愿做您成功路上的基石！

——百杰科技

HelperBoard A733 开发板

用户手册

V1.2

2026-06

百杰科技 BAIJIE TECHNOLOGY

www.szbaijie.com

www.szbaijie.cn

深圳市百杰科技有限公司

地址：深圳市光明区马田街道钟表基地飞亚达钟表大厦 1A 栋 602

修订记录

日期	版本	修订说明	作者
2025.08.11	V1.0	新建	段竞成
2026.06.01	V1.1	增加屏幕连接图	段竞成
2026.06.12	V1.2	增加 ubuntu 系统屏幕旋转说明	段竞成

目录

1 HelperBoard A733 开发板介绍

1.1 简介

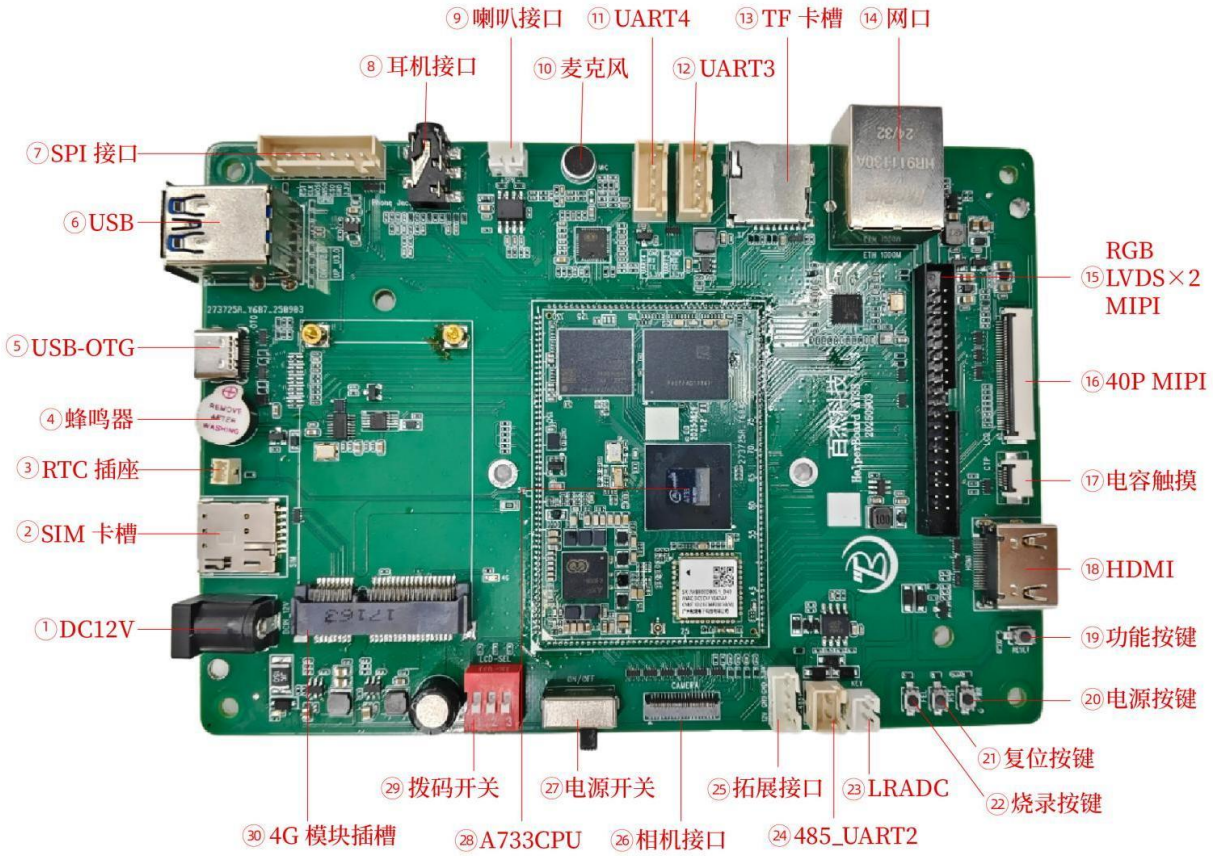
HelperBoard A733 开发板（**以下简称 HelperA733**）是百杰科技 HelperBoard 系列开发板的第七款，继承了 HelperA523 的所有优秀特性，简洁、美观、稳定、高效，适合更高端的应用需求。

1.2 系统特性

- 主控芯片：全志 A733
- CPU：64 位 双核 A76+六核 A55 最高主频：2.0GHz
- NPU：3T
- 内存：4GB/6GB （最大 16GB）LPDDR5
- 贮存：64GB （最大 1T）UFS2.2（可选 UFS3.0）
- 图形处理器：IMG BXM-4-64 MC1
- 视频处理器：H.265 MP/VP9/AVS2 decoder 8K@24fps、H.264 BL/MP/HP decoder 4K@30fps、JPEG decoder 1080P@60fps、H.264/H.265 encoder 4K@30fps、MJPEG/JPEG encoder 4K@15fps、JPEG encoder 4K@15fps
- 解码：8K@24fps H.265、4K@30fps H.264
- 显示：40 针 MIPI/LVDSx2/RGB 复用接口、HDMI
- 以太网：10/100/1000M BASE
- WIFI/BT：5G WIFI/BT 5.0
- 通讯模块：4G（EC20 模块）
- USB：一路超高速 USB3.0，一路高速 USB OTG
- 一路 TYPE-C 接口，内部复合 ADB 和 UART0 调试串口
- 一路 TF 卡接口
- 一路 MIC 接口
- 一路 3.5mm 耳机输出接口
- 一路喇叭输出接口
- 一路 MIPI 屏幕接口
- 一路电容触摸接口
- 一路 485 接口（UART2）
- 一路 CSI 摄像头双摄接口，最高支持 13M 像素
- 一路蜂鸣器
- 扩展接口：2 组 UART、1 组 SPI、2 组 3.3V 电源、1 组 12V 电源、1 组 LRADC
- 一个电源按键（PWR）、一个复位按键（RST）、一个烧录按键（FEL）、一个功能按键

- 一路外接 RTC 时钟插座
- 供电：12v 电压输入
- 系统：Android15.0、ubuntu+qt、ubuntu+xfce

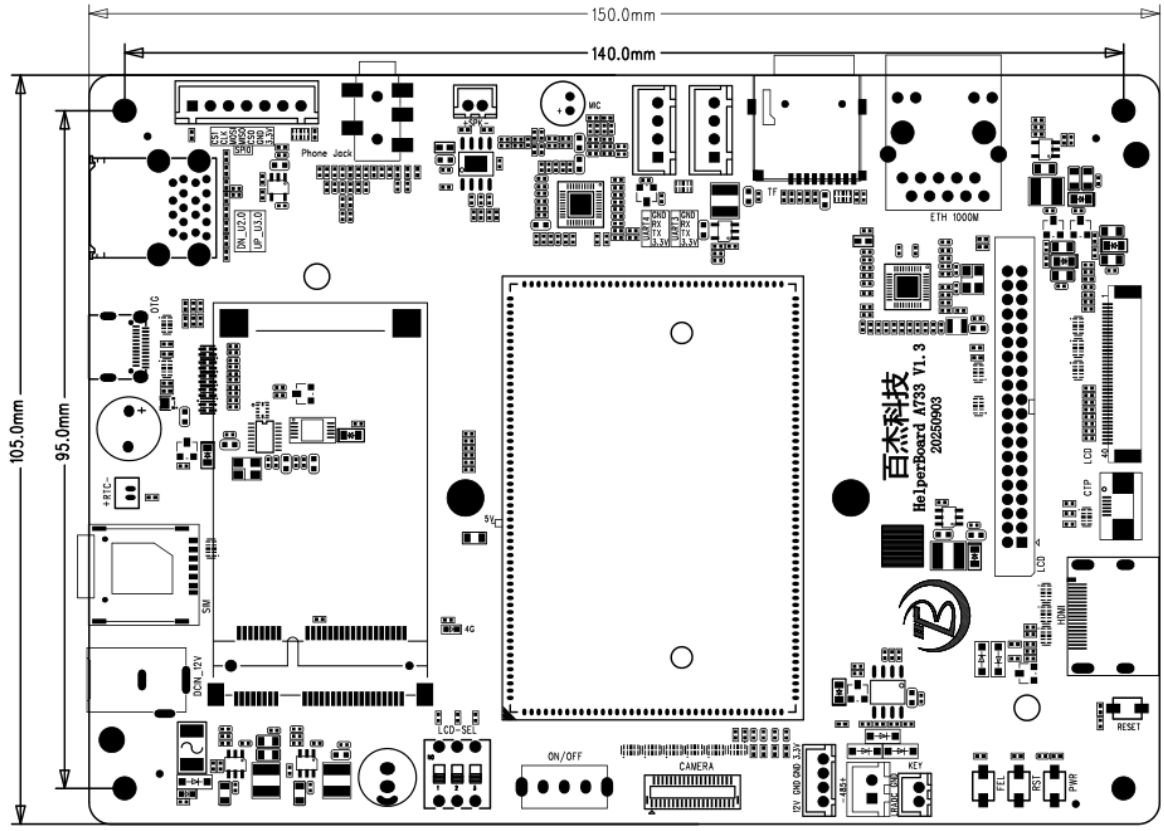
1.3 硬件资源描述



位置	描述
①	DC9~12V 电源
②	SIM 卡槽
③	RTC 电池插座
④	蜂鸣器
⑤	USB-OTG 烧录，DEBUG 调试二合一 type-c 口
⑥	扩展 USB HOST
⑦	SPI 接口
⑧	耳机接口
⑨	喇叭接口
⑩	麦克风
⑪	UART4
⑫	UART3
⑬	TF 卡槽
⑭	千兆网口
⑮	40 针液晶屏接口，LVDSx2/MIPI/RGB 复用接口（引脚定义参见原理图）
⑯	标准 40P MIPI
⑰	电容触摸接口
⑱	HDMI 接口
⑲	功能按键
⑳	电源按键
㉑	复位按键
㉒	烧录按键
㉓	LRADC
㉔	485(UART2)

㉕	扩展接口（一组 3.3v 供电、一组 12v 供电、两个 GND）
㉖	相机接口
㉗	电源开关
㉘	A733CPU
㉙	拨码开关
㉚	4G 模块插槽

1.4 硬件尺寸图

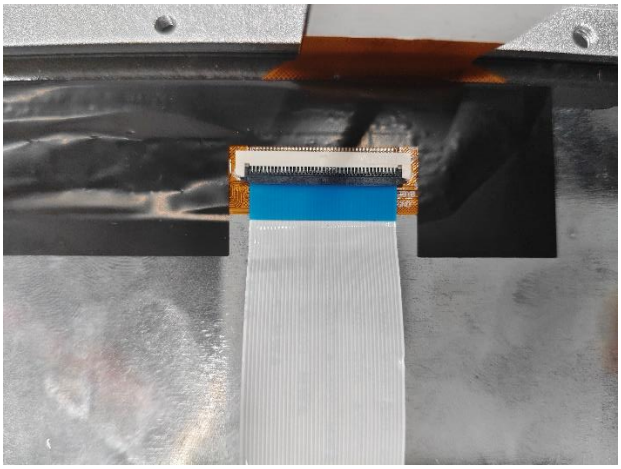


1.5 屏幕连接图

（1）lvds7.0 1024x600 连接图

（2）mipi10.1 800x1280 和 mipi10.1 1200x1920 连接图

mipi 屏幕排线背面连接图



（3）mipi4.3 480x800 连接图

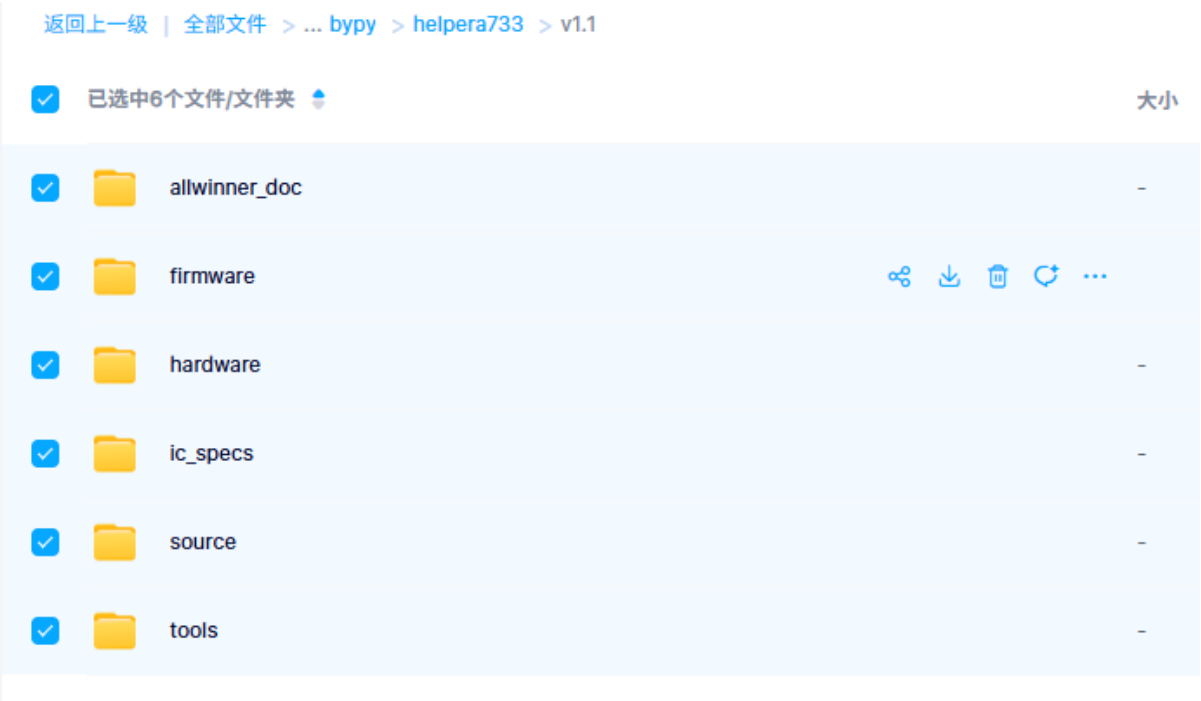
（4）mipi5.0_720x1280 连接图

2 终端调试

2.1 网盘资源下载

本手册所有用到的软件、硬件、源码和固件资源都在百度网盘中，网盘链接请咨询客服获取。

网盘目录如下：



2.2 Xshell 的安装与使用

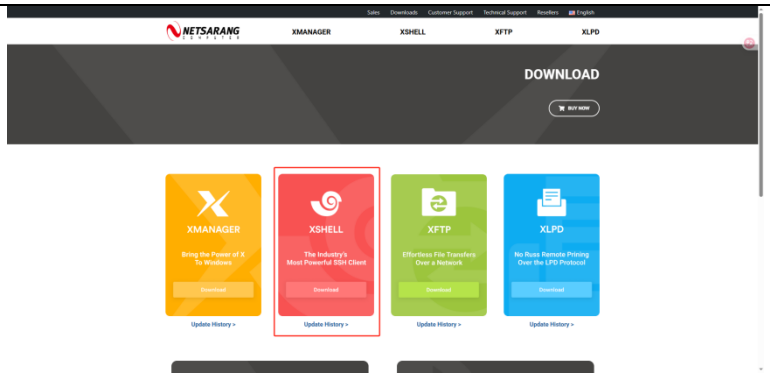
2.2.1 Xshell 安装

在进行测试前，需要安装 Xshell，方便进入终端操作，以下说明 Xhsell 的下载安装。

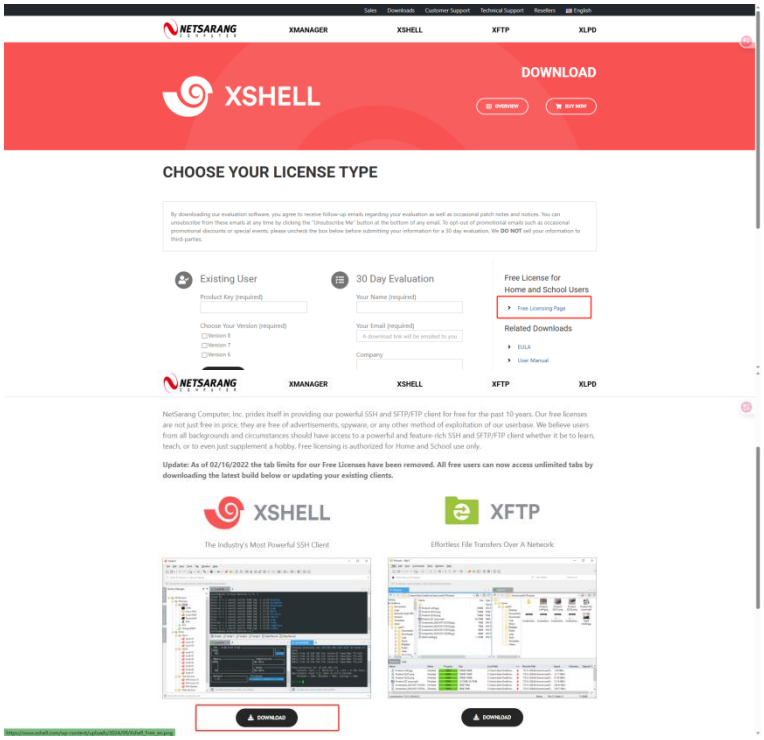
1、进入 Xshell 官网：https://www.xshell.com/

2、点击 DOWNLOAD 进入下载页面

3、选择下载 XSHELL



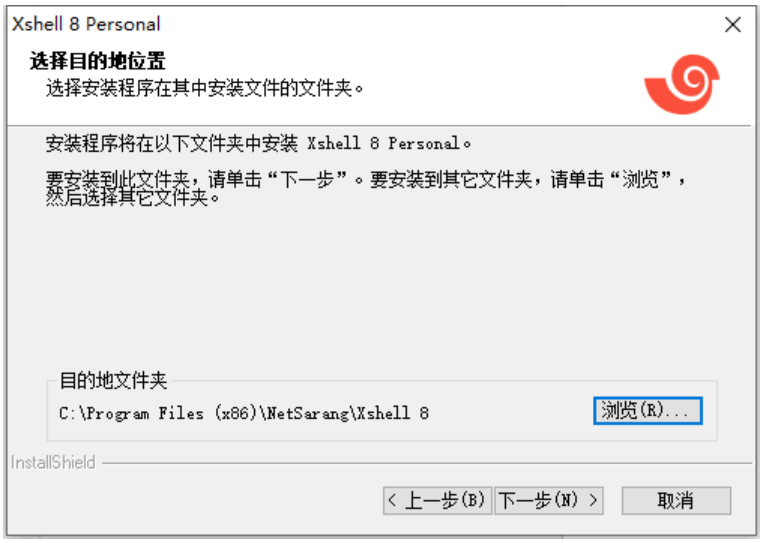
4、选择免费版本下载



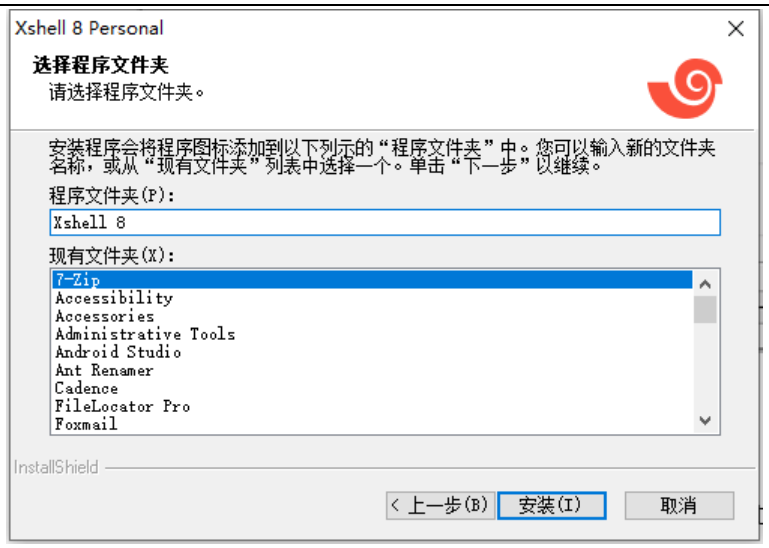
5、下载完成后双击.exe 文件进入安装界面，点击下一步

选择我接受许可协议后继续点击下一步

选择好安装路径后再次点击下一步，默认如下



点击安装



最后点击完成，自动启动 Xshell

第一次启动需要进行注册，填写自己的邮箱提交即可

点击回复的的链接完成注册后重新启动 Xshell 就可以正常使用了

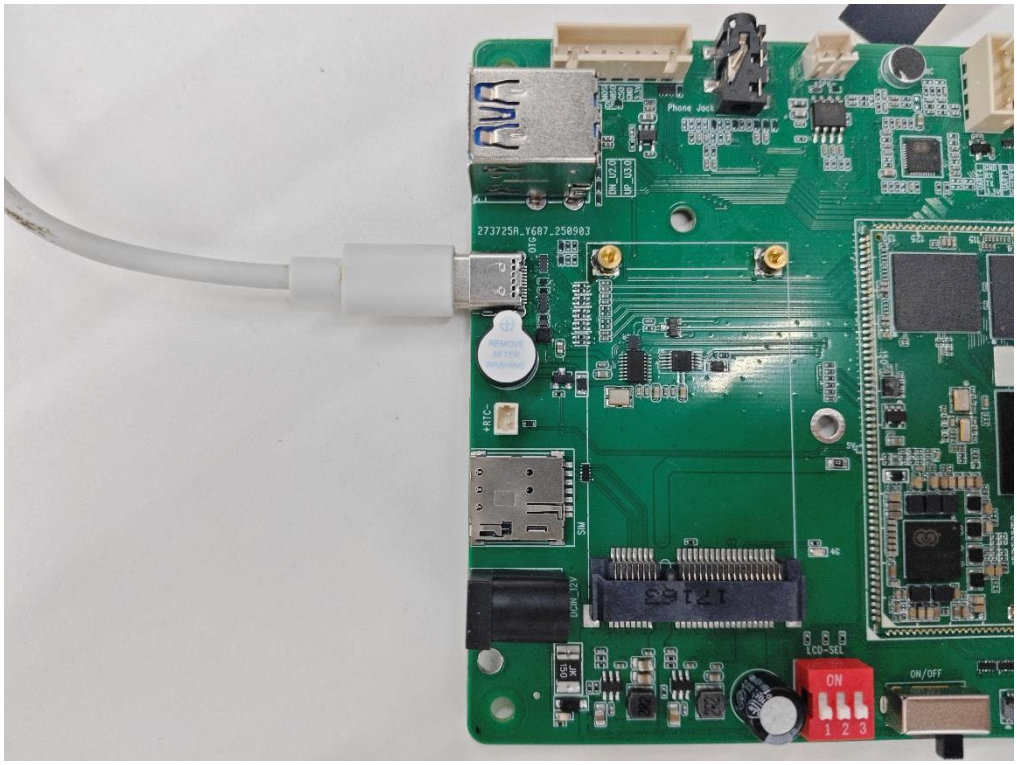
6、拓展软件

串口连接工具不仅仅局限于 Xshell，可以根据用户喜好选择其他的工具连接，例如 MobaXterm 等，篇幅有限，具体安装教程请自行百度搜索。

2.2.2 开发板串口连接

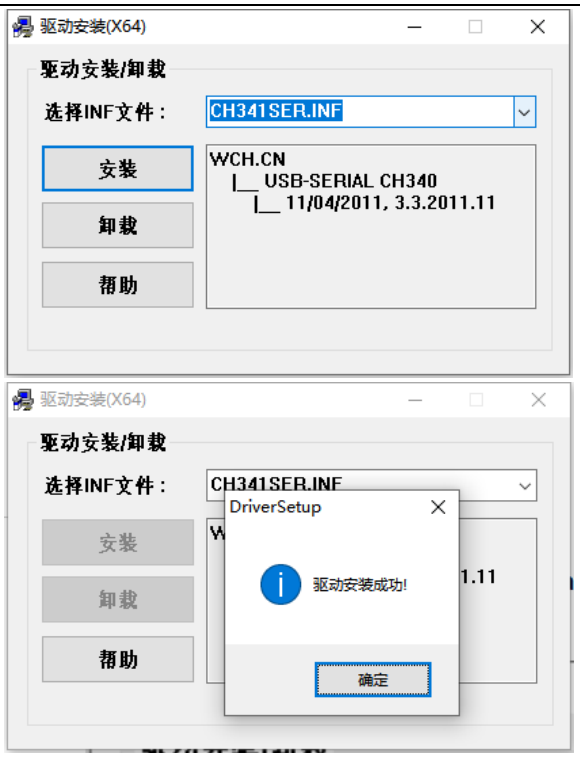
1、准备工作

按图将 USB 线与 1.3 节中的⑤USB-OTG 口连接好



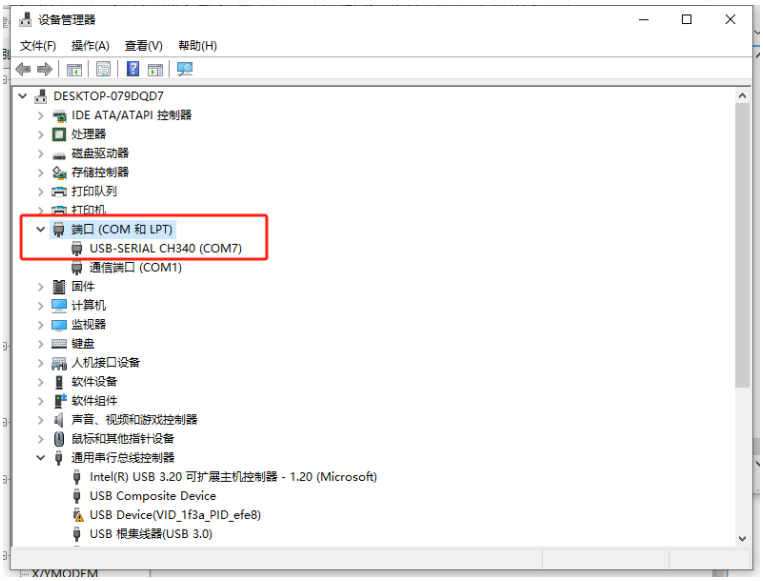
2、安装串口驱动程序

在百度网盘资源下载目录中找到 tools/usb2 serial_driver/CH-340 driver.EXE 驱动程序，下载并安装好。



2.2.3 Xshell 连接开发板

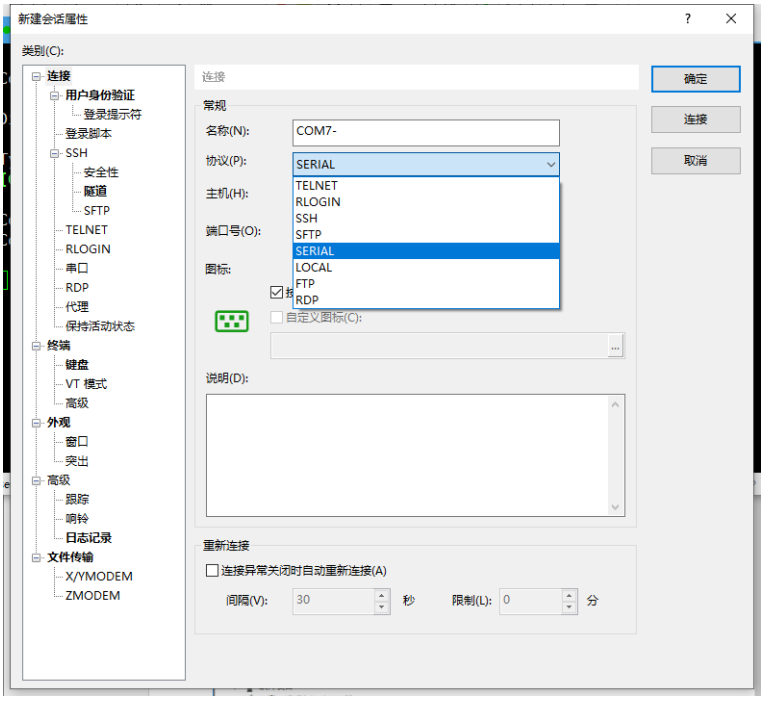
用 USB 线将开发板 USB-OTG 接口和 PC 机 USB 接口相连接后，打开设备管理器查看端口是否新增加了。我电脑上跟开发板通信的端口对应为 COM7，举例也用 COM7，用户根据自己的端口号进行配置。



然后网上下载 Xshell 自行安装好之后，点击文件 ->新建。

在“**连接**“中修改“**名称**”为 COM7-。

在“**协议**“目录下选择 SERIAL



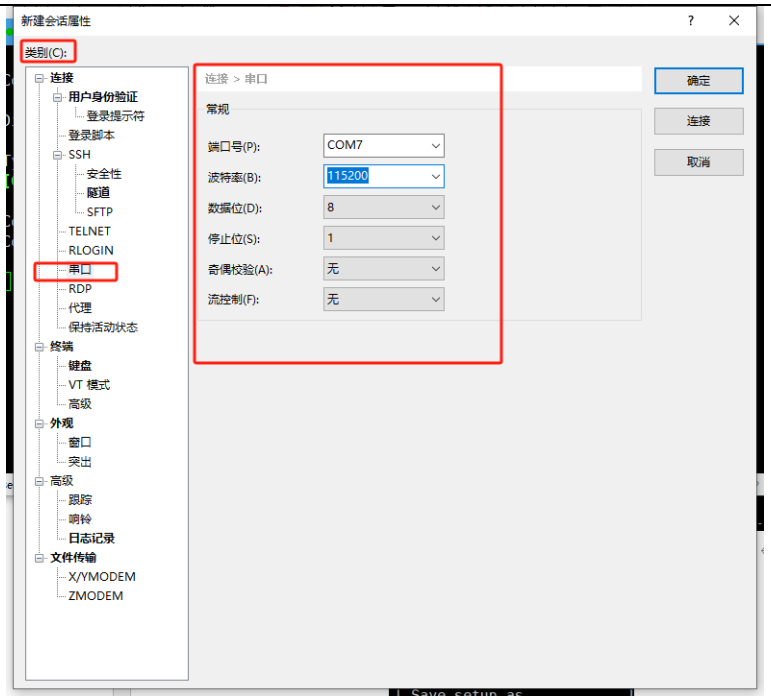
当设置好协议后，点击“**类别**”目录下选择“串口“。

”**端口号**“项选择 PC 上的调试串口端口号“**COM7**”。

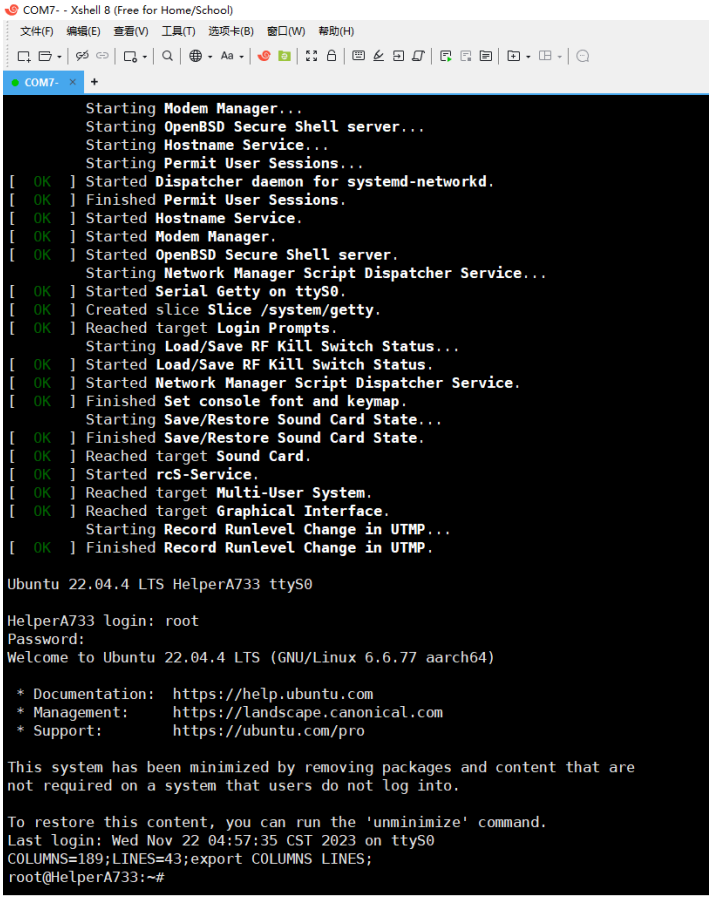
”波特率“项设置为 115200。

”**数据位**“项选择 8 位。

“**停止位**“项选择 1 位。最后两项为“无”。



当 Xshell 成功设置好串口后点击连接，开发板重新开机，就会打印如下图信息。这时候你就可以通过“串口”操控开发板了。



2.3 adb 调试

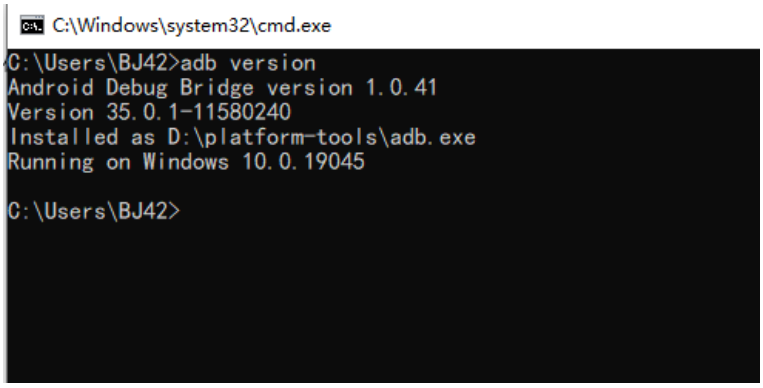
使用 USB 数据线与 1.3 节图中的⑤USB-OTG 口连接好

2.3.1 adb 基础使用

在 PC 电脑端执行组合按键“Win + R”，打开运行窗口，输入“cmd”并回车。

输入以下命令查看 adb 版本

```
$ adb version
```



如果没有安装 adb，可下载网盘 **tools\adb\adb_V1.0.41.zip**，解压该文件，在解压目录中（或者配置好环境变量就可以在任何路径下），按住 shift+鼠标右键，选择“在此处打开 Powershell 窗口”。

在 A733 系统中输入以下命令打开 adb 功能（出厂固件默认打开）

```
$ cat /sys/devices/platform/soc@3000000/10.usbc0/usb_device
```

出现以下打印则打开成功

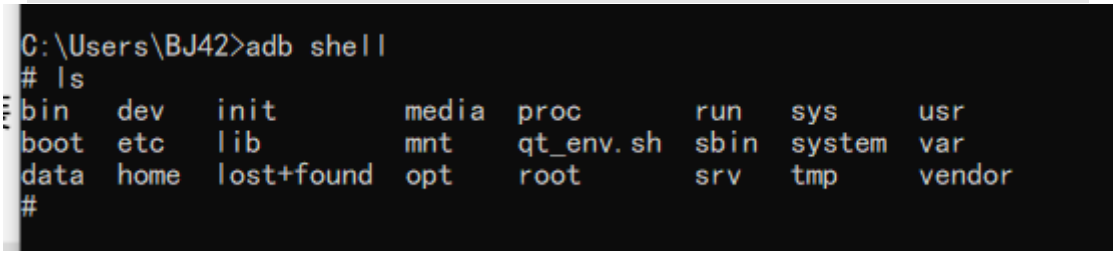
```
root@HelperA733:/# cat /sys/devices/platform/soc@3000000/10.usbc0/usb_device
device_chose finished, otg disabled!
root@HelperA733:/#
```

接着在 cmd 命令行终端输入以下命令查看 adb 设备

```
$ adb devices
```

执行以下命令连接开发板，连接正常后状态如下所示

```
$ adb shell
```



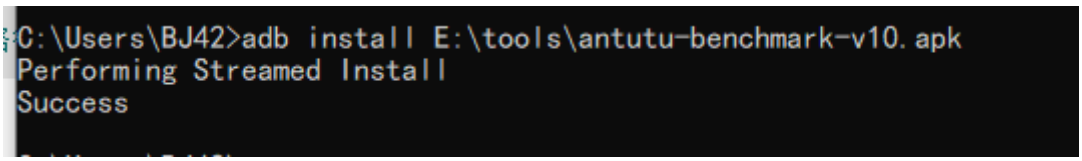
更多 adb 命令请自行百度学习

2.3.2 使用 adb 安装 apk

设备连接好后就可以安装 apk，apk 存放的目录中不能出现中文名的目录名，否则会导致安装失败，安装命令如下：

```
$ adb install apk 路径和文件名
```

安装成功如下图所示：



当使用 adb devices 出现多个设备时，需要加选项-s 区分设备号，例如：

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


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

C:\Users\BJ42>
```

3 硬件功能测试

HelperA733 出厂时默认烧写的是 ubuntu 系统，开发板可以使用 DC 电源供电，开发板上电后即可进行硬件系统测试，以便确保开发板各功能部件工作正常。

DC 电源供电：在 1.3 节图中的①为 DC 电源接口，使用配套的 12V 电源适配器连接后并打开 1.3 节图中的㉙电源开关启动系统。

3.1 开发板 Ubuntu 系统功能测试

3.1.1 开发板 Ubuntu 账户与密码

开发板超级用户账号：**root**

开发板普通用户账号：**szbaijie**

开发板超级用户和普通用户密码：**szbaijie**

3.1.2 485 测试

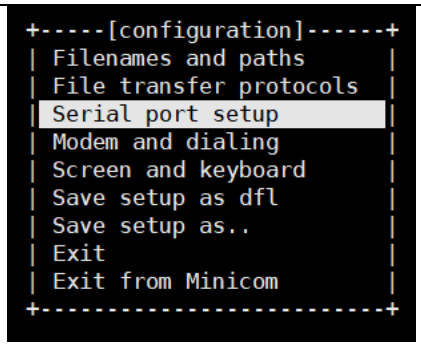
在 1.3 节图中的㉔为 485_UART2 接口。485 测试，需要两个 485 接口对接，只有一个接口无法测试，此测试是在 A 和 B 两块开发板对接的环境下测试的。A 开发板和 B 开发板都需要连接 Xshell（注：Xshell 的使用查看 2.2 节）。

485 通信采用 minicom 软件进行测试，两块开发板都需要安装 minicom，命令如下：

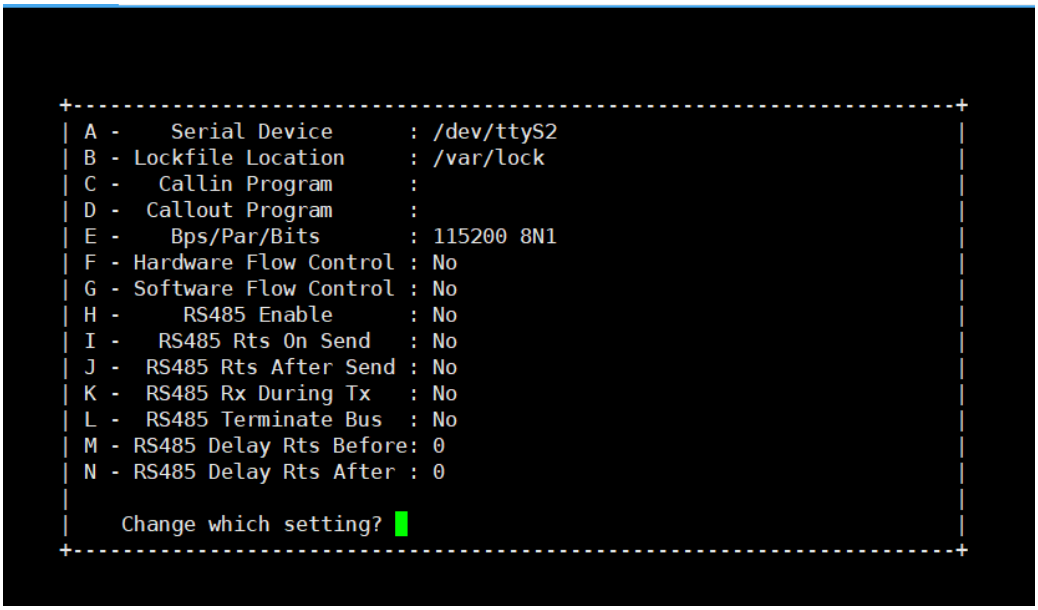
```
$ apt-get install minicom
```

安装完后进入进行配置，首先进入配置界面，命令如下：

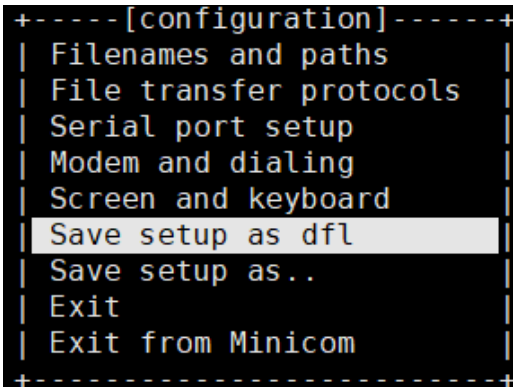
```
$ minicom -s
```



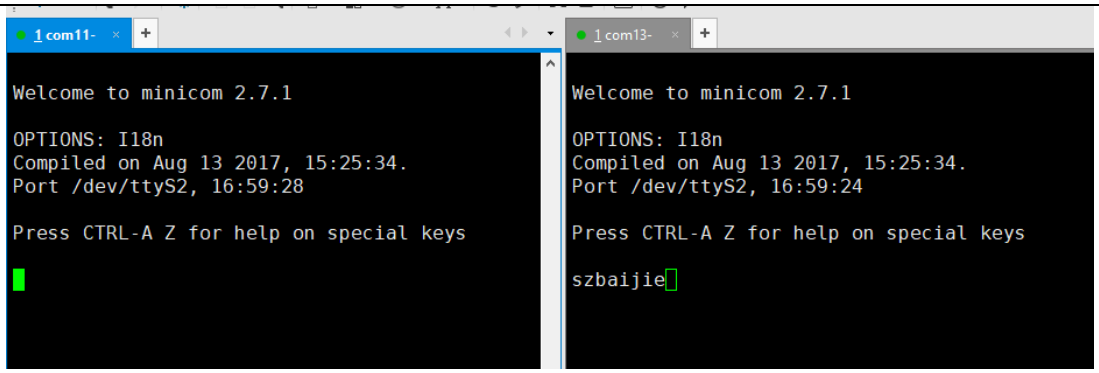
选择 Serial port setup 按回车进入端口配置，按“A”进入通信端口设置，将端口设置为 /dev/ttyS2；再按“F”将 Hardware Flow Control 改为“No”，最后配置如下图：



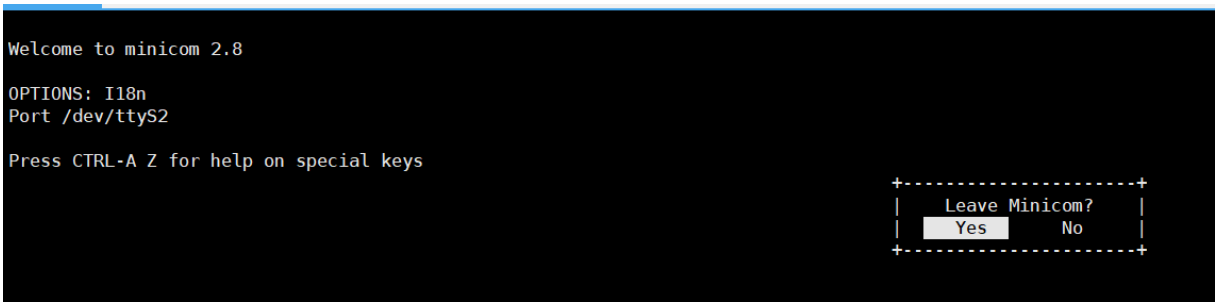
设置完后按回车键退出端口配置界面，选择 Save setup as dfl 保存为默认配置，如下：



选择 Exit 退出配置模式，进入串口通信界面，在 A 开发板输入 szbaijie 时，B 开发板会显示输入的信息，在 B 开发板输入 szbaijie 时，A 开发板会显示输入的信息，测试通过。



退出 minicom：首先按下“Ctrl+A”组合键，再按“X’键，选择”Yes”即可退出。



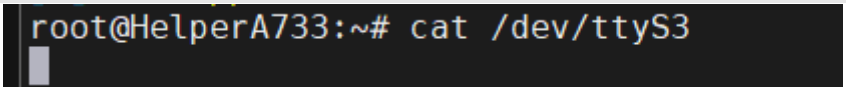
3.1.3 UART 串口测试

在 1.3 节图中的为⑫UART3 和⑪UART4 接口。串口测试采用两块开发板连接，测试发送和接受的效果；串口连接接线：GND——GND,RX——TX,TX——RX；也就是 RX 和 TX 需要交换接线。以 UART3 为例，测试步骤如下：

1、测试开发板 1 接收数据，开发板 2 发送数据。UART3 在设备下生成的节点为 ttyS3，所以发送和接受都用 ttyS3。

开发板 1 上运行以下命令，等待数据接收

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



在开发板 2 上运行以下命令，发送数据

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

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

此时看开发板 1 就接收到开发板 2 发送过来的数据

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

2、测试开发板 1 发送数据，开发板 2 接收数据。

只需在开发板 2 上先运行上边第一条命令，再去开发板 1 运行上边第二条命令即可

3.1.4 网口测试

在 1.3 节图中的⑭为千兆网接口。用普通网线连接开发板的以太网口。首先查看开发板有无 ip，命令如下：

```
$ ifconfig
```

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

在开发板中 ping 地址，有数据产生证明网络和接口测试通过。

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

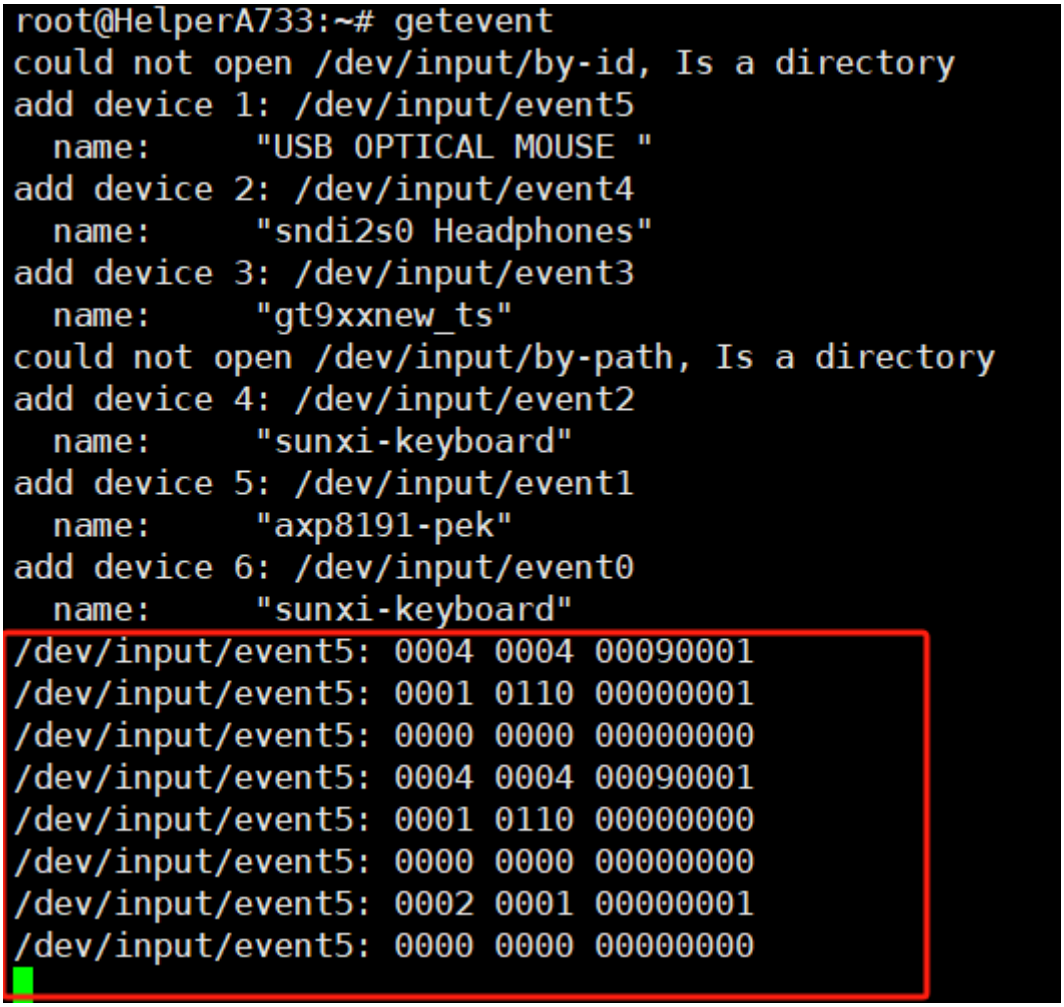
```
root@HelperA733:~# ping www.baidu.com
PING www.a.shifen.com (183.2.172.17) 56(84) bytes of data.
64 bytes from 183.2.172.17 (183.2.172.17): icmp_seq=1 ttl=54 time=8.27 ms
64 bytes from 183.2.172.17 (183.2.172.17): icmp_seq=2 ttl=54 time=8.29 ms
64 bytes from 183.2.172.17 (183.2.172.17): icmp_seq=3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

注：此处 ping 的是国内网站，可根据自己情况 ping 不同地址

3.1.5 USB-HOST 口测试

在 1.3 节图中的⑥为开发板的 2 个 USB-HOST 接口，插上鼠标或者键盘后，在 Xshell 输入命令，滑动鼠标或者在键盘上按任意功能键，会有数据产生。命令如下：

```
$ getevent
```



其中底板标号 DN_U2.0 对应的 USB-HOST 接口使用需要先输入指令运行脚本，再输入 getevent 查看是否有数据产生。命令如下：

```
$ usb2.0_use.sh
```

不使用时执行如下命令：

```
$ usb2.0_del.sh
```

注：执行了 usb2.0_use.sh 后，开发板 adb 调试功能将会关闭

3.1.6 相机测试

在 1.3 节图中的㉖为相机接口，连接好相机再使用脚本命令测试，此脚本拍照后会在屏幕静态显示一张照片，照片“yuv.jpg”会保存在当前目录下。命令如下：

```
$ photo_csi.sh
```

3.1.7 TF 卡接口测试

在 1.3 节图中的⑬为 TF 卡接口。插入 TF 卡后，在 Xshell 输入命令，会在终端显示出 TF 卡设备和容量。命令如下：

```
$ fdisk -l
```

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

3.1.8 麦克风，耳机，喇叭测试

在 1.3 节图中的⑧为耳机接口、⑨为喇叭接口、⑩为麦克风。在 Xshell 输入如下命令后测试，输入命令后会录音 3 秒后自动播放，出现声音则正常。命令如下：

```
$ sound_aplay.sh
```

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

注：ubuntu 系统下耳机和喇叭同时插入都会有声音，建议同一时间只插入一个。

3.1.9 蜂鸣器测试

在 1.3 节图中的④是蜂鸣器。在 Xshell 串口输入如下命令，控制蜂鸣器的打开和关闭

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

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

3.1.10 触摸屏测试

在 1.3 节图中的⑮支持 RGB/LVDS/MIPI 复用接口液晶屏。开机后可在界面点击应用或者滑动进行触摸测试。

触摸屏幕时就会产生触摸数据，则测试通过。命令如下：

```
$ getevent
```

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

3.1.11 WIFI 测试

开发板开机后可在 Xshell 中串口会话输入下图命令，检测到 WIFI，说明测试通过。命令如下：

```
$ nmcli device wifi list
```

```
root@HelperA733:~# nmcli device wifi list
IN-USE  BSSID              SSID                MODE   CHAN  RATE        SIGNAL  BARS  SECURITY
        9E:47:82:16:D3:15  --                  Infra  1     270 Mbit/s  100           WPA1 WPA2
        9C:47:82:10:D3:15  baijiekeji          Infra  1     270 Mbit/s  100           WPA1 WPA2
        9C:47:82:10:D3:16  baijiekeji          Infra  36    405 Mbit/s  100           WPA1 WPA2
        9E:47:82:16:D3:16  --                  Infra  36    405 Mbit/s  100           WPA1 WPA2
        90:B9:42:BA:B0:0F  ChinaNet-bzhX       Infra  11    130 Mbit/s  94            WPA1 WPA2
        90:B9:42:BA:B0:10  ChinaNet-bzhX-5G    Infra  40    270 Mbit/s  94            WPA1 WPA2
        92:B9:42:7A:B0:10  --                  Infra  40    270 Mbit/s  94            WPA1 WPA2
```

3.1.12 4G 模块测试

在 1.3 节图中的②为 SIM 卡槽，㉚为开发板 4G 模块选用 EC20 模块（注：需要另外购买）；在开发板上插上 4G 模块和 SIM 卡并固定好 4G 模块，保证 4G 模块不会松动，开发板上电，准备就绪后按以下步骤执行：与 4G 有关的脚本有两个，一个是启动 4G，一个是删除 4G，启动 4G 后 ifconfig 会出现 ppp0 与 IP 地址，且能 ping www.baidu.com 说明测试通过。

1、首先确保相关软件已经安装，如果没安装则执行一下命令安装并重启系统

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

2、重新启动系统后，执行一下命令启动 4G，第一次连接 4G 需要等待一段时间获取 ip

```
$ 4g_set.sh
```

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

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

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

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

3、删除 4G

```
$ 4g_del.sh
```

删除成功后用 ifconfig 查看网络时没有 ppp0。

3.1.13 蓝牙测试

使用手机蓝牙与开发板蓝牙进行通信测试。开发板中与蓝牙有关的脚本有两个，一个是启动并设置连接、一个启动通信端口。Xshell 终端运行设置蓝牙脚本，命令如下：

```
$ bluetooth_set.sh
```

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

注：进入 bluetooth 后可输入 help 查看相应的命令信息。

1、搜索蓝牙设备

```
$ scan on
```

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

搜索到蓝牙设备 szbaijie_bluetooth，获取到其手机蓝牙的设备 ID：F0:C4:2F:39:6E:55，如下所示：

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

找到对应设备 id 后可以停止搜索，命令如下：

```
$ scan off
```

如果搜索报错，请检查蓝牙设备 hci0 是否打开，如果没有打开，请重新执行 bluetooth_set.sh 脚本。检查命令如下：

```
$ hciconfig -a
```

hci0 正常打开时如下图

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

2、配对蓝牙

搜索到手机蓝牙设备 ID 后，输入配对命令短暂后会出现确定界面，在开发板命令行输入 yes。

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

注：F0:C4:2F:39:6E:55 是蓝牙设备对应的 ID，在 scan on 命令后出现

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

这时候手机上也会弹出配对请求，点击配对。配对成功后的界面如下：

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

配对成功后输入 exit 退出蓝牙连接界面。命令如下：

```
$ exit
```

3、连接蓝牙

配对退出后在开发板输入蓝牙端口注册命令，命令如下：

```
$ bluetooth_rfcomm.sh
```

输入以上命令后会出现等待界面，此时手机上下载一个蓝牙串口 app，在 app 中点击连接配对过的开发板蓝牙，连接后就开发板会显示连接成功终止等待。此时手机蓝牙串口 app 和开发板可以进行数据的接收和发送。连接成功如下所示

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

注：没下载蓝牙串口 app 连接时，会一直处于等待状态

4、蓝牙通信

（1）接受数据

在 Xshell 终端输入如下命令，通过手机蓝牙串口 app 中发送数据，终端会显示相应的数据。

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

（2）发送数据

在 Xshell 终端输入 echo 发送数据后，在手机蓝牙串口 app 中也能显示。命令如下：

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

3.1.14 背光调节

背光的大小可以通过以下命令进行调节，value 为数字 0~255，一般情况下 0 为最暗的背光，255 为最亮的

```
$ set_brightness.64 value
```

使用上面命令设置背光为 150，可以在屏幕上看到背光的变化

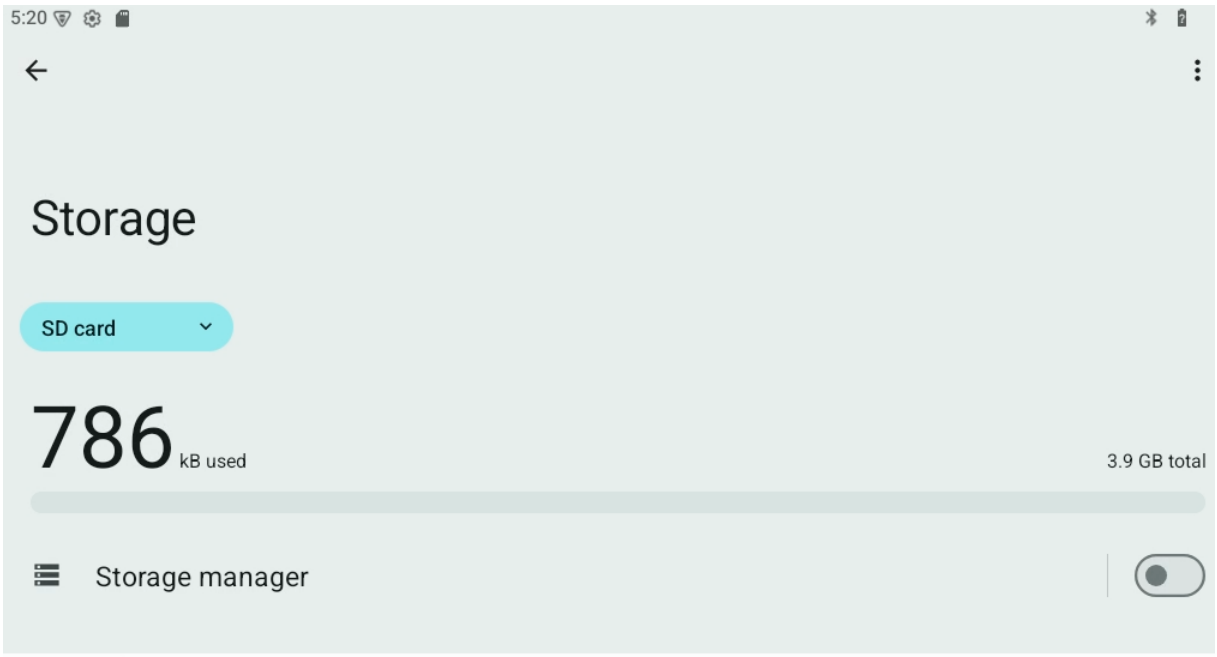
```
root@HelperA733:~# set_brightness.64 150
Brightness set to 150 (max: 255)
root@HelperA733:~#
```

3.2 开发板 android 系统功能测试

android 测试与 ubuntu 测试有很多相同的地方，相同的地方的测试方式如下：485 测试、网口测试、USB-HOST 口测试、触摸屏测试。

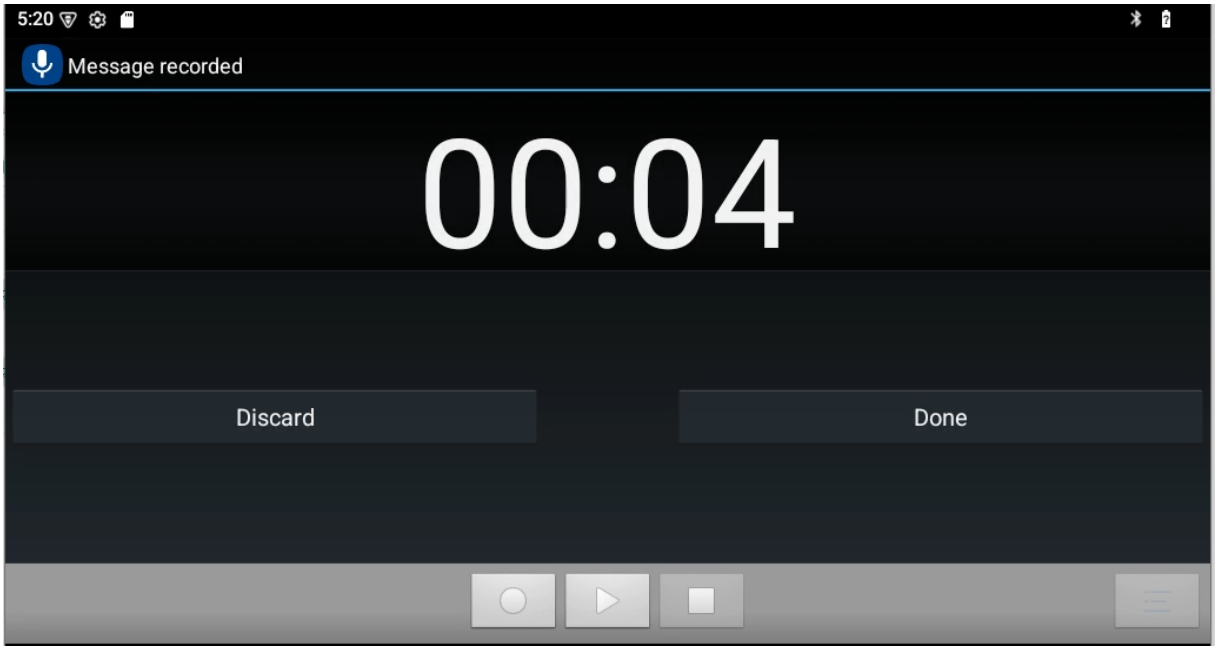
3.2.1 TF 卡测试

在 1.3 节图中⑬为 TF 卡接口。插入 SD 卡后点击“设置“中的“存储“，里面会显示出 TF 卡存储容量。



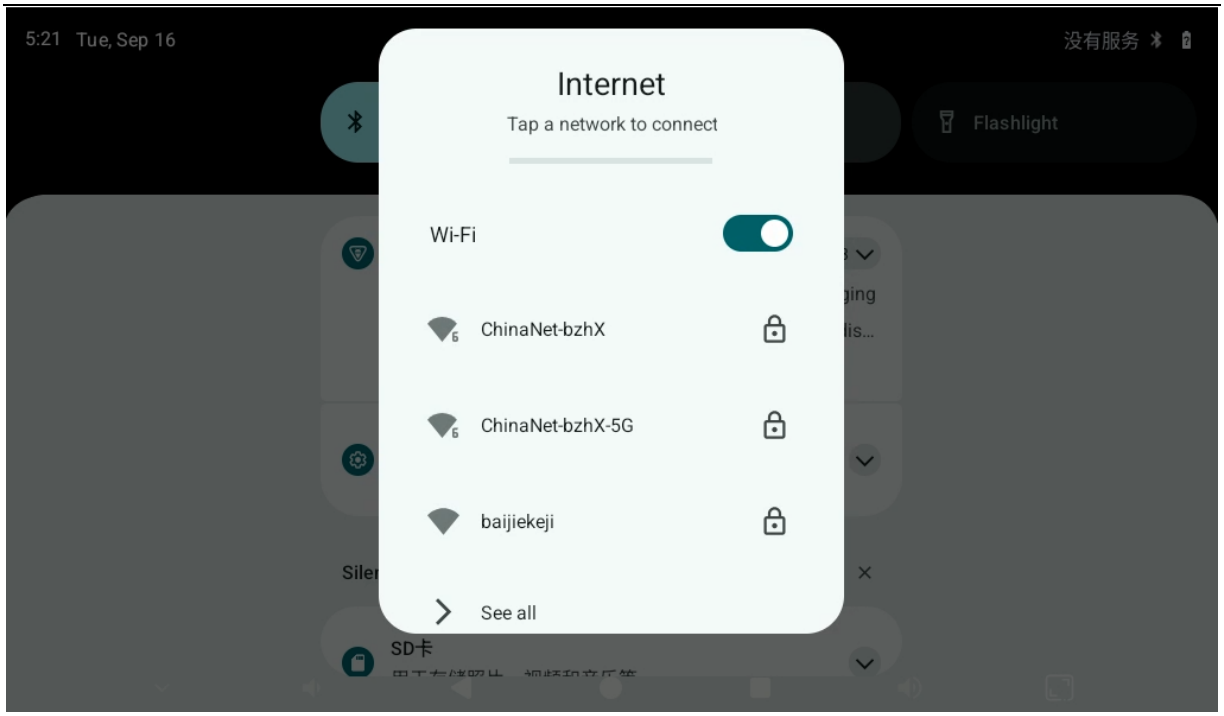
3.2.2 麦克风，耳机，喇叭测试

在 1.3 节图中的⑧为耳机接口，⑨为喇叭接口，⑩为麦克风。我们使用录音机 app 可以同时测试麦克风，喇叭等接口，如下图所示:



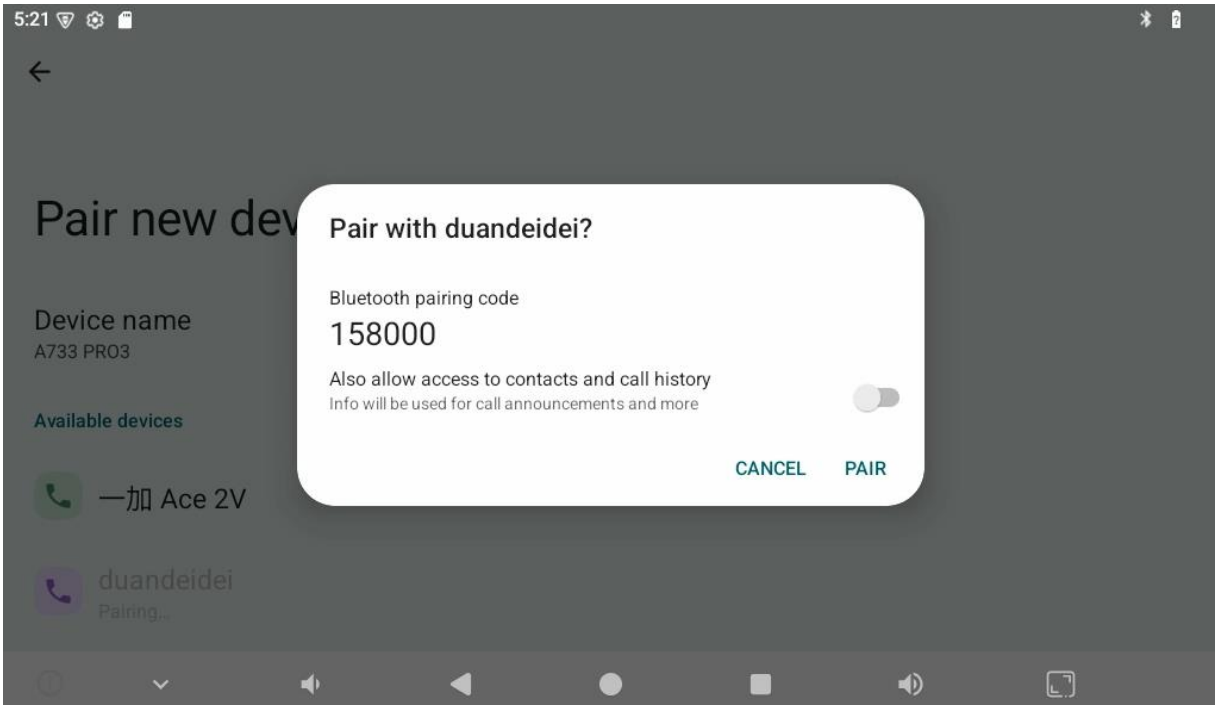
3.2.3 WIFI 测试

开发板开机后可在界面下拉点开 WIFI 和关闭 WIFI。



3.2.4 蓝牙测试

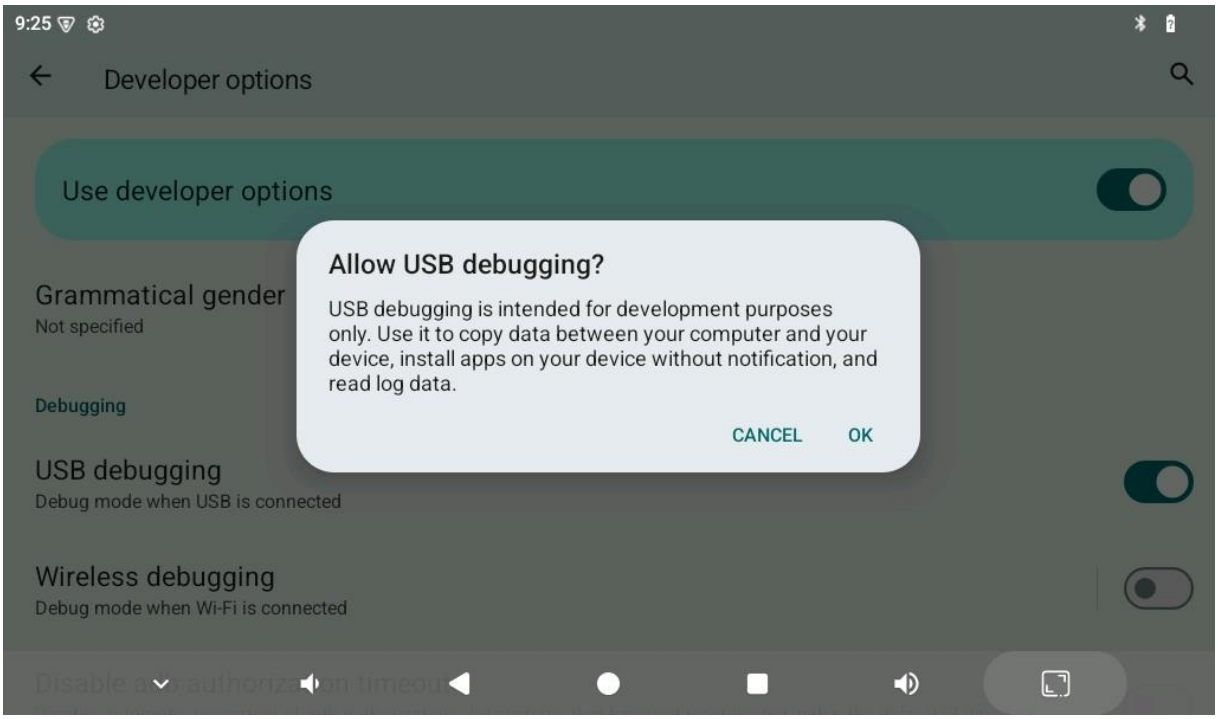
开发板开机后可在界面下拉点“与新设备配对”搜索到蓝牙设备进行配对，配对图如下：

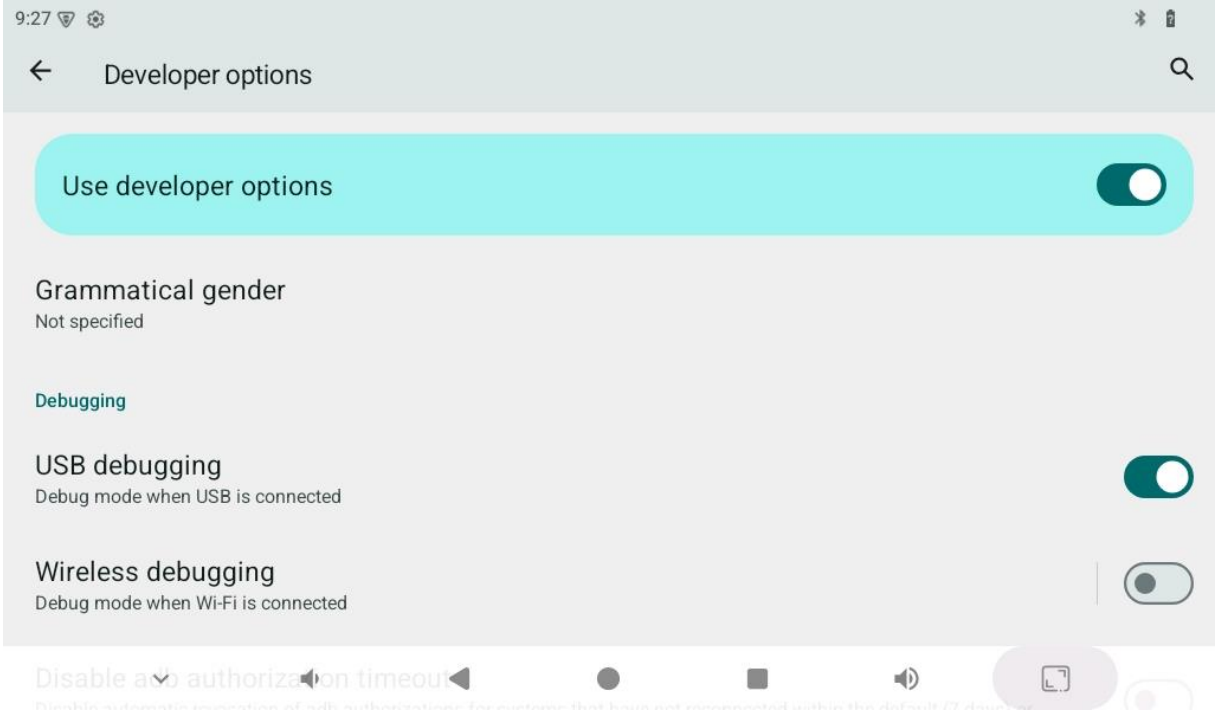


3.2.5 USB-OTG 口测试

在 1.3 节图中的⑤为 USB-OTG。系统开机后在界面打开：设置->系统->高级->开发者选项->USB 调试（默认为打开的）。当你打开时会提示如下图所示信息，选择”是“就可以进行安卓的”ADB 调试“了，说明测试通过。这个口也可以作为烧录口，具体烧录步骤请查看第 4.3 节。

注：关于安卓的 ADB 调试请查看第 2.3 节内容





当开发板用 adb shell 不能连接安卓时（可以直接连接时，跳过这一步），adb devices 查看设备也没有设备号，这时连接安卓需要先在开发板的串口终端下打开 OTG 功能,操作如下：

```
$ su

$ cat /sys/devices/platform/soc@3000000/10.usbc0/usb_device
```

```
console:/ # cat /sys/devices/platform/soc@3000000/10.usbc0/usb_device
device_chose finished, otg disabled!
console:/ #
```

显示 device chose finished 后就可以用 adb 连接安卓。这时在电脑的 terminal 用命令：adb devices 就能显示设备号。

3.2.6 USB-HOST 口测试

参考 3.1.5 章节描述，要注意的是 android 系统下没有 usb2.0_use.sh 和 usb2.0_del.sh 这两个脚本，所以要使用底板标号 DN_U2.0 对应的 USB-HOST 接口的话需要执行以下命令

```
$ su
```

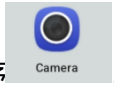
```
$ echo 354 > /sys/class/gpio/export

$ echo out > /sys/class/gpio/gpio354/direction

$ echo 0 > /sys/class/gpio/gpio354/value

$ cat /sys/devices/platform/soc@3000000/10.usbc0/usb_host
```

不使用时执行如下命令。

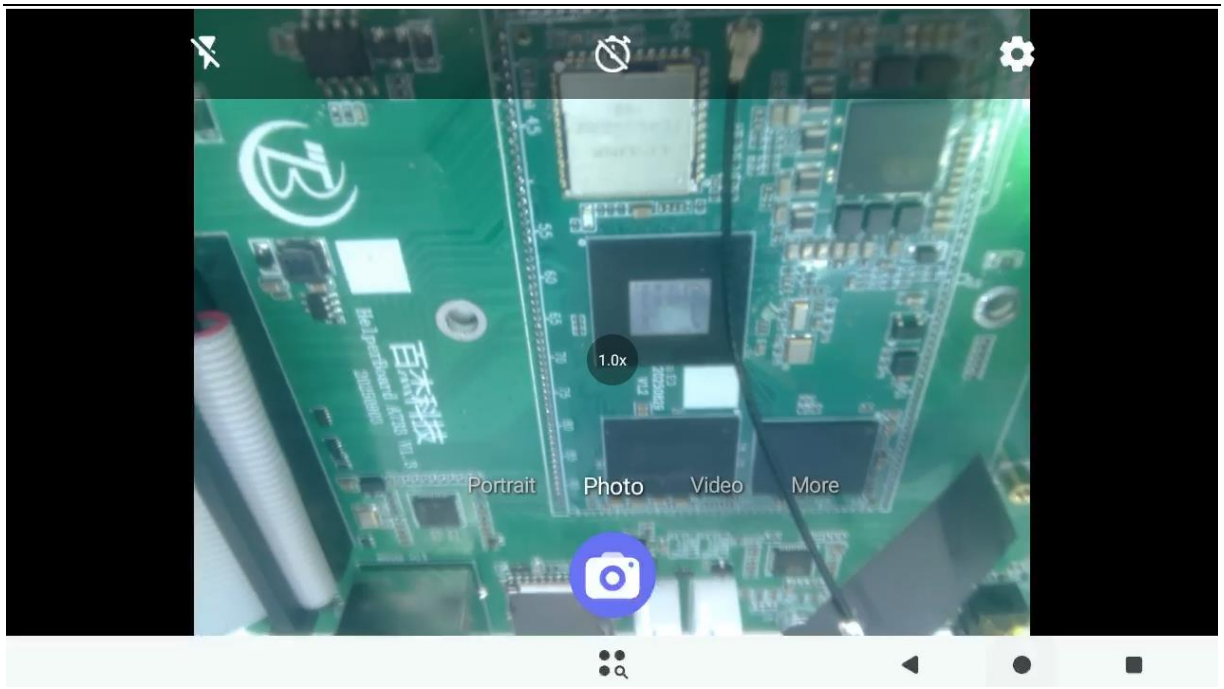
```
$ su

$ echo 1 > /sys/class/gpio/gpio354/value

$ cat /sys/devices/platform/soc@3000000/10.usbc0/usb_device
```

注：使用 DN_U2.0 这个接口作为 HOST 的话，开发板 adb 调试功能将会关闭

3.2.7 相机测试

打开应用程序界面，点击“Camera“程序 

,进入相机界面即可拍照或者录像，如下图：



注：需插入相机设备相机应用才会出现

4 软件工具使用

4.1 vmware 开发环境搭建

4.1.1 硬件需求

内存：大于等于 12G（Android 编译大于等于 16G 最优）

存储：大于等于 500G

4.1.2 软件需求

为了简化开发流程，我们特地制作了 ubuntu 20.04 的虚拟机镜像，已经在里边安装了必要的软件，如果不使用我司提供虚拟机需要下载如下软件

Ubuntu16.04 下载

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

Unubtu18.04 下载

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

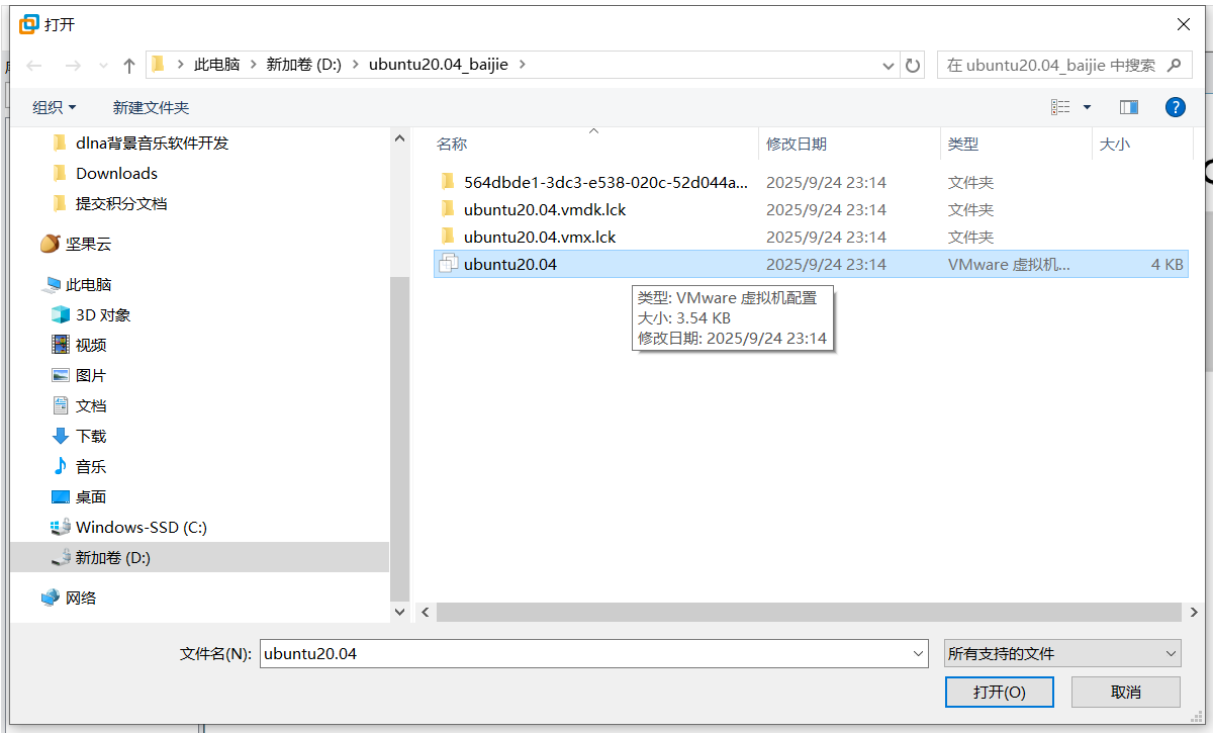
Ubuntu20.04 下载

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

安装虚拟机软件 vmware17 很简单，在网盘共享里有相应的安装文件，请自行安装，虚拟机镜像在网盘共享目录里：**tools/vmware17/helperboard_ubuntu20.04.zip**，解压缩之后即可使用。下边介绍在 vmware17 里配置开发环境的过程。

打开 vmware 17 软件，点击菜单：**文件->打开**，选择解压后的虚拟机镜像文件：**ubuntu20.04.vmx**，如下图：

注：打开虚拟机后，默认内存是 4G



默认的配置是 2 个 CPU，3 核心数，12G 内存，500G 硬盘，其中有 24G 交换分区。大家可以根据自己的需要做适当的调整，比如增大内存、CPU 核心数。我用的 CPU 是 i7 6700K，各位在运行虚拟机的时候可能会出错，可选择拷贝虚拟机，然后自动配置成自己对应的 CPU 应该就可以了。

注：

1、虚拟机 ubuntu20.04 中用户名密码都是：szbaijie。

2、安装虚拟机后不要更新，不然 Python 版本发生变化导致不能编译。

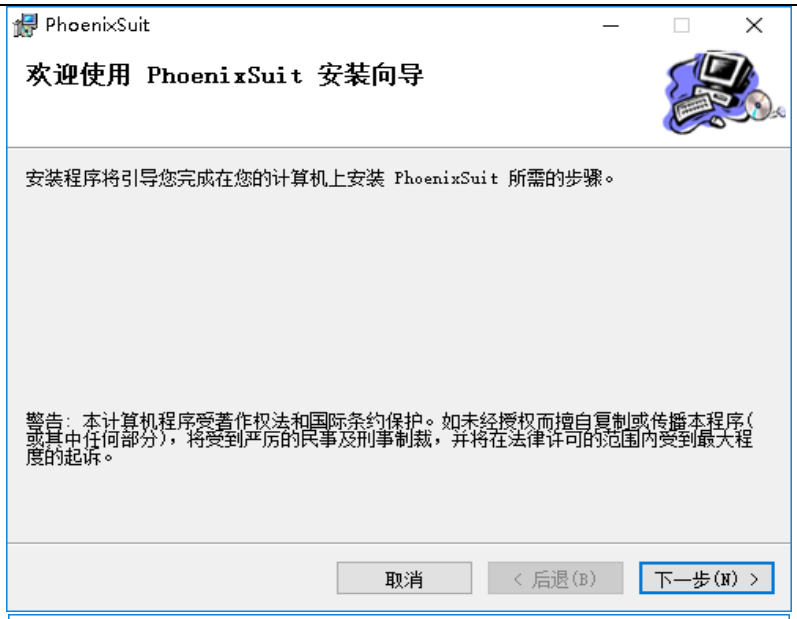
然后点击 vmware 17 工具栏的  按钮，即可启动 HelperBoard 虚拟机了。

若出现以下界面点击“获取所有权”即可

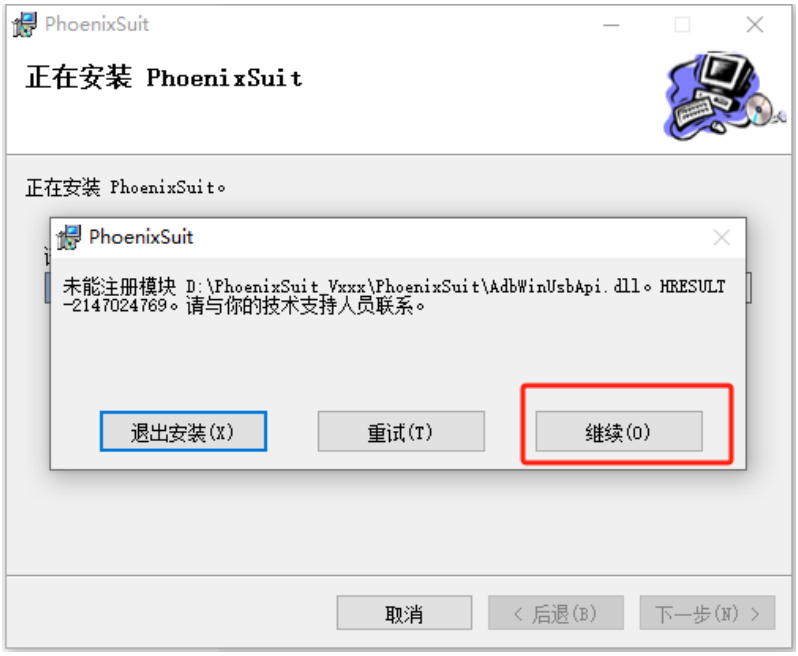
接着点击“我已复制该虚拟机”即可进入系统

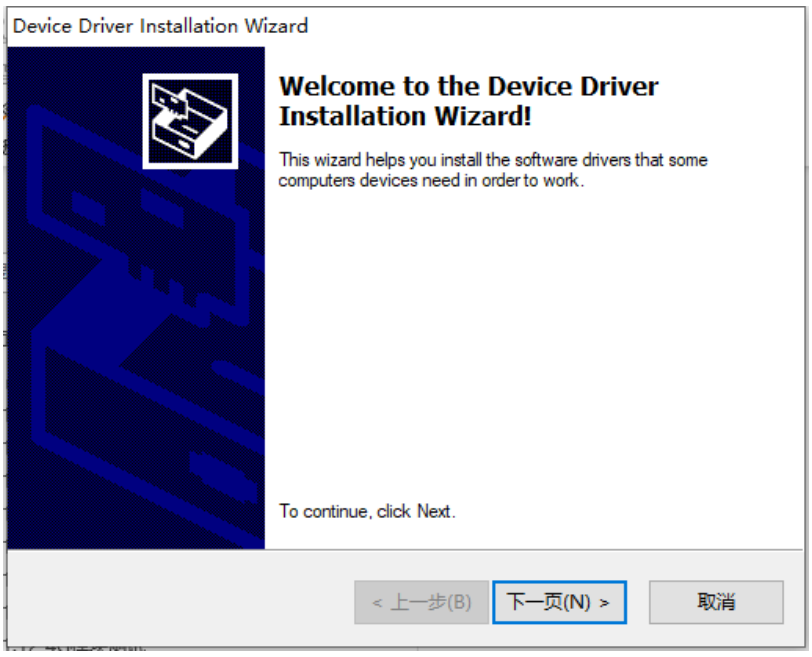
4.2 安装 PhoenixSuit

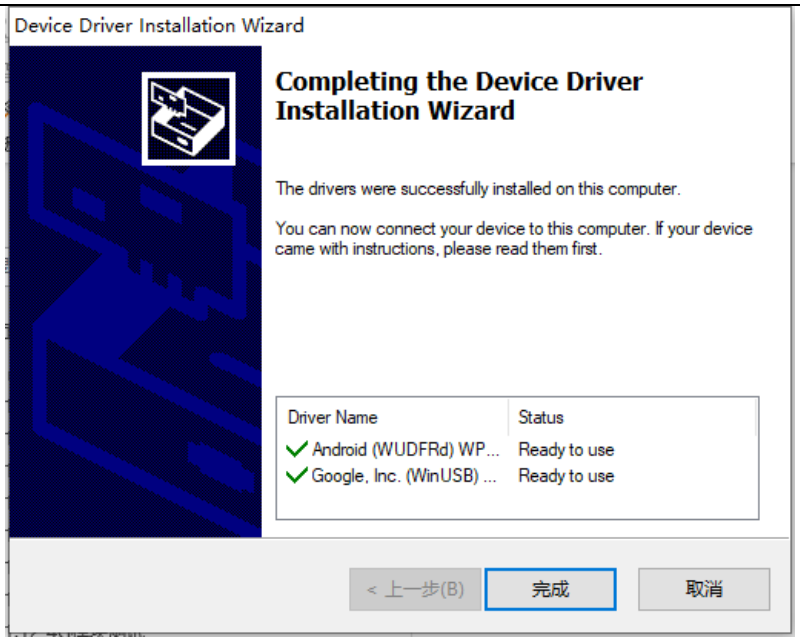
下载安装网盘共享目录里：tools/windows_burning_tool/PhoenixSuit_CN.msi。安装过程中会出现安装驱动的界面，依次进行：



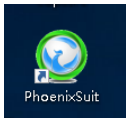
若出现下图所示，点击继续即可


PhoenixSuit
正在安装 PhoenixSuit
正在安装 PhoenixSuit。
PhoenixSuit
未能注册模块 D:\PhoenixSuit_Vxxx\PhoenixSuit\AdbWinUsbApi.dll。HRESULT -2147024769。请与你的技术支持人员联系。
退出安装(X)
重试(T)
继续(O)
取消
< 后退(B)
下一步(N) >


Device Driver Installation Wizard
Welcome to the Device Driver Installation Wizard!
This wizard helps you install the software drivers that some computers devices need in order to work.
To continue, click Next.
< 上一步(B)
下一页(N) >
取消



安装完后会在桌面生成 PhoenixSuit 图标



4.3 烧写开发板固件

按照 4.2 节要求安装 PhoneixSuit，打开软件会出现界面如下图

注：烧录固件不同的版本可能步骤略有不同，此文将说明 2.0.1 版本方式。

解压下载的固件，点击一键刷机，再点击浏览，选择要用的固件文件，比如 a733_dragonboard_pro3_uart0.img，选择好固件后，烧录方式选择“**分区擦除升级**“。

烧写步骤：

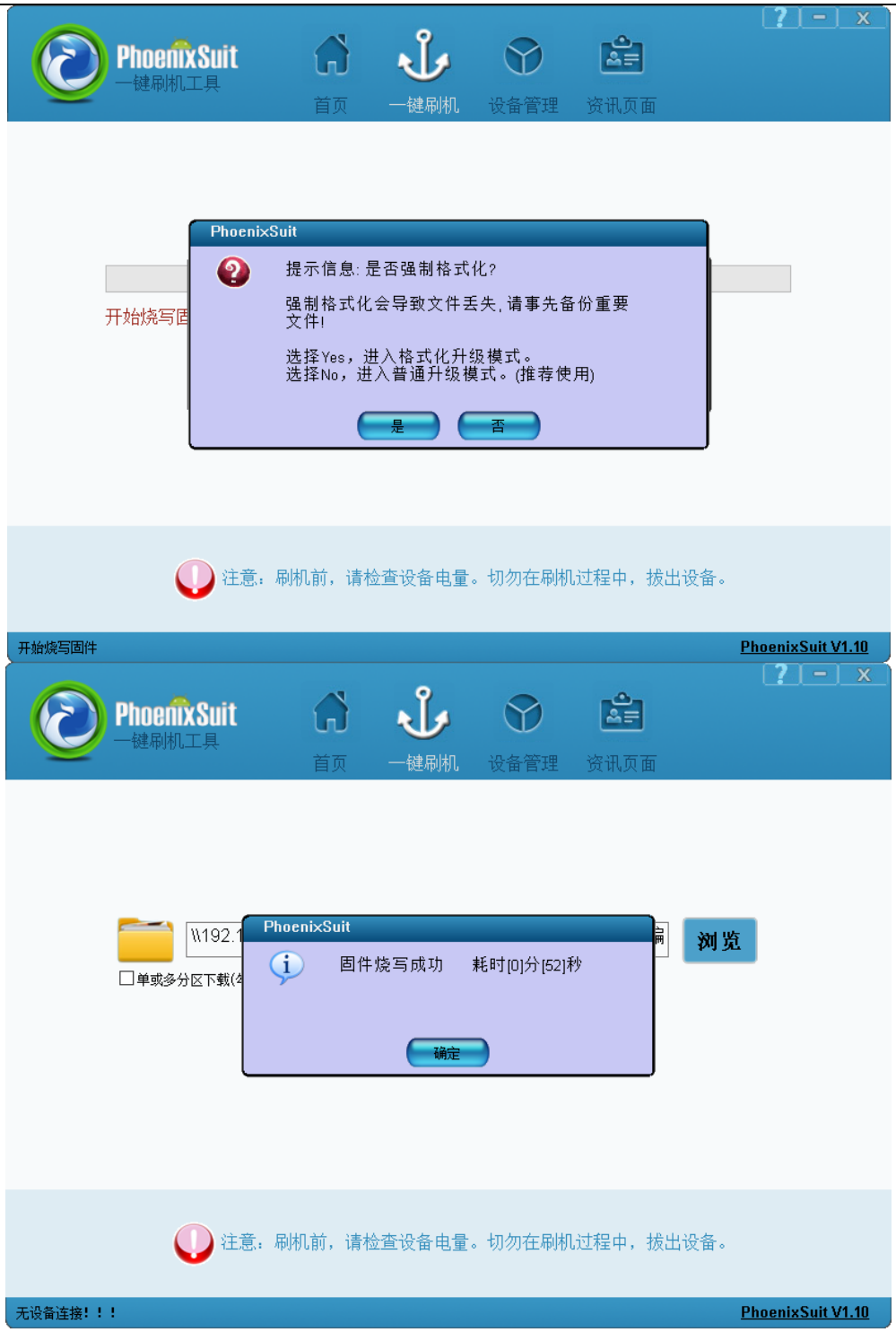
1、插上开发板的烧写 USB-OTG 口（**在 1.3 节图中的⑤**）

2、上电启动开发板

3、待系统启动完成后先按住烧录键 FEL（**在 1.3 节图中的㉒**）不要松开，再按复位键 RST（**在 1.3 节图中的㉑**），松开时先松开复位键 RST，再松开烧录键 FEL，成功进入烧写模式时软件如下图状态。

注：烧录过程不要断开开发板的电源，否则会导致烧录固件失败，等待几分钟后烧录成功，如下图示，重启开发板进入系统，第一次启动可能会久一点。

低于 1.19 版本的 PhoneixSuit 进入烧录状态时会出现确认界面，点击“**是**”即可。烧录结束后也会有提示。如下：

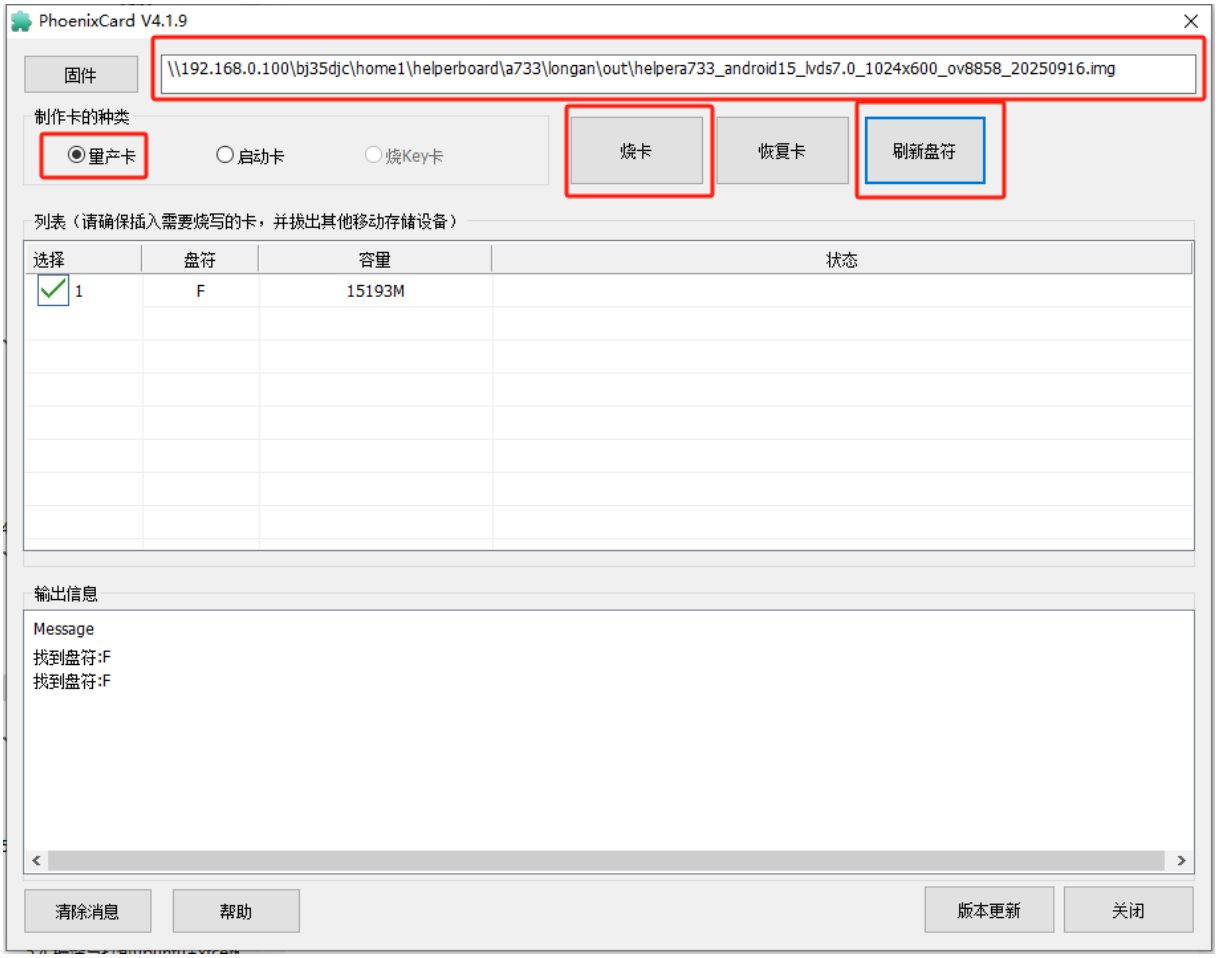


注：由于全志下载程序的原因，有时候会出现进度条走了一部分就不动了，遇到这种情况只能重新启动板子，再重复上边的过程。

当烧写成功、开机成功后，正常出现系统界面。

4.4 PhoenixCard 制作 SD 卡烧录盘

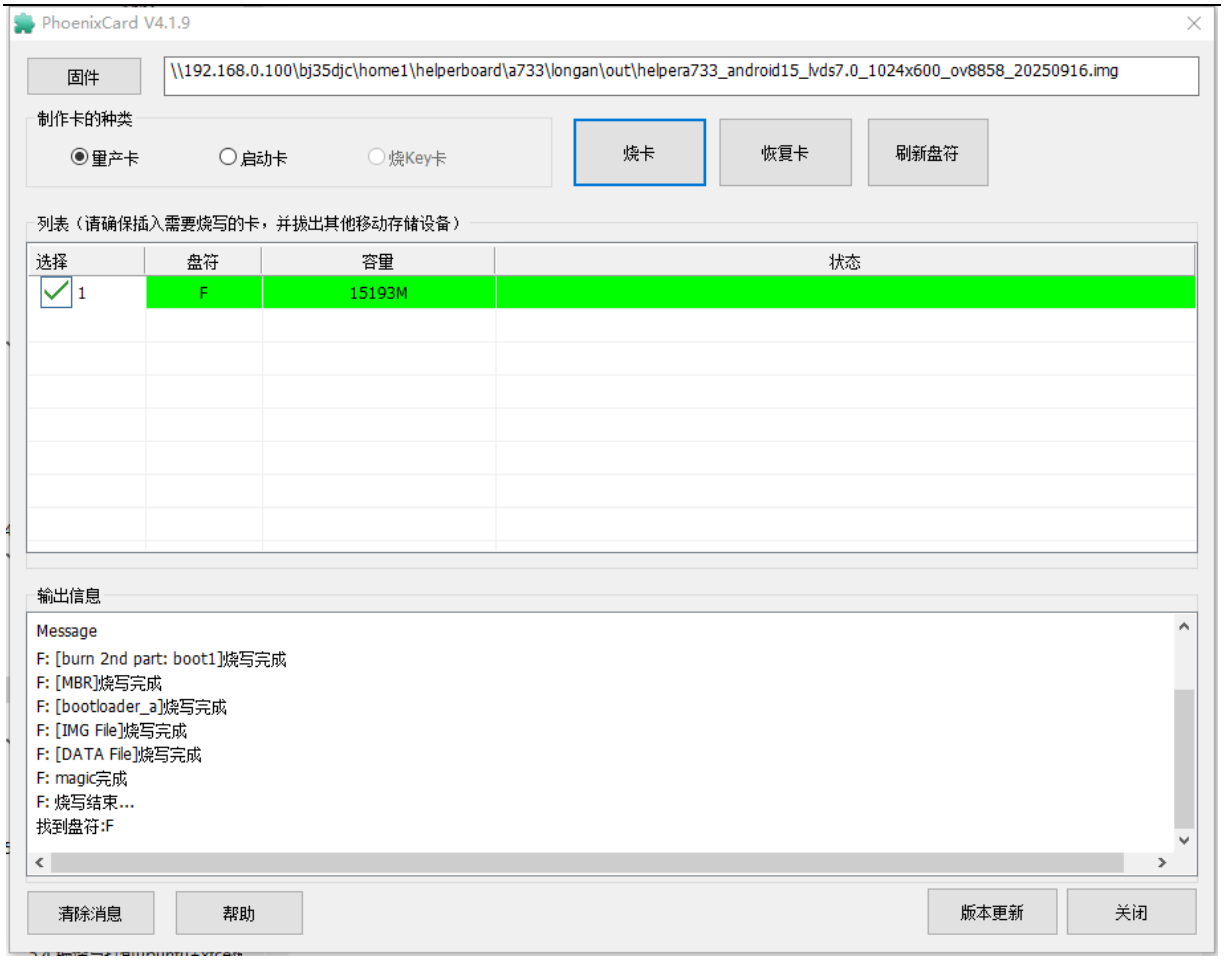
网盘共享目录：**tools/windows_burning_tool/ PhoenixCard_V4.1.9_20190227.zip**，下载并解压后进入到安装目录找到 PhoenixCard.exe 可直接运行程序。界面如下：



鼠标单击“刷新盘符”就可以自动搜索读卡器的盘符，然后选择你需要制作的烧录盘。

当找到盘符之后，鼠标单击“固件”就会弹出 img 文件选择对话框，用户选择需要烧写的 img 固件。

当选择好对应的固件 img 的时候，将烧写模式切换为“量产卡”，再单击“烧卡”按钮进行卡的烧写，就可以将你插入的盘制作成 SD 卡烧录系统盘，下图则表示烧录成功；烧录成功后 SD 卡就可以插在开发板上烧写程序到开发板上。



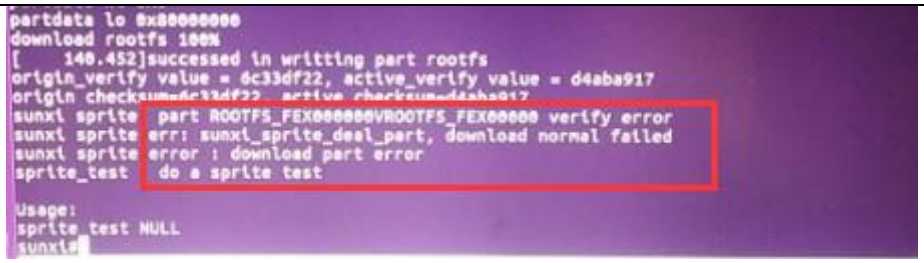
注：当不需要用作 SD 卡启动盘的时候，单击“恢复卡”可将 SD 卡启动盘格式化为普通卡。

```
partdata hi 0x0
partdata lo 0x1729000
sparse: bad magic
[07.077]successed in writting part boot
origin_verify value = 10830c90, active_verify value = 10830c90
[07.339]successed in verify part boot
[07.342]successed in download part boot
[07.346]begin to download part rootfs
partdata hi 0x0
partdata lo 0x39000000
sparse: bad magic
```

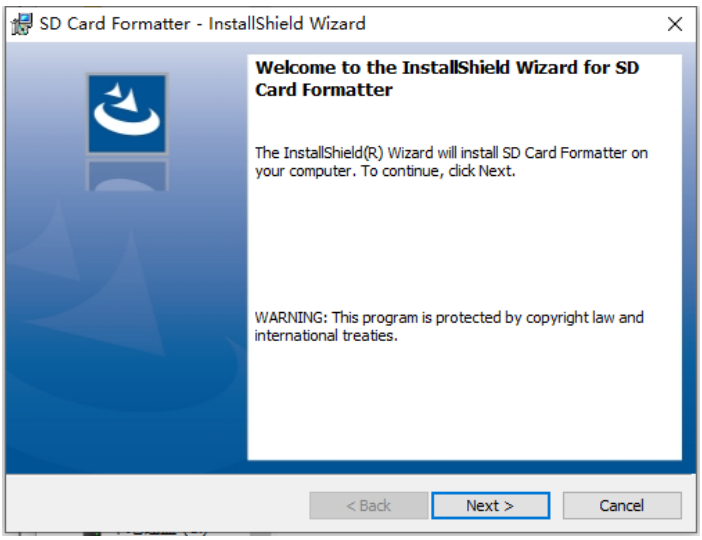
注：如果用卡烧写失败，请使用 SDFormatter 来格式化 sd 卡，如果格式化后还不行，换卡试一试。

4.5 卡烧写失败用 SDFormatter 格式化内存卡

用卡烧写固件时出现下图错误不能正常启动时，需要使用 SDformatter 格式化内存卡，然后用 PhoenixCard 重新烧写镜像到内存卡，再用内存卡烧写到开发板上

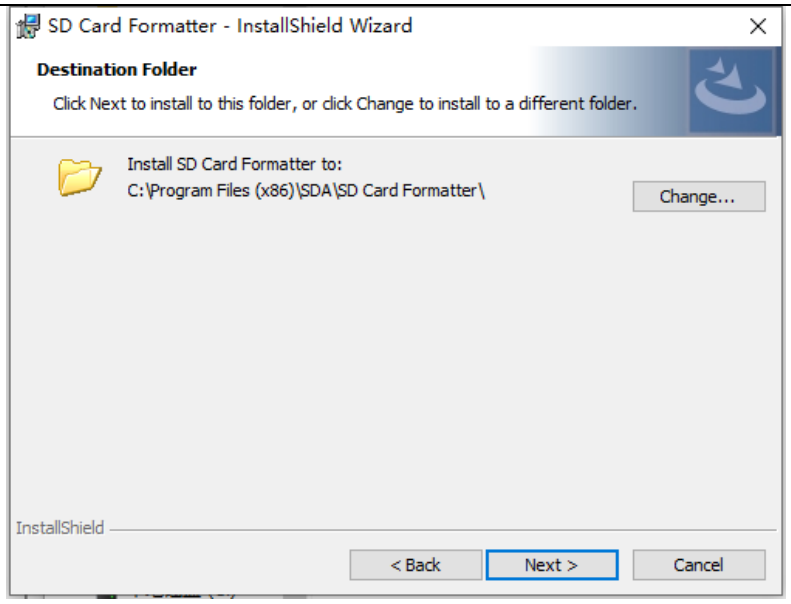


网盘共享目录里: **tools/SD_card_formatterl/SDCardFormatterv5_WinEN.zip**，下载解压后进入到目录找到 SD Card Formatter 5.0.2 Setup EN.exe 双击文件进行安装。安装流程如下图所示:

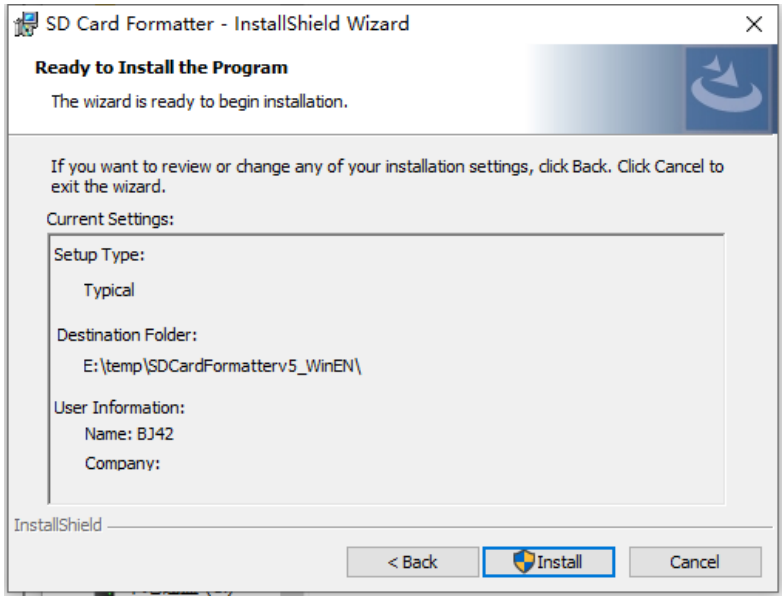


点击“Next”

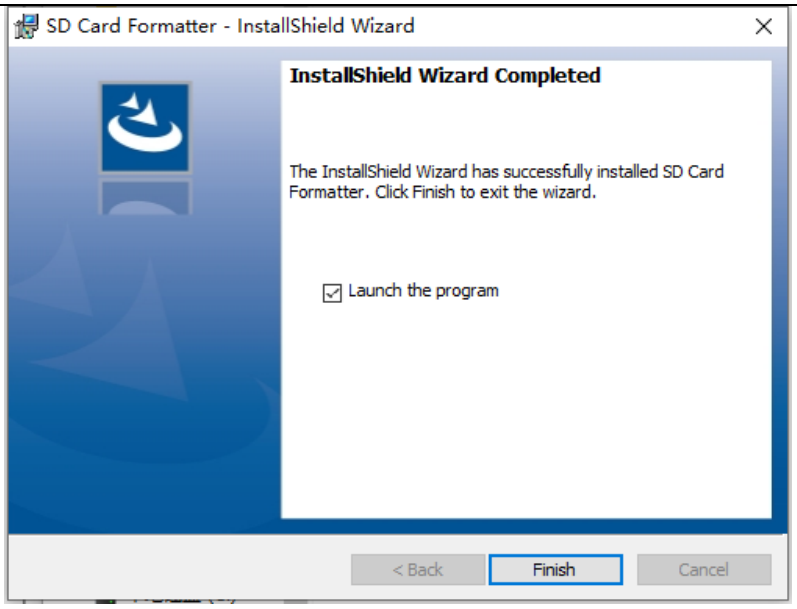
选择“I accept”，点击“Next”



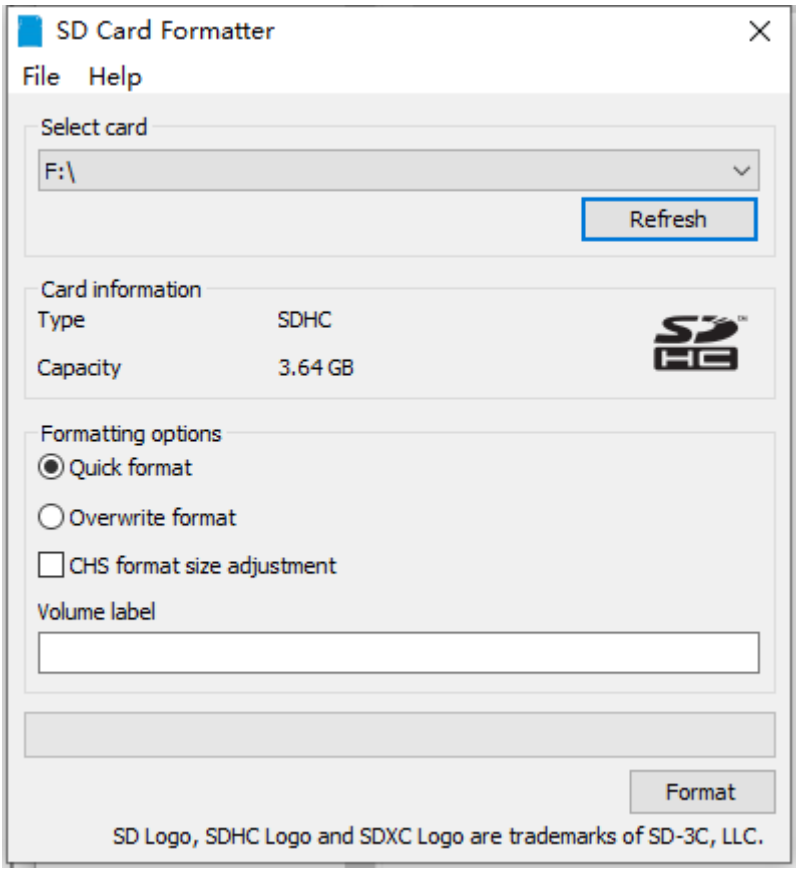
选择安装路径



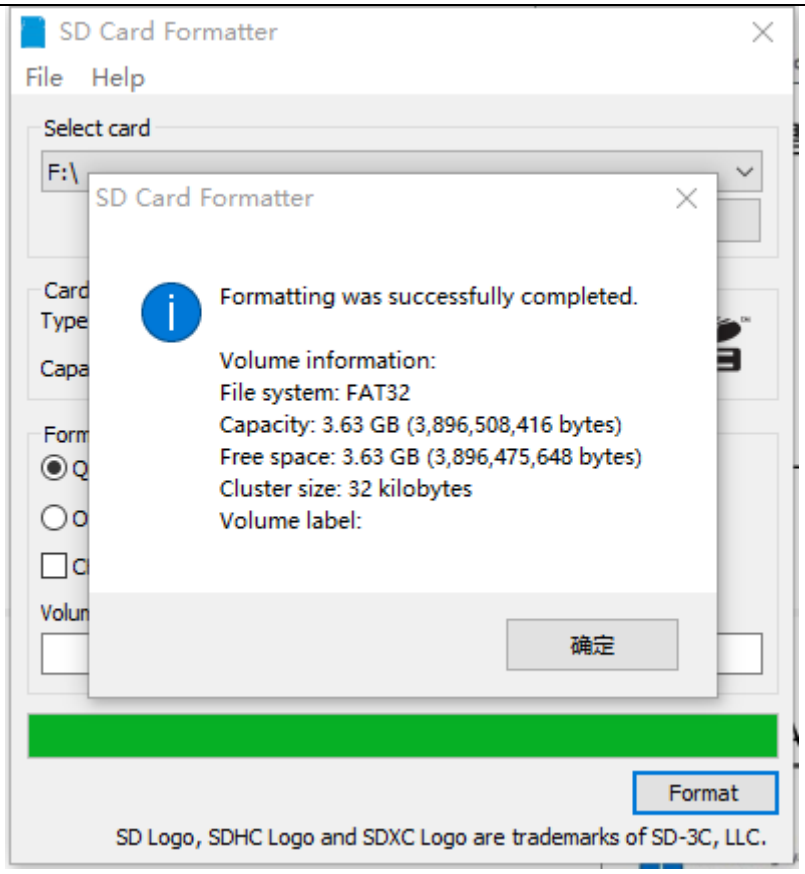
点击“Install”进入安装



安装完成后点击“Finish”启动

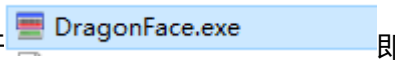


插入内存卡后会自动读取，如果没有读取到，单击“Refresh”重新读取，读取到内存卡后单击“Format”按钮，使用默认选项格式化。中途出现错误请检查接口是否损坏，内存卡是否插好，是否出现松动情况。出现以下界面则格式化成功。

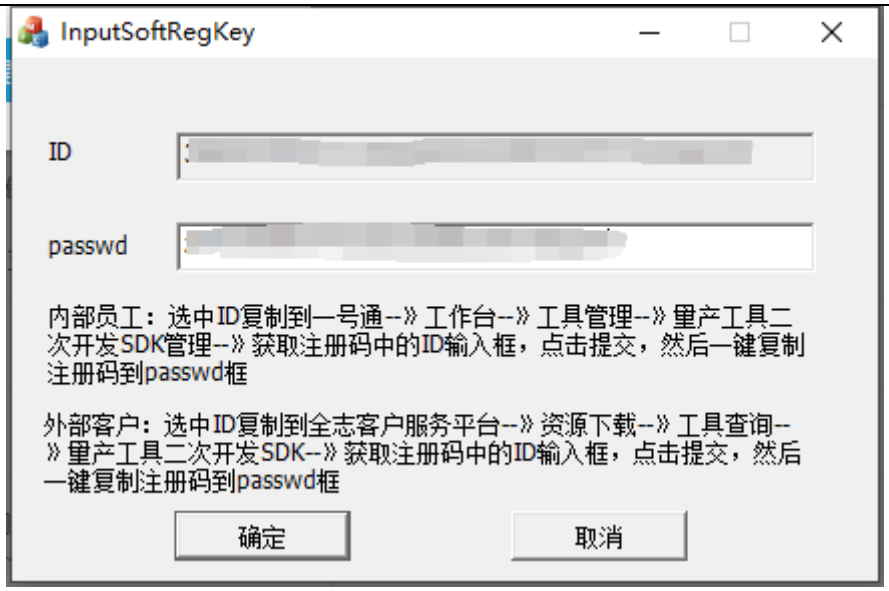


注：如果格式化后还不行，请换卡尝试！

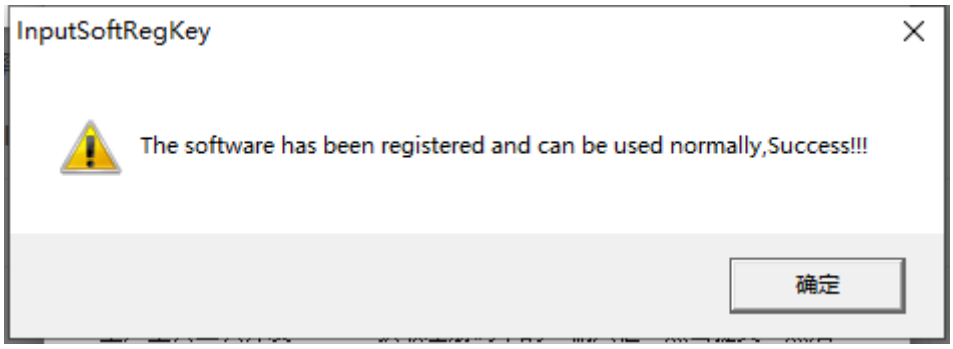
4.6 Dragonface 的使用

目前 dragonface 只支持 android 系统的修改，从网盘中下载并解压 **tools/ firmware_modify _tool/DragonFace_V4.1.6.zip** 到本地。双击打开 

即可进入注册界面，将 ID 复制给客服获取注册码。



注册成功提示如下



4.6.1 更改 Android 开机 logo

打开 Dragonface：

（1）点击“配置”选择对应配置

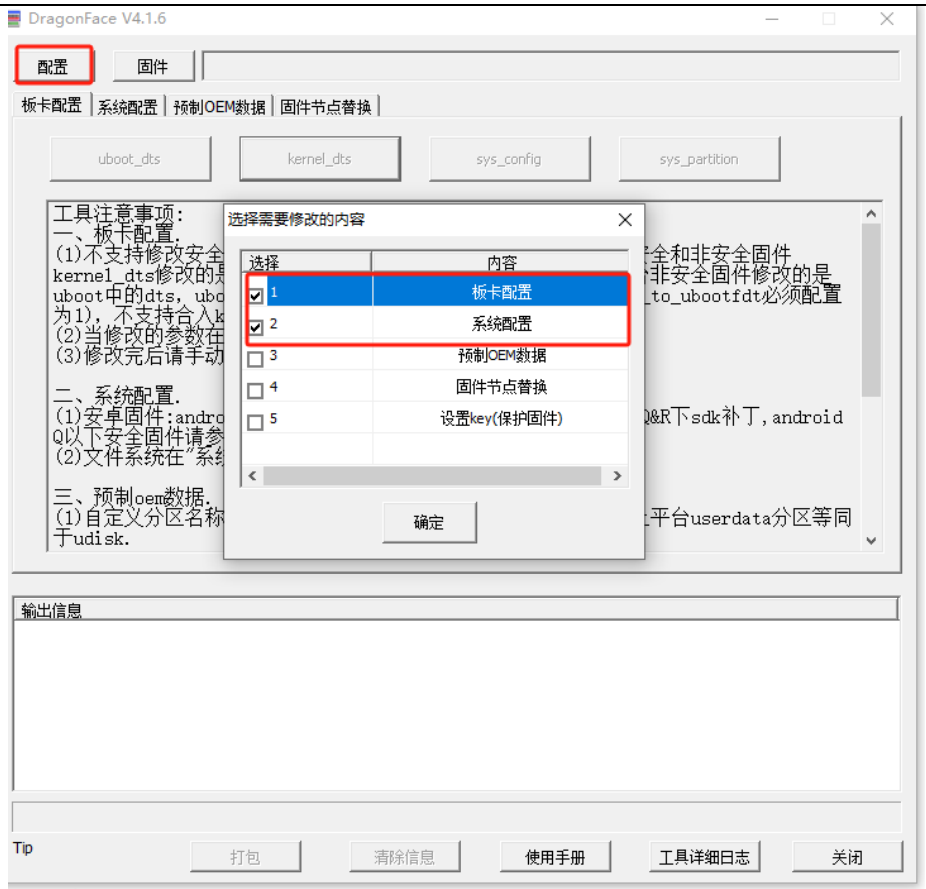
（2）点击“固件”选择要更改开机 logo 的固件，等待固件加载

（3）固件加载完毕后选择“系统配置”和“画面音乐”

（4）点击“替换 Bootlogo”，选择需要替换的开机 logo

（5）替换完后点击左下角“打包”，选择保存路径即可

注：选择图片时默认显示 bmp,png,jpg 格式的图片，需要其他格式可在右下角选择 All files


DragonFace V4.1.6
配置
固件
板卡配置
系统配置
预制OEM数据
固件节点替换
uboot_dts
kernel_dts
sys_config
sys_partition
选择需要修改的内容
选择
内容
1 板卡配置
2 系统配置
3 预制OEM数据
4 固件节点替换
5 设置key(保护固件)
确定
输出信息
Tip
打包
清除信息
使用手册
工具详细日志
关闭


DragonFace V4.1.6
配置
固件
板卡配置
系统配置
预制OEM数据
固件节点替换
基本设置
画面音乐
APK
高级设置
Rootfs
替换BootLogo
替换kernelLogo
替换开机动画
替换开机视频
替换开机音乐
替换关机Logo
替换关机音乐
Bootlogo
Kernellogo
开机动画
关机logo
开机视频
壁纸
pixels: 600 * 420
百杰科技
BAIJIE TECHNOLOGY
输出信息
14:25:12 解析super分区中...
14:25:20 super分区解析成功
14:25:51 系统分区解析成功
14:25:57 固件打开成功...
100%
打包
清除信息
使用手册
工具详细日志
关闭

4.6.2 更改开机动画

打开 Dragonface：

（1）点击“固件”选择要更改开机动画的固件，等待固件加载

（2）固件加载完毕后选择“系统配置”和“画面音乐”

（3）点击“替换开机动画”，选择需要替换的开机动画压缩包（zip 格式）

（4）替换完后点击左下角“打包”，选择保存路径即可

注：开机动画 bootanimation.zip 压缩包的制作过程见 8.2 节

4.6.3 增加内置 APK

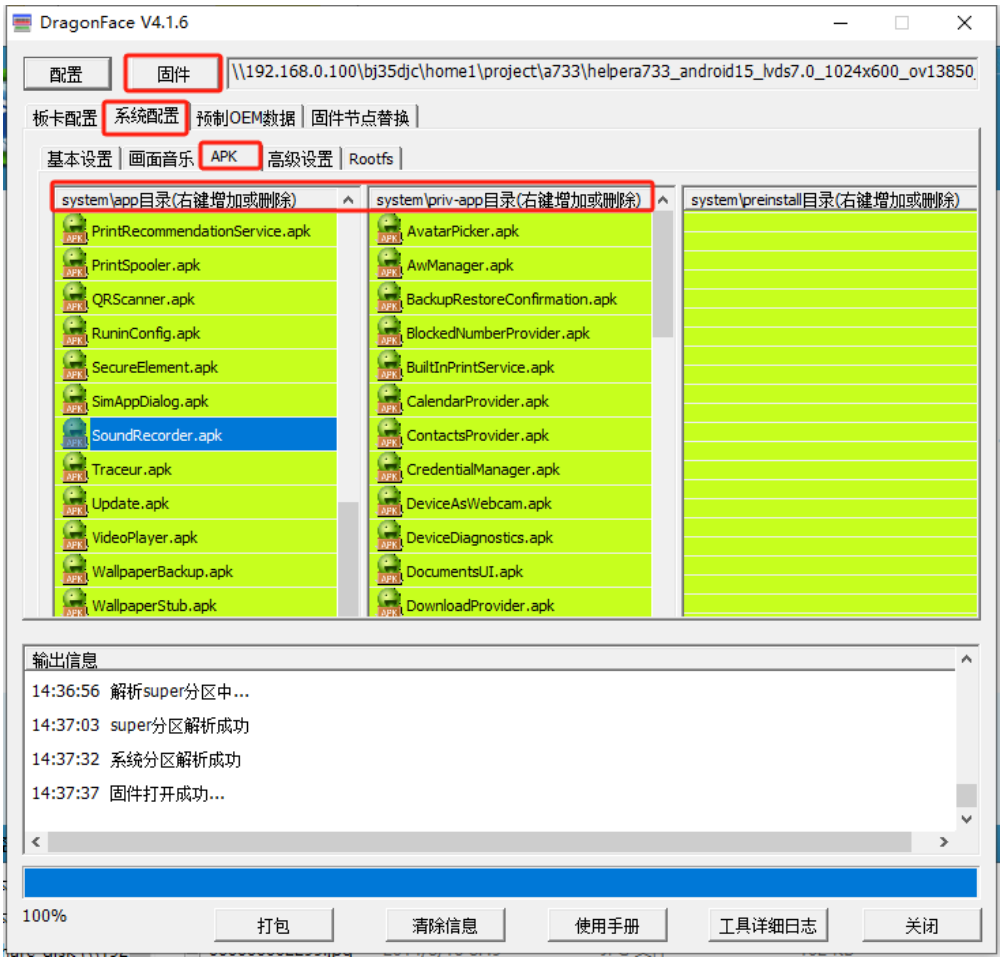
打开 Dragonface：

（1）点击“固件”选择要更改内置 APK 的固件，等待固件加载

（2）固件加载完毕后选择“系统配置”和“APK”

（3）鼠标右键单击预览区的 apk，增加或者删除

（4）修改完后点击左下角“打包”，选择保存路径即可



5 制作 ubuntu 系统固件

已经编译好的 ubuntu+qt 系统放在 **firmware/gz** 目录里,名称为：**helpera733_ubuntu24.04_qt6.8.3_lvds7.0_1024x600_20251204.tar.gz**，解压完后就可以使用 PhoenixSuit 烧写到板子。如需要在板子上运行自己的 qt 应用程序可以直接参考第 6 章： “远程运行与调试 Qt 应用程序” 。

5.1 编译 linux 内核

在 sources 目录中下载 helpera733_android15_repo_v1.1_20251204_1952.tar.gz 和 repo.tar.gz 到本地，里面包含 HelperA733 的 u-boot 代码、内核代码和板级配置文件等。把该文件拷贝到虚拟机系统非 root 用户目录下，解压缩该文件，比如解压缩到~/workspace/a733/下边，执行如下命令：

```
$ mkdir -p ~/workspace/a733/
$ cp helpera733_android15_repo_v1.1_20251204_1952.tar.gz repo.tar.gz
~/workspace/a733/
$ cd ~/workspace/a733/
$ tar -xvf helpera733_android15_repo_v1.1_20251204_1952.tar.gz
$ tar -xvf repo.tar.gz -C .repo/
$ .repo/repo/repo sync -l
```

同步完成之后，依次执行如下命令选择对应的开发板配置

注：所有编译都不能使用 root 用户编译，只能使用非 root 用户编译

```
$ cd longan
$ ./build.sh config
```

输入上面命令后需要配置 ubuntu 信息，如下：

1、**platform** 输入“1”， 选择 **linux。**

2、**linux_dev** 输入“2”，选择 **dragonboard。**

3、**ic** 输入“0”，选择 **a733。**

4、**board** 输入“10”，选择 **pro3**，对应 HelperA733 开发板的硬件配置路径。

5、**flash** 输入“0” ，选择 **default。**

6、**devicetree** 输入“1”，选择 **lvds7.0_1024x600_bj。**可按需更改。

7、**baijie_extra** 输入“0”，选择 **extra-qt6.8.3。**可按需更改

8、**baijie_rootfs** 输入“0”，选择 **ubuntu24.04_base.tar.gz。**可按需更改

具体如下图所示：

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

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

注:

All available devicetree 对应不同屏幕的配置，百杰默认五种屏幕，说明如下:

1、hdmi_bj 选择 0，对应百杰 hdmi 屏幕（需购买支持 hdmi 的开发板版本）

2、lvds7.0_1024x600_bj 选择 1，对应百杰 lvds 7 寸屏幕

3、mipi10.1_1200x1920_bj 高清屏选择 2，对应百杰 mipi10.1 寸高清屏

4、mipi10.1_800x1280_bj 选择 3，对应百杰 mipi 10.1 寸屏

5、mipi4.3_480x800_bj 选择 4，对应百杰 mipi 4.3 寸屏

6、mipi5.0_720x1280_bj 选择 5，对应百杰 mipi5 寸屏

7、mipi8_800x1280_bj 选择 6，对应百杰 mipi8 寸屏

All available baijie_extra 对应不同的文件系统补充，百杰默认两种，说明如下:

1、extra-qt6.8.3 选择 0，对应 qt6.8.3 系统补充

2、extra-xfce 选择 1，对应 xfce 桌面系统补充

3、no_extra 选择 2，对应不使用 extra 补充

All available baijie_rootfs 对应不同文件系统，百杰默认两种，说明如下:

ubuntu24.04_base.tar.gz 选择 0，对应 ubuntu24.04 系统（需配合 extra-qt6.8.3）

ubuntu24.04_xfce.tar.gz 选择 1，对应 ubuntu24.04+xfce 桌面系统（需配合 extra-xfce）

此配置步骤只是第一次编译时的配置选项，后期更改设备树或者驱动后无需再配置，直接编译即可。配置选择完后执行如下命令开始编译:

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

编译注意事项：

1、 请使用我们提供的虚拟机 helperboard_ubuntu20.04，否则会编译失败。

2、 同步源码后第一次编译内核完成之后，需要进入 bsp 目录执行以下命令，之后重新编译一遍内核

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

3、选择屏幕配置后会覆盖 device/config/chips/a733/configs/pro3/linux-6.6 目录下的 board.dts 和 device/config/chips/a733/configs/pro3 目录下的 uboot-board.dts 配置，不更改平台（Android，linux）的情况下，修改代码后直接编译即可，不需要重复选择。

4、如需有更改平台，请备份好 board.dts 和 uboot-board.dts，否则会被覆盖，之前修改全无。

5.2 编译 uboot

进入到 uboot，直接编译

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

5.3 编译与打包 ubuntu+qt 文件系统

在 5.1 节执行./build.sh config 配置时，对应的 baijie_extra 选择 extra-qt6.8.3，baijie_rootfs 选择 ubuntu24.04_base.tar.gz

编译打包执行

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

5.4 编译与打包 ubuntu+xfce 文件系统

在 5.1 节执行./build.sh config 配置时，对应的 baijie_extra 选择 extra-xfce，baijie_rootfs 选择 ubuntu24.04_xfce.tar.gz

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

5.5 注意事项

1、烧录后的开发板账户密码如下：

开发板超级用户账号：root

开发板普通用户账号：szbaijie

开发板超级用户和普通用户密码：szbaijie

2、如果修改了 longan 下的代码，需要全部重新编译再打包，不能直接打包，如下所示：

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

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

6 远程运行与调试 qt 程序

6.1 拷贝编译器到虚拟机中

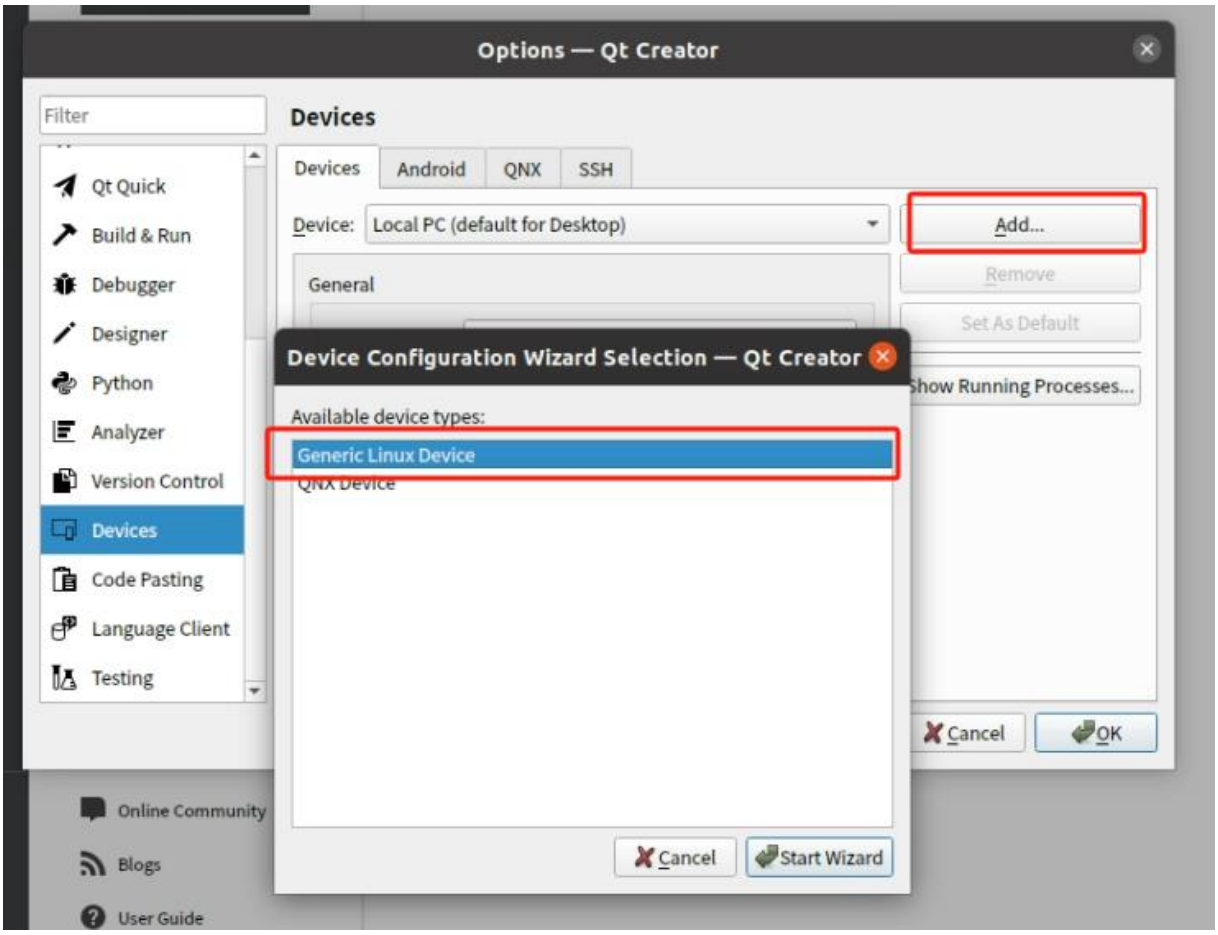
在网盘中 **tools/toolchain_for_linux** 目录里把 **helpera733_ubuntu24.04_aarch64_glibc2.39_gcc13.3_qt6.8.3.tar.gz** 拷贝到虚拟机 helperboard_ubuntu20.04 的/opt/toolchain 目录下，解压文件并设置环境变量,命令如下

```
$ sudo mkdir -p /opt/toolchain
$ cd /opt/toolchain
$ sudo tar -xf helpera733_ubuntu24.04_aarch64_glibc2.39_gcc13.3_qt6.8.3.tar.gz
```

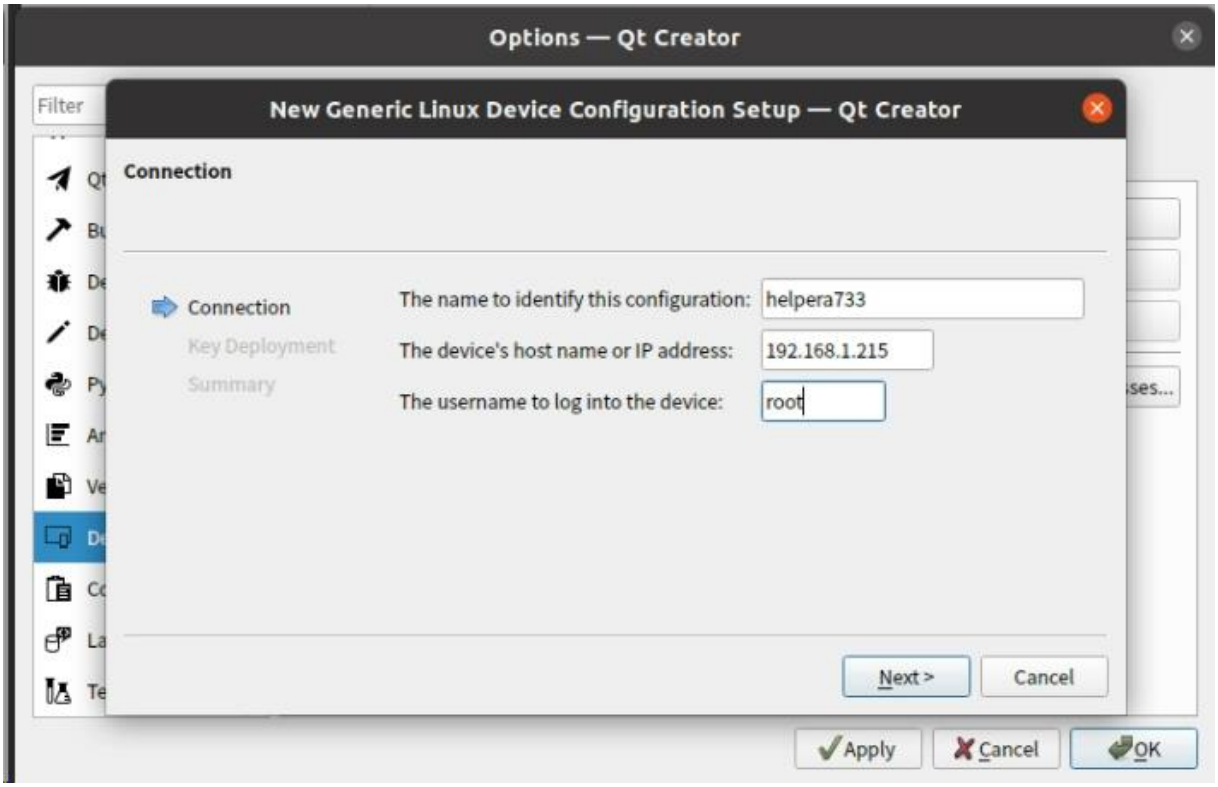
6.2 虚拟机中 qtcreator 环境配置

6.2.1 添加远程设备

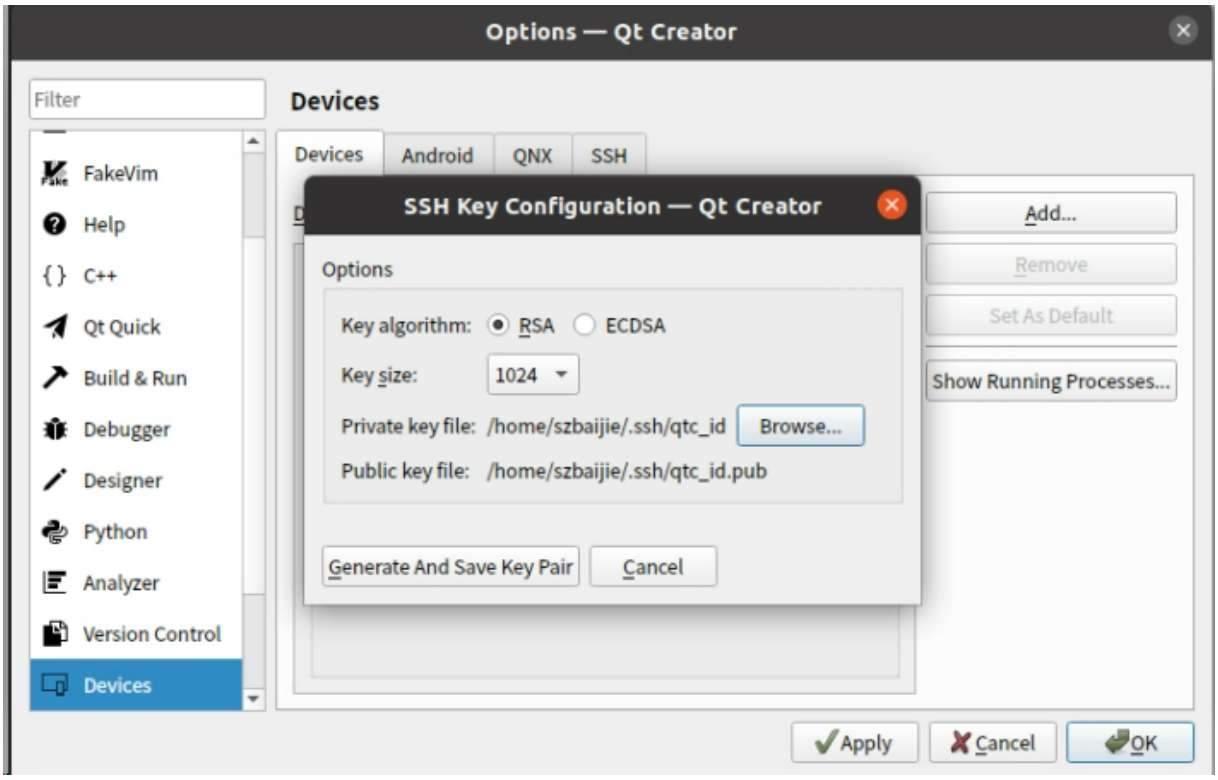
我们需要远程运行程序，还需要在 qtcreator 中添加远程设备。打开我们虚拟机中内置的 qtcreator 软件，然后依次点击 **Tools->Options->Devices->Devices->Add**，新增远程设备。如下所示:



我们需要输入：**开发板用户名、开发板密码、开发板 ip 地址。**如下所示：

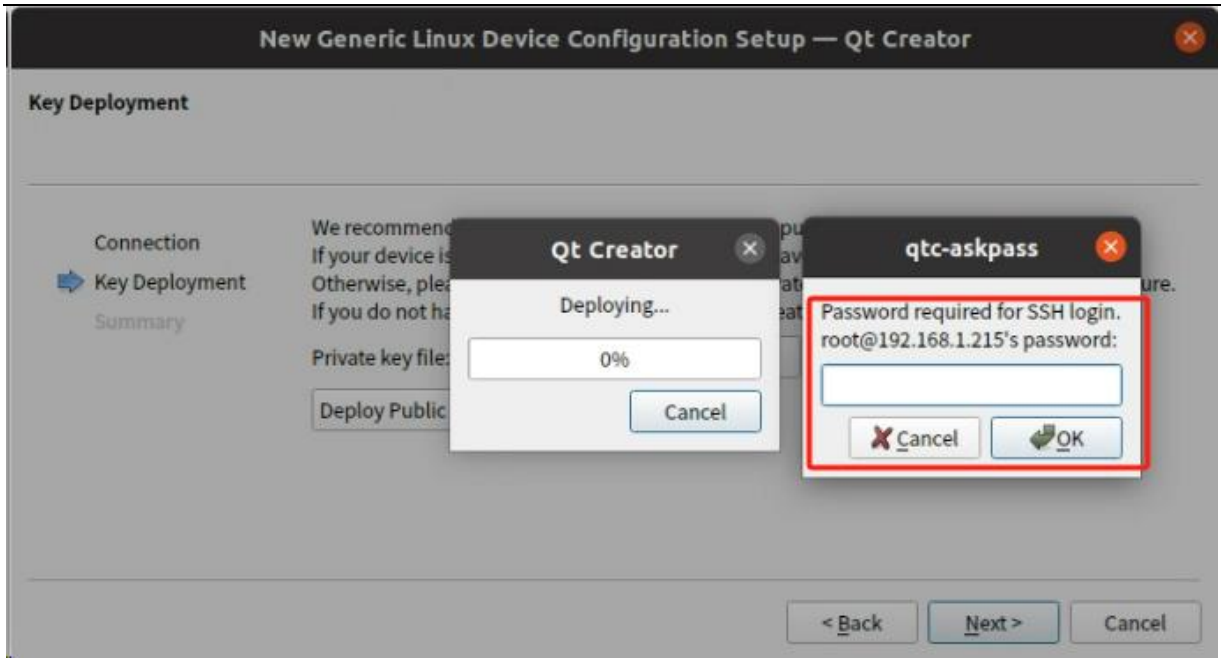


创建新的秘钥



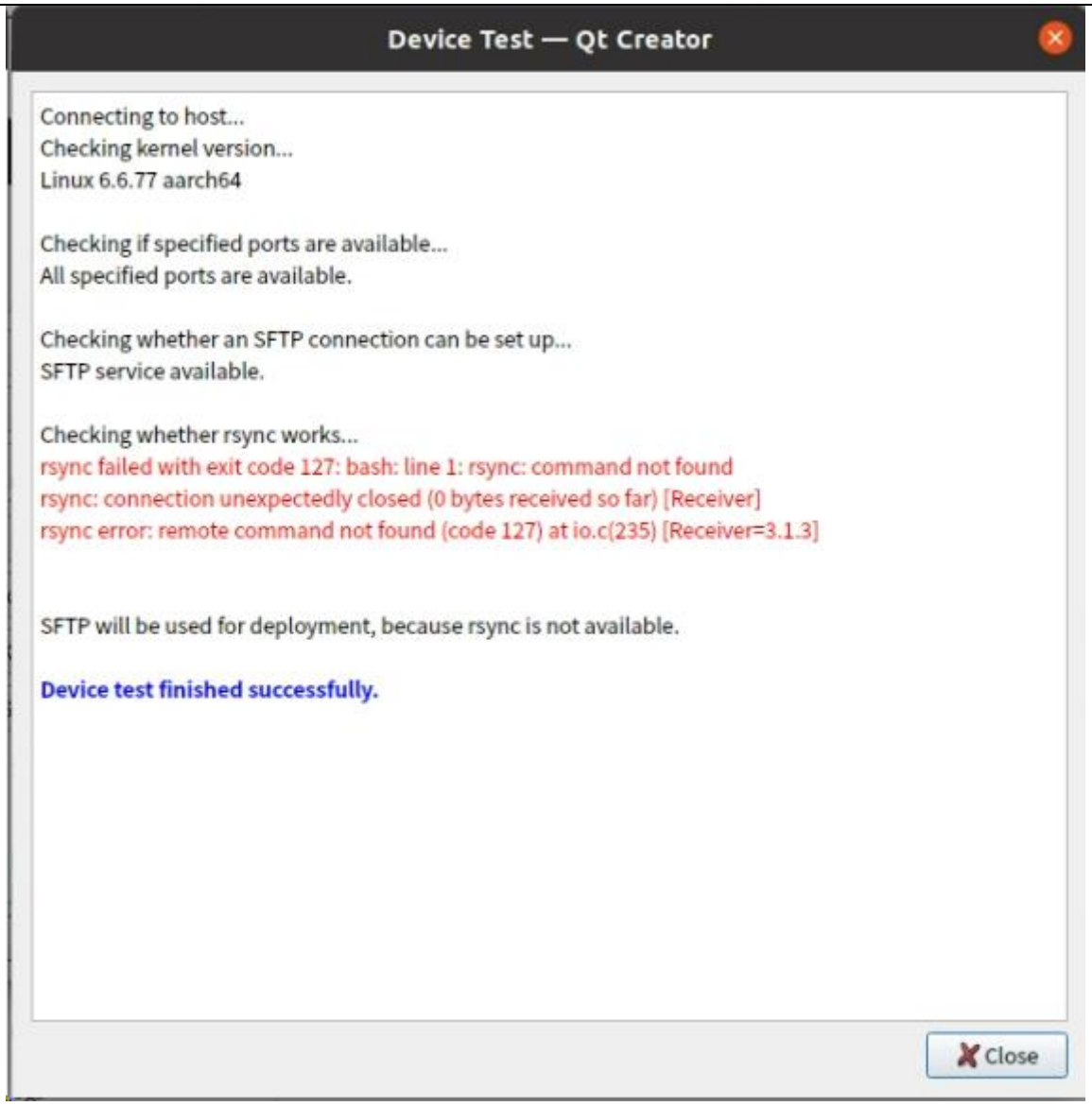
创建完成之后点击部署公钥

输入开发板登录用户密码



部署成功后点击下一步

点击完成，成功连接状态如下



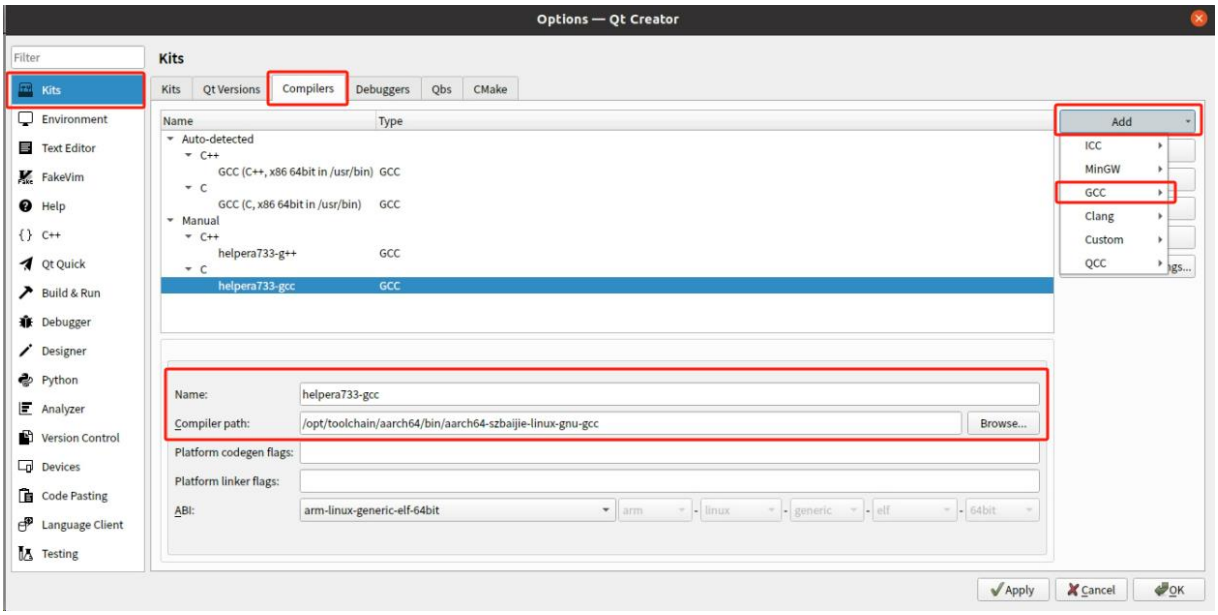
注：开发板的 ip 地址需要使用串口登录（参考 2.2 xshell 使用）到开发板中，通过下面命令查看：

```
$ ifconfig
```

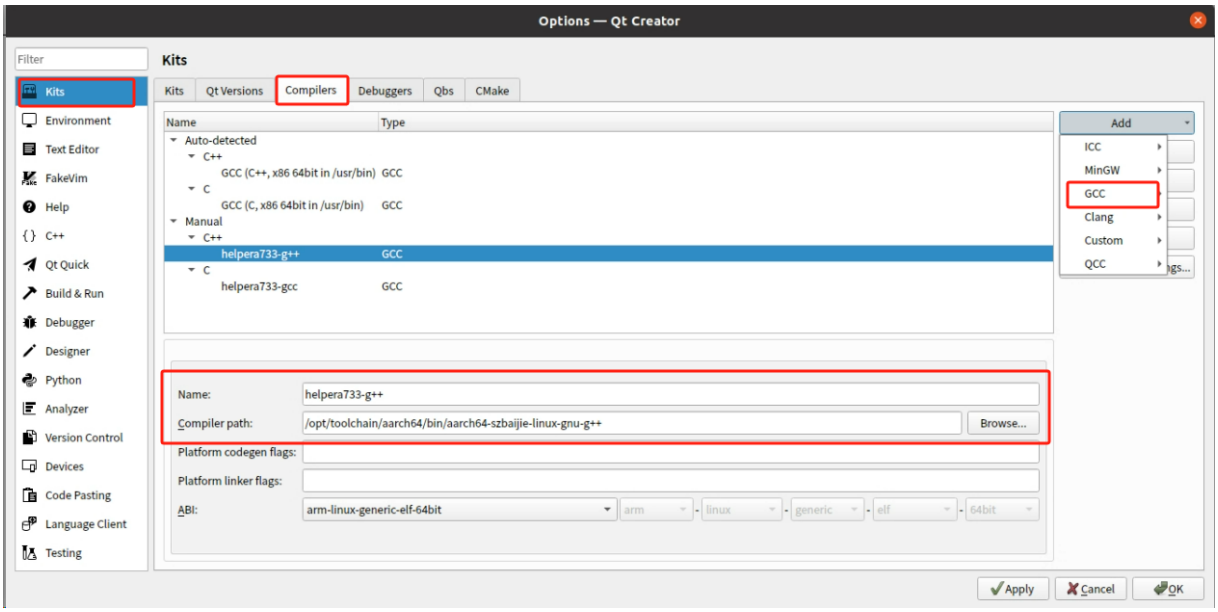
6.2.2 设置编译器

在 6.1 节中拷贝完编译器后，我们还需要配置 qtcreator 环境变量。打开我们虚拟机中内置的 qtcreator 软件，然后依次点击 **Tools->Options->Kits->Compilers**。c 编译器设置如下：

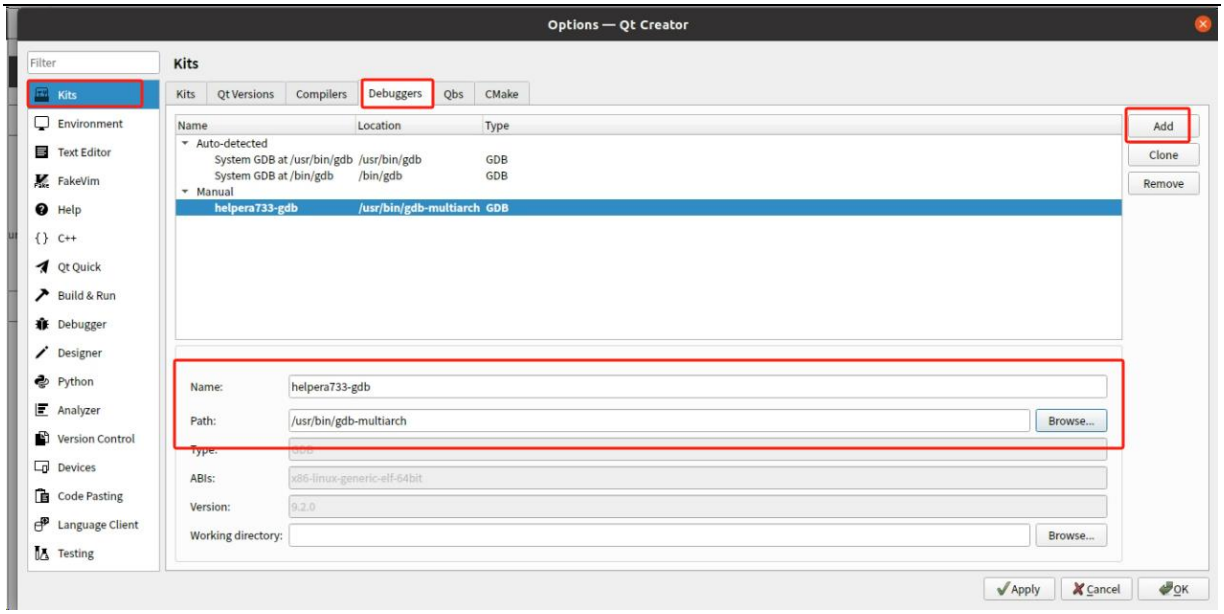
（注：编译器设置时 ABI 选项选为 **arm-linux-generic-elf-64bit**）



然后 **g++**编译器设置如下：

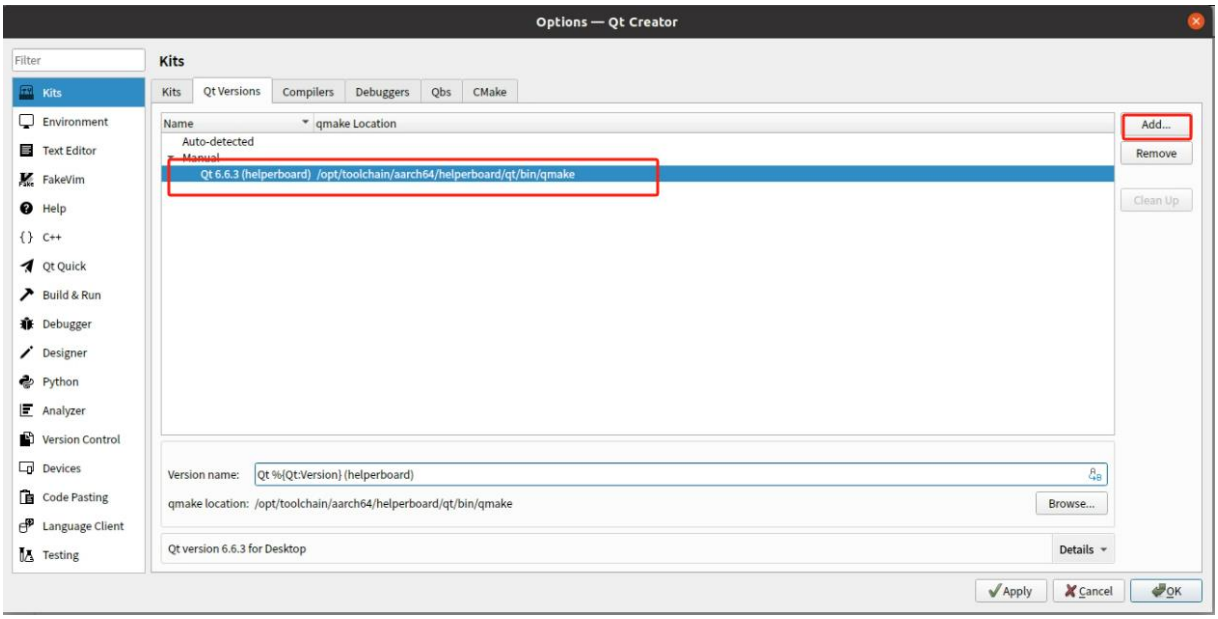


然后 **gdb** 调试器设置如下：



6.2.3 设置 qt 版本

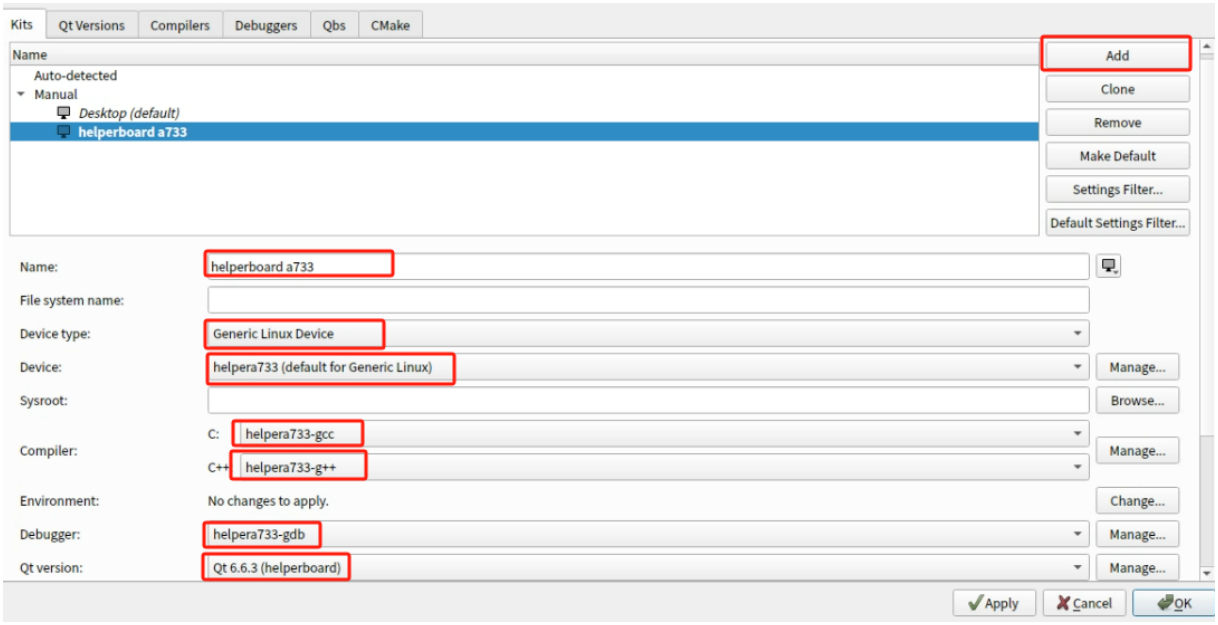
如果设置 qt 版本出现感叹号等警示标志，表明相关库有问题，请检查是否选对编译器



6.2.4 设置 kits

设置完其他的之后，最后设置 kits 设置如下：

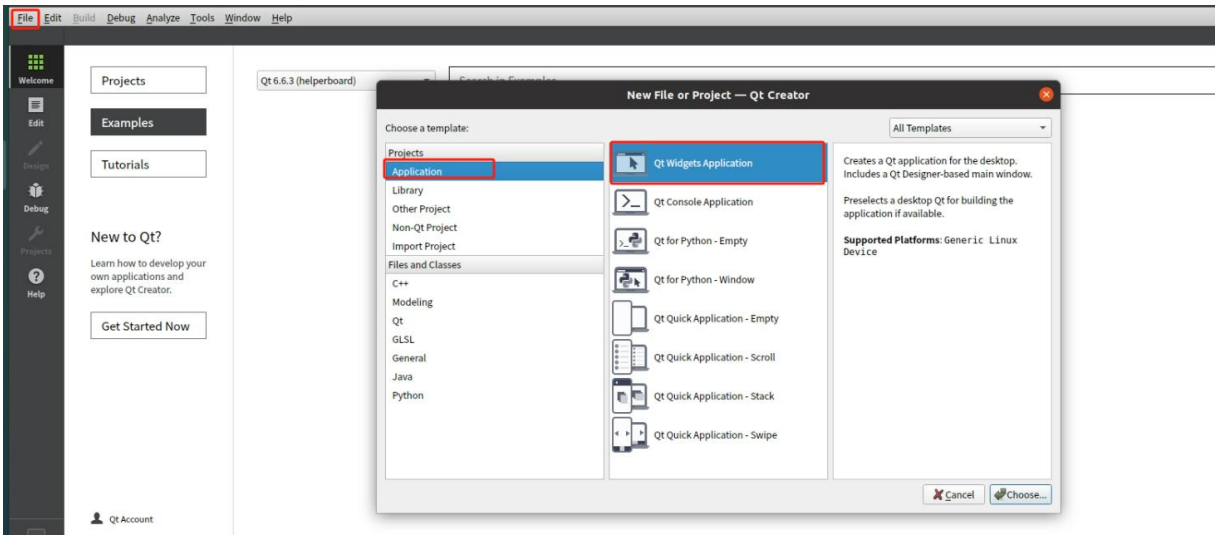
（注：如果没有配置成功，会有感叹号的标志，这时候检查下我们提供的编译器。）

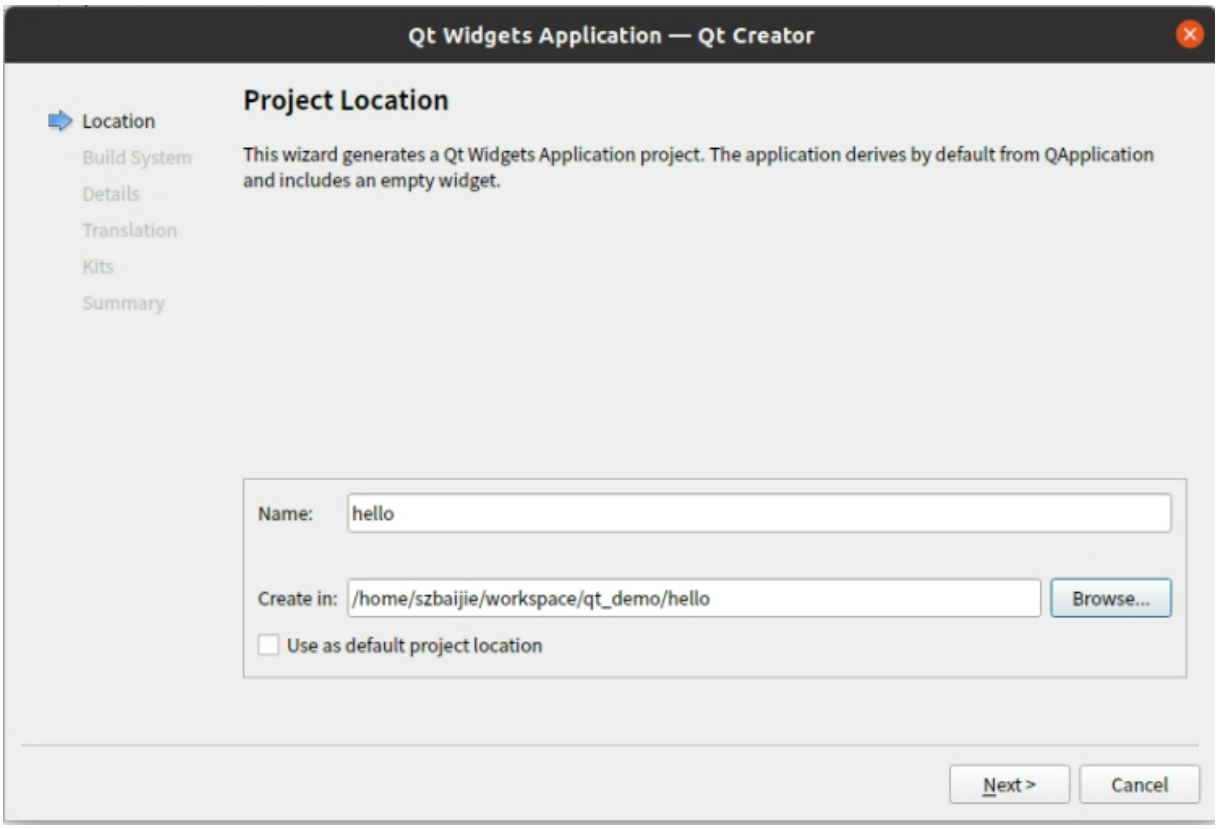


虚拟机中 qtcreator 远程调试与运行程序

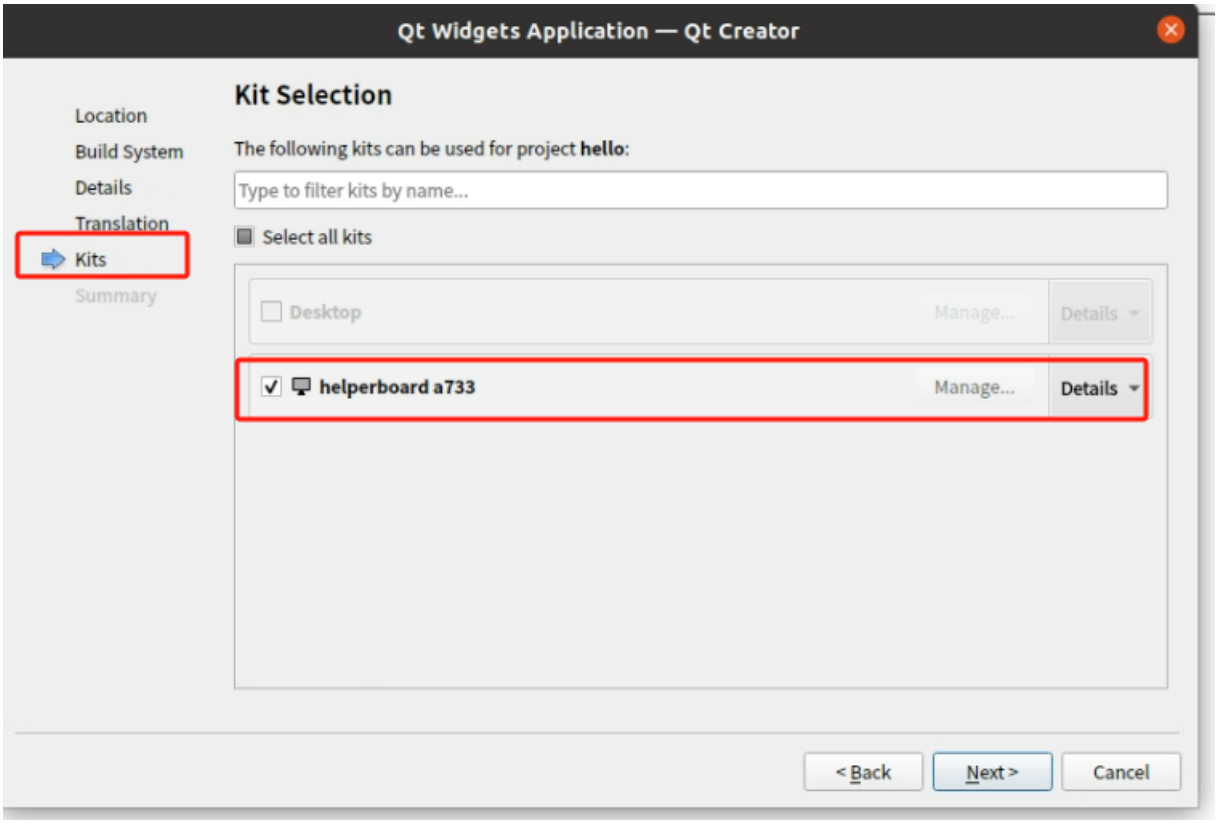
6.2.5 新建 qt 工程

打开 qtcreator，依次点击 **File->New File or Project** 来新建工程。如下所示





上图点击 next 之后，直到跳到如下图所示的状态，这时候需要将我们刚才建立的 kits 勾选上，我这里是 **helperboard a733**



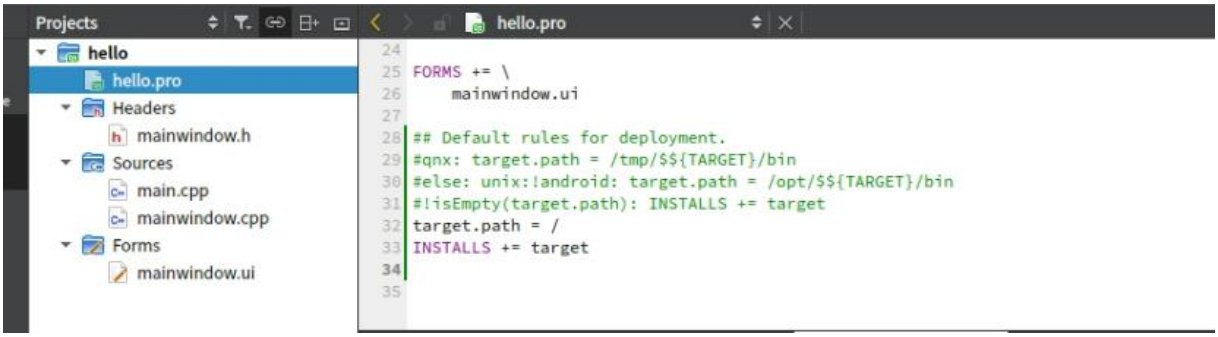
勾选完 kits 后，点击 **Next->Finish** 工程就建立完毕了

6.2.6 设置运行程序路径

这时候我们需要在工程文件 **hello.pro** 中修改对应代码。如下命令所示：

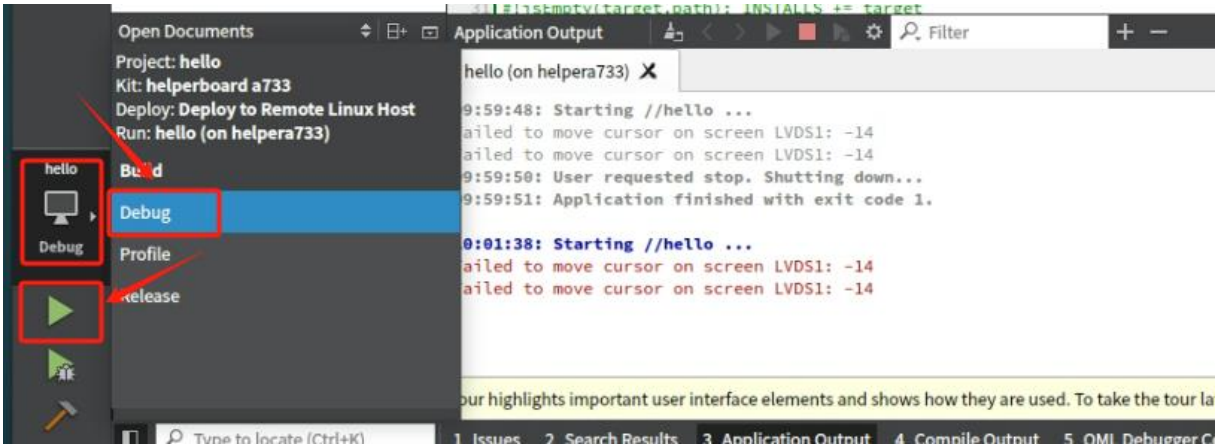
```
target.path = /

INSTALLS += target
```

这两条命令表示可以远程运行在开发板上的**根目录。**

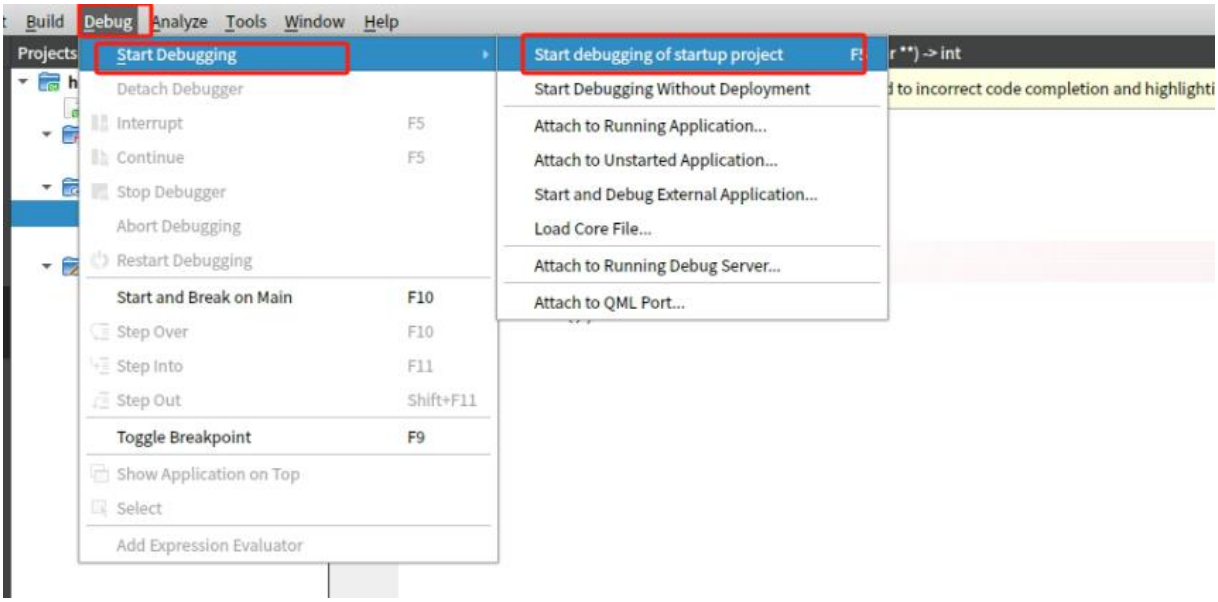


6.2.7 运行与调试

依次点击 **Debug->Debug 选择编译套件**。确定后点击三角形按钮来运行程序。如下图所示，可以看到程序已经被成功执行：



如果需要调试，点击 **Debug->Start Debugging->Start debugging of startup project** 来调试，具体的调试步骤自行学习。

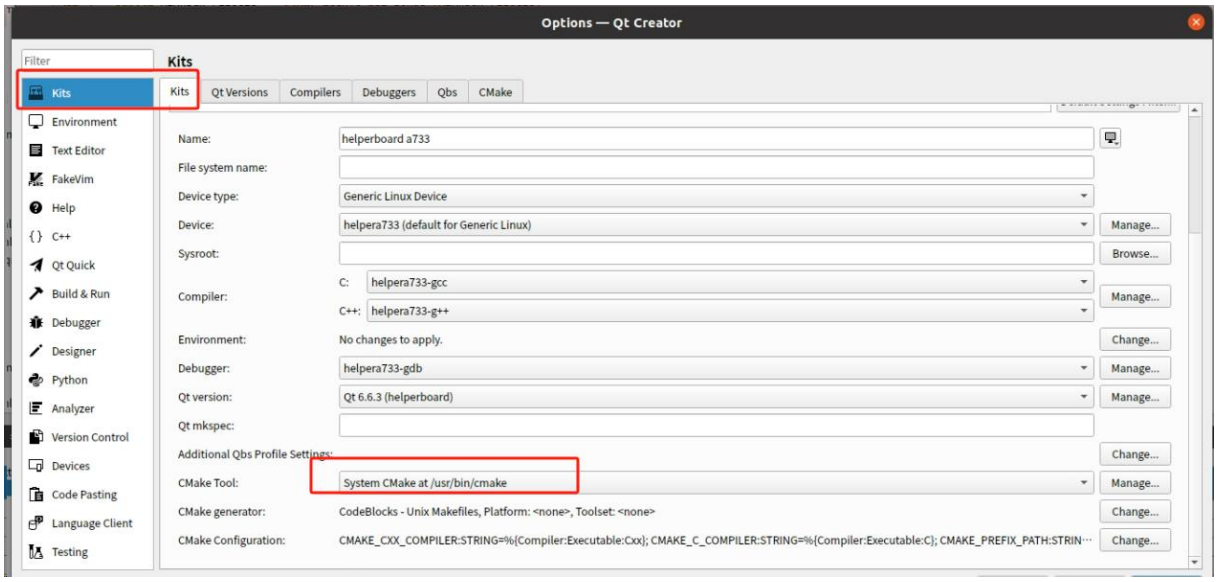


6.3 Qtcreator 中使用 Cmake 交叉编译

若使用 cmake 构建编译，需要增加以下配置，首先在确保虚拟机系统安装 cmake 工具

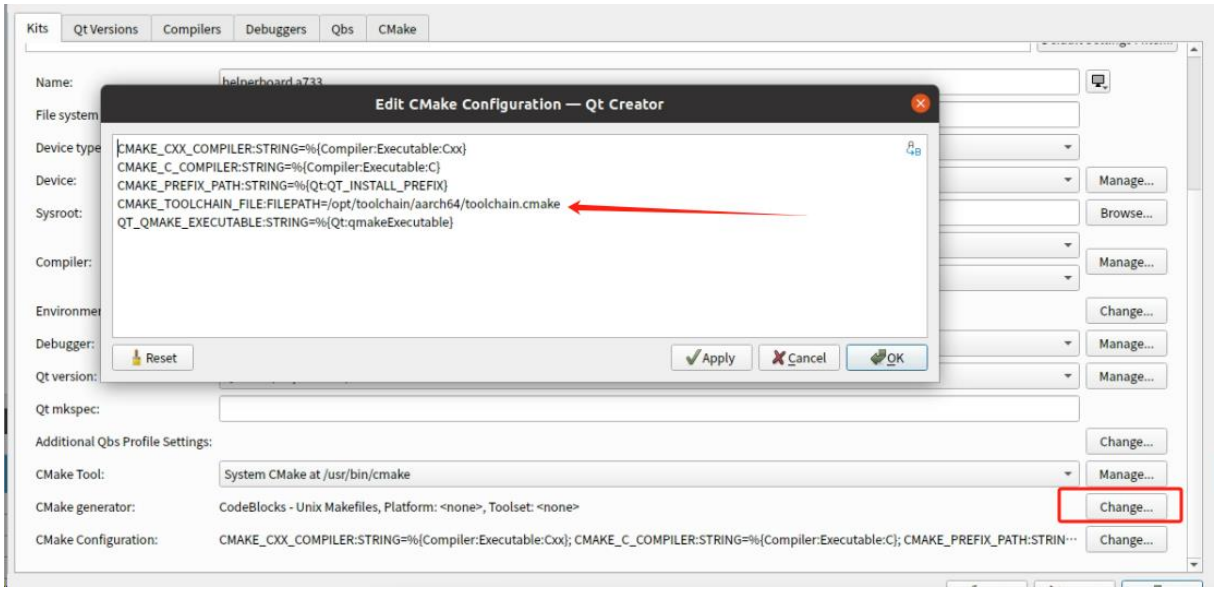
```
$ sudo apt update && apt install cmake -y
```

打开 QtCreator，进入如下配置，选择系统 cmake 工具

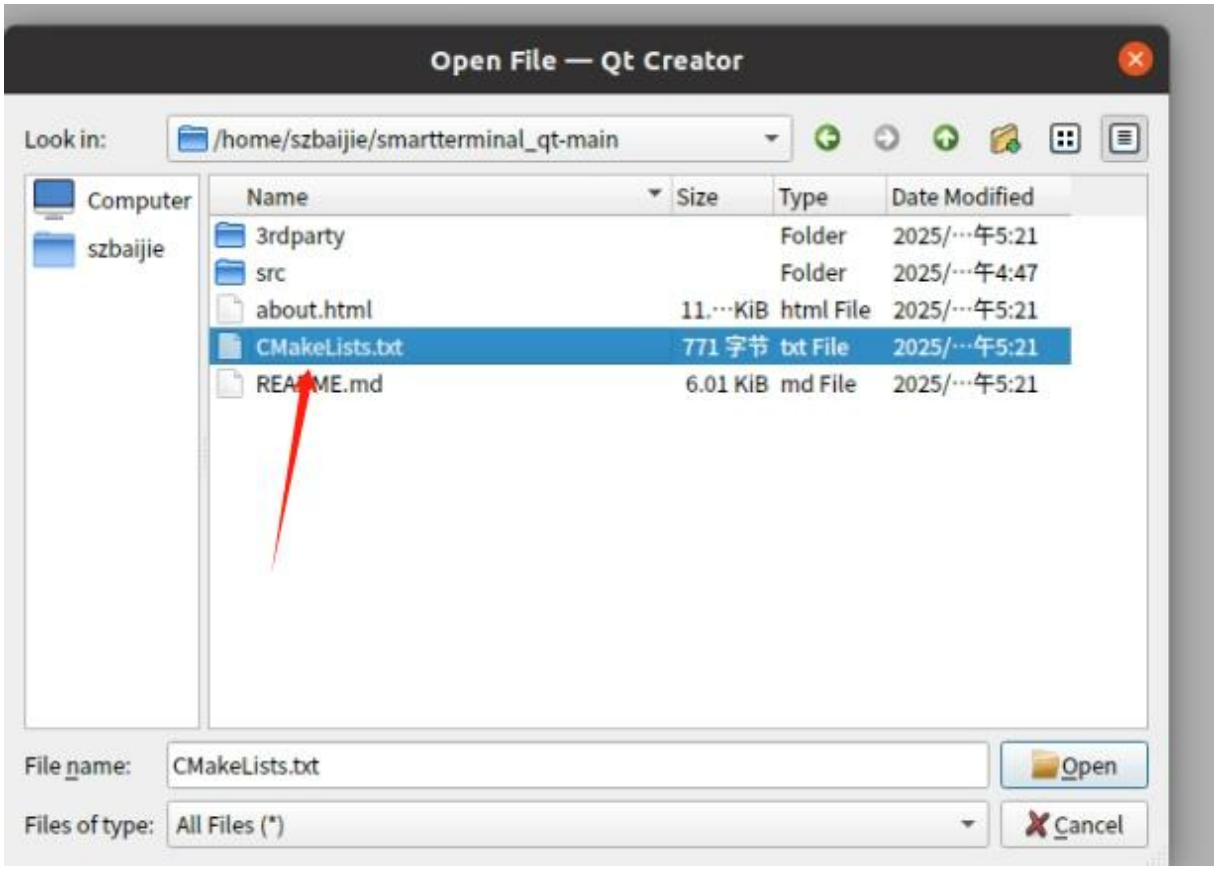


修改 Cmake Configuration 配置如下，增加一行指定 TOOLCHAIN_FILE 配置文件，文件内容跟上文一样，建议先用命令编译通过再指定使用。

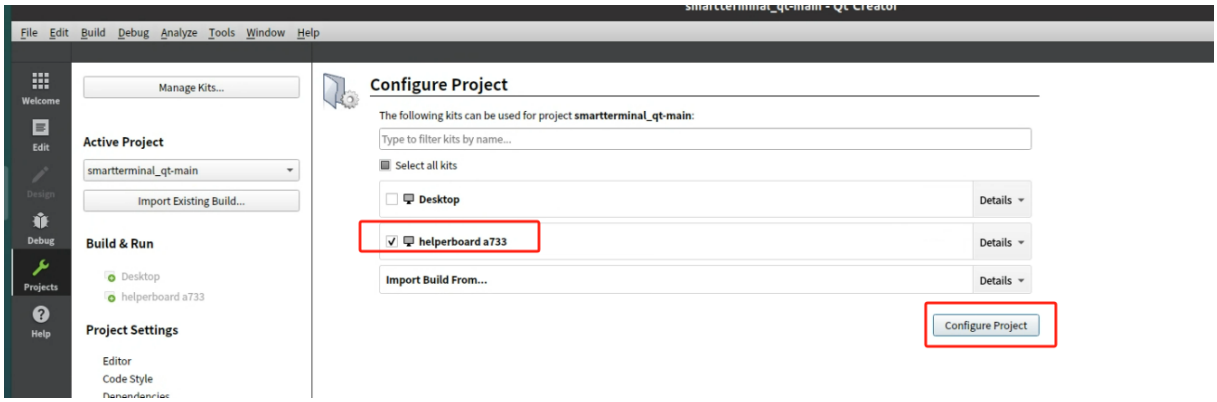
```
$ CMAKE_TOOLCHAIN_FILE:FILEPATH=/opt/toolchain/aarch64/toolchain.cmake
```



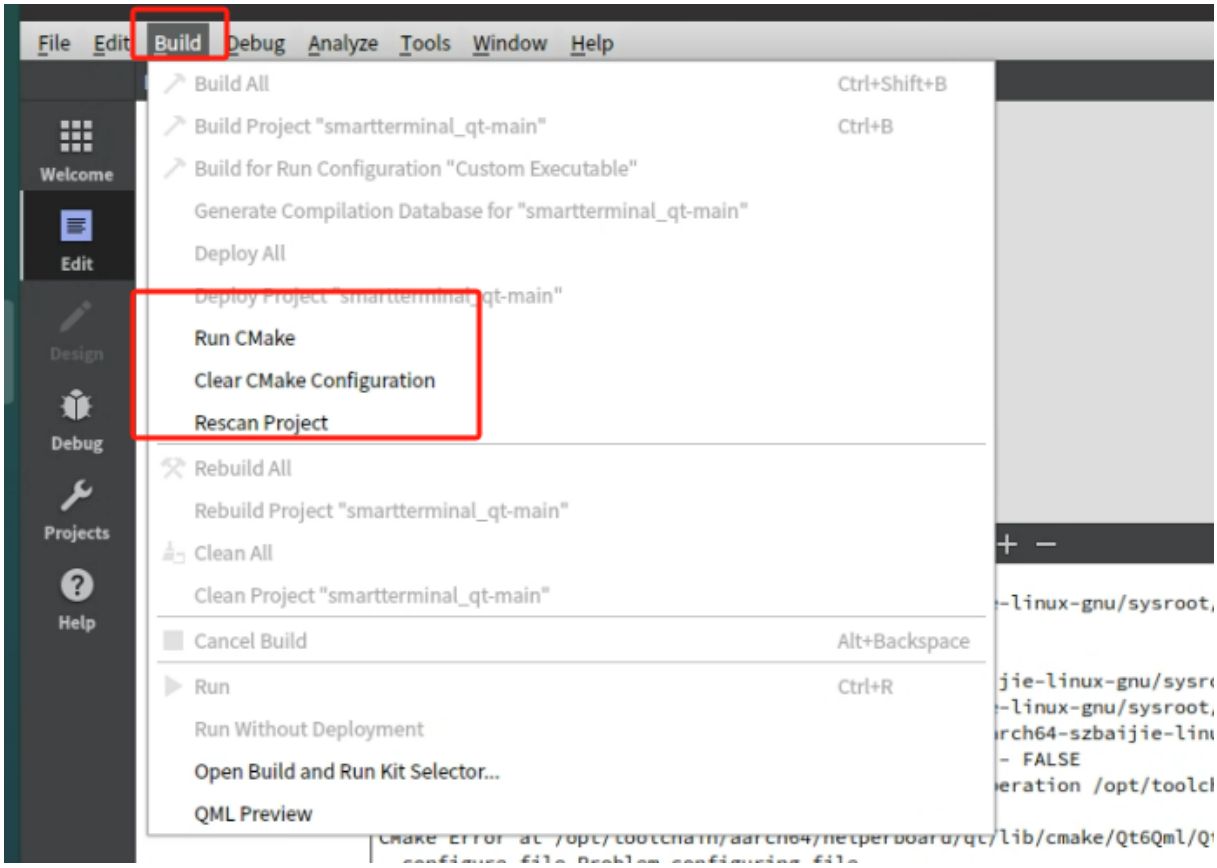
修改完配置后打开 Qt 工程



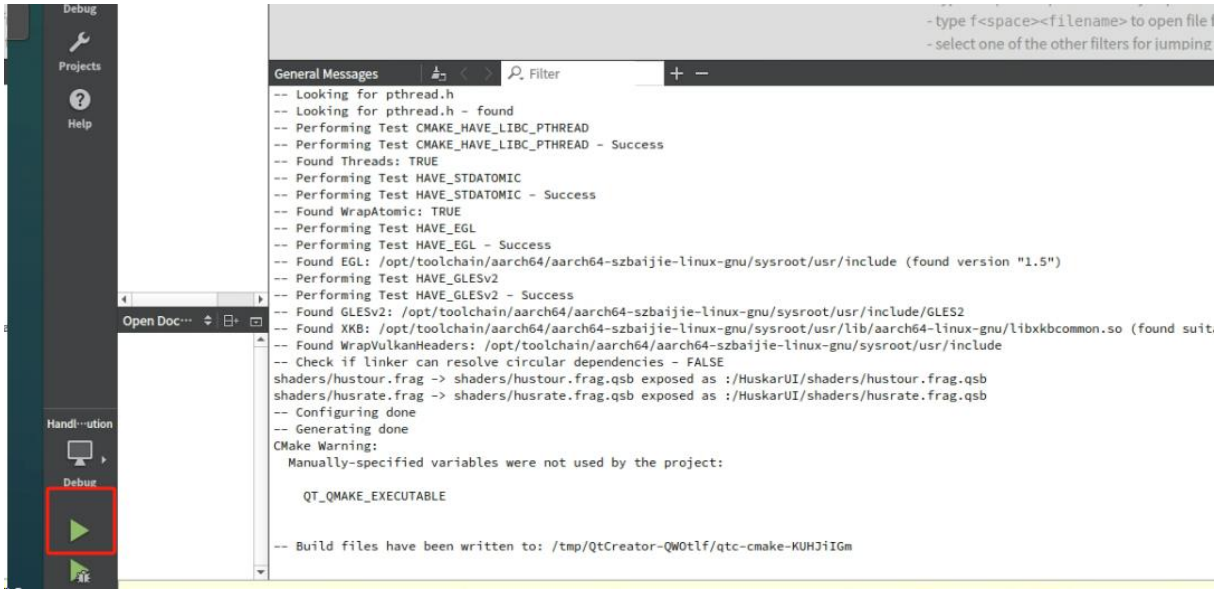
选择目标开发板，点击 Configure Project



修改配置后可以清理一下再重新构建



构建成功后点击编译即可



6.4 开发板设置 qt 开机自启动

6.4.1 在已经烧录的板子上自启动

默认的 **qt_env.sh** 在**/usr/helperboard/**目录下，如果需要更改环境变量，可自行更改。开发板默认启动的 qt 程序在**/etc/init.d/rcS** 中，如果需要将修改自己的 qt 程序，需要先使用 scp 命令将 qt 程序拷贝到开发板，这里以 BaijieDemo 为例，

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

注意：BaijieDemo 需要修改为自己的 qt 程序，root 为开发板登录的用户名，“@”后面的是开发板的 IP 地址，冒号“:”后面的是开发板的目录，视具体情况修改，传输成功如下

```
bj35djc@LinuxServer3:~/A733$ scp BaijieDemo root@192.168.1.187:~
BaijieDemo                                    100% 1543KB  11.0MB/s   00:00
bj35djc@LinuxServer3:~/A733$
```

传输到开发板后，将自己的 qt 程序执行命令替换**/etc/init.d/rcS** 中下图红框显示位置：

```
. /usr/helperboard/qt_env.sh
#/usr/helperboard/examples/opengl/hellogles3/hellogles3 &
/usr/helperboard/examples/quick3d/helloqtquick3d/helloqtquick3d &
```

6.4.2 在固件中设置自启动

按 5.3 节选择对应的 extra 后，进入到 longan/test/dragonboard/extra，在编译时 extra 中内容会替换 rootfs 中文件。

```
data  etc  init  lib  media  qt_env.sh  root  usr  vendor
```

所以这里我们只需要更改 extra 中的**/etc/init.d/rcS** 就能将 qt 程序集成到固件中，这时候烧录到板子上就能开机自启动了。

7 制作 android 系统固件

7.1 编译 linux 内核

Android 在编译之前需要先编译 linux 内核。Linux 内核编译和 5.1 节一致，只不过 android 配置选项不同，配置过程如下：

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

$ ./build.sh config
```

输入上面命令后需要配置 ubuntu 信息，如下：

1、**platform** 输入“0”， 选择 **android**。

2、**ic** 输入“0”，选择 **a733**。

3、**board** 输入“10”，选择 **pro3**，对应 HelperA733 开发板的硬件配置路径。

4、**flash** 输入“0” ，选择 **default**。

5、**devicetree** 输入“1”，选择 **lvds7.0_1024x600_bj**。可按需更改。

具体如下图所示：

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

选择完配置后执行如下命令即可开始编译：

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

注：请使用我们提供的虚拟机 helperboard_ubuntu20.04，否则会编译失败！

7.2 编译 uboot

进入到 uboot，直接编译

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

$ cd brandy/brandy-2.0/u-boot-2018/

$ make sun60iw2p1_a733_defconfig && make -j
```

7.3 编译与打包 android

回到主目录设置开发环境：（退出终端或者新打开终端都需要设置环境）

```
$ cd ~/workspace/a733/

$ source build/envsetup.sh
```

运行 lunch：

```
$ lunch
```

注：输入“**20**“，选择开发板对应的配置：”**a733_pro3_arm64-ap3a-userdebug**“，也可以直接输入”**lunch 20**“

编译 android：

```
$ make -j8
```

然后就开始漫长的等待，直到编译结束，根据电脑的配置情况，可能会是从一小时到几小时不等的时间。

注：编译不通过的话就去掉-j 和后边的数字（数字代表用多少线程编译，此处所用的最大线程数是 8，可以是 CPU 核心数的 2 倍），加的任务数越多需要的内存也越多。去掉 –j 后还是失败的话就运行以下命令后再编译,直到编译成功：

```
$ make installclean
$ make
```

注：如果还是编译不过，尝试多编译几次

编译结束后，用以下命令可以打包生成安卓烧写镜像：

```
$ pack
```

最终生成的镜像在 **longan/out/a733_android15_pro3_uart0.img**，然后就可以使用烧写工具“PhoenixSuit”（**参考 4.3 节**）烧录到板子。

8 其他操作

8.1 Android 修改开机 logo

开机 logo 图片名字只能命名为 bootlogo.bmp，替换掉 longan/device/config/chips/a733/configs/pro3/dragonboard 和 longan/device/config/chips/a733/configs/pro3/android 目录下的 bootlogo.bmp。

注：

1、 在更换 bootlogo.bmp 图片时，转换 bmp 格式的深度必须选择 32 位

2、 开机 logo 图片的名字错误，编译时会报错

图片格式可以使用转换软件，我使用的是 photoshop 转换，如下：

转换完成后，替换 bootlogo.bmp，回到 longan 目录下编译。如下所示：

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

编译完成后按照 android 或者 ubuntu 固件打包方式打包就行。打包后的镜像在 longan/out 目录下（**注：参考第 5 章 ubuntu 固件制作和第 7 章 android 固件制作**）。

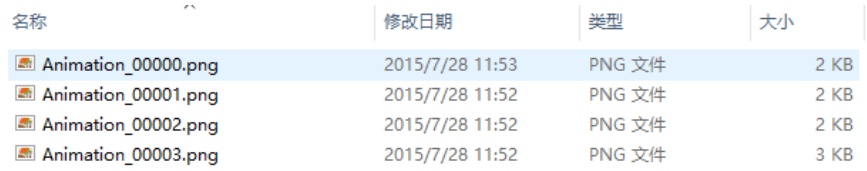
8.2 Android 修改开机动画和开机音乐

开机动画 bootanimation.zip 打包文件按以下步骤制作。

8.2.1 目录结构说明

bootanimation.zip 包含 part0 、part1 文件夹和 desc.txt 文件。part0、part1 文件夹里面放的是动画拆分的图片，part0 只显示一次，part1 为循环显示图片，格式为 png 或 jpg。动画目录示例如下：

part0 目录示例：



desc.txt 文件内容如下：

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

说明：第一行 500 350 和图片分辨率一致，500 为图片宽度，350 为图片高度（**注：图片的分辨率不要超过屏幕分辨率**）20 为帧数。第二行开始 p 为标志符，接下来第二列为循环次数（0 为无限循环），第三项为两次循环之间间隔的帧数，第四项为对应的目录名。播放动画时会按照图片文件名顺序自动播放。

8.2.2 开机音乐

如需开机音乐，将开机音乐放入 part0 目录中，命名为 audio.wav。

在根目录中加入 audio_conf.txt，内容如下：

```
card=0

device=0

period_size=4096

period_count=8

#Input_Event or Switch_State

Headset detection=Switch_State

switch_path=/sys/module/snd_soc_sunxi_common/parameters/jack_stat

e

single_volume_control=1

speaker_headphones_out=0

mixer "SPK Volume"=20

mixer "OutputR Mixer DACR Switch"=1

mixer "OutputL Mixer DACL Switch"=1

mixer "LINEOUTL Switch"=1

mixer "LINEOUTR Switch"=1

mixer "SPK Switch"=1

headset mixer "OutputR Mixer DACR Switch"=1

headset mixer "OutputL Mixer DACL Switch"=1
```

```
headset mixer "HPOUTL Switch"=1

headset mixer "HPOUTR Switch"=1

headset mixer "SPK Switch"=0
```

8.2.3 打包与编译

windows 使用打包工具打包，选择 ZIP 格式，压缩标准要选“储存”。

linux 下进入 bootanimation 目录，使用如下命令进行打包：

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

注：linux 命令使用-0 指定压缩等级为最低等级 stored，即只归档不压缩，否则可能由于包格式问题引起动画显示为黑屏

然后将打包好的文件替换 device/softwinner/jupiter/common/media/bootanimation 目录下的 bootanimation.zip，并修改权限：

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

回到 android 目录下,清除以前的编译配置后再打包：

```
$ make installclean

$ make -j8 && pack
```

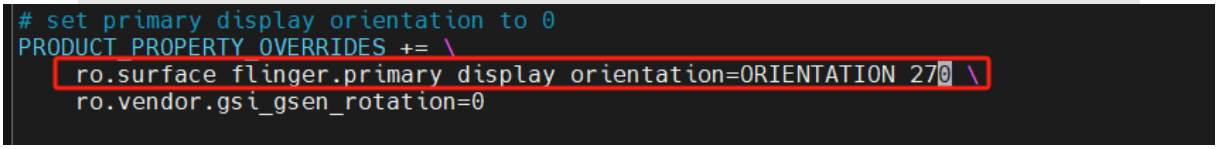
最终生成的镜像在 longan/out/ a733_android15_pro3_uart0.img

8.3 Android 修改屏幕旋转

A733 屏幕旋转需要在 device/softwinner/jupiter/a733-pro3/device-common.mk 文件中修改旋转方向，在源码路径 device/softwinner/jupiter/a733-pro3/ input/ gt9xxnew_ts.idc 中修改触摸方向。开机 logo 是图片固定方向，所以开机 logo 不会更改方向。

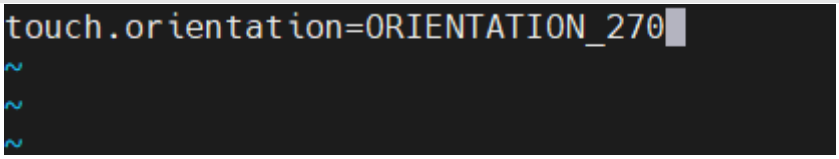
打开 device-common.mk 找到 RODUCT_PROPERTY_OVERRIDES 属性，旋转角度为顺时针旋转，例如顺时针旋转 270°，添加如下代码：

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



打开 gt9xxnew_ts.idc 文件，里面只有一条触摸属性值设置，旋转角度为顺时针旋转，例如顺时针旋转 270°，添加如下代码：

```
touch.orientation=ORIENTATION_270
```



修改完成后重新按第 7 章编译安卓固件烧录即可

8.4 Ubuntu 设置网卡静态 ip

与网卡静态 ip 有关的脚本共有三个，分别是设置静态 ip、修改静态 ip、删除静态 ip。

1、设置静态 ip

运行脚本后根据提示输入 ip 地址/24、掩码和 DNS。命令如下：

```
$ eth0_set.sh
```

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

2、修改静态 ip

修改时只需要重设 ip，不需要重设掩码。命令如下：

```
$ eth0_mod.sh
```

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

3、删除静态 ip

```
$ eth0_del.sh
```

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

8.5 Ubuntu 控制 wifi

与 wifi 有关的脚本就一个/usr/bin/wifi.sh

1、查看 wifi 列表

```
$ nmcli device wifi list
```

2、设置 wifi

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

```
root@HelperA733:~# wifi.sh -f c -n baijiekeji -p
Device 'wlan0' successfully activated with 'b508f158-c245-4393-a0f3-9bcdf10f4d71'.
root@HelperA733:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.140  netmask 255.255.254.0  broadcast 192.168.1.255
        inet6 fe80::5b97:eddc:4dc0:5855  prefixlen 64  scopeid 0x20<link>
        ether d4:bf:48:ac:9a:57  txqueuelen 1000  (Ethernet)
        RX packets 20465  bytes 1755880 (1.7 MB)
        RX errors 0  dropped 240  overruns 0  frame 0
        TX packets 213  bytes 22260 (22.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 7

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

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

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

当 wifi 设置完毕后，每次开机都会自动连接。

3、修改 wifi

如果当前连接的 wifi 密码被修改了，开发板对应的连接密码也要同步修改，执行下面命令：

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

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

4、删除 wifi

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

```
root@HelperA733:~# wifi.sh -f d -n baijiekeji
Connection 'baijiekeji' (b508f158-c245-4393-a0f3-9bcdf10f4d71) successfully deleted.
```

5、设置静态 ip

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

```
root@HelperA733:~# wifi.sh -f s -i 192.168.1.253 -n baijiekeji -p [redacted] -g 192.168.1.1
Connection 'baijiekeji' (7de53b40-915a-4d8b-affe-581772dc6102) successfully deleted.
Device 'wlan0' successfully activated with '4e545168-b504-4e25-ad4c-7d39cffb9b1b'.
Connection 'baijiekeji' successfully deactivated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/19)
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/20)
root@HelperA733:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.140  netmask 255.255.254.0  broadcast 192.168.1.255
        inet6 fe80::5b97:eddc:4dc0:5855  prefixlen 64  scopeid 0x20<link>
        ether d4:bf:48:ac:9a:57  txqueuelen 1000  (Ethernet)
        RX packets 29850  bytes 2532832 (2.5 MB)
        RX errors 0  dropped 387  overruns 0  frame 0
        TX packets 263  bytes 25760 (25.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 7

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

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

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

8.6 Ubuntu 根文件系统备份

当我们需要保存板子上 ubuntu 系统所做的修改时，可以把整个文件系统备份，然后编译固件时替换掉 rootfs 文件，这样新的固件也拥有之前所做的修改，不需要再重新操作一遍，可以高效的实现功能的移植。

1、文件打包

运行 szbaijie_rootfs_backup 命令，将在根目录下得到 rootfs.tar.gz 文件，详情如下：

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

注：当有一些重要文件是在其他挂载目录下时，比如 /dev/sda5 挂载到/data 目录，打包时并不会把/data 目录下的这些文件包含进去。

2、编译

打包完成之后，使用 U 盘或 scp 命令等方式，把 rootfs.tar.gz 文件拷贝到 linux 源码 longan/test/dragonboard/baijie_system 目录下（建议重新命名），之后回到 longan 目录下重新进行./build.sh config 配置，baijie_extra 选择 no_extra，baijie_rootfs 选择新增的文件系统压缩包即可。

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

```
All available baijie_extra:
   0. extra-24.04-core
   1. extra-a733-mate
   2. extra-qt5.15.14
   3. no_extra
Choice [extra-24.04-core]: 3
All available baijie_rootfs:
   0. rootfs_new.tar.gz
   1. rootfs_ubuntu22.04_core.tar.gz
   2. rootfs_ubuntu22.04_mate.tar.gz
   3. ubuntu24.04_core.tar.gz
Choice [rootfs_new.tar.gz]: 0
rootfs decompression done!
```

至此就可以按用户开发手册操作编译生成新的固件，运行命令如下：

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

将新编译好的固件烧录到开发板即可。

8.7 Ubuntu 修改开机 logo

Ubuntu 系统需要更改开机 logo 时，可使用此方法更改，重启后就能看到更改的 logo

要求：bootlogo.bmp 必须为 32 位图片，图片的分辨率需小于显示器的分辨率。

使用方法：szbaijie_logo_update_64 ./bootlogo.jpg 1024 600 0

说明如下：第一个参数为图片路径，第二个参数为屏幕宽，第三个参数为屏幕高，第四个参数为图片旋转角度（逆时针旋转）

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

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

运行成功后如上图所示，重启系统后可以看到 bootlogo 已经改变

8.8 Ubuntu 修改屏幕旋转

8.8.1 Qt 系统修改

1、修改显示方向，设置如下环境变量后重启 Qt 程序，其中 90 为旋转角度，按需求修改即可

```
$ export QT_QPA_EGLFS_ROTATION=90
```

2、修改触摸方向，执行如下脚本，其中 90 是旋转角度，按需求修改即可

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

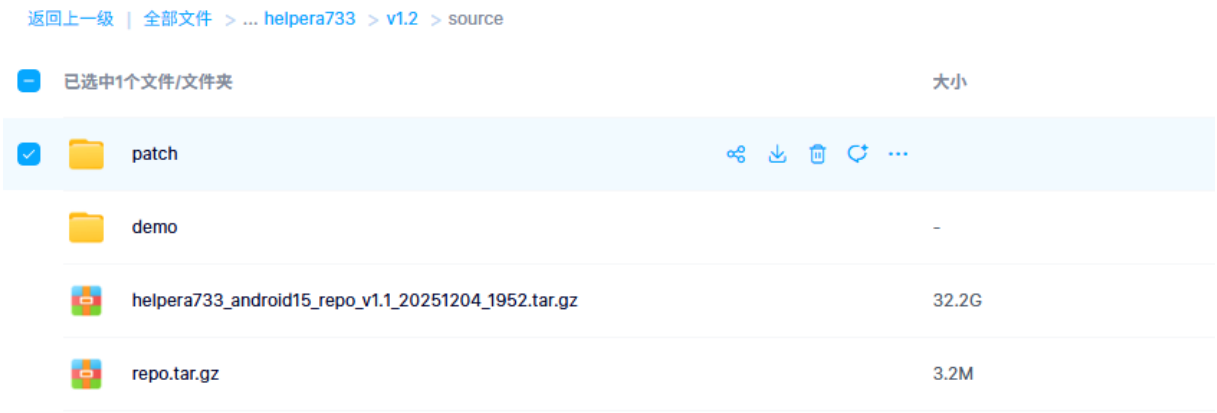
8.8.2 Xfce 系统修改

执行如下脚本命令，后面的参数代表顺时针旋转角度

```
$ /usr/bin/rotate-screen.sh 0

$ /usr/bin/rotate-screen.sh 90

$ /usr/bin/rotate-screen.sh 180

$ /usr/bin/rotate-screen.sh 270
```

8.9 打补丁

系统发布的版本可能存在需要用户去打补丁的情况，补丁目录在 source->patch



比如我们要打补丁 example.patch 一般如下图所示：

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

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

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

打补丁时，我们先进入到相关目录，比如上图中的 patch 为在 longan/bsp 目录中生成的 patch。patch 修改的是 drivers/vin/modules/sensor/ov8858_r2a_4lane.c 和 drivers/vin/vin-cci/sunxi_cci.c 文件，所以我们只需要进入 longan/bsp 目录就能正确打 patch 了（**注：在其他目录打 patch 会报错**）。命令如下：

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

$ patch -p1 < example.patch
```

8.10 内核配置

A733 所需的内核配置都已经完成，下边的举例只是让用户熟悉内核配置的过程。执行内核配置前可先加载我们提供的默认内核配置 config 文件（**参考 5.1 和 7.1**）

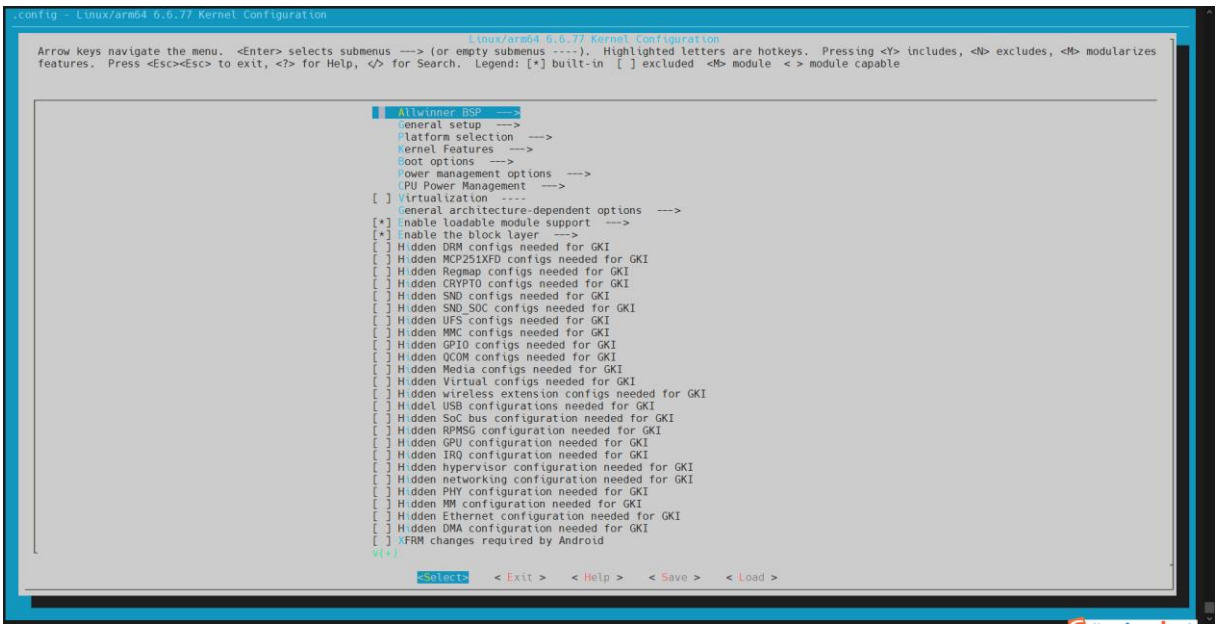
Linux 系统默认内核配置“longan/device/config/chips/a733/configs/default/linux-6.6/dragonboard_defconfig”

Android 系统默认内核配置“longan/device/config/chips/a733/configs/default/linux-6.6/android15_arm64_defconfig”

在此基础上进行修改，修改的配置内容会保存到内核目录下的“longan/out/kernel/build/.config”文件中。配置命令如下所示

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

$ ./build.sh menuconfig
```

注：可以在上图前几行文字描述中看到 Y 表示选择，N 是取消选择，M 是编译为模块等信息



9 关于定制服务

本公司提供基于全志 A733 平台项目的定制服务，如有需要请联系：13510178868 陈先生。