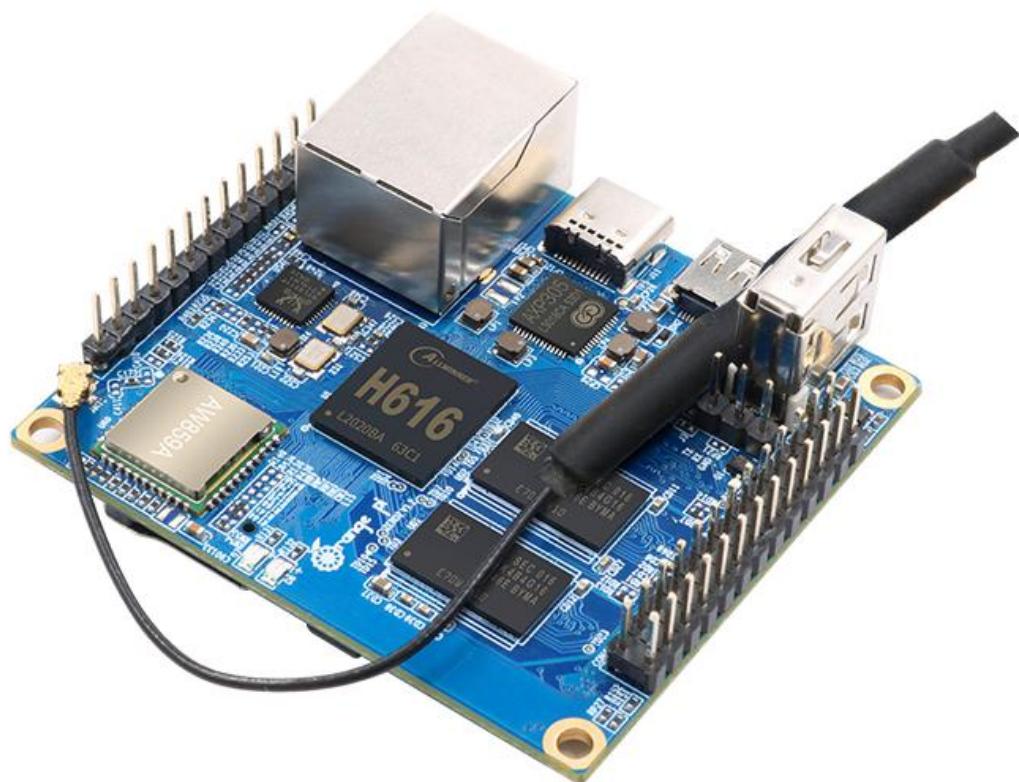




Orange Pi Zero 2

用户手册





目录

1. Orange Pi Zero 2 的基本特性.....	1
1. 1. 什么是 Orange Pi Zero 2.....	1
1. 2. Orange Pi Zero 2 的用途.....	1
1. 3. Orange Pi Zero 2 是为谁设计的.....	1
1. 4. Orange Pi Zero 2 的硬件特性.....	2
1. 5. Orange Pi Zero 2 的顶层视图和底层视图.....	3
1. 6. Orange Pi Zero 2 的接口详情图.....	4
2. 开发板使用介绍.....	6
2. 1. 准备需要的配件.....	6
2. 2. 下载开发板的镜像和相关的资料.....	10
2. 3. 基于 Windows PC 将 Linux 镜像烧写到 TF 卡的方法.....	11
2. 4. 基于 Ubuntu PC 将 Linux 镜像烧写到 TF 卡的方法.....	13
2. 5. 烧写 Android 固件到 TF 卡的方法.....	16
2. 6. 启动香橙派开发板.....	21
2. 7. 调试串口的使用方法.....	22
2. 7. 1. 调试串口的连接说明.....	22
2. 7. 2. Ubuntu 平台调试串口的使用方法.....	23
2. 7. 3. Windows 平台调试串口的使用方法.....	26
2. 8. 使用开发板 26pin 或 13pin 接口中的 5v 引脚供电说明.....	29
3. Linux 系统使用说明.....	30
3. 1. 已支持的 linux 发行版类型和内核版本.....	30
3. 2. linux 内核驱动适配情况.....	30
3. 2. 1. linux4.9 驱动适配情况.....	30
3. 2. 2. linux5.13 驱动适配情况.....	31
3. 3. linux 系统默认登录账号和密码.....	31



3. 4. Linux 桌面版系统自动登录说明.....	32
3. 5. 第一次启动自动扩容 TF 卡中的 rootfs.....	33
3. 6. 修改 linux 日志级别（loglevel）的方法.....	34
3. 7. 以太网口测试.....	35
3. 8. 设置静态 IP 地址的方法.....	36
3. 9. SSH 远程登录开发板.....	43
3. 9. 1. Ubuntu 下 SSH 远程登录开发板.....	43
3. 9. 2. Windows 下 SSH 远程登录开发板.....	44
3. 10. HDMI 显示测试.....	46
3. 11. HDMI 转 VGA 显示测试.....	46
3. 12. HDMI 分辨率设置.....	47
3. 13. Framebuffer 宽度和高度的修改方法.....	49
3. 14. Framebuffer 光标设置.....	52
3. 15. 香橙派 5 寸 TFT 液晶屏测试.....	53
3. 16. WIFI 连接测试.....	54
3. 16. 1. 服务器版镜像通过命令连接 WIFI.....	55
3. 16. 2. 服务器版镜像通过图形化方式连接 WIFI.....	56
3. 16. 3. 桌面板镜像的测试方法.....	59
3. 17. 蓝牙使用方法.....	61
3. 17. 1. 桌面板镜像的测试方法.....	61
3. 17. 2. 服务器版镜像的使用方法.....	65
3. 18. 板载 LED 灯测试说明.....	68
3. 19. USB 接口测试.....	69
3. 19. 1. 连接鼠标或键盘测试.....	69
3. 19. 2. 连接 USB 存储设备测试.....	69
3. 20. USB 无线网卡测试.....	70
3. 21. USB 以太网卡测试.....	72
3. 22. USB 摄像头测试.....	73



3. 23. 虚拟 USB 网卡测试.....	76
3. 24. 虚拟串口测试.....	78
3. 25. 音频测试.....	81
3. 25. 1. 耳机接口播放音频测试.....	81
3. 25. 2. HDMI 音频播放测试.....	83
3. 26. 红外接收测试.....	84
3. 27. 温度传感器.....	86
3. 28. 安装 Docker 的方法.....	86
3. 29. 13 Pin 扩展板接口引脚说明.....	88
3. 30. 26 Pin 接口引脚说明.....	89
3. 31. 安装 wiringOP 的方法.....	90
3. 32. 26pin 接口 GPIO、I2C、UART、SPI 测试.....	91
3. 32. 1. 26pin GPIO 口测试.....	91
3. 32. 2. 26pin SPI 测试.....	92
3. 32. 3. 26pin I2C 测试.....	94
3. 32. 4. 26pin 的 UART 测试.....	95
3. 33. Linux4.9 PWM 的测试方法.....	97
3. 34. SPI LCD 显示屏使用方法.....	102
3. 34. 1. 2.4 寸 SPI LCD 显示屏.....	102
3. 34. 2. 3.2 寸 RPi SPI LCD 显示屏.....	107
3. 34. 3. 3.5 寸 SPI LCD 显示屏.....	111
3. 35. 将内核打印信息输出到 26pin 串口的方法.....	115
3. 36. I2C 接口的 0.96 寸 OLED 模块使用方法.....	116
3. 37. 香橙派 DS1307 RTC 时钟模块使用方法.....	118
3. 38. 硬件看门狗测试.....	123
3. 39. 设置中文环境以及安装中文输入法.....	124
3. 39. 1. Ubuntu 系统的安装方法.....	124
3. 39. 2. Debian 系统的安装方法.....	129



3. 40. 查看 H616 芯片的 chipid.....	137
3. 41. 关机和重启开发板的方法.....	137
3. 42. Linux4.9 系统修改启动阶段显示的图片的方法.....	138
3. 43. Linux 系统启动后 TF 卡存储空间和内存的使用情况.....	138
3. 44. 使用 Kiauh 安装 Klipper 固件上位机的方法.....	140
3. 45. 宝塔 Linux 面板的安装方法.....	163
3. 46. Linux 系统支持的部分编程语言测试.....	169
4. Android10 系统使用说明.....	172
4. 1. 已支持的 Android 版本.....	172
4. 2. Android 10 功能适配情况.....	172
4. 3. 板载 LED 灯显示说明.....	172
4. 4. Android 返回上一级界面的方法.....	173
4. 5. ADB 的使用方法.....	173
4. 5. 1. 打开 USB debugging 选项.....	173
4. 5. 2. 使用网络连接 adb 调试.....	175
4. 5. 3. 使用数据线连接 adb 调试.....	176
4. 6. 香橙派 5 寸 TFT 液晶屏测试.....	176
4. 7. HDMI 4K 显示说明.....	179
4. 8. HDMI 转 VGA 显示测试.....	180
4. 9. WI-FI 的连接方法.....	182
4. 10. WI-FI hotspot 的使用方法.....	184
4. 11. 蓝牙的连接方法.....	186
4. 12. Android 视频客户端软件安装说明.....	189
4. 13. USB 摄像头使用方法.....	190
4. 14. Android 系统 ROOT 说明.....	191
5. Linux SDK 使用说明.....	194
5. 1. 编译系统需求.....	194



5.2. 获取 linux sdk 的源码.....	194
5.2.1. 从 github 下载 orangepi-build.....	194
5.2.2. 下载交叉编译工具链.....	195
5.2.3. orangepi-build 完整目录结构说明.....	197
5.2.4. 从百度云盘下载 orangepi-build.....	198
5.3. 编译 u-boot.....	200
5.4. 编译 linux 内核.....	204
5.5. 编译 rootfs.....	208
5.6. 编译 linux 镜像.....	211
6. Android SDK 使用说明.....	215
6.1. 下载 Android SDK 的源码.....	215
6.2. 搭建 Android 编译环境.....	216
6.3. 编译 android 镜像.....	217
6.3.1. 编译内核.....	217
6.3.2. 编译 Android 源码.....	218



1. Orange Pi Zero 2 的基本特性

1. 1. 什么是 Orange Pi Zero 2

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 10.0、Ubuntu 和 Debian 等操作系统。香橙派开发板（Orange Pi Zero 2）使用全志 H616 系统级芯片，同时拥有 1GB DDR3 内存。

1. 2. Orange Pi Zero 2 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，因为 Orange Pi 开发板可以安装 Debian 和 Ubuntu 这样的 Linux 系统，以及 Android TV 这样的系统，也就意味着我们可以在开发板硬件和软件支持的范围内，来实现各种各样的功能

1. 3. Orange Pi Zero 2 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



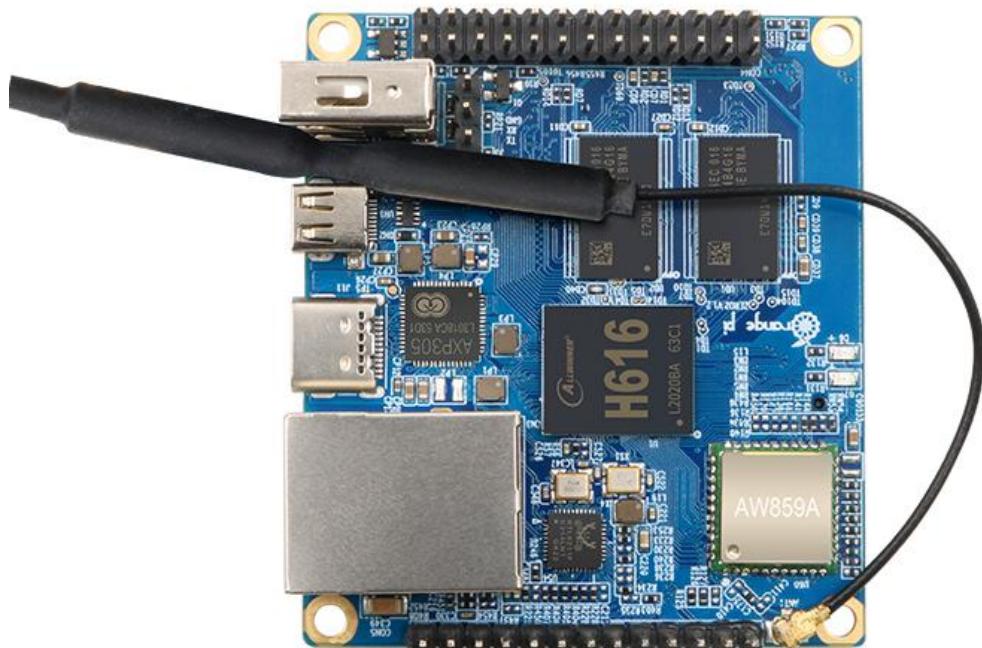
1. 4. Orange Pi Zero 2 的硬件特性

硬件特性介绍	
CPU	全志 H616 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 DDR3 (与 GPU 共享)
板载存储	TF 卡插槽、2MB SPI Flash
以太网	支持 10/100M/1000M 以太网
WIFI+蓝牙	• AW859A 芯片、支持 IEEE 802.11 a/b/g/n/ac、BT5.0
视频输出	• Micro HDMI 2.0a • TV CVBS output, 支持 PAL/NTSC (通过 13pin 扩展板)
音频输出	• Micro HDMI 输出 • 3.5mm 音频口 (通过 13pin 扩展板)
电源	USB Type C 接口输入
USB 2.0 端口	3 个 USB 2.0 HOST (其中两个通过 13pin 扩展板)
26pin 接头	带有 I2Cx1、SPIx1、UARTx1 以及多个 GPIO 口
13pin 接头	带有 USB 2.0 HOSTx2、TV-OUT、LINE OUT、IR-RX、以及 3 个 GPIO 口
调试串口	UART-TX、UART-RX 以及 GND
LED 灯	电源指示灯和状态指示灯
红外接收	支持红外遥控器 (通过 13pin 扩展板)
支持的操作系统	Android10.0、Ubuntu、Debian 等操作系统
外观规格介绍	
产品尺寸	85mm×56mm
重量	30g
range Pi™ 是深圳市迅龙软件有限公司的注册商标	

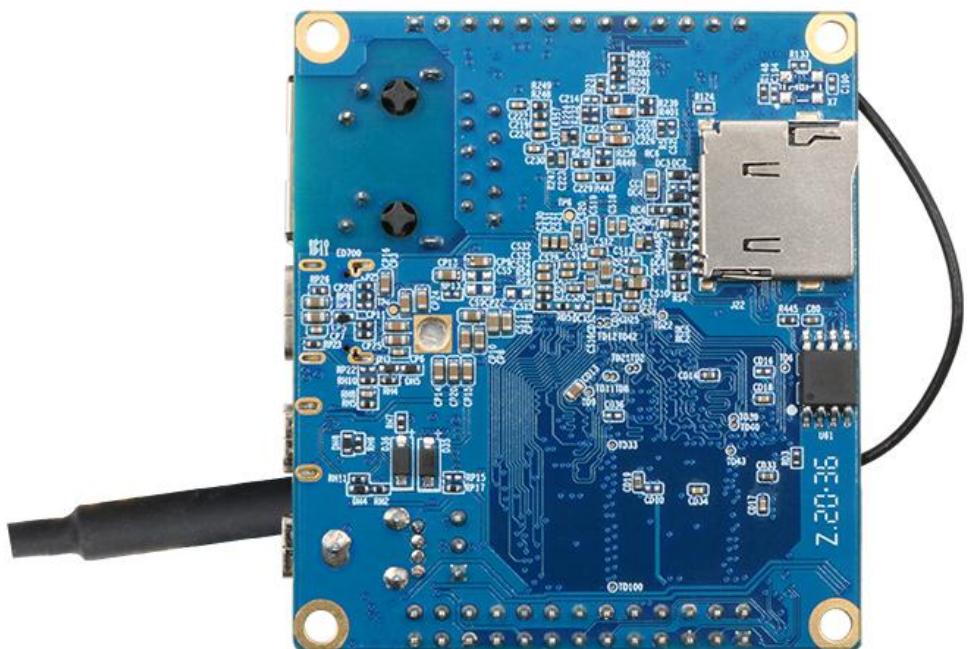


1. 5. Orange Pi Zero 2 的顶层视图和底层视图

顶层视图：



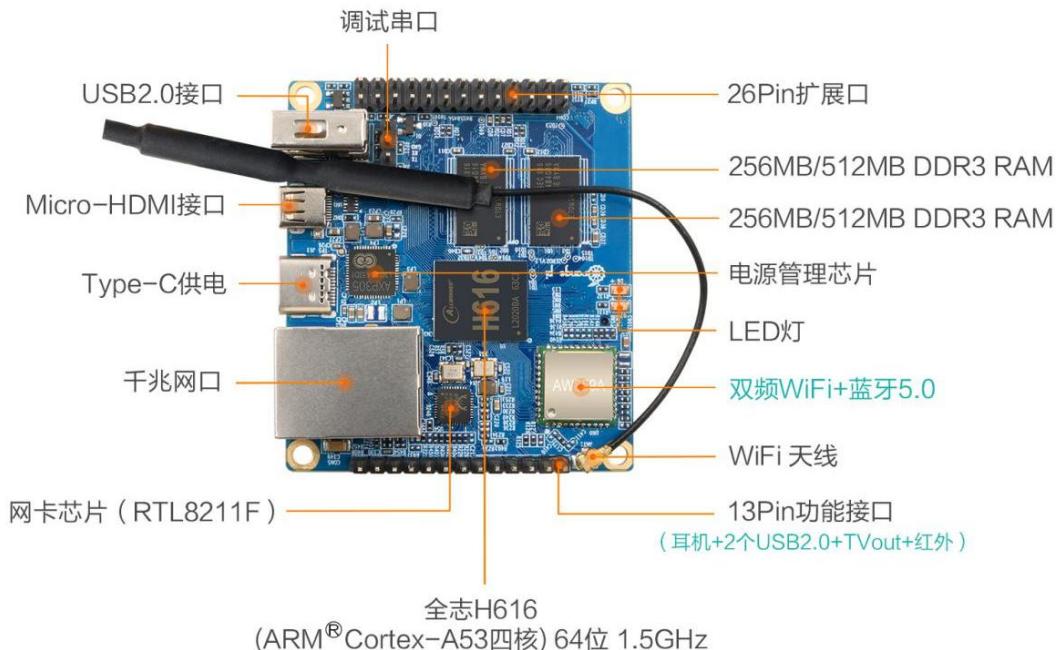
底层视图：



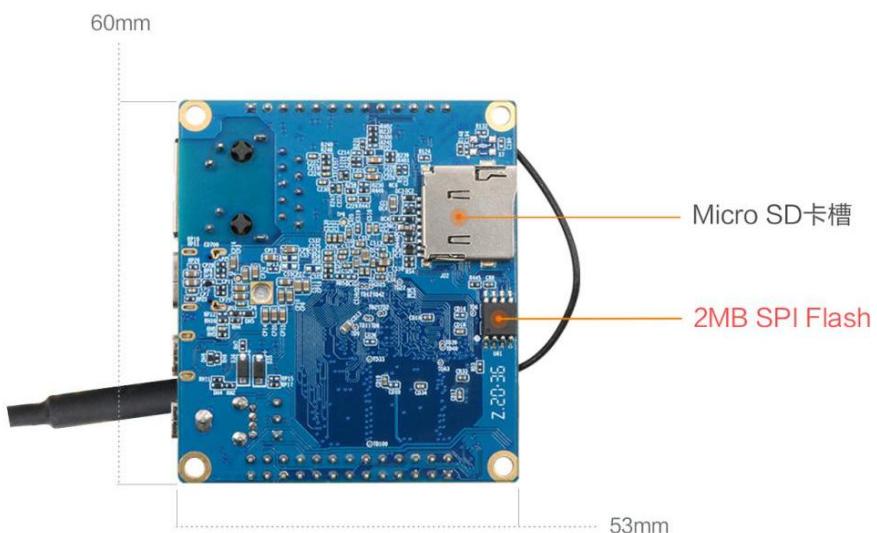


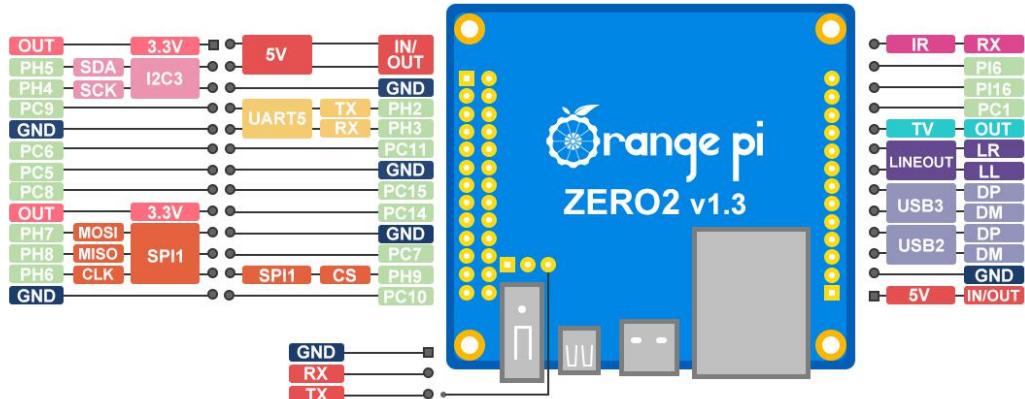
1.6. Orange Pi Zero 2 的接口详情图

正面视图



背面视图





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 2 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 2 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) Micro 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

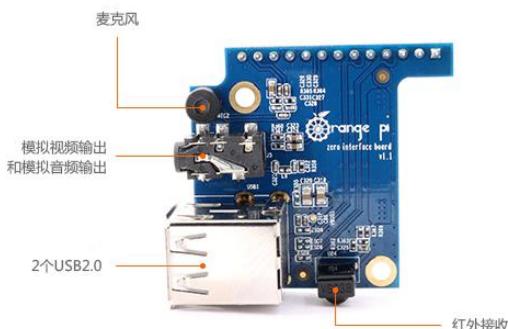


4) 电源，如果有 5V/2A 或 5V/3A 的电源头那就只需要准备一根下面左边图片所示的 USB 转 Type C 接口的数据线，另外也可以使用类似下面右边图片所示的线和电源头一体的 5V/2A 或者 5V/3A 的高品质 USB Typc C 接口电源适配器



5) 13pin 扩展板

a. 扩展板实物如下所示



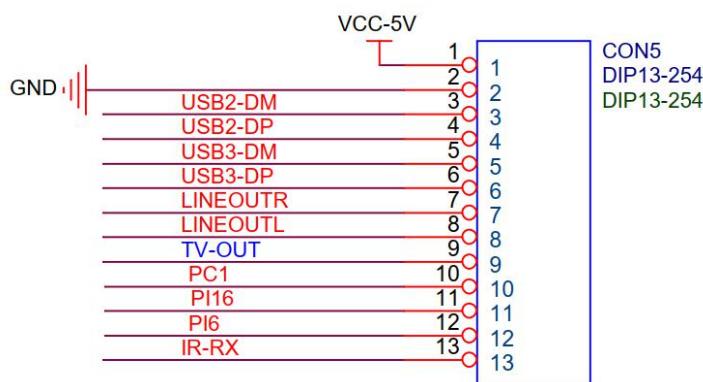
b. 扩展板插入开发板的方式如下所示，切记不要插反了



c. Orange Pi Zero 2 开发板上的 13pin 排针可以接上扩展板来扩展开发板上没有的功能，扩展板可以使用的功能有

1	麦克风（Mic）	不支持，不支持，不支持！！！ 13pin 扩展板是一个通用型号的扩展板，适用于 Orange Pi 多款开发板，但是 Orange Pi Zero2 的 13pin 接口是没有 Mic 功能的，所以 13pin 扩展板上虽然有 Mic，但是在 Orange Pi Zero 2 上是不能用的，13pin 扩展板在 Orange Pi Zero 2 上主要用来扩展除 Mic 以外的其他功能
2	模拟音视频输出接口	支持，可用于接耳机播放音乐，或者通过 AV 线接电视输出模拟音视频信号（仅安卓系统）
3	USB 2.0 Host x 2	支持，用于接 USB 键盘、鼠标以及 USB 存储设备
4	红外接收功能	支持，通过红外遥控可以控制 Android 系统

d. Orange Pi Zero 2 开发板 13pin 排针的原理图如下所示



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



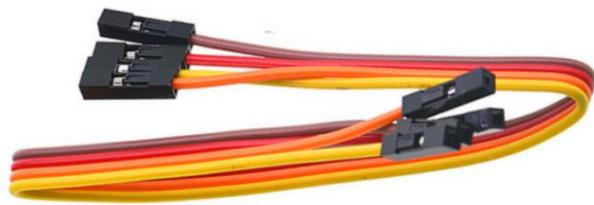
注意，空调的遥控或者电视机的遥控是无法控制Orange Pi开发板的，只有Orange Pi提供的遥控才可以

8) 百兆或者千兆网线，用于将开发板连接到因特网

9) AV 视频线，如果希望通过 AV 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) 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

11) 安装有 Ubuntu 和 Windows 操作系统的个人电脑

1	Ubuntu14.04.6 PC	可选，用于编译 Android 源码
2	Ubuntu18.04 PC	可选，用于编译 Linux 源码
3	Windows PC	用于烧录 Android 和 Linux 镜像

2. 2. 下载开发板的镜像和相关的资料

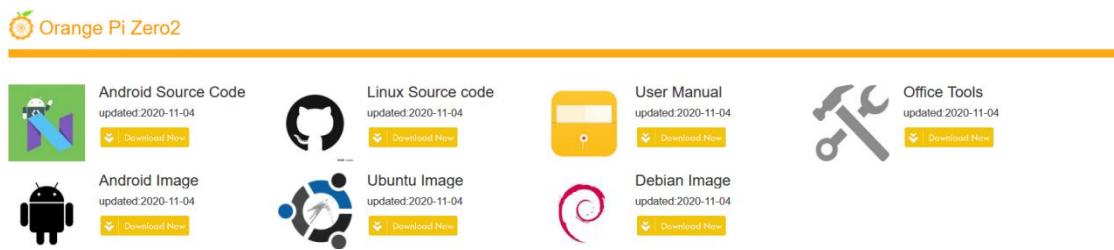
1) 中文版资料的下载网址为

<http://www.orangepi.cn/downloadresourcescn/>



2) 英文版资料的下载网址为

<http://www.orangepi.org/downloadresources/>



3) 资料主要包含

- Android 源码：**保存在百度云盘和谷歌网盘上
- Linux 源码：**保存在 Github 上
- 用户手册和原理图：**芯片相关的数据手册也会放在这里
- 官方工具：**主要包括开发板使用过程中需要用到的软件
- Android 镜像：**保存在百度云盘和谷歌网盘上
- Ubuntu 镜像：**保存在百度云盘和谷歌网盘上



- g. **Debian 镜像**: 保存在百度云盘和谷歌网盘上

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

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** 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

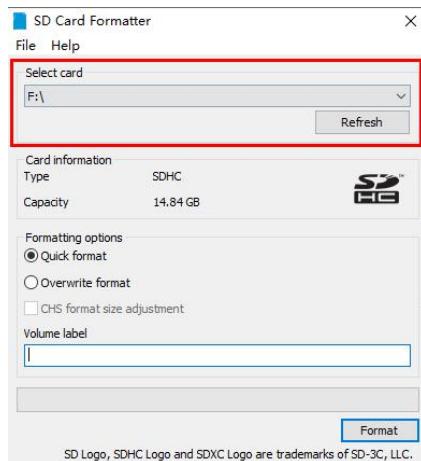
3) 接着格式化 TF 卡

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

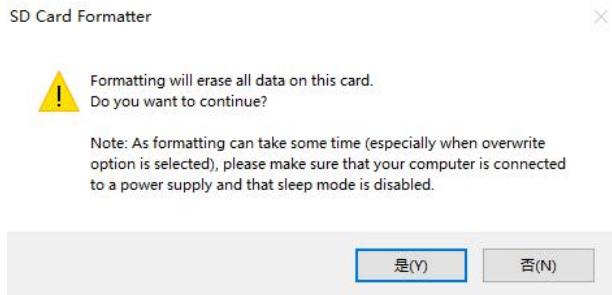
https://www.sdcard.org/downloads/formatter/eula_windows/SDCardFormatterv5_WinEN.zip

- 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

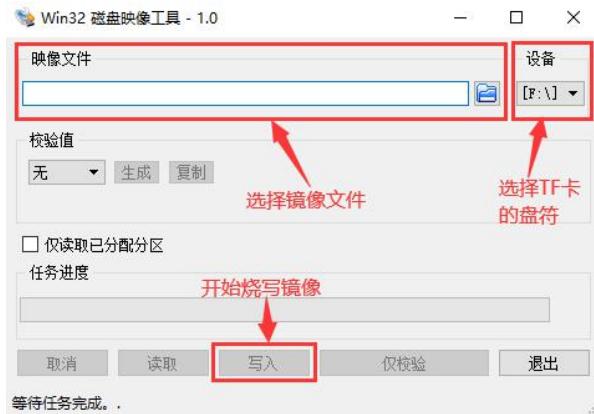
5) 使用 **Win32Diskimager** 烧录 Linux 镜像到 TF 卡

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

- a) 首先选择镜像文件的路径
- b) 然后确认下 TF 卡的盘符和“设备”一栏中显示的一致
- c) 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

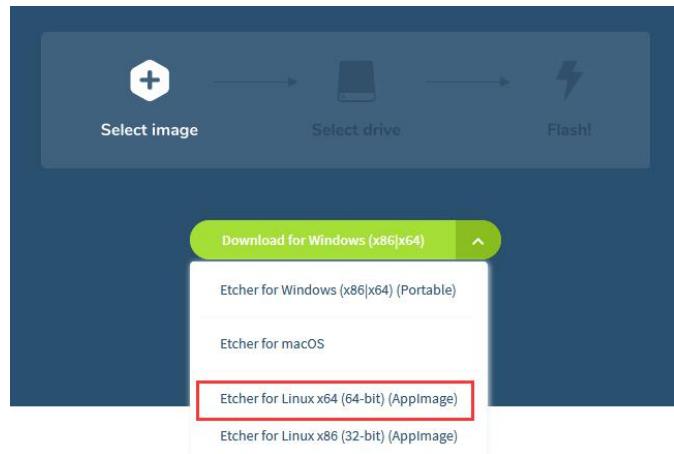
1) 首先准备一张 8GB 或更大容量的 TF 卡，TF 卡的传输速度必须为 **class10** 以上，建议使用闪迪等品牌的 TF 卡

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，请通过下拉框选择 Linux 版本的软件进行下载



- 5) 下载完后使用 **unzip** 进行解压，解压后的 **balenaEtcher-1.5.109-x64.AppImage** 就



是烧录 Linux 镜像需要的软件

```
test@test:~$ unzip balena-etcher-electron-1.5.109-linux-x64.zip
Archive: balena-etcher-electron-1.5.109-linux-x64.zip
  inflating: balenaEtcher-1.5.109-x64.AppImage
test@test:~$ ls
balenaEtcher-1.5.109-x64.AppImage  balena-etcher-electron-1.5.109-linux-x64.zip
```

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

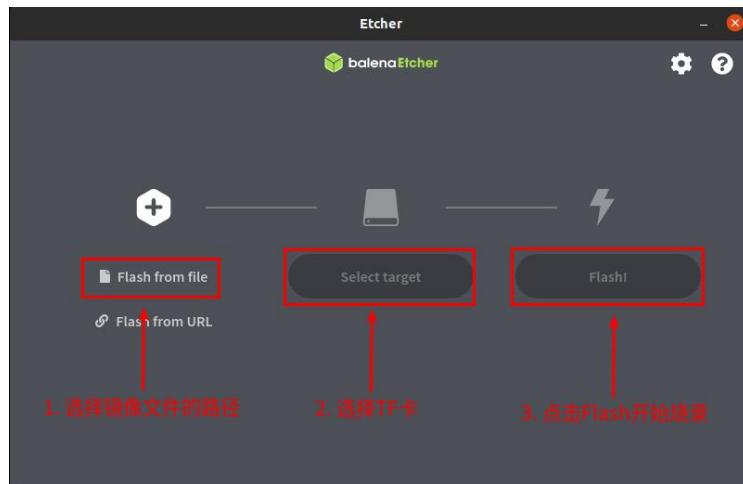
```
test@test:~$ 7z x Orangeepizero2_2.1.8_ubuntu_focal_desktop_linux4.9.170.7z
test@test:~$ ls Orangeepizero2_2.1.8_ubuntu_focal_desktop_linux4.9.170.*
Orangeepizero2_2.1.8_ubuntu_focal_desktop_linux4.9.170.7z
Orangeepizero2_2.1.8_ubuntu_focal_desktop_linux4.9.170.sha  #校验和文件
Orangeepizero2_2.1.8_ubuntu_focal_desktop_linux4.9.170.img  #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
orangeepizero2_2.1.8_ubuntu_bionic_server_linux4.9.170.img: 成功
```

8) 然后在 Ubuntu PC 的图形界面双击 **balenaEtcher-1.5.109-x64.AppImage** 即可打开 **balenaEtcher**（无需安装），打开后的界面如下图所示

- a. 首先选择 linux 镜像文件的路径
- b. 然后选择 TF 卡的设备号
- c. 最后点击 Flash 开始烧录



9) 烧录过程会提示写入的速度和剩余时间



10) 烧录完后会显示下面的界面,此时就可以把TF卡从电脑中拔出来插到开发板中启动了

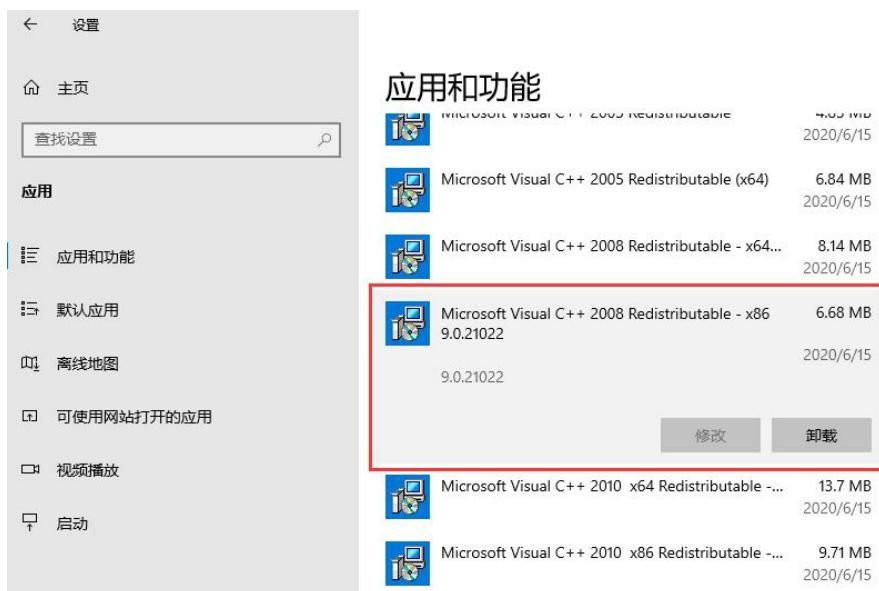




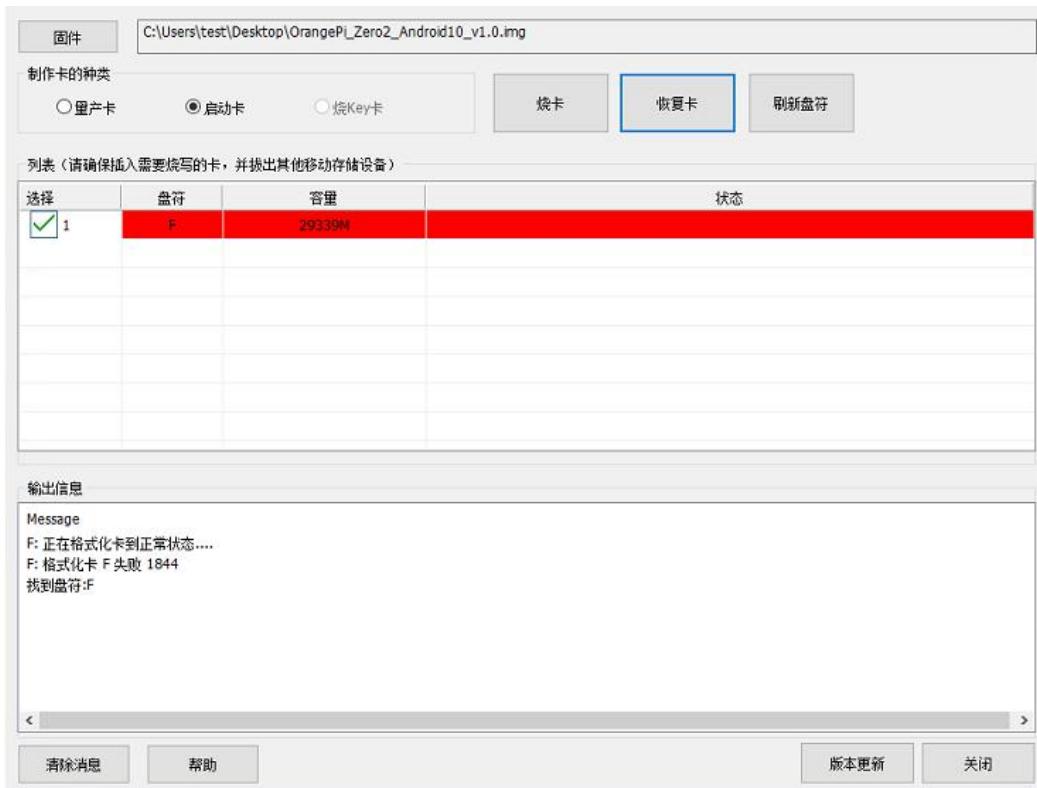
2. 5. 烧写 Android 固件到 TF 卡的方法

开发板的 Android 固件只能在 Windows 平台下使用 PhoenixCard 软件烧录到 TF 卡中，另外 PhoenixCard 这款软件没有 Linux 平台的版本，所以在 Linux 平台上是无法烧录安卓系统到 TF 卡中的

- 1) 首先请确保 Windows 系统已经安装了 **Microsoft Visual C++ 2008 Redistributable - 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 2 的[官方工具](#)中下载到，也可以去[微软官网](#)下载

<input type="checkbox"/> 文件名	大小
<input type="checkbox"/> Balena-etcher	-
<input type="checkbox"/> Android测试APP	-
<input type="checkbox"/> win32diskimager-1.0.0-install.exe	12M
<input type="checkbox"/> vcredist_x86.exe	4.3M
<input type="checkbox"/> SDCardFormaterv5_WinEN.zip	6M

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** 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 10 系统和 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 PhoenixCard 软件来烧录 Orange Pi Zero 2 的 Android 10 镜像，低于这个版本的 PhoenixCard 工具烧写的 Android 10 系统可能会有问题

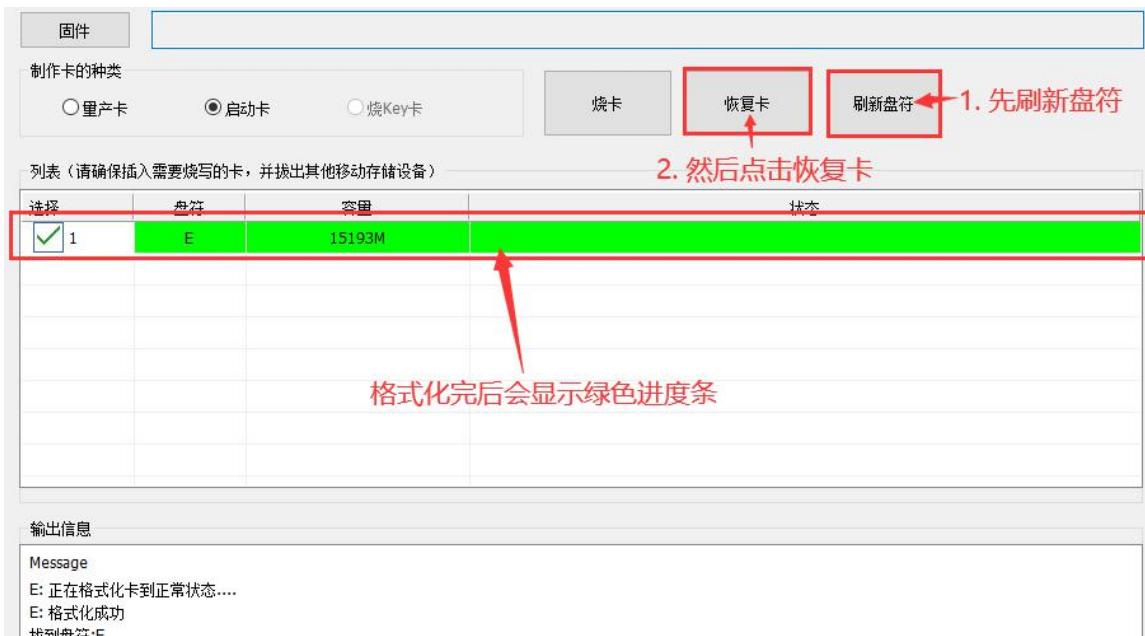
- 7) 使用解压软件解压下载的 Android 系统的压缩包，解压后的文件中，以 “.img” 结尾的文件就是 Android 镜像文件
- 8) 使用解压软件解压 PhoenixCard4.2.8.zip，此软件无需安装，在解压后的文件夹中找到 PhoenixCard 打开即可

文件名	修改日期	类型	大小
ludosocket.dll	2019/4/22 11:53	应用程序扩展	4 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
PhoenixCardManager.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

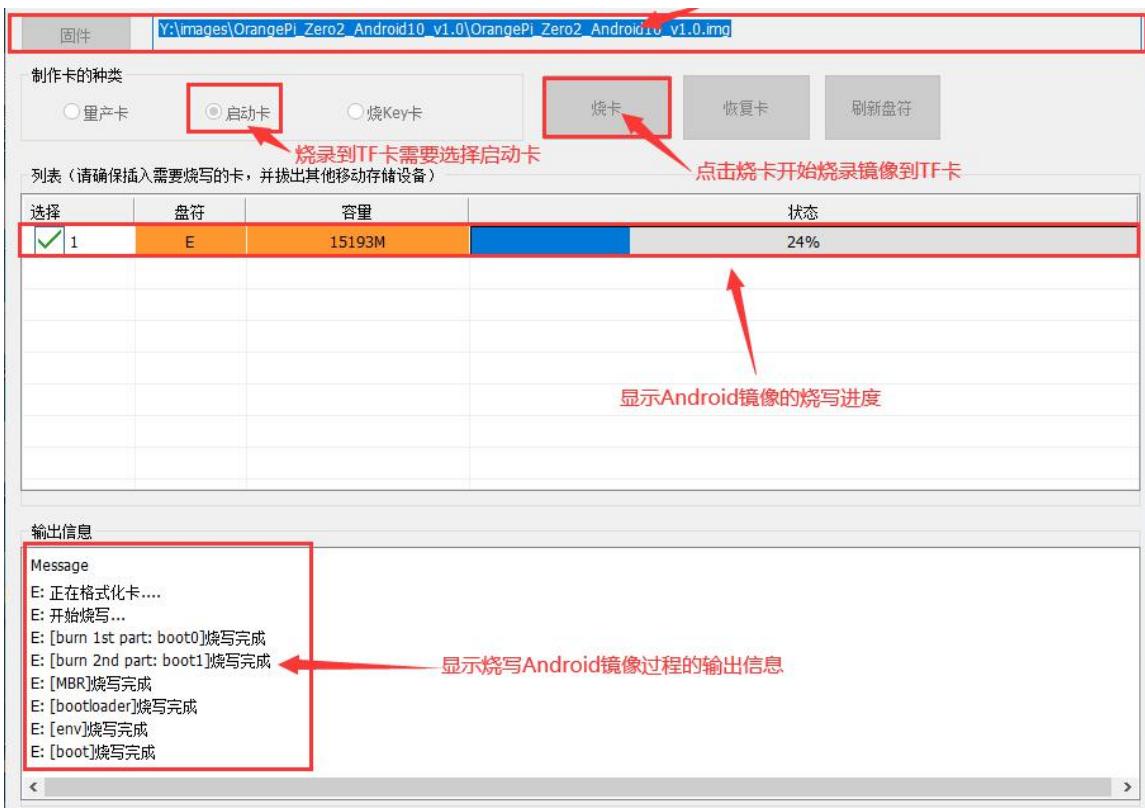


- 10) 确认完盘符后，先格式化 TF 卡，点击 PhoenixCard 中“恢复卡”按钮即可（如果“恢复卡”按钮为灰色的无法按下，可以先点击下“刷新盘符”按钮）

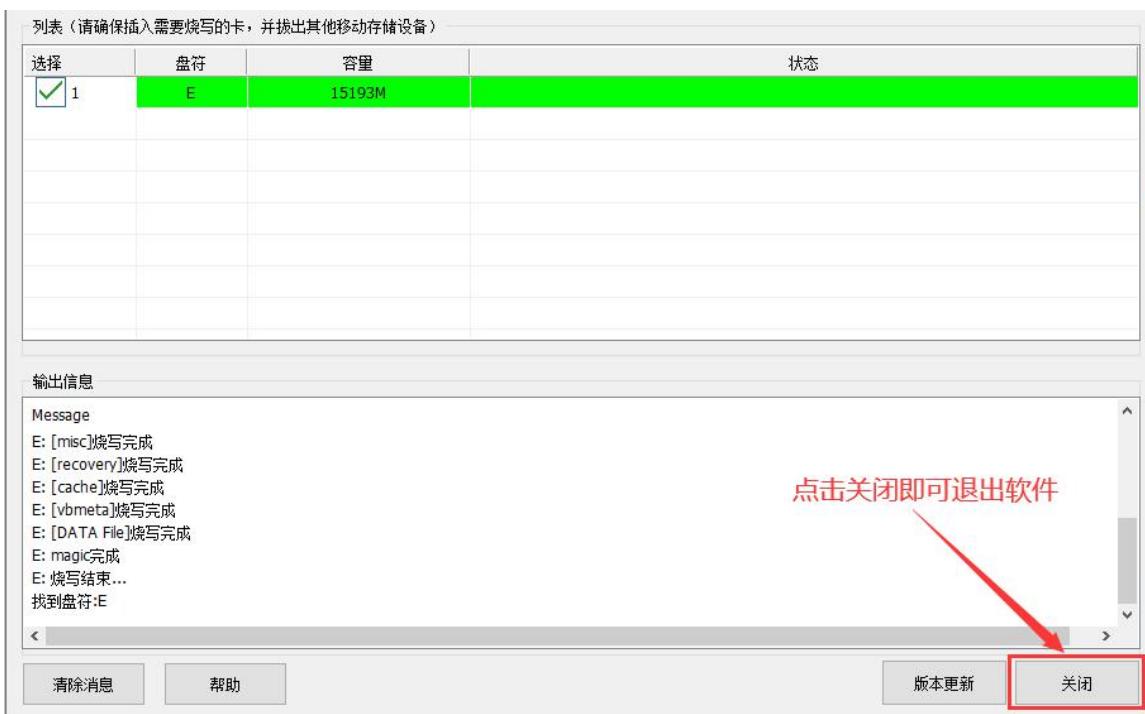


如果格式化有问题, 请尝试拔插 TF 卡后再测试, 如果重新拔插 TF 卡后还是有问题, 请换一台电脑再试下

- 11) 然后开始将 Android 系统写入 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





2.6. 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 Micro HDMI 接口，可以通过 Micro 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 13pin 的扩展板，可以将 13pin 的扩展板插到开发板的 13pin 接口中
- 4) 接上 USB 鼠标和键盘，用于控制香橙派开发板
- 5) 开发板有以太网口，可以插入网线用来上网
- 6) 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
 - a. 切记不要插入 12V 的电源适配器，如果插入了 12V 的电源适配器，会烧坏开发板
 - b. 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和 USB Type C 数据线很重要
- 7) 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
- 8) 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

首先需要注意的是，如果没有插入已经烧录好系统的 TF 卡就上电，开发板是不会有任何灯亮的，包括开发板上的两个 LED 灯和网口灯，所以请不要通过这种方法来判断开发板的好坏。

判断系统是否已经正常启动的方法：

- 1) 如果接了 HDMI 显示器，那么判断的方法就很简单，只要 HDMI 显示器正常显示了系统的界面，就说明系统已经正常启动了；
- 2) 如果没有 HDMI 显示器，可以将开发板通过串口线连接到电脑，通过调试串口输出的 log 信息来查看系统的启动情况。如果串口输出停在了终端的登录界面，就说明系统已经正常启动了；
- 3) 如果 HDMI 显示器和串口线都没有，可以通过开发板上面的两个 LED 灯来判断系统的启动情况，如果绿灯的 LED 灯亮了，这时系统一般已经正常启动了，如果上电等待一段时间后，还是只有红色的 LED 灯亮了，或者红色和绿色的 LED 灯都没亮，说明系统没有正常启动。

如果系统启动失败或者无法正常进入登录界面，请首先检测下面的东西：

- 1) 检查下载的镜像是否有损坏，可以通过计算镜像附带的校验和来判断；



- 2) 镜像烧录镜像到 TF 卡的过程是否有问题，可以重新烧录镜像再测试一遍；
- 3) 确定电源适配器和电源线没有问题，可以尝试换一个试下；
- 4) 确定 TF 卡符合 Orange Pi 开发板的要求，如果有多余的 TF 卡，可以尝试换个 TF 卡然后烧录镜像再测试一遍；
- 5) 如果以上都确定没问题后，请保存下系统启动过程中调试串口的输出 log（最好是以 txt 文本的形式，而不是拍照），然后向客服反馈问题。

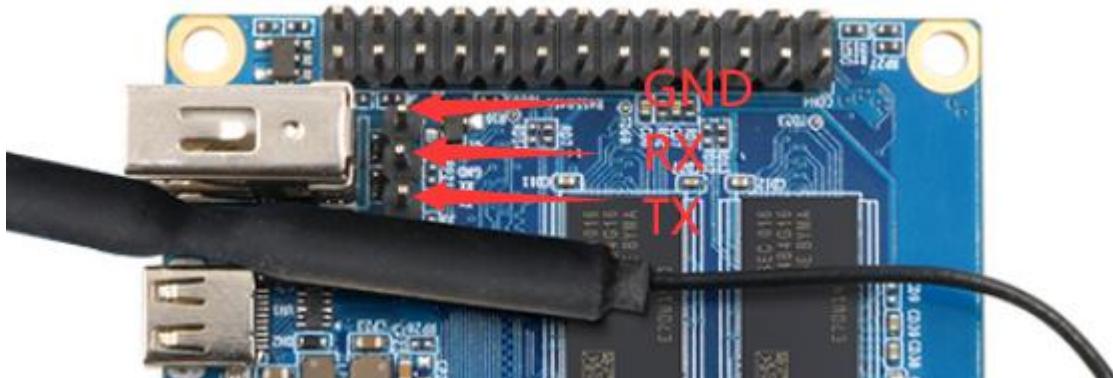
2.7. 调试串口的使用方法

2.7.1. 调试串口的连接说明

- 1) 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



- 2) 开发板的调试串口 GND、TX 和 RX 引脚的对应关系如下图所示

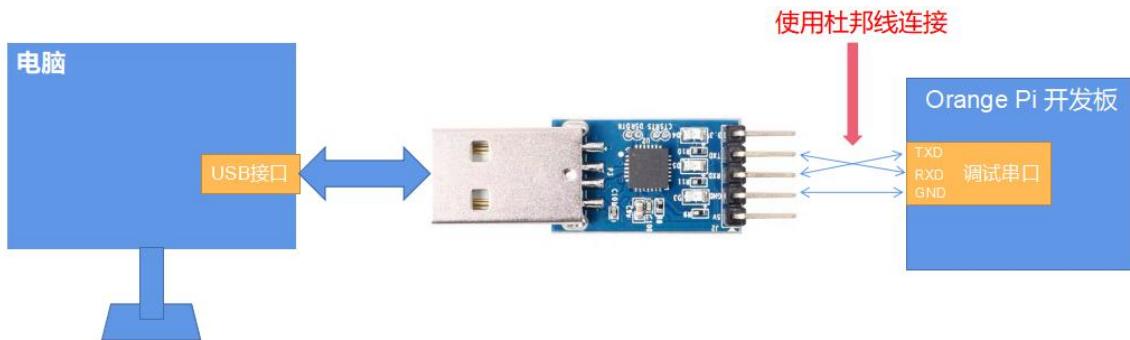




3) USB 转 TTL 模块 GND、TX 和 RX 引脚需要通过杜邦线连接到开发板的调试串口上

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

4) 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

2.7.2. Ubuntu 平台调试串口的使用方法

1) 如果 USB 转 TTL 模块连接正常，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/ttys000
```

2) linux 下可以使用的串口调试软件有很多，如 putty、minicom 等，下面演示 putty 的使用方法

3) 首先在 Ubuntu PC 上安装 putty

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

4) 然后运行 putty，记得加 sudo 权限

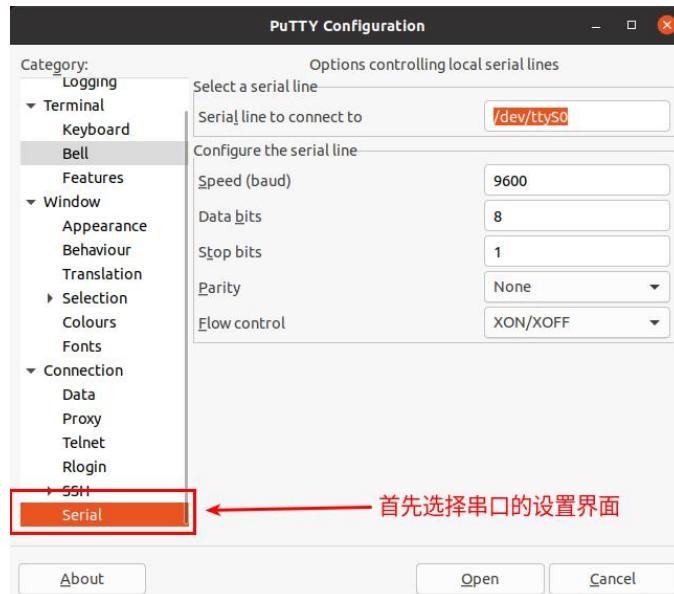


```
test@test:~$ sudo putty
```

5) 执行 putty 命令后会弹出下面的界面



6) 首先选择串口的设置界面

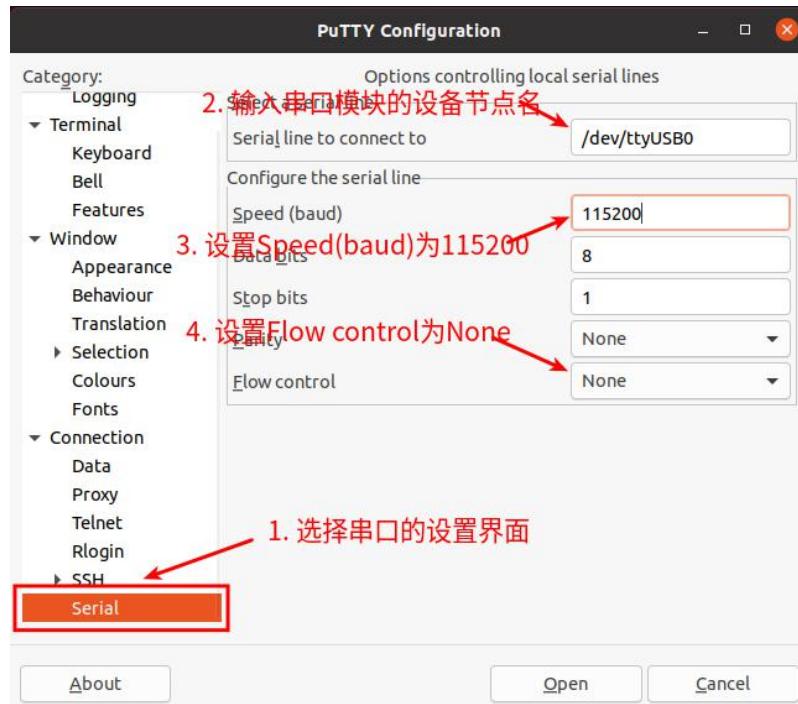


7) 然后设置串口的参数

- a. 设置 **Serial line to connect to** 为 **/dev/ttyUSB0** (修改为对应的节点名, 一般为 **/dev/ttyUSB0**)
- b. 设置 **Speed(baud)** 为 **115200** (串口的波特率)

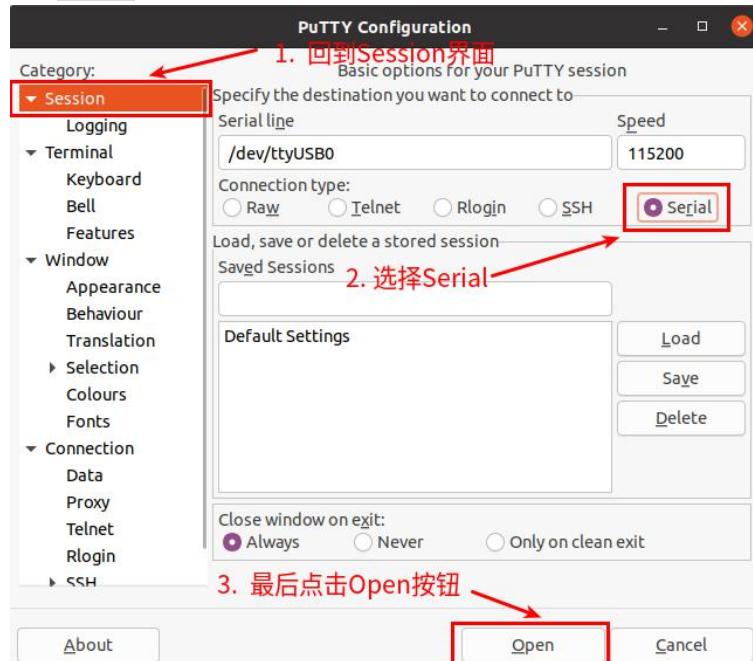


c. 设置 Flow control 为 None



8) 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



9) 启动开发板后，就能从打开的串口终端中看到系统输出的 Log 信息了

2.7.3. Windows 平台调试串口的使用方法

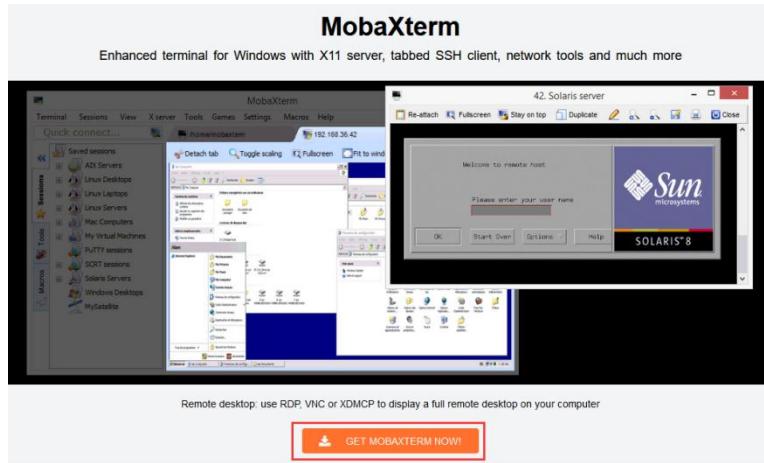
Windows 下可以使用的串口调试软件有很多，如 SecureCRT、MobaXterm 等，下面演示 MobaXterm 的使用方法，这款软件有免费版本，无需购买序列号即可使用。

1) 下载 MobaXterm

- a. 下载 MobaXterm 网址如下

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

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



- c. 然后选择下载 Home 版本



The chart compares the features of the Home Edition (Free) and Professional Edition. The Home Edition offers basic remote desktop, SSH, and X11-forwarding support. The Professional Edition adds advanced features like unlimited sessions, tunnels, and security enhancements.

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 download page for MobaXterm Home Edition. It offers two options: 'MobaXterm Home Edition v20.3 (Portable edition)' (blue button) and 'MobaXterm Home Edition v20.3 (Installer edition)' (green button). Below these are links for previous versions and a preview version. A note at the bottom encourages users to subscribe for professional support.

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

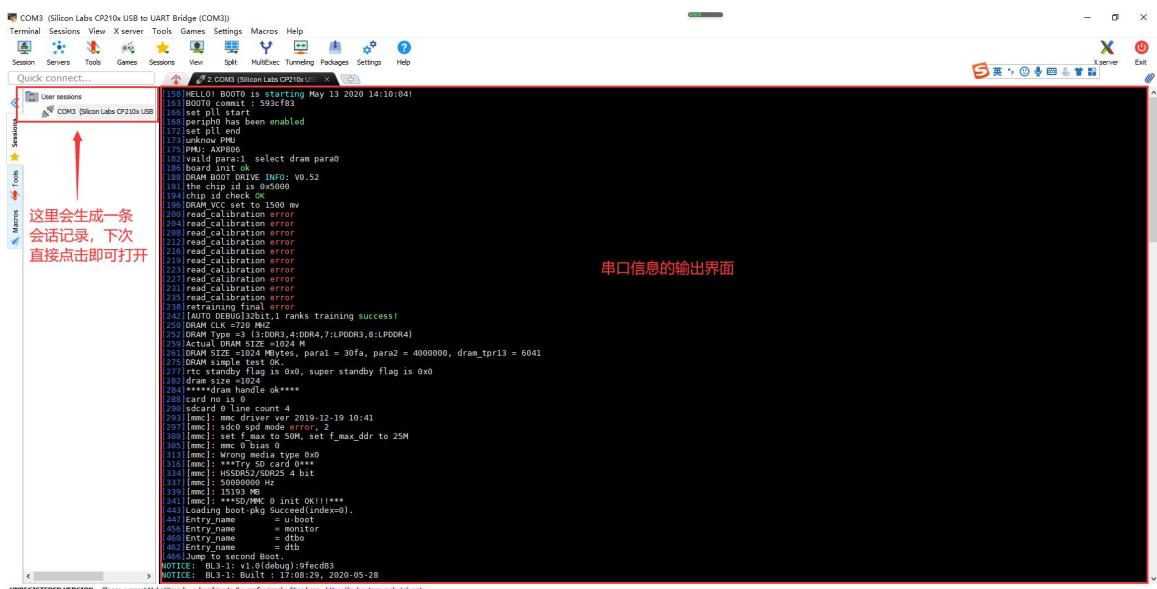
名称	修改日期	类型	大小
CygUtils.plugin	2020/5/21 4:06	PLUGIN 文件	15,570 KB
MobaXterm_Personal_20.3	2020/6/5 4:30	应用程序	14,104 KB

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





2.8. 使用开发板 26pin 或 13pin 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 26pin 或者 13pin 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

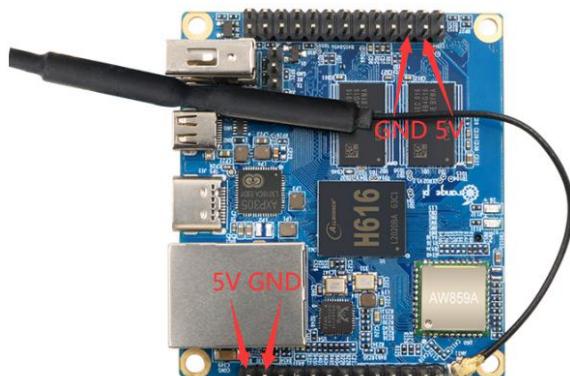
1) 首先需要准备一根下图所示的电源线



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 26pin 或者 13pin 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（不建议插到电脑的 USB 接口来供电，如果开发板接的外设过多，使用会不稳定）
- b. 红色的杜邦线需要插到开发板 26pin 或者 13pin 接口的 5V 引脚上
- c. 黑色的杜邦线需要插到 26pin 或者 13pin 接口的 GND 引脚上
- d. 26pin 和 13pin 接口 5V 引脚和 GND 引脚在开发板中的位置下图所示，**切记不要接反了**





3. Linux 系统使用说明

3.1. 已支持的 linux 发行版类型和内核版本

发行版类型	内核版本	服务器版	桌面版
Ubuntu 18.04	linux4.9	支持	支持
Ubuntu 20.04	linux4.9	支持	支持
Debian 10	linux4.9	支持	支持
Ubuntu 20.04	linux5.13	支持大部分功能，提供测试版本的镜像，目前不建议用在生产环境中	
Ubuntu 18.04	linux5.13		
Debian 10	linux5.13		

3.2. linux 内核驱动适配情况

3.2.1. linux4.9 驱动适配情况

功能	状态
HDMI 视频	OK
HDMI 音频	OK
USB2.0 x 3	OK
TF 卡启动	OK
网卡	OK
红外接收	OK
WIFI	OK
蓝牙	OK
耳机音频	OK
USB 摄像头	OK
LED 灯	OK
26pin GPIO	OK
I2C3	OK
SPI1	OK
UART5	OK
温度传感器	OK



硬件看门狗	OK
TV-OUT	NO
Mali GPU	NO
视频编解码	NO

3. 2. 2. linux5.13 驱动适配情况

Linux5.13 系统目前还在开发中，由于没有大规模测试，暂时不能保证系统的稳定性，所以不建议在生产环境中使用，目前主要提供给开发者测试开发使用

功能	状态
HDMI 视频	OK
HDMI 音频	OK
USB2.0 x 3	OK
TF 卡启动	OK
网卡	OK
红外接收	OK
WIFI	OK
蓝牙	OK
耳机音频	OK
USB 摄像头	OK
LED 灯	OK
26pin GPIO	OK
I2C3	OK
SPI1	OK
UART5	OK
温度传感器	OK
硬件看门狗	OK
TV-OUT	NO
Mali GPU	NO
视频编解码	NO

3. 3. linux 系统默认登录账号和密码

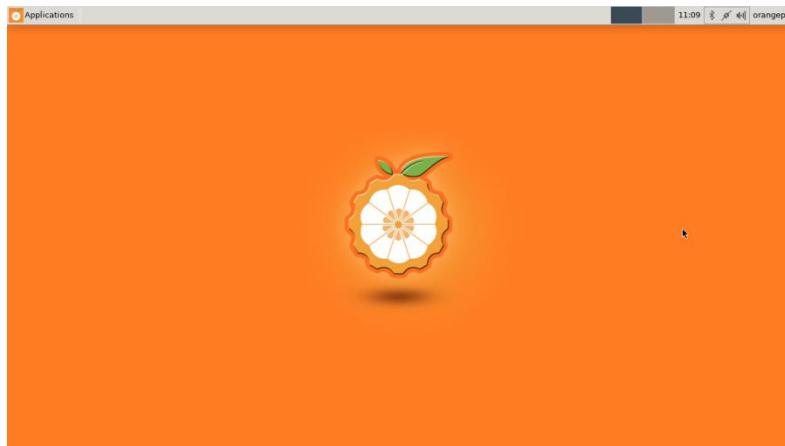
账号	密码
----	----



root	orangeipi
orangeipi	orangeipi

3. 4. Linux 桌面板系统自动登录说明

1) 桌面板系统默认启动后会自动登录进入桌面，无需输入密码



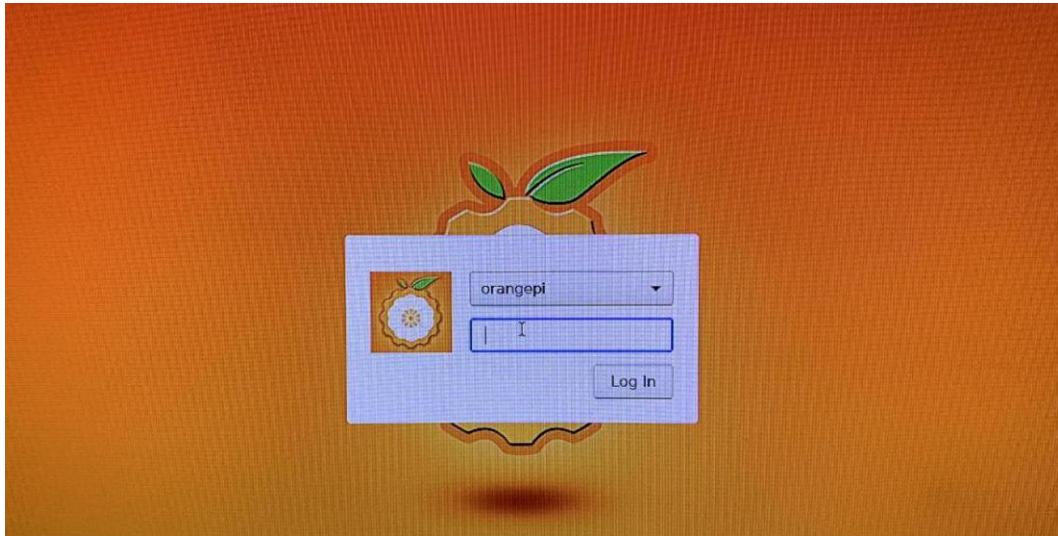
2) 修改/etc/lightdm/lightdm.conf.d/22-orangeipi-autologin.conf 中的配置可以禁止桌面版系统自动登录桌面，修改命令如下，也可以打开配置文件直接修改

```
root@orangeipi:~# sed -i "s/autologin-user=orangeipi/#autologin-user=orangeipi/"  
/etc/lightdm/lightdm.conf.d/22-orangeipi-autologin.conf
```

3) 修改完后/etc/lightdm/lightdm.conf.d/22-orangeipi-autologin.conf 的配置如下所示

```
root@orangeipi:~# cat /etc/lightdm/lightdm.conf.d/22-orangeipi-autologin.conf  
[Seat: *]  
#autologin-user=orangeipi  
autologin-user-timeout=0  
user-session=xfce
```

4) 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



3.5. 第一次启动自动扩容 TF 卡中的 rootfs

- 1) 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 的扩容，所以无需再手动扩容
- 2) 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

```
root@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
```

- 3) 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，也就不存在 BOOT 分区扩容的问题
- 4) 另外如果不需自动扩容 rootfs，可以使用下面的方法来禁止
 - a. 首先将 linux 镜像烧录到 TF 卡中
 - b. 然后将 TF 卡插入 Ubuntu PC 中（Windows 不行），Ubuntu PC 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，



TF 卡的分区名和下面命令所示名字不一定相同，请根据实际情况进行修改

```
test@test:~$ ls /media/test/27e62f92-8250-4ef1-83db-3d8f0c2e23db/
bin  boot  dev  etc  home  lib  lost+found  media  mnt  opt  proc  root  run
sbin  selinux  srv  sys  tmp  usr  var
```

c. 然后在 Ubuntu PC 中将当前用户切换成 root 用户

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

d. 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/27e62f92-8250-4ef1-83db-3d8f0c2e23db
root@test:/media/test/27e62f92-8250-4ef1-83db-3d8f0c2e23db# cd root
root@test:/media/test/27e62f92-8250-4ef1-83db-3d8f0c2e23db/root# touch .no_rootfs_resize
root@test:/media/test/27e62f92-8250-4ef1-83db-3d8f0c2e23db/root# ls .no_rootfs*
.no_rootfs_resize
```

e. 然后就可以卸载 TF 卡，再拔出 TF 插到开发板启动，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。

f. 禁止 rootfs 自动扩容后可以看到 TF 卡可用容量只有 200M 左右

```
root@orangepi:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            927M    0   927M   0% /dev
tmpfs           200M  5.6M  194M   3% /run
/dev/mmcblk0p1  1.5G  1.3G  196M  87% /
tmpfs           997M    0   997M   0% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           997M    0   997M   0% /sys/fs/cgroup
tmpfs           997M  4.0K  997M   1% /tmp
/dev/zram0       49M  1.5M   44M   4% /var/log
tmpfs           200M    0   200M   0% /run/user/0
```

3. 6. 修改 linux 日志级别（**loglevel**）的方法

1) linux 系统的 loglevel 默认设置为 1，当使用串口查看启动信息时，内核输出 log 如下所示，基本全部屏蔽了

```
Starting kernel ...
```



Uncompressing Linux... done, booting the kernel.

Orange Pi 2.1.8 Focal ttyS0

orangeipi login:

- 2) 当 linux 系统启动出现问题时，可以使用下面的方法来修改 loglevel 的值，从而打印更多的 log 信息到串口显示，方便调试。如果 linux 系统启动失败，无法进入系统，可以把 TF 卡通过读卡器插入 Ubuntu PC 中，然后在 Ubuntu PC 中挂载 TF 卡后直接修改 TF 卡中的 linux 系统的配置，修改完后，再把 TF 卡插入开发板中启动

```
root@orangeipi:~# sed -i "s/verbosity=1/verbosity=7/" /boot/orangeipiEnv.txt  
root@orangeipi:~# sed -i "s/console=both/console=serial/" /boot/orangeipiEnv.txt
```

- 3) 上面的命令其实都是设置 **/boot/orangeipiEnv.txt** 中的变量，设置完后可以打开 **/boot/orangeipiEnv.txt** 检查下

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

- 4) 然后重启开发板，内核的输出信息就都会打印到串口输出了

3. 7. 以太网口测试

- 1) 首先将网线插入开发板的以太网接口，并确保网络是畅通的

- 2) 系统启动后会通过 DHCP 自动给以太网卡分配 IP 地址

- 3) 查看 IP 地址的命令如下

```
root@orangeipi:~# ifconfig eth0  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500  
          inet 192.168.1.47  netmask 255.255.255.0  broadcast 192.168.1.255  
            inet6 fe80::e56:c34d:62f0:8d6e  prefixlen 64  scopeid 0x20<link>  
              ether 02:81:3e:a8:58:d8  txqueuelen 1000  (Ethernet)
```



```
RX packets 2165 bytes 177198 (177.1 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 312 bytes 40435 (40.4 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
device interrupt 39
```

4) 测试网络连通性的命令如下

```
root@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
```

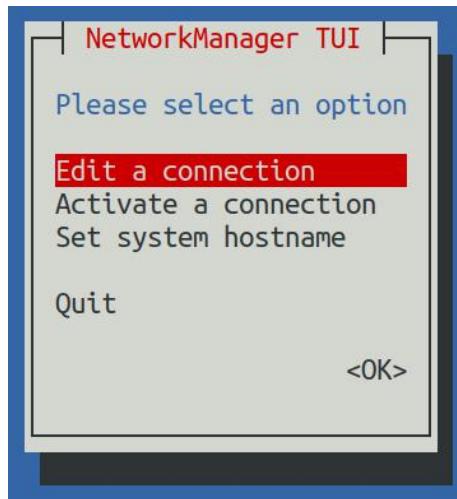
3.8. 设置静态 IP 地址的方法

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

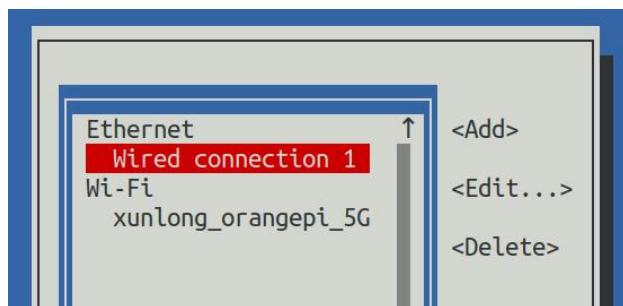
1) 首先运行 mutui 命令

```
root@orangepi:~# nmtui
```

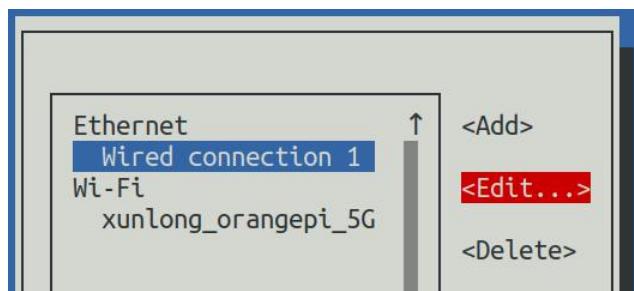
2) 然后选择 **Edit a connection** 并按下回车键



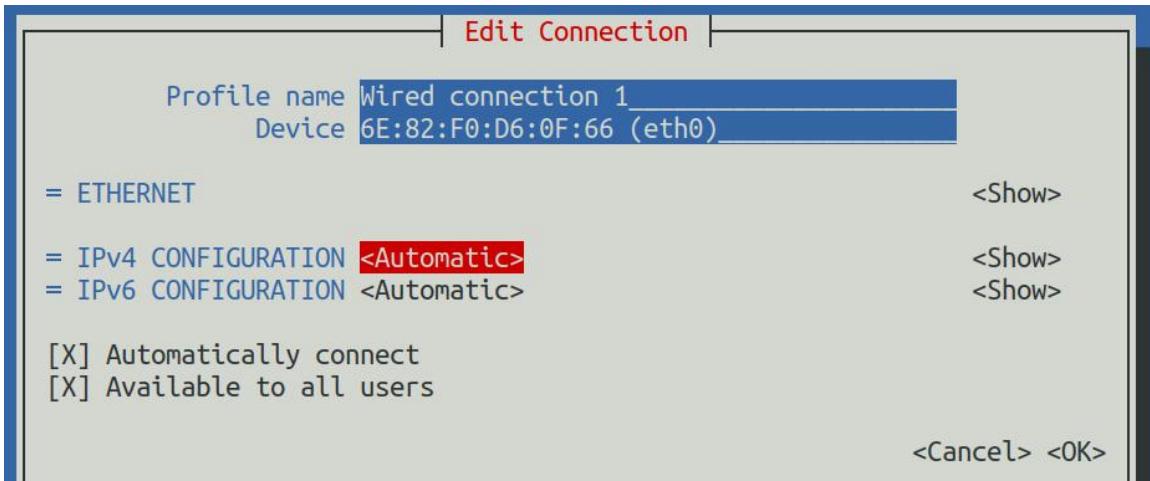
- 3) 然后选择需要设置静态 IP 地址的网络接口，比如设置 **Ethernet** 接口的静态 IP 地址选择 **Wired connection 1** 就可以了



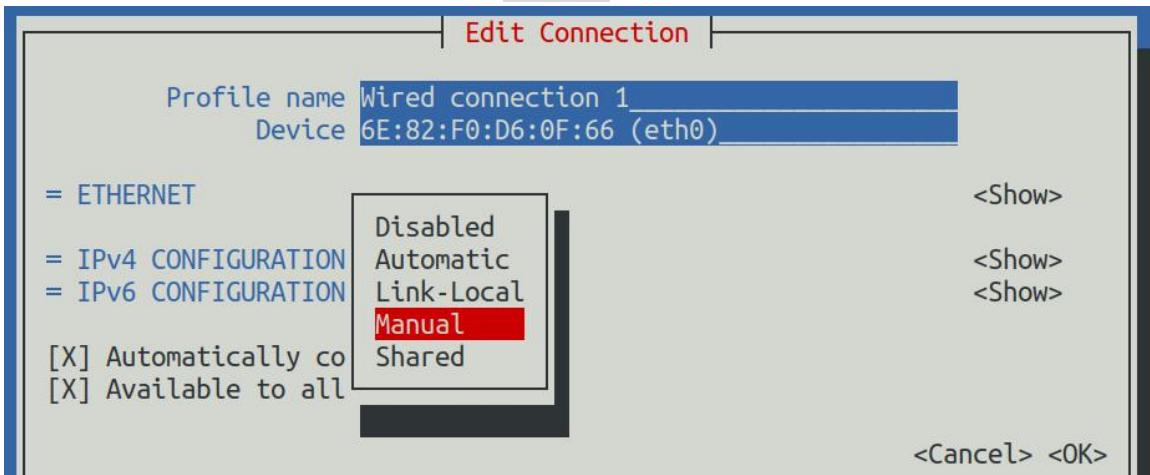
- 4) 然后通过 **Tab** 键选择 **Edit** 并按下回车键



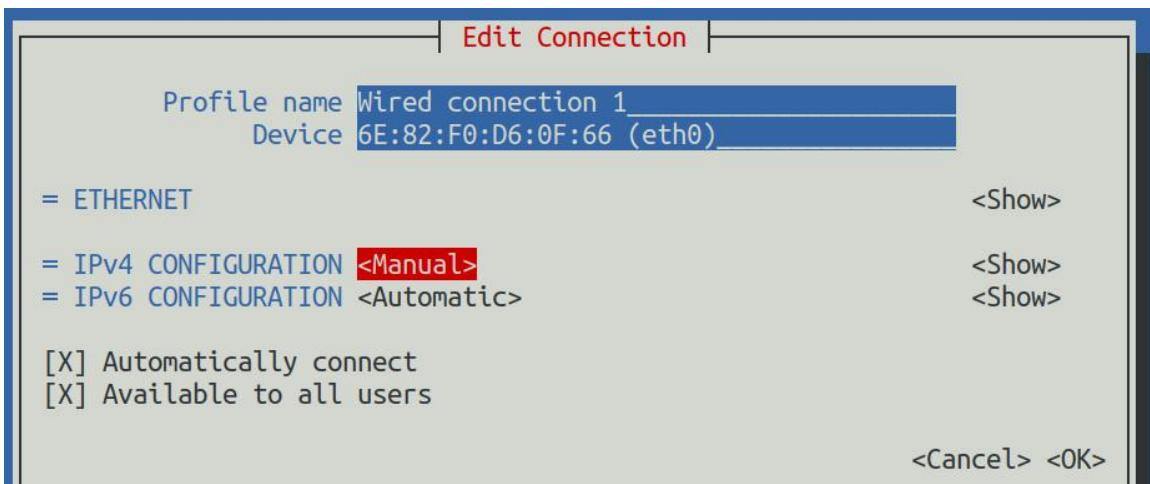
- 5) 然后通过 Tab 键将光标移动到下图所示的<Automatic>位置进行 IPv4 的配置



6) 然后回车，通过上下方向键选择 **Manual**，然后回车确定

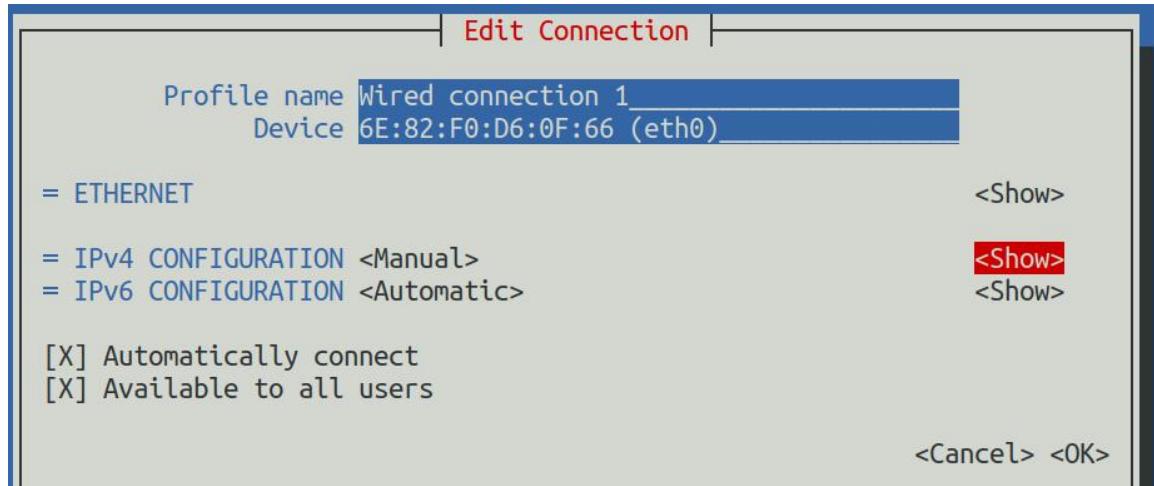


7) 选择完后的显示如下图所示

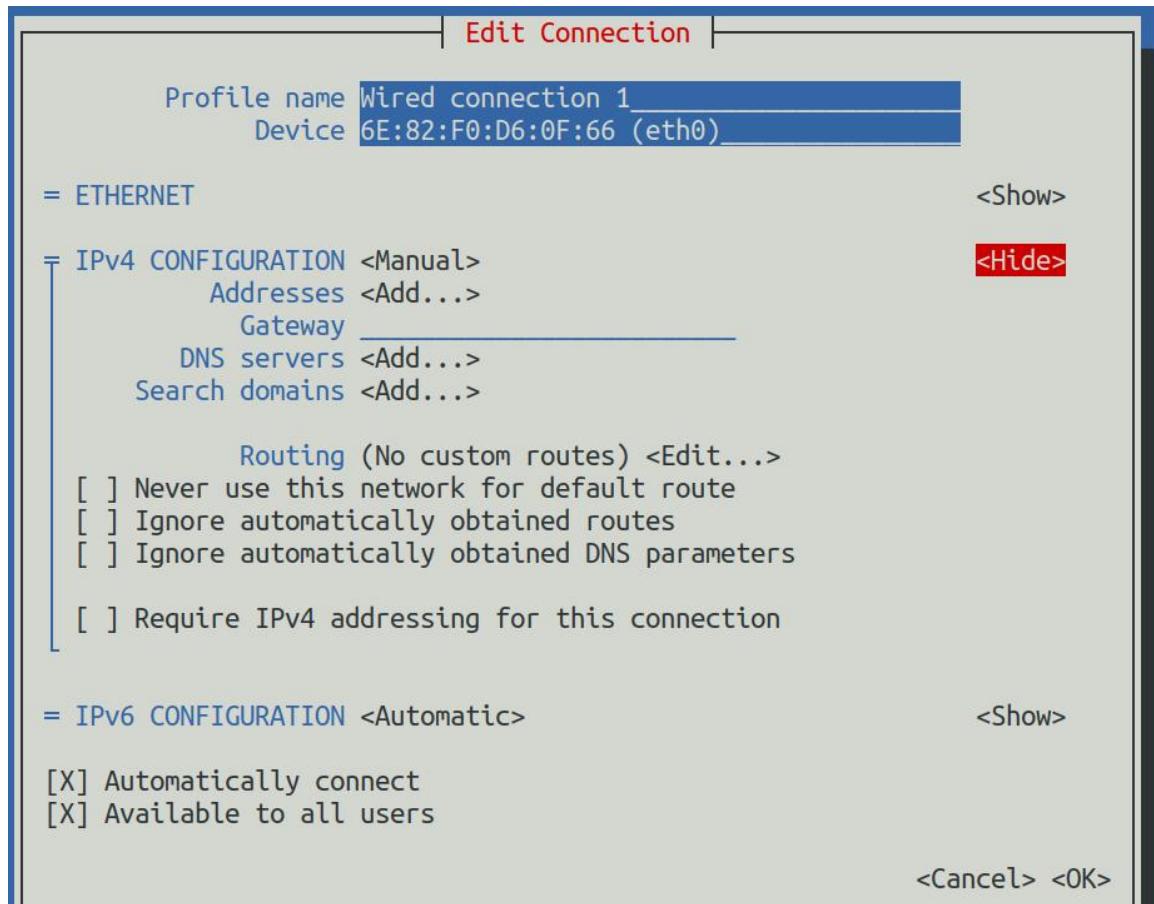




8) 然后通过 Tab 键将光标移动到<Show>



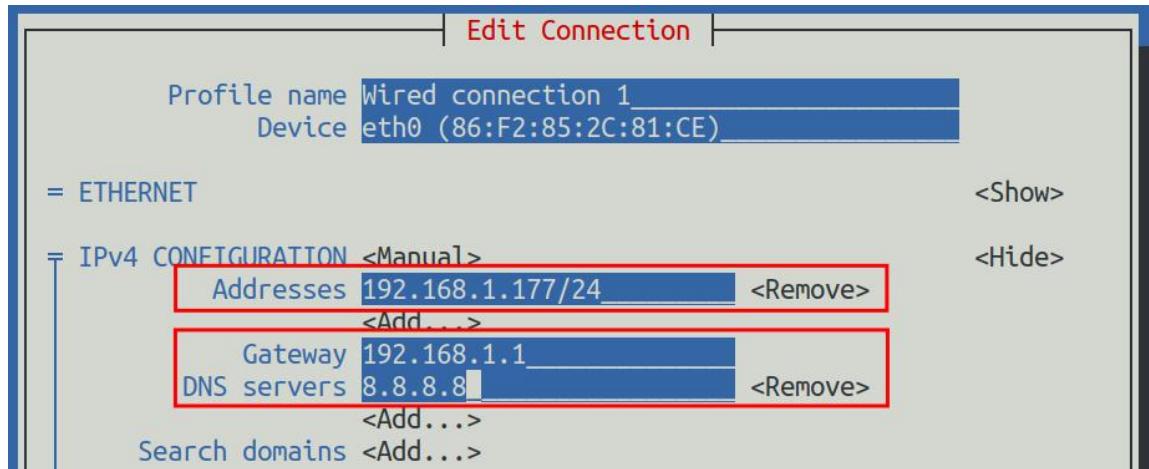
9) 然后回车，回车后会弹出下面的设置界面



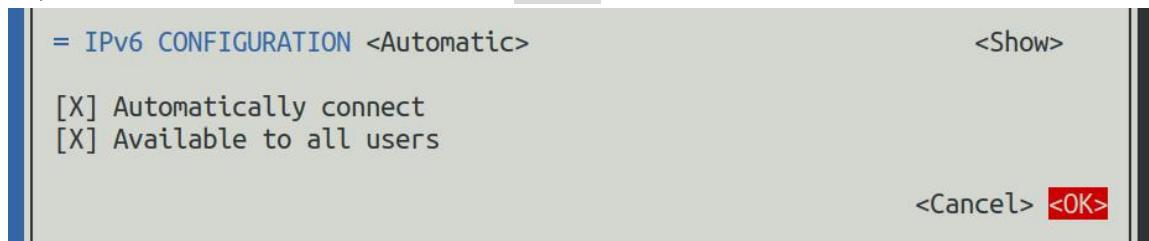
10) 然后就可以在下图所示的位置设置 IP 地址(Addresses)、网关(Gateway)和 DNS



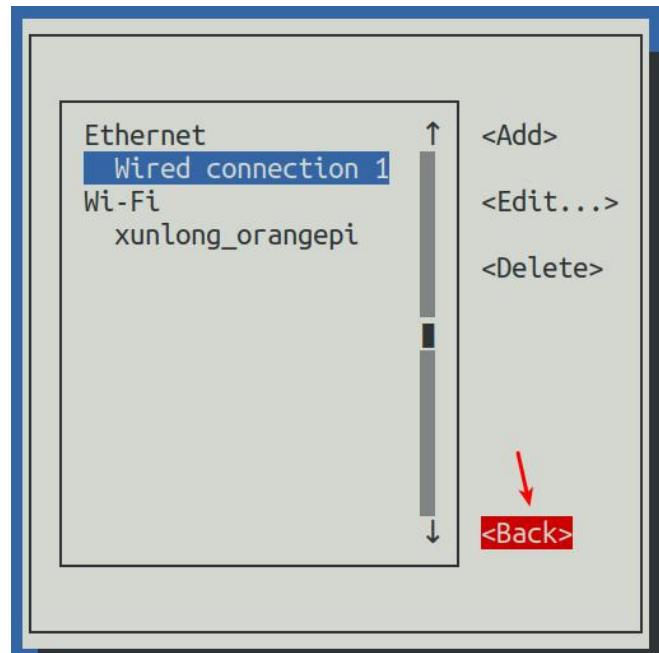
服务器的地址（里面还有很多其他设置选项，请自行探索），**请根据自己的具体需求来设置，下图中设置的值只是一个示例**



11) 设置完后将光标移动到右下角的**<OK>**，然后回车确认

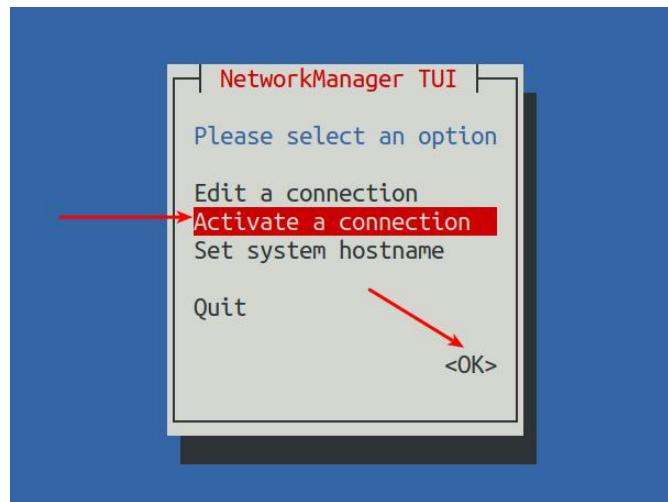


12) 然后点击**<Back>**回退到上一级选择界面

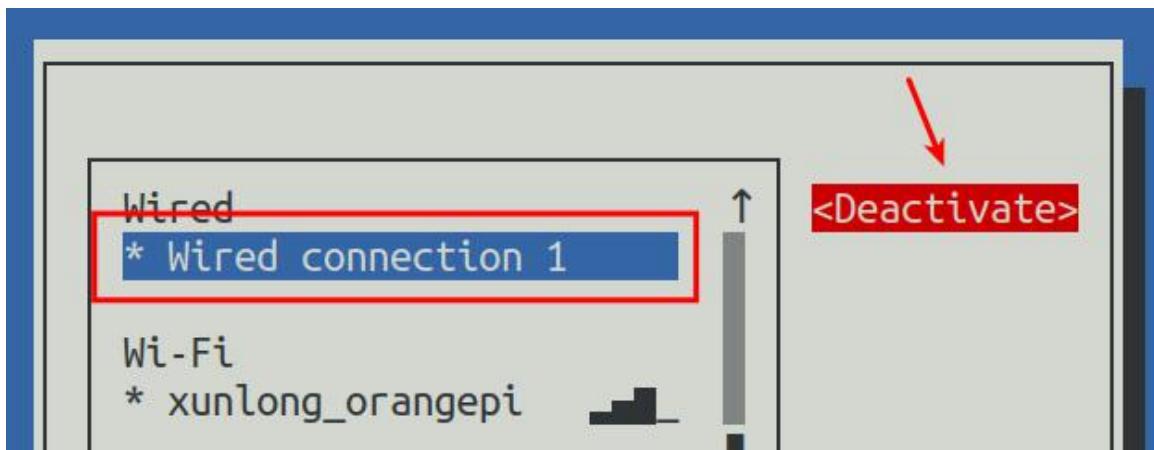




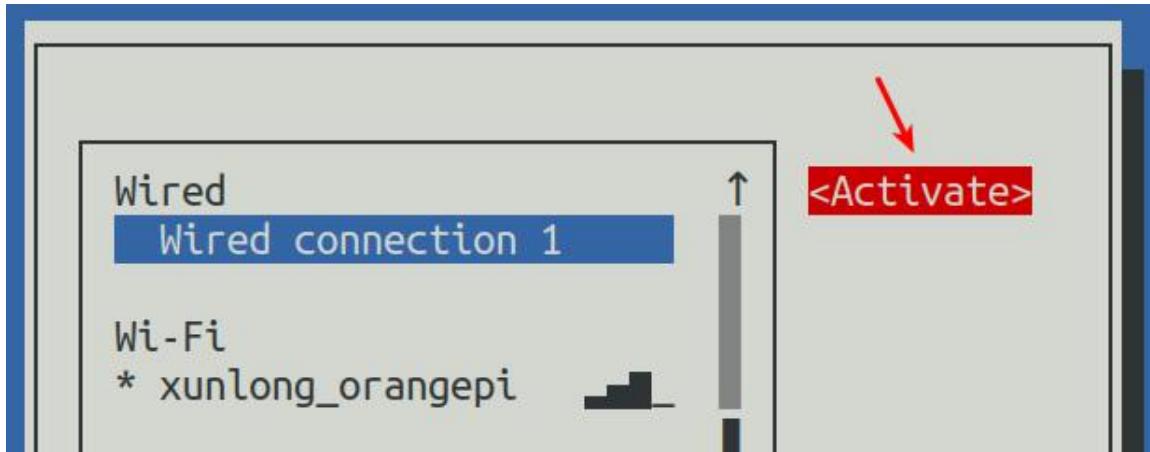
13) 然后选择 **Activate a connection**, 再将光标移动到**<OK>**, 最后点击回车



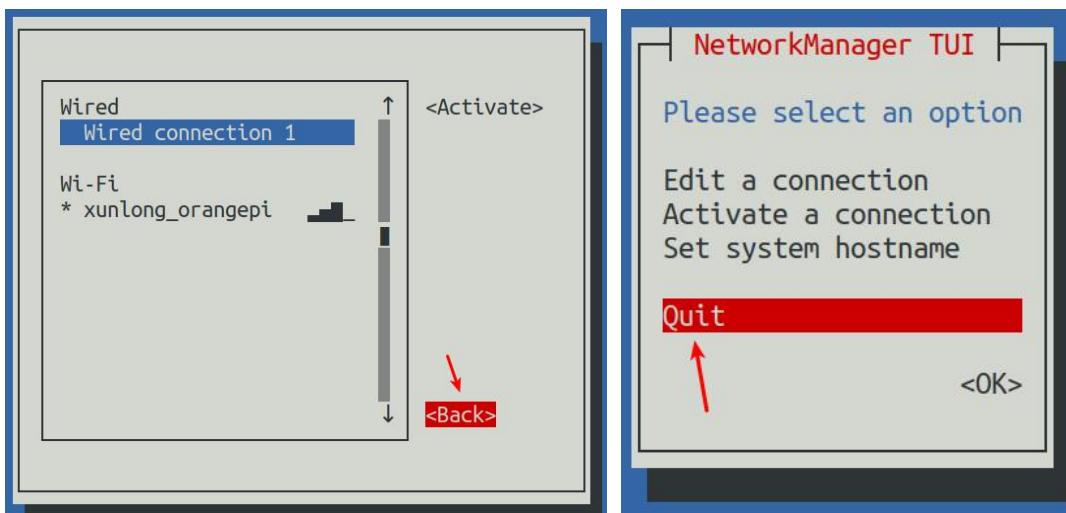
14) 然后选择需要设置的网络接口, 比如 **Wired connection 1**, 然后将光标移动到**<Deactivate>**, 再按下回车键禁用 **Wired connection 1**



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 ifconfig 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

```
root@orangepi:~# ifconfig eth0
eth0: 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::69f6:23d3:a3f3:9772  prefixlen 64  scopeid 0x20<link>
                  ether 86:f2:85:2c:81:ce  txqueuelen 1000  (Ethernet)
                  RX packets 47817  bytes 4047732 (4.0 MB)
                  RX errors 0  dropped 1563  overruns 0  frame 0
                  TX packets 1815  bytes 295954 (295.9 KB)
                  TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
                  device interrupt 64
```



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

```
root@orangepi:~# ping 192.168.1.47 -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. 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 ifconfig 命令或者通过查看路由器的方式获取开发板的 IP 地址

3. 9. 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 root@192.168.1.xxx      (需要替换为开发板的 IP 地址)
root@192.168.1.36's password:      (在这里输入密码， 默认密码为 orangepi)
```

3) 成功登录系统后的显示如下图所示



```
test@test:~$ ssh root@192.168.1.213
root@192.168.1.213's password:

[REDACTED]

Welcome to Orange Pi Bionic with Linux 5.13.0-sun50iw9

System load:  1.00 1.00 1.00   Up time:          55 min   Local users:  2
Memory usage: 30 % of 984MB   IP:      192.168.1.213
CPU temp:     63°C
Usage of /:    16% of 15G

Last login: Wed Oct 13 00:58:48 2021

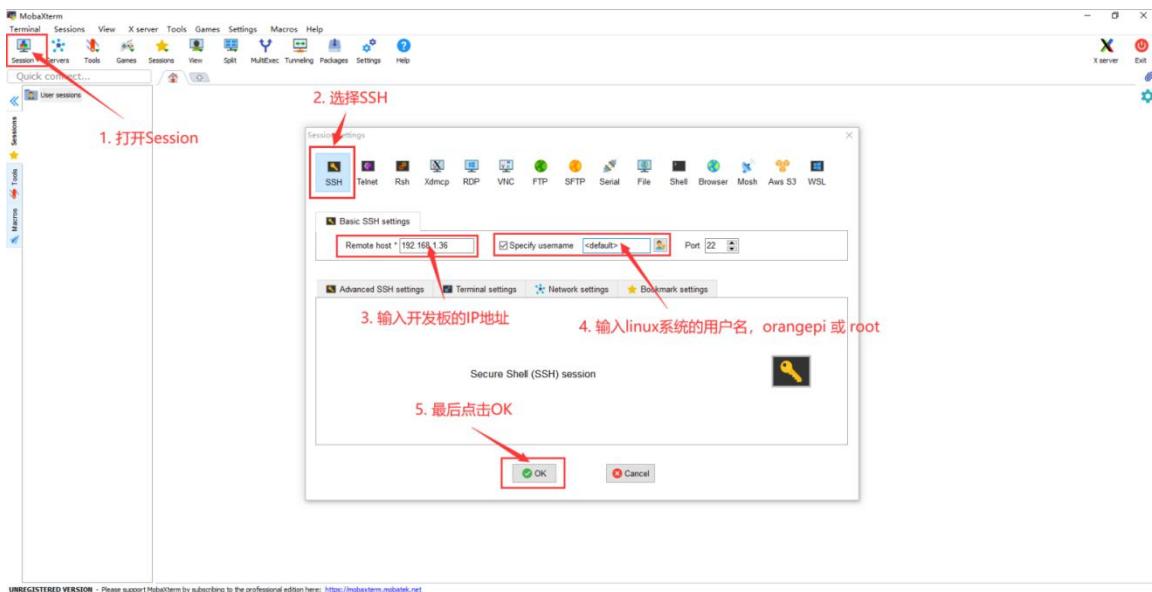
root@orangepizero2:~#
```

如果 ssh 无法正常登陆 Linux 系统，可以在开发板上输入下面的命令后再尝试是否能连接：

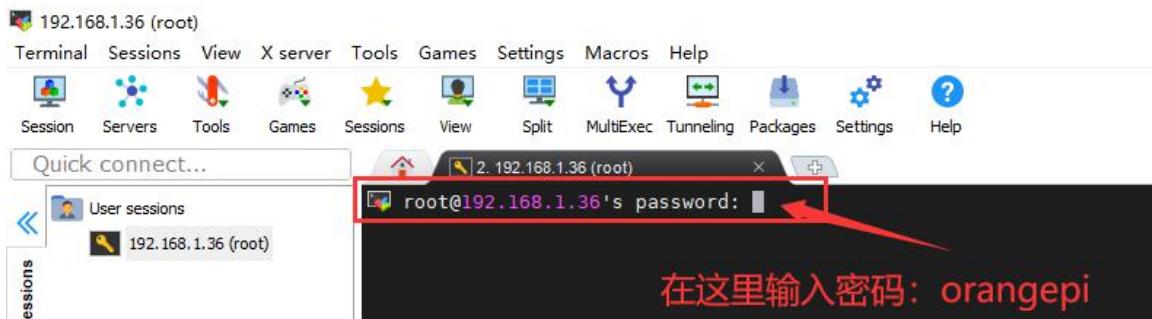
```
root@orangepi:~# rm /etc/ssh/ssh_host_*
root@orangepi:~# dpkg-reconfigure openssh-server
```

3.9.2. Windows 下 SSH 远程登录开发板

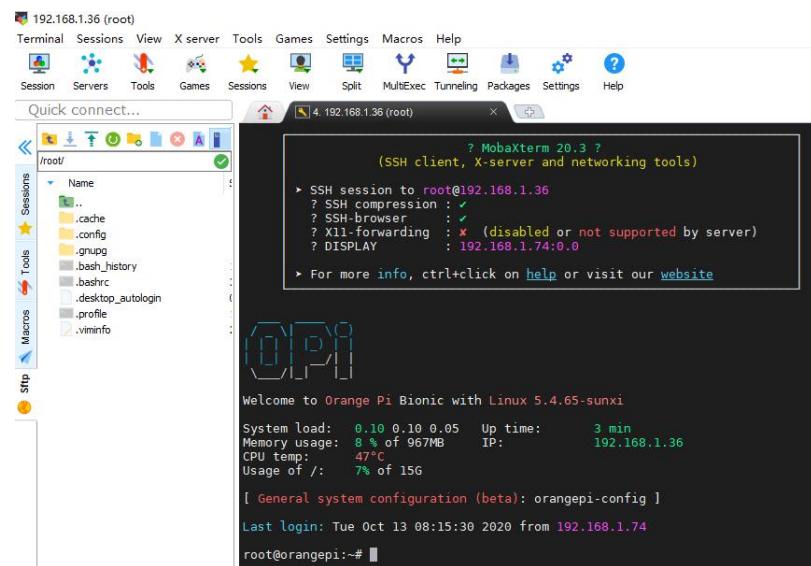
- 1) 首先获取开发板的 IP 地址
 - 2) 在 windows 下可以使用 MobaXterm 远程登录开发板，首先新建一个 ssh 会话
 - a. 打开 **Session**
 - b. 然后在 **Session Setting** 中选择 **SSH**
 - c. 然后在 **Remote host** 中输入开发板的 IP 地址
 - d. 然后 **Specify username** 中输入 linux 系统的用户名 **root** 或 **orangepi**
 - e. 最后点击 **OK** 即可



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



4) 成功登录系统后的显示如下图所示



3. 10. HDMI 显示测试

- 1) 使用 Micro 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

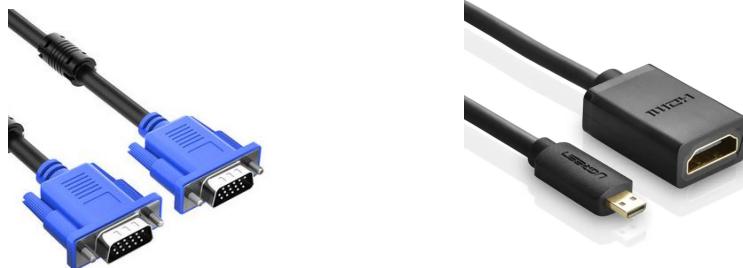
3. 11. HDMI 转 VGA 显示测试

- 1) 首先需要准备下面的配件

- a. HDMI 转 VGA 转换器

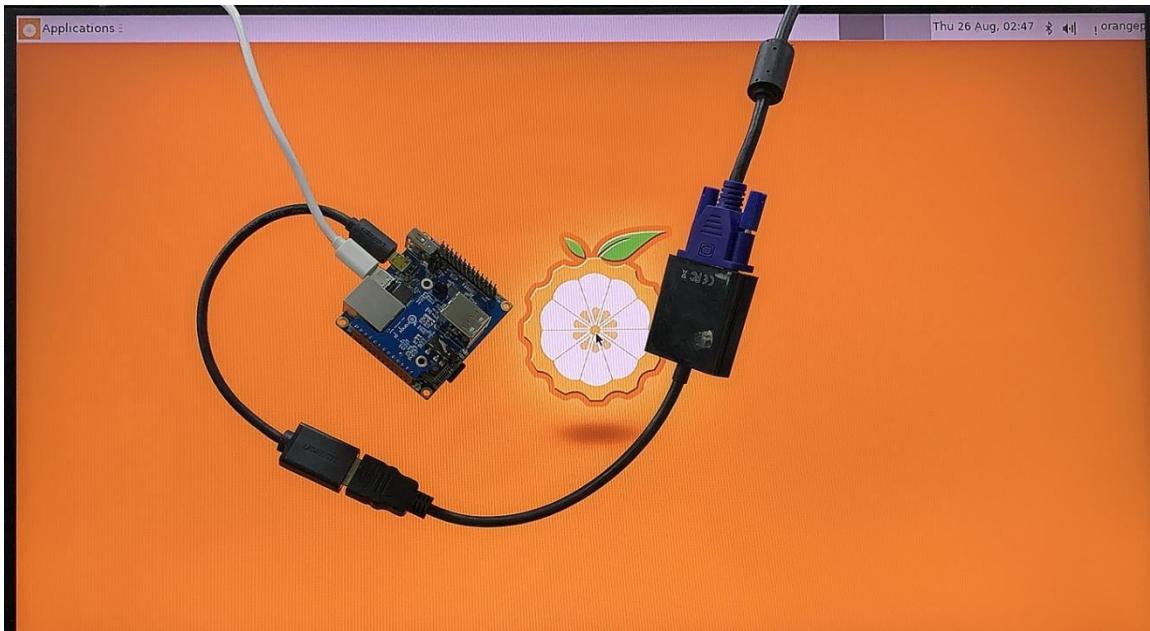


- b. 一根 VGA 线和一根 Micro HDMI 公转 HDMI 母转接线



- 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 Micro 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

3.12. HDMI 分辨率设置

注意：此方法只适用于 linux4.9 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

```
root@orangepi:/boot# cat orangepiEnv.txt
verbosity=1
console=both
disp_mode=1080p60
```

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

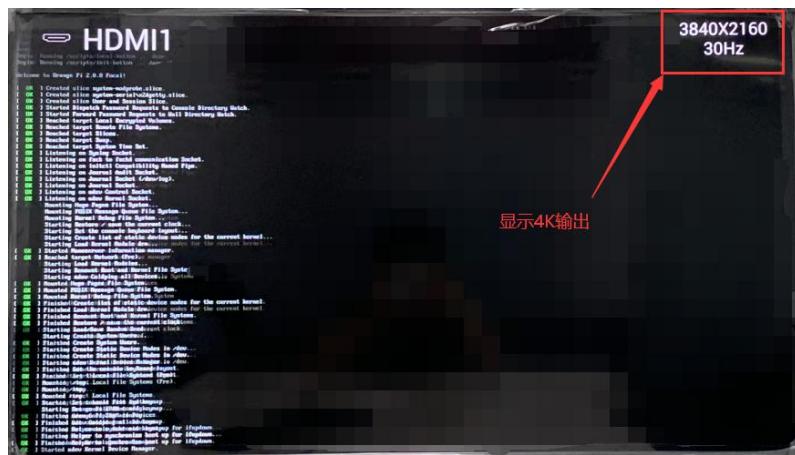


480p	720x480	60
576p	720x576	60
720p50	1280x720	50
720p60	1280x720	60
1080i50	1920x1080	50
1080i60	1920x1080	60
1080p24	1920x1080	24
1080p50	1920x1080	50
1080p60	1920x1080	60
2160p24	3840x2160	24
2160p25	3840x2160	25
2160p30	3840x2160	30

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) 如果设置了 HDMI 的输出分辨率为 **2160p24**、**2160p25** 或者 **2160p30**，HDMI 需要接到支持 4K 显示的电视或者显示器才能正常显示

a. 接到 4K 电视显示如下所示



b. 如果接到不支持 4K 显示的电视或者显示器，会无法显示，如下图所示接到 1080p 的电视直接显示不支持格式



5) 查看 HDMI 输出分辨率的方法如下所示（下图显示的 HDMI 输出的分辨率为 2160p25），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

```
root@orangeipi:~# cat /sys/class/disp/disp/attr/sys
```

```
root@orangepi:/sys/class/disp/disp/attr# cat sys
screen 0:
de rate 6960000000 hz, ref fps:25
mgr0: 3840x2160 fmt[rgb] cs[0x0] range[limit] eotf[0x0] bits[8bits] err[0] force_sync[0] unblank direct_show[false]
dmabuf: cache[0] cache_max[0] unmmap skip[0] overflow[0]
      hdmidi output mode[29] fps:25.2 3840x2160
err[0] skip[1] irq[69215] vsync[0] vsync_skip[0]
      BUF enable ch[1] tyr[0] z[0] prem[N] aglob[255] fmt[ 0] fb[1920,1080;1920,1080;1920,1080] crop[ 0, 0,1920,1080] frame[ 0, 0,3840,2160] addr[fe000000,
0, 0] flags[0x 0] trd[0,0]
depth[ 0] trans[0] -
```

3.13. Framebuffer 宽度和高度的修改方法

注意：此方法只适用于 linux4.9 内核的系统

1) 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=1280`、`fb0_height=720`

```
root@orangepi:~# cat /boot/orangepiEnv.txt
verbosity=1
console=both
disp_mode=1080p60
fb0_width=1280
fb0_height=720
```

2) fb0 width 和 fb0 height 不同分辨率对应的参考值如下所示

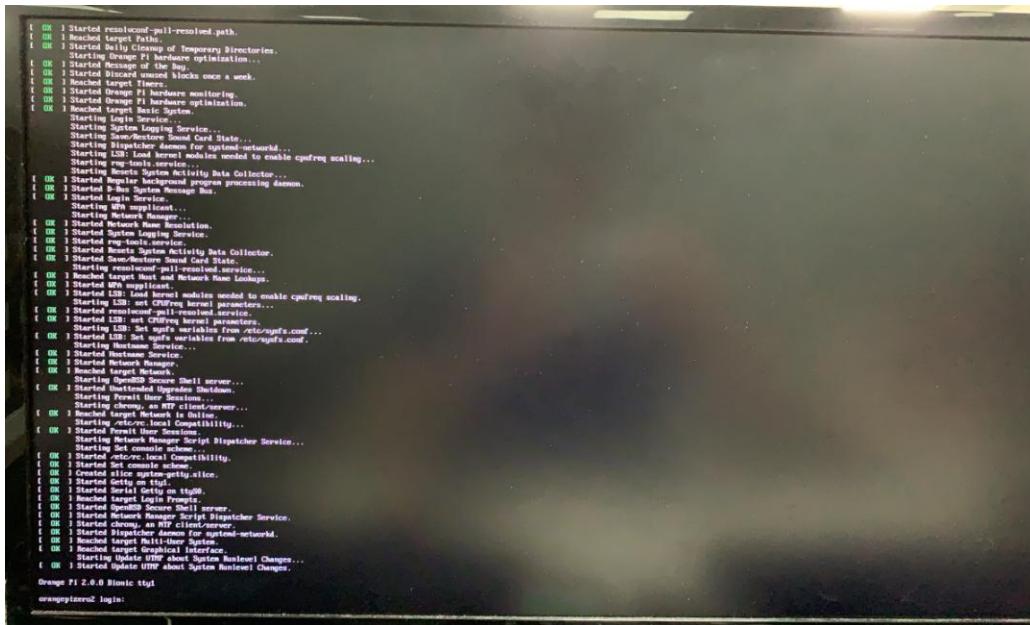
注意，当 HDMI 的分辨率设置为 2160p24、2160p25 和 2160p30 时，Framebuffer

**最高只能设置为 1920x1080**

HDMI 分辨率	fb0_width	fb0_height
480p	720	480
576p	720	576
720p	1280	720
1080p	1920	1080
2160p	1920	1080

3) 在相同的 HDMI 分辨下，不同的 fb0_width 和 fb0_height 的显示情况如下所示，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

a. HDMI 分辨率为 1080p60, fb0_width 和 fb0_height 为 1920x1080 的显示情况



b. HDMI 分辨率为 1080p60, fb0_width 和 fb0_height 为 1280x720 的显示情况



```
[ OK ] Started System Logging Service.
[ OK ] Started Resnets System Activity Data Collector.
[ OK ] Started rsys-tools.service.
[ OK ] Started Save/Restore Sound Card State.
[ OK ] Started Login Service.
  Starting resolvconf-pull-resolved.service...
[ OK ] Reached target Host and Network Name Lookups.
[ OK ] Started WPA supplicant.
[ OK ] Started LSB: Load kernel modules needed to enable cpufreq scaling.
  Starting kernel: set CPUFreq kernel parameters...
[ OK ] Started LSB: set CPUFreq kernel parameters.
[ OK ] Started LSB: Set sysfs variables from /etc/sysfs.conf...
[ OK ] Started LSB: Set sysfs variables from /etc/sysfs.conf.
  Starting Hostname Service...
[ OK ] Started Hostname Service.
[ OK ] Started Network Manager.
[ OK ] Reached target Network.
  Starting chrony, an NTP client/server...
  Starting Permit User Sessions...
[ OK ] Started LSB: Unattended Upgrades Shutdown.
[ OK ] Started LSB: Set sysfs variables from /etc/sysfs.conf...
  Starting OpenBSD Secure Shell server...
[ OK ] Reached target Network is Online.
  Starting /etc/rc.local Compatibility...
[ OK ] Started Permit User Sessions.
[ OK ] Started /etc/rc.local Compatibility.
  Starting Network Manager Script Dispatcher Service...
  Starting Set console scheme...
[ OK ] Started Serial Getty on ttyS0.
[ OK ] Created slice system-getty.slice.
[ OK ] Started Getty on ttym1.
[ OK ] Reached target Login Prompts.
[ OK ] Started Network Manager Script Dispatcher Service.
[ OK ] Started OpenBSD Secure Shell server.
[ OK ] Started chrony, an NTP client/server...
  Started Dispatcher daemon for systemd-networkd.
[ OK ] Reached target Multi-User System.
[ OK ] Reached target Graphical Interface.
  Starting Update UTMP about System Runlevel Changes...
[ OK ] Started Update UTMP about System Runlevel Changes.

Orange Pi 2.0.0 Bionic ttym1
orangepizero2 login:
```

c. HDMI 分辨率为 1080p60, fb0_width 和 fb0_height 为 720x576 的显示情况

```
[ OK ] Started LSB: Load kernel modules needed to enable cpufreq scaling.
  Starting LSB: set CPUFreq kernel parameters...
[ OK ] Started LSB: set CPUFreq kernel parameters.
  Starting LSB: Set sysfs variables from /etc/sysfs.conf...
[ OK ] Started LSB: Set sysfs variables from /etc/sysfs.conf.
  Starting Hostname Service...
[ OK ] Started Hostname Service.
[ OK ] Started Network Manager.
[ OK ] Reached target Network.
[ OK ] Reached target Network is Online.
[ OK ] Started Unattended Upgrades Shutdown.
  Starting chrony, an NTP client/server...
  Starting OpenBSD Secure Shell server...
  Starting /etc/rc.local Compatibility...
  Starting Permit User Sessions...
[ OK ] Started Permit User Sessions.
  Starting Network Manager Script Dispatcher Service...
  Starting Set console scheme...
[ OK ] Started /etc/rc.local Compatibility.
[ OK ] Started Set console scheme.
[ OK ] Created slice system-getty.slice.
[ OK ] Started Serial Getty on ttym0.
[ OK ] Started Getty on ttym1.
[ OK ] Reached target Login Prompts.
[ OK ] Started Network Manager Script Dispatcher Service.
[ OK ] Started OpenBSD Secure Shell server.
[ OK ] Started chrony, an NTP client/server.
  Started Dispatcher daemon for systemd-networkd.
[ OK ] Reached target Multi-User System.
[ OK ] Reached target Graphical Interface.
  Starting Update UTMP about System Runlevel Changes...
[ OK ] Started Update UTMP about System Runlevel Changes.

Orange Pi 2.0.0 Bionic ttym1
orangepizero2 login:
```

d. HDMI 分辨率为 1080p60, fb0_width 和 fb0_height 为 720x480 的显示情况



```
[ OK ] Started Hostname Service.  
[ OK ] Started Network Manager.  
[ OK ] Reached target Network.  
      Starting OpenBSD Secure Shell server...  
[ OK ] Reached target Network is Online.  
[ OK ] Started Unattended Upgrades Shutdown.  
      Starting /etc/rc.local Compatibility...  
      Starting chrony, an NTP client/server...  
      Starting Permit User Sessions...  
[ OK ] Started Permit User Sessions.  
      Starting Set console scheme...  
      Starting Network Manager Script Dispatcher Service...  
[ OK ] Started /etc/rc.local Compatibility.  
[ OK ] Started Serial Getty on ttyS0.  
[ OK ] Started Set console scheme.  
[ OK ] Created slice system-getty.slice.  
[ OK ] Started Getty on tty1.  
[ OK ] Reached target Login Prompts.  
[ OK ] Started Network Manager Script Dispatcher Service.  
[ OK ] Started chrony, an NTP client/server.  
[ OK ] Started OpenBSD Secure Shell server.  
[ OK ] Started Dispatcher daemon for systemd-networkd.  
[ OK ] Reached target Multi-User System.  
[ OK ] Reached target Graphical Interface.  
      Starting Update UTMP about System Runlevel Changes...  
[ OK ] Started Update UTMP about System Runlevel Changes.  
  
Orange Pi 2.0.8 Bionic tty1  
orangepizero2 login:
```

3.14. 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

```
root@orangepi:~# cat /boot/orangepiEnv.txt  
verbosity=1  
console=both  
disp_mode=1080p60  
fb0_width=1280  
fb0_height=720  
extraargs=vt.global_cursor_default=0
```



3.15. 香橙派 5 寸 TFT 液晶屏测试

注意： 此方法只适用于 linux4.9 内核的系统

- 1) 首先准备好香橙派的 5 寸 TFT 液晶屏，屏幕排线和转接板的接线方式如下图所示，请勿接反了，另外请确保屏幕排线和排线插座接触都到位了，如果接触不到位，屏幕输出会有问题



- 2) 然后使用 Micro HDMI 线将开发板的 Micro HDMI 接口连接到转接板的 HDMI 接口中，屏幕需要单独供电，请在转接板的 Micro USB 接口中插入 5V/2A 的电源



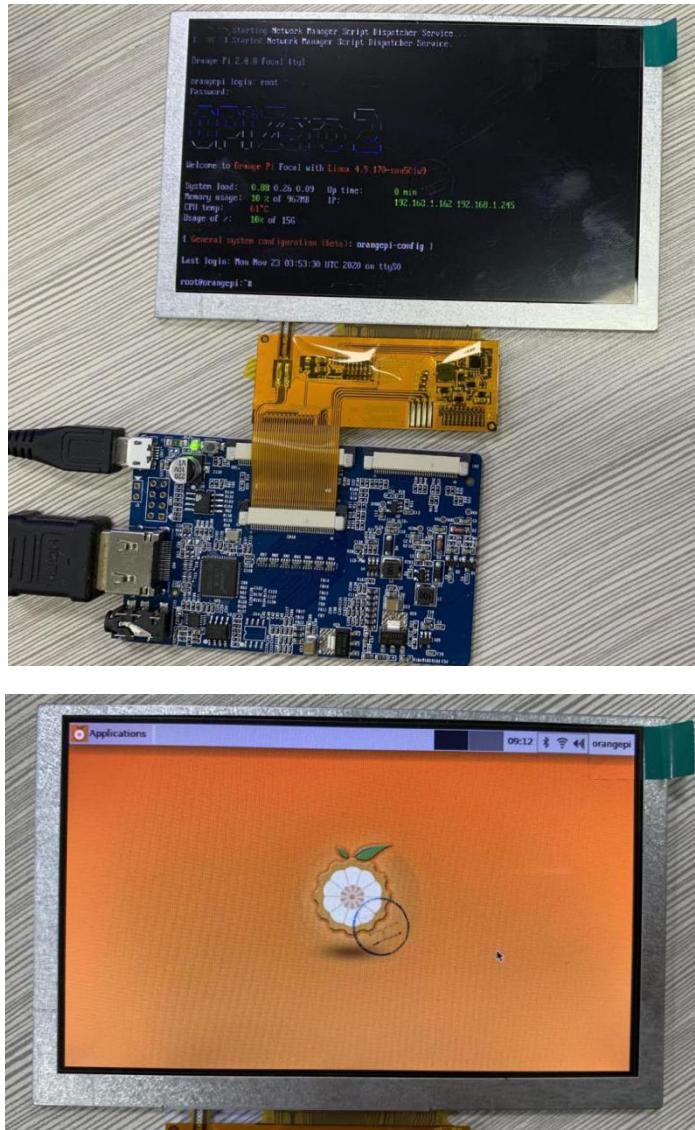
- 3) 在 /boot/orangepiEnv.txt 中修改 HDMI 输出分辨率为 480p，Framebuffer 的长度和宽度可以设置为 800x480，如果设置为 1280x720，TFT 液晶屏显示的字体会很小

```
root@orangepi:~# cat /boot/orangepiEnv.txt
disp_mode=480p
fb0_width=800
```



fb0_height=480

4) 开发板启动后，屏幕就可以看到启动画面了



5) 关闭开发板的电源后，请记得同时关闭屏幕转接板的电源，下次启动开发板时再打开

3.16. 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

3.16.1. 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 显示屏，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

```
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_TC15	Infra	1	130 Mbit/s	79	[REDACTED]	WPA1 WPA2
	A0:40:A0:A1:72:31	NETGEAR04	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

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

```
root@orangeipi:~# nmcli dev wifi connect wifi_name password wifi_passwd
```

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

4) 通过 **ifconfig** 命令可以查看 wifi 的 IP 地址



```
root@orangepi:~# ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.1.49 netmask 255.255.255.0 broadcast 192.168.1.255
              inet6 fe80::76bb:f67d:ef98:2f9a prefixlen 64 scopeid 0x20<link>
                ether 12:81:3e:a8:58:d8 txqueuelen 1000 (Ethernet)
                  RX packets 185 bytes 109447 (109.4 KB)
                  RX errors 0 dropped 61 overruns 0 frame 0
                  TX packets 27 bytes 14783 (14.7 KB)
                  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

5) 使用 ping 命令可以测试 wifi 网络的连通性

```
root@orangepi:~# ping www.orangepi.org -I wlan0
PING www.orangepi.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.orangepi.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
```

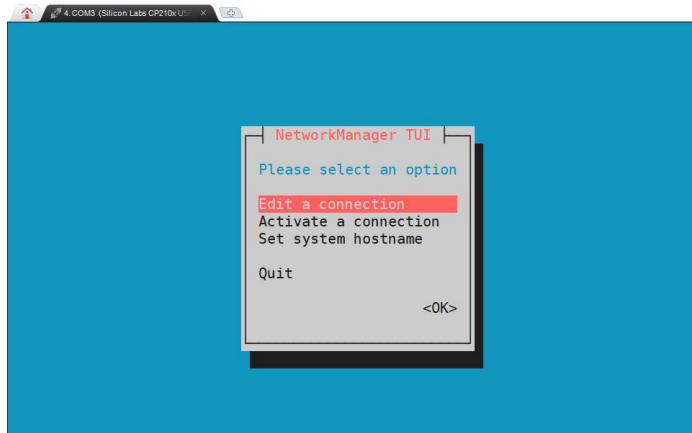
3.16.2. 服务器版镜像通过图形化方式连接 WIFI

- 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. 如果开发板连接了开发板到 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

```
root@orangepi:~# nmtui
```



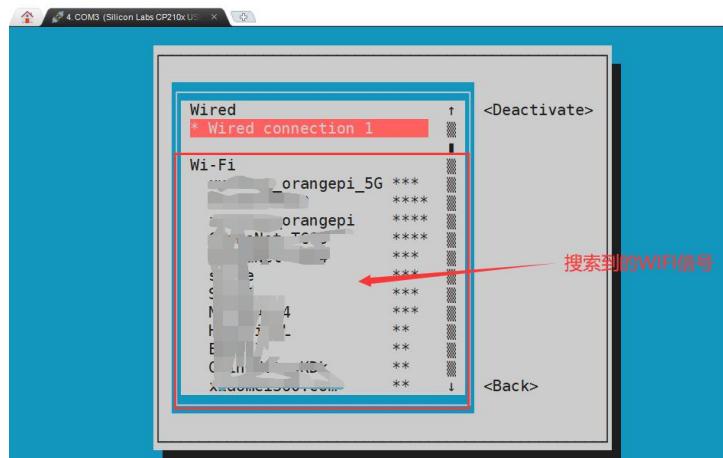
3) 输入 nmtui 命令打开的界面如下所示



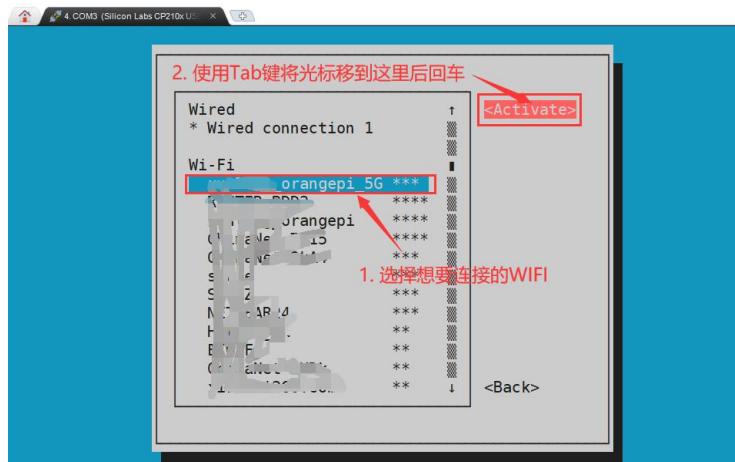
4) 选择 **Activate a connect** 后回车



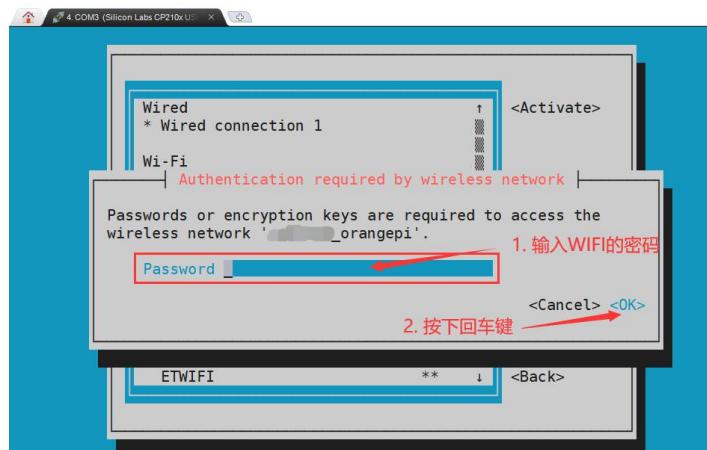
5) 然后就能看到所有搜索到的 WIFI 热点



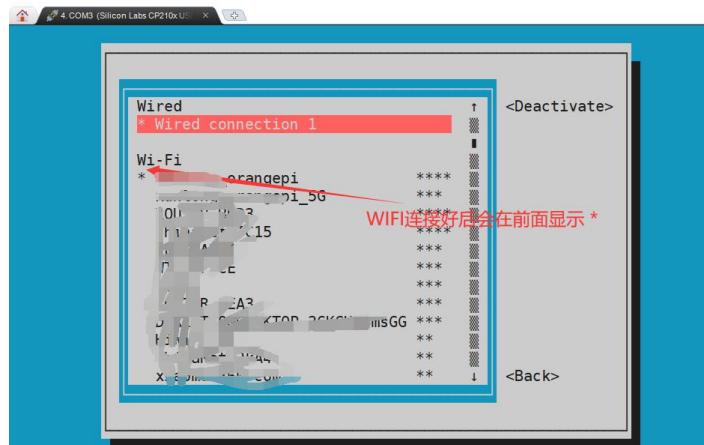
6) 选择想要连接的 WIFI 热点后再使用 Tab 键将光标定位到 **Activate** 后回车



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 **ifconfig** 命令可以查看 wifi 的 IP 地址



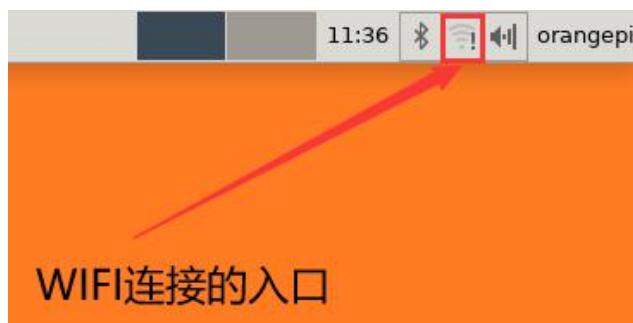
```
root@orangepi:~# ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.1.49 netmask 255.255.255.0 broadcast 192.168.1.255
              inet6 fe80::76bb:f67d:ef98:2f9a prefixlen 64 scopeid 0x20<link>
                ether 12:81:3e:a8:58:d8 txqueuelen 1000 (Ethernet)
                  RX packets 185 bytes 109447 (109.4 KB)
                  RX errors 0 dropped 61 overruns 0 frame 0
                  TX packets 27 bytes 14783 (14.7 KB)
                  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

10) 使用 **ping** 命令可以测试 wifi 网络的连通性

```
root@orangepi:~# ping www.orangepi.org -I wlan0
PING www.orangepi.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.orangepi.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
```

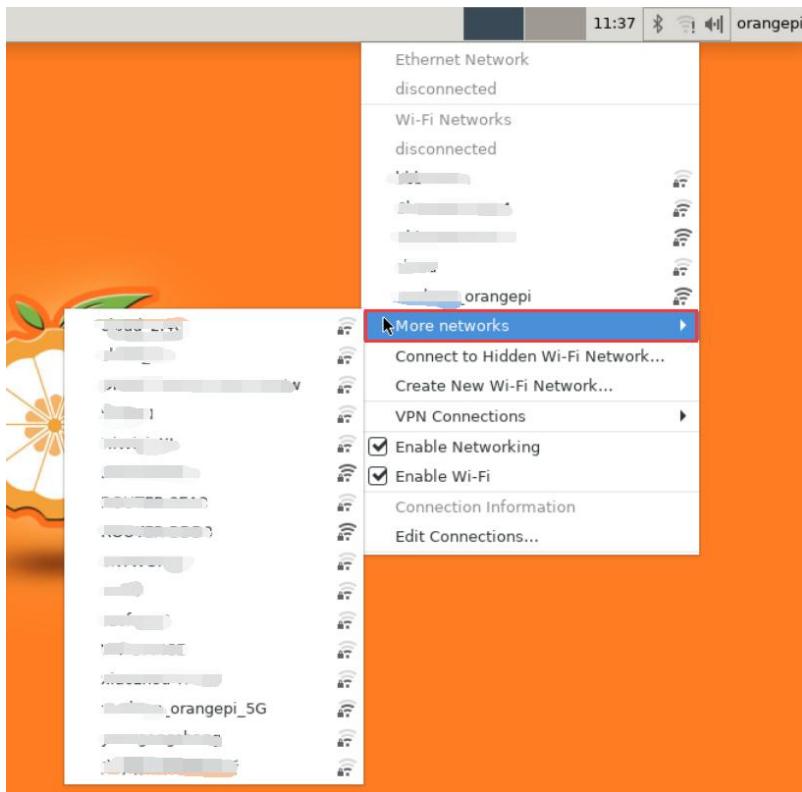
3.16.3. 桌面板镜像的测试方法

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）





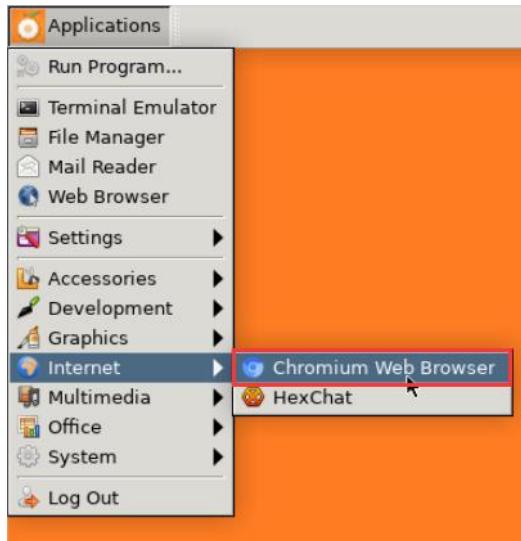
2) 在弹出的下拉框中点击 **More networks** 可以看到所有扫描到的 WIFI 热点，然后选择想要连接的 WIFI 热点



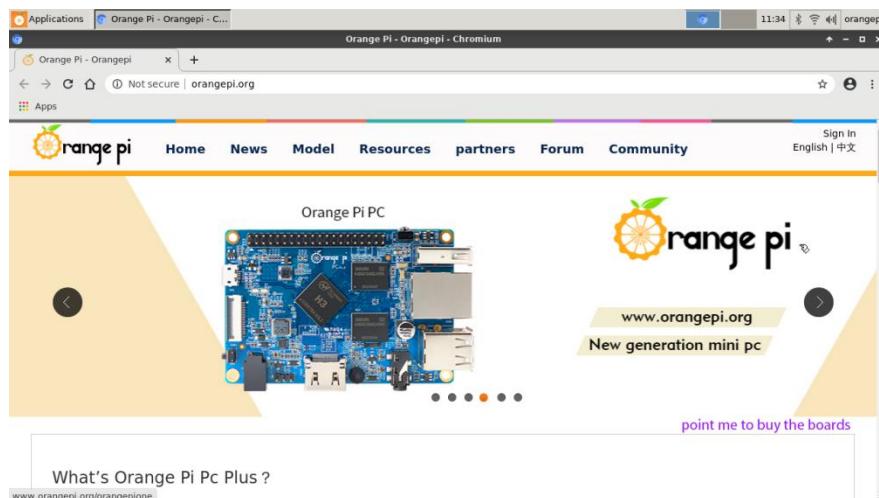
3) 然后输入 WIFI 热点的密码，再点击 **Connect** 就会开始连接 WIFI



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看到 Orange Pi 网站的页面，或者能打开其他网页说明 WIFI 连接正常



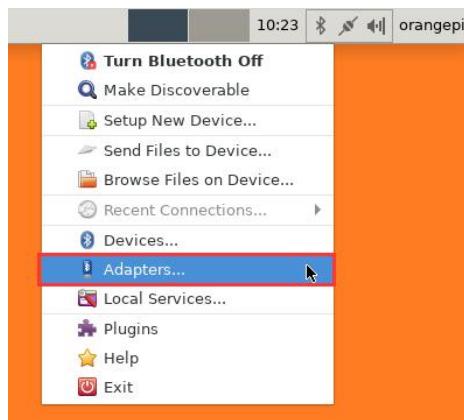
3.17. 蓝牙使用方法

3.17.1. 桌面版镜像的测试方法

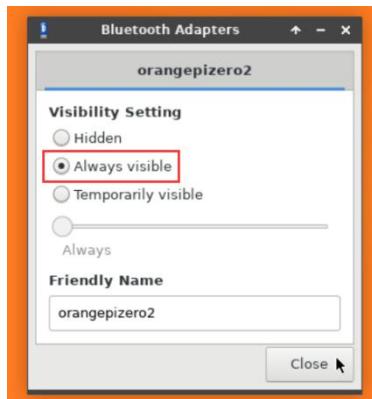
1) 点击桌面右上角的蓝牙图标



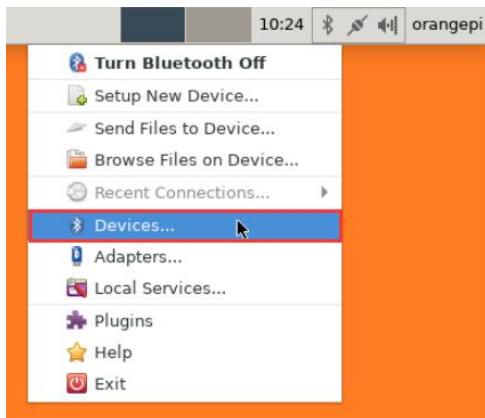
2) 然后选择适配器



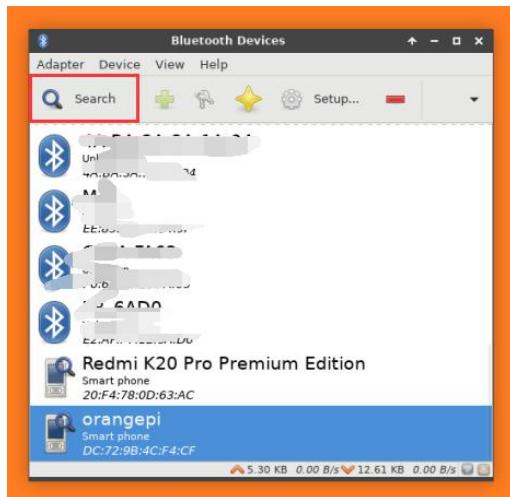
3) 在蓝牙的适配器设置界面中设置 **Visibility Setting** 为 **Always visible**, 然后点击 **close** 关闭



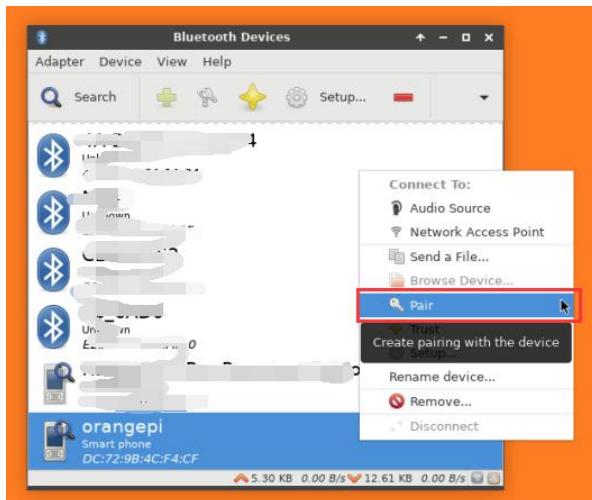
4) 然后打开蓝牙设备的配置界面



5) 点击 **Search** 即可开始扫描周围的蓝牙设备

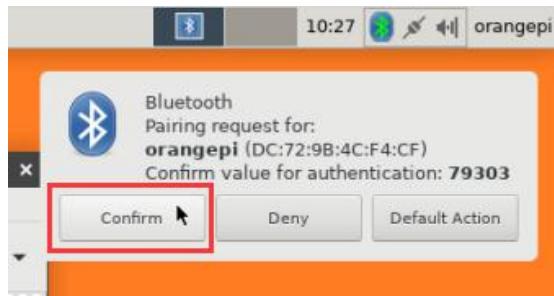


6) 然后选择想要连接的蓝牙设备，再点击鼠标右键就会弹出对此蓝牙设备的操作界面，选择 **Pair** 即可开始配对，这里演示的是和 Android 手机配对

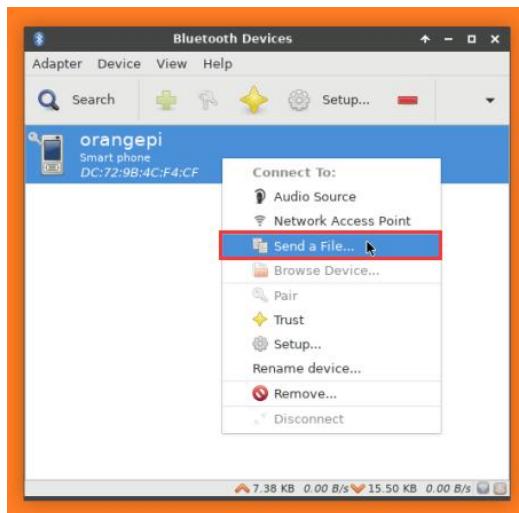




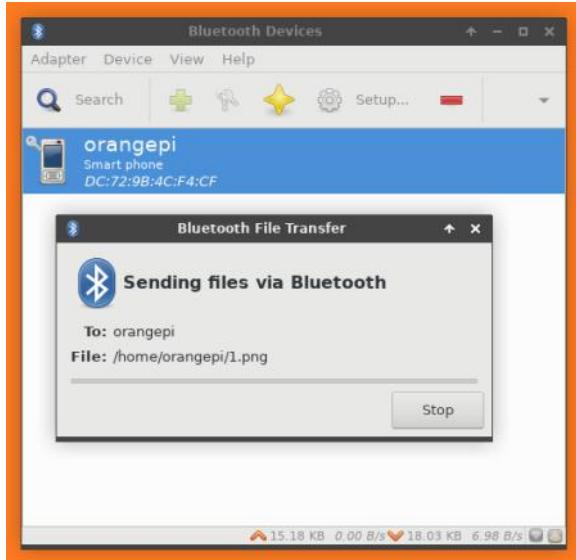
7) 配对时，桌面的右上角会弹出配对确认框，选择 **Confirm** 确认即可，此时手机上也同样需要进行确认



8) 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



9) 发送图片的界面如下所示



3.17.2. 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

```
root@orangeipi:~# hciconfig -a
hci0:  Type: Primary  Bus: UART
          BD Address: 10:11:12:13:14:15  ACL MTU: 1021:8  SCO MTU: 240:3
          UP RUNNING
          RX bytes:646 acl:0 sco:0 events:37 errors:0
          TX bytes:2650 acl:0 sco:0 commands:37 errors:0
          Features: 0xbff 0xff 0x8d 0xfe 0xdb 0x3d 0x7b 0xc7
          Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
          Link policy:
          Link mode: SLAVE ACCEPT
          Name: 'orangepirzero2'
          Class: 0x000000
          Service Classes: Unspecified
          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

```
root@orangeipi:~# bluetoothctl
```



```
[NEW] Controller 10:11:12:13:14:15 orangepizero2 [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 orangepi
[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
orangepi，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

```
root@orangeipi:~# apt update  
root@orangeipi:~# apt -y install pulseaudio-module-bluetooth  
root@orangeipi:~# pulseaudio --start
```

6) 连接蓝牙设备的方法

```
root@orangeipi:~# 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



3.18. 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	亮	灭
GPIO 口	PC13	PC12

2) 设置绿灯亮灭和闪烁的方法如下所示

a. 首先进入绿灯的设置目录

```
root@orangeipi:~# cd /sys/class/leds/orangeipi:green\:status
```

b. 设置绿灯熄灭的命令如下

```
root@orangeipi:/sys/class/leds/orangeipi:green:status# echo 0 > brightness
```

c. 设置绿灯常亮的命令如下

```
root@orangeipi:/sys/class/leds/orangeipi:green:status# echo 1 > brightness
```

d. 设置绿灯闪烁的命令如下

```
root@orangeipi:/sys/class/leds/orangeipi:green:status# echo heartbeat > trigger
```

e. 设置绿灯停止闪烁的命令如下

```
root@orangeipi:/sys/class/leds/orangeipi:green:status# echo none > trigger
```

3) 设置红灯亮灭和闪烁的方法如下所示

a. 首先进入红灯的设置目录



```
root@orangepi:~# cd /sys/class/leds/orangepi:red\:status
```

b. 设置红灯熄灭的命令如下

```
root@orangepi:/sys/class/leds/orangepi:red:status# echo 0 > brightness
```

c. 设置红灯常亮的命令如下

```
root@orangepi:/sys/class/leds/orangepi:red:status# echo 1 > brightness
```

d. 设置红灯闪烁的命令如下

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

e. 设置红灯停止闪烁的命令如下

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

3.19. USB 接口测试

3.19.1. 连接鼠标或键盘测试

- 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系统说明 USB 接口使用正常（鼠标只有在桌面版的系统中才能使用）

3.19.2. 连接 USB 存储设备测试

- 1) 首先将 U 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

```
root@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

```
root@orangepi:~# mount /dev/sda1 /mnt/
```

```
root@orangepi:~# 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

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

3. 20. USB 无线网卡测试

目前测试过的能用的 USB 无线网卡如下所示，其他型号的 USB 无线网卡请自行测试，如果无法使用就需要移植对应的 USB 无线网卡驱动

序号	型号
1	RTL8723BU

3. 20. 1. 1. RTL8723BU 测试

1) 首先将 RTL8723BU 无线网卡模块插入开发板的 USB 接口中

2) 然后 linux 系统会自动加载 RTL8723BU 蓝牙和 WIFI 相关的内核模块，通过 lsmod 命令可以看到下面输出

```
root@orangeipi:~# lsmod
Module           Size  Used by
rtl8xxxu        143360  0
mac80211        491520  1 rtl8xxxu
rtk_btusb       65536   0
btusb           49152   0
btrtl           16384   1 btusb
btbcm           16384   1 btusb
btintel         20480   1 btusb
```

3) 通过 dmesg 命令可以看到 RTL8723BU 模块的加载信息

```
root@orangeipi:~# dmesg
.....
[ 166.867806] sunxi-ehci 5311000.ehci3-controller: ehci_irq: highspeed device connect
[ 167.090509] usb 3-1: new high-speed USB device number 2 using sunxi-ehci
[ 167.228930] usb 3-1: New USB device found, idVendor=0bda, idProduct=b720
```



```
[ 167.228955] usb 3-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 167.228971] usb 3-1: Product: 802.11n WLAN Adapter
[ 167.228985] usb 3-1: Manufacturer: Realtek
[ 167.229000] usb 3-1: SerialNumber: 00e04c000001
[ 167.336331] usbcore: registered new interface driver btusb
[ 167.337532] Bluetooth: hci1: rtl: examining hci_ver=06 hci_rev=000b lmp_ver=06
lmp_subver=8723
[ 167.337544] Bluetooth: hci1: rtl: loading rtl_bt/rtl8723b_config.bin
[ 167.342938] Bluetooth: hci1: rtl: loading rtl_bt/rtl8723b_fw.bin
[ 167.347539] Bluetooth: hci1: rom_version status=0 version=1
[ 167.347629] Bluetooth: cfg_sz 68, total size 24088
[ 167.352702] rtk_btusb: RTKBT_RELEASE_NAME: 20170427_TV_ANDROID_5.x
[ 167.352717] rtk_btusb: Realtek Bluetooth USB driver module init, version 4.1.4
[ 167.352721] rtk_btusb: Register usb char device interface for BT driver
[ 167.366566] usbcore: registered new interface driver rtk_btusb
[ 167.384087] usb 3-1: This Realtek USB WiFi dongle (0x0bda:0xb720) is untested!
[ 167.384100] usb 3-1: Please report results to Jes.Sorensen@gmail.com
[ 167.593523] usb 3-1: Vendor: Realtek
[ 167.593533] usb 3-1: Product: 802.11n WLAN Adapter
.....
[ 167.594031] usb 3-1: RTL8723BU rev E (SMIC) 1T1R, TX queues 3, WiFi=1, BT=1,
GPS=0, HI PA=0
[ 167.594041] usb 3-1: RTL8723BU MAC: 00:13:ef:f4:58:ae
[ 167.594052] usb 3-1: rtl8xxxu: Loading firmware rtlwifi/rtl8723bu_nic.bin
[ 167.599402] usb 3-1: Firmware revision 35.0 (signature 0x5301)
[ 168.488312] usbcore: registered new interface driver rtl8xxxu
```

4) 然后通过 ifconfig 命令可以看到 RTL8723BU WIFI 的设备节点, WIFI 的连接和测试方法请参看 [WIFI 连接测试](#)一节, 这里不再赘述

```
root@orangeipi:~# ifconfig wlx0013eff458ae
wlx0013eff458ae: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
          ether 00:13:ef:f4:58:ae  txqueuelen 1000  (Ethernet)
          RX packets 0  bytes 0 (0.0 B)
          RX errors 0  dropped 0  overruns 0  frame 0
          TX packets 0  bytes 0 (0.0 B)
```



```
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

5) 然后通过 **hciconfig** 命令可以看到两个蓝牙设备，其中 Bus 类型为 USB 的节点就是 RTL8723BU 的蓝牙节点，蓝牙的测试方法请参看[蓝牙使用方法](#)一节，这里不再赘述

```
root@orangeipi:~# hciconfig
hci1:  Type: Primary  Bus: USB
        BD Address: 00:13:EF:F4:58:AE  ACL MTU: 1021:8  SCO MTU: 255:16
        UP RUNNING PSCAN ISCAN
        RX bytes:3994 acl:0 sco:0 events:206 errors:0
        TX bytes:29459 acl:0 sco:0 commands:173 errors:0

hci0:  Type: Primary  Bus: UART
        BD Address: 10:11:12:13:14:15  ACL MTU: 1021:8  SCO MTU: 240:3
        UP RUNNING
        RX bytes:2981 acl:0 sco:0 events:127 errors:0
        TX bytes:5423 acl:0 sco:0 commands:61 errors:0
```

3. 21. 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

```
root@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

```
root@orangepi:~# 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

```
root@orangepi:~# 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. 22. 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

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

3) 通过 v4l2-ctl (注意 v4l2 中的 l 是小写字母 l, 不是数字 1) 命令可以看到 USB 摄像头的设备节点信息为 /dev/video0

```
root@orangeipi:~# apt update
root@orangeipi:~# apt install -y v4l-utils
root@orangeipi:~# v4l2-ctl --list-devices
USB 2.0 Camera (usb-sunxi-ehci-1):
/dev/video0
```

4) 使用 fswebcam 测试 USB 摄像头

a. 安装 fswebcam

```
root@orangeipi:~# apt update
root@orangeipi:~# 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

```
root@orangeipi:~# 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

```
root@orangeipi:~# 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 motion 测试 USB 摄像头

a. 安装摄像头测试软件 motion

```
root@orangeipi:~# apt update
root@orangeipi:~# apt install -y motion
```



- b. 修改 `/etc/default/motion` 的配置，将 `start_motion_daemon=no` 修改为 `start_motion_daemon=yes`

```
root@orangepi:~# sed -i "s/start_motion_daemon=no/start_motion_daemon=yes/" \
/etc/default/motion      (这是一条命令)
```

- c. 修改 `/etc/motion/motion.conf` 的配置

```
root@orangepi:~# sed -i "s/stream_localhost on/stream_localhost off/" \
/etc/motion/motion.conf      (这是一条命令)
```

- d. 然后重启 motion 服务

```
root@orangepi:~# /etc/init.d/motion restart
[ ok ] Restarting motion (via systemctl): motion.service.
```

- e. 使用 motion 前请先确保 Orange Pi 开发板能正常连接网络，然后通过 ifconfig 命令获取开发板的 IP 地址

- f. 然后在和开发板同一局域网的 Ubuntu PC 或者 Windows PC 或者手机的火狐浏览器中输入【开发板的 IP 地址:8081】就能看到摄像头输出的视频了



6) 使用 mjpg-streamer 测试 USB 摄像头

- a. 下载 mjpg-streamer

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

- b. 安装依赖的软件包

```
root@orangepi:~# apt-get install -y cmake libjpeg8-dev
```

- c. 编译安装 mjpg-streamer

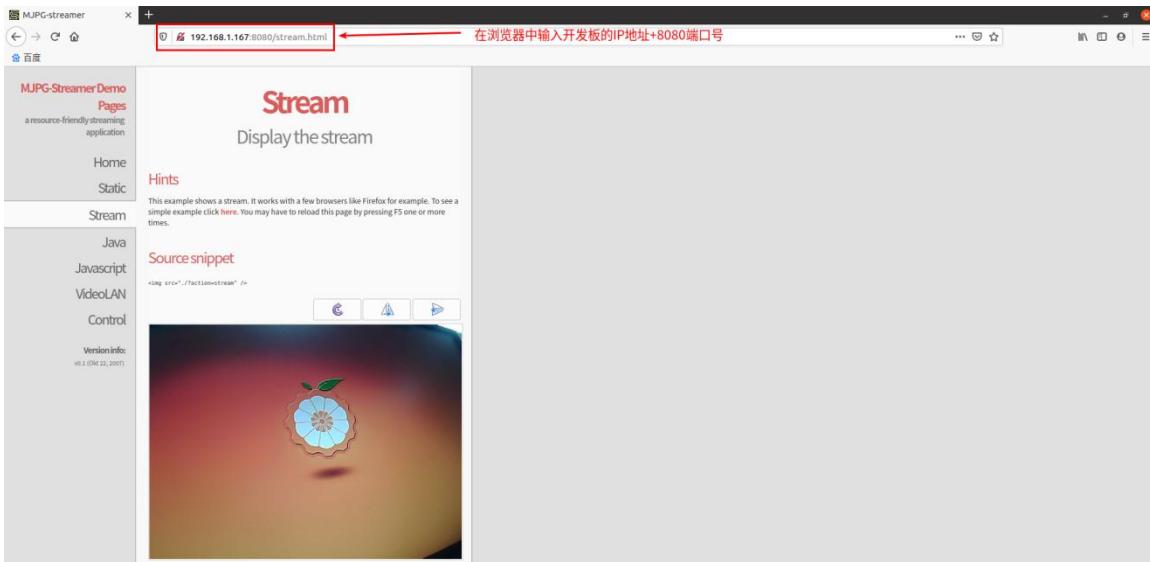
```
root@orangepi:~# cd mjpg-streamer/mjpg-streamer-experimental
root@orangepi:~/mjpg-streamer/mjpg-streamer-experimental# make
root@orangepi:~/mjpg-streamer/mjpg-streamer-experimental# make install
```



d. 然后输入下面的命令启动 mjpg_streamer

```
root@orangepi:~/mjpg-streamer/mjpg-streamer-experimental# export  
LD_LIBRARY_PATH=.          (这是一条命令)  
root@orangepi:~/mjpg-streamer/mjpg-streamer-experimental# ./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



f. 推荐使用 mjpg-streamer 来测试 USB 摄像头，比 motion 流畅很多，使用 mjpg-streamer 感觉不到任何卡顿

3. 23. 虚拟 USB 网卡测试

注意：此方法只适用于 linux4.9 内核的系统

1) 首先需要使用 USB Type C 线将开发板连接到电脑的 USB 接口中，在这种情况下是由电脑的 USB 接口给开发板供电的，**所以需要确保电脑的 USB 接口能提供足够的功率驱动开发板**，如果开发板启动有问题，则需要更换 USB 接口或者电脑

2) linux 系统默认配置 USB0 为 **usb_device** 模式，可以通过下面的命令来查看 **otg_role** 的状态

```
root@orangepi:~# cat /sys/devices/platform/soc/usbc0/otg_role  
usb_device
```



3) 如果 otg_role 没有设置为 usb_device 模式，可以使用下面的命令打开

```
root@orangepi:~# cat /sys/devices/platform/soc/usbc0/usb_device  
device_chose finished!
```

4) 然后加载 g_ether 内核模块

```
root@orangepi:~# modprobe g_ether
```

5) 加载完内核模块后开发板就会多出一个名为 usb0 的网络接口，首先使能这个网络接口，然后给其设置 IP 地址

```
root@orangepi:~# ifconfig usb0 up  
root@orangepi:~# ifconfig usb0 192.168.10.10  
root@orangepi:~# ifconfig usb0  
usb0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500  
          inet 192.168.10.10  netmask 255.255.255.0  broadcast 192.168.10.255  
            inet6 fe80::180e:b0ff:fe5a:1ee4  prefixlen 64  scopeid 0x20<link>  
              ether 1a:0e:b0:5a:1e:e4  txqueuelen 1000  (Ethernet)  
                RX packets 47  bytes 8223 (8.2 KB)  
                RX errors 0  dropped 0  overruns 0  frame 0  
                TX packets 47  bytes 8223 (8.2 KB)  
                TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

6) 然后回到 Ubuntu PC 中，通过 dmesg 命令可以看到下面的 log 信息，从中可以得知 USB 虚拟网卡的设备名为 **enxaaf52849335**

```
test@test:~$ dmesg | tail  
[33055.681514] usb 2-1.2: new high-speed USB device number 17 using ehci-pci  
[33055.791512] usb 2-1.2: New USB device found, idVendor=0525, idProduct=a4a2,  
bcdDevice= 4.09  
[33055.791515] usb 2-1.2: New USB device strings: Mfr=1, Product=2, SerialNumber=0  
[33055.791516] usb 2-1.2: Product: RNDIS/Ethernet Gadget  
[33055.791517] usb 2-1.2: Manufacturer: Linux 4.9.170-sun50iw9 with sunxi_usb_udc  
[33055.792258] cdc_subset: probe of 2-1.2:1.0 failed with error -22  
[33055.793063] cdc_ether 2-1.2:1.0 usb0: register 'cdc_ether' at usb-0000:00:1d.0-1.2,  
CDC Ethernet Device, aa:fd:52:84:93:35  
[33055.862338] cdc_ether 2-1.2:1.0 enxaaf52849335: renamed from usb0
```



7) 然后在 Ubuntu PC 中给 USB 虚拟网卡分配 IP 地址, Ubuntu PC 的 IP 地址请和开发板的 IP 保持在同一网段内

```
test@test:~$ sudo ifconfig enxaaf52849335 192.168.10.12
test@test:~$ ifconfig enxaaf52849335
enxaaf52849335: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>      mtu
1500
          inet 192.168.10.12  netmask 255.255.255.0  broadcast 192.168.10.255
                  ether aa:fd:52:84:93:35  txqueuelen 1000  (以太网)
                  RX packets 56  bytes 8542 (8.5 KB)
                  RX errors 0  dropped 0  overruns 0  frame 0
                  TX packets 113  bytes 21351 (21.3 KB)
                  TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

8) 然后就可以测试开发板和 Ubuntu PC 是否可以相互通信

```
root@orangepi:~# ping 192.168.10.12
PING 192.168.10.12 (192.168.10.12) 56(84) bytes of data.
64 bytes from 192.168.10.12: icmp_seq=1 ttl=64 time=0.376 ms
64 bytes from 192.168.10.12: icmp_seq=2 ttl=64 time=0.549 ms
64 bytes from 192.168.10.12: icmp_seq=3 ttl=64 time=0.460 ms
64 bytes from 192.168.10.12: icmp_seq=4 ttl=64 time=0.460 ms
64 bytes from 192.168.10.12: icmp_seq=5 ttl=64 time=0.493 ms
^C
--- 192.168.10.12 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4045ms
rtt min/avg/max/mdev = 0.376/0.467/0.549/0.056 ms
root@orangepi:~#
```

3.24. 虚拟串口测试

注意：此方法只适用于 linux4.9 内核的系统

1) 首先需要使用 USB Type C 线将开发板连接到电脑的 USB 接口中, 在这种情况下是由电脑的 USB 接口给开发板供电的, **所以需要确保电脑的 USB 接口能提供足够的功率驱动开发板**, 如果开发板启动有问题, 则需要更换 USB 接口或者电脑



2) linux 系统默认配置 USB0 为 **usb_device** 模式，可以通过下面的命令来查看 **otg_role** 的状态

```
root@orangeipi:~# cat /sys/devices/platform/soc/usbc0/otg_role  
usb_device
```

3) 如果 **otg_role** 没有设置为 **usb_device** 模式，可以使用下面的命令打开

```
root@orangeipi:~# cat /sys/devices/platform/soc/usbc0/usb_device  
device_chose finished!
```

4) 然后加载 **g_serial** 内核模块

```
root@orangeipi:~# modprobe g_serial
```

5) 加载完内核模块后开发板 linux 系统的 **/dev** 下面就会多出一个名为 **ttyGS0** 的设备节点

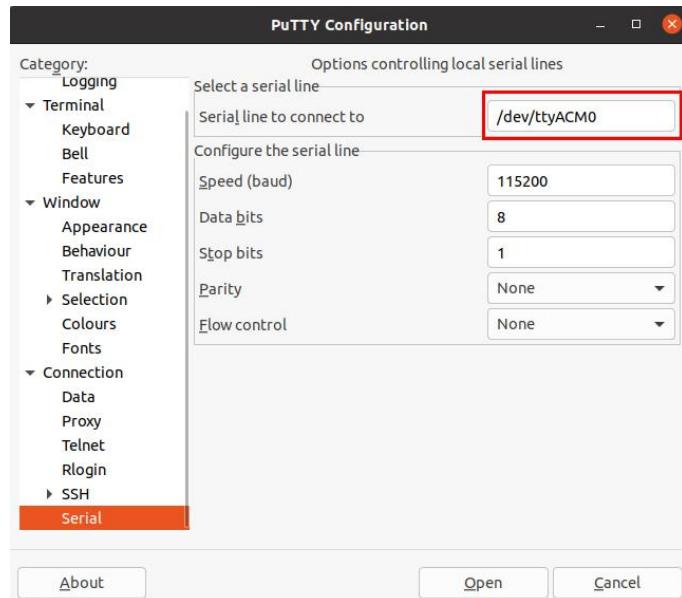
```
root@orangeipi:~# ls /dev/ttyGS*  
/dev/ttyGS0
```

6) 然后回到 Ubuntu PC，可以看到 **/dev** 下面会多出了一个名为 **ttyACM0** 的设备节点

```
test@test:~$ ls /dev/ttyACM*  
/dev/ttyACM0
```

7) 然后打开 Ubuntu PC 的 **putty**，连接 **ttyACM0**

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



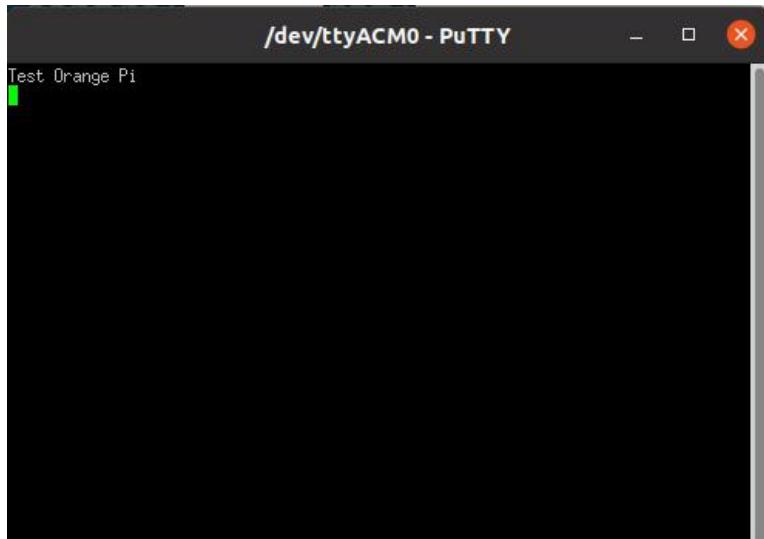
Putty 连接好 ttyACM0 后会打开下图所示的窗口



8) 然后回到开发板的 linux 系统，给 /dev/ttys0 发送一串字符

```
root@orangeipi:~# echo "Test Orange Pi" > /dev/ttys0
```

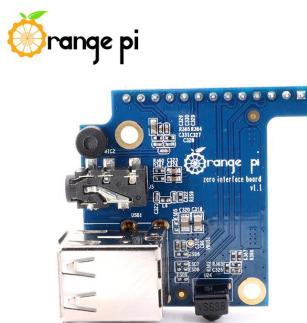
9) 如果一切正常，Ubuntu PC 的 Putty 就会收到开发板发送过来的字符串



3.25. 音频测试

3.25.1. 耳机接口播放音频测试

- 1) 首先需要将 13pin 扩展板插入到 Orange Pi 开发板的 13pin 接口中，然后在音频接口中插入耳机



- 2) 通过 **aplay -l** 命令可以查看 linux 系统支持的声卡设备
 - a. linux4.9 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@orangepi:~# aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: audiocodec [audiocodec], device 0: SUNXI-CODEC sun50iw9-codec-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```



- b. linux5.13 系统的输出如下所示，其中 **H616 Audio Codec** 就是耳机播放需要的声卡设备

```
root@orangepi:~# aplay -l
card 2: Codec [H616 Audio Codec], device 0: CDC PCM Codec-0 [CDC PCM Codec-0]
  Subdevices: 0/1
  Subdevice #0: subdevice #0
```

- 3) 上传需要播放的音频文件到 linux 系统的/root 文件夹中，在 Ubuntu PC 中可以使用 scp 命令上传(命令中的 IP 地址为 Orange Pi 开发板的 IP 地址)，或者使用 U 盘拷贝

```
test@test:~/AudioTest$ scp audio.wav  root@192.168.1.xx:/root  (根据实际情况修改 IP 地址和路径)
```

- 4) 然后使用 aplay 命令播放音频，耳机就能听到声音了

- a. Linux4.9 的播放命令如下所示

```
root@orangepi:~# aplay -D hw:0,0 audio.wav
Playing WAVE 'audio.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
```

- b. Linux5.13 的播放命令如下所示

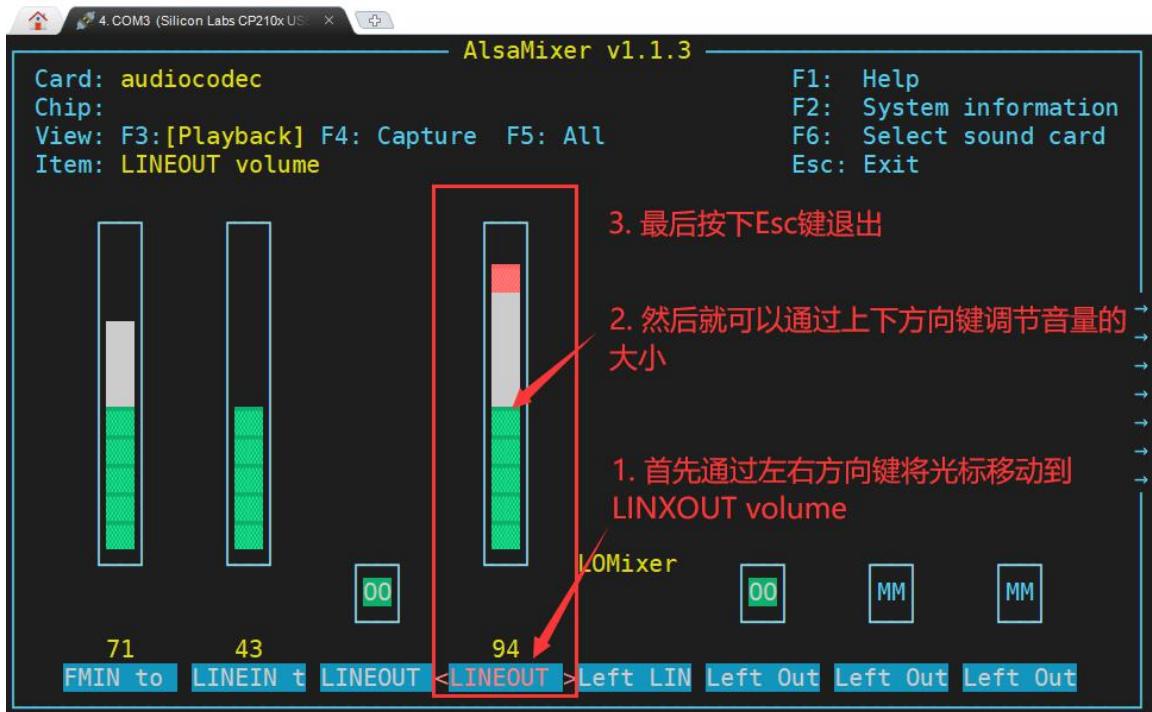
```
root@orangepi:~# aplay -D hw:2,0 audio.wav
Playing WAVE 'audio.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
```

- 5) 通过 alsamixer 命令可以调节耳机的音量

- a. 在终端中输入 alsamixer 命令

```
root@orangepi:~# alsamixer
```

- b. 输入 alsamixer 命令后会弹出下面的音频设置界面，然后可以通过上下左右的方向键调节音频的大小，具体步骤见下图的说明



3.25.2. HDMI 音频播放测试

1) 首先使用 Micro 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) 上传需要播放的音频文件到 linux 系统的/root 文件夹中，在 Ubuntu PC 中可以使用 scp 命令上传(命令中的 IP 地址为 Orange Pi 开发板的 IP 地址)，或者使用 U 盘拷贝

```
test@test:~/AudioTest$ scp audio.wav root@192.168.1.xx:/root (根据实际情况修改 IP 地址和路径)
```

3) HDMI 音频播放无需其他设置，直接使用 aplay 命令播放即可

a. Linux4.9 系统的播放命令如下所示

```
root@orangeipi:~# aplay -D hw:1,0 audio.wav
```

b. Linux5.13 系统的播放命令如下所示

```
root@orangeipi:~# aplay -D hw:0,0 audio.wav
```



3.26. 红外接收测试

- 首先需要将 13pin 扩展板插入到 Orange Pi 开发板的 13pin 接口中, 插入扩展板后, Orange Pi Zero 2 才能使用红外接收功能



- 安装 ir-keytable 红外测试软件

```
root@orangepi:~# apt update  
root@orangepi:~# apt-get install -y ir-keytable
```

- 然后执行 ir-keytable 可以查看红外设备的信息

- linux4.9 系统输出如下所示

```
root@orangepi:~# ir-keytable  
Found /sys/class/rc/rc0/ (/dev/input/event1) with:  
    Driver: sunxi-rc-recv, table: rc_map_sunxi  
    lirc device: /dev/lirc0  
    Supported protocols: lirc nec  
    Enabled protocols: lirc nec  
    Name: sunxi_ir_recv  
    bus: 25, vendor/product: 0001:0001, version: 0x0100  
    Repeat delay = 500 ms, repeat period = 125 ms
```

- linux5.13 系统的输出如下所示

```
root@orangepi:~# ir-keytable  
Found /sys/class/rc/rc0/ with:  
    Name: sunxi-ir  
    Driver: sunxi-ir  
    Default keymap: rc-empty  
    Input device: /dev/input/event4  
    LIRC device: /dev/lirc0
```



```
Attached BPF protocols: Operation not supported
```

```
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. linux4.9 系统输出如下所示

```
root@orangeipi:/# 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. linux5.13 系统输出如下所示

```
root@orangepizero2:~# 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).
```



3.27. 温度传感器

1) H616 总共有 4 个温度传感器，查看温度的命令如下

a. sensor0: CPU

```
root@orangepi:~# cat /sys/class/thermal/thermal_zone0/type  
cpu_thermal_zone  
root@orangepi:~# cat /sys/class/thermal/thermal_zone0/temp  
57734
```

b. sensor1: GPU

```
root@orangepi:~# cat /sys/class/thermal/thermal_zone1/type  
gpu_thermal_zone  
root@orangepi:~# cat /sys/class/thermal/thermal_zone1/temp  
57410
```

c. sensor2: VE

```
root@orangepi:~# cat /sys/class/thermal/thermal_zone2/type  
ve_thermal_zone  
root@orangepi:~# cat /sys/class/thermal/thermal_zone2/temp  
59273
```

d. sensor3: DDR

```
root@orangepi:~# cat /sys/class/thermal/thermal_zone3/type  
ddr_thermal_zone  
root@orangepi:~# cat /sys/class/thermal/thermal_zone3/temp  
58949
```

3.28. 安装 Docker 的方法

1) 首先安装下面的软件包

```
root@orangepi:~# apt update  
root@orangepi:~# apt install -y apt-transport-https ca-certificates curl \  
software-properties-common
```

2) 再添加阿里云 docker 的密钥

```
root@orangepi:~# curl -fsSL http://mirrors.aliyun.com/docker-ce/linux/ubuntu/gpg \  
| sudo apt-key add -
```



3) 在 ubuntu 的系统源中添加对应的 docker 源

```
root@orangepi:~# add-apt-repository "deb [arch=arm64] \
https://mirrors.aliyun.com/docker-ce/linux/ubuntu ${lsb_release -cs} stable"
```

4) 安装最新版本的 docker-ce

```
root@orangepi:~# apt update
root@orangepi:~# apt install -y docker-ce
```

5) 验证 docker 的状态

```
root@orangepi:~# systemctl status docker
● docker.service - Docker Application Container Engine
  Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2020-08-24 10:29:22 UTC; 26min ago
    Docs: https://docs.docker.com
   Main PID: 3145 (dockerd)
     Tasks: 15
    CGroup: /system.slice/docker.service
            └─3145 /usr/bin/dockerd -H fd://
--containerd=/run/containerd/containerd.sock
```

6) 测试 docker

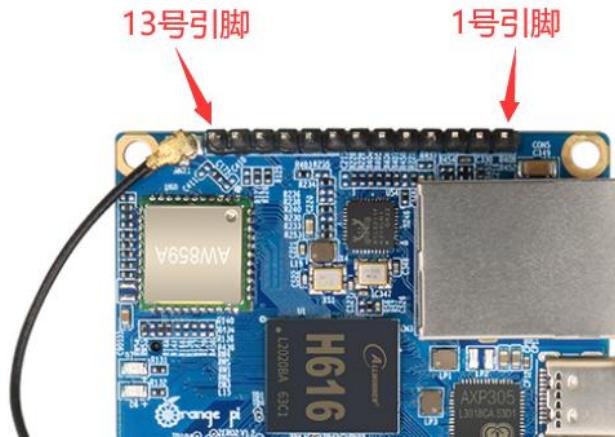
```
root@orangepi:~# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
256ab8fe8778: Pull complete
Digest:
sha256:7f0a9f93b4aa3022c3a4c147a449ef11e0941a1fd0bf4a8e6c9408b2600777c5
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

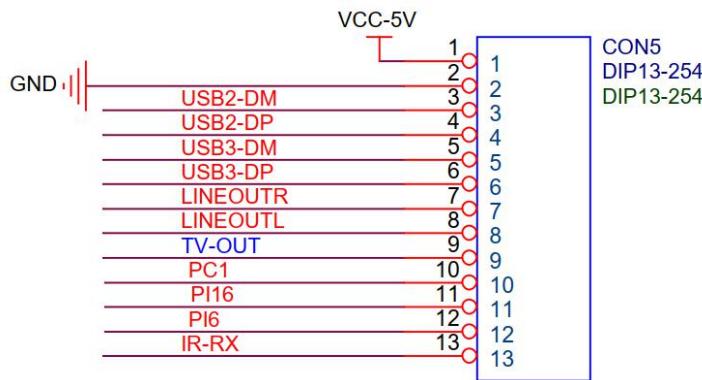


3.29. 13 Pin 扩展板接口引脚说明

1) Orange Pi Zero 2 开发板 13 pin 扩展板接口引脚的顺序请参考下图



2) Orange Pi Zero 2 开发板 13pin 接口的原理图如下所示



3) Orange Pi Zero 2 开发板 13 pin 扩展板接口引脚的功能说明如下

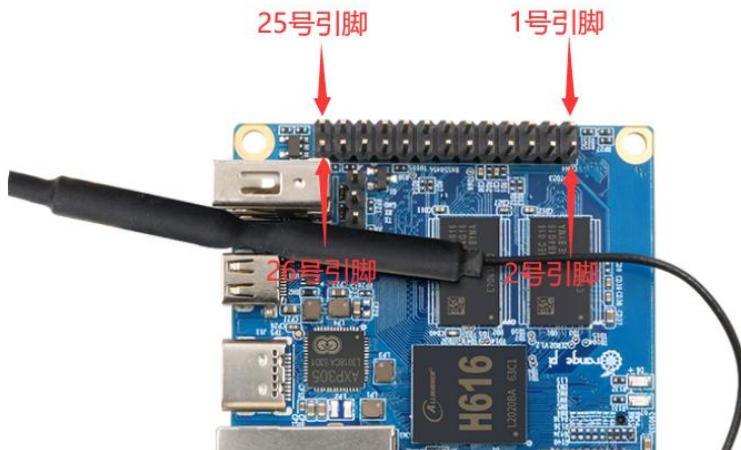
- 13pin 引脚接扩展板时，可以额外提供
 - 2 个 USB 2.0 Host
 - 耳机左右声道音频输出
 - TV-OUT 视频输出
 - 红外接收功能
 - 接了扩展板后 13pin 接口的 10、11 和 12 号引脚就无法使用了
 - 另外需要注意 13pin 扩展板上的 MIC 在 Orange Pi Zero 2 上是无法使用的
- 13pin 引脚不接扩展板时，10、11、12 和 13 号引脚可当作普通 GPIO 口来使用



GPIO序号	功能	引脚
	5V	1
	GND	2
	USB2-DM	3
	USB2-DP	4
	USB3-DM	5
	USB3-DP	6
	LINEOUTR	7
	LINEOUTL	8
	TV-OUT	9
65	PC1	10
272	PI16	11
262	PI6	12
234	IR-RX/PH10	13

3.30. 26 Pin 接口引脚说明

1) Orange Pi Zero 2 开发板 26 pin 接口引脚的顺序请参考下图



2) Orange Pi Zero 2 开发板 26 pin 接口引脚的功能如下表所示

GPIO序号	GPIO	功能	引脚
		3.3V	1
229	PH5	TWI3-SDA	3
228	PH4	TWI3-SCK	5

引脚	功能	GPIO	GPIO序号
2	5V		
4	5V		
6	GND		



73	PC9	PC9	7
		GND	9
70	PC6	PC6	11
69	PC5	PC5	13
72	PC8	PC8	15
		3.3V	17
231	PH7	SPI1_MOSI	19
232	PH8	SPI1_MISO	21
230	PH6	SPI1_CLK	23
		GND	25

8	UART5_TX	PH2	226
10	UART5_RX	PH3	227
12	PC11	PC11	75
14	GND		
16	PC15	PC15	79
18	PC14	PC14	78
20	GND		
22	PC7	PC7	71
24	SPI1_CS	PH9	233
26	PC10	PC10	74

3.31. 安装 wiringOP 的方法

1) 下载 wiringOP 的代码

```
root@orangepi:~# apt update
root@orangepi:~# apt install -y git
root@orangepi:~# git clone https://github.com/orangepi-xunlong/wiringOP
```

2) 编译安装 wiringOP

```
root@orangepi:~# cd wiringOP
root@orangepi:~/wiringOP# ./build clean
root@orangepi:~/wiringOP# ./build
```

3) 测试 gpio readall 命令的输出如下

- a. 其中 1 到 26 号引脚与开发板上的 26 Pin 引脚是一一对应的
- b. 27 号引脚对应开发板上 13pin 的 10 号引脚
- c. 29 号引脚对应开发板上 13pin 的 11 号引脚
- d. 31 号引脚对应开发板上 13pin 的 12 号引脚
- e. 33 号引脚对应开发板上 13pin 的 13 号引脚
- f. **28、30、32、34 号引脚为空，请直接忽略**



root@orangepizero2: ~# gpio readall												
GPIO	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	GPIO	Zero 2	
		3.3V			1	2		5V				
229	0	SDA.3	OFF	0	3	4		5V				
228	1	SCL.3	OFF	0	5	6		GND				
73	2	PC9	OUT	0	7	8	0	OUT	TXD.5	3	226	
		GND			9	10	0	OUT	RXD.5	4	227	
70	5	PC6	OUT	0	11	12	0	OUT	PC11	6	75	
69	7	PC5	OUT	0	13	14		GND				
72	8	PC8	OUT	1	15	16	0	OUT	PC15	9	79	
		3.3V			17	18	0	OUT	PC14	10	78	
231	11	MOSI.1	OUT	0	19	20		GND				
232	12	MISO.1	OUT	0	21	22	0	OUT	PC7	13	71	
230	14	SCLK.1	OUT	0	23	24	0	OUT	CE.1	15	233	
		GND			25	26	0	OUT	PC10	16	74	
65	17	PC1	OUT	0	27	28						
272	18	PI16	OUT	0	29	30						
262	19	PI6	OUT	0	31	32						
234	20	PH10	OUT	0	33	34						
GPIO	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	GPIO	Zero 2	

3.32. 26pin 接口 GPIO、I2C、UART、SPI 测试

wiringOP 已适配 Orange Pi 开发板，使用 wiringOP 可以测试 GPIO、I2C、UART 和 SPI 的功能。

开始测试前，请确保已经参照[安装 wiringOP](#)一节编译安装好了 wiringOP

3.32.1. 26pin GPIO 口测试

- 1) 下面以 7 号引脚——对应 GPIO 为 PC9 ——对应 wPi 序号为 2——为例演示如何设置 GPIO 口的高低电平

root@orangepizero2: ~/wiringOP# gpio readall												
GPIO	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	GPIO	Zero 2	
		3.3V			1	2		5V				
229	0	SDA.3	OUT	1	3	4		5V				
228	1	SCL.3	OUT	1	5	6		GND				
73	2	PC9	OUT	1	7	8	1	OUT	TXD.5	3	226	
		GND			9	10	1	OUT	RXD.5	4	227	
70	5	PC6	OUT	1	11	12	1	OUT	PC11	6	75	

- 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
```

使用 gpio readall 可以看到 7 号引脚的值(V)变为了 0

```
root@orangepizero2: # gpio readall
+-----+-----+-----+-----+-----+-----+-----+
| GPIO | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | GPIO |
+-----+-----+-----+-----+-----+-----+-----+
| 229 | 0 | SDA.3 | OFF | 0 | 3 | 4 | | 5V | | | |
| 228 | 1 | SCL.3 | OFF | 0 | 5 | 6 | | 5V | | |
| 73 | 2 | PC9 | OUT | 0 | 7 | 8 | 0 | ALT2 | TXD.5 | 3 | 226 |
| 70 | 5 | GND | | | 9 | 10 | 0 | ALT2 | RXD.5 | 4 | 227 |
| 70 | 5 | PC6 | ALT5 | 0 | 11 | 12 | 0 | OFF | PC11 | 6 | 75 |
+-----+-----+-----+-----+-----+-----+-----+
```

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
```

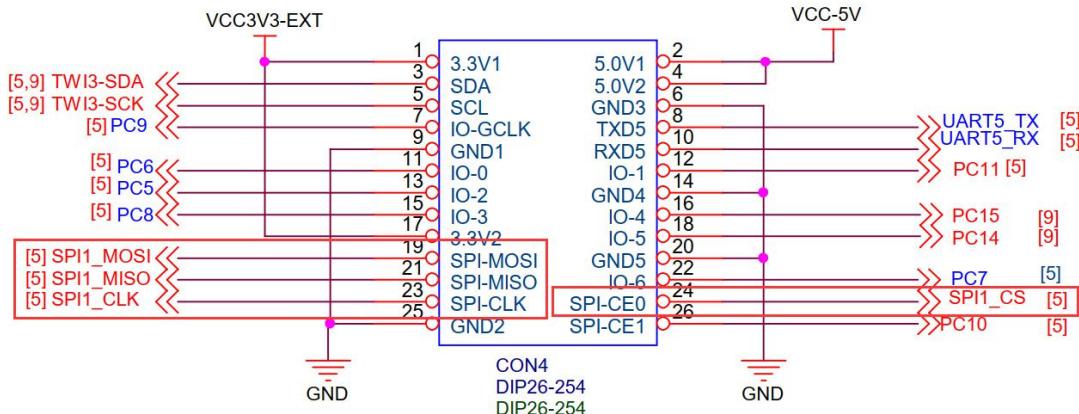
使用 gpio readall 可以看到 7 号引脚的值(V)变为了 1

```
root@orangepizero2: # gpio readall
+-----+-----+-----+-----+-----+-----+-----+
| GPIO | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | GPIO |
+-----+-----+-----+-----+-----+-----+-----+
| 229 | 0 | SDA.3 | OFF | 0 | 3 | 4 | | 5V | | | |
| 228 | 1 | SCL.3 | OFF | 0 | 5 | 6 | | 5V | | |
| 73 | 2 | PC9 | OUT | 1 | 7 | 8 | 0 | ALT2 | TXD.5 | 3 | 226 |
| 70 | 5 | GND | | | 9 | 10 | 0 | ALT2 | RXD.5 | 4 | 227 |
| 70 | 5 | PC6 | ALT5 | 0 | 11 | 12 | 0 | OFF | PC11 | 6 | 75 |
+-----+-----+-----+-----+-----+-----+-----+
```

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

3.32.2. 26pin SPI 测试

1) 由 26pin 接口的原理图可知, Orange Pi Zero 2 可用的 spi 为 spi1



2) 先查看下 linux 系统中是否存在 **spidev1.x** (**x 可能为 0 或者 1**) 的设备节点，如果存在，说明 SPI1 已经设置好了，可以直接使用

```
root@orangepi:~/wiringOP/examples# ls /dev/spidev1*
```

/dev/spidev1.x

3) 再在 wiringOP 的 examples 中编译 spidev_test 测试程序

```
root@orangepi:~/wiringOP/examples# make spidev_test
```

[CC] spidev_test.c

[link]

4) 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

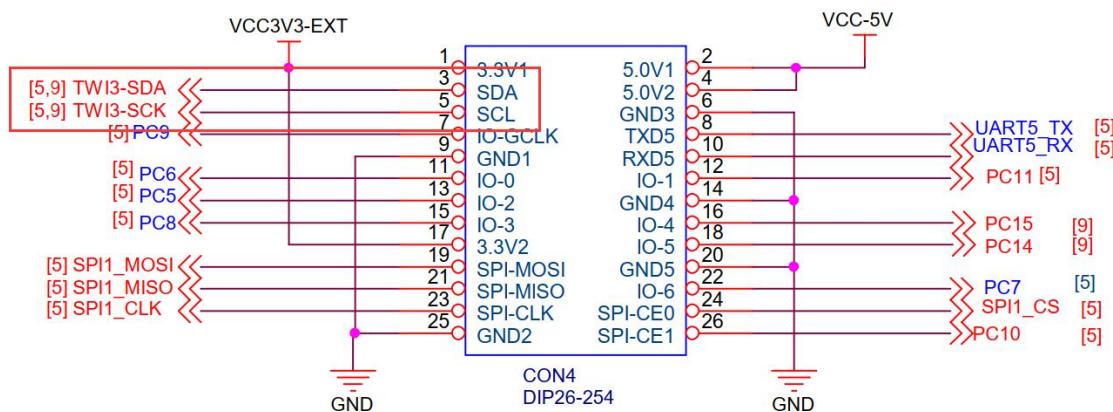
5) 然后短接 SPI1 的 mosi (26pin 接口中的第 19 号引脚) 和 miso (26pin 接口中的第 21 号引脚) 两个引脚再运行 spidev_test 的输出如下, 可以看到发送和接收的数据一样

```
root@orangepi:~/wiringOP/examples# ./spidev test -v -D /dev/spidev1.x
```



3. 32. 3. 26pin I2C 测试

1) 由 26pin 的原理图可知, Orange Pi Zero 2 可用的 i2c 为 i2c3



2) 启动 linux 系统后, 先确认下 /dev 下存在 i2c3 的设备节点

a. linux4.9 系统 i2c3 对应的设备节点为 /dev/i2c-3

```
root@orangepi:~# ls /dev/i2c-*
```

| **/dev/i2c-3** /dev/i2c-5

b. linux5.13 系统 i2c3 对应的设备节点为 /dev/i2c-1

```
root@orangepeizero2:~# ls /dev/i2c-*
```

/dev/i2c-0 /dev/i2c-1

3) 然后开始测试 i2c，首先安装 i2c-tools

root@orangepi:~# apt update

```
root@orangeipi:~# apt install -y i2c-tools
```

4) 然后在 26pin 接头的 i2c3 引脚上接一个 i2c 设备

	i2c3
sda 引脚	对应 3 号引脚



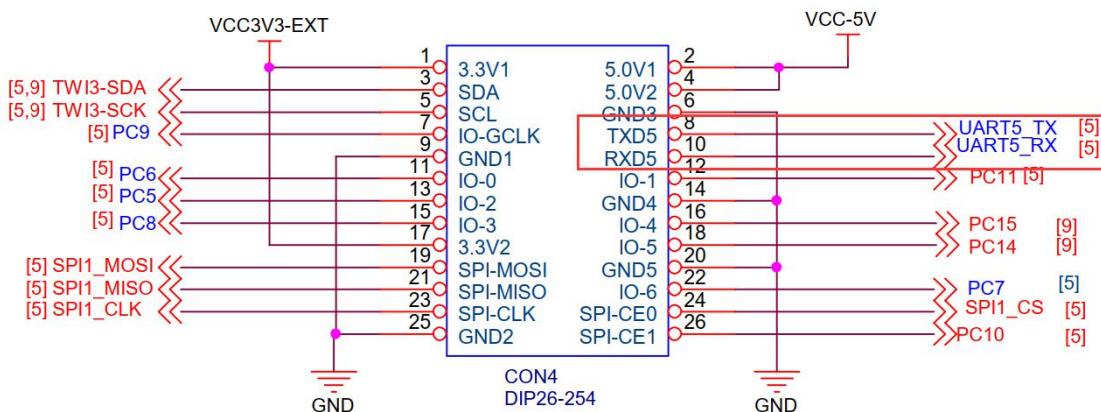
sck 引脚	对应 5 号引脚
vcc 引脚	对应 1 号引脚
gnd 引脚	对应 6 号引脚

5) 然后使用 **i2cdetect -y 3** 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（请注意 linux5.13 需要使用命令 **i2cdetect -y 1**）

```
root@orangepi:~# 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: --
```

3.32.4. 26pin 的 UART 测试

1) 由 26pin 接口的原理图可知，Orange Pi Zero 2 可用的 uart 为 uart5



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

```
root@orangepi:~# ls /dev/ttys5
/dev/ttys5
```

3) 然后开始测试 uart5 接口，先使用杜邦线短接要测试的 uart5 接口的 rx 和 tx





tx 引脚	对应 8 号引脚
rx 引脚	对应 10 号引脚

4) 然后修改 wiringOP 中串口测试程序 serialTest 打开的串口设备节点名为/**dev/ttyS5**

```
root@orangeipi:~/wiringOP/examples# vim serialTest.c
```

```
int main ()
{
    int fd ;
    int count ;
    unsigned int nextTime ;

    if ((fd = serialOpen ("/dev/ttyS5", 115200)) < 0)
    {
        fprintf (stderr, "Unable to open serial device: %s\n", strerror (errno)) ;
        return 1 ;
    }
```

5) 再编译 wiringOP 中的串口测试程序 serialTest

```
root@orangeipi:~/wiringOP/examples# make serialTest
```

```
[CC] serialTest.c
```

```
[link]
```

```
root@orangeipi:~/wiringOP/examples#
```

6) 最后运行 serialTest，如果能看到下面的打印，说明串口通信正常

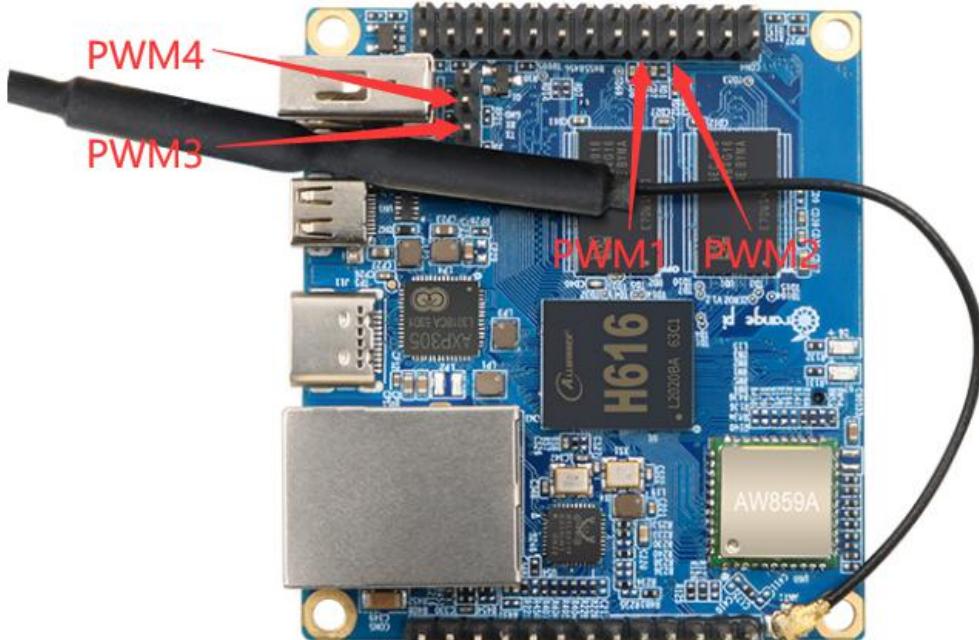
```
root@orangeipi:~/wiringOP/examples# ./serialTest
```

```
Out: 0: -> 0
Out: 1: -> 1
Out: 2: -> 2
Out: 3: -> 3
Out: 4: -> 4
Out: 5: -> 5
Out: 6: -> 6
Out: 7: -> 7
Out: 8: -> 8^C
```



3. 33. Linux4.9 PWM 的测试方法

- 1) Orange Pi Zero 2 最多可以使用 4 路 PWM，它们所在引脚的位置如下图所示
 - a. 其中 PWM1、PWM2 和 26pin 接口中 UART5 的 RX、TX 引脚是复用的，所以使用 PWM1 和 PWM2 前需要先在 dts 中将 UART5 的配置关掉
 - b. PWM3、PWM4 和调试串口中的 TX、RX 引脚是复用的，所以使用 PWM3 和 PWM4 前需要在 dts 中先将 UART0 的配置关掉，此时调试串口就无法使用了



- 2) PWM1 和 PWM2 测试方法如下所示

- a. 首先需要在 dts 中关掉 UART5 的配置，linux 系统中预装了一个名为 **orangeipi-add-overlay** 的脚本，通过这个脚本我们可以使用 DT overlay 来动态的修改 dts 中的配置。首先编写 `uart5_disable.dts` 文件，内容如下所示

```
root@orangeipi:~# cat uart5_disable.dts
/dts-v1/;
/plugin/;

{
    compatible = "allwinner,h616", "arm,sun50iw9p1";

    fragment@0 {
```



```
target = <&uart5>;
__overlay__ {
    status = "disabled";
};
};
```

- b. 然后就可以使用 **orangeipi-add-overlay** 将 uart5_disable.dts 编译成 uart5_disable.dtbo

```
root@orangeipi:~# orangeipi-add-overlay uart5_disable.dts
Compiling the overlay
Copying the compiled overlay file to /boot/overlay-user/
Reboot is required to apply the changes
```

- c. 然后重启 Linux 系统，在串口输出的 log 中可以看到下面的打印信息，说明 uart5_disable.dtbo 加载成功

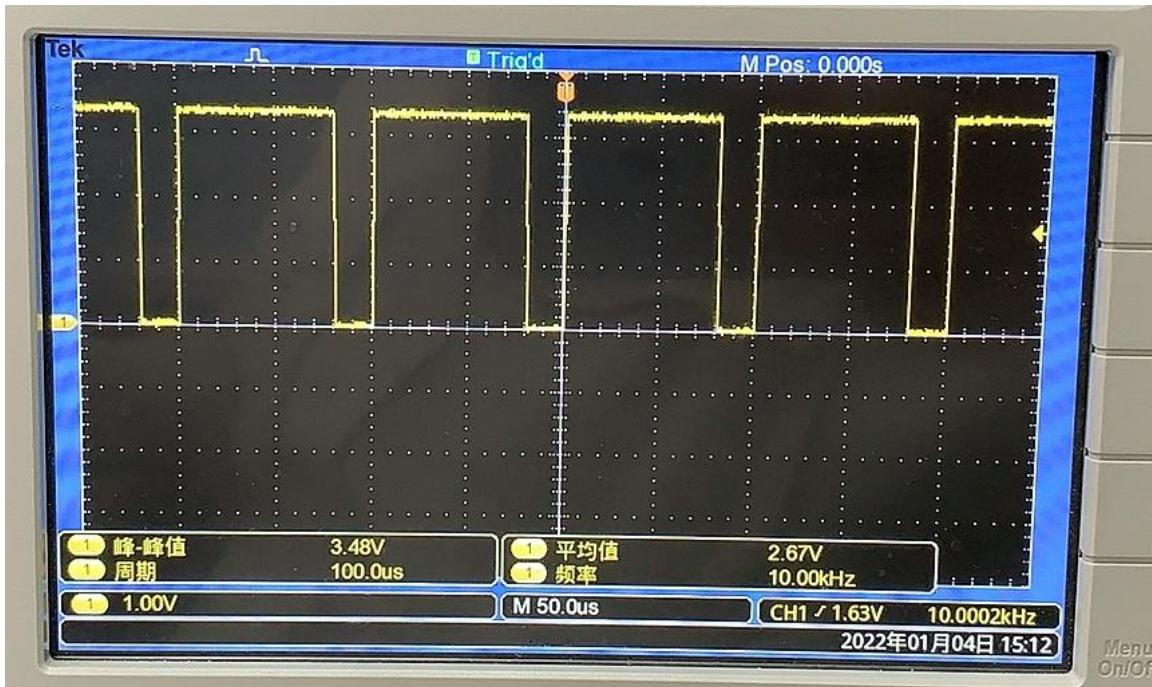
```
216 bytes read in 7 ms (29.3 KiB/s)
283 bytes read in 11 ms (24.4 KiB/s)
Applying user provided DT overlay uart5_disable.dtbo
8472451 bytes read in 367 ms (22 MiB/s)
24125512 bytes read in 1020 ms (22.6 MiB/s)
```

- d. 进入 Linux 系统后在/dev 下就看不到 ttyS5 这个设备节点了

```
root@orangeipi:~# ls /dev/ttys*
/dev/ttyS0  /dev/ttyS1
```

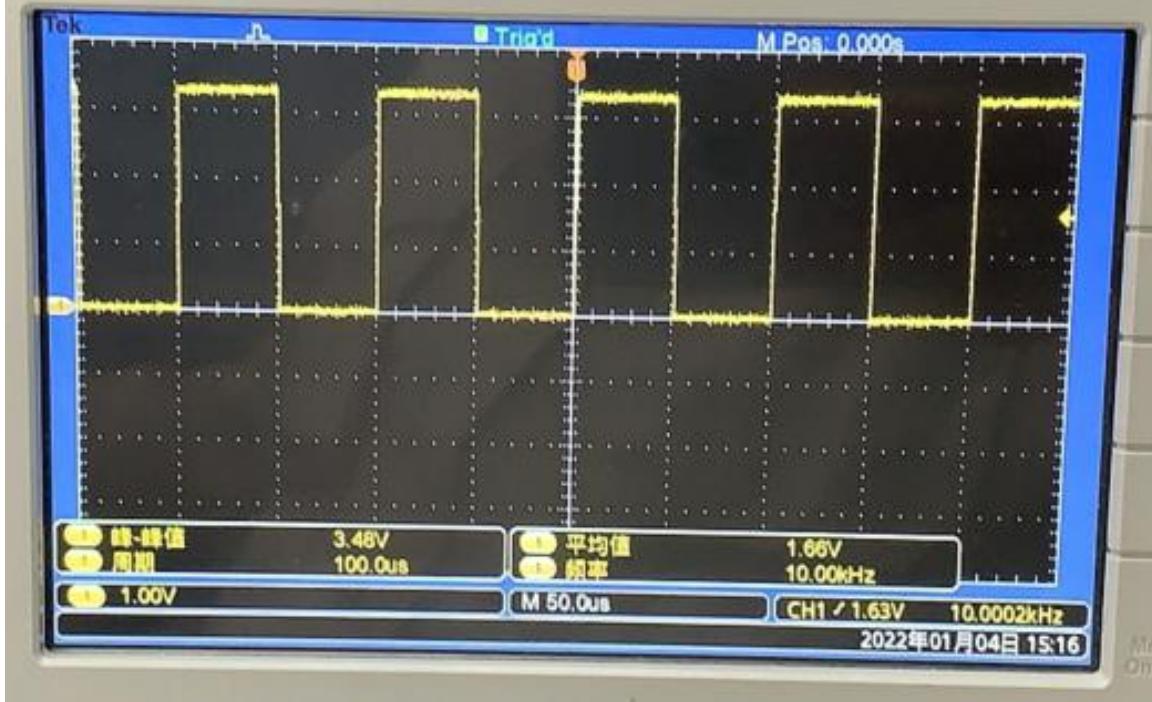
- e. 然后就可以开始 PWM 的测试，在 Linux4.9 系统中输入下面的命令可以让 pwm1 输出一个 10kHz 的矩形波

```
root@orangeipi:~# echo 1 > /sys/class/pwm/pwmchip0/export
root@orangeipi:~# echo 100000 > /sys/class/pwm/pwmchip0/pwm1/period
root@orangeipi:~# echo 20000 > /sys/class/pwm/pwmchip0/pwm1/duty_cycle
root@orangeipi:~# echo 1 > /sys/class/pwm/pwmchip0/pwm1/enable
```



f. 在 Linux4.9 系统中输入下面的命令可以让 pwm2 输出一个 10kHz 的方波

```
root@orangepi:~# echo 2 > /sys/class/pwm/pwmchip0/export
root@orangepi:~# echo 100000 > /sys/class/pwm/pwmchip0/pwm2/period
root@orangepi:~# echo 50000 > /sys/class/pwm/pwmchip0/pwm2/duty_cycle
root@orangepi:~# echo 1 > /sys/class/pwm/pwmchip0/pwm2/enable
```





3) PWM3 和 PWM4 测试方法如下所示

- 首先需要在 dts 中关掉 UART0 的配置，linux 系统中预装了一个名为 **orangeipi-add-overlay** 的脚本，通过这个脚本我们可以使用 DT overlay 来动态的修改 dts 中的配置。首先编写 `uart0_disable.dts` 文件，内容如下所示

```
root@orangepi:~# cat uart0_disable.dts
/dts-v1/;
/plugin/;

{
    compatible = "allwinner,h616", "arm,sun50iw9p1";

    fragment@0 {
        target = <&uart0>;
        __overlay__ {
            status = "disabled";
        };
    };
};

};
```

- 然后就可以使用 **orangeipi-add-overlay** 将 `uart0_disable.dts` 编译成 `uart0_disable.dtbo`

```
root@orangepi:~# orangeipi-add-overlay uart0_disable.dts
Compiling the overlay
Copying the compiled overlay file to /boot/overlay-user/
Reboot is required to apply the changes
```

- 然后重启 Linux 系统，在串口输出的 log 中可以看到下面的打印信息，说明 `uart0_disable.dtbo` 加载成功

```
216 bytes read in 7 ms (29.3 KiB/s)
283 bytes read in 12 ms (22.5 KiB/s)
Applying user provided DT overlay uart0_disable.dtbo
8472451 bytes read in 367 ms (22 MiB/s)
24125512 bytes read in 1020 ms (22.6 MiB/s)
```

- 进入 Linux 系统后在 `/dev` 下就看不到 `ttyS0` 这个设备节点了

```
root@orangepi:~# ls /dev/ttys*
/dev/ttyS1  /dev/ttyS5
```



e. 注意此时调试串口会卡在下面的地方，不会再有任何输出，也不能输入了

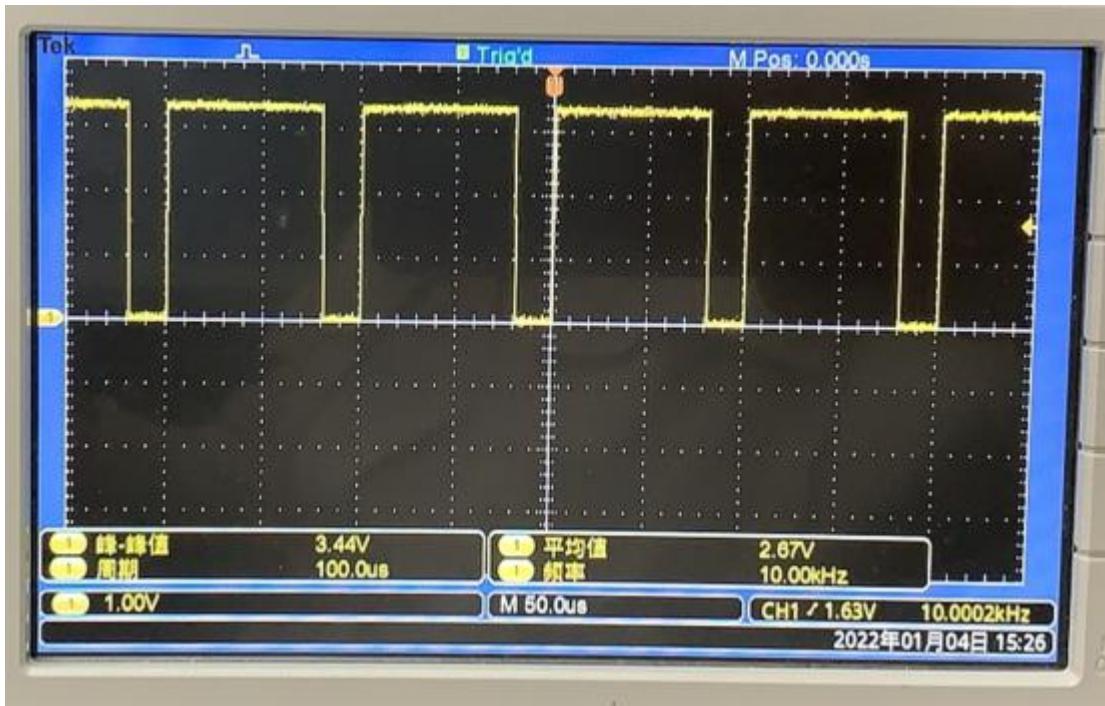
```
## Loading init Ramdisk from Legacy Image at 43300000 ...
Image Name: uInitrd
Image Type: ARM Linux RAMDisk Image (gzip compressed)
Data Size: 8472387 Bytes = 8.1 MiB
Load Address: 00000000
Entry Point: 00000000
Verifying Checksum ... OK
Loading Ramdisk to 497eb000, end 4ffff743 ... OK
reserving fdt memory region: addr=48000000 size=1000000
## Linux machid: 00000000, FDT addr: 7be88d60

Starting kernel ...

[
```

f. 然后就可以开始 PWM 的测试，在 Linux4.9 系统中输入下面的命令可以让 pwm3 输出一个 10kHz 的矩形波

```
root@orangepi:~# echo 3 > /sys/class/pwm/pwmchip0/export
root@orangepi:~# echo 100000 > /sys/class/pwm/pwmchip0/pwm3/period
root@orangepi:~# echo 20000 > /sys/class/pwm/pwmchip0/pwm3/duty_cycle
root@orangepi:~# echo 1 > /sys/class/pwm/pwmchip0/pwm3/enable
```

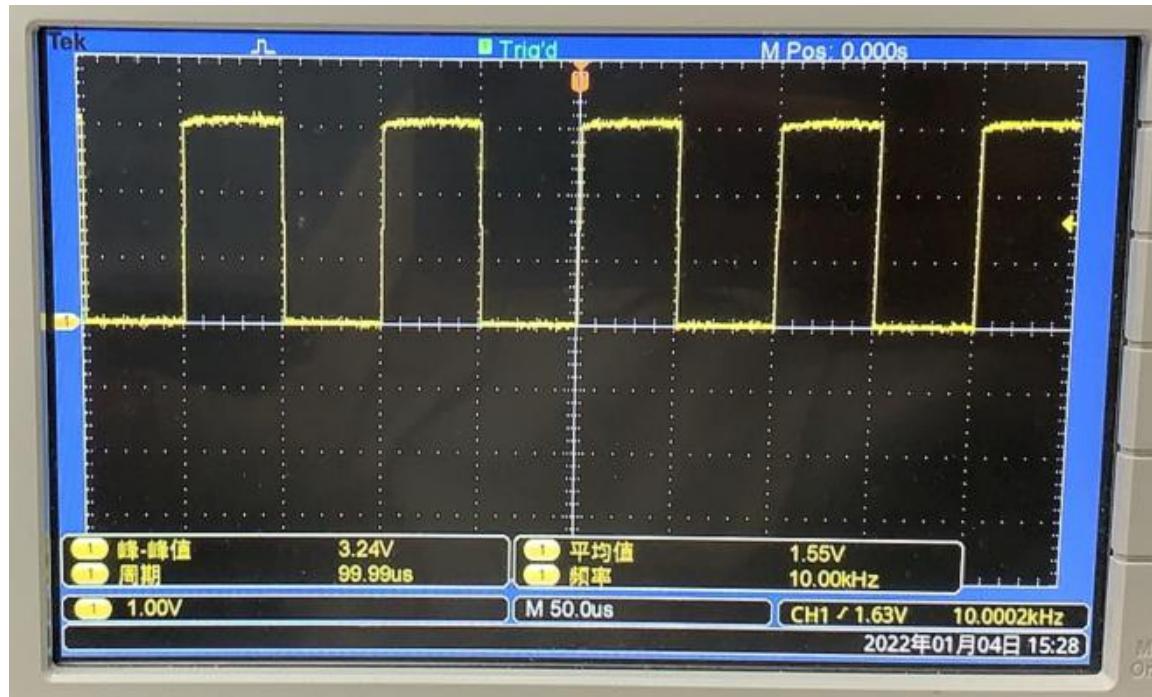


g. 在 Linux4.9 系统中输入下面的命令可以让 pwm2 输出一个 10kHz 的方波

```
root@orangepi:~# echo 4 > /sys/class/pwm/pwmchip0/export
root@orangepi:~# echo 100000 > /sys/class/pwm/pwmchip0/pwm4/period
root@orangepi:~# echo 50000 > /sys/class/pwm/pwmchip0/pwm4/duty_cycle
```



```
root@orangepi:~# echo 1 > /sys/class/pwm/pwmchip0/pwm4/enable
```



3.34. SPI LCD 显示屏使用方法

注意：此方法只适用于 linux4.9 内核的系统，Linux5.13 内核的系统没有适配

3.34.1. 2.4 寸 SPI LCD 显示屏

1) 测试的 LCD 显示屏详情页链接如下

http://www.lcdwiki.com/2.4inch_SPI_Module_ILI9341_SKU:MSP2402

2) LCD 显示屏和开发板的接线方式如下所示

TFT SPI 模块引脚	开发板 26pin 对应的引脚	GPIO -- GPIO num
VCC	1 号引脚	
GND	6 号引脚	
CS	24 号引脚	
RESET	7 号引脚	PC9 -- 73
D/C	11 号引脚	PC6 -- 70
SDI(MOSI)	19 号引脚	
SCK	23 号引脚	



LED	13 号引脚	PC5 -- 69
SDO(MISO)	21 引脚	

3) 将显示屏接到开发板后，再使用下面的命令加载 **fbtft_device** 内核模块

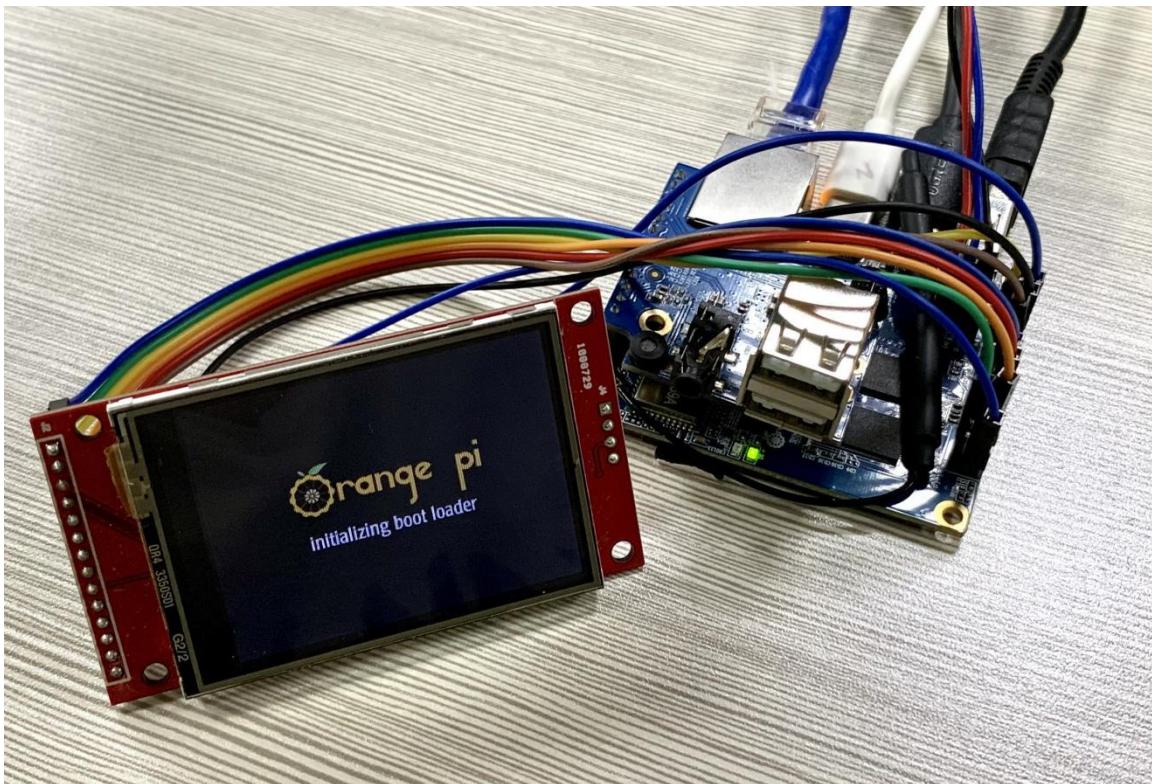
```
root@orangepi:~# modprobe fbtft_device custom name=fb ili9341 busnum=1 cs=1  
gpios=reset:73,dc:70,led:69 rotate=90 speed=65000000 bgr=1 txbuflen=65536
```

4) **fbtft_device** 内核模块加载时 dmesg 命令正确的输出 log 如下所示，而且由 log 可以知道 LCD 显示屏使用的 framebuffer 为 **fb1**

```
root@orangepi:~# dmesg | tail  
[ 391.862343] spidev spi1.1: dh2228fv spi1.1 16777kHz 8 bits mode=0x00  
[ 391.862773] spidev spi1.1: Deleting spi1.1  
[ 391.864506] fbtft_device: GPIOs used by 'fb ili9341':  
[ 391.864529] fbtft_device: 'reset' = GPIO73  
[ 391.864540] fbtft_device: 'dc' = GPIO70  
[ 391.864550] fbtft_device: 'led' = GPIO69  
[ 391.864579] spidev spi0.0: dh2228fv spi0.0 16777kHz 8 bits mode=0x00  
[ 391.864598] spi spi1.1: fb ili9341 spi1.1 65000kHz 8 bits mode=0x00  
[ 391.883881] fb ili9341: module is from the staging directory, the quality is unknown,  
you have been warned.  
[ 392.159982] graphics fb1: fb ili9341 frame buffer, 320x240, 150 KiB video memory,  
64 KiB buffer memory, fps=20, spi1.1 at 65 MHz
```

5) 然后使用下面的命令就可以在 LCD 显示屏上显示 Orange Pi 的 logo 图片

```
root@orangepi:~# apt update  
root@orangepi:~# apt -y install fbi  
root@orangepi:~# fbi -vt 1 -noverbose -d /dev/fb1 /boot/boot.bmp
```



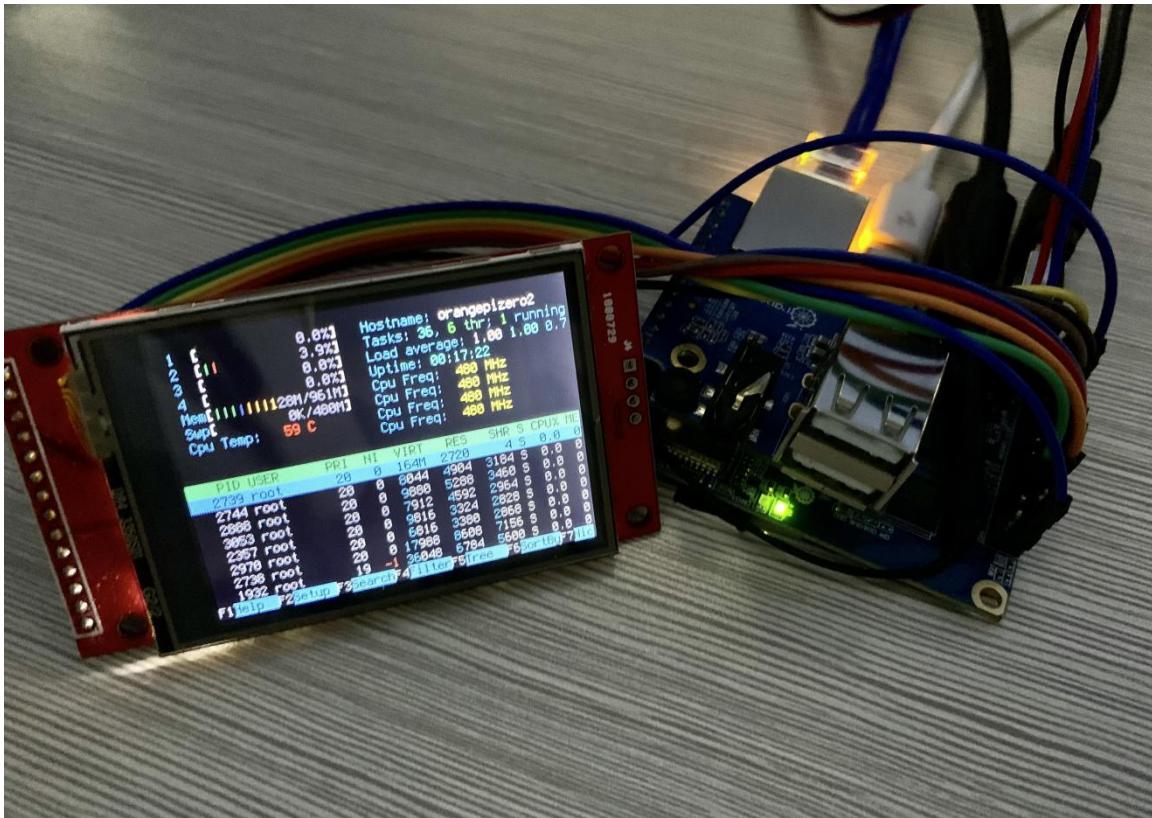
6) 还可以将 `tty1` 的输出映射到 LCD 显示屏的 `fb` 设备——`fb1`，映射完后，HDMI 就不会再有图像输出了

```
root@orangeipi:~# con2fbmap 1 1
```

如果要切换回 HDMI 显示，请使用下面的命令

```
root@orangeipi:~# con2fbmap 1 0
```

下面是运行 `htop` 命令的输出



7) 由于默认的终端字体太大，导致显示屏无法显示太多的内容，可以通过下面的方法来缩小终端的字体

a. 首先运行 **dpkg-reconfigure console-setup**

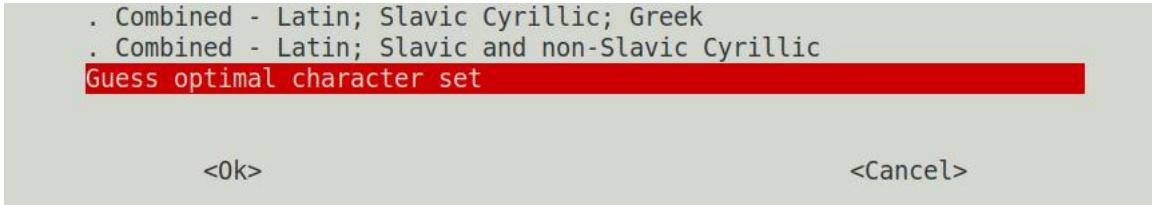
```
root@orangeipi:~# apt-get update  
root@orangeipi:~# apt-get install kbd  
root@orangeipi:~# dpkg-reconfigure console-setup
```

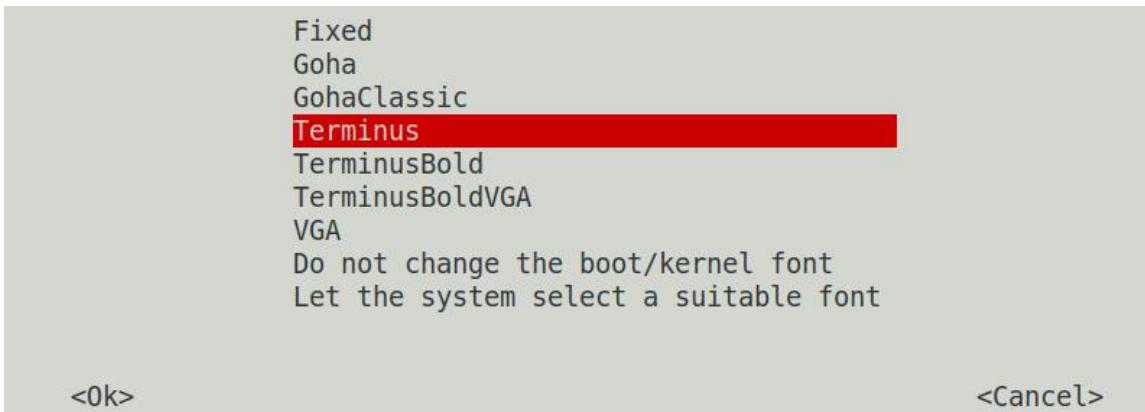
b. 终端编码选择 **UTF-8**



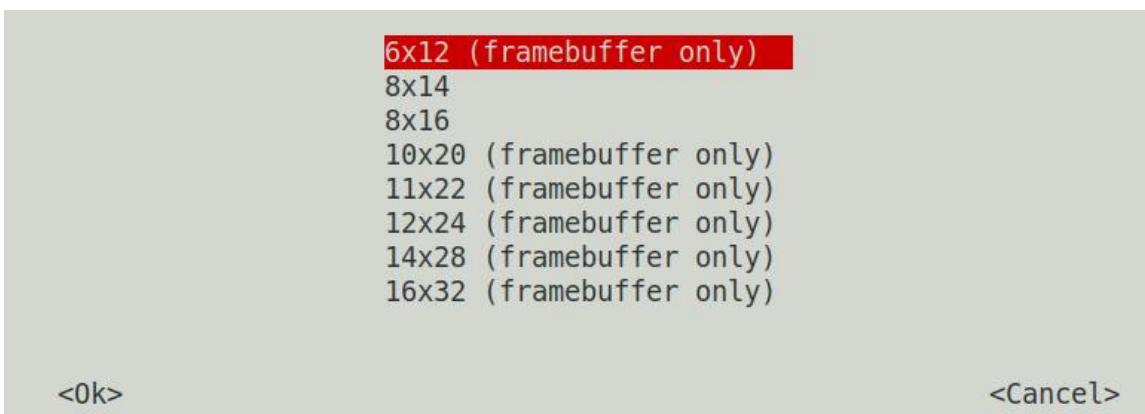
c. 然后选择 **Guess optimal character set**

- . Combined - Latin; Slavic Cyrillic; Greek
 - . Combined - Latin; Slavic and non-Slavic Cyrillic
- Guess optimal character set



d. 然后选择 **Terminus**

e. 最后选择字体大小为 6x12



f. 设置完后就能看到 LCD 显示屏上的字体变小了

8) 设置系统启动自动加载 fbtft_device 模块的方法

a. 新建/etc/modules-load.d/fbtft.conf 配置文件，文件内容如下所示

```
root@orangepi:~# cat /etc/modules-load.d/fbtft.conf
fbtft_device
```

b. 新建/etc/modprobe.d/fbtft.conf 配置文件，文件内容如下所示

```
root@orangepi:~# cat /etc/modprobe.d/fbtft.conf
options fbtft_device custom name=fb ili9341 busnum=1 cs=1
gpis=reset:73,dc:70,led:69 rotate=90 speed=65000000 bgr=1 txbuflen=65536
```

c. 然后重启 linux 系统就能看到 fbtft_device 相关的内核模块都已自动加载

9) 如果希望 linux 系统启动后自动将 console 映射到 LCD 显示屏，请在 /boot/orangepiEnv.txt 中加入下面的配置，然后重启系统就能看到 LCD 显示屏有输出了



```
root@orangepi:~# cat /boot/orangepiEnv.txt | grep "fbcon"  
extraargs=fbcon=map:1
```

3.34.2. 3.2 寸 RPi SPI LCD 显示屏

1) 测试的 LCD 显示屏详情页链接如下

http://www.lcdwiki.com/3.2inch_RPi_Display

2) LCD 显示屏和开发板接线方式如下所示



3) 将 LCD 显示屏接到开发板后，再使用下面的命令加载 **fbtft_device** 内核模块

```
root@orangepi:~# modprobe fbtft_device custom name=fb ili9341 busnum=1 cs=1  
gpios=reset:69,dc:72 rotate=90 speed=65000000 bgr=1 txbuflen=65536
```

4) **fbtft_device** 内核模块加载时 dmesg 命令正确的输出 log 如下所示，而且由 log 可以知道 LCD 屏幕使用的 framebuffer 为 **fb1**

```
root@orangepi:~# dmesg | tail  
[ 271.924571] spidev spi0.0: dh2228fv spi0.0 16777kHz 8 bits mode=0x00  
[ 271.924598] spidev spi1.1: dh2228fv spi1.1 16777kHz 8 bits mode=0x00  
[ 271.925034] spidev spi1.1: Deleting spi1.1  
[ 271.926925] fbtft_device: GPIOs used by 'fb ili9341':  
[ 271.926957] fbtft_device: 'reset' = GPIO69  
[ 271.926968] fbtft_device: 'dc' = GPIO72  
[ 271.926997] spidev spi0.0: dh2228fv spi0.0 16777kHz 8 bits mode=0x00  
[ 271.927016] spi spi1.1: fb ili9341 spi1.1 65000kHz 8 bits mode=0x00  
[ 271.946173] fb ili9341: module is from the staging directory, the quality is unknown,  
you have been warned.  
[ 272.220982] graphics fb1: fb ili9341 frame buffer, 320x240, 150 KiB video memory,
```



64 KiB buffer memory, fps=20, spi1.1 at 65 MHz

5) 然后使用下面的命令就可以在 LCD 屏幕上显示 Orange Pi 的 logo 图片

```
root@orangeipi:~# apt update  
root@orangeipi:~# apt -y install fbi  
root@orangeipi:~# fbi -vt 1 -noverbose -d /dev/fb1 /boot/boot.bmp
```



6) 还可以将 tty1 的输出映射到 LCD 屏幕的 fb 设备——**fb1**, 映射完后, HDMI 就不会再有图像输出了

```
root@orangeipi:~# con2fbmap 1 1
```

如果要切换回 HDMI 显示, 请使用下面的命令

```
root@orangeipi:~# con2fbmap 1 0
```

下面是运行 htop 命令的输出



7) 由于默认的终端字体太大，导致屏幕无法显示太多的内容，可以通过下面的方法来缩小终端的字体

a. 首先运行 **dpkg-reconfigure console-setup**

```
root@orangepi:~# apt-get update
root@orangepi:~# apt-get -y install kbd
root@orangepi:~# dpkg-reconfigure console-setup
```

b. 终端编码选择 **UTF-8**



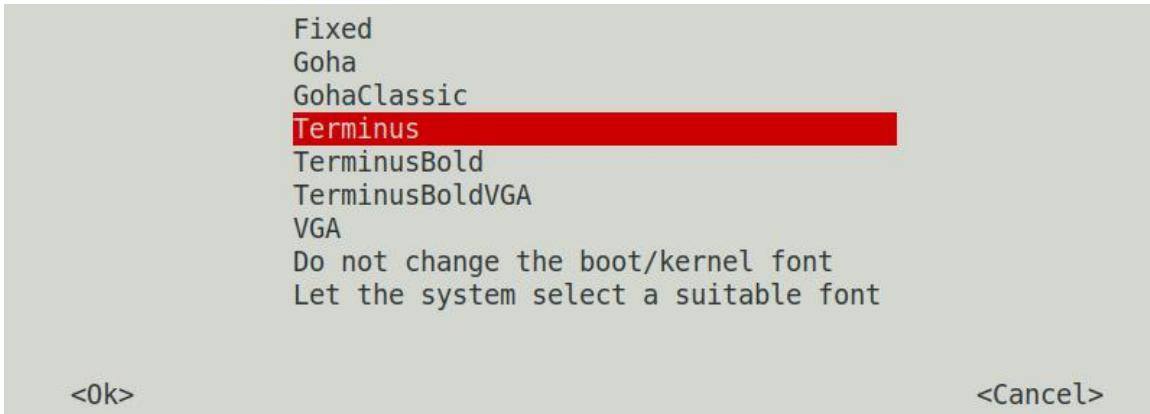
c. 然后选择 **Guess optimal character set**

- . Combined - Latin; Slavic Cyrillic; Greek
- . Combined - Latin; Slavic and non-Slavic Cyrillic

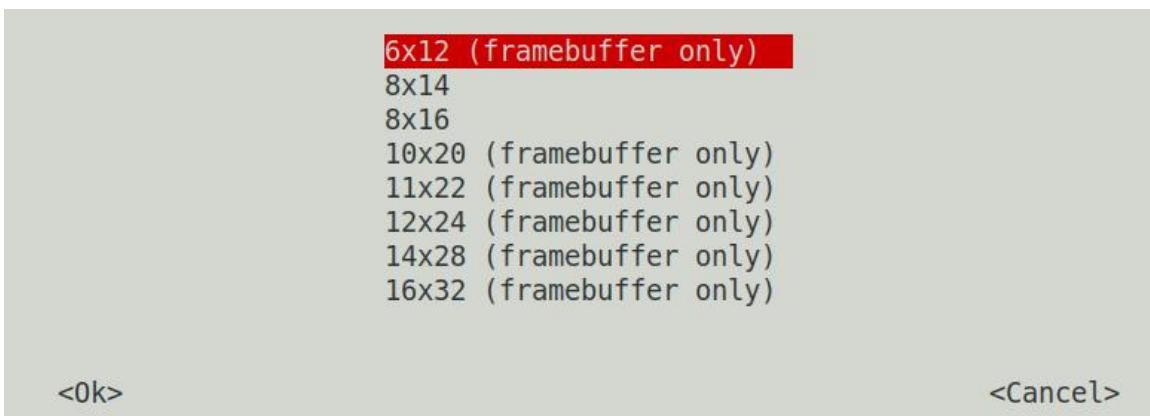
Guess optimal character set

<Ok> <Cancel>

d. 然后选择 **Terminus**



e. 最后选择字体大小为 6x12



f. 设置完后就能看到 LCD 屏幕上的字体变小了

8) 设置系统启动自动加载 fbtft_device 模块的方法

a. 新建/etc/modules-load.d/fbtft.conf 配置文件，文件内容如下所示

```
root@orangeipi:~# cat /etc/modules-load.d/fbtft.conf
fbtft_device
```

b. 新建/etc/modprobe.d/fbtft.conf 配置文件，文件内容如下所示

```
root@orangeipi:~# cat /etc/modprobe.d/fbtft.conf
options fbtft_device custom name=fb ili9341 busnum=1 cs=1 gpios=reset:69,dc:72
rotate=90 speed=65000000 bgr=1 txbuflen=65536
```

c. 然后重启 linux 系统就能看到 fbtft_device 相关的内核模块都已自动加载

9) 如果希望 linux 系统启动后自动将 console 映射到 LCD 屏幕，请在 /boot/orangeipiEnv.txt 中加入下面的配置，然后重启系统就能看到 LCD 屏幕有输出了

```
root@orangeipi:~# cat /boot/orangeipiEnv.txt | grep "fbcon"
```

**extraargs=fbcon=map:1**

3.34.3. 3.5 寸 SPI LCD 显示屏

1) 测试的 LCD 显示屏详情页链接如下

http://www.lcdwiki.com/3.5inch_SPI_Module_ILI9488_SKU:MSP3520

2) LCD 显示屏和开发板的接线方式如下所示

TFT SPI 模块引脚	开发板 26pin 对应的引脚	GPIO -- GPIO num
VCC	1 号引脚	
GND	6 号引脚	
CS	24 号引脚	
RESET	7 号引脚	PC9 -- 73
DC/RS	11 号引脚	PC6 -- 70
SDI(MOSI)	19 号引脚	
SCK	23 号引脚	
LED	13 号引脚	PC5 -- 69
SDO(MISO)	21 引脚	

3) 将显示屏接到开发板后，再使用下面的命令加载 **fbtft_device** 内核模块

```
root@orangepi:~# modprobe fbtft_device custom name=fb_il9488 busnum=1 cs=1  
gpios=reset:73,dc:70,led:69 rotate=90 speed=65000000 bgr=1 txbuflen=65536
```

4) **fbtft_device** 内核模块加载时 **dmesg** 命令正确的输出 log 如下所示，而且由 log 可以知道 LCD 显示屏使用的 framebuffer 为 **fb1**

```
root@orangepi:~# dmesg | tail  
[ 378.953595] spidev spi1.1: dh2228fv spi1.1 16777kHz 8 bits mode=0x00  
[ 378.953952] spidev spi1.1: Deleting spi1.1  
[ 378.955865] fbtft_device: GPIOs used by 'fb_il9488':  
[ 378.955881] fbtft_device: 'reset' = GPIO73  
[ 378.955890] fbtft_device: 'dc' = GPIO70  
[ 378.955898] fbtft_device: 'led' = GPIO69  
[ 378.955924] spidev spi0.0: dh2228fv spi0.0 16777kHz 8 bits mode=0x00  
[ 378.955939] spi spi1.1: fb_il9488 spi1.1 65000kHz 8 bits mode=0x00  
[ 378.971754] fb_il9488: module is from the staging directory, the quality is unknown,
```



you have been warned.

```
[ 379.318032] graphics fb1: fb ili9488 frame buffer, 480x320, 300 KiB video memory,  
64 KiB buffer memory, fps=60, spi1.1 at 65 MHz
```

5) 然后使用下面的命令就可以在 LCD 显示屏上显示 Orange Pi 的 logo 图片

```
root@orangeipi:~# apt update  
root@orangeipi:~# apt -y install fbi  
root@orangeipi:~# fbi -vt 1 -noverbose -d /dev/fb1 /boot/boot.bmp
```

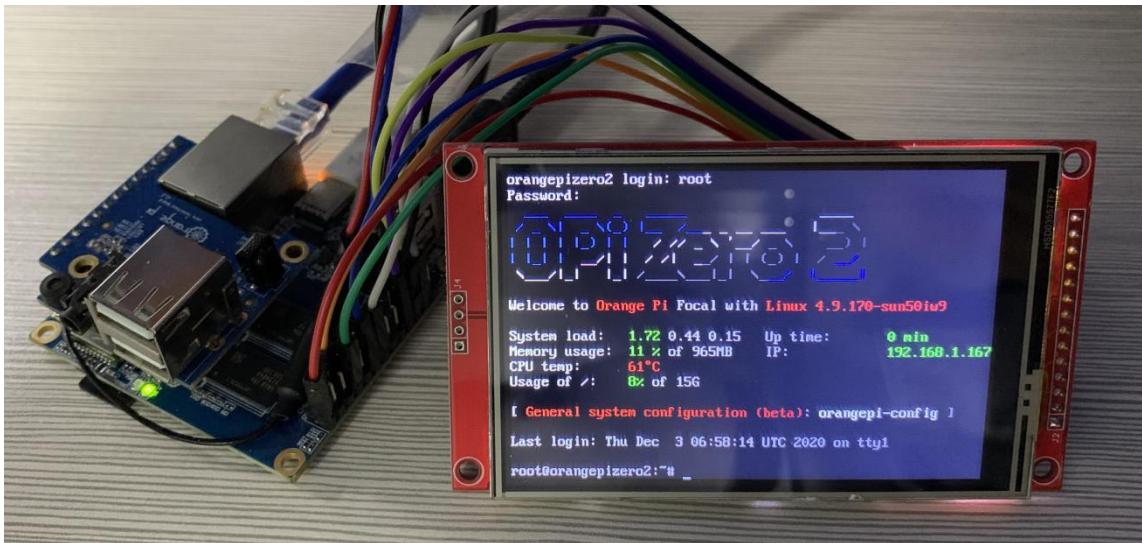


6) 还可以将 tty1 的输出映射到 LCD 显示屏的 fb 设备——**fb1**，映射完后，LCD 屏幕将会显示终端的输出，HDMI 就不会再有图像输出了

```
root@orangeipi:~# con2fbmap 1 1
```

如果要切换回 HDMI 显示，请使用下面的命令

```
root@orangeipi:~# con2fbmap 1 0
```



7) 设置系统启动自动加载 fbtft_device 模块的方法

- 新建/etc/modules-load.d/fbtft.conf 配置文件，文件内容如下所示

```
root@orangeipi:~# cat /etc/modules-load.d/fbtft.conf
fbtft_device
```

- 新建/etc/modprobe.d/fbtft.conf 配置文件，文件内容如下所示

```
root@orangeipi:~# cat /etc/modprobe.d/fbtft.conf
options fbtft_device custom name=fb ili9488 busnum=1 cs=1
gpios=reset:73,dc:70,led:69 rotate=90 speed=65000000 bgr=1 txbuflen=65536
```

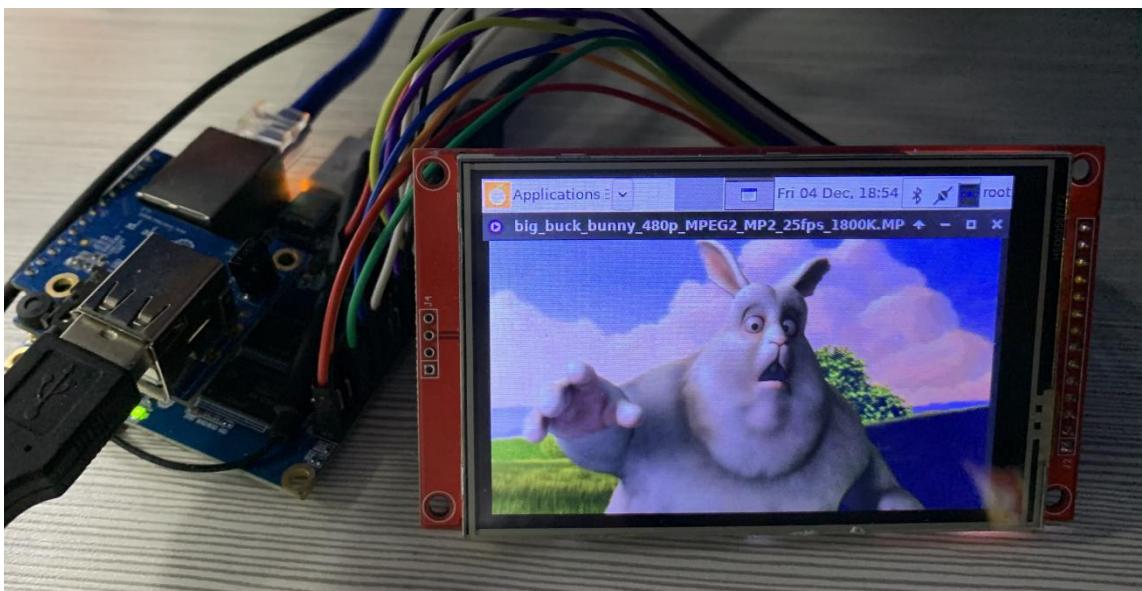
- 然后重启 linux 系统就能看到 fbtft_device 相关的内核模块都已自动加载

8) 如果希望 linux 系统启动后自动将 console 映射到 LCD 显示屏，请在 /boot/orangepiEnv.txt 中加入下面的配置，然后重启系统就能看到 LCD 显示屏有输出了

```
root@orangeipi:~# cat /boot/orangepiEnv.txt | grep "fbcon"
extraargs=fbcon=map:1
```

9) 如果需要将桌面版系统显示到 LCD 屏幕，可以执行下面的命令，等待几秒钟后，LCD 屏幕就能看到 linux 系统的桌面了

```
root@orangeipi:~# FRAMEBUFFER=/dev/fb1 startx
```



10) 如果希望 linux 系统启动后自动将桌面显示到 LCD 显示屏, 请在 linux 系统中添加下面的配置文件, 然后重启系统就能看到 LCD 显示屏有显示输出了

```
root@orangeipi:~# cat /usr/share/X11/xorg.conf.d/99-fbdev.conf
Section "Device"
    Identifier "myfb"
    Driver "fbdev"
    Option "fbdev" "/dev/fb1"
EndSection
```



3. 35. 将内核打印信息输出到 26pin 串口的方法

内核 `console` 默认输出到 `ttyS0`, 也就是开发板上的 3pin 调试串口。我们也可以设置内核 `console` 输出重定向到 26pin 接口中的 **UART5**, 具体方法请参考下面的步骤

- 1) 修改 `/boot/boot.cmd` 中的 `console=ttyS0` 为 `console=ttyS5`

```
root@orangepi:~# vim /boot/boot.cmd
```

```
if test "${console}" = "display" || test "${console}" = "both"; then setenv consoleargs "console=ttyS5 115200 console=tty1"; fi  
if test "${console}" = "serial"; then setenv consoleargs "console=ttyS5 115200"; fi  
if test "${bootlogo}" = "true"; then setenv consoleargs "bootsplash.bootfile=bootsplash.orangepi ${consoleargs}"; fi
```

- 2) 然后将 `/boot/boot.cmd` 重新编译为 `/boot/boot.scr` (在开发板的 linux 系统中操作)

```
root@orangepi:~# mkimage -C none -A arm -T script -d /boot/boot.cmd /boot/boot.scr
```

Image Name:

Created: Tue Nov 3 01:45:17 2020

Image Type: ARM Linux Script (uncompressed)

Data Size: 2247 Bytes = 2.19 KiB = 0.00 MiB

Load Address: 00000000

Entry Point: 00000000

Contents:

Image 0: 2239 Bytes = 2.19 KiB = 0.00 MiB

- 3) 然后将 USB 转 TTL 模块通过杜邦线接到 26pin 接口的 **UART5** 引脚上

a. USB 转 TTL 模块的 GND 接到开发板 26pin 接口的 GND 上

b. USB 转 TTL 模块的 **RX** 接到开发板 **UART5 的 TX 上**

c. USB 转 TTL 模块的 **TX** 接到开发板 **UART5 的 RX 上**



- 4) 然后重启开发板, 可以看到内核 `console` 默认输出到了 `ttyS5`。注意此时 u-boot



的输出 log 还是输出到 ttyS0，不会输出到 ttyS5

```
test@test: ~/OrangePi
Orange Pi 2.1.4 Focal ttyS5
orangepi@orangepi: ~
```

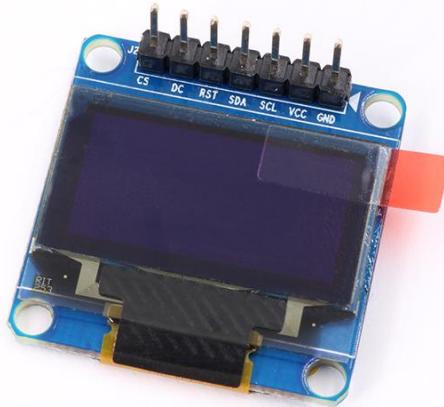
3.36. I2C 接口的 0.96 寸 OLED 模块使用方法

1) 香橙派的 0.96 寸 OLED 模块如下图所示，其 7 位的 i2c 从机地址为 0x3c



2) 先将 0.96 寸 OLED 模块通过杜邦线连接到 Orange Pi 开发板的 26pin 接口上，接线方式如下所示

OLED 模块的引脚	描述	开发板 26pin 接口对应引脚
GND	电源地	6 号引脚
VCC	5V	2 号引脚
SCL	I2C 时钟线	5 号引脚
SDA	I2C 数据线	3 号引脚
RST	接 3.3V	1 号引脚
DC	接 GND	9 号引脚
CS	接 GND	25 号引脚



3) 将 OLED 模块连接到开发板后, 先使用 i2c-tools 工具检查下是否能扫描到 OLED 模块的地址

```
root@orangeipi:~# apt update
root@orangeipi:~# apt install -y i2c-tools
root@orangeipi:~# i2cdetect -y 3      (linux5.13 需要使用 i2cdetect -y 1)
root@orangepirzero2:~# i2cdetect -y 3
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -----
10: --
20: --
30: --          3c
40: --
50: --
60: --
70: --
root@orangepirzero2:~#
```

4) 然后就可以使用 wiringOP 中的 **oled_demo** 来测试 OLED 模块, 测试步骤如下所示

```
root@orangeipi:~# git clone https://github.com/orangepi-xunlong/wiringOP
root@orangeipi:~# cd wiringOP
root@orangeipi:~/wiringOP# ./build clean && ./build
root@orangeipi:~/wiringOP# cd examples
```



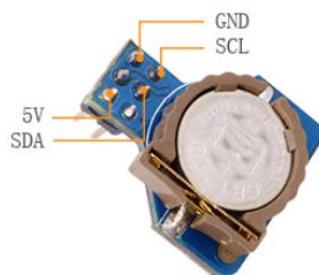
```
root@orangePi:~/wiringOP/examples# make oled_demo
root@orangePi:~/wiringOP/examples# ./oled_demo /dev/i2c-3 (linux5.13 需要使用 ./oled_demo /dev/i2c-1)
-----start-----
-----end-----
```

5) 运行 oled_demo 后，在 OLED 屏幕上就能看到下面的输出



3.37. 香橙派 DS1307 RTC 时钟模块使用方法

1) 香橙派 DS1307 RTC 时钟模块如下图所示，使用 i2c 接口和开发板通信，i2c 设备地址为 0x68。RTC 模块默认不配电池，使用前需要准备一块纽扣电池



2) 首先将 RTC 模块接到开发板的 26pin 上，接线方式如下所示



RTC 模块的引脚	开发板 26pin 对应的引脚
5V	2 号引脚
GND	6 号引脚
SDA	3 号引脚
SCL	5 号引脚

3) 接好 RTC 模块后, 先用 i2cdetect 命令查看下是否能检测到 RTC 模块的设备地址

```
root@orangeipi:~# apt update
```

```
root@orangeipi:~# apt install -y i2c-tools
```

```
root@orangeipi:~# i2cdetect -y 3      (linux5.13 需要使用 i2cdetect -y 1)
```

```
root@orangepirzero2:~# 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:          - - - - - - - - - - - - - - - - - - - -
60:          - - - - - - - - - - - - - - - - - - - - 68
70:          - - - - - - - - - - - - - - - - - - - -
```

4) 然后在 dts 中添加 **rtc-ds1307** 模块的配置。linux 系统中预装了一个名为 **orangeipi-add-overlay** 的脚本, 通过这个脚本我们可以使用 DT overlay 来动态的添加某些 dts 中没有的功能。首先编写 **rtc-ds1307** 模块的 dts 文件, 内容如下所示

a. linux4.9 系统 rtc-ds1307 模块 dts 文件的内容

```
root@orangeipi:~# cat i2c-ds1307.dts
/dts-v1/;
/plugin/;

{
    compatible = "allwinner,h616", "arm,sun50iw9p1";

    fragment@1 {
        target = <&twi3>;
        __overlay__ {
```



```
#address-cells = <1>;
#size-cells = <0>;
ds1307@68 {
    compatible = "dallas,ds1307";
    reg = <0x68>;
    status = "okay";
};
};

};

};
```

b. linux5.13 系统 rtc-ds1307 模块 dts 文件的内容

```
root@orangepi:~# cat i2c-ds1307.dts
/dts-v1/;
/plugin/;

{
    compatible = "xunlong,orangepi-zero2", "allwinner,sun50i-h616";

    fragment@1 {
        target = "&i2c3";
        __overlay__ {
            #address-cells = <1>;
            #size-cells = <0>;
            ds1307@68 {
                compatible = "dallas,ds1307";
                reg = <0x68>;
                status = "okay";
            };
        };
    };
};
```

c. 然后使用 **orangepi-add-overlay** 将 i2c-ds1307.dts 编译成 i2c-ds1307.dtbo，并且设置好相关的启动变量

```
root@orangepi:~# orangepi-add-overlay i2c-ds1307.dts
Compiling the overlay
```



Copying the compiled overlay file to /boot/overlay-user/
Reboot is required to apply the changes

- d. i2c-ds1307.dtbo 会被复制到/**boot/overlay-user** 中，运行完
orangeipi-add-overlay 后可以查看下/**boot/overlay-user** 中是否有
i2c-ds1307.dtbo 这个文件

```
root@orangeipi:~# cd /boot/overlay-user/
root@orangeipi:/boot/overlay-user# ls
i2c-ds1307.dtbo
```

- e. **orangeipi-add-overlay** 还会在/**boot/orangeipiEnv** 中添加 **user_overlays** 变量，
并设置值为 i2c-ssd1307

```
root@orangeipi:~# cat /boot/orangeipiEnv.txt | grep "user"
user_overlays=i2c-ds1307
```

- f. 然后重启 linux 系统，启动时，在 u-boot 的 log 中可以看到 DT overlay 相关的输出

```
U-boot loaded from SD
Boot script loaded from mmc
214 bytes read in 8 ms (25.4 KiB/s)
645 bytes read in 13 ms (47.9 KiB/s)
Applying user provided DT overlay i2c-ds1307.dtbo
8482593 bytes read in 369 ms (21.9 MiB/s)
23638088 bytes read in 1005 ms (22.4 MiB/s)
## Booting kernel from Legacy Image at 41000000 ...
```

5) 重启后，从 dmesg 输出的 log 中可看到 ds1307 模块的加载信息，ds1307 对应的设备节点为 **rtc0** (**linux5.13** 为 **rtc1**)

```
root@orangeipi:~# dmesg | grep "rtc"
[    2.131445] rtc-ds1307 3-0068: rtc core: registered ds1307 as rtc0
[    2.131470] rtc-ds1307 3-0068: 56 bytes nvram
[    2.132256] sunxi-rtc rtc: rtc core: registered sunxi-rtc as rtc1
[    2.132329] sunxi-rtc rtc: RTC enabled
[    2.307120] rtc-ds1307 3-0068: setting system clock to 2000-00-00 06:33:46 UTC
(1607063626)
```

6) linux 系统启动时，如果开发板连接了网络，linux 系统会通过网络自动同步系统时间为正确的时间，linux 系统默认时间为世界标准时间 UTC，在中国，需要将时区

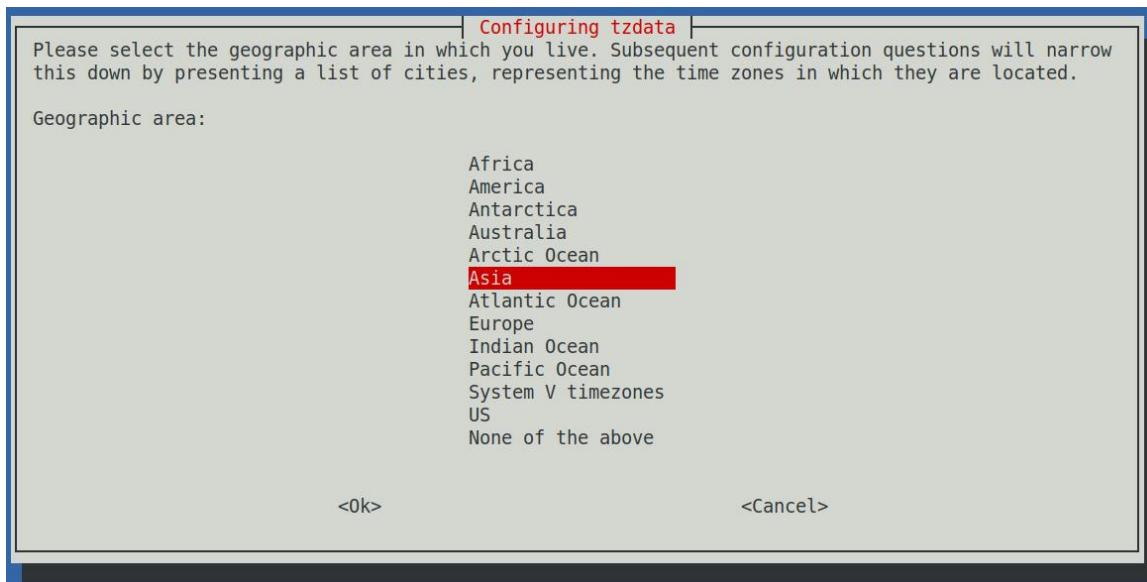


修改为 **Asia/Shanghai**，使用 date 命令获取到的时间才正确，方法如下

a. 执行下面的命令

```
root@orangeipi:~# dpkg-reconfigure tzdata
```

b. 然后选择地理区域为 **Asia**



c. 再选择时区为 **Shanghai**



d. 配置完后再使用 date 命令查看时间就会正常了

```
root@orangeipi:~# date
```

7) 如果系统当前时间不正确，首先请连接网络，然后使用下面的命令同步时间，这里之所以先要将系统时间设置正确，是为了后面同步 RTC 模块的时间做准备

```
root@orangeipi:~# apt-get update  
root@orangeipi:~# apt install -y ntpdate  
root@orangeipi:~# ntpdate 0.cn.pool.ntp.org
```



8) 查看 RTC 模块当前时间的命令如下所示

```
root@orangepi:~# hwclock -r
```

9) 第一次使用 RTC 模块读取到的时间肯定是不对的，可以通过下面的命令将系统当前的时间同步到 RTC 模块，同步前，需要保证系统当前的时间是正确的

```
root@orangepi:~# date          #首先确定当前系统时间是正确的  
root@orangepi:~# hwclock -w      #然后将系统时间写入 RTC 模块  
root@orangepi:~# hwclock -r      #最后读取 RTC 模块的时间确认设置正确
```

10) 此时就可以断开开发板所有的网络连接，然后等待几分钟，再重启系统，然后查看系统时间就会发现即使没有网络，系统的时间也是正确的

3.38. 硬件看门狗测试

1) 下载 wiringOP 的代码

```
root@orangepi:~# apt update  
root@orangepi:~# apt install -y git  
root@orangepi:~# git clone https://github.com/orangepi-xunlong/wiringOP
```

2) 编译 watchdog 测试程序

```
root@orangepi:~# cd wiringOP/examples/  
root@orangepi:~/wiringOP/examples# gcc watchdog.c -o watchdog
```

3) 运行看门狗测试程序

- 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

```
root@orangepi:~/wiringOP/examples# ./watchdog 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
```

3.39. 设置中文环境以及安装中文输入法

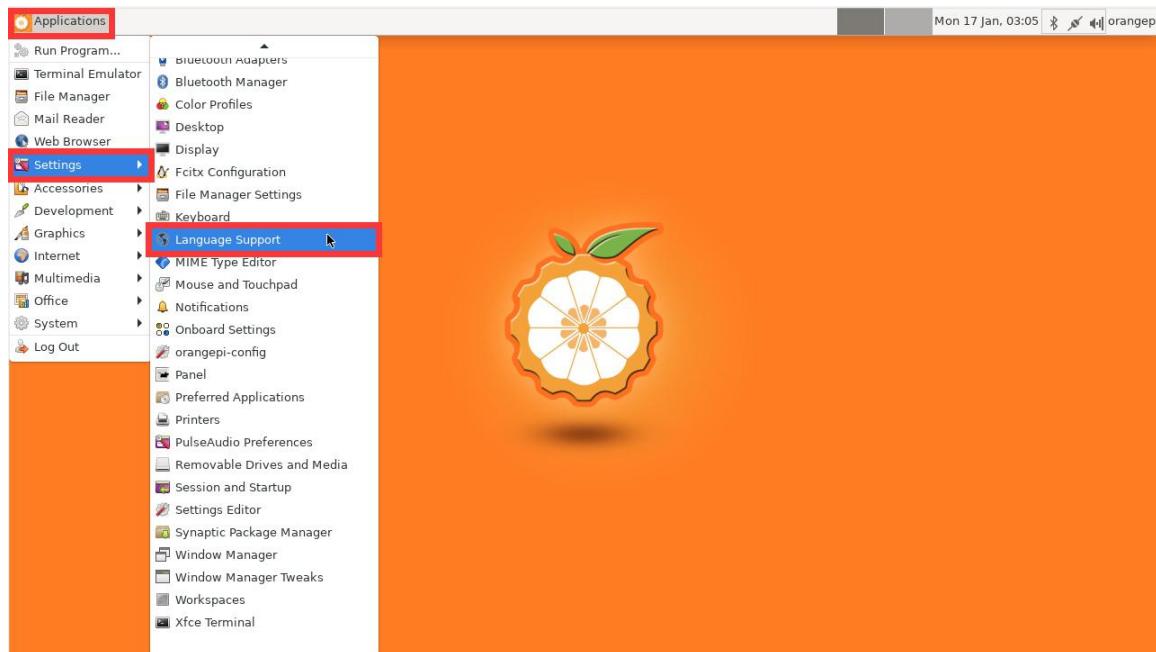
注意，安装中文输入法前请确保开发板使用的 Linux 系统为桌面版系统

3.39.1. Ubuntu 系统的安装方法

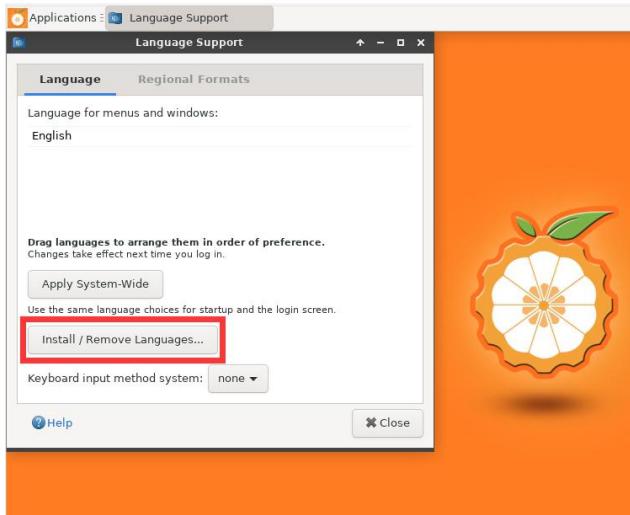
1) 首先更新下系统的软件源

```
root@orangeipi:~# apt update
```

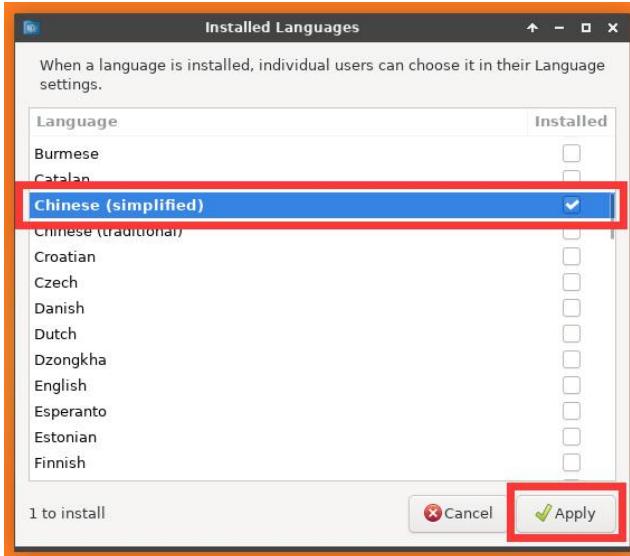
2) 然后打开 **Language Support**



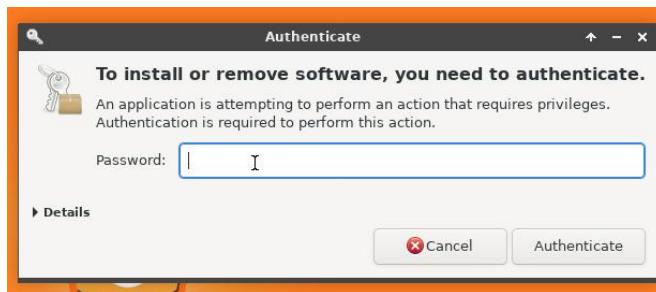
3) 然后打开 **Install/Remove Languages...**



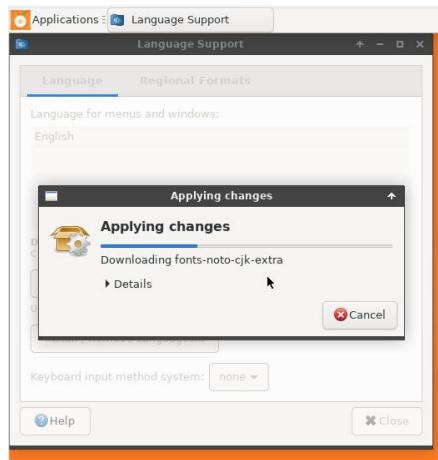
4) 然后找到 **Chinese (simplified)**, 点击右边的方框选上, 再点击右下角的 **Apply**



5) 然后在弹出的密码输入界面中输入 Linux 系统的密码, 默认为 **orangepi**

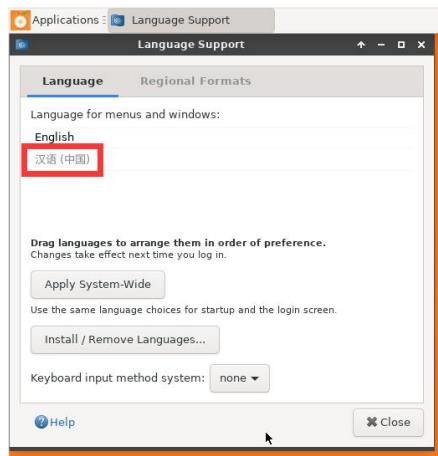


6) 然后就会开始安装需要的软件包, 此时耐心等待安装完成即可

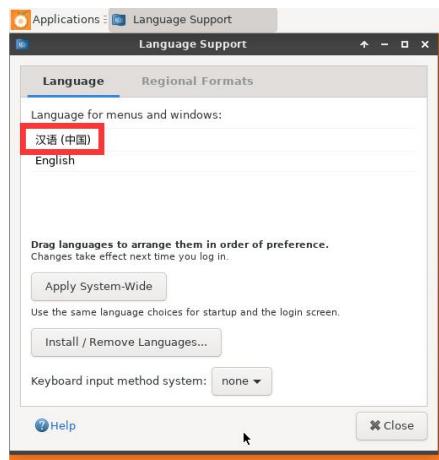


如果这里提示软件安装失败，一般是最开始有没有执行 `apt update` 命令

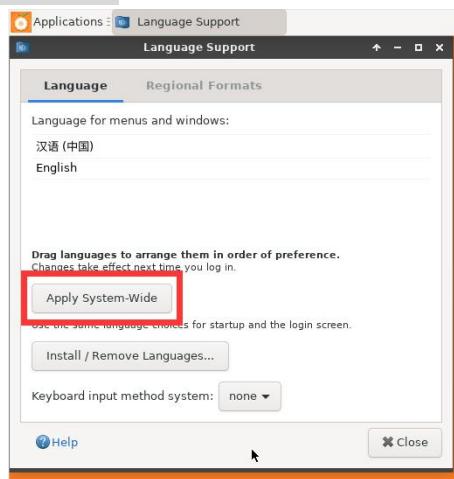
7) 安装完成后就可以看到汉语（中国）的选项了



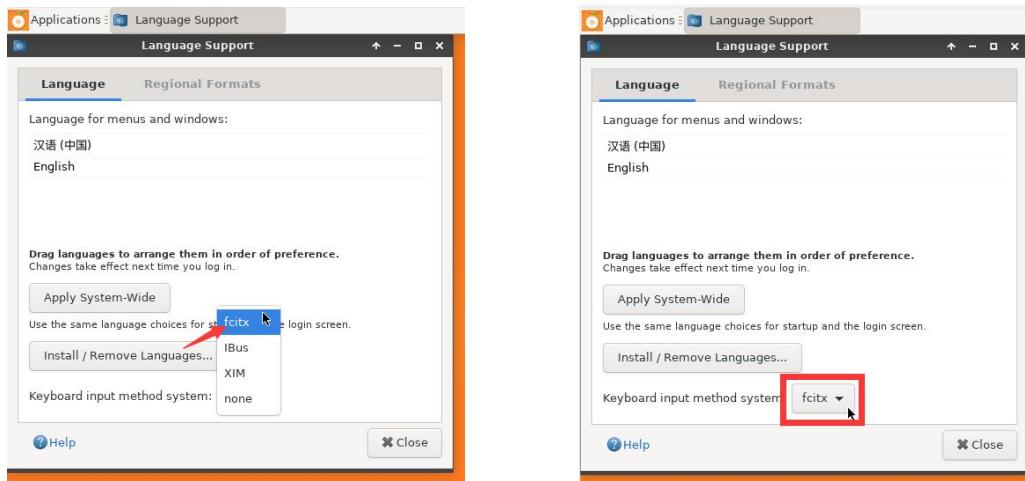
8) 然后请使用鼠标左键选中汉语（中国）并按住不动，然后往上将其拖到最开始的位置，拖完后的显示如下图所示



9) 然后选择 **Apply System-Wide** 将中文设置应用到整个系统



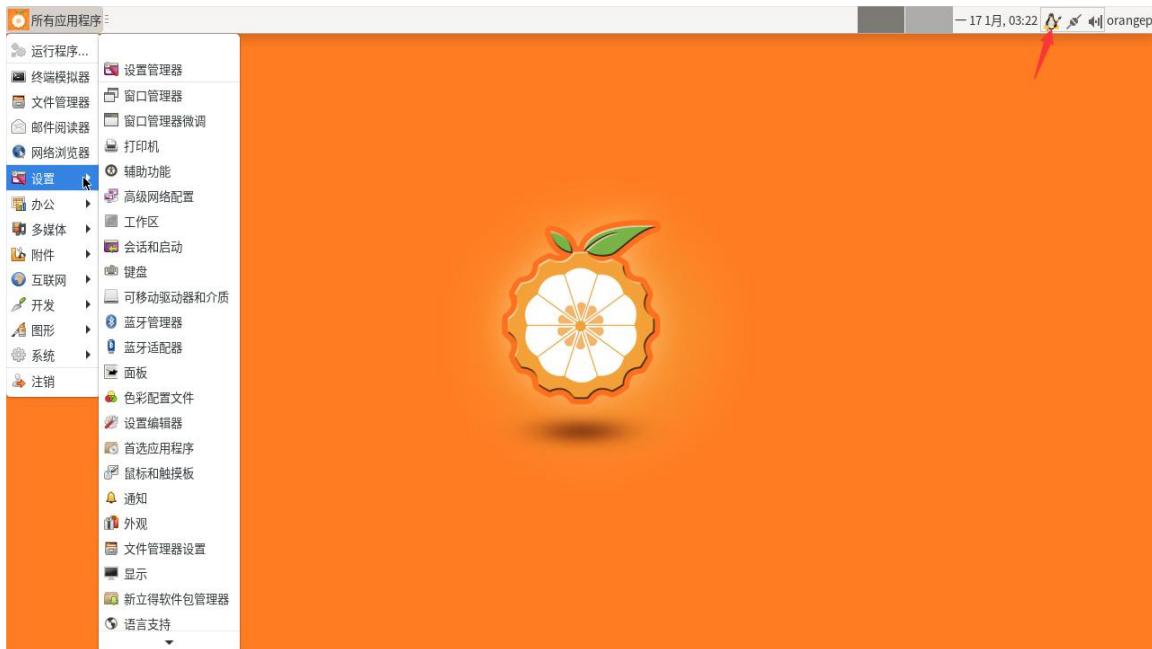
10) 然后选择 **Keyboard input method system** 为 **fcitx**



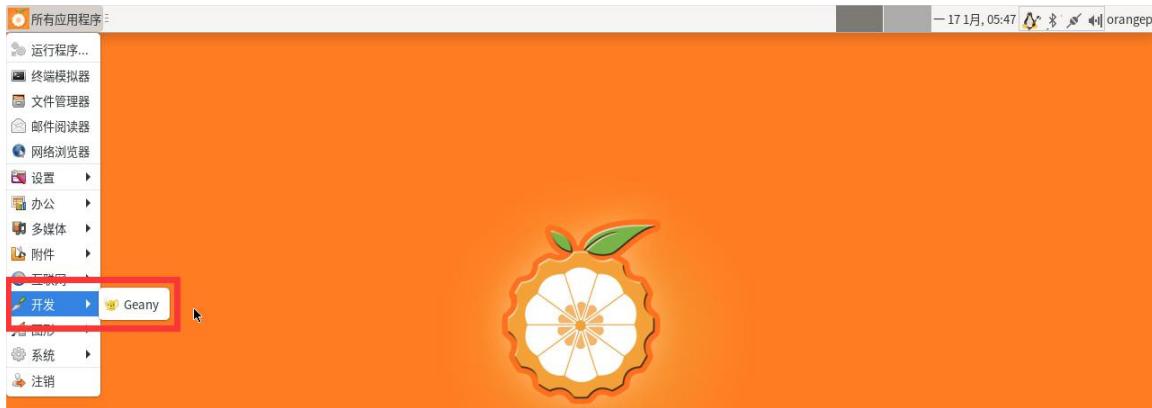


11) 然后重启 Linux 系统使配置生效

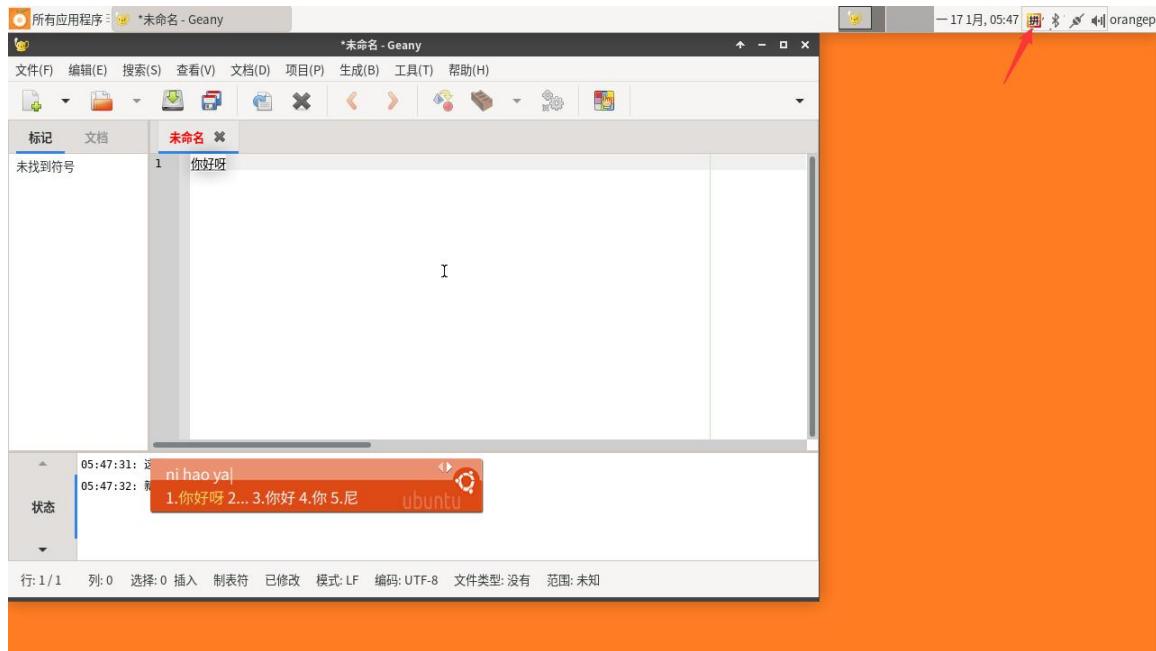
12) 重新进入系统后可以看到桌面都显示为中文了，在系统的右上角还能看到一只企鹅



13) 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



14) 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



15) 另外可以将鼠标放在桌面右上角的企鹅上，然后点击**鼠标右键**，可以查看系统支持的所有输入法，也可以选择需要使用的输入法



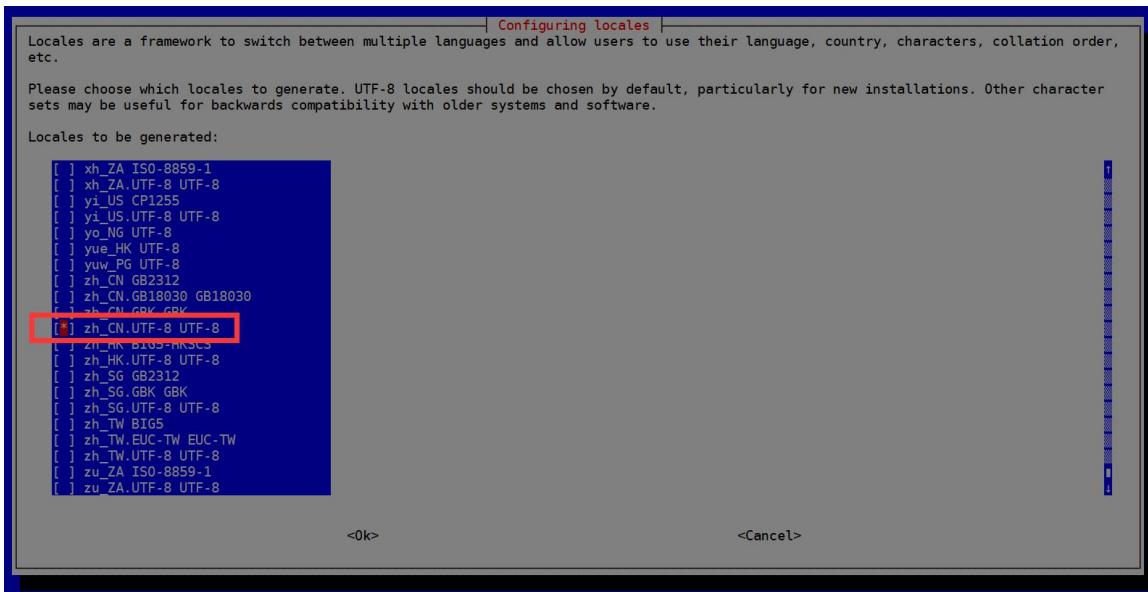
3.39.2. Debian 系统的安装方法

1) 首先设置默认 **locale** 为中文

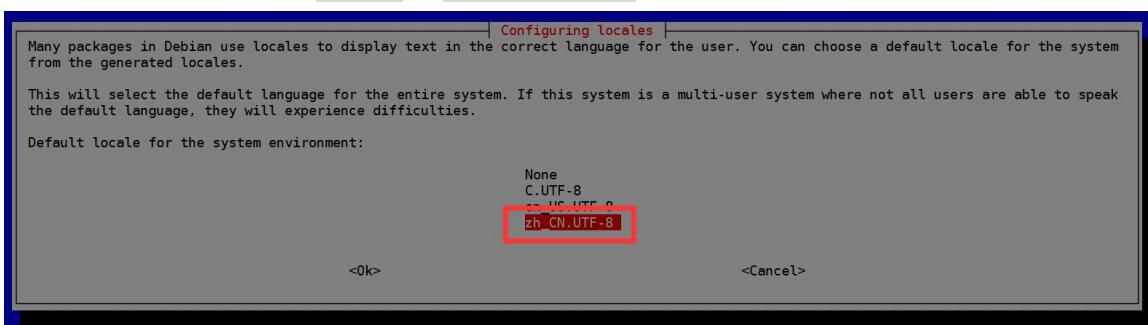
a. 输入下面的命令可以开始配置 **locale**

```
root@orangepi:~# 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**



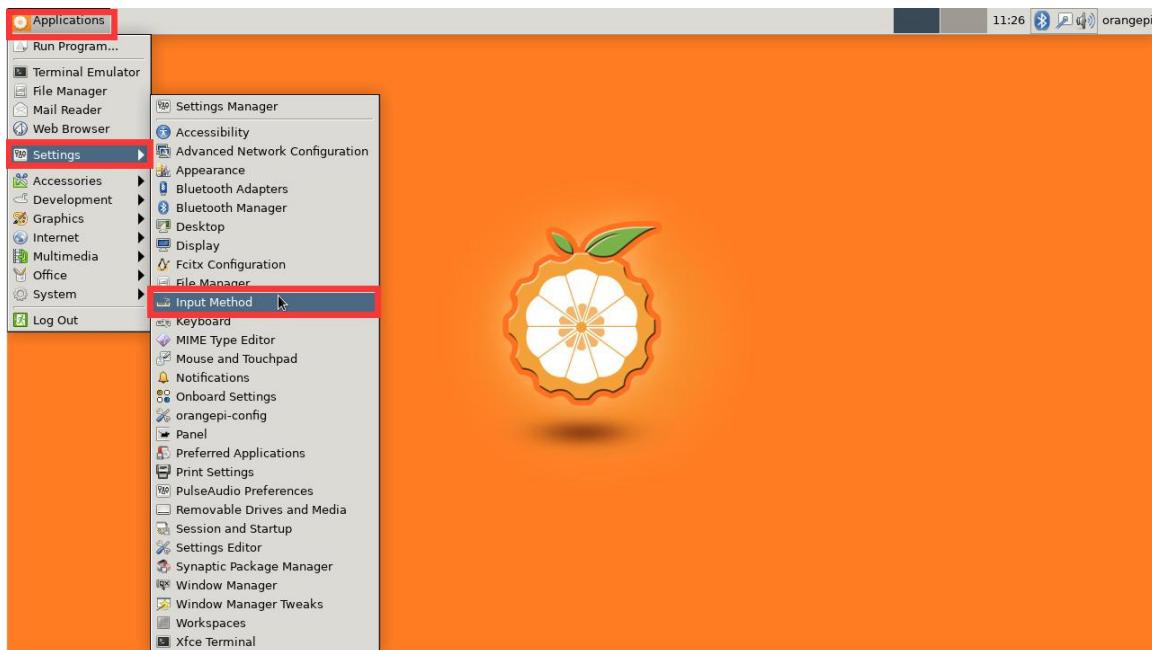
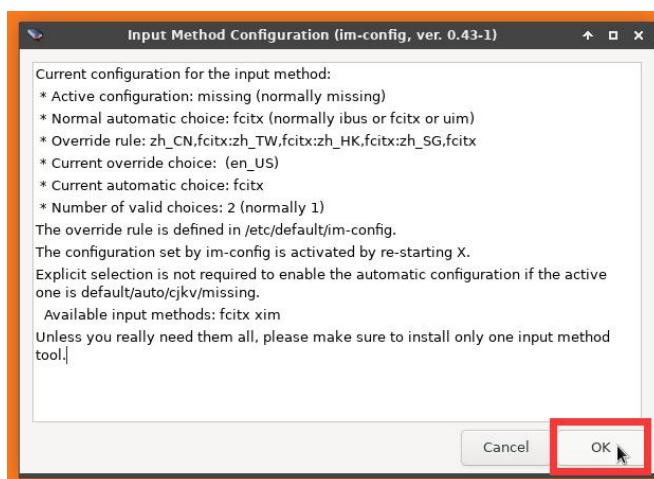
d. 退出界面后就会开始 **locale** 的设置，命令行显示的输出如下所示

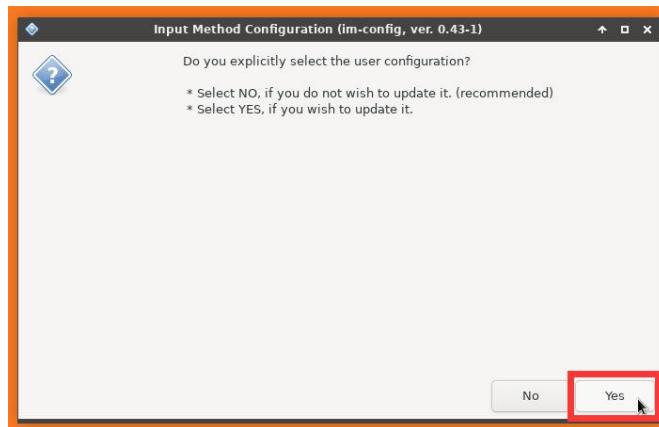
```
root@orangeipi:~# dpkg-reconfigure locales
Generating locales (this might take a while)...
  en_US.UTF-8... done
  zh_CN.UTF-8... done
Generation complete.
```

2) 然后安装下面的软件包

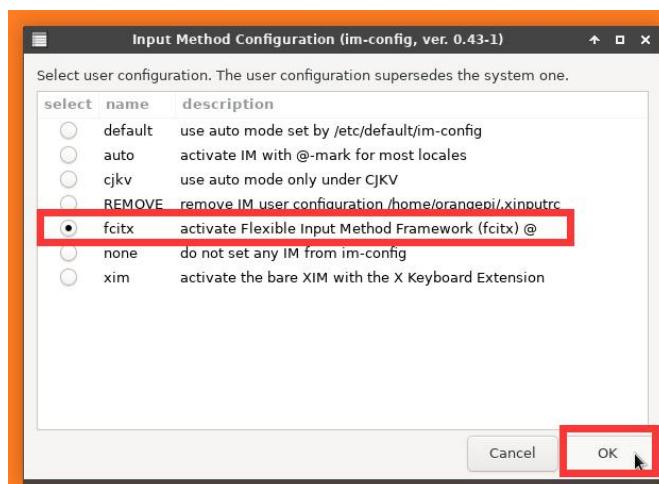
```
root@orangeipi:~# apt update
root@orangeipi:~# apt install -y fonts-archic-bsmi00lp fonts-archic-gbsn00lp
fonts-archic-gkai00mp fcitx fcitx-table* fcitx-frontend-gtk* fcitx-frontend-qt*
fcitx-config-gtk* im-config fcitx-googlepinyin fcitx-ui*
```

3) 然后打开 **Input Method**

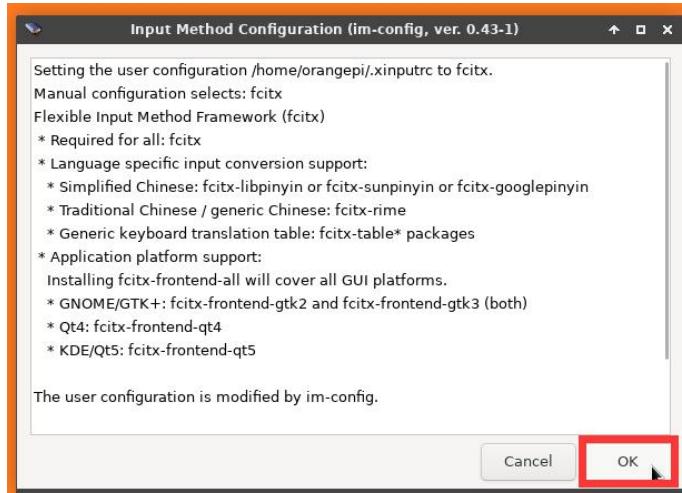
4) 然后选择 **OK**5) 然后选择 **Yes**



6) 然后选择 **fcitx**



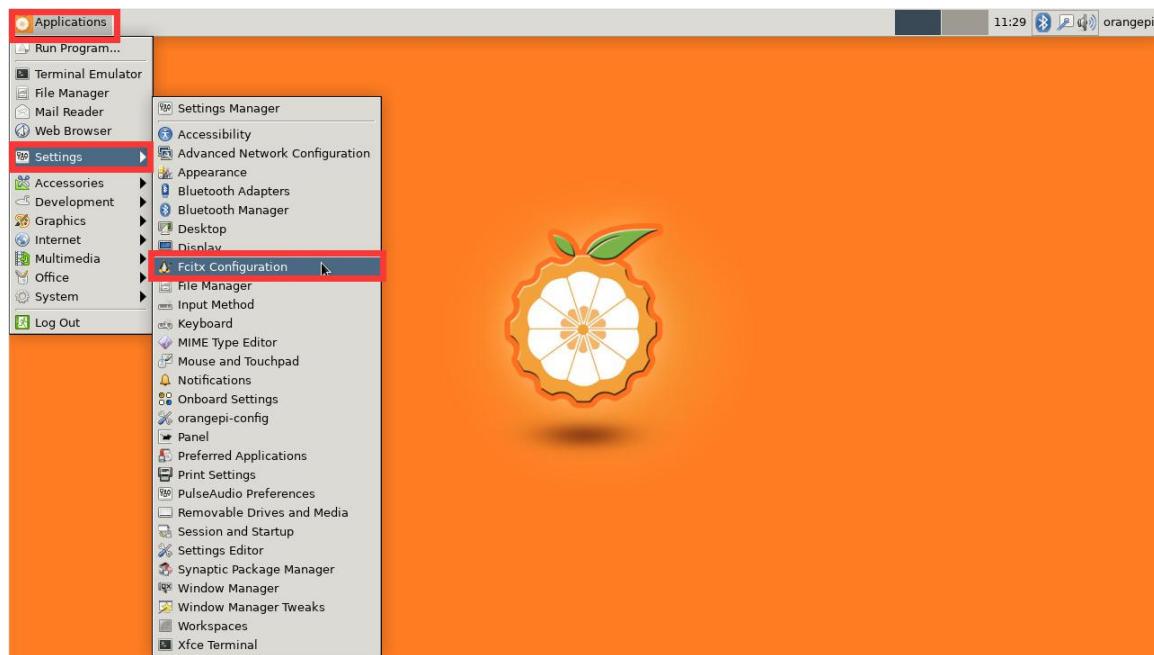
7) 然后选择 **OK**



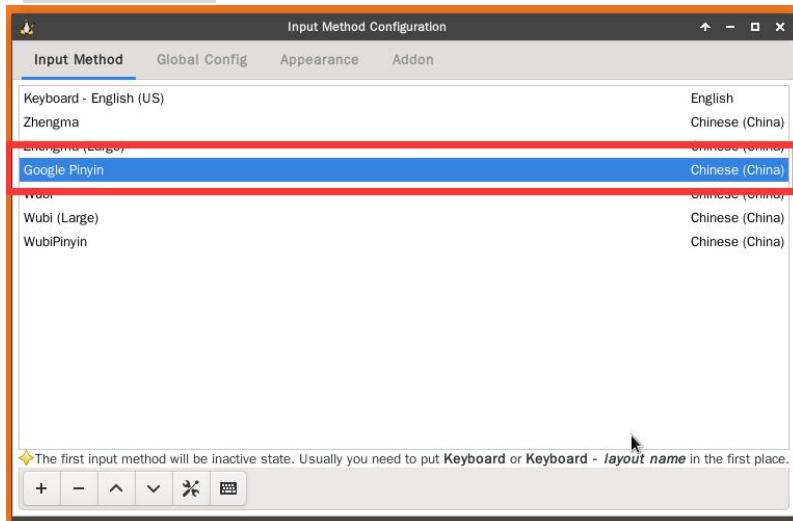


8) 然后重启 Linux 系统才能使配置生效

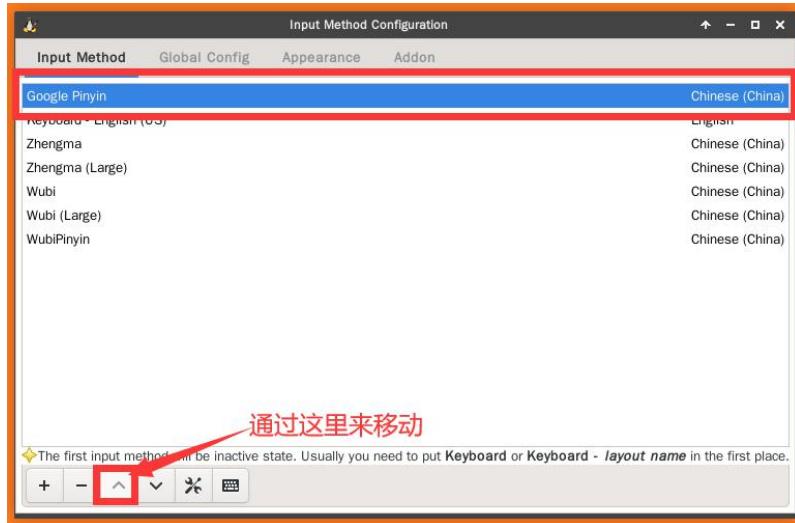
9) 然后打开 **Fcitx configuration**



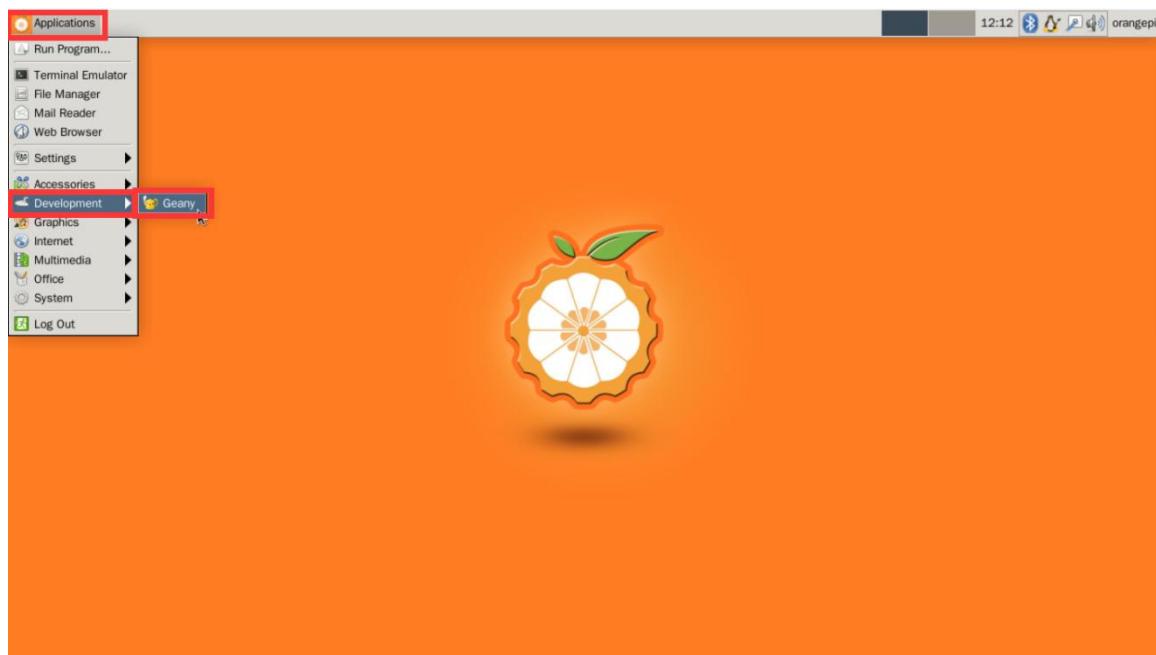
10) 然后选择 **Google Pinyin**



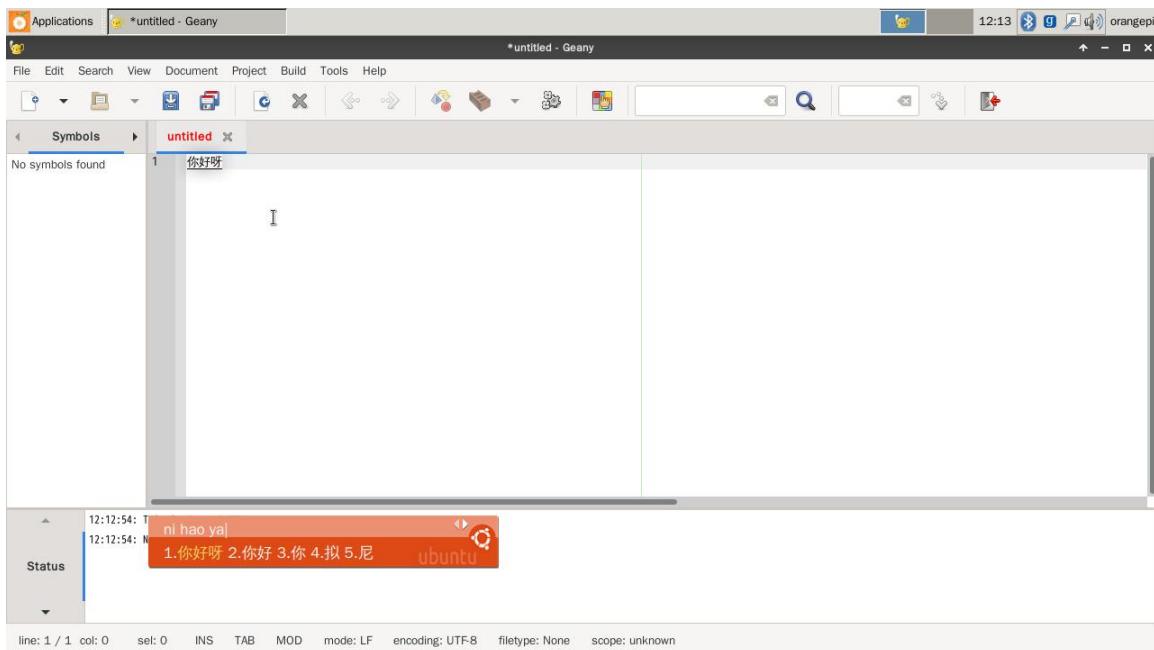
11) 然后将 **Google Pinyin** 放到最前面，再在右上角关闭这个配置界面



12) 然后打开 **Geany** 这个编辑器测试下中文输入法



13) 中文输入测试如下所示



14) 在桌面的右上角可以切换回英文输入

- 首先将鼠标光标放在下图所示的企鹅位置



- 然后点击鼠标右键就可以看到下面的选项,然后选择 **Keyboard-English(US)** 就会切换回英文输入了



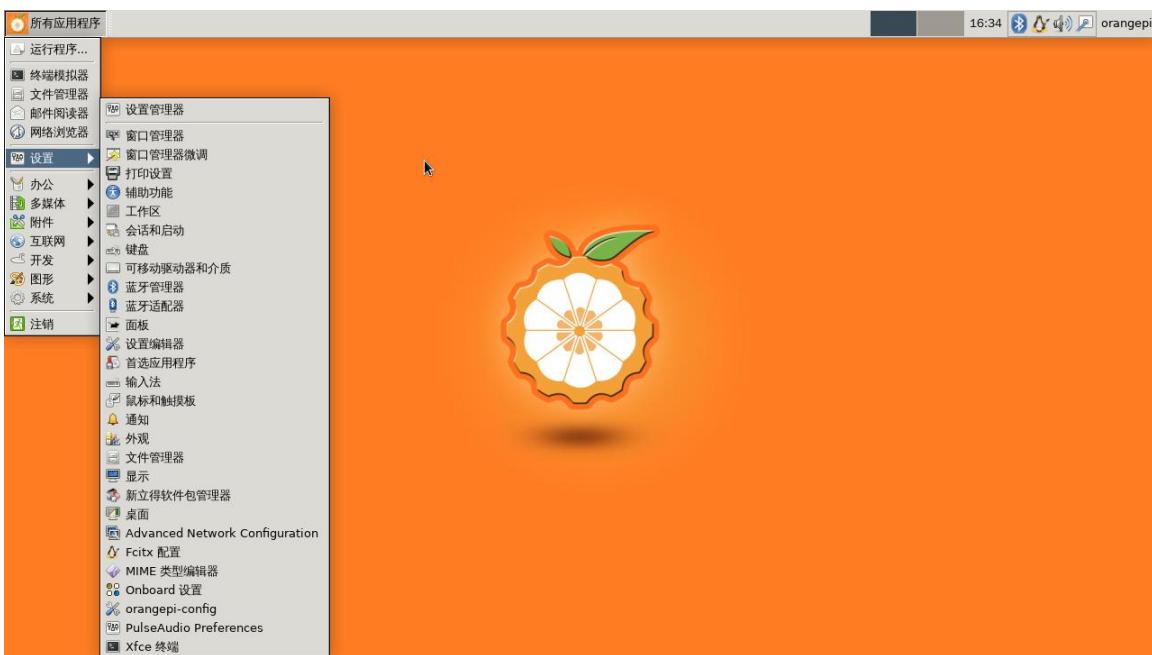


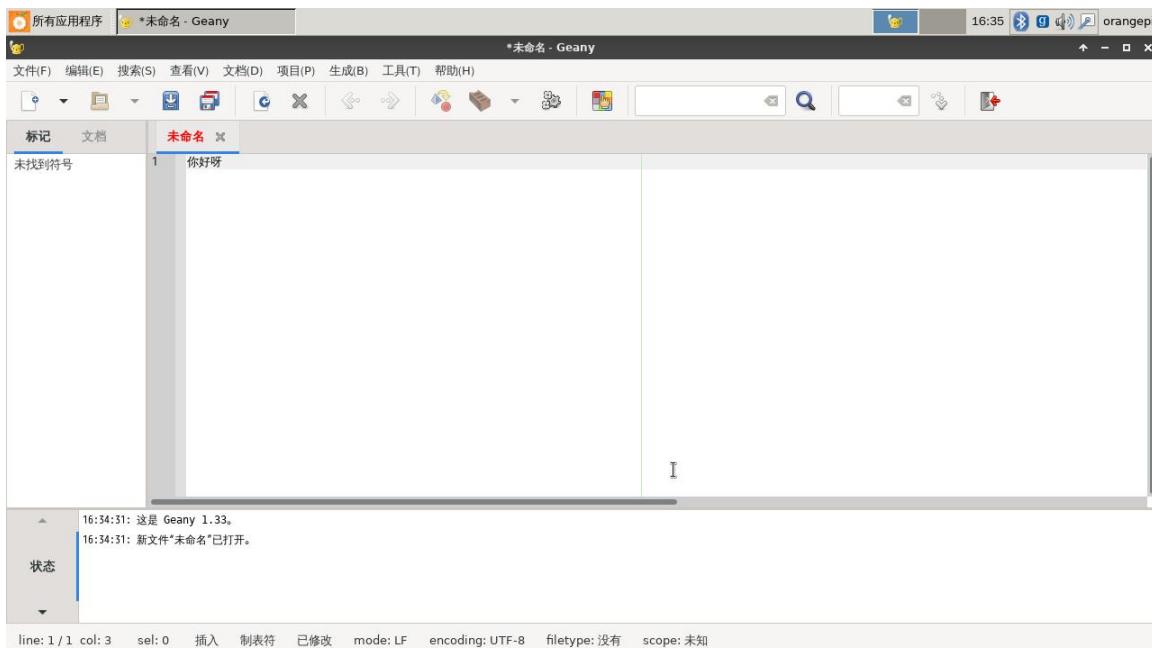
15) 此时就可以通过 **Ctrl+Space** 快捷键来切换中英文输入法了

16) 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**

```
root@orangeipi:~# cat /etc/default/locale
# File generated by update-locale
LC_MESSAGES=zh_CN.UTF-8
LANG=zh_CN.UTF-8
LANGUAGE=zh_CN.UTF-8
```

17) 然后**重启系统**就能看到系统显示为中文了





3. 40. 查看 H616 芯片的 chipid

查看 h616 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

```
root@orangepi:~# cat /sys/class/sunxi_info/sys_info | grep "chipid"
sunxi_chipid      : 32c05000dc00480801411a64298e1dce
```

3. 41. 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

```
root@orangepi:~# poweroff
```

关闭开发板后需要重新拔插电源才能开机

2) 使用 **reboot** 命令即可重启开发板中的 Linux 系统

```
root@orangepi:~# reboot
```



3. 42. Linux4.9 系统修改启动阶段显示的图片的方法

1) Linux4.9 系统 U-boot 启动阶段会在 HDMI 显示器中显示下面的图片



2) 此图片在 linux 系统中的位置如下所示

```
root@orangeipi:~# ls /boot/boot.bmp  
/boot/boot.bmp
```

3) 在 orangepi-build 源码中的位置如下所示，编译 linux 系统时，orangepi-build 中的脚本会将 orangepi-u-boot.bmp 复制到最终生成的 linux 系统的 **/boot** 目录下，并重命名为 **boot.bmp**

```
orangepi-build/external/packages/blobs/splash/orangepi-u-boot.bmp
```

4) boot.bmp 图片相关属性如下所示，如果需要替换 boot.bmp，只需按照下面的属性值制作好相应的图片，然后替换 TF 卡中的 linux 系统中的 boot.bmp 即可，如果是自己编译 linux 系统，只需要替换 orangepi-build 中的 orangepi-u-boot.bmp 即可



3. 43. Linux 系统启动后 TF 卡存储空间和内存的使用情况

1) 服务器版镜像



a. 测试的镜像全名为

Orangepizero2_2.1.6_ubuntu_focal_server_linux5.13.0.img

b. 系统启动后查看到的存储空间和内存的使用情况如下所示

```
root@orangepizero2:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            424M    0  424M  0% /dev
tmpfs           99M   3.2M  96M  4% /run
/dev/mmcblk1p1   15G  1.2G  14G  8% /
tmpfs           493M    0  493M  0% /dev/shm
tmpfs           5.0M  4.0K  5.0M  1% /run/lock
tmpfs           493M    0  493M  0% /sys/fs/cgroup
tmpfs           493M  4.0K  493M  1% /tmp
/dev/zram0       49M  1.4M  44M  4% /var/log
tmpfs           99M    0   99M  0% /run/user/0

root@orangepizero2:~#
root@orangepizero2:~# free -h
              total        used        free      shared  buff/cache   available
Mem:      984Mi       111Mi      768Mi       3.0Mi     104Mi      803Mi
Swap:      492Mi          0B      492Mi

root@orangepizero2:~#
root@orangepizero2:~# lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 20.04.3 LTS
Release:        20.04
Codename:       focal
root@orangepizero2:~#
root@orangepizero2:~# uname -r
5.13.0-sun50iw9
root@orangepizero2:~#
```

2) 桌面板镜像

a. 测试的镜像全名为

Orangepizero2_2.1.6_ubuntu_bionic_desktop_linux5.13.0.img

b. 系统启动后查看到的存储空间和内存的使用情况如下所示



```
root@orangepirzero2:~# df -h
Filesystem      Size   Used  Avail Use% Mounted on
udev            424M     0    424M  0% /dev
tmpfs           99M  3.1M   96M  4% /run
/dev/mmcblk1p1  15G  2.1G   13G 15% /
tmpfs           493M     0   493M  0% /dev/shm
tmpfs           5.0M  4.0K   5.0M  1% /run/lock
tmpfs           493M     0   493M  0% /sys/fs/cgroup
tmpfs           493M   12K   493M  1% /tmp
/dev/zram0       49M  1.9M   44M  5% /var/log
tmpfs           99M   12K   99M  1% /run/user/1000
tmpfs           99M     0   99M  0% /run/user/0
root@orangepirzero2:~#
root@orangepirzero2:~# free -h
              total        used        free      shared  buff/cache   available
Mem:      984M       273M       436M        3.6M       273M       635M
Swap:      492M          0B       492M
root@orangepirzero2:~#
root@orangepirzero2:~# lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 18.04.5 LTS
Release:        18.04
Codename:       bionic
root@orangepirzero2:~#
root@orangepirzero2:~# uname -r
5.13.0-sun50iw9
root@orangepirzero2:~# █
```

3.44. 使用 Kiauh 安装 Klipper 固件上位机的方法

Klipper 是一个 3D 打印机固件。它可以将 Orange Pi 开发板的功能与一个或多个微控制器结合在一起使用。有关 Klipper 的更多信息，请参考下 [Klipper 的官方文档](#)。

Kiauh 是 [Klipper Installation And Update Helper](#) 的缩写，它提供了一个很直观的选择界面，通过简单的选择就能完成 Klipper 等软件的安装。有关 Kiauh 的更多信息，请参考下 Kiauh 源码中的 [README](#) 文档。

这一小节只会演示下在 Orange Pi 开发板的 Linux 系统中使用 Kiauh 安装 Klipper 固件上位机的过程，并不涉及微控制器或者 3D 打印机如何使用的方法

另外请确保开发板使用的 Linux 系统为 **Ubuntu Focal** 或者 **Debian Buster**，**Ubuntu Bionic** 由于默认的 Python3 版本不符合要求会有问题

- 1) 首先请确保开发板的 Linux 系统已经连接好了 WIFI 或者有线网络，并且能正常访问 GitHub，Klipper 等软件的安装需要从 GitHub 下载很多代码，如果访问 GitHub 有问题，会导致安装失败，然后通过 ssh 远程登录到 Linux 系统



```
test@test:~$ ssh orangepi@192.168.1.xx
```

Kiauh 不支持在 root 用户下使用，所以请使用 orangepi 用户登录 Linux 系统

2) 然后下载 Kiauh 的源码

```
orangepi@orangepi:~$ sudo apt update  
orangepi@orangepi:~$ sudo apt install -y git  
orangepi@orangepi:~$ git clone https://github.com/th33xitus/kiauh.git
```

3) 进入 Kiauh 的源码目录，可以看到其包含的文件和文件夹如下所示

- a. kiauh.sh: kiauh 的启动脚本
- b. scripts: 包含 klipper 等软件的安装脚本

```
orangepi@orangepi:~$ cd kiauh/  
orangepi@orangepi:~/kiauh$ ls  
LICENSE README.md docs kiauh.sh resources scripts
```

4) 然后就可以在 Kiauh 的源码目录下运行 kiauh.sh 脚本

```
orangepi@orangepi:~/kiauh$ ./kiauh.sh
```

5) 运行 kiauh.sh 后会弹出下面的选择界面，可以看到 Kiauh 提供了很多的操作选项，还能显示 Klipper 等相关软件的安装状态



```
/=-----\ [ KIAUH ] -----\
| Klipper Installation And Update Helper
| -----
\-----/ [ Main Menu ]
| -----
| 0) [Upload Log] Klipper: Not installed!
| Branch: -----
| 1) [Install] Moonraker: Not installed!
| 2) [Update] Mainsail: Not installed!
| 3) [Remove] Fluidd: Not installed!
| 4) [Advanced] KlipperScreen: Not installed!
| 5) [Backup] Telegram Bot: Not installed!
| 6) [Settings] DWC2: Not installed!
| Octoprint: Not installed!
| -----
| v3.1.0-85
| -----
| Q) Quit
\-----/
Perform action: |
```

6) 然后就可以在 **Perform action:** 后面输入 **1** 后回车，选择 **1) [Install]**

```
/=-----\ [ KIAUH ] -----\
| Klipper Installation And Update Helper
| -----
\-----/ [ Main Menu ]
| -----
| 0) [Upload Log] Klipper: Not installed!
| Branch: -----
| 1) [Install] Moonraker: Not installed!
| 2) [Update] Mainsail: Not installed!
| 3) [Remove] Fluidd: Not installed!
| 4) [Advanced] KlipperScreen: Not installed!
| 5) [Backup] Telegram Bot: Not installed!
| 6) [Settings] DWC2: Not installed!
| Octoprint: Not installed!
| -----
| v3.1.0-85
| -----
| Q) Quit
\-----/
Perform action: 1| ←
```

7) 然后会弹出下面的安装选择界面



```
/=====
|----- [ KIAUH ] -----
|----- Klipper Installation And Update Helper -----
|----- [ Installation Menu ] -----
|
| You need this menu usually only for installing
| all necessary dependencies for the various
| functions on a completely fresh system.
|
|----- Firmware: | Touchscreen GUI: -----
| 1) [Klipper] | 5) [KlipperScreen]
|
|----- Klipper API: | Other: -----
| 2) [Moonraker] | 6) [Duet Web Control]
|----- Klipper Webinterface: | 7) [OctoPrint]
| 3) [Mainsail] | 8) [PrettyGCode]
| 4) [Fluidd] | 9) [Telegram Bot]
|
|----- Webcam: -----
| 10) [MJPG-Streamer]
|
|----- B) << Back -----
\=====
```

Perform action: █

8) 我们首先在 **Perform action:** 后面输入 **1** 然后回车就会开始安装 Klipper 的过程

```
/=====
|----- [ KIAUH ] -----
|----- Klipper Installation And Update Helper -----
|----- [ Installation Menu ] -----
|
| You need this menu usually only for installing
| all necessary dependencies for the various
| functions on a completely fresh system.
|
|----- Firmware: | Touchscreen GUI: -----
| 1) [Klipper] | 5) [KlipperScreen]
|
|----- Klipper API: | Other: -----
| 2) [Moonraker] | 6) [Duet Web Control]
|----- Klipper Webinterface: | 7) [OctoPrint]
| 3) [Mainsail] | 8) [PrettyGCode]
| 4) [Fluidd] | 9) [Telegram Bot]
|
|----- Webcam: -----
| 10) [MJPG-Streamer]
|
|----- B) << Back -----
\=====
```

Perform action: 1 ←

9) 开始安装 Klipper 后首先会提醒需要设置 Klipper 配置文件所在文件夹的路径，默



认为`/home/orangepi/klipper_config`, 如果不需要修改, 直接回车即可

```
/=====
|           [ KIAUH ] ~~~~~
|           Klipper Installation And Update Helper
| 
\=====

##### Initializing Klipper installation ...

/=====
|           !!! WARNING !!!
|           No Klipper configuration directory set!
| 
|           Before we can continue, KIAUH needs to know where
|           you want your printer configuration to be.
| 
|           Please specify a folder where your Klipper configu-
|           ration is stored or, if you don't have one yet, in
|           which it should be saved after the installation.
\=====

/=====
|           IMPORTANT:
|           Please enter the new path in the following format:
|           /home/orangepi/your_config_folder
| 
|           By default 'klipper_config' is recommended!
\=====

##### Please set the Klipper config directory:
/home/orangepi/klipper_config
```

然后会弹出下面的提示信息, 默认为 Y, 直接回车即可

```
/=====
|           [ KIAUH ] ~~~~~
|           Klipper Installation And Update Helper
| 
\=====

##### Initializing Klipper installation ...

/=====
|           !!! WARNING !!!
|           No Klipper configuration directory set!
| 
|           Before we can continue, KIAUH needs to know where
|           you want your printer configuration to be.
| 
|           Please specify a folder where your Klipper configu-
|           ration is stored or, if you don't have one yet, in
|           which it should be saved after the installation.
\=====

/=====
|           IMPORTANT:
|           Please enter the new path in the following format:
|           /home/orangepi/your_config_folder
| 
|           By default 'klipper_config' is recommended!
\=====

##### Please set the Klipper config directory:
/home/orangepi/klipper_config

##### Set config directory to '/home/orangepi/klipper_config' ? (Y/n): ■
```

10) 然后会提示需要输入 Linux 系统的密码, 默认为 orangepi



```
##### Please set the Klipper config directory:  
/home/orangepi/klipper_config  
  
##### Set config directory to '/home/orangepi/klipper_config' ? (Y/n):  
##### > Yes  
  
##### Create KIAUH backup directory ...  
>>>> Directory created!  
>>>> No config directory found! Skipping backup ...  
  
##### Directory set to '/home/orangepi/klipper_config'!  
[sudo] password for orangepi:  在这里输入linux系统的默认密码: orangepi
```

11) 然后会提示设置需要的 Klipper 实例的个数，这里我们输入 **1** 即可

```
##### Directory set to '/home/orangepi/klipper_config'!  
[sudo] password for orangepi:  
  
>>>> Config directory changed!  
=====\\  
| Please select the number of Klipper instances to set |  
| up. The number of Klipper instances will determine |  
| the amount of printers you can run from this machine. |  
|  
| WARNING: There is no limit on the number of instances |  
| you can set up with this script. |  
=====/  
##### Number of Klipper instances to set up: 1
```

然后继续回车确认即可

```
>>>> Config directory changed!  
=====\\  
| Please select the number of Klipper instances to set |  
| up. The number of Klipper instances will determine |  
| the amount of printers you can run from this machine. |  
|  
| WARNING: There is no limit on the number of instances |  
| you can set up with this script. |  
=====/  
##### Number of Klipper instances to set up: 1  
  
##### Install 1 instance(s)? (Y/n): 
```

12) 然后就会正式开始 Klipper 的安装过程



- a. Kiauh 中的脚本首先会自动从 GitHub 下载 Klipper 的源码，这一步很容易由于网络问题下载失败，请特别注意下输出 log 和下图一致，没有报错

```
##### Installing 1 Klipper instance(s) ...
#####
## Checking for the following dependencies:
• git
>>>> Dependencies already met! Continue...

#####
## Downloading Klipper ...
Cloning into 'Klipper'...
remote: Enumerating objects: 28306, done.
remote: Counting objects: 100% (76/76), done.
remote: Compressing objects: 100% (32/32), done.
remote: Total 28306 (delta 49), reused 67 (delta 44), pack-reused 28230
Receiving objects: 100% (28306/28306), 20.32 MiB | 396.00 KiB/s, done.
Resolving deltas: 100% (21377/21377), done.
Updating files: 100% (1526/1526), done.

#####
## Download complete!
```

- b. 然后会自动安装依赖包，请确保下载和安装的过程不会由于网络的原因出错

```
#####
## Installing packages...
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'python-dev-is-python2' instead of 'python-dev'
Note, selecting 'libusb-1.0-0' for regex 'libusb-1.0'
Note, selecting 'libusb-1.0-0-dev' for regex 'libusb-1.0'
Note, selecting 'libusb-1.0-doc' for regex 'libusb-1.0'
libusb-1.0-0 is already the newest version (2:1.0.23-2build1).
libusb-1.0-0 set to manually installed.
build-essential is already the newest version (12.8ubuntu1.1).
```

- c. 然后自动会安装 Python 的虚拟环境，请确保下载和安装的过程不会由于网络的原因出错

```
#####
## Installing python virtual environment...
created virtual environment CPython2.7.18.final.0-64 in 1637ms
  creator CPython2Posix(dest=/home/orangepi/klippy-env, clear=False, global=
    seeder FromAppData(download=False, pip=latest, setuptools=latest, wheel=la
    ed-app-data/v1.0.1.debian.1)
  activators BashActivator,CShellActivator,FishActivator,PowerShellActivator
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Pl
l drop support for Python 2.7. More details about Python 2 support in pip, c
Collecting cffi==1.14.6
  Downloading cffi-1.14.6.tar.gz (475 kB)
    |██████████| 475 kB 401 kB/s
Collecting pyserial==3.4
  Downloading pyserial-3.4-py2.py3-none-any.whl (193 kB)
    |██████████| 193 kB 328 kB/s
```

- d. 安装完成后会显示下面的信息，并会自动返回 Kiauh 的安装界面



```
##### Creating Klipper Service ...
Created symlink /etc/systemd/system/multi-user.target.wants/klipper.service → /etc/systemd/system/klipper.service.
>>>> Single Klipper instance created!

##### Launching Klipper instance ...

#####
Klipper has been set up!
#####

/===== [ Installation Menu ] =====\
| You need this menu usually only for installing
| all necessary dependencies for the various
| functions on a completely fresh system.
| -----
| Firmware: | Touchscreen GUI:
| 1) [Klipper] | 5) [KlipperScreen]
| Klipper API: | Other:
| 2) [Moonraker] | 6) [Duet Web Control]
| Klipper Webinterface: | 7) [OctoPrint]
| 3) [Mainsail] | 8) [PrettyGCode]
| 4) [Fluidd] | 9) [Telegram Bot]
| | Webcam:
| | 10) [MJPG-Streamer]
| -----
| B) << Back
| -----
Perform action: |
```

13) 然后开始安装 Klipper API——Moonraker，在 **Perform action:** 后面输入 **2** 后回车即可

```
/=====
~~~~~ [ Installation Menu ] ~~~~~
| You need this menu usually only for installing
| all necessary dependencies for the various
| functions on a completely fresh system.
| -----
| Firmware: | Touchscreen GUI:
| 1) [Klipper] | 5) [KlipperScreen]
| Klipper API: | Other:
| 2) [Moonraker] | 6) [Duet Web Control]
| Klipper Webinterface: | 7) [OctoPrint]
| 3) [Mainsail] | 8) [PrettyGCode]
| 4) [Fluidd] | 9) [Telegram Bot]
| | Webcam:
| | 10) [MJPG-Streamer]
| -----
| B) << Back
\=====
Perform action: 2 | ←
```

14) 然后会提示需要设置 Moonraker 实例的个数，这个需要和前面设置的 Klipper 实例的个数相对应，这里我们设置为 **1** 然后回车即可



```
/=====
|           [ KIAUH ] ~~~~~
|           Klipper Installation And Update Helper
|~~~~~\

##### Initializing Moonraker installation ...

##### Your Python 3 version is: Python 3.8.10
/=====
| 1 Klipper instance was found!
| Usually you need one Moonraker instance per Klipper
| instance. Though you can install as many as you wish.
\=====

##### Number of Moonraker instances to set up: 1 ←
```

然后继续回车确认即可

```
/=====
|           [ KIAUH ] ~~~~~
|           Klipper Installation And Update Helper
|~~~~~\

##### Initializing Moonraker installation ...

##### Your Python 3 version is: Python 3.8.10
/=====
| 1 Klipper instance was found!
| Usually you need one Moonraker instance per Klipper
| instance. Though you can install as many as you wish.
\=====

##### Number of Moonraker instances to set up: 1

##### Install 1 instance(s)? (Y/n):
```

然后输入 Linux 系统的默认密码 orangepi 就会开始正式的安装过程



```
##### Installing Moonraker ...  
  
##### Checking for the following dependencies:  
● wget  
● curl  
● unzip  
● dfu-util  
● virtualenv  
● libjpeg-dev  
● zlib1g-dev  
  
##### Installing the following dependencies:  
● libjpeg-dev  
● zlib1g-dev  
  
[sudo] password for orangepi: [← 在这里输入linux系统的密码: orangepi]
```

15) Moonraker 具体安装过程如下

- Kiauh 中的脚本首先会自动安装依赖包，请确保下载的过程不会由于网络的原因出错

```
##### Checking for the following dependencies:  
● wget  
● curl  
● unzip  
● dfu-util  
● virtualenv  
● libjpeg-dev  
● zlib1g-dev  
  
##### Installing the following dependencies:  
● libjpeg-dev  
● zlib1g-dev  
  
[sudo] password for orangepi:  
Hit:1 http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports focal InRelease  
Hit:2 http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports focal-security InRelease  
Hit:3 http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports focal-updates InRelease  
Hit:4 http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports focal-backports InRelease  
Reading package lists... Done  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  libjpeg-turbo8-dev libjpeg8-dev  
The following NEW packages will be installed:  
  libjpeg-dev libjpeg-turbo8-dev libjpeg8-dev zlib1g-dev  
0 upgraded, 4 newly installed, 0 to remove and 124 not upgraded.
```

- 然后会自动从 GitHub 下载 Moonraker 的源码，这一步很容易由于网络问题下载失败，请特别注意下输出 log 和下图一致，没有报错



```
##### Downloading Moonraker ...
Cloning into 'moonraker'...
remote: Enumerating objects: 5413, done.
remote: Counting objects: 100% (188/188), done.
remote: Compressing objects: 100% (86/86), done.
remote: Total 5413 (delta 118), reused 144 (delta 102), pack-reused 5225
Receiving objects: 100% (5413/5413), 1.65 MiB | 1.38 MiB/s, done.
Resolving deltas: 100% (3955/3955), done.
>>>> Download complete!
```

c. 下载出错的情况如下所示

```
##### Downloading Moonraker ...
Cloning into 'moonraker'...
fatal: unable to access 'https://github.com/Arksine/moonraker.git/': GnuTLS recv error (-110): The TLS connection was non-properly terminated.
>>>> Download complete!
```

d. 然后会继续安装依赖包，请确保下载安装的过程不会由于网络的原因出错

```
##### Installing dependencies ...
##### Reading dependencies...
python3-virtualenv
python3-dev
libopenjp2-7
python3-libgpiod
curl
libcurl4-openssl-dev
libssl-dev
liblmdb-dev
libsodium-dev
zlib1g-dev
libjpeg-dev

##### Running apt-get update...
[sudo] password for orangepi:
Hit:1 http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports focal InRelease
Hit:2 http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports focal-security InRelease
Hit:3 http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports focal-updates InRelease
Hit:4 http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports focal-backports InRelease
Reading package lists... Done

##### Installing packages...
```

e. 然后会自动安装 Python 的虚拟环境，请确保下载和安装的过程不会由于网络的原因出错

```
##### Installing python virtual environment...
created virtual environment CPython3.8.10.final.0-64 in 1075ms
  creator CPython3Posix(dest=/home/orangepi/moonraker-env, clear=False, global=False)
  seeder FromAppData(download=False, pip=latest, setuptools=latest, wheel=latest, pkg_resources=eed-app-data/v1.0.1.debian.1)
  activators BashActivator,CShellActivator,FishActivator,PowerShellActivator,PythonActivator,XonshActivator
Collecting tornado==6.1.0
  Downloading tornado-6.1-cp38-cp38-manylinux2014_aarch64.whl (427 kB)
    [██████████| 427 kB 609 kB/s]
Collecting pyserial==3.4
  Using cached pyserial-3.4-py3-none-any.whl (193 kB)
Collecting pillow==8.3.2
  Downloading Pillow-8.3.2-cp38-cp38-manylinux_2_17_aarch64.manylinux2014_aarch64.whl (3.0 MB)
    [██████████| 3.0 MB 7.9 MB/s]
```

f. 安装完成后会显示下面的信息，并会自动返回 Kiauh 的安装界面



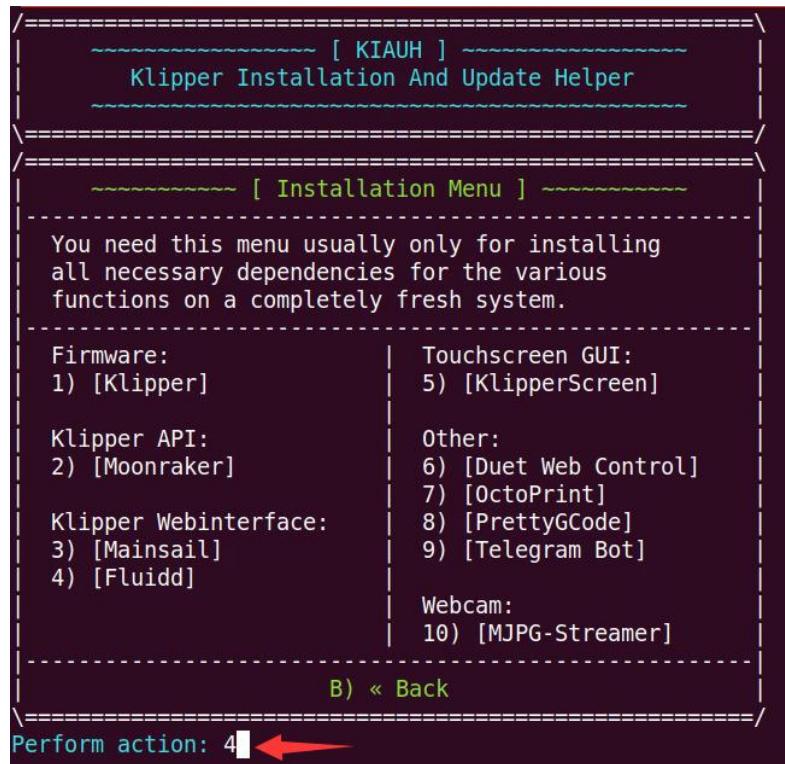
```
##### Enabling moonraker.service ...
Created symlink /etc/systemd/system/multi-user.target.wants/moonraker.service -> /etc/systemd/system/moonraker.service.
>>>> moonraker.service enabled!
>>>> Single Moonraker instance created!

##### Starting moonraker.service ...
>>>> moonraker.service started!

#####
Moonraker has been set up!
#####

● Instance 1: 192.168.1.11:7125
```

- 16) 然后开始安装 Klipper 的 Web 操作界面, 默认可选的有 **3) [Mainsail]** 和 **4) [Fluiddd]** 等, 这里我们测试下安装 **4) [Fluiddd]** 的过程, 要安装 **4) [Fluiddd]** 在 **Perform action:** 后面输入 **4**, 然后回车即可



- 17) Klipper 的 Web 操作界面——**Fluiddd** 具体安装过程如下

- a. Kiauh 中的脚本首先会自动下载安装依赖包, 请确保下载的过程不会由于网络的原因出错



```
/=====
| ~~~~~ [ KIAUH ] ~~~~~ |
| Klipper Installation And Update Helper |
\=====

##### Checking for the following dependencies:
● nginx

##### Installing the following dependencies:
● nginx

Hit:1 http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports focal InRelease
Hit:2 http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports focal-security InRelease
Hit:3 http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports focal-updates InRelease
Hit:4 http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports focal-backports InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter libnginx-mod-mail
```

- b. 然后会提示是否需要安装 MJGP-Streamer 来显示摄像头的输出视频, 如果需要直接回车即可, 如果不需要, 可以在下图所示的地方输入 **n** 然后回车。

```
##### Initializing Fluidd installation ...

/=====
| Install MJGP-Streamer for webcam support? |
\=====

##### Install MJPG-Streamer? (Y/n): █
```

- c. 这里我们选择安装 MJGP-Streamer, 然后会提示下面的信息, 回车即可

```
/=====
| Install MJGP-Streamer for webcam support? |
\=====

##### Install MJPG-Streamer? (Y/n):
##### > Yes

/=====
| It is recommended to have some important macros set |
| up in your printer configuration to have Fluidd |
| fully functional and working. |
|
| Those macros are: |
| ● [gcode_macro PAUSE] |
| ● [gcode_macro RESUME] |
| ● [gcode_macro CANCEL_PRINT] |
|
| If you already have these macros in your config file |
| you can skip this step and choose 'no'. |
| Otherwise you should consider to answer with 'yes' to |
| add the recommended example macros to your config. |
\=====

##### Add the recommended macros? (Y/n): █
```



- d. 然后会从 GitHub 下载 **fluidd.zip** 并解压安装，这一步很容易由于网络问题下载失败，请特别注意下输出 log 和下图一致，没有报错

```
##### Downloading Fluidd ...
--2022-01-13 13:33:33-- https://github.com/fluidd-core/fluidd/releases/download/v1.16.2/fluidd.zip
Resolving github.com (github.com) ... 149.82.112.4
Connecting to github.com (github.com) |149.82.112.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/295836951/5d248200-e0dd-11eb-9d39-b4a76797fd0a?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVHEH53A%2F20220101%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20220101T133334Z&X-Amz-Expires=3086X-Amz-Signature=76e8c06f5c7a375dc51fd37056c9b0f5c07b37e62665db2334b1a12361e2a746X-Amz-SignedHeaders=host&actor_id=0&repo_id=295836951&response-content-disposition=attachment%3B%2Bfilename%3Dfluidd.zip&response-content-type=application%2Foctet-stream [following]
--2022-01-13 13:33:34-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/295836951/5d248200-e0dd-11eb-9d39-b4a76797fd0a?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVHEH53A%2F20220101%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20220101T133334Z&X-Amz-Expires=3086X-Amz-Signature=76e8c06f5c7a375dc51fd37056c9b0f5c07b37e62665db2334b1a12361e2a746X-Amz-SignedHeaders=host&actor_id=0&repo_id=295836951&response-content-disposition=attachment%3B%2Bfilename%3Dfluidd.zip&response-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 9518655 (9.1M) [application/octet-stream]
Saving to: "fluidd.zip"

fluidd.zip          100%[=====]  9.08M  15.3KB/s   in 10m 54s

13:44:30 (14.2 KB/s) - 'fluidd.zip' saved [9518655/9518655]

>>>> Download complete!
>>>> Extracting archive ...
>>>> Done!
>>>> Remove downloaded archive ...
>>>> Done!
```

- e. 然后会安装依赖包，请确保下载的过程不会由于网络的原因出错

```
##### Checking for the following dependencies:
● git
● cmake
● build-essential
● imagemagick
● libv4l-dev
● ffmpeg
● libjpeg8-dev

##### Installing the following dependencies:
● cmake
● imagemagick
● libv4l-dev
● ffmpeg

Hit:1 http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports focal InRelease
Hit:2 http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports focal-security InRelease
Hit:3 http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports focal-updates InRelease
Hit:4 http://mirrors.tuna.tsinghua.edu.cn/ubuntu-ports focal-backports InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
```

- f. 然后会下载编译安装 MJPG-Streamer，这一步很容易由于网络问题下载失败，请特别注意下输出 log 和下图一致，没有报错

```
##### Downloading MJPG-Streamer ...
Cloning into 'mjpg-streamer'...
remote: Enumerating objects: 2964, done.
remote: Total 2964 (delta 0), reused 0 (delta 0), pack-reused 2964
Receiving objects: 100% (2964/2964), 3.48 MiB | 1.96 MiB/s, done.
Resolving deltas: 100% (1885/1885), done.
>>>> Download complete!

##### Compiling MJPG-Streamer ...
[ -d _build ] || mkdir _build
[ -f _build/Makefile ] || (cd _build && cmake -DCMAKE_BUILD_TYPE=Release ..)
-- The C compiler identification is GNU 9.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
```

- g. MJPG-Streamer 的访问地址如下所示，如果开发板有接 USB 摄像头，在浏



览器中输入下面的地址就能看到 USB 摄像头的视频输出了

```
#####
# MJPG-Streamer has been set up!
#####
● Webcam URL: http://192.168.1.11:8080/?action=stream
● Webcam URL: http://192.168.1.11/webcam/?action=stream
```

h. 安装完后的输出如下图所示

```
#####
# Fluidd has been set up!
#####

/===== \
| ~~~~~ [ Installation Menu ] ~~~~~ |
|-----|
| You need this menu usually only for installing |
| all necessary dependencies for the various |
| functions on a completely fresh system. |
|-----|
| Firmware: | Touchscreen GUI: |
| 1) [Klipper] | 5) [KlipperScreen] |
|-----|
| Klipper API: | Other: |
| 2) [Moonraker] | 6) [Duet Web Control] |
|-----|
| Klipper Webinterface: | 7) [OctoPrint] |
| 3) [Mainsail] | 8) [PrettyGCode] |
| 4) [Fluidd] | 9) [Telegram Bot] |
|-----|
| | Webcam: |
| | 10) [MJPG-Streamer] |
|-----|
| B) « Back |
\===== /
```

Perform action: █

18) 然后在 **Perform action:** 后面输入 **B** 就能返回安装的主界面



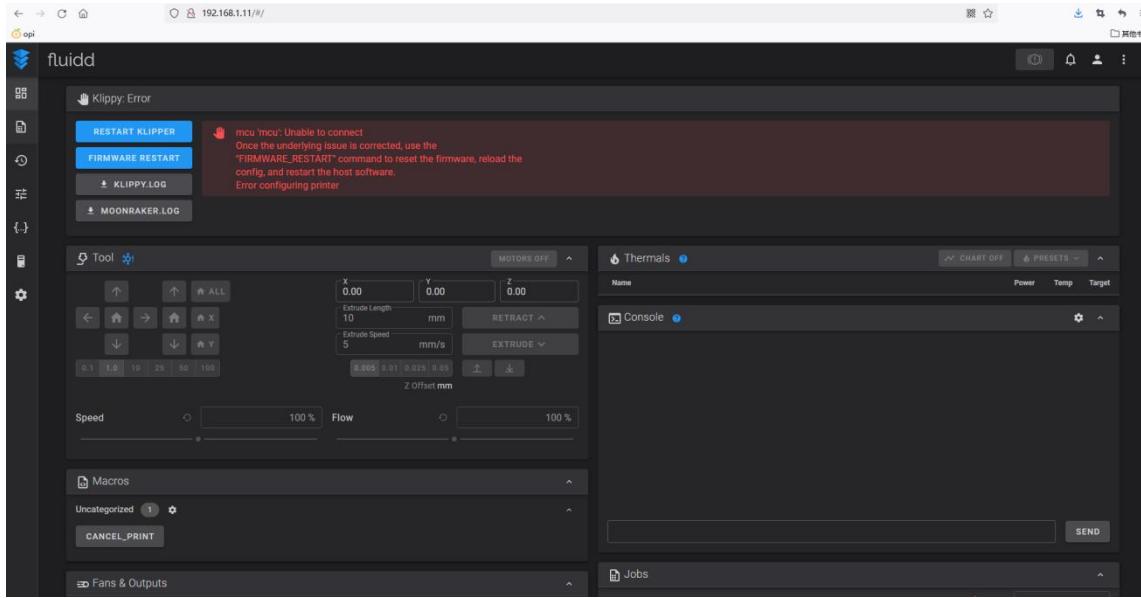
```
/===== [ Installation Menu ] =====\n\nYou need this menu usually only for installing\nall necessary dependencies for the various\nfunctions on a completely fresh system.\n\nFirmware:\n  1) [Klipper]\n\nKlipper API:\n  2) [Moonraker]\n\nKlipper Webinterface:\n  3) [Mainsail]\n  4) [Fluidd]\n\nTouchscreen GUI:\n  5) [KlipperScreen]\n\nOther:\n  6) [Duet Web Control]\n  7) [OctoPrint]\n  8) [PrettyGCode]\n  9) [Telegram Bot]\n\nWebcam:\n  10) [MJPG-Streamer]\n\nB) << Back\n\\=====/\nPerform action: b
```

```
/===== [ KIAUH ] =====\n  Klipper Installation And Update Helper\n\\=====/\n\\===== [ Main Menu ] =====\n\n  0) [Upload Log] | Klipper: Installed: 1\n  1) [Install]   | Branch: master\n  2) [Update]    | Moonraker: Installed: 1\n  3) [Remove]    | Mainsail: Not installed!\n  4) [Advanced]  | Fluidd: Installed!\n  5) [Backup]    | Klipperscreen: Not installed!\n  6) [Settings] | Telegram Bot: Not installed!\n\n  v3.1.0-85     | DWC2: Not installed!\n                  | Octoprint: Not installed!\n\nQ) Quit\n\\=====/\nPerform action: 
```

返回主界面后可以看到 **Klipper**、**Moonraker** 和 **Fluidd** 都已显示为 **Installed**。不过需要注意的是，即使 **Klipper**、**Moonraker** 和 **Fluidd** 都已显示为 **Installed** 也不能确保所有软件就都已安装成功了。测试过程中发现，就算由于网络问题导致代码下载失败后，返回主界面也能看到 **Klipper**、**Moonraker** 和 **Fluidd** 都会显示为 **Installed**。所以保证安装没问题的最好方法就是确保安装过程的输出 log 和上面的所有截图都一样，没有因为网络问题导致的源码下载失败的报错。



19) 最后在浏览器中输入开发板的 IP 地址就能看到 fluidd 的 Web 控制界面了



20) 以上演示的是使用 Kiauh 安装 Klipper 固件上位机的标准方法，如果由于网络问题导致安装失败，又无法解决开发板 Linux 系统访问 GitHub 的问题，可以考虑使用 Orange Pi 提供的 Kiauh 源码压缩包进行测试

- Kiauh 源码压缩包可以从下面的百度云盘链接下载，进入 **kiauh_klipper** 文件夹中就能看到

链接：<https://pan.baidu.com/s/1vWQmCmSYdH7iCDFyKpJtVw>

提取码：zero

返回上一级 全部文件 > orangepi-build		
□ 文件名	大小	修改日期
□ u-boot	-	2020-11-05 13:54
□ toolchains	-	2021-04-02 11:53
□ orangepi-build-h6	-	2021-12-27 16:12
□ orangepi-build	-	2020-12-09 20:37
□ OpenWRT	-	2021-04-16 14:42
□ linux我&使用的rootfs压缩包	-	2020-11-05 12:08
□ kiauh_klipper	-	2022-01-04 10:21
□ kernel	-	2020-11-05 13:54
□ orangepi-firmware-git.tar.gz	33.4M	2020-11-05 14:52

- 下载完 **kiauh.tar.gz** 压缩包后，可以使用下面的命令解压

```
orangeipi@orangeipi:~$ tar zxf kiauh.tar.gz
```

```
orangeipi@orangeipi:~$ cd kiauh
```

```
orangeipi@orangeipi:~/kiauh$ ls
```



```
docs  github_src  kiauh.sh  LICENSE  README.md  resources  scripts
```

- c. Orange Pi 提供的 Kiauh 源码的压缩包和 Kiauh 原始源码的区别如下所示
 - a) Orange Pi 提供的 Kiauh 源码中有个 `github_src` 文件夹，其中缓存了 Klipper 等软件安装过程中需要的源码和压缩包以及 USB 摄像头使用需要的配置文件

```
orangeipi@orangeipi:~/kiauh$ ls github_src/
DuetWebControl-SD.zip  fluidd.zip  mainsail.zip  moonraker  webcamd
dwc2-for-klipper-socket  mjpg-streamer  moonraker-telegram-bot  webcam.txt
klipper
```

- b) 修改了部分 kiauh 的脚本代码，通过 `git status .` 命令可以看到所有修改的文件

```
orangeipi@orangepirzero2:~/kiauh$ git status .
On branch master
Your branch is up to date with 'origin/master'.
```

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

```
modified:   kiauh.sh
modified:   scripts/dwc2-for-klipper-socket-installer/install-debian.sh
modified:   scripts/dwc2-for-klipper-socket-installer/install-octopi.sh
modified:   scripts/install_dwc2.sh
modified:   scripts/install_klipper.sh
modified:   scripts/install_klipper_webui.sh
modified:   scripts/install_mjpg-streamer.sh
modified:   scripts/install_moonraker-telegram-bot.sh
modified:   scripts/install_moonraker.sh
modified:   scripts/install_octoprint.sh
modified:   scripts/update.sh
```

Untracked files:

(use "git add <file>..." to include in what will be committed)

```
github_src/
```



c) 修改的代码实现的功能主要有

- i. 禁止 Kiauh 代码自动更新，kiauh.sh 每次运行时，会自动和 GitHub 同步代码，如果网络有问题，会导致卡住不动。如果需要自动更新的话，可以把下面修改的代码改回来

```
orangepi@orangepi:~/kiauh$ git diff kiauh.sh
diff --git a/kiauh.sh b/kiauh.sh
index 0a61f54..a9b51da 100755
--- a/kiauh.sh
+++ b/kiauh.sh
-kiauh_status
+#kiauh_status
```

- ii. 将 pip 的安装源改为 <https://pypi.tuna.tsinghua.edu.cn/simple>，加快 python 库的安装
- iii. 由于 **github_src** 文件夹中缓存了需要从 GitHub 下载的源码等文件，所以屏蔽了 Kiauh 脚本中部分需要从 GitHub 下载源码的代码，并修改为使用 cp 命令从 **github_src** 中复制到对应位置。这样就无需通过网络来从 GitHub 下载部分安装 Klipper 等软件需要的源码了。比如 **install_klipper.sh** 脚本修改了下面的两行代码

```
orangepi@orangepi:~/kiauh$ git diff scripts/install_klipper.sh
diff --git a/scripts/install_klipper.sh b/scripts/install_klipper.sh
index 4d6ade0..6bbfb09 100755
--- a/scripts/install_klipper.sh
+++ b/scripts/install_klipper.sh
- cd "${HOME}" && git clone "$KLIPPER_REPO"
+ #cd "${HOME}" && git clone "$KLIPPER_REPO"
+ cd "${HOME}" && cp -r ${GITHUB_SRC}/klipper .
```

21) 使用 Orange Pi 提供的 Kiauh 源码压缩包来安装 Klipper 和使用前面演示的标准方法来安装没有任何区别，只是部分下载源码的 log 输出会有所不同，比如安装 Klipper 时，下载 Klipper 源码这里是没有 log 输出的

```
##### Downloading Klipper ...
#####
Download complete!
```

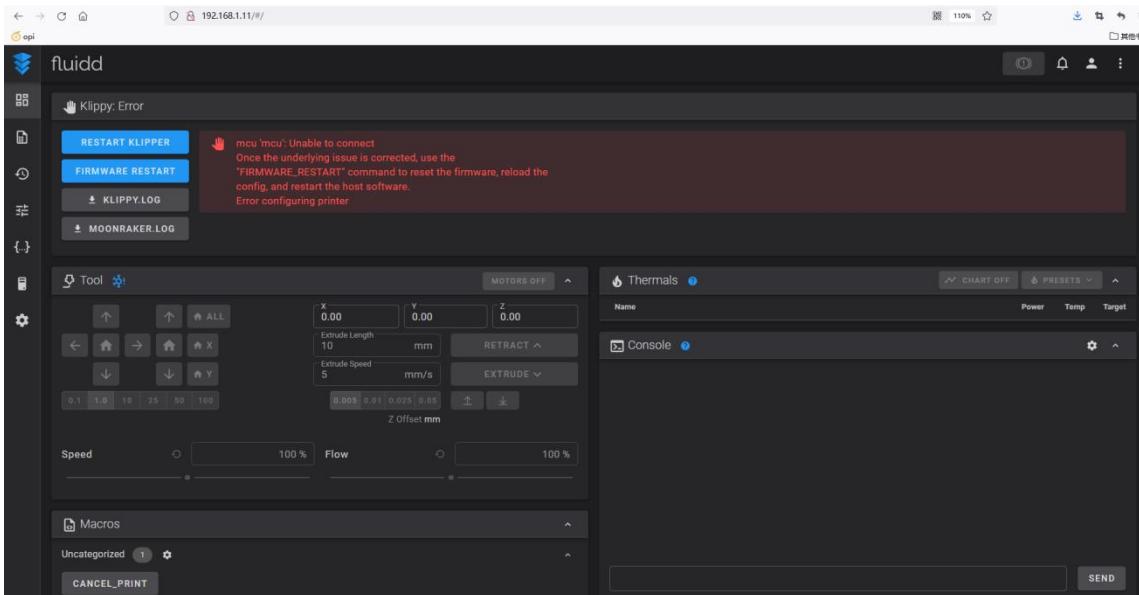
22) 使用 Orange Pi 提供的 Kiauh 源码压缩包安装 Fluddd 示例



- a. 首先请安装 **1) [Klipper]** 和 **2) [Moonraker]**
- b. 安装 **4) [Fluidd]** 选项如下所示

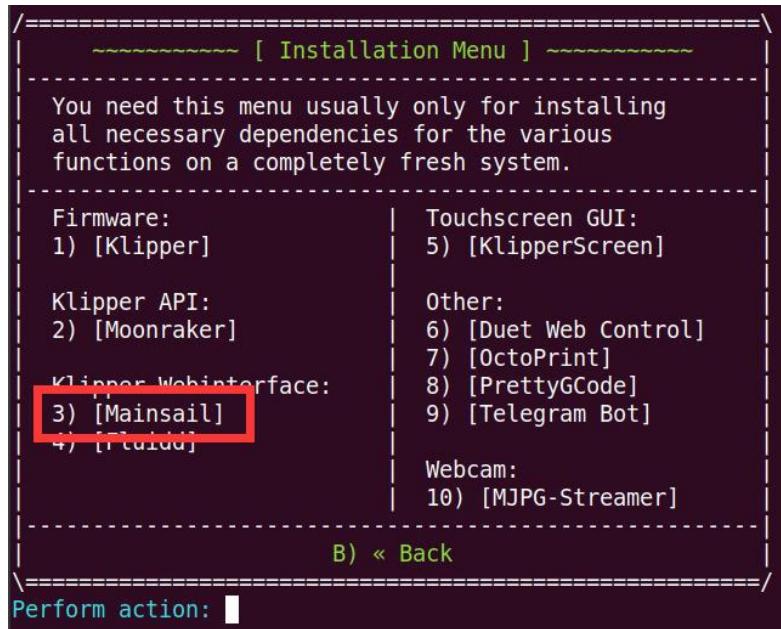
```
/=====
| ===== [ Installation Menu ] ===== |
|-----|
| You need this menu usually only for installing |
| all necessary dependencies for the various |
| functions on a completely fresh system. |
|-----|
| Firmware: | Touchscreen GUI: |
| 1) [Klipper] | 5) [KlipperScreen] |
|-----|
| Klipper API: | Other: |
| 2) [Moonraker] | 6) [Duet Web Control] |
|-----|
| Klipper Webinterface: | 7) [OctoPrint] |
| 3) [Mainsail] | 8) [PrettyGCode] |
|-----|
| 4) [Fluidd] | 9) [Telegram Bot] |
|-----|
| Webcam: |
| 10) [MJPG-Streamer] |
|-----|
| B) << Back |
\=====/
Perform action: 4
```

- c. 安装完后在浏览器中输入开发板的 IP 地址就能看到 Fluidd 的 Web 控制界面

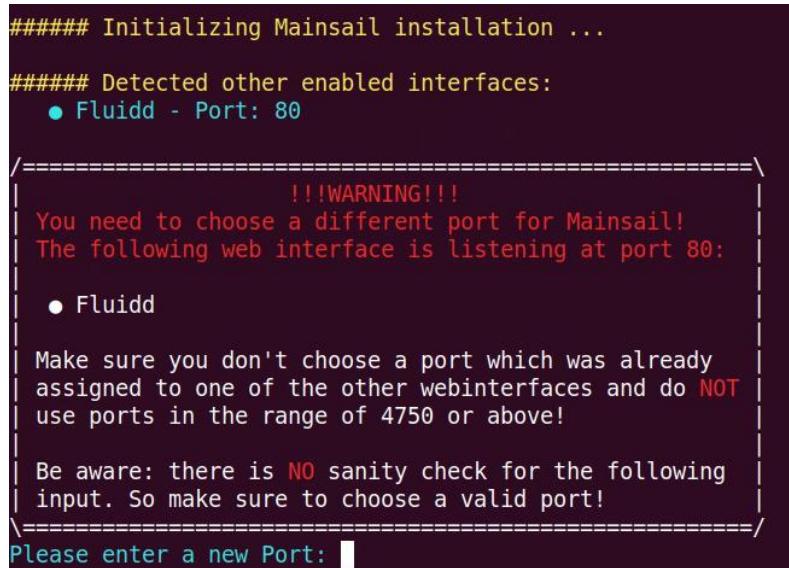


23) 使用 Orange Pi 提供的 Kiauh 源码压缩包安装 Mainsail 示例

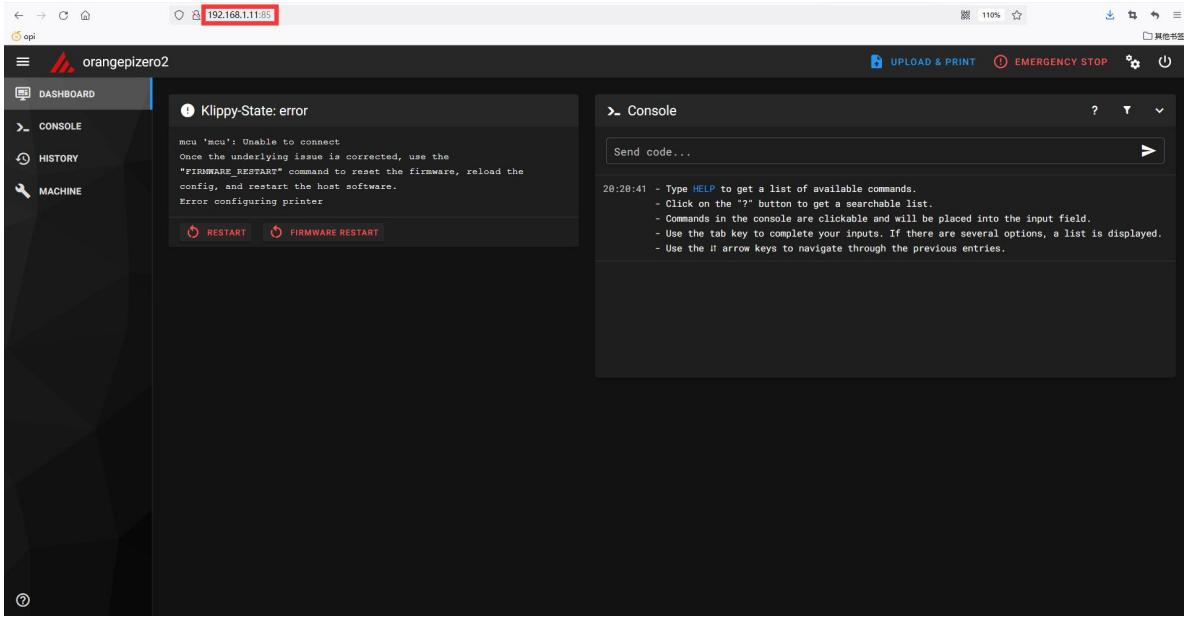
- a. 首先请安装 **1) [Klipper]** 和 **2) [Moonraker]**
- b. 安装 **3) [Mainsail]** 选项如下所示



- c. 如果前面已经安装了 Fluide, 80 端口就已经被占用了, 首先需要设置下端口号, 比如设置为 85

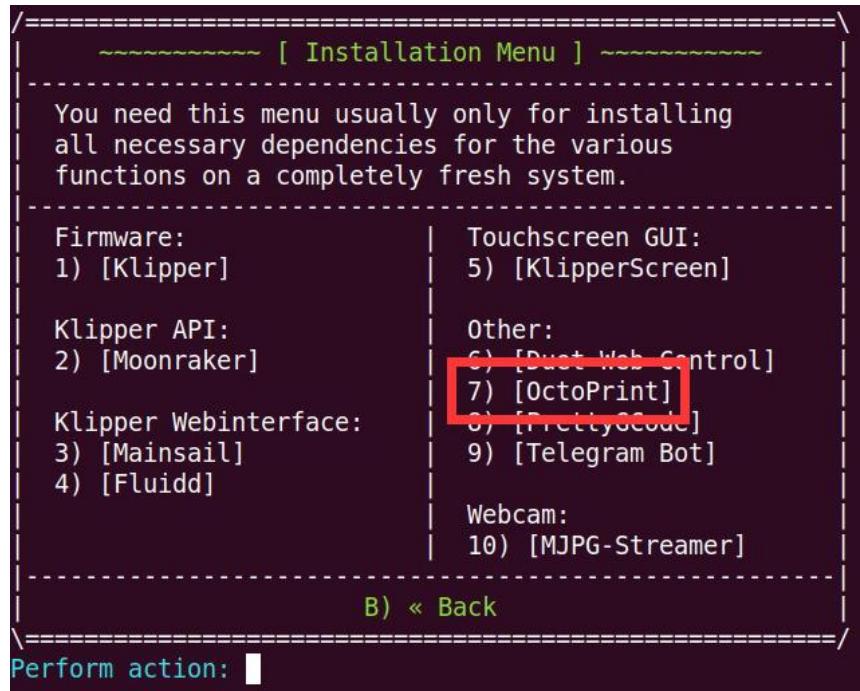


- d. 安装完后在浏览器中输入开发板的 IP 地址:端口号 就能看到 Mainsail 的 Web 控制界面



24) 使用 Orange Pi 提供的 Kiauh 源码压缩包安装 OctoPrint 示例

- 首先请安装 1) [Klipper] 和 2) [Moonraker]
- 安装 7) [OctoPrint] 选项如下所示

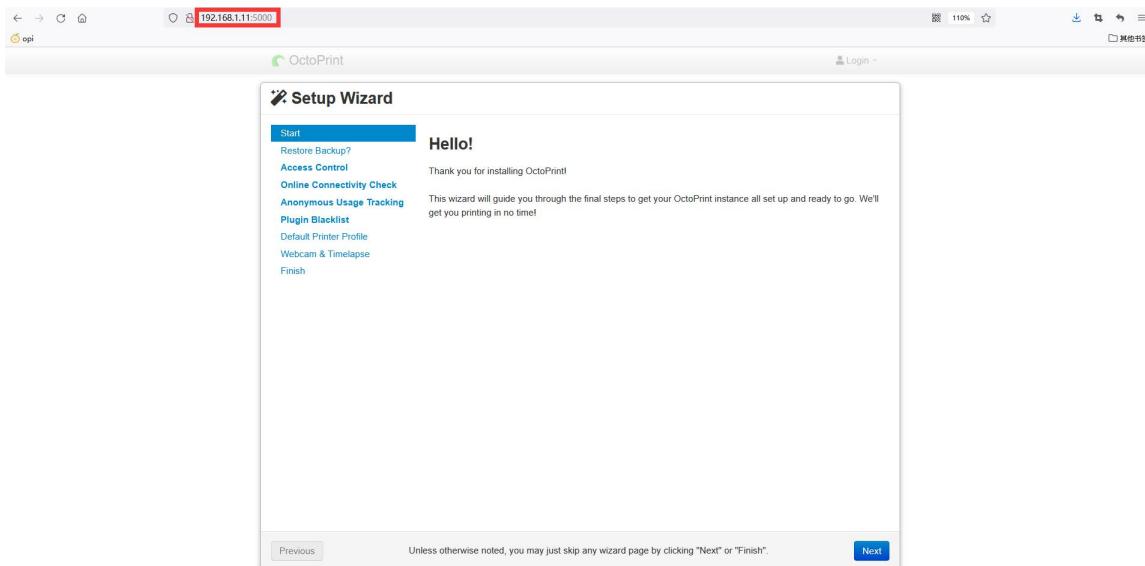


- 安装完后就可以看到 OctoPrint 的 Web 界面访问地址，端口号为 5000

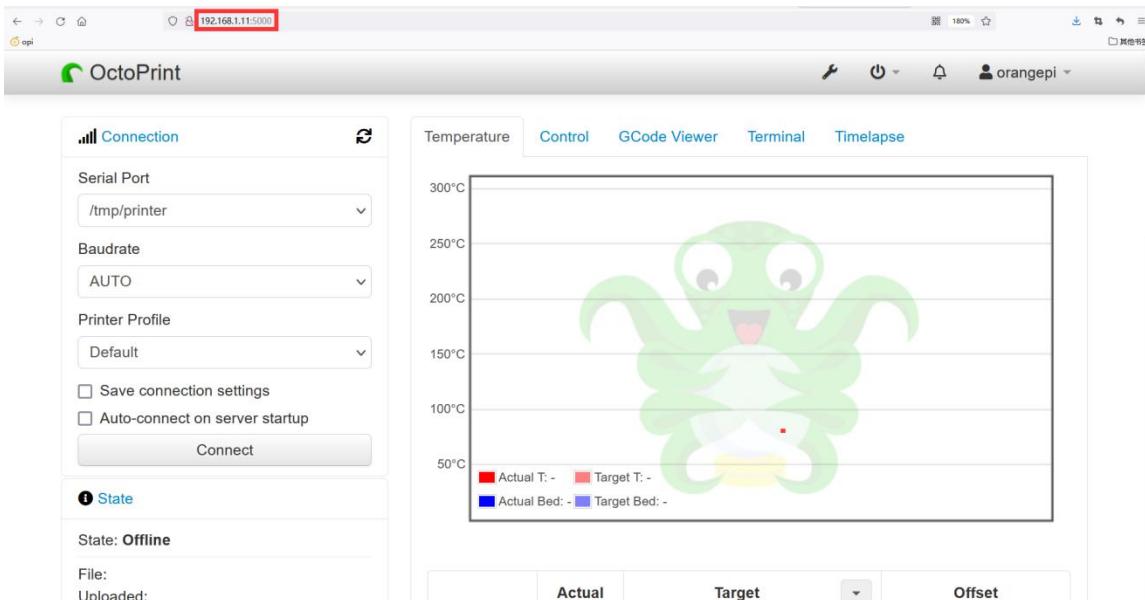


```
#####
Single OctoPrint instance has been set up!
#####
● Instance 1: 192.168.1.11:5000
```

d. 然后在浏览器中输入上图显示的地址就能看到 OctoPrint 的 Web 控制界面



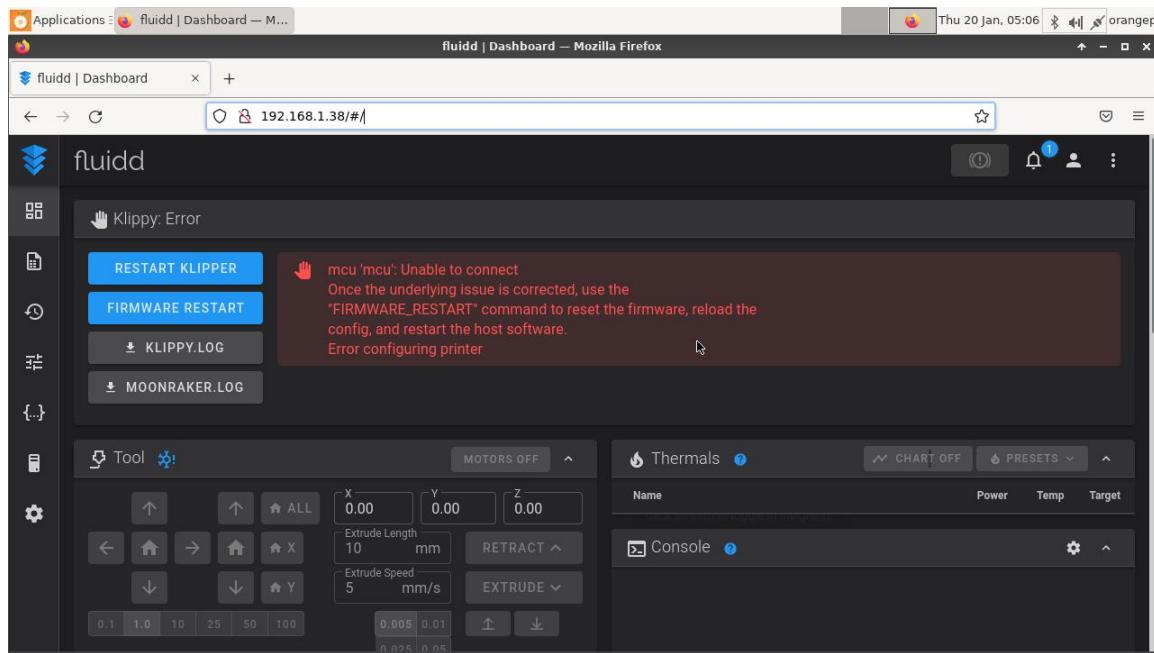
e. 根据 OctoPrint 的安装向导设置完后的界面显示如下所示



25) 如果使用的是桌面版的 Linux 系统安装的 Klipper，我们还可以把开发板通过 HDMI 线接到 HDMI 显示器或者电视上，然后打开 Linux 系统中的浏览器，再在浏览器中输入开发板的 IP 地址后也是可以看到 Fluidd（前提是已经安装了 Fluidd）的



Web 管理界面的。如果当安装完 Klipper 后发现打开 Web 界面提示了错误无法正常连接，这种情况一般都是 Klipper 等软件的安装过程中没有仔细按照手册的说明对比查看源码的下载和软件包的安装等有没有报错就以为安装全部正常导致的。此时请在 Kiauh 中先卸载（Kiauh 中有 Remove 的选项）所有已安装的软件，然后再重新安装，并仔细查看安装过程的输出打印信息是否有错误，确保没问题后再进行下一步操作。



有关 Klipper 怎么连接微控制器以及 3D 打印机的使用本小节就无法具体测试了，请自行查找相关的资料或者视频进行测试和调试

26) 相关资料

Klipper 源码: <https://github.com/Klipper3d/klipper>

Klipper 官方文档: <http://www.klipper3d.org>

Kiauh 源码: <https://github.com/th33xitus/kiauh>

3. 45. 宝塔 Linux 面板的安装方法

宝塔 Linux 面板是提升运维效率的服务器管理软件，支持一键 LAMP/LNMP/集群/监控/网站/FTP/数据库/JAVA 等 100 多项服务器管理功能（摘抄自[宝塔官网](#)）

- 1) 宝塔 Linux 系统兼容性推荐的顺序为



Debian10 > Ubuntu 20.04 > Ubuntu 18.04 > 其它系统

2) 安装宝塔前, 请确保开发板的 linux 系统是新安装, 另外请先关闭开发板 linux 系统的 orangepi-ramlog 和 orangepi-zram 服务, 如果不关闭, 安装宝塔时会报设备空间不足的错误。在开发板的 linux 系统中输入下面的两条命令就可以关闭 orangepi-ramlog 和 orangepi-zram 服务

```
orangepi@orangepi:~$ sudo sed -i "s/ENABLED=true/ENABLED=false/"  
    /etc/default/orangepi-ramlog          #这是一条命令  
orangepi@orangepi:~$ sudo sed -i "s/ENABLED=true/ENABLED=false/"  
    /etc/default/orangepi-zram-config      #这是一条命令
```

另外还需要屏蔽下面的配置, 设置完后需要重启开发板的 linux 系统

```
orangepi@orangepi:~$ sudo vim /etc/fstab  
#tmpfs /tmp tmpfs defaults,nosuid 0 0      #屏蔽 tmpfs 的配置
```

3) 重启后, 登录 linux 系统输入下面的命令就可以开始宝塔的安装

```
orangepi@orangepi:~$ wget -O install.sh  
http://download.bt.cn/install/install-ubuntu_6.0.sh && sudo bash install.sh
```

4) 然后宝塔安装程序会提醒是否安装 **Bt-Panel** 到 **/www** 文件夹, 此时输入 **y** 即可

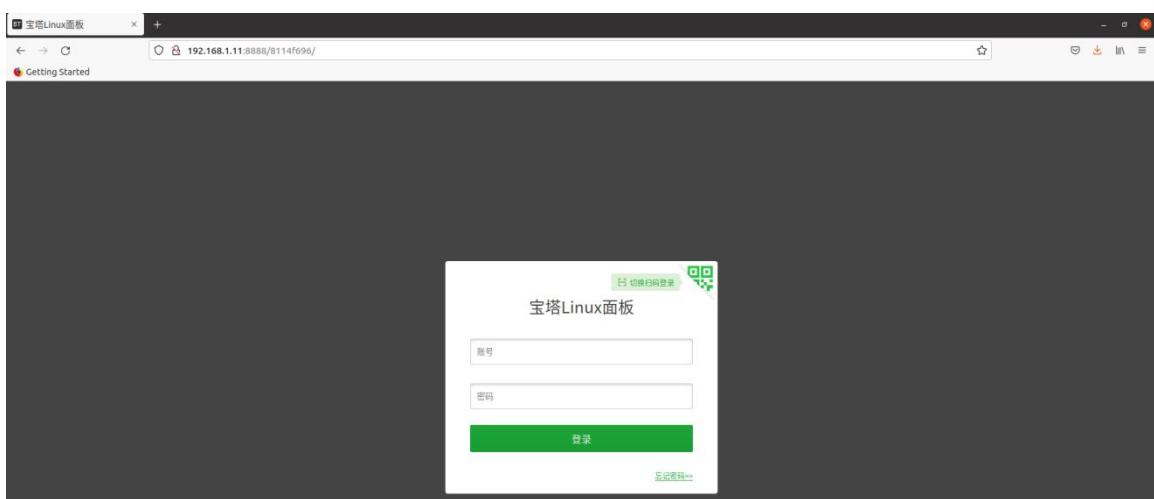
```
+-----  
| Bt-WebPanel FOR CentOS/Ubuntu/Debian  
+-----  
| Copyright © 2015-2099 BT-SOFT(http://www.bt.cn) All rights reserved.  
+-----  
| The WebPanel URL will be http://SERVER_IP:8888 when installed.  
+-----  
  
Do you want to install Bt-Panel to the /www directory now?(y/n): y
```

5) 然后要做的就是耐心等待, 当看到终端输出下面的打印信息时, 说明宝塔已经安装完成, 整个安装过程大约耗时 59 分钟, 根据网络速度的不同可能会有一些差别



```
=====
Congratulations! Installed successfully!
=====
外网面板地址: http://[240e:3b7:3240:c3a0:d81f:613c:1739:3d6a]:8888/8114f696
内网面板地址: http://192.168.1.11:8888/8114f696
username: klpyxjy6
password: ae287263
If you cannot access the panel,
release the following panel port [8888] in the security group
若无法访问面板, 请检查防火墙/安全组是否有放行面板[8888]端口
=====
Time consumed: 59 Minute!
orangepi@orangepirzero2:~$
```

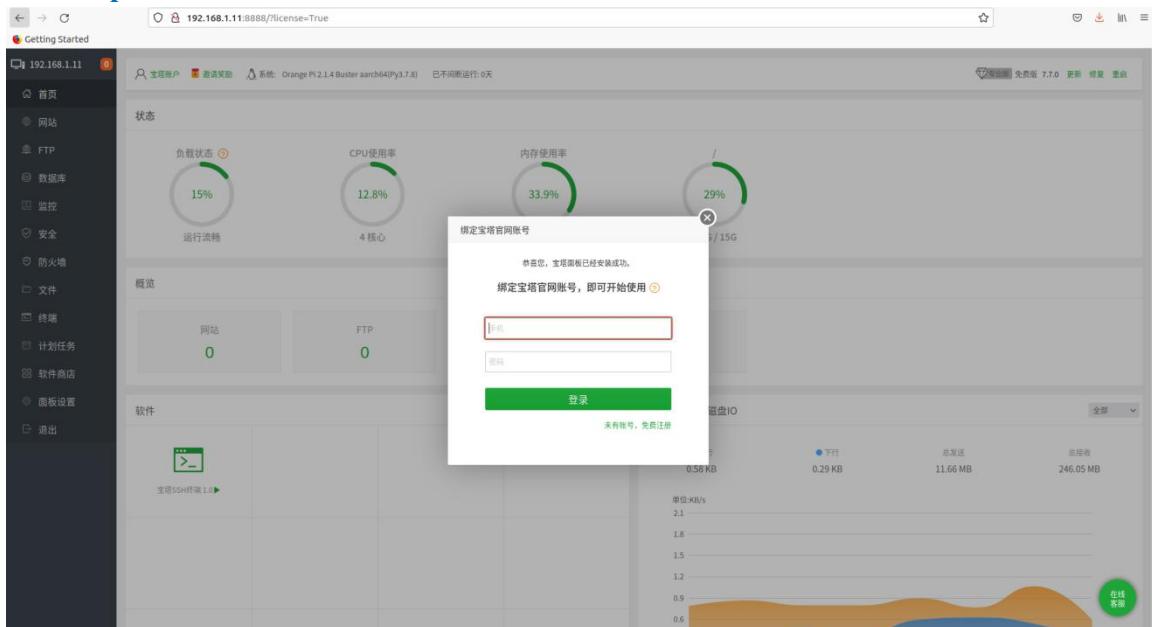
6) 此时在浏览器中输入上面显示的面板地址就可以打开宝塔 Linux 面板的登录界面，然后在对应的位置输入上图显示的 **username** 和 **password** 就可以登录进宝塔



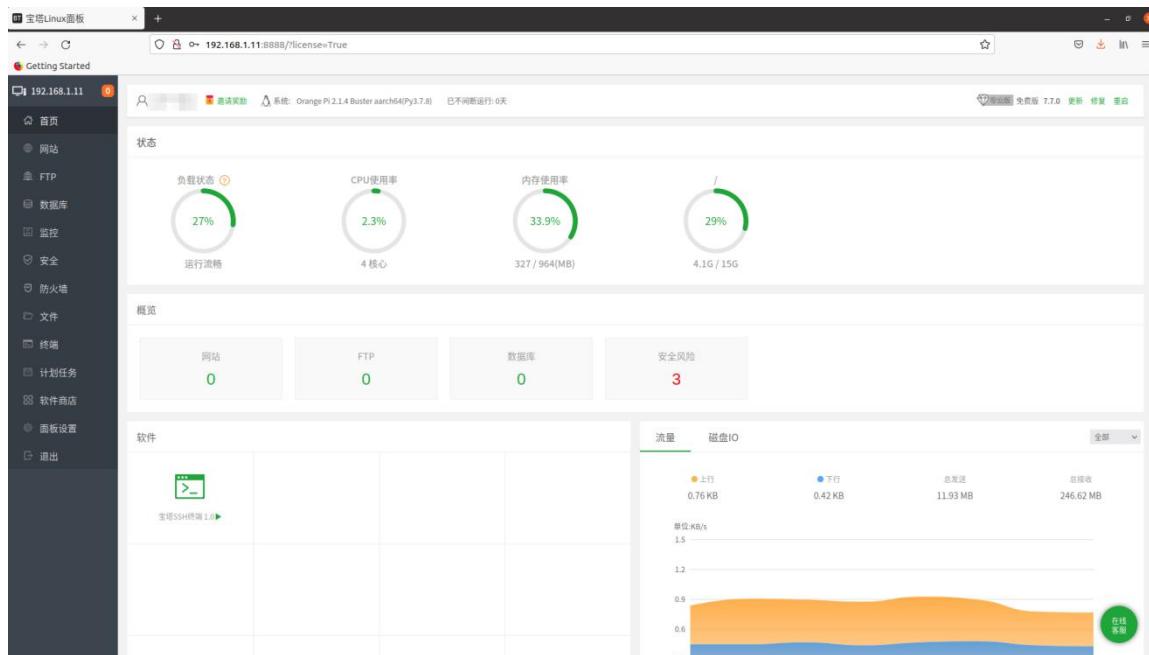
7) 成功登录宝塔后的会弹出下面的欢迎界面，首先请将中间的用户须知阅读完拖到最下面，然后就可以选择“我已同意并阅读《用户协议》”，接着点击“进入面板”就可以进入宝塔了



8) 进入宝塔后首先会提示需要绑定宝塔官网的账号，如果没有账号可以去宝塔的官网 (<https://www.bt.cn>) 注册一个

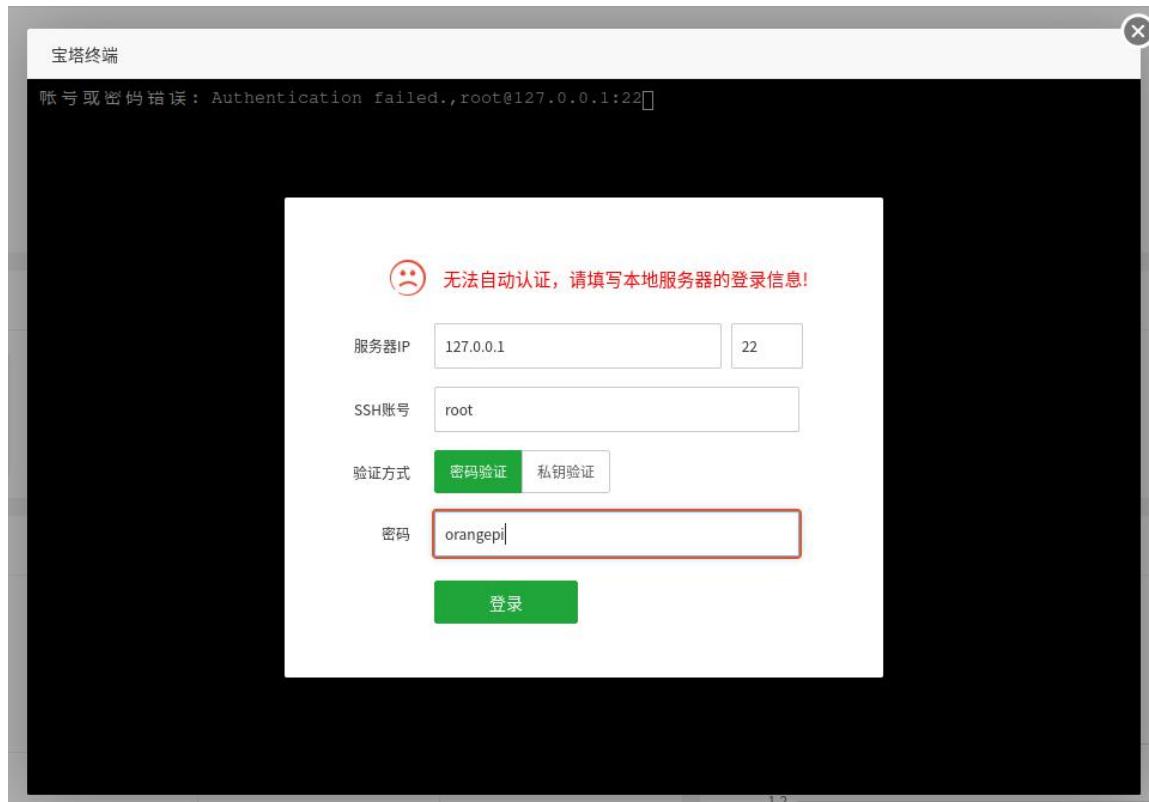


9) 最终显示的界面如下图所示，可以很直观的看到开发板 Linux 系统的一些状态信息，比如负载状态、CPU 的使用率、内存使用率和存储空间的使用情况等

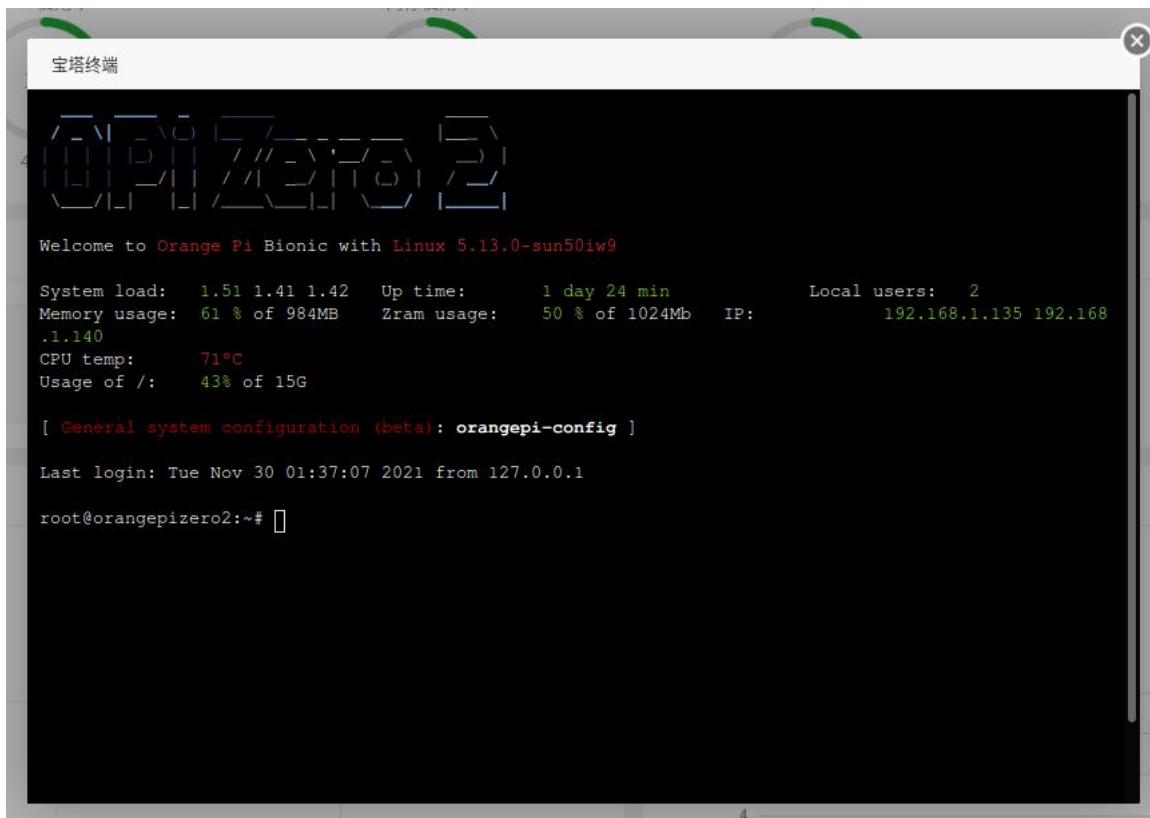


10) 测试宝塔的 SSH 终端登录

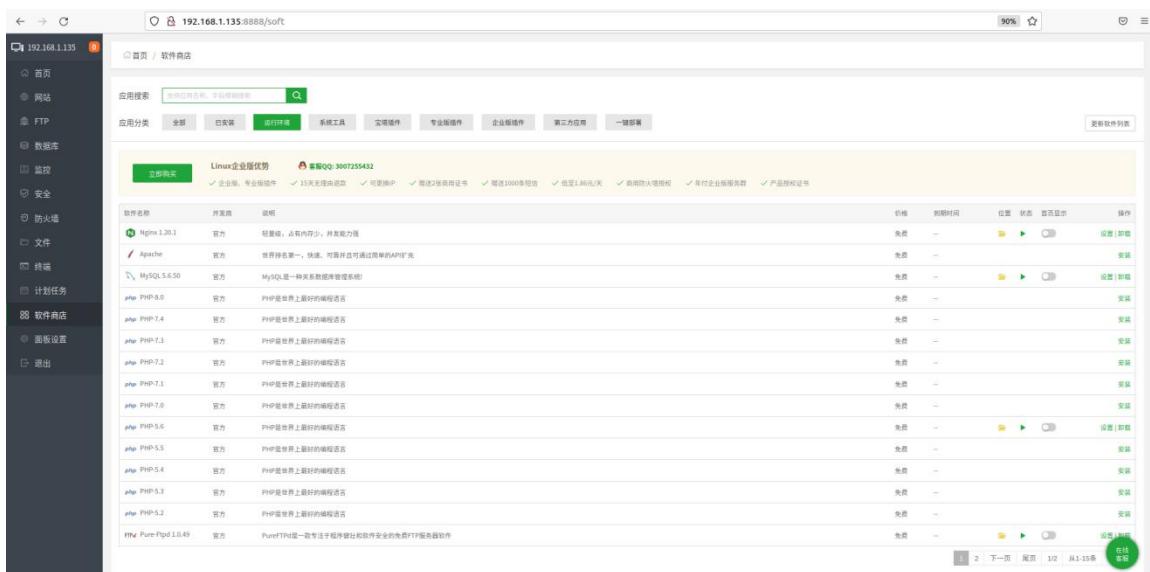
- 打开宝塔的 SSH 终端后首先会提示需要输入开发板系统的密码，此时在密码框中输入 **orangeipi**（默认密码，如果有修改请填写修改后的）即可



- 成功登录后的显示如下图所示



11) 在宝塔的软件商店中可以安装 Apache、MySQL 和 PHP 等软件，也可以一键部署各种应用程序，这部分功能请自行探索，这里就不一一演示了



12) 宝塔命令行工具测试



```
root@orangepizero2:~# bt
=====
宝塔面板命令行 =====
(1) 重启面板服务          (8) 改面板端口
(2) 停止面板服务          (9) 清除面板缓存
(3) 启动面板服务          (10) 清除登录限制
(4) 重载面板服务          (11) 取消入口限制
(5) 修改面板密码          (12) 取消域名绑定限制
(6) 修改面板用户名        (13) 取消IP访问限制
(7) 强制修改MySQL密码      (14) 查看面板默认信息
(22) 显示面板错误日志      (15) 清理系统垃圾
(23) 关闭BasicAuth认证      (16) 修复面板(检查错误并更新面板文件到最新版)
(24) 关闭谷歌认证          (17) 设置日志切割是否压缩
(25) 设置是否保存文件历史副本 (18) 设置是否自动备份面板
(0) 取消
=====
请输入命令编号: 14
=====
正在执行(14)...
=====
BT-Panel default info!
=====
外网面板地址: http://[240e:3b7:3240:c3a0:d81f:613c:1739:3d6a]:8888/8114f696
内网面板地址: http://192.168.1.11:8888/8114f696
*以下仅为初始默认账户密码, 若无法登录请执行bt命令重置账户/密码登录
username: klpypyjy6
password: ae287263
If you cannot access the panel,
release the following panel port [8888] in the security group
若无法访问面板, 请检查防火墙/安全组是否有放行面板[8888]端口
=====
root@orangepizero2:~#
```

13) 宝塔的更多功能可以参考下面资料自行探索

使用手册: <http://docs.bt.cn>

论坛地址: <https://www.bt.cn/bbs>

GitHub 链接: <https://github.com/aaPanel/BaoTa>

3. 46. Linux 系统支持的部分编程语言测试

1) 测试的镜像全名如下所示

[Orangepizero2_2.1.8_debian_buster_desktop_linux4.9.170.7z](#)

2) Debian Buster 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



```
root@orangepi:~# gcc --version
gcc (Debian 8.3.0-6) 8.3.0
Copyright (C) 2018 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

```
root@orangepi:~# cat 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**

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

3) Debian Buster 默认安装有 Python2 和 Python3

a. Python 具体版本如下所示

```
root@orangepi:~# python
Python 2.7.16 (default, Oct 10 2019, 22:02:15)
[GCC 8.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
root@orangepi:~# python3
Python 3.7.3 (default, Jan 22 2021, 20:04:44)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

b. 编写 Python 语言的 **hello_world.py** 程序

```
root@orangepi:~# vim hello_world.py
```



```
print('Hello World!')
```

c. 运行 **hello_world.py** 的结果如下所示

```
root@orangepi:~# python hello_world.py
```

```
Hello World!
```

```
root@orangepi:~# python3 hello_world.py
```

```
Hello World!
```

4) Debian Buster 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 Buster 中默认版本为 openjdk-11

```
root@orangepi:~# apt install -y openjdk-11-jdk
```

b. 安装完后可以查看下 Java 的版本

```
root@orangepi:~# java --version
```

```
openjdk 11.0.13 2021-10-19
```

```
OpenJDK Runtime Environment (build 11.0.13+8-post-Debian-1deb10u1)
```

```
OpenJDK 64-Bit Server VM (build 11.0.13+8-post-Debian-1deb10u1, mixed mode)
```

c. 编写 Java 版本的 **hello_world.java**

```
root@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**

```
root@orangepi:~# javac hello_world.java
```

```
root@orangepi:~# java hello_world
```

```
Hello World!
```



4. Android10 系统使用说明

4. 1. 已支持的 Android 版本

Android 版本	内核版本
Android 10.0 TV 版	linux4.9

4. 2. Android 10 功能适配情况

功能	状态
HDMI 视频	OK
HDMI 音频	OK
USB2.0 x 3	OK
TF 卡启动	OK
网卡	OK
红外	OK
WIFI	OK
WIFI hotspot	OK
蓝牙	OK
BLE 蓝牙	OK
耳机音频	OK
TV-OUT	OK
USB 摄像头	OK
LED 灯	OK
温度传感器	OK
硬件看门狗	OK
Mali GPU	OK
视频编解码	OK

4. 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	灭
GPIO 口	PC13	PC12

4. 4. Android 返回上一级界面的方法

- 1) 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
- 2) 如果有购买开发板配套的红外遥控（其他遥控不行）和扩展板，将扩展板插入开发板后，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



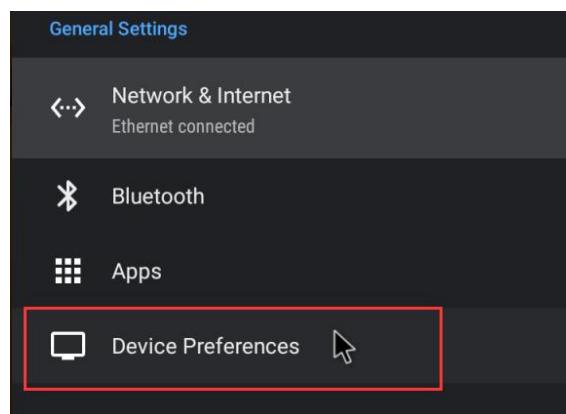
4. 5. ADB 的使用方法

4. 5. 1. 打开 USB debugging 选项

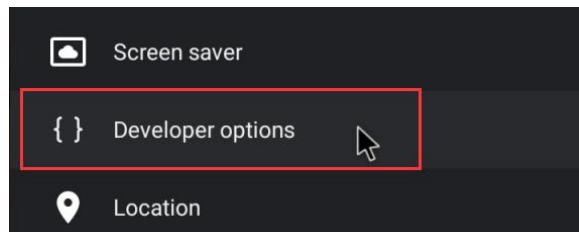
- 1) 选择 **Settings**



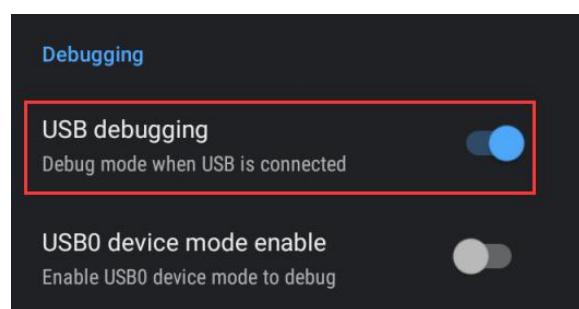
2) 然后选择 **Device Preferences**



3) 然后选择 **Developer options**



4) 最后找到 **USB debugging**, 确保其已经打开





4.5.2. 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) 确保已经打开 **USB debugging** 选项

2) 确保 Android 系统的 **service.adb.tcp.port** 设置为 5555 端口号

```
cupid-p2:/ # getprop | grep "adb.tcp"  
[service.adb.tcp.port]: [5555]
```

3) 如果 **service.adb.tcp.port** 没有设置，可以使用下面的命令设置网络 adb 的端口号

```
cupid-p2:/ # setprop service.adb.tcp.port 5555  
cupid-p2:/ # stop adbd  
cupid-p2:/ # start adbd
```

4) 在 Ubuntu PC 上安装 adb 工具

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

5) 然后在 Ubuntu PC 上连接网络 adb

```
test@test:~$ adb connect 192.168.1.xxx  (IP 地址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
```

6) 然后在 Ubuntu PC 上通过 adb shell 就可以登录 android 系统

```
test@test:~$ adb shell  
cupid-p2:/ #
```



4. 5. 3. 使用数据线连接 adb 调试

1) 首先确保打开 **USB debugging 选项**

2) 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电源接口中。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



3) 在 Ubuntu PC 上安装 adb 工具

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

4) 查看识别到 ADB 设备

```
test@test:~$ adb devices  
List of devices attached  
8c00141167058911ccd    device
```

5) 然后在 Ubuntu PC 上通过 adb shell 就可以登录 android 系统

```
test@test:~$ adb shell  
cupid-p2:/ #
```

4. 6. 香橙派 5 寸 TFT 液晶屏测试

1) 首先准备好香橙派的 5 寸 TFT 液晶屏，屏幕排线和转接板的接线方式如下图所示，请勿接反了，另外请确保屏幕排线和排线插座接触都到位了，如果接触不到位，屏幕输出会有问题



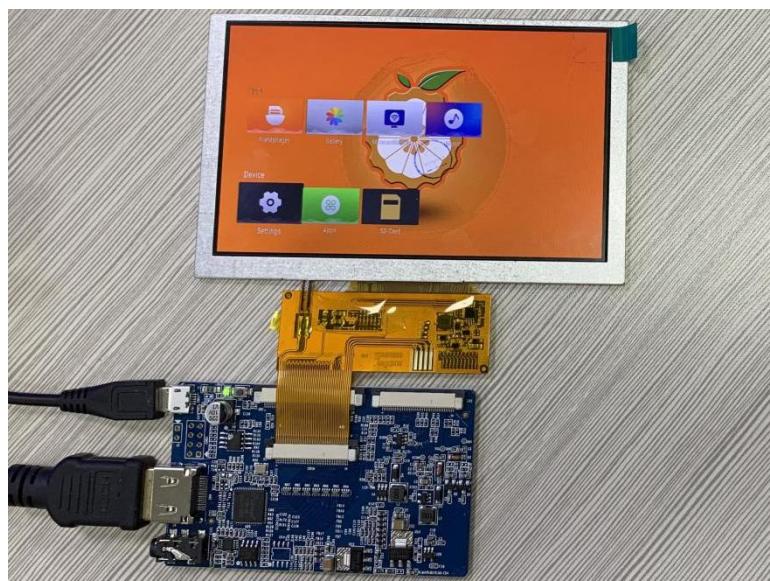
香橙派



2) 然后使用 Micro HDMI 线将开发板的 Micro HDMI 接口连接到转接板的 HDMI 接口中，屏幕需要单独供电，请在转接板的 Micro USB 接口中插入 5V/2A 的电源



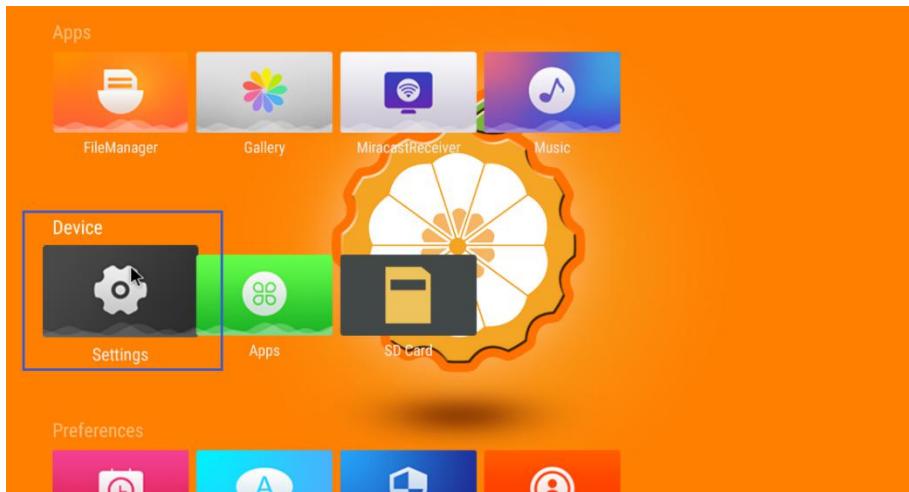
3) 开发板启动后，屏幕就可以看到 Android 系统的桌面了



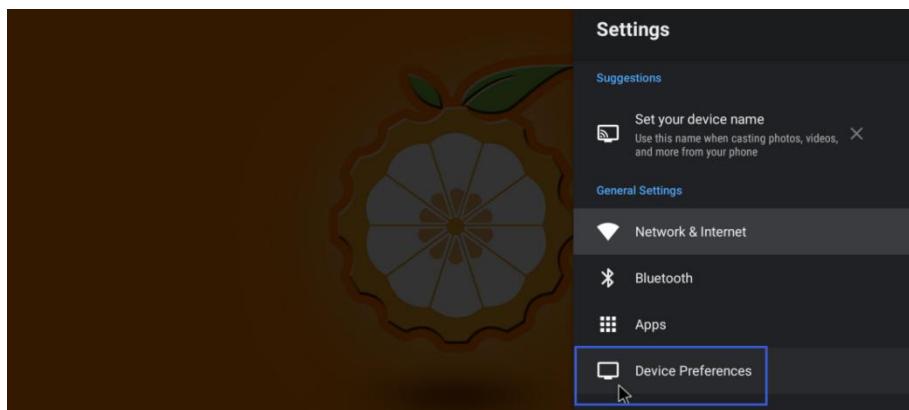


4) Android 系统第一次启动时，屏幕显示有抖动或者模糊不清是正常的，因为 Android 系统 HDMI 分辨率默认为 1080p，但是 TFT 液晶屏不支持这个分辨率，第一次启动进入 Android 系统后请首先修改 HDMI 输出的分辨率为 **480p**，然后屏幕显示就会很稳定了

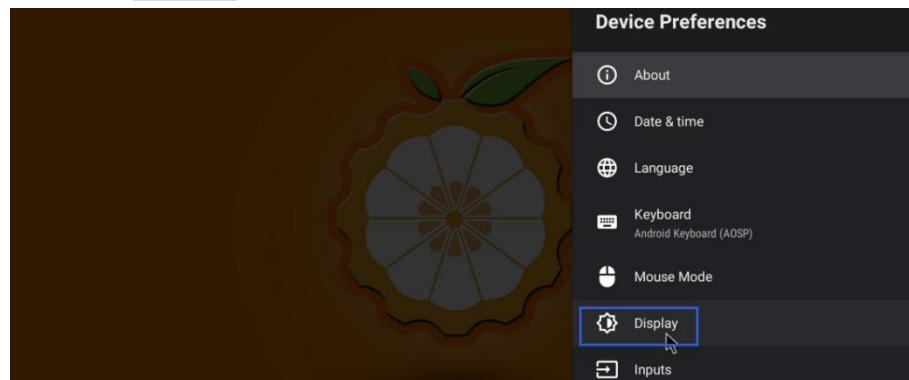
a. 首先选择 **Setting**



b. 然后选择 **Device Preferences**



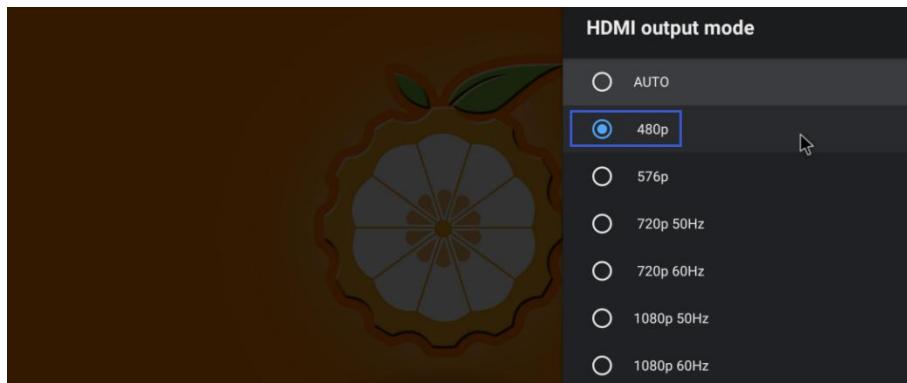
c. 然后选择 **Display**



d. 然后选择 **HDMI output mode**



- e. 最后选择 480p 即可将 HDMI 的分辨率设置为 480p

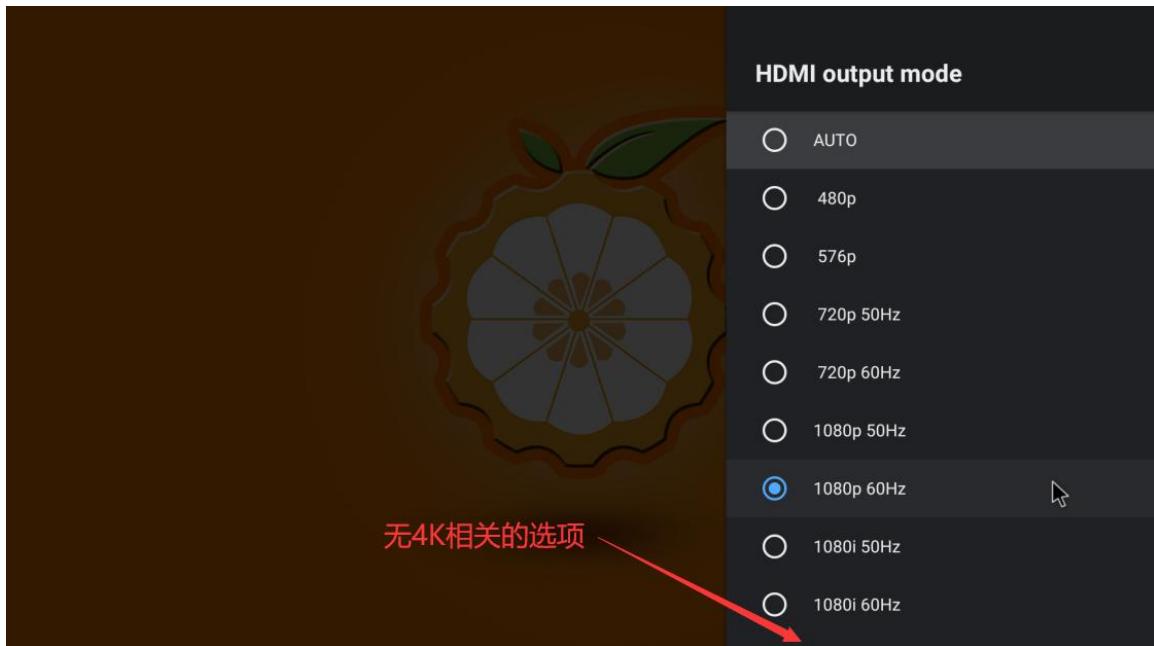


- f. 此时再查看液晶屏的显示输出就会很稳定了

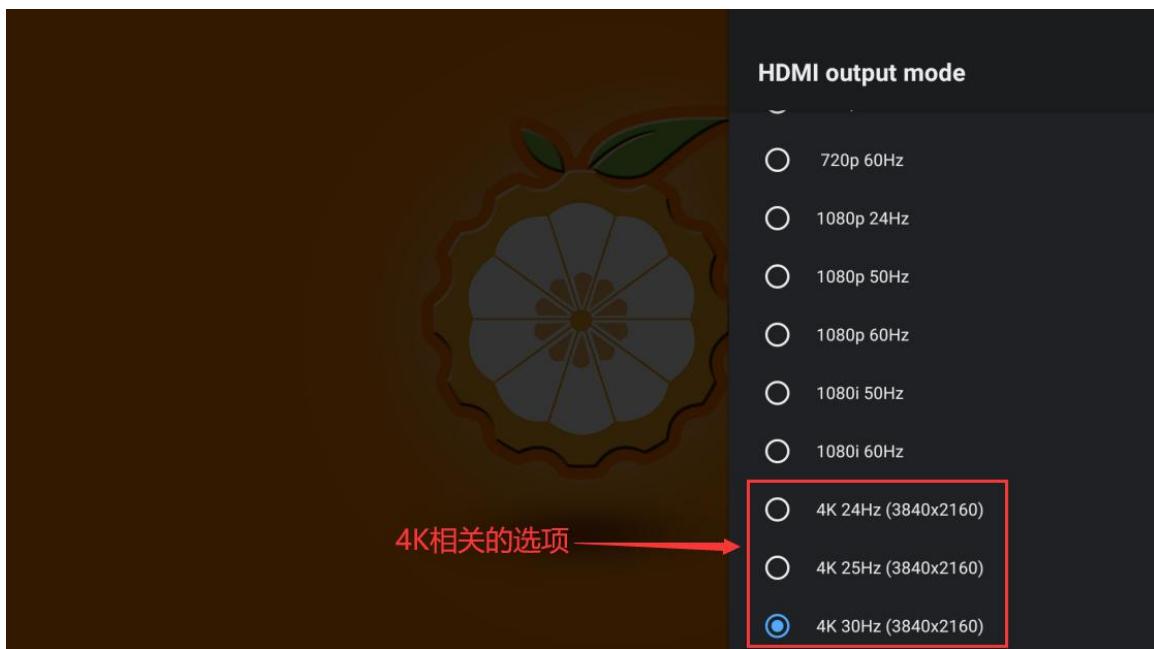
关闭开发板的电源后，请记得同时关闭屏幕转接板的电源，下次启动开发板前再打开

4.7. HDMI 4K 显示说明

- 1) 如果开发板的 Micro HDMI 接到不支持 4K 的电视上或者显示器上，在设置中查看 HDMI 支持的分辨率时是看不到 4K 相关的选项的



- 2) 只有将开发板的 Micro HDMI 接到支持 4K 的电视上或者显示器上，在 HDMI 支持的分辨率中才能看到 4K 相关的选项



4.8. HDMI 转 VGA 显示测试

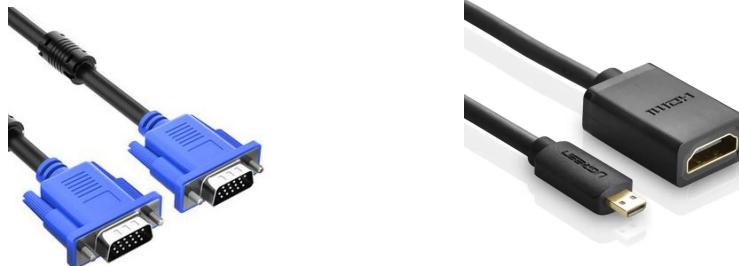
- 3) 首先需要准备下面的配件



a. HDMI 转 VGA 转换器

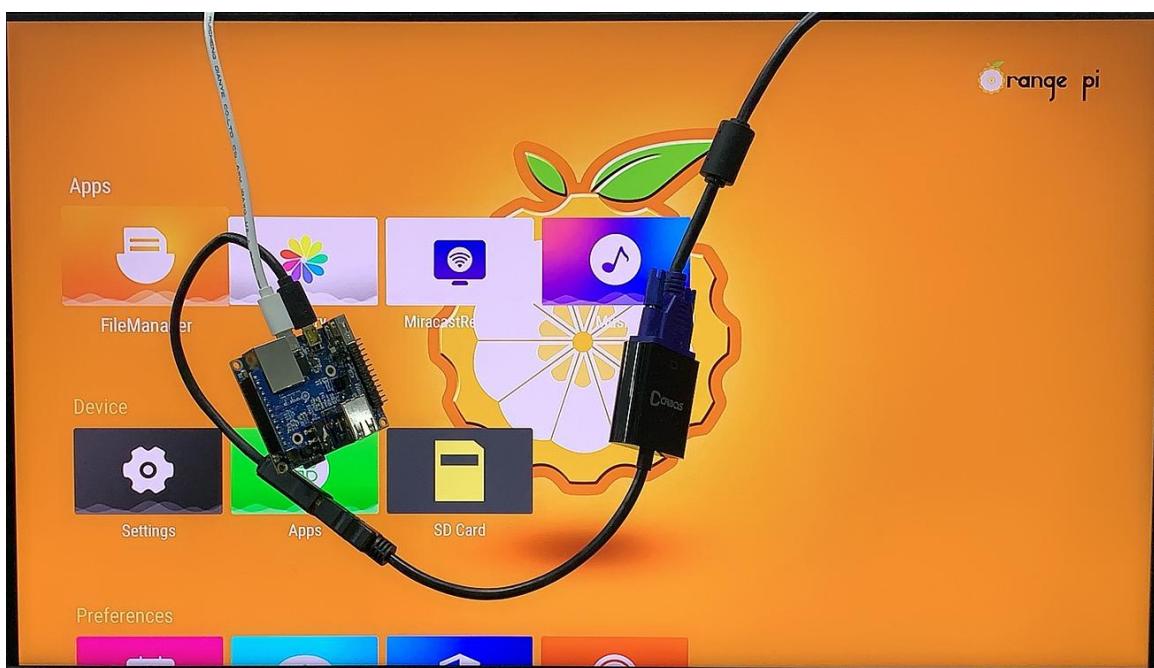


b. 一根 VGA 线和一根 Micro HDMI 公转 HDMI 母转接线



c. 一个支持 VGA 接口的显示器或者电视

4) 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 Micro 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

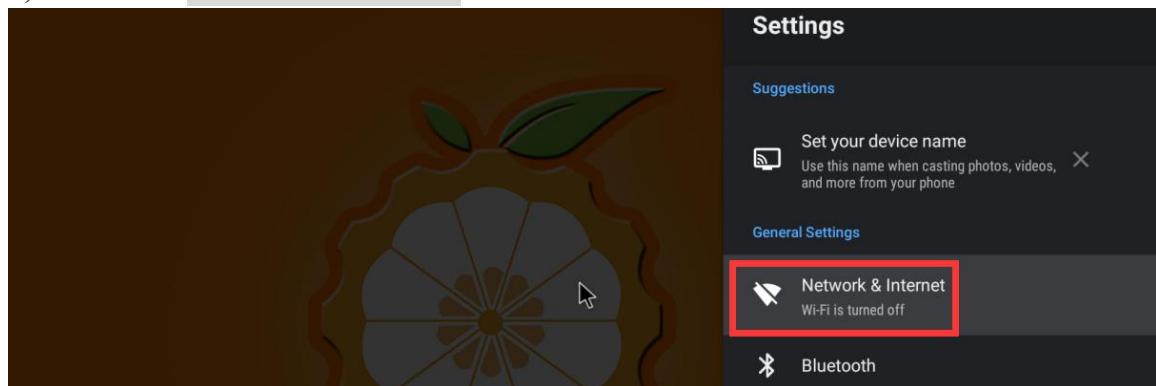


4. 9. WI-FI 的连接方法

1) 首先选择 **Settings**



2) 然后选择 **Network & Internet**



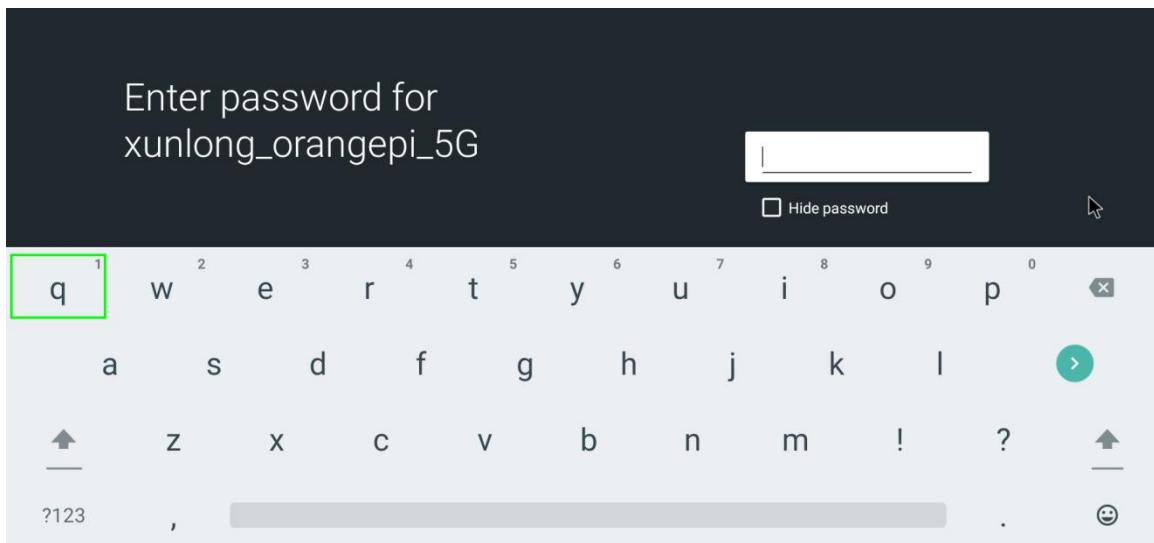
3) 然后打开 WI-FI



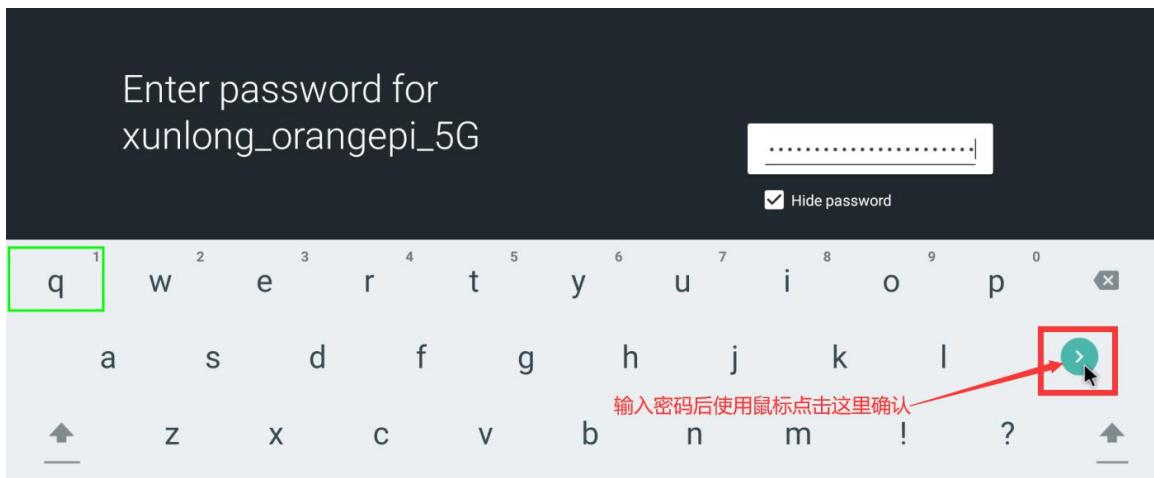
4) 打开 WI-FI 后在 **Available networks** 下面就可以看到搜索到的信号



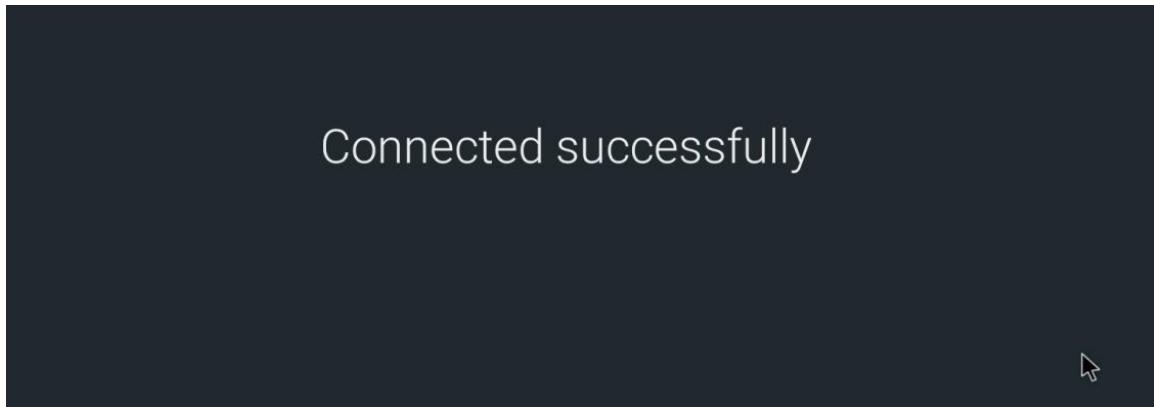
5) 选择想连接的 WI-FI 后会弹出下图所示的密码输入界面



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



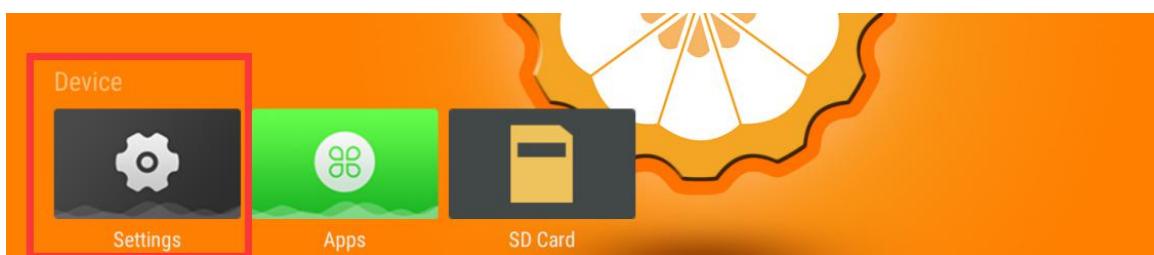
7) WI-FI 连接成功后的显示如下图所示



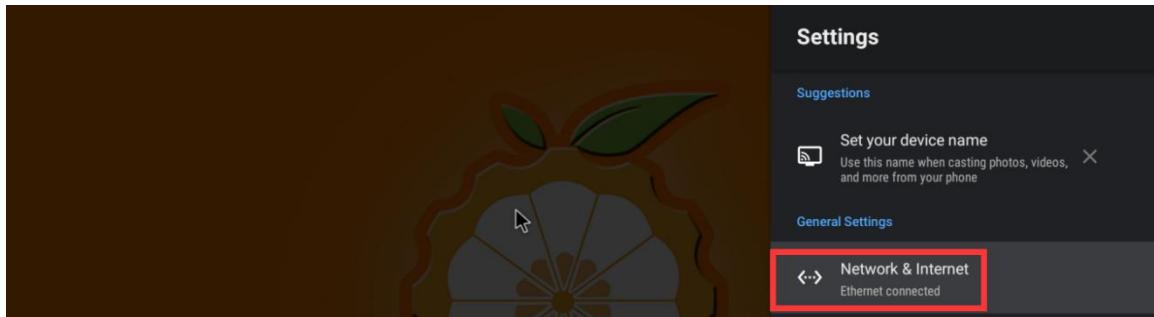
4. 10. WI-FI hotspot 的使用方法

1) 首先请确保以太网口已连接网线，并且能正常上网

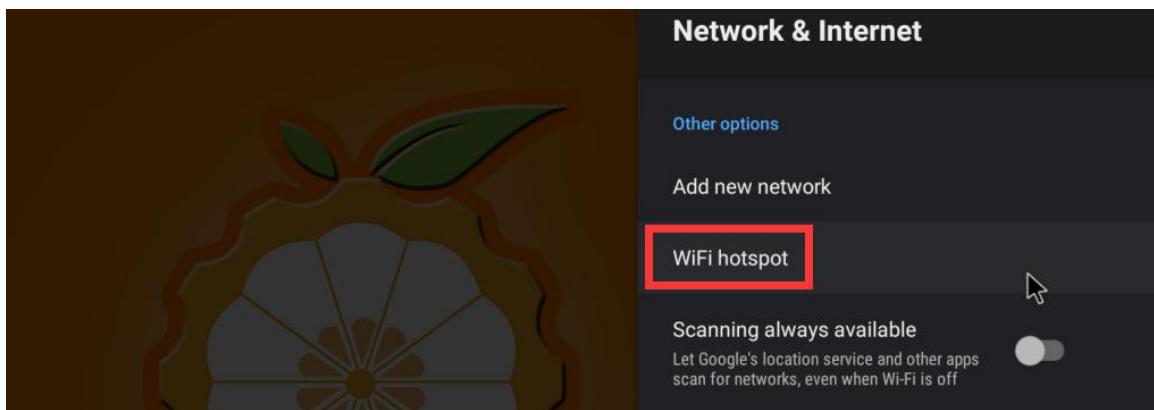
2) 然后选择 **Settings**



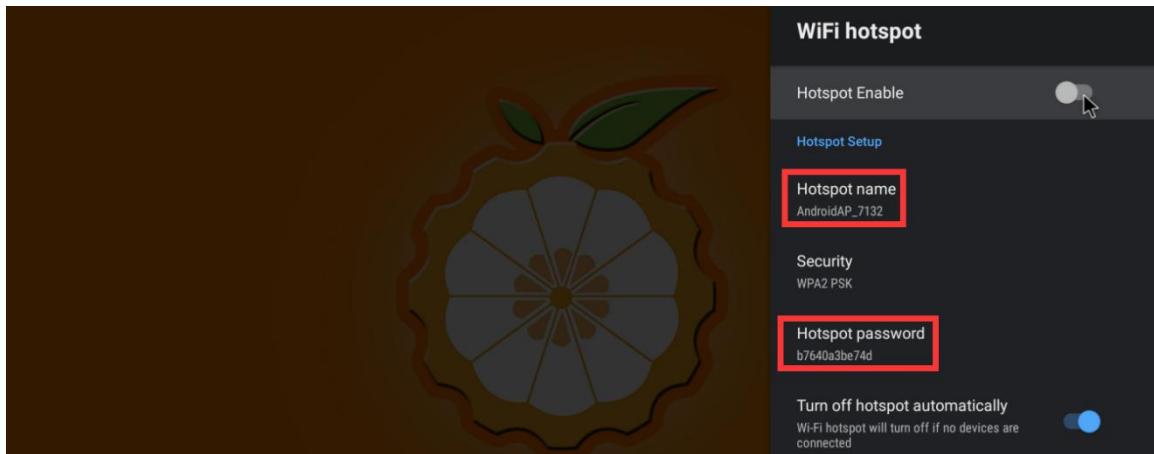
3) 然后选择 **Network & Internet**



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)



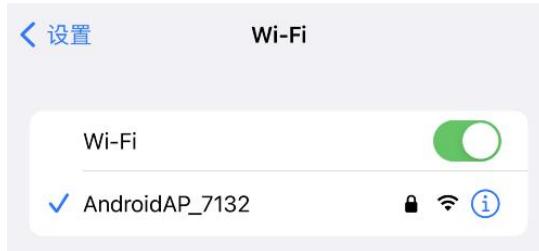
6) 此时可以拿出你的手机, 如果一切正常, 在手机搜索到的 WI-FI 列表中就能找到上图 **Hotspot name** 下面显示的同名 (这里为 **AndroidAP_7132**) 的 WIFI 热点了。然后可以点击 **AndroidAP_7132** 连接热点, 密码在上图的 **Hotspot password** 下面可



以看到

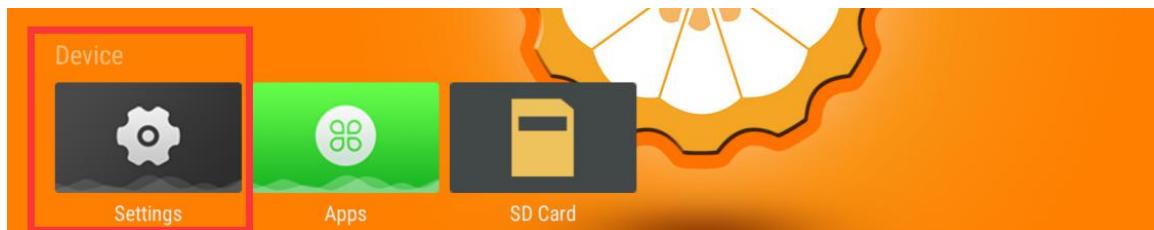


7) 连接成功后显示如下图所示（不同手机界面会有区别，具体界面以你手机显示的为准）。此时就可以在手机上打开一个网页看下能否上网了，如果能正常打开网页，说明开发板的 **WI-FI Hotspot** 能正常使用



4. 11. 蓝牙的连接方法

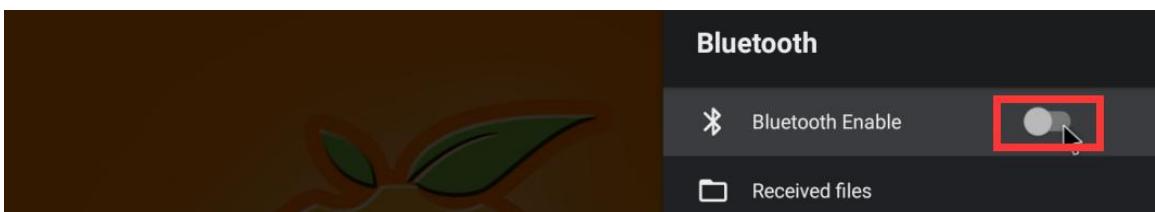
1) 首先选择 **Settings**



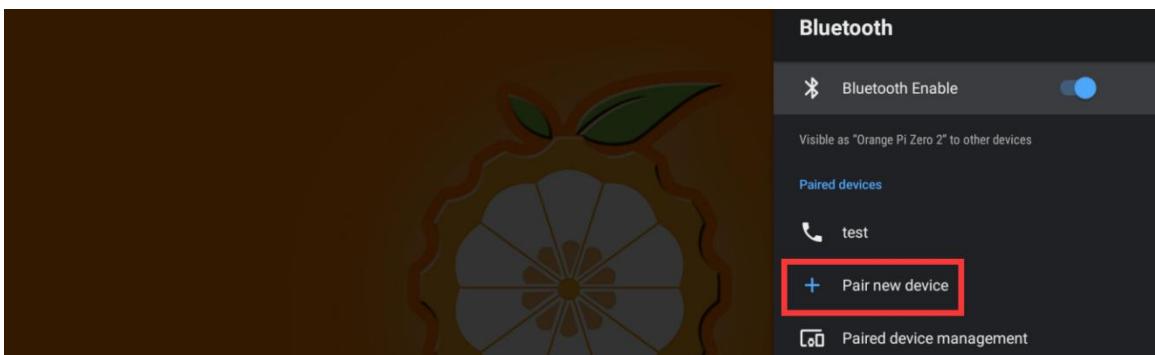
2) 然后选择 **Bluetooth**



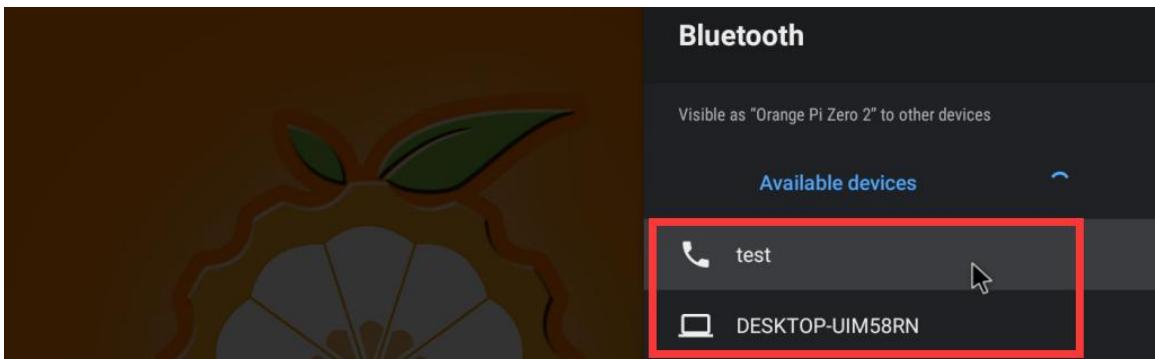
3) 然后打开 **Bluetooth Enable**



4) 然后点击 **Pair new device** 开始扫描周围的蓝牙设备

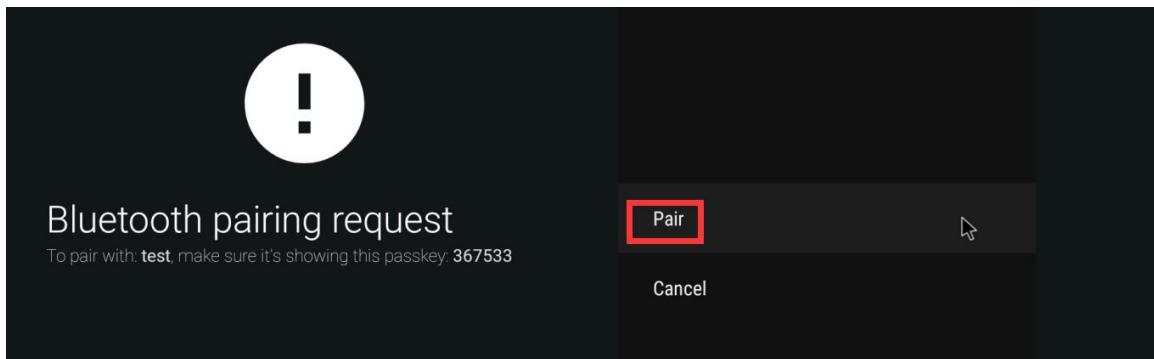


5) 搜索到的蓝牙设备会在 **Available devices** 下面显示出来





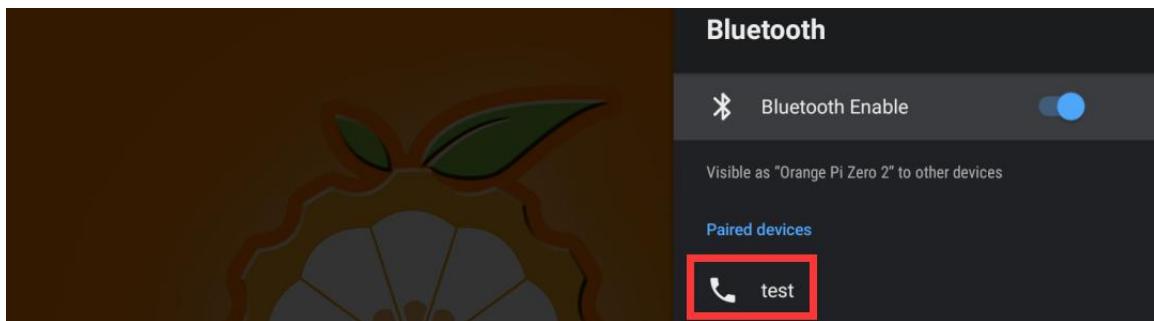
6) 然后点击想要连接的蓝牙设备就可以开始配对了，当弹出下面的界面时，请使用鼠标选择 **Pair** 选项



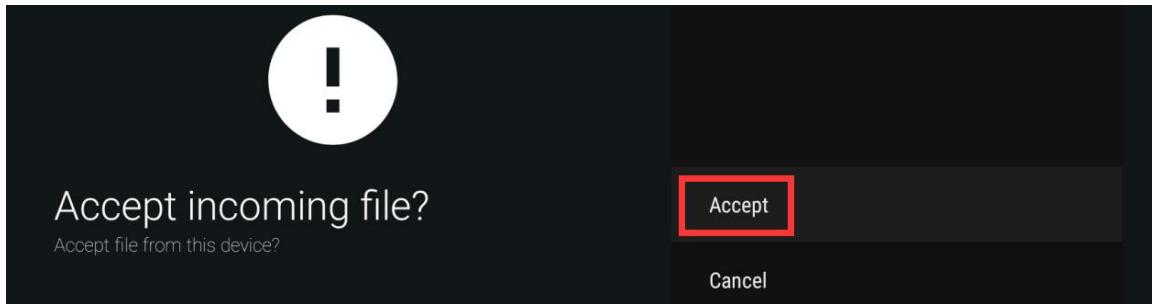
7) 这里测试的是开发板和**安卓手机**蓝牙的配置过程，此时在手机上会弹出下面的确认界面，在手机上也点击配对按钮后就会开始配对过程



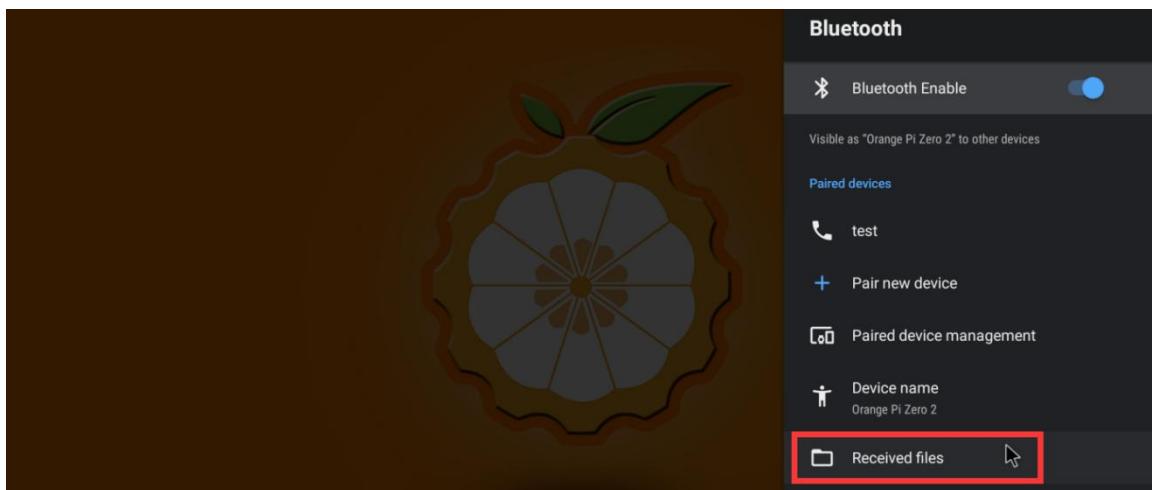
8) 配对完成后，再打开 **Paired devices** 下面就可以看到已配对的蓝牙设备



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



4. 12. Android 视频客户端软件安装说明

1) 如果需要安装腾讯视频用来看电视剧和电影，请安装腾讯视频电视端的 TV 版，安装移动端的 Android 版本是无法正常使用的，腾讯视频电视端的 TV 版下载地址如下所示

<http://v.qq.com/download.html>

2) 如果需要安装优酷用来看电视剧和电影，请安装优酷提供的电视端的酷喵 TV 版，安装移动端的 Android 版本是无法正常使用的，酷喵 TV 版的下载地址如下所示

https://youku.com/product/index?spm=a2ha1.14919748_WEBHOME_GRAY.uerCenter.5!5~5~5~A

3) 如果需要安装爱奇艺用来看电视剧和电影，请使用爱奇艺的奇异果 TV 版本，下载地址如下所示



<http://app.iqiyi.com/tv/player/>

4. 13. USB 摄像头使用方法

- 1) 先在开发板的 USB 接口中插入 USB 摄像头，然后确认下 USB 摄像头相关的内核模块已正常加载

```
console:/ # lsmod
Module           Size  Used by
sprdw1_ng        405504  0
sprdbt_tty       36864   2
uwe5622_bsp_sdio 274432  2 sprdw1_ng,sprdbt_tty
uvcvideo          102400  0
videobuf2_v4l2    28672   1 uvcvideo
videobuf2_vmalloc  16384   1 uvcvideo
videobuf2_memops   16384   1 videobuf2_vmalloc
videobuf2_core     49152   2 uvcvideo,videobuf2_v4l2
mali_kbase        532480   7
```

- 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
console:/ # ls -l /sys/class/video4linux/
total 0
lrwxrwxrwx 1 root root 0 2020-11-02 20:46:01.187678078 +0800 video0 -> ../../devices/platform/soc/5200000.ehci1-controller/usb1/1-1/1-1:1.0/video4linux/video0
console:/ #
```

- 3) 然后确保 Ubuntu PC 和开发板的 adb 连接正常

- 4) 在 Orange Pi Zero 2 资料下载页面的官方工具中下载 USB 摄像头测试 APP

The screenshot shows a web page with a header "官方工具" and a "保存到网盘" button. Below the header, there is a timestamp "2020-11-03 14:09" and a note "失效时间: 永久有效". A breadcrumb navigation bar shows "返回上一级 | 全部文件 > 官方工具 > Android测试APP". The main content is a table listing four APK files:

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

4. 14. Android 系统 ROOT 说明

Orange Pi 发布的 Android 10.0 系统已经 ROOT，可以使用下面的方法来测试

1) 在 Orange Pi Zero 2 资料下载页面的官方工具中下载 rootcheck.apk

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

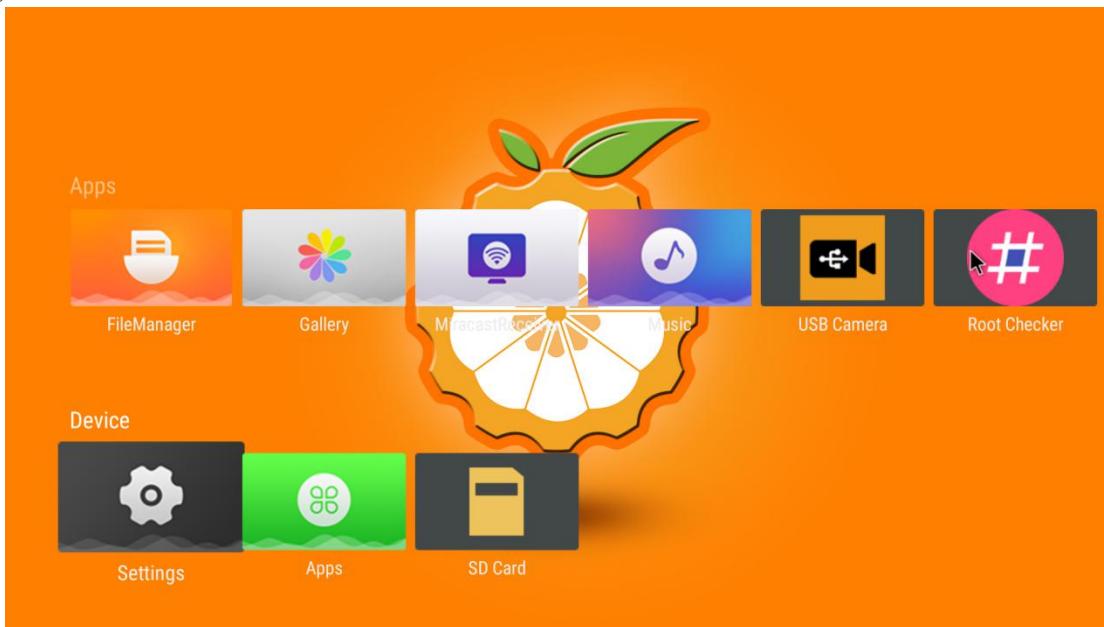
2) 然后确保 Ubuntu PC 和开发板的 adb 连接正常

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

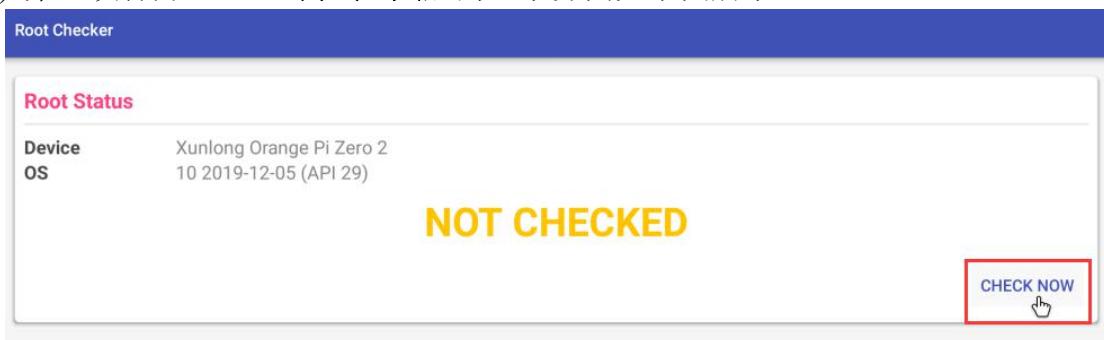
```
test@test:~$ adb install rootcheck.apk
```



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“立刻检查”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



Root Checker

Root Status

Device	Xunlong Orange Pi Zero 2
OS	10 2019-12-05 (API 29)

ROOTED

CHECK NOW

Root Details

Root access	Yes
SELinux	Permissive



5. Linux SDK 使用说明

5.1. 编译系统需求

1) Linux SDK, 即 **orangeipi-build**, 只支持在安装有 **Ubuntu18.04** 的电脑上运行, 所以下载 **orangeipi-build** 前, 请首先确保自己电脑已安装的 Ubuntu 版本是 Ubuntu18.04。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 **18.04**, 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 18.04.5 LTS
Release:        18.04
Codename:       bionic
test@test:~$
```

2) 如果电脑安装的是 Windows 系统, 没有安装有 Ubuntu18.04 的电脑, 可以考虑使用 **VirtualBox** 或者 **VMware** 来在 Windows 系统中安装一个 Ubuntu18.04 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

5.2. 获取 linux sdk 的源码

5.2.1. 从 **github** 下载 **orangeipi-build**

1) linux sdk 其实指的就是 orangeipi-build 这套代码, orangeipi-build 是基于 armbian build 编译系统修改而来的, 使用 orangeipi-build 可以编译出多个版本的 linux 镜像。首先下载 orangeipi-build 的代码, 目前 H616 系列开发板已经支持 legacy 分支和 current 分支

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



```
test@test:~$ git clone https://github.com/orangepi-xunlong/orangepi-build.git
```

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。

2) `legacy` 分支一般使用全志提供的 `BSP` 版本的 `u-boot` 和 `linux` 内核代码，`current` 分支一般使用接近 `linux` 主线版本的 `u-boot` 和内核代码，`H616` 系列开发板当前使用的 `u-boot` 和 `linux` 内核如下所示

分支	u-boot 版本	linux 内核版本
legacy	u-boot 2018.05	linux4.9
current	u-boot 2021.07	linux5.13

3) `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
```

如果你是从 `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。

5.2.2. 下载交叉编译工具链

1) 第一次运行 `orangepi-build` 中的 `build.sh` 脚本时候会自动下载交叉编译工具链，



并将其存放在 orangepi-build 下的 **toolchains** 文件夹中，此后每次运行 orangepi-build 中的 **build.sh** 脚本时，也都会检查 **toolchains** 中的交叉编译工具链是否都存在，如果不存在或者有更新则会重新开始下载，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 ]
#e39eec 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 [9.66MiB/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_aarch64-linux-gnu.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
```

2) 交叉编译工具链在中国境内的默认镜像网址为清华大学的开源软件镜像站，如果需要单独下载交叉编译工具链的压缩包，可以打开下面的网址，然后选择需要的版本下载即可

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

3) **toolchains** 下载完后会包含多个版本的交叉编译工具链

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

4) 编译 H616 Linux 内核源码使用的交叉编译工具链为

a. linux4.9



```
gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu
```

- b. linux5.13

```
gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu
```

5) 编译 H616 u-boot 源码使用的交叉编译工具链为

- a. u-boot 2018.05

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

- b. u-boot 2021.07

```
gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu
```

5. 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

- a) linux4.9

```
https://github.com/orangepi-xunlong/linux-orangepi/tree/orange-pi-4.9-sun50iw9
```

- b) linux5.13

```
https://github.com/orangepi-xunlong/linux-orangepi/tree/orange-pi-5.13-sunxi64
```

- b. u-boot 源码存放的 git 仓库如下

- a) u-boot 2018.05

```
https://github.com/orangepi-xunlong/u-boot-orangepi/tree/v2018.05-sun50iw9
```

- b) u-boot 2021.07

```
https://github.com/orangepi-xunlong/u-boot-orangepi/tree/v2021.07-sunxi
```

如果你对 orangepi-build 不熟悉，不清楚其编译 linux 内核和 u-boot 的详细过程，请不要单独下载使用上面的 linux 内核和 u-boot 源码进行编译操作，因为 orangepi-build 的编译脚本和配置文件会对 u-boot 和 linux 进行一些调整和优化，如果不使用 orangepi-build 来编译 u-boot 和 linux，可能会遇到编译失败或者无法启动的问题。

2) 第一次运行 orangepi-build 中的 **build.sh** 脚本时会自动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，里面名为 **orange-pi-4.9-sun50iw9** 的文件夹存放的就是 H616 系列开发板 legacy 分支的内核源码，里面名为 **orange-pi-5.13-sunxi64** 的文件夹存放的就是 H616 开发板 current 分支的内核源码（如果只编译了 legacy 分支的 linux 镜像，那么则只能看到 legacy 分支的内核源码；如果只编译了 current 分支的 linux 镜像那么则只能看到 current 分支的内核源码），内核源码的文件夹的名字请不要手动修改，如果修改了，编译系统运行时会重新下载内核源码
- 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，里面名为 **v2018.05-sun50iw9** 的文件夹存放的就是 H616 系列开发板 legacy 分支的 u-boot 源码，里面名为 **v2021.07-sunxi** 的文件夹存放的就是 H616 开发板 current 分支的 u-boot 源码（如果只编译了 legacy 分支的 linux 镜像，那么则只能看到 legacy 分支的 u-boot 源码；如果只编译了 current 分支的 linux 镜像，那么则只能看到 current 分支的 u-boot 源码），u-boot 源码的文件夹的名字请不要手动修改，如果修改了，编译系统运行时会重新下载 u-boot 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
```

5. 2. 4. 从百度云盘下载 orangepi-build

1) 如果从 github 下载 orangepi-build 源码的速度很慢，或者提示网络无法连接，还可以从百度云盘下载 orangepi-build 的压缩包，百度云盘的下载链接如下所示

链接: <https://pan.baidu.com/s/1vWQmCmSYdH7iCDFyKpJtVw>

提取码: zero



A screenshot of a Baidu Cloud storage interface. At the top, there's a blue button labeled "保存到网盘". Below it, a message says "① 2020-11-05 12:06 失效时间: 永久有效". A link "返回上一级 | 全部文件 > orangepi-build" is shown. The main area shows a file list with three items:

文件名	大小	修改日期
u-boot	-	2020-11-05 13:54
orangepi-build	-	2020-12-09 20:37
Linux镜像使用的rootfs压缩包	-	2020-11-05 12:08

2) 百度云盘的 orangepi-build 文件夹下有两个文件

- orangepi-build.tar.gz** 为 orangepi-build 源码的压缩包
- orangepi-build.tar.gz.md5sum** 为 orangepi-build 源码的压缩包的 MD5 校验和文件
- 下载完后, 请首先检查下 orangepi-build.tar.gz 压缩包的 MD5 校验和是否正确, 这样可以防止下载的压缩包有问题, 如果不正确, 请重新下载, 检查校验和是否正确的命令为

```
test@test:~$ md5sum -c orangepi-build.tar.gz.md5sum
orangepi-build.tar.gz: 成功
```

A screenshot of a Baidu Cloud storage interface. At the top, there's a blue button labeled "保存到网盘". Below it, a message says "① 2020-11-05 12:06 失效时间: 永久有效". A link "返回上一级 | 全部文件 > orangepi-build > orangepi-build" is shown. The main area shows a file list with two items:

文件名	大小	修改日期
orangepi-build.tar.gz.md5sum	568	2020-12-09 20:37
orangepi-build.tar.gz	3,456	2020-12-09 20:37

3) 然后就可以使用 **tar -zxf** 命令解压 orangepi-build.tar.gz

```
test@test:~$ tar -zxf orangepi-build.tar.gz
test@test:~$ cd orangepi-build/
test@test:~/orangepi-build$ ls
build.sh    external    kernel    LICENSE    README.md    scripts    toolchains
u-boot    userpatches
```

4) 使用 orangepi-build 编译系统前, 请先将 orangepi-build 和 github 服务器进行同步, 确保代码为最新的状态

```
test@test:~/orangepi-build$ git pull
```

如果 orangepi-build 和 github 服务器同步过程有问题, 可以尝试下面的方法

```
test@test:~/orangepi-build$ sudo rm -rf external scripts
```



```
test@test:~/orangepi-build$ git checkout .
test@test:~/orangepi-build$ git pull
```

百度云盘上的 **orangepi-build.tar.gz** 压缩包除了包含 **orangepi-build** 编译系统的代码外，还缓存了交叉编译工具链，**u-boot** 和 **linux** 内核的源码。所以在编译镜像的过程中就不会去 **github** 服务器上从头开始下载 **u-boot** 和 **linux** 内核的源码以及交叉编译工具链，这样可以节省大量的时间。**orangepi-build** 编译系统开始运行时，默认会自动从 **github** 同步 **u-boot** 和 **linux** 内核的源码（请注意，这里不会同步 **orangepi-build** 本身仓库的代码），以确保代码为最新状态，所以无需手动同步 **u-boot** 和 **linux** 内核的源码

5) 百度云盘上的 **orangepi-build.tar.gz** 解压后的 **kernel** 文件夹中会包含多个版本的内核源码，请不要修改这些内核源码文件夹的名字，否则会导致编译系统找不到内核源码进而重新从 **github** 下载内核源码。**kernel** 文件夹下的多个内核源码并非都能被 H616 系列开发板使用，H616 系列开发板使用到的内核源码有：

orange-pi-4.9-sun50iw6	H6 系列开发板 legacy 分支使用可以忽略
orange-pi-5.4	H6 系列开发板 current 分支使用可以忽略
orange-pi-3.4-sun8i	H2/H3 系列开发板 legacy 分支使用可以忽略
orange-pi-5.4	H2/H3/H5/H6 系列开发板 current 分支使用可以忽略
orange-pi-4.9-sun50iw9	H616 系列开发板 legacy 分支使用

6) 百度云盘上的 **orangepi-build.tar.gz** 解压后的 **u-boot** 文件夹中会包含多个版本的 **u-boot** 源码，请不要修改这些 **u-boot** 源码文件夹的名字，否则会导致编译系统找不到 **u-boot** 源码进而重新从 **github** 下载 **u-boot** 源码。**u-boot** 文件夹下的多个版本的 **u-boot** 源码并非都能被 H616 系列开发板使用，H616 系列开发板使用到的 **u-boot** 源码有：

v2014.07-sun50iw6-linux4.9	H6 系列开发板 legacy 分支使用可以忽略
v2020.04	H6 系列开发板 current 分支使用可以忽略
v2018.05-sun50iw9	H616 系列开发板 legacy 分支使用

5.3. 编译 **u-boot**

首先需要注意，**legacy** 分支的 **u-boot 2018.05** 的源码中没有 **boot0** 的源码，这部分源码全志是不开放的，只提供了 **boot0** 的 **bin** 文件。**current** 分支的 **u-boot 2021.07**

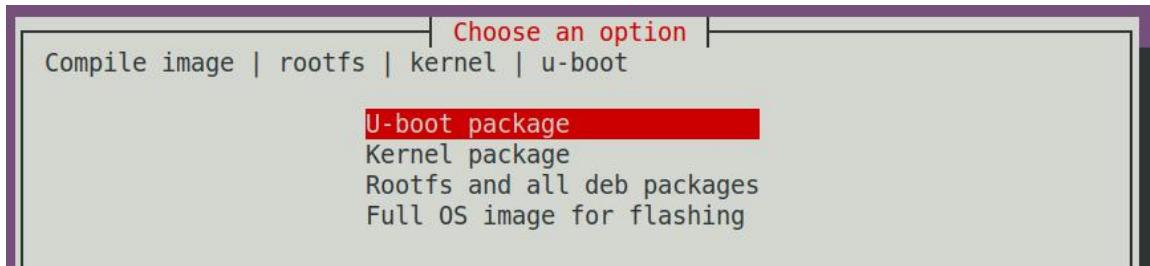


没有这个问题，所有代码都可以获取到

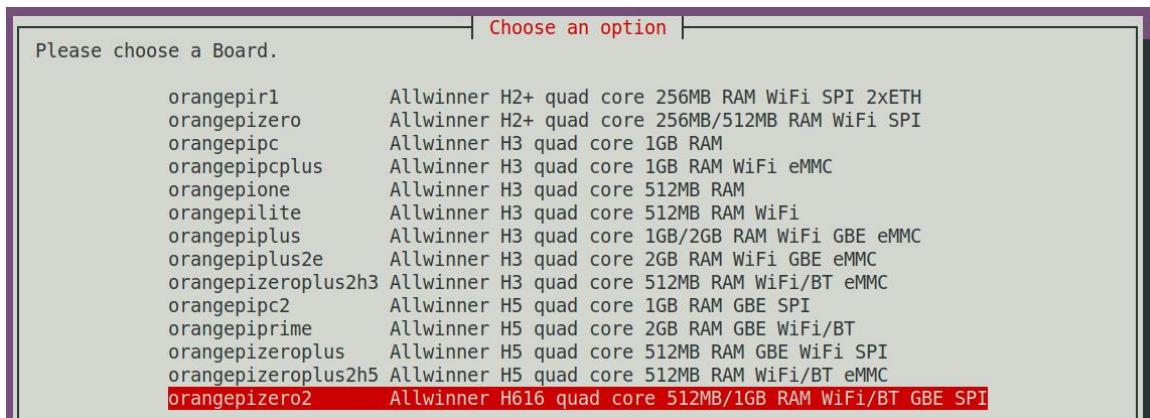
1) 运行 build.sh 脚本，记得加 sudo 权限

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

2) 选择 **U-boot package**，然后回车



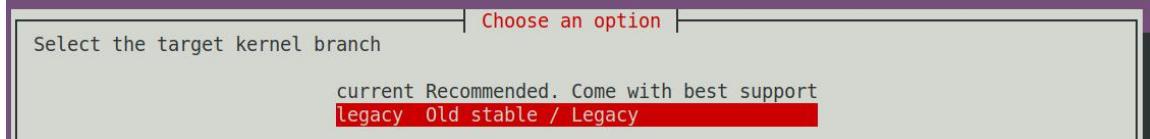
3) 接着选择开发板的型号



4) 然后选择分支

a. current 会编译 u-boot v2021.07

b. legacy 会编译 u-boot v2018.05



5) 然后就会开始编译 u-boot，编译时提示的部分信息说明如下(以 legacy 分支为例)

a. u-boot 源码的版本

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

b. 交叉编译工具链的版本



[o.k.] Compiler version [**arm-linux-gnueabi-gcc 7.4.1**]

c. 编译生成的 u-boot deb 包的路径

[o.k.] Target directory [**orangepi-build/output/debs/u-boot**]

d. 编译生成的 u-boot deb 包的包名

[o.k.] File name [**linux-u-boot-legacy-orangepizero2_2.1.8_arm64.deb**]

e. 编译使用的时间

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

f. 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=orangepi2**]

BRANCH=legacy BUILD_OPT=u-boot KERNEL_CONFIGURE=no]

6) 查看编译生成的 u-boot deb 包

```
test@test:~/orangepi-build$ ls output/debs/u-boot/  
linux-u-boot-legacy-orangepizero2_2.1.8_arm64.deb
```

7) 生成的 u-boot 的 deb 包包含的文件如下所示

a. 使用下面的命令可以解压 deb 包

```
test@test:~/orangepi-build$ cd output/debs/u-boot  
test@test:~/orangepi_build/output/debs/u-boot$ $ dpkg -x \  
linux-u-boot-legacy-orangepizero2_2.1.8_arm64.deb . (注意命令最后有个 ".")  
test@test:~/orangepi_build/output/debs/u-boot$ ls  
linux-u-boot-legacy-orangepizero2_2.1.8_armhf.deb usr
```

b. 解压后的文件如下所示

```
test@test:~/orangepi-build/output/debs/u-boot$ tree usr/  
usr/  
└── lib  
    ├── linux-u-boot-legacy-orangepizero2_2.1.8_arm64  
    |   ├── boot0_sdcard.fex      //boot0 的二进制文件  
    |   └── boot_package.fex     //u-boot 的二进制文件  
    └── u-boot  
        ├── LICENSE  
        ├── orangepi_zero2_defconfig      //编译 u-boot 源码使用的配置文件  
        ├── orangepizero2-u-boot.dts      //u-boot 源码使用的 dts 文件  
        └── platform_install.sh          //u-boot 的烧录脚本
```



3 directories, 6 files

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. 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-legacy-orangepizero2_2.1.8_arm64.deb root@192.168.1.xxx:/root
```

b. 然后登录到开发板，卸载已安装的 u-boot 的 deb 包

```
root@orangepi:~# apt purge -y linux-u-boot-orangepizero2-legacy
```

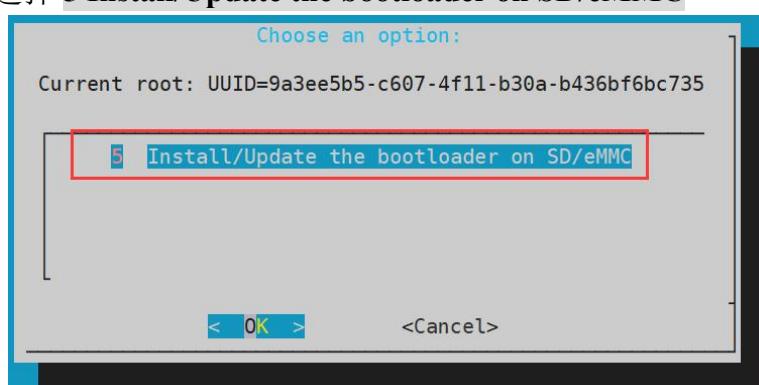
c. 再安装刚才上传的新的 u-boot 的 deb 包

```
root@orangepi:~# dpkg -i linux-u-boot-legacy-orangepizero2_2.1.8_arm64.deb
```

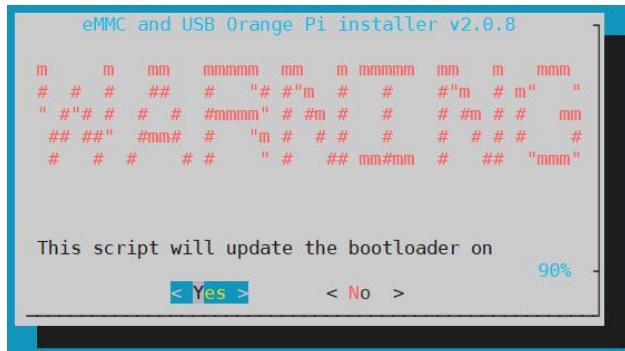
d. 然后运行 nand-sata-install 脚本

```
root@orangepi:~# nand-sata-install
```

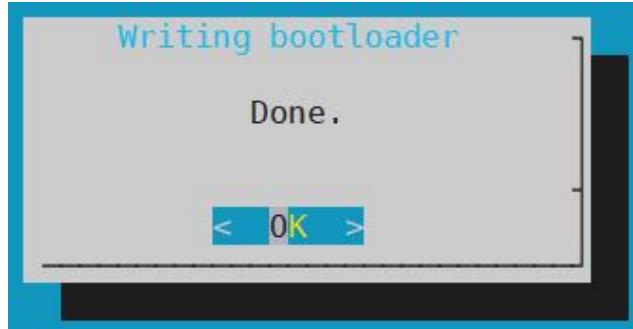
e. 然后选择 **5 Install/Update the bootloader on SD/eMMC**



f. 按下回车键后首先会弹出一个 Warring



g. 再按下回车键就会开始更新 u-boot，更新完后会显示下面的信息



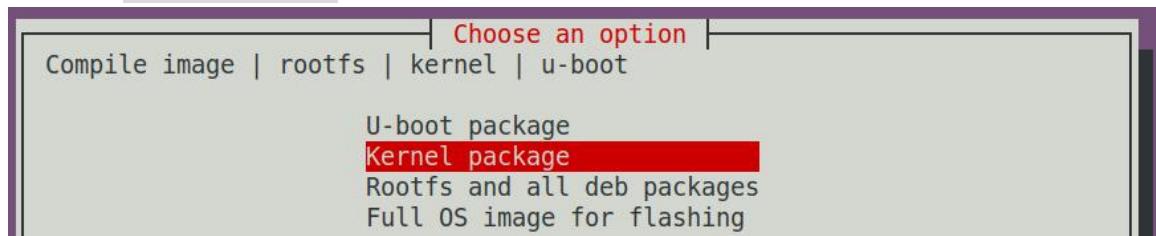
h. 然后就可以重启开发板来测试 u-boot 的修改是否生效了

5.4. 编译 linux 内核

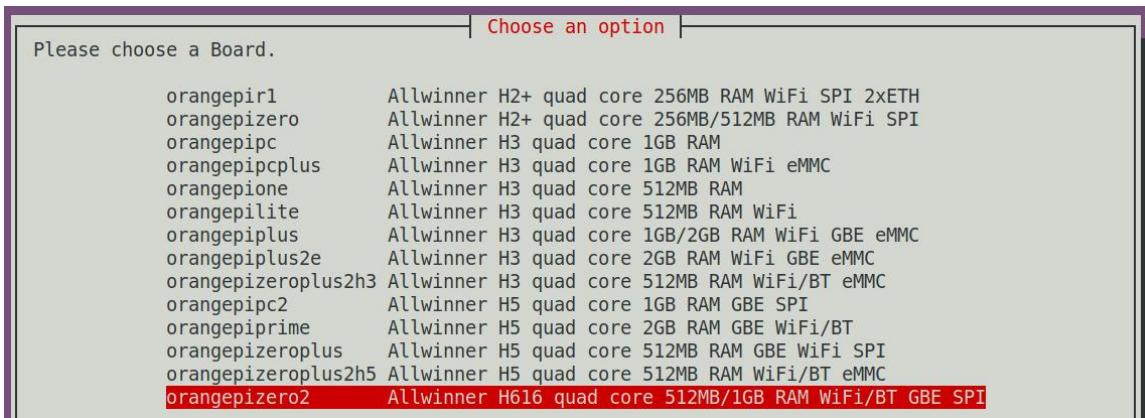
1) 运行 build.sh 脚本，记得加 sudo 权限

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

2) 选择 **Kernel package**，然后回车

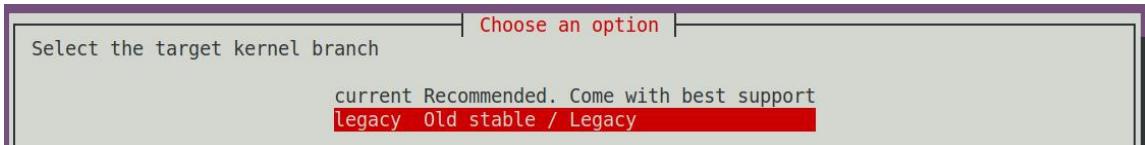


3) 接着选择开发板的型号

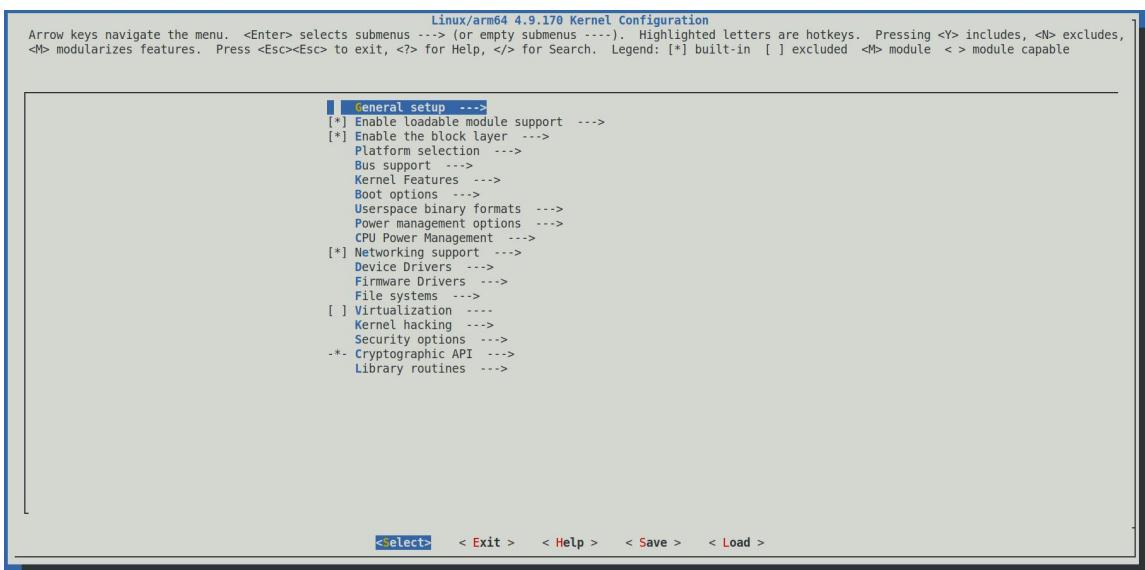


4) 然后选择分支

- current 会编译 linux 5.13
- legacy 会编译 linux4.9



5) 然后会弹出通过 **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，如果不需要修改内核配置，直接退出即可，退出后会开始编译内核源码



- 如果不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
```

- 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
```

6) 编译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 legacy kernel [4.9.170]

b. 使用的交叉编译工具链的版本

[o.k.] Compiler version [aarch64-none-linux-gnu-gcc 9.2.1]

c. 内核默认使用的配置文件以及它存放的路径

[o.k.] Using kernel config file [config/kernel/linux-sun50iw9-legacy.config]

d. 如果 `KERNEL_CONFIGURE=yes`，内核最终使用的配置文件.config 会复制到 `output/config` 中，如果没有对内核配置进行修改，最终的配置文件和默认的配置文件是一致的

[o.k.] Exporting new kernel config [output/config/linux-sun50iw9-legacy.config]

e. 编译生成的内核相关的 deb 包的路径

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

f. 编译生成的内核镜像 deb 包的包名

[o.k.] File name [linux-image-legacy-sun50iw9_2.1.8_arm64.deb]

g. 编译使用的时间

[o.k.] Runtime [5 min]

h. 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=orangepirzero2

BRANCH=legacy BUILD_OPT=kernel KERNEL_CONFIGURE=yes]



7) 查看编译生成的内核相关的 deb 包

- a. **linux-dtb-legacy-sun50iw9_2.1.8_arm64.deb** 暂未使用，先不用关心
- b. **linux-headers-legacy-sun50iw9_2.1.8_arm64.deb** 包含内核头文件
- c. **linux-image-legacy-sun50iw9_2.1.8_arm64.deb** 包含内核镜像和内核模块

```
test@test:~/orangeipi-build$ ls output/debs/linux-*
output/debs/linux-dtb-legacy-sun50iw9_2.1.8_arm64.deb
output/debs/linux-headers-legacy-sun50iw9_2.1.8_arm64.deb
output/debs/linux-image-legacy-sun50iw9_2.1.8_arm64.deb
```

8) 生成的 linux-image 的 deb 包包含的文件如下所示

- a. 使用下面的命令可以解压 deb 包

```
test@test:~/orangeipi-build$ cd output/debs
test@test:~/orangeipi_build/output/debs$ mkdir test
test@test:~/orangeipi_build/output/debs$ cp \
linux-image-legacy-sun50iw9_2.1.8_arm64.deb test/
test@test:~/orangeipi_build/output/debs$ cd test
test@test:~/orangeipi_build/output/debs/test$ dpkg -x \
linux-image-legacy-sun50iw9_2.1.8_arm64.deb .
test@test:~/orangeipi_build/output/debs/test$ ls
boot  etc  lib  linux-image-legacy-sun50iw9_2.1.8_arm64.deb  usr
```

- b. 解压后的文件如下所示

```
test@test:~/orangeipi_build/output/debs/test$ tree -L 2
.
├── boot
│   ├── config-4.9.170-sun50iw9      //编译内核源码使用的配置文件
│   ├── System.map-4.9.170-sun50iw9
│   └── vmlinuz-4.9.170-sun50iw9     //编译生成的内核镜像文件
└── etc
    └── kernel
└── lib
    └── modules                      //编译生成的内核模块
└── linux-image-legacy-sun50iw9_2.1.8_arm64.deb
└── usr
    └── lib
```



```
└─ share  
8 directories, 4 files
```

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

a. 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-legacy-sun50iw9_2.1.8_arm64.deb root@192.168.1.207:/root
```

b. 然后登录到开发板，卸载已安装的 linux 内核的 deb 包

```
root@orangepi:~# apt purge -y linux-image-legacy-sun50iw9
```

c. 再安装刚才上传的新的 linux 内核的 deb 包

```
root@orangepi:~# dpkg -i linux-image-legacy-sun50iw9_2.1.8_arm64.deb
```

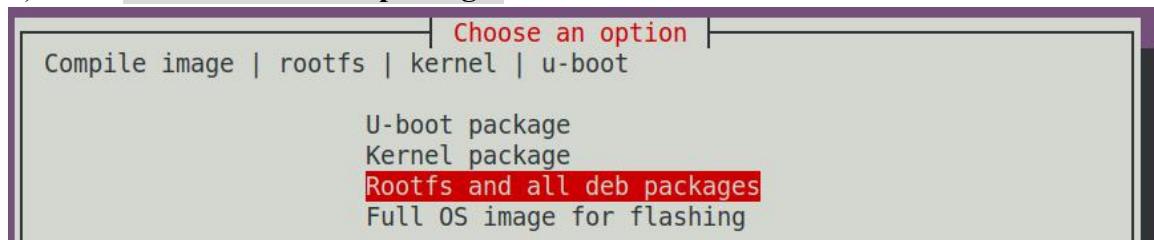
d. 然后重启开发板，再查看内核相关的修改是否已生效

5. 5. 编译 rootfs

1) 运行 build.sh 脚本，记得加 sudo 权限

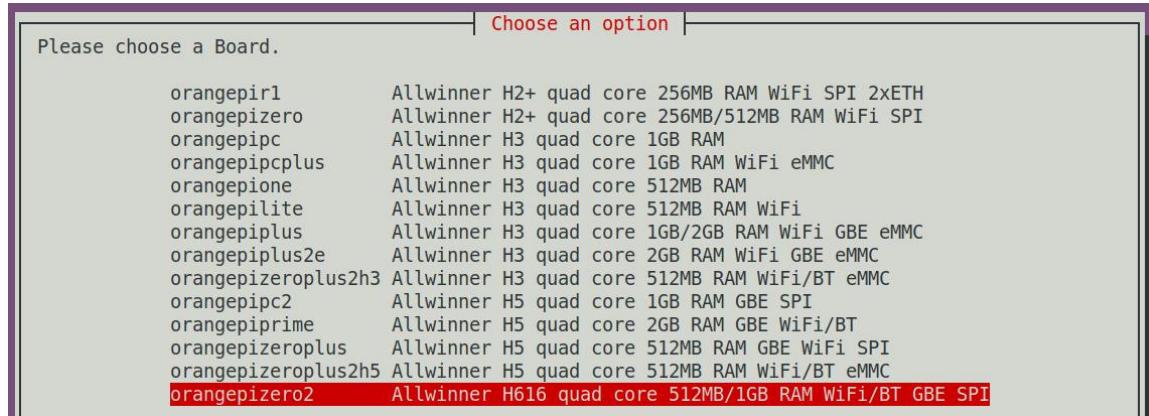
```
test@test:~/orangepi-build$ sudo ./build.sh
```

2) 选择 **Rootfs and all deb packages**，然后回车

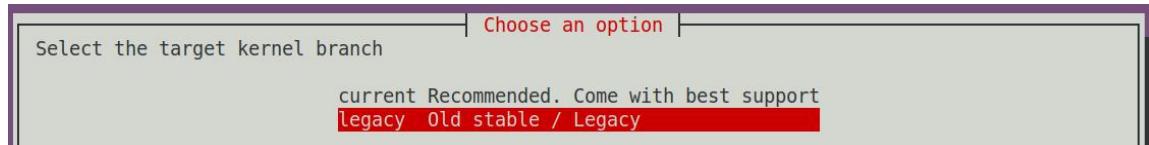




3) 接着选择开发板的型号

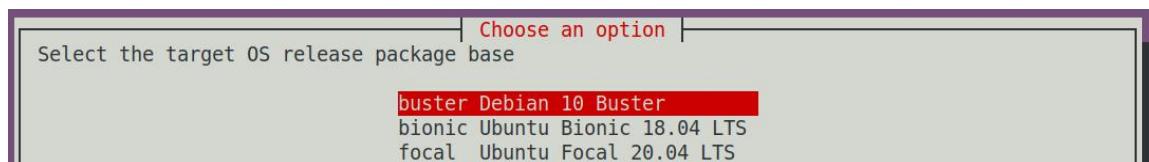


4) 然后选择分支类型



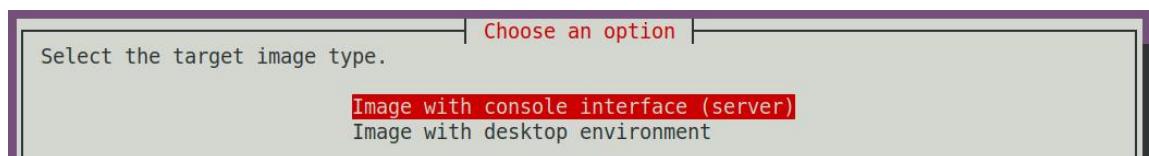
5) 然后选择 rootfs 的类型

- buster** 表示 Debian 10
- bionic** 表示 Ubuntu 18.04
- focal** 表示 Ubuntu 20.04

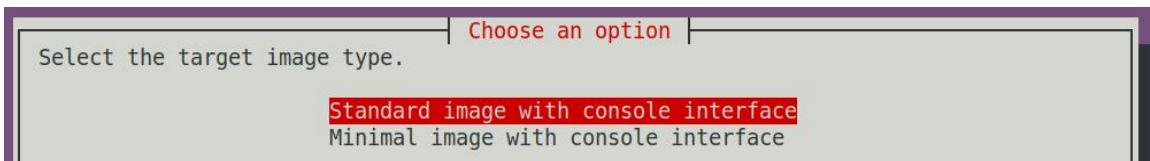


6) 然后选择镜像的类型

- 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



8) 选择镜像的类型后就会开始编译 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 **bionic**]

b. 编译生成的 rootfs 压缩包的存放路径

[o.k.] Target directory [**external/cache/rootfs**]

c. 编译生成的 rootfs 压缩包的名字

[o.k.] File name [**bionic-cli-arm64.153618961f14c28107ca023429aa0eb9.tar.lz4**]

d. 编译使用的时间

[o.k.] Runtime [**13 min**]

e. 重复编译 rootfs 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 rootfs

```
[ o.k. ] Repeat Build Options [ sudo ./build.sh BOARD=orangezerop2  
BRANCH=legacy BUILD_OPT=rootfs RELEASE=bionic BUILD_MINIMAL=no  
BUILD_DESKTOP=no KERNEL_CONFIGURE=yes ]
```

9) 查看编译生成的 rootfs 压缩包

a. **bionic-cli-arm64.153618961f14c28107ca023429aa0eb9.tar.lz4** 是 rootfs 的压缩包，名字各字段的含义为

- a) **bionic** 表示 rootfs 的 linux 发行版的类型
- b) **cli** 表示 rootfs 为服务器版的类型，如果为 **desktop** 则表示桌面版类型
- c) **arm64** 表示 rootfs 的架构类型
- d) **153618961f14c28107ca023429aa0eb9** 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. **bionic-cli-arm64.153618961f14c28107ca023429aa0eb9.tar.lz4.list** 列出了 rootfs 安装的所有软件包的包名

```
test@test:~/orangepi-build$ ls external/cache/rootfs/  
bionic-cli-arm64.153618961f14c28107ca023429aa0eb9.tar.lz4  
bionic-cli-arm64.153618961f14c28107ca023429aa0eb9.tar.lz4.list
```



10) 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

11) 由于编译 rootfs 的时间较长，如果不想从头开始编译 rootfs，或者编译 rootfs 的过程有问题，可以直接下载 Orange Pi 缓存的 rootfs 压缩包，rootfs 压缩包百度云盘的下载链接如下所示，下载好的 rootfs 压缩包需要放在 orangepi-build 的 **external/cache/rootfs** 目录下才能被编译脚本正常使用

链接: <https://pan.baidu.com/s/1vWQmCmSYdH7iCDFyKpJtVw>

提取码: zero

orangepi-build

① 2020-11-05 12:06 失效时间: 永久有效

返回上一级 全部文件 > orangepi-build

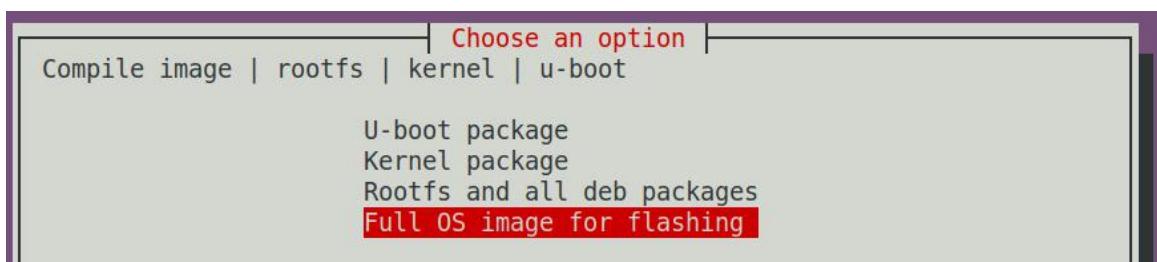
文件名	大小
linux镜像使用的rootfs压缩包	-
toolchains.tar.gz	1.71G
orangepi-build.tar.gz	151.7M

5. 6. 编译 linux 镜像

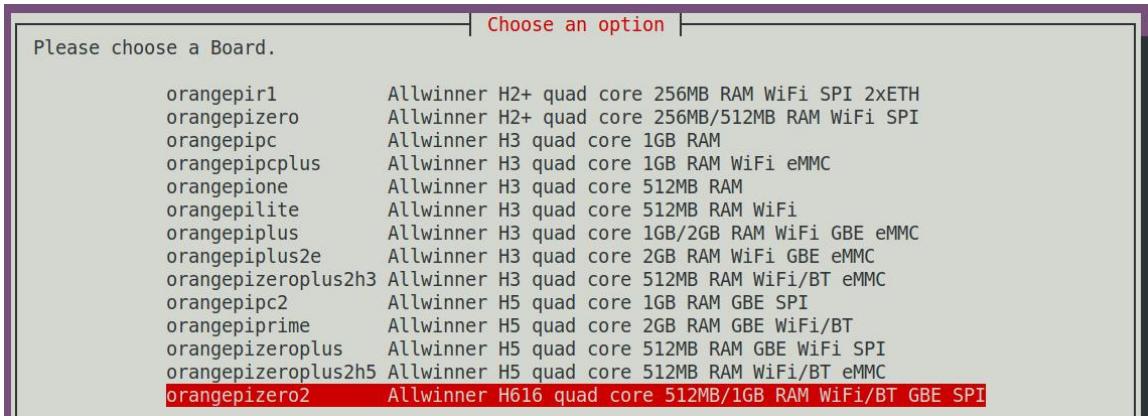
1) 运行 build.sh 脚本，记得加 sudo 权限

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

2) 选择 **Full OS image for flashing**，然后回车

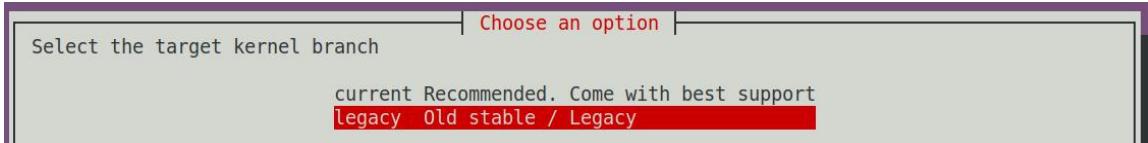


3) 然后选择开发板的型号



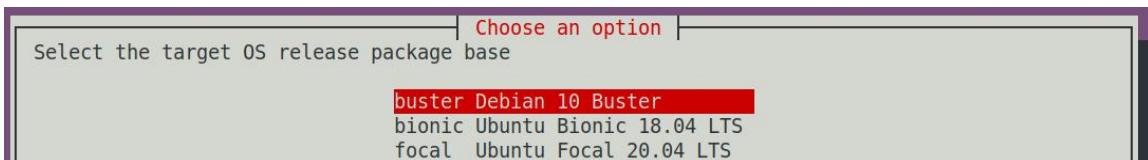
4) 然后选择分支

- a. current 会编译 linux 5.13
- b. legacy 会编译 linux4.9



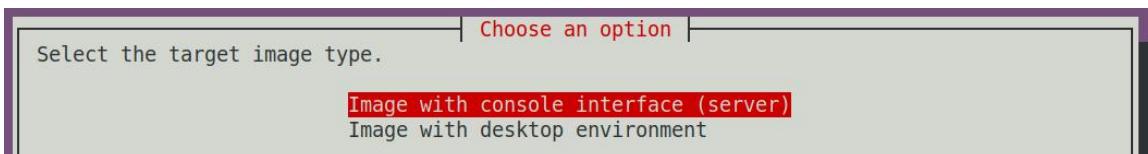
5) 然后选择 rootfs 的类型

- a. **buster** 表示 Debian 10
- b. **bionic** 表示 Ubuntu 18.04
- c. **focal** 表示 Ubuntu 20.04

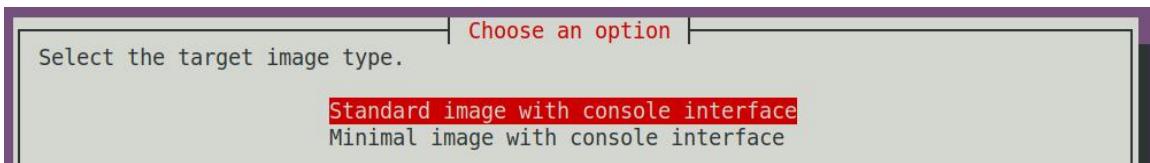


6) 然后选择镜像的类型

- a. **Image with console interface (server)** 表示服务器版的镜像，体积比较小
- b. **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



- 8) 选择镜像的类型后就会开始编译 linux 镜像，编译的大致流程如下
- 初始化 Ubuntu PC 的编译环境，安装编译过程需要的软件包
 - 下载 u-boot 和 linux 内核的源码（如果已经缓存，则只更新代码）
 - 编译 u-boot 源码，生成 u-boot 的 deb 包
 - 编译 linux 源码，生成 linux 相关的 deb 包
 - 制作 linux firmware 的 deb 包
 - 制作 orangepi-config 工具的 deb 包
 - 制作板级支持的 deb 包
 - 如果是编译 desktop 版镜像，还会制作 desktop 相关的 deb 包
 - 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
 - 安装前面生成的 deb 包到 rootfs 中
 - 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
 - 然后制作镜像文件，并格式化分区，默认类型为 ext4
 - 再将配置好的 rootfs 拷贝到镜像的分区中
 - 然后更新 initramfs
 - 最后将 u-boot 的 bin 文件通过 dd 命令写入到镜像中

- 9) 编译完镜像后会提示下面的信息

- 编译生成的镜像的存放路径

```
[ o.k. ] Done building  
[ output/images/orangepizero2_2.1.8_ubuntu_bionic_server_linux4.9.170/orangepizero2_2.1.8_ubuntu_bionic_server_linux4.9.170.img ]
```

- 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=orangepizero2  
BRANCH=legacy BUILD_OPT=image RELEASE=bionic BUILD_MINIMAL=no ]
```



```
BUILD_DESKTOP=no KERNEL_CONFIGURE=yes ]
```



6. Android SDK 使用说明

Android SDK 的编译是在安装有 **Ubuntu 14.04** 的 PC 上进行的，其它版本的 Ubuntu 系统可能会有一些区别，Ubuntu14.04 amd64 版本的镜像下载地址如下所示

<https://repo.huaweicloud.com/ubuntu-releases/14.04/ubuntu-14.04.6-desktop-amd64.iso>

Android SDK 是全志释放的原始 SDK，如果想在 Orange Pi 开发板上使用 Android SDK 编译出来的 Android 镜像，需要针对 Orange Pi 开发板进行适配，才能保证所有功能使用正常

6.1. 下载 Android SDK 的源码

- 1) H616 的 Android 源码文件夹中包含如下 4 个文件
 - a. **android.tar.gz**: android 相关的源码
 - b. **android.tar.gz.md5sum**: android.tar.gz 的 MD5 校验和文件
 - c. **longan.tar.gz**: 包含 u-boot, linux 内核等源码（不包含 boot0 的源码）
 - d. **longan.tar.gz.md5sum**: longan.tar.gz 的 MD5 校验和文件

H616_Android_Source_Code		
返回上一级 全部文件 > H616_Android_Source_Code		
<input type="checkbox"/> 文件名	大小	修改日期
<input type="checkbox"/> longan.tar.gz.md5sum	49B	2020-11-04 13:47
<input type="checkbox"/> longan.tar.gz	1.31G	2020-11-04 13:47
<input type="checkbox"/> android.tar.gz.md5sum	49B	2020-11-04 13:47
<input type="checkbox"/> android.tar.gz	20.74G	2020-11-04 13:47

- 2) 下载完 Android 源码后请先检查下 MD5 校验和是否正确，如果不正确，请重新下载源码

```
test@test:~$ md5sum -c android.tar.gz.md5sum
```

android.tar.gz: 确定

```
test@test:~$ md5sum -c longan.tar.gz.md5sum
```

longan.tar.gz: 确定



3) 然后解压 Android 源码

- a. android: 存放 android 相关的源码
- b. longan: 存放 linux 内核和 u-boot 的源码（不包含 boot0 的源码），以及其他
的配置文件

```
test@test:~$ tar -zxf android.tar.gz
test@test:~$ tar -zxf longan.tar.gz
test@test:~$ ls
android  longan
```

6. 2. 搭建 Android 编译环境

1) 使用 Ubuntu 14.04 编译 Android 10 源码，需要确保 Ubuntu 14.04 使用的是 **linux 4.4** 的内核，否则编译时会报错。安装 Ubuntu14.04 系统时如果使用的是 **ubuntu-14.04.6-desktop-amd64.iso** 这个最新版本的 Ubuntu14.04 镜像文件，那么安装好系统后内核默认就是 linux 4.4，如果你现在的 Ubuntu14.04 系统不是使用的最新的镜像文件安装的，那么首先请检查下内核版本是不是 linux 4.4，如果不是请升级内核到 linux 4.4，或者使用 **ubuntu-14.04.6-desktop-amd64.iso** 重新安装 Ubuntu 系统

```
test@test:~$ uname -a
Linux ubuntu 4.4.0-142-generic #168~14.04.1-Ubuntu SMP Sat Jan 19 11:26:28 UTC
2019 x86_64 x86_64 x86_64 GNU/Linux
```

2) 安装 JDK

```
test@test:~$ sudo add-apt-repository ppa:openjdk-r/ppa
test@test:~$ sudo apt-get update
test@test:~$ sudo apt-get install openjdk-8-jdk
```

3) 配置 JAVA 环境变量

- a. 首先确定 java 的安装路径，一般为

```
test@test:~$ ls /usr/lib/jvm/java-8-openjdk-amd64
ASSEMBLY_EXCEPTION  bin  docs  include  jre  lib  man  src.zip
THIRD_PARTY_README
```

- b. 然后使用下面的命令导出 java 的环境变量

```
test@test:~$ export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```



```
test@test:~$ export PATH=$JAVA_HOME/bin:$PATH  
test@test:~$ export CLASSPATH=.:$JAVA_HOME/lib:$JAVA_HOME/lib/tools.jar
```

4) 安装平台支持软件

```
test@test:~$ sudo apt-get update  
test@test:~$ sudo apt-get install 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  
  
test@test:~$ sudo apt-get install u-boot-tools
```

6. 3. 编译 android 镜像

6. 3. 1. 编译内核

1) 首先配置编译环境

```
test@test:~$ cd longan  
test@test:~/longan$ ./build.sh config
```

Welcome to mkscript setup progress

All available platform:

- 0. android
- 1. linux

Choice [android]: **0**

All available ic:

- 0. h313
- 1. h616
- 2. h700

Choice [h616]: **1**

All available board:

- 0. fpga
- 1. ft
- 2. p1
- 3. p2



- 4. perf1
- 5. perf1_axp152
- 6. perf2
- 7. perf3
- 8. qa

Choice [p2]: **3**

```
INFO: kernel defconfig: generate longan/kernel/linux-4.9/.config by
longan/kernel/linux-4.9/arch/arm64/configs/sun50iw9p1smp_h616_android_defconfig
*** Default configuration is based on 'sun50iw9p1smp_h616_android_defconfig'
#
# configuration written to .config
#
```

2) 然后开始编译

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

3) 编译完后的输出如下所示

```
sun50iw9p1 compile Kernel successful

INFO: build kernel OK.
INFO: build rootfs ...
INFO: skip make rootfs for android
INFO: -----
INFO: build lichee OK.
INFO: -----
```

6. 3. 2. 编译 Android 源码

1) 编译 Android 的命令如下

```
test@test:~$ cd android
test@test:~/android$ source build/envsetup.sh
test@test:~/android$ lunch cupid_p2-eng
test@test:~/android$ extract-bsp
test@test:~/android$ make -j8
```



2) 编译完成会打印下面的信息

```
##### build completed successfully (01:51 (mm:ss)) #####
```

3) 然后使用 **pack** 命令打包生成 Android 镜像

```
test@test:~/android$ pack
.....
-----image is at-----
longan/out/h616_android10_p2_uart0.img

pack finish
use pack4dist for release
```

4) 生成的安卓镜像存放的路径为

```
longan/out/h616_android10_p2_uart0.img
```