test whether the USB-to-serial tools are functioning properly.

6.3 Common USB Device Issues

USB devices typically undergo automatic recognition when connected to the internet. In the absence of an internet connection, manual confirmation is necessary. When the USB interface of a PC is linked to the DEBUG/OTG interface of an Android screen, the Android screen can be identified automatically.

Issue One: Excessive USB connections to the current computer may lead to connectivity anomalies.

Issue Two: Upon powering up, the computer's device manager will automatically refresh and present new devices for installation. If the device manager fails to refresh, please inspect the USB connection.

Issue Three: If a prompt dialog emerges, select the option to install drivers and initiate an automatic driver search.

Issue Four: Upon successful installation, an additional entry for 'Android Tablet' will appear in the device manager. If an exclamation mark is displayed, uninstall the driver and reconnect the device.

6.4 Product connectivity malfunction

Resolution one: Driver errors necessitate a reinstallation of drivers.

Resolution two: Reconnect all interfaces by reinserting them.

Resolution three: Reboot the Android screen, computer, Wandoujia, mobile assistant, and so forth.

Resolution four: It's important to note that a maximum of one Android device can be connected to the PC at any given time.

6.5 Other issues

1. In instances where you encounter issues such as product screen blackout, white screen anomalies, delayed touchscreen responsiveness, or the inability to access the main interface, please refer to Section 2.1.4 regarding firmware flashing procedures. If the problem persists, kindly get in touch with the sales engineer assigned to you for return and repair arrangements.

2. Should the product display screen exhibit aberrations, it is advisable to reseat the LCD screen connection ribbon cable or contact the sales engineer designated for your support for arrangements related to return and repair. Pixelation or irregular screen patterns are often attributable to loose or aging ribbon cables.

3. If you observe breakpoints in the touchscreen operation, where specific columns or rows are unresponsive to touch input, navigate to the "Settings" menu, access the "Developer Options," and select "Pointer Location" to verify the status of touch points. In the event of breakpoint occurrences, please contact your designated sales engineer for return and repair assistance.