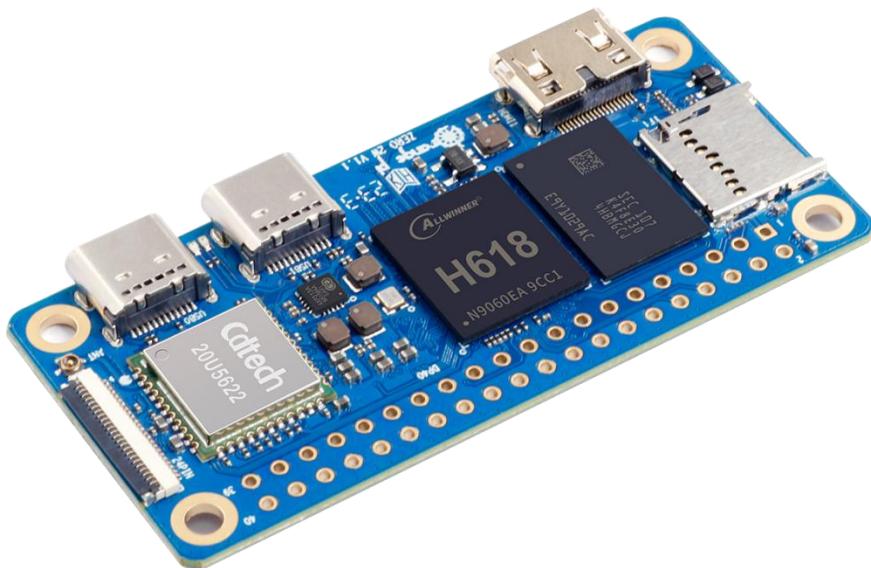




Orange Pi Zero 2W

用户手册





目录

1. Orange Pi Zero 2w 的基本特性	1
1. 1. 什么是 Orange Pi Zero 2w	1
1. 2. Orange Pi Zero 2w 的用途	1
1. 3. Orange Pi Zero 2w 是为谁设计的	1
1. 4. Orange Pi Zero 2w 的硬件特性	2
1. 5. Orange Pi Zero 2w 的顶层视图和底层视图	3
1. 6. Orange Pi Zero 2w 的接口详情图	4
1. 7. Orange Pi Zero 2w 24pin 扩展板的接口详情图	5
2. 开发板使用介绍	7
2. 1. 准备需要的配件	7
2. 2. 下载开发板的镜像和相关的资料	11
2. 3. 基于 Windows PC 将 Linux 镜像烧写到 TF 卡的方法	11
2. 3. 1. 使用 balenaEtcher 烧录 Linux 镜像的方法	11
2. 3. 2. 使用 Win32Diskimager 烧录 Linux 镜像的方法	15
2. 4. 基于 Ubuntu PC 将 Linux 镜像烧写到 TF 卡的方法	18
2. 5. 烧写 Android 镜像到 TF 卡的方法	22
2. 6. 板载 SPI Flash 中的微型 linux 系统使用说明	28
2. 7. 启动香橙派开发板	29
2. 8. 调试串口的使用方法	30
2. 8. 1. 调试串口的连接说明	30
2. 8. 2. Ubuntu 平台调试串口的使用方法	31
2. 8. 3. Windows 平台调试串口的使用方法	34
2. 9. 使用开发板 40pin 接口中的 5v 引脚供电说明	37
3. Debian/Ubuntu Server 和 Xfce 桌面系统使用说明	38
3. 1. 已支持的 linux 镜像类型和内核版本	38



3. 2. linux 内核驱动适配情况	39
3. 3. 本手册 linux 命令格式说明	40
3. 4. linux 系统登录说明	41
3. 4. 1. linux 系统默认登录账号和密码	41
3. 4. 2. 设置 linux 系统终端自动登录的方法	42
3. 4. 3. linux 桌面版系统自动登录说明	42
3. 4. 4. Linux 桌面版系统 root 用户自动登录的设置方法	43
3. 4. 5. Linux 桌面版系统禁用桌面的方法	44
3. 5. 板载 LED 灯测试说明	44
3. 6. TF 卡中 linux 系统 rootfs 分区容量操作说明	47
3. 6. 1. 第一次启动会自动扩容 TF 卡中 rootfs 分区的容量	47
3. 6. 2. 禁止自动扩容 TF 卡中 rootfs 分区容量的方法	48
3. 6. 3. 手动扩容 TF 卡中 rootfs 分区容量的方法	50
3. 6. 4. 缩小 TF 卡中 rootfs 分区容量的方法	55
3. 7. 24Pin 扩展板接口引脚说明	59
3. 8. 24pin 扩展板上的 2 个 LRADC 按键的使用方法	61
3. 9. 网络连接测试	65
3. 9. 1. 以太网口测试	65
3. 9. 2. WIFI 连接测试	67
3. 9. 3. 通过 create_ap 创建 WIFI 热点的方法	74
3. 9. 4. 设置静态 IP 地址的方法	81
3. 9. 5. 设置 Linux 系统第一次启动自动连接网络的方法	89
3. 10. SSH 远程登录开发板	92
3. 10. 1. Ubuntu 下 SSH 远程登录开发板	92
3. 10. 2. Windows 下 SSH 远程登录开发板	93
3. 11. HDMI 测试	95
3. 11. 1. HDMI 显示测试	95
3. 11. 2. HDMI 转 VGA 显示测试	96
3. 11. 3. Linux5.4 系统 HDMI 分辨率设置的方法	97
3. 11. 4. Linux5.4 系统 Framebuffer 宽度和高度的修改方法	98
3. 11. 5. Framebuffer 光标设置	98



3.12. 蓝牙使用方法	99
3.12.1. 桌面板镜像的测试方法	99
3.12.2. 服务器版镜像的使用方法	102
3.13. USB 接口测试	105
3.13.1. USB 接口扩展说明	105
3.13.2. USB0 设置为 HOST 模式的方法	106
3.13.3. 连接 USB 鼠标或键盘测试	108
3.13.4. 连接 USB 存储设备测试	108
3.13.5. USB 以太网卡测试	109
3.13.6. USB 摄像头测试	110
3.14. 音频测试	112
3.14.1. 使用命令行播放音频的方法	112
3.14.2. 在桌面系统中测试音频方法	114
3.15. 红外接收测试	116
3.16. 温度传感器	118
3.16.1. linux5.4 系统查看温度的方法	118
3.16.2. linux6.1 系统查看温度的方法	119
3.17. 40 Pin 接口引脚说明	119
3.18. 安装 wiringOP 的方法	120
3.19. 40pin 接口 GPIO、I2C、UART、SPI 和 PWM 测试	122
3.19.1. 40pin GPIO 口测试	122
3.19.2. 40 Pin GPIO 口上下拉电阻的设置方法	123
3.19.3. 40pin SPI 测试	124
3.19.4. 40pin I2C 测试	127
3.19.5. 40pin 的 UART 测试	130
3.19.6. 使用/sys/class/pwm/测试 PWM 的方法	133
3.20. wiringOP 硬件 PWM 的使用方法	137
3.20.1. 使用 wiringOP 的 gpio 命令设置 PWM 的方法	138
3.20.2. PWM 测试程序的使用方法	142
3.21. wiringOP-Python 的安装使用方法	143
3.21.1. wiringOP-Python 的安装方法	143



3.21. 2. 40pin GPIO 口测试	145
3.21. 3. 40pin SPI 测试	148
3.21. 4. 40pin I2C 测试	151
3.21. 5. 40pin 的 UART 测试	155
3.22. 硬件看门狗测试	158
3.23. 查看 H618 芯片的 chipid	159
3.24. Python 相关说明	159
3.24. 1. Python 源码编译安装的方法	159
3.24. 2. Python 更换 pip 源的方法	160
3.25. 安装 Docker 的方法	161
3.26. Home Assistant 的安装方法	162
3.26. 1. 通过 docker 安装	162
3.26. 2. 通过 python 安装	166
3.27. OpenCV 的安装方法	167
3.27. 1. 使用 apt 来安装 OpenCV	167
3.28. 设置中文环境以及安装中文输入法	168
3.28. 1. Debian 系统的安装方法	168
3.28. 2. Ubuntu 20.04 系统的安装方法	174
3.28. 3. Ubuntu 22.04 系统的安装方法	179
3.29. 远程登录 Linux 系统桌面的方法	184
3.29. 1. 使用 NoMachine 远程登录	184
3.29. 2. 使用 VNC 远程登录	188
3.30. QT 的安装方法	190
3.31. ROS 安装方法	198
3.31. 1. Ubuntu20.04 安装 ROS 1 Noetic 的方法	198
3.31. 2. Ubuntu20.04 安装 ROS 2 Galactic 的方法	202
3.31. 3. Ubuntu22.04 安装 ROS 2 Humble 的方法	205
3.32. 安装内核头文件的方法	207
3.33. Linux 系统支持的部分编程语言测试	209
3.33. 1. Debian Bullseye 系统	209



3.33. 2. Debian Bookworm 系统	210
3.33. 3. Ubuntu Focal 系统	212
3.33. 4. Ubuntu Jammy 系统	214
3.34. 上传文件到开发板 Linux 系统中的方法	216
3.34. 1. 在 Ubuntu PC 中上传文件到开发板 Linux 系统中的方法	216
3.34. 2. 在 Windows PC 中上传文件到开发板 Linux 系统中的方法	219
3.35. 开关机 logo 使用说明	223
3.36. Linux5.4 打开开关机按键的方法	224
3.37. 关机和重启开发板的方法	226
4. Linux SDK——orangeipi-build 使用说明	227
4.1. 编译系统需求	227
4.2. 获取 linux sdk 的源码	229
4.2. 1. 从 github 下载 orangeipi-build	229
4.2. 2. 下载交叉编译工具链	230
4.2. 3. orangeipi-build 完整目录结构说明	232
4.3. 编译 u-boot	233
4.4. 编译 linux 内核	236
4.5. 编译 rootfs	240
4.6. 编译 linux 镜像	243
5. Orange Pi OS Arch 系统的使用说明	248
5.1. Orange Pi OS Arch 系统功能适配情况	248
5.2. Orange Pi OS Arch 系统用户向导使用说明	249
5.3. 设置 DT overlays 的方法	254
5.4. 安装软件的方法	256
6. Android 12 TV 系统使用说明	257
6.1. 已支持的 Android 版本	257
6.2. Android 12 TV 功能适配情况	257
6.3. 板载 LED 灯显示说明	258



6. 4. Android 返回上一级界面的方法	258
6. 5. ADB 的使用方法	258
6. 5. 1. 使用网络连接 adb 调试	258
6. 5. 2. 使用数据线连接 adb 调试	259
6. 6. 查看设置 HDMI 显示分辨率的方法	260
6. 6. 1. HDMI 转 VGA 显示测试	262
6. 7. WI-FI 的连接方法	263
6. 8. WI-FI hotspot 的使用方法	265
6. 9. 查看以太网口 IP 地址的方法	267
6. 10. 蓝牙的连接方法	268
6. 11. USB0 设置为 HOST 模式的方法	271
6. 12. USB 摄像头使用方法	272
6. 13. Android 系统 ROOT 说明	273
6. 14. 使用 MiracastReceiver 将手机屏幕投屏到开发板的方法	274
6. 15. 通过按键或红外遥控开关机的方法	277
6. 16. 40pin 接口 GPIO、UART、SPI 测试	278
6. 16. 1. 40pin 的 GPIO 口测试方法	278
6. 16. 2. 40pin 的 UART 测试方法	282
6. 16. 3. 40pin 的 SPI 测试方法	285
6. 16. 4. 40pin 的 I2C 测试方法	288
6. 16. 5. 40pin 的 PWM 测试	292
7. Android 12 源码的编译方法	296
7. 1. 下载 Android 12 的源码	296
7. 2. 编译 Android 12 的源码	297
8. 附录	300
8. 1. 用户手册更新历史	300
8. 2. 镜像更新历史	300



1. Orange Pi Zero 2w 的基本特性

1. 1. 什么是 Orange Pi Zero 2w

香橙派是一款开源的单板卡片电脑，新一代的arm64开发板，它可以运行Android TV 12、Ubuntu 和 Debian 等操作系统。香橙派开发板（Orange Pi Zero 2w）使用全志 H618 系统级芯片，同时可选 1GB 或 1.5GB 或 2GB 或 4GB LPDDR4 内存。

1. 2. Orange Pi Zero 2w 的用途

我们可以用它实现：

- 一台小型的 Linux 桌面计算机
- 一台小型的 Linux 网络服务器
- 安装 Klipper 上位机控制 3D 打印机
- Android TV 电视盒子

当然还有其他更多的功能，依托强大的生态系统以及各式各样的扩展配件，Orange Pi 可以帮助用户轻松实现从创意到原型再到批量生产的交付，是创客、梦想家、业余爱好者的理想创意平台。

1. 3. Orange Pi Zero 2w 是为谁设计的

Orange Pi 开发板不仅仅是一款消费品，同时也是给任何想用技术来进行创作创新的人设计的。它是一款简单、有趣、实用的工具，你可以用它去打造你身边的世界。



1. 4. Orange Pi Zero 2w 的硬件特性

硬件特性介绍	
CPU	全志 H618 四核 64 位 1.5GHz 高性能 Cortex-A53 处理器
GPU	Mali G31 MP2 Supports OpenGL ES 1.0/2.0/3.2、OpenCL 2.0
内存	1GB/1.5GB/2GB/4GB LPDDR4 (与 GPU 共享)
板载存储	TF 卡插槽、16MB SPI Flash
WIFI+蓝牙	• 20U5622 芯片、支持 IEEE 802.11 a/b/g/n/ac、BT5.0
视频输出	• Mini HDMI 2.0 接口
音频输出	• Mini HDMI 输出
电源	Type-C 5V/2A
USB 2.0 端口	Type-C USB2.0 x 2
40pin 扩展接口	用于扩展 GPIO、UART、I2C、SPI、PWM
24pin 扩展接口	用于扩展 USB2.0 x 2、100M 以太网、红外接收、音频输出、TV-OUT 输出、开关机按键、LRADC 按键 x 2
LED 灯	电源指示灯和状态指示灯
支持的操作系统	Android 12 TV, Debian11, Debian12, Ubuntu22.04, Ubuntu20.04, Orange Pi OS (Arch) 等
外观规格介绍	
PCB 尺寸	30mm x 65mm x 1.2mm
重量	12.5g
range Pi™ 是深圳市迅龙软件有限公司的注册商标	

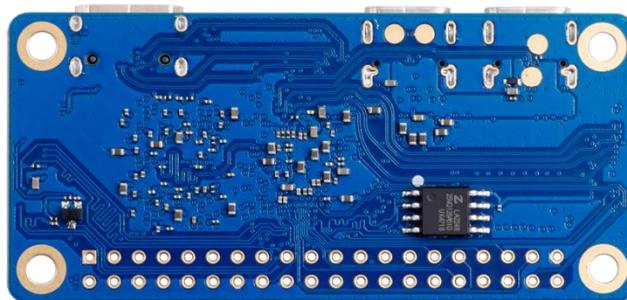


1. 5. Orange Pi Zero 2w 的顶层视图和底层视图

顶层视图：



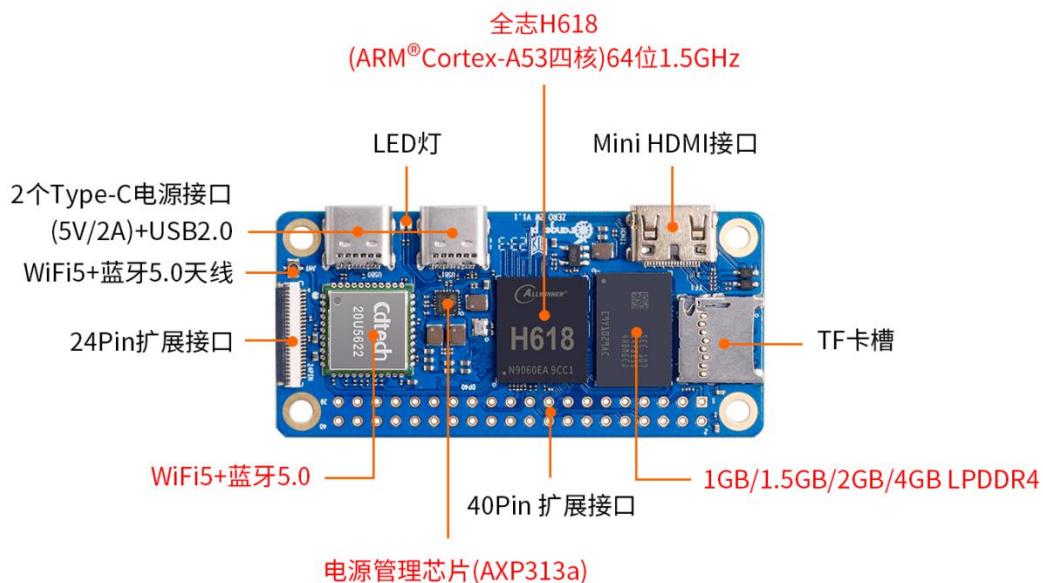
底层视图：



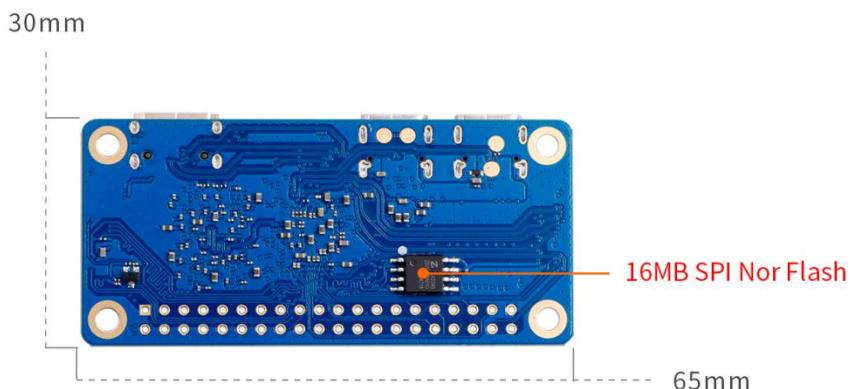


1. 6. Orange Pi Zero 2w 的接口详情图

正面视图

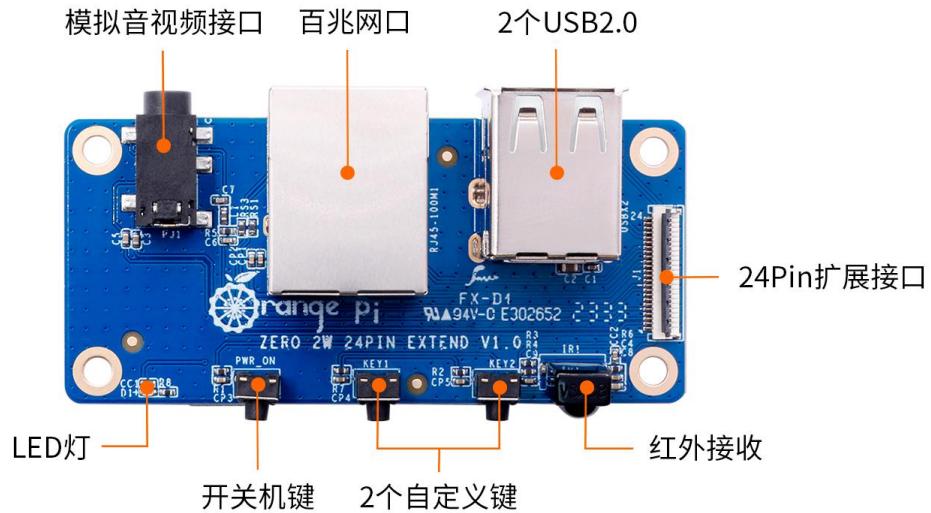


背面视图



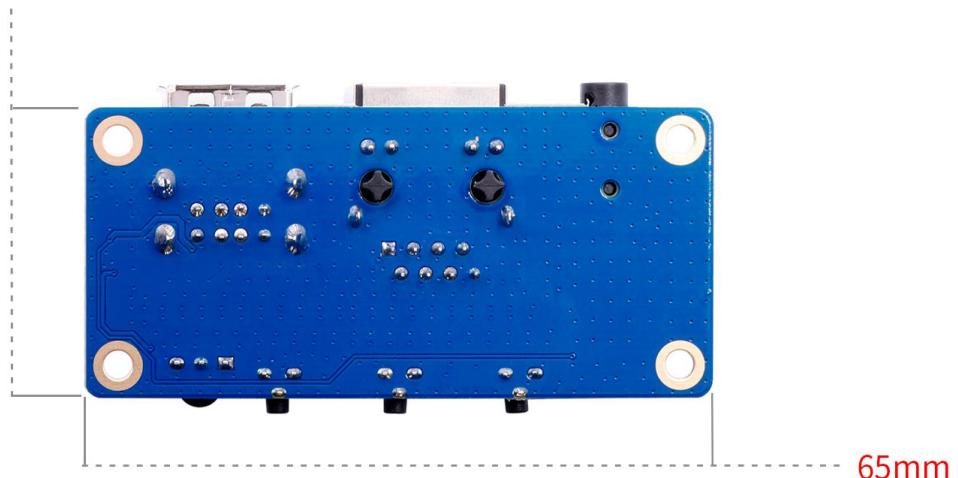
1. 7. Orange Pi Zero 2w 24pin 扩展板的接口详情图

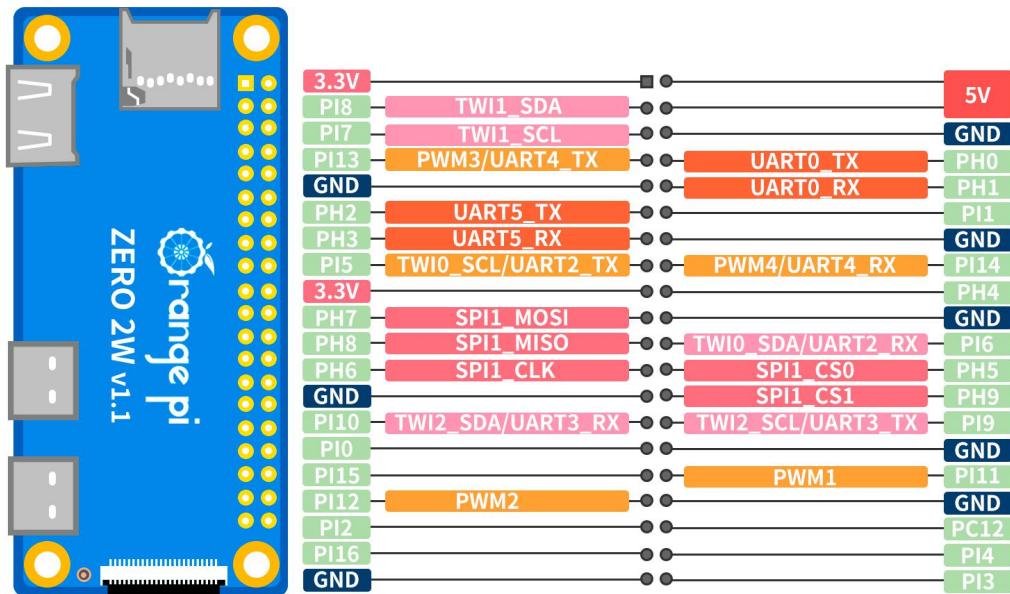
正面视图



背面视图

30mm





四个定位孔的直径都是 3.0mm。



2. 开发板使用介绍

2.1. 准备需要的配件

- 1) TF 卡，最小 8GB 容量的 **class10** 级或以上的高速闪迪卡

SanDisk 闪迪



使用其他品牌的TF卡（非闪迪的TF卡），如下图所示（包含但不仅限这些卡），已经有朋友反馈系统启动过程中会出现问题，比如系统启动到一半卡住不动，或者reboot命令无法正常使用，最后都是换了闪迪牌的TF卡后才解决的。所以如果您使用的是非闪迪牌的TF卡发现系统启动或者使用过程有问题，请更换闪迪牌的TF卡后再测试。



目前反馈在Orange Pi Zero 2w上启动有问题的部分TF卡

另外，在其他型号的开发板上能正常使用的TF卡并不能保证在Orange Pi Zero 2w上也一定能正常启动，这点请特别注意。

- 2) TF 卡读卡器，用于读写 TF 卡



- 3) Mini HDMI 转 HDMI 连接线，用于将开发板连接到 HDMI 显示器或者电视进行显示



4) 电源，如果有 5V/2A 或 5V/3A 的电源头那就只需要准备一根下面左边图片所示的 USB 转 Type C 接口的数据线，另外也可以使用类似下面右边图片所示的线和电源头一体的 5V/2A 或者 5V/3A 的高品质 USB Typc C 接口电源适配器。

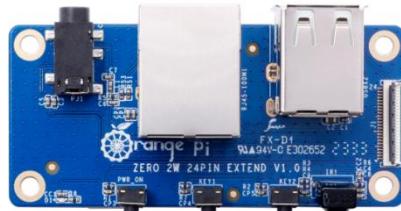


开发板上的两个Type-C接口都可以用来供电。

两个Type-C接口都可供电，二选一即可



5) 24pin 扩展板



6) USB接口的鼠标和键盘，只要是标准USB接口的鼠标和键盘都可以，鼠标和键盘可以用来控制Orange Pi开发板



7) 红外遥控器，主要用于控制安卓TV系统



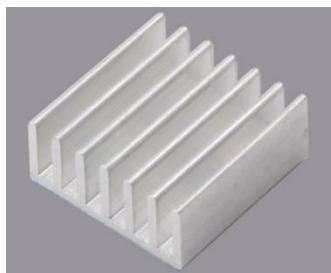
注意，空调的遥控或者电视机的遥控是无法控制Orange Pi开发板的，默认只有Orange Pi提供的遥控才可以。

8) 网线，用于将开发板连接到因特网

9) AV 视频线，如果希望通过 TV-OUT 接口而不是 HDMI 接口来显示视频，那么就需要通过 AV 视频线将开发板连接到电视



10) 散热片，如果担心开发板的温度过高，可以加些散热片，散热片贴在 H618 芯片和内存芯片上即可





11) 5V 的散热风扇，如下图所示，开发板的 40pin 接口上有 5V 和 GND 引脚可以接散热风扇，40pin 排针的间距为 **2.54mm**，散热风扇的电源接口参照这个规格去购买即可。



注意，开发板插上电源后 **5V**引脚就可以直接使用，无需其他设置，另外 **5V**引脚输出的电压是无法通过软件调节和关闭的。

40pin接口上的排针默认是不焊接的，需要自己焊接上去才能使用。

12) Type-C 转 USB 线，用于接 USB 设备



13) USB 转 TTL 模块和杜邦线，使用串口调试功能时，需要 USB 转 TTL 模块和杜邦线来连接开发板和电脑



注意，开发板使用的TTL电平是 **3.3v**的，除了上图所示的USB转TTL模块外，其他类似的 **3.3v**的USB转TTL模块一般也都是可以的。

14) 安装有 Ubuntu 和 Windows 操作系统的 X64 电脑

1	Ubuntu22.04 PC	可选，用于编译 Android 和 Linux 源码
---	----------------	----------------------------



2. 2. 下载开发板的镜像和相关的资料

1) 中文版资料的下载网址为

<http://www.orangepi.cn/html/hardWare/computerAndMicrocontrollers/service-and-support/Orange-Pi-Zero-2W.html>

2) 英文版资料的下载网址为

<http://www.orangepi.org/html/hardWare/computerAndMicrocontrollers/service-and-support/Orange-Pi-Zero-2W.html>

3) 资料主要包含

- a. **Android 源码**: 保存在百度云盘和谷歌网盘上
- b. **Linux 源码**: 保存在 Github 上
- c. **Android 镜像**: 保存在百度云盘和谷歌网盘上
- d. **Ubuntu 镜像**: 保存在百度云盘和谷歌网盘上
- e. **Debian 镜像**: 保存在百度云盘和谷歌网盘上
- f. **Orange Pi OS (Arch)镜像**: 保存在百度云盘和谷歌网盘上
- g. **用户手册和原理图**: 芯片相关的数据手册也会放在这里
- h. **官方工具**: 主要包括开发板使用过程中需要用到的软件

2. 3. 基于 Windows PC 将 Linux 镜像烧写到 TF 卡的方法

注意，这里说的Linux镜像具体指的是从Orange Pi资料下载页面下载的Debian或者Ubuntu这样的Linux发行版镜像。

2. 3. 1. 使用 balenaEtcher 烧录 Linux 镜像的方法

- 1) 首先准备一张 8GB 或更大容量的 TF 卡，TF 卡的传输速度必须为 **class10** 级或 **class10** 级以上，建议使用闪迪等品牌的 TF 卡
- 2) 然后使用读卡器把 TF 卡插入电脑
- 3) 从 [Orange Pi 的资料下载页面](#)下载想要烧录的 Linux 操作系统镜像文件压缩包，

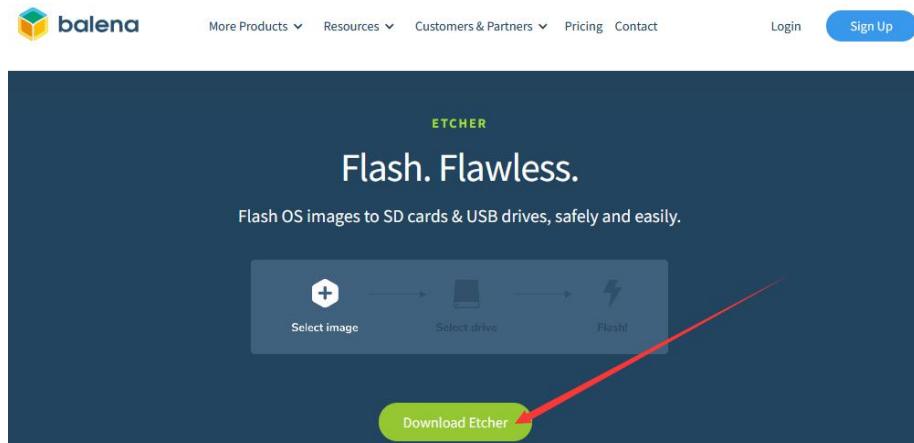


然后使用解压软件解压，解压后的文件中，以“.img”结尾的文件就是操作系统的镜像文件，大小一般都在1GB以上

4) 然后下载Linux镜像的烧录软件——**balenaEtcher**，下载地址为

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

5) 进入 balenaEtcher 下载页面后，点击绿色的下载按钮会跳到软件下载的地方



6) 然后可以选择下载 balenaEtcher 的 Portable 版本的软件，Portable 版本无需安装，双击打开就可以使用

[DOWNLOAD](#)

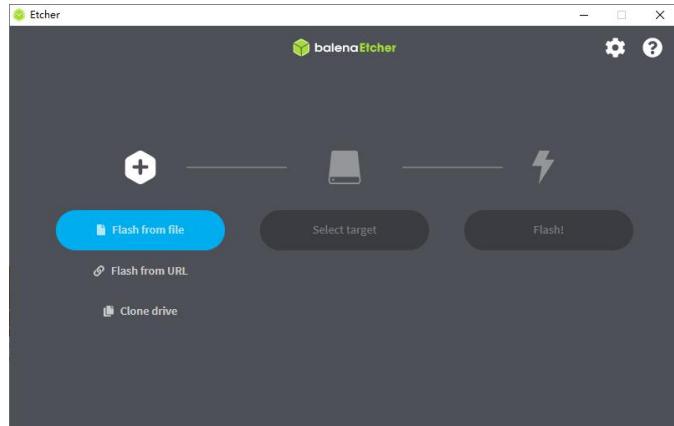
Download Etcher

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

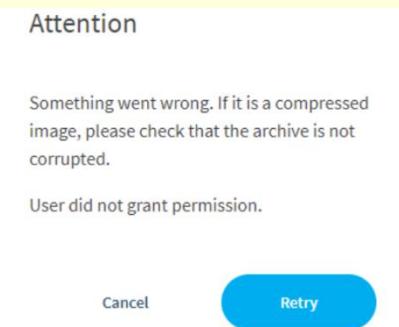
Looking for [Debian \(.deb\) packages](#) or [Red Hat \(.rpm\) packages](#)?

OSS hosting by [cloudsmith](#)

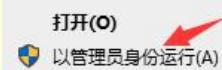
7) 如果下载的是需要安装版本的 balenaEtcher，请先安装再使用。如果下载的 Portable 版本 balenaEtcher，直接双击打开即可，打开后的 balenaEtcher 界面如下图所示



打开 **balenaEtcher** 时如果提示下面的错误：



请选择 **balenaEtcher** 后点击右键，然后选择以管理员身份运行。

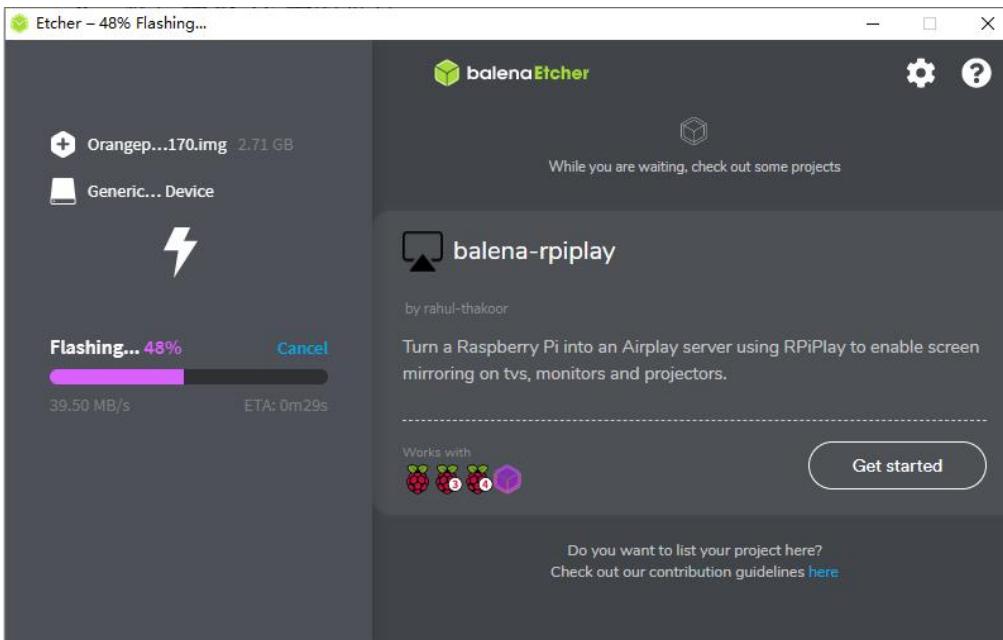


8) 使用 **balenaEtcher** 烧录 Linux 镜像的具体步骤如下所示

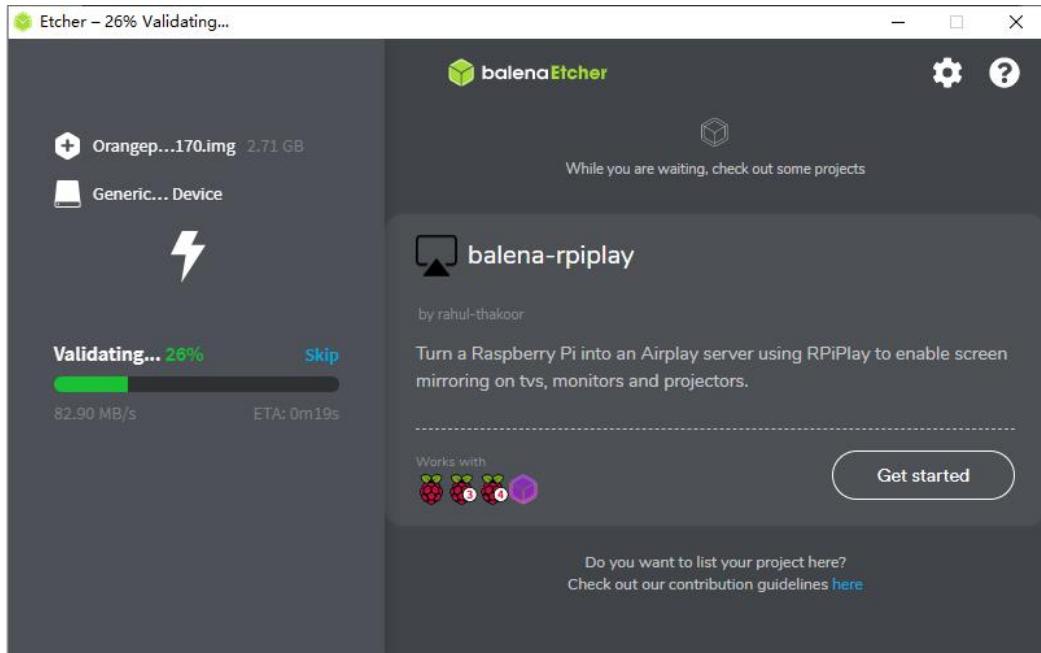
- a. 首先选择要烧录的 Linux 镜像文件的路径
- b. 然后选择 TF 卡的盘符
- c. 最后点击 Flash 就会开始烧录 Linux 镜像到 TF 卡中



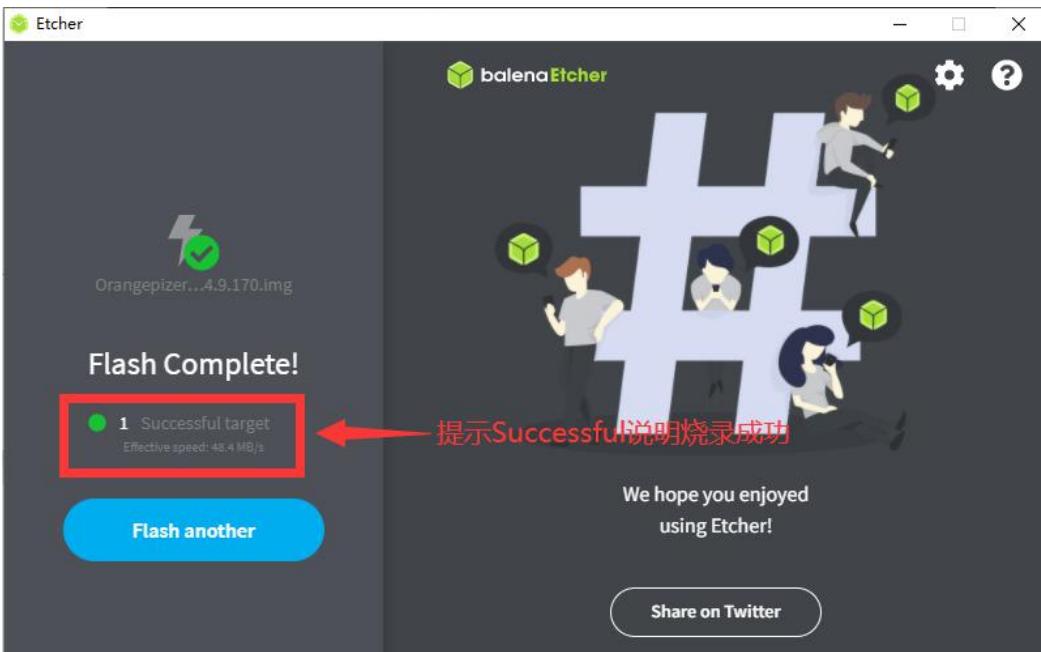
9) balenaEtcher 烧录 Linux 镜像的过程显示的界面如下图所示，另外进度条显示紫色表示正在烧录 Linux 镜像到 TF 卡中



10) Linux 镜像烧录完后，balenaEtcher 默认还会对烧录到 TF 卡中的镜像进行校验，确保烧录过程没有出问题。如下图所示，显示绿色的进度条就表示镜像已经烧录完成，balenaEtcher 正在对烧录完成的镜像进行校验



11) 成功烧录完成后 balenaEtcher 的显示界面如下图所示, 如果显示绿色的指示图标说明镜像烧录成功, 此时就可以退出 balenaEtcher, 然后拔出 TF 卡插入到开发板的 TF 卡槽中使用了



2. 3. 2. 使用 Win32Diskimager 烧录 Linux 镜像的方法

1) 首先准备一张 8GB 或更大容量的 TF 卡, TF 卡的传输速度必须为 **class10** 级或 **class10** 级以上, 建议使用闪迪等品牌的 TF 卡



2) 然后使用读卡器把 TF 卡插入电脑

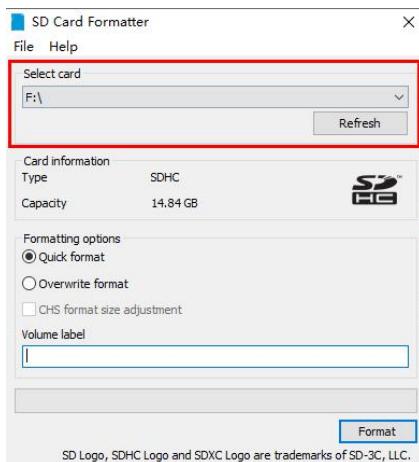
3) 接着格式化 TF 卡

a. 可以使用 **SD Card Formatter** 这个软件格式化 TF 卡，其下载地址为

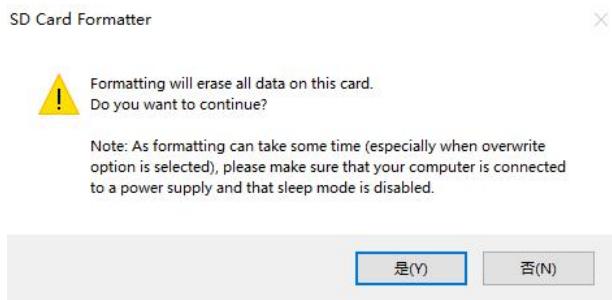
https://www.sdcard.org/downloads/formatter/eula_windows/SDCardFormatterv5_WinEN.zip

b. 下载完后直接解压安装即可，然后打开软件

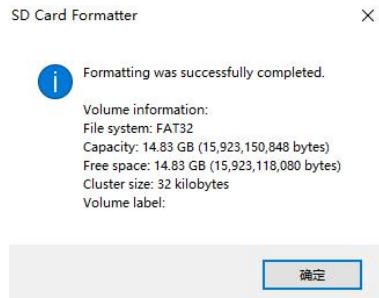
c. 如果电脑只插入了 TF 卡，则“**Select card**”一栏中会显示 TF 卡的盘符，如果电脑插入了多个 USB 存储设备，可以通过下拉框选择 TF 卡对应的盘符



d. 然后点击“**Format**”，格式化前会弹出一个警告框，选择“是(Y)”后就会开始格式化



e. 格式化完 TF 卡后会弹出下图所示的信息，点击确定即可



4) 从[Orange Pi的资料下载页面](#)下载想要烧录的Linux操作系统镜像文件压缩包，然后使用解压软件解压，解压后的文件中，以“.img”结尾的文件就是操作系统的镜像文件，大小一般都在1GB以上

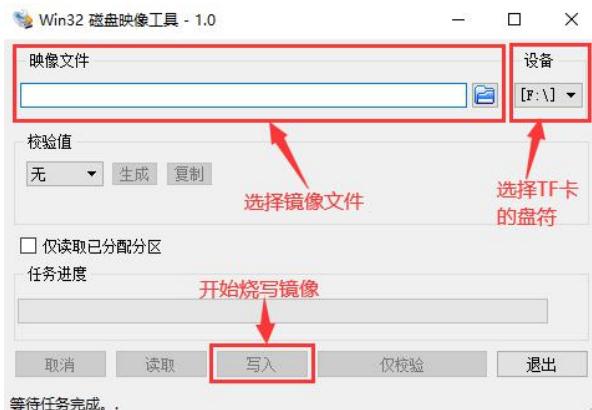
5) 使用 **Win32Diskimager** 烧录 Linux 镜像到 TF 卡

a. Win32Diskimager 的下载页面为

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

b. 下载完后直接安装即可，Win32Diskimager 界面如下所示

- 首先选择镜像文件的路径
- 然后确认下 TF 卡的盘符和“设备”一栏中显示的一致
- 最后点击“写入”即可开始烧录



c. 镜像写入完成后，点击“退出”按钮退出即可，然后就可以拔出 TF 卡插到开发板中启动



2. 4. 基于 Ubuntu PC 将 Linux 镜像烧写到 TF 卡的方法

注意，这里说的Linux镜像具体指的是从Orange Pi资料下载页面下载的Debian或者Ubuntu这样的Linux发行版镜像，Ubuntu PC指的是安装了Ubuntu系统的个人电脑。

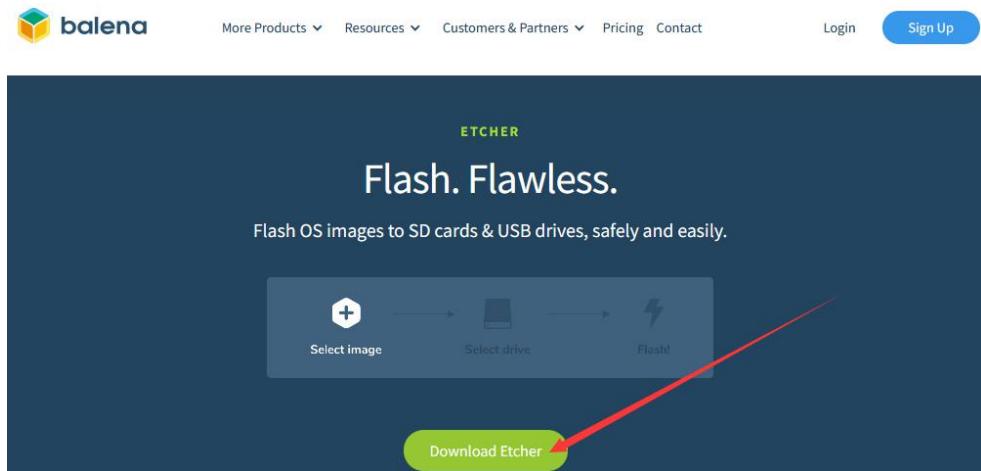
1) 首先准备一张 8GB 或更大容量的 TF 卡，TF 卡的传输速度必须为 **class10** 级或 **class10** 级以上，建议使用闪迪等品牌的 TF 卡

2) 然后使用读卡器把 TF 卡插入电脑

3) 下载 balenaEtcher 软件，下载地址为

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

4) 进入 balenaEtcher 下载页面后，点击绿色的下载按钮会跳到软件下载的地方



5) 然后选择下载 Linux 版本的软件即可

[DOWNLOAD](#)

Download Etcher

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

[Looking for Debian \(.deb\) packages or Red Hat \(.rpm\) packages?](#)

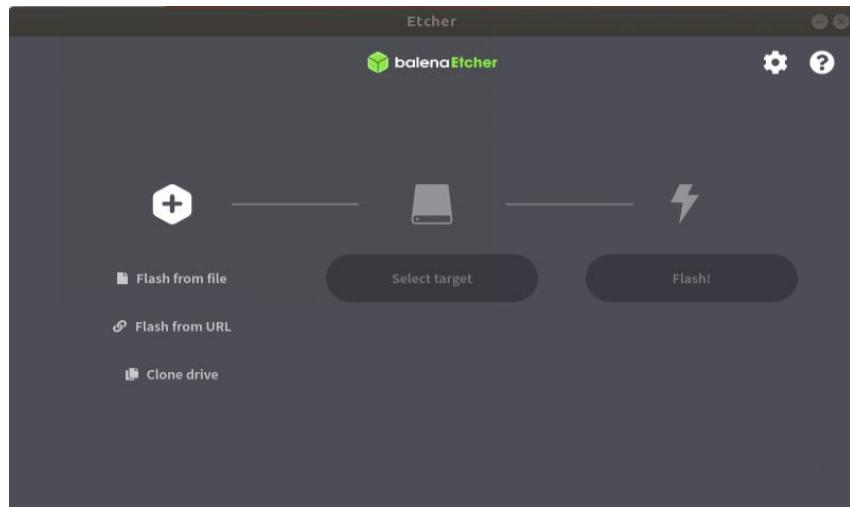
- 6) 从 [Orange Pi 的资料下载页面](#) 下载想要烧录的 Linux 操作系统镜像文件压缩包，然后使用解压软件解压，解压后的文件中，以 “.img” 结尾的文件就是操作系统的镜像文件，大小一般都在 1GB 以上。7z 结尾的压缩包的解压命令如下所示：

```
test@test:~$ 7z x orangepizero2w_1.0.0_ubuntu_focal_desktop_linux6.1.31.7z
test@test:~$ ls orangepizero2w_1.0.0_ubuntu_focal_desktop_linux6.1.31.*
orangepizero2w_1.0.0_ubuntu_focal_desktop_linux6.1.31.7z
orangepizero2w_1.0.0_ubuntu_focal_desktop_linux6.1.31.sha #校验和文件
orangepizero2w_1.0.0_ubuntu_focal_desktop_linux6.1.31.img #镜像文件
```

- 7) 解压镜像后可以先用 **sha256sum -c *.sha** 命令计算下校验和是否正确，如果提示成功说明下载的镜像没有错，可以放心的烧录到 TF 卡，如果提示校验和不匹配说明下载的镜像有问题，请尝试重新下载

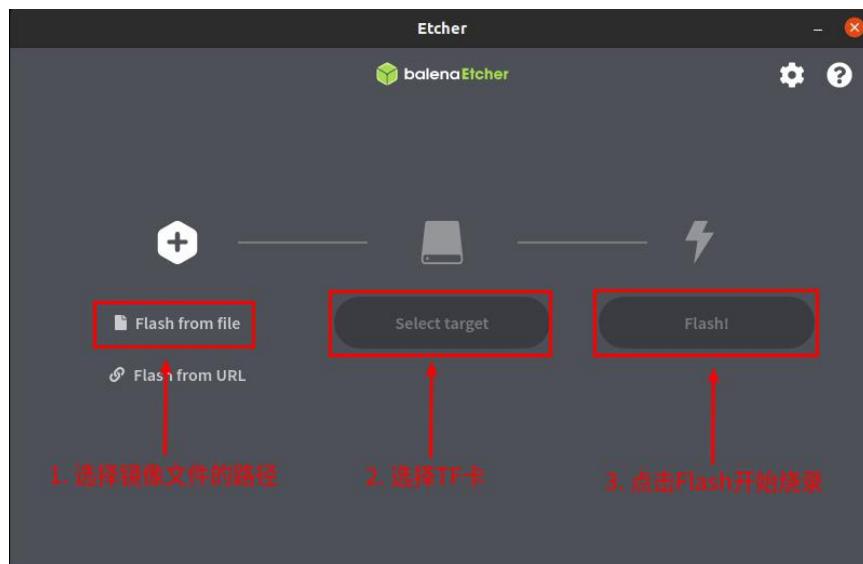
```
test@test:~$ sha256sum -c *.sha
orangepizero2w_1.0.0_ubuntu_focal_desktop_linux6.1.31.img: 成功
```

- 8) 然后在 Ubuntu PC 的图形界面双击 **balenaEtcher-1.14.3-x64.AppImage** 即可打开 balenaEtcher（无需安装），balenaEtcher 打开后的界面显示如下图所示

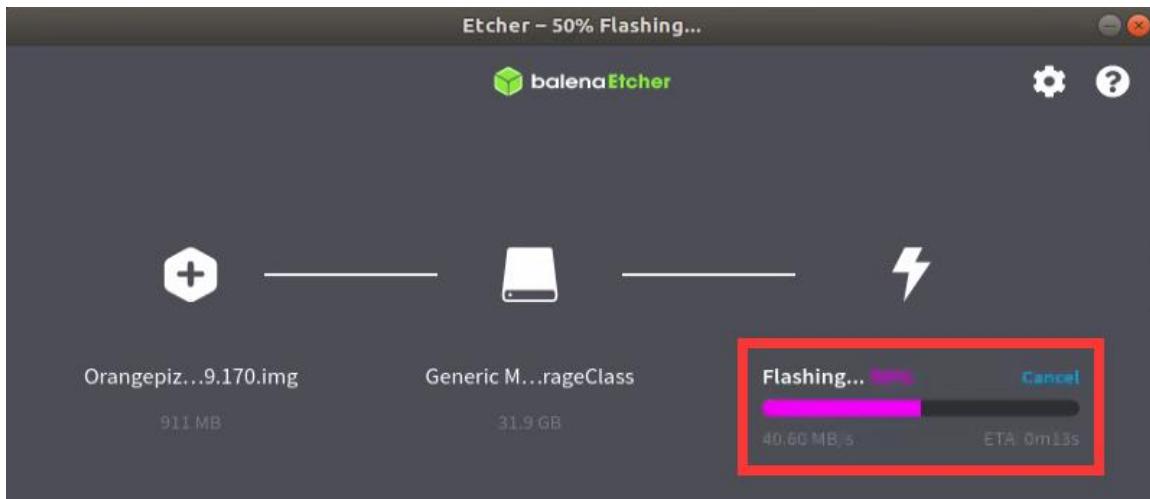


9) 使用 balenaEtcher 烧录 Linux 镜像的具体步骤如下所示

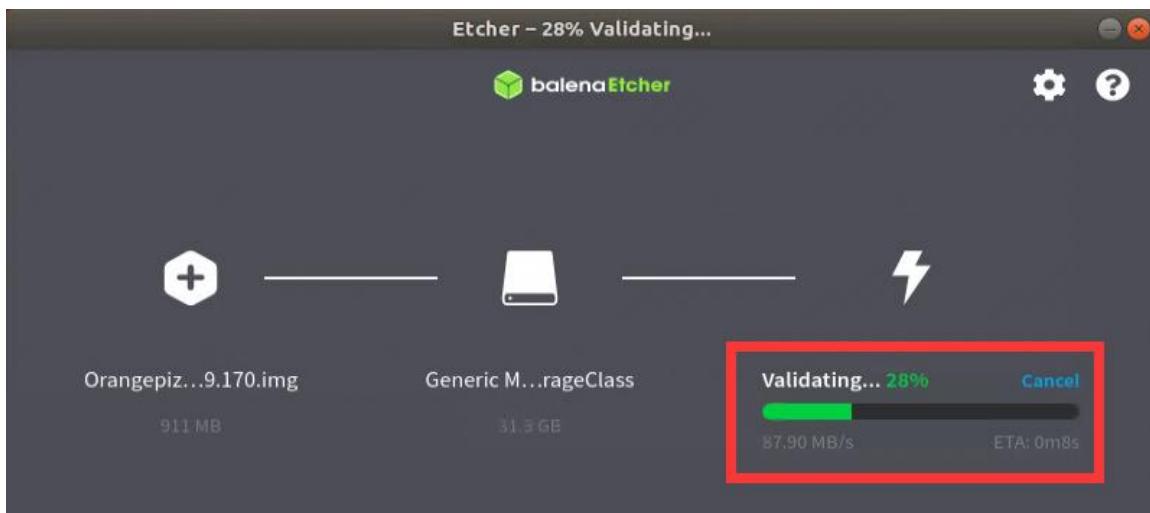
- 首先选择要烧录的 Linux 镜像文件的路径
- 然后选择 TF 卡的盘符
- 最后点击 Flash 就会开始烧录 Linux 镜像到 TF 卡中



10) balenaEtcher 烧录 Linux 镜像的过程显示的界面如下图所示，另外进度条显示紫色表示正在烧录 Linux 镜像到 TF 卡中



11) Linux 镜像烧录完后，balenaEtcher 默认还会对烧录到 TF 卡中的镜像进行校验，确保烧录过程没有出问题。如下图所示，显示绿色的进度条就表示镜像已经烧录完成，balenaEtcher 正在对烧录完成的镜像进行校验



12) 成功烧录完成后 balenaEtcher 的显示界面如下图所示，如果显示绿色的指示图标说明镜像烧录成功，此时就可以退出 balenaEtcher，然后拔出 TF 卡插入到开发板的 TF 卡槽中使用了



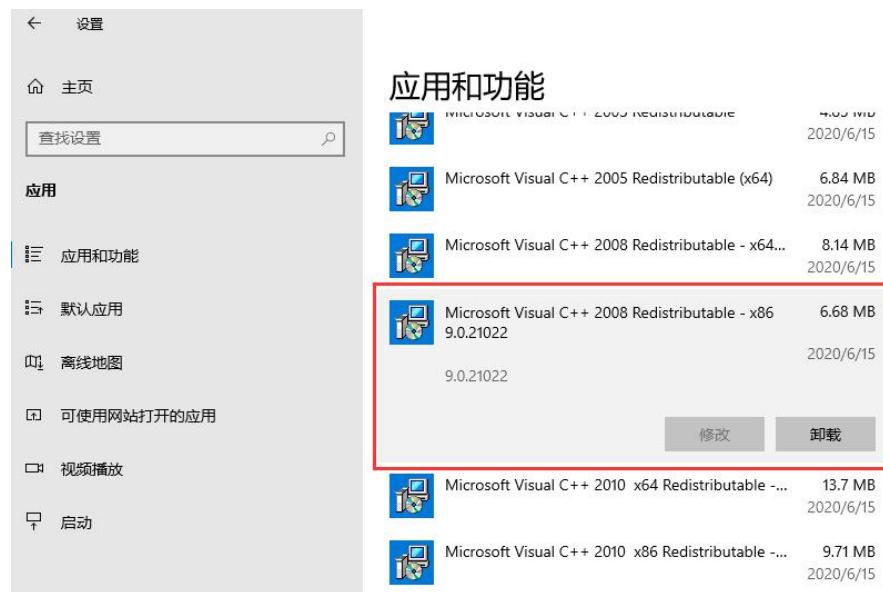
2.5. 烧写 Android 镜像到 TF 卡的方法

开发板的 Android 镜像只能在 Windows 平台下使用 **PhoenixCard** 软件烧录到 TF 卡中，**PhoenixCard** 软件的版本必须为 **PhoenixCard-4.2.8**。

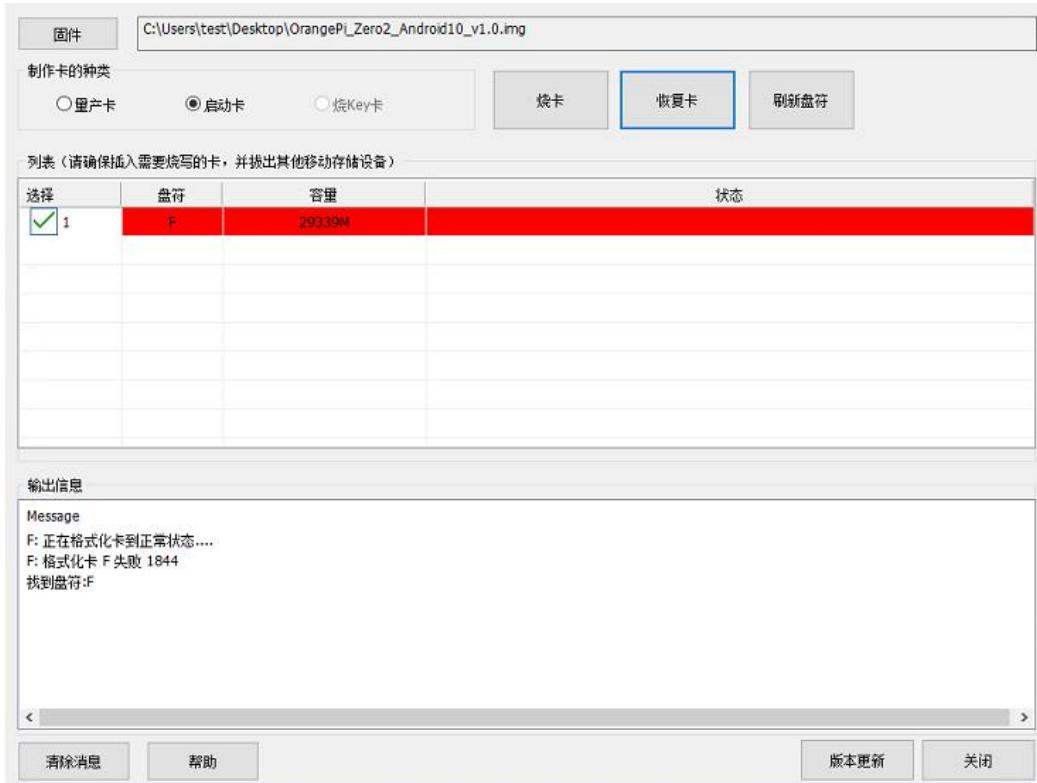
请不要用烧录 Linux 镜像的软件，如 Win32Diskimager 或者 balenaEtcher 来烧录安卓镜像。

另外 **PhoenixCard** 这款软件没有 Linux 和 Mac 平台的版本，所以在 Linux 和 Mac 平台下是无法烧录安卓镜像到 TF 卡中的。

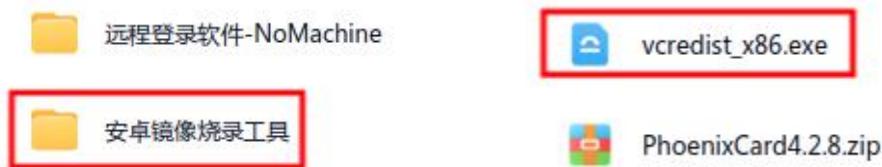
- 1) 首先请确保 Windows 系统已经安装了 **Microsoft Visual C++ 2008 Redistrbutable - x86**



2) 如果没有安装 **Microsoft Visual C++ 2008 Redistributable - x86**，使用 PhoenixCard 格式化 TF 卡或者烧录 Android 镜像会提示下面的错误



3) Microsoft Visual C++ 2008 Redistributable - x86 的安装包可以从 Orange Pi Zero 2w 的官方工具中下载到，也可以去[微软官网](#)下载



4) 然后准备一张 8GB 或更大容量的 TF 卡，TF 卡的传输速度必须为 **class10** 级或 **class10** 级以上，建议使用闪迪等品牌的 TF 卡

5) 然后使用读卡器把 TF 卡插入电脑

6) 从 [Orange Pi 的资料下载页面](#) 下载 Android 镜像和 PhoenixCard 烧写工具，请确保 PhonenixCrad 工具的版本为 **PhonixCard-4.2.8**，**请不要用低于 4.2.8 版本的 PhonixCard 软件来烧录 Android 镜像**，低于这个版本的 PhonixCard 工具烧写的 Android 镜像可能会有问题

<input type="checkbox"/> Balena-etcher	-	2020-11-04 13:48
<input type="checkbox"/> Android测试APP	-	2020-11-04 13:48
<input type="checkbox"/> win32diskimager-1.0.0-install.exe	12M	2020-11-04 13:48
<input type="checkbox"/> vcredist_x86.exe	4.3M	2021-04-25 21:25
<input type="checkbox"/> security.tar.gz	2.3M	2021-06-16 14:07
<input type="checkbox"/> SDCardFormatterv5_WinEN.zip	6M	2020-11-04 13:48
<input type="checkbox"/> PhonixCard-4.2.5.zip	4.9M	2021-03-08 18:07
<input type="checkbox"/> PhoenixCard4.2.8.zip	10.2M	2022-01-05 13:33
<input type="checkbox"/> MobaXterm_Portable_v20.3.zip	24.9M	2020-11-04 13:48

请下载这个最新版本的软件

7) 然后使用解压软件解压下载的 Android 镜像的压缩包，解压后的文件中，以“.img”结尾的文件就是 Android 镜像文件，大小在 1GB 以上。如果不知道怎么解压 Android 镜像的压缩包，可以安装一个 [360 压缩软件](#) 来解压镜像。



8) 然后使用解压软件解压 **PhoenixCard4.2.8.zip**，此软件无需安装，在解压后的文件夹中找到 PhoenixCard 打开即可

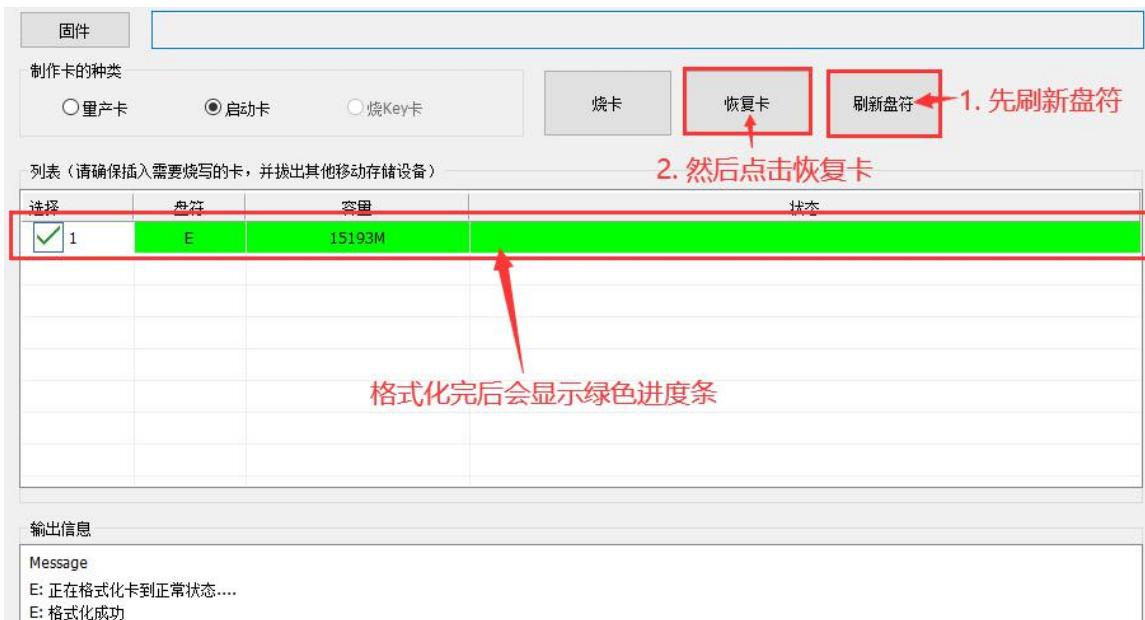


ludosocket.dll	2019/4/21 11:03	应用程序	24 KB
Mbr2Gpt.dll	2019/2/27 13:34	应用程序扩展	9 KB
option.cfg	2019/4/22 15:57	CFG 文件	1 KB
ParserManager.dll	2019/1/10 14:51	应用程序扩展	81 KB
PhoenixCard	2019/12/31 11:29	应用程序	1,748 KB
PhoenixCard.exe	2019/12/31 10:42	LAN 文件	3 KB

9) 打开 PhoenixCard 后, 如果 TF 卡识别正常, 会在中间的列表中显示 TF 卡的盘符和容量, **请务必确认显示的盘符和你想烧录的 TF 卡的盘符是一致的**, 如果没有显示可以尝试拔插下 TF 卡, 或者点击 PhoenixCard 中的“刷新盘符”按钮



10) 确认完盘符后, 先格式化 TF 卡, 点击 PhoenixCard 中“恢复卡”按钮即可 (如果“恢复卡”按钮为灰色的无法按下, 可以先点击下“刷新盘符”按钮)

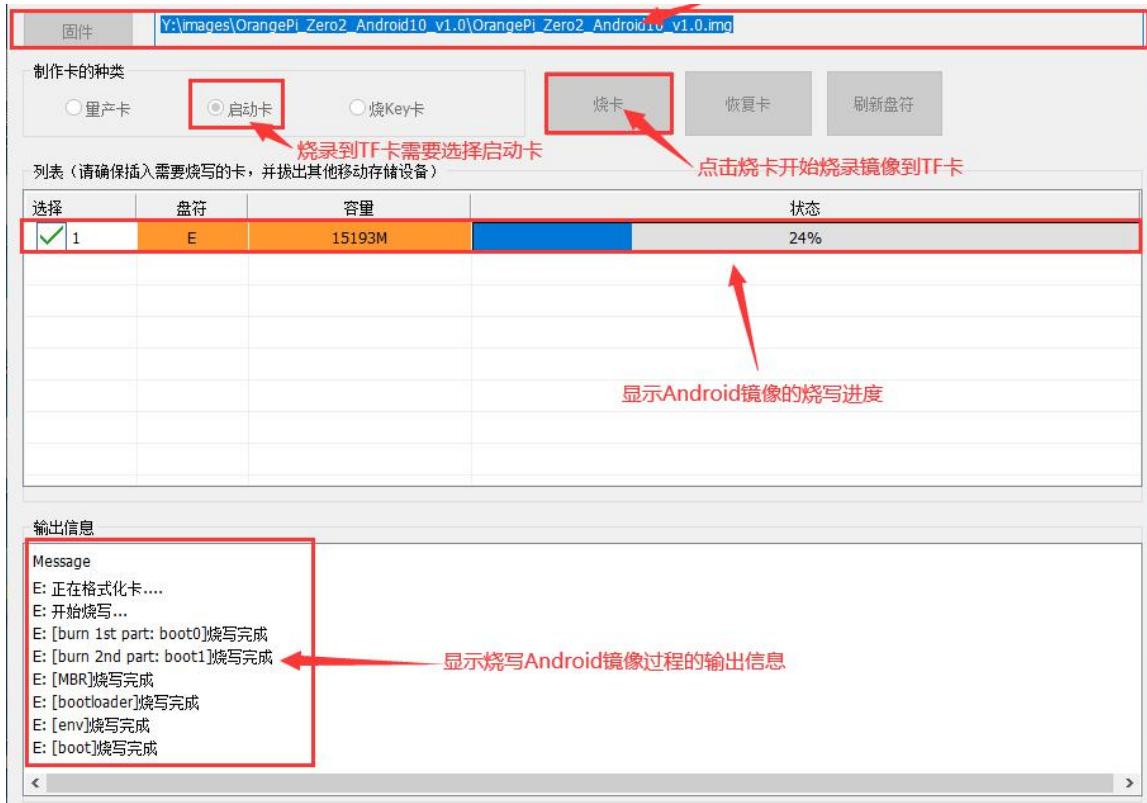


如果格式化有问题, 请尝试拔插下 TF 卡后再测试, 如果重新拔插 TF 卡后还是有问题, 可以重启下 Window 电脑或者换一台电脑再试下。

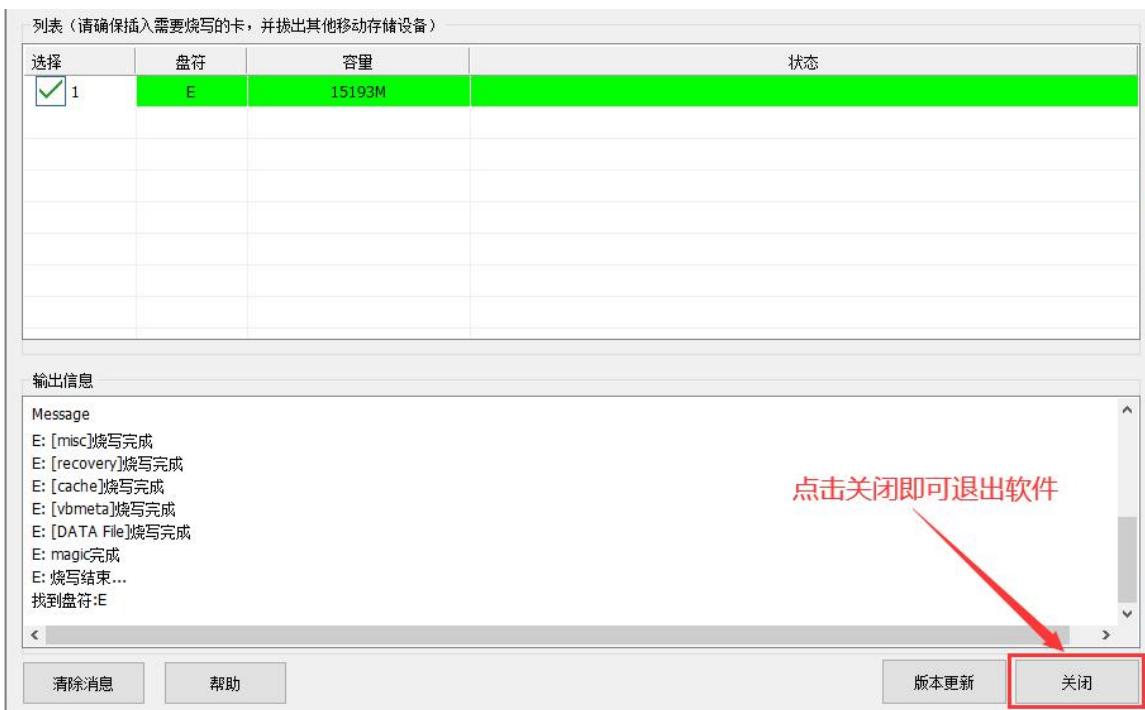


11) 然后开始将 Android 镜像写入 TF 卡

- a. 首先在“固件”一栏中选择 Android 镜像的路径
- b. 在“制作卡的种类”中选择“启动卡”
- c. 然后点击“烧卡”按钮就会开始烧录



12) 烧录完后 PhoenixCard 的显示如下图所示，此时点击“关闭”按钮即可退出 PhoenixCard，然后就可以把 TF 卡从电脑中拔出来插到开发板中启动了



烧录完Android系统后在Windows中TF卡只能看到一个 128 MB的分区，显示的分区如下图所示（有些电脑可能会弹出二十几个磁盘分区，但也只能打开 128 MB的那个分区），请注意，这是正常的，请不要以为TF卡烧坏了。之所以这样，是因为安卓系统总共有二十几个分区，但大部分分区在Windows系统中是无法正常识别的。此时，请放心的拔下TF卡然后插入开发板中启动即可。



安卓系统启动后，使用下面的命令可以看到TF卡中的这二十几个分区：

```
console:/ # ls /dev/block/mmcblk0*
/dev/block/mmcblk0      /dev/block/mmcblk0p17  /dev/block/mmcblk0p25
/dev/block/mmcblk0p1    /dev/block/mmcblk0p18  /dev/block/mmcblk0p3
/dev/block/mmcblk0p10   /dev/block/mmcblk0p19  /dev/block/mmcblk0p4
/dev/block/mmcblk0p11   /dev/block/mmcblk0p20  /dev/block/mmcblk0p5
/dev/block/mmcblk0p12   /dev/block/mmcblk0p21  /dev/block/mmcblk0p6
/dev/block/mmcblk0p13   /dev/block/mmcblk0p22  /dev/block/mmcblk0p7
/dev/block/mmcblk0p14   /dev/block/mmcblk0p23  /dev/block/mmcblk0p8
/dev/block/mmcblk0p15   /dev/block/mmcblk0p24  /dev/block/mmcblk0p9
/dev/block/mmcblk0p16
console:/ #
```

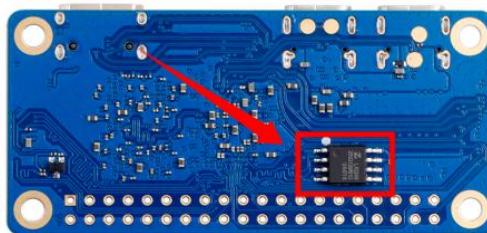
使用 **df -h** 命令可以看到 16GB 的TF卡烧录完安卓系统后大概还有 11 GB 的空间可以用（二十几个分区并不会都挂载到安卓系统中，重点关注这些能看到的分区即可）。



```
console:/ # df -h
Filesystem           Size  Used  Avail Use% Mounted on
tmpfs                 727M  1.1M  726M   1% /dev
tmpfs                 727M     0  727M   0% /mnt
/dev/block/mmcblk0p19      11M 136K   11M   2% /metadata
/dev/block/dm-0          782M 779M   2.4M 100% /
/dev/block/dm-1          104M 103M  332K 100% /vendor
/dev/block/dm-3          6.5M  6.5M  24K 100% /vendor_dlkm
/dev/block/dm-2          250M 249M  788K 100% /product
/dev/block/mmcblk0p23      16M     0   16M   0% /oem
tmpfs                 727M  8.0K  727M   1% /apex
tmpfs                 727M 532K  726M   1% /linkerconfig
/dev/block/mmcblk0p25      11G 904M   11G   8% /data
tmpfs                 727M     0  727M   0% /data_mirror
/dev/block/mmcblk0p24      16M     0   16M   0% /Reserve0
/dev/fuse                11G 904M   11G   8% /mnt/user/0/emulated
/dev/block/vold/public:179,1 128M 5.3M  122M   5% /mnt/media_rw/0000-0000
/dev/fuse                128M 5.3M  122M   5% /mnt/user/0/0000-0000
console:/ #
```

2. 6. 板载 SPI Flash 中的微型 linux 系统使用说明

开发板上有一个 16MB 大小的 SPI Flash，其所在位置如下图所示：



SPI Flash 中默认烧录有一个微型的 Linux 系统，此系统主要用于证明开发板是能正常启动的。当拿到开发板后，不用烧录系统到 TF 卡中，只需要给开发板接上 Type-C 电源就能启动 SPI Flash 中的微型 Linux 系统。此系统的主要功能有：

- 开机进入内核后，会设置绿色的 LED 灯闪烁；
- 如果开发板接了 HDMI 屏幕，系统启动完成后，在 HDMI 屏幕中能看到微型 Linux 系统的命令行界面。

再强调下，SPI Flash 中微型的 Linux 系统只是用于证明开发板是能正常启动的（不需要烧录系统就可以点亮开发板），如果你发现 SPI Flash 中的系统存在其他问题（比如串口无法登录），请直接忽略。

如果想正常使用开发板，请烧录 Ubuntu、Debian 等 Linux 镜像或者安卓镜像到 TF 卡中，然后再使用。



2.7. 启动香橙派开发板

- 1) 将烧录好镜像的 TF 卡插入香橙派开发板的 TF 卡插槽中。
- 2) 开发板有 Mini HDMI 接口，可以通过 Mini HDMI 转 HDMI 连接线把开发板连接到电视或者 HDMI 显示器。
- 3) 如果购买了 24pin 的扩展板，可以将 24pin 的扩展板通过排线接到开发板的 24pin 接口中。
- 4) 接上 USB 鼠标和键盘，用于控制香橙派开发板。
- 5) 连接一个 5V/2A (5V/3A 的也可以) 的 USB Type C 接口的高品质的电源适配

切记不要插入电压输出大于 5V 的电源适配器，会烧坏开发板。

系统上电启动过程中很多不稳定的现象基本都是供电有问题导致的，所以一个靠谱的电源适配器很重要。如果启动过程中发现有不断重启的现象，请更换下电源或者 Type C 数据线再试下。

开发板上的两个Type-C接口都可以用来供电。



- 6) 然后打开电源适配器的开关，如果一切正常，此时 HDMI 显示器就能看到系统的启动画面了。
- 7) 如果想通过调试串口查看系统的输出信息，请使用串口线将开发板连接到电脑，串口的连接方法请参看[调试串口的使用方法](#)一节。



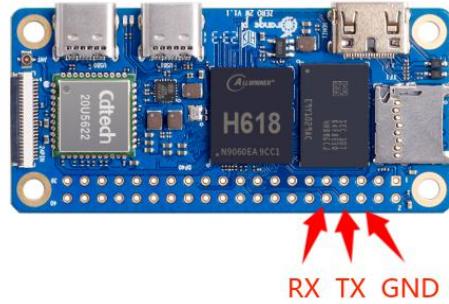
2.8. 调试串口的使用方法

2.8.1. 调试串口的连接说明

- 首先需要准备一个 3.3V 的 USB 转 TTL 模块，然后将 USB 转 TTL 模块的 USB 接口一端插入到电脑的 USB 接口中



- 开发板的调试串口 GND、TX 和 RX 引脚的对应关系如下图所示

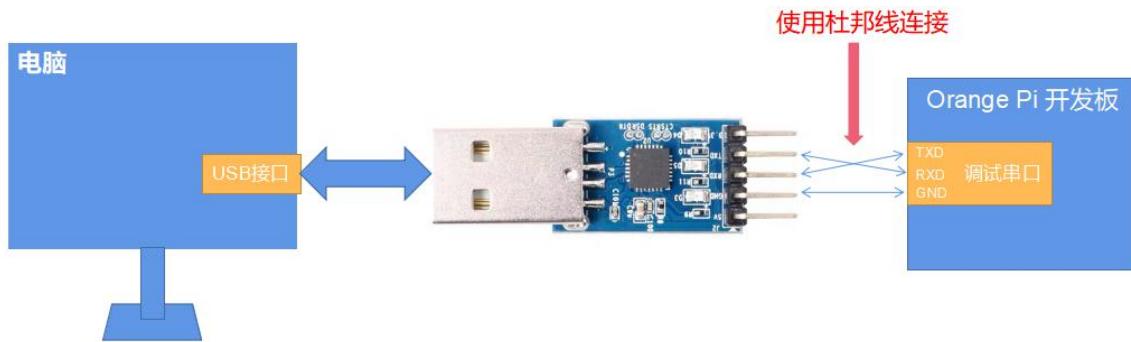


注意：40pin接口上的排针默认是不焊接的，需要自己焊接上去才能使用。

- USB 转 TTL 模块 GND、TX 和 RX 引脚需要通过杜邦线连接到开发板的调试串口上

- USB 转 TTL 模块的 GND 接到开发板的 GND 上
- USB 转 TTL 模块的 **RX** 接到开发板的 **TX** 上
- USB 转 TTL 模块的 **TX** 接到开发板的 **RX** 上

- USB 转 TTL 模块连接电脑和 Orange Pi 开发板的示意图如下所示



USB转TTL模块连接电脑和 Orange Pi 开发板的示意图

串口的 TX 和 RX 是需要交叉连接的，如果不仔细区分 TX 和 RX 的顺序，可以把串口的 TX 和 RX 先随便接上，如果测试串口没有输出再交换下 TX 和 RX 的顺序，这样就总有一种顺序是对的。

2.8.2. Ubuntu 平台调试串口的使用方法

Linux 下可以使用的串口调试软件有很多，如 **putty**、**minicom** 等，下面演示下 **putty** 的使用方法。

1) 首先将 USB 转 TTL 模块插入 Ubuntu 电脑的 USB 接口，如果 USB 转 TTL 模块连接识别正常，在 Ubuntu PC 的 **/dev** 下就可以看到对应的设备节点名，记住这个节点名，后面设置串口软件时会用到

```
test@test:~$ ls /dev/ttys  
/dev/ttys
```

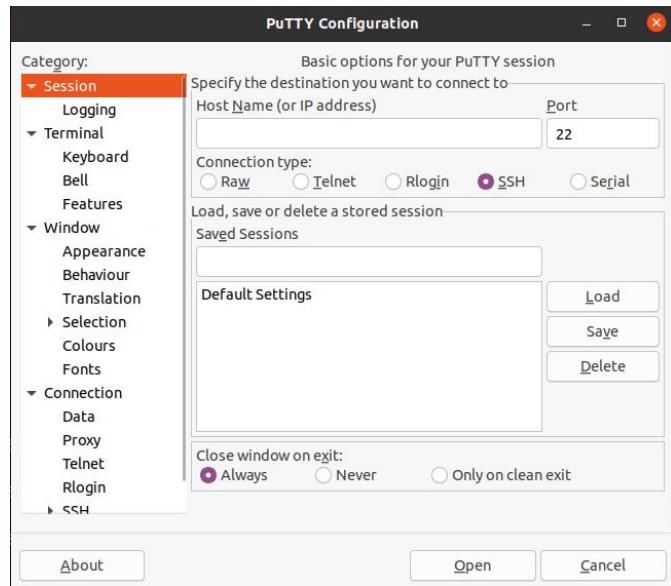
2) 然后使用下面的命令在 Ubuntu PC 上安装下 **putty**

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

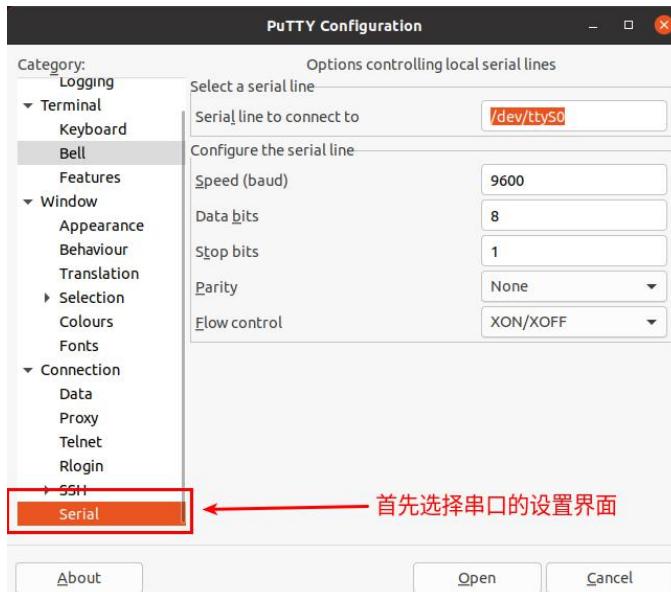
3) 然后运行 **putty**，记得加 **sudo** 权限

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

4) 执行 **putty** 命令后会弹出下面的界面

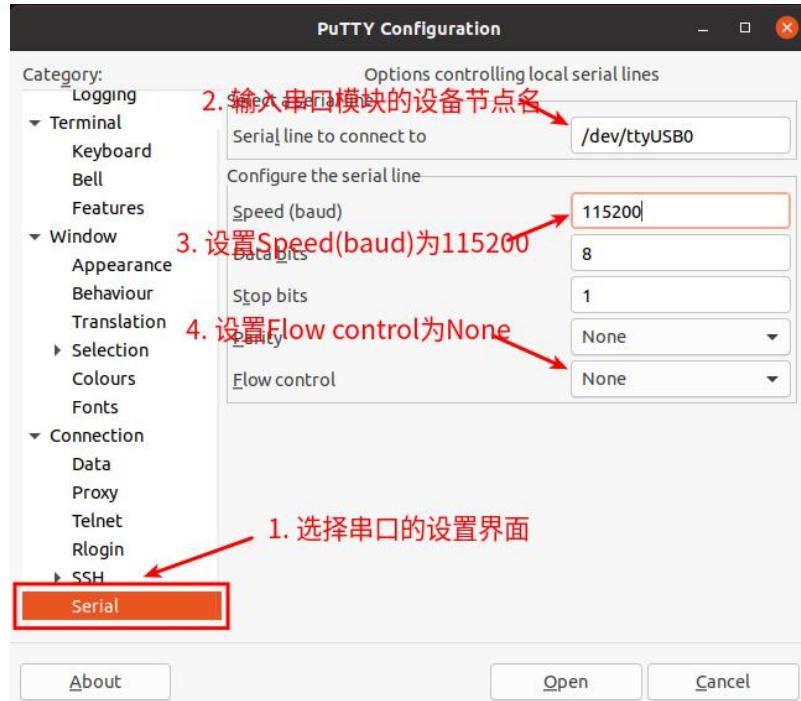


5) 首先选择串口的设置界面



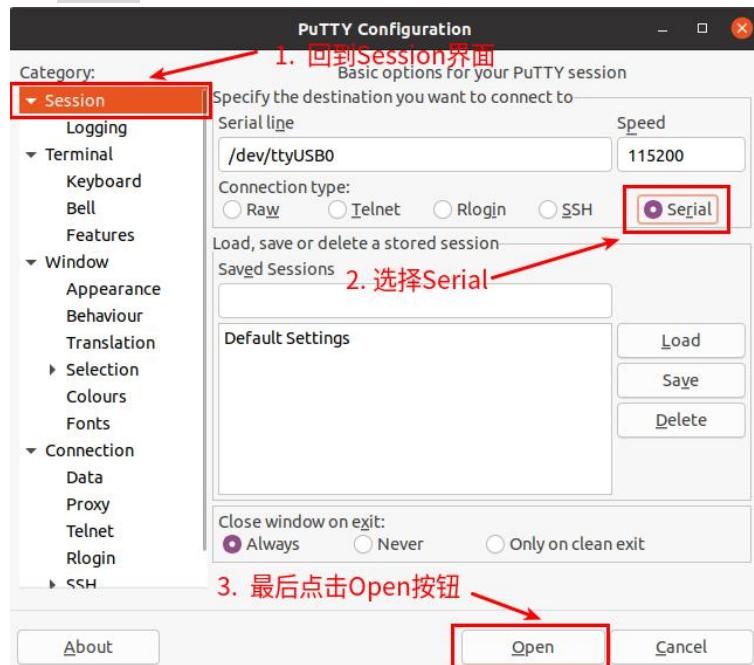
6) 然后设置串口的参数

- a. 设置 **Serial line to connect to** 为 **/dev/ttys0** (修改为对应的节点名, 一般为**/dev/ttys0**)
- b. 设置 **Speed(baud)** 为 **115200** (串口的波特率)
- c. 设置 **Flow control** 为 **None**



7) 在串口的设置界面设置完后，再回到 Session 界面

- 首先选择 Connection type 为 Serial
- 然后点击 Open 按钮连接串口



8) 然后启动开发板，就能从打开的串口终端中看到系统输出的 Log 信息了



```
1189[EL10] R6070 is starting May 13 2020 14:10:04|  
1183[boot0] commit : 9336e38  
1186[set pll start]  
1188[periph0] has been enabled  
1172[set pll end]  
1174[PLL] 100M  
1175[PMM] AP9006  
1182[valid para1] select dram para0  
1186[board init ok]  
1186[mem blc init] INFO: W0,52  
1181[chip id] 0x00000000  
1194[chip id check] OK  
1196[DRAM_VCC set to 1500 mV]  
200[read_calibration error]  
204[read_calibration error]  
208[read_calibration error]  
212[read_calibration error]  
216[read_calibration error]  
220[read_calibration error]  
224[read_calibration error]  
228[read_calibration error]  
232[read_calibration error]  
236[read_calibration error]  
240[read_calibration error]  
244[read_calibration error]  
248[read_calibration error]  
252[read_calibration error]  
256[read_calibration error]  
260[read_calibration error]  
264[read_calibration error]  
268[read_calibration error]  
272[read_calibration error]  
276[read_calibration error]  
280[read_calibration error]  
284[read_calibration error]  
288[read_calibration error]  
292[read_calibration error]  
296[read_calibration error]  
300[read_calibration error]  
304[read_calibration error]  
308[read_calibration error]  
312[read_calibration error]  
316[read_calibration error]  
320[read_calibration error]  
324[read_calibration error]  
328[read_calibration error]  
332[read_calibration error]  
336[read_calibration error]  
340[read_calibration error]  
344[read_calibration error]  
348[read_calibration error]  
352[read_calibration error]  
356[read_calibration error]  
360[read_calibration error]  
364[read_calibration error]  
368[read_calibration error]  
372[read_calibration error]  
376[read_calibration error]  
380[read_calibration error]  
384[read_calibration error]  
388[read_calibration error]  
392[read_calibration error]  
396[read_calibration error]  
400[read_calibration error]  
404[read_calibration error]  
408[read_calibration error]  
412[read_calibration error]  
416[read_calibration error]  
420[read_calibration error]  
424[read_calibration error]  
428[read_calibration error]  
432[read_calibration error]  
436[read_calibration error]  
440[read_calibration error]  
444[read_calibration error]  
448[read_calibration error]  
452[read_calibration error]  
456[read_calibration error]  
460[read_calibration error]  
464[read_calibration error]  
468[read_calibration error]  
472[read_calibration error]  
476[read_calibration error]  
480[read_calibration error]  
484[read_calibration error]  
488[read_calibration error]  
492[read_calibration error]  
496[read_calibration error]  
500[read_calibration error]  
504[read_calibration error]  
508[read_calibration error]  
512[read_calibration error]  
516[read_calibration error]  
520[read_calibration error]  
524[read_calibration error]  
528[read_calibration error]  
532[read_calibration error]  
536[read_calibration error]  
540[read_calibration error]  
544[read_calibration error]  
548[read_calibration error]  
552[read_calibration error]  
556[read_calibration error]  
560[read_calibration error]  
564[read_calibration error]  
568[read_calibration error]  
572[read_calibration error]  
576[read_calibration error]  
580[read_calibration error]  
584[read_calibration error]  
588[read_calibration error]  
592[read_calibration error]  
596[read_calibration error]  
600[read_calibration error]  
604[read_calibration error]  
608[read_calibration error]  
612[read_calibration error]  
616[read_calibration error]  
620[read_calibration error]  
624[read_calibration error]  
628[read_calibration error]  
632[read_calibration error]  
636[read_calibration error]  
640[read_calibration error]  
644[read_calibration error]  
648[read_calibration error]  
652[read_calibration error]  
656[read_calibration error]  
660[read_calibration error]  
664[read_calibration error]  
668[read_calibration error]  
672[read_calibration error]  
676[read_calibration error]  
680[read_calibration error]  
684[read_calibration error]  
688[read_calibration error]  
692[read_calibration error]  
696[read_calibration error]  
700[read_calibration error]  
704[read_calibration error]  
708[read_calibration error]  
712[read_calibration error]  
716[read_calibration error]  
720[read_calibration error]  
724[read_calibration error]  
728[read_calibration error]  
732[read_calibration error]  
736[read_calibration error]  
740[read_calibration error]  
744[read_calibration error]  
748[read_calibration error]  
752[read_calibration error]  
756[read_calibration error]  
760[read_calibration error]  
764[read_calibration error]  
768[read_calibration error]  
772[read_calibration error]  
776[read_calibration error]  
780[read_calibration error]  
784[read_calibration error]  
788[read_calibration error]  
792[read_calibration error]  
796[read_calibration error]  
800[read_calibration error]  
804[read_calibration error]  
808[read_calibration error]  
812[read_calibration error]  
816[read_calibration error]  
820[read_calibration error]  
824[read_calibration error]  
828[read_calibration error]  
832[read_calibration error]  
836[read_calibration error]  
840[read_calibration error]  
844[read_calibration error]  
848[read_calibration error]  
852[read_calibration error]  
856[read_calibration error]  
860[read_calibration error]  
864[read_calibration error]  
868[read_calibration error]  
872[read_calibration error]  
876[read_calibration error]  
880[read_calibration error]  
884[read_calibration error]  
888[read_calibration error]  
892[read_calibration error]  
896[read_calibration error]  
900[read_calibration error]  
904[read_calibration error]  
908[read_calibration error]  
912[read_calibration error]  
916[read_calibration error]  
920[read_calibration error]  
924[read_calibration error]  
928[read_calibration error]  
932[read_calibration error]  
936[read_calibration error]  
940[read_calibration error]  
944[read_calibration error]  
948[read_calibration error]  
952[read_calibration error]  
956[read_calibration error]  
960[read_calibration error]  
964[read_calibration error]  
968[read_calibration error]  
972[read_calibration error]  
976[read_calibration error]  
980[read_calibration error]  
984[read_calibration error]  
988[read_calibration error]  
992[read_calibration error]  
996[read_calibration error]  
1000[read_calibration error]
```

2. 8. 3. Windows 平台调试串口的使用方法

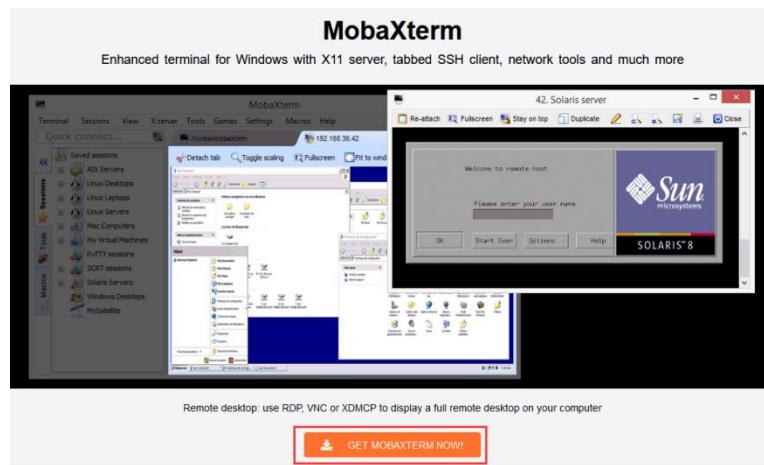
Windows 下可以使用的串口调试软件有很多，如 SecureCRT、MobaXterm 等，下面演示 MobaXterm 的使用方法，这款软件有免费版本，无需购买序列号即可使用。

1) 下载 MobaXterm

- 下载 MobaXterm 网址如下

<https://mobaxterm.mobatek.net/>

- 进入 MobaXterm 下载网页后点击 **GET XOBATERM NOW!**



- 然后选择下载 Home 版本



The chart compares the features of MobaXterm Home Edition (Free) and Professional Edition.

Home Edition	Professional Edition
Free	\$69 / 49€ per user*
Full X server and SSH support	* Excluding tax. Volume discounts available
Remote desktop (RDP, VNC, Xdmcp)	
Remote terminal (SSH, telnet, rlogin, Mosh)	
X11-Forwarding	Every feature from Home Edition +
Automatic SFTP browser	Customize your startup message and logo
Master password protection	Modify your profile script
Plugins support	Remove unwanted games, screensaver or tools
Portable and installer versions	Unlimited number of sessions
Full documentation	Unlimited number of tunnels and macros
Max. 12 sessions	Unlimited run time for network daemons
Max. 2 SSH tunnels	Enhanced security settings
Max. 4 macros	12-months updates included
Max. 360 seconds for Tftp, Nfs and Cron	Deployment inside company
Download now	Lifetime right to use

d. 然后选择 Portable 便携式版本，下载完后无需安装，直接打开就可以使用

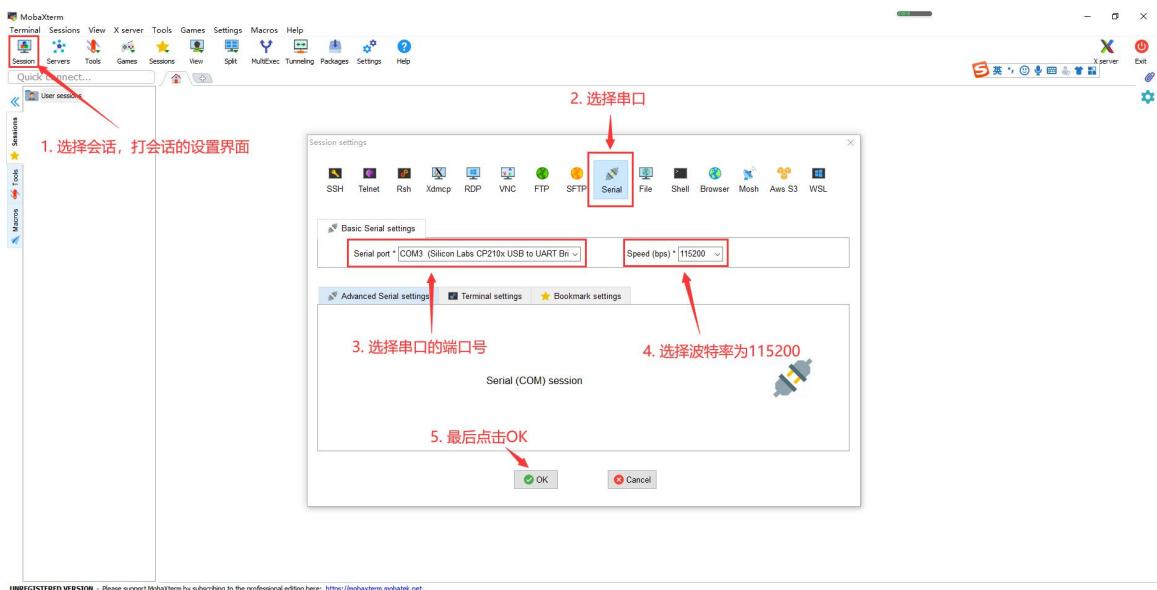
The screenshot shows the MobaXterm Home Edition download page. It highlights the "MobaXterm Home Edition v20.3 (Portable edition)" link, which is enclosed in a red box. Other links like "MobaXterm Home Edition v20.3 (Installer edition)" and "MobaXterm Preview Version" are also visible.

2) 下载完后使用解压缩软件解压下载的压缩包，即可得到 MobaXterm 的可执软件，然后双击打开

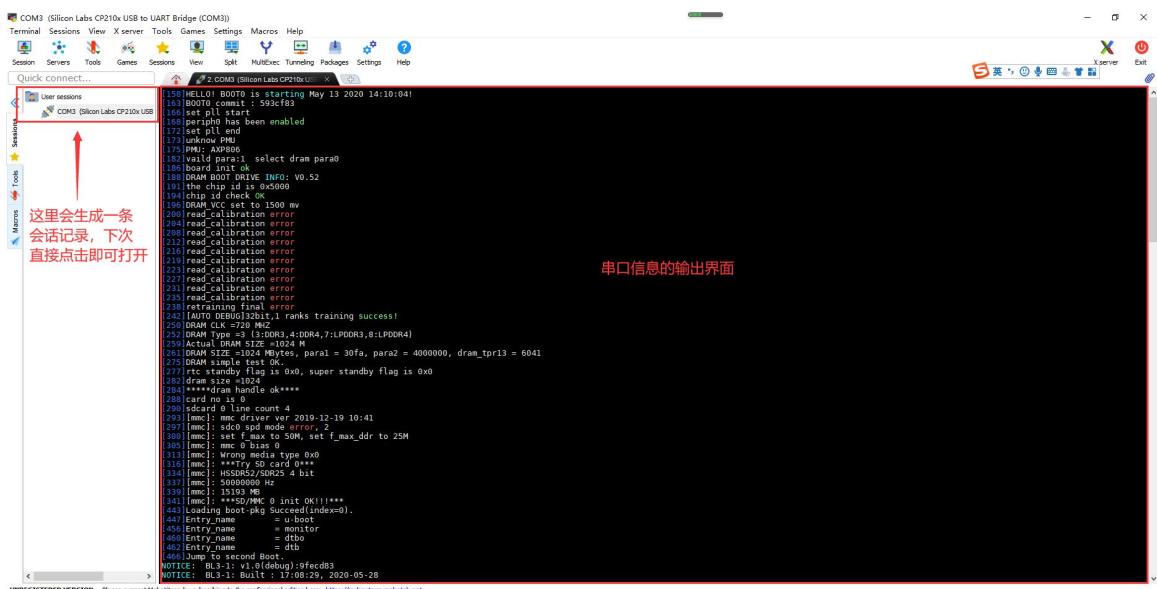
名称	修改日期	类型	大小
CygUtils.plugin	2020/5/21 4:06	PLUGIN 文件	15,570 KB
MobaXterm_Personal_20.3	2020/6/5 4:30	应用程序	14,104 KB

3) 打开软件后，设置串口连接的步骤如下

- 打开会话的设置界面
- 选择串口类型
- 选择串口的端口号（根据实际的情况选择对应的端口号），如果看不到端口号，请使用 **360 驱动大师** 扫描安装 USB 转 TTL 串口芯片的驱动
- 选择串口的波特率为 **115200**
- 最后点击“OK”按钮完成设置



4) 点击“OK”按钮后会进入下面的界面，此时启动开发板就能看到串口的输出信息了





2. 9. 使用开发板 40pin 接口中的 5v 引脚供电说明

我们推荐的开发板的供电方式是使用 5V/2A 或者 5V/3A 的 Type C 接口的电源线插到开发板的 Type C 电源接口来供电的。如果需要使用 40pin 接口中的 5V 引脚来给开发板供电，请确保使用的电源线能满足开发板的供电需求。如果有使用不稳定的情况，请换回 Type C 电源供电。

注意：40pin接口上的排针默认是不焊接的，需要自己焊接上去才能使用。

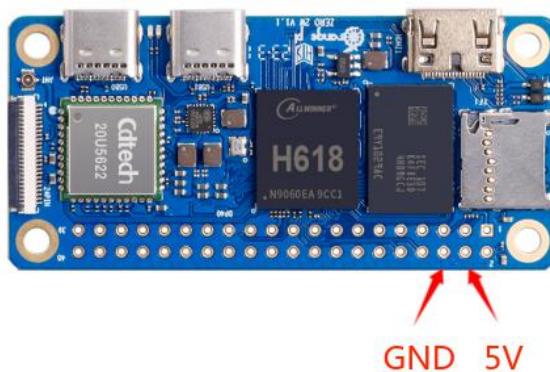
1) 首先需要准备一根下图所示的电源线



上图所示的电源线在淘宝可以买到，请自行搜索购买。

2) 使用 40pin 接口中的 5V 引脚来给开发板供电，电源线的接法如下所示：

- a. 上图所示的电源线 USB-A 口需要插到 5V/2A 或者 5V/3A 的电源适配器接头上
- b. 红色的杜邦线需要插到开发板 40pin 接口的 5V 引脚上
- c. 黑色的杜邦线需要插到 40pin 接口的 GND 引脚上
- d. 40pin 接口 5V 引脚和 GND 引脚在开发板中的位置如下图所示，**切记不要接反了**





3. Debian/Ubuntu Server 和 Xfce 桌面系统使用说明

3.1. 已支持的 linux 镜像类型和内核版本

Linux 镜像类型	内核版本	服务器版	桌面版
Ubuntu 20.04 - Focal	Linux5.4	支持	支持
Debian 11 - Bullseye	Linux5.4	支持	支持
Ubuntu 22.04 - Jammy	Linux6.1	支持	支持
Debian 11 - Bullseye	Linux6.1	支持	支持
Debian 12 - Bookworm	Linux6.1	支持	支持

在 [Orange Pi 的资料下载页面](#)进入对应开发板的下载页面后可以看到下面的下载选项，在下文的描述中，Ubuntu 镜像和 Debian 镜像一般统称为 Linux 镜像。



ubuntu镜像



debian镜像

[下载](#)[下载](#)

Linux 镜像的命名规则为：

开发板型号_版本号_Linux 发行版类型_发行版代号_服务器或桌面_内核版本

- a. **开发板的型号：**都是 **orangepirzero2w**。不同开发板的型号名一般都是不同的，烧录镜像前，请确保所选择镜像的这个型号名和开发板是匹配的。
- b. **版本号：**如 **1.x.x**，这个版本号会随着镜像功能的更新而递增，另外开发板 Linux 镜像的版本号最后一个数字都是偶数。
- c. **Linux 发行版的类型：**目前支持 **Ubuntu** 和 **Debian**。由于 Ubuntu 源自 Debian，所以两个系统在使用上来说总体区别不大。但部分软件的默认配置和命令的使用上还是有些许区别的，另外 Ubuntu 和 Debian 都各自有维护所支持的软件仓库，在支持的可安装的软件包上也是有些许差异的。这些需要亲自去使用体验才会有比较深刻的认识。有关更多的细节，可以参考下 Ubuntu 和 Debian 官方提供的文档。
- d. **发行版代号：**用来区分 Ubuntu 或者 Debian 这样具体的 Linux 发行版的不同版本。其中 **focal** 和 **jammy** 都是 Ubuntu 发行版，focal 表示 Ubuntu20.04，jammy 表示 Ubuntu22.04，不同版本的最大的区别是新版本的 Ubuntu 系统



维护的软件仓库中的软件很多都比旧版本的 Ubuntu 系统中的要新，比如 Python 和 GCC 编译工具链等。**bullseye** 是 Debian 的具体版本代号，**bullseye** 表示 Debian11，**bookworm** 表示 Debian12。

- e. **服务器或桌面：**用来表示系统是否带桌面环境，如果为 **server** 就表示系统没有安装桌面环境，镜像占用的存储空间和资源比较小，主要使用命令行来操作控制系统。如果为 **desktop_xfce** 就表示系统默认安装有 XFCE 桌面环境，镜像占用的存储空间和资源比较大，可以接显示器和鼠标键盘通过界面来操作系统。当然 desktop 版本的系统也可以像 server 版本的系统一样通过命令行来操作。
- f. **内核版本：**用来表示 linux 内核的版本号，目前支持 **linux5.4** 和 **linux6.1**。

3. 2. linux 内核驱动适配情况

主板功能	Linux5.4	Linux6.1
HDMI 视频	OK	OK
HDMI 音频	OK	OK
Type-C USB2.0 x 2	OK	OK
TF 卡启动	OK	OK
WIFI	OK	OK
蓝牙	OK	OK
USB 摄像头	OK	OK
LED 灯	OK	OK
40pin GPIO	OK	OK
40pin I2C	OK	OK
40pin SPI	OK	OK
40pin UART	OK	OK
40pin PWM	OK	OK
温度传感器	OK	OK
硬件看门狗	OK	OK
Mali GPU	NO	NO
视频编解码	NO	NO



24pin 扩展板功能	Linux5.4	Linux6.1
百兆网口	OK	OK
百兆网口灯	OK	OK
USB2.0 HOST x 2	OK	OK
红外接收	OK	OK
耳机音频播放	OK	OK
开关机按键	OK	OK
LRADC 自定义按键 x 2	OK	OK
TV-OUT	NO	NO

3. 3. 本手册 linux 命令格式说明

1) 本手册中所有需要在 Linux 系统中输入的命令都会使用下面的方框框起来

如下所示，黄色方框里内容表示需要特别注意的内容，这里面的命令除外。

2) 命令前面的提示符类型说明

- 命令前面提示符指的是下面方框内红色部分的内容，这部分内容不是 linux 命令的一部分，所以在 linux 系统中输入命令时，请不要把红色字体部分的内容也输入进去。

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

- root@orangepi:~\$** 提示符表示这个命令是在开发板的 linux 系统中输入的，提示符最后的 **\$** 表示系统当前用户为普通用户，当执行特权命令时，需要加上 **sudo**
- root@orangepi:~#** 提示符表示这个命令是在开发板的 linux 系统中输入的，提示符最后的 **#** 表示系统当前用户为 root 用户，可以执行任何想要执行的命令
- test@test:~\$** 提示符表示这个命令是在 Ubuntu PC 或者 Ubuntu 虚拟机中输入的，而不是开发板的 linux 系统中。提示符最后的 **\$** 表示系统当前用户



为普通用户，当执行特权命令时，需要加上 **sudo**

- e. **root@test:~#** 提示符表示这个命令是在 Ubuntu PC 或者 Ubuntu 虚拟机中输入的，而不是开发板的 linux 系统中。提示符最后的 **#** 表示系统当前用户为 root 用户，可以执行任何想要执行的命令

3) 哪些是需要输入的命令？

- a. 如下所示，**黑色加粗部分**是需要输入的命令，命令下面的是输出的内容（有些命令有输出，有些可能没有输出），这部分内容是不需要输入的

```
root@orangepi:~# cat /boot/orangepiEnv.txt
verbosity=7
bootlogo=false
console=serial
```

- b. 如下所示，有些命令一行写不下会放到下一行，只要黑色加粗的部分就都是需要输入的命令。当这些命令输入到一行的时候，每行最后的“\”是需要去掉的，这个不是命令的一部分。另外命令的不同部分都是有空格的，请别漏了

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

3. 4. linux 系统登录说明

3. 4. 1. linux 系统默认登录账号和密码

账号	密码
root	orangepi
orangepi	orangepi

注意，输入密码的时候，**屏幕上是不会显示输入的密码的具体内容的**，请不要以为是有什么故障，输入完后直接回车即可。

当输入密码提示错误，或者 ssh 连接有问题，请注意，只要使用的是 Orange Pi



提供的 Linux 镜像，就请不要怀疑上面的密码不对，而是要找其它的原因。

3.4.2. 设置 linux 系统终端自动登录的方法

1) linux 系统默认就是自动登录终端的， 默认登录的用户名是 orangepi

2) 使用下面的命令可以设置 root 用户自动登录终端

```
orangeipi@orangeipi:~$ sudo auto login cli.sh root
```

3) 使用下面的命令可以禁止自动登录终端

```
orangepi@orangepi:~$ sudo auto login cli.sh -d
```

4) 使用下面的命令可以再次设置 orangepi 用户自动登录终端

```
orangepi@orangepi:~$ sudo auto login cli.sh orangepi
```

3. 4. 3. linux 桌面板系统自动登录说明

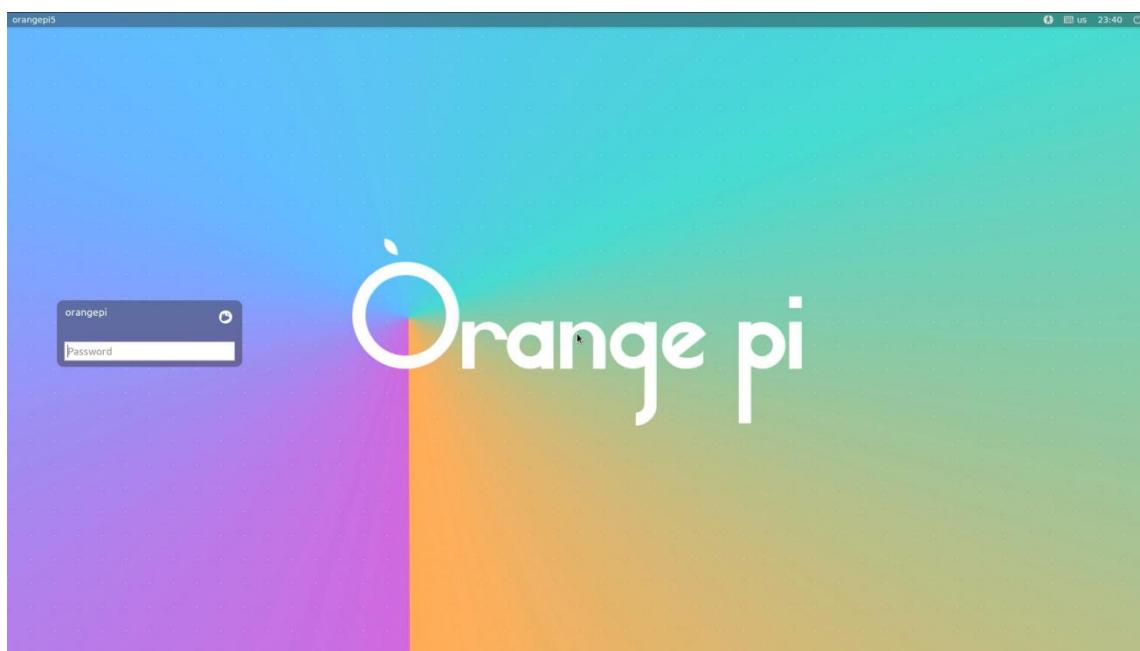
1) 桌面版系统启动后会自动登录进入桌面，无需输入密码



2) 运行下面的命令可以禁止桌面版系统自动登录桌面

```
orangepi@orangepi:~$ sudo disable_desktop_autologin.sh
```

3) 然后重启系统就会出现登录对话框，此时需要输入**密码**才能进入系统



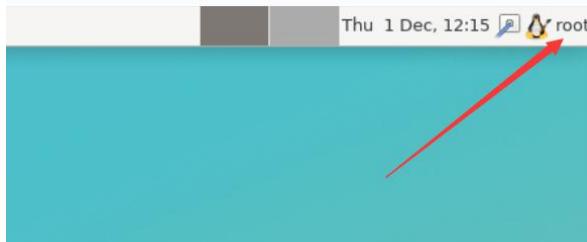
3. 4. 4. Linux 桌面版系统 root 用户自动登录的设置方法

1) 执行下面的命令可以设置桌面版系统使用 root 用户自动登录



```
orangeipi@orangeipi:~$ sudo desktop_login.sh root
```

2) 然后重启系统，就会自动使用 root 用户登录桌面了



注意，如果使用 root 用户登录桌面系统，是无法使用右上角的 pulseaudio 来管理音频设备的。

另外请注意这并不是一个 bug，因为 pulseaudio 本来就不允许在 root 用户下运行。

3) 执行下面的命令可以再次设置桌面版系统使用 orangeipi 用户自动登录

```
orangeipi@orangeipi:~$ sudo desktop_login.sh orangeipi
```

3. 4. 5. Linux 桌面版系统禁用桌面的方法

1) 首先在命令行中输入下面的命令，**请记得加 sudo 权限**

```
orangeipi@orangeipi:~$ sudo systemctl disable lightdm.service
```

2) 然后重启 Linux 系统就会发现不会显示桌面了

```
orangeipi@orangeipi:~$ sudo reboot
```

3) 重新打开桌面的命令如下所示，**请记得加 sudo 权限**

```
orangeipi@orangeipi:~$ sudo systemctl start lightdm.service
```

```
orangeipi@orangeipi:~$ sudo systemctl enable lightdm.service
```

3. 5. 板载 LED 灯测试说明

1) 开发板上有两个 LED 灯，一个绿灯，一个红灯，系统启动时 LED 灯默认显示情况如下所示：

	绿灯	红灯
--	----	----



u-boot 启动阶段	灭	亮
内核启动到进入系统	闪烁	亮

开发板上的绿灯可以通过软件来控制，红灯上电后就会常亮，无法通过软件来控制。

当拿到开发板后，您可能会发现开发板上就算没有插入烧录有系统的 TF 卡，给开发板接上电源后，绿灯也会闪烁，这是因为开发板上的 16MB SPI Flash 出厂默认会烧录一个微型的 Linux 系统，此系统在进入内核后会设置绿灯闪烁。

如果 SPI Flash 中的 Linux 系统被清空了，那么不插入烧录有系统的 TF 卡，接通电源后，开发板上就只会看到红灯常亮了。

2) 设置绿灯亮灭和闪烁的方法如下所示：

注意，下面的操作请在 root 用户下进行。

- 首先进入绿灯的设置目录

```
root@orangepi:~# cd /sys/class/leds/green_led
```

- 设置绿灯停止闪烁的命令如下

```
root@orangepi:/sys/class/leds/green_led# echo none > trigger
```

- 设置绿灯常亮的命令如下

```
root@orangepi:/sys/class/leds/green_led# echo default-on > trigger
```

- 设置绿灯闪烁的命令如下

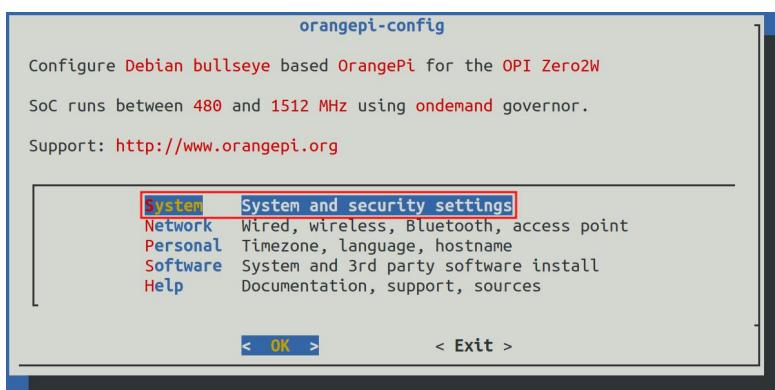
```
root@orangepi:/sys/class/leds/green_led# echo heartbeat > trigger
```

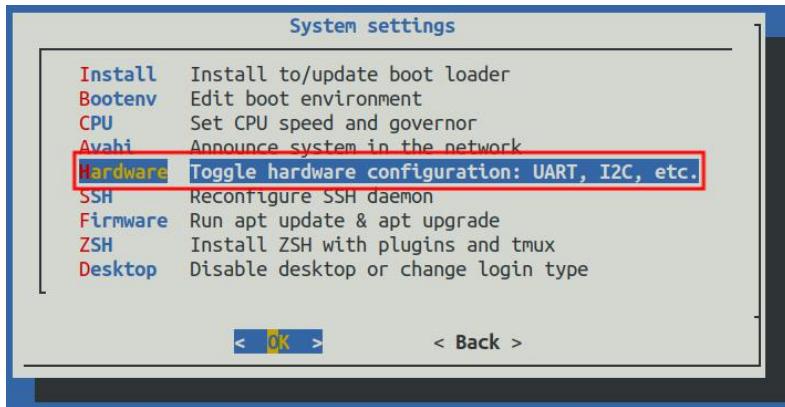
3) 如果开机后不需要 LED 灯闪烁，可以使用下面的方法来关闭绿灯闪烁

- 首先运行下 orangepi-config，普通用户记得加 sudo 权限

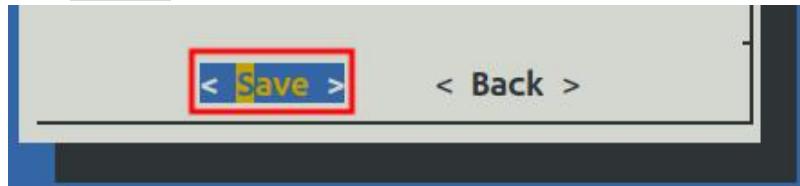
```
orangepi@orangepi:~$ sudo orangepi-config
```

- 然后选择 System

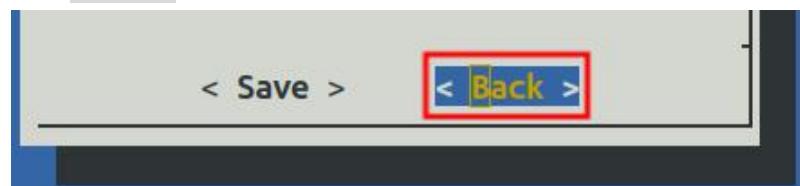


c. 然后选择 **Hardware**d. 然后使用键盘的方向键定位到下图所示的位置，再使用空格选中 **disable-leds**

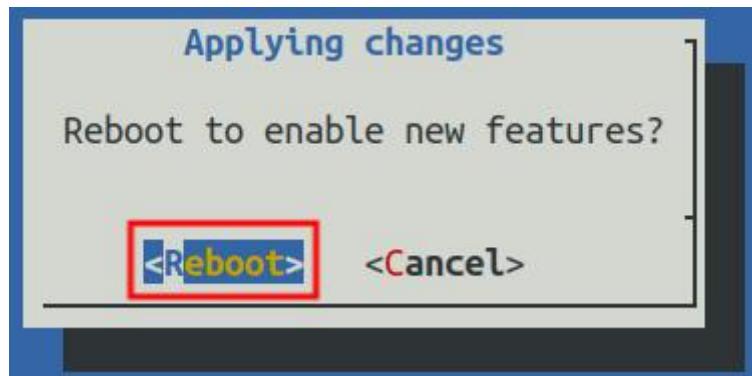
e. 然后选择 <Save> 保存



f. 然后选择 <Back>



g. 然后选择 <Reboot> 重启系统使配置生效



h. 重启后进入系统就可以看到开发板上的绿色 LED 灯都不会亮了



3.6. TF 卡中 linux 系统 rootfs 分区容量操作说明

3.6.1. 第一次启动会自动扩容 TF 卡中 rootfs 分区的容量

1) 将开发板的 Linux 镜像烧录到 TF 卡中后, 可以在 **Ubuntu 电脑** 中查看下 TF 卡容量的使用情况, 步骤如下所示:

注意, 这一步不操作是不影响开发板的 Linux 系统自动扩容的。这里只是想说明 TF 卡烧录完 Linux 镜像后, 怎么查看 TF 卡容量的方法。

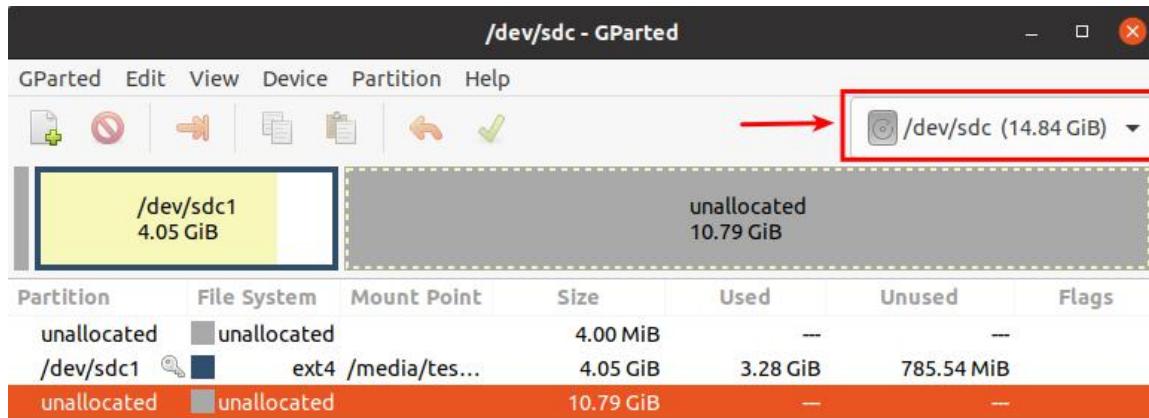
- 首先在 Ubuntu 电脑中安装下 gparted 这个软件

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

- 然后打开 gparted

```
test@test:~$ sudo gparted
```

- 打开 gparted 后在右上角可以选择 TF 卡, 然后就可以看到 TF 卡容量的使用情况



- 上图显示的是烧录完 Linux 桌面版系统后 TF 卡的情况, 可以看到, 虽然 TF 卡的总容量是 16GB 的 (在 GParted 中显示为 14.84GiB), 但是 rootfs 分区 (/dev/sdc1) 实际只分配了 4.05GiB, 还剩下 10.79GiB 未分配

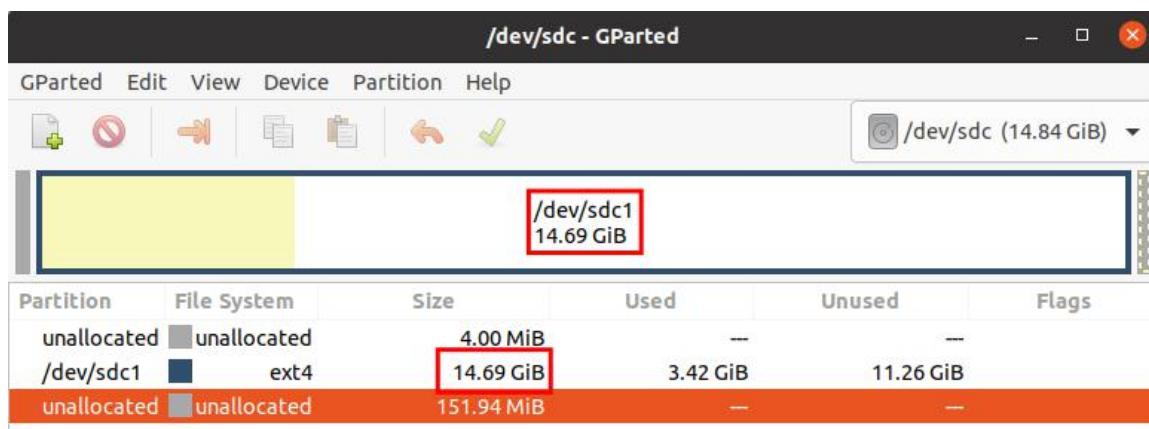
2) 然后可以将烧录好 Linux 系统的 TF 卡插入开发板中启动, TF 卡第一次启动 linux 系统时会通过 **orangeipi-resize-filesystem.service** 这个 systemd 服务来调用 **orangeipi-resize-filesystem** 脚本自动进行 rootfs 分区的扩容, 所以**无需再手动扩容**

3) 登录系统后可以通过 **df -h** 命令来查看 rootfs 的大小, 如果和 TF 卡的实际容量一致, 说明自动扩容运行正确



```
orangeipi@orangeipi:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            430M    0   430M   0% /dev
tmpfs           100M   5.6M   95M   6% /run
/dev/mmcblk0p1   15G  915M   14G   7% /
tmpfs           500M    0   500M   0% /dev/shm
```

- 4) 第一次启动完 Linux 系统后，我们还可以将 TF 卡从开发板中取下来重新插入 **Ubuntu 电脑**，然后再次使用 gparted 查看下 TF 卡的情况，如下图所示，rootfs 分区（/dev/sdc1）的容量已经扩展到了 14.69GiB 了



需要注意的是，linux 系统只有一个 ext4 格式的分区，没有使用单独的 **BOOT** 分区来存放内核镜像等文件，所以也就不存在 **BOOT** 分区扩容的问题。

3. 6. 2. 禁止自动扩容 TF 卡中 rootfs 分区容量的方法

- 1) 首先在 **Ubuntu 电脑**（Windows 不行）中将开发板的 linux 镜像烧录到 TF 卡中，
然后重新拔插下 TF 卡
- 2) 然后 Ubuntu 电脑一般会自动挂载 TF 卡的分区，如果自动挂载正常，使用 ls 命令可以看到下面的输出

```
test@test:~$ ls /media/test/opi_root/
bin  boot  dev  etc  home  lib  lost+found  media  mnt  opt  proc  root  run
sbin  selinux  srv  sys  tmp  usr  var
```

- 3) 然后在 Ubuntu 电脑中将当前用户切换成 root 用户

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



```
[sudo] test 的密码:  
root@test:~#
```

4) 然后进入 TF 卡中的 linux 系统的 root 目录下新建一个名为.**no_rootfs_resize** 的文件

```
root@test:~# cd /media/test/opi_root/  
root@test:/media/test/opi_root/# cd root  
root@test:/media/test/opi_root/root# touch .no_rootfs_resize  
root@test:/media/test/opi_root/root# ls .no_rootfs*  
.no_rootfs_resize
```

5) 然后就可以卸载 TF 卡，再拔出 TF 卡插到开发板中启动，linux 系统启动时，当检测到/**root** 目录下有**.no_rootfs_resize** 这个文件就不会再自动扩容 rootfs 了

6) 禁止 rootfs 自动扩容后进入 Linux 系统可以看到 rootfs 分区的总容量只有 4GB(这里测试的是桌面版本的镜像)，远小于 TF 卡的实际容量，说明禁止 rootfs 自动扩容成功

```
orangepi@orangepi:~$ df -h  
Filesystem      Size  Used Avail Use% Mounted on  
udev            925M    0   925M   0% /dev  
tmpfs           199M   3.2M  196M   2% /run  
/dev/mmcblk0p1  4.0G  3.2G  686M  83% /
```

7) 如果需要重新扩容 TF 卡中 rootfs 分区的容量，只需要执行下面的命令，然后重新启动开发板的 Linux 系统即可

注意，请在 root 用户下执行下面的命令。

```
root@orangepi:~# rm /root/.no_rootfs_resize  
root@orangepi:~# systemctl enable orangepi-resize-filesystem.service  
root@orangepi:~# sudo reboot
```

重启后再次进入开发板的 Linux 系统就可以看到 rootfs 分区已经扩展为 TF 卡的实际容量了

```
root@orangepi:~# df -h  
Filesystem      Size  Used Avail Use% Mounted on  
udev            925M    0   925M   0% /dev
```



tmpfs	199M	3.2M	196M	2%	/run
/dev/mmcblk0p1	15G	3.2G	12G	23%	/

3. 6. 3. 手动扩容 TF 卡中 rootfs 分区容量的方法

如果 TF 卡的总容量很大，比如为 128GB，不想 Linux 系统 rootfs 分区使用 TF 卡所有的容量，只想分配一部分容量，比如 16GB，给 Linux 系统使用，然后 TF 卡的剩余容量就可以用作其他用途。那么可以使用此小节介绍的内容来手动扩容 TF 中 rootfs 分区的容量。

1) 首先在 **Ubuntu 电脑** (Windows 不行) 中将开发板的 linux 镜像烧录到 TF 卡中，
然后重新拔插下 TF 卡

2) 然后 Ubuntu 电脑一般会自动挂载 TF 卡的分区，如果自动挂载正常，使用 **ls** 命令可以看到下面的输出

```
test@test:~$ ls /media/test/opi_root/
bin  boot  dev  etc  home  lib  lost+found  media  mnt  opt  proc  root  run
sbin  selinux  srv  sys  tmp  usr  var
```

3) 然后在 Ubuntu 电脑中将当前用户切换成 root 用户

```
test@test:~$ sudo -i
[sudo] test 的密码:
root@test:~#
```

4) 然后进入 TF 卡中的 linux 系统的 root 目录下新建一个名为 **.no_rootfs_resize** 的文件

```
root@test:~# cd /media/test/opi_root/
root@test:/media/test/opi_root# cd root
root@test:/media/test/opi_root/root# touch .no_rootfs_resize
root@test:/media/test/opi_root/root# ls .no_rootfs*
.no_rootfs_resize
```

5) 然后在 Ubuntu 电脑中安装 gparted 这个软件

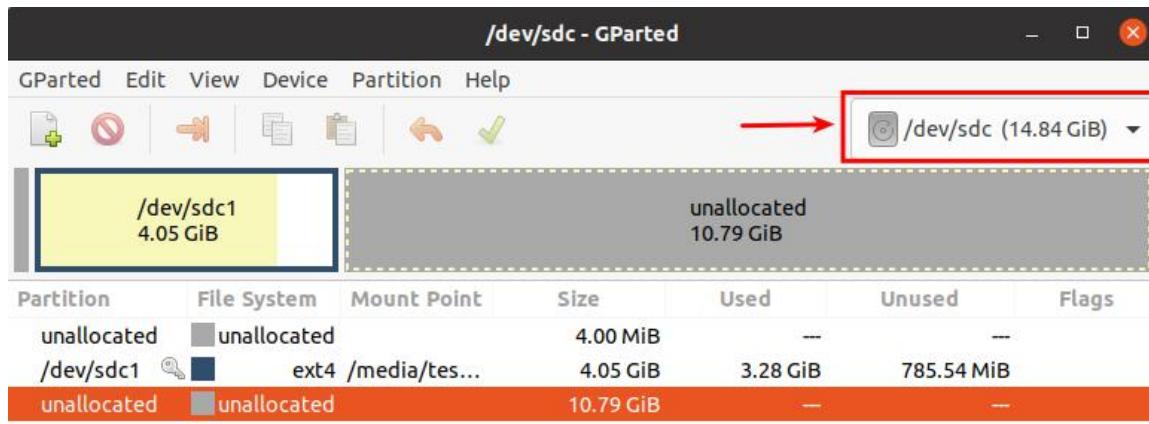
```
test@test:~$ sudo apt install -y gparted
```

6) 然后打开 gparted

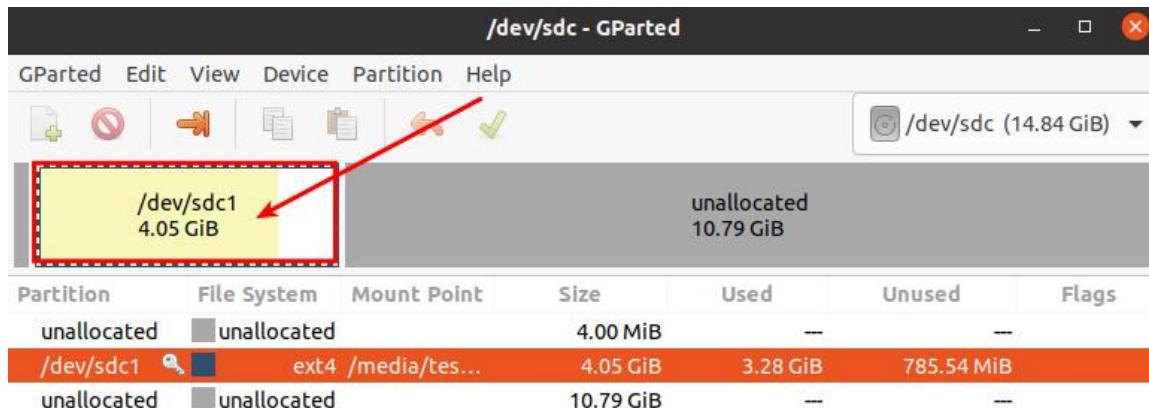


```
test@test:~$ sudo gparted
```

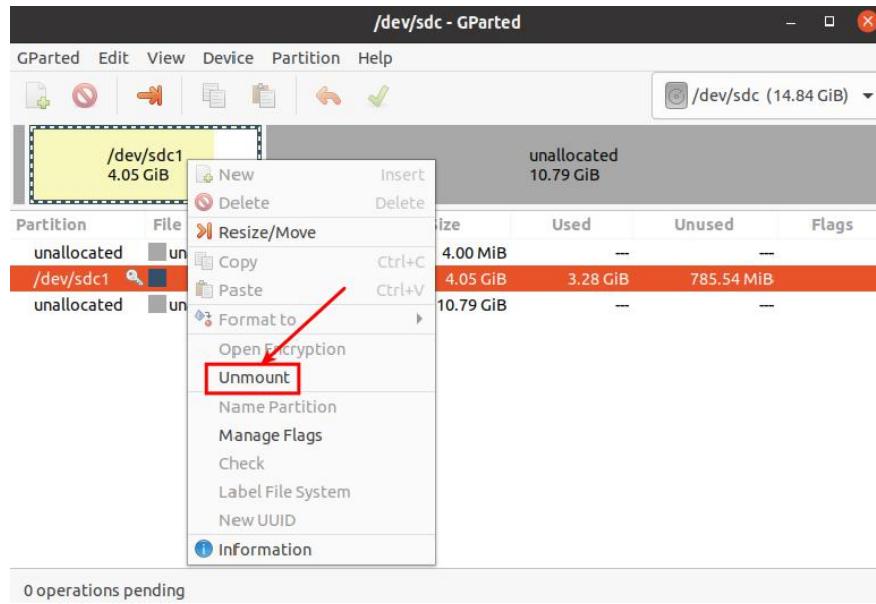
7) 打开 gparted 后在右上角可以选择 TF 卡，然后就可以看到 TF 卡容量的使用情况。下图显示的是烧录完 Linux 桌面版系统后 TF 卡的情况，可以看到，虽然 TF 卡的总容量是 16GB 的（在 GParted 中显示为 14.84GiB），但是 rootfs 分区（/dev/sdc1）实际只分配了 4.05GiB，还剩下 10.79GiB 未分配



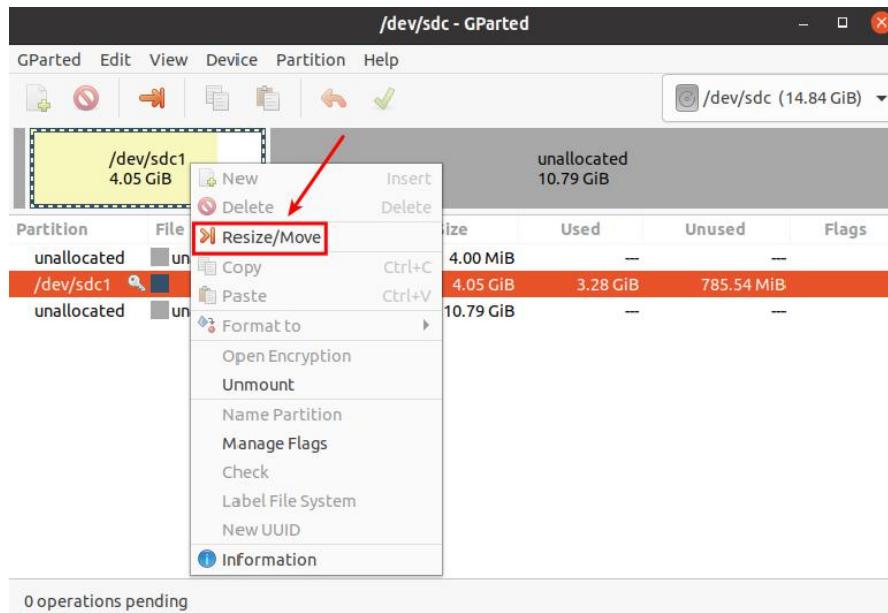
8) 然后选中 rootfs 分区（/dev/sdc1）



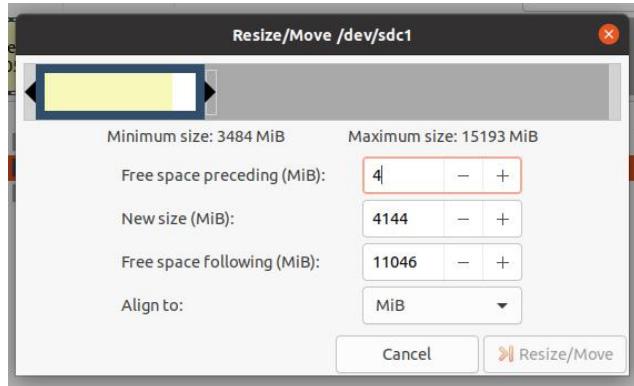
9) 再点击鼠标右键就可以看到下图所示的操作选项，如果 TF 卡已经挂载了，首先需要 Umount 掉 TF 卡的 rootfs 分区



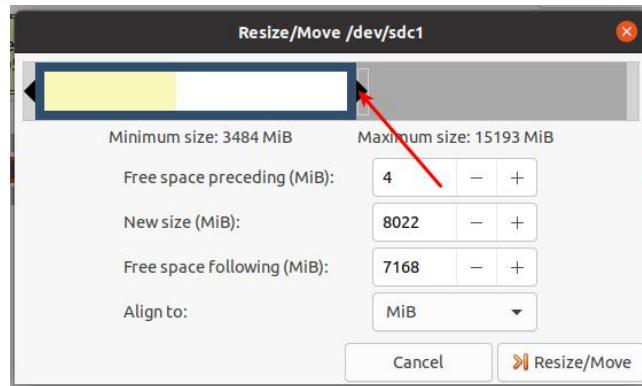
10) 然后再次选中 rootfs 分区，再点击鼠标右键，然后选择 **Resize/Move** 开始扩容 rootfs 分区的大小



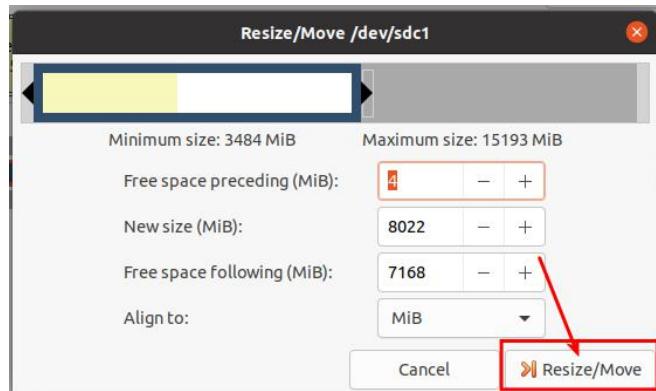
11) **Resize/Move** 选项打开后会弹出下面的设置界面



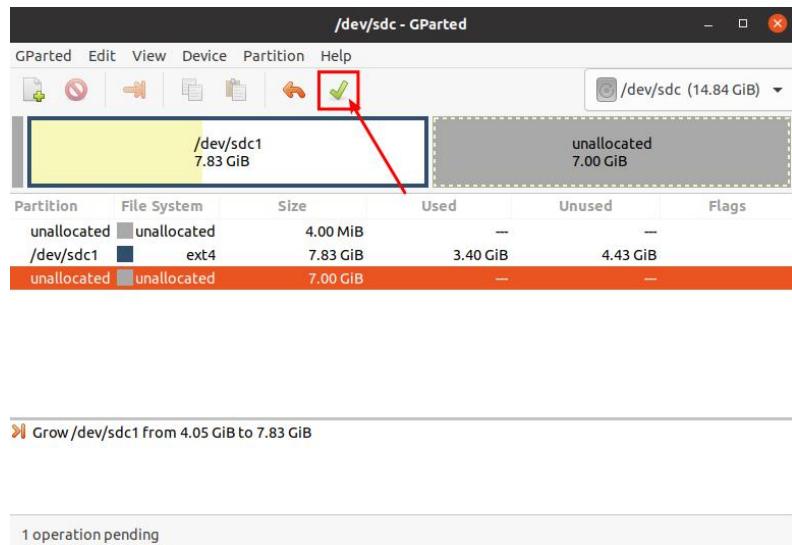
12) 然后可以直接拖动下图所示的位置来设置容量的大小，也可以通过设置 **New size(MiB)** 中的数字来设置 rootfs 分区的大小



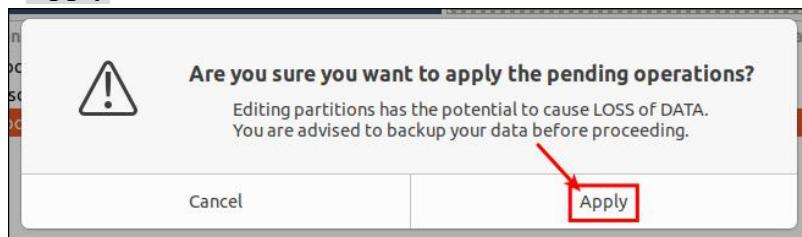
13) 设置好容量后，再点击右下角的 **Resize/Move** 即可



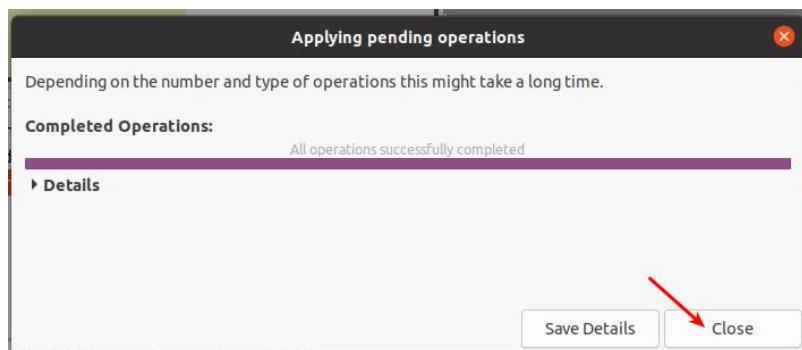
14) 最后确认无误后，再点击下图所示的绿色 ✓



15) 然后选择 **Apply**, 就会正式开始扩容 rootfs 分区的容量



16) 扩容完成后点击 **Close** 关闭即可



17) 然后就可以把 TF 卡拔下来, 再插到开发板中启动, 进入开发板的 Linux 系统中后如果使用 **df -h** 命令可以看到 rootfs 分区的大小和前面设置的大小一致的话就说明手动扩容成功

```
root@orangeipi:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            925M    0  925M   0% /dev
```



```
tmpfs           199M  3.2M  196M   2% /run  
/dev/mmcblk0p1  7.7G  3.2G  4.4G  42% /
```

3. 6. 4. 缩小 TF 卡中 rootfs 分区容量的方法

在 TF 卡的 Linux 系统中配置好应用程序或者其他开发环境后，如果想备份下 TF 卡中的 Linux 系统，可以使用此小节的方法先缩小下 rootfs 分区的大小，然后再开始备份。

1) 首先在 **Ubuntu 电脑** (Windows 不行) 中插入想要操作的 TF 卡

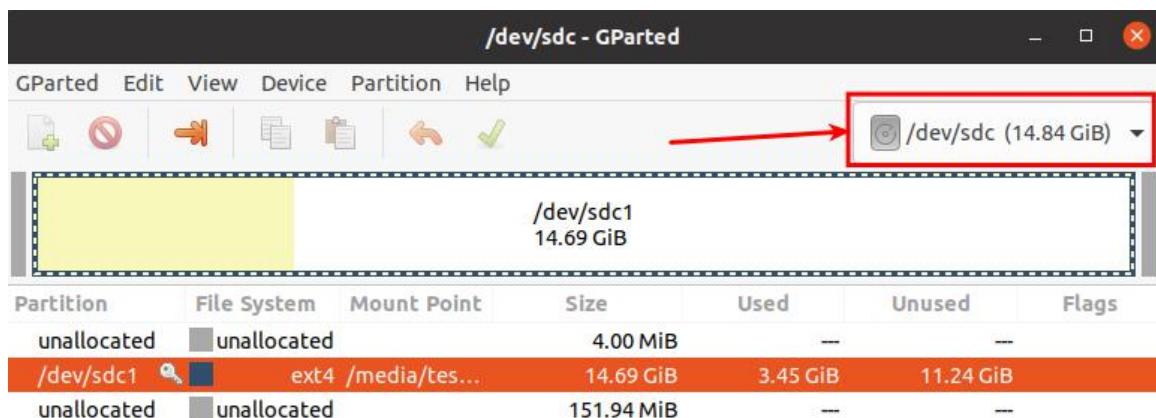
2) 然后在 Ubuntu 电脑中安装下 gparted 这个软件

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

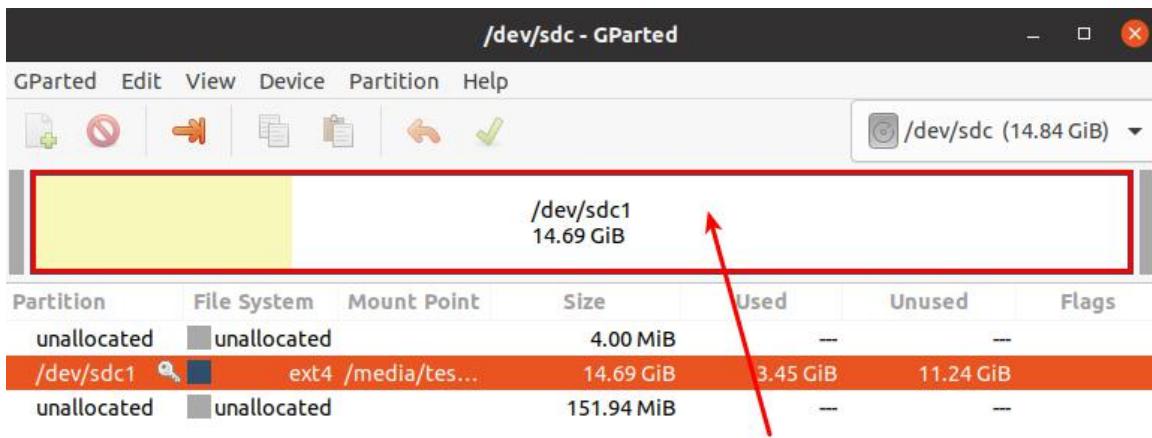
3) 然后打开 gparted

```
test@test:~$ sudo gparted
```

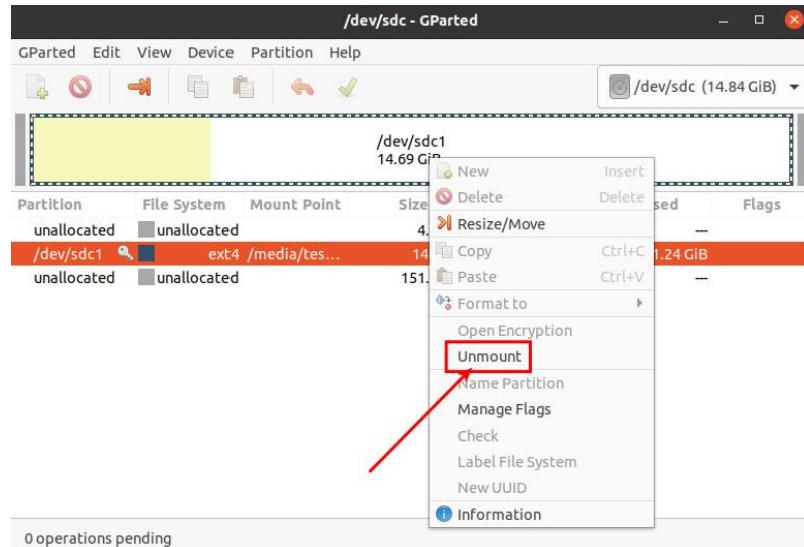
4) 打开 gparted 后在右上角可以选择 TF 卡，然后就可以看到 TF 卡容量的使用情况



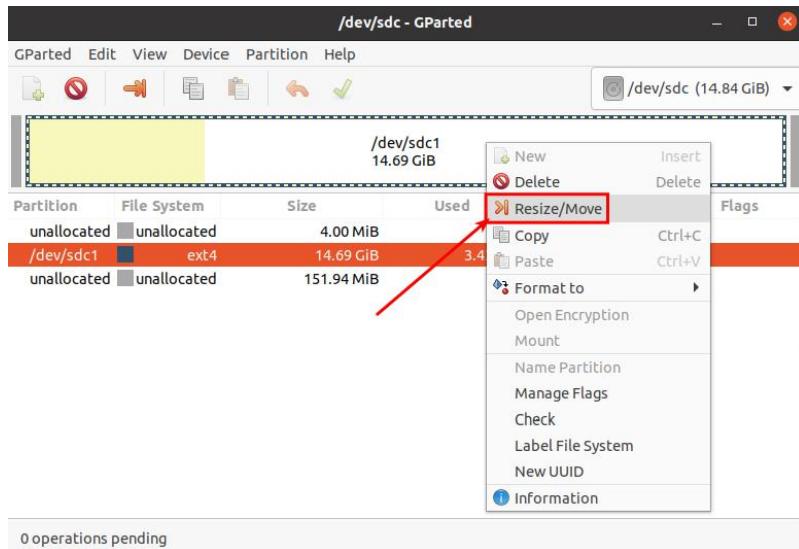
5) 然后选中 rootfs 分区 (/dev/sdc1)



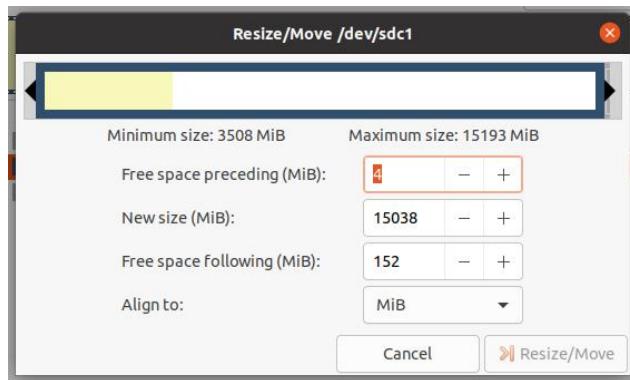
6) 再点击鼠标右键就可以看到下图所示的操作选项，如果 TF 卡已经挂载了，首先需要 Umount 掉 TF 卡的 rootfs 分区



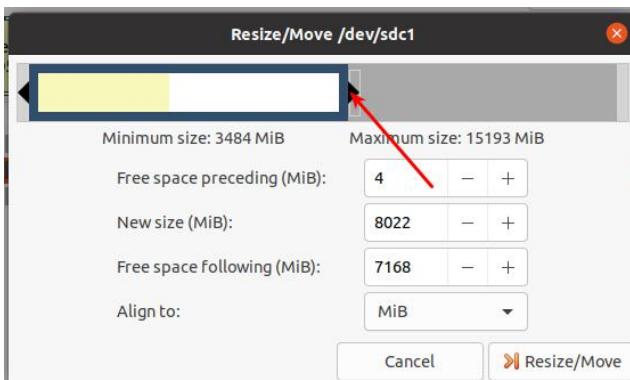
7) 然后再次选中 rootfs 分区，再点击鼠标右键，然后选择 **Resize/Move** 开始设置 rootfs 分区的大小



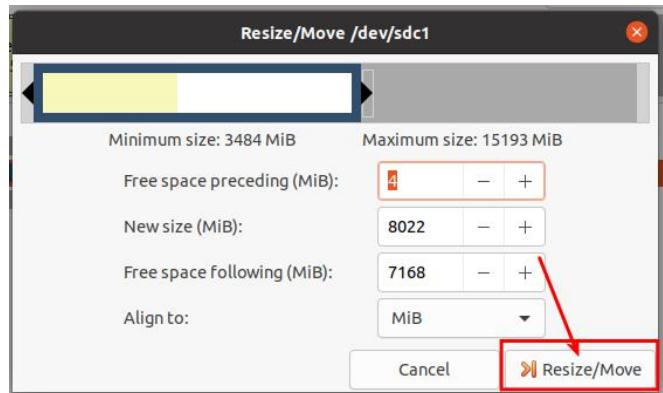
8) **Resize/Move** 选项打开后会弹出下面的设置界面



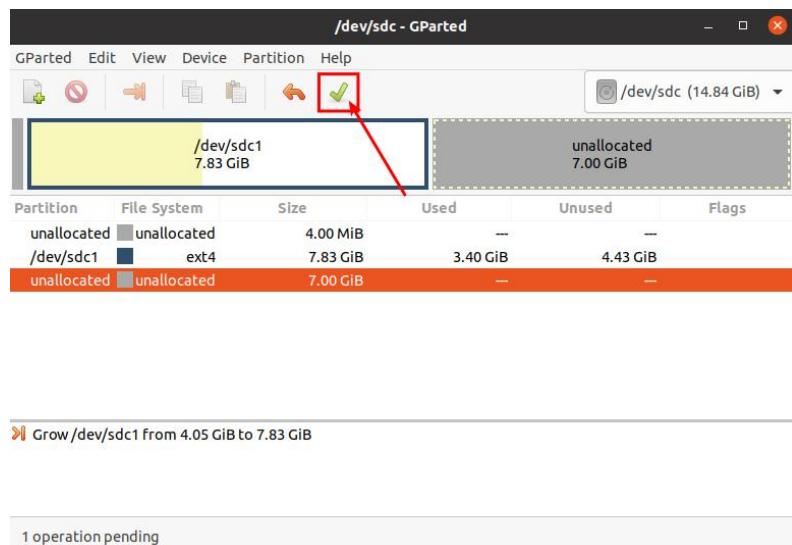
9) 然后可以直接拖动下图所示的位置来设置容量的大小，也可以通过设置 **New size(MiB)** 中的数字来设置 rootfs 分区的大小



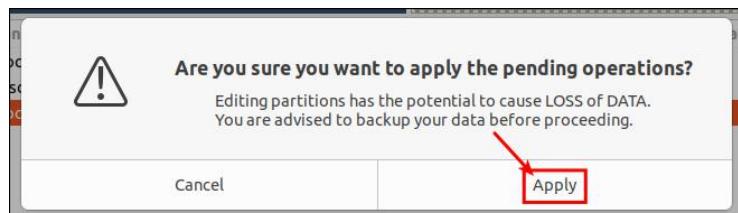
10) 设置好容量后，再点击右下角的 **Resize/Move** 即可



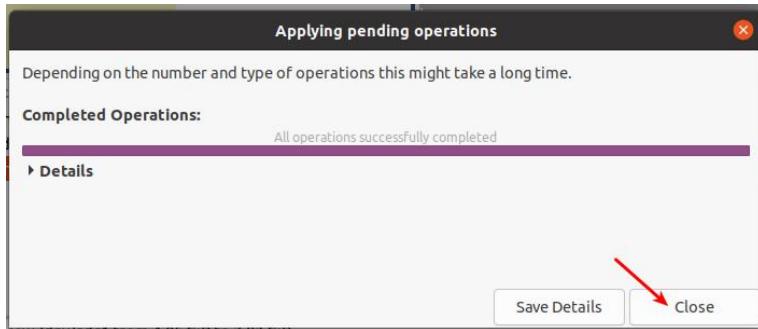
11) 最后确认无误后，再点击下图所示的绿色 ✓



12) 然后选择 **Apply**，就会正式开始扩容 rootfs 分区的容量



13) 扩容完成后点击 **Close** 关闭即可

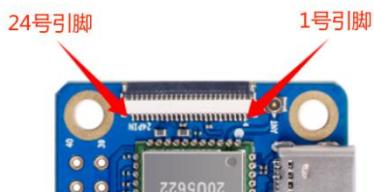


14) 然后就可以把 TF 卡拔下来，再插到开发板中启动，进入开发板的 Linux 系统中后如果使用 **df -h** 命令可以看到 rootfs 分区的大小和前面设置的大小一致的话就说明缩小容量成功

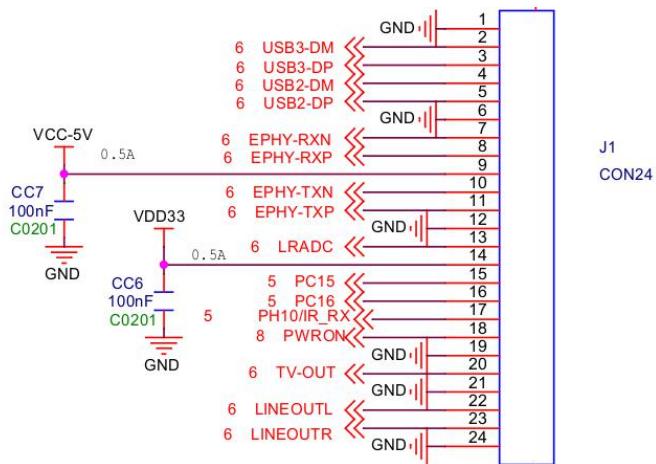
```
root@orangeipi:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            925M    0   925M   0% /dev
tmpfs           199M  3.2M  196M   2% /run
/dev/mmcblk0p1  7.7G  3.2G  4.4G  42% /
```

3. 7. 24Pin 扩展板接口引脚说明

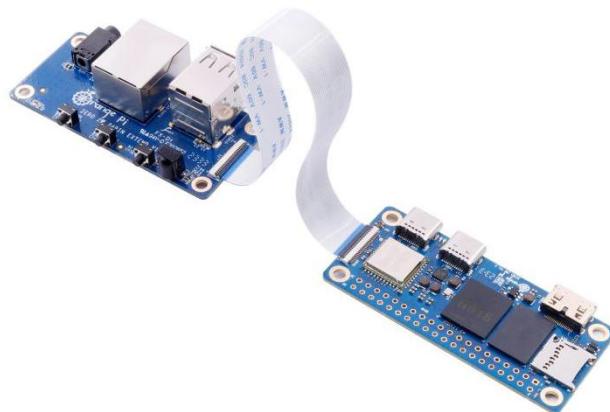
1) 开发板 24 pin 扩展板接口引脚的顺序请参考下图



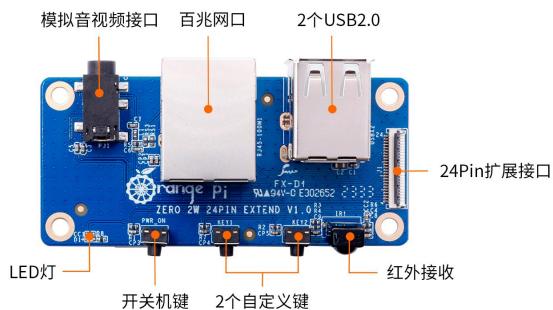
2) 开发板 24pin 扩展板接口的原理图如下所示



3) 扩展板接入开发板的方式如下所示，请注意排线的正反方向，不要插反了



4) 扩展板可以扩展的功能有



1	百兆网口	用于连接有线网络来上网
2	模拟音视频输出接口	在 Linux 系统中可用于接耳机播放音乐，TV-OUT 功能无法使用
3	USB 2.0 Host x 2	用于接 USB 键盘、鼠标以及 USB 存储设备
4	红外接收功能	目前主要用于 Android 系统，Linux 系统主要适



		配了内核驱动
5	开关机按键	用于关闭或开启开发板
6	LRADC 自定义按键 x 2	Linux 系统默认设置为 KEY_1 (数字 1 按键) 和 KEY_ENTER (回车按键)，可以通过修改 dts 配置自定义为其他功能按键

5) Linux5.4 和 Linux6.1 系统对扩展板的适配情况如下表所示

24pin 扩展板功能	Linux5.4	Linux6.1
百兆网口	OK	OK
百兆网口灯	OK	OK
USB2.0 HOST x 2	OK	OK
红外接收	OK	OK
耳机音频播放	OK	OK
开关机按键	OK	OK
LRADC 自定义按键 x 2	OK	OK
TV-OUT	NO	NO

3.8. 24pin 扩展板上的 2 个 LRADC 按键的使用方法

1) 24pin 扩展板上有两个 LRADC 按键，位置如下图所示：



2) 在 Linux 系统中，KEY1 和 KEY2 默认设置的键值为

Linux 内核	KEY1	KEY2
Linux5.4	KEY_1, 即键盘上的数字 1	KEY_ENTER, 即回车键
Linux6.1	KEY_1, 即键盘上的数字 1	KEY_ENTER, 即回车键

3) 通过 `evtest` 命令我们可以查看下 KEY1 和 KEY2 按下后上报的键值



a. linux5.4

```
orangepi@orangepizero2w:~$ evtest
No device specified, trying to scan all of /dev/input/event*
Not running as root, no devices may be available.
Available devices:
/dev/input/event0: sunxi-keyboard
/dev/input/event1: sunxi-ir
/dev/input/event2: axp2101-pek
/dev/input/event3: SONiX USB Keyboard
/dev/input/event4: SONiX USB Keyboard Consumer Control
/dev/input/event5: SONiX USB Keyboard System Control
/dev/input/event6: PixArt USB Optical Mouse
/dev/input/event7: BRLTTY 6.3 Linux Screen Driver Keyboard
Select the device event number [0-7]: 0 #需要输入 sunxi-keyboard 对应的序号
Input driver version is 1.0.1
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "sunxi-keyboard"
Supported events:
    Event type 0 (EV_SYN)
    Event type 1 (EV_KEY)
        Event code 2 (KEY_1)
        Event code 28 (KEY_ENTER)
Properties:
Testing ... (interrupt to exit)

#下面是按下 KEY1 和 KEY2 后上报的键值

Event: time 1693555298.132314, type 1 (EV_KEY), code 2 (KEY_1), value 1
Event: time 1693555298.132314, ----- SYN_REPORT -----
Event: time 1693555298.226071, type 1 (EV_KEY), code 2 (KEY_1), value 0
Event: time 1693555298.226071, ----- SYN_REPORT -----
Event: time 1693555298.601042, type 1 (EV_KEY), code 28 (KEY_ENTER), value 1
Event: time 1693555298.601042, ----- SYN_REPORT -----
Event: time 1693555298.710415, type 1 (EV_KEY), code 28 (KEY_ENTER), value 0
Event: time 1693555298.710415, ----- SYN_REPORT -----
```

b. linux6.1



```
orangeipi@orangepirzero2w:~$ evtest
No device specified, trying to scan all of /dev/input/event*
Not running as root, no devices may be available.

Available devices:
/dev/input/event0:      axp20x-pek
/dev/input/event1:      5070800.lradc
/dev/input/event2:      SONiX USB Keyboard
/dev/input/event3:      SONiX USB Keyboard Consumer Control
/dev/input/event4:      SONiX USB Keyboard System Control
/dev/input/event5:      PixArt USB Optical Mouse
/dev/input/event6:      sunxi-ir

Select the device event number [0-6]: 1 #需要输入 5070800.lradc 对应的序号
Input driver version is 1.0.1
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "5070800.lradc"

Supported events:
    Event type 0 (EV_SYN)
    Event type 1 (EV_KEY)
        Event code 2 (KEY_1)
        Event code 28 (KEY_ENTER)

Properties:
Testing ... (interrupt to exit)

#下面是按下 KEY1 和 KEY2 后上报的键值

Event: time 1694075818.810877, type 1 (EV_KEY), code 2 (KEY_1), value 1
Event: time 1694075818.810877, ----- SYN_REPORT -----
Event: time 1694075818.961345, type 1 (EV_KEY), code 2 (KEY_1), value 0
Event: time 1694075818.961345, ----- SYN_REPORT -----
Event: time 1694075819.536128, type 1 (EV_KEY), code 28 (KEY_ENTER), value 1
Event: time 1694075819.536128, ----- SYN_REPORT -----
Event: time 1694075819.705009, type 1 (EV_KEY), code 28 (KEY_ENTER), value 0
Event: time 1694075819.705009, ----- SYN_REPORT -----
```

- 4) 如果需要修改 KEY1 和 KEY2 按下后上报的按键值，可以使用下面的方法：
 - a. 在/usr/src/路径下有一个 **sun50i-h618-lradc-keys.dts** 文件，我们可以通过它



来定义 KEY1 和 KEY2 为想要的按键值

```
orangepi@orangepizero2w:~$ cd /usr/src/  
orangepi@orangepizero2w:/usr/src$ ls *.dts  
sun50i-h618-lradc-keys.dts
```

b. linux5.4 系统 **sun50i-h618-lradc-keys.dts** 文件的内容如下所示：

- a) KEY1 对应：修改 **key0 = <600 2>**; 中的 2 为想要的按键值对应的数字
- b) KEY2 对应：修改 **key1 = <800 28>**; 中的 28 为想要的按键值对应的数字

```
orangepi@orangepizero2w:/usr/src$ sudo vim sun50i-h618-lradc-keys.dts
```

```
/dts-v1;/  
/plugin/;  
  
{  
    fragment@0 {  
        target = <&keyboard>;  
  
        __overlay__ {  
            status = "okay";  
  
            key0 = <600 2>;  
            key1 = <800 28>;  
        };  
    };  
};
```

c. linux6.1 系统 **sun50i-h618-lradc-keys.dts** 文件的内容如下所示：

- a) KEY1 对应：修改 **linux,code = <2>**; 中的 2 为想要的按键值对应的数字
- b) KEY2 对应：修改 **linux,code = <28>**; 中的 28 为想要的按键值对应的数字

```
orangepi@orangepizero2w:/usr/src$ sudo vim sun50i-h618-lradc-keys.dts
```

```
/dts-v1;/  
/plugin/;  
  
{  
    fragment@0 {  
        target = <&r_lradc>;  
    };  
};
```



```
__overlay__ {
    status = "okay";
```



```
button-500 {
    label = "KEY_1";
    linux,code = <2>;
};
```



```
button-800 {
    label = "KEY_ENTER";
    linux,code = <28>;
};
```

```
};
```

```
};
```

- d. 可以设置的按键值请参考下 **input-event-codes.h** 头文件中的宏定义，其在内核源码中的路径为：

orange-pi-5.4-sun50iw9/include/uapi/linux/input-event-codes.h
orange-pi-6.1-sun50iw9/include/uapi/linux/input-event-codes.h

- e. 修改完后，再使用 **orangepi-add-overlay** 命令让 sun50i-h618-lradc-keys.dts 配置添加到系统中

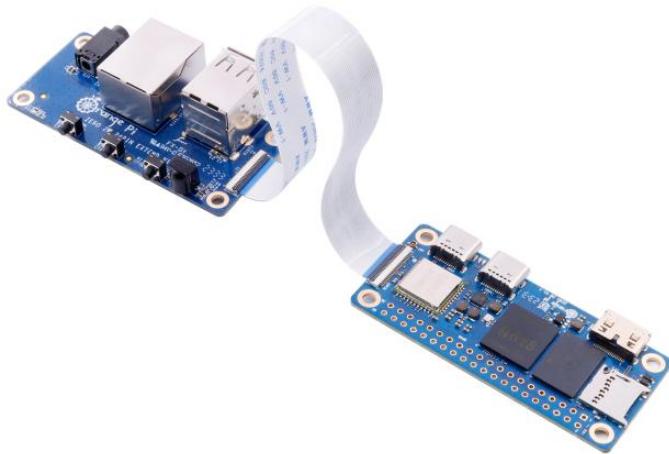
```
orangepi@orangepirzero2w:/usr/src$ sudo orangepi-add-overlay sun50i-h618-lrade-keys.dts  
Compiling the overlay  
Copying the compiled overlay file to /boot/overlay-user/  
Reboot is required to apply the changes
```

- f. 然后重启系统自定义的按键值就会生效了

3.9. 网络连接测试

3.9.1. 以太网口测试

- 1) 开发板主板上是没有有线网络接口的，我们可以通过 24pin 扩展板来扩展百兆有线网络接口



2) 然后将网线的一端插入扩展板的以太网接口，网线的另一端接入路由器，并确保网络是畅通的

3) 系统启动后会通过 **DHCP** 自动给以太网卡分配 IP 地址，**不需要其他任何配置**

4) 在开发板的 Linux 系统中查看 IP 地址的命令如下所示：

下面的命令请不要照抄，比如 **debian12** 中的网络节点名为 **end0**，下面的命令就需要修改为 **ip a s end0**。

```
orangepi@orangepi:~$ ip a s eth0
3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 5e:ac:14:a5:b3 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.16/24 brd 192.168.1.255 scope global dynamic noprefixroute eth0
        valid_lft 259174sec preferred_lft 259174sec
    inet6 240e:3b7:3240:c3a0:e269:8305:dc08:135e/64 scope global dynamic
noprefixroute
        valid_lft 259176sec preferred_lft 172776sec
    inet6 fe80::957d:bbbd:4928:3604/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

开发板启动后查看 IP 地址有三种方法：

1. 接 **HDMI** 显示器，然后登录系统使用 **ip a s eth0** 命令查看 IP 地址
2. 在调试串口终端输入 **ip a s eth0** 命令来查看 IP 地址
3. 如果没有调试串口，也没有 **HDMI** 显示器，还可以通过路由器的管理界面来查



看开发板网口的 IP 地址。不过这种方法经常有人会无法正常看到开发板的 IP 地址。如果看不到，调试方法如下所示：

- A) 首先检查 Linux 系统是否已经正常启动，如果开发板的绿灯闪烁了，一般是正常启动了，如果只亮红灯，或者红灯绿灯都没亮，说明系统都没正常启动；
- B) 检查网线有没有插紧，或者换根网线试下；
- C) 换个路由器试下（路由器的问题有遇到过很多，比如路由器无法正常分配 IP 地址，或者已正常分配 IP 地址但在路由器中看不到）；
- D) 如果没有路由器可换就只能连接 HDMI 显示器或者使用调试串口来查看 IP 地址。

另外需要注意的是开发板 DHCP 自动分配 IP 地址是不需要任何设置的。

5) 测试网络连通性的命令如下，**ping** 命令可以通过 **Ctrl+C** 快捷键来中断运行

下面的命令请不要照抄，比如 debian12 中的网络节点名为 **end0**，下面的命令就需要修改为 **ping www.baidu.com -I end0**。

```
orangepi@orangepi:~$ ping www.baidu.com -I eth0
PING www.a.shifen.com (14.215.177.38) from 192.168.1.12 eth0: 56(84) bytes of data.
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=1 ttl=56 time=6.74 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=2 ttl=56 time=6.80 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=3 ttl=56 time=6.26 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=4 ttl=56 time=7.27 ms
^C
--- www.a.shifen.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 6.260/6.770/7.275/0.373 ms
```

3.9.2. WIFI 连接测试

请不要通过修改/etc/network/interfaces 配置文件的方式来连接 WIFI，通过这种方式连接 WIFI 网络使用会有问题。

3.9.2.1. 服务器版镜像通过命令连接 WIFI

当开发板没有连接以太网，没有连接 HDMI 显示器，只连接了串口时，推荐使用此小节演示的命令来连接 WIFI 网络。因为 nmtui 在某些串口软件（如 minicom）



中只能显示字符，无法正常显示图形界面。当然，如果开发板连接了以太网或者 HDMI 显示屏，也可以使用此小节演示的命令来连接 WIFI 网络的。

1) 先登录 linux 系统，有下面三种方式

- a. 如果开发板连接了网线，可以通过 **ssh 远程登录 linux 系统**
- a. 如果开发板连接好了调试串口，可以使用串口终端登录 linux 系统
- b. 如果连接了开发板到HDMI 显示器，可以通过 HDMI 显示的终端登录到linux 系统

2) 首先使用 **nmcli dev wifi** 命令扫描周围的 WIFI 热点

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

```
root@orangeipi:~# nmcli dev wifi
IN-USE   BSSID           SSID      MODE   CHAN  RATE    SIGNAL  BARS  SECURITY
28:6C:07:6E:87:2E  [REDACTED]_orangeipi  Infra   9     260 Mbit/s 97      [REDACTED] WPA1 WPA2
D8:D8:66:A5:BD:D1  [REDACTED]          Infra   10    270 Mbit/s 90      [REDACTED] WPA1 WPA2
A0:40:A0:A1:72:20  [REDACTED]          Infra   4     405 Mbit/s 82      [REDACTED] WPA2
28:6C:07:6E:87:2F  [REDACTED]_orangeipi_5G Infra  149    540 Mbit/s 80      [REDACTED] WPA1 WPA2
CA:50:E9:89:E2:44  ChinaNet_TG15       Infra   1     130 Mbit/s 79      [REDACTED] WPA1 WPA2
A0:40:A0:A1:72:31  NETGEAR00          Infra  100    405 Mbit/s 67      [REDACTED] WPA2
D4:EE:07:08:A9:E0  [REDACTED]          Infra   4     130 Mbit/s 55      [REDACTED] WPA1 WPA2
88:C3:97:49:25:13  [REDACTED]          Infra   6     130 Mbit/s 52      [REDACTED] WPA1 WPA2
00:BD:82:51:53:C2  [REDACTED]          Infra   12    130 Mbit/s 49      [REDACTED] WPA1 WPA2
C0:61:18:FA:49:37  [REDACTED]          Infra  149    270 Mbit/s 47      [REDACTED] WPA1 WPA2
04:79:70:8D:0C:B8  [REDACTED]          Infra  153    270 Mbit/s 47      [REDACTED] WPA2
04:79:70:FD:0C:B8  [REDACTED]          Infra  153    270 Mbit/s 47      [REDACTED] WPA2
9C:A6:15:DD:E6:0C  [REDACTED]          Infra   10    270 Mbit/s 45      [REDACTED] WPA1 WPA2
B4:0F:3B:45:D1:F5  [REDACTED]          Infra   48     270 Mbit/s 45      [REDACTED] WPA1 WPA2
E8:CC:18:4F:7B:44  [REDACTED]          Infra  157    135 Mbit/s 45      [REDACTED] WPA1 WPA2
B0:95:8E:D8:2F:ED  [REDACTED]          Infra   11     405 Mbit/s 39      [REDACTED] WPA1 WPA2
C0:61:18:FA:49:36  [REDACTED]          Infra   11    270 Mbit/s 24      [REDACTED] WPA1 WPA2
root@orangeipi:~#
```

3) 然后使用 **nmcli** 命令连接扫描到的 WIFI 热点，其中：

- a. **wifi_name** 需要换成想连接的 WIFI 热点的名字
- b. **wifi_passwd** 需要换成想连接的 WIFI 热点的密码

```
orangeipi@orangeipi:~$ sudo nmcli dev wifi connect wifi_name password wifi_passwd
```

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

4) 通过 **ip addr show wlan0** 命令可以查看 wifi 的 IP 地址

```
orangeipi@orangeipi:~$ ip a s wlan0
1: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
    link/ether 23:8c:d6:ae:76:bb brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.11/24 brd 192.168.1.255 scope global dynamic noprefixroute wlan0
```



```
valid_lft 259192sec preferred_lft 259192sec
inet6 240e:3b7:3240:c3a0:c401:a445:5002:ccdd/64 scope global dynamic
noprefixroute
    valid_lft 259192sec preferred_lft 172792sec
    inet6 fe80::42f1:6019:a80e:4c31/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

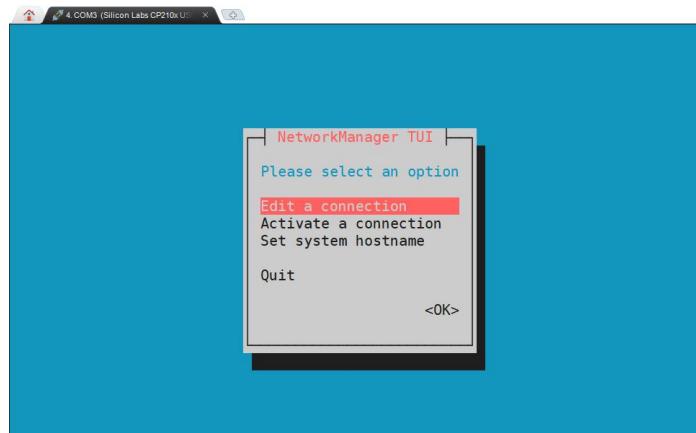
- 5) 使用 **ping** 命令可以测试 wifi 网络的连通性，**ping** 命令可以通过 **Ctrl+C** 快捷键来中断运行

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

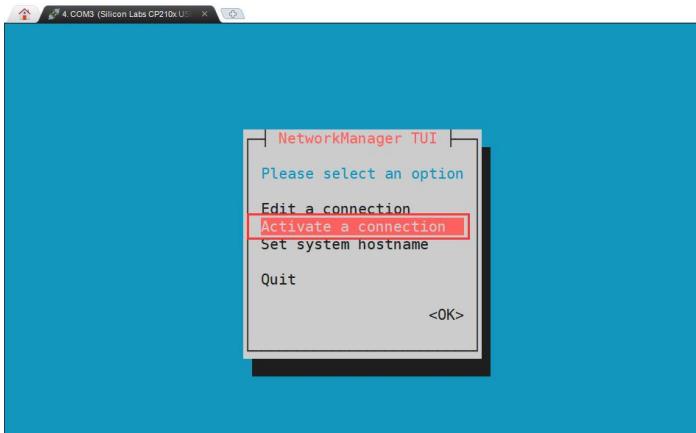
3. 9. 2. 2. 服务器版镜像通过图形化方式连接 WIFI

- 1) 先登录 linux 系统，有下面三种方式
 - a. 如果开发板连接了网线，可以通过 **ssh 远程登录 linux 系统**
 - b. 如果开发板连接好了调试串口，可以使用串口终端登录 linux 系统（串口软件请使用 MobaXterm，使用 minicom 无法显示图形界面）
 - c. 如果连接了开发板到HDMI 显示器，可以通过 HDMI 显示的终端登录到 linux 系统
- 2) 然后在命令行中输入 nmtui 命令打开 wifi 连接的界面

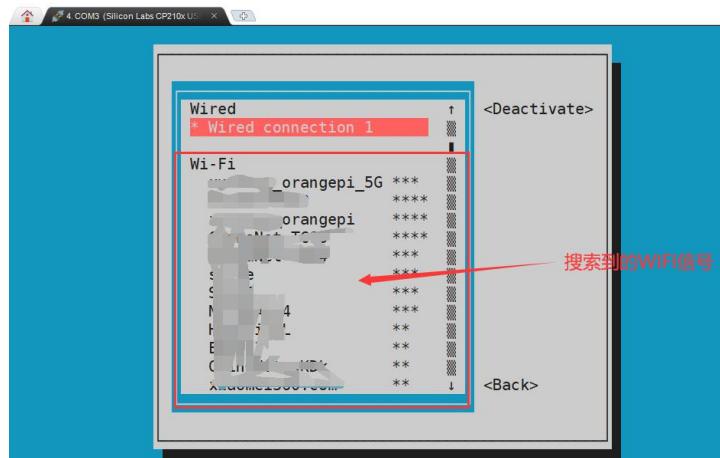
```
orangeipi@orangeipi:~$ sudo nmtui
```
- 3) 输入 nmtui 命令打开的界面如下所示



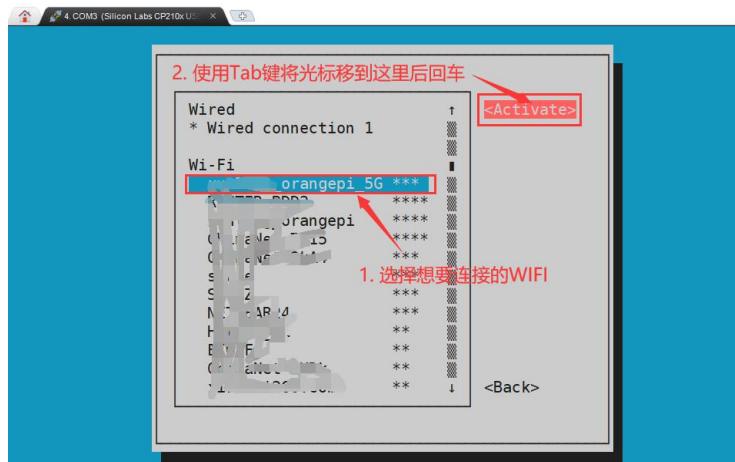
4) 选择 **Activate a connect** 后回车



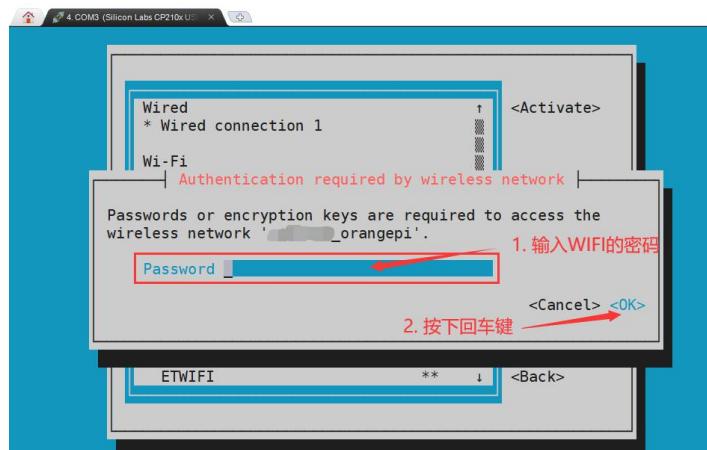
5) 然后就能看到所有搜索到的 WIFI 热点



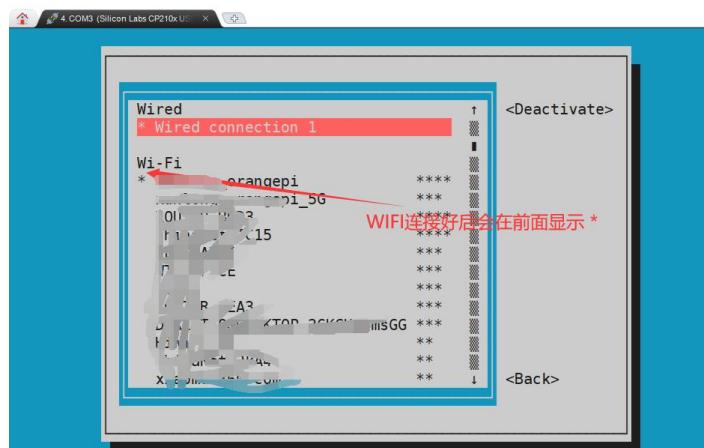
6) 选择想要连接的 WIFI 热点后再使用 Tab 键将光标定位到 **Activate** 后回车



7) 然后会弹出输入密码的对话框，在 **Password** 内输入对应的密码然后回车就会开始连接 WIFI



8) WIFI 连接成功后会在已连接的 WIFI 名称前显示一个 “*”



9) 通过 **ip a s wlan0** 命令可以查看 wifi 的 IP 地址



```
orangeipi@orangeipi:~$ ip a s wlan0
1: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
    link/ether 24:8c:d3:aa:76:bb brd ff:ff:ff:ff:ff:ff
        inet 192.168.1.11/24 brd 192.168.1.255 scope global dynamic noprefixroute wlan0
            valid_lft 259069sec preferred_lft 259069sec
        inet6 240e:3b7:3240:c4a0:c401:a445:5002:ccdd/64 scope global dynamic
noprefixroute
            valid_lft 259071sec preferred_lft 172671sec
        inet6 fe80::42f1:6019:a80e:4c31/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
```

- 10) 使用 **ping** 命令可以测试 wifi 网络的连通性，**ping** 命令可以通过 **Ctrl+C** 快捷键来中断运行

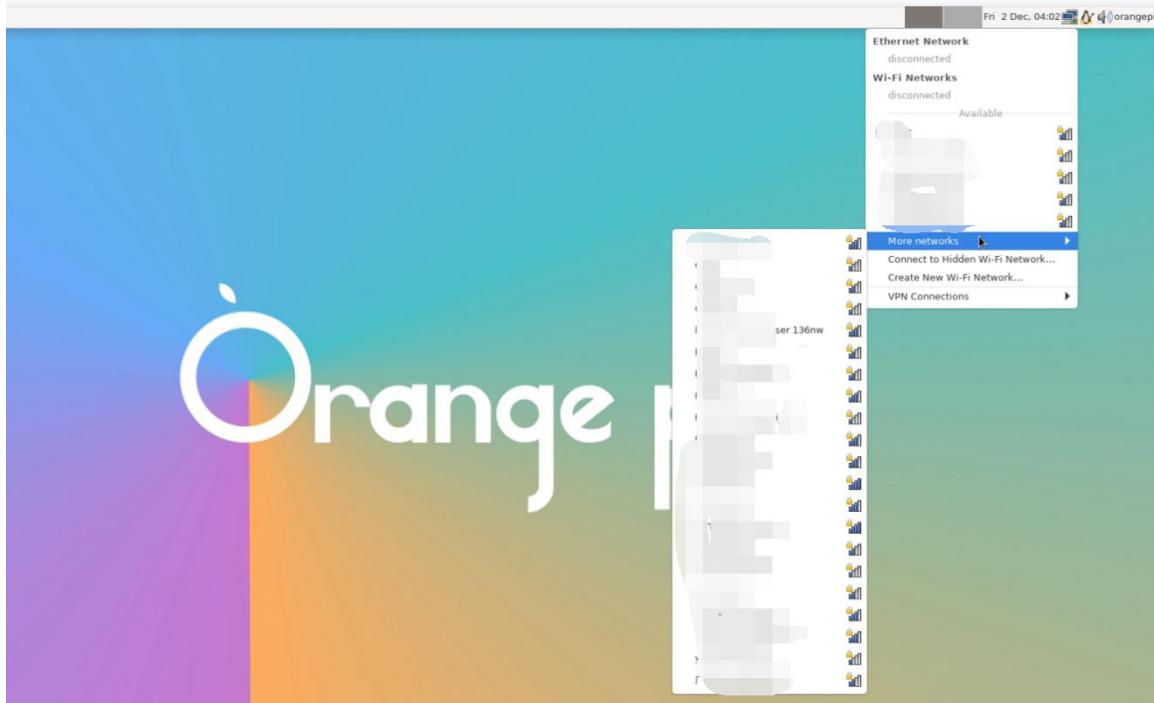
```
orangeipi@orangeipi:~$ ping www.orangeipi.org -I wlan0
PING www.orangeipi.org (182.92.236.130) from 192.168.1.49 wlan0: 56(84) bytes of
data.
64 bytes from 182.92.236.130 (182.92.236.130): icmp_seq=1 ttl=52 time=43.5 ms
64 bytes from 182.92.236.130 (182.92.236.130): icmp_seq=2 ttl=52 time=41.3 ms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

3. 9. 2. 3. 桌面版镜像的测试方法

- 1) 点击桌面右上角的网络配置图标（测试 WIFI 时请不要连接网线）



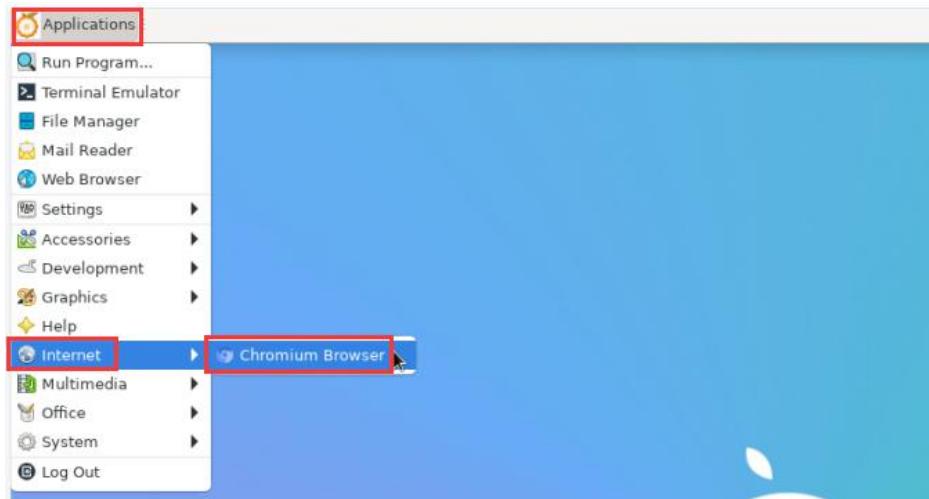
2) 在弹出的下拉框中点击 **More networks** 可以看到所有扫描到的 WIFI 热点，然后选择想要连接的 WIFI 热点



3) 然后输入 WIFI 热点的密码，再点击 **Connect** 就会开始连接 WIFI



4) 连接好 WIFI 后，可以打开浏览器查看是否能上网，浏览器的入口如下图所示



5) 打开浏览器后如果能打开其他网页说明 WIFI 连接正常



3.9.3. 通过 `create_ap` 创建 WIFI 热点的方法

`create_ap`是一个帮助快速创建Linux上的WIFI热点的脚本，并且支持bridge和NAT模式，能够自动结合hostapd, dnsmasq和iptables完成WIFI热点的设置，避免了用户进行复杂的配置，github地址如下：

https://github.com/oblique/create_ap

OPi发布的Linux镜像已经预装了`create_ap`脚本，可以通过`create_ap`命令来创建WIFI热点，`create_ap`的基本命令格式如下所示：



```
create_ap [options] <wifi-interface> [<interface-with-internet>]
[<access-point-name> [<passphrase>]]
```

* **options:** 可以通过该参数指定加密方式、WIFI热点的频段、频宽模式、网络共享方式等，具体可以通过**create_ap -h**获取到有哪些option

* **wifi-interface:** 无线网卡的名称

* **interface-with-internet:** 可以联网的网卡名称，一般是eth0

* **access-point-name:** 热点名称

* **passphrase:** 热点的密码

3. 9. 3. 1. **create_ap** 以 NAT 模式创建 WIFI 热点的方法

1) 输入下面的命令以 NAT 模式创建名称为 **orangeipi**、密码为 **orangeipi** 的 WIFI 热点

注意，下面的命令中，Debian12 需要修改 eth0 为 end0。

```
orangeipi@orangeipi:~$ sudo create_ap -m nat wlan0 eth0 orangeipi orangeipi --no-virt
```

2) 如果有下面的信息输出，说明 WIFI 热点创建成功

```
orangeipi@orangeipi:~$ sudo create_ap -m nat wlan0 eth0 orangeipi orangeipi --no-virt
Config dir: /tmp/create_ap.wlan0.conf.TQkJtsz1
PID: 26139
Network Manager found, set wlan0 as unmanaged device... DONE
Sharing Internet using method: nat
hostapd command-line interface: hostapd_cli -p
/tmp/create_ap.wlan0.conf.TQkJtsz1/hostapd_ctrl
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
wlan0: STA ce:bd:9a:dd:a5:86 IEEE 802.11: associated
wlan0: AP-STA-CONNECTED ce:bd:9a:dd:a5:86
wlan0: STA ce:bd:9a:dd:a5:86 RADIUS: starting accounting session
D4FBF7E5C604F169
wlan0: STA ce:bd:9a:dd:a5:86 WPA: pairwise key handshake completed (RSN)
wlan0: EAPOL-4WAY-HS-COMPLETED ce:bd:9a:dd:a5:86
```



3) 此时拿出手机，在搜索到的 WIFI 列表中就能找到开发板创建的名为 **orangeipi** 的 WIFI 热点，然后可以点击 **orangeipi** 连接热点，密码就是上面设置的 **orangeipi**



4) 连接成功后的显示如下图所示



5) 在 NAT 模式下，连接到开发板热点的无线设备是向开发板的 DHCP 服务请求 IP 地址的，所以会有两个不同的网段，如这里开发板的 IP 是 192.168.1.X

注意，下面的命令中，Debian12 需要修改 eth0 为 end0。

```
orangeipi@orangeipi:~$ sudo ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
      inet 192.168.1.150  netmask 255.255.255.0  broadcast 192.168.1.255
          inet6 fe80::938f:8776:5783:afa2  prefixlen 64  scopeid 0x20<link>
              ether 4a:a0:c8:25:42:82  txqueuelen 1000  (Ethernet)
                  RX packets 25370  bytes 2709590 (2.7 MB)
                  RX errors 0  dropped 50  overruns 0  frame 0
                  TX packets 3798  bytes 1519493 (1.5 MB)
                  TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
                  device interrupt 83
```

而开发板的 DHCP 服务默认会给接入热点的设备分配 **192.168.12.0/24** 的 IP 地址，这时点击已经连接的 WIFI 热点 **orangeipi**，然后就可以看到手机的 IP 地址是 **192.168.12.X**。



6) 如果想要为接入的设备指定不同的网段，可以通过-g 参数指定，如通过-g 参数指定接入点 AP 的网段为 192.168.2.1

注意，下面的命令中，Debian12 需要修改 eth0 为 end0。

```
orangeipi@orangeipi:~$ sudo create_ap -m nat wlan0 eth0 orangeipi orangeipi -g 192.168.2.1 --no-virt
```

此时通过手机连接到热点后，点击已经连接的 WIFI 热点 **orangeipi**，然后可以看到手机的 IP 地址是 **192.168.2.X**。





7) 在不指定**--freq-band** 参数的情况下，默认创建的热点是 2.4G 频段的，如果想要创建 5G 频段的热点可以通过**--freq-band 5** 参数指定，具体命令如下

注意，下面的命令中，Debian12 需要修改 eth0 为 end0。

```
orangeipi@orangeipi:~$ sudo create_ap -m nat wlan0 eth0 orangeipi orangeipi --freq-band 5 --no-virt
```

8) 如果需要隐藏 SSID，可以指定**--hidden** 参数，具体命令如下

注意，下面的命令中，Debian12 需要修改 eth0 为 end0。

```
orangeipi@orangeipi:~$ sudo create_ap -m nat wlan0 eth0 orangeipi orangeipi --hidden --no-virt
```

此时手机是搜索不到 WIFI 热点的，需要手动指定 WIFI 热点名称，并输入密码来连接 WIFI 热点



3.9.3.2. **create_ap** 以 bridge 模式创建 WIFI 热点的方法

1) 输入下面的命令以 bridge 模式创建名称为 **orangeipi**、密码为 **orangeipi** 的 WIFI 热点

注意，下面的命令中，Debian12 需要修改 eth0 为 end0。

```
orangeipi@orangeipi:~$ sudo create_ap -m bridge wlan0 eth0 orangeipi orangeipi --no-virt
```

2) 如果有下面的信息输出，说明 WIFI 热点创建成功

```
orangeipi@orangeipi:~$ sudo create_ap -m bridge wlan0 eth0 orangeipi orangeipi --no-virt
```

```
Config dir: /tmp/create_ap.wlan0.conf.zAcFlYTx
```

```
PID: 27707
```

```
Network Manager found, set wlan0 as unmanaged device... DONE
```



Sharing Internet using method: bridge

Create a bridge interface... br0 created.

hostapd command-line interface: hostapd_cli -p

/tmp/create_ap.wlan0.conf.zAcFIYTx/hostapd_ctrl

wlan0: interface state UNINITIALIZED->ENABLED

wlan0: AP-ENABLED

wlan0: STA ce:bd:9a:dd:a5:86 IEEE 802.11: associated

wlan0: AP-STA-CONNECTED ce:bd:9a:dd:a5:86

wlan0: STA ce:bd:9a:dd:a5:86 RADIUS: starting accounting session

937BF40E51897A7B

wlan0: STA ce:bd:9a:dd:a5:86 WPA: pairwise key handshake completed (RSN)

wlan0: EAPOL-4WAY-HS-COMPLETED ce:bd:9a:dd:a5:86

3) 此时拿出手机，在搜索到的 WIFI 列表中就能找到开发板创建的名为 **orangeipi** 的 WIFI 热点，然后可以点击 **orangeipi** 连接热点，密码就是上面设置的 **orangeipi**



4) 连接成功后的显示如下图所示



5) 在 bridge 模式下，连接到开发板热点的无线设备也是向主路由（开发板连接的路由器）的 DHCP 服务请求 IP 地址的，如这里开发板的 IP 是 **192.168.1.X**

orangeipi@orangeipi:~\$ sudo ifconfig eth0



```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.150 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::938f:8776:5783:afa2 prefixlen 64 scopeid 0x20<link>
        ether 4a:a0:c8:25:42:82 txqueuelen 1000 (Ethernet)
        RX packets 25370 bytes 2709590 (2.7 MB)
        RX errors 0 dropped 50 overruns 0 frame 0
        TX packets 3798 bytes 1519493 (1.5 MB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
        device interrupt 83
```

而接入 WIFI 热点的设备的 IP 也是由主路由分配的，所以连接 WIFI 热点的手机和开发板处于相同的网段，这时点击已经连接的 WIFI 热点 **orangeipi**，然后就可以看到手机的 IP 地址也是 **192.168.1.X**。



6) 在不指定**--freq-band**参数的情况下，默认创建的热点是 2.4G 频段的，如果想要创建 5G 频段的热点可以通过**--freq-band 5** 参数指定，具体命令如下

注意，下面的命令中，Debian12 需要修改 eth0 为 end0。

```
orangeipi@orangeipi:~$ sudo create_ap -m bridge wlan0 eth0 orangeipi orangeipi --freq-band 5 --no-virt
```

7) 如果需要隐藏 SSID，可以指定**--hidden** 参数，具体命令如下

注意，下面的命令中，Debian12 需要修改 eth0 为 end0。

```
orangeipi@orangeipi:~$ sudo create_ap -m bridge wlan0 eth0 orangeipi orangeipi --hidden --no-virt
```



此时手机是搜索不到 WIFI 热点的，需要手动指定 WIFI 热点名称，并输入密码来连接 WIFI 热点



3.9.4. 设置静态 IP 地址的方法

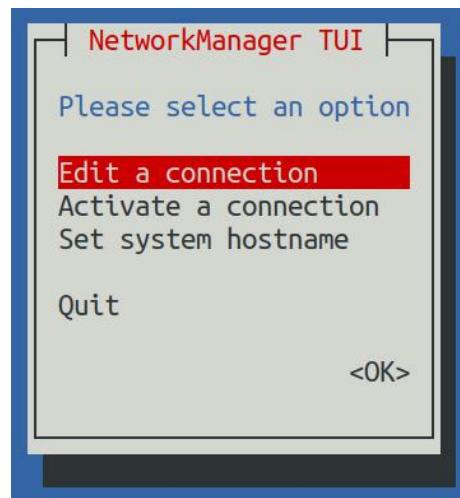
请不要通过修改/etc/network/interfaces 配置文件的方式来设置静态 IP 地址。

3.9.4.1. 使用 nmtui 命令来设置静态 IP 地址

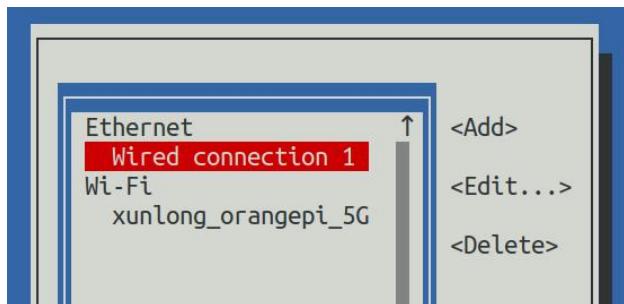
1) 首先运行 **nmtui** 命令

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

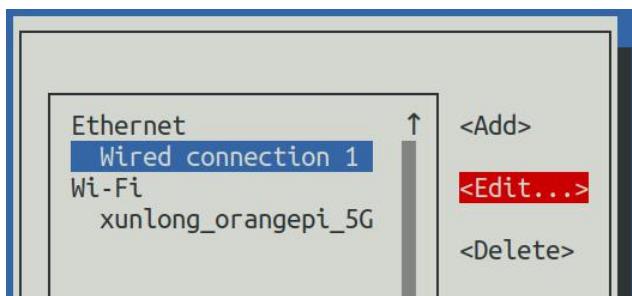
2) 然后选择 **Edit a connection** 并按下回车键



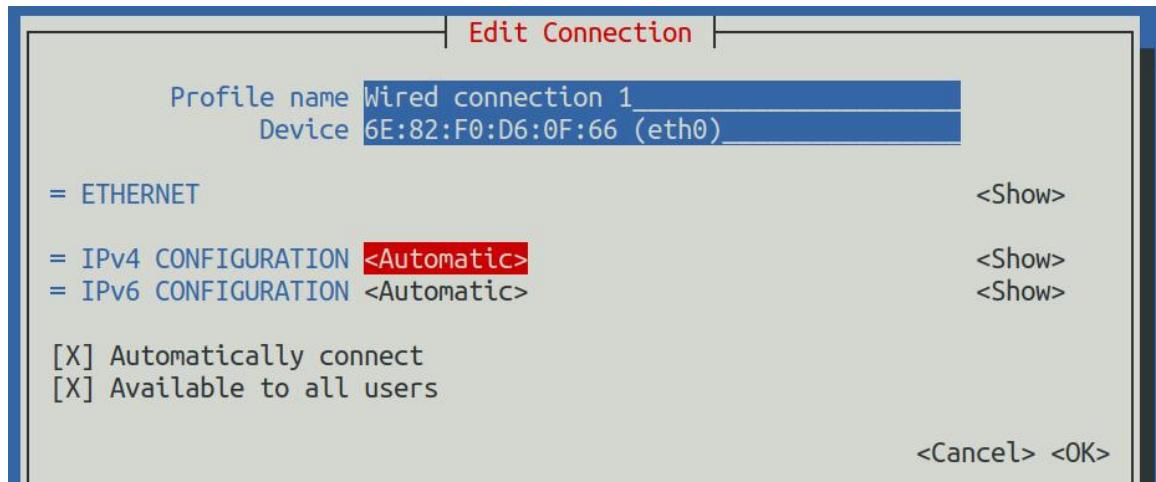
3) 然后选择需要设置静态 IP 地址的网络接口，比如设置 **Ethernet** 接口的静态 IP 地址选择 **Wired connection 1** 就可以了



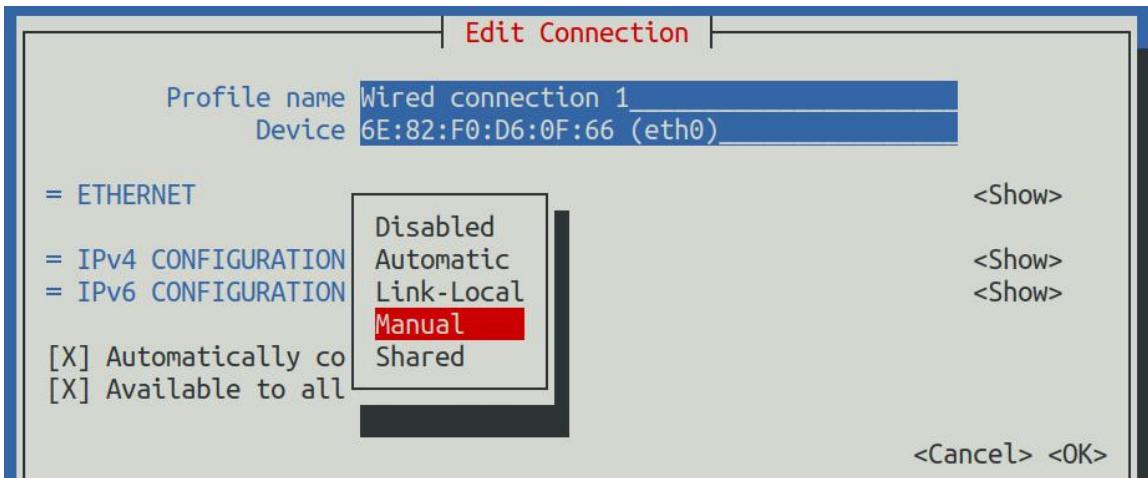
4) 然后通过 **Tab** 键选择 **Edit** 并按下回车键



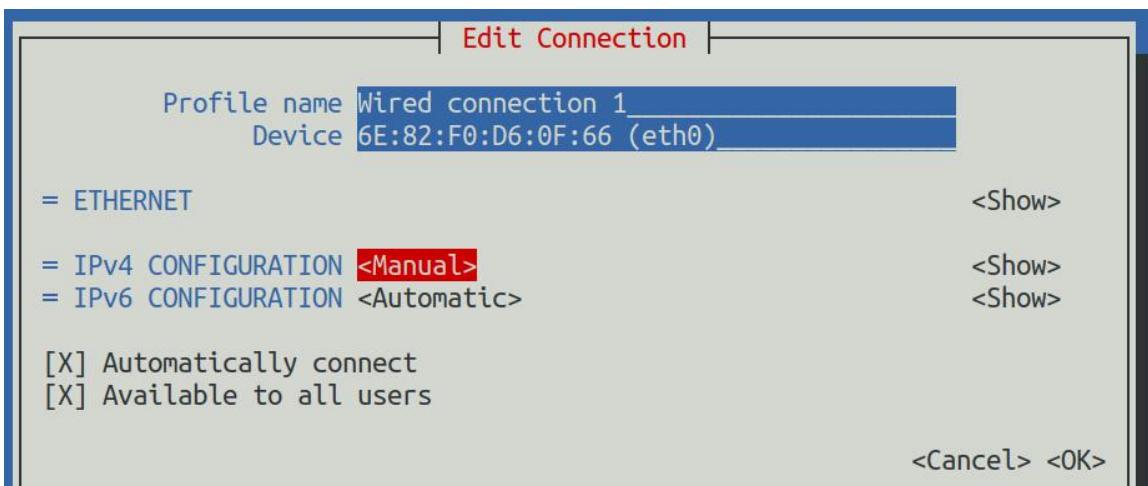
5) 然后通过 Tab 键将光标移动到下图所示的<Automatic>位置进行 IPv4 的配置



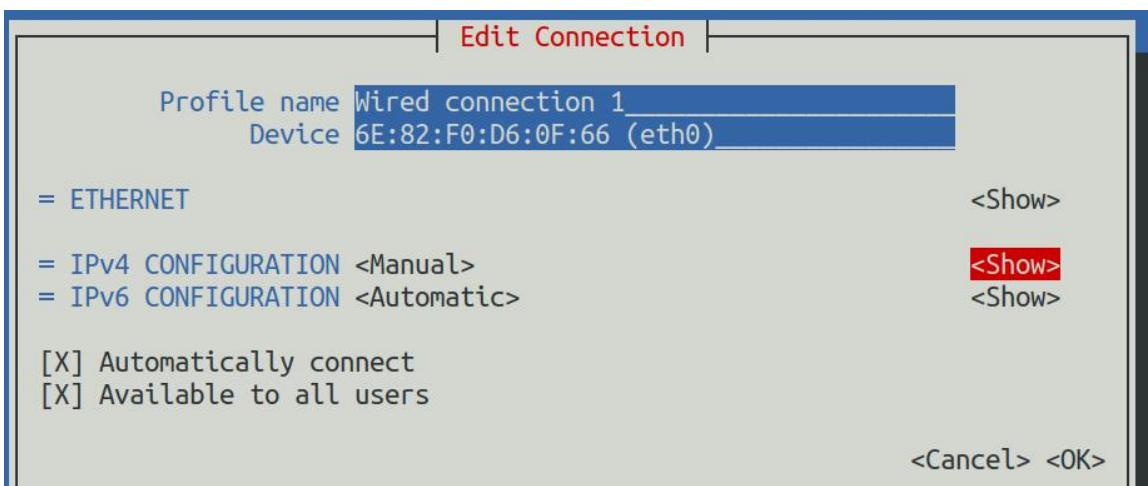
6) 然后回车，通过上下方向键选择 **Manual**，然后回车确定



7) 选择完后的显示如下图所示

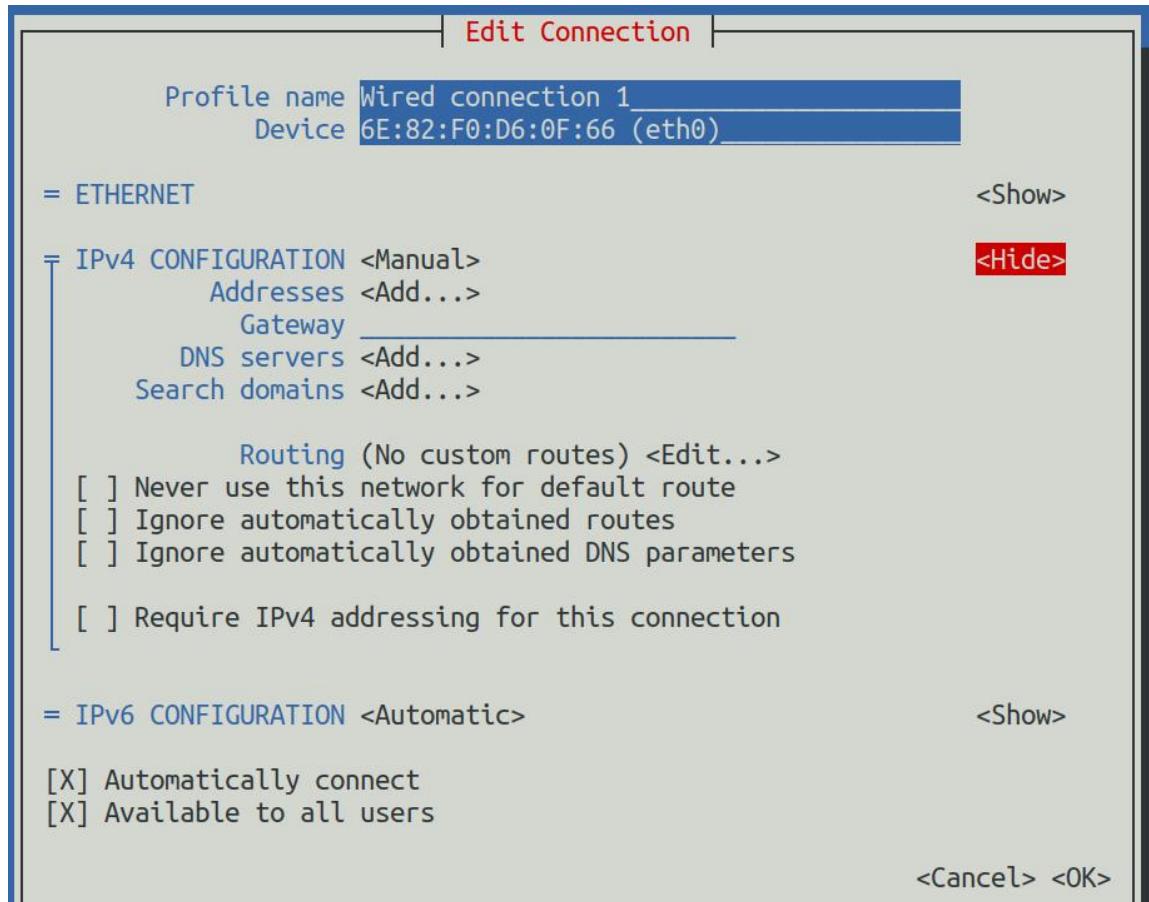


8) 然后通过 Tab 键将光标移动到<Show>

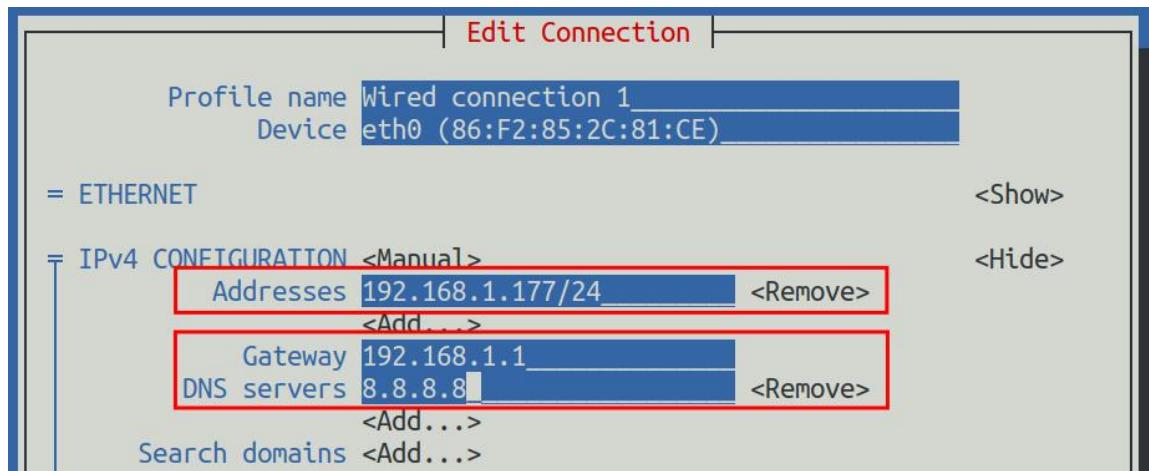




9) 然后回车，回车后会弹出下面的设置界面

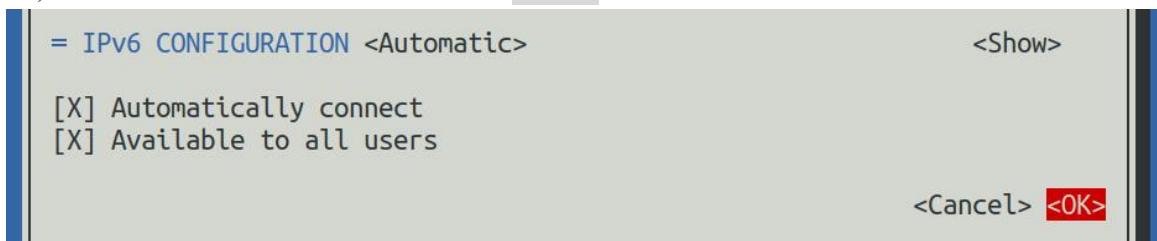


10) 然后就可以在下图所示的位置设置 IP 地址(Addresses)、网关(Gateway)和 DNS 服务器的地址（里面还有很多其他设置选项，请自行探索），**请根据自己的具体需求来设置，下图中设置的值只是一个示例**

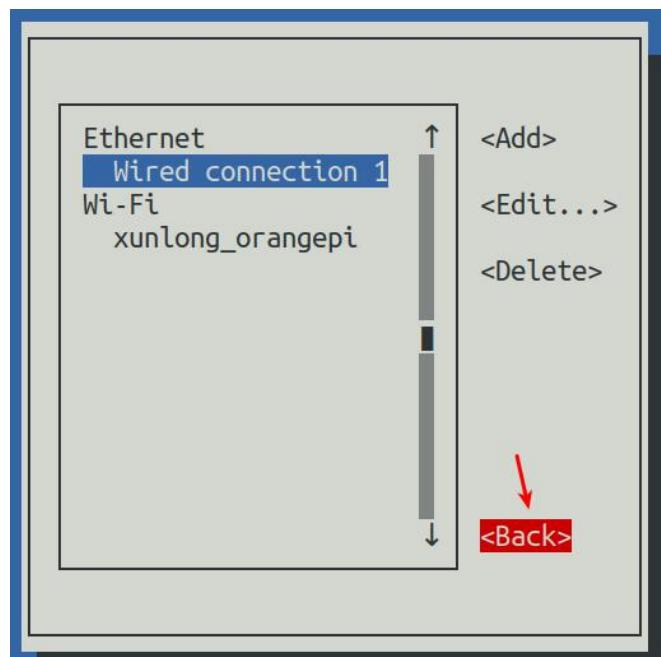




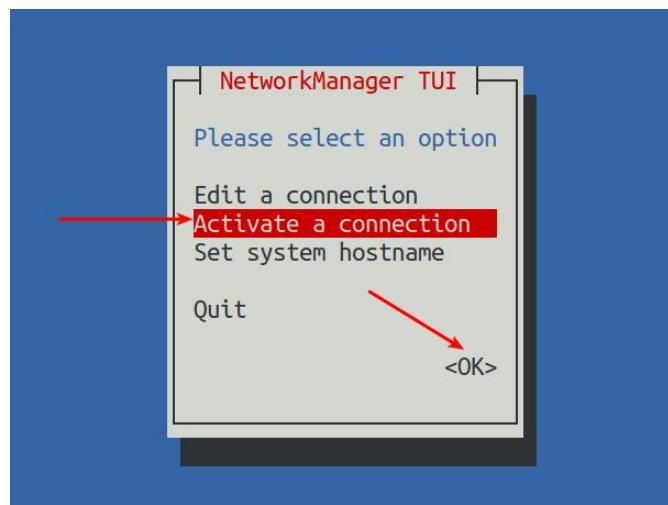
11) 设置完后将光标移动到右下角的<OK>, 然后回车确认



12) 然后点击<Back>回退到上一级选择界面

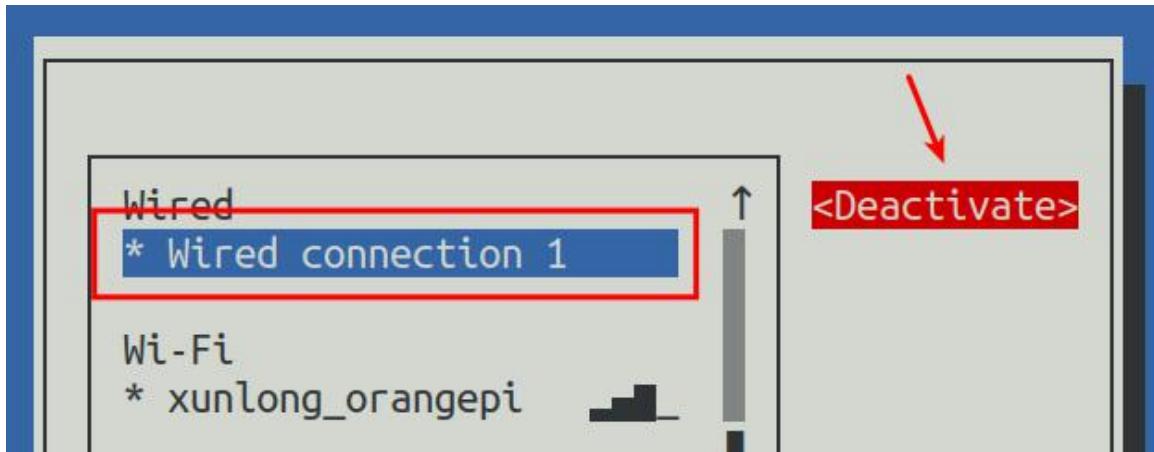


13) 然后选择 **Activate a connection**, 再将光标移动到<OK>, 最后点击回车

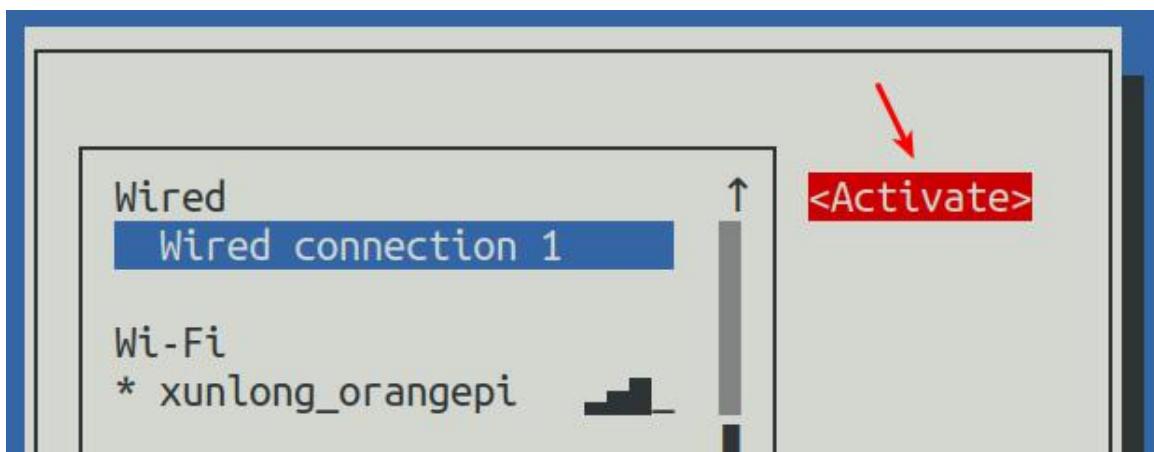




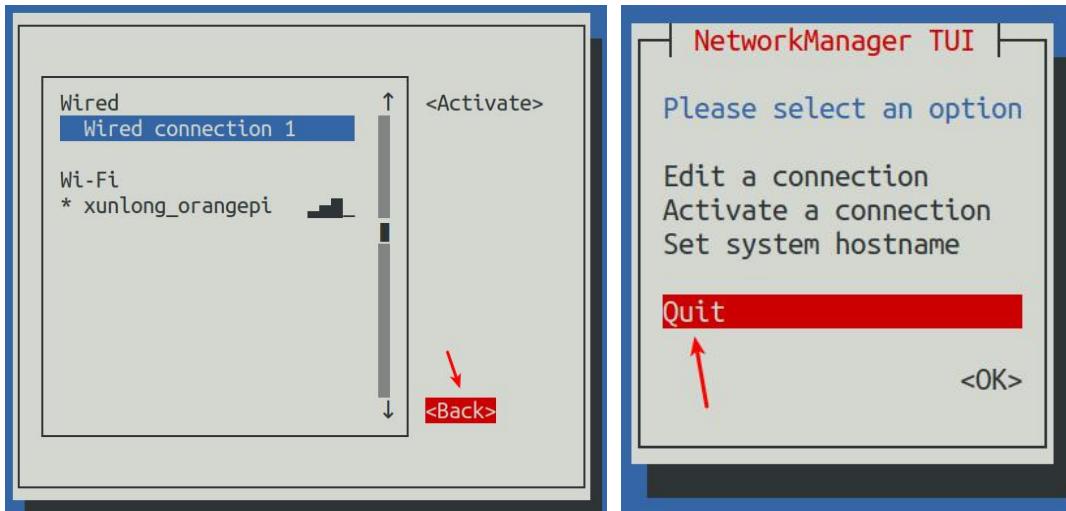
14) 然后选择需要设置的网络接口，比如 **Wired connection 1**，然后将光标移动到 **<Deactivate>**，再按下回车键禁用 **Wired connection 1**



15) 然后请不要移动光标，再按下回车键重新使能 **Wired connection 1**，这样前面设置的静态 IP 地址就会生效了



16) 然后通过**<Back>**和**Quit**按钮就可以退出 nmtui



17) 然后通过 **ip a s eth0** 就能看到网口的 IP 地址已经变成前面设置的静态 IP 地址了

注意，下面的命令中，Debian12 需要修改 eth0 为 end0。

```
orangeipi@orangepi:~$ ip a s eth0
3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 5e:ac:14:a5:92:b3 brd ff:ff:ff:ff:ff:ff
        inet 192.168.1.177/24 brd 192.168.1.255 scope global noprefixroute eth0
            valid_lft forever preferred_lft forever
        inet6 241e:3b8:3240:c3a0:e269:8305:dc08:135e/64 scope global dynamic
noprefixroute
            valid_lft 259149sec preferred_lft 172749sec
        inet6 fe80::957d:bbbe:4928:3604/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
```

18) 然后就可以测试网络的连通性来检查 IP 地址是否配置 OK 了，**ping** 命令可以通过 **Ctrl+C** 快捷键来中断运行

注意，下面的命令中，Debian12 需要修改 eth0 为 end0。

```
orangeipi@orangepi:~$ ping 192.168.1.177 -I eth0
PING 192.168.1.47 (192.168.1.47) from 192.168.1.188 eth0: 56(84) bytes of data.
64 bytes from 192.168.1.47: icmp_seq=1 ttl=64 time=0.233 ms
64 bytes from 192.168.1.47: icmp_seq=2 ttl=64 time=0.263 ms
64 bytes from 192.168.1.47: icmp_seq=3 ttl=64 time=0.273 ms
64 bytes from 192.168.1.47: icmp_seq=4 ttl=64 time=0.269 ms
```



```
64 bytes from 192.168.1.47: icmp_seq=5 ttl=64 time=0.275 ms
^C
--- 192.168.1.47 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4042ms
rtt min/avg/max/mdev = 0.233/0.262/0.275/0.015 ms
```

3. 9. 4. 2. 使用 nmcli 命令来设置静态 IP 地址

1) 如果要设置网口的静态 IP 地址, 请先将网线插入开发板, **如果需要设置 WIFI 的静态 IP 地址, 请先连接好 WIFI**, 然后再开始设置静态 IP 地址

2) 然后通过 **nmcli con show** 命令可以查看网络设备的名字, 如下所示

- a. **orangeipi** 为 WIFI 网络接口的名字 (名字不一定相同)
- b. **Wired connection 1** 为以太网接口的名字

```
orangeipi@orangeipi:~$ nmcli con show
NAME           UUID                                  TYPE      DEVICE
orangeipi      cfc4f922-ae48-46f1-84e1-2f19e9ec5e2a    wifi      wlan0
Wired connection 1  9db058b7-7701-37b8-9411-efc2ae8bfa30  ethernet   eth0
```

3) 然后输入下面的命令, 其中

- a. "**Wired connection 1**" 表示设置以太网口的静态 IP 地址, 如果需要设置 WIFI 的静态 IP 地址, 请修改为 WIFI 网络接口对应的名字 (通过 **nmcli con show** 命令可以获取到)
- b. **ipv4.addresses** 后面是要设置的静态 IP 地址, 可以修改为自己想要设置的值
- c. **ipv4.gateway** 表示网关的地址

```
orangeipi@orangeipi:~$ sudo nmcli con mod "Wired connection 1" \
ipv4.addresses "192.168.1.110" \
ipv4.gateway "192.168.1.1" \
ipv4.dns "8.8.8.8" \
ipv4.method "manual"
```

4) 然后重启 linux 系统

```
orangeipi@orangeipi:~$ sudo reboot
```

5) 然后重新进入 linux 系统使用 **ip addr show eth0** 命令就可以看到 IP 地址已经设置



为想要的值了

```
orangepi@orangepi:~$ ip addr show eth0
3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 5e:ae:14:a5:91:b3 brd ff:ff:ff:ff:ff:ff
        inet 192.168.1.110/32 brd 192.168.1.110 scope global noprefixroute eth0
            valid_lft forever preferred_lft forever
        inet6 240e:3b7:3240:c3a0:97de:1d01:b290:fe3a/64 scope global dynamic noprefixroute
            valid_lft 259183sec preferred_lft 172783sec
        inet6 fe80::3312:861a:a589:d3c/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
```

3.9.5. 设置 Linux 系统第一次启动自动连接网络的方法

开发板有以太网口，如果想通过以太网口来远程登录开发板的 Linux 系统，只需要给以太网口插上能正常上网的网线，在启动完 Linux 系统后会自动通过 DHCP 给以太网口分配一个 IP 地址，然后我们通过 HDMI 屏幕、串口或者查看路由器后台的方式就可以获取以太网口的 IP 地址，然后就能远程登录 Linux 系统。

开发板也有无线 WIFI，如果想通过 WIFI 来远程登录开发板的 Linux 系统，则需要通过以太网口的 IP 地址 ssh 远程登录 Linux 系统后通过命令来连接 WIFI，或者在 HDMI 屏幕或串口中通过命令来连接 WIFI。

但如果 HDMI 屏幕和串口模块都没有，虽然有网线，但无法通过路由器后台查看到开发板的 IP 地址。或者 HDMI 屏幕、串口模块和网线都没有，只有 WIFI 可以连接，则可以使用此小节介绍的方法来自动连接 WIFI 并且还能设置 WIFI 的静态 IP 地址或者自动设置以太网口的静态 IP 地址。

要使用此小节的方法，首先需要准备一台 Linux 系统的机器。比如一台安装有 Ubuntu 系统的电脑或者虚拟机。

为什么需要 Linux 系统的机器，因为 TF 卡中烧录的开发板 Linux 系统的根文件系统是 ext4 格式的，Linux 系统的机器可以正常的挂载它，然后对其中的配置文件进行修改。

如果想要在 Windows 系统中来修改，可以使用 **Paragon ExtFS for Windows** 这款软件，由于此软件需要付费，而目前又没有比较好用的类似的免费软件，这里就不具体演示了。



另外如果尝试 **Paragon ExtFS for Windows** 这款软件使用有问题请自行解决，
我们不答疑。

- 1) 首先烧录想使用的开发板的 Linux 镜像到 TF 卡中，然后使用读卡器，将烧录好开发板 Linux 镜像的 TF 卡插入安装有 Linux 系统的机器中（比如安装有 Ubuntu 系统的电脑，下面都以 Ubuntu 电脑为例来演示）
- 2) 当 TF 卡插入 Ubuntu 电脑后，Ubuntu 电脑一般会自动挂载 TF 卡中的 Linux 根文件系统的分区，由下面的命令可以知道，**/media/test/opi_root** 即为 TF 卡中的 Linux 根文件系统挂载的路径

```
test@test:~$ df -h | grep "media"
/dev/sdd1  1.4G  1.2G  167M  88% /media/test/opi_root
test@test:~$ ls /media/test/opi_root
bin  boot  dev  etc  home  lib  lost+found  media  mnt  opt  proc  root  run
sbin  selinux  srv  sys  tmp  usr  var
```

- 3) 然后进入 TF 卡中烧录的 Linux 系统的**/boot** 目录中

```
test@test:~$ cd /media/test/opi_root/boot/
```

- 4) 然后将其中的 **orangeipi_first_run.txt.template** 复制为 **orangeipi_first_run.txt**，通过 orangeipi_first_run.txt 配置文件可以设置开发板 Linux 系统第一次启动时自动连接某个 WIFI 热点，也可以设置 WIFI 或者以太网口的静态 IP 地址

```
test@test:/media/test/opi_root/boot$ sudo cp orangeipi_first_run.txt.template orangeipi_first_run.txt
```

- 5) 通过下面的命令可以打开 orangeipi_first_run.txt 文件，然后就可以查看修改其中的内容

```
test@test:/media/test/opi_root/boot$ sudo vim orangeipi_first_run.txt
```

- 6) orangeipi_first_run.txt 文件中的变量使用说明

- a. **FR_general_delete_this_file_after_completion** 变量用来设置第一次启动完后是否删除 orangeipi_first_run.txt 这个文件，默认为 1，也就是删除，如果设置为 0，第一次启动后会将 orangeipi_first_run.txt 重命名为 orangeipi_first_run.txt.old，一般保持默认值即可
- b. **FR_net_change_defaults** 变量用于设置是否改变默认网络设置，这个必须要设置为 1，否则所有的网络设置都不会生效



- c. **FR_net_etherent_enabled** 变量用来控制是否使能以太网口的配置，如果需要设置以太网口的静态 IP 地址，请设置为 1
- d. **FR_net_wifi_enabled** 变量用来控制是否使能 WIFI 的配置，如果需要设置开发板自动连接 WIFI 热点，则必须将其设置为 1，另外请注意，如果此变量设置为 1，则以太网口的设置就会失效。也就是说 WIFI 和以太网口不能同时设置（为什么，因为没必要...）
- e. **FR_net_wifi_ssid** 变量用于设置想要连接的 WIFI 热点的名字
- f. **FR_net_wifi_key** 变量用于设置想要连接的 WIFI 热点的密码
- g. **FR_net_use_static** 变量用于设置是否需要设置 WIFI 或者以太网口的静态 IP 地址
- h. **FR_net_static_ip** 变量用于设置静态 IP 的地址，请根据自己的实际情况设置
- i. **FR_net_static_gateway** 变量用于设置网关，请根据自己的实际情况设置

7) 下面演示几个具体的设置示例：

- a. 比如想要开发板的 Linux 系统第一次启动后自动连接 WIFI 热点，可以这样设置：
 - a) 设置 **FR_net_change_defaults** 为 1
 - b) 设置 **FR_net_wifi_enabled** 为 1
 - c) 设置 **FR_net_wifi_ssid** 为想要连接的 WIFI 热点的名字
 - d) 设置 **FR_net_wifi_key** 为想要连接的 WIFI 热点的密码
- b. 比如想要开发板的 Linux 系统第一次启动后自动连接 WIFI 热点，并且设置 WIFI 的 IP 地址为特定的静态 IP 地址（这样当 Linux 系统启动后，可以直接使用设置的静态 IP 地址 ssh 远程登录开发板，无需通过路由器后台来查看开发板的 IP 地址），可以这样设置：
 - a) 设置 **FR_net_change_defaults** 为 1
 - b) 设置 **FR_net_wifi_enabled** 为 1
 - c) 设置 **FR_net_wifi_ssid** 为想要连接的 WIFI 热点的名字
 - d) 设置 **FR_net_wifi_key** 为想要连接的 WIFI 热点的密码
 - e) 设置 **FR_net_use_static** 为 1
 - f) 设置 **FR_net_static_ip** 为想要的 IP 地址
 - g) 设置 **FR_net_static_gateway** 为对应的网关地址
- c. 比如想要开发板的 Linux 系统第一次启动后自动设置以太网口的 IP 地址为想要的静态 IP 地址，可以这样设置：
 - a) 设置 **FR_net_change_defaults** 为 1



- b) 设置 **FR_net_ethernet_enabled** 为 1
- c) 设置 **FR_net_use_static** 为 1
- d) 设置 **FR_net_static_ip** 为想要的 IP 地址
- e) 设置 **FR_net_static_gateway** 为对应的网关地址

8) 修改完 `orangeipi_first_run.txt` 文件后, 就可以退出 TF 卡中开发板 Linux 系统的/`boot` 目录, 再卸载 TF 卡, 然后就可以将 TF 卡插入开发板中启动了

9) 如果没有设置静态 IP 地址, 则还是需要通过路由器后台来查看 IP 地址, 如果设置了静态 IP 地址, 则可以在电脑上 ping 下设置的静态 IP 地址, 如果能 ping 说明系统已经正常启动, 并且网络也已设置正确, 然后就可以使用设置的 IP 地址 ssh 远程登录开发板的 Linux 系统了

开发板的 Linux 系统第一次启动完后, `orangeipi_first_run.txt` 会被删除或者重命名为 `orangeipi_first_run.txt.old`, 此时就算重新设置 `orangeipi_first_run.txt` 配置文件, 然后重新启动开发板的 Linux 系统, `orangeipi_first_run.txt` 中的配置也不会再次生效, 因为此配置只在烧录完 Linux 系统后第一次启动才会有作用, 这点请特别注意。

3.10. SSH 远程登录开发板

Linux 系统默认都开启了 ssh 远程登录, 并且允许 root 用户登录系统。ssh 登录前首先需要确保以太网或者 wifi 网络已连接, 然后使用 `ip addr` 命令或者通过查看路由器的方式获取开发板的 IP 地址。

3.10.1. Ubuntu 下 SSH 远程登录开发板

1) 获取开发板的 IP 地址

2) 然后就可以通过 ssh 命令远程登录 linux 系统

```
test@test:~$ ssh orangeipi@192.168.1.xxx      (需要替换为开发板的 IP 地址)
orangeipi@192.168.1.xx's password:    (在这里输入密码, 默认密码为 orangeipi)
```

注意, 输入密码的时候, 屏幕上是不会显示输入的密码的具体内容的, 请不要以为是有什么故障, 输入完后直接回车即可。



如果提示拒绝连接，只要使用的是 Orange Pi 提供的镜像，就请不要怀疑 orangepi 这个密码是不是不对，而是要找其他原因。

3) 成功登录系统后的显示如下图所示

```
test@test:~$ ssh orangepi@192.168.1.121
orangepi@192.168.1.121's password:
[REDACTED]
Welcome to Orange Pi 1.0.0 Bullseye with Linux 6.1.31-sun50iw9

System load: 39%          Up time:      21 min  Local users: 4
Memory usage: 31% of 1.45G   IP:          192.168.1.121
CPU temp:      54°C         Usage of /:    25% of 15G

Last login: Thu Jun  8 08:03:08 2023 from 192.168.1.119
orangepi@orangepi:~$
```

如果 ssh 无法正常登陆 linux 系统，首先请检查下开发板的 IP 地址是否能 ping 通，如果 ping 通没问题的话，可以通过串口或者 HDMI 显示器登录 linux 系统然后在开发板上输入下面的命令后再尝试是否能连接：

root@orangepi:~# reset ssh.sh

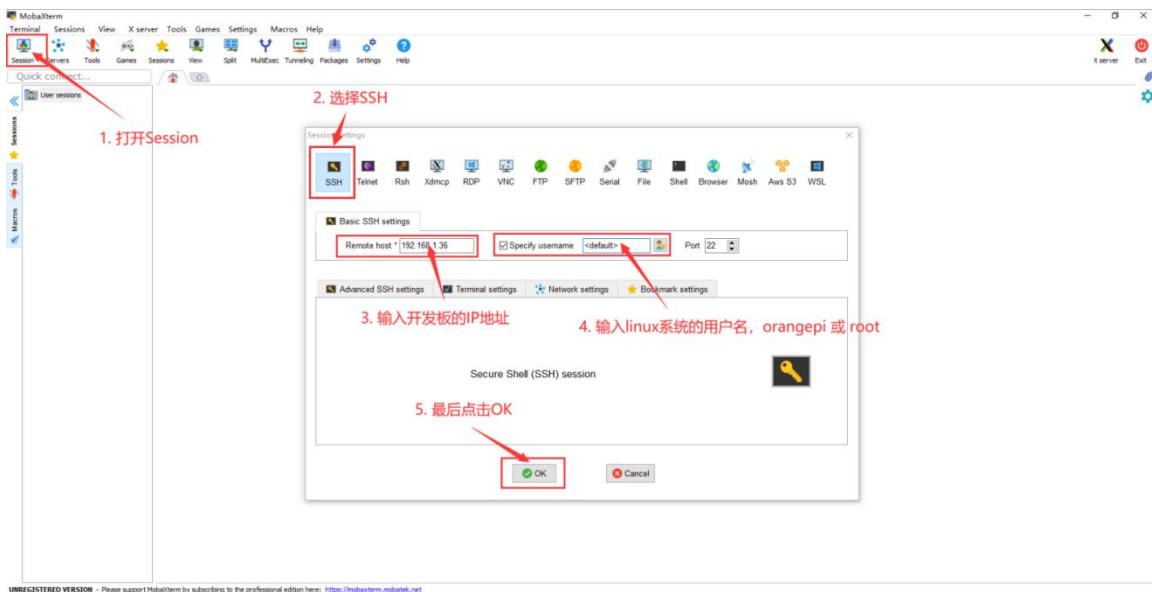
如果还不行，请重烧系统试下。

3. 10. 2. Windows 下 SSH 远程登录开发板

1) 首先获取开发板的 IP 地址

2) 在 windows 下可以使用 MobaXterm 远程登录开发板，首先新建一个 ssh 会话

- a. 打开 **Session**
 - b. 然后在 **Session Setting** 中选择 **SSH**
 - c. 然后在 **Remote host** 中输入开发板的 IP 地址
 - d. 然后在 **Specify username** 中输入 linux 系统的用户名 **root** 或 **orangepi**
 - e. 最后点击 **OK** 即可

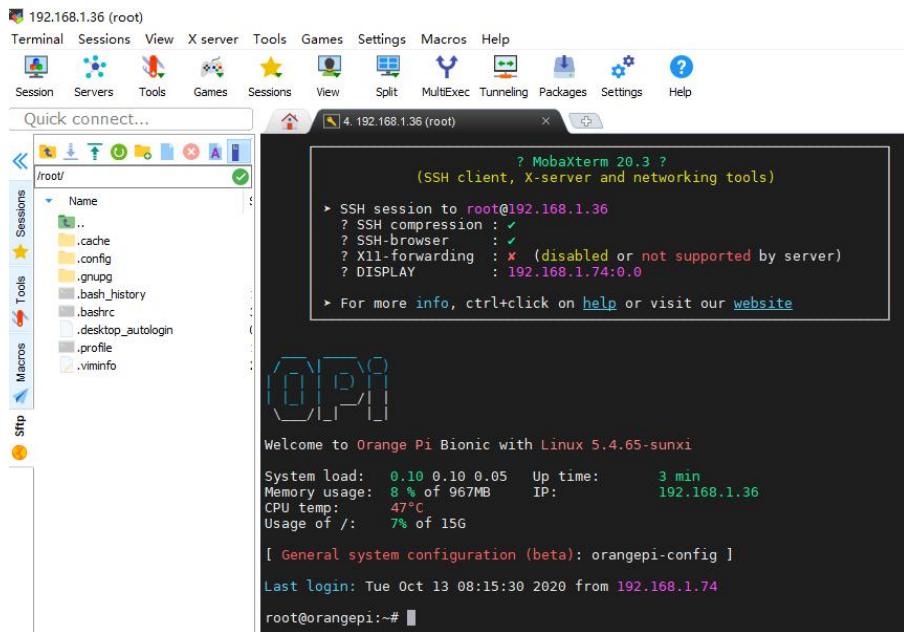


3) 然后会提示输入密码， 默认 root 和 orangepi 用户的密码都为 orangepi

注意，输入密码的时候，屏幕上是不会显示输入的密码的具体内容的，请不要以为是有什么故障，输入完后直接回车即可。



4) 成功登录系统后的显示如下图所示



3.11. HDMI 测试

3.11.1. HDMI 显示测试

- 1) 使用 Mini HDMI 转 HDMI 线连接 Orange Pi 开发板和 HDMI 显示器



- 2) 启动 linux 系统后如果 HDMI 显示器有图像输出说明 HDMI 接口使用正常

注意，很多笔记本电脑虽然带有 HDMI 接口，但是笔记本的 HDMI 接口一般只有输出功能，并没有 HDMI in 的功能，也就是说并不能将其他设备的 HDMI 输出显示到笔记本的屏幕上。

当想把开发板的 HDMI 接到笔记本电脑 HDMI 接口时，请先确认清楚您的笔记本是支持 HDMI in 的功能。



当 HDMI 没有显示的时候，请先检查下 HDMI 线有没有插紧，确认接线没问题后，可以换一个不同的屏幕试下有没有显示。

3.11.2. HDMI 转 VGA 显示测试

1) 首先需要准备下面的配件

- a. HDMI 转 VGA 转换器



- b. 一根 VGA 线和一个 Mini HDMI 公转 HDMI 母的转接头



- c. 一个支持 VGA 接口的显示器或者电视

2) HDMI 转 VGA 显示测试如下所示





使用 HDMI 转 VGA 显示时，开发板以及开发板的 Linux 系统是不需要做任何设置的，只需要开发板 Mini HDMI 接口能正常显示就可以了。所以如果测试有问题，请检查 HDMI 转 VGA 转换器、VGA 线以及显示器是否有问题。

3.11.3. Linux5.4 系统 HDMI 分辨率设置的方法

注意：此方法只适用于 linux5.4 内核的系统。

1) 在 linux 系统的 /boot/orangepiEnv.txt 中有个 disp_mode 变量，可以通过它来设置 HDMI 输出的分辨率，linux 系统默认设置的分辨率为 1080p60

```
orangepi@orangepi:~$ sudo vim /boot/orangepiEnv.txt
verbosity=1
console=both
disp_mode=1080p60
fb0_width=1920
fb0_height=1080
```

2) disp_mode 变量支持设置的值如下表所示

disp_mode 支持的值	HDMI 分辨率	HDMI 刷新率
480i	720x480	60
576i	720x480	50
480p	720x480	60
576p	720x576	60
720p50	1280x720	50
720p60	1280x720	60
1080i50	1920x1080	50
1080i60	1920x1080	60
1080p24	1920x1080	24
1080p50	1920x1080	50
1080p60	1920x1080	60

注意：Linux 系统目前不支持 4K 分辨率。

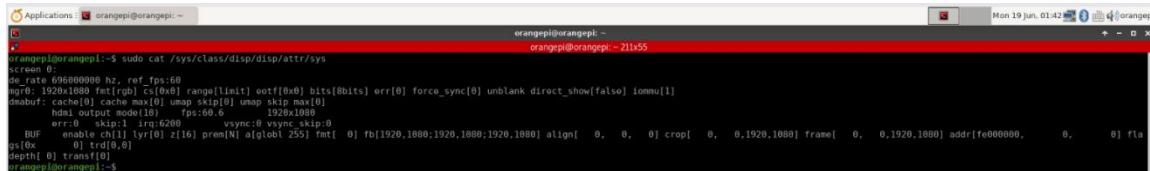
3) 将 disp_mode 变量的值修改为想要输出的分辨率，然后重启系统，HDMI 就会输



出所设置的分辨率了

4) 查看 HDMI 输出分辨率的方法如下所示，如果显示的分辨率和设置的分辨率一样，说明开发板这端的设置正确

```
orangepi@orangepi:~$ sudo cat /sys/class/disp/disp/attr/sys
```



```
orangepi@orangepi:~$ sudo cat /sys/class/disp/disp/attr/sys
screen 0
freq 60000000 hz, ref_fps:60
negr: 1920x1080 fenc[rgb] c[0x0] range[limit] outf[0x0] bits[8bits] err[0] force_sync[0] unblank direct_show[false] ionmu[1]
dmabuf: cache[0] cache_max[0] umap skip[0] umap skip_max[0]
        hdmi output mode[18]   fps:60.6   1920x1080
        BIF enable ch[1] lyr[0] z[16] prem[N] alg[obl 255] fmt[ 0] fb[1920,1080:1920,1080;1920,1080] align[ 0, 0, 0] crop[ 0, 0, 0,1920,1080] frame[ 0, 0, 0,1920,1080] addr[fe000000, 0, 0] fla
        depth[ 0] trans[0]
        gsc[ 0] trd[0,0]
        depth[ 0] trans[0]
```

3.11.4. Linux5.4 系统 Framebuffer 宽度和高度的修改方法

注意：此方法只适用于 linux5.4 内核的系统。

在 linux 系统的/**/boot/orangepiEnv.txt** 中有 **fb0_width** 和 **fb0_height** 两个变量，可以通过它们来设置 Framebuffer 的宽度和高度，linux 系统默认设置 **fb0_width=1920**、**fb0_height=1080**。

```
orangepi@orangepi:~$ sudo vim /boot/orangepiEnv.txt
```

```
verbosity=1
console=both
disp_mode=1080p60
fb0_width=1920
fb0_height=1080
```

fb0_width 和 **fb0_height** 不同分辨率对应的参考值如下所示：

HDMI 分辨率	fb0_width	fb0_height
480p	720	480
576p	720	576
720p	1280	720
1080p	1920	1080

在相同的 HDMI 分辨下，当 **fb0_width** 和 **fb0_height** 设置的值越大时，屏幕显示的文字就越小，当 **fb0_width** 和 **fb0_height** 设置的值越小时，屏幕显示的文字就越大。

3.11.5. Framebuffer 光标设置

1) Framebuffer 使用的 softcursor，设置光标闪烁或者不闪烁的方法如下所示



```
root@orangepi:~# echo 1 > /sys/class/graphics/fbcon/cursor_blink #光标闪烁  
root@orangepi:~# echo 0 > /sys/class/graphics/fbcon/cursor_blink #光标不闪烁
```

2) 如果需要隐藏光标，可以在`/boot/orangepiEnv.txt`的`extraargs`变量（`extraargs`的值会赋值给`bootargs`环境变量最终传递给内核）中加入`vt.global_cursor_default=0`（如果`vt.global_cursor_default=1`则是显示光标），然后重启系统就能看到光标已经消失

```
orangepi@orangepi:~$ sudo vim /boot/orangepiEnv.txt  
verbosity=1  
console=both  
disp_mode=1080p60  
fb0_width=1920  
fb0_height=1080  
extraargs=vt.global_cursor_default=0
```

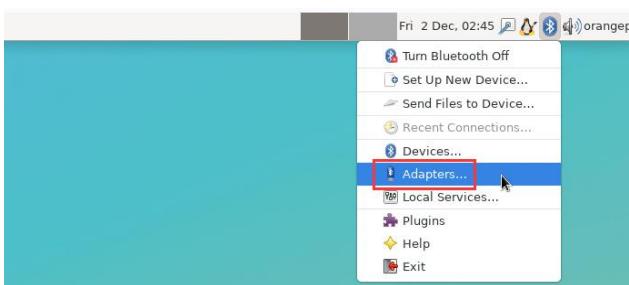
3. 12. 蓝牙使用方法

3. 12. 1. 桌面版镜像的测试方法

1) 点击桌面右上角的蓝牙图标

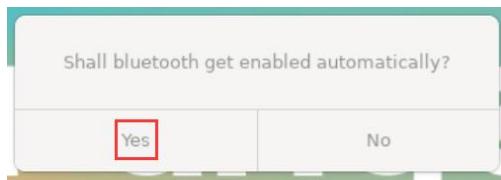


2) 然后选择适配器

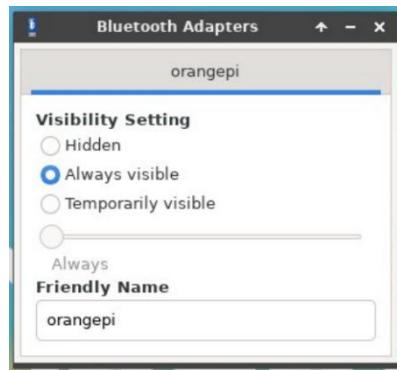




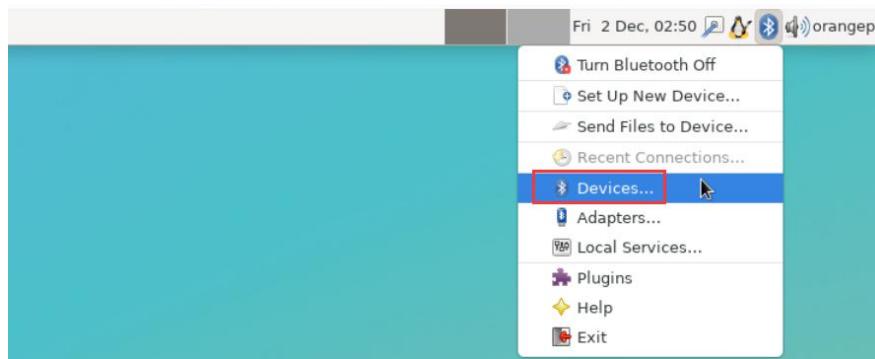
3) 如果有提示下面的界面, 请选择 **Yes**



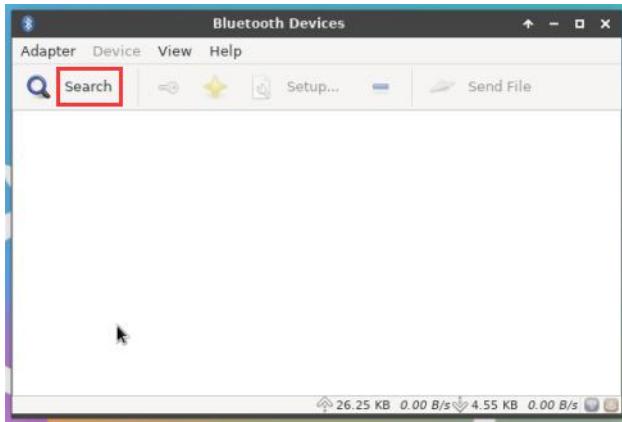
4) 然后在蓝牙的适配器设置界面中设置 **Visibility Setting** 为 **Always visible**, 然后关闭即可



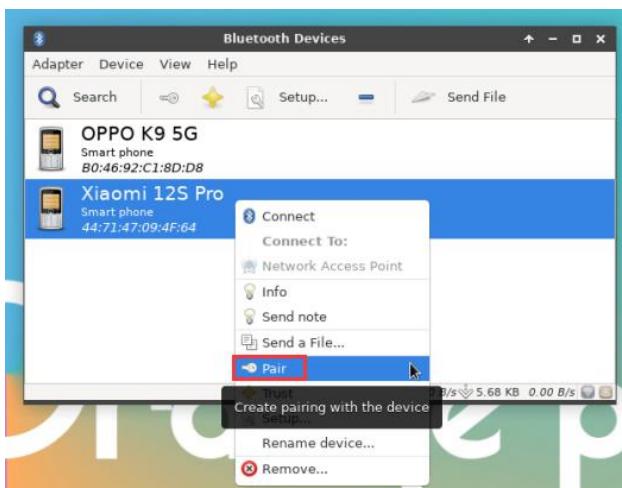
5) 然后打开蓝牙设备的配置界面



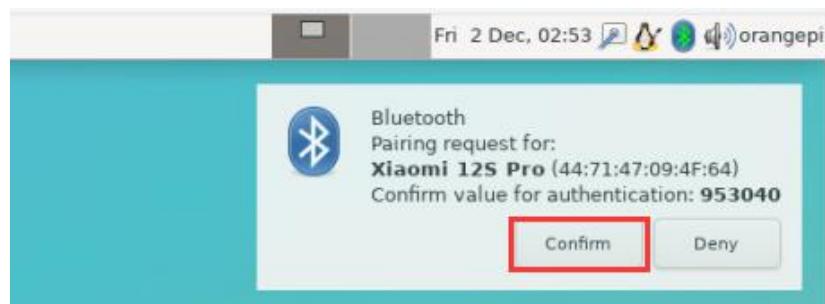
6) 点击 **Search** 即可开始扫描周围的蓝牙设备



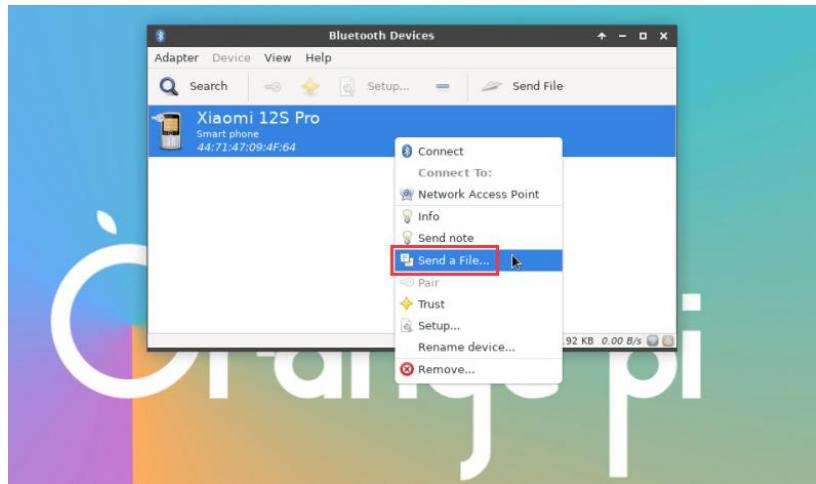
7) 然后选择想要连接的蓝牙设备，再点击鼠标右键就会弹出对此蓝牙设备的操作界面，选择 **Pair** 即可开始配对，这里演示的是和 Android 手机配对



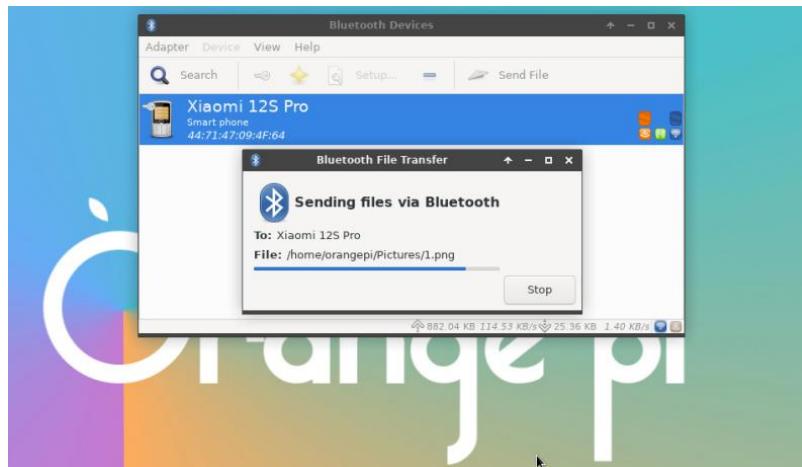
8) 配对时，桌面的右上角会弹出配对确认框，选择 **Confirm** 确认即可，此时手机上也同样需要进行确认



9) 和手机配对完后，可以选择已配对的蓝牙设备，然后右键选择 **Send a File** 即可开始给手机发送一张图片



10) 发送图片的界面如下所示



3. 12. 2. 服务器版镜像的使用方法

1) 进入系统后首先可以通过 **hciconfig** 命令来查看是否存在蓝牙的设备节点，如果存在，说明蓝牙初始化正常

```
orangepi@orangepi:~$ sudo apt update && sudo apt install -y bluez
orangepi@orangepi:~$ hciconfig -a
hci0:  Type: Primary  Bus: UART
        BD Address: 3E:61:3D:19:0E:52  ACL MTU: 1021:8  SCO MTU: 240:3
        UP RUNNING
        RX bytes:925 acl:0 sco:0 events:72 errors:0
        TX bytes:5498 acl:0 sco:0 commands:72 errors:0
        Features: 0xbff 0xff 0x8d 0xfe 0xdb 0x3d 0x7b 0xc7
        Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
        Link policy: RSWITCH SNIFF
```



```
Link mode: SLAVE ACCEPT
Name: 'orangeipi'
Class: 0x3c0000
Service Classes: Rendering, Capturing, Object Transfer, Audio
Device Class: Miscellaneous,
HCI Version: 5.0 (0x9) Revision: 0x400
LMP Version: 5.0 (0x9) Subversion: 0x400
Manufacturer: Spreadtrum Communications Shanghai Ltd (492)
```

2) 使用 **bluetoothctl** 扫描蓝牙设备

```
orangeipi@orangeipi:~$ sudo bluetoothctl
[NEW] Controller 10:11:12:13:14:15 orangepirzero2w [default]
Agent registered
[bluetooth]# power on      #使能控制器
Changing power on succeeded
[bluetooth]# discoverable on    #设置控制器为可被发现的
Changing discoverable on succeeded
[CHG] Controller 10:11:12:13:14:15 Discoverable: yes
[bluetooth]# pairable on     #设置控制器为可配对的
Changing pairable on succeeded
[bluetooth]# scan on       #开始扫描周围的蓝牙设备
Discovery started
[CHG] Controller 10:11:12:13:14:15 Discovering: yes
[NEW] Device 76:60:79:29:B9:31 76-60-79-29-B9-31
[NEW] Device 9C:2E:A1:42:71:11 小米手机
[NEW] Device DC:72:9B:4C:F4:CF orangeipi
[bluetooth]# scan off      #扫描到想连接的蓝牙设备后就可以关闭扫描了，然后记
下蓝牙设备的 MAC 地址，这里测试的蓝牙设备为 Android 手机，蓝牙的名字为
orangeipi，对应的 MAC 地址为 DC:72:9B:4C:F4:CF
Discovery stopped
[CHG] Controller 10:11:12:13:14:15 Discovering: no
[CHG] Device DC:72:9B:4C:F4:CF RSSI is nil
```

3) 扫描到想配对的设备后就可以进行配对了，配对需要使用设备的 MAC 地址

```
[bluetooth]# pair DC:72:9B:4C:F4:CF    #使用扫描到的蓝牙设备的 MAC 地址进
```



行配对

```
Attempting to pair with DC:72:9B:4C:F4:CF
[CHG] Device DC:72:9B:4C:F4:CF Connected: yes
Request confirmation
[leeb1m[agent] Confirm passkey 764475 (yes/no): yes #在这里输入 yes, 在手机上也需要确认
[CHG] Device DC:72:9B:4C:F4:CF Modalias: bluetooth:v010Fp107Ed1436
[CHG] Device DC:72:9B:4C:F4:CF UUIDs: 0000046a-0000-1000-8000-00805f9b34fb
[CHG] Device DC:72:9B:4C:F4:CF ServicesResolved: yes
[CHG] Device DC:72:9B:4C:F4:CF Paired: yes
Pairing successful #提示配对成功
[CHG] Device DC:72:9B:4C:F4:CF ServicesResolved: no
[CHG] Device DC:72:9B:4C:F4:CF Connected: no
```

4) 配对成功后，手机蓝牙界面的显示如下所示



5) 连接蓝牙设备需要安装 **pulseaudio-module-bluetooth** 软件包，然后再启动 **pulseaudio** 服务

```
orangeipi@orangeipi:~$ sudo apt update
orangeipi@orangeipi:~$ sudo apt -y install pulseaudio-module-bluetooth
orangeipi@orangeipi:~$ pulseaudio --start
```

6) 连接蓝牙设备的方法

```
orangeipi@orangeipi:~$ sudo bluetoothctl
Agent registered
[bluetooth]# paired-devices      #查看已配对的蓝牙设备的 MAC 地址
Device DC:72:9B:4C:F4:CF orangeipi
[bluetooth]# connect DC:72:9B:4C:F4:CF    #使用 MAC 地址连接蓝牙设备
Attempting to connect to DC:72:9B:4C:F4:CF
```



```
[CHG] Device DC:72:9B:4C:F4:CF Connected: yes  
Connection successful  
[CHG] Device DC:72:9B:4C:F4:CF ServicesResolved: yes  
[CHG] Controller 10:11:12:13:14:15 Discoverable: no  
[orangeipi]# #出现这个提示符说明连接成功
```

7) 连接完蓝牙设备后，Android 手机的蓝牙配置界面就可以看到已连接用于通话和媒体的音频的提示

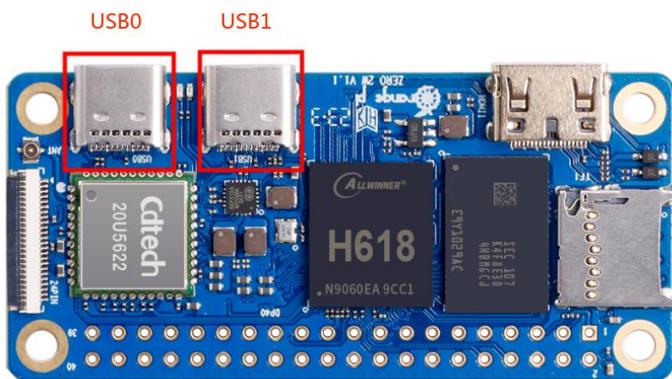


3. 13. USB 接口测试

USB 接口是可以接 USB hub 来扩展 USB 接口的数量的。

3. 13. 1. USB 接口扩展说明

如下图所示，开发板的主板上只有两个 Type-C 类型的 USB2.0 接口，是无法直接接 USB Type-A 类型的鼠标、键盘等 USB 设备的。

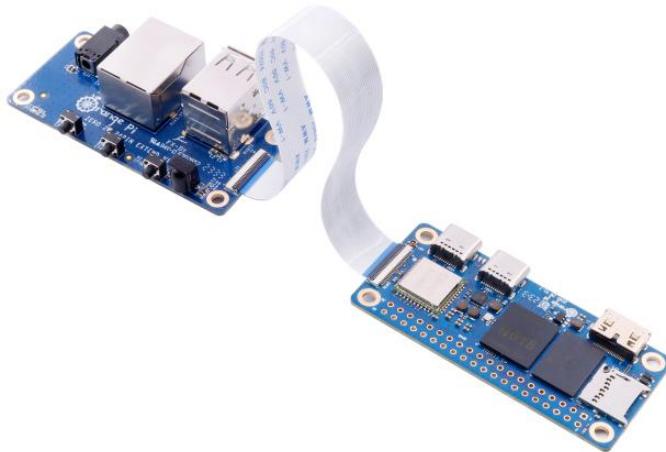




如果只购买了主板，没有购买 24pin 扩展板，可以准备一根线下图所示的 Type-C 转 USB 线，将其 Type-C 接口的一端插入主板的 Type-C 接口中，然后另一端就可以接鼠标键盘等 USB 设备了，如果觉得一个 USB 接口不够用，还可以通过 USB Hub 来扩展多个 USB 接口。

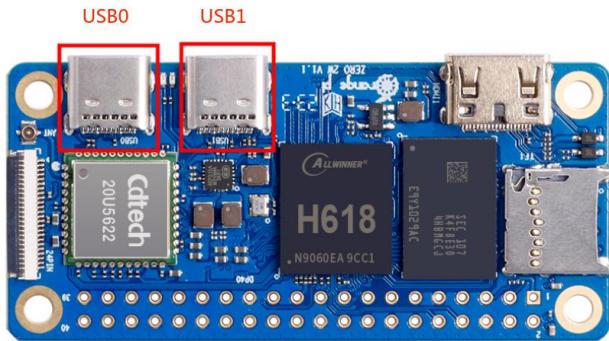


如果购买了 24pin 扩展板，就无需 Type-C 转 USB 线，因为 24pin 扩展板可以扩展出两个 USB2.0 接口。



3.13.2. USB0 设置为 HOST 模式的方法

如下图所示，开发板的主板上有两个 Type-C 类型的接口：USB0 和 USB1，这两个接口都可以用来给开发板供电，也都可以用来当做 USB2.0 HOST 接口。USB0 和 USB1 的区别是：USB0 除了可以设置为 HOST 模式外，还可以设置为 Device 模式，而 USB1 只有 HOST 模式。



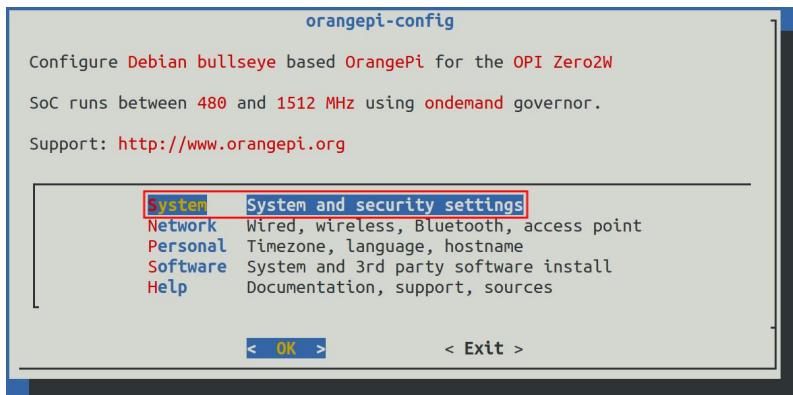
Orange Pi 发布的 Linux 系统 USB0 默认设置为 Device 模式，所以在不需要使用 USB0 Device 模式时，建议使用 USB0 来供电，这样 USB1 就可以直接用来接 USB 设备。

如果需要使用 USB0 来接 USB 设备，需要把 USB0 设置为 HOST 模式，方法如下所示：

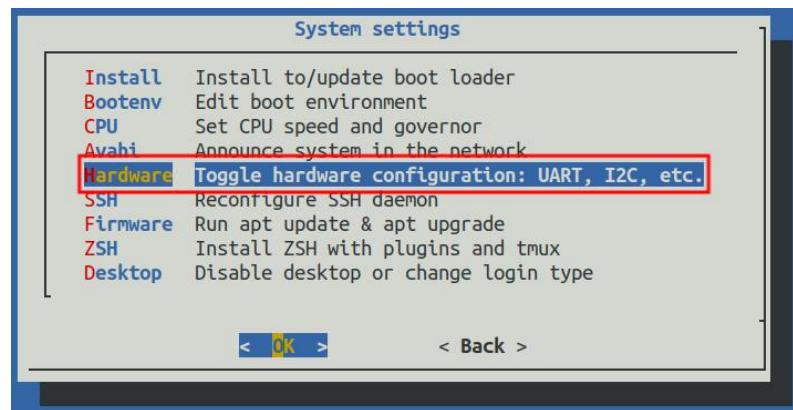
- 首先运行下 **orangepi-config**，普通用户记得加 **sudo** 权限

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

- 然后选择 **System**



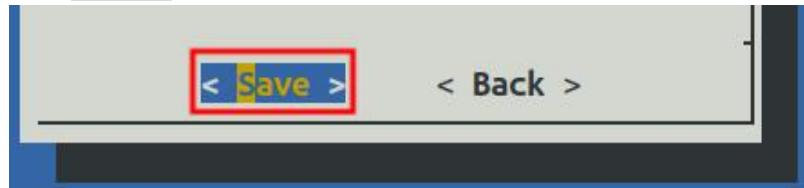
- 然后选择 **Hardware**



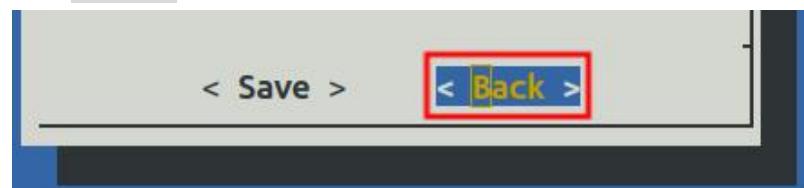
- 然后使用键盘的方向键定位到下图所示的位置，再使用空格选中 **usb0-host**



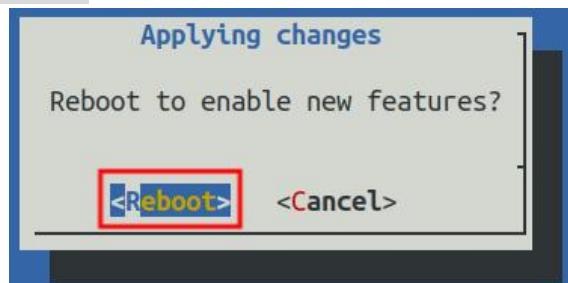
e. 然后选择<Save>保存



f. 然后选择<Back>



g. 然后选择<Reboot>重启系统使配置生效



h. 重启后 USB0 就能正常使用鼠标键盘等 USB 设备了

3.13.3. 连接 USB 鼠标或键盘测试

1) 将 USB 接口的键盘插入 Orange Pi 开发板的 USB 接口中

2) 连接 Orange Pi 开发板到 HDMI 显示器

3) 如果鼠标或键盘能正常操作说明 USB 接口使用正常（鼠标只有在桌面版的系统中才能使用）

3.13.4. 连接 USB 存储设备测试

1) 首先将 U 盘或者 USB 移动硬盘插入 Orange Pi 开发板的 USB 接口中

2) 执行下面的命令如果能看到 sdX 的输出说明 U 盘识别成功

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



8	0	30044160	sda
8	1	30043119	sda1

3) 使用 mount 命令可以将 U 盘挂载到/mnt 中，然后就能查看 U 盘中的文件了

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

4) 挂载完后通过 df -h 命令就能查看 U 盘的容量使用情况和挂载点

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

3.13.5. USB 以太网卡测试

1) 目前测试过能用的 USB 以太网卡如下所示，其中 RTL8153 USB 千兆网卡插入开发板的 USB 2.0 Host 接口中测试可以正常使用，但是速率是达不到千兆的，这点请注意

序号	型号
1	RTL8152B USB 百兆网卡
2	RTL8153 USB 千兆网卡

2) 首先将 USB 网卡插入开发板的 USB 接口中，然后在 USB 网卡中插入网线，确保网线能正常上网，如果通过 dmesg 命令可以看到下面的 log 信息，说明 USB 网卡识别正常

```
orangeipi@orangeipi:~$ dmesg | tail
[ 121.985016] usb 3-1: USB disconnect, device number 2
[ 126.873772] sunxi-ehci 5311000.ehci3-controller: ehci_irq: highspeed device connect
[ 127.094054] usb 3-1: new high-speed USB device number 3 using sunxi-ehci
[ 127.357472] usb 3-1: reset high-speed USB device number 3 using sunxi-ehci
[ 127.557960] r8152 3-1:1.0 eth1: v1.08.9
[ 127.602642] r8152 3-1:1.0 enx00e04c362017: renamed from eth1
[ 127.731874] IPv6: ADDRCONF(NETDEV_UP): enx00e04c362017: link is not ready
[ 127.763031] IPv6: ADDRCONF(NETDEV_UP): enx00e04c362017: link is not ready
[ 129.892465] r8152 3-1:1.0 enx00e04c362017: carrier on
[ 129.892583] IPv6: ADDRCONF(NETDEV_CHANGE): enx00e04c362017: link becomes ready
```



3) 然后通过 ifconfig 命令可以看到 USB 网卡的设备节点，以及自动分配的 IP 地址

```
orangeipi@orangeipi:~$ sudo ifconfig
enx00e04c362017: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>      mtu
1500
          inet 192.168.1.177  netmask 255.255.255.0 broadcast 192.168.1.255
              inet6 fe80::681f:d293:4bc5:e9fd  prefixlen 64  scopeid 0x20<link>
                  ether 00:e0:4c:36:20:17  txqueuelen 1000  (Ethernet)
                      RX packets 1849  bytes 134590 (134.5 KB)
                      RX errors 0  dropped 125  overruns 0  frame 0
                      TX packets 33  bytes 2834 (2.8 KB)
                      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

4) 测试网络连通性的命令如下

```
orangeipi@orangeipi:~$ ping www.baidu.com -I enx00e04c362017
PING www.a.shifen.com (14.215.177.38) from 192.168.1.12 eth0: 56(84) bytes of data.
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=1 ttl=56 time=6.74 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=2 ttl=56 time=6.80 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=3 ttl=56 time=6.26 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=4 ttl=56 time=7.27 ms
^C
--- www.a.shifen.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 6.260/6.770/7.275/0.373 ms
```

3. 13. 6. USB 摄像头测试

1) 首先将 USB 摄像头插入到 Orange Pi 开发板的 USB 接口中

2) 然后通过 lsmod 命令可以看到内核自动加载了下面的模块

```
orangeipi@orangeipi:~$ lsmod
Module           Size  Used by
uvcvideo        106496  0
```

3) 通过 v4l2-ctl 命令可以看到 USB 摄像头的设备节点信息为 /dev/video0

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



```
orangepi@orangepi:~$ sudo apt install -y v4l-utils  
orangepi@orangepi:~$ v4l2-ctl --list-devices  
USB 2.0 Camera (usb-sunxi-ehci-1):  
/dev/video0
```

注意 v4l2 中的 l 是小写字母 l，不是数字 1。

另外 video 的序号不一定都是 video0，请以实际看到的为准。

4) 使用 fswebcam 测试 USB 摄像头

a. 安装 fswebcam

```
orangepi@orangepi:~$ sudo apt update  
orangepi@orangepi:~$ sudo apt-get install -y fswebcam
```

b. 安装完 fswebcam 后可以使用下面的命令来拍照

- a) -d 选项用于指定 USB 摄像头的设备节点
- b) --no-banner 用于去除照片的水印
- c) -r 选项用于指定照片的分辨率
- d) -S 选项用设置于跳过前面的帧数
- e) ./image.jpg 用于设置生成的照片的名字和路径

```
orangepi@orangepi:~$ sudo fswebcam -d /dev/video0 \  
--no-banner -r 1280x720 -S 5 ./image.jpg
```

c. 在服务器版的 linux 系统中，拍完照后可以使用 scp 命令将拍好的图片传到 Ubuntu PC 上镜像观看

```
orangepi@orangepi:~$ scp image.jpg test@192.168.1.55:/home/test (根据实际情况修改 IP 地址和路径)
```

d. 在桌面版的 linux 系统中，可以通过 HDMI 显示器直接查看拍摄的图片

5) 使用 mjpg-streamer 测试 USB 摄像头

a. 下载 mjpg-streamer

- a) Github 的下载地址：

```
orangepi@orangepi:~$ git clone https://github.com/jacksonliam/mjpg-streamer
```

- b) Gitee 的镜像下载地址为：

```
orangepi@orangepi:~$ git clone https://gitee.com/leeboby/mjpg-streamer
```

b. 安装依赖的软件包

- a) Ubuntu 系统

```
orangepi@orangepi:~$ sudo apt-get install -y cmake libjpeg8-dev
```



b) Debian 系统

```
orangepi@orangepi:~$ sudo apt-get install -y cmake libjpeg62-turbo-dev
```

c. 编译安装 mjpg-streamer

```
orangepi@orangepi:~$ cd mjpg-streamer/mjpg-streamer-experimental
```

```
orangepi@orangepi:~/mjpg-streamer/mjpg-streamer-experimental$ make -j4
```

```
orangepi@orangepi:~/mjpg-streamer/mjpg-streamer-experimental$ sudo make install
```

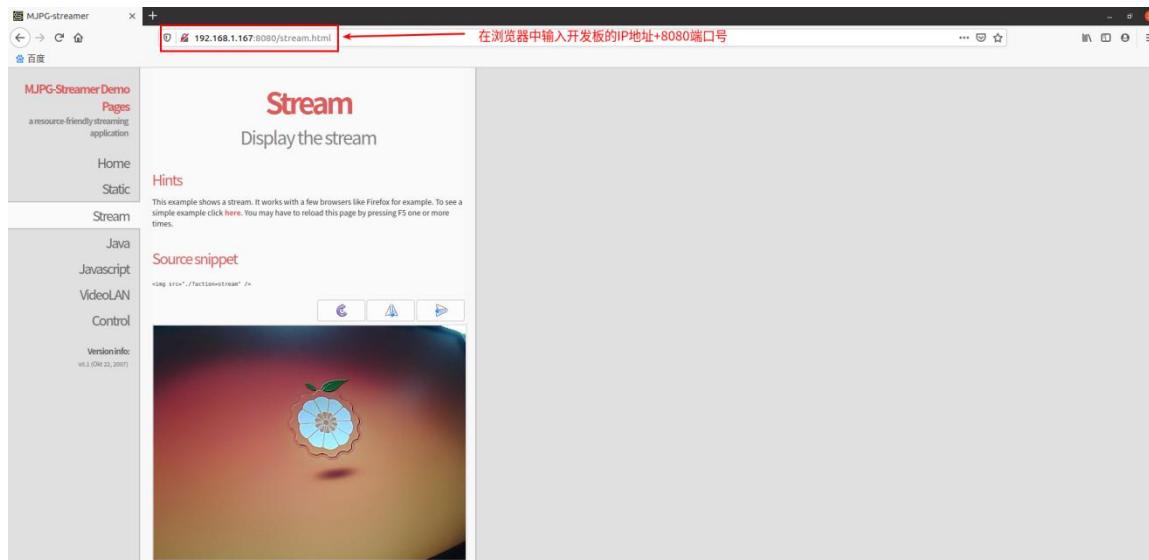
d. 然后输入下面的命令启动 mjpg_streamer

注意，video 的序号不一定都是 video0，请以实际看到的为准。

```
orangepi@orangepi:~/mjpg-streamer/mjpg-streamer-experimental$ export LD_LIBRARY_PATH=.
```

```
orangepi@orangepi:~/mjpg-streamer/mjpg-streamer-experimental$ sudo ./mjpg_streamer -i "./input_uvc.so -d /dev/video0 -u -f 30" -o "./output_http.so -w /www"
```

e. 然后在和开发板同一局域网的 Ubuntu PC 或者 Windows PC 或者手机的浏览器中输入【开发板的 IP 地址:8080】就能看到摄像头输出的视频了

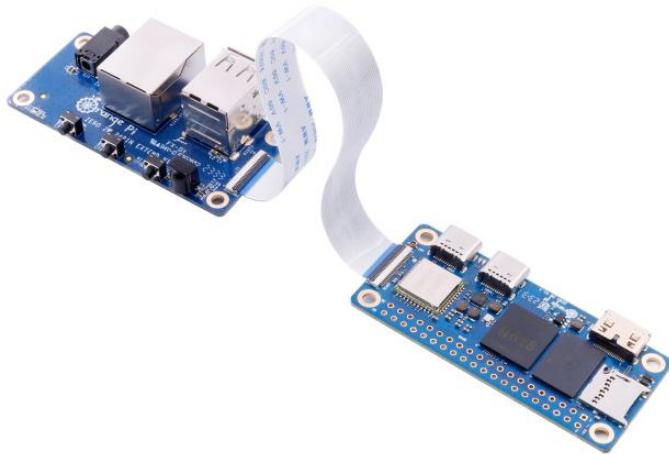


3.14. 音频测试

3.14.1. 使用命令行播放音频的方法

3.14.1.1. 耳机接口播放音频测试

- 1) 开发板主板上是没有耳机接口的，我们可以通过 24pin 扩展板来扩展



2) 通过 **aplay -l** 命令可以查看 linux 系统支持的声卡设备

a. linux5.4 系统的输出如下所示，其中 **card 0: audiocodec** 就是耳机播放需要的声卡设备

```
root@orangeipi:~# aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: audiocodec [audiocodec], device 0: soc@3000000:codec_plat-5096000.codec5096000.codec-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

b. linux6.1 系统的输出如下，其中 **audiocodec** 就是耳机播放需要的声卡设备

```
root@orangeipi:~# aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: audiocodec [audiocodec], device 0: CDC PCM Codec-0 [CDC PCM Codec-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

3) 然后使用 **aplay** 命令播放音频，耳机就能听到声音了

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

如果耳机测试有杂音，请将耳机拔出来一些，不要全部插到底。



3. 14. 1. 2. HDMI 音频播放测试

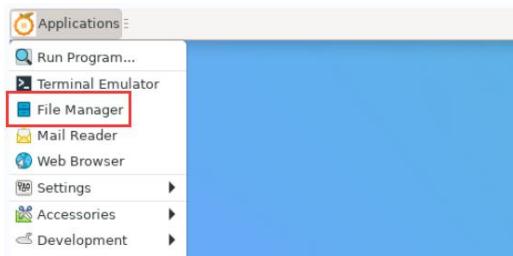
1) 首先使用 Mini HDMI 转 HDMI 线将 Orange Pi 开发板连接到电视机上（其他的 HDMI 显示器需要确保可以播放音频）

2) HDMI 音频播放无需其他设置，直接使用 **aplay** 命令播放即可

```
root@orangepi:~# aplay -D hw:2,0 /usr/share/sounds/alsa/audio.wav
```

3. 14. 2. 在桌面系统中测试音频方法

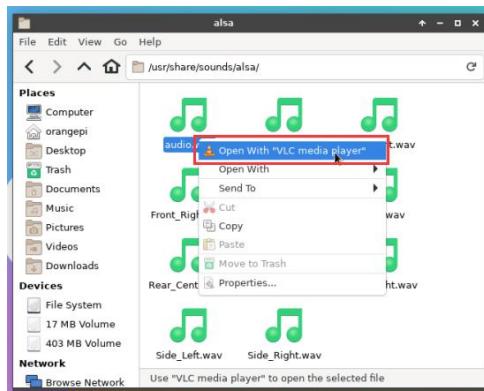
1) 首先打开文件管理器



2) 然后找到下面这个文件（如果系统中没有这个音频文件，可以自己上传一个音频文件到系统中）

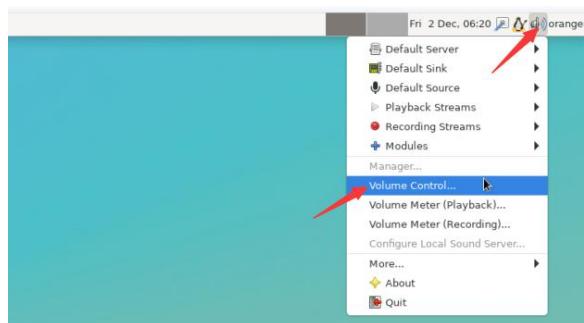


3) 然后选中 **audio.wav** 文件，右键选择使用 **vlc** 打开就可以开始播放

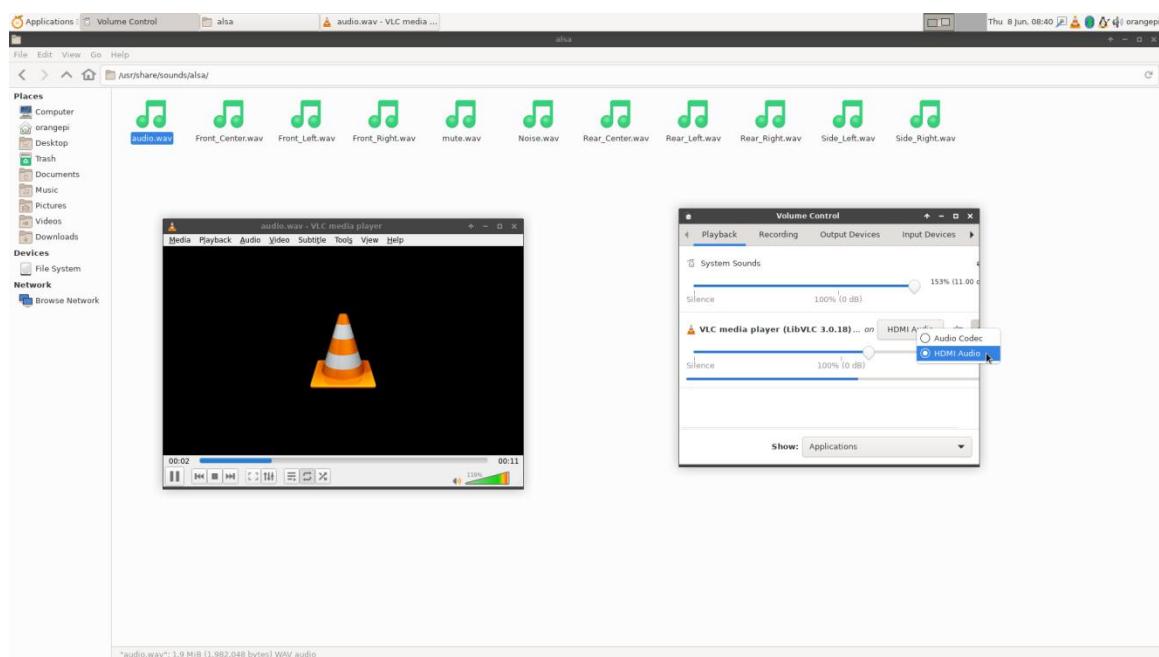


4) 切换 HDMI 播放和耳机播放等不同音频设备的方法

a. 首先打开音量控制界面



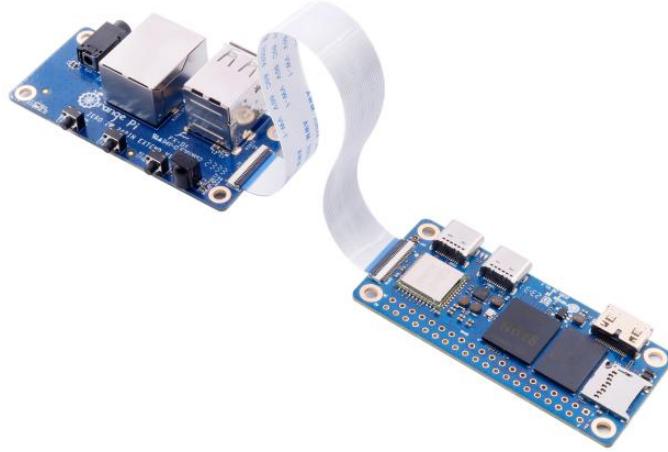
b. 播放音频的时候，在 **Playback** 中会显示播放软件可以使用的音频设备选项，如下图所示，在这里可以设置需要播放到哪个音频设备





3. 15. 红外接收测试

1) 开发板主板上是没有红外接收器的，我们可以通过 24pin 扩展板来扩展



2) 安装 ir-keytable 红外测试软件

```
orangepi@orangepi:~$ sudo apt update  
orangepi@orangepi:~$ sudo apt-get install -y ir-keytable
```

3) 然后执行 ir-keytable 可以查看红外设备的信息

a. linux5.4 系统输出如下所示

```
orangepi@orangepi:~$ ir-keytable  
Found /sys/class/rc/rc0/ with:  
    Name: sunxi-ir  
    Driver: sunxi-rc-recv  
    Default keymap: rc_map_sunxi  
    Input device: /dev/input/event1  
    LIRC device: /dev/lirc0  
    Attached BPF protocols: Operation not permitted  
    Supported kernel protocols: lirc nec  
    Enabled kernel protocols: lirc nec  
    bus: 25, vendor/product: 0001:0001, version: 0x0100  
    Repeat delay = 500 ms, repeat period = 125 ms
```

b. linux6.1 系统的输出如下所示

```
orangepi@orangepi:~$ ir-keytable  
Found /sys/class/rc/rc0/ with:
```



```
Name: sunxi-ir
Driver: sunxi-ir
Default keymap: rc-empty
Input device: /dev/input/event5
LIRC device: /dev/lirc0
Attached BPF protocols: Operation not permitted
Supported kernel protocols: lirc rc-5 rc-5-sz jvc sony nec sanyo mce_kbd rc-6 sharp
xmp imon rc-mm
Enabled kernel protocols: lirc
bus: 25, vendor/product: 0001:0001, version: 0x0100
Repeat delay = 500 ms, repeat period = 125 ms
```

4) 测试红外接收功能前需要准备一个 Orange Pi 专用的红外遥控器，**其他遥控器不支持**



5) 然后在终端中输入 **ir-keytable -t** 命令，再使用红外遥控器对着 Orange Pi 开发板的红外接收头按下按键就能在终端中看到接收到的按键编码了

a. linux5.4 系统输出如下所示

```
orangepi@orangepi:~$ sudo ir-keytable -t
Testing events. Please, press CTRL-C to abort.
1598339152.260376: event type EV_MSC(0x04): scancode = 0xfb0413
1598339152.260376: event type EV_SYN(0x00).
1598339152.914715: event type EV_MSC(0x04): scancode = 0xfb0410
```

b. linux6.1 系统输出如下所示

```
orangepi@orangepi:~$ sudo ir-keytable -c -p NEC -t
Old keytable cleared
Protocols changed to nec
Testing events. Please, press CTRL-C to abort.
202.063219: lirc protocol(nec): scancode = 0x45c
```



```
202.063249: event type EV_MSC(0x04): scancode = 0x45c  
202.063249: event type EV_SYN(0x00).
```

3. 16. 温度传感器

3. 16. 1. linux5.4 系统查看温度的方法

H618 总共有 4 个温度传感器，查看温度的命令如下所示：

显示的温度值需要除以 1000，单位才是摄氏度。

- a. sensor0: CPU 的温度传感器，第一条命令用于查看温度传感器的类型，第二条命令用于查看温度传感器的数值

```
orangeipi@orangeipi:~$ cat /sys/class/thermal/thermal_zone0/type  
cpu_thermal_zone  
orangeipi@orangeipi:~$ cat /sys/class/thermal/thermal_zone0/temp  
57734
```

- b. sensor1: DDR 的温度传感器，第一条命令用于查看温度传感器的类型，第二条命令用于查看温度传感器的数值

```
orangeipi@orangeipi:~$ cat /sys/class/thermal/thermal_zone1/type  
ddr_thermal_zone  
orangeipi@orangeipi:~$ cat /sys/class/thermal/thermal_zone1/temp  
57410
```

- c. sensor2: GPU 的温度传感器，第一条命令用于查看温度传感器的类型，第二条命令用于查看温度传感器的数值

```
orangeipi@orangeipi:~$ cat /sys/class/thermal/thermal_zone2/type  
gpu_thermal_zone  
orangeipi@orangeipi:~$ cat /sys/class/thermal/thermal_zone2/temp  
59273
```

- d. sensor3: VE 的温度传感器，第一条命令用于查看温度传感器的类型，第二条命令用于查看温度传感器的数值

```
orangeipi@orangeipi:~$ cat /sys/class/thermal/thermal_zone3/type  
ve_thermal_zone  
orangeipi@orangeipi:~$ cat /sys/class/thermal/thermal_zone3/temp  
58949
```



3. 16. 2. linux6.1 系统查看温度的方法

```
orangepi@orangepi:~$ sensors
cpu_thermal-virtual-0
Adapter: Virtual device
temp1:      +47.4°C  (crit = +110.0°C)

gpu_thermal-virtual-0
Adapter: Virtual device
temp1:      +48.7°C  (crit = +110.0°C)

ddr_thermal-virtual-0
Adapter: Virtual device
temp1:      +47.8°C  (crit = +110.0°C)

ve_thermal-virtual-0
Adapter: Virtual device
temp1:      +47.2°C  (crit = +110.0°C)
```

3. 17. 40 Pin 接口引脚说明

注意：40pin接口上的排针默认是不焊接的，需要自己焊接上去才能使用。

1) Orange Pi Zero 2w 开发板 40 Pin 接口引脚的顺序请参开发板上的丝印图



2) 开发板 40 Pin 接口引脚的功能如下表所示

GPIO序号	GPIO	功能	引脚	引脚	功能	GPIO	GPIO序号



		3.3V	1
264	PI8	TWI1-SDA	3
263	PI7	TWI1-SCL	5
269	PI13	PWM3/UART4_TX	7
		GND	9
226	PH2	UART5_TX	11
227	PH3	UART5_RX	13
261	PI5	TWI0_SCL/UART2_TX	15
		3.3V	17
231	PH7	SPI1_MOSI	19
232	PH8	SPI1_MISO	21
230	PH6	SPI1_CLK	23
		GND	25
266	PI10	TWI2-SDA/UART3_RX	27
256	PI10		29
271	PI15		31
268	PI12	PWM2	33
258	PI2		35
272	PI16		37
		GND	39

2	5V		
4	5V		
6	GND		
8	UART0_TX	PH0	224
10	UART0_RX	PH1	225
12		PI1	257
14	GND		
16	PWM4/UART4_RX	PI14	270
18		PH4	228
20	GND		
22	TWI0_SDA/UART2_RX	PI6	262
24	SPI1_CS0	PH5	229
26	SPI1_CS1	PH9	233
28	TWI2-SCL/UART3_TX	PI9	265
30	GND		
32	PWM1	PI11	267
34	GND		
36		PC12	76
38		PI4	260
40		PI3	259

3) 40pin 接口中总共有 28 个 GPIO 口，所有 GPIO 口的高电平电压都是 **3.3v**

3.18. 安装 wiringOP 的方法

注意，Orange Pi 发布的 linux 镜像中已经预装了 wiringOP，除非 wiringOP 的代码有更新，否则无需重新下载编译安装，直接使用即可。

编译好的 wiringOP 的 deb 包在 orangepi-build 中的存放路径为：

orangepi-build/external/cache/debs/arm64/wiringpi_x.xx.deb

进入系统后可以运行下 `gpio readall` 命令，如果能看到下面的输出，说明 wiringOP 已经预装并且能正常使用。



```
orangepi@orangepizero2w:~$ gpio readall
+-----+-----+-----+-----+-----+-----+-----+-----+
| GPIO | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | GPIO |
+-----+-----+-----+-----+-----+-----+-----+-----+
|      |     | 3.3V |      |   |      |   |      | 5V  |      | | |
| 264  | 0  | SDA.1 | OFF  | 0 | 3     | 4 |      | 5V  |      |
| 263  | 1  | SCL.1 | OFF  | 0 | 5     | 6 |      | GND |      |
| 269  | 2  | PWM3  | OFF  | 0 | 7     | 8 | 0    | ALT2 | TXD.0 | 3  | 224 |
|      |     | GND   |      |   | 9     | 10| 0   | ALT2 | RXD.0 | 4  | 225 |
| 226  | 5  | TXD.5 | OFF  | 0 | 11    | 12| 0   | OFF  | PI01  | 6  | 257 |
| 227  | 7  | RXD.5 | OFF  | 0 | 13    | 14|      |      | GND  |      |
| 261  | 8  | TXD.2 | OFF  | 0 | 15    | 16| 0   | OFF  | PWM4  | 9  | 270 |
|      |     | 3.3V  |      |   | 17    | 18| 0   | OFF  | PH04  | 10 | 228 |
| 231  | 11 | MOSI.1| OFF  | 0 | 19    | 20|      |      | GND  |      |
| 232  | 12 | MISO.1| OFF  | 0 | 21    | 22| 0   | OFF  | RXD.2 | 13 | 262 |
| 230  | 14 | SCLK.1| OFF  | 0 | 23    | 24| 0   | OFF  | CE.0   | 15 | 229 |
|      |     | GND   |      |   | 25    | 26| 0   | ALT3 | CE.1   | 16 | 233 |
| 266  | 17 | SDA.2 | OFF  | 0 | 27    | 28| 0   | OFF  | SCL.2 | 18 | 265 |
| 256  | 19 | PI00  | OFF  | 0 | 29    | 30|      |      | GND  |      |
| 271  | 20 | PI15  | OFF  | 0 | 31    | 32| 0   | OFF  | PWM1  | 21 | 267 |
| 268  | 22 | PI12  | OFF  | 0 | 33    | 34|      |      | GND  |      |
| 258  | 23 | PI02  | OFF  | 0 | 35    | 36| 1   | OUT  | PC12  | 24 | 76  |
| 272  | 25 | PI16  | OFF  | 0 | 37    | 38| 0   | OFF  | PI04  | 26 | 260 |
|      |     | GND   |      |   | 39    | 40| 0   | OFF  | PI03  | 27 | 259 |
+-----+-----+-----+-----+-----+-----+-----+-----+
| GPIO | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | GPIO |
+-----+-----+-----+-----+-----+-----+-----+-----+
|      |     | ZERO2W |      |   |      |   |      |      |      |
orangepi@orangepizero2w:~$
```

1) 下载 wiringOP 的代码

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

注意，源码需要下载 wiringOP next 分支的代码，请别漏了 -b next 这个参数。

如果从 GitHub 下载代码有问题，可以直接使用 Linux 镜像中自带的 wiringOP 源码，存放位置为： /usr/src/wiringOP。

2) 编译安装 wiringOP

```
orangepi@orangepi:~$ cd wiringOP
orangepi@orangepi:~/wiringOP$ sudo ./build clean
orangepi@orangepi:~/wiringOP$ sudo ./build
```

3) 测试 gpio readall 命令的输出如下



```
orangepi@orangepizero2w:~$ gpio readall
+-----+-----+-----+-----+-----+-----+-----+-----+
| GPIO | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | GPIO |
+-----+-----+-----+-----+-----+-----+-----+-----+
|       |     3.3V |       |   | 1 | 2 |       |   | 5V |   | |
| 264 | 0 | SDA.1 | OFF | 0 | 3 | 4 |       |   | 5V |   |
| 263 | 1 | SCL.1 | OFF | 0 | 5 | 6 |       |   | GND |   |
| 269 | 2 | PWM3 | OFF | 0 | 7 | 8 | 0 | ALT2 | TXD.0 | 3 | 224
|       |     GND |       |   | 9 | 10 | 0 | ALT2 | RXD.0 | 4 | 225
| 226 | 5 | TXD.5 | OFF | 0 | 11 | 12 | 0 | OFF | PI01 | 6 | 257
| 227 | 7 | RXD.5 | OFF | 0 | 13 | 14 |       |   | GND |   |
| 261 | 8 | TXD.2 | OFF | 0 | 15 | 16 | 0 | OFF | PWM4 | 9 | 270
|       |     3.3V |       |   | 17 | 18 | 0 | OFF | PH04 | 10 | 228
| 231 | 11 | MOSI.1 | OFF | 0 | 19 | 20 |       |   | GND |   |
| 232 | 12 | MISO.1 | OFF | 0 | 21 | 22 | 0 | OFF | RXD.2 | 13 | 262
| 230 | 14 | SCLK.1 | OFF | 0 | 23 | 24 | 0 | OFF | CE.0 | 15 | 229
|       |     GND |       |   | 25 | 26 | 0 | ALT3 | CE.1 | 16 | 233
| 266 | 17 | SDA.2 | OFF | 0 | 27 | 28 | 0 | OFF | SCL.2 | 18 | 265
| 256 | 19 | PI00 | OFF | 0 | 29 | 30 |       |   | GND |   |
| 271 | 20 | PI15 | OFF | 0 | 31 | 32 | 0 | OFF | PWM1 | 21 | 267
| 268 | 22 | PI12 | OFF | 0 | 33 | 34 |       |   | GND |   |
| 258 | 23 | PI02 | OFF | 0 | 35 | 36 | 1 | OUT | PC12 | 24 | 76
| 272 | 25 | PI16 | OFF | 0 | 37 | 38 | 0 | OFF | PI04 | 26 | 260
|       |     GND |       |   | 39 | 40 | 0 | OFF | PI03 | 27 | 259
+-----+-----+-----+-----+-----+-----+-----+-----+
| GPIO | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | GPIO |
+-----+-----+-----+-----+-----+-----+-----+-----+
|       |     ZERO2W |       |   |       |   |       |   |       |   |
orangepi@orangepizero2w:~$
```

3.19. 40pin 接口 GPIO、I2C、UART、SPI 和 PWM 测试

注意：40pin接口上的排针默认是不焊接的，需要自己焊接上去才能使用。

3.19.1. 40pin GPIO 口测试

- 1) 下面以 7 号引脚——对应 GPIO 为 PI13——对应 wPi 序号为 2——为例演示如何设置 GPIO 口的高低电平

```
orangepi@orangepizero2w:~$ gpio readall
+-----+-----+-----+-----+-----+-----+-----+-----+
| GPIO | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | GPIO |
+-----+-----+-----+-----+-----+-----+-----+-----+
|       |     3.3V |       |   | 1 | 2 |       |   | 5V |   | |
| 264 | 0 | SDA.1 | OFF | 0 | 3 | 4 |       |   | 5V |   |
| 263 | 1 | SCL.1 | OFF | 0 | 5 | 6 |       |   | GND |   |
| 269 | 2 | PWM3 | OFF | 0 | 7 | 8 | 0 | ALT2 | TXD.0 | 3 | 224
|       |     GND |       |   | 9 | 10 | 0 | ALT2 | RXD.0 | 4 | 225
|       |     ZERO2W |       |   |       |   |       |   |       |   |
orangepi@orangepizero2w:~$
```

- 2) 首先设置 GPIO 口为输出模式，其中第三个参数需要输入引脚对应的 wPi 的序号

```
root@orangepi:~/wiringOP# gpio mode 2 out
```



- 3) 然后设置 GPIO 口输出低电平, 设置完后可以使用万用表测量引脚的电压的数值, 如果为 0v, 说明设置低电平成功

```
root@orangepi:~/wiringOP# gpio write 2 0
```

- 4) 然后设置 GPIO 口输出高电平, 设置完后可以使用万用表测量引脚的电压的数值, 如果为 3.3v, 说明设置高电平成功

```
root@orangepi:~/wiringOP# gpio write 2 1
```

- 5) 其他引脚的设置方法类似, 只需修改 wPi 的序号为引脚对应的序号即可

3.19.2. 40 Pin GPIO 口上下拉电阻的设置方法

- 1) 下面以 7 号引脚——对应 GPIO 为 PI13——对应 wPi 序号为 2——为例演示如何设置 GPIO 口的上下拉电阻

ZERO2W											
GPIO	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	GPIO	
264	0	3.3V			1	2		5V			
263	1	SDA.1	OFF	0	3	4		5V			
269	2	SCL.1	OFF	0	5	6		GND			
		PWM3	OFF	0	7	8	0	ALT2	TXD.0	3	224
		GND			9	10	0	ALT2	RXD.0	4	225

- 2) 首先需要设置 GPIO 口为输入模式, 其中第三个参数需要输入引脚对应的 wPi 的序号

```
root@orangepi:~/wiringOP# gpio mode 2 in
```

- 3) 设置为输入模式后, 执行下面的命令可以设置 GPIO 口为上拉模式

```
root@orangepi:~/wiringOP# gpio mode 2 up
```

- 4) 然后输入下面的命令读取 GPIO 口的电平, 如果电平为 1, 说明上拉模式设置成功

```
root@orangepi:~/wiringOP# gpio read 2
```

1

- 5) 然后执行下面的命令可以设置 GPIO 口为下拉模式

```
root@orangepi:~/wiringOP# gpio mode 2 down
```



6) 然后输入下面的命令读取 GPIO 口的电平，如果电平为 0，说明下拉模式设置成功

```
root@orangepi:~/wiringOP# gpio read 2  
0
```

3.19.3. 40pin SPI 测试

1) 由下表可知，40pin 接口可用的 spi 为 spi1，有两个片选引脚 cs0 和 cs1

GPIO序号	GPIO	功能	引脚
		3.3V	1
264	PI8	TWI1-SDA	3
263	PI7	TWI1-SCL	5
269	PI13	PWM3/UART4_TX	7
		GND	9
226	PH2	UART5_TX	11
227	PH3	UART5_RX	13
261	PI5	TWI0_SCL/UART2_TX	15
		3.3V	17
231	PH7	SPI1_MOSI	19
232	PH8	SPI1_MISO	21
230	PH6	SPI1_CLK	23
		GND	25
266	PI10	TWI2-SDA/UART3_RX	27
256	PI0		29
271	PI15		31
268	PI12	PWM2	33
258	PI2		35
272	PI16		37
		GND	39

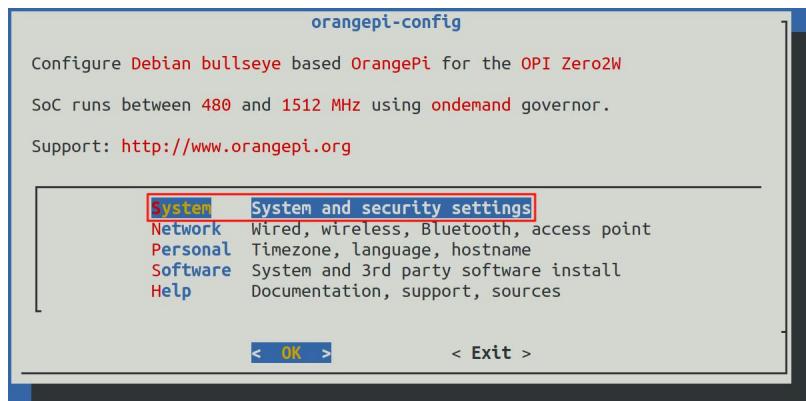
引脚	功能	GPIO	GPIO序号
2	5V		
4	5V		
6	GND		
8	UART0_TX	PH0	224
10	UART0_RX	PH1	225
12		PI1	257
14	GND		
16	PWM4/UART4_RX	PI14	270
18		PH4	228
20	GND		
22	TWI0_SDA/UART2_RX	PI6	262
24	SPI1_CS0	PH5	229
26	SPI1_CS1	PH9	233
28	TWI2-SCL/UART3_TX	PI9	265
30	GND		
32	PWM1	PI11	267
34	GND		
36		PC12	76
38		PI4	260
40		PI3	259

2) Linux 系统中 spi1 默认是关闭的，需要手动打开才能使用。打开步骤如下所示：

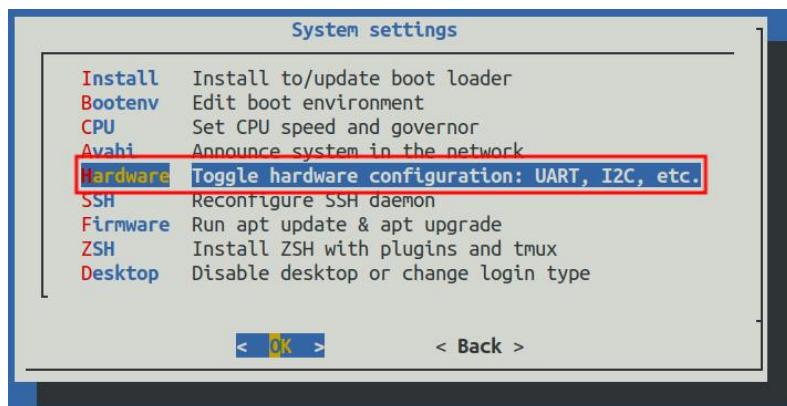
a. 首先运行下 **orangepi-config**，普通用户记得加 **sudo** 权限

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

b. 然后选择 **System**

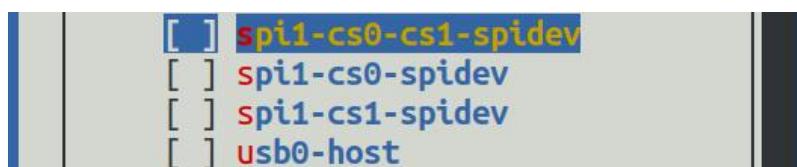


c. 然后选择 **Hardware**

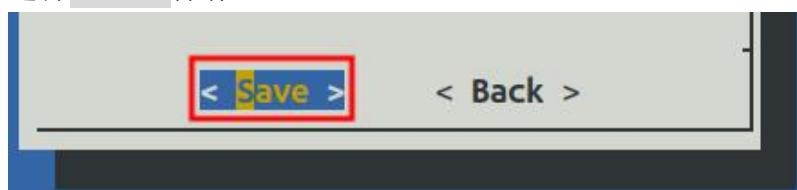


d. 然后使用键盘的方向键定位到下图所示的位置,再使用空格选中想要打开的 SPI 的 dtbo 配置

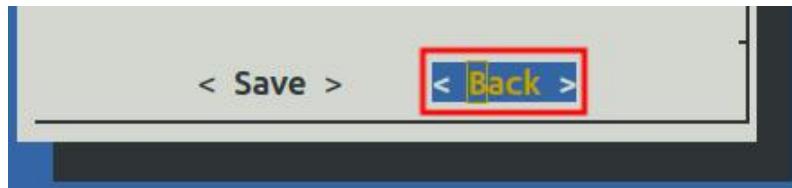
dtbo 配置	说明
spi1-cs0-cs1-spidev	同时打开 spi1 的 cs0 和 cs1
spi1-cs0-spidev	只打开 spi1 的 cs0
spi1-cs1-spidev	只打开 spi1 的 cs1



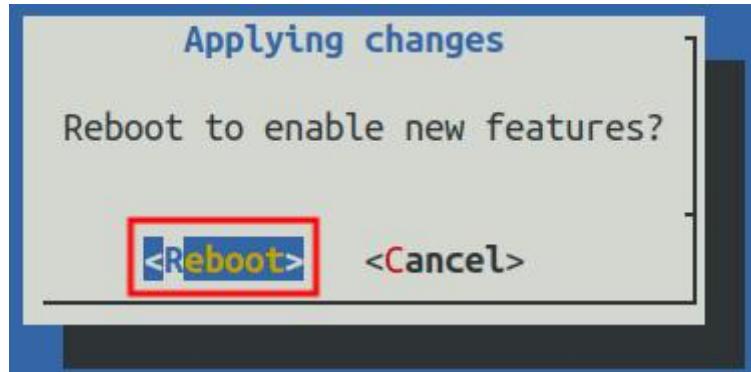
e. 然后选择<Save>保存



f. 然后选择<Back>



g. 然后选择**<Reboot>**重启系统使配置生效



3) 然后查看下 linux 系统中是否存在 **spidev1.x** 的设备节点，如果存在，说明 SPI1 的配置已经生效了

```
orangeipi@orangeipi:~$ ls /dev/spidev1*  
/dev/spidev1.0  /dev/spidev1.1
```

注意，只有打开 spi1-cs0-cs1-spidev 时，才会看到两个 spi 的设备节点。

4) 下面开始 spi 的回环测试, 先不短接 SPI1 的 mosi 和 miso 两个引脚, 运行 spidev_test 的输出结果如下所示, 可以看到 TX 和 RX 的数据不一致

5) 然后短接 SPI1 的 mosi (40pin 接口中的第 19 号引脚) 和 miso (40pin 接口中的第 21 号引脚) 两个引脚再运行 spidev_test 的输出如下，可以看到发送和接收的数据一样，说明回环测试通过。



3. 19. 4. 40pin I2C 测试

1) 由下表可知, 40pin 接口可用的 i2c 为 i2c0、i2c1 和 i2c2

GPIO序号	GPIO	功能	引脚
		3.3V	1
264	PI8	TWI1-SDA	3
263	PI7	TWI1-SCL	5
269	PI13	PWM3/UART4_TX	7
		GND	9
226	PH2	UART5_TX	11
227	PH3	UART5_RX	13
261	PI5	TWI0_SCL/UART2_TX	15
		3.3V	17
231	PH7	SPI1_MOSI	19
232	PH8	SPI1_MISO	21
230	PH6	SPI1_CLK	23
		GND	25
266	PI10	TWI2-SDA/UART3_RX	27
256	PIO		29
271	PI15		31
268	PI12	PWM2	33
258	PI2		35
272	PI16		37
		GND	39

引脚	功能	GPIO	GPIO序号
2	5V		
4	5V		
6	GND		
8	UART0_TX	PH0	224
10	UART0_RX	PH1	225
12		PI1	257
14	GND		
16	PWM4/UART4_RX	PI14	270
18		PH4	228
20	GND		
22	TWI0_SDA/UART2_RX	PI6	262
24	SPI1_CS0	PH5	229
26	SPI1_CS1	PH9	233
28	TWI2-SCL/UART3_TX	PI9	265
30	GND		
32	PWM1	PI11	267
34	GND		
36		PC12	76
38		PI4	260
40		PI3	259

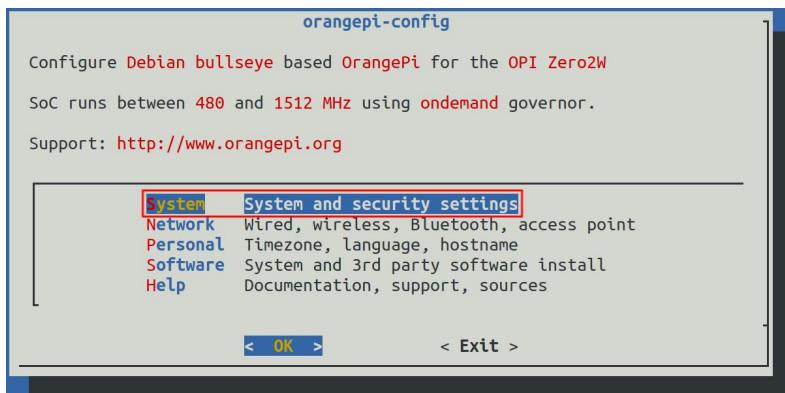
2) Linux 系统中 i2c 默认都是关闭的，需要手动打开才能使用。打开步骤如下所示：



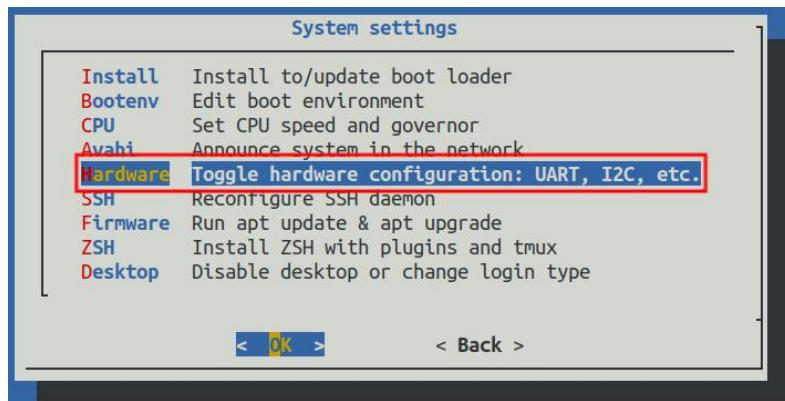
- a. 首先运行下 **orangepi-config**, 普通用户记得加 **sudo** 权限

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

- b. 然后选择 **System**



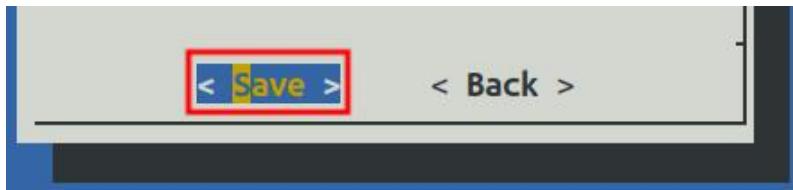
- c. 然后选择 **Hardware**



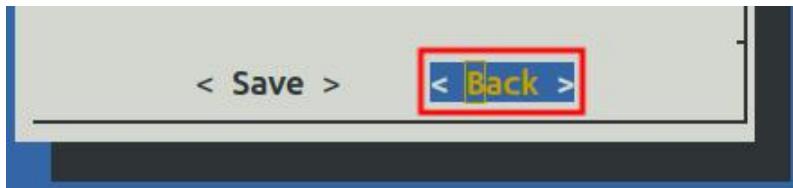
- d. 然后使用键盘的方向键定位到下图所示的位置, 再使用空格选中下图中对应的 i2c 的配置

40pin 中的复用功能	对应的 dtbo 配置
40pin - i2c0	pi-i2c0
40pin - i2c1	pi-i2c1
40pin - i2c2	pi-i2c2

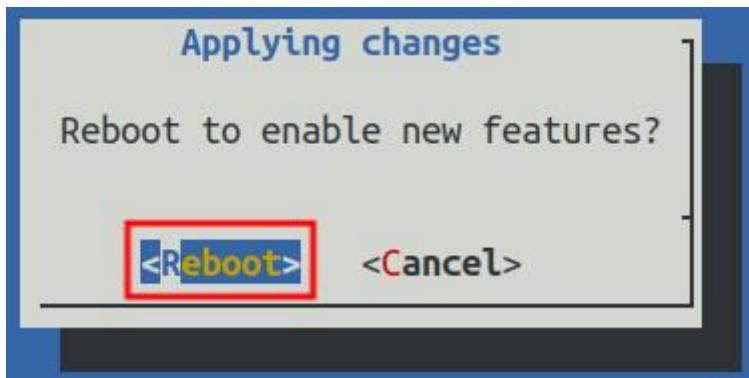
- e. 然后选择<Save>保存



f. 然后选择<Back>



g. 然后选择<Reboot>重启系统使配置生效



3) 启动 linux 系统后，先确认下/dev 下存在已打开的 i2c 的设备节点

```
orangepi@orangepi:~$ ls /dev/i2c-*
```

```
/dev/i2c-*
```

有时 i2c 的设备节点和 i2c 总线的序号不是一一对应的，比如 i2c1 总线的 i2c 设备节点可能是 /dev/i2c-3。

准确确认 i2c 总线对应的 /dev 下设备节点方法为：

a. 首先运行下面的命令，查看 i2c 的对应关系

```
orangepi@orangepizero2w:~$ ls /sys/devices/platform/soc/*/*i2c-* | grep "i2c-[0-9]"
```

```
/sys/devices/platform/soc/5002000.i2c/i2c-0:
```

```
/sys/devices/platform/soc/5002400.i2c/i2c-3:
```

```
/sys/devices/platform/soc/5002800.i2c/i2c-4:
```

```
/sys/devices/platform/soc/5002c00.i2c/i2c-5:
```

```
/sys/devices/platform/soc/6000000.hdmi/i2c-2:
```

```
/sys/devices/platform/soc/7081400.i2c/i2c-1:
```



b. 上面的输出中

- a) 5002000 为 i2c0 总线的寄存器地址，其后面显示 i2c-0 就是其对应的 i2c 设备节点
- b) 5002400 为 i2c1 总线的寄存器地址，其后面显示 i2c-3 就是其对应的 i2c 设备节点
- c) 5002800 为 i2c2 总线的寄存器地址，其后面显示 i2c-4 就是其对应的 i2c 设备节点

4) 然后开始测试 i2c，首先安装 i2c-tools

```
orangeipi@orangeipi:~$ sudo apt-get update  
orangeipi@orangeipi:~$ sudo apt-get install -y i2c-tools
```

5) 然后在 40pin 接头的 i2c 引脚上接一个 i2c 设备

6) 然后使用 **i2cdetect -y x** 命令如果能检测到连接的 i2c 设备的地址，就说明 i2c 能正常使用

注意，**i2cdetect -y x** 命令中的 x 需要替换为 i2c 总线对应的设备节点的序号。

不同的 i2c 设备地址是不同的，下图 0x50 地址只是一个示例。请以实际看到的为准。

```
root@orangeipi:~# i2cdetect -y 3  
      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  
00: ---  
10: ---  
20: ---  
30: ---  
40: ---  
50: 50  
60: ---  
70: ---
```

3.19.5. 40pin 的 UART 测试

1) 由下表可知，可用的 uart 为 uart2、uart3、uart4 和 uart5。请注意 uart0 默认设置为调试串口，请不要把 uart0 当成普通串口使用

GPIO序号	GPIO	功能	引脚
		3.3V	1
264	PI8	TWI1-SDA	3
263	PI7	TWI1-SCL	5
269	PI13	PWM3/ UART4_TX	7
		GND	9
226	PH2	UART5_TX	11
227	PH3	UART5_RX	13

引脚	功能	GPIO	GPIO序号
2	5V		
4	5V		
6	GND		
8	UART0_TX	PH0	224
10	UART0_RX	PH1	225
12		PI1	257
14	GND		



261	PI5	TWI0_SCL/UART2_TX	15
		3.3V	17
231	PH7	SPI1_MOSI	19
232	PH8	SPI1_MISO	21
230	PH6	SPI1_CLK	23
		GND	25
266	PI10	TWI2-SDA/UART3_RX	27
256	PI0		29
271	PI15		31
268	PI12	PWM2	33
258	PI2		35
272	PI16		37
		GND	39

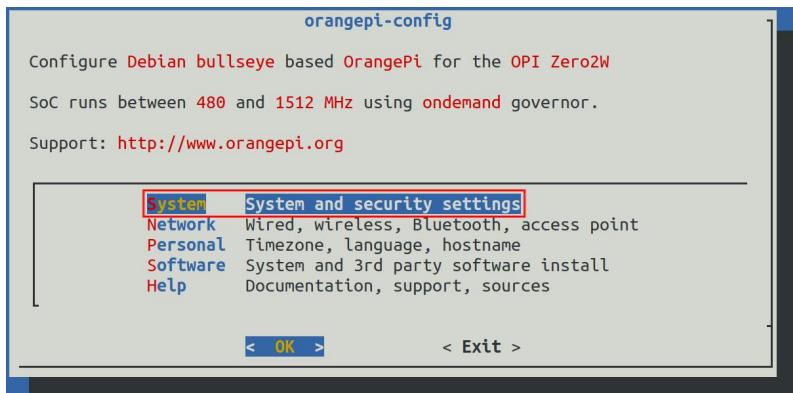
16	PWM4/UART4_RX	PI14	270
18		PH4	228
20	GND		
22	TWI0_SDA/UART2_RX	PI6	262
24	SPI1_CS0	PH5	229
26	SPI1_CS1	PH9	233
28	TWI2-SCL/UART3_TX	PI9	265
30	GND		
32	PWM1	PI11	267
34	GND		
36		PC12	76
38		PI4	260
40		PI3	259

2) Linux 系统中 uart 默认都是关闭的，需要手动打开才能使用。打开步骤如下所示：

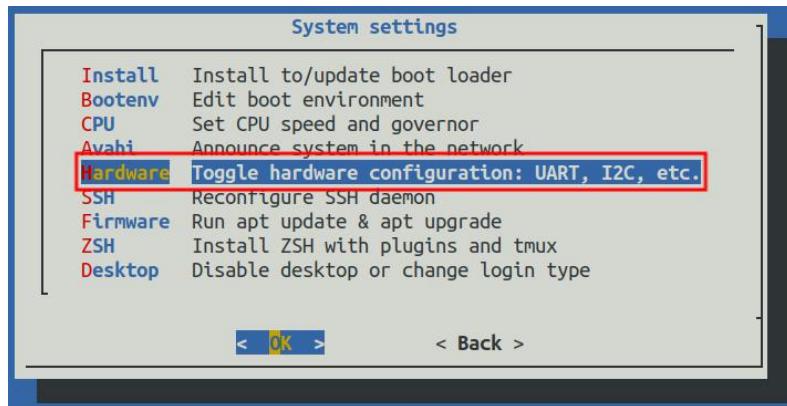
a. 首先运行下 **orangepi-config**，普通用户记得加 **sudo** 权限

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

b. 然后选择 **System**

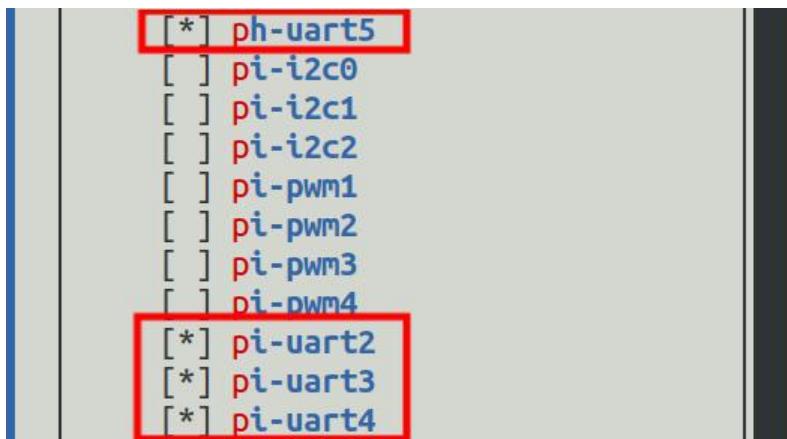


c. 然后选择 **Hardware**

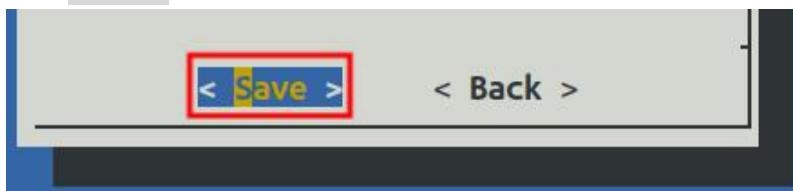


- d. 然后使用键盘的方向键定位到下图所示的位置,再使用空格选中想要打开的串口

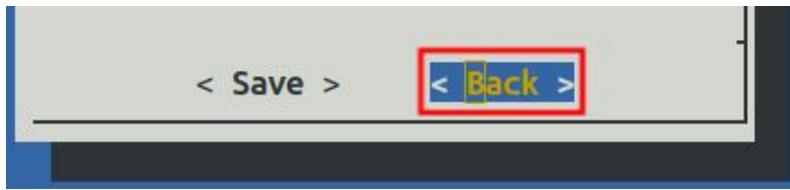
40pin 中的复用功能	对应的 dtbo 配置
40pin - uart2	pi-uart2
40pin - uart3	pi-uart3
40pin - uart4	pi-uart4
40pin - uart5	ph-uart5



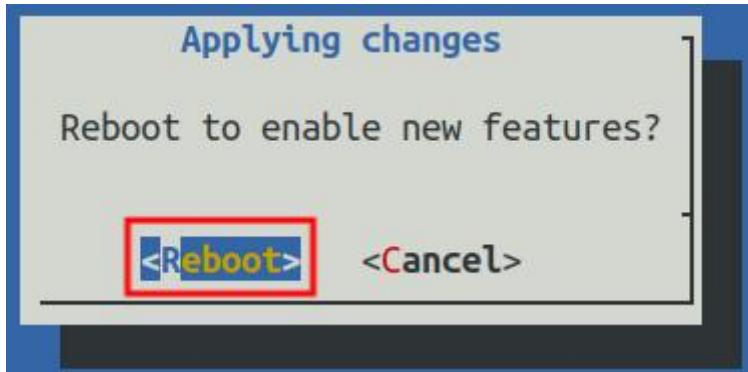
- e. 然后选择<Save>保存



- f. 然后选择<Back>



g. 然后选择<Reboot>重启系统使配置生效



3) 进入 linux 系统后，先确认下/dev 下是否存在 uart5 的设备节点

注意，linux5.4 系统为/dev/ttYSx。

```
orangepi@orangepi:~$ ls /dev/ttYS*
/dev/ttYSx
```

4) 然后开始测试 uart 接口，先使用杜邦线短接要测试的 uart 接口的 rx 和 tx 引脚

5) 使用 wiringOP 中的 **gpio** 命令测试串口的回环功能如下所示，如果能看到下面的打印，说明串口通信正常

注意，**gpio serial /dev/ttYSx** 命令中最后的 x 需要替换为对应的 uart 设备节点的序号。

```
orangepi@orangepi:~$ gpio serial /dev/ttYSx      # linux-6.1 测试命令
orangepi@orangepi:~$ gpio serial /dev/ttYSx      # linux-5.4 测试命令

Out:  0:  ->  0
Out:  1:  ->  1
Out:  2:  ->  2
Out:  3:  ->  3^C
```

3.19.6. 使用/sys/class/pwm/测试 PWM 的方法

1) 由下表可知，可用的 pwm 为 pwm1、pwm2、pwm3 和 pwm4。



GPIO序号	GPIO	功能	引脚
		3.3V	1
264	PI8	TWI1-SDA	3
263	PI7	TWI1-SCL	5
269	PI13	PWM3/UART4_TX	7
		GND	9
226	PH2	UART5_TX	11
227	PH3	UART5_RX	13
261	PI5	TWI0_SCL/UART2_TX	15
		3.3V	17
231	PH7	SPI1_MOSI	19
232	PH8	SPI1_MISO	21
230	PH6	SPI1_CLK	23
		GND	25
266	PI10	TWI2-SDA/UART3_RX	27
256	PI0		29
271	PI15		31
268	PI12	PWM2	33
258	PI2		35
272	PI16		37
		GND	39

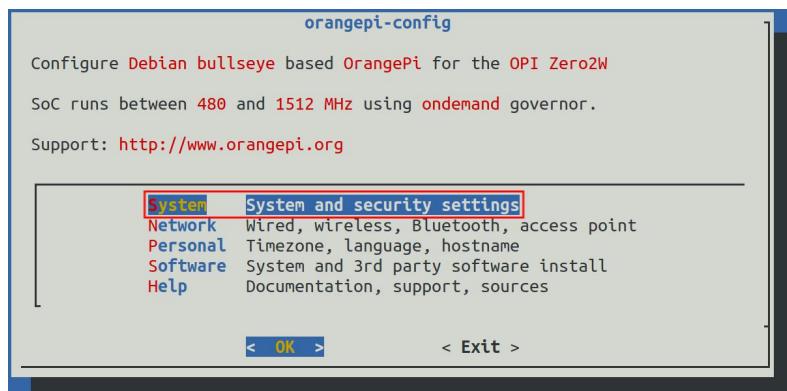
引脚	功能	GPIO	GPIO序号
2	5V		
4	5V		
6	GND		
8	UART0_TX	PH0	224
10	UART0_RX	PH1	225
12		PI1	257
14	GND		
16	PWM4/UART4_RX	PI14	270
18		PH4	228
20	GND		
22	TWI0_SDA/UART2_RX	PI6	262
24	SPI1_CS0	PH5	229
26	SPI1_CS1	PH9	233
28	TWI2-SCL/UART3_TX	PI9	265
30	GND		
32	PWM1	PI11	267
34	GND		
36		PC12	76
38		PI4	260
40		PI3	259

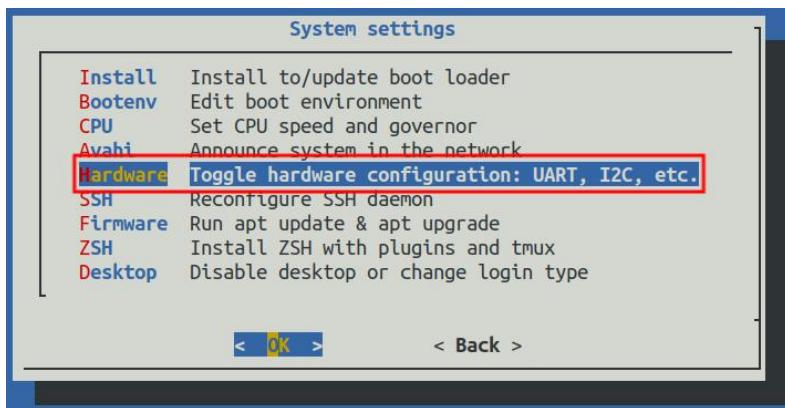
2) Linux 系统中 pwm 默认是关闭的，需要手动打开才能使用。打开步骤如下所示：

a. 首先运行下 **orangepi-config**，普通用户记得加 **sudo** 权限

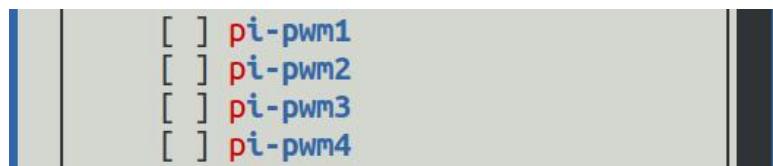
```
orangepi@orangepi:~$ sudo orangepi-config
```

b. 然后选择 **System**

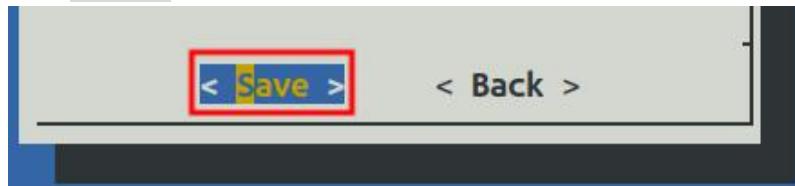


c. 然后选择 **Hardware**

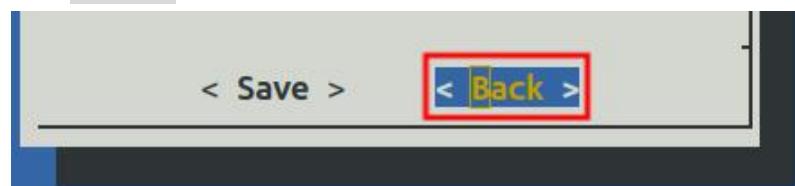
d. 然后使用键盘的方向键定位到下图所示的位置,再使用空格选中想要打开的 pwm 对应的配置



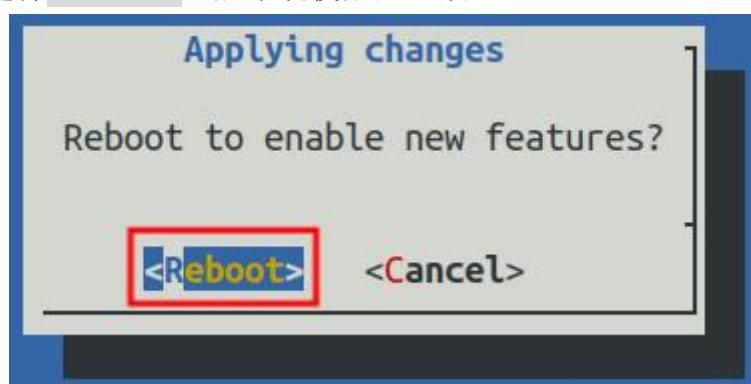
e. 然后选择<Save>保存



f. 然后选择<Back>



g. 然后选择<Reboot>重启系统使配置生效





3) 重启后就可以开始 PWM 的测试

下面的命令请在 **root** 用户下执行。

- a. 在命令行中输入下面的命令可以让 pwm1 输出一个 50Hz 的方波

```
root@orangeipi:~# echo 1 > /sys/class/pwm/pwmchip0/export
root@orangeipi:~# echo 20000000 > /sys/class/pwm/pwmchip0/pwm1/period
root@orangeipi:~# echo 1000000 > /sys/class/pwm/pwmchip0/pwm1/duty_cycle
root@orangeipi:~# echo 1 > /sys/class/pwm/pwmchip0/pwm1/enable
```

- b. 在命令行中输入下面的命令可以让 pwm2 输出一个 50Hz 的方波

```
root@orangeipi:~# echo 2 > /sys/class/pwm/pwmchip0/export
root@orangeipi:~# echo 20000000 > /sys/class/pwm/pwmchip0/pwm2/period
root@orangeipi:~# echo 1000000 > /sys/class/pwm/pwmchip0/pwm2/duty_cycle
root@orangeipi:~# echo 1 > /sys/class/pwm/pwmchip0/pwm2/enable
```

- c. 在命令行中输入下面的命令可以让 pwm3 输出一个 50Hz 的方波

```
root@orangeipi:~# echo 3 > /sys/class/pwm/pwmchip0/export
root@orangeipi:~# echo 20000000 > /sys/class/pwm/pwmchip0/pwm3/period
root@orangeipi:~# echo 1000000 > /sys/class/pwm/pwmchip0/pwm3/duty_cycle
root@orangeipi:~# echo 1 > /sys/class/pwm/pwmchip0/pwm3/enable
```

- d. 在命令行中输入下面的命令可以让 pwm4 输出一个 50Hz 的方波

```
root@orangeipi:~# echo 4 > /sys/class/pwm/pwmchip0/export
root@orangeipi:~# echo 20000000 > /sys/class/pwm/pwmchip0/pwm4/period
root@orangeipi:~# echo 1000000 > /sys/class/pwm/pwmchip0/pwm4/duty_cycle
root@orangeipi:~# echo 1 > /sys/class/pwm/pwmchip0/pwm4/enable
```

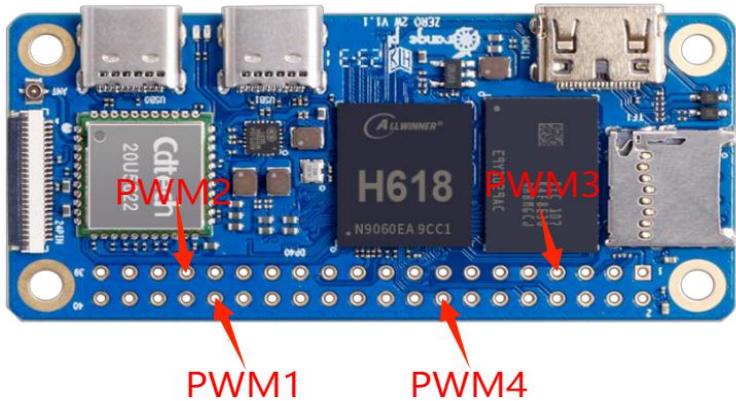


3. 20. wiringOP 硬件 PWM 的使用方法

使用 wiringOP 硬件 PWM 功能前, 请先下载最新的 wiringOP 的源码然后编译安装下。v1.0.0 版本的 linux 镜像中预装的 wiringOP 是用不了硬件 PWM 功能的。

下载安装 wiringOP 的方法请参考[安装 wiringOP 的方法](#)一小节的说明。

开发板最多可以使用 4 通道 PWM, 它们所在引脚的位置如下图所示:





3.20.1. 使用 wiringOP 的 gpio 命令设置 PWM 的方法

3.20.1.1. 设置对应引脚为 PWM 模式

1) 4 个 PWM 引脚与 wPi 序号对应关系如下表所示:

PWM 引脚	wPi 序号
PWM1	21
PWM2	22
PWM3	2
PWM4	9

2) 设置引脚为 PWM 模式的命令如下，以 PWM1 为例，其中第三个参数需要输入 PWM1 引脚对应的 wPi 的序号

```
orangepi@orangepi:~$ gpio mode 21 pwm
```

3) 引脚设置为 PWM 模式后，默认会输出一个频率为 23475Hz，占空比为 50% 的方波，此时，我们使用示波器测量对应的 PWM 引脚，就可以看到下面的波形



3.20.1.2. 调节 PWM 占空比的方法

1) PWM 占空比的计算公式如下所示，我们可以通过设置 CCR 和 ARR 的值来调节



PWM 占空比

$$\text{PWM 占空比} = \text{CCR} / \text{ARR}$$

其中：

CCR 的取值范围是 0~65535，默认值是 512。

ARR 的取值范围是 1~65536，默认值是 1024。

需要注意的是，我们设置的 CCR 值需要小于 ARR，因为占空比不能大于 1，当设置 CCR > ARR 时，会提示如下错误信息：

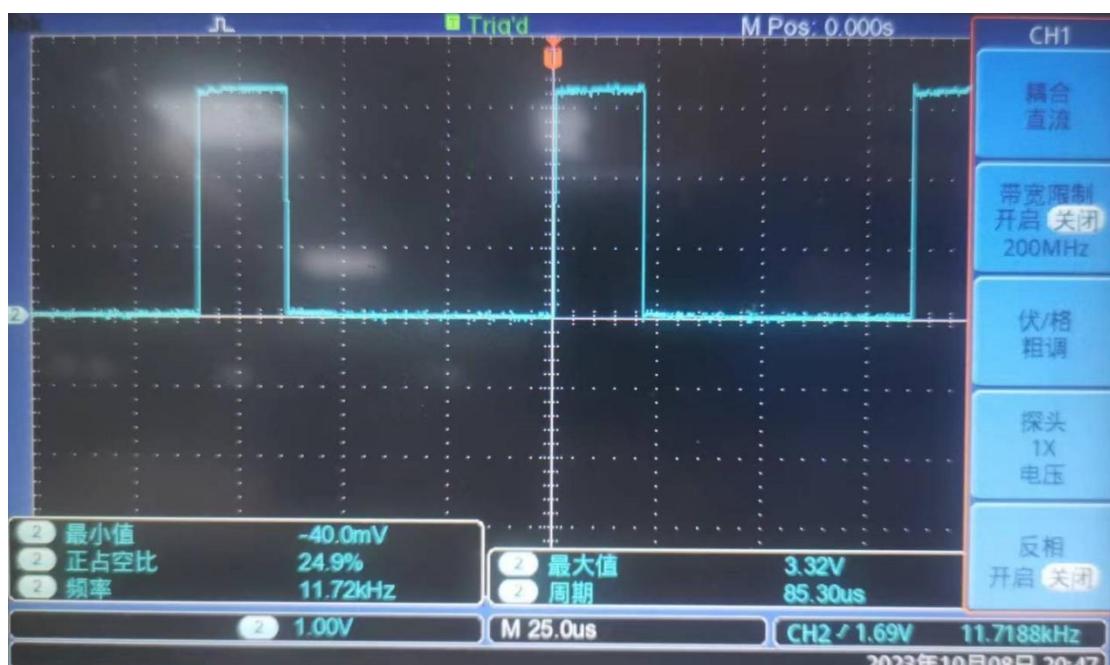
val pwmWrite 0 <= X <= 1024

Or you can set new range by yourself by pwmSetRange(range)

2) 我们可以使用下面的命令设置 PWM1 引脚的 ARR 为 2048

```
orangeipi@orangeipi:~$ gpio pwmr 21 2048
```

3) 运行上面的命令后，通过示波器可以观察到 PWM 占空比从默认的 50% (512/1024) 变为 25% (512/2048)

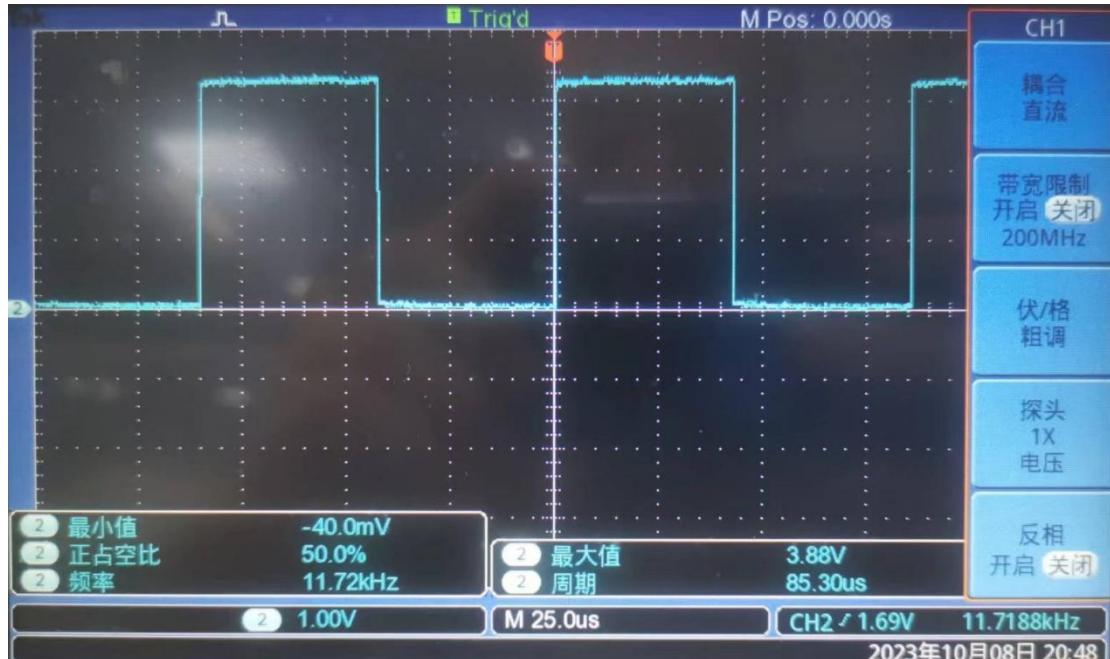


4) 我们可以使用下面的命令设置 PWM1 引脚的 CCR 为 1024



```
orangeipi@orangeipi:~$ gpio pwm 21 1024
```

5) 运行上面的命令后, 通过示波器可以观察到 PWM 占空比从 25% (512/2048) 变为 50% (1024/2048)



3.20.1.3. 调节 PWM 频率的方法

3.20.1.3.1. 通过设置分频系数来调节 PWM 频率的方法

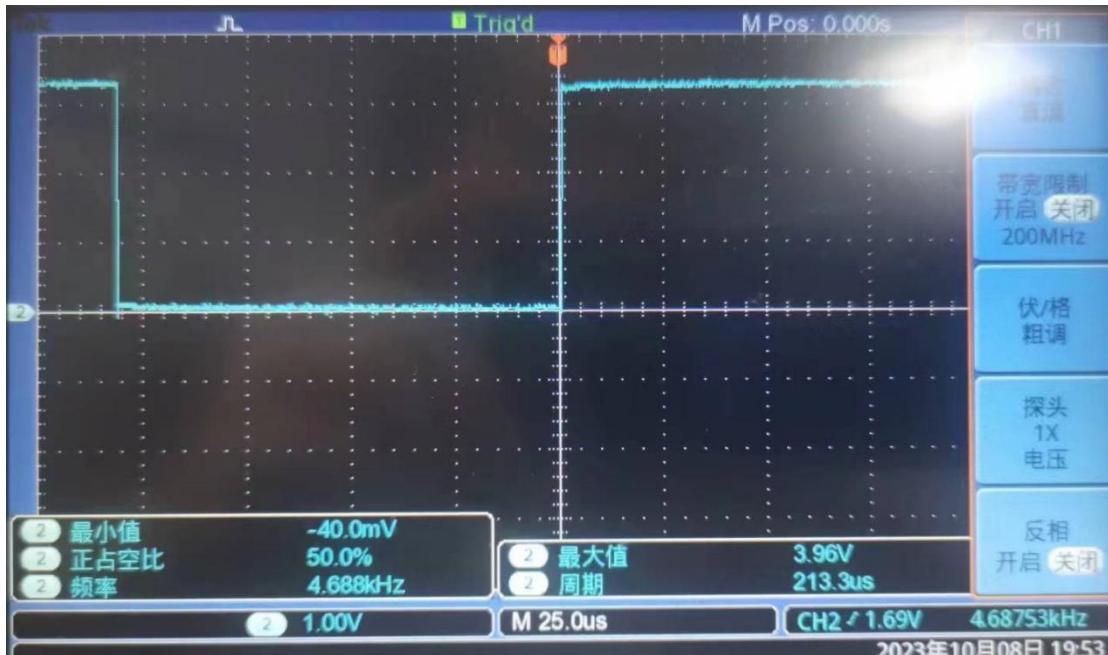
1) 设置分频系数后, PWM 频率会变为分频系数分之一

分频系数的取值范围是 1~256, 默认是 1。

2) 比如可以使用下面的命令设置 PWM1 引脚的分频系数为 5

```
orangeipi@orangeipi:~$ gpio pwmc 21 5
```

3) PWM 默认频率为 23475Hz, 经过 5 分频后计算值为 4695Hz, 通过示波器可以观察到 PWM 频率实际值为 4688Hz, 误差可以忽略



3.20.1.3.2. 直接设置 PWM 频率的方法

1) 我们可以使用 **gpio pwmTone** 命令来设置 PWM 引脚的频率，比如使用下面的命令可以设置 PWM1 引脚的 PWM 频率为 20000Hz

```
orangepi@orangepi:~$ gpio pwmTone 21 20000
```

在设置 PWM 频率时，需要保证：

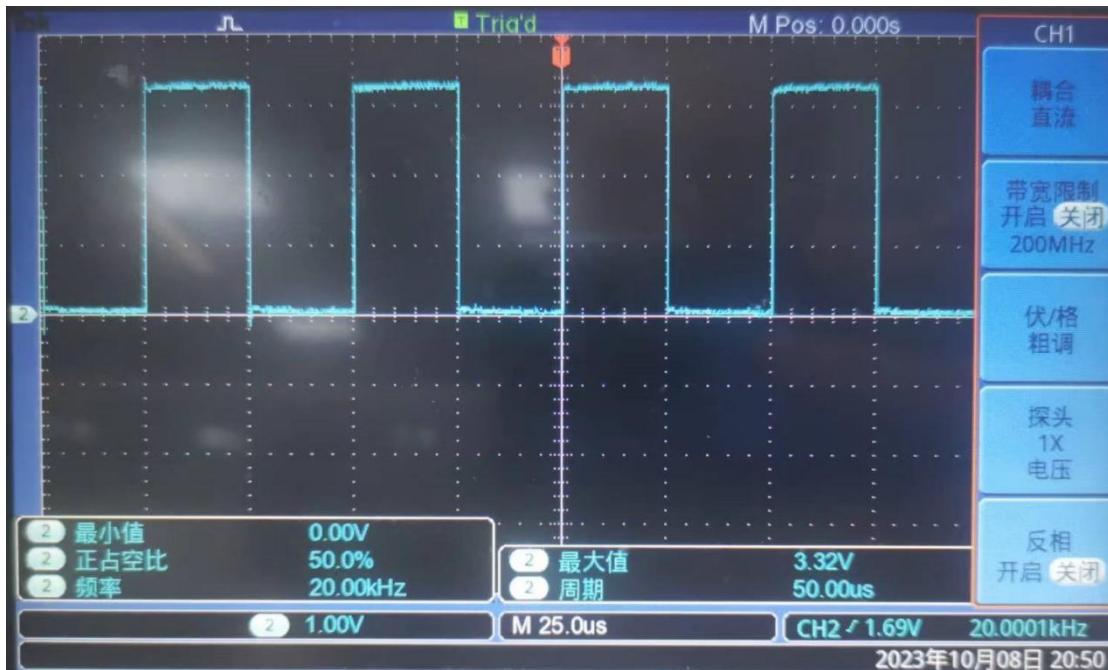
设置的频率值 $> 24000000 / (65536 * \text{分频系数})$ 。

比如，默认的分频系数为 1，在没有修改分频系数的情况下，设置的频率值应大于 366。

如果设置值过小，会出现如下报错：

```
gpio: range must be between 1 and 65536
```

2) 然后通过示波器可以观察到 PWM 频率变为 20000Hz 了



3.20.2. PWM 测试程序的使用方法

- 1) 在 wiringOP 的 example 目录下，有一个名为 pwm.c 的程序，此程序演示了使用 wiringOP 中 PWM 相关的 API 来操作 PWM 的方法

```
orangeipi@orangeipi:~/wiringOP/examples$ cd wiringOP/examples/
orangeipi@orangeipi:~/wiringOP/examples$ ls pwm.c
pwm.c
```

- 2) 编译 **pwm.c** 为可执行程序的命令如下所示

```
sorangeipi@orangeipi:~/wiringOP/examples$ gcc -o pwm pwm.c -lwiringPi
```

- 3) 然后就可以执行 PWM 测试程序了，在执行 PWM 测试程序时，需要指定 PWM 引脚，比如可以使用下面的命令对 PWM1 引脚进行测试：

```
sorangeipi@orangeipi:~/wiringOP/examples$ sudo ./pwm 21
```

- 4) pwm 程序执行后会对以下内容依次进行测试：

- 通过设置 ARR 调节 PWM 占空比
- 通过设置 CCR 调节 PWM 占空比
- 通过设置分频系数调节 PWM 频率
- 直接设置 PWM 频率



5) 在每完成一项测试后，会停止输出 pwm 波形 5 秒钟，在完成所有测试内容后，会重新开始新一轮测试

6) PWM 测试程序的详细执行过程如下所示：

- a. 通过设置 ARR 调节 PWM 占空比：通过示波器可以观察到 PWM 波形每隔 0.5 秒产生变化，在变化了 8 次后，PWM 占空比从 50% 变为 25%，保持 5 秒，然后 PWM 波形每隔 0.5 秒产生变化，在变化了 8 次后，PWM 占空比从 25% 变为 50%，保持 5 秒。
- b. 通过设置 CCR 调节 PWM 占空比：通过示波器可以观察到 PWM 波形每隔 0.5 秒产生变化，在变化了 8 次后，PWM 占空比从 50% 变为 100%，保持 5 秒，然后 PWM 波形每隔 0.5 秒产生变化，在变化了 8 次后，PWM 占空比从 100% 变为 50%，保持 5 秒。
- c. 通过设置分频系数调节 PWM 频率：通过示波器可以观察到 PWM 波形每隔 0.5 秒产生变化，在变化了 9 次后，PWM 频率变为默认 PWM 频率的 1/10，即 2347Hz，保持 5 秒，然后 PWM 波形每隔 0.5 秒产生变化，在变化了 9 次后，PWM 频率变为默认 PWM 频率，即 23475Hz，保持 5 秒。
- d. 直接设置 PWM 频率：通过示波器可以观察到 PWM 频率首先变为 2000Hz，然后每隔两秒 PWM 频率增加 2000Hz，在变化了 9 次后，PWM 频率变为 20000Hz，保持 5 秒。

3. 21. wiringOP-Python 的安装使用方法

注意：40pin 接口上的排针默认是不焊接的，需要自己焊接上去才能使用。

wiringOP-Python 是 wiringOP 的 Python 语言版本的库，用于在 Python 程序中操作开发板的 GPIO、I2C、SPI 和 UART 等硬件资源。

另外请注意下面所有的命令都是在 **root** 用户下操作的。

3. 21. 1. wiringOP-Python 的安装方法

1) 首先安装依赖包

```
root@orangepi:~# sudo apt-get update  
root@orangepi:~# sudo apt-get -y install git swig python3-dev python3-setuptools
```

2) 然后使用下面的命令下载 wiringOP-Python 的源码



注意，下面的 **git clone--recursive** 命令会自动下载 **wiringOP** 的源码，因为 **wiringOP-Python** 是依赖 **wiringOP** 的。请确保下载过程没有因为网络问题而报错。

如果从 **GitHub** 下载代码有问题，可以直接使用 **Linux** 镜像中自带的 **wiringOP-Python** 源码，存放位置为：**/usr/src/wiringOP-Python**。

```
root@orangepi:~# git clone --recursive https://github.com/orangepi-xunlong/wiringOP-Python -b next
root@orangepi:~# cd wiringOP-Python
root@orangepi:~/wiringOP-Python# git submodule update --init --remote
```

3) 然后使用下面的命令编译 **wiringOP-Python** 并将其安装到开发板的 **Linux** 系统中

```
root@orangepi:~# cd wiringOP-Python
root@orangepi:~/wiringOP-Python# python3 generate-bindings.py > bindings.i
root@orangepi:~/wiringOP-Python# sudo python3 setup.py install
```

4) 然后输入下面的命令，如果有帮助信息输出，说明 **wiringOP-Python** 安装成功，按下 **q** 键可以退出帮助信息的界面

```
root@orangepi:~/wiringOP-Python# python3 -c "import wiringpi; help(wiringpi)"
Help on module wiringpi:
```

NAME

```
wiringpi
```

DESCRIPTION

```
# This file was automatically generated by SWIG (http://www.swig.org).
# Version 4.0.2
#
# Do not make changes to this file unless you know what you are doing--modify
# the SWIG interface file instead.
```

5) 在 **python** 命令行下测试 **wiringOP-Python** 是否安装成功的步骤如下所示：

a. 首先使用 **python3** 命令进入 **python3** 的命令行模式

```
root@orangepi:~# python3
```

b. 然后导入 **wiringpi** 的 **python** 模块

```
>>> import wiringpi;
```

c. 最后输入下面的命令可以查看下 **wiringOP-Python** 的帮助信息，按下 **q** 键可



以退出帮助信息的界面

```
>>> help(wiringpi)
```

```
Help on module wiringpi:
```

NAME

```
wiringpi
```

DESCRIPTION

```
# This file was automatically generated by SWIG (http://www.swig.org).  
# Version 4.0.2  
#  
# Do not make changes to this file unless you know what you are doing--modify  
# the SWIG interface file instead.
```

CLASSES

```
builtins.object  
    GPIO  
    I2C  
    Serial  
    nes
```

```
class GPIO(builtins.object)  
    |   GPIO(pinmode=0)  
    |
```

```
>>>
```

3.21.2. 40pin GPIO 口测试

wiringOP-Python 跟 wiringOP 一样，也是可以通过指定 wPi 号来确定操作哪一个 GPIO 引脚，因为 wiringOP-Python 中没有查看 wPi 号的命令，所以只能通过 wiringOP 中的 gpio 命令来查看板子 wPi 号与物理引脚的对应关系。



ZERO2W													
GPIO	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	GPIO			
		3.3V			1	2		5V					
264	0	SDA.1	OFF	0	3	4		5V					
263	1	SCL.1	OFF	0	5	6		GND					
269	2	PWM3	OFF	0	7	8	0	ALT2	TXD.0	3	224		
		GND			9	10	0	ALT2	RXD.0	4	225		
226	5	TXD.5	OFF	0	11	12	0	OFF	PI01	6	257		
227	7	RXD.5	OFF	0	13	14			GND				
261	8	TXD.2	OFF	0	15	16	0	OFF	PWM4	9	270		
		3.3V			17	18	0	OFF	PH04	10	228		
231	11	MOSI.1	OFF	0	19	20			GND				
232	12	MISO.1	OFF	0	21	22	0	OFF	RXD.2	13	262		
230	14	SCLK.1	OFF	0	23	24	0	OFF	CE.0	15	229		
		GND			25	26	0	ALT3	CE.1	16	233		
266	17	SDA.2	OFF	0	27	28	0	OFF	SCL.2	18	265		
256	19	PI00	OFF	0	29	30			GND				
271	20	PI15	OFF	0	31	32	0	OFF	PWM1	21	267		
268	22	PI12	OFF	0	33	34			GND				
258	23	PI02	OFF	0	35	36	1	OUT	PC12	24	76		
272	25	PI16	OFF	0	37	38	0	OFF	PI04	26	260		
		GND			39	40	0	OFF	PI03	27	259		
ZERO2W													
orangepi@orangepirzero2w:~\$													

- 1) 下面以 7 号引脚——对应 GPIO 为 PI13 ——对应 wPi 序号为 2——为例演示如何设置 GPIO 口的高低电平

ZERO2W													
GPIO	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	GPIO			
		3.3V			1	2		5V					
264	0	SDA.1	OFF	0	3	4		5V					
263	1	SCL.1	OFF	0	5	6		GND					
269	2	PWM3	OFF	0	7	8	0	ALT2	TXD.0	3	224		
		GND			9	10	0	ALT2	RXD.0	4	225		

- 2) 直接用命令测试的步骤如下所示：

- 首先设置 GPIO 口为输出模式，其中 **pinMode** 函数的第一个参数是引脚对应的 wPi 的序号，第二个参数是 GPIO 的模式

```
root@orangepi:~/wiringOP-Python# python3 -c "import wiringpi;
from wiringpi import GPIO; wiringpi.wiringPiSetup();
wiringpi.pinMode(2, GPIO.OUTPUT);"
```

- 然后设置 GPIO 口输出低电平，设置完后可以使用万用表测量引脚的电压的数值，如果为 0v，说明设置低电平成功



```
root@orangepi:~/wiringOP-Python# python3 -c "import wiringpi; \
from wiringpi import GPIO; wiringpi.wiringPiSetup(); \
wiringpi.digitalWrite(2, GPIO.LOW)"
```

- c. 然后设置 GPIO 口输出高电平，设置完后可以使用万用表测量引脚的电压的数值，如果为 3.3v，说明设置高电平成功

```
root@orangepi:~/wiringOP-Python# python3 -c "import wiringpi; \
from wiringpi import GPIO; wiringpi.wiringPiSetup(); \
wiringpi.digitalWrite(2, GPIO.HIGH)"
```

3) 在 python3 的命令行中测试的步骤如下所示：

- a. 首先使用 python3 命令进入 python3 的命令行模式

```
root@orangepi:~# python3
```

- b. 然后导入 wiringpi 的 python 模块

```
>>> import wiringpi
>>> from wiringpi import GPIO
```

- c. 然后设置 GPIO 口为输出模式，其中 **pinMode** 函数的第一个参数是引脚对应的 wPi 的序号，第二个参数是 GPIO 的模式

```
>>> wiringpi.wiringPiSetup()
0
>>> wiringpi.pinMode(2, GPIO.OUTPUT)
```

- d. 然后设置 GPIO 口输出低电平，设置完后可以使用万用表测量引脚的电压的数值，如果为 0v，说明设置低电平成功

```
>>> wiringpi.digitalWrite(2, GPIO.LOW)
```

- e. 然后设置 GPIO 口输出高电平，设置完后可以使用万用表测量引脚的电压的数值，如果为 3.3v，说明设置高电平成功

```
>>> wiringpi.digitalWrite(2, GPIO.HIGH)
```

4) wiringOP-Python 在 python 代码中设置 GPIO 高低电平的方法可以参考下 examples 中的 **blink.py** 测试程序，**blink.py** 测试程序会设置开发板 40 Pin 中所有的 GPIO 口的电压不断的高低变化

```
root@orangepi:~/wiringOP-Python# cd examples
root@orangepi:~/wiringOP-Python/examples# ls blink.py
blink.py
root@orangepi:~/wiringOP-Python/examples# python3 blink.py
```



3.21.3. 40pin SPI 测试

1) 由下表可知，40pin 接口可用的 spi 为 spi1，有两个片选引脚 cs0 和 cs1

GPIO序号	GPIO	功能	引脚
		3.3V	1
264	PI8	TWI1-SDA	3
263	PI7	TWI1-SCL	5
269	PI13	PWM3/UART4_TX	7
		GND	9
226	PH2	UART5_TX	11
227	PH3	UART5_RX	13
261	PI5	TWI0_SCL/UART2_TX	15
		3.3V	17
231	PH7	SPI1_MOSI	19
232	PH8	SPI1_MISO	21
230	PH6	SPI1_CLK	23
		GND	25
266	PI10	TWI2-SDA/UART3_RX	27
256	PI0		29
271	PI15		31
268	PI12	PWM2	33
258	PI2		35
272	PI16		37
		GND	39

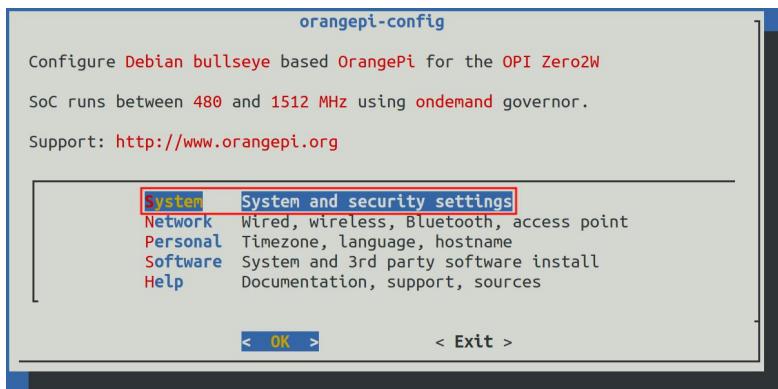
引脚	功能	GPIO	GPIO序号
2	5V		
4	5V		
6	GND		
8	UART0_TX	PH0	224
10	UART0_RX	PH1	225
12		PI1	257
14	GND		
16	PWM4/UART4_RX	PI14	270
18		PH4	228
20	GND		
22	TWI0_SDA/UART2_RX	PI6	262
24	SPI1_CS0	PH5	229
26	SPI1_CS1	PH9	233
28	TWI2-SCL/UART3_TX	PI9	265
30	GND		
32	PWM1	PI11	267
34	GND		
36		PC12	76
38		PI4	260
40		PI3	259

2) Linux 系统中 spi1 默认是关闭的，需要手动打开才能使用。打开步骤如下所示：

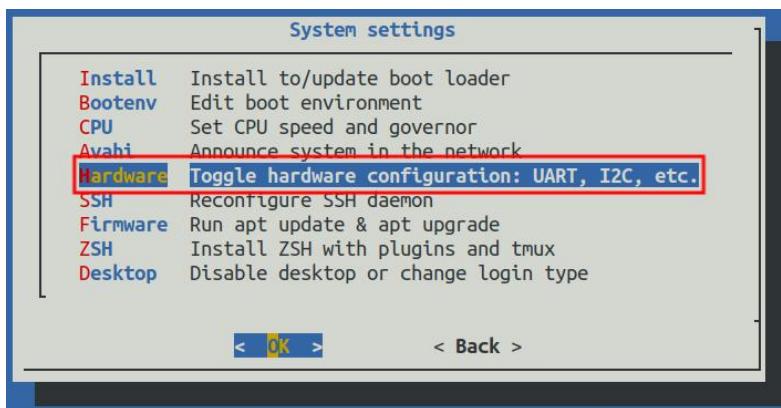
a. 首先运行下 **orangepi-config**，普通用户记得加 **sudo** 权限

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

b. 然后选择 **System**



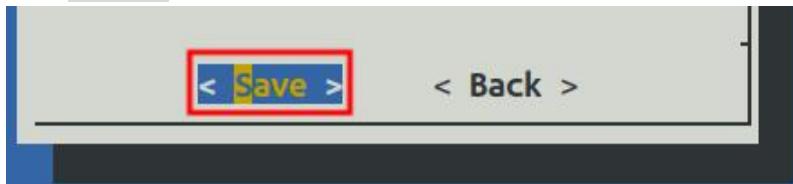
c. 然后选择 **Hardware**



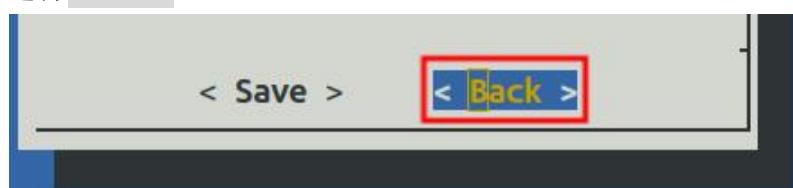
d. 然后使用键盘的方向键定位到下图所示的位置,再使用空格选中想要打开的 SPI 的 dtbo 配置

dtbo 配置	说明
spi1-cs0-cs1-spidev	同时打开 spi1 的 cs0 和 cs1
spi1-cs0-spidev	只打开 spi1 的 cs0
spi1-cs1-spidev	只打开 spi1 的 cs1

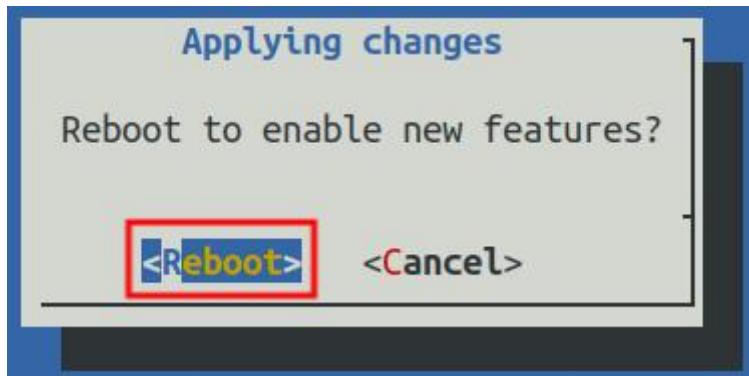
e. 然后选择<Save>保存



f. 然后选择<Back>



g. 然后选择<Reboot>重启系统使配置生效



3) 然后查看下 linux 系统中是否存在 **spidev1.x** 的设备节点，如果存在，说明 SPI1 的配置已经生效了

```
orangepi@orangepi:~$ ls /dev/spidev1*
/dev/spidev1.0  /dev/spidev1.1
```

注意，只有打开 spi1-cs0-cs1-spidev 时，才会看到两个 spi 的设备节点。

4) 然后可以使用 examples 中的 `spidev_test.py` 程序测试下 SPI 的回环功能，`spidev_test.py` 程序需要指定下面的两个参数：

- a. **--channel:** 指定 SPI 的通道号
 - b. **--port:** 指定 SPI 的端口号

5) 先不短接 SPI1 的 mosi 和 miso 两个引脚, 运行 spidev_test.py 的输出结果如下所示, 可以看到 TX 和 RX 的数据不一致



6) 然后使用杜邦线短接 SPI1 的 txd (40pin 接口中的第 19 号引脚) 和 rxd (40pin 接口中的第 21 号引脚) 两个引脚再运行 spidev_test.py 的输出如下，可以看到发送和接收的数据一样，说明 SPI1 回环测试正常

```
root@orangepi:~/wiringOP-Python# cd examples
root@orangepi:~/wiringOP-Python/examples# python3 spidev_test.py \
--channel 1 --port 0
spi mode: 0x0
max speed: 500000 Hz (500 KHz)
Opening device /dev/spidev1.1
TX | FF FF FF FF FF FF 40 00 00 00 00 95 FF F0 0D  |.....@.....|
RX | FF FF FF FF FF FF 40 00 00 00 00 95 FF F0 0D  |.....@.....|
```

3.21.4. 40pin I2C 测试

1) 由下表可知，40pin 接口可用的 i2c 为 i2c0、i2c1 和 i2c2

GPIO序号	GPIO	功能	引脚
		3.3V	1
264	PI8	TWI1-SDA	3
263	PI7	TWI1-SCL	5
269	PI13	PWM3/UART4_TX	7
		GND	9
226	PH2	UART5_TX	11
227	PH3	UART5_RX	13
261	PI5	TWI0_SCL/UART2_TX	15
		3.3V	17
231	PH7	SPI1_MOSI	19
232	PH8	SPI1_MISO	21
230	PH6	SPI1_CLK	23
		GND	25
266	PI10	TWI2-SDA/UART3_RX	27
256	PI0		29
271	PI15		31
268	PI12	PWM2	33
258	PI2		35

引脚	功能	GPIO	GPIO序号
2	5V		
4	5V		
6	GND		
8	UART0_TX	PH0	224
10	UART0_RX	PH1	225
12		PI1	257
14	GND		
16	PWM4/UART4_RX	PI14	270
18		PH4	228
20	GND		
22	TWI0_SDA/UART2_RX	PI6	262
24	SPI1_CS0	PH5	229
26	SPI1_CS1	PH9	233
28	TWI2-SCL/UART3_TX	PI9	265
30	GND		
32	PWM1	PI11	267
34	GND		
36		PC12	76



272	PI16		37
		GND	39

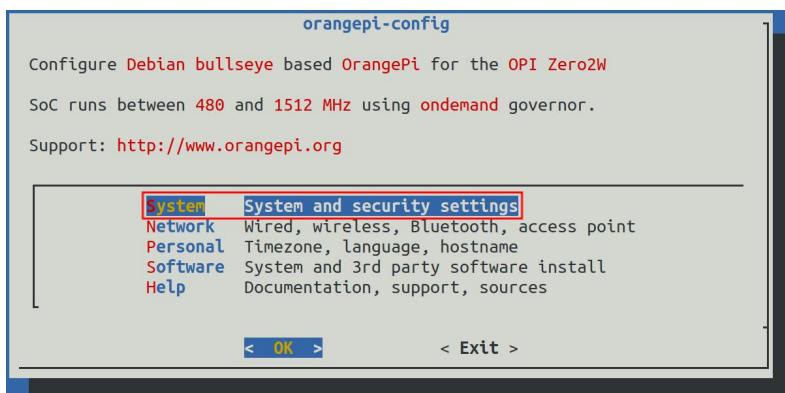
38		PI4	260
40		PI3	259

2) Linux 系统中 i2c 默认都是关闭的，需要手动打开才能使用。打开步骤如下所示：

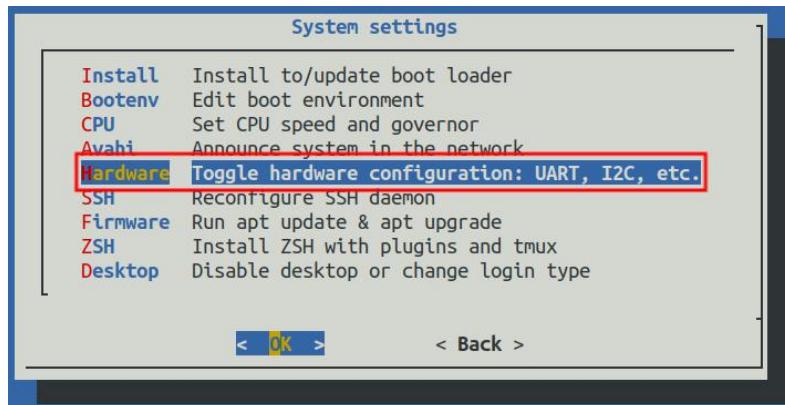
- a. 首先运行下 **orangepi-config**，普通用户记得加 **sudo** 权限

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

- b. 然后选择 **System**

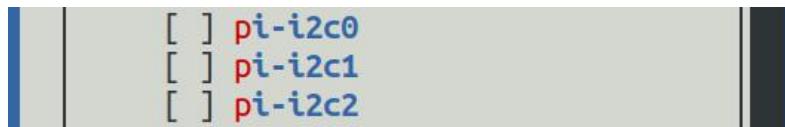


- c. 然后选择 **Hardware**



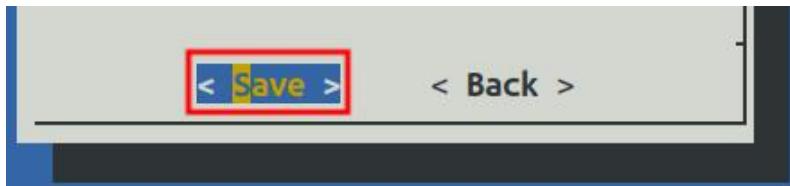
- d. 然后使用键盘的方向键定位到下图所示的位置，再使用空格选中下图中对应的 i2c 的配置

40pin 中的复用功能	对应的 dtbo 配置
40pin - i2c0	pi-i2c0
40pin - i2c1	pi-i2c1
40pin - i2c2	pi-i2c2

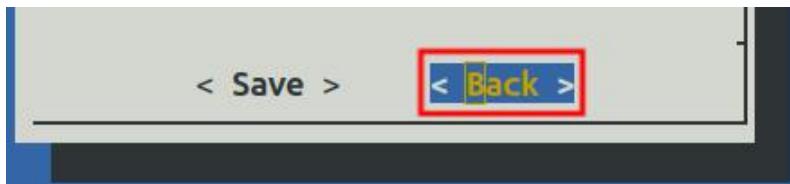




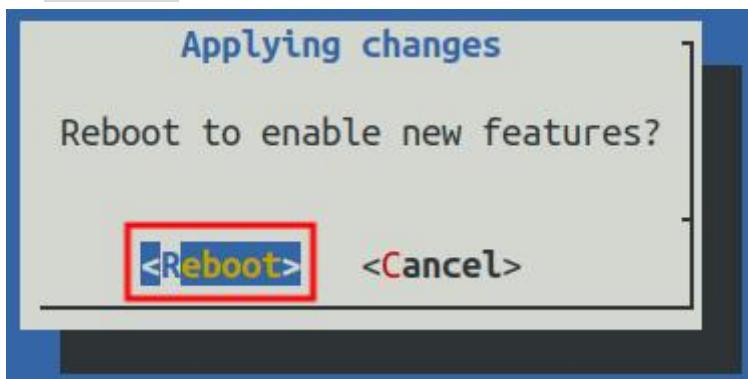
e. 然后选择<Save>保存



f. 然后选择<Back>



g. 然后选择<Reboot>重启系统使配置生效



3) 启动 linux 系统后，先确认下/dev 下存在已打开的 i2c 的设备节点

```
orangepi@orangepi:~$ ls /dev/i2c-*  
/dev/i2c-*
```

有时 i2c 的设备节点和 i2c 总线的序号不是一一对应的，比如 i2c1 总线的 i2c 设备节点可能是 /dev/i2c-3。
准确确认 i2c 总线对应的 /dev 下设备节点方法为：

c. 首先运行下面的命令，查看 i2c 的对应关系

```
orangepi@orangepi:~$ ls /sys/devices/platform/soc*/*/i2c-* | grep "i2c-[0-9]"  
/sys/devices/platform/soc/5002000.i2c/i2c-0:  
/sys/devices/platform/soc/5002400.i2c/i2c-3:  
/sys/devices/platform/soc/5002800.i2c/i2c-4:  
/sys/devices/platform/soc/5002c00.i2c/i2c-5:  
/sys/devices/platform/soc/6000000.hdmi/i2c-2:  
/sys/devices/platform/soc/7081400.i2c/i2c-1:
```



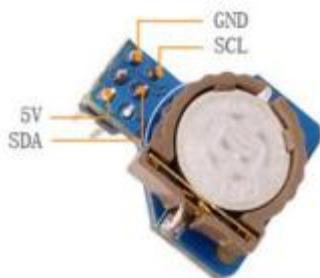
d. 上面的输出中

- d) 5002000 为 i2c0 总线的寄存器地址，其后面显示 i2c-0 就是其对应的 i2c 设备节点
- e) 5002400 为 i2c1 总线的寄存器地址，其后面显示 i2c-3 就是其对应的 i2c 设备节点
- f) 5002800 为 i2c2 总线的寄存器地址，其后面显示 i2c-4 就是其对应的 i2c 设备节点

4) 然后开始测试 i2c，首先安装下 i2c-tools

```
orangepi@orangepi:~$ sudo apt-get update  
orangepi@orangepi:~$ sudo apt-get install -y i2c-tools
```

5) 然后在 40pin 接头的 i2c 引脚上接一个 i2c 设备，这里以 DS1307 RTC 模块为例

6) 然后使用 **i2cdetect -y x** 命令如果能检测到连接的 i2c 设备的地址，就说明 i2c 设备连接正确

注意，**i2cdetect -y x** 命令中的 x 需要替换为 i2c 总线对应的设备节点的序号。

```
orangepi@orangepi:~$ sudo i2cdetect -y 3  
[sudo] password for orangepi:  
      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  
00:          - - - - - - - -  
10:          - - - - - - - -  
20:          - - - - - - - -  
30:          - - - - - - - -  
40:          - - - - - - - -  
50:          - - - - - - - -  
60:          - - - - - - 68 - - - - -  
70:          - - - - - - - -  
orangepi@orangepi:~$
```

7) 然后可以运行 examples 中的 **ds1307.py** 测试程序读取 RTC 的时间

注意，下面命令中 **i2c-x** 中的 x 需要替换为 i2c 总线对应的设备节点的序号。

```
root@orangepi:~/wiringOP-Python# cd examples  
root@orangepi:~/wiringOP-Python/examples# python3 ds1307.py --device \
```



```
"/dev/i2c-x"
Thu 2022-06-16 04:35:46
Thu 2022-06-16 04:35:47
Thu 2022-06-16 04:35:48
^C
exit
```

3.21.5. 40pin 的 UART 测试

1) 由下表可知，可用的 uart 为 uart2、uart3、uart4 和 uart5。请注意 uart0 默认设置为调试串口，请不要把 uart0 当成普通串口使用

GPIO序号	GPIO	功能	引脚
		3.3V	1
264	PI8	TWI1-SDA	3
263	PI7	TWI1-SCL	5
269	PI13	PWM3/ UART4_TX	7
		GND	9
226	PH2	UART5_TX	11
227	PH3	UART5_RX	13
261	PI5	TWI0_SCL/ UART2_TX	15
		3.3V	17
231	PH7	SPI1_MOSI	19
232	PH8	SPI1_MISO	21
230	PH6	SPI1_CLK	23
		GND	25
266	PI10	TWI2-SDA/ UART3_RX	27
256	PI0		29
271	PI15		31
268	PI12	PWM2	33
258	PI2		35
272	PI16		37
		GND	39

引脚	功能	GPIO	GPIO序号
2	5V		
4	5V		
6	GND		
8	UART0_TX	PH0	224
10	UART0_RX	PH1	225
12		PI1	257
14	GND		
16	PWM4/ UART4_RX	PI14	270
18		PH4	228
20	GND		
22	TWI0_SDA/ UART2_RX	PI6	262
24	SPI1_CS0	PH5	229
26	SPI1_CS1	PH9	233
28	TWI2-SCL/ UART3_TX	PI9	265
30	GND		
32	PWM1	PI11	267
34	GND		
36		PC12	76
38		PI4	260
40		PI3	259

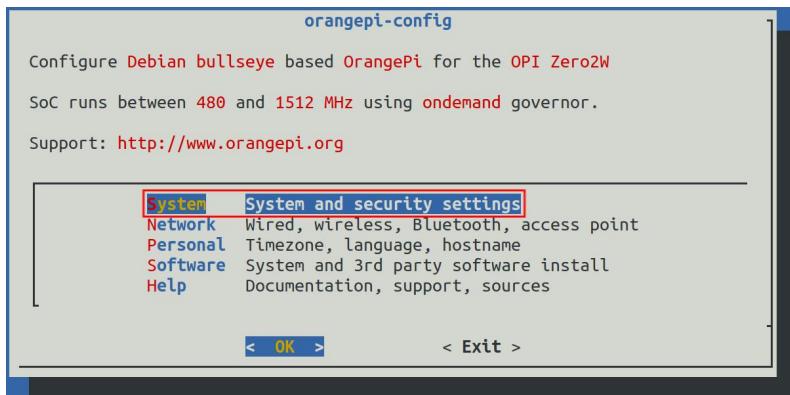
2) Linux 系统中 uart 默认都是关闭的，需要手动打开才能使用。打开步骤如下所示：

a. 首先运行下 **orangepi-config**，普通用户记得加 **sudo** 权限

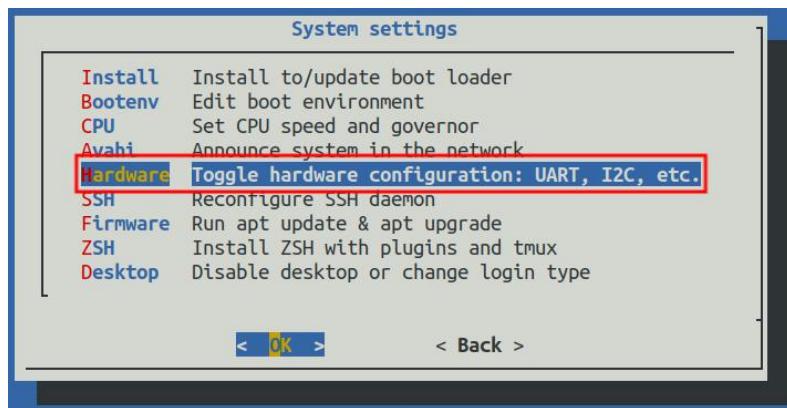
```
orangepi@orangepi:~$ sudo orangepi-config
```



b. 然后选择 **System**

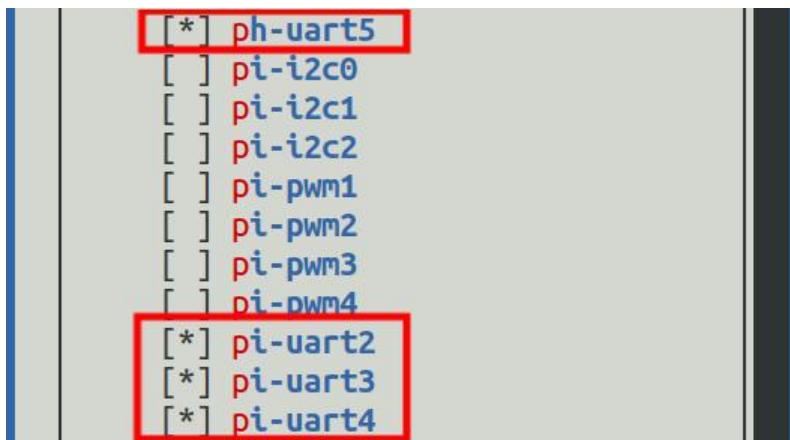


c. 然后选择 **Hardware**

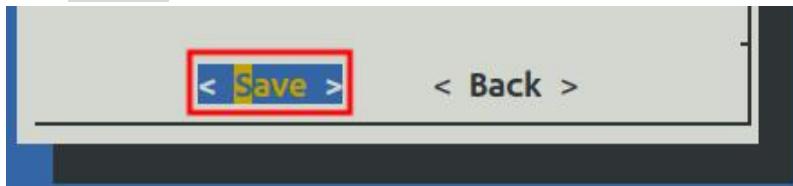


d. 然后使用键盘的方向键定位到下图所示的位置,再使用空格选中想要打开的串口

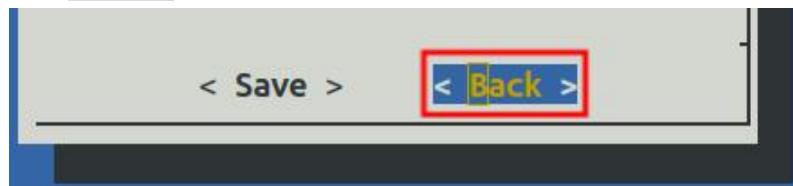
40pin 中的复用功能	对应的 dtbo 配置
40pin - uart2	pi-uart2
40pin - uart3	pi-uart3
40pin - uart4	pi-uart4
40pin - uart5	ph-uart5



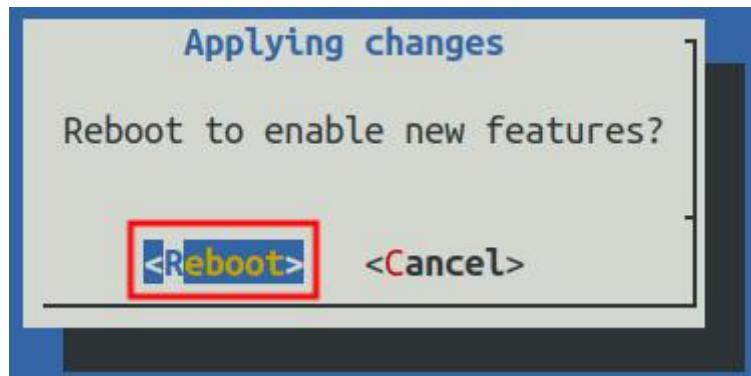
- e. 然后选择<Save>保存



- f. 然后选择<Back>



- g. 然后选择<Reboot>重启系统使配置生效



3) 进入 linux 系统后，先确认下/dev 下是否存在 uart5 的设备节点

注意，linux5.4 系统为/dev/ttysX。

```
orangeipi@orangeipi:~$ ls /dev/ttys*  
/dev/ttysX
```



- 4) 然后开始测试 uart 接口，先使用杜邦线短接要测试的 uart 接口的 rx 和 tx 引脚
- 5) 使用 wiringOP 中的 **gpio** 命令测试串口的回环功能如下所示，如果能看到下面的打印，说明串口通信正常

注意，**gpio serial /dev/ttYSx** 命令中最后的 x 需要替换为对应的 uart 设备节点的序号。

```
orangepi@orangepi:~$ gpio serial /dev/ttYSx      # linux-6.1 测试命令  
orangepi@orangepi:~$ gpio serial /dev/ttYSx      # linux-5.4 测试命令  
  
Out: 0: -> 0  
Out: 1: -> 1  
Out: 2: -> 2  
Out: 3: -> 3^C
```

- 6) 最后可以运行 examples 中的 **serialTest.py** 程序来测试下串口的回环功能，如果能看到下面的打印，说明串口回环测试正常

注意，命令中**/dev/ttYSx** 或**/dev/ttYSx** 中的 x 需要替换为对应的 uart 设备节点的序号。

```
root@orangepi:~/wiringOP-Python# cd examples  
root@orangepi:~/wiringOP-Python/examples# python3 serialTest.py --device "/dev/ttYSx"      # linux6.1 使用  
root@orangepi:~/wiringOP-Python/examples# python3 serialTest.py --device "/dev/ttYSx"      # linux5.4 使用  
  
Out: 0: -> 0  
Out: 1: -> 1  
Out: 2: -> 2  
Out: 3: -> 3  
Out: 4:^C  
exit
```

3.22. 硬件看门狗测试

Orange Pi 发布的 linux 系统中预装了 **watchdog_test** 程序，可以直接测试。

运行 **watchdog_test** 程序的方法如下所示：



- a. 第二个参数 10 表示看门狗的计数时间，如果这个时间内没有喂狗，系统会重启
- b. 我们可以通过按下键盘上的任意键（ESC 除外）来喂狗，喂狗后，程序会打印一行 keep alive 表示喂狗成功

```
orangepi@orangepi:~$ sudo watchdog_test 10
open success
options is 33152,identity is sunxi-wdt
put_usr return,if 0,success:0
The old reset time is: 16
return ENOTTY,if -1,success:0
return ENOTTY,if -1,success:0
put_user return,if 0,success:0
put_usr return,if 0,success:0
keep alive
keep alive
keep alive
```

3. 23. 查看 H618 芯片的 chipid

查看 H618 芯片 chipid 的命令如下所示，每个芯片的 chipid 都是不同的，所以可以使用 chipid 来区分多个开发板。

```
orangepi@orangepi:~$ cat /sys/class/sunxi_info/sys_info | grep "chipid"
sunxi_chipid      : 338020004c0048080147478824681ed1
```

3. 24. Python 相关说明

3. 24. 1. Python 源码编译安装的方法

如果使用的 Ubuntu 或者 Debian 系统软件仓库中的 Python 版本不符合开发的要求，想要使用最新版本的 Python，可以使用下面的方法下载 Python 的源码包来编译安装最新版本的 Python。

下面演示的是编译安装 Python3.9 的最新版本，如果要编译安装其他的版本的 Python，方法也是一样的（需要下载想要安装的 Python 对应的源码）。



1) 首先安装编译 Python 需要的依赖包

```
orangepi@orangepi:~$ sudo apt-get update  
orangepi@orangepi:~$ sudo apt-get install -y build-essential zlib1g-dev \\\nlibncurses5-dev libgdbm-dev libnss3-dev libssl-dev libsqlite3-dev \\\nlibreadline-dev libffi-dev curl libbz2-dev
```

2) 然后下载最新版本的 Python3.9 源码并解压

```
orangepi@orangepi:~$ wget \\\nhttps://www.python.org/ftp/python/3.9.10/Python-3.9.10.tgz  
orangepi@orangepi:~$ tar xvf Python-3.9.10.tgz
```

3) 然后运行配置命令

```
orangepi@orangepi:~$ cd Python-3.9.10  
orangepi@orangepi:~$ ./configure --enable-optimizations
```

4) 然后编译安装 Python3.9， 编译时间大概需要半个小时左右

```
orangepi@orangepi:~$ make -j4  
orangepi@orangepi:~$ sudo make altinstall
```

5) 安装完后可以使用下面的命令查看下刚安装的 Python 的版本号

```
orangepi@orangepi:~$ python3.9 --version  
Python 3.9.10
```

6) 然后更新下 pip

```
orangepi@orangepi:~$ /usr/local/bin/python3.9 -m pip install --upgrade pip
```

3. 24. 2. Python 更换 pip 源的方法

Linux 系统 pip 默认使用的源为 Python 官方的源，但是国内访问 Python 官方的源速度是很慢的，并且经常会由于网络原因导致 Python 软件包安装失败。所以在使用 pip 安装 Python 库时，请记得更换下 pip 源。

1) 首先安装下 **python3-pip**

```
orangepi@orangepi:~$ sudo apt-get update  
orangepi@orangepi:~$ sudo apt-get install -y python3-pip
```



2) Linux 下永久更换 pip 源的方法

- 先新建`~/.pip` 目录，然后添加 `pip.conf` 配置文件，并在其中设置 pip 的源为清华源

```
orangeipi@orangeipi:~$ mkdir -p ~/.pip
orangeipi@orangeipi:~$ cat <<EOF > ~/.pip/pip.conf
[global]
timeout = 6000
index-url = https://pypi.tuna.tsinghua.edu.cn/simple
trusted-host = pypi.tuna.tsinghua.edu.cn
EOF
```

- 然后使用 pip3 安装 Python 库速度就会很快了

3) Linux 下临时更换 pip 源的方法，其中的<packagename>需要替换为具体的包名

```
orangeipi@orangeipi:~$ pip3 install <packagename> -i \
https://pypi.tuna.tsinghua.edu.cn/simple --trusted-host pypi.tuna.tsinghua.edu.cn
```

3.25. 安装 Docker 的方法

Orange Pi 提供的 linux 镜像已经预装了 Docker，只是 Docker 服务默认没有打开。使用 `enable_docker.sh` 脚本可以使能 docker 服务，然后就可以开始使用 docker 命令了，并且在下次启动系统时也会自动启动 docker 服务。

```
orangeipi@orangeipi:~$ enable_docker.sh
```

可以使用下面的命令测试下 docker，如果能运行 `hello-world` 说明 docker 能正常使用了。

```
orangeipi@orangeipi:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
256ab8fe8778: Pull complete
Digest:
sha256:7f0a9f93b4aa3022c3a4c147a449ef11e0941a1fd0bf4a8e6c9408b2600777c5
Status: Downloaded newer image for hello-world:latest
```

Hello from Docker!

This message shows that your installation appears to be working correctly.



.....

使用 docker 命令时，如果提示 **permission denied**，请将当前用户加入到 docker 用户组，这样不需要 sudo 就能运行 docker 命令了。

```
orangeipi@orangeipi:~$ sudo usermod -aG docker $USER
```

注意：需要退出重新登录系统才能生效，重启系统也可以。

3. 26. Home Assistant 的安装方法

注意，这里只会提供在 Ubuntu 或者 Debian 系统中安装 Home Assistant 的方法，Home Assistant 详细的使用方法请参考官方文档或者相应的书籍。

3. 26. 1. 通过 docker 安装

1) 首先请安装好 docker，并确保 docker 能正常运行。docker 的安装步骤请参考[安装 Docker 的方法](#)一节的说明。

2) 然后可以搜索下 Home Assistant 的 docker 镜像

```
orangeipi@orangeipi:~$ docker search homeassistant
```

3) 然后使用下面的命令下载 Home Assistant 的 docker 镜像到本地，镜像大小大概有 1GB 多，下载时间会比较长，请耐心等待下载完成

```
orangeipi@orangeipi:~$ docker pull homeassistant/home-assistant
Using default tag: latest
latest: Pulling from homeassistant/home-assistant
be307f383ecc: Downloading
5fbc4c07ac88: Download complete
..... (省略部分输出)
3cc6a1510c9f: Pull complete
7a4e4d5b979f: Pull complete
Digest:
sha256:81d381f5008c082a37da97d8b08dd8b358dae7ecf49e62ce3ef1eeaefc4381bb
Status: Downloaded newer image for homeassistant/home-assistant:latest
docker.io/homeassistant/home-assistant:latest
```



4) 然后可以使用下面的命令查看下刚下载的 Home Assistant 的 docker 镜像

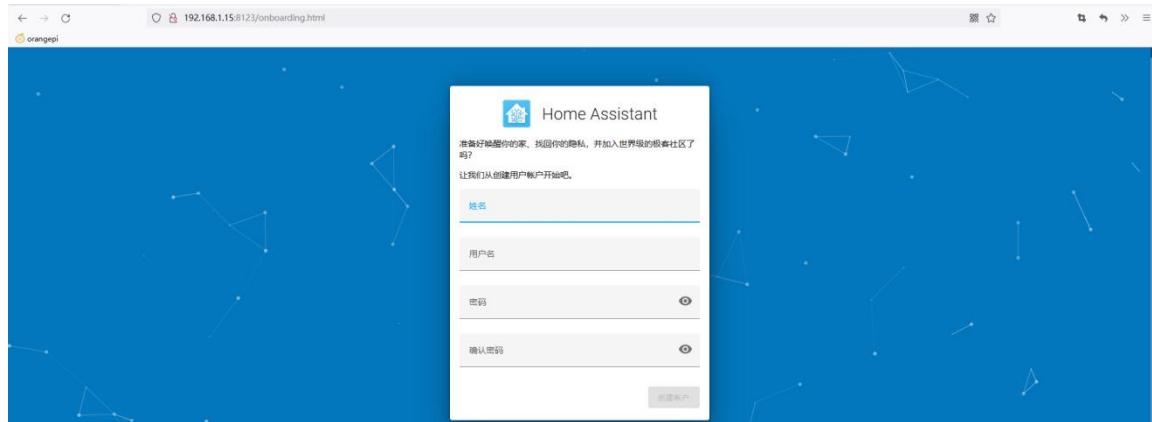
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
homeassistant/home-assistant	latest	bfa0ab9e1cf5	2 months ago	1.17GB

5) 此时就可以运行 Home Assistant 的 docker 容器了

```
orangeipi@orangeipi:~$ docker run -d \
--name homeassistant \
--privileged \
--restart=unless-stopped \
-e TZ=Asia/Shanghai \
-v /home/orangeipi/home-assistant:/config \
--network=host \
homeassistant/home-assistant:latest
```

6) 然后在浏览器中输入【开发板的 IP 地址:8123】就能看到 Home Assistant 的界面

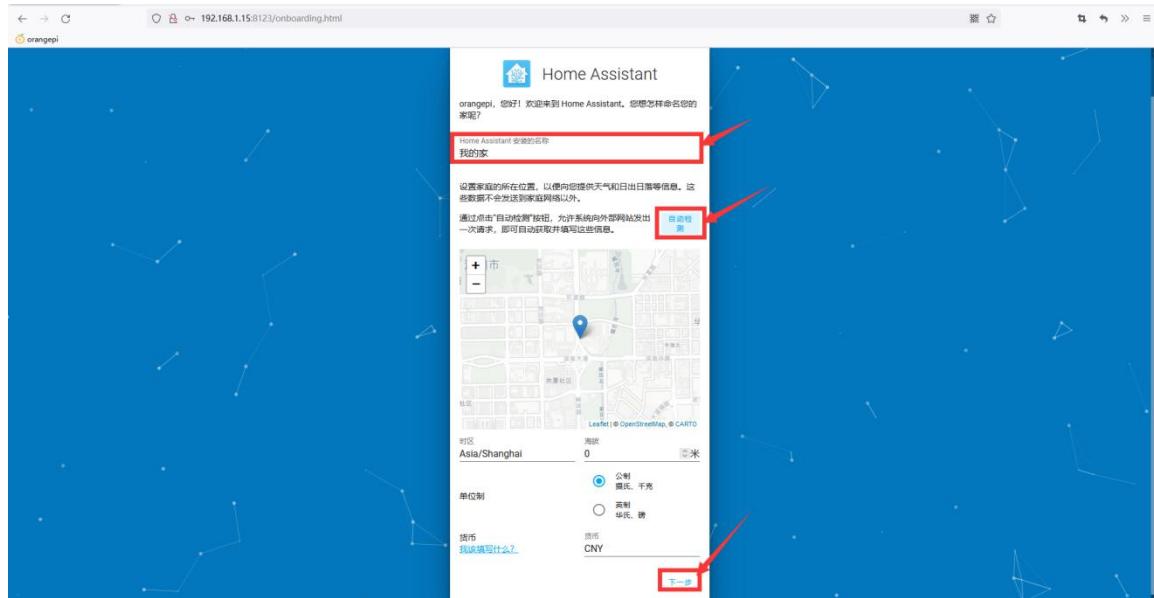
Home Assistant 容器的启动需要一段时间，如果下面的界面没有正常显示，请等待几秒钟再刷新。如果等待一分钟以上还没有正常显示下面的界面说明 Home Assistant 安装有问题，此时需要去检查前面的安装设置过程是否有问题了。



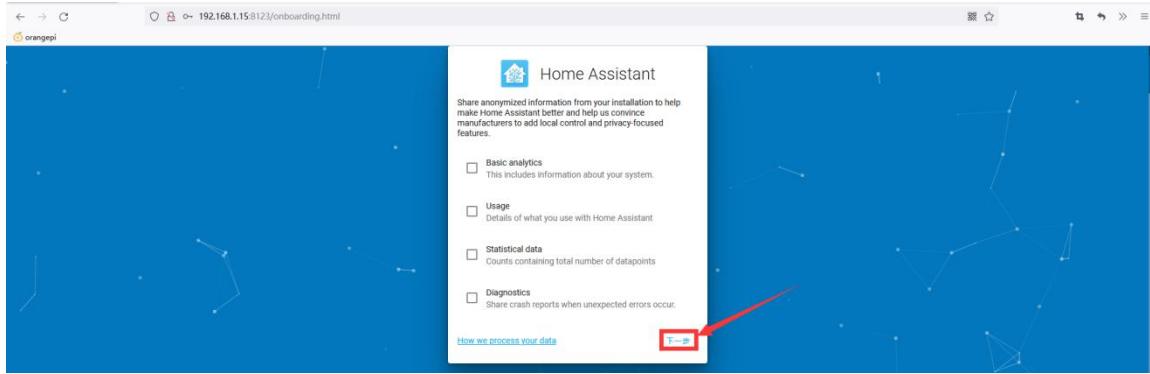
7) 然后输入姓名、用户名和密码再点击创建账号



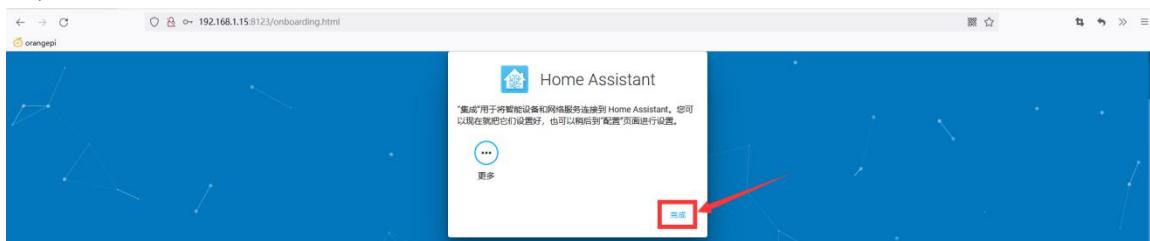
8) 然后按照界面提示根据自己的喜好设置，再点击下一步



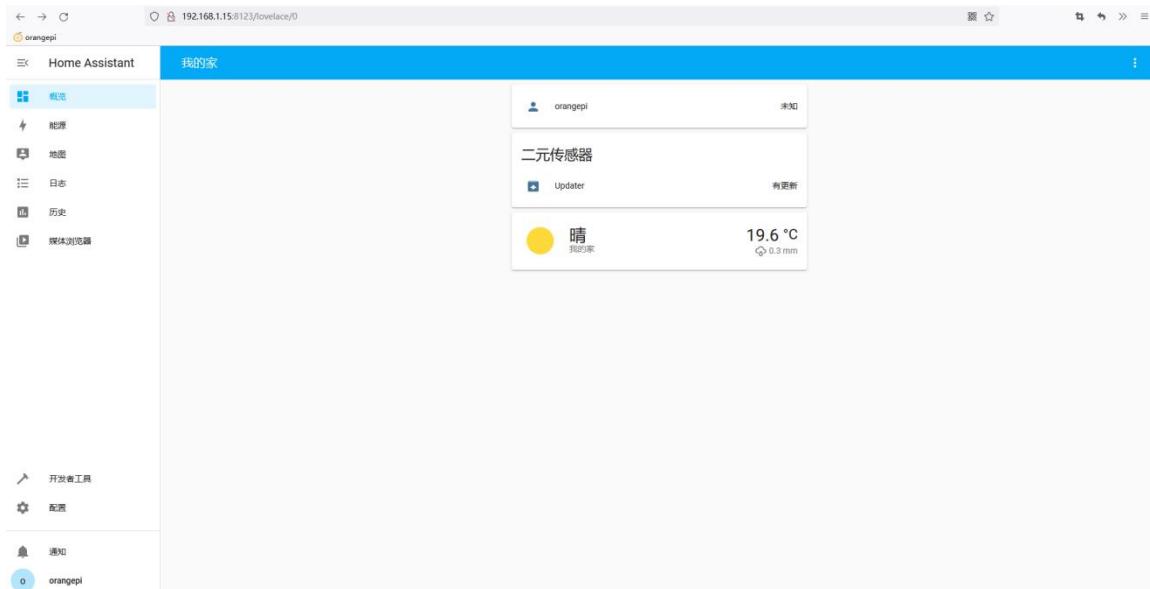
9) 然后点击下一步



10) 然后点击完成



11) Home Assistant 最终显示的主界面如下图所示



12) 停止 Home Assistant 容器的方法

a. 查看 docker 容器的命令如下所示

```
orangepi@orangepi:~$ docker ps -a
```

b. 停止 Home Assistant 容器的命令如下所示



```
orangeipi@orangeipi:~$ docker stop homeassistant
```

c. 删除 Home Assistant 容器的命令如下所示

```
orangeipi@orangeipi:~$ docker rm homeassistant
```

3. 26. 2. 通过 python 安装

安装前请先更换下 pip 的源为国内源，加快 Python 包的安装速度，配置方法见[Python 更换 pip 源的方法](#)一节的说明。

1) 首先安装依赖包

```
orangeipi@orangeipi:~$ sudo apt-get update  
orangeipi@orangeipi:~$ sudo apt-get install -y python3 python3-dev python3-venv \\\npython3-pip libffi-dev libssl-dev libjpeg-dev zlib1g-dev autoconf build-essential \\\nlibopenjp2-7 libtiff5 libturbojpeg0-dev tzdata
```

如果是 debian12 请使用下面的命令：

```
orangeipi@orangeipi:~$ sudo apt-get update  
orangeipi@orangeipi:~$ sudo apt-get install -y python3 python3-dev python3-venv \\\npython3-pip libffi-dev libssl-dev libjpeg-dev zlib1g-dev autoconf build-essential \\\nlibopenjp2-7 libturbojpeg0-dev tzdata
```

2) 然后需要编译安装 Python3.9，方法请参考[Python 源码编译安装的方法](#)一节

Debian Bullseye 默认的 Python 版本就是 Python3.9，所以无需编译安装。

Ubuntu Jammy 默认的 Python 版本就是 Python3.10，所以也无需编译安装。

Debian Bookworm 默认的 Python 版本就是 Python3.11，所以也无需编译安装。

3) 然后创建 Python 虚拟环境

Debian Bookworm 中是 python3.11，请记得替换对应的命令。

```
orangeipi@orangeipi:~$ sudo mkdir /srv/homeassistant  
orangeipi@orangeipi:~$ sudo chown orangeipi:orangeipi /srv/homeassistant  
orangeipi@orangeipi:~$ cd /srv/homeassistant  
orangeipi@orangeipi:~$ python3.9 -m venv .  
orangeipi@orangeipi:~$ source bin/activate  
(homeassistant) orangeipi@orangeipi:/srv/homeassistant$
```



4) 然后安装需要的 Python 包

```
(homeassistant) orangepi@orangepi:/srv/homeassistant$ python3 -m pip install wheel
```

5) 然后就可以安装 Home Assistant Core

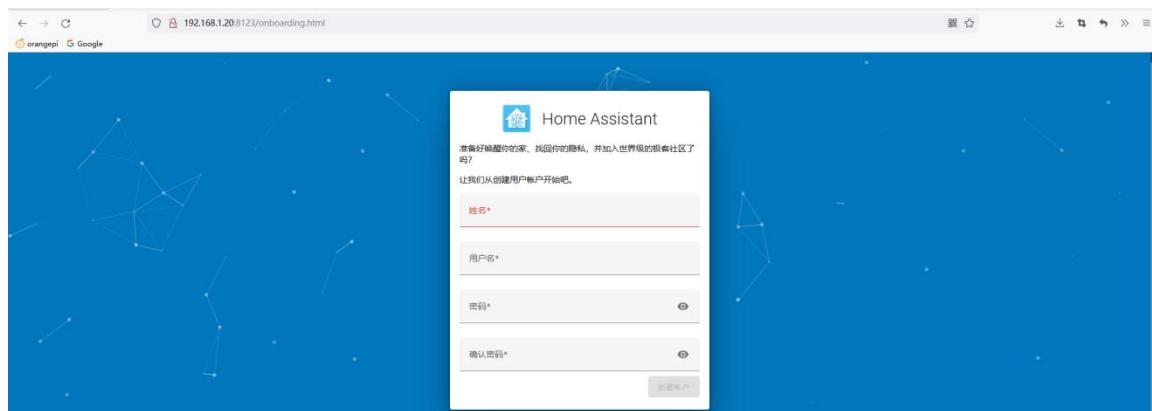
```
(homeassistant) orangepi@orangepi:/srv/homeassistant$ pip3 install homeassistant
```

6) 然后输入下面的命令就可以运行 Home Assistant Core

```
(homeassistant) orangepi@orangepi:/srv/homeassistant$ hass
```

7) 然后在浏览器中输入【开发板的 IP 地址:8123】就能看到 Home Assistant 的界面

第一次运行 **hass** 命令时，会下载安装和缓存一些运行必须的库和依赖包。这个过程可能会花费几分钟的时间。注意，此时在浏览器中是无法看到 **Home Assistant** 的界面的，请等待一段时间后再刷新下。



3.27. OpenCV 的安装方法

3.27.1. 使用 apt 来安装 OpenCV

1) 安装命令如下所示

```
orangepi@orangepi:~$ sudo apt-get update  
orangepi@orangepi:~$ sudo apt-get install -y libopencv-dev python3-opencv
```

2) 然后使用下面的命令打印 OpenCV 的版本号输出正常，说明 OpenCV 安装成功

a. Ubuntu22.04 中 OpenCV 的版本如下所示：

```
orangepi@orangepi:~$ python3 -c "import cv2; print(cv2.__version__)"
```



4.5.4

b. Ubuntu20.04 中 OpenCV 的版本如下所示:

```
orangeipi@orangeipi:~$ python3 -c "import cv2; print(cv2.__version__)"
```

4.2.0

c. Debian11 中 OpenCV 的版本如下所示:

```
orangeipi@orangeipi:~$ python3 -c "import cv2; print(cv2.__version__)"
```

4.5.1

d. Debian12 中 OpenCV 的版本如下所示:

```
orangeipi@orangeipi:~$ python3 -c "import cv2; print(cv2.__version__)"
```

4.6.0

3. 28. 设置中文环境以及安装中文输入法

注意，安装中文输入法前请确保开发板使用的 Linux 系统为桌面版系统。

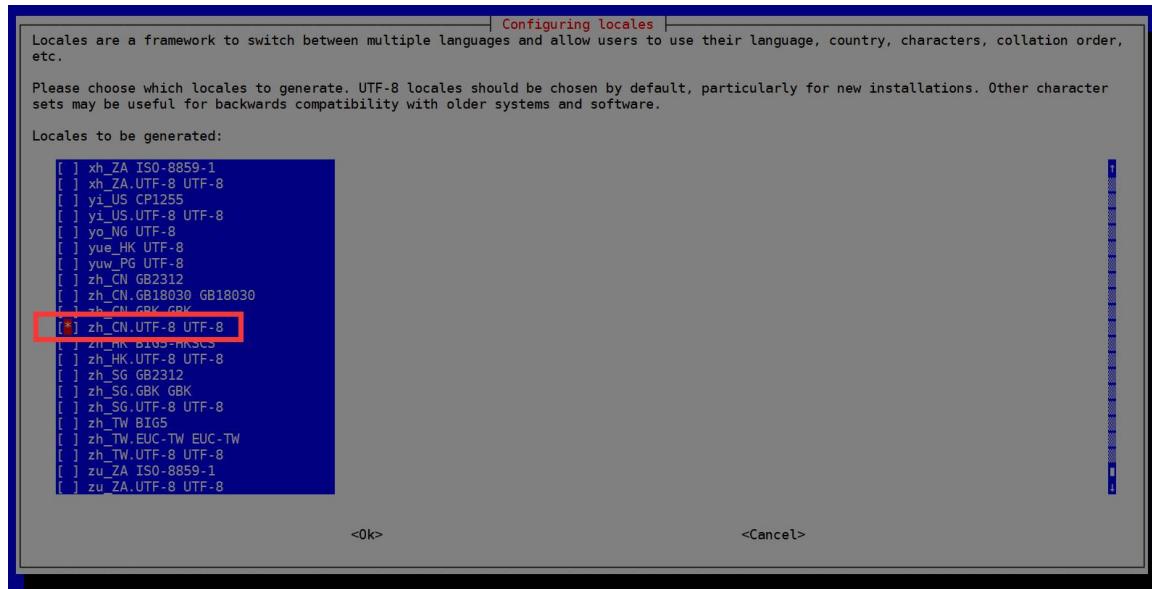
3. 28. 1. Debian 系统的安装方法

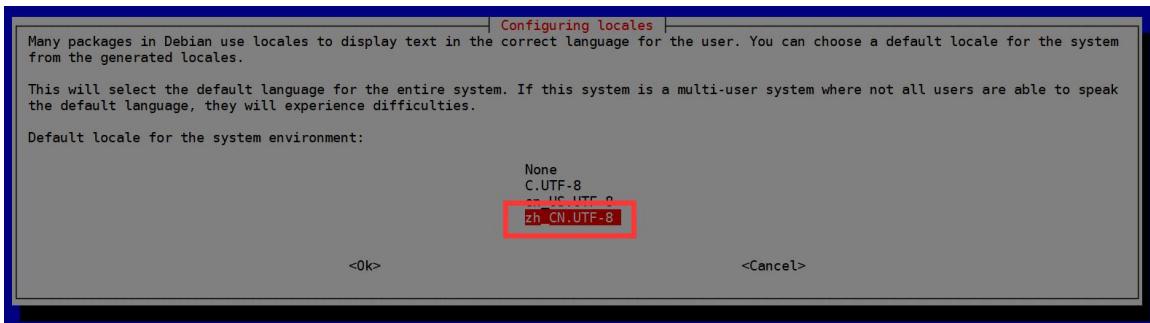
1) 首先设置默认 **locale** 为中文

a. 输入下面的命令可以开始配置 **locale**

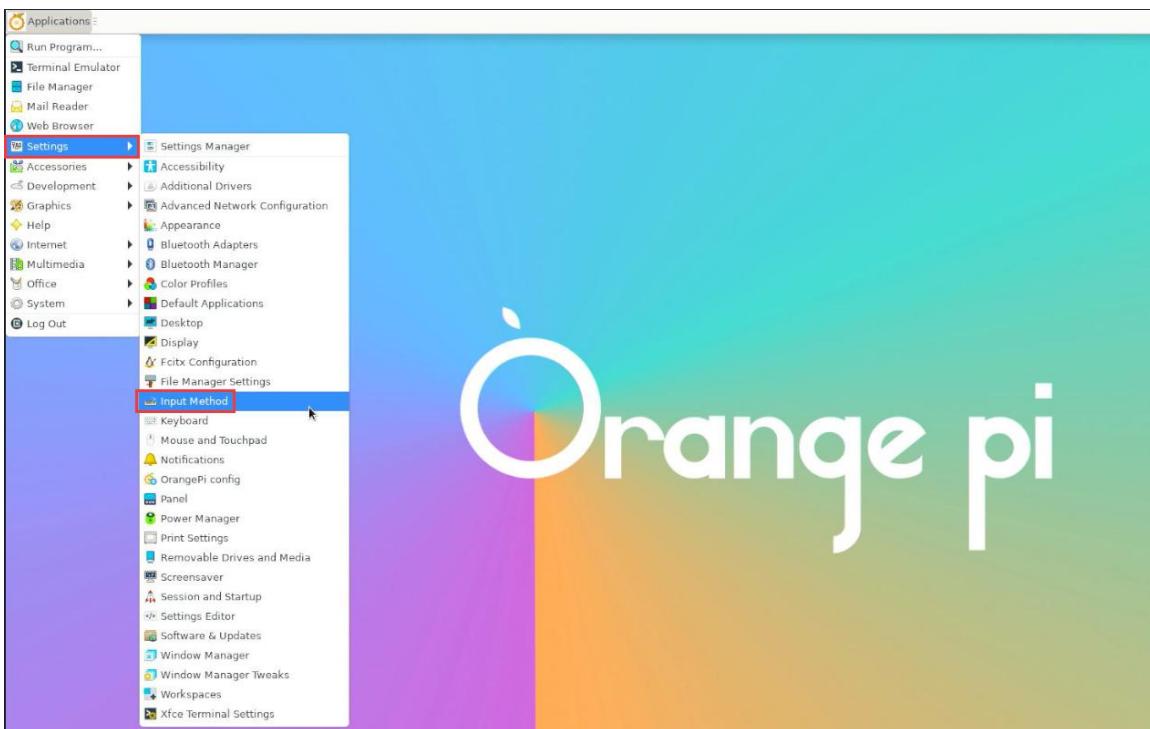
```
orangeipi@orangeipi:~$ sudo dpkg-reconfigure locales
```

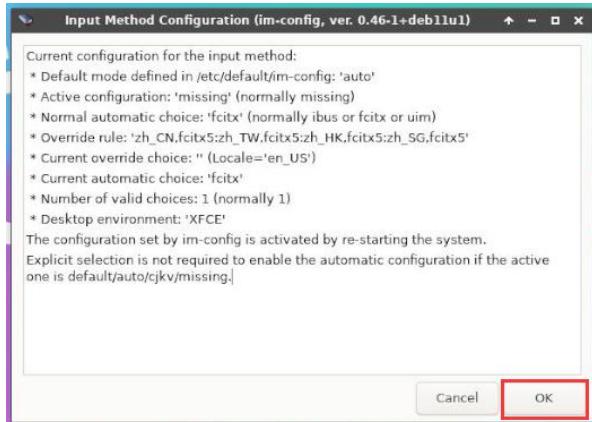
b. 然后在弹出的界面中选择 **zh_CN.UTF-8 UTF-8**（通过键盘上的上下方向按键来上下移动，通过空格键来选择，最后通过 Tab 键可以将光标移动到 **<OK>**，然后回车即可）



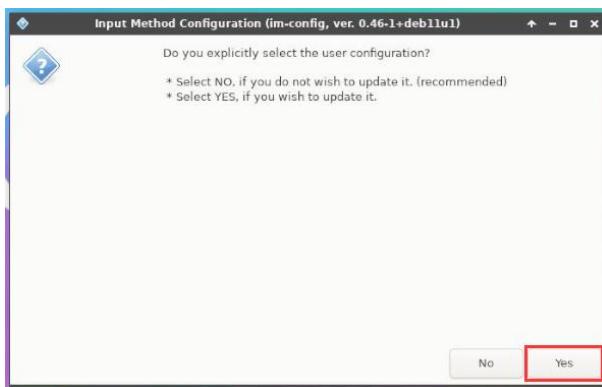
c. 然后设置默认 **locale** 为 **zh_CN.UTF-8**d. 退出界面后就会开始 **locale** 的设置，命令行显示的输出如下所示

```
orangeipi@orangeipi:~$ sudo dpkg-reconfigure locales
Generating locales (this might take a while)...
en_US.UTF-8... done
zh_CN.UTF-8... done
Generation complete.
```

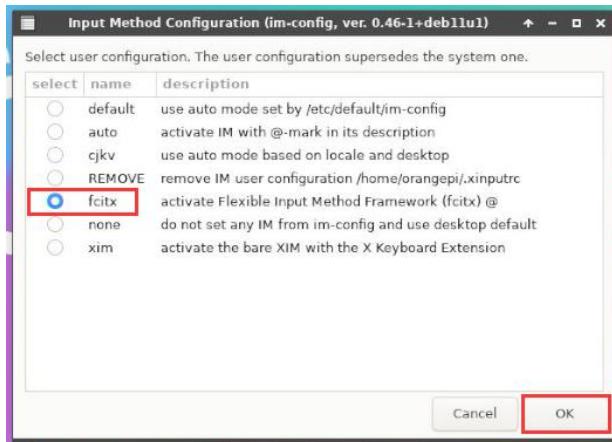
2) 然后打开 **Input Method**3) 然后选择 **OK**



4) 然后选择 Yes



5) 然后选择 fcitx

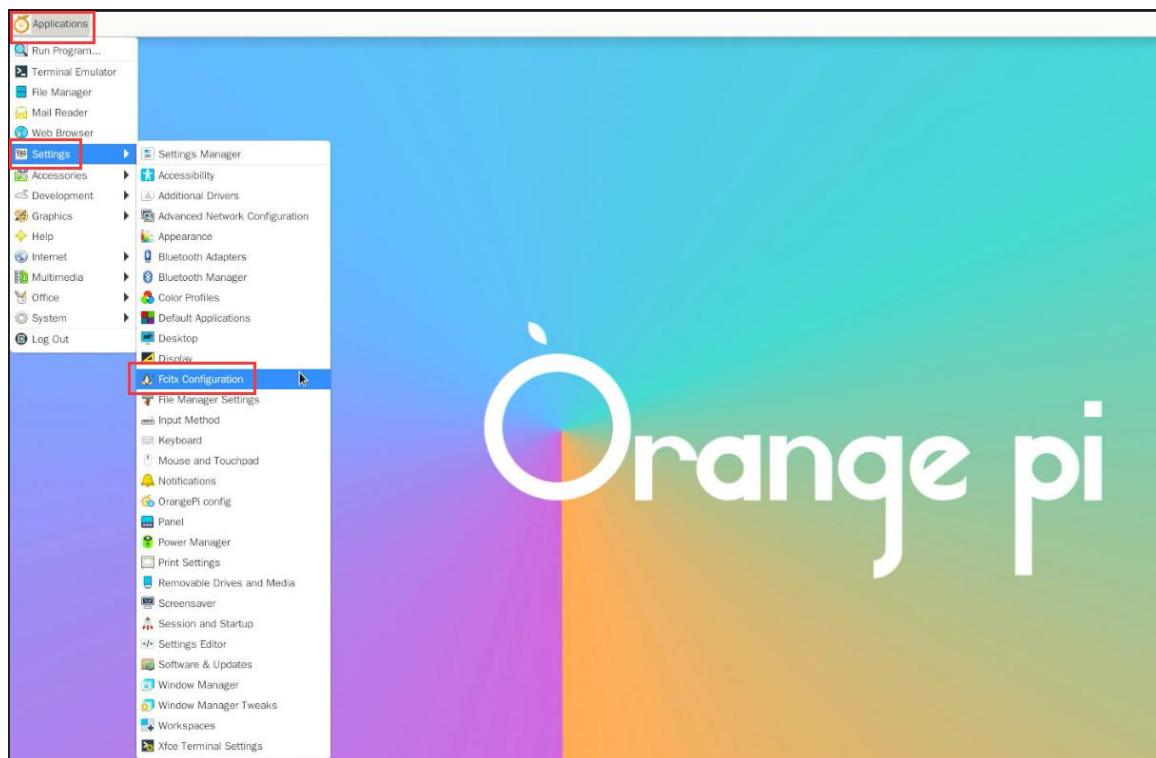


6) 然后选择 OK

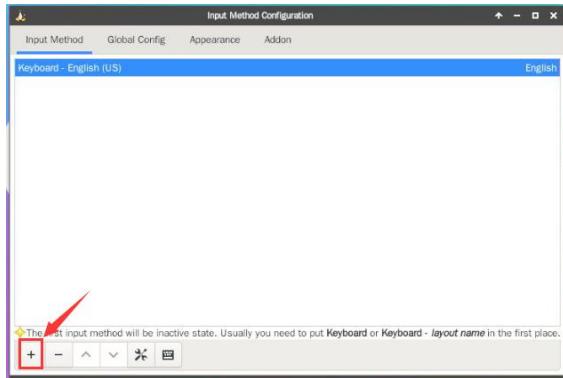


7) 然后重启 Linux 系统才能使配置生效

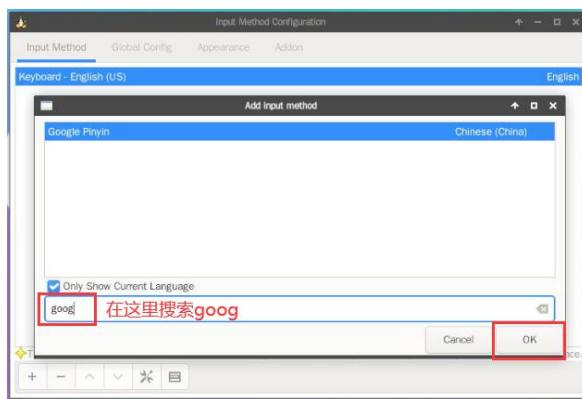
8) 然后打开 **Fcitx configuration**



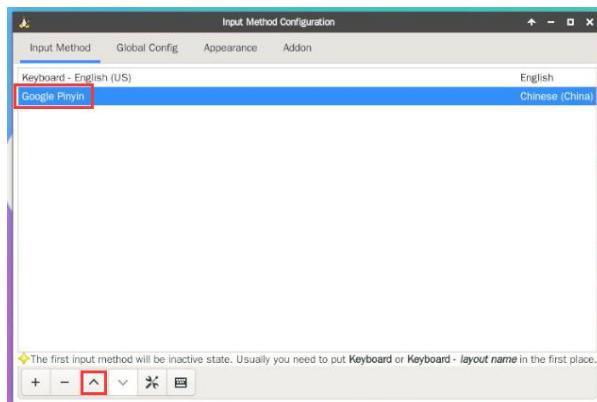
9) 然后点击下图所示位置的+号

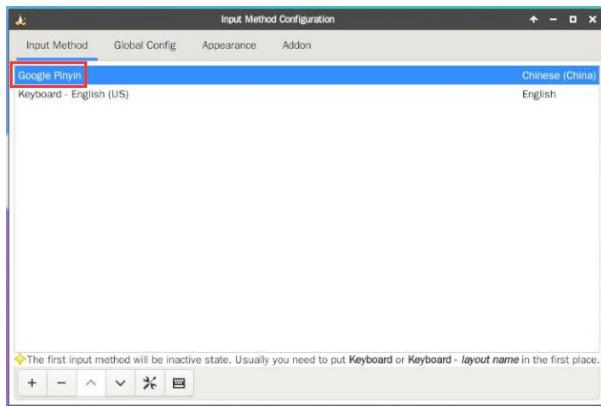


10) 然后搜索 **Google Pinyin** 再点击 **OK**



11) 然后将 **Google Pinyin** 放到最前面

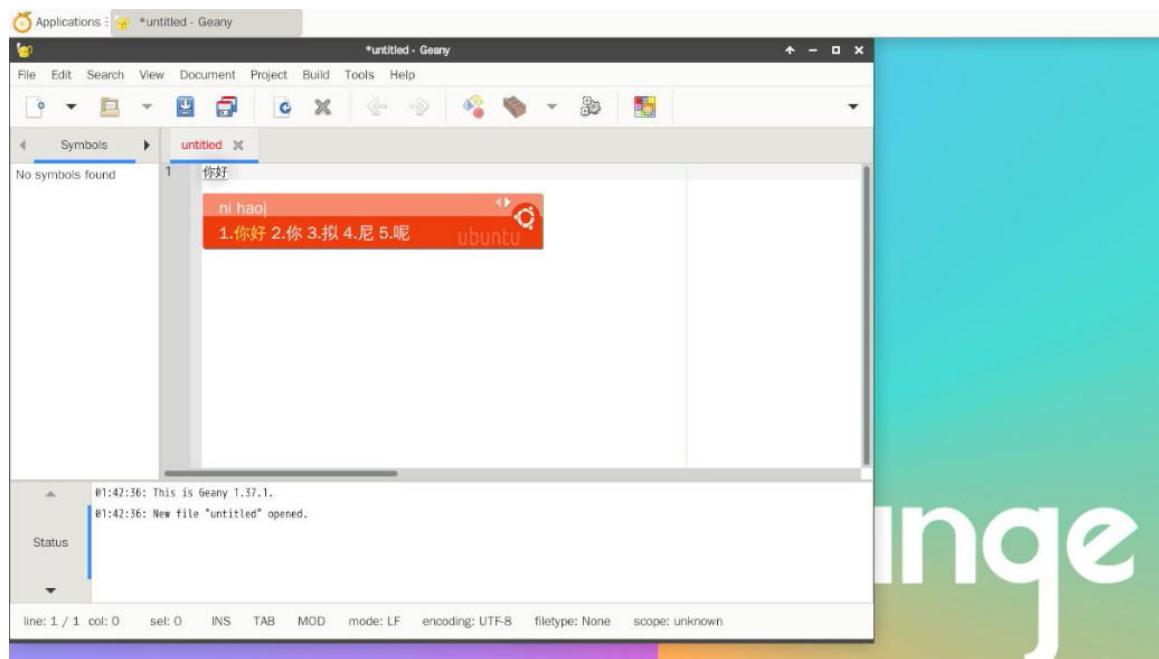




12) 然后打开 Geany 这个编辑器测试下中文输入法



13) 中文输入法测试如下所示



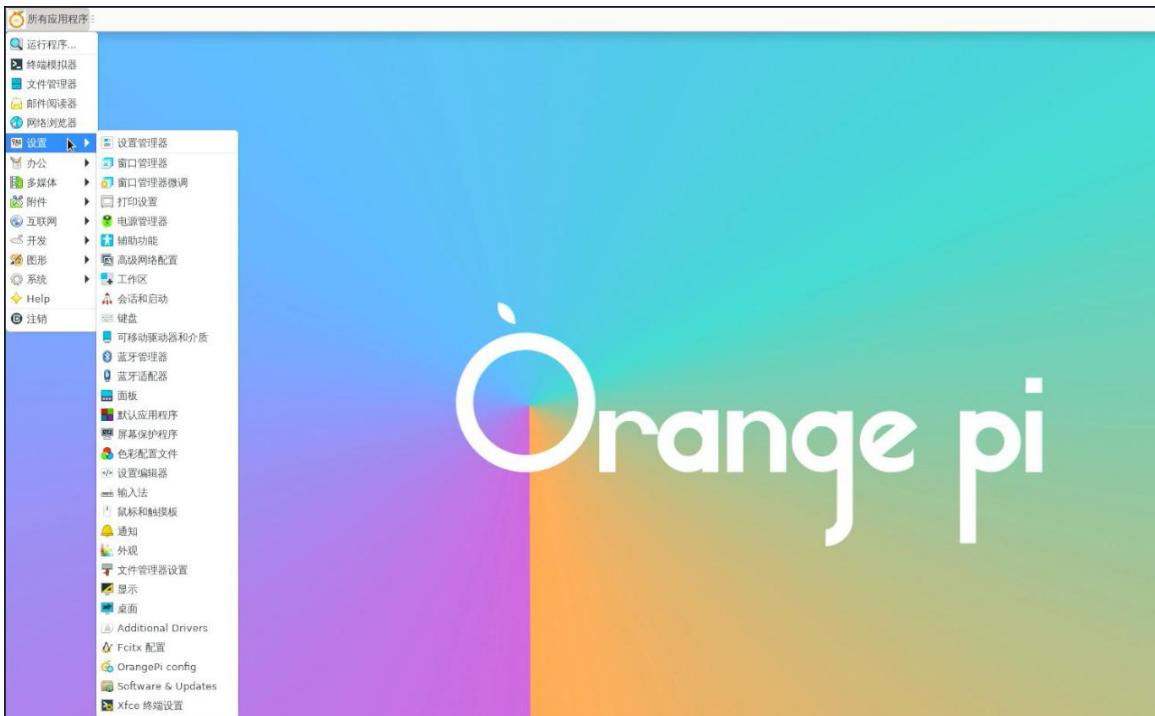


14) 通过 **Ctrl+Space** 快捷键可以切换中英文输入法

15) 如果需要整个系统都显示为中文，可以将 **/etc/default/locale** 中的变量都设置为 **zh_CN.UTF-8**

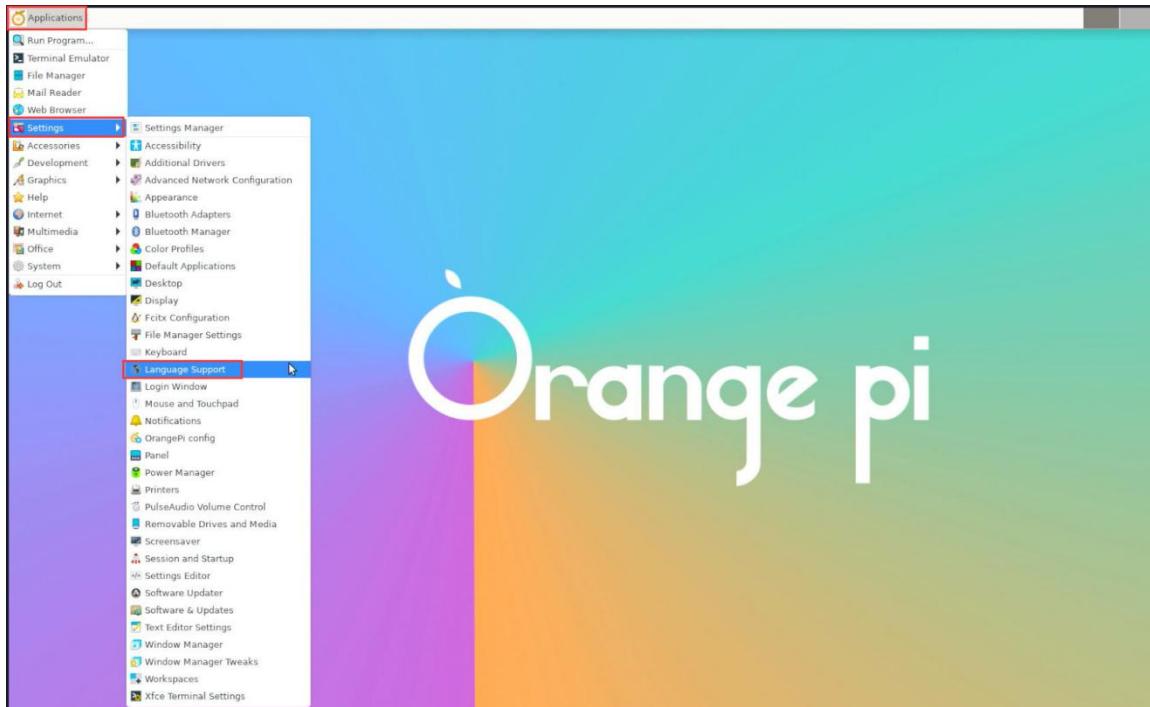
```
orangepi@orangepi:~$ sudo vim /etc/default/locale
# File generated by update-locale
LC_MESSAGES=zh_CN.UTF-8
LANG=zh_CN.UTF-8
LANGUAGE=zh_CN.UTF-8
```

16) 然后**重启系统**就能看到系统显示为中文了

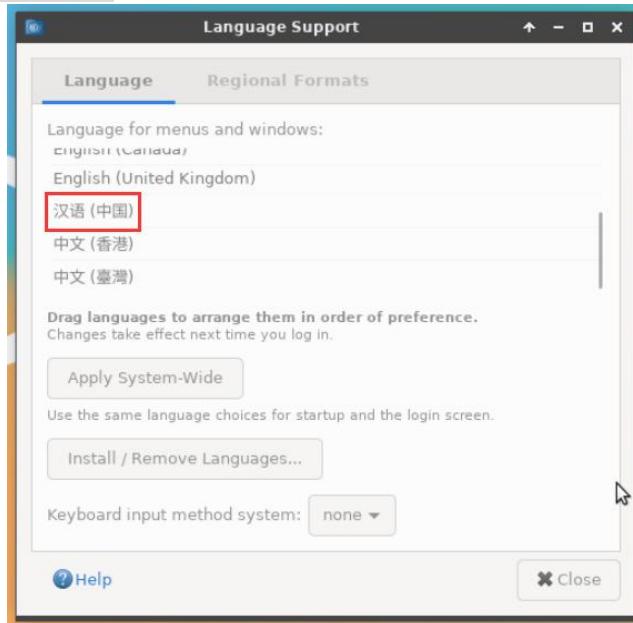


3.28.2. Ubuntu 20.04 系统的安装方法

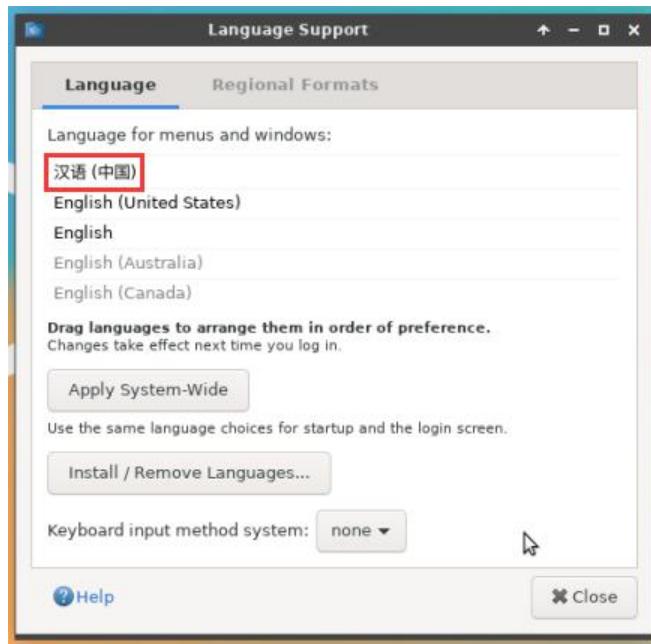
1) 首先打开 **Language Support**



2) 然后找到汉语（中国）选项

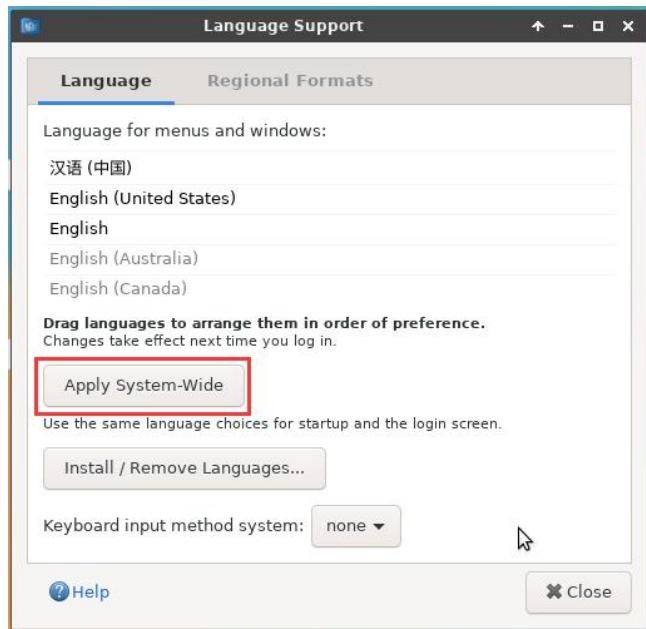


3) 然后请使用鼠标左键选中汉语（中国）并按住不动，然后往上将其拖到最开始的位置，拖完后的显示如下图所示：

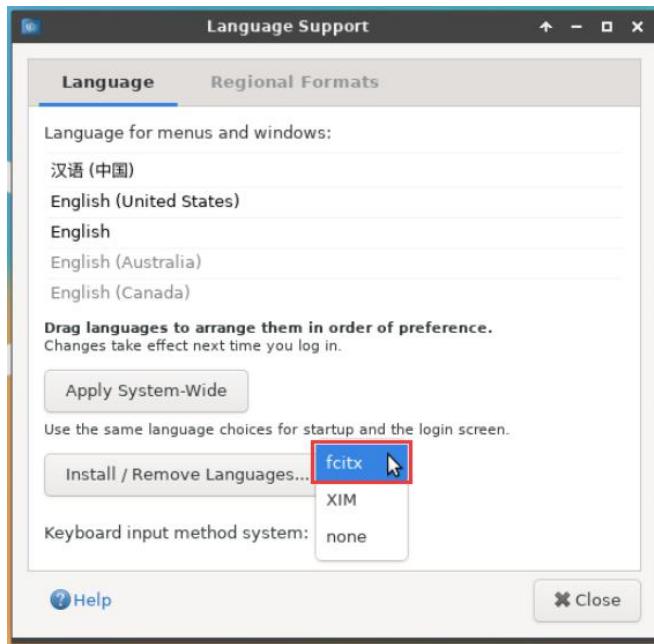


注意，这一步不是很好拖动的，请耐心多试几次。

4) 然后选择 **Apply System-Wide** 将中文设置应用到整个系统



5) 然后设置 **Keyboard input method system** 为 **fcitx**

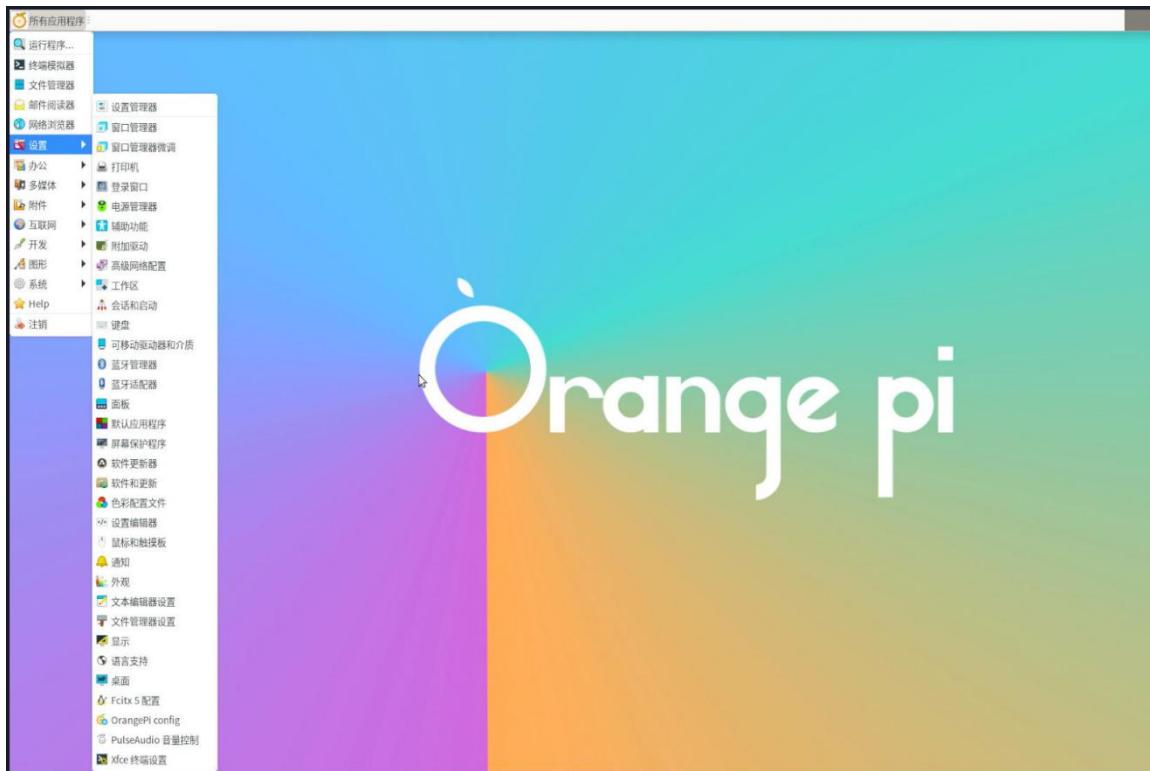


6) 然后重启 Linux 系统使配置生效

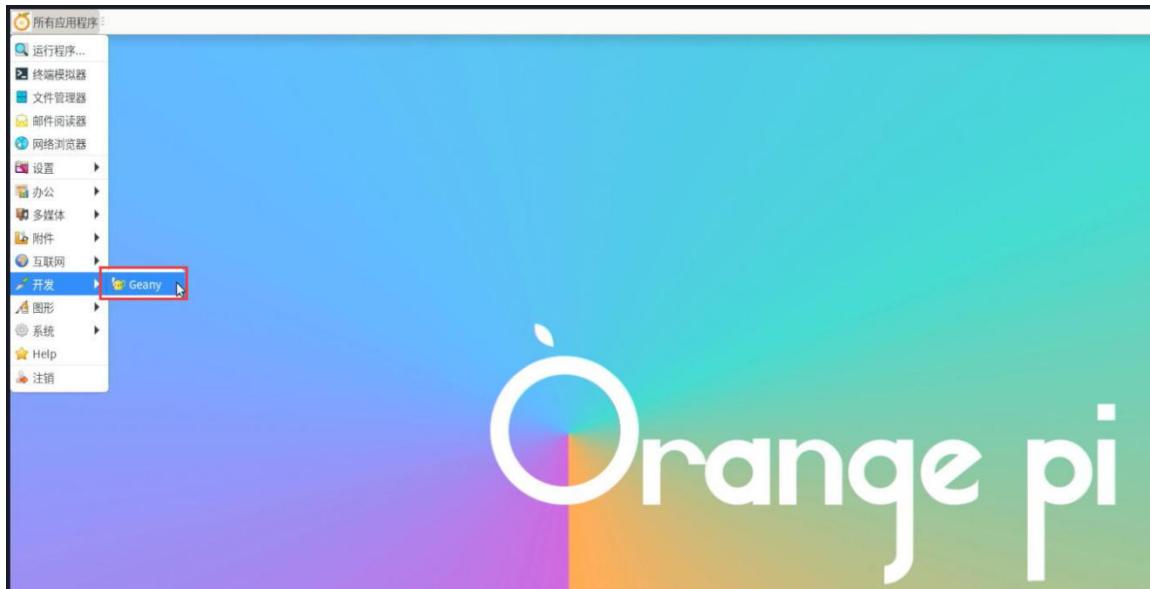
7) 重新进入系统后，在下面的界面请选择**不要再次询问我**，然后请根据自己的喜好决定标准文件夹是否也要更新为中文



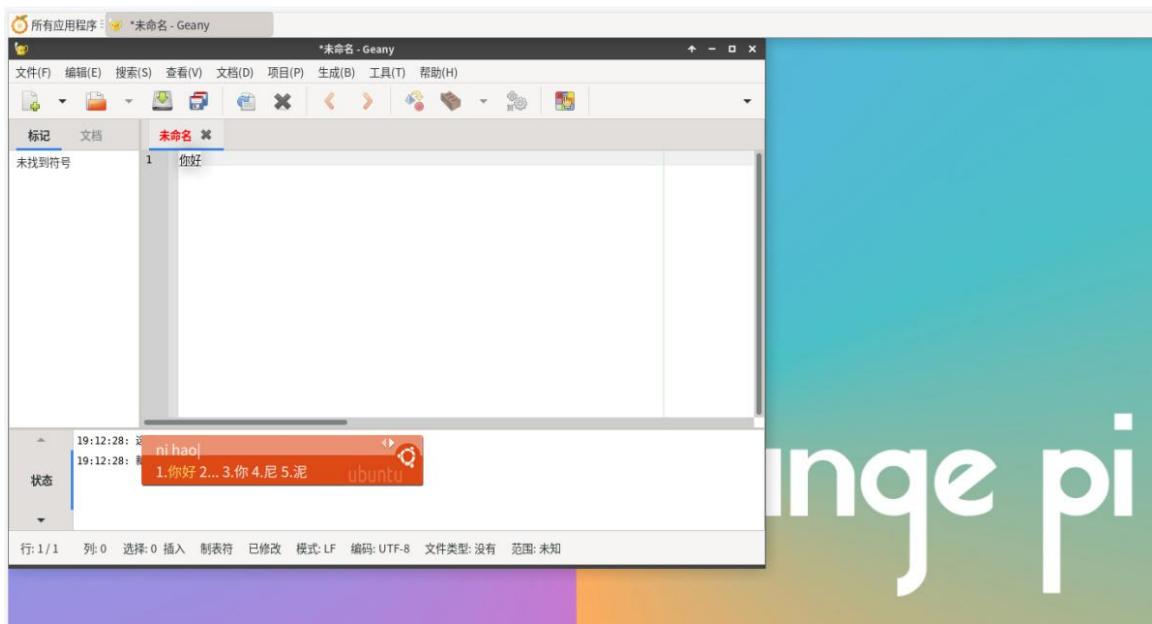
8) 然后可以看到桌面都显示为中文了



9) 然后我们可以打开 **Geany** 测试下中文输入法，打开方式如下图所示

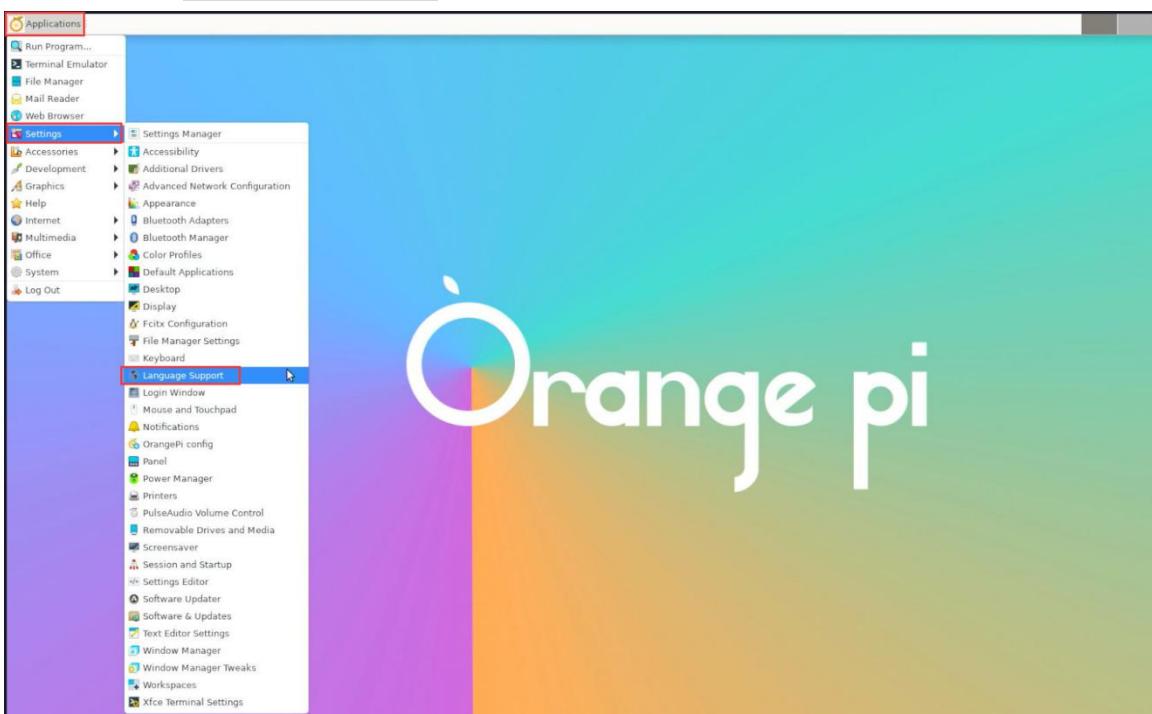


10) 打开 **Geany** 后，默认还是英文输入法，我们可以通过 **Ctrl+Space** 快捷键来切换成中文输入法，然后就能输入中文了

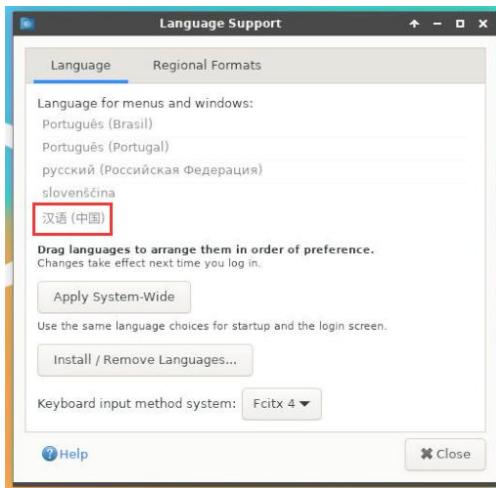


3.28.3. Ubuntu 22.04 系统的安装方法

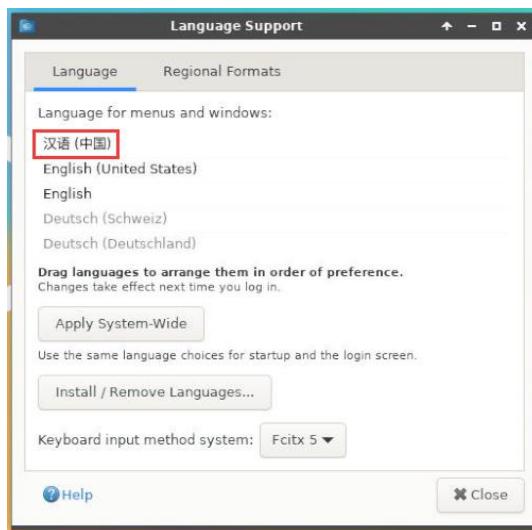
1) 首先打开 Language Support



2) 然后找到汉语（中国）选项

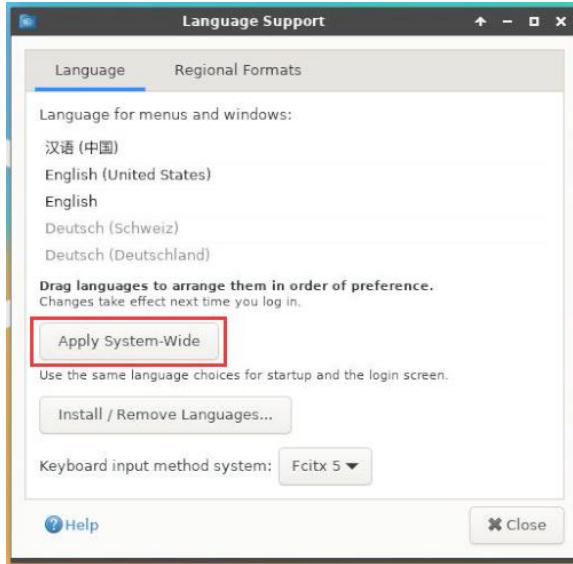


3) 然后请使用鼠标左键选中**汉语 (中国)**并按住不动，然后往上将其拖到最开始的位置，拖完后的显示如下图所示：



注意，这一步不是很好拖动的，请耐心多试几次。

4) 然后选择 **Apply System-Wide** 将中文设置应用到整个系统

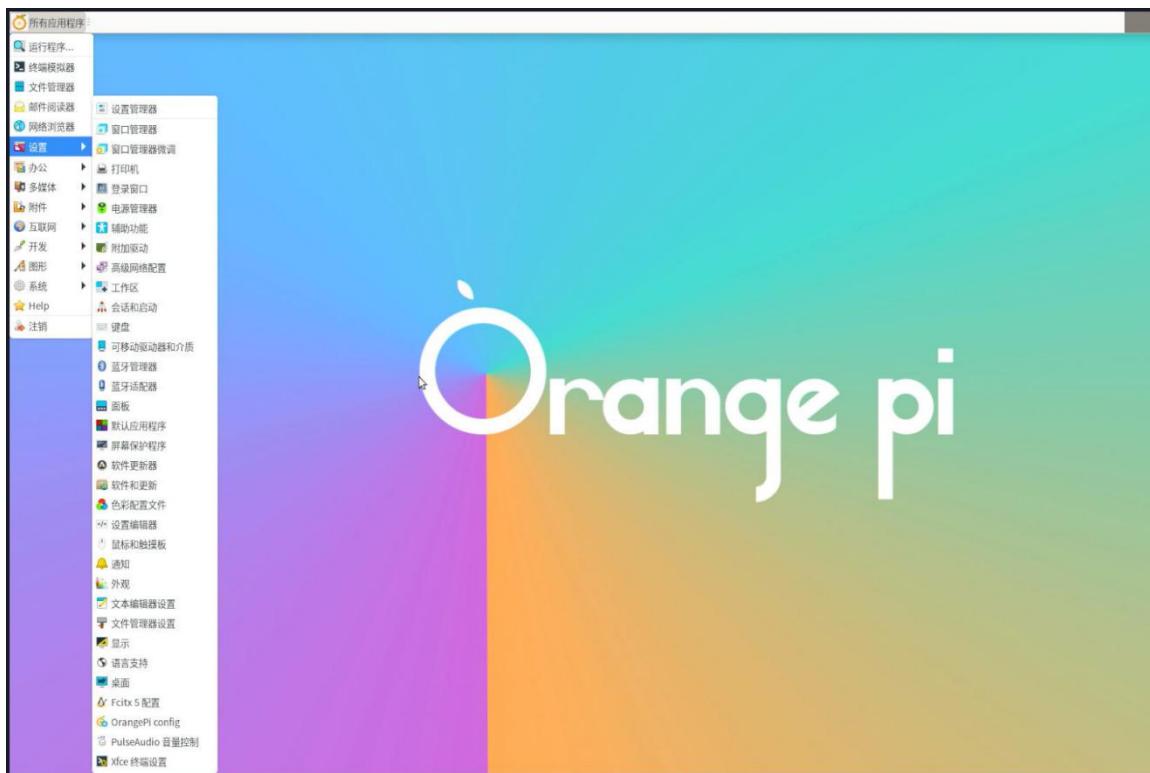


5) 然后重启 Linux 系统使配置生效

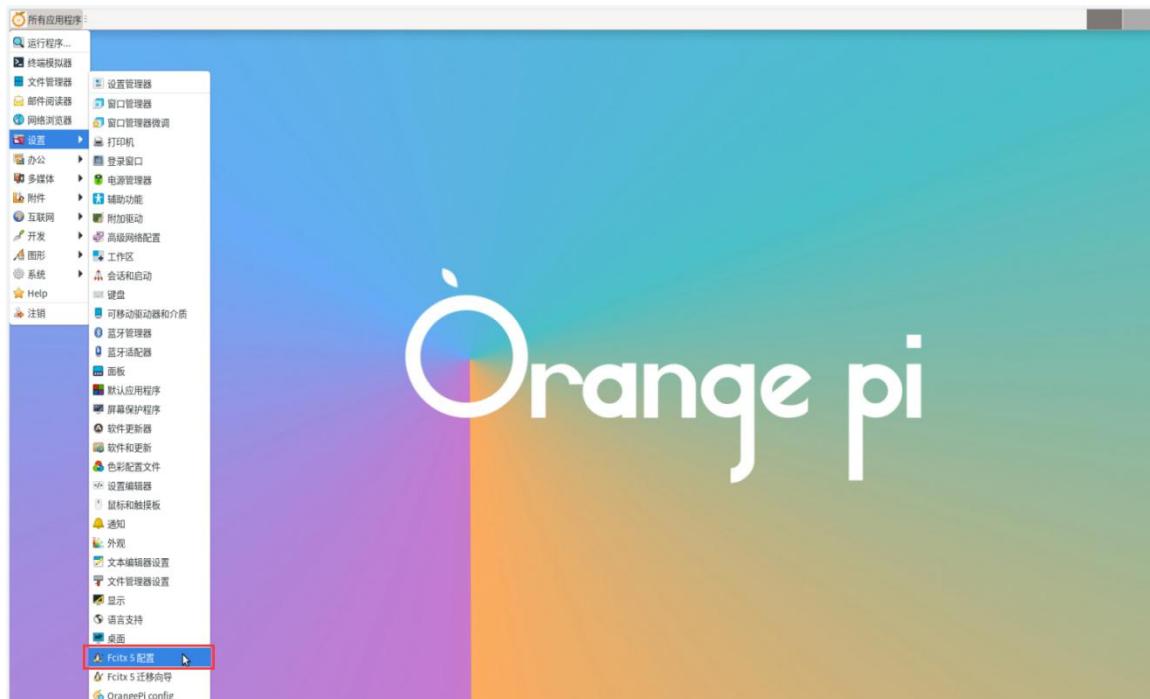
6) 重新进入系统后，在下面的界面请选择**不要再次询问我**，然后请根据自己的喜好决定标准文件夹是否也要更新为中文



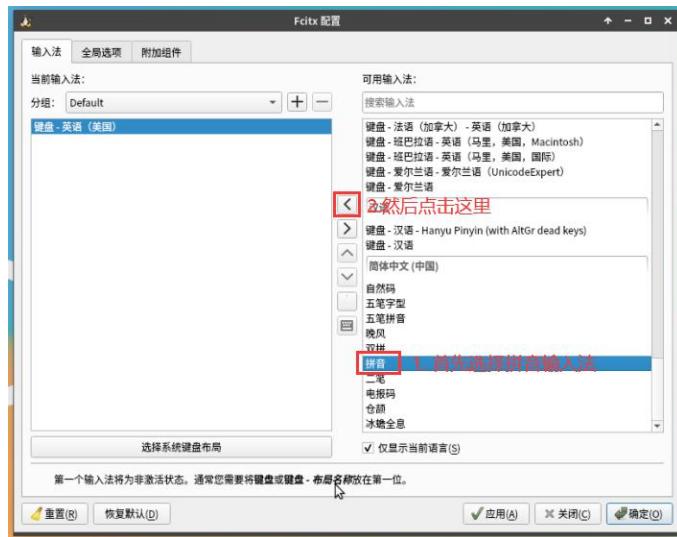
7) 然后可以看到桌面都显示为中文了



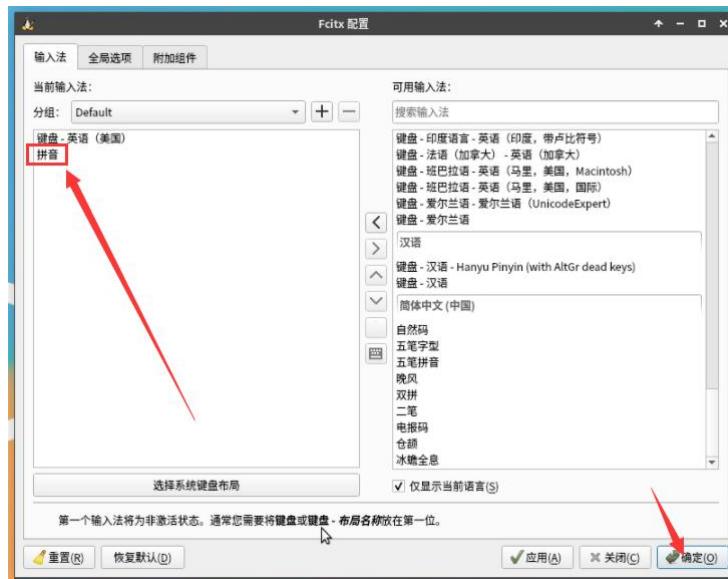
8) 然后打开 Fcitx5 配置程序



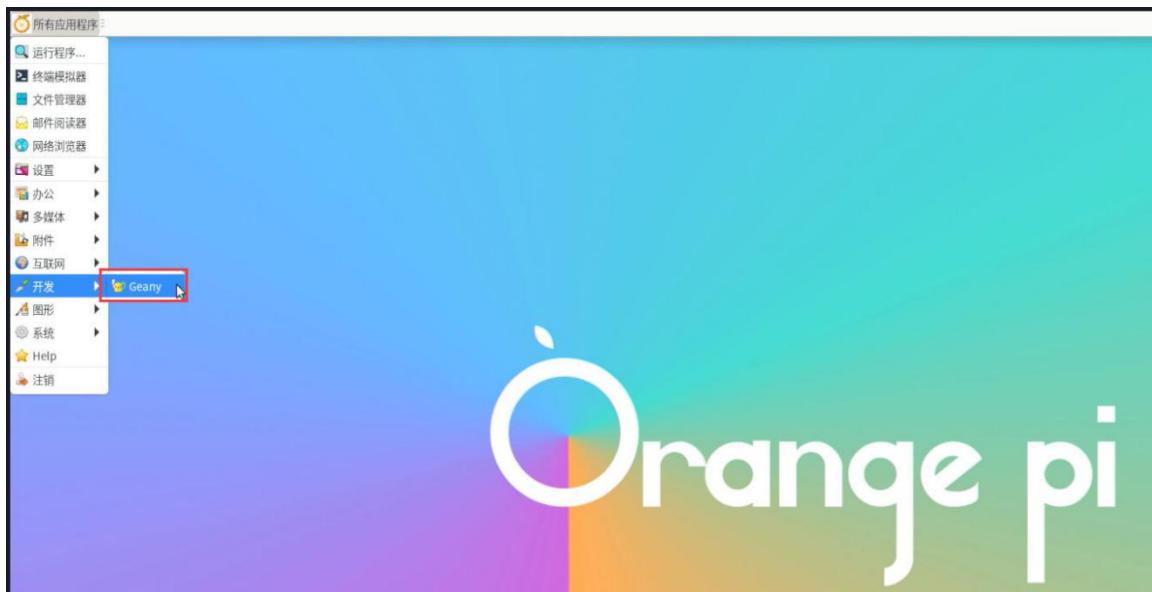
9) 然后选择使用拼音输入法



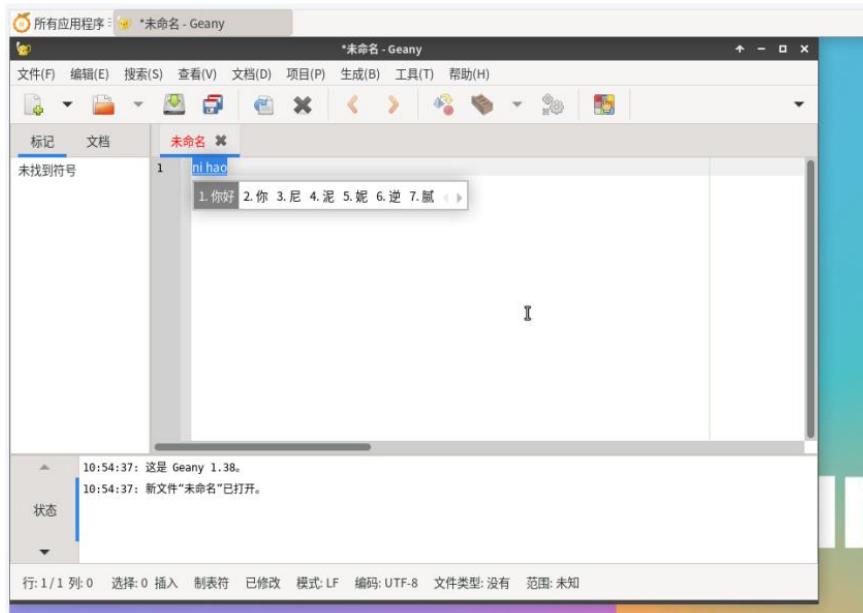
10) 选择后的界面如下所示，再点击确定即可



11) 然后我们可以打开 **Geany** 测试下中文输入法，打开方式如下图所示



12) 打开 **Geany** 后，默认还是英文输入法，我们可以通过 **Ctrl+Space** 快捷键来切换成中文输入法，然后就能输入中文了



3.29. 远程登录 Linux 系统桌面的方法

3.29.1. 使用 NoMachine 远程登录

请确保开发板安装的 Ubuntu 或者 Debian 系统为桌面版本的系统。另外



NoMachine 也提供了详细的使用文档，强烈建议通读此文档来熟悉 NoMachine 的使用，文档链接如下所示：

<https://knowledgebase.nomachine.com/DT10R00166>

NoMachine 支持 Windows、Mac、Linux、iOS 和安卓平台，所以我们可以 在多种设备上通过 NoMachine 来远程登录控制 Orange Pi 开发板。下面演示下在 Windows 中通过 NoMachine 来远程登录 Orange Pi 开发板的 Linux 系统桌面。其他平台的安装方法请参考下 NoMachine 的官方文档。

操作前请先确保 Windwos 电脑和开发板在同一局域网内，并且能正常 ssh 登录开发板的 Ubuntu 或者 Debian 系统。

1) 首先下载 NoMachine 软件 Linux **arm64** deb 版本的安装包，然后安装到开发板的 Linux 系统中

- a. 由于 H618 是 ARMv8 架构的 SOC，我们使用的系统为 Ubuntu 或者 Debian，所以这里需要下载 **NoMachine for ARM ARMv8 DEB** 安装包，下载链接如下所示：

注意，这个下载链接可能会变，请认准 Armv8/Arm64 版本的 deb 包。

<https://downloads.nomachine.com/download/?id=118&distro=ARM>

Home / Download / NoMachine for ARM - arm64

NoMachine for ARM - arm64



Version:	8.5.3_1
Package size:	48.34 MB
Package type:	DEB
MD5 signature:	2291f8d8ec76f0a914285acaaa93e34d
For:	Ubuntu 14.04/16.04/18.04/20.04, Debian 8/9/10

Although your ARMv8 device may not be listed here, we encourage you to try the packages. Please consult the installation and configuration notes about Linux for ARM packages for more details about devices and specific distributions we have tested.

[Download](#)

- b. 另外在官方工具中也可以下载到 **NoMachine** 的安装包



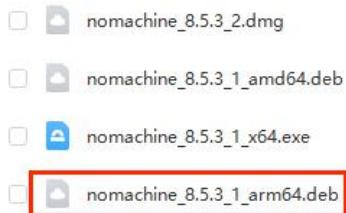
官方工具

[下载](#)

先进入远程登录软件-NoMachine 文件夹



然后下载 arm64 版本的 deb 安装包



- c. 然后将下载的 **nomachine_x.x.x_x_arm64.deb** 上传到开发板的 Linux 系统中
- d. 然后使用下面的命令在开发板的 Linux 系统中安装 **NoMachine**

```
orangeipi@orangeipi:~$ sudo dpkg -i nomachine_x.x.x_x_arm64_arm64.deb
```

2) 然后下载 NoMachine 软件 Windows 版本的安装包，下载地址如下所示

注意，这个下载链接可能会变。

<https://downloads.nomachine.com/download/?id=9>

NoMachine for Windows - 64bit



Version:	8.5.3_1
Package size:	57.4 MB
Package type:	EXE
MDS signature:	d585ad1e4f341444cacd3ae8add3b6ee
For:	Windows 7/8/8.1/10/11/Windows Server 2008/2012/2016/2019

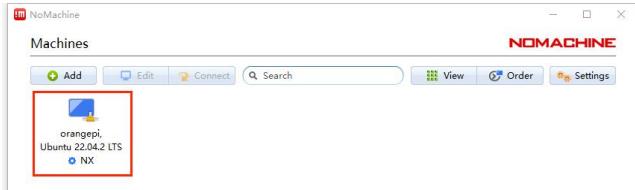
Download

3) 然后在 Windows 中安装 NoMachine，安装完后请重启下电脑

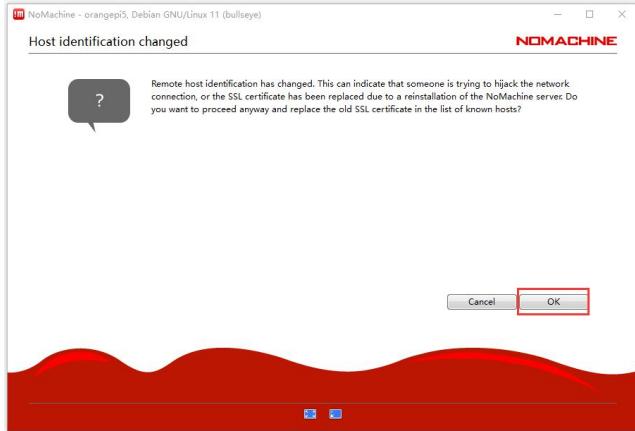
4) 然后在 Window 中打开 **NoMachine**



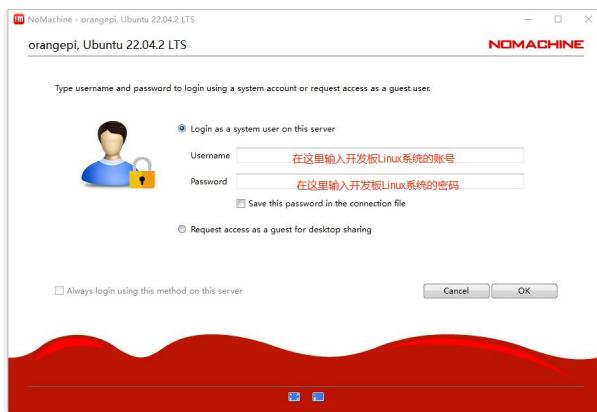
5) NoMachine 启动后会自动扫描局域网内其他安装有 NoMachine 的设备，进入 NoMachine 的主界面后就可以看到开发板已经在可连接的设备列表里了，然后点击下图红色方框所示的位置即可开始登录开发板的 Linux 系统桌面



6) 然后点击 **OK**

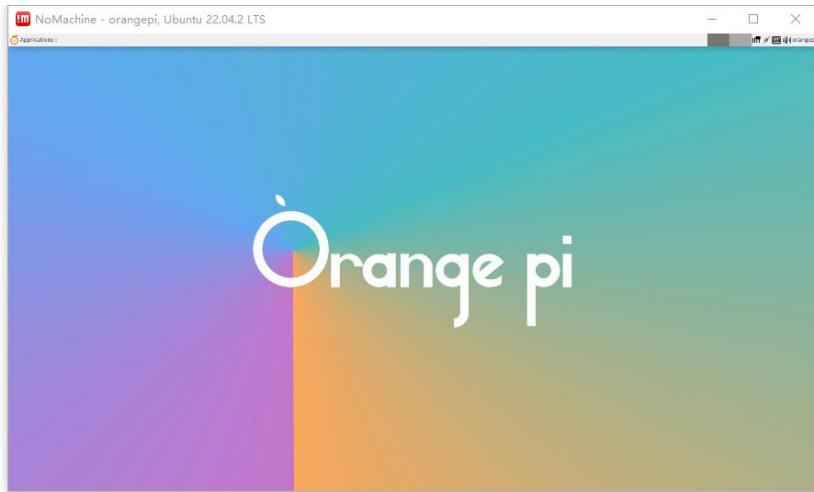


7) 然后在下图对应的位置输入开发板 Linux 系统的用户名和密码，再点击 **OK** 开始登陆



8) 然后在接下来的界面中都点击 **OK**

9) 最后就能看到开发板 Linux 系统的桌面了



3.29.2. 使用 VNC 远程登录

操作前请先确保 Windows 电脑和开发板在同一局域网内，并且能正常 ssh 登录开发板的 Ubuntu 或者 Debian 系统。

Ubuntu20.04 测试 VNC 很多问题，请不要使用这种方法。

1) 首先运行 **set_vnc.sh** 脚本设置下 vnc，**记得加 sudo 权限**

```
orangepi@orangepi:~$ sudo set_vnc.sh
```

```
You will require a password to access your desktops.
```

```
Password:      #在这里设置 vnc 的密码，8 位字符
```

```
Verify:       #在这里设置 vnc 的密码，8 位字符
```

```
Would you like to enter a view-only password (y/n)? n
```

```
xauth: file /root/.Xauthority does not exist
```

```
New 'X' desktop is orangepi:1
```

```
Creating default startup script /root/.vnc/xstartup
```

```
Starting applications specified in /root/.vnc/xstartup
```

```
Log file is /root/.vnc/orangepi:1.log
```

```
Killing Xtightvnc process ID 3047
```

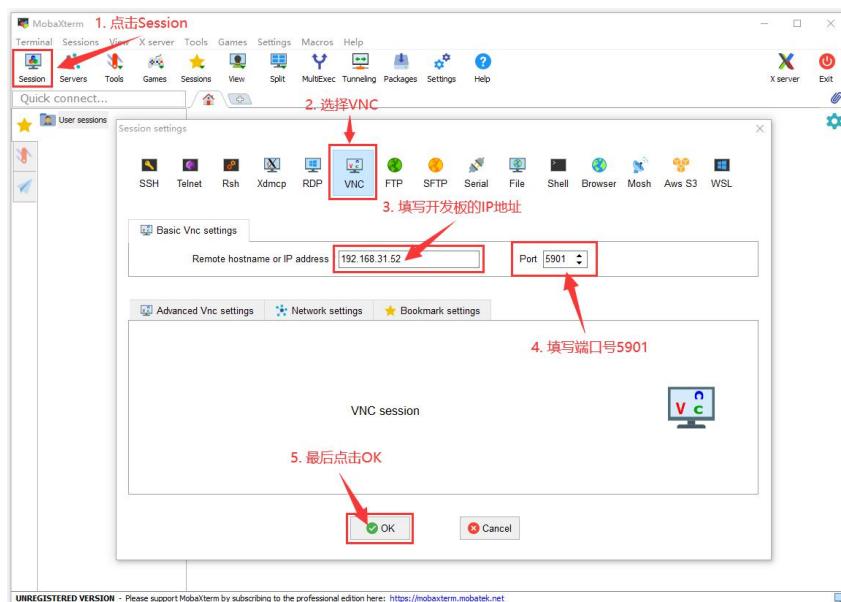


New 'X' desktop is orangepi:1

Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/orangepi:1.log

2) 使用 MobaXterm 软件连接开发板 linux 系统桌面的步骤如下所示：

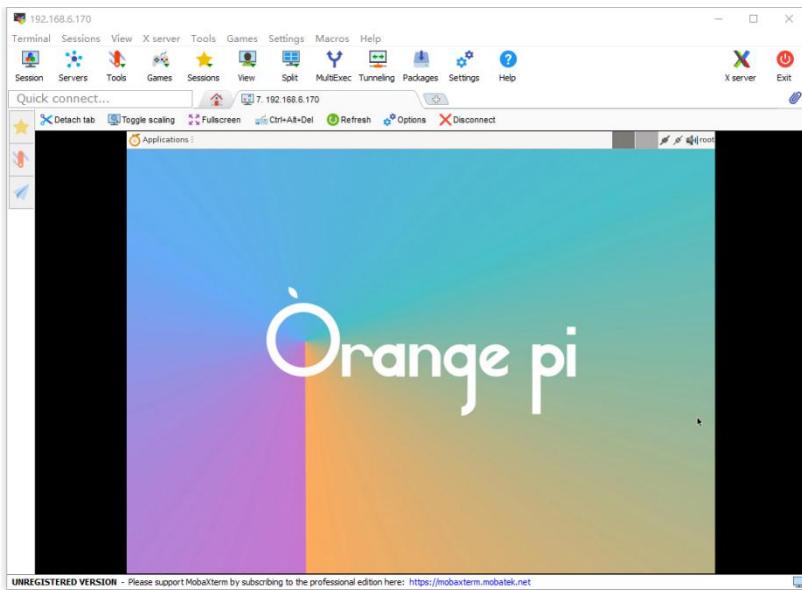
- 首先点击 Session，然后选择 VNC，再填写开发板的 IP 地址和端口，最后点击 OK 确认



- 然后输入前面设置的 VNC 的密码



- 登录成功后的界面显示如下图所示，然后就可以远程操作开发板 linux 系统的桌面了



3.30. QT 的安装方法

1) 使用下面的脚本可以安装 QT5 和 QT Creator

```
orangeipi@orangeipi:~$ install_bt.sh
```

2) 安装完后会自动打印 QT 的版本号

a. Ubuntu20.04 自带的 qt 版本为 **5.12.8**

```
orangeipi@orangeipi:~$ install_bt.sh
```

.....

QMake version 3.1

Using Qt version **5.12.8** in /usr/lib/aarch64-linux-gnu

b. Ubuntu22.04 自带的 QT 版本为 **5.15.3**

```
orangeipi@orangeipi:~$ install_bt.sh
```

.....

QMake version 3.1

Using Qt version **5.15.3** in /usr/lib/aarch64-linux-gnu

c. Debian11 自带的 QT 版本为 **5.15.2**

```
orangeipi@orangeipi:~$ install_bt.sh
```

.....

QMake version 3.1

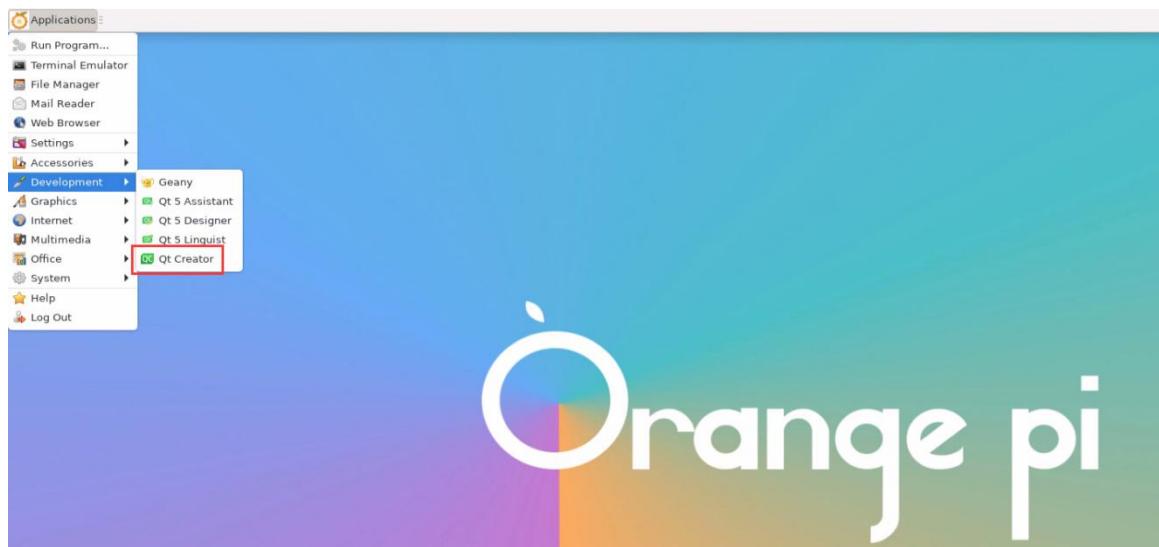
Using Qt version **5.15.2** in /usr/lib/aarch64-linux-gnu



d. Debian12 自带的 QT 版本为 **5.15.8**

```
orangepi@orangepi:~$ install_qt.sh  
.....  
QMake version 3.1  
Using Qt version 5.15.8 in /usr/lib/aarch64-linux-gnu
```

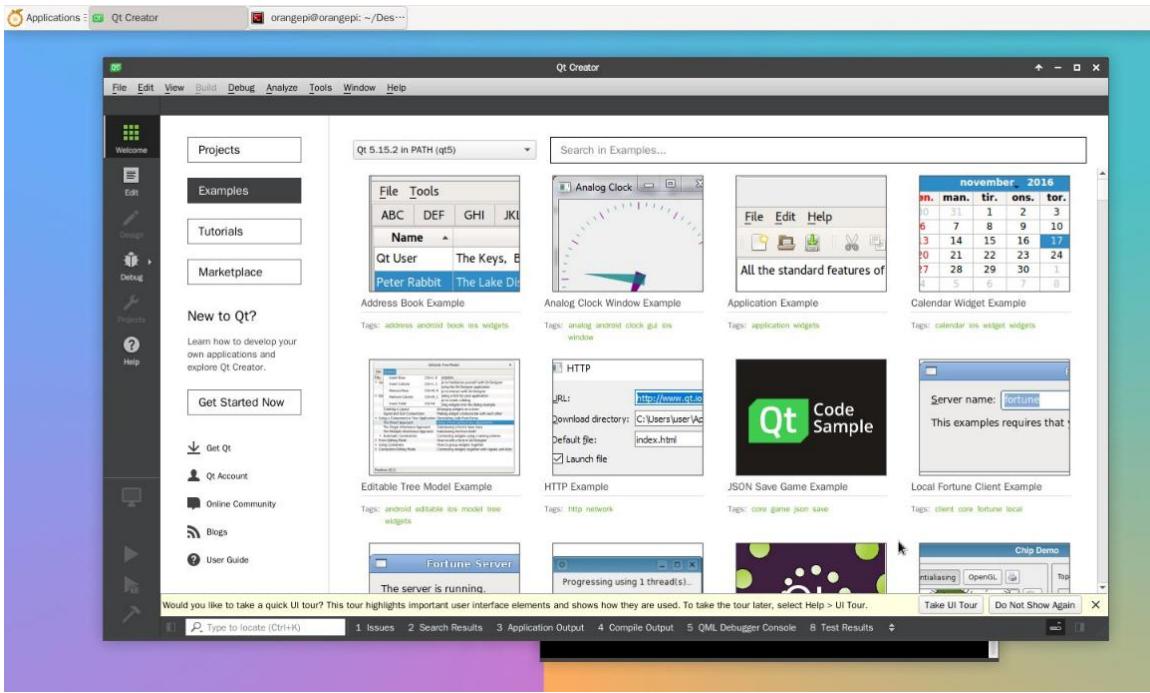
3) 然后在 **Applications** 中就可以看到 QT Creator 的启动图标



也可以使用下面的命令打开 QT Creator

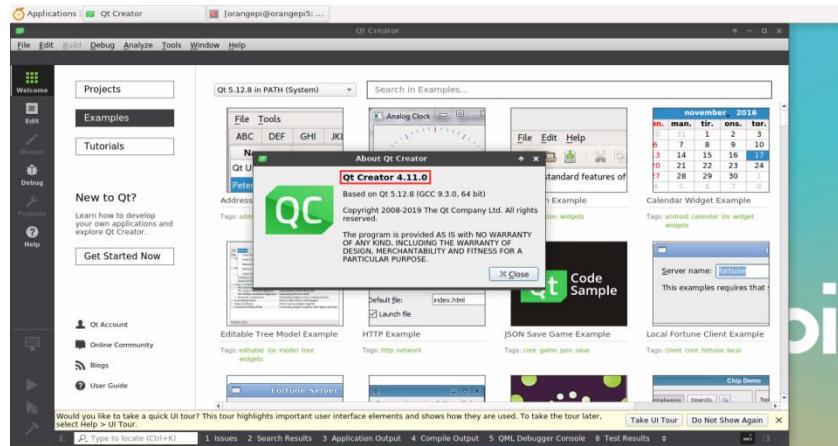
```
orangepi@orangepi:~$ qtcreator
```

4) QT Creator 打开后的界面如下所示

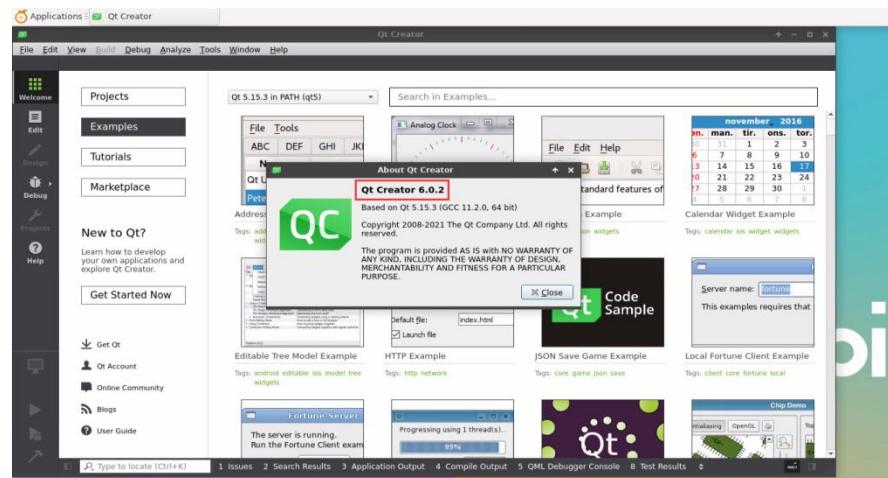


5) QT Creator 的版本如下所示

a. QT Creator 在 Ubuntu20.04 中的默认版本如下所示



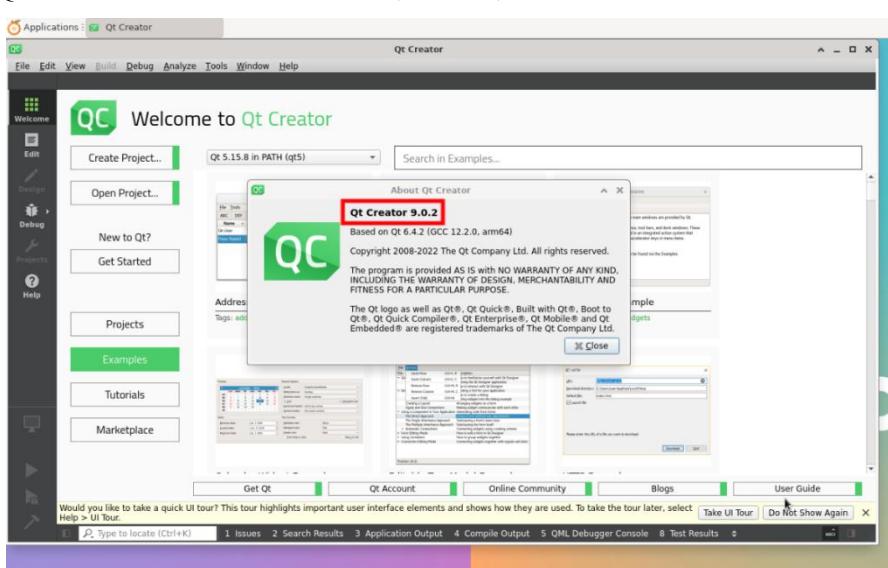
b. QT Creator 在 Ubuntu22.04 中的默认版本如下所示



c. QT Creator 在 Debian11 中的默认版本如下所示



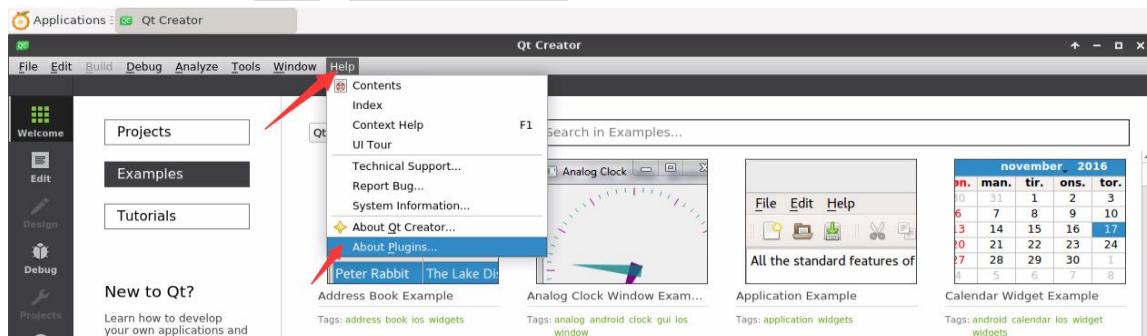
d. QT Creator 在 Debian12 中的默认版本如下所示



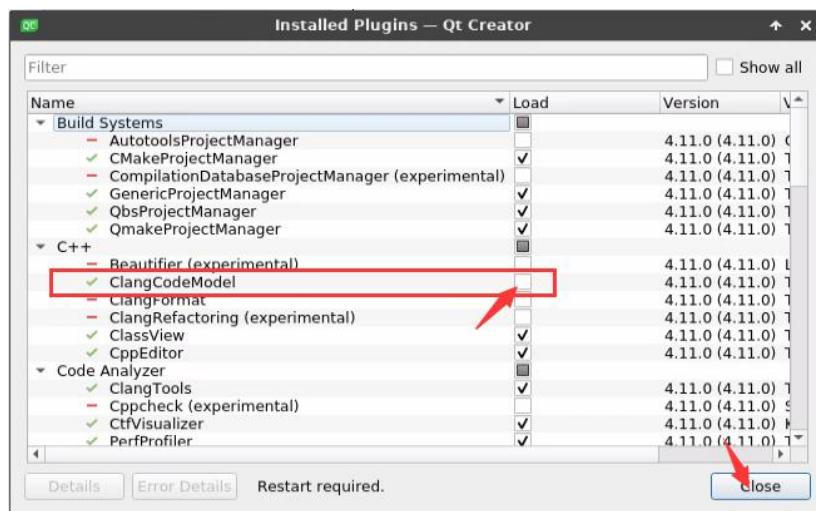
6) 然后设置下 QT



a. 首先打开 **Help->About Plugins...**



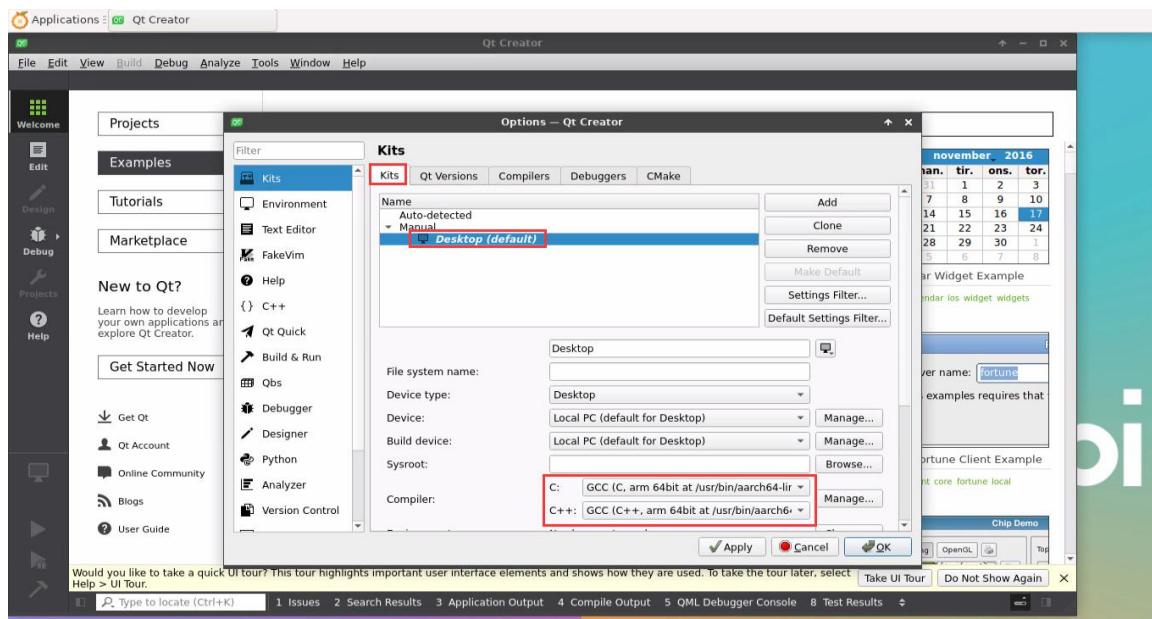
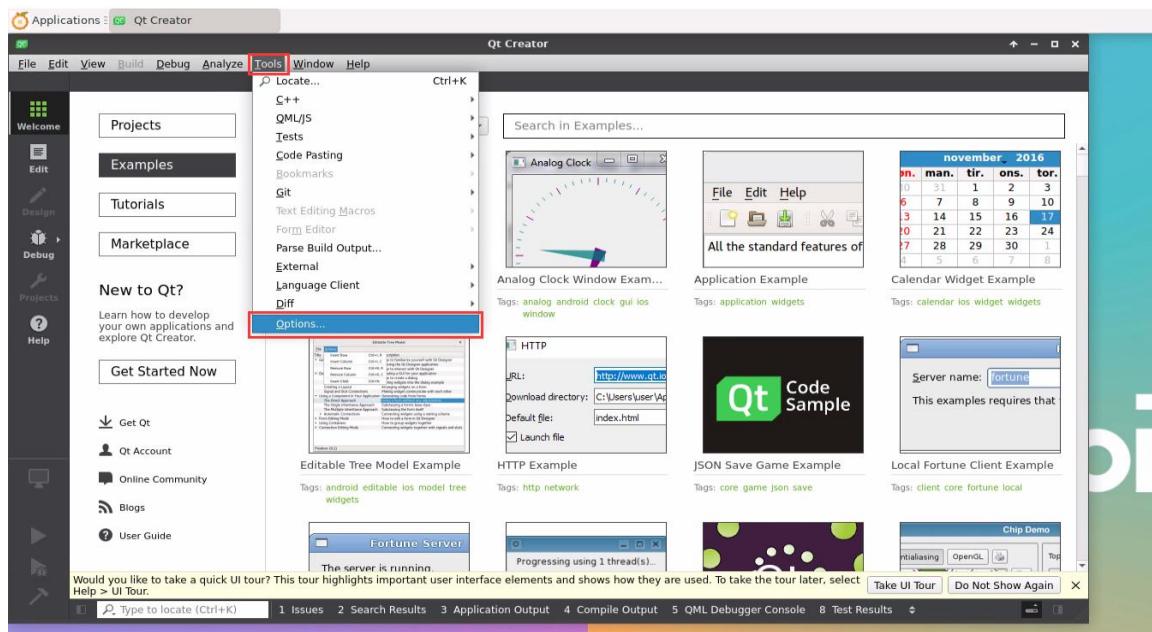
b. 然后去掉 **ClangCodeModel** 的那个勾



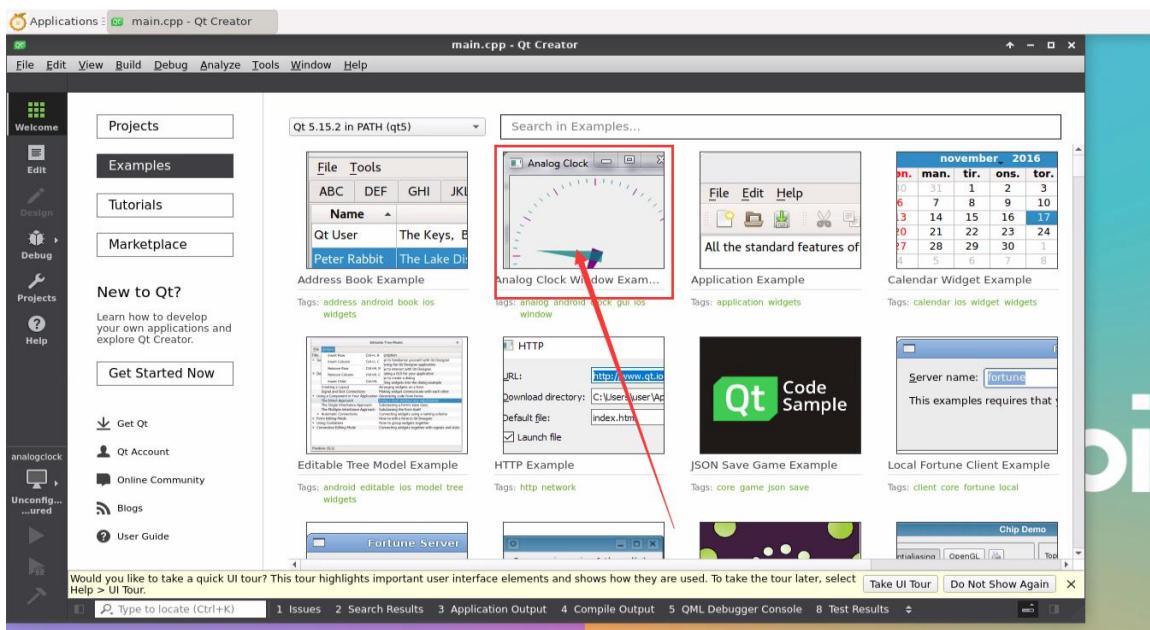
c. 设置完后需要重启下 QT Creator

d. 然后确保 QT Creator 使用的 GCC 编译器, 如果默认为 Clang, 请修改为 GCC

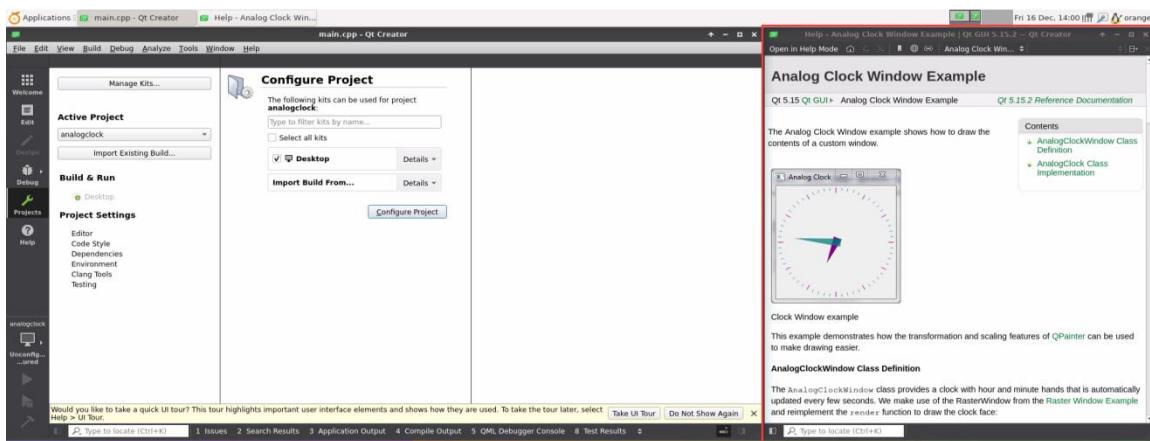
Debian12 请跳过这步。



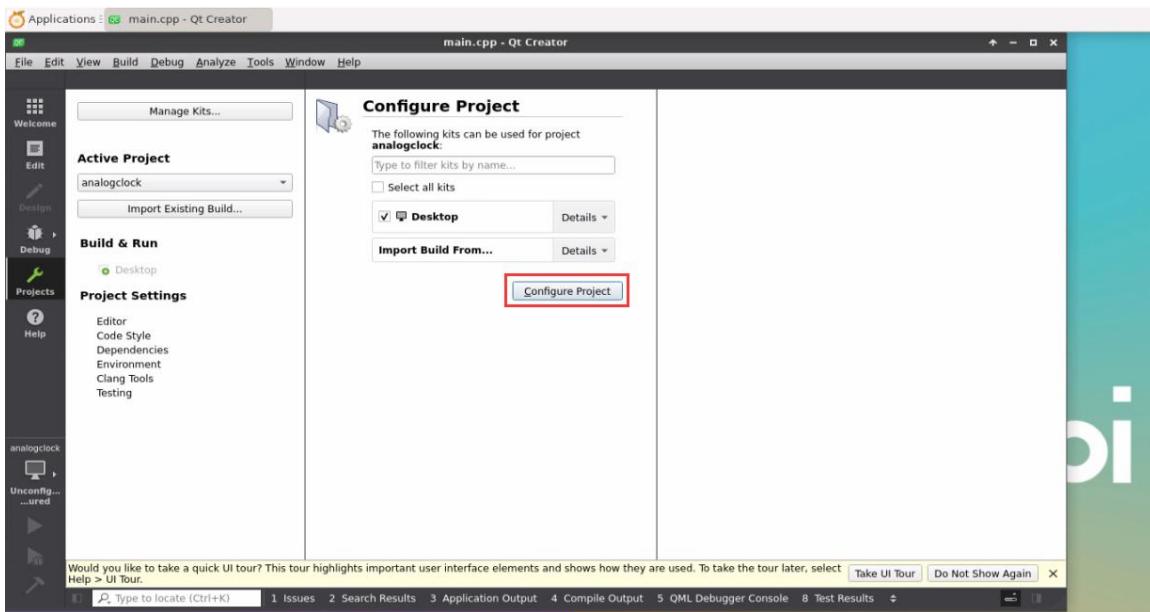
7) 然后就可以打开一个示例代码



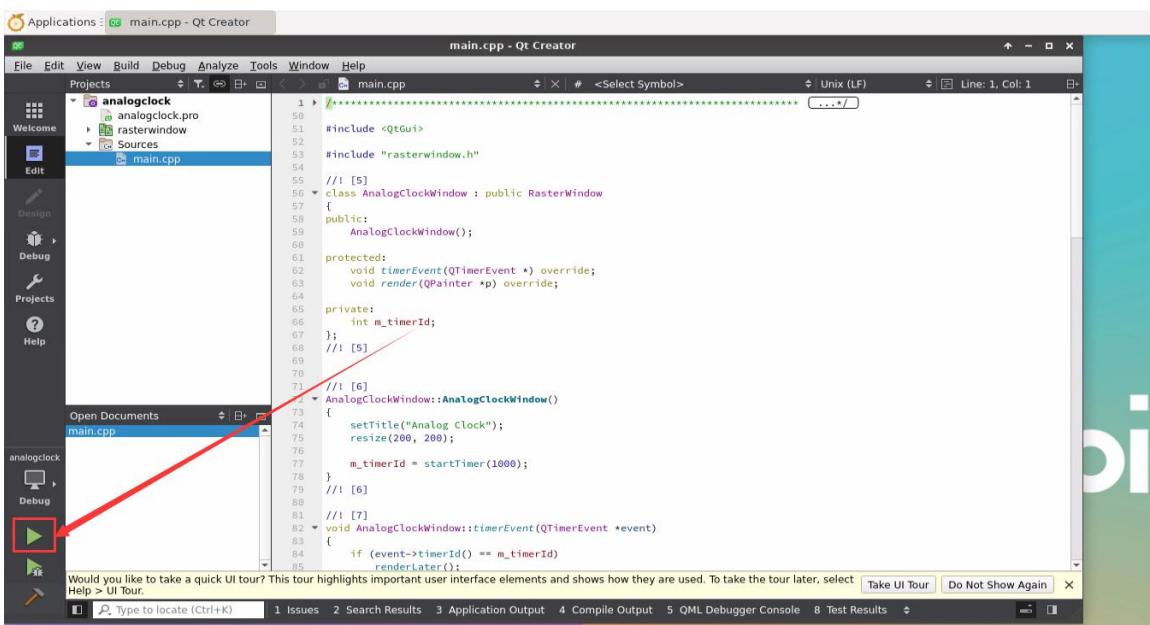
8) 点击示例代码后会自动打开对应的说明文档，可以仔细看下其中的使用说明



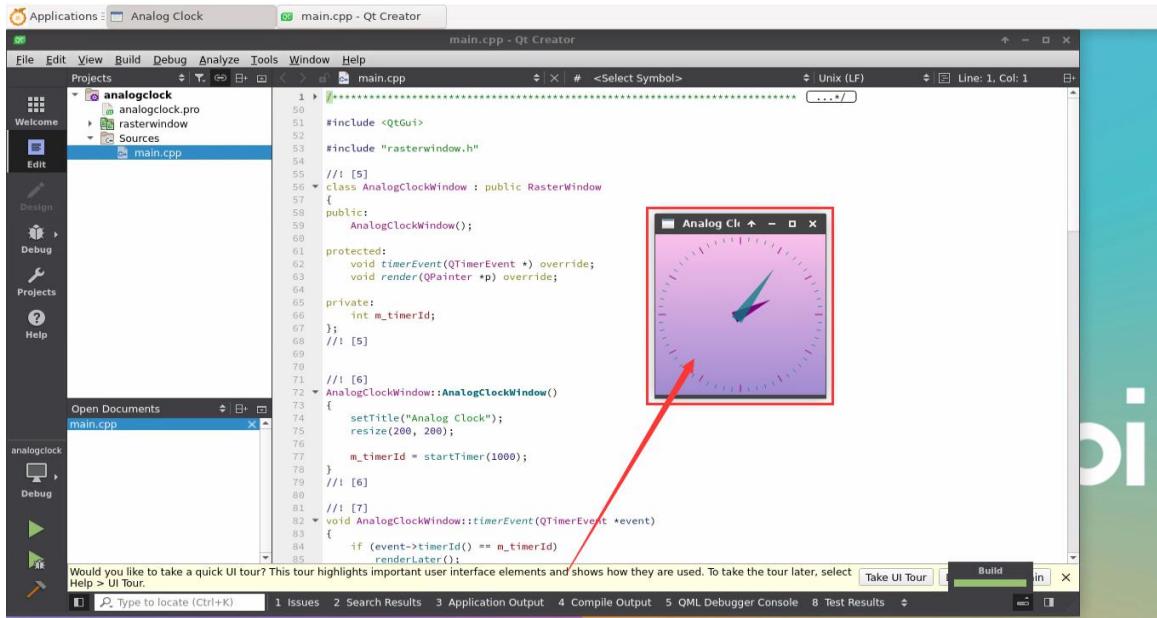
9) 然后点击下 **Configure Project**



10) 然后点击左下角的绿色三角形编译运行下示例代码



11) 等待一段时间后，会弹出下图所示的界面，此时就说明 QT 能正常编译运行



12) 参考资料

https://wiki.qt.io/Install_Qt_5_on_Ubuntu

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

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

3. 31. ROS 安装方法

3. 31. 1. Ubuntu20.04 安装 ROS 1 Noetic 的方法

1) ROS 1 当前活跃的版本如下所示，推荐版本为 **Noetic Ninjemys**

Active ROS 1 distributions

Recommended





Distro	Release date	Poster	Tuturtle, turtle in tutorial	EOL date
ROS Noetic Ninjemys (Recommended)	May 23rd, 2020			May, 2025 (Focal EOL)
ROS Melodic Morenia	May 23rd, 2018			May, 2023 (Bionic EOL)

<http://docs.ros.org>

<https://wiki.ros.org/Distributions>

2) ROS 1 **Noetic Ninjemys** 官方安装文档链接如下所示:

<http://wiki.ros.org/noetic/Installation/Ubuntu>

3) ROS **Noetic Ninjemys** 官方安装文档中 Ubuntu 推荐使用 Ubuntu20.04，所以请确保开发板使用的系统为 **Ubuntu20.04 桌面版系统**

<http://wiki.ros.org/noetic/Installation>

Select Your Platform

Supported:



4) 然后使用下面的脚本安装 ros1

`orangepi@orangepi:~$ install_ros.sh ros1`

5) 使用 ROS 工具前，首先需要初始化下 rosdep，然后编译源码时就能快速的安装一些系统依赖和一些 ROS 中的核心组件

注意，运行下面的命令需要确保开发板能正常访问 **github**，否则会由于网络问题而报错。



install_ros.sh 脚本会尝试修改 `/etc/hosts` 并自动运行下面的命令。但是这种方法无法保证每次都能正常访问 `github`，如果 `install_ros.sh` 安装完 `ros1` 后有提示下面的错误，请自己想其它办法让开发板的 `linux` 系统能正常访问 `github`，然后再手动运行下面的命令。

<https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/osx-homebrew.yaml>

Hit <https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/base.yaml>

ERROR: error loading sources list:

The read operation timed out

```
orangepi@orangepi:~$ source /opt/ros/noetic/setup.bash
```

```
orangepi@orangepi:~$ sudo rosdep init
```

```
Wrote /etc/ros/rosdep/sources.list.d/20-default.list
```

```
Recommended: please run
```

```
rosdep update
```

```
orangepi@orangepi:~$ rosdep update
```

```
reading in sources list data from /etc/ros/rosdep/sources.list.d
```

```
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/osx-homebrew.yaml
```

```
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/base.yaml
```

```
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/python.yaml
```

```
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/ruby.yaml
```

```
Hit https://raw.githubusercontent.com/ros/rosdistro/master/releases/fuerte.yaml
```

```
Query rosdistro index
```

```
https://raw.githubusercontent.com/ros/rosdistro/master/index-v4.yaml
```

```
Skip end-of-life distro "ardent"
```

```
Skip end-of-life distro "bouncy"
```

```
Skip end-of-life distro "crystal"
```

```
Skip end-of-life distro "dashing"
```

```
Skip end-of-life distro "eloquent"
```

```
Add distro "foxy"
```

```
Add distro "galactic"
```

```
Skip end-of-life distro "groovy"
```

```
Add distro "humble"
```

```
Skip end-of-life distro "hydro"
```

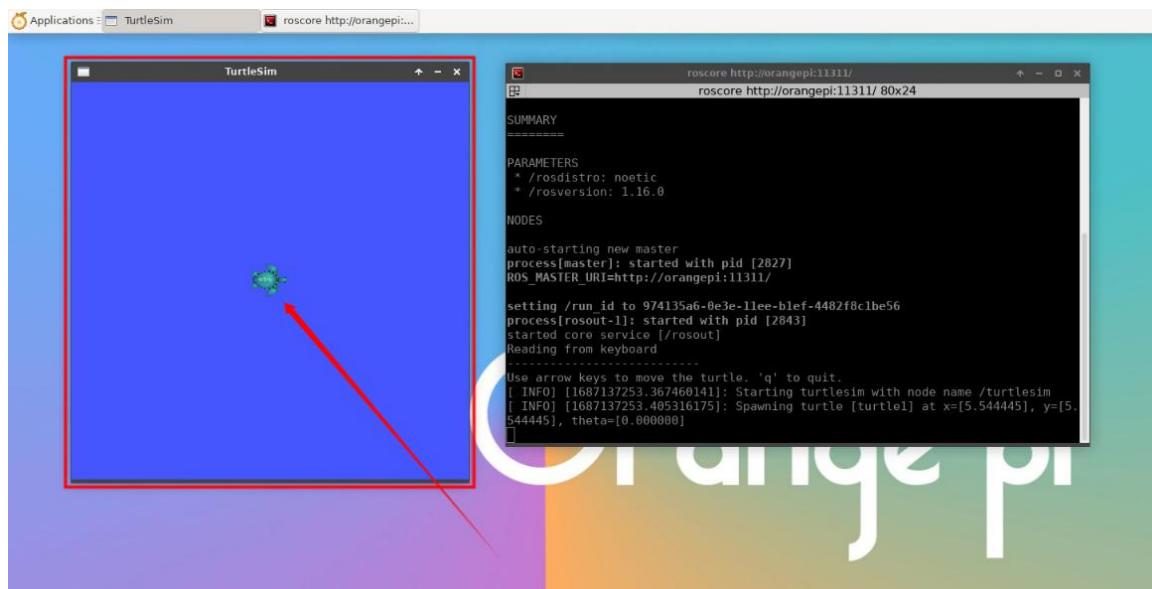


```
Skip end-of-life distro "indigo"
Skip end-of-life distro "jade"
Skip end-of-life distro "kinetic"
Skip end-of-life distro "lunar"
Add distro "melodic"
Add distro "noetic"
Add distro "rolling"
updated cache in /home/orangepi/.ros/rosdep/sources.cache
```

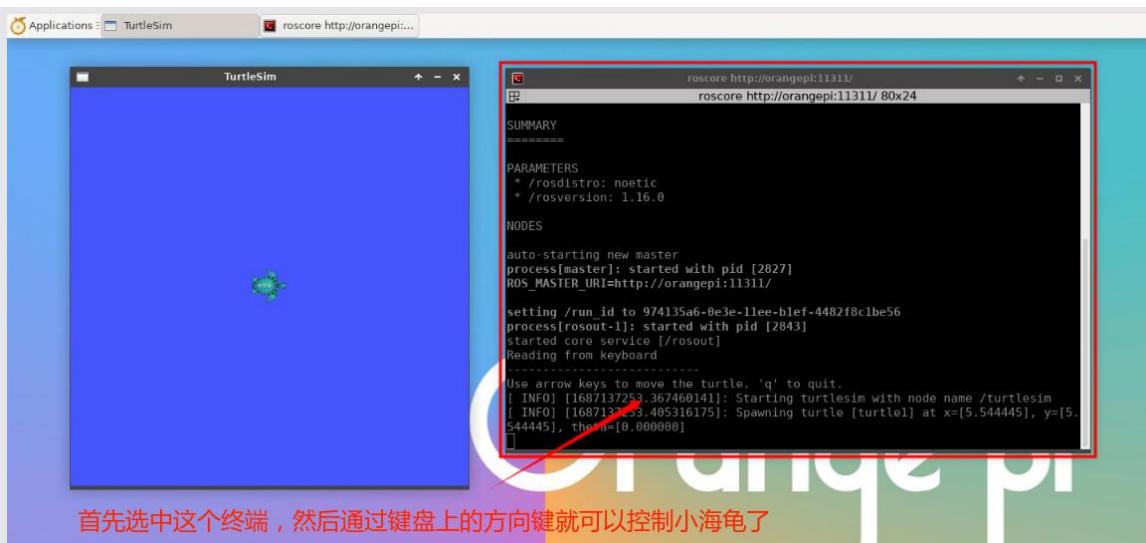
- 6) 然后在桌面中打开一个命令行终端窗口，再使用 **test_ros.sh** 脚本可以启动一个小海龟的例程来测试下 ROS 是否能正常使用

```
orangepi@orangepi:~$ test_ros.sh
```

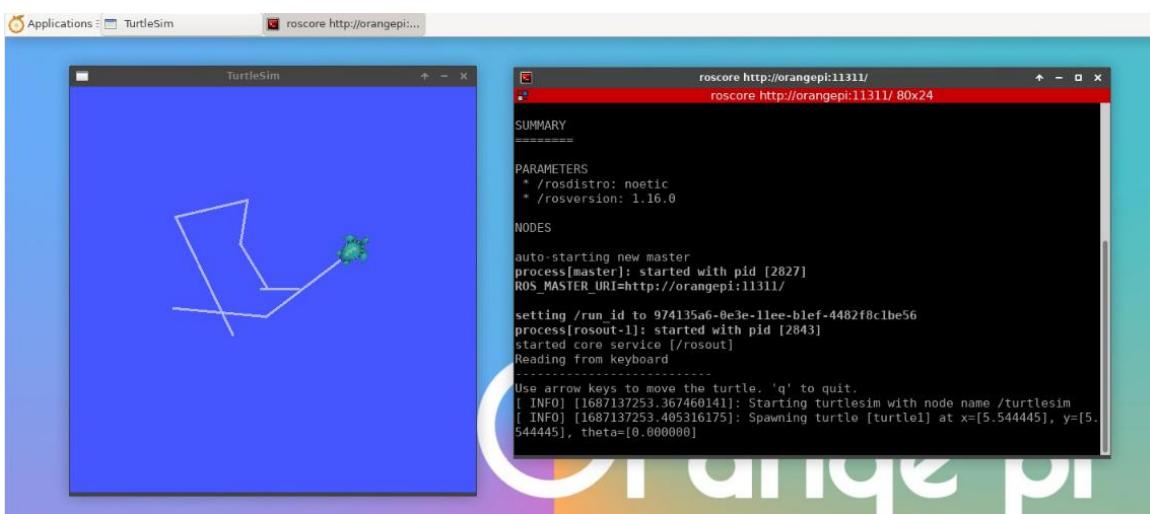
- 7) 运行完 **test_ros.sh** 脚本后，会弹出下图所示的一个小海龟



- 8) 然后请保持刚才打开终端窗口在最上面



9) 此时按下键盘上的方向按键就可以控制小海龟上下左右移动了



3.31.2. Ubuntu20.04 安装 ROS 2 Galactic 的方法

1) ROS 2 当前活跃的版本如下所示，推荐版本为 **Galactic Geochelone**

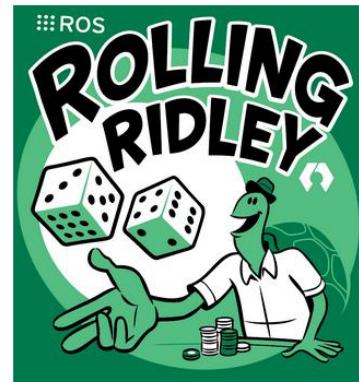


Active ROS 2 distributions

Recommended



Development



Distro	Release date	Logo	EOL date
Humble Hawksbill	May 23rd, 2022		May 2027
Galactic Geochelone	May 23rd, 2021		November 2022
Foxy Fitzroy	June 5th, 2020		May 2023

<http://docs.ros.org>

<http://docs.ros.org/en/galactic/Releases.html>

2) ROS 2 **Galactic Geochelone** 官方安装文档链接如下所示:

docs.ros.org/en/galactic/Installation.html

[http://docs.ros.org/en/galactic/Installation/Ubuntu-Install-Debians.html](https://docs.ros.org/en/galactic/Installation/Ubuntu-Install-Debians.html)

3) ROS 2 **Galactic Geochelone** 官方安装文档中 Ubuntu Linux 推荐使用 Ubuntu20.04, 所以请确保开发板使用的系统为 **Ubuntu20.04 桌面版系统**。安装 ROS 2 有几种方法, 下面演示下通过 **Debian packages** 的方式来安装 ROS 2 **Galactic Geochelone**



4) 使用 **install_ros.sh** 脚本可以安装 ros2

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

5) **install_ros.sh** 脚本安装完 ros2 后会自动运行下 **ros2 -h** 命令，如果能看到下面的打印，说明 ros2 安装完成

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

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

optional arguments:

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

Commands:

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

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

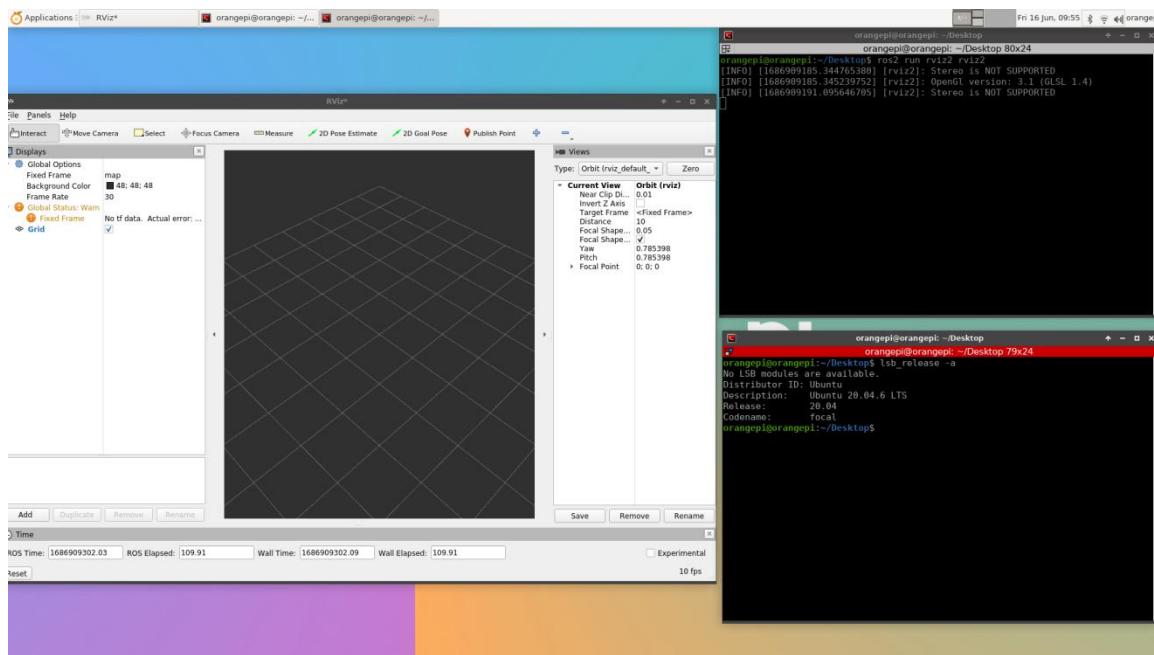
6) 然后可以使用 **test_ros.sh** 脚本测试下 ROS 2 是否安装成功，如果能看到下面的打印，说明 ROS 2 能正常运行



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

7) 运行下面的命令可以打开 rviz2

```
orangeipi@orangeipi:~$ source /opt/ros/galactic/setup.bash
orangeipi@orangeipi:~$ ros2 run rviz2 rviz2
```



8) ROS 的使用方法请参考下 ROS 2 的文档

<http://docs.ros.org/en/galactic/Tutorials.html>

3.31.3. Ubuntu22.04 安装 ROS 2 Humble 的方法

1) 使用 `install_ros.sh` 脚本可以安装 ros2

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

2) `install_ros.sh` 脚本安装完 ros2 后会自动运行下 `ros2 -h` 命令，如果能看到下面的打印，说明 ros2 安装完成



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

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

optional arguments:

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

Commands:

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

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

3) 然后可以使用 **test_ros.sh** 脚本测试下 ROS 2 是否安装成功，如果能看到下面的打印，说明 ROS 2 能正常运行

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

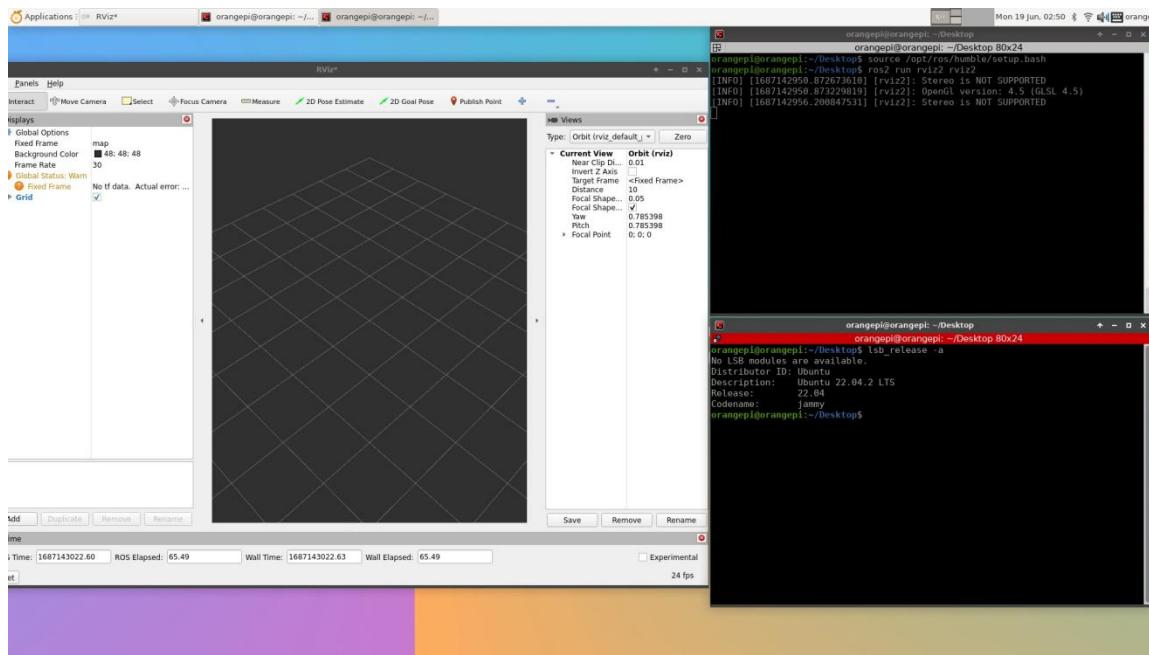


```
[INFO] [1671174103.204019965] [listener]: I heard: [Hello World: 3]
```

4) 运行下面的命令可以打开 rviz2

```
orangeipi@orangeipi:~$ source /opt/ros/humble/setup.bash
```

```
orangeipi@orangeipi:~$ ros2 run rviz2 rviz2
```



5) 参考文档

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

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

3. 32. 安装内核头文件的方法

Linux6.1 内核的 **Debian11** 系统编译内核模块时会报 **GCC** 的错误。所以如果要编译内核模块请使用 **Debian12** 或者 **Ubuntu22.04**。

1) OPi 发布的 Linux 镜像默认自带了内核头文件的 deb 包，存放的位置为/**/opt/**

```
orangeipi@orangeipi:~$ ls /opt/linux-headers*
```

```
/opt/linux-headers-xxxx-sun50iw9_x.x.x_arm64.deb
```

2) 使用下面的命令可以安装内核头文件的 deb 包

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



3) 安装完后在 **/usr/src** 下就能看到内核头文件所在的文件夹

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

4) 然后可以编译下 Linux 镜像中自带的 hello 内核模块的源码，hello 模块的源码在 **/usr/src/hello** 中，进入此目录后，然后使用 make 命令编译即可。

```
orangeipi@orangeipi:~$ cd /usr/src/hello/  
orangeipi@orangeipi:/usr/src/hello$ sudo make  
make -C /lib/modules/5.4.125/build M=/usr/src/hello modules  
make[1]: Entering directory '/usr/src/linux-headers-5.4.125'  
  CC [M]  /usr/src/hello/hello.o  
Building modules, stage 2.  
MODPOST 1 modules  
  CC [M]  /usr/src/hello/hello.mod.o  
  LD [M]  /usr/src/hello/hello.ko  
make[1]: Leaving directory '/usr/src/linux-headers-5.4.125'
```

5) 编译完后会生成 **hello.ko** 内核模块

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

6) 使用 **insmod** 命令可以将 **hello.ko** 内核模块插入内核中

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

7) 然后使用 **dmesg** 命令可以查看下 **hello.ko** 内核模块的输出，如果能看到下面的输出说明 **hello.ko** 内核模块加载正确

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

8) 使用 **rmmmod** 命令可以卸载 **hello.ko** 内核模块

```
orangeipi@orangeipi:/usr/src/hello$ sudo rmmmod hello  
orangeipi@orangeipi:/usr/src/hello$ dmesg | grep "Hello"  
[ 2871.893988] Hello Orange Pi -- init  
[ 3173.800892] Hello Orange Pi -- exit
```



3. 33. Linux 系统支持的部分编程语言测试

3. 33. 1. Debian Bullseye 系统

1) Debian Bullseye 默认安装有 gcc 编译工具链，可以直接在开发板的 Linux 系统中编译 C 语言的程序

a. gcc 的版本如下所示

```
orangeipi@orangeipi:~$ gcc --version
gcc (Debian 10.2.1-6) 10.2.1 20210110
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE.
```

b. 编写 C 语言的 **hello_world.c** 程序

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

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

    return 0;
}
```

c. 然后编译运行 **hello_world.c**

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

2) Debian Bullseye 默认安装有 Python3

a. Python 具体版本如下所示

```
orangeipi@orangeipi:~$ python3
Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```



使用 **Ctrl+D** 快捷键可退出 python 的交互模式。

b. 编写 Python 语言的 **hello_world.py** 程序

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

c. 运行 **hello_world.py** 的结果如下所示

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

3) Debian Bullseye 默认没有安装 Java 的编译工具和运行环境

a. 可以使用下面的命令安装 openjdk, Debian Bullseye 中最新版本为 openjdk-17

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

b. 安装完后可以查看下 Java 的版本

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

c. 编写 Java 版本的 **hello_world.java**

```
orangeipi@orangeipi:~$ vim hello_world.java
public class hello_world
{
    public static void main(String[] args)
    {
        System.out.println("Hello World!");
    }
}
```

d. 然后编译运行 **hello_world.java**

```
orangeipi@orangeipi:~$ javac hello_world.java
```

```
orangeipi@orangeipi:~$ java hello_world
```

```
Hello World!
```

3.33.2. Debian Bookworm 系统

1) Debian Bookworm 默认安装有 gcc 编译工具链，可以直接在开发板的 Linux 系统中编译 C 语言的程序

a. gcc 的版本如下所示

```
orangeipi@orangeipi:~$ gcc --version
gcc (Debian 12.2.0-14) 12.2.0
Copyright (C) 2022 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
```



warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

b. 编写 C 语言的 **hello_world.c** 程序

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

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

    return 0;
}
```

c. 然后编译运行 **hello_world.c**

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

2) Debian Bookworm 默认安装有 Python3

a. Python 具体版本如下所示

```
orangeipi@orangeipi:~$ python3
Python 3.11.2 (main, Mar 13 2023, 12:18:29) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

使用 **Ctrl+D** 快捷键可退出 python 的交互模式。

b. 编写 Python 语言的 **hello_world.py** 程序

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

c. 运行 **hello_world.py** 的结果如下所示

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

3) Debian Bookworm 默认没有安装 Java 的编译工具和运行环境

a. 可以使用下面的命令安装 openjdk，Debian Bookworm 中最新版本为 openjdk-17

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



b. 安装完后可以查看下 Java 的版本

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

c. 编写 Java 版本的 **hello_world.java**

```
orangepi@orangepi:~$ vim hello_world.java
```

```
public class hello_world
{
    public static void main(String[] args)
    {
        System.out.println("Hello World!");
    }
}
```

d. 然后编译运行 **hello_world.java**

```
orangepi@orangepi:~$ javac hello_world.java
```

```
orangepi@orangepi:~$ java hello_world
```

```
Hello World!
```

3. 33. 3. Ubuntu Focal 系统

1) Ubuntu Focal 默认安装有 gcc 编译工具链，可以直接在开发板的 Linux 系统中编译 C 语言的程序

a. gcc 的版本如下所示

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

```
gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0
```

```
Copyright (C) 2019 Free Software Foundation, Inc.
```

```
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE.
```

b. 编写 C 语言的 **hello_world.c** 程序

```
orangepi@orangepi:~$ vim hello_world.c
```

```
#include <stdio.h>
```

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

    return 0;
}
```



{}

c. 然后编译运行 **hello_world.c**

```
orangeipi@orangeipi:~$ gcc -o hello_world hello_world.c
```

```
orangeipi@orangeipi:~$ ./hello_world
```

```
Hello World!
```

2) Ubuntu Focal 默认安装有 Python3

a. Python3 具体版本如下所示

```
orangeipi@orangeipi:~$ python3
```

```
Python 3.8.10 (default, Nov 14 2022, 12:59:47)
```

```
[GCC 9.4.0] on linux
```

```
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>>
```

使用 **Ctrl+D** 快捷键可退出 python 的交互模式。

b. 编写 Python 语言的 **hello_world.py** 程序

```
orangeipi@orangeipi:~$ vim hello_world.py
```

```
print('Hello World!')
```

c. 运行 **hello_world.py** 的结果如下所示

```
orangeipi@orangeipi:~$ python3 hello_world.py
```

```
Hello World!
```

3) Ubuntu Focal 默认没有安装 Java 的编译工具和运行环境

a. 可以使用下面的命令安装 openjdk-17

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

b. 安装完后可以查看下 Java 的版本

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

```
openjdk 17.0.2 2022-01-18
```

```
OpenJDK Runtime Environment (build 17.0.2+8-Ubuntu-120.04)
```

```
OpenJDK 64-Bit Server VM (build 17.0.2+8-Ubuntu-120.04, mixed mode, sharing)
```

c. 编写 Java 版本的 **hello_world.java**

```
orangeipi@orangeipi:~$ vim hello_world.java
```

```
public class hello_world
```

```
{
```

```
    public static void main(String[] args)
```

```
{
```



```
        System.out.println("Hello World!");  
    }  
}
```

d. 然后编译运行 **hello_world.java**

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

3.33.4. Ubuntu Jammy 系统

1) Ubuntu Jammy 默认安装有 gcc 编译工具链, 可以直接在开发板的 Linux 系统中编译 C 语言的程序

a. gcc 的版本如下所示

```
orangeipi@orangeipi:~$ gcc --version  
gcc (Ubuntu 11.3.0-1ubuntu1~22.04.1) 11.3.0  
Copyright (C) 2021 Free Software Foundation, Inc.  
This is free software; see the source for copying conditions. There is NO  
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR  
PURPOSE.
```

b. 编写 C 语言的 **hello_world.c** 程序

```
orangeipi@orangeipi:~$ vim hello_world.c  
#include <stdio.h>  
  
int main(void)  
{  
    printf("Hello World!\n");  
  
    return 0;  
}
```

c. 然后编译运行 **hello_world.c**

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

2) Ubuntu Jammy 默认安装有 Python3

a. Python3 具体版本如下所示



```
orangeipi@orangeipi:~$ python3
Python 3.10.6 (main, May 29 2023, 11:10:38) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

使用 **Ctrl+D** 快捷键可退出 python 的交互模式。

b. 编写 Python 语言的 **hello_world.py** 程序

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

c. 运行 **hello_world.py** 的结果如下所示

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

3) Ubuntu Jammy 默认没有安装 Java 的编译工具和运行环境

a. 可以使用下面的命令安装 openjdk-18

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

b. 安装完后可以查看下 Java 的版本

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

c. 编写 Java 版本的 **hello_world.java**

```
orangeipi@orangeipi:~$ vim hello_world.java
public class hello_world
{
    public static void main(String[] args)
    {
        System.out.println("Hello World!");
    }
}
```

d. 然后编译运行 **hello_world.java**

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



3. 34. 上传文件到开发板 Linux 系统中的方法

3. 34. 1. 在 Ubuntu PC 中上传文件到开发板 Linux 系统中的方法

3. 34. 1. 1. 使用 scp 命令上传文件的方法

1) 使用 scp 命令可以在 Ubuntu PC 中上传文件到开发板的 Linux 系统中，具体命令如下所示

- a. **file_path:** 需要替换为要上传文件的路径
- b. **orangeipi:** 为开发板 linux 系统的用户名，也可以替换成其它的，比如 root
- c. **192.168.xx.xx:** 为开发板的 IP 地址，请根据实际情况进行修改
- d. **/home/orangeipi:** 开发板 linux 系统中的路径，也可以修改为其它的路径

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

2) 如果要上传文件夹，需要加上-r 参数

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

3) scp 还有更多的用法，请使用下面的命令查看 man 手册

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

3. 34. 1. 2. 使用 filezilla 上传文件的方法

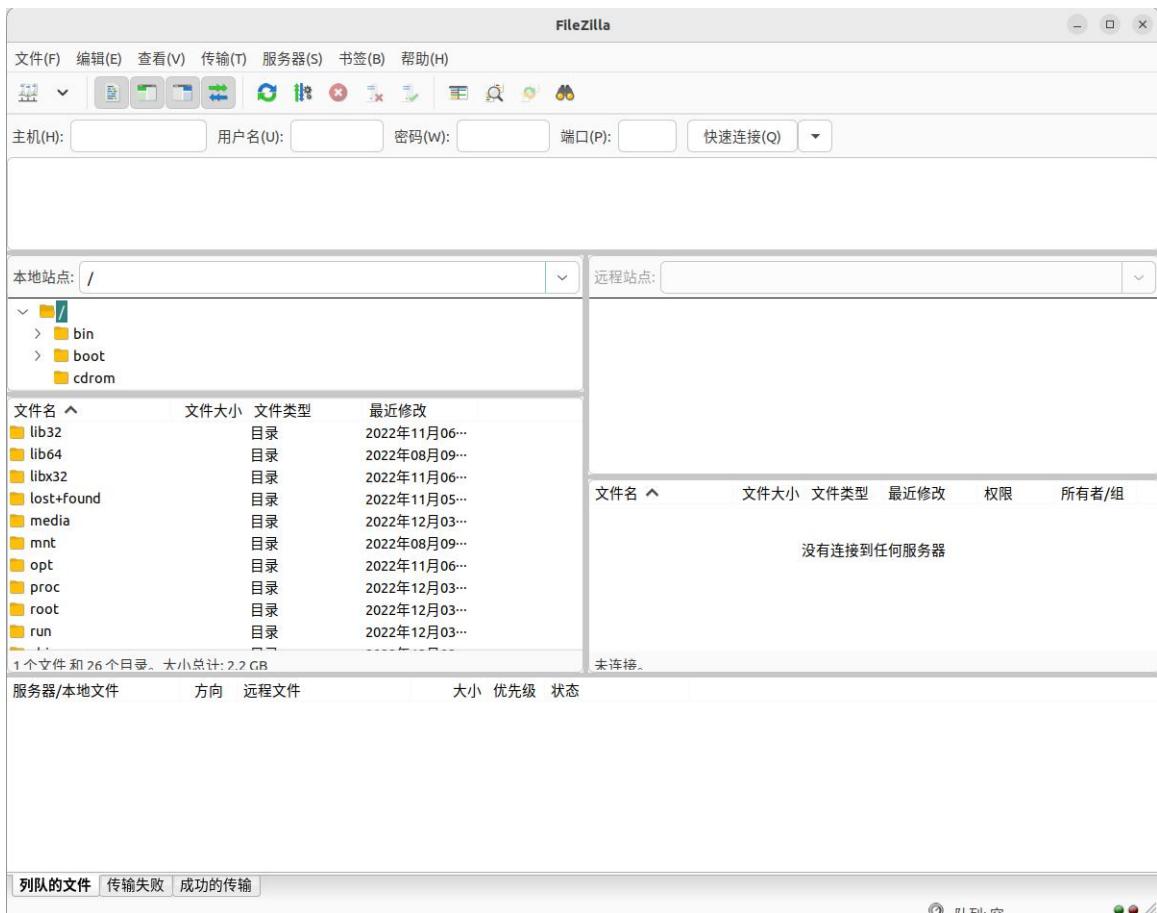
1) 首先在 Ubuntu PC 中安装 filezilla

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

2) 然后使用下面的命令打开 filezilla

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

3) filezilla 打开后的界面如下所示，此时右边远程站点下面显示的是空的



4) 连接开发板的方法如下图所示



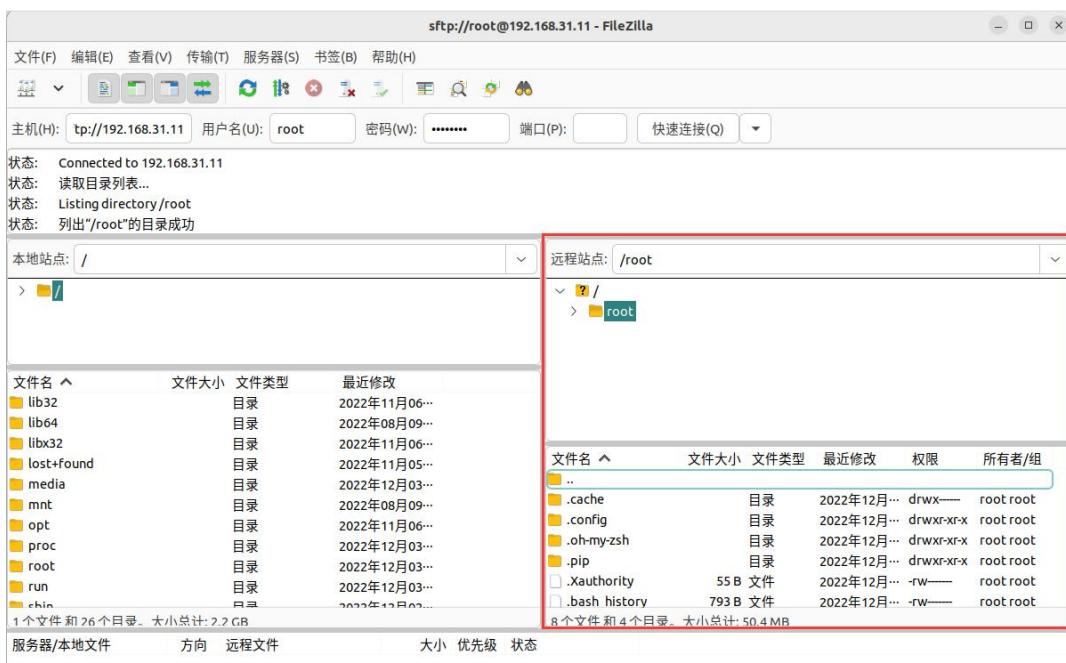
5) 然后选择保存密码，再点击确定



6) 然后选择总是信任该主机，再点击确定



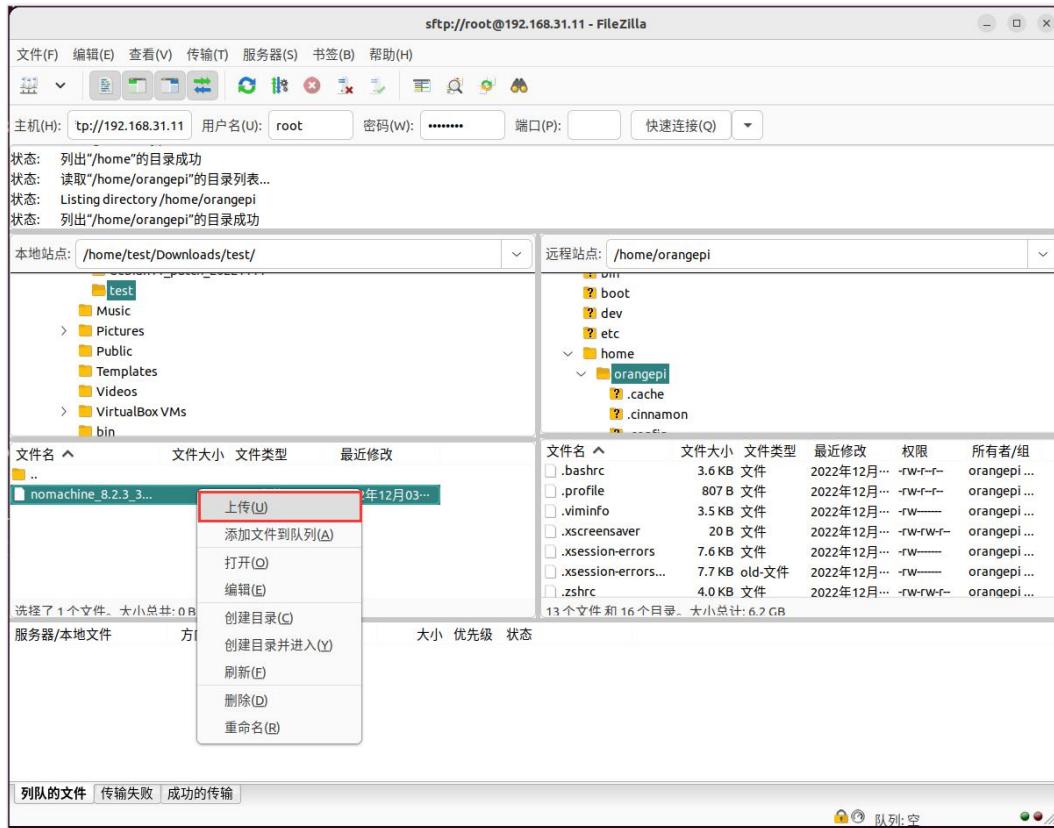
7) 连接成功后在 filezilla 软件的右边就可以看到开发板 linux 文件系统的目录结构了



8) 然后在 filezilla 软件的右边选择要上传到开发板中的路径，再在 filezilla 软件的左边选中 Ubuntu PC 中要上传的文件，再点击鼠标右键，再点击上传选项就会开始上



传文件到开发板中了。



9) 上传完成后就可以去开发板 linux 系统中的对应路径中查看上传的文件了

10) 上传文件夹的方法和上传文件的方法是一样的，这里就不再赘述了

3.34.2. 在 Windows PC 中上传文件到开发板 Linux 系统中的方法

3.34.2.1. 使用 filezilla 上传文件的方法

1) 首先下载 filezilla 软件 Windows 版本的安装文件，下载链接如下所示

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

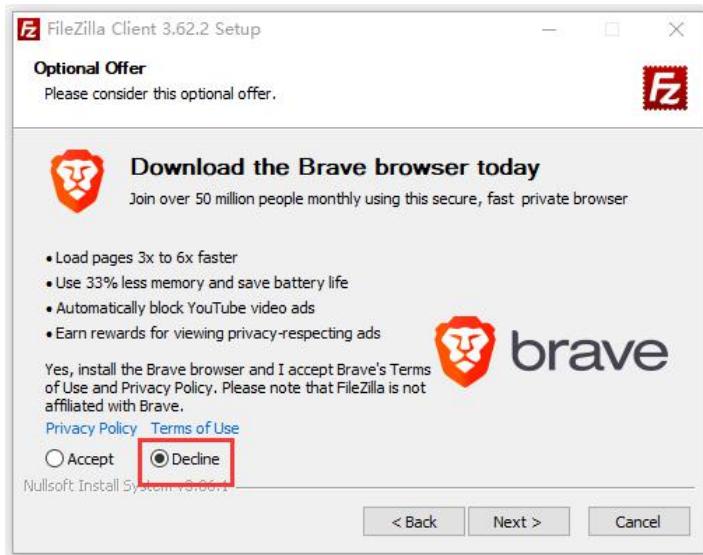


	FileZilla	FileZilla with manual	FileZilla Pro	FileZilla Pro + CLI
Standard FTP	Yes	Yes	Yes	Yes
FTP over TLS	Yes	Yes	Yes	Yes
SFTP	Yes	Yes	Yes	Yes
Comprehensive PDF manual	-	Yes	Yes	Yes
Amazon S3	-	-	Yes	Yes
Backblaze B2	-	-	Yes	Yes
Dropbox	-	-	Yes	Yes
Microsoft OneDrive	-	-	Yes	Yes
Google Drive	-	-	Yes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

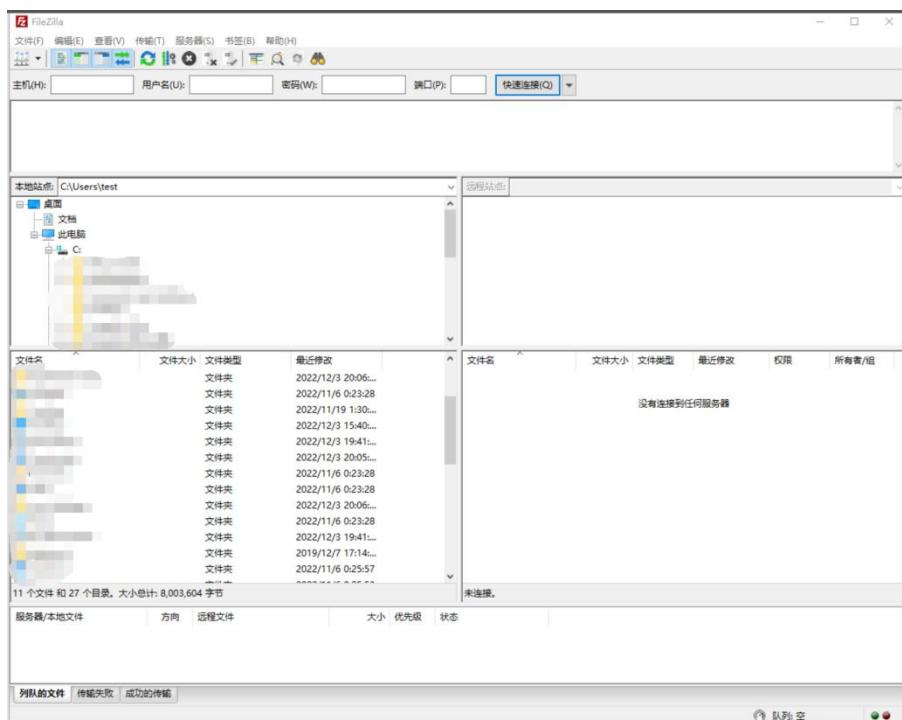
2) 下载的安装包如下所示，然后双击直接安装即可

FileZilla_Server_1.5.1_win64-setup.exe

安装过程中，下面的安装界面请选择 Decline，然后再选择 Next>



3) filezilla 打开后的界面如下所示，此时右边远程站点下面显示的是空的



4) 连接开发板的方法如下图所示：





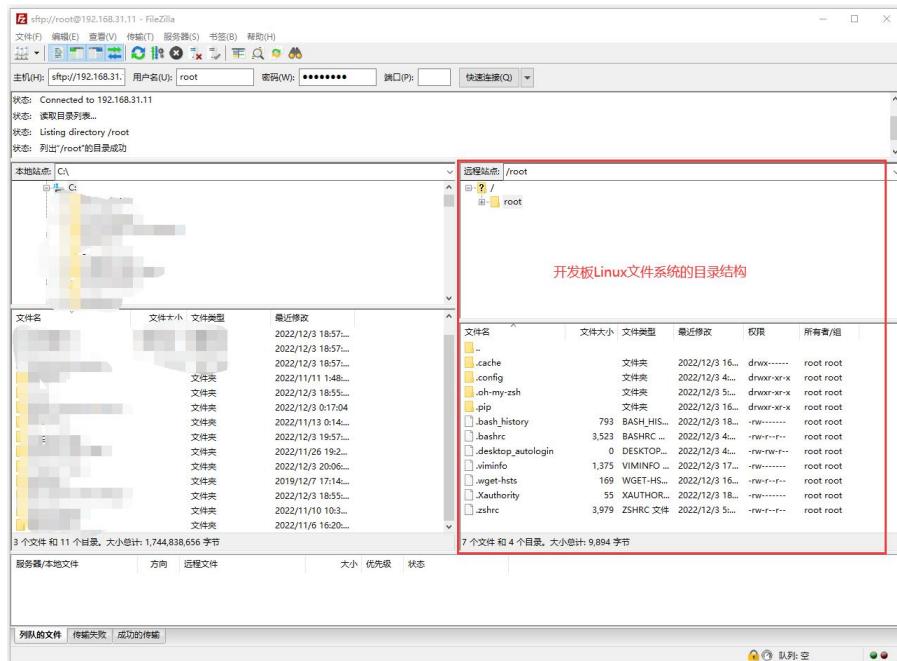
5) 然后选择保存密码，再点击确定



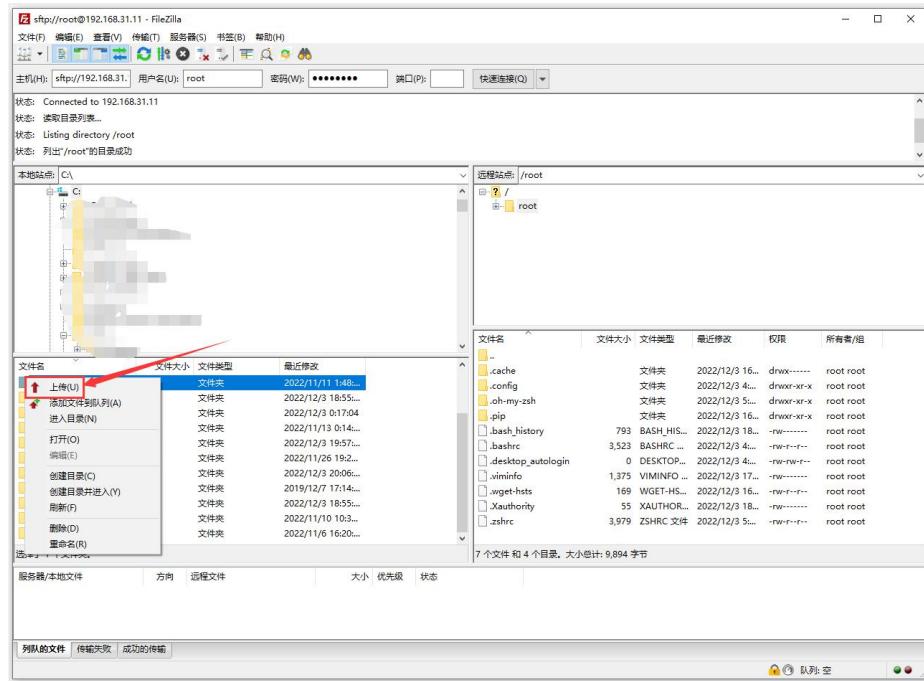
6) 然后选择总是信任该主机，再点击确定



7) 连接成功后在 filezilla 软件的右边就可以看到开发板 linux 文件系统的目录结构了



8) 然后在 filezilla 软件的右边选择要上传到开发板中的路径，再在 filezilla 软件的左边选中 Windows PC 中要上传的文件，再点击鼠标右键，再点击上传选项就会开始上传文件到开发板中了



9) 上传完成后就可以去开发板 linux 系统中的对应路径中查看上传的文件了

10) 上传文件夹的方法和上传文件的方法是一样的，这里就不再赘述了

3. 35. 开关机 logo 使用说明

1) 开关机 logo 默认只在桌面版的系统中才会显示

2) 在`/boot/orangepiEnv.txt` 中设置 `bootlogo` 变量为 `false` 可以关闭开关机 logo

```
orangepi@orangepi:~$ sudo vim /boot/orangepiEnv.txt
verbosity=1
bootlogo=false
```

3) 在`/boot/orangepiEnv.txt` 中设置 `bootlogo` 变量为 `true` 可以开启开关机 logo

```
orangepi@orangepi:~$ sudo vim /boot/orangepiEnv.txt
verbosity=1
bootlogo=true
```

4) 开机 logo 图片在 linux 系统中的位置为



```
/usr/share/plymouth/themes/orangepi/watermark.png
```

5) 替换开机 logo 图片后需要运行下命令才能生效

```
orangepi@orangepi:~$ sudo update-initramfs -u
```

3. 36. Linux5.4 打开开关机按键的方法

开发板主板上没有开关机按键，我们可以通过 24pin 扩展板来扩展。开关机按键在扩展板上的位置如下所示：

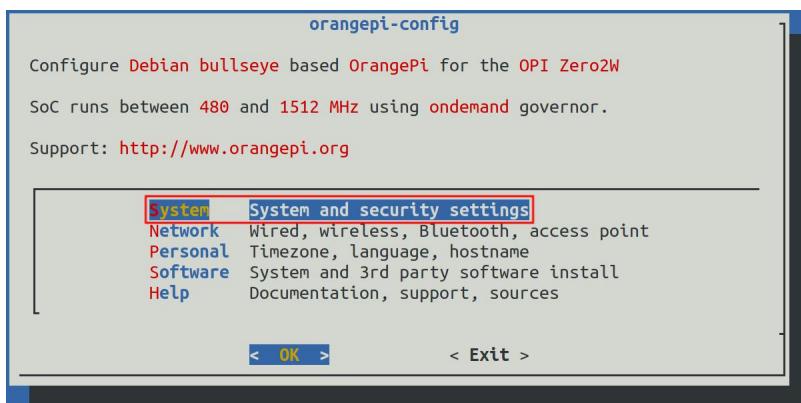


Linux6.1 镜像开关机按键默认是打开的，但 Linux5.4 内核镜像的开关机按键默认是关闭的，需要手动打开才能正常使用。步骤如下所示：

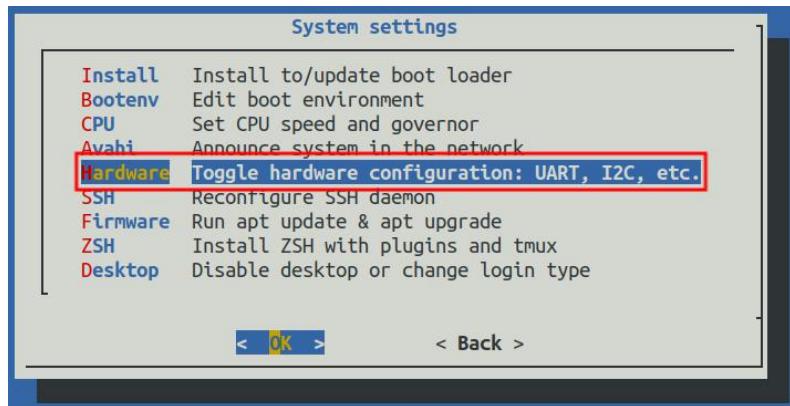
1) 首先运行下 **orangepi-config**，普通用户记得加 **sudo** 权限

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

2) 然后选择 **System**



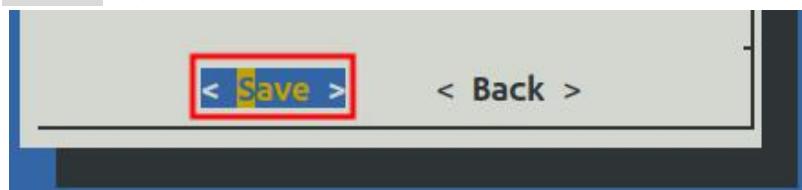
3) 然后选择 **Hardware**



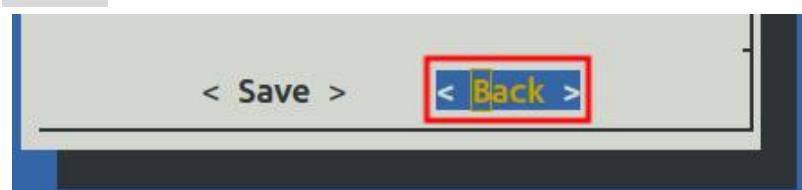
4) 然后使用键盘的方向键定位到下图所示的位置，再使用空格选中想要打开的 SPI 的 dtbo 配置



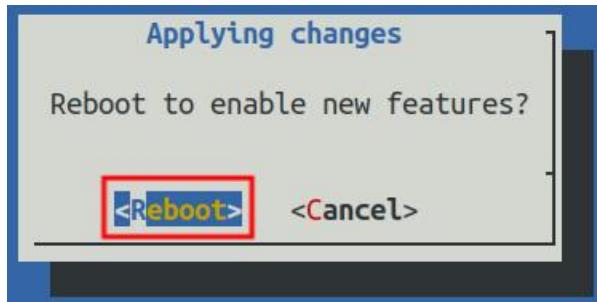
5) 然后选择<Save>保存



6) 然后选择<Back>



7) 然后选择<Reboot>重启系统使配置生效



3. 37. 关机和重启开发板的方法

- 1) 在 Linux 系统运行的过程中，如果直接拔掉电源断电，可能会导致文件系统丢失某些数据，建议断电前先使用 **poweroff** 命令关闭开发板的 Linux 系统，然后再拔掉电源

```
orangeipi@orangeipi:~$ sudo poweroff
```

注意，关闭开发板后需要重新拔插电源才能开机。

- 2) 除了使用 poweroff 命令关机外，还可以使用扩展板上的开关机按键来关机



注意，Linux5.4 需要手动打开开关机按键的配置才能使用。打开方法请参考 [Linux5.4 打开开关机按键的方法。](#)

- 3) 使用 **reboot** 命令即可重启开发板中的 Linux 系统

```
orangeipi@orangeipi:~$ sudo reboot
```



4. Linux SDK——orangeipi-build 使用说明

4.1. 编译系统需求

Linux SDK，即 **orangeipi-build**，只支持在安装有 **Ubuntu 22.04** 的 X64 电脑上运行，所以下载 orangeipi-build 前，请首先确保自己电脑已安装的 Ubuntu 版本是 Ubuntu 22.04。查看电脑已安装的 Ubuntu 版本的命令如下所示，如果 Release 字段显示的不是 **22.04**，说明当前使用的 Ubuntu 版本不符合要求，请更换系统后再进行下面的操作。

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

如果电脑安装的是 Windows 系统，没有安装有 Ubuntu 22.04 的电脑，可以考虑使用 **VirtualBox** 或者 **VMware** 来在 Windows 系统中安装一个 Ubuntu 22.04 虚拟机。但是请注意，不要在 WSL 虚拟机上编译 orangeipi-build，因为 orangeipi-build 没有在 WSL 虚拟机中测试过，所以无法确保能正常在 WSL 中使用 orangeipi-build，另外请不要在开发板的 Linux 系统中使用 orangeipi-build。Ubuntu 22.04 **amd64** 版本的安装镜像下载地址为：

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

在电脑中或者虚拟机中安装完 Ubuntu 22.04 后，请先设置 Ubuntu 22.04 的软件源为清华源（或者其它你觉得速度快的国内源），不然后面安装软件的时候很容易由于网络原因而出错。替换清华源的步骤如下所示：

- 替换清华源的方法参考这个网页的说明即可。

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

- 注意 Ubuntu 版本需要切换到 22.04。



Ubuntu 镜像使用帮助

Ubuntu 的软件源配置文件是 `/etc/apt/sources.list`。将系统自带的该文件做个备份，将该文件替换为下面内容，即可使用 TUNA 的软件源镜像。

选择你的ubuntu版本:

22.04 LTS

```
# 默认注释了源码镜像以提高 apt update 速度，如有需要可自行取消注释
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-updates main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-updates main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-backports main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-backports main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-security main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-security main restricted universe multiverse

# 预发布软件源，不建议启用
# deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-proposed main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-proposed main restricted universe multiverse
```

c. 需要替换的`/etc/apt/sources.list`文件的内容为:

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

# 默认注释了源码镜像以提高 apt update 速度，如有需要可自行取消注释
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-updates main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-updates main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-backports main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-backports main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-security main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-security main restricted universe multiverse

# 预发布软件源，不建议启用
# deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-proposed main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-proposed main restricted universe multiverse
```

d. 替换完后需要更新下包信息，并确保没有报错。

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

e. 另外，由于内核和 U-boot 等源码都是存放在 GitHub 上的，所以编译镜像的时候请确保电脑能正常从 GitHub 下载代码，这点是非常重要的。



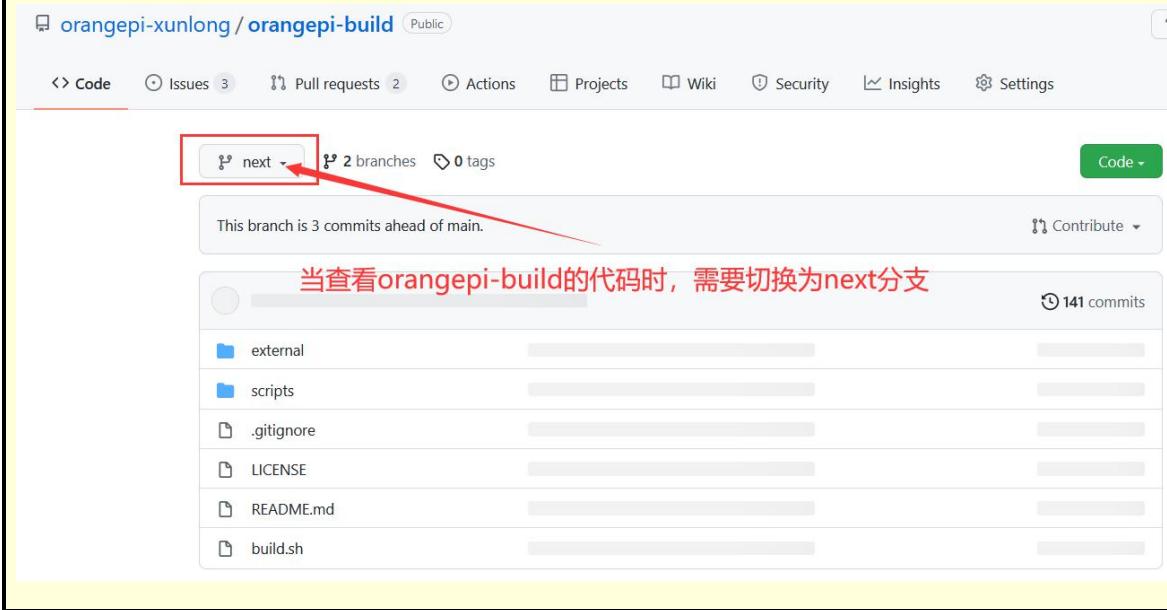
4.2. 获取 linux sdk 的源码

4.2.1. 从 github 下载 orangepi-build

linux sdk 指的是 orangepi-build 这套代码，orangepi-build 是基于 armbian build 编译系统修改而来的，使用 orangepi-build 可以编译出多个版本的 linux 镜像。使用下面的命令可以下载 orangepi-build 的代码：

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

注意，使用 H618 Soc 的开发板是需要下载 orangepi-build 的 **next** 分支源码的，上面的 **git clone** 命令需要指定 orangepi-build 源码的分支为 **next**。



通过 **git clone** 命令下载 orangepi-build 的代码是不需要输入 **github** 账号的用户名和密码的（下载本手册中的其他代码也是一样的），如果如输入 **git clone** 命令后 Ubuntu PC 提示需要输入 **github** 账号的用户名和密码，一般都是 **git clone** 后面的 orangepi-build 仓库的地址输入错误了，请仔细检查命令拼写是否有错误，而不是以为我们这里忘了提供 **github** 账号的用户名和密码。



H618 系列开发板当前使用的 u-boot 和 linux 内核版本如下所示：

分支	u-boot 版本	linux 内核版本
current	u-boot v2018.05	linux5.4
next	u-boot v2021.07	linux6.1

这里所说的分支和 orangepi-build 源代码的分支不是同一个东西，请不要搞混了。此分支主要是用来区分不同内核源码版本的。

目前全志提供的 linux5.4 bsp 内核我们定义为 current 分支。最新的 linux6.1 LTS 内核定义为 next 分支。

orangepi-build 下载完后会包含下面的文件和文件夹：

- a. **build.sh**: 编译启动脚本
- b. **external**: 包含编译镜像需要用的配置文件、特定的脚本以及部分程序的源码等
- c. **LICENSE**: GPL 2 许可证文件
- d. **README.md**: orangepi-build 说明文件
- e. **scripts**: 编译 linux 镜像的通用脚本

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

如果是从 github 下载的 orangepi-build 的代码，下载完后你可能会发现 orangepi-build 中并没有包含 u-boot 和 linux 内核的源码，也没有编译 u-boot 和 linux 内核需要用到交叉编译工具链，这是正常的，因为这些东西都存放在其它单独的 github 仓库或者某些服务器上了（下文会详述其地址）。orangepi-build 在脚本和配置文件中会指定 u-boot、linux 内核和交叉编译工具链的地址，运行 orangepi-build 时，当其发现本地没有这些东西，会自动去相应的地方下载的。

4. 2. 2. 下载交叉编译工具链

orangepi-build 第一次运行的时候会自动下载交叉编译工具链放在 **toolchains** 文件夹中，每次运行 orangepi-build 的 build.sh 脚本后，都会检查 **toolchains** 中的交叉编译工具链是否都存在，如果不存在则会重新开始下载，如果存在则直接使用，不会重复下载。



```
[ o.k. ] Checking for external GCC compilers
[ ... ] downloading using https(s) network [ gcc-linaro-aarch64-none-elf-4.8-2013.11_linux.tar.xz ]
#8d7029 16MiB/24MiB (65%) CN:1 DL:7.9MiB ETA:1s
[ o.k. ] Verified [ PGP ]
[ ... ] decompressing
[ ... ] gcc-linaro-aarch64-none-elf-4.8-2013.11_linux.tar.xz: 24.9MiB [14.4MiB/s] [=====>] 100%
[ ... ] downloading using https(s) network [ gcc-linaro-arm-none-eabi-4.8-2014.04_linux.tar.xz ]
#e30eec 17MiB/33MiB (50%) CN:1 DL:10MiB ETA:1s
[ o.k. ] Verified [ PGP ]
[ ... ] decompressing
[ ... ] gcc-linaro-arm-none-eabi-4.8-2014.04_linux.tar.xz: 33.9MiB [9.6MiB/s] [=====>] 100%
[ ... ] downloading using https(s) network [ gcc-linaro-arm-linux-gnueabihf-4.8-2014.04_linux.tar.xz ]
#041c24 49MiB/49MiB (99%) CN:1 DL:2.7MiB
[ o.k. ] Verified [ PGP ]
[ ... ] decompressing
[ ... ] gcc-linaro-arm-linux-gnueabihf-4.8-2014.04_linux.tar.xz: 48.8MiB [13.0MiB/s] [=====>] 100%
[ ... ] downloading using https(s) network [ gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabi.tar.xz ]
#3dee3e 72MiB/76MiB (93%) CN:1 DL:3.7MiB ETA:1s
[ o.k. ] Verified [ MD5 ]
[ ... ] decompressing
[ ... ] gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabi.tar.xz: 77.0MiB [14.2MiB/s] [=====>] 100%
[ ... ] downloading using https(s) network [ gcc-linaro-7.4.1-2019.02-x86_64_arm-linux-gnueabi.tar.xz ]
#42c728 104MiB/104MiB (99%) CN:1 DL:2.8MiB
[ o.k. ] Verified [ MD5 ]
[ ... ] decompressing
[ ... ] gcc-linaro-7.4.1-2019.02-x86_64_arm-linux-gnueabi.tar.xz: 104MiB [13.9MiB/s] [=====>] 100%
[ ... ] downloading using https(s) network [ gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu.tar.xz ]
#2c065e 108MiB/111MiB (97%) CN:1 DL:3.9MiB
[ o.k. ] Verified [ MD5 ]
[ ... ] decompressing
[ ... ] gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu.tar.xz: 111MiB [13.4MiB/s] [=====>] 100%
[ ... ] downloading using https(s) network [ gcc-arm-9.2-2019.12-x86_64-arm-none-linux-gnueabihf.tar.xz ]
#d232ee 250MiB/251MiB (99%) CN:1 DL:2.0MiB
[ o.k. ] Verified [ MD5 ]
[ ... ] decompressing
[ ... ] gcc-arm-9.2-2019.12-x86_64-arm-none-linux-gnueabihf.tar.xz: 251MiB [13.7MiB/s] [=====>] 100%
[ ... ] downloading using https(s) network [ gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu.tar.xz ]
#88b441 268MiB/269MiB (99%) CN:1 DL:0.9MiB
[ o.k. ] Verified [ MD5 ]
[ ... ] decompressing
```

交叉编译工具链在中国境内的镜像网址为清华大学的开源软件镜像站：

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

toolchains 下载完后会包含多个版本的交叉编译工具链：

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

编译 H618 Linux 内核源码使用的交叉编译工具链为：

a. linux5.4

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

b. linux6.1

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

编译 H618 u-boot 源码使用的交叉编译工具链为：

- a. v2018.05

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

- b. v2021.07

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

4. 2. 3. orangepi-build 完整目录结构说明

1) orangepi-build 仓库下载完后并不包含 linux 内核、u-boot 的源码以及交叉编译工具链，linux 内核和 u-boot 的源码存放在独立的 git 仓库中

- a. linux 内核源码存放的 git 仓库如下，注意切换 linux-orangepi 仓库的分支为

- a) Linux5.4

<https://github.com/orangepi-xunlong/linux-orangepi/tree/orange-pi-5.4-sun50iw9>

- b) Linux6.1

<https://github.com/orangepi-xunlong/linux-orangepi/tree/orange-pi-6.1-sun50iw9>

- b. u-boot 源码存放的 git 仓库如下，注意切换 u-boot-orangepi 仓库的分支为

- a) v2018.05

<https://github.com/orangepi-xunlong/u-boot-orangepi/tree/v2018.05-h618>

- b) v2021.07

<https://github.com/orangepi-xunlong/u-boot-orangepi/tree/v2021.07-sunxi>

2) orangepi-build 第一次运行的时候会去下载交叉编译工具链、u-boot 和 linux 内核源码，成功编译完一次 linux 镜像后在 orangepi-build 中可以看到的文件和文件夹有

- a. **build.sh**: 编译启动脚本
- b. **external**: 包含编译镜像需要用的配置文件、特定功能的脚本以及部分程序的源码，编译镜像过程中缓存的 rootfs 压缩包也存放在 external 中
- c. **kernel**: 存放 linux 内核的源码
- d. **LICENSE**: GPL 2 许可证文件
- e. **README.md**: orangepi-build 说明文件
- f. **output**: 存放编译生成的 u-boot、linux 等 deb 包、编译日志以及编译生成的镜像等文件
- g. **scripts**: 编译 linux 镜像的通用脚本
- h. **toolchains**: 存放交叉编译工具链
- i. **u-boot**: 存放 u-boot 的源码



j. **userpatches**: 存放编译脚本需要用到的配置文件

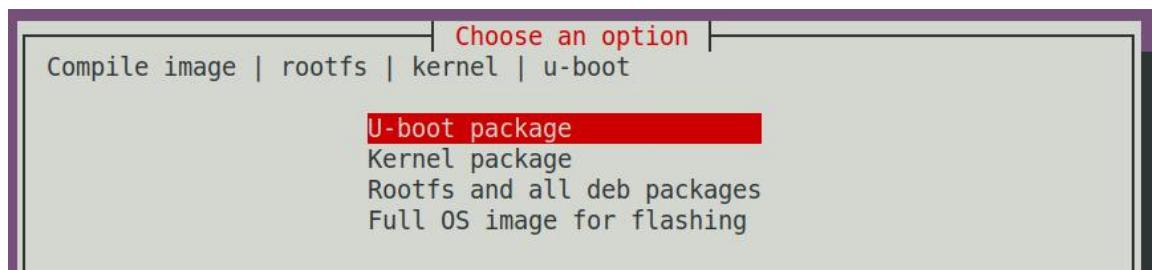
```
test@test:~/orangepi-build$ ls
build.sh  external  kernel  LICENSE  output  README.md  scripts  toolchains
u-boot  userpatches
```

4. 3. 编译 u-boot

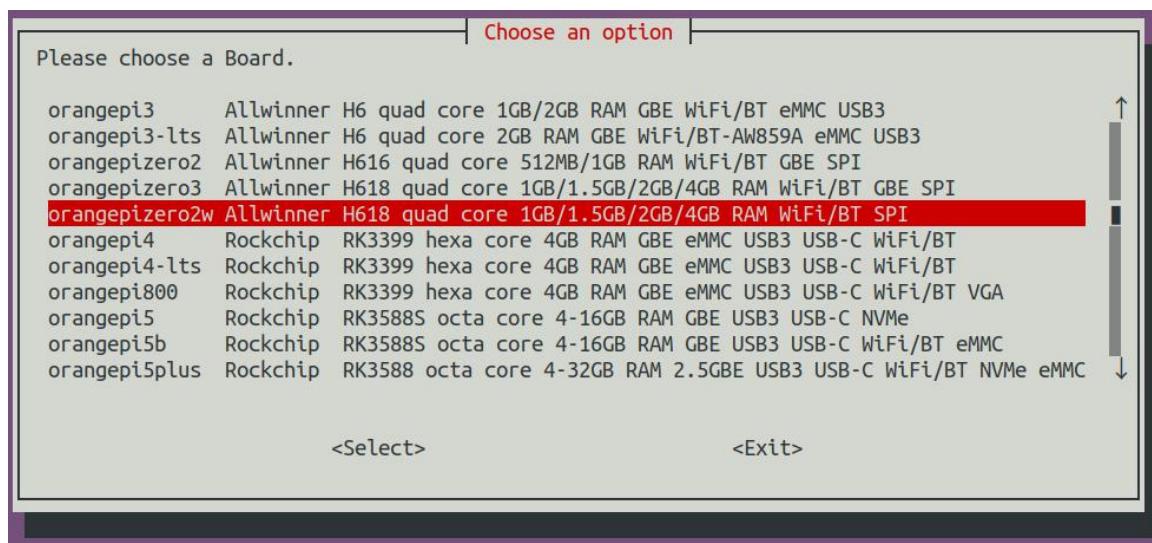
1) 运行 build.sh 脚本，记得加 sudo 权限

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

2) 选择 **U-boot package**, 然后回车

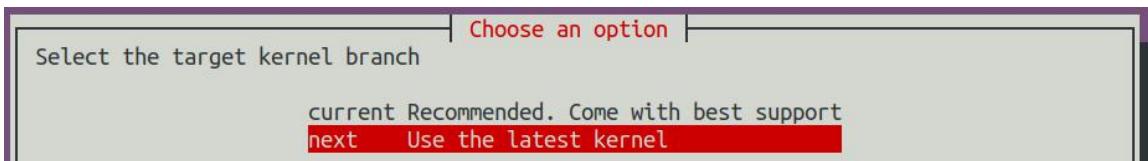


3) 接着选择开发板的型号

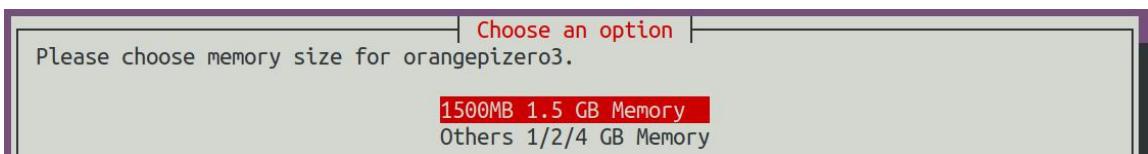


4) 然后选择 u-boot 的分支类型

- current 分支会编译 linux5.4 镜像需要使用的 u-boot v2018.05 版本的代码
- next 分支会编译 linux6.1 镜像需要使用的 u-boot v2021.07 版本的代码



- 5) 如果选择的 next 分支还会提示需要选择内存的大小，current 分支不需要选择
- 如果购买的开发板为 1.5GB 内存大小的，请选择第一项
 - 如果购买的开发板为 1GB 或 2GB 或 4GB 内存大小的，请选择第二项



- 6) 然后就会开始编译 u-boot，编译 next 分支时提示的部分信息说明如下所示：

- u-boot 源码的版本

[o.k.] Compiling u-boot [v2021.07]

- 交叉编译工具链的版本

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

- 编译生成的 u-boot deb 包的路径

[o.k.] Target directory [orangepi-build/output/debs/u-boot]

- 编译生成的 u-boot deb 包的包名

[o.k.] File name [linux-u-boot-next-orangepizero2w_x.x.x_arm64.deb]

- 编译使用的时间

[o.k.] Runtime [1 min]

- 重复编译 u-boot 的命令，使用下面的命令无需通过图形界面选择，可以直接开始编译 u-boot

[o.k.] Repeat Build Options [sudo ./build.sh BOARD=orangepizero2w

BRANCH=next BUILD_OPT=u-boot]

- 7) 查看编译生成的 u-boot deb 包

```
test@test:~/orangepi-build$ ls output/debs/u-boot/  
linux-u-boot-next-orangepizero2w_x.x.x_arm64.deb
```

- 8) orangepi-build 编译系统编译 u-boot 源码时首先会将 u-boot 的源码和 github 服务器的 u-boot 源码进行同步，所以如果想修改 u-boot 的源码，首先需要关闭源码的下



载更新功能（需要完整编译过一次 **u-boot** 后才能关闭这个功能，否则会提示找不到 **u-boot** 的源码），否则所作的修改都会被还原，方法如下：

设置 **userpatches/config-default.conf** 中的 IGNORE_UPDATES 变量为 “yes”

```
test@test:~/orangepi-build$ vim userpatches/config-default.conf
```

```
.....
```

```
IGNORE_UPDATES="yes"
```

```
.....
```

9) 调试 u-boot 代码时，可以使用下面的方法来更新 linux 镜像中的 u-boot 进行测试

a. 首先将编译好的 u-boot 的 deb 包上传到开发板的 linux 系统中

```
test@test:~/orangepi-build$ cd output/debs/u-boot
```

```
test@test:~/orangepi_build/output/debs/u-boot$ scp \
```

```
linux-u-boot-next-orangepizero2w_x.x.x_arm64.deb root@192.168.1.xxx:/root
```

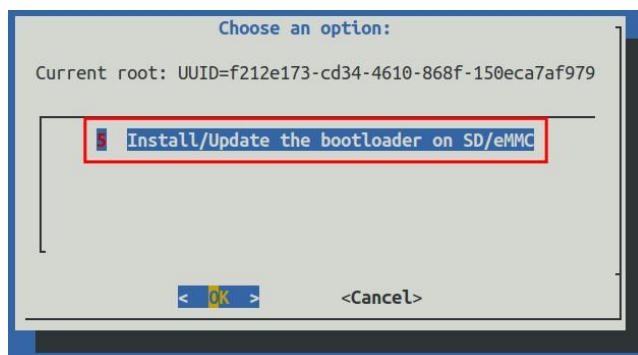
b. 再安装刚才上传的新的 u-boot 的 deb 包

```
orangepi@orangepi:~$ sudo dpkg -i linux-u-boot-next-orangepizero2w_x.x.x_arm64.deb
```

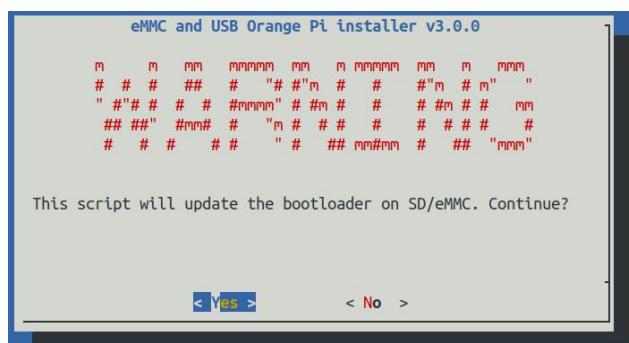
c. 然后运行 nand-sata-install 脚本

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

d. 然后选择 **5 Install/Update the bootloader on SD/eMMC**

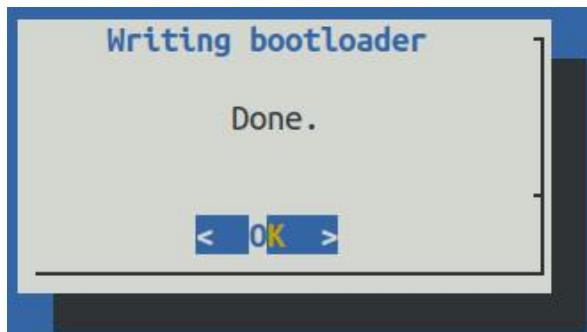


e. 按下回车键后首先会弹出一个 Warning





f. 再按下次回车键就会开始更新 u-boot，更新完后会显示下面的信息



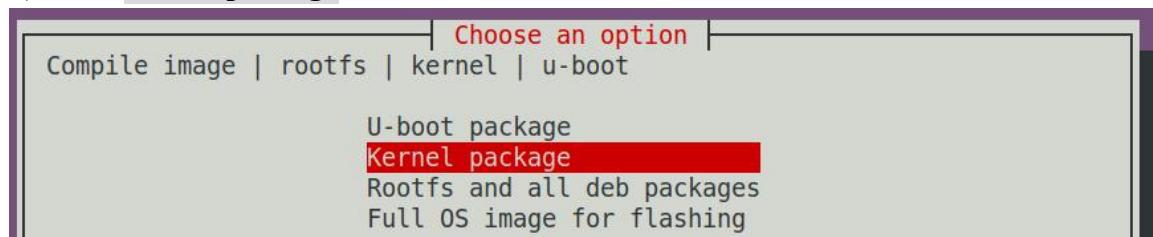
g. 然后就可以重启开发板来测试 u-boot 的修改是否生效了

4. 4. 编译 linux 内核

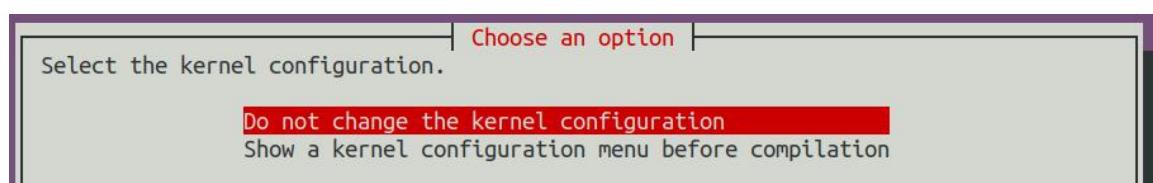
1) 运行 **build.sh** 脚本，记得加 sudo 权限

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

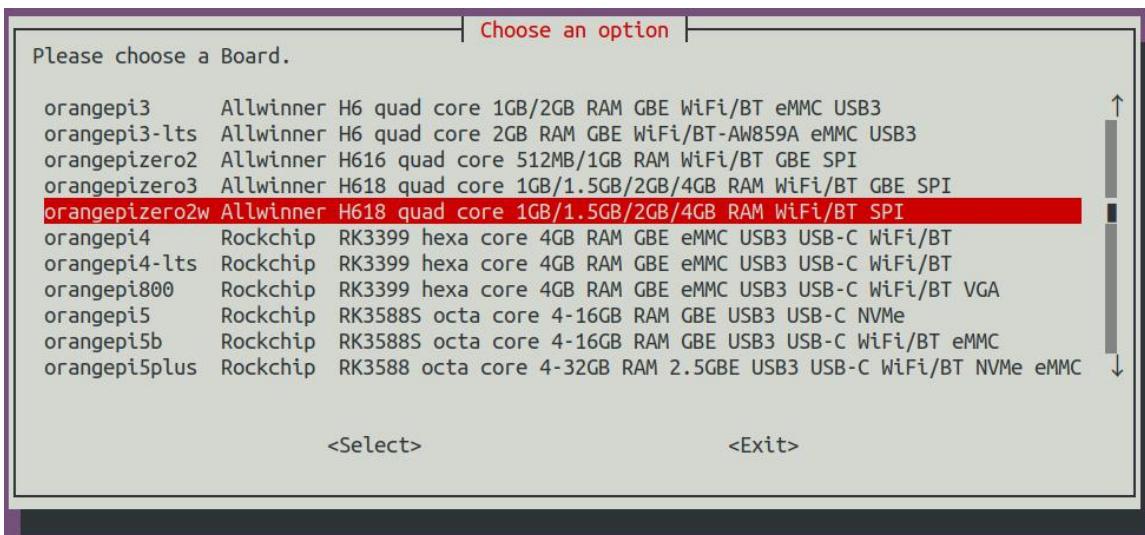
2) 选择 **Kernel package**，然后回车



3) 然后会提示是否需要显示内核配置界面，如果不需要修改内核配置，则选择第一个即可，如果需要修改内核配置，则选择第二个

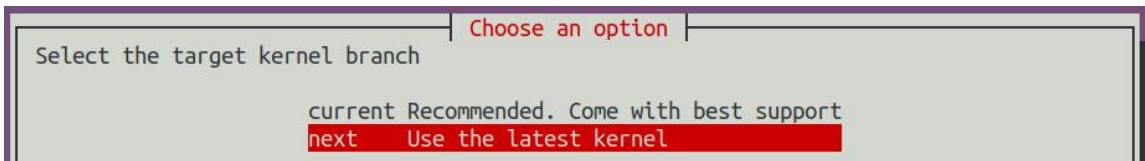


4) 接着选择开发板的型号

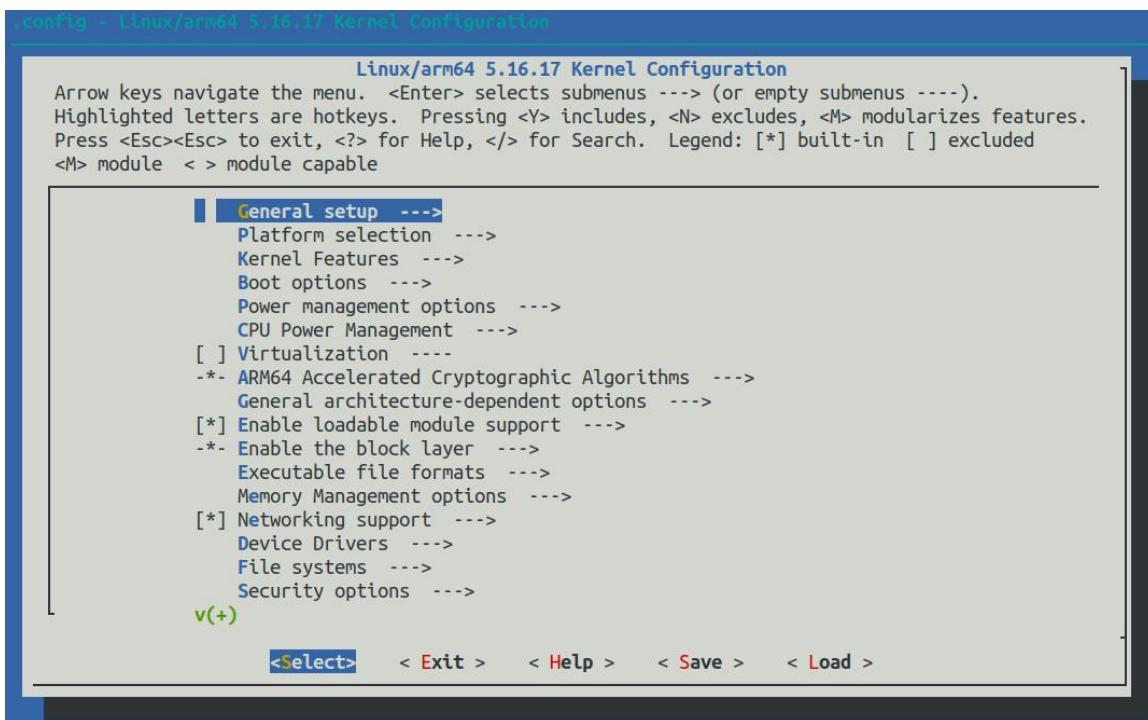


5) 然后选择内核源码的分支类型

- current 分支会编译 linux5.4 内核源码
- next 分支会编译 linux6.1 内核源码



6) 如果第 3)步选择了需要显示内核配置菜单（第二个选项），则会弹出通过 **make menuconfig** 打开的内核配置的界面，此时可以直接修改内核的配置，修改完后再保存退出即可，退出后会开始编译内核源码。



- a. 如果不需要修改内核的配置选项，在运行 build.sh 脚本时，传入 **KERNEL_CONFIGURE=no** 就可临时屏蔽弹出内核的配置界面了

```
test@test:~/orangepi-build$ sudo ./build.sh KERNEL_CONFIGURE=no
```

- b. 也可以设置 orangepi-build/userpatches/config-default.conf 配置文件中的 **KERNEL_CONFIGURE=no**，这样可以永久禁用这个功能
c. 编译内核的时候如果提示下面的错误，这是由于 Ubuntu PC 的终端界面太小，导致 **make menuconfig** 的界面无法显示，请把 Ubuntu PC 的终端调到最大，然后重新运行 build.sh 脚本

```
HOSTCC scripts/kconfig/mconf.o
HOSTCC scripts/kconfig/lxdialog/checklist.o
HOSTCC scripts/kconfig/lxdialog/util.o
HOSTCC scripts/kconfig/lxdialog/inputbox.o
HOSTCC scripts/kconfig/lxdialog/textbox.o
HOSTCC scripts/kconfig/lxdialog/yesno.o
HOSTCC scripts/kconfig/lxdialog/menubox.o
HOSTLD scripts/kconfig/mconf
scripts/kconfig/mconf Kconfig
Your display is too small to run Menuconfig!
It must be at least 19 lines by 80 columns.
scripts/kconfig/Makefile:28: recipe for target 'menuconfig' failed
make[1]: *** [menuconfig] Error 1
Makefile:560: recipe for target 'menuconfig' failed
make: *** [menuconfig] Error 2
[ error ] ERROR in function compile_kernel [ compilation.sh:376 ]
[ error ] Error kernel menuconfig failed
[ o.k. ] Process terminated
```

7) 编译 next 分支内核源码时提示的部分信息说明如下：

- a. linux 内核源码的版本



[o.k.] Compiling current kernel [**6.1.31**]

b. 使用的交叉编译工具链的版本

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

c. 内核默认使用的配置文件以及它存放的路径如下所示

[o.k.] Using kernel config file

[**orangepi-build/external/config/kernel/linux-6.1-sun50iw9-next.config**]

d. 编译生成的内核相关的 deb 包的路径

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

e. 编译生成的内核镜像 deb 包的包名

[o.k.] File name [**linux-image-next-sun50iw9_x.x.x_arm64.deb**]

f. 编译使用的时间

[o.k.] Runtime [**10 min**]

g. 最后会显示重复编译上一次选择的内核的编译命令, 使用下面的命令无需通过图形界面选择, 可以直接开始编译内核源码

[o.k.] Repeat Build Options [**sudo ./build.sh BOARD=orangepizero2w**]

BRANCH=next BUILD_OPT=kernel KERNEL_CONFIGURE=no]

8) 查看编译生成的内核相关的 deb 包

a. **linux-dtb-next-sun50iw9_x.x.x_arm64.deb** 包含有内核使用的 dtb 文件

b. **linux-headers-next-sun50iw9_x.x.x_arm64.deb** 包含内核头文件

c. **linux-image-next-sun50iw9_x.x.x_arm64.deb** 包含内核镜像和内核模块

```
test@test:~/orangepi-build$ ls output/debs/linux-*
output/debs/linux-dtb-next-sun50iw9_x.x.x_arm64.deb
output/debs/linux-headers-next-sun50iw9_x.x.x_arm64.deb
output/debs/linux-image-next-sun50iw9_x.x.x_arm64.deb
```

9) orangepi-build 编译系统编译 linux 内核源码时首先会将 linux 内核源码和 github 服务器的 linux 内核源码进行同步, 所以如果想修改 linux 内核的源码, 首先需要关闭源码的更新功能 (需要完整编译过一次 linux 内核源码后才能关闭这个功能, 否则会提示找不到 linux 内核的源码), 否则所作的修改都会被还原, 方法如下:

设置 **userpatches/config-default.conf** 中的 IGNORE_UPDATES 变量为 "yes"

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

10) 如果对内核做了修改, 可以使用下面的方法来更新开发板 linux 系统的内核和内



核模块

- 将编译好的 linux 内核的 deb 包上传到开发板的 linux 系统中

```
test@test:~/orangepi-build$ cd output/debs  
test@test:~/orangepi-build/output/debs$ scp \  
linux-image-next-sun50iw9_x.x.x_arm64.deb root@192.168.1.xxx:/root
```

- 再安装刚才上传的新的 linux 内核的 deb 包

```
orangepi@orangepi:~$ sudo dpkg -i linux-image-next-sun50iw9_x.x.x_arm64.deb
```

- 然后重启开发板，再查看内核相关的修改是否已生效

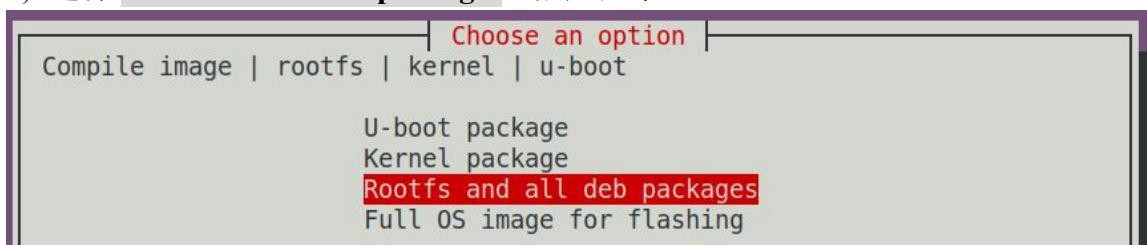
```
orangepi@orangepi:~$ sudo reboot
```

4. 5. 编译 rootfs

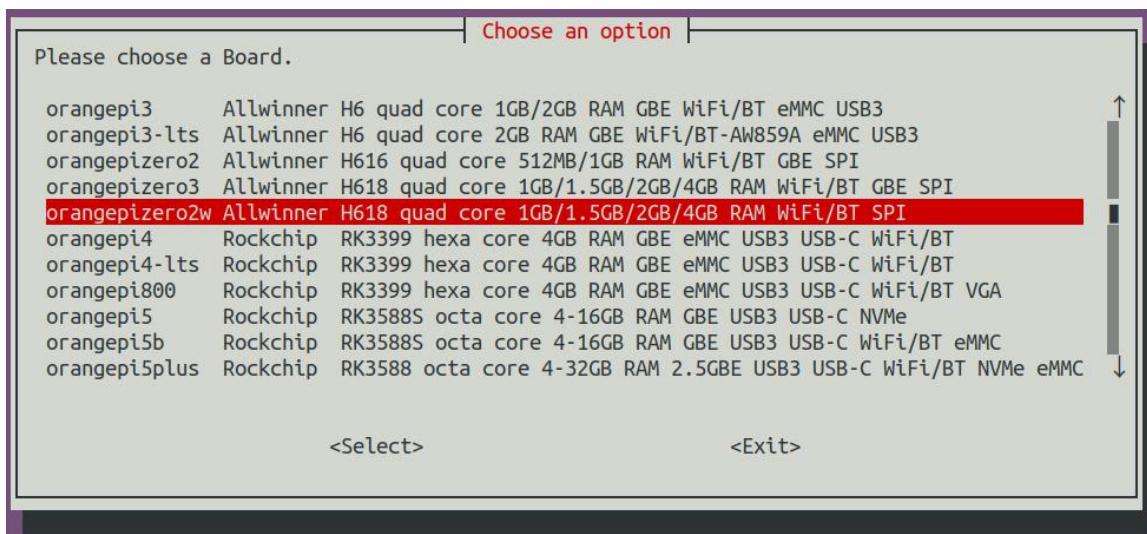
- 运行 build.sh 脚本，记得加 sudo 权限

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

- 选择 **Rootfs and all deb packages**，然后回车

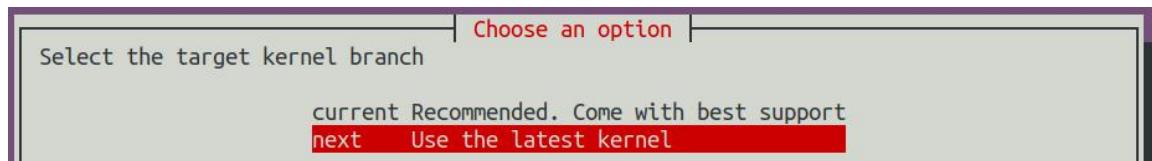


- 接着选择开发板的型号

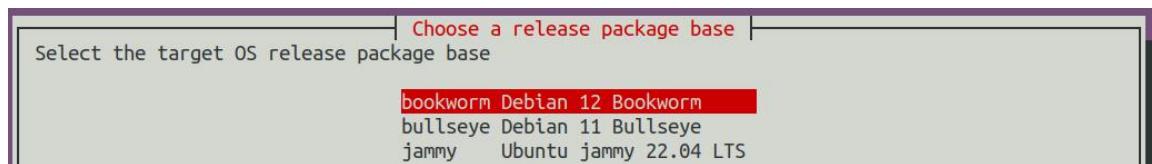




- 4) 然后选择内核源码的分支类型，不同版本的内核源码维护的 rootfs 类型有区别
- current 分支可以看到 debian11、ubuntu20.04、ubuntu22.04 三个选项
 - next 分支可以看到 debian11、debian12、ubuntu22.04 三个选项

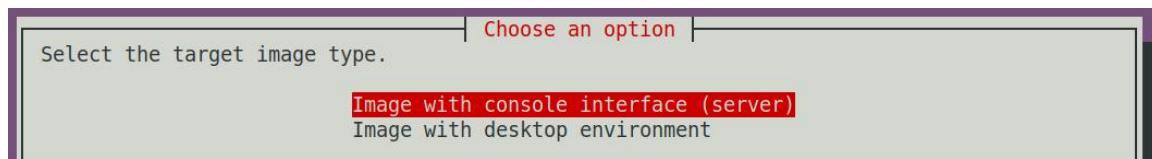


- 5) 然后选择 rootfs 的类型

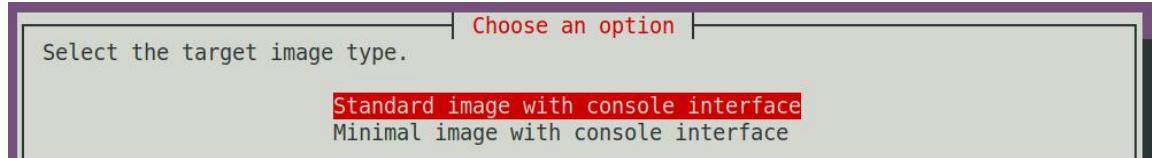


- 6) 然后选择镜像的类型

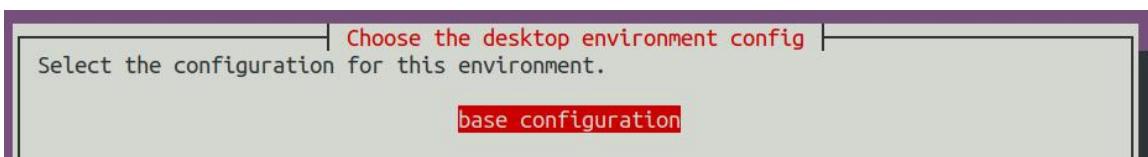
- Image with console interface (server)** 表示服务器版的镜像，体积比较小
- Image with desktop environment** 表示带桌面的镜像，体积比较大



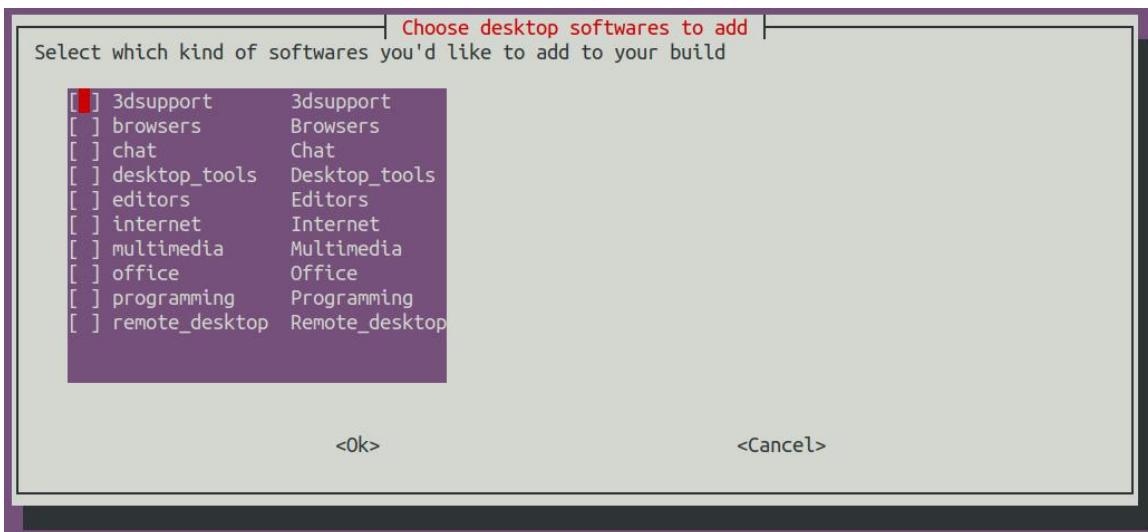
- 7) 如果是编译服务器版的镜像，还可以选择编译 Standard 版本或者 Minimal 版本，Minimal 版本预装的软件会比 Standard 版本少很多(没特殊需求请不要选择 Minimal 版本，因为很多东西默认没有预装，部分功能可能用不了)



- 8) 如果是编译桌面版本的镜像还需要选择桌面环境的类型，目前只维护 XFCE，所以请选择 XFCE 类型的桌面



然后可以选择需要安装的额外的软件包。这里请按下车键直接跳过。



9) 然后就会开始编译 rootfs，编译时提示的部分信息说明如下

a. rootfs 的类型

[o.k.] local not found [Creating new rootfs cache for **bullseye**]

b. 编译生成的 rootfs 压缩包的存放路径

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

c. 编译生成的 rootfs 压缩包的名字

[o.k.] File name [**bullseye-xfce-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4**]

10) 查看编译生成的 rootfs 压缩包

a. **bullseye-xfce-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4** 是 rootfs 的压缩包，名字各字段的含义为

a) **bullseye** 表示 rootfs 的 linux 发行版的类型



- b) **xfce** 表示 rootfs 为桌面版的类型，如果为 **cli** 则表示服务器版类型
 - c) **arm64** 表示 rootfs 的架构类型
 - d) **25250ec7002de9e81a41de169f1f89721** 是由 rootfs 安装的所有软件包的包名生成的 MD5 哈希值，只要没有修改 rootfs 安装的软件包的列表，那么这个值就不会变，编译脚本会通过这个 MD5 哈希值来判断是否需要重新编译 rootfs
- b. **bullseye-xfce-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4.list** 列出了 rootfs 安装的所有软件包的包名

```
test@test:~/orangeipi-build$ ls external/cache/rootfs/
bullseye-xfce-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4
bullseye-xfce-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4.current
bullseye-xfce-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4.list
```

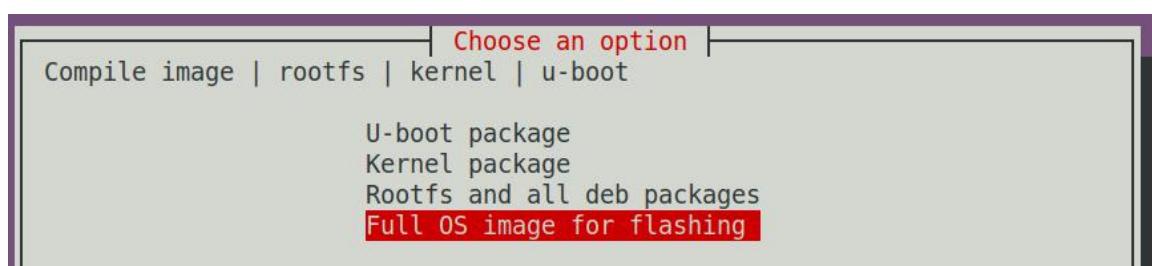
11) 如果需要的 rootfs 在 **external/cache/rootfs** 下已经存在，那么再次编译 rootfs 就会直接跳过编译过程，不会重新开始编译，编译镜像的时候也会去 **external/cache/rootfs** 下查找是否已经有缓存可用的 rootfs，如果有就直接使用，这样可以节省大量的下载编译时间

4. 6. 编译 linux 镜像

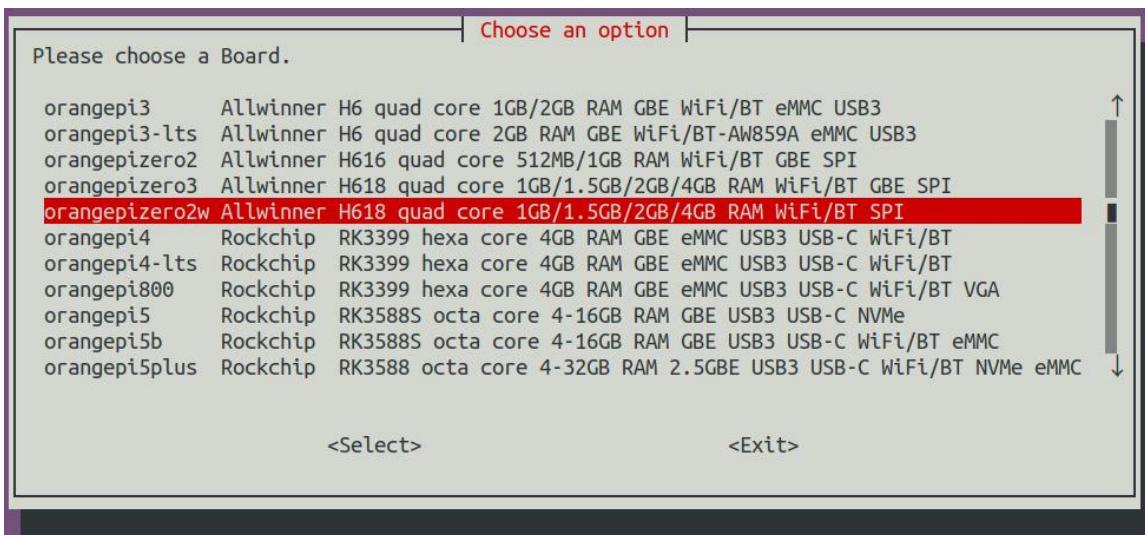
1) 运行 **build.sh** 脚本，记得加 sudo 权限

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

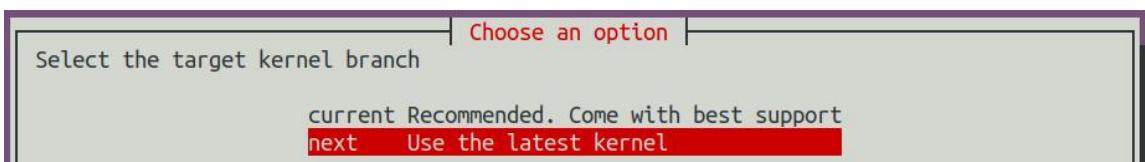
2) 选择 **Full OS image for flashing**，然后回车



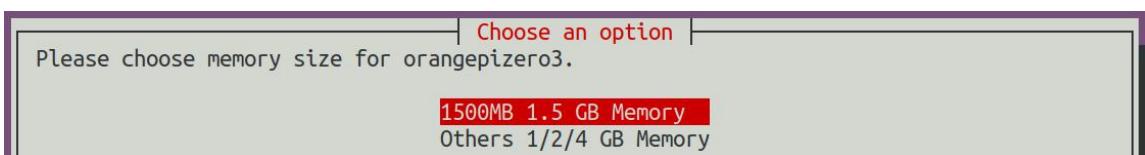
3) 然后选择开发板的型号



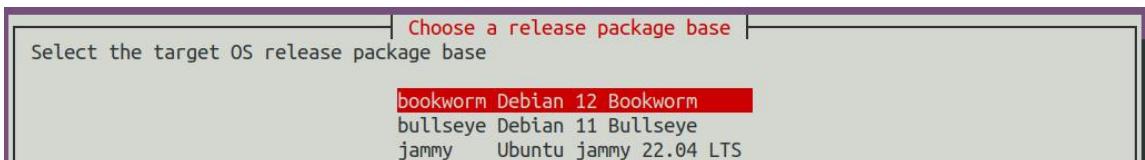
- 4) 然后选择内核源码的分支类型，不同版本的内核源码维护的 rootfs 类型有区别
- current 分支可以看到 debian11、ubuntu20.04、ubuntu22.04 三个选项
 - next 分支可以看到 debian11、debian12、ubuntu22.04 三个选项



- 5) 如果选择的 next 分支还会提示需要选择内存的大小，current 分支不需要选择
- 如果购买的开发板为 1.5GB 内存大小的，请选择第一项
 - 如果购买的开发板为 1GB 或 2GB 或 4GB 内存大小的，请选择第二项



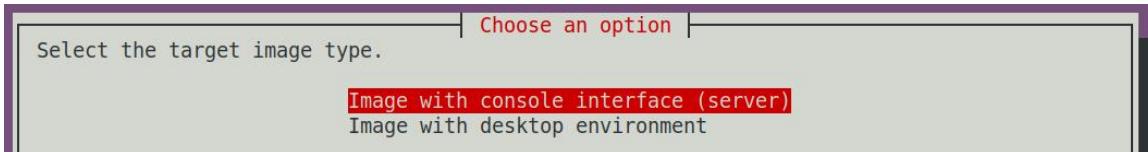
- 6) 然后选择 rootfs 的类型



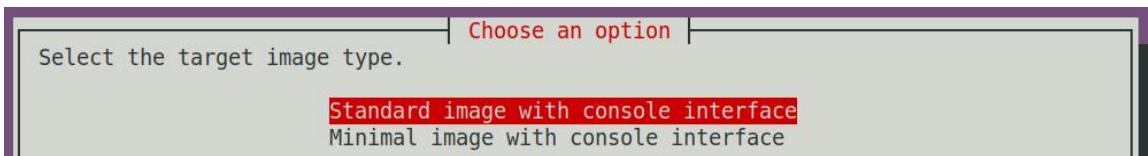
- 7) 然后选择镜像的类型



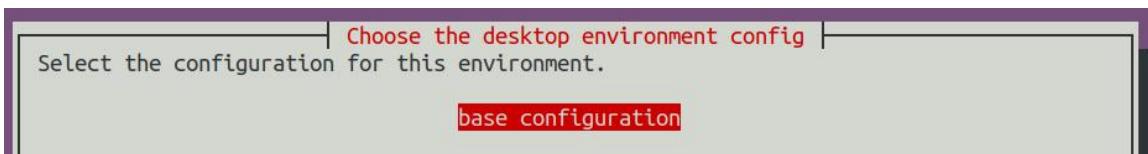
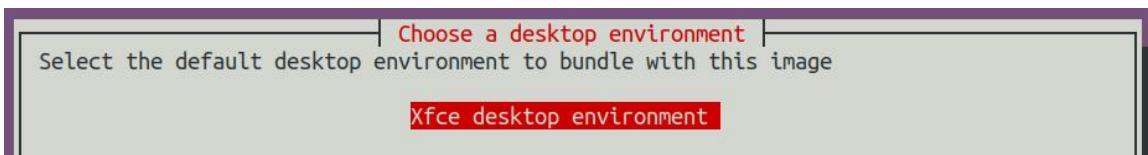
- c. **Image with console interface (server)** 表示服务器版的镜像，体积比较小
- d. **Image with desktop environment** 表示带桌面的镜像，体积比较大



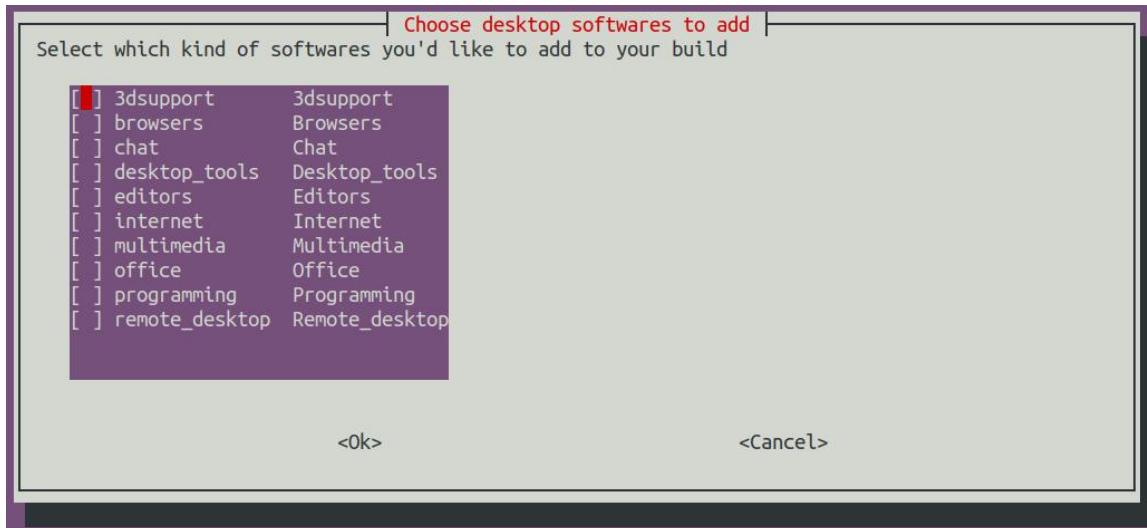
8) 如果是编译服务器版的镜像，还可以选择编译 Standard 版本或者 Minimal 版本，Minimal 版本预装的软件会比 Standard 版本少很多(没特殊需求请不要选择 Minimal 版本，因为很多东西默认没有预装，部分功能可能用不了)



9) 如果是编译桌面版本的镜像还需要选择桌面环境的类型，目前只维护 XFCE，所以请选择 XFCE 类型的桌面



然后可以选择需要安装的额外的软件包。这里请按下回车键直接跳过。



10) 然后就会开始编译 linux 镜像，编译的大致流程如下

- a. 初始化 Ubuntu PC 的编译环境，安装编译过程需要的软件包
- b. 下载 u-boot 和 linux 内核的源码（如果已经缓存，则只更新代码）
- c. 编译 u-boot 源码，生成 u-boot 的 deb 包
- d. 编译 linux 源码，生成 linux 相关的 deb 包
- e. 制作 linux firmware 的 deb 包
- f. 制作 orangepi-config 工具的 deb 包
- g. 制作板级支持的 deb 包
- h. 如果是编译 desktop 版镜像，还会制作 desktop 相关的 deb 包
- i. 检查 rootfs 是否已经缓存，如果没有缓存，则重新制作 rootfs，如果已经缓存，则直接解压使用
- j. 安装前面生成的 deb 包到 rootfs 中
- k. 对不同的开发板和不同类型镜像做一些特定的设置，如预装额外的软件包，修改系统配置等
- l. 然后制作镜像文件，并格式化分区，默认类型为 ext4
- m. 再将配置好的 rootfs 拷贝到镜像的分区中
- n. 然后更新 initramfs
- o. 最后将 u-boot 的 bin 文件通过 dd 命令写入到镜像中

11) 编译完镜像后会提示下面的信息

- a. 编译生成的镜像的存放路径

```
[ o.k. ] Done building  
[ output/images/orangepizero2w_x.x.x_debian_bullseye_linux6.1.xx_xfce_desktop/or
```



angeepizero2w_x.x.x_debian_bullseye_linux6.1.xx_xfce_desktop.img]

b. 编译使用的时间

[o.k.] Runtime [19 min]

c. 重复编译镜像的命令，使用下面的命令无需通过图形界面选择，可以直接开始编译镜像

```
[ o.k. ] Repeat Build Options [ sudo ./build.sh BOARD=orangeepizero2w  
BRANCH=next BUILD_OPT=image RELEASE=bullseye BUILD_MINIMAL=no  
BUILD_DESKTOP=no KERNEL_CONFIGURE=yes ]
```



5. Orange Pi OS Arch 系统的使用说明

5. 1. Orange Pi OS Arch 系统功能适配情况

主板功能	OPi OS Arch
HDMI 视频	OK
HDMI 音频	OK
Type-C USB2.0 x 2	OK
TF 卡启动	OK
WIFI	OK
蓝牙	OK
LED 灯	OK
40pin GPIO	OK
40pin I2C	OK
40pin SPI	OK
40pin UART	OK
40pin PWM	OK
温度传感器	OK
硬件看门狗	OK
Mali GPU	NO
视频编解码	NO

24pin 扩展板功能	OPi OS Arch
百兆网口	OK
百兆网口灯	OK
USB2.0 HOST x 2	OK
红外接收	OK
耳机音频播放	OK
开关机按键	OK
LRADC 自定义按键 x 2	OK
TV-OUT	NO

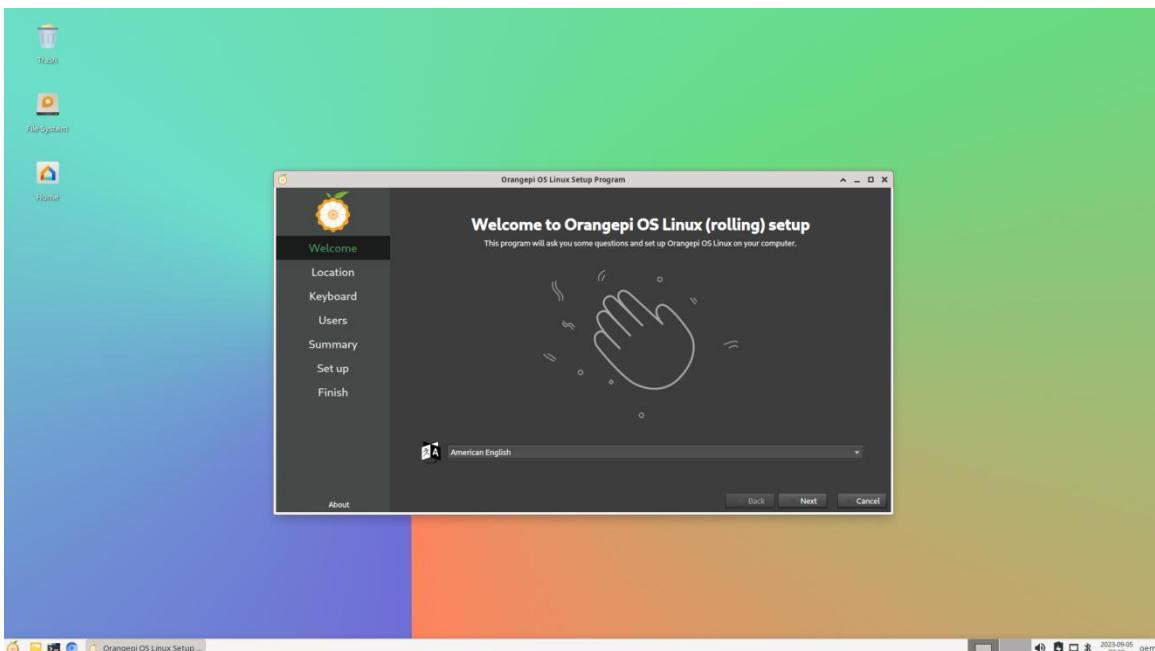


5.2. Orange Pi OS Arch 系统用户向导使用说明

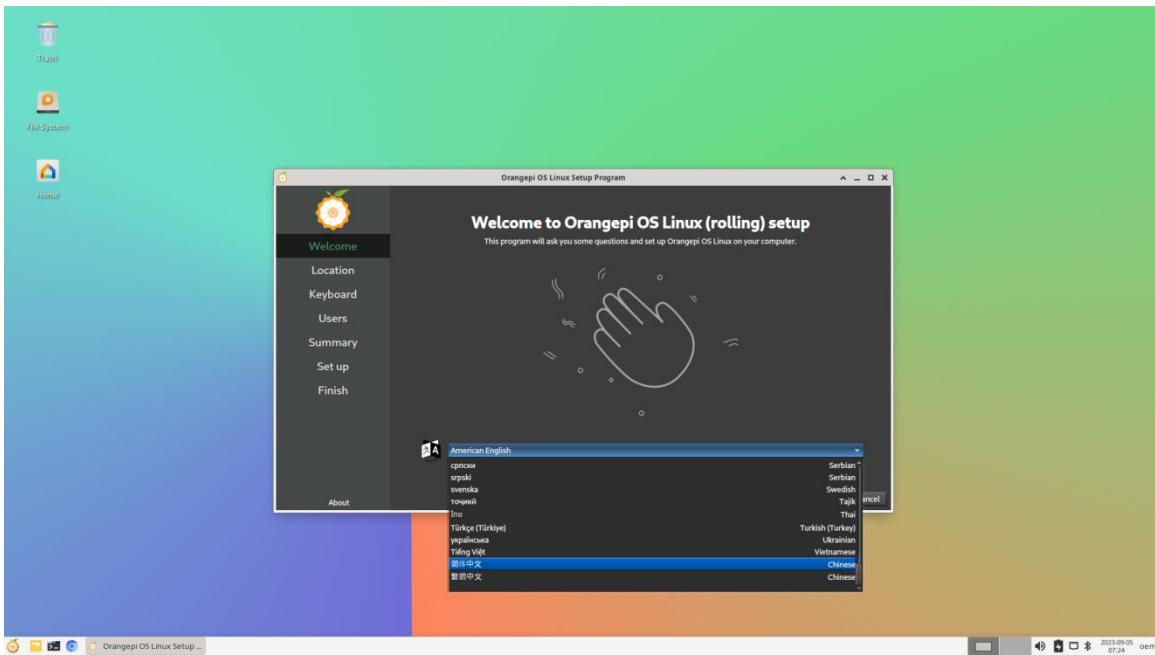
首先请注意，OPi OS Arch 系统是没有设置默认的 orangepi 用户和密码的，所以烧录完系统启动后是无法直接通过串口和 ssh 远程登录的（root 用户也不行）。这一点和 Ubuntu、Debian 系统是有区别的。

OPi OS Arch 系统第一次启动时需要接上 HDMI 显示器，然后通过用户向导来初始化系统设置（其中包括新建用户名和设置密码）。用户向导的设置步骤如下所示：

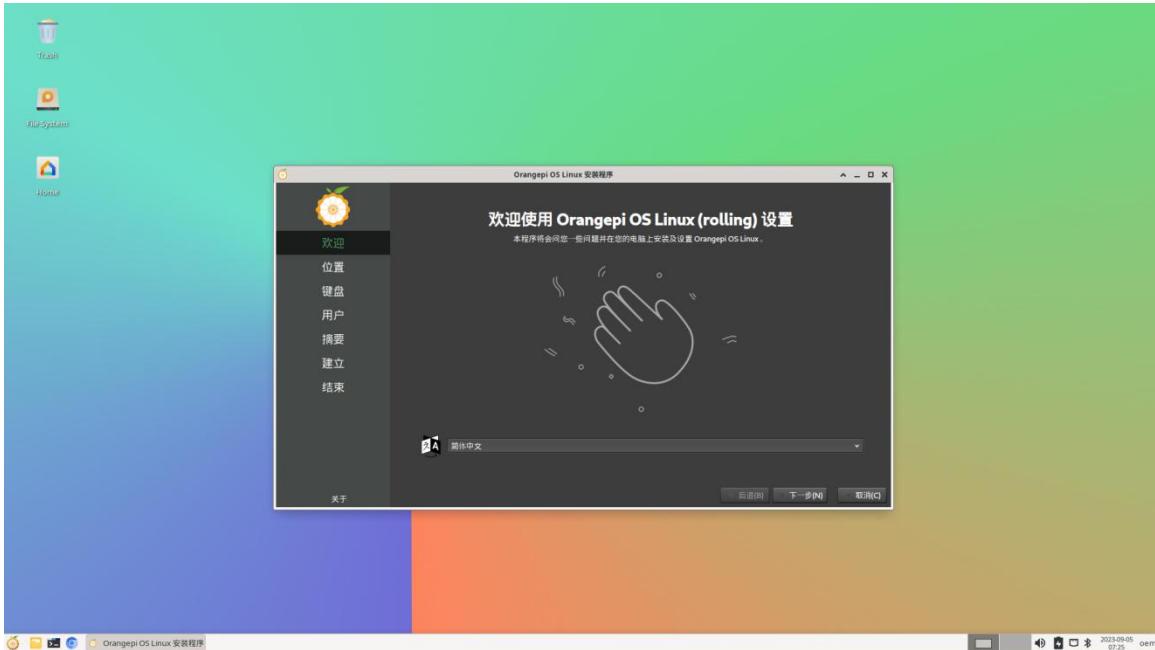
- 烧录完系统第一次启动进入桌面后会看到下图所示的用户向导程序



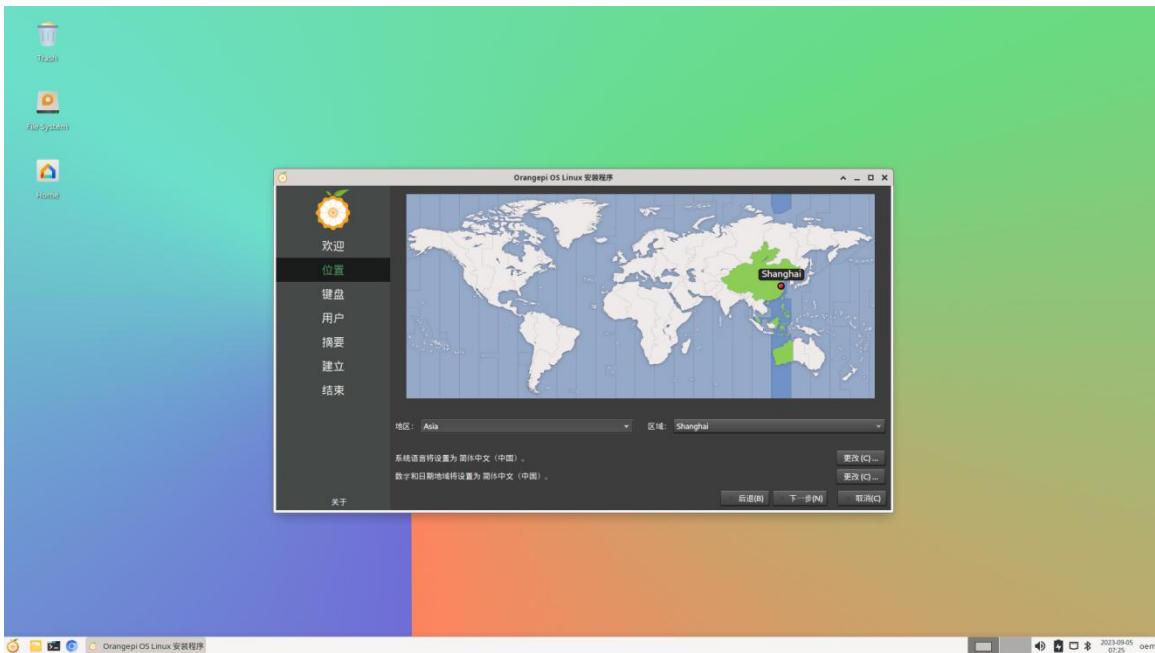
- 首先需要选择想要语言



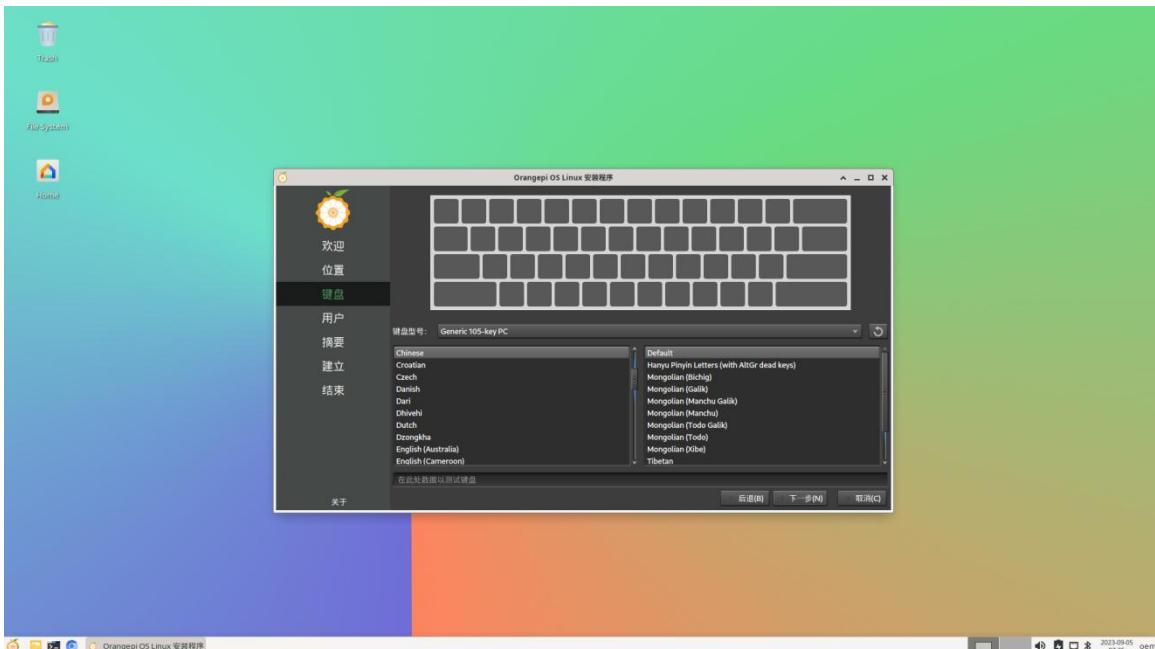
c) 在选择完语言后，用户向导会立即切换为对应的语言界面，如中文显示如下所示



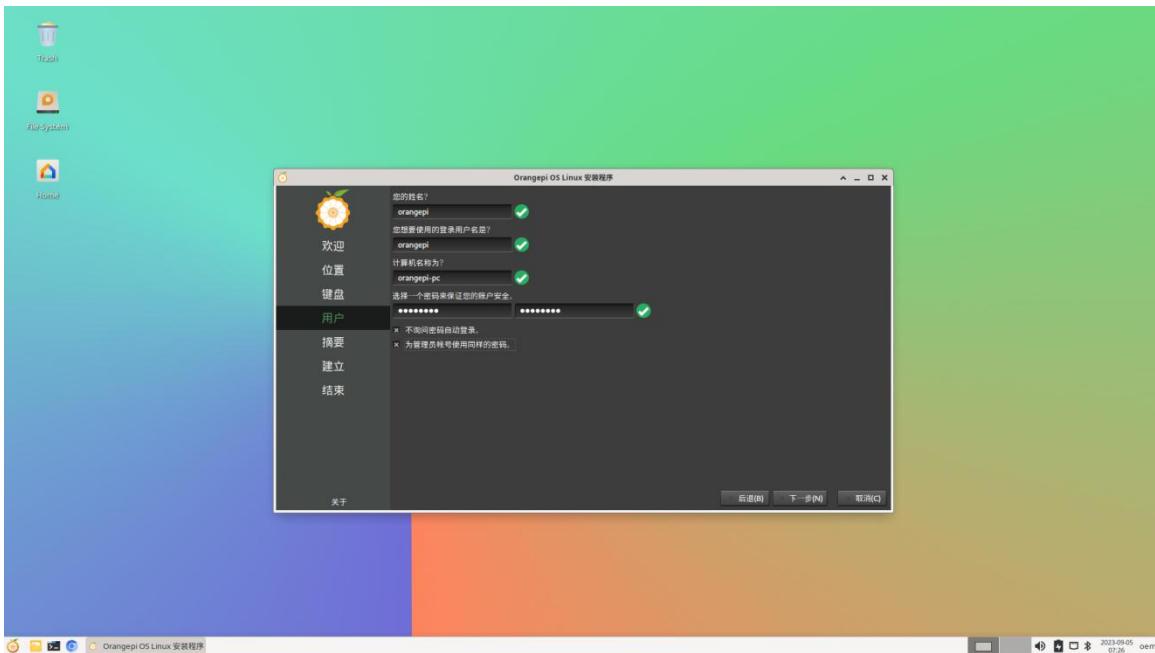
d) 然后选择区域



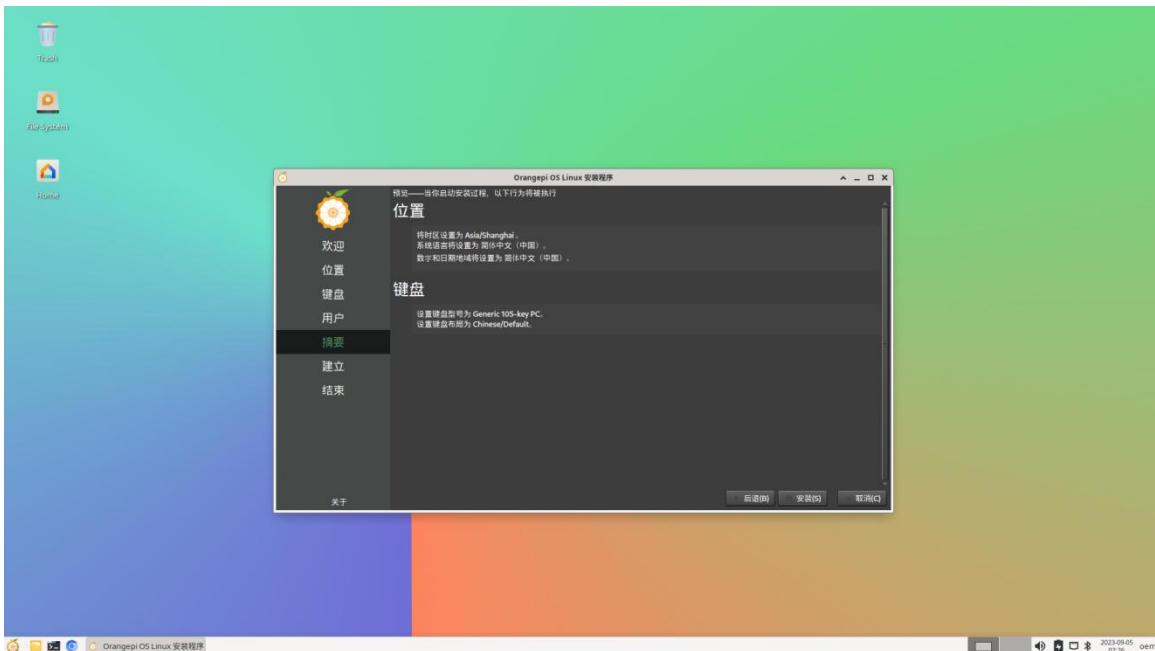
e) 然后选择键盘型号



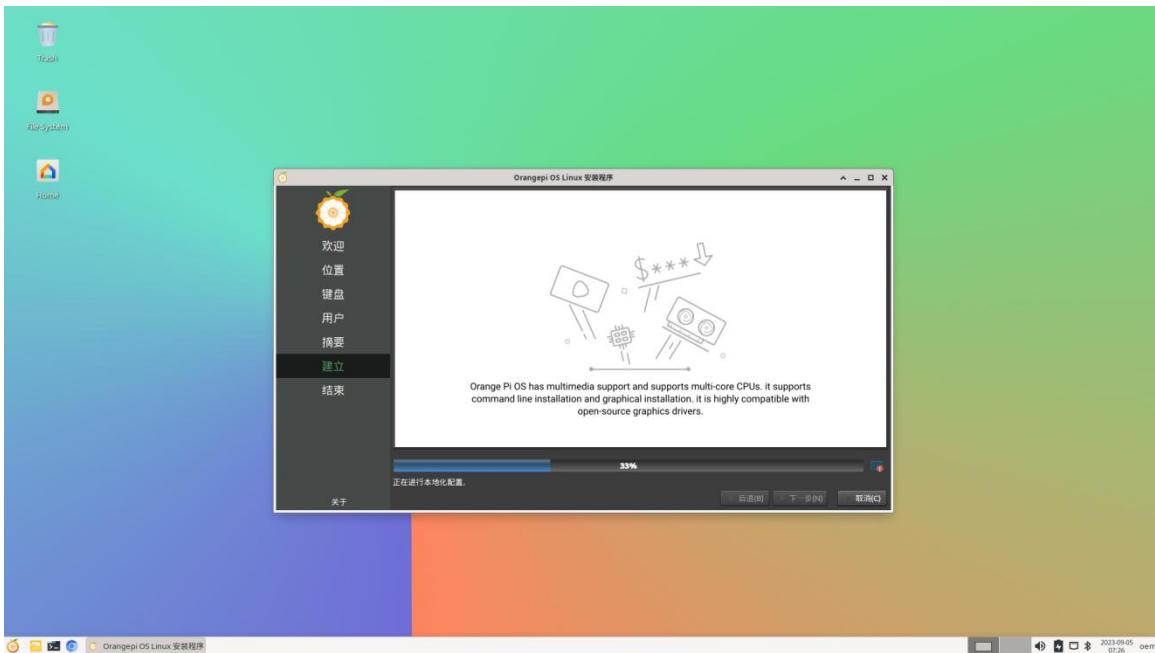
f) 然后新建用户名和设置密码



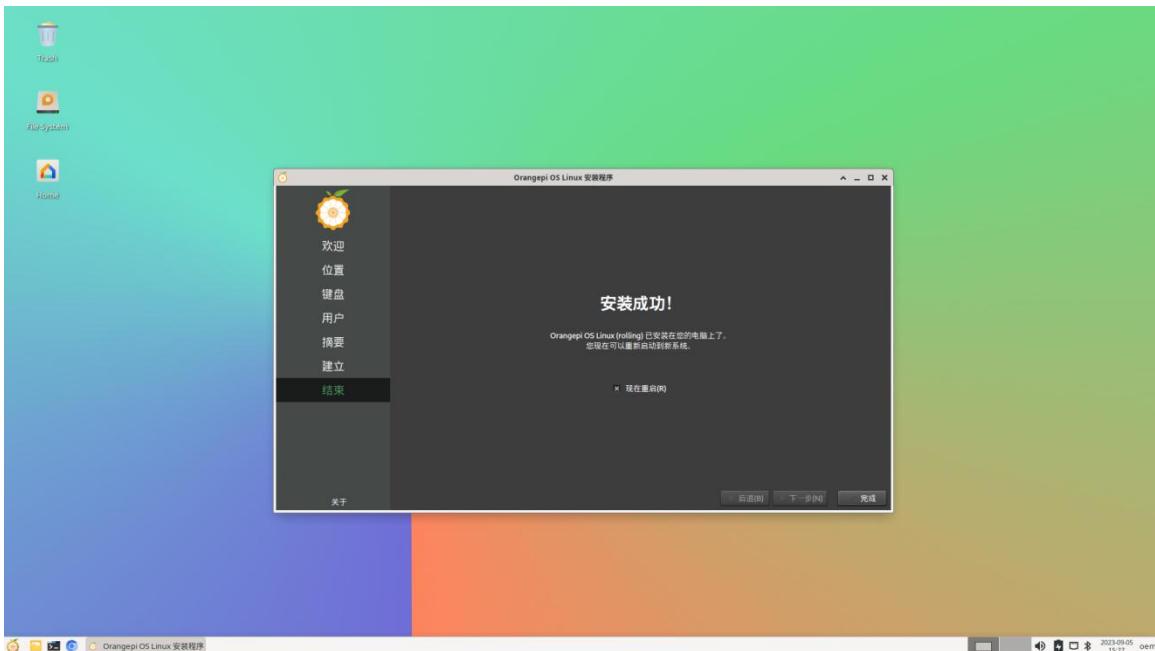
g) 然后确保选择没问题后，再点击**安装**按钮



h) 然后等待安装完成



i) 安装完成需要点击**完成**按钮重启系统



j) 重启后会自动启动 Orange Pi Hello 程序，此时需要去掉右下角**开机时启动**的勾选状态，不然每次启动都需要手动关闭 Orange Pi Hello 程序



此时就可以使用刚才新建的用户名和密码通过串口或者 ssh 登录 OPi OS 系统了。

5.3. 设置 DT overlays 的方法

开发板 40pin 中的 I2C/SPI/UART/PWM 等复用功能默认在内核的 dts 中都是关闭的，需要手动打开对应的 DT overlays 才能使用。

在 OPi OS Arch 系统中打开 DT overlays 的方法如下所示：

1) 首先打开 /boot/extlinux/extlinux.conf 配置文件

```
[orangepi@orangepi-pc ~]$ sudo vim /boot/extlinux/extlinux.conf
```

2) 然后在 /boot/extlinux/extlinux.conf 中通过添加 **FDTOVERLAYS**
/dtbs/allwinner/overlay/xxx.dtbo 来打开对应的配置

注意 **FDTOVERLAYS** **/dtbs/allwinner/overlay/xxx.dtbo** 中的 **xxx.dtbo** 需要替换为具体的 dtbo 配置，请不要照抄。

```
[orangepi@orangepi-pc ~]$ sudo vim /boot/extlinux/extlinux.conf
LABEL Orange Pi
KERNEL /Image
```



FDT /dtbs/allwinner/sun50i-h616-orangepi-zero2w.dtb

FDTOVERLAYS /dtbs/allwinner/overlay/xxx.dtbo #需要添加的配置

3) xxx.dtbo 在 OPi OS Arch 镜像中的存放路径如下所示, 请注意, 此路径下面不是所有的 dtbo 都可以使用的。

/boot/dtbs/allwinner/overlay/

4) 开发板可以使用的 DT overlays 配置如下所示

开发板上的功能	对应的 DT overlays 配置
40pin - i2c0	sun50i-h616-pi-i2c0.dtbo
40pin - i2c1	sun50i-h616-pi-i2c1.dtbo
40pin - i2c2	sun50i-h616-pi-i2c2.dtbo
40pin - uart2	sun50i-h616-pi-uart2.dtbo
40pin - uart3	sun50i-h616-pi-uart3.dtbo
40pin - uart4	sun50i-h616-pi-uart4.dtbo
40pin - uart5	sun50i-h616-ph-uart5.dtbo
40pin - pwm1	sun50i-h616-pi-pwm1.dtbo
40pin - pwm2	sun50i-h616-pi-pwm2.dtbo
40pin - pwm3	sun50i-h616-pi-pwm3.dtbo
40pin - pwm4	sun50i-h616-pi-pwm4.dtbo
40pin - spi1 cs0	sun50i-h616-spi1-cs0-spidev.dtbo
40pin - spi1 cs1	sun50i-h616-spi1-cs1-spidev.dtbo
40pin - spi1 cs0 cs1	sun50i-h616-spi1-cs0-cs1-spidev.dtbo
设置 USB0 为 Host 模式	sun50i-h616-usb0-host.dtbo
关闭绿色的 LED 灯	sun50i-h616-zero2w-disable-led.dtbo
关闭 UART0 调试串口的方法	sun50i-h616-disable-uart0.dtbo

5) 如果需要同时打开多个配置, 直接在 **FDTOVERLAYS** 后面将多个配置的路径加上即可, 比如同时打开 i2c1 和 uart5 的配置如下所示

```
[orangepi@orangepi-pc ~]$ sudo vim /boot/extlinux/extlinux.conf
```

```
LABEL Orange Pi
```

```
KERNEL /Image
```

```
FDT /dtbs/allwinner/sun50i-h616-orangepi-zero2w.dtb
```

```
FDTOVERLAYS /dtbs/allwinner/overlay/sun50i-h616-pi-i2c1.dtbo /dtbs/allwinner/overlay/sun50i-h616-ph-uart5.dtbo
```



6) 设置好后需要重启系统才能让配置生效

```
[orangeipi@orangeipi-pc ~]$ sudo reboot
```

5. 4. 安装软件的方法

使用 pacman 包管理工具可以安装 OPi OS 中没有的软件，比如安装 vim 编辑器的命令如下所示，如果想安装其他软件，只需要把 vim 替换想要安装的软件的包名即可。

```
[orangeipi@orangeipi-pc ~]$ sudo pacman -Syy vim
```



6. Android 12 TV 系统使用说明

6. 1. 已支持的 Android 版本

Android 版本	内核版本
Android 12 TV 版	linux5.4

6. 2. Android 12 TV 功能适配情况

主板功能	Android12 TV
HDMI 视频	OK
HDMI 音频	OK
Type-C USB2.0 x 2	OK
TF 卡启动	OK
WIFI	OK
蓝牙	OK
USB 摄像头	OK
LED 灯	OK
40pin GPIO	OK
40pin I2C	OK
40pin SPI1	OK
40pin UART	OK
40pin PWM	OK
温度传感器	OK
硬件看门狗	OK
Mali GPU	OK
视频编解码	OK

24pin 扩展板功能	Android12 TV
百兆网口	OK
百兆网口灯	OK
USB2.0 HOST x 2	OK
红外接收	OK



耳机音频播放	OK
开关机按键	OK
LRADC 自定义按键 x 2	OK, 默认设置为音量加减按键
TV-OUT	OK

6.3. 板载 LED 灯显示说明

	绿灯	红灯
u-boot 启动阶段	灭	亮
内核启动到进入系统	亮	亮

6.4. Android 返回上一级界面的方法

我们一般都是使用鼠标和键盘来控制开发板的安卓系统，当进入某些界面，需要返回上一级界面或者桌面时，只能通过**鼠标右键**来返回，键盘是无法返回的。

如果有购买开发板配套的红外遥控（其他遥控不行）和 24pin 扩展板，将 24pin 扩展板接入开发板后，还可以通过遥控中的返回键来返回上一级菜单，返回键的位置如下图所示：



6.5. ADB 的使用方法

6.5.1. 使用网络连接 adb 调试

使用网络 adb 无需 USB Type C 接口的数据线来连接电脑和开发板，而是通过网络来通信，所以首先请确保开发板的有线或者无线网络已经连接好了，然后获取开发板的 IP 地址，后面要用到。



1) 确保 Android 系统的 **service.adb.tcp.port** 设置为 5555 端口号

```
apollo-p2:/ # getprop | grep "adb.tcp"  
[service.adb.tcp.port]: [5555]
```

2) 如果 **service.adb.tcp.port** 没有设置，可以在串口中使用下面的命令设置网络 adb 的端口号

```
apollo-p2:/ # setprop service.adb.tcp.port 5555  
apollo-p2:/ # stop adbd  
apollo-p2:/ # start adbd
```

3) 在 Ubuntu PC 上安装 adb 工具

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

4) 然后在 Ubuntu PC 上连接网络 adb

```
test@test:~$ adb connect 192.168.1.xxx:5555 (需要修改为开发板的 IP 地址)  
* daemon not running; starting now at tcp:5037  
* daemon started successfully  
connected to 192.168.1.xxx:5555  
  
test@test:~$ adb devices  
List of devices attached  
192.168.1.xxx:5555 device
```

5) 然后在 Ubuntu PC 上通过 adb shell 就可以登录 android 系统

```
test@test:~$ adb shell  
apollo-p2:/ #
```

6. 5. 2. 使用数据线连接 adb 调试

1) 准备一根 USB Type C 接口的数据线，USB 接口一端插入电脑的 USB 接口中，USB Type C 接口一端插入开发板的 USB0 接口中（USB0 的位置请见下面右边图片的说明）。在这种情况下是由电脑的 USB 接口给开发板供电，所以请确保电脑的 USB 接口能提供足够的功率驱动开发板。



只有此Type-C接口有device功能，可以使用adb



2) 在 Ubuntu PC 上安装 adb 工具

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

3) 查看识别到 ADB 设备

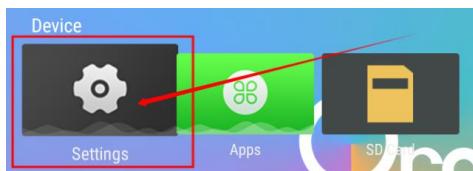
```
test@test:~$ adb devices  
List of devices attached  
4c00146473c28651dd0    device
```

4) 然后在 Ubuntu PC 上通过 adb shell 就可以登录 android 系统

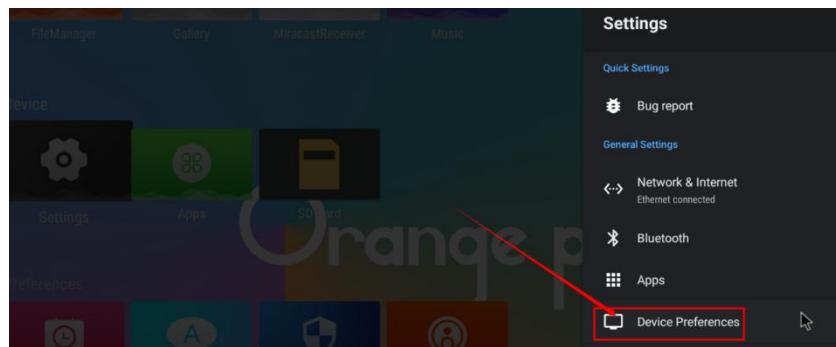
```
test@test:~$ adb shell  
apollo-p2:/ $
```

6.6. 查看设置 HDMI 显示分辨率的方法

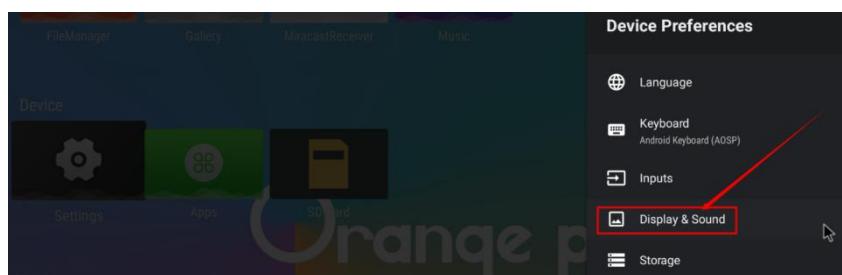
1) 首先进入 Settings



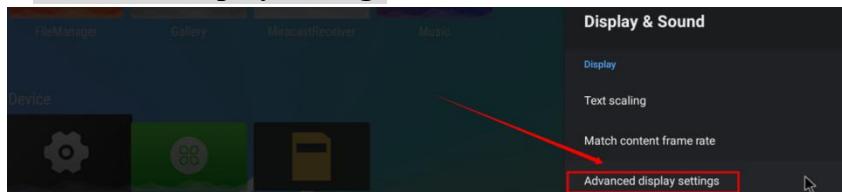
2) 然后选择 Device Preferences



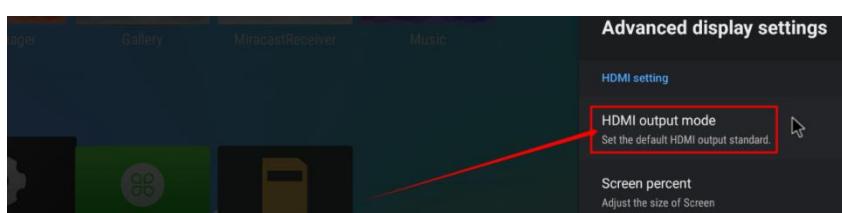
3) 然后选择 **Display & Sound**



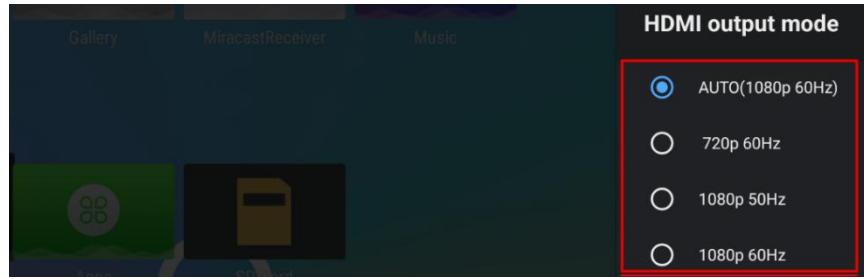
4) 然后选择 **Advanced display settings**



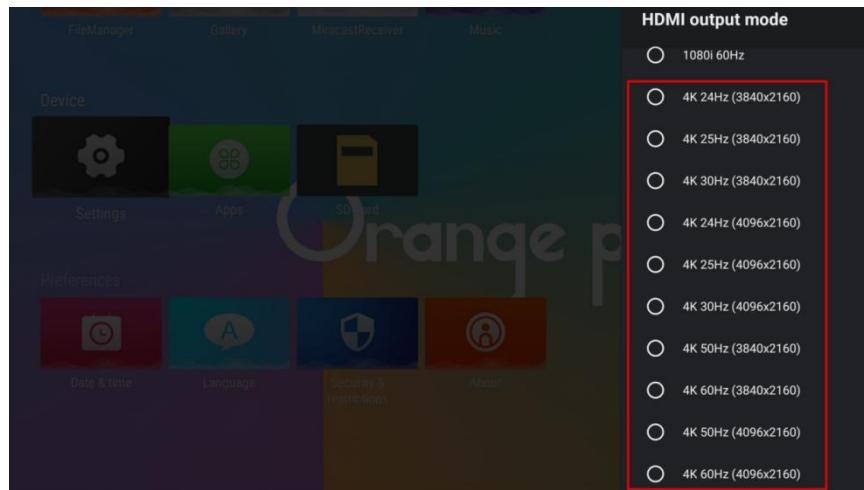
5) 然后选择 **HDMI output mode**



6) 然后就能看到显示器支持的分辨率列表了。此时点击对应的选项就会切换到对应的分辨率。请注意，不同显示器支持的分辨率可能是不同的，如果接到电视上，一般会看到比下图更多的分辨率选项。



7) 开发板的 HDMI 输出是支持 4K 显示的，当接到 4K 电视时就可以看到 4K 分辨率的选项



6. 6. 1. HDMI 转 VGA 显示测试

1) 首先需要准备下面的配件

a. HDMI 转 VGA 转换器



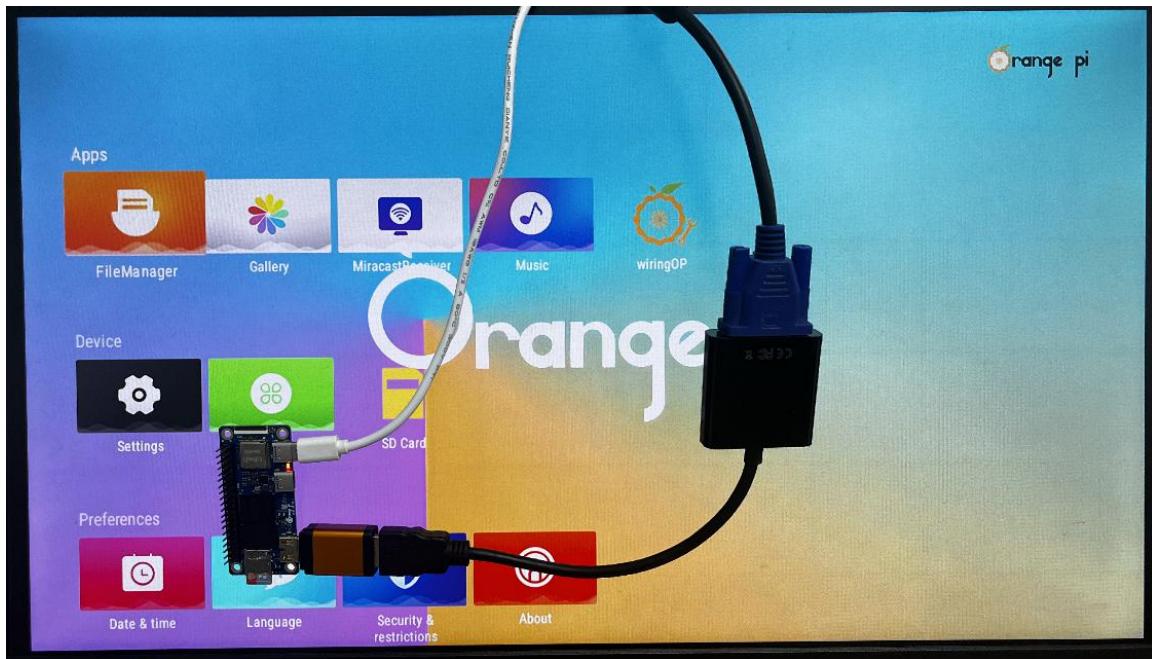
b. 一根 VGA 线和一个 Mini HDMI 公转 HDMI 母的转接头





c. 一个支持 VGA 接口的显示器或者电视

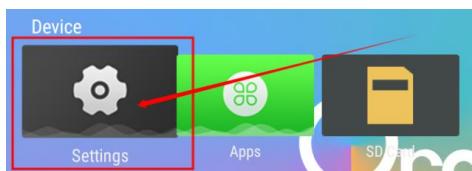
2) HDMI 转 VGA 显示测试如下所示



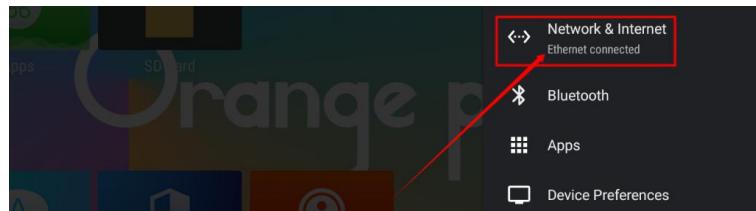
使用 HDMI 转 VGA 显示时，开发板以及开发板的 Android 系统是不需要做任何设置的，只需要开发板 Mini HDMI 接口能正常显示就可以了。所以如果测试有问题，请检查 HDMI 转 VGA 转换器、VGA 线以及显示器是否有问题。

6. 7. WI-FI 的连接方法

1) 首先选择 **Settings**



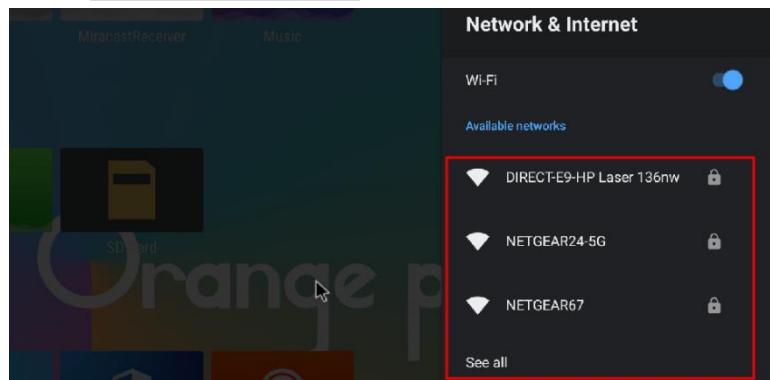
2) 然后选择 **Network & Internet**



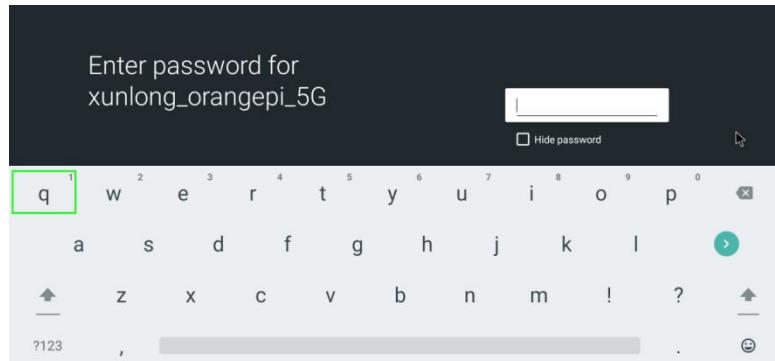
3) 然后打开 WI-FI



4) 打开 WI-FI 后在 Available networks 下面就可以看到搜索到的信号



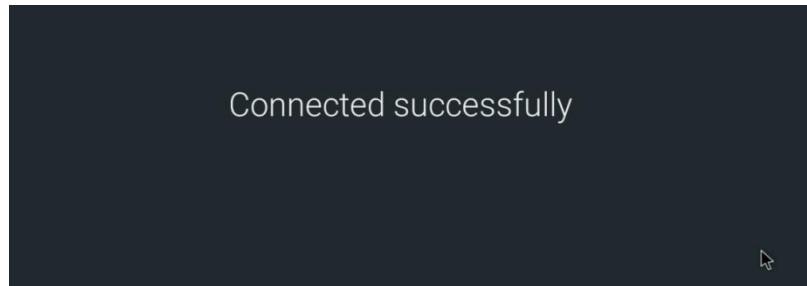
5) 选择想连接的 WI-FI 后会弹出下图所示的密码输入界面



6) 然后使用键盘输入 WI-FI 对应的密码，再使用鼠标点击虚拟键盘中的回车按钮就会开始连接 WI-FI 了



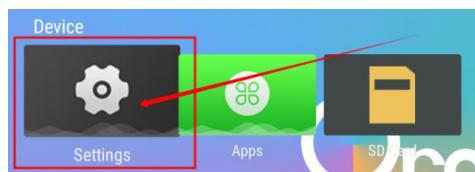
7) WI-FI 连接成功后的显示如下图所示



6.8. WI-FI hotspot 的使用方法

1) 首先请确保以太网口已连接网线，并且能正常上网

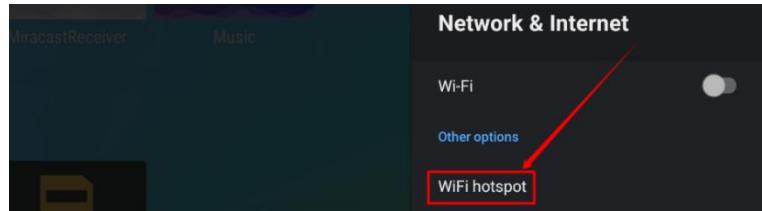
2) 然后选择 **Settings**



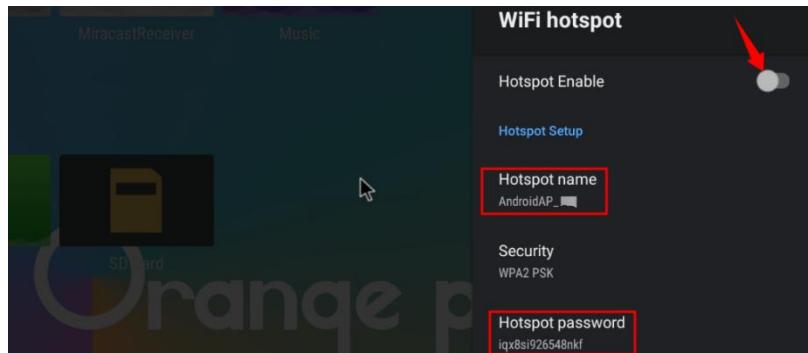
3) 然后选择 **Network & Internet**



4) 然后选择 **WIFI hotspot**



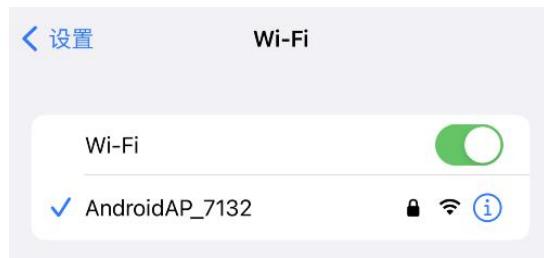
5) 然后打开 **Hotspot Enable**, 下图中还可以看到生成的热点的名字和密码, 记住它们, 在连接热点的时候要用到(如果需要修改热点的名字和密码, 需要先关闭 **Hotspot Enable**, 然后才能修改)



6) 此时可以拿出你的手机, 如果一切正常, 在手机搜索到的 WI-FI 列表中就能找到上图 **Hotspot name** 下面显示的同名 (这里为 **AndroidAP_7132**) 的 WIFI 热点。然后可以点击 **AndroidAP_7132** 连接热点, 密码在上图的 **Hotspot password** 下面可以看到

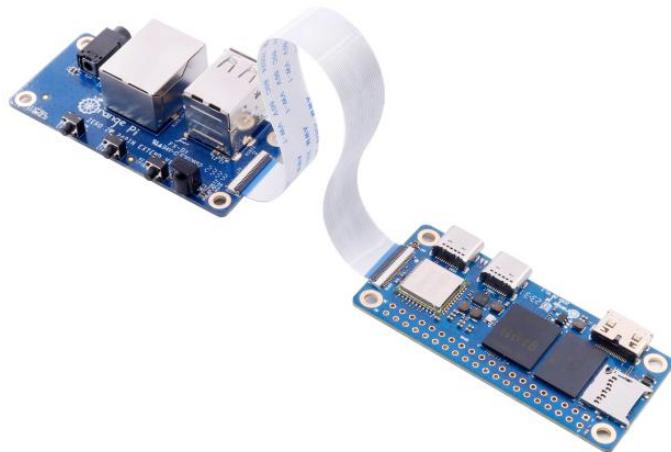


7) 连接成功后显示如下图所示 (不同手机界面会有区别, 具体界面以你手机显示的为准)。此时就可以在手机上打开一个网页看下能否上网了, 如果能正常打开网页, 说明开发板的 **WI-FI Hotspot** 能正常使用



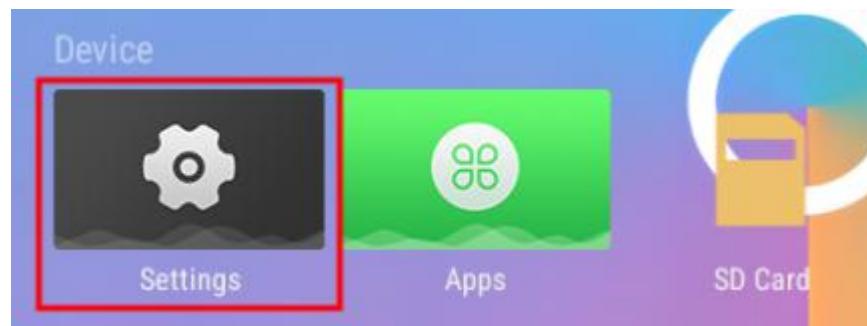
6.9. 查看以太网口 IP 地址的方法

- 1) 开发板主板上是没有有线网络接口的，我们可以通过 24pin 扩展板来扩展百兆以太网

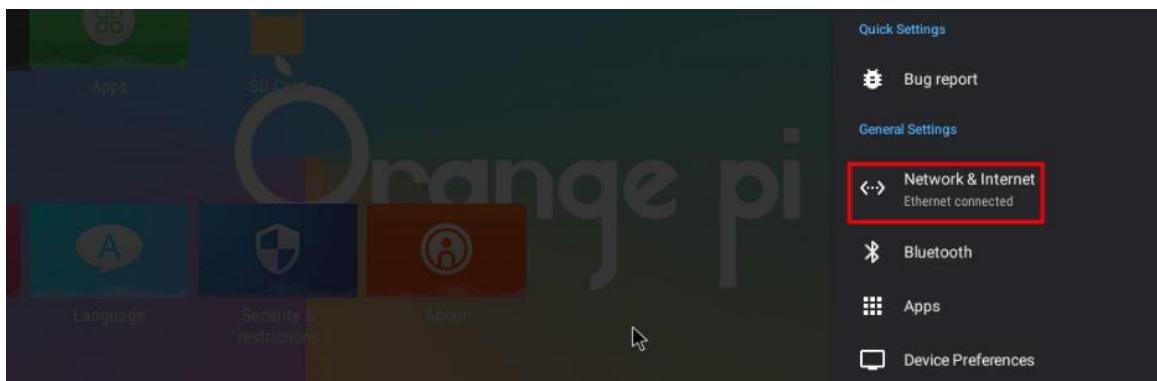


- 2) 然后请确保扩展板的网口连接到了路由器或者交换机

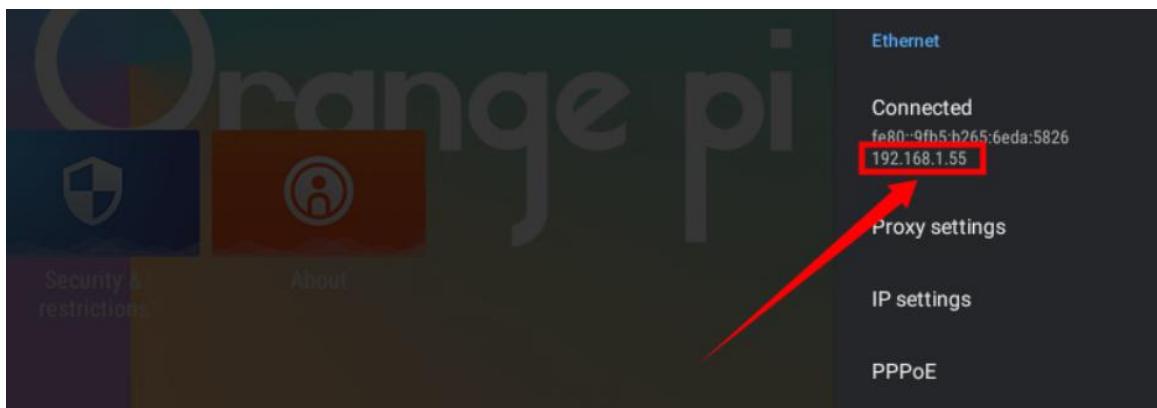
- 3) 然后打开 **Settings**



- 4) 然后选择 **Network & Internet**



5) 然后在下图所示的位置就能看到开发板有线网口的 IP 地址了

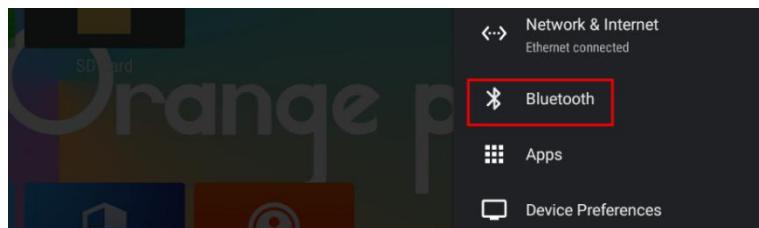


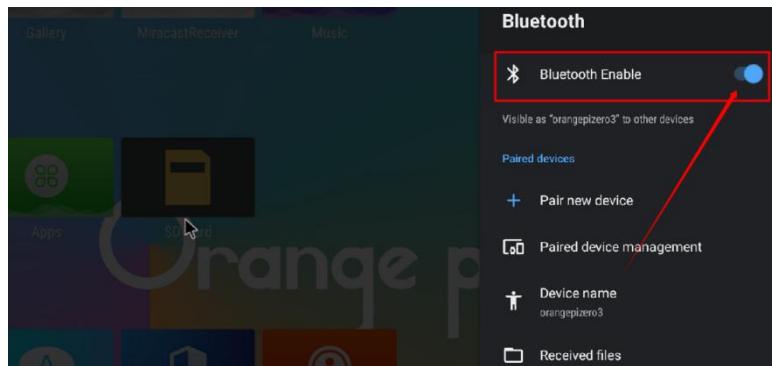
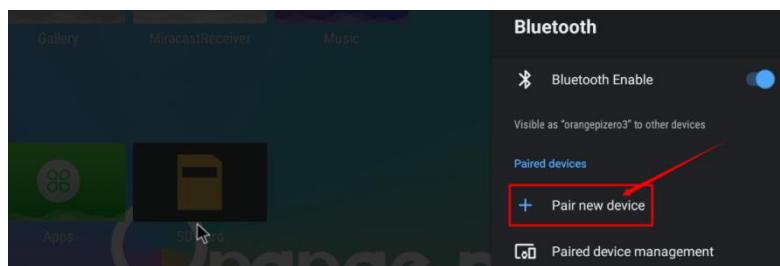
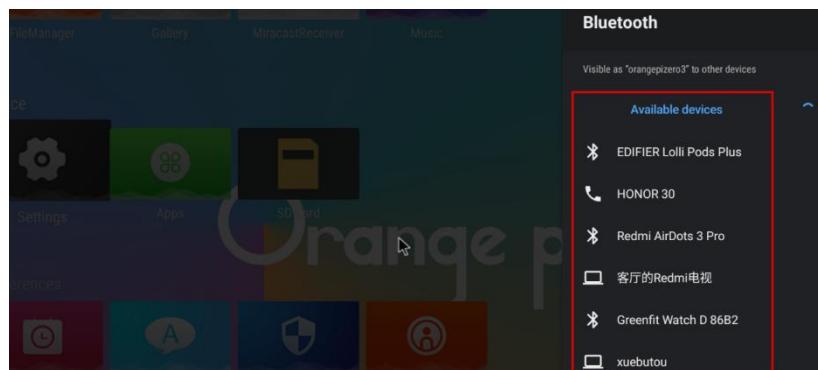
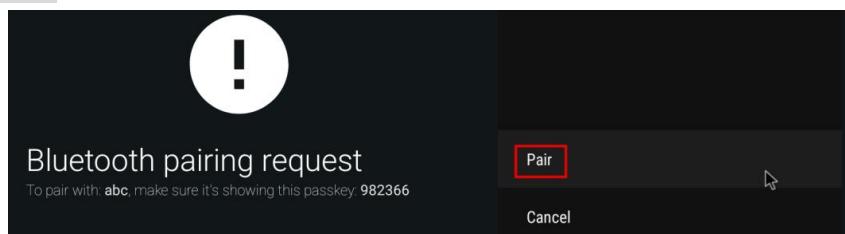
6. 10. 蓝牙的连接方法

1) 首先选择 **Settings**



2) 然后选择 **Bluetooth**



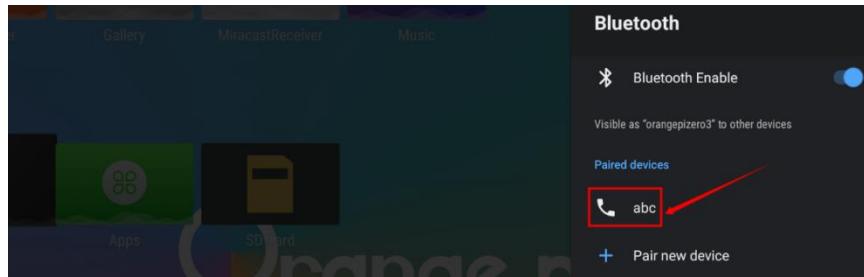
3) 然后打开 **Bluetooth Enable**4) 然后点击 **Pair new device** 开始扫描周围的蓝牙设备5) 搜索到的蓝牙设备会在 **Available devices** 下面显示出来6) 然后点击想要连接的蓝牙设备就可以开始配对了，当弹出下面的界面时，请使用鼠标选择 **Pair** 选项



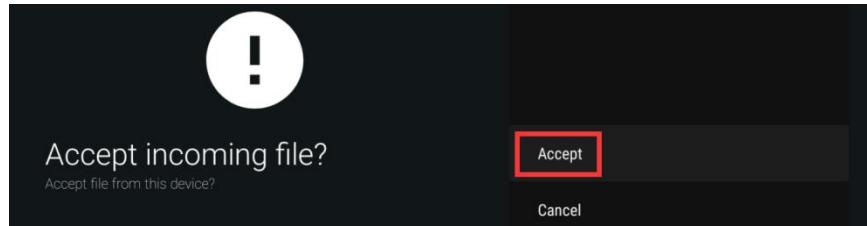
7) 这里测试的是开发板和**安卓手机**蓝牙的配置过程，此时在手机上会弹出下面的确认界面，在手机上也点击配对按钮后就会开始配对过程



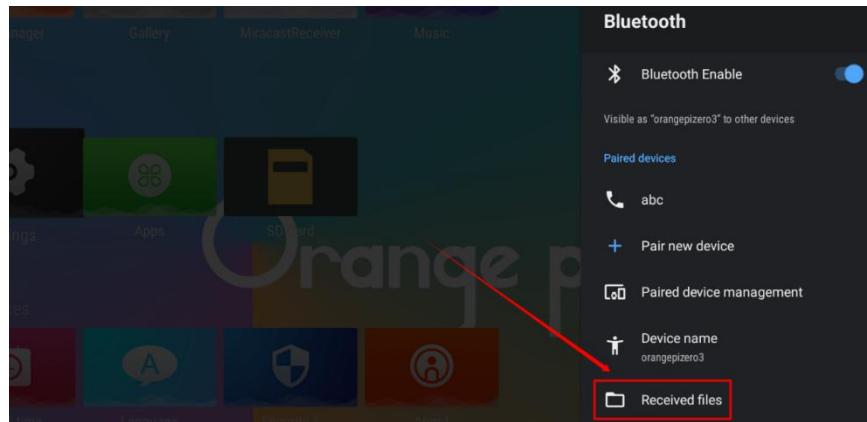
8) 配对完成后，再打开 **Paired devices** 下面就可以看到已配对的蓝牙设备



9) 此时可以使用手机蓝牙给开发板发送一张图片，发送后，在开发板的安卓系统中可以看到下面的确认界面，然后点击 **Accept** 就可以开始接收手机发过来的图片了

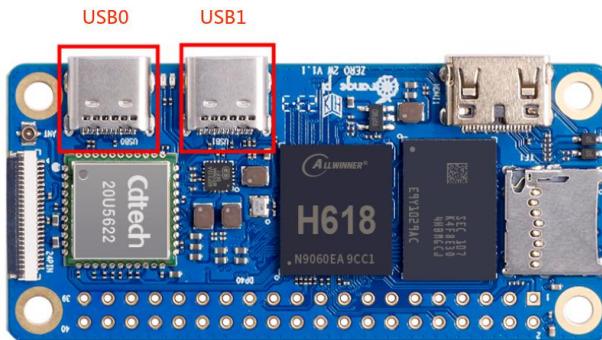


10) 开发板 Android 系统蓝牙接收到的图片可以打开 **Received files** 中查看



6. 11. USB0 设置为 HOST 模式的方法

如下图所示，开发板的主板上有两个 Type-C 类型的接口：USB0 和 USB1，这两个接口都可以用来给开发板供电，也都可以用来当做 USB2.0 HOST 接口。USB0 和 USB1 的区别是：USB0 除了可以设置为 HOST 模式外，还可以设置为 Device 模式，而 USB1 只有 HOST 模式。



Orange Pi 发布的 Android12 TV 系统 USB0 默认设置为 Device 模式，所以在不需要使用 USB0 Device 模式时（ADB 功能需要确保 USB0 为 Device 模式），建议使用 USB0 来供电，这样 USB1 就可以直接用来接 USB 设备。

如果想使用 USB0 来接 USB 设备，需要把 USB0 设置为 HOST 模式，方法如下所示：

- e. 运行下面的命令可以将 USB0 设置为 HOST 模式：

```
apollo-p2:/ # cat /sys/devices/platform/soc@3000000/soc@3000000:usbc0@0/usb_host
host_chose finished!
apollo-p2:/ #
```

- f. 运行下面的命令可以切回 Device 模式

```
apollo-p2:/ # cat /sys/devices/platform/soc@3000000/soc@3000000:usbc0@0/usb_device
```



```
device_chose finished!
```

```
apollo-p2:/ #
```

g. 查看 USB0 当前模式的命令为

```
apollo-p2:/ # cat /sys/devices/platform/soc@3000000/soc@3000000\::usbc0@0/otg_role
```

```
usb_host
```

6. 12. USB 摄像头使用方法

1) 首先在开发板的 USB 接口中插入 USB (UVC 协议) 摄像头

2) USB 摄像头如果识别正常, 在 /dev 下会生成相应的 video 设备节点

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

```
/dev/video0
```

3) 然后确保 Ubuntu PC 和开发板的 adb 连接正常, adb 的使用方法请参考 [ADB 的使用方法](#) 一小节的说明

4) 在开发板资料下载页面的 [官方工具](#) 中下载 USB 摄像头测试 APP

官方资料



官方工具

[\[下载\]](#)



用户手册

[\[下载\]](#)



原理图

[\[下载\]](#)

[保存到网盘](#)

官方工具

① 2020-11-03 14:09 失效时间: 永久有效

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

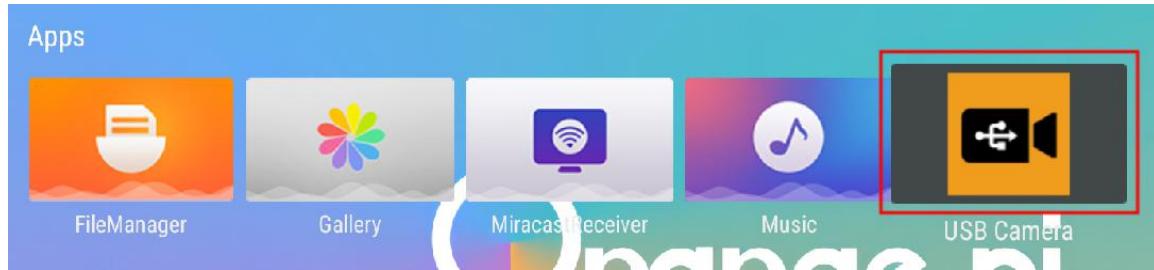
文件名	大小	修改日期
usbcamera.apk	20M	2020-11-04 13:56
rootcheck.apk	2M	2020-11-04 13:48
REFile.apk	4.4M	2020-11-04 13:48
bledemo.apk	4.1M	2020-11-04 13:48

5) 然后使用 adb 命令安装 USB 摄像头测试 APP 到 Android 系统中, 当然也可以使用 U 盘拷贝的方式进行安装

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



6) 安装完后在 Android 的桌面可以看到 USB 摄像头的启动图标



7) 然后双击打开 USB 摄像头 APP 就可以看到 USB 摄像头的输出视频了

6. 13. Android 系统 ROOT 说明

Orange Pi 发布的 Android 系统已经 ROOT，可以使用下面的方法来测试。

1) 在开发板资料下载页面的官方工具中下载 **rootcheck.apk**

官方资料


官方工具
[\[下载\]](#)


用户手册
[\[下载\]](#)


原理图
[\[下载\]](#)

官方工具

① 2020-11-03 14:09 失效时间: 永久有效

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

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

[保存到网盘](#)

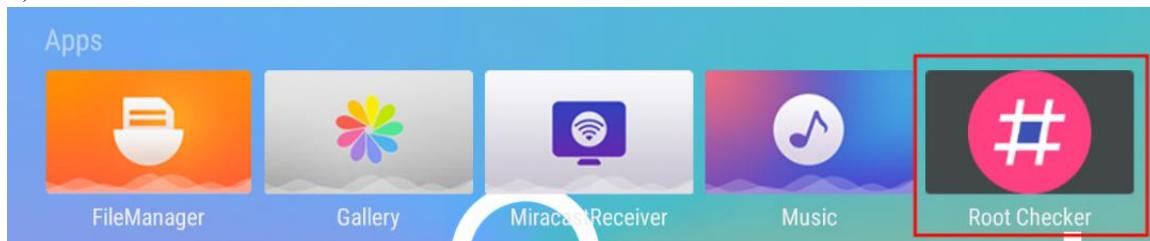
2) 然后确保 Ubuntu PC 和开发板的 adb 连接正常，adb 的使用方法请参考 [ADB 的使用方法](#)一小节的说明

3) 然后使用 adb 命令安装 rootcheck.apk 到 Android 系统中，当然也可以使用 U 盘拷贝的方式进行安装

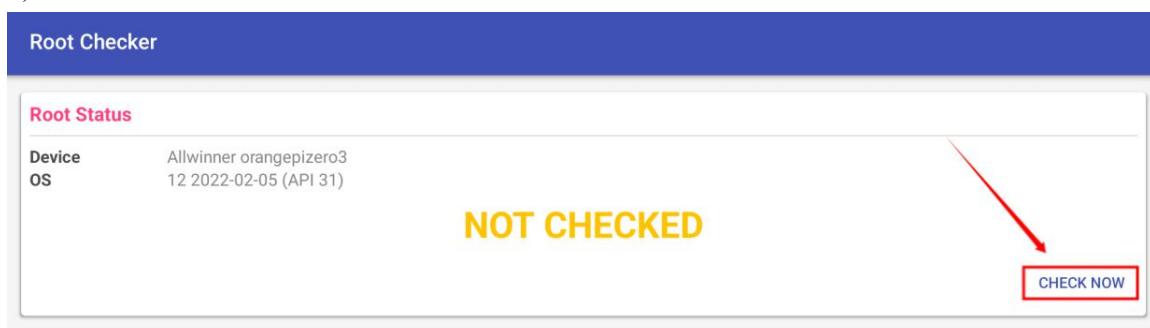
```
test@test:~$ adb install rootcheck.apk
```



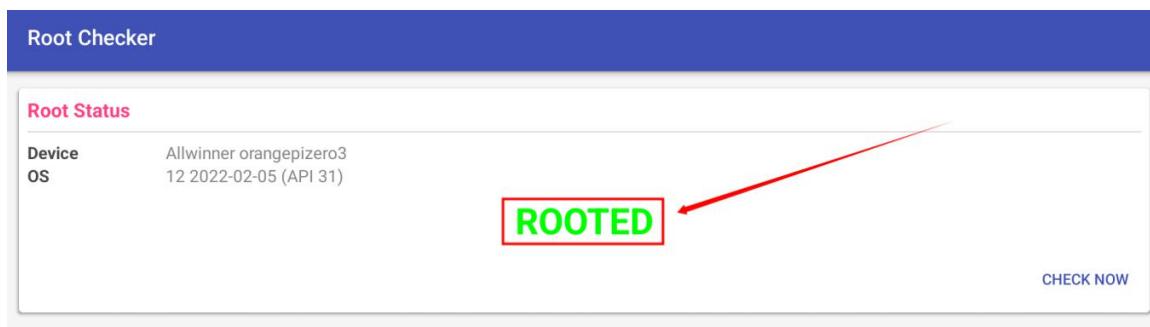
4) 安装完后在 Android 的桌面可以看到 ROOT 测试工具的启动图标



5) 第一次打开 ROOT 测试工具后的显示界面如下图所示



6) 然后就可以点击 **CHECK NOW** 开始 Android 系统的 ROOT 状态的检查, 检查完后的显示如下所示, 可以看到 Android 系统已取得 ROOT 权限

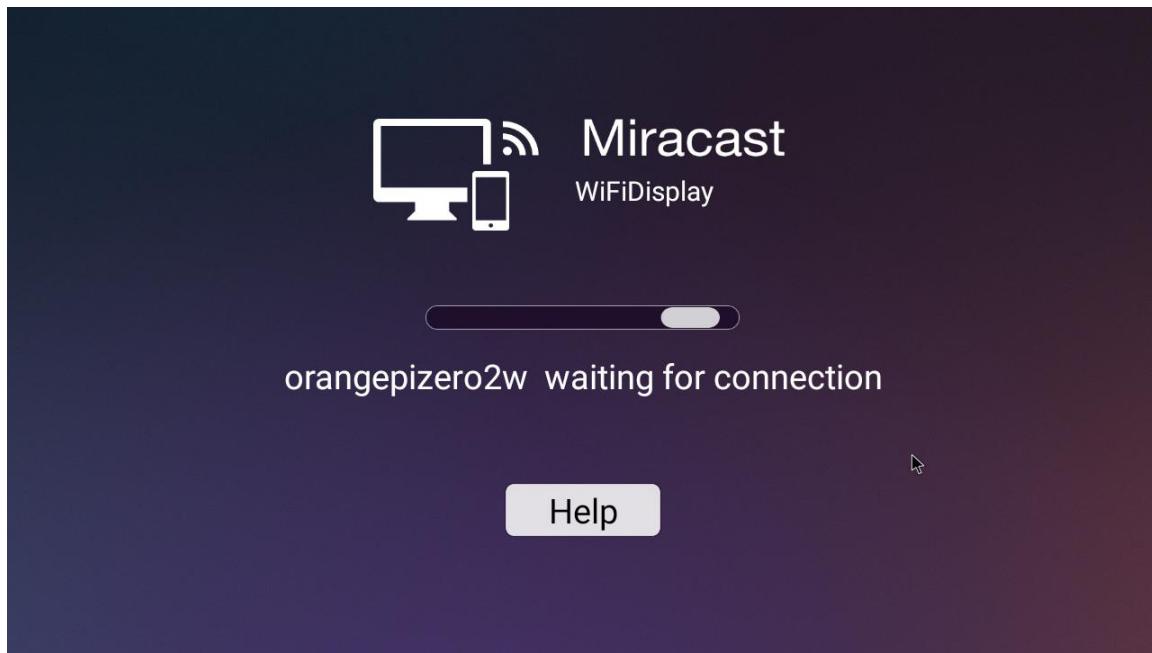


6. 14. 使用 MiracastReceiver 将手机屏幕投屏到开发板的方法

- 1) 首先请确保开发板和手机都连接了同一个 WIFI 热点, 开发板连接 WIFI 的方法请参考 [WI-FI 的连接方法](#)一小节的说明
- 2) 然后打开开发板安卓系统中的 **MiracastReceiver** 应用



3) MiracastReceiver 打开后的界面如下所示



4) 然后在手机设置中找到投屏功能，这里以小米 12S Pro 手机为例，其他品牌的手机请自行研究下，如下图所示，点击红色方框位置的按钮即可打开手机的投屏功能

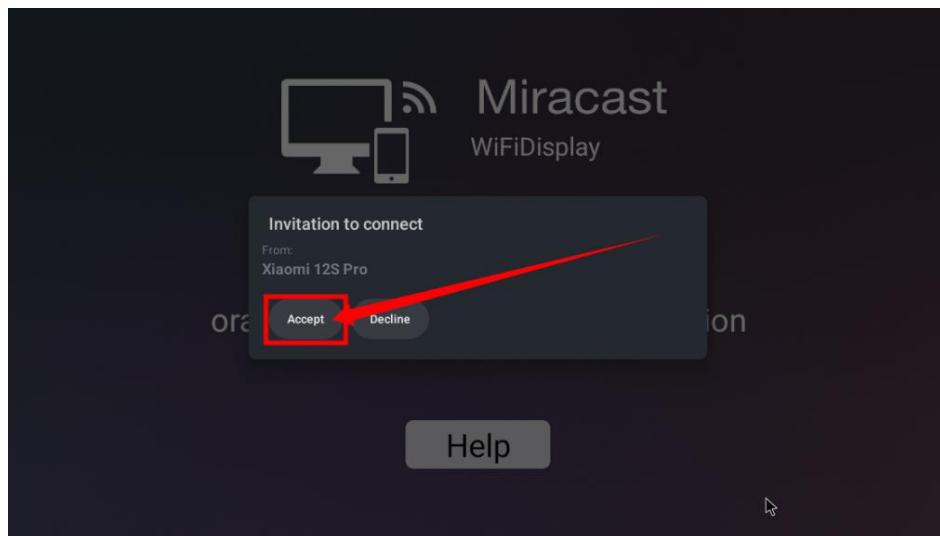




5) 等待一段时间后在手机上就能看到搜索到的可连接的设备，然后我们选择开发板对应的设备连接即可



6) 然后在开发板的 **MiracastReceiver** 应用界面会弹出下图所示的选择框，这里我们选择 **Accept** 即可

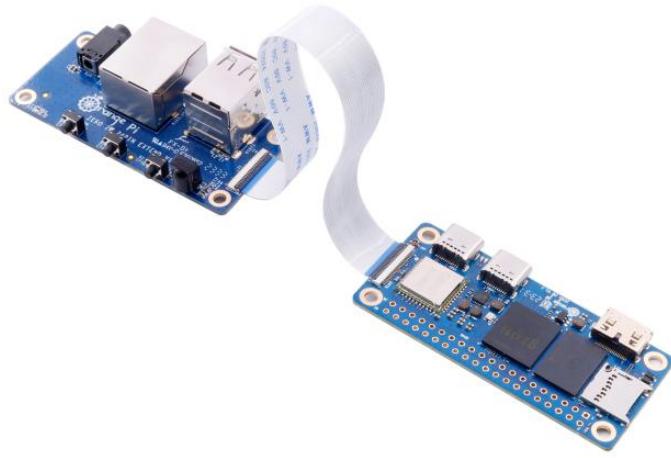


7) 然后就能在开发板连接的 HDMI 屏幕上看到手机屏幕的内容了



6.15. 通过按键或红外遥控开关机的方法

我们可以通过开关机按键或者红外遥控来关闭或开启开发板的安卓系统。但是需要注意的是，开发板主板上是没有开关机按键和红外接收器的，需要通过 24pin 扩展板来扩展。



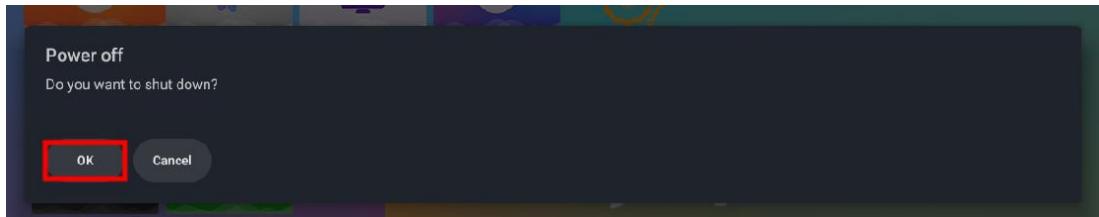
24pin 扩展板上的开关机按键所在位置如下图所示：



红外遥控电源按键所在位置如下所示：



关机时，我们需要长按开关机按键或者红外遥控上的电源按键，然后安卓系统会弹出下图所示的确认对话框，然后选择 **OK** 就会关闭安卓系统了。



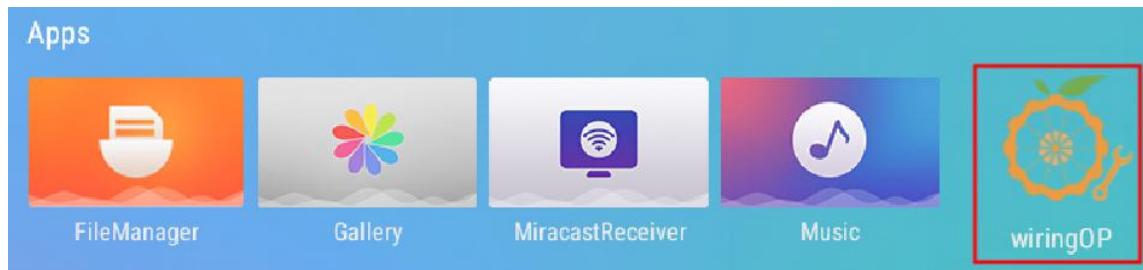
关机后，再次长按开关机按键或者红外遥控上的电源按键就会开机了。

6.16. 40pin 接口 GPIO、UART、SPI 测试

注意：40pin接口上的排针默认是不焊接的，需要自己焊接上去才能使用。

6.16.1. 40pin 的 GPIO 口测试方法

- 1) 首先在桌面中打开 wiringOP APP



2) 然后点击 **GPIO_TEST** 按钮打开 GPIO 测试界面



3) GPIO 测试界面如下图所示，左边的两排 **CheckBox** 按钮和 40pin 引脚是一一对应的关系。当勾选 **CheckBox** 按钮时，对应的 GPIO 引脚会被设置为 **OUT** 模式，引脚电平设置为高电平；当取消勾选时，GPIO 引脚电平会设置为低电平；当点击 **GPIO READALL** 按钮时，可以获取到 wPi 号、GPIO 模式、引脚电平等信息；当点击 **BLINK ALL GPIO** 按钮时，会让所有的 GPIO 口循环输出高低电平，使用这个功能可以用来测试 40pin 中所有的 GPIO 口。





4) 然后点击 **GPIO READALL** 按钮，输出信息如下图所示：

wiringOP											
GPIO READALL											
BLINK ALL GPIO											
GPIO	wPi	Name	Mode	V	ZERO2W	Physical	V	Mode	Name	wPi	GPIO
3.3V		5V									
SDA.1		5V									
SCL.1		GND									
PWM3		TXD.0									
GND		RXD.0									
TXD.5		PI01									
RXD.5		GND									
TXD.2		PWM4									
3.3V		PH04									
MOSI.1		GND									
MISO.1		RXD.2									
SCLK.1		CE.0									
GND		CE.1									
SDA.2		SCL.2									
PI00		GND									
PI15		PWM1									
PI12		GND									
PI02		PC12									
PI16		PI04									
GND		PI03									

5) 开发板 40pin 中总共有 28 个 GPIO 口可以使用，下面以 12 号引脚——对应 GPIO 为 PI01——对应 wPi 序号为 6——为例演示如何设置 GPIO 口的高低电平。首先点击 12 号引脚对应的 **CheckBox** 按钮，当按钮为选中状态时，12 号引脚会设置为高电平，设置完后可以使用万用表测量引脚的电压的数值，如果为 **3.3v**，说明设置高电平成功。

wiringOP											
GPIO READALL											
BLINK ALL GPIO											
GPIO	wPi	Name	Mode	V	ZERO2W	Physical	V	Mode	Name	wPi	GPIO
3.3V		5V									
SDA.1		5V									
SCL.1		GND									
PWM3		TXD.0									
GND		RXD.0									
TXD.5	<input checked="" type="checkbox"/>	PI01									
RXD.5		GND									
TXD.2		PWM4									
3.3V		PH04									
MOSI.1		GND									
MISO.1		RXD.2									
SCLK.1		CE.0									
GND		CE.1									
SDA.2		SCL.2									
PI00		GND									
PI15		PWM1									
PI12		GND									
PI02		PC12									
PI16		PI04									
GND		PI03									

6) 然后点击 **GPIO READALL** 按钮，可以看到当前的 12 号引脚模式为 **OUT**，引脚



电平为高电平

wiringOP															
3.3V		5V		GPIO READALL			BLINK ALL GPIO								
SDA.1		5V													
PWM3	<input type="checkbox"/>	RXD.0	<input type="checkbox"/>	GPIO	wPi	Name	Mode	V	ZERO2W	Physical	V	Mode	Name	wPi	GPIO
264	0	3.3V		SDA.1		IN	1	1	2			5V			
263	1	SCL.1		SCL.1		IN	1	3	4			5V			
269	2	PWM3		PWM3		OFF	0	7	8	0	ALT2	TxD.0	3	224	
226	5	TxD.5		TXD.5		ALT2	0	11	12	1	OUT	RxD.0	4	225	
227	7	RxD.5		RxD.5		ALT2	0	13	14	0	ALT2	PI01	6	257	
261	8	TxD.2		TXD.2		ALT3	0	15	16	0	OFF	PWM4	9	270	
3.3V		PH04		3.3V				17	18	0	OFF	PH04	10	228	
MOSI.1		GND		MOSI.1				19	20	0	ALT3	RxD.2	13	262	
MISO.1		RxD.2		MISO.1		ALT4	0	21	22	0	OFF	CE.0	15	229	
SCLK.1		CE.0		SCLK.1		ALT4	0	23	24	0	ALT4	CE.1	16	233	
GND		CE.1		GND		GND		25	26	0	IN	SCL.2	18	265	
266	17	SDA.2		SDA.2		IN	1	27	28	1	OFF	GND			
256	19	RxD.0		PI00		OFF	0	29	30	0	OFF	PWM1	21	267	
271	20	TXD.5		PI15		OFF	0	31	32	0	OFF	PC12	24	76	
268	22	RxD.5		PI12		OFF	0	33	34	0	OUT	PI04	26	260	
258	23	TxD.2		PI02		OFF	0	35	36	1	OFF	PI03	27	259	
272	25	SDA.2		PI16		OFF	0	37	38	0	OFF				
PI00		GND				GND		39	40	0	OFF				
PI15		PWM1													
PI12		GND													
PI02		PC12													
PI16		PI04													
GND		PI03													

7) 再次点击下图的 **CheckBox** 按钮取消勾选状态，12 号引脚就会设置为低电平，设置完后可以使用万用表测量引脚的电压的数值，如果为 **0v**，说明设置低电平成功

wiringOP															
3.3V		5V		GPIO READALL			BLINK ALL GPIO								
SDA.1		5V													
PWM3	<input type="checkbox"/>	RXD.0	<input type="checkbox"/>	GPIO	wPi	Name	Mode	V	ZERO2W	Physical	V	Mode	Name	wPi	GPIO
264	0	3.3V		SDA.1		IN	1	1	2			5V			
263	1	SCL.1		SCL.1		IN	1	3	4			5V			
269	2	PWM3		PWM3		OFF	0	7	8	0	ALT2	TxD.0	3	224	
226	5	TxD.5		TXD.5		ALT2	0	11	12	1	OUT	PI01	6	257	
227	7	RxD.5		RxD.5		ALT2	0	13	14	0	OFF	GND			
261	8	TxD.2		TXD.2		ALT3	0	15	16	0	OFF	PWM4	9	270	
3.3V		PH04		3.3V				17	18	0	OFF	PH04	10	228	
MOSI.1		GND		MOSI.1		ALT4	0	19	20	0	ALT3	RxD.2	13	262	
MISO.1		RxD.2		MISO.1		ALT4	0	21	22	0	OFF	CE.0	15	229	
SCLK.1		CE.0		SCLK.1		ALT4	0	23	24	0	OFF	CE.1	16	233	
GND		CE.1		GND		GND		25	26	0	IN	SCL.2	18	265	
266	17	SDA.2		SDA.2		IN	1	27	28	1	OFF	PWM1	21	267	
256	19	RxD.0		PI00		OFF	0	29	30	0	OFF	PC12	24	76	
271	20	TXD.5		PI15		OFF	0	31	32	0	OUT	PI04	26	260	
268	22	RxD.5		PI12		OFF	0	33	34	0	OFF	PI03	27	259	
258	23	TxD.2		PI02		OFF	0	35	36	1	OUT				
272	25	SDA.2		PI16		OFF	0	37	38	0	OFF				
PI00		GND				GND		39	40	0	OFF				
PI15		PWM1													
PI12		GND													
PI02		PC12													
PI16		PI04													
GND		PI03													

8) 然后点击 **GPIO READALL** 按钮，可以看到当前的 12 号引脚模式为 OUT，引脚电平为低电平



wiringOP											
GPIO READALL				BLINK ALL GPIO							
GPIO	vP1	Name	Mode	V	Physical	V	Mode	Name	vP1	GPIO	
3.3V		3.3V		1	2			5V			
SDA.1		SDA.1	IN	1	3	4		5V			
SCL.1		SCL.1	IN	5	6			GND			
PWM3		PWM3	OFF	0	7	8	0	ALT2	TXO.0	3	
GND		GND		9	10	0	ALT2	RXD.0	4	224	
TXD.5		TXD.5	PI01	0	11	12	0	OUT	PI01	5	225
RXD.5		RXD.5	GND	0	13	14	0	IN	GND	6	257
TXD.2		TXD.2	PWM4	0	15	16	0	OFF	PWM4	9	270
3.3V		3.3V	PH04	0	17	18	0	OFF	PH04	10	228
MOSI.1		MOSI.1	ALT4	0	19	20	0	AL1_3	RXD.2	13	262
MISO.1		MISO.1	ALT4	0	21	22	0	OFF	CE_0	15	229
SCLK.1		SCLK.1	ALT4	0	23	24	0	OFF	CE_1	16	233
GND		GND		0	25	26	0	ALT4	SCLK.2	18	265
SDA.2		SDA.2	IN	0	27	28	1	IN	GND	21	267
PI00		PI00	OFF	0	29	30	1	OFF	PWM1	24	76
PI15		PI15	OFF	0	31	32	0	OFF	PI14	26	260
PI12		PI12	OFF	0	33	34	0	OFF	PI04	27	259
PI02		PI02	OFF	0	35	36	1	OUT	PI03	27	
PI16		PI16	OFF	0	37	38	0	OFF			
GND		GND		0	39	40	0	OFF			
PI03		PI03									

6. 16. 2. 40pin 的 UART 测试方法

1) 由下表可知，Android12 TV 系统默认可用的 uart 有 uart2 和 uart5。请注意 uart0 默认设置为调试串口，请不要把 uart0 当成普通串口使用。

GPIO序号	GPIO	功能	引脚
		3.3V	1
264	PI8	TWI1-SDA	3
263	PI7	TWI1-SCL	5
269	PI13	PWM3	7
		GND	9
226	PH2	UART5_TX	11
227	PH3	UART5_RX	13
261	PI5	UART2_TX	15
		3.3V	17
231	PH7	SPI1_MOSI	19
232	PH8	SPI1_MISO	21
230	PH6	SPI1_CLK	23
		GND	25
266	PI10	TWI2-SDA	27
256	PI0		29
271	PI15		31
268	PI12	PWM2	33
258	PI2		35

引脚	功能	GPIO	GPIO序号
2	5V		
4	5V		
6	GND		
8	UART0_TX	PH0	224
10	UART0_RX	PH1	225
12		PI1	257
14	GND		
16	PWM4	PI14	270
18		PH4	228
20	GND		
22	UART2_RX	PI6	262
24	SPI1_CS0	PH5	229
26	SPI1_CS1	PH9	233
28	TWI2-SCL	PI9	265
30	GND		
32	PWM1	PI11	267
34	GND		
36		PC12	76



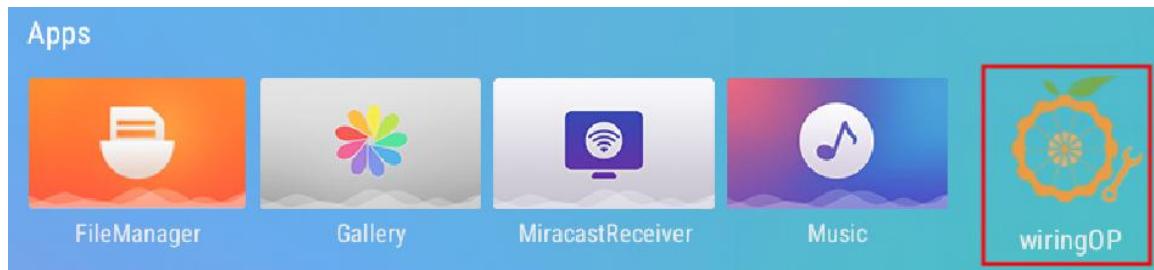
272	PI16		37
		GND	39

38		PI4	260
40		PI3	259

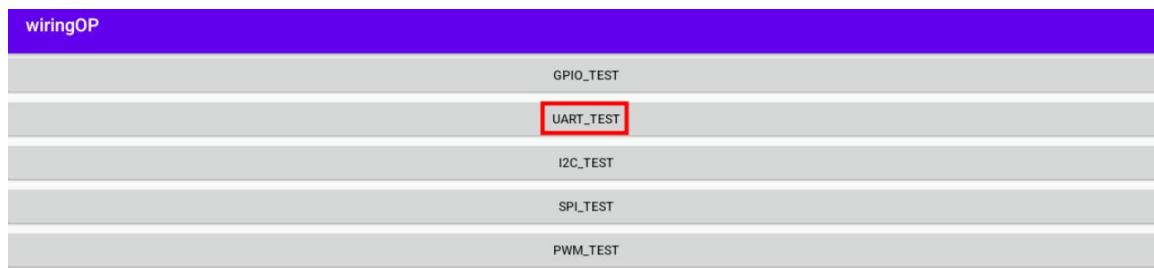
2) uart2 对应的设备节点为 **/dev/ttyAS2**, uart5 对应的设备节点为 **/dev/ttyAS5**

```
apollo-p2:/ # ls /dev/ttyAS*
/dev/ttyAS0  /dev/ttyAS1  /dev/ttyAS2  /dev/ttyAS5
```

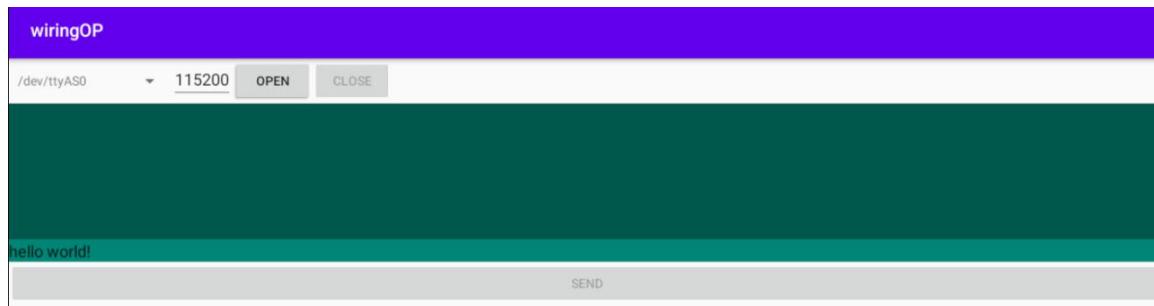
3) 首先在桌面中打开 wiringOP APP



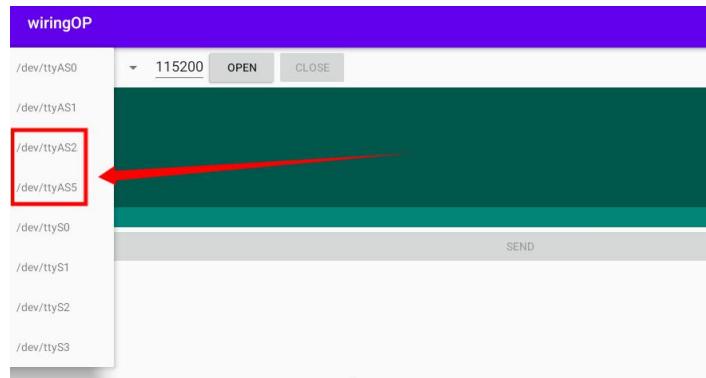
4) 然后点击 **UART_TEST** 按钮打开 UART 测试界面



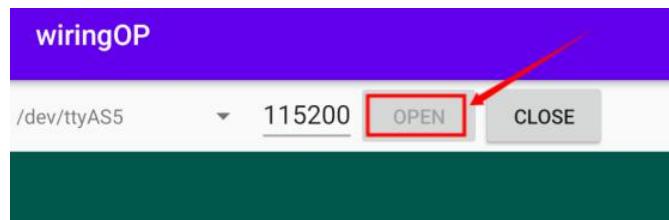
5) wiringOP 的串口测试界面如下图所示



6) 然后在选择框中选择 **/dev/ttyAS2** 或者 **/dev/ttyAS5** 节点

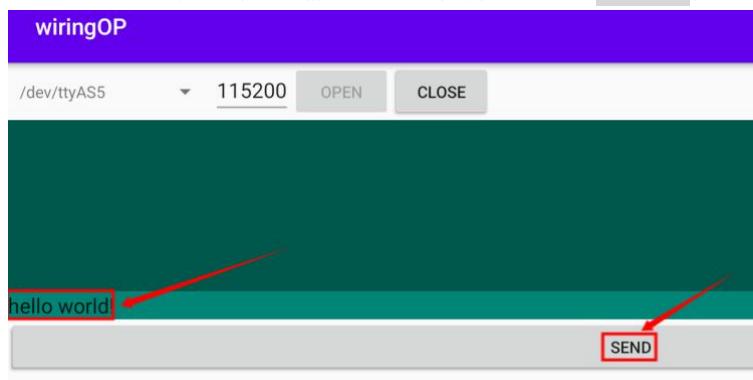


7) 再在编辑框中输入想要设置的波特率，然后点击 **OPEN** 按钮打开 uart 节点，打开成功后，**OPEN** 按钮变为不可选中状态，**CLOSE** 按钮和 **SEND** 按钮变为可选中状态

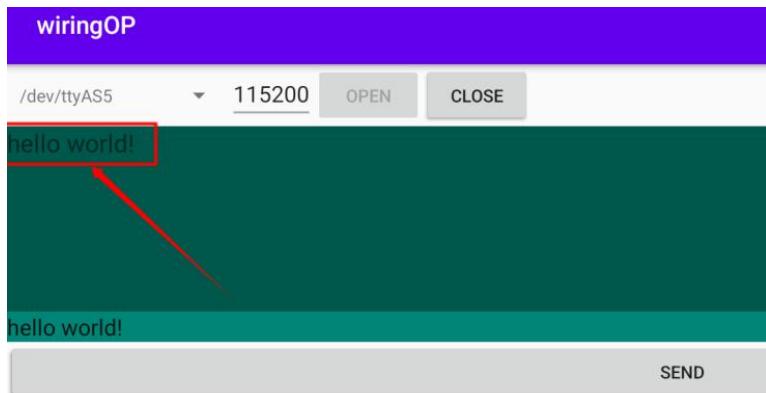


8) 然后使用杜邦线短接 uart 的 rx 和 tx 引脚

9) 然后可以在下面的发送编辑框中输入一段字符，点击 **SEND** 按钮开始发送



10) 如果一切正常，接收框内会显示已接收到的字符串



6.16.3. 40pin 的 SPI 测试方法

1) 由下表可知，40pin 接口可用的 spi 为 spi1，有两个片选引脚 cs0 和 cs1

GPIO序号	GPIO	功能	引脚
		3.3V	1
264	PI8	TWI1-SDA	3
263	PI7	TWI1-SCL	5
269	PI13	PWM3	7
		GND	9
226	PH2	UART5_TX	11
227	PH3	UART5_RX	13
261	PI5	UART2_TX	15
		3.3V	17
231	PH7	SPI1_MOSI	19
232	PH8	SPI1_MISO	21
230	PH6	SPI1_CLK	23
		GND	25
266	PI10	TWI2-SDA	27
256	PI0		29
271	PI15		31
268	PI12	PWM2	33
258	PI2		35

引脚	功能	GPIO	GPIO序号
2	5V		
4	5V		
6	GND		
8	UART0_TX	PH0	224
10	UART0_RX	PH1	225
12		PI1	257
14	GND		
16	PWM4	PI14	270
18		PH4	228
20	GND		
22	UART2_RX	PI6	262
24	SPI1_CS0	PH5	229
26	SPI1_CS1	PH9	233
28	TWI2-SCL	PI9	265
30	GND		
32	PWM1	PI11	267
34	GND		
36		PC12	76

2) SPI1 CS0 对应的设备节点为 /dev/spidev1.0， SPI1 CS1 对应的设备节点为
/dev/spidev1.1

```
apollo-p2:/ # ls /dev/spidev1.*
```



/dev/spidev1.0 /dev/spidev1.1

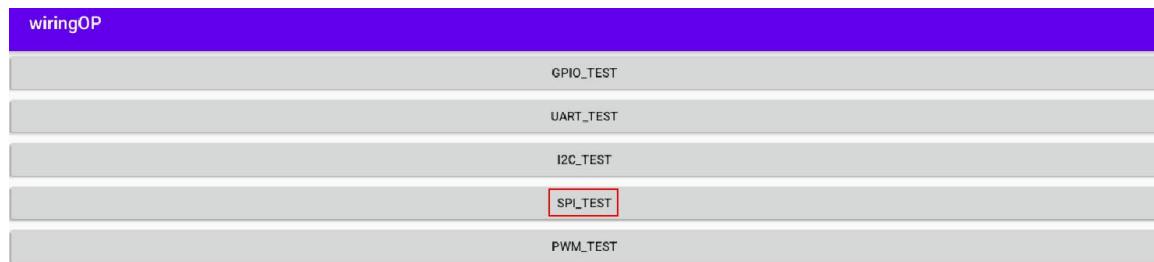
3) 这里演示下通过 **w25qxx** 模块来测试 SPI1 接口，首先在 SPI1 接口接入 w25qxx 模块

如果没有 **w25qxx** 模块也没关系，因为开发板上有一个 **SPIFlash** 接在了 **SPI0** 上，在安卓中 **SPI0** 的配置默认也打开了，所以我们也可以直接使用板载的 **SPIFlash** 测试。

4) 然后在桌面中打开 wiringOP APP



5) 然后点击 **SPI_TEST** 按钮打开 SPI 的测试界面



6) 然后在左上角选择 spi 的设备节点，如果直接测试板载的 **SPIFlash**，那么保持默认的 **/dev/spidev0.0** 即可，如果在 40pin 的 spi1 cs0 上接了 **w25qxx** 模块，那么就请选择 **/dev/spidev1.0**，如果在 40pin 的 spi1 cs1 上接了 **w25qxx** 模块，那么就请选择 **/dev/spidev1.1**





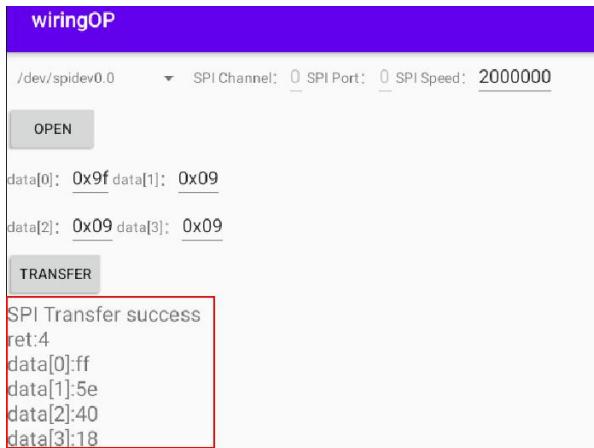
7) 然后点击 **OPEN** 按钮初始化 SPI



8) 然后填充需要发送的字节，比如读取板载 SPIFlash 的 ID 信息，在 data[0]中填入地址 0x9f，然后点击 **TRANSFER** 按钮



9) 最后 APP 会显示读取到的板载 SPI Flash 的 ID 信息



10) 如果是读取接在 40pin SPI1 上的 w25qxx 模块，和板载 SPI Flash 的 ID 信息也是类似的



6. 16. 4. 40pin 的 I2C 测试方法

1) 由下表可知，Android12 TV 系统默认打开了 i2c1 和 i2c2

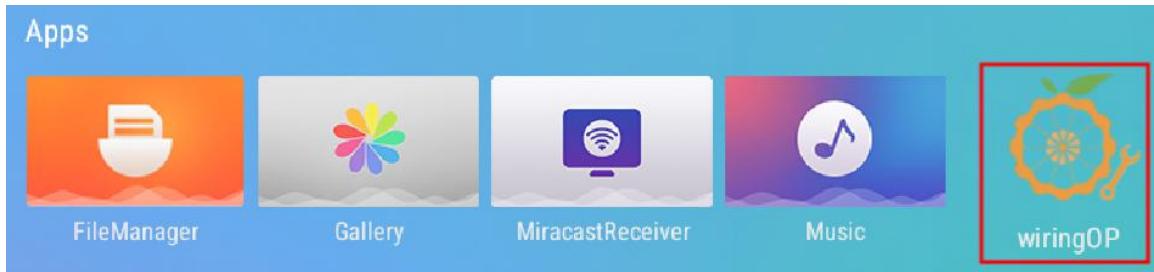
GPIO序号	GPIO	功能	引脚
		3.3V	1
264	PI8	TWI1-SDA	3
263	PI7	TWI1-SCL	5
269	PI13	PWM3	7
		GND	9
226	PH2	UART5_TX	11
227	PH3	UART5_RX	13
261	PI5	UART2_TX	15
		3.3V	17
231	PH7	SPI1_MOSI	19
232	PH8	SPI1_MISO	21
230	PH6	SPI1_CLK	23
		GND	25
266	PI10	TWI2-SDA	27
256	PI0		29
271	PI15		31
268	PI12	PWM2	33
258	PI2		35
272	PI16		37
		GND	39

引脚	功能	GPIO	GPIO序号
2	5V		
4	5V		
6	GND		
8	UART0_TX	PH0	224
10	UART0_RX	PH1	225
12		PI1	257
14	GND		
16	PWM4	PI14	270
18		PH4	228
20	GND		
22	UART2_RX	PI6	262
24	SPI1_CS0	PH5	229
26	SPI1_CS1	PH9	233
28	TWI2-SCL	PI9	265
30	GND		
32	PWM1	PI11	267
34	GND		
36		PC12	76
38		PI4	260
40		PI3	259

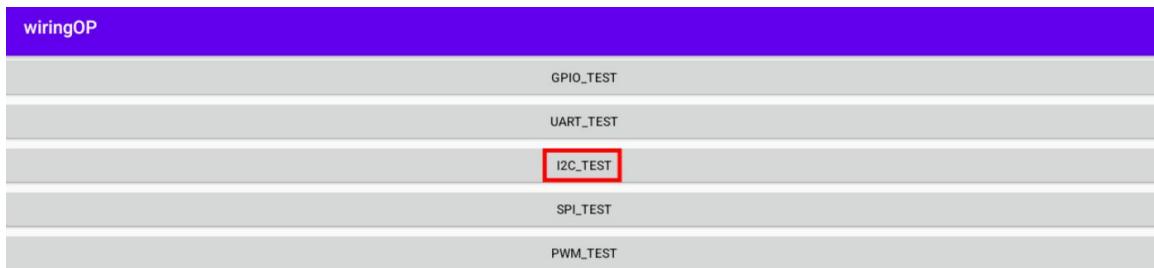
2) i2c1 对应的设备节点为 /dev/i2c-1，i2c2 对应的设备节点为 /dev/i2c-2

```
apollo-p2:/ # ls /dev/i2c-*
/dev/i2c-1  /dev/i2c-2  /dev/i2c-5
```

3) 首先在桌面中打开 wiringOP APP



4) 然后点击 **I2C_TEST** 按钮打开 i2c 的测试界面



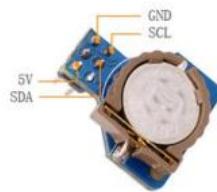
5) wiringOP 的 i2c 测试界面如下图所示



6) 然后点击左上角的设备节点选择框选择想要测试的 i2c



7) 然后在 40pin 的 i2c 引脚上接一个 i2c 设备，这里以 ds1307 rtc 模块为例



8) ds1307 rtc 模块的 i2c 地址为 0x68，接好线后，我们可以在串口命令行中使用 **i2cdetect -y 1** 或 **i2cdetect -y 2** 命令查看下是否能扫描到 ds1307 rtc 模块的 i2c 地址。如果能看到 0x68 这个地址，说明 ds1307 rtc 模块接线正确。

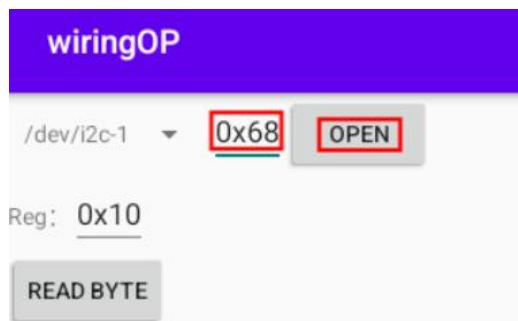
```
apollo-p2:/ # i2cdetect -y 1
```

或

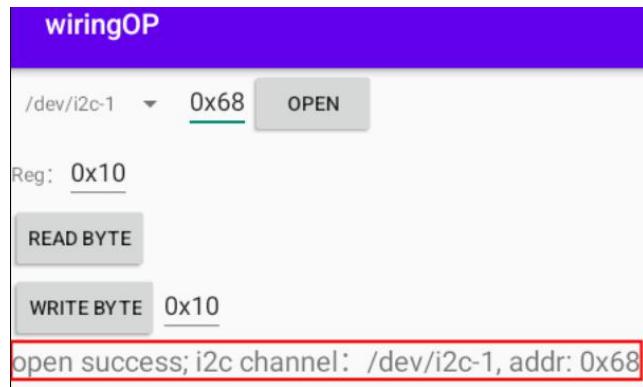
```
apollo-p2:/ # i2cdetect -y 2
```

```
apollo-p2:/ # i2cdetect -y 1
      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          - - - - - - - - - - - - - - - - - - - -
10:          - - - - - - - - - - - - - - - - - - - -
20:          - - - - - - - - - - - - - - - - - - - -
30:          - - - - - - - - - - - - - - - - - - - -
40:          - - - - - - - - - - - - - - - - - - - -
50:          - - - - - - - - - - - - - - - - - - - -
60:          - - - - - - - - - - 68 - - - - - - - -
70:          - - - - - - - - - - - - - - - - - - - -
apollo-p2:/ #
```

9) 然后在 wiringOP 中设置 i2c 的地址为 0x68，再点击 **OPEN** 按钮打开 i2c



10) 点击 **OPEN** 按钮打开 i2c 后的显示如下所示



- 11) 然后我们测试下往 rtc 模块的寄存器中写入一个值，比如往 0x1c 地址写入 0x55
a. 我们首先设置需要写入的寄存器的地址为 0x1c



- b. 然后设置需要写入的值为 0x55



- c. 然后点击 **WRITE BYTE** 按钮执行写入的动作



12) 然后点击 **READ BYTE** 按钮读取下 0x1c 寄存器的值，如果显示为 0x55，就说明 i2c 读写测试通过



6. 16. 5. 40pin 的 PWM 测试

1) 由下表可知，可用的 pwm 为 pwm1、pwm2、pwm3 和 pwm4。

GPIO序号	GPIO	功能	引脚
		3.3V	1
264	PI8	TWI1-SDA	3
263	PI7	TWI1-SCL	5
269	PI13	PWM3	7
		GND	9
226	PH2	UART5_TX	11
227	PH3	UART5_RX	13
261	PI5	UART2_TX	15
		3.3V	17
231	PH7	SPI1_MOSI	19
232	PH8	SPI1_MISO	21
230	PH6	SPI1_CLK	23

引脚	功能	GPIO	GPIO序号
2	5V		
4	5V		
6	GND		
8	UART0_TX	PH0	224
10	UART0_RX	PH1	225
12		PI1	257
14	GND		
16	PWM4	PI14	270
18		PH4	228
20	GND		
22	UART2_RX	PI6	262
24	SPI1_CS0	PH5	229



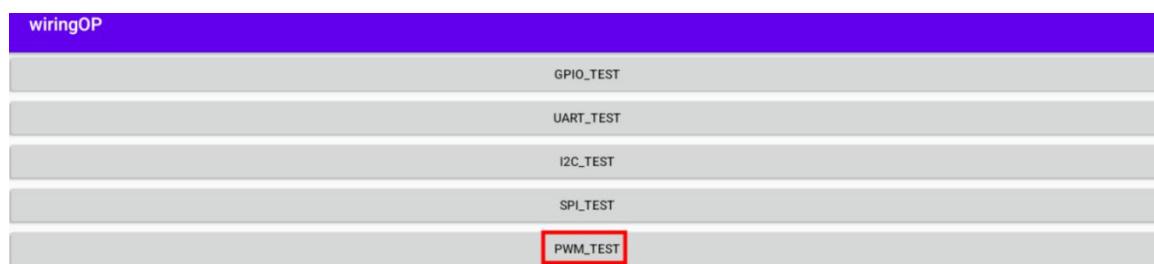
		GND	25
266	PI10	TWI2-SDA	27
256	PI0		29
271	PI15		31
268	PI12	PWM2	33
258	PI2		35
272	PI16		37
		GND	39

26	SPI1_CS1	PH9	233
28	TWI2-SCL	PI9	265
30	GND		
32	PWM1	PI11	267
34	GND		
36		PC12	76
38		PI4	260
40		PI3	259

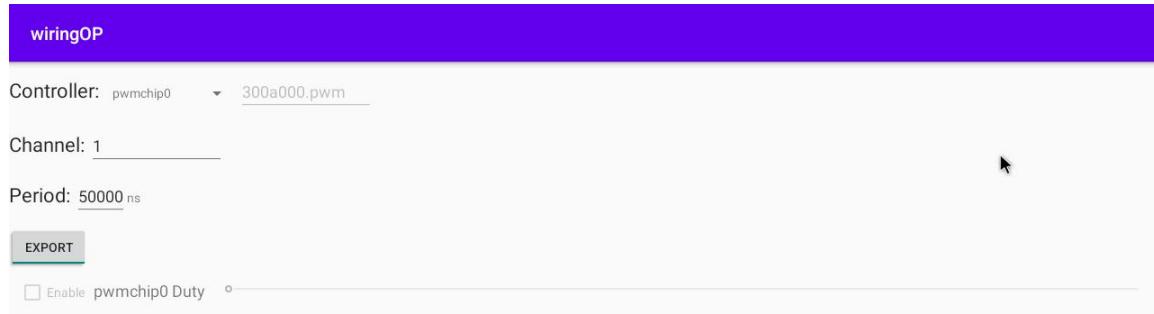
2) 首先点击 wiringOP 图标打开 wiringOP APP



3) 然后在 wiringOP 的主界面点击 **PWM_TEST** 按钮进入 PWM 的测试界面



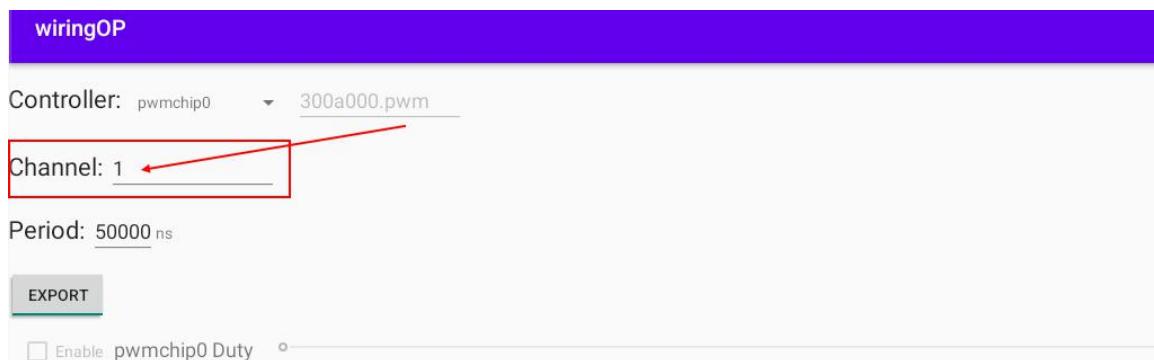
4) PWM 测试界面如下所示



5) 然后在 Channel 中设置下想用哪个 PWM, 默认是 PWM1, 如果想设置为 PWM2,



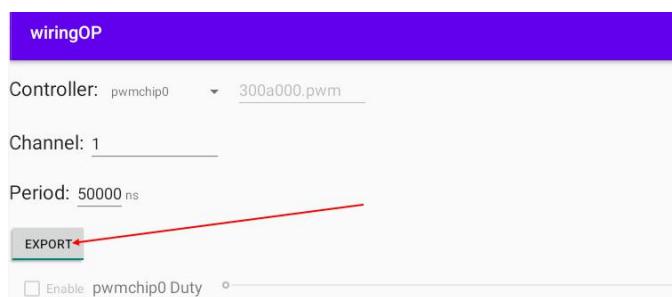
则在 Channel 中输入 2 即可， PWM3 和 PWM4 以此类推



6) 然后可以设置下 PWM 的周期，默认的配置是 **50000ns**，转换为 PWM 频率是 **20KHz**



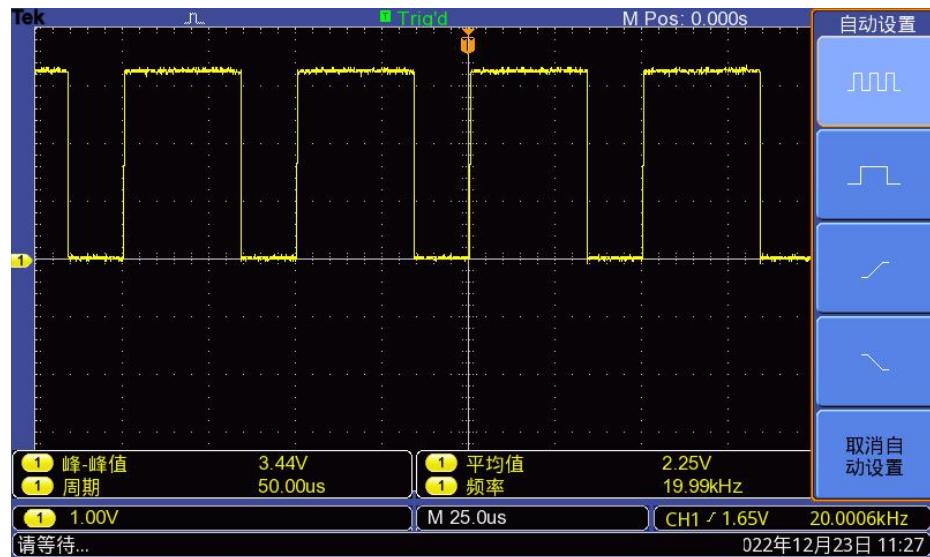
7) 然后点击 **EXPORT** 按钮导出 PWM



8) 然后拖动下面的进度条，就可以改变 PWM 的占空比，然后勾选 **Enable** 就可以输出 PWM 波形了



9) 然后使用示波器测量开发板 40pin 中的对应引脚就可以看到下面的波形了





7. Android 12 源码的编译方法

7.1. 下载 Android 12 的源码

1) 首先从百度或者谷歌网盘下载 Android 12 源码的分卷压缩包和 Orange Pi Zero2w 修改的文件的压缩包

a. 百度网盘

文件名	大小	修改日期
H618_Android12源码	212.8M	2023-07-14 09:18
H618-Android12-Src.tar.gzar	1.86G	2023-07-14 09:18
H618-Android12-Src.tar.gzaq	1.86G	2023-07-14 09:18
H618-Android12-Src.tar.gzip	1.86G	2023-07-14 09:18
H618-Android12-Src.tar.gzo	1.86G	2023-07-14 09:18
H618-Android12-Src.tar.gzan	1.86G	2023-07-14 09:18
H618-Android12-Src.tar.gzam	1.86G	2023-07-14 09:18
H618-Android12-Src.tar.gzal	1.86G	2023-07-14 09:18
H618-Android12-Src.tar.gzak	1.86G	2023-07-14 09:18
H618-Android12-Src.tar.gzaj	1.86G	2023-07-14 09:18
H618-Android12-Src.tar.gzai	1.86G	2023-07-14 09:18
H618-Android12-Src.tar.gzah	1.86G	2023-07-14 09:18
H618-Android12-Src.tar.gzag	1.86G	2023-07-14 09:18
H618-Android12-Src.tar.gzaf	1.86G	2023-07-14 09:18
H618-Android12-Src.tar.gzae	1.86G	2023-07-14 09:18
H618-Android12-Src.tar.gzad	1.86G	2023-07-14 09:18
H618-Android12-Src.tar.gzar	1.86G	2023-07-14 09:18
H618-Android12-Src.tar.gzab	1.86G	2023-07-14 09:18
H618-Android12-Src.tar.gzaa	1.86G	2023-07-14 09:18
H618-Android12-Src.tar.gzsum	1KB	2023-07-14 09:18

b. 谷歌网盘

名称	所有者	上次修改日期	文件大小
H618-Android12-Src.tar.gz.sum	所有者已隐藏	09:20	1KB
H618-Android12-Src.tar.gzaa	所有者已隐藏	09:20	1.86 GB
H618-Android12-Src.tar.gzab	所有者已隐藏	09:20	1.86 GB
H618-Android12-Src.tar.gzac	所有者已隐藏	09:20	1.86 GB
H618-Android12-Src.tar.grad	所有者已隐藏	09:20	1.86 GB
H618-Android12-Src.tar.graa	所有者已隐藏	09:20	1.86 GB
H618-Android12-Src.tar.grab	所有者已隐藏	09:21	1.86 GB
H618-Android12-Src.tar.grag	所有者已隐藏	09:21	1.86 GB
H618-Android12-Src.tar.grah	所有者已隐藏	09:21	1.86 GB
H618-Android12-Src.tar.grai	所有者已隐藏	09:21	1.86 GB
H618-Android12-Src.tar.graj	所有者已隐藏	09:21	1.86 GB
H618-Android12-Src.tar.grak	所有者已隐藏	09:21	1.86 GB
H618-Android12-Src.tar.gral	所有者已隐藏	09:21	1.86 GB
H618-Android12-Src.tar.gram	所有者已隐藏	09:21	1.86 GB
H618-Android12-Src.tar.gran	所有者已隐藏	09:21	1.86 GB
H618-Android12-Src.tar.grao	所有者已隐藏	09:22	1.86 GB
H618-Android12-Src.tar.grap	所有者已隐藏	09:22	1.86 GB
H618-Android12-Src.tar.gzaq	所有者已隐藏	09:22	1.86 GB
H618-Android12-Src.tar.gzar	所有者已隐藏	09:22	212.8 MB

2) Android 12 源码的分卷压缩包下载完后, 请先检查下 MD5 校验和是否正确, 如



果不正确，请重新下载源码。检查 MD5 校验和的方法如下所示：

```
test@test:~$ md5sum -c H618-Android12-Src.tar.gz.md5sum
H618-Android12-Src.tar.gzaa: OK
H618-Android12-Src.tar.gzab: OK
.....
```

3) 然后需要将多个压缩文件合并成一个，再解压出安卓源码。命令如下所示：

```
test@test:~$ cat H618-Android12-Src.tar.gza* > H618-Android12-Src.tar.gz
test@test:~$ tar -xvf H618-Android12-Src.tar.gz
```

4) 然后解压 Orange Pi Zero2w 修改的文件的压缩包

```
test@test:~$ tar zxf opizero2w_android12_patches.tar.gz
test@test:~$ ls
opizero2w_android12_patches  opizero2w_android12_patches.tar.gz
```

5) 然后将 Orange Pi Zero2w 修改的文件复制到安卓源码中

```
test@test:~$ cp -rf opizero2w_android12_patches/*  H618-Android12-Src/
```

7.2. 编译 Android 12 的源码

Android12 的编译是在安装有 **Ubuntu 22.04** 的 **x86_64** 电脑上进行的，其它版本的 Ubuntu 系统包依赖可能会有一些区别，Ubuntu 22.04 **amd64** 版本的镜像下载地址如下所示：

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

编译 Android12 源码的 **x86_64** 电脑硬件配置建议内存为 16GB 或 16GB 以上，硬盘空间建议预留 200GB 或以上，CPU 核心数越多越好。

1) 首先安装编译 Android12 源码需要的软件包

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

**libssl-dev libncurses5 clang gawk**

2) 然后编译 **longan** 文件夹中的代码，里面主要包含 u-boot 和 linux 内核

a. 首先运行 **./build.sh config** 设置编译选项

```
test@test:~$ cd H618-Android12-Src/longan
test@test:~/H618-Android12-Src/longan$ ./build.sh config

Welcome to mkscript setup progress

All available platform:
  0. android
  1. linux
Choice [android]: 0

All available ic:
  0. h618
Choice [h618]: 0

All available board:
  0. ft
  1. p1
  2. p2
  3. p7
  4. p7l
  5. perf1
  6. perf2
  7. perf3
  8. qa
Choice [p2]: 2

All available flash:
  0. default
  1. nor
Choice [default]: 0

All available kern_ver:
  0. linux-5.4
Choice [linux-5.4]: 0

All available arch:
  0. arm
  1. arm64
```



```
Choice [arm64]: 1  
.....  
*** Default configuration is based on 'sun50iw9p1smp_h618_android_defconfig'  
#  
# configuration written to .config  
#  
make[1]: Leaving directory '/home/test/H618-Android12-Src/longan/out/kernel/build'  
make: Leaving directory '/home/test/H618-Android12-Src/longan/kernel/linux-5.4'  
INFO: clean buildserver  
INFO: prepare_buildserver
```

b. 然后运行**./build.sh** 脚本就可以开始编译了

```
test@test:~/H618-Android12-Src/longan$ ./build.sh
```

c. 编译完成后会看到下面的输出

```
sun50iw9p1 compile Kernel successful  
  
INFO: Prepare toolchain ...  
.....  
INFO: build kernel OK.  
INFO: build rootfs ...  
INFO: skip make rootfs for android  
INFO: -----  
INFO: build lichee OK.  
INFO: -----
```

3) 然后使用下面的命令编译安卓源码并生成最终的安卓镜像

```
test@test:~$ cd H618-Android12-Src  
test@test:~/H618-Android12-Src$ source build/envsetup.sh  
test@test:~/H618-Android12-Src$ lunch apollo_p2-userdebug  
test@test:~/H618-Android12-Src$ make -j8  
test@test:~/H618-Android12-Src$ pack
```

4) 编译生成的安卓镜像存放路径为:

```
longan/out/h618_android12_p2_uart0.img
```



8. 附录

8.1. 用户手册更新历史

版本	日期	更新说明
v1.0	2023-09-14	初始版本
v1.1	2023-10-13	wiringOP 硬件 PWM 的使用方法

8.2. 镜像更新历史

日期	更新说明
2023-09-14	orangepizero2w_1.0.0_debian_bullseye_server_linux5.4.125.7z orangepizero2w_1.0.0_ubuntu_focal_server_linux5.4.125.7z orangepizero2w_1.0.0_ubuntu_focal_desktop_xfce_linux5.4.125.7z orangepizero2w_1.0.0_debian_bullseye_desktop_xfce_linux5.4.125.7z orangepizero2w_1.0.0_ubuntu_jammy_server_linux6.1.31.7z orangepizero2w_1.0.0_debian_bookworm_server_linux6.1.31.7z orangepizero2w_1.0.0_debian_bullseye_server_linux6.1.31.7z orangepizero2w_1.0.0_ubuntu_jammy_desktop_xfce_linux6.1.31.7z orangepizero2w_1.0.0_debian_bookworm_desktop_xfce_linux6.1.31.7z orangepizero2w_1.0.0_debian_bullseye_desktop_xfce_linux6.1.31.7z OrangePi_Zero2w_Android12_v1.0.tar.gz Opios-arch-aarch64-xfce-opizero2w-23.09-linux6.1.31.img.xz * 初始版本