Orange Pi Zero 2
用户手册

目录

版本更新历史

版本	日期	更新说明
v3.6	2022-04-20	1. Linux 系统 USB 麦克风测试方法 2. 在桌面 Linux 系统中测试音频的方法 3. Linux5.16 I2C、SPI、UART 和 PWM 的测试方法 4. Linux: 腾讯 ncnn 神经网络前向计算框架测试方法 5. Ubuntu20.04 ROS 2 Galactic 的安装方法 6. Debian12 Linux5.16 系统 GPU 测试说明 7. 添加新版本 orangepi-build 使用说明 8. 添加部分注意事项说明
v3.7	2022-05-20	1. 支持 Ubuntu22.04 的服务器和桌面版镜像 2. Ubuntu22.04 Linux5.16 GPU 测试说明 3. Ubuntu22.04 安装浏览器的方法 4. Linux 桌面版系统 root 用户自动登录的设置方法 5. Linux 桌面版系统禁用桌面的方法 6. Linux:手动扩容 TF 卡中 rootfs 分区容量的方法 7. Linux:缩小 TF 卡中 rootfs 分区容量的方法 8. Linux 桌面版系统触摸屏隐藏鼠标光标的方法 9. Linux:face_recognition 人脸识别库的安装和测试方法 10. Debian: OpenMediaVault 5.x 和 6.x 的安装方法 11. Ubuntu 20.04 ROS 1 Noetic 的安装方法 12. Linux: Pi-hole 的安装方法 13. Debian10: Tensorflow 的安装方法 14. Linux:GotoHTTP 使用介绍
v3.8	2022-06-16	1. Linux: 安装内核头文件的方法 2. 设置 Linux 系统第一次启动自动连接网络的方法 3. Linux: KlipperScreen 打开鼠标光标方法 4. Linux: wiringOP-Python 的安装使用方法 5. Linux: USB 触摸屏的使用方法
v3.9	2022-07-08	1. Ubuntu20.04/22.04: 向日葵远程控制软件的安装方法 2. 更新部分链接和图片
v3.10	2022-07-29	1. orangepi-build next 分支：支持在 Ubuntu22.04 PC 上使用
v4.0	2022-08-10	1. 添加 Orange Pi Zero2 v1.5 版本的说明图片

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 TV 10、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。

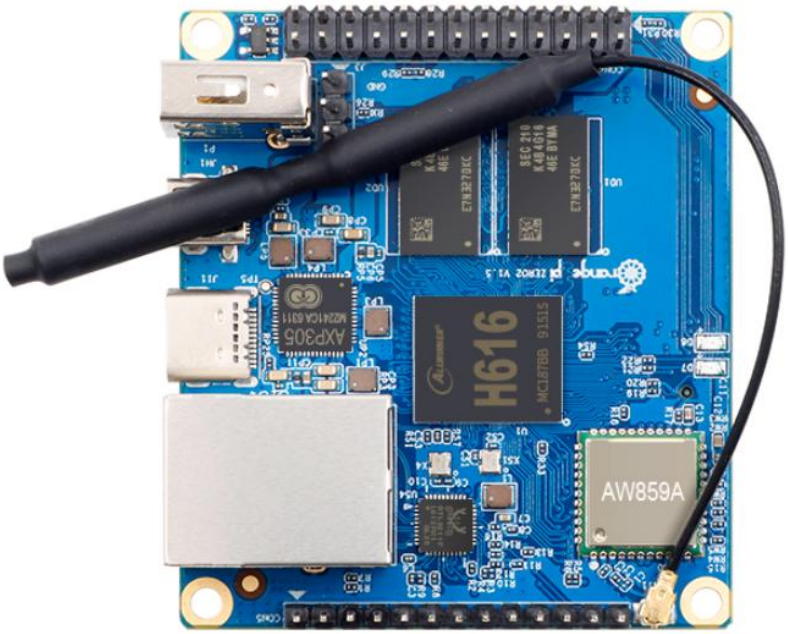
1.4. Orange Pi Zero 2 的硬件特性

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 TV、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	

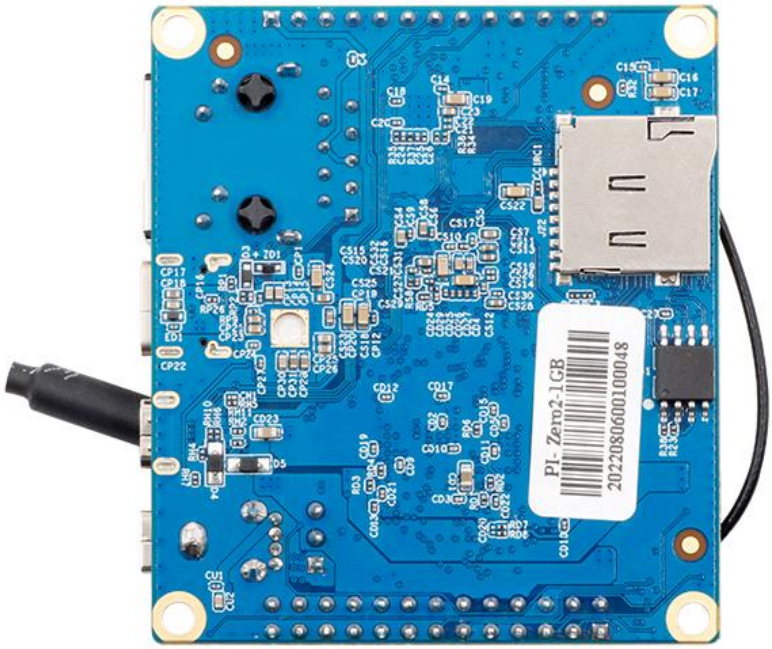
1.5. Orange Pi Zero 2 的顶层视图和底层视图

1.5.1. Orange Pi Zero 2 v1.5 版本的顶层视图和底层视图

顶层视图：

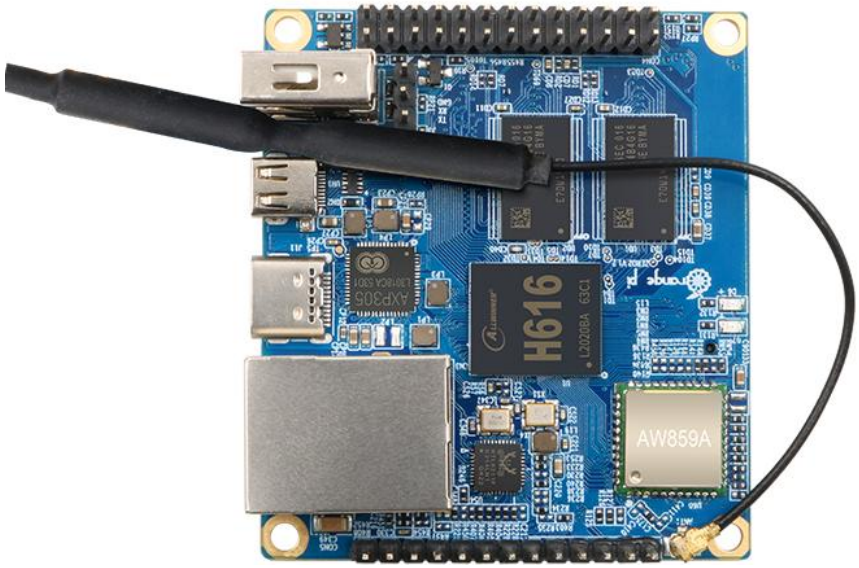


底层视图：

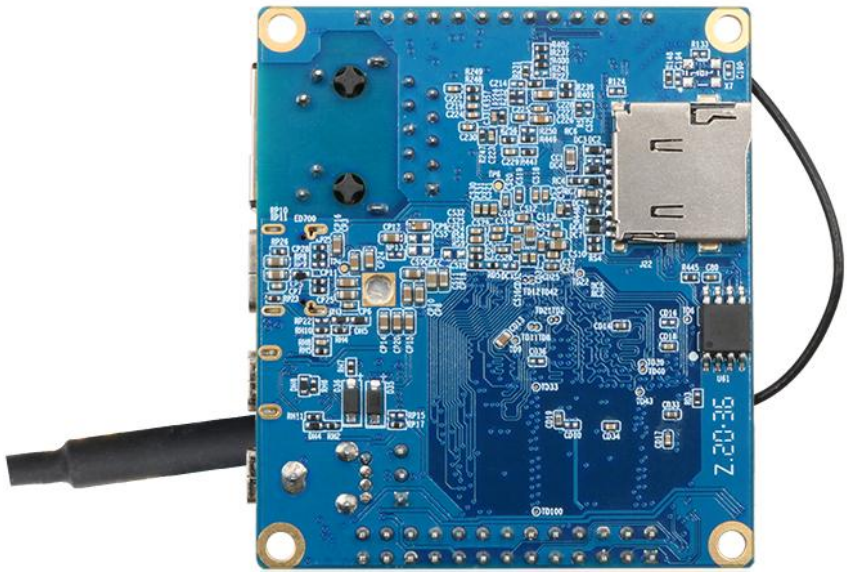


1.5.2. Orange Pi Zero 2 v1.3 版本的顶层视图和底层视图

顶层视图：



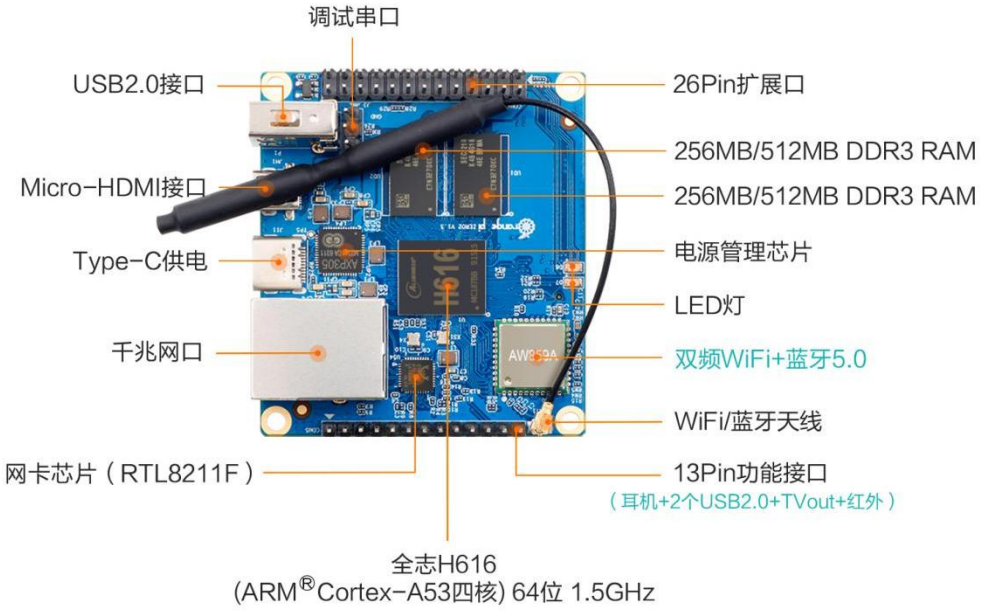
底层视图：



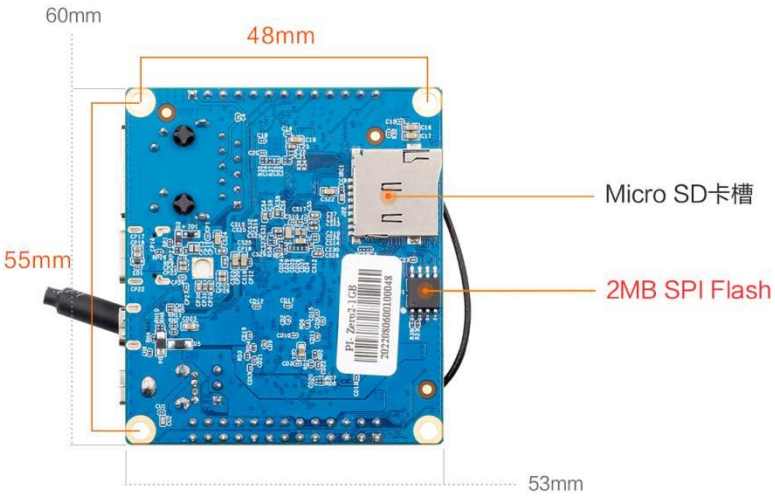
1.6. Orange Pi Zero 2 的接口详情图

1.6.1. Orange Pi Zero 2 v1.5 版本的接口详情图

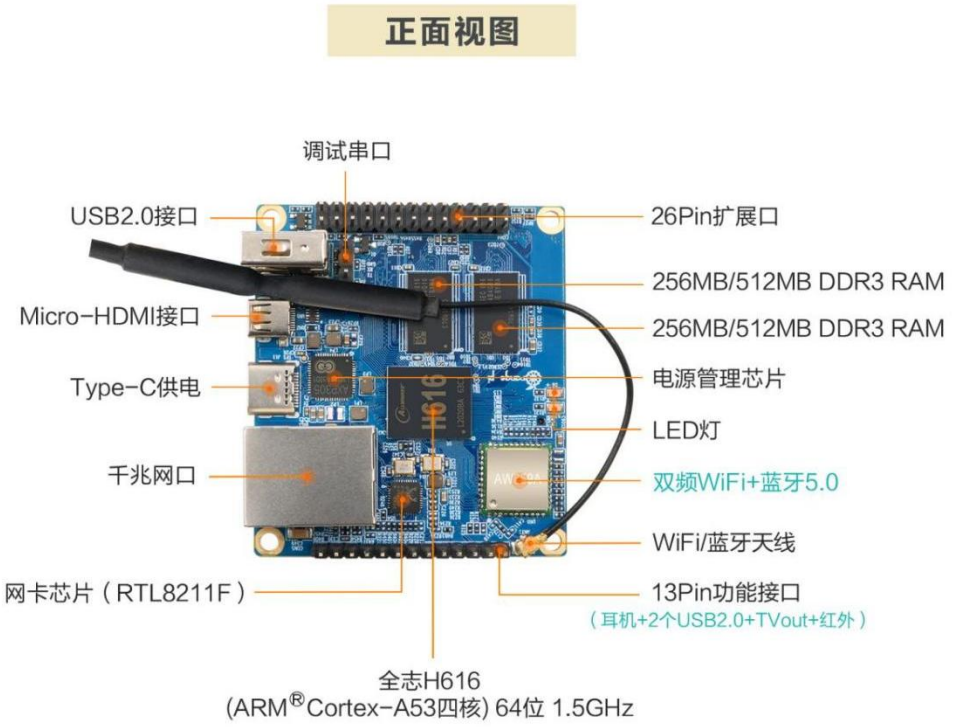
正面视图

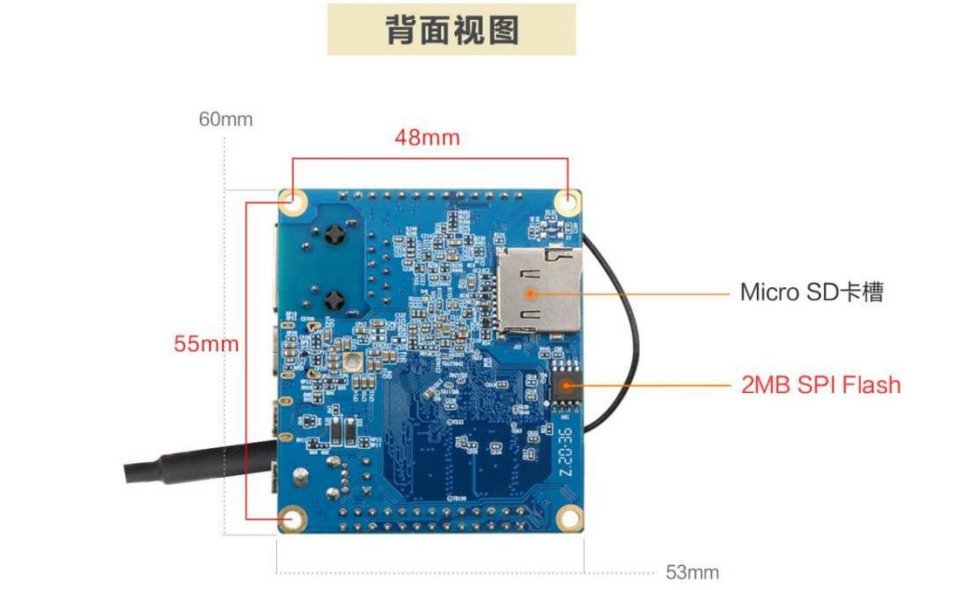


背面视图

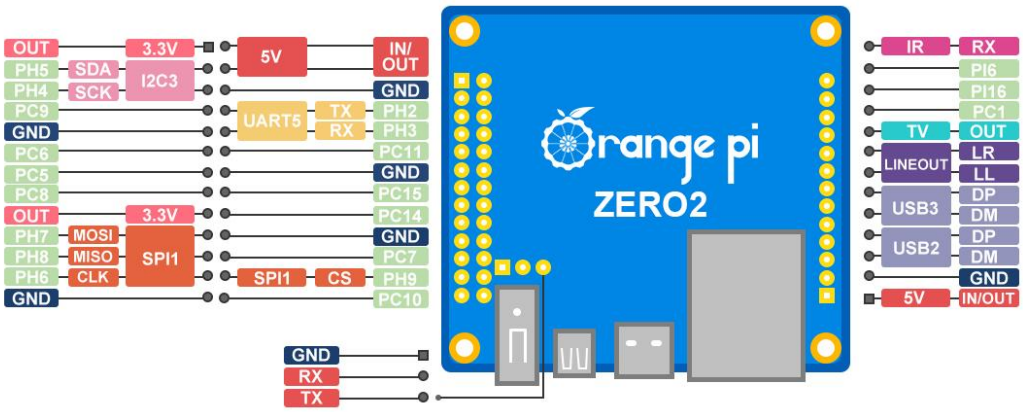


1.6.2. Orange Pi Zero 2 v1.3 版本的接口详情图





Orange Pi Zero2 v1.3 和v1.5 版本的结构都是一样的，功能也没有区别。最新版本的镜像也是通用的。v1.5 版本主要是把电容电阻小型化了，优化了部分走线。



四个定位孔的直径都是 3.0mm。

2. 开发板使用介绍

2.1. 准备需要的配件

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

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

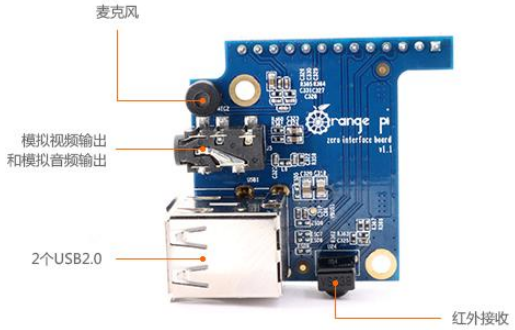
3) Micro HDMI 转 HDMI 连接线，用于将开发板连接到 HDMI 显示器或者电视进行显示

注意，请不要使用下图所示的这种比较宽的Micro HDMI转接头，由于开发板的USB接口、Micro HDMI接口和Type-C电源接口之间的间距比较小，可能会导致三者无法同时插入到开发板。

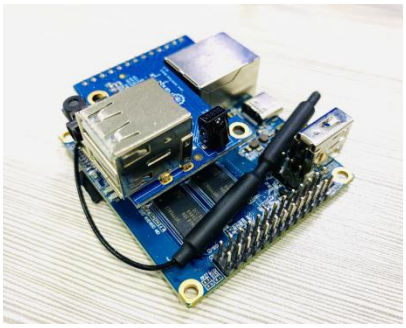
4) 电源，如果有 5V/2A 或 5V/3A 的电源头那就只需要准备一根下面左边图片所示的 USB 转 Type C 接口的数据线，另外也可以使用类似下面右边图片所示的线和电源头一体的 5V/2A 或者 5V/3A 的高品质 USB Typc C 接口电源适配器

5) 13pin 扩展板

a. 扩展板实物如下所示



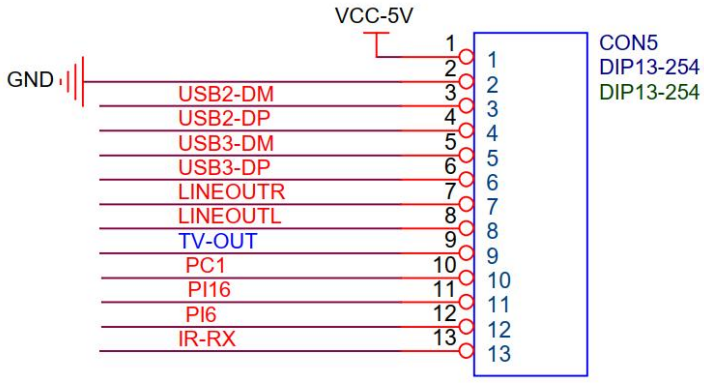
b. 扩展板插入开发板的方式如下所示，切记不要插反了



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。 如果需要使用 MIC 功能，可以通过 USB 接口来扩展，**手册第三章中有 USB MIC 的使用方法**。
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

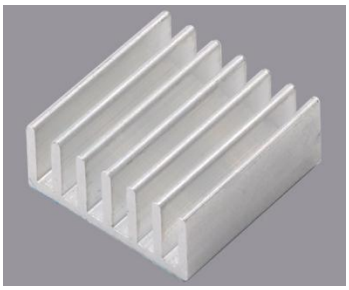
7) 红外遥控器，主要用于控制安卓系统

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

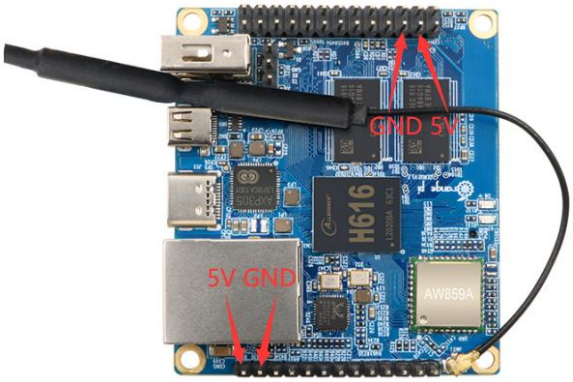
9) AV 视频线，如果希望通过 AV 接口而不是 HDMI 接口来显示视频，那么就需要通过 AV 视频线将开发板连接到电视

10) 散热片，如果担心开发板的温度过高，可以加个散热片，散热片贴在 H616 芯片上即可



11) 5V 的散热风扇，如下图所示，开发板的 26pin 和 13pin 接口上都有 5V 和 GND 引脚可以接散热风扇，26pin 和 13pin 排针的间距为 **2.54mm**，散热风扇的电源接口参照这个规格去淘宝购买即可

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



12) 配套外壳，有透明外壳和黑色外壳可供选择

注意，Orange Pi Zero 2 的配套外壳是无法装下 13pin扩展板的。

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

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

14) 安装有 Ubuntu 和 Windows 操作系统的个人电脑

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（旧版 orangepi-build）
3	Ubuntu22.04 PC	可选，用于编译 Linux 源码（新版 orangepi-build）
4	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/html/hardWare/computerAndMicrocontrollers/service-and-support/Orange-Pi-Zero-2.html

2) 英文版资料的下载网址为

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

3) 资料主要包含

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

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

2.3.1. 使用 balenaEtcher 烧录 Linux 镜像的方法

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

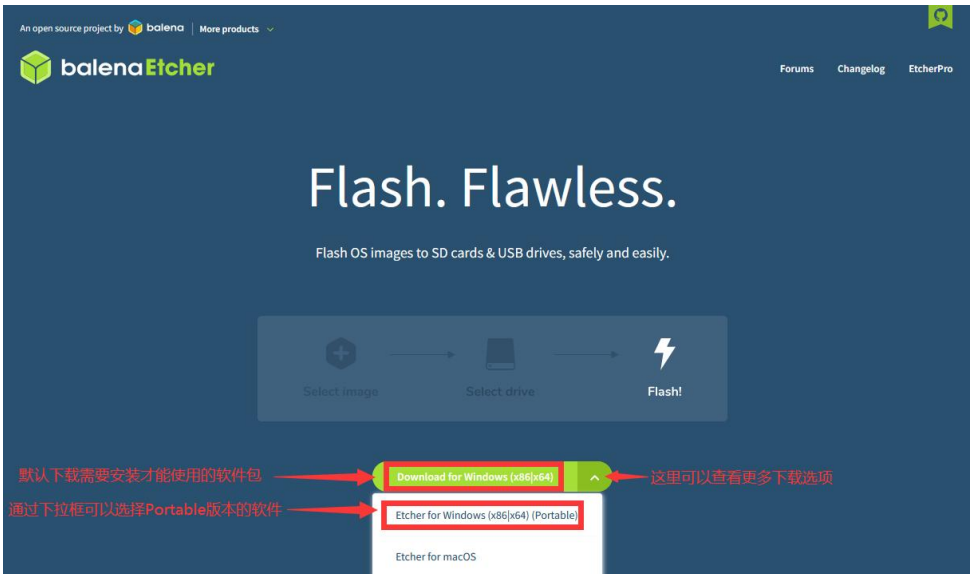
2) 然后使用读卡器把 TF 卡插入电脑

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

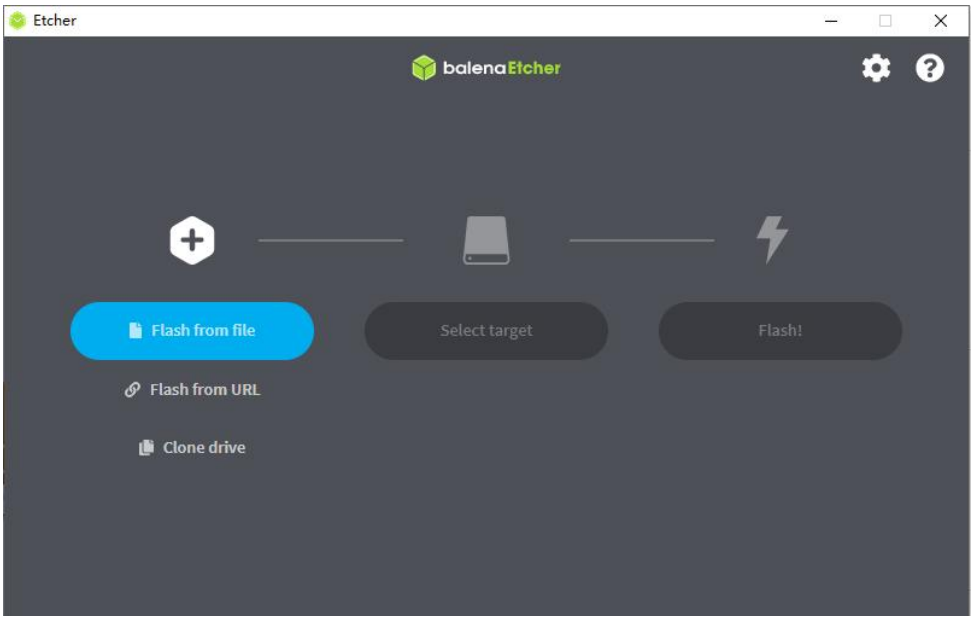
4) 然后下载 Linux 镜像的烧录软件——**balenaEtcher**，下载地址为

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

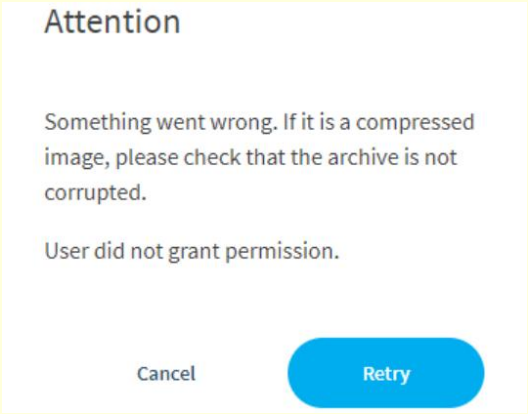
5) 进入 balenaEtcher 下载页面后，点击绿色的下载按钮就可以下载 balenaEtcher 的安装包，也可以通过下拉框选择 balenaEtcher 的 Portable 版本的软件，Portable 版本无需安装，双击打开就可以使用



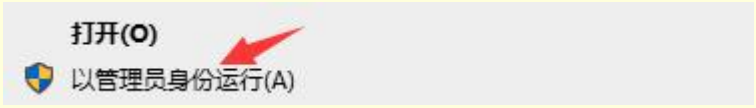
6) 如果下载的是需要安装版本的 balenaEtcher，请先安装再使用。如果下载的 Portable 版本 balenaEtcher，直接双击打开即可，打开后的 balenaEtcher 界面如下图所示



打开 balenaEtcher 时如果提示下面的错误：



请选择 balenaEtcher 后点击右键，然后选择以管理员身份运行。

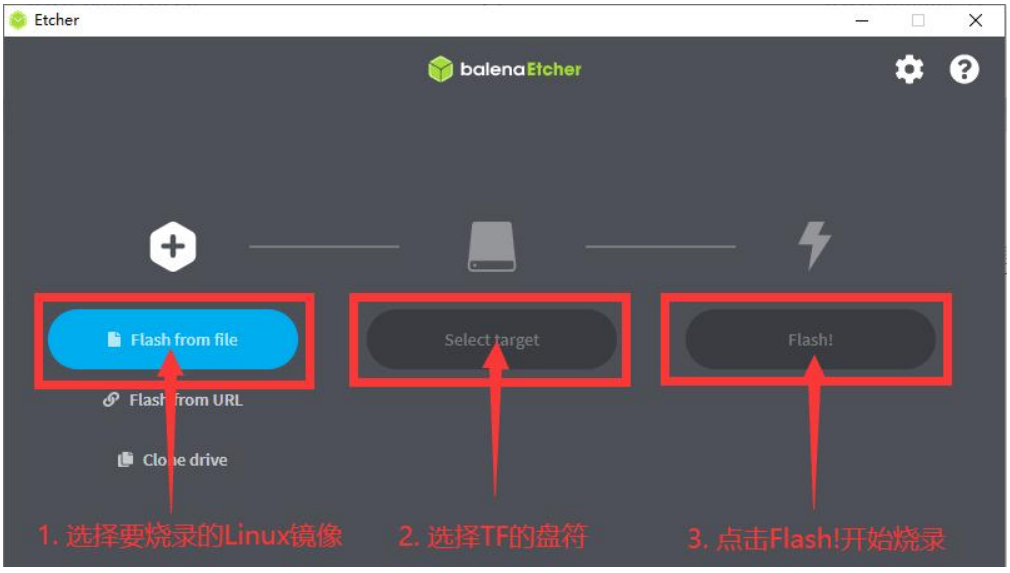


7) 使用 balenaEtcher 烧录 Linux 镜像的具体步骤如下所示

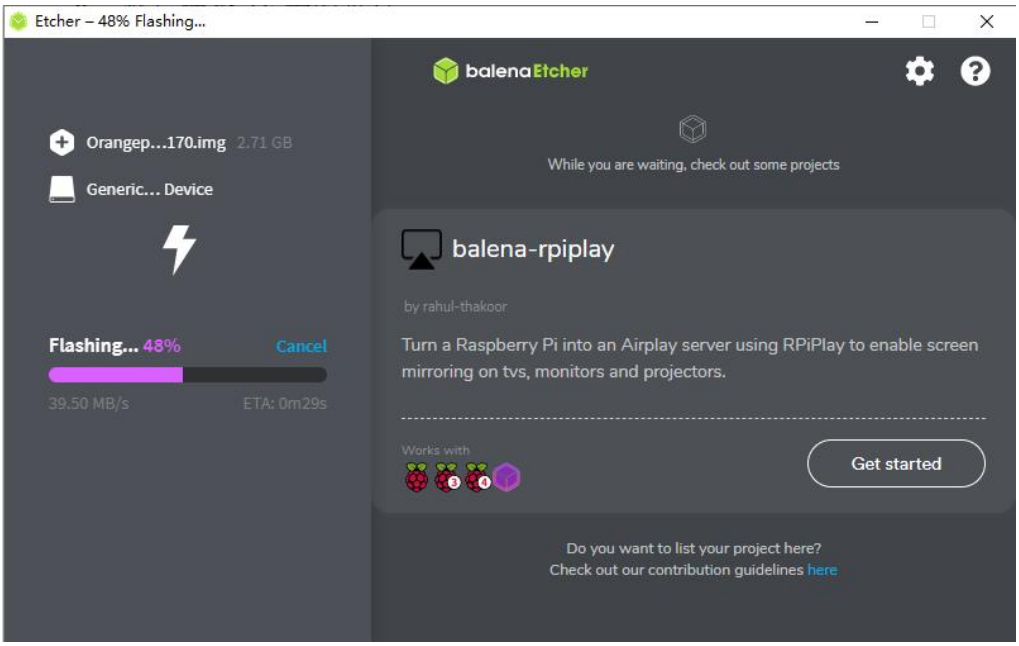
a. 首先选择要烧录的 Linux 镜像文件的路径

b. 然后选择 TF 卡的盘符

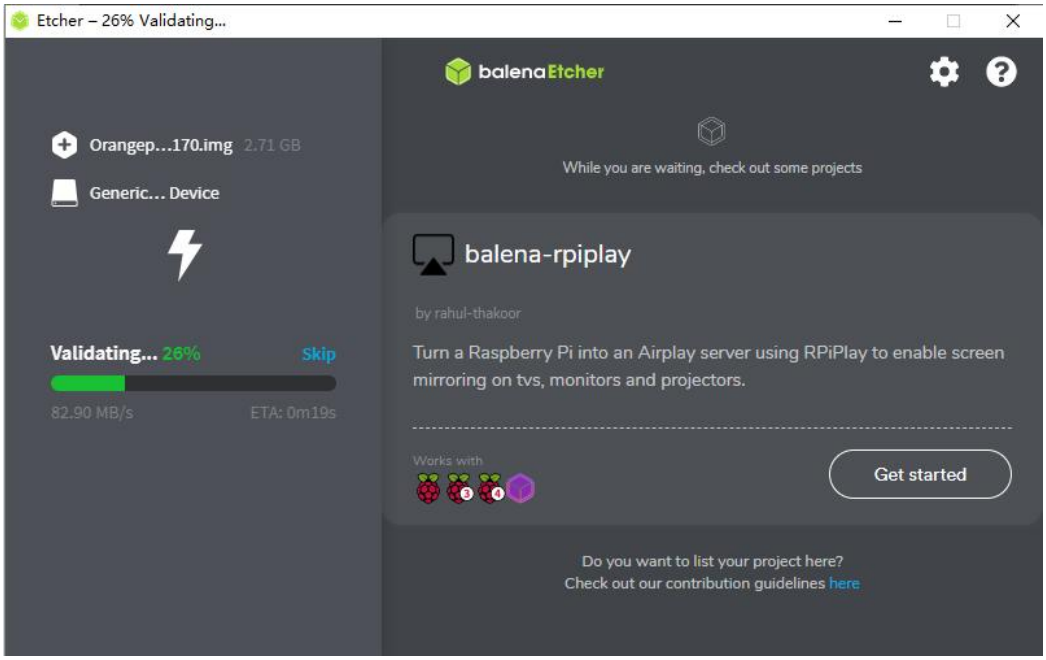
c. 最后点击 Flash 就会开始烧录 Linux 镜像到 TF 卡中



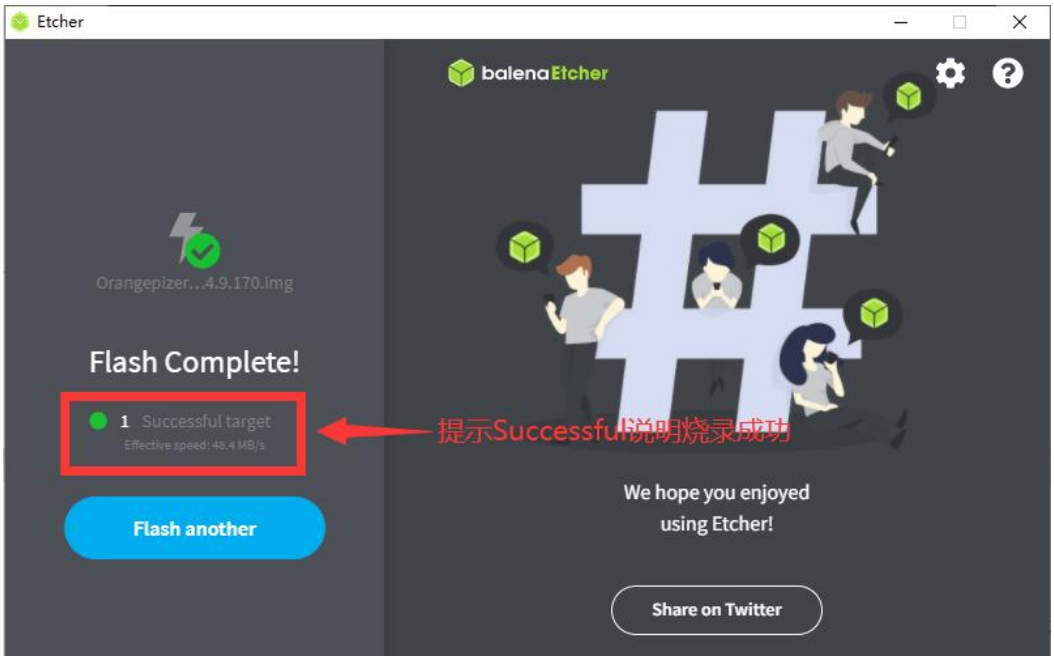
8) balenaEtcher 烧录 Linux 镜像的过程显示的界面如下图所示，另外进度条显示紫色表示正在烧录 Linux 镜像到 TF 卡中



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



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



2.3.2. 使用 Win32Diskimager 烧录 Linux 镜像的方法

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

2) 然后使用读卡器把 TF 卡插入电脑

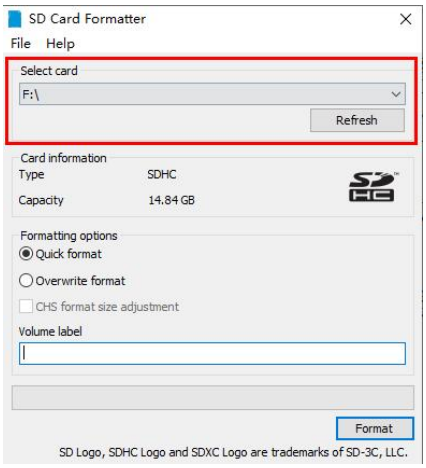
3) 接着格式化 TF 卡

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

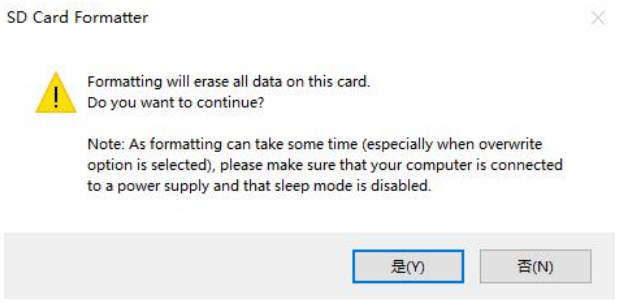
https://www.sdcard.org/downloads/formatter/eula_windows/SDCardFormatterv5_WinEN.zip

b. 下载完后直接解压安装即可，然后打开软件

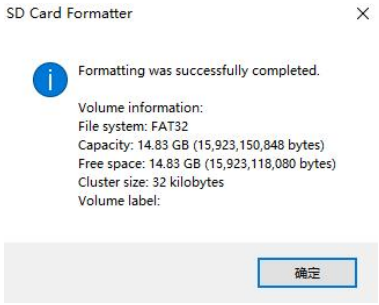
c. 如果电脑只插入了 TF 卡，则“**Select card**”一栏中会显示 TF 卡的盘符，如果电脑插入了多个 USB 存储设备，可以通过下拉框选择 TF 卡对应的盘符



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



e. 格式化完 TF 卡后会弹出下图所示的信息，点击确定即可



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

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

a. Win32Diskimager 的下载页面为

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

b. 下载完后直接安装即可，Win32Diskimager 界面如下所示
   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

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

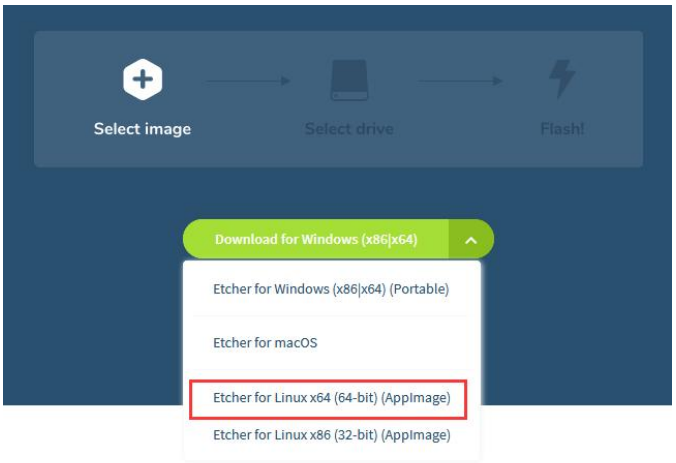
1) 首先准备一张 8GB 或更大容量的 TF 卡，TF 卡的传输速度必须为 **class10** 级或 **class10** 级以上，建议使用闪迪等品牌的 TF 卡

2) 然后使用读卡器把 TF 卡插入电脑

3) 下载 balenaEtcher 软件，下载地址为

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

4) 进入 balenaEtcher 下载页面后，请通过下拉框选择 Linux 版本的软件进行下载



5) 下载完后请先使用 **unzip** 命令解压下载的压缩包，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

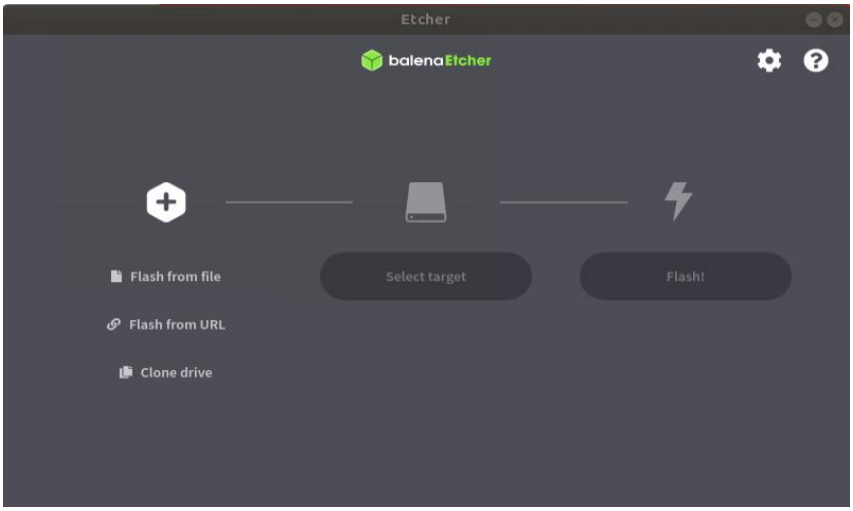
```
test@test:~$ 7z x Orangepizero2_2.2.0_ubuntu_focal_desktop_linux4.9.170.7z
test@test:~$ ls Orangepizero2_2.2.0_ubuntu_focal_desktop_linux4.9.170.*
Orangepizero2_2.2.0_ubuntu_focal_desktop_linux4.9.170.7z
Orangepizero2_2.2.0_ubuntu_focal_desktop_linux4.9.170.sha     #校验和文件
Orangepizero2_2.2.0_ubuntu_focal_desktop_linux4.9.170.img     #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
orangepizero2_2.2.0_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**），balenaEtcher 打开后的界面显示如下图所示

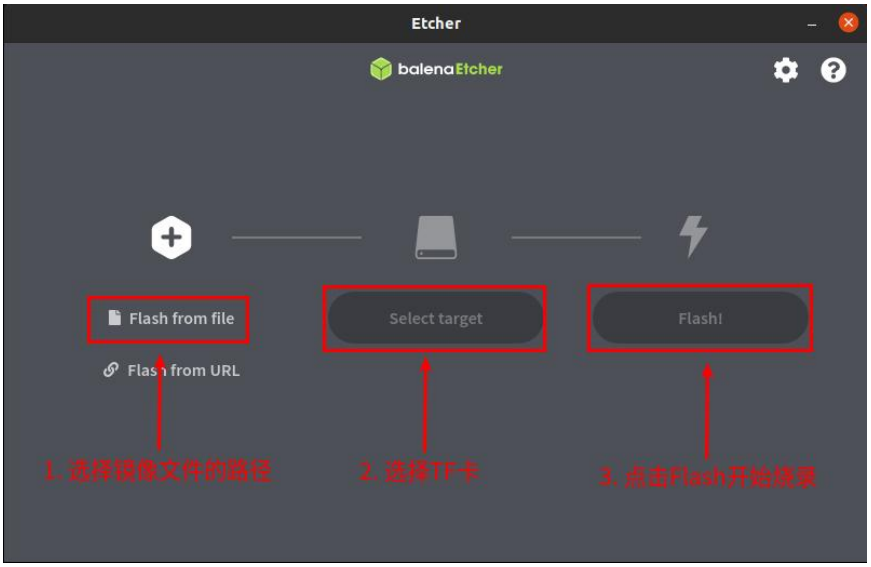


9) 使用 balenaEtcher 烧录 Linux 镜像的具体步骤如下所示

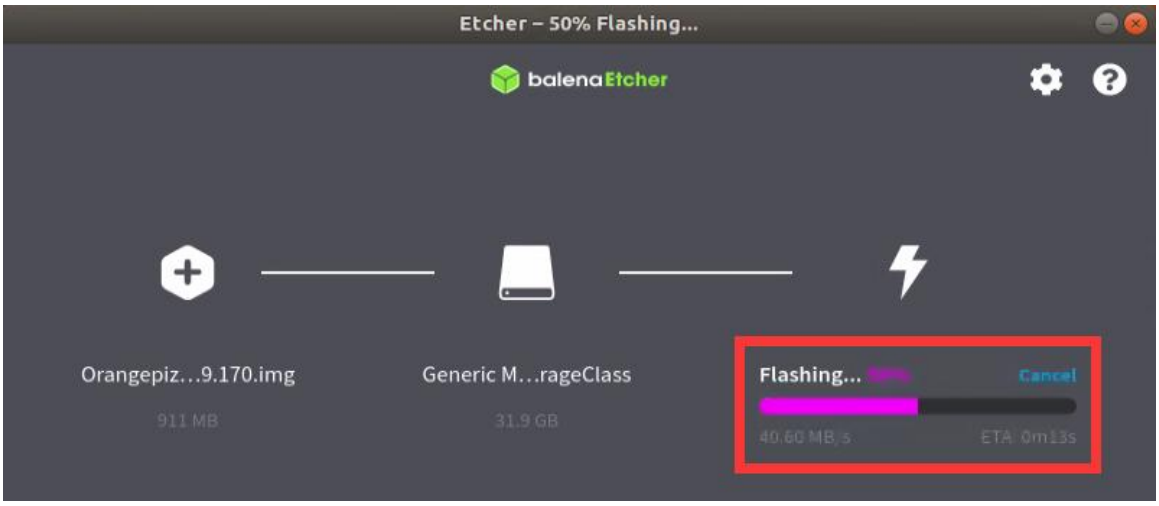
a. 首先选择要烧录的 Linux 镜像文件的路径

b. 然后选择 TF 卡的盘符

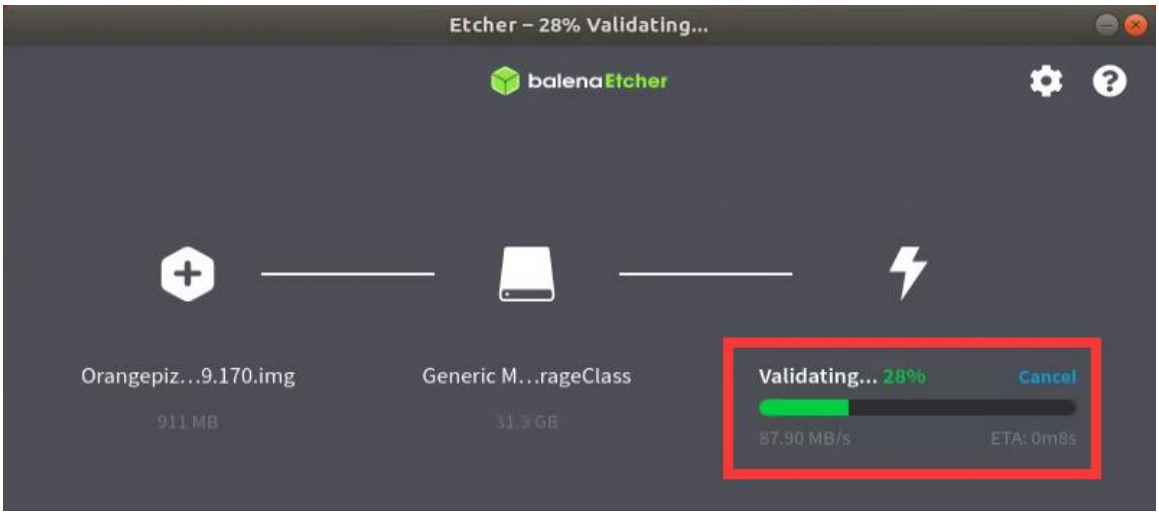
c. 最后点击 Flash 就会开始烧录 Linux 镜像到 TF 卡中



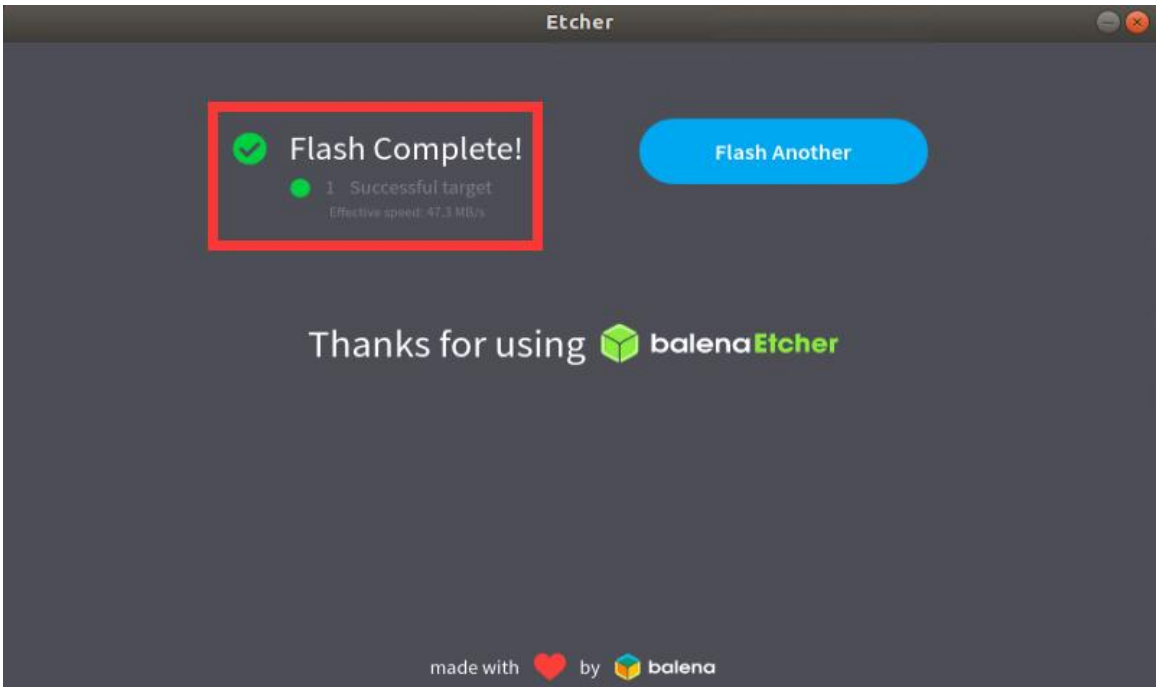
10) balenaEtcher 烧录 Linux 镜像的过程显示的界面如下图所示，另外进度条显示紫色表示正在烧录 Linux 镜像到 TF 卡中



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



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



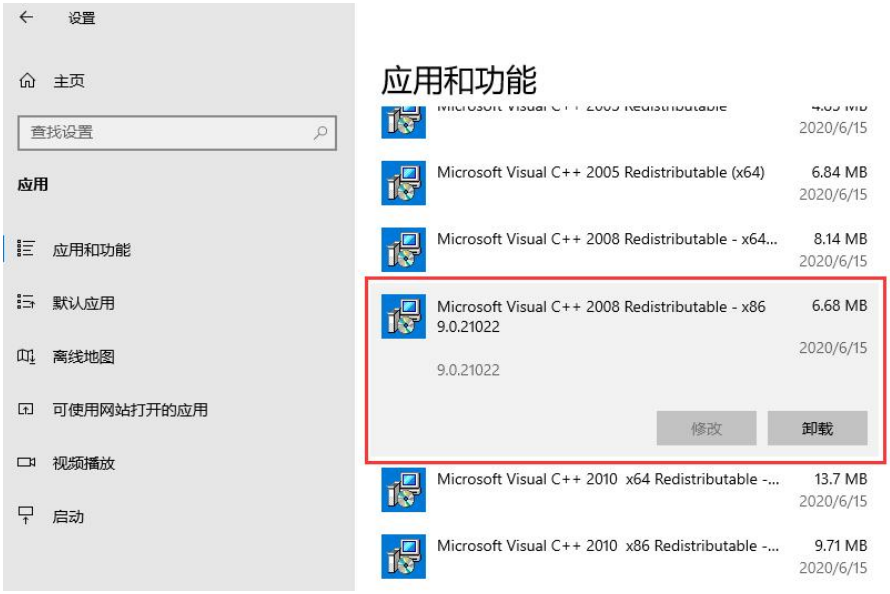
2.5. 烧写 Android 镜像到 TF 卡的方法

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

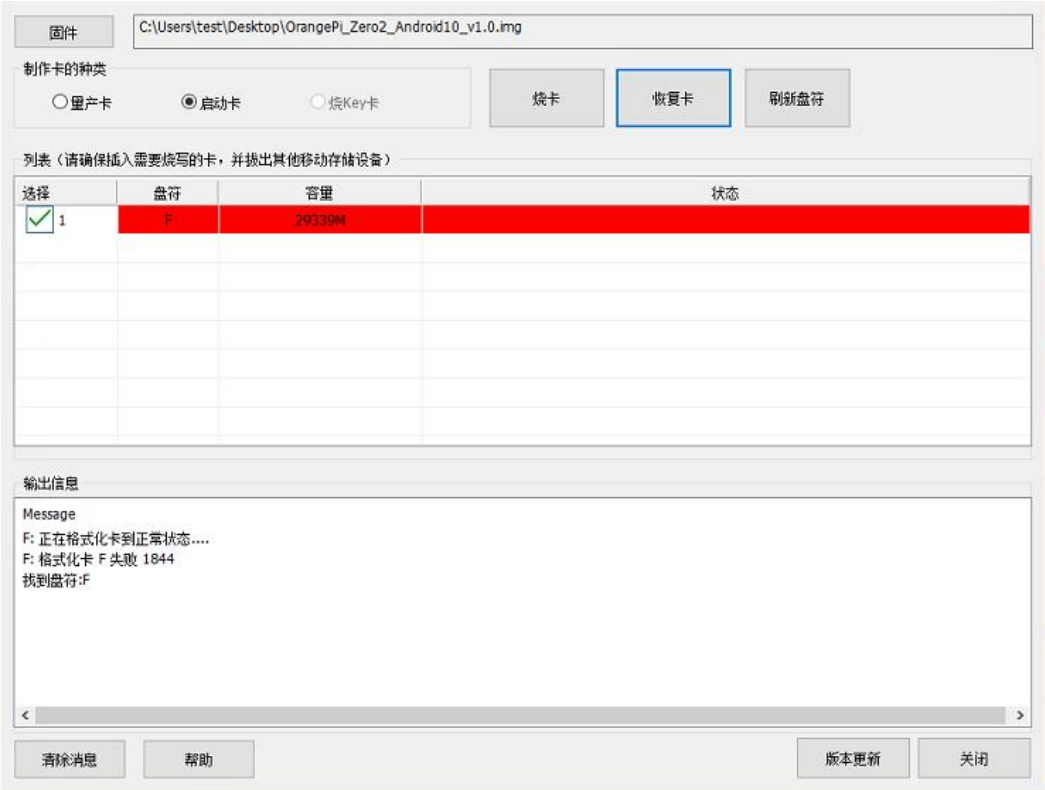
请不要用烧录 Linux 镜像的软件，如 Win32Diskimager 或者 balenaEtcher 来烧录安卓镜像。

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

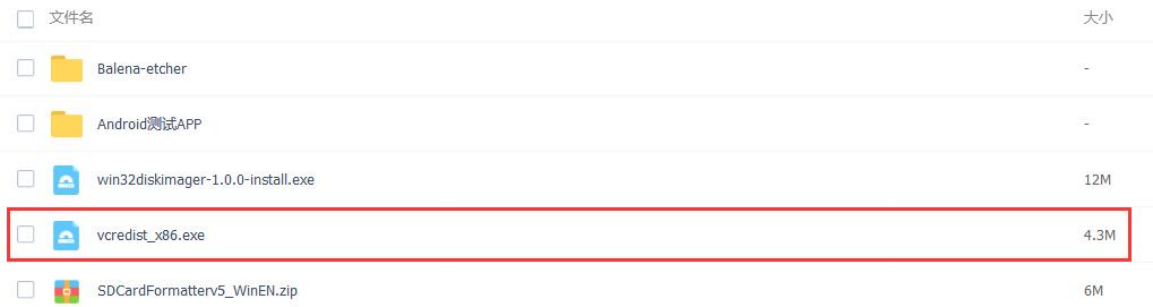
1) 首先请确保 Windows 系统已经安装了 **Microsoft Visual C++ 2008 Redistrbutable - x86**



2) 如果没有安装 **Microsoft Visual C++ 2008 Redistrbutable - 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 Redistrbutable - 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



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

5) 然后使用读卡器把 TF 卡插入电脑

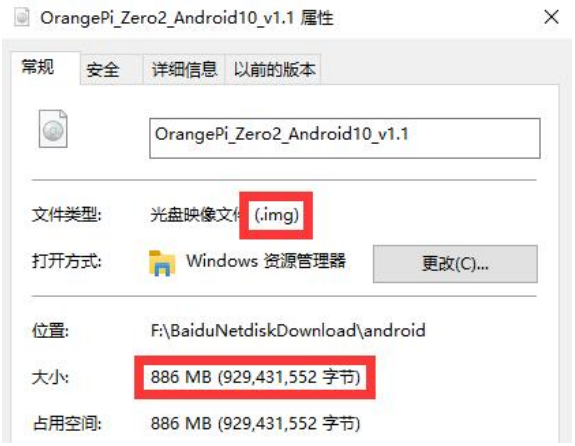
6) 从 Orange Pi 的资料下载页面下载 Android10 镜像和 PhoenixCard 烧写工具，请确保 PhonenixCrad 工具的版本为 PhonixCard-4.2.8，请不要用低于 4.2.8 版本的 PhonixCard 软件来烧录 Orange Pi Zero 2 的 Android 10 镜像，低于这个版本的 PhonixCard 工具烧写的 Android10 镜像可能会有问题

7）然后使用解压软件解压下载的 Android 镜像的压缩包，解压后的文件中，以“**.img**”结尾的文件就是 Android 镜像文件

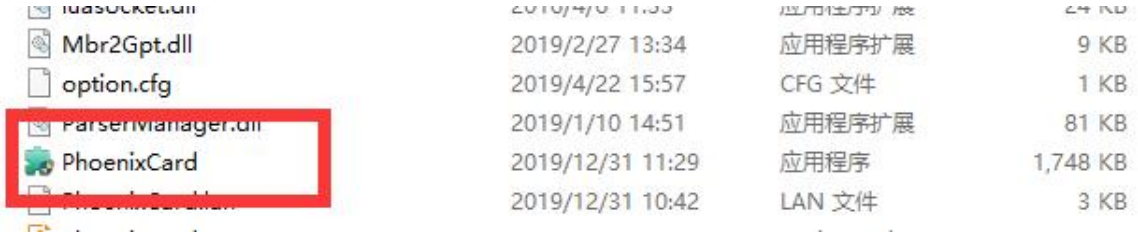
a. 如果不知道怎么解压 Android 镜像的压缩包，可以安装一个 360 压缩软件

b. 使用 360 压缩软件解压后可以看到下面的几个文件

c. 其中第一个接近 900M 左右的文件就是要烧录的安卓镜像文件，查看其属性如下图所示



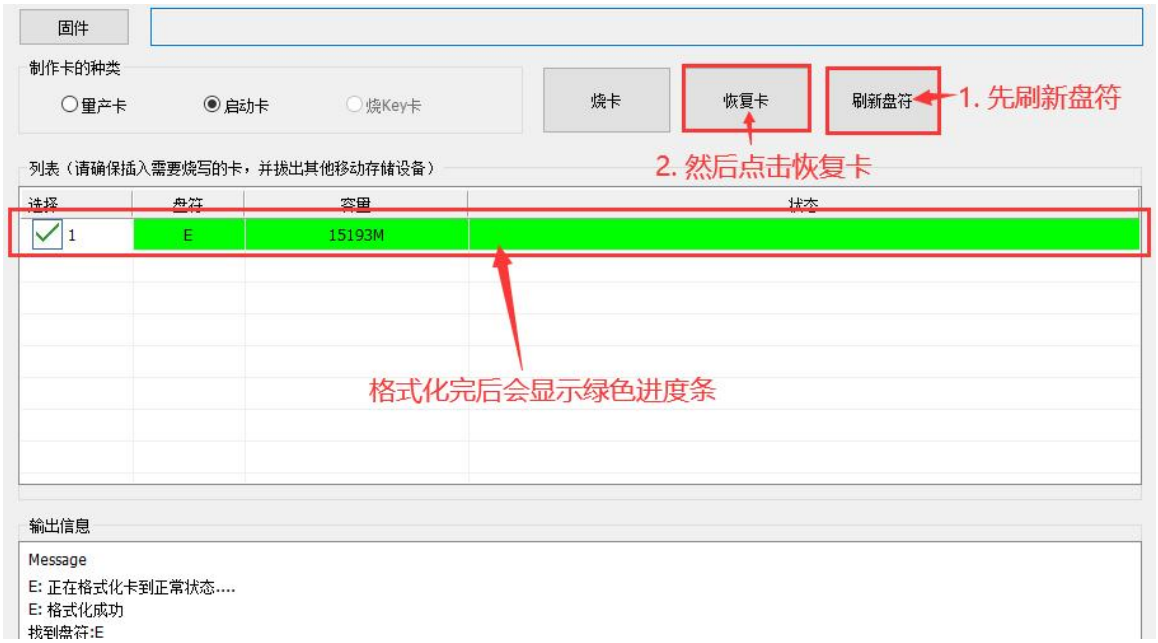
8) 然后使用解压软件解压 PhonixCard4.2.8.zip，此软件无需安装，在解压后的文件夹中找到 PhoenixCard 打开即可



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

10) 确认完盘符后，先格式化 TF 卡，点击 PhoenixCard 中“**恢复卡**”按钮即可（如

果“**恢复卡**”按钮为灰色的无法按下，可以先点击下“**刷新盘符**”按钮）



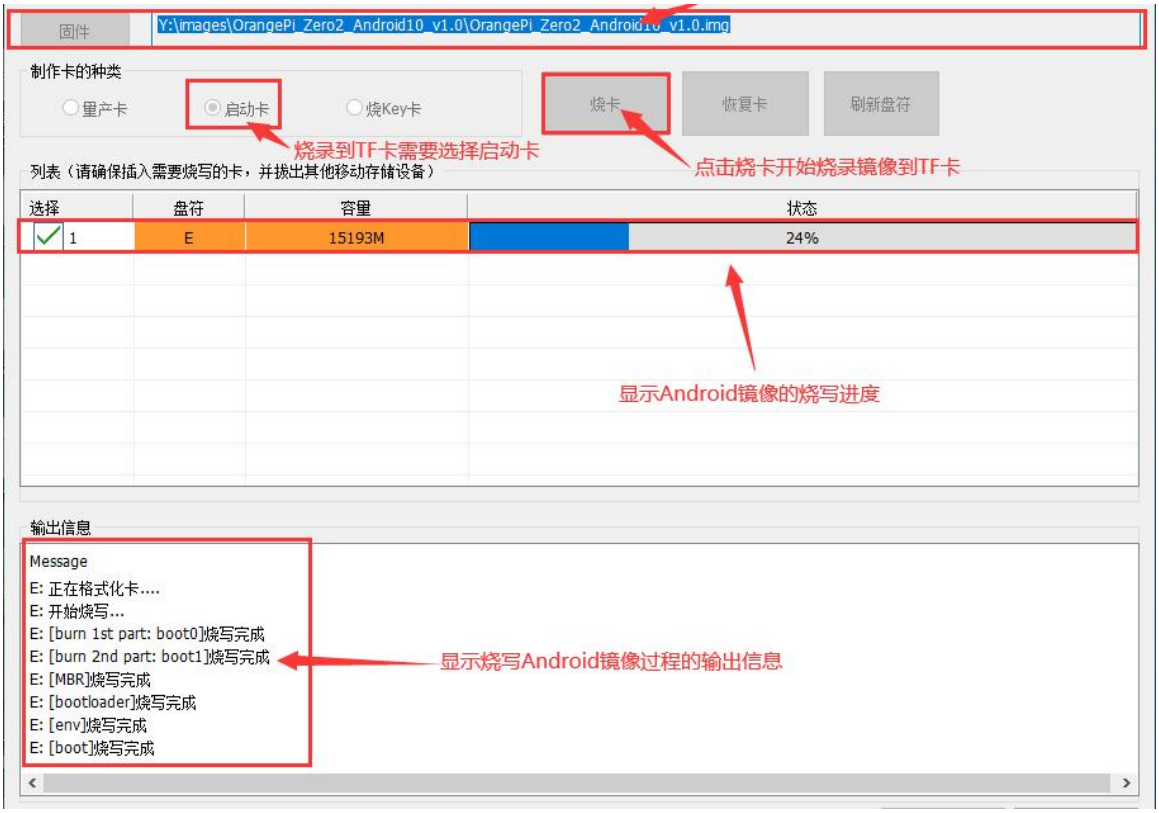
如果格式化有问题，请尝试拔插下 TF 卡后再测试，如果重新拔插 TF 卡后还是有问题，可以重启下 Window 电脑或者换一台电脑再试下。

11) 然后开始将 Android 镜像写入 TF 卡

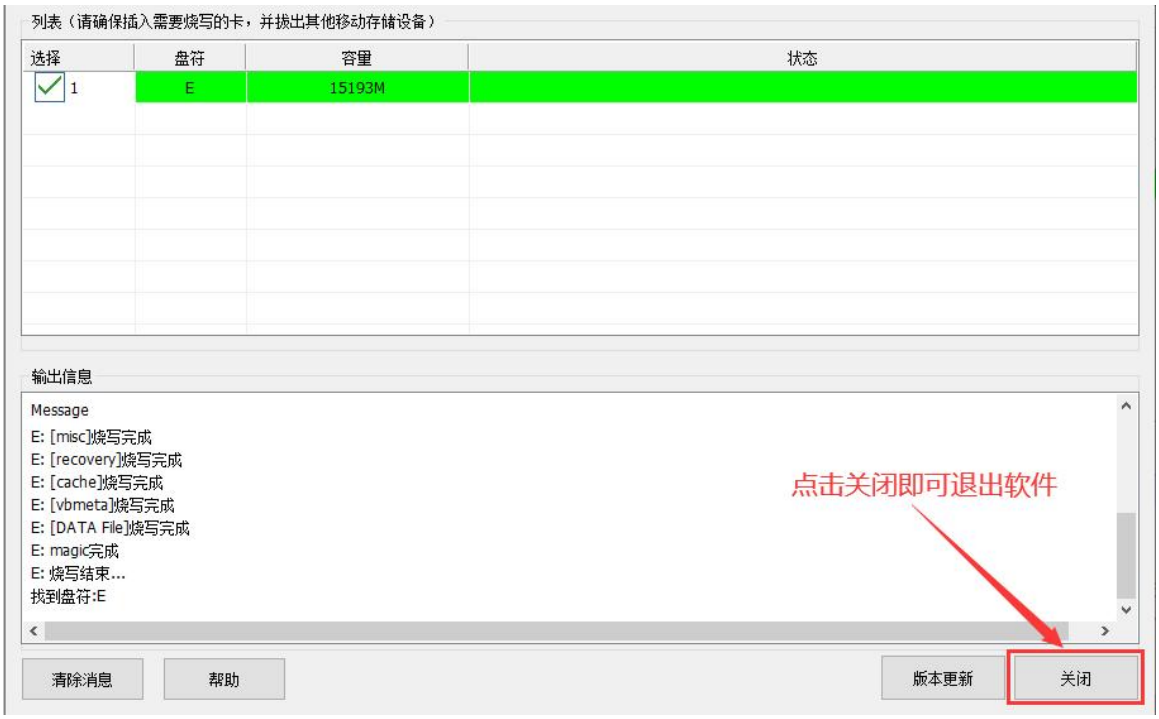
a. 首先在“**固件**”一栏中选择 Android 镜像的路径

b. 在“**制作卡的种类**”中选择“**启动卡**”

c. 然后点击“**烧卡**”按钮就会开始烧录



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



烧录完Android系统后在Windows中TF卡只能看到一个 128 MB的分区，显示的分区如下图所示（有些电脑可能会弹出十几个磁盘分区，但也只能打开 128 MB的那个分区），请注意，这是正常的，请不要以为TF卡烧坏了。之所以这样，是因为安卓系统总共有 17 个分区，其他 16 分区在Windows系统中是无法正常识别的。此时，请放心的拔下TF卡然后插入开发板中启动即可。

安卓系统启动后，使用下面的命令可以看到TF卡中的这 17 个分区：

```
console:/ # ls /dev/block/mmcblk0*
/dev/block/mmcblk0    /dev/block/mmcblk0p14 /dev/block/mmcblk0p4
/dev/block/mmcblk0p1  /dev/block/mmcblk0p15 /dev/block/mmcblk0p5
/dev/block/mmcblk0p10 /dev/block/mmcblk0p16 /dev/block/mmcblk0p6
/dev/block/mmcblk0p11 /dev/block/mmcblk0p17 /dev/block/mmcblk0p7
/dev/block/mmcblk0p12 /dev/block/mmcblk0p2  /dev/block/mmcblk0p8
/dev/block/mmcblk0p13 /dev/block/mmcblk0p3  /dev/block/mmcblk0p9
console:/ #
```

使用df -h命令可以看到 32 GB的TF卡烧录完安卓系统后大概还有 26 GB的空间可以用使用（17 个分区并不会都挂载到安卓系统中，重点关注这些能看到的分区即可）。

```
console:/ # df -h
Filesystem              Size  Used Avail Use% Mounted on
/dev/block/dm-0         588M  586M  1.7M 100% /
tmpfs                   485M  616K  484M   1% /dev
tmpfs                   485M     0  485M   0% /apex
/dev/block/dm-1         112M  112M  404K 100% /vendor
/dev/block/dm-2         106M  106M  332K 100% /product
/dev/block/mmcblk0p17    27G  895M   26G   4% /data
/dev/block/mmcblk0p7    614M  512K  613M   1% /cache
/dev/block/mmcblk0p11    11M   72K   11M   1% /metadata
tmpfs                   485M     0  485M   0% /storage
/data/media              27G  895M   26G   4% /storage/emulated
/mnt/media_rw/0000-0000 128M  5.3M  122M   5% /storage/0000-0000
/dev/block/mmcblk0p16    16M     0   16M   0% /Reserve0
/dev/block/mmcblk0p1    128M  5.3M  122M   5% /mnt
console:/ #
```

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

切记不要插入电压输出大于 5V 的电源适配器，会烧坏开发板。

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

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；

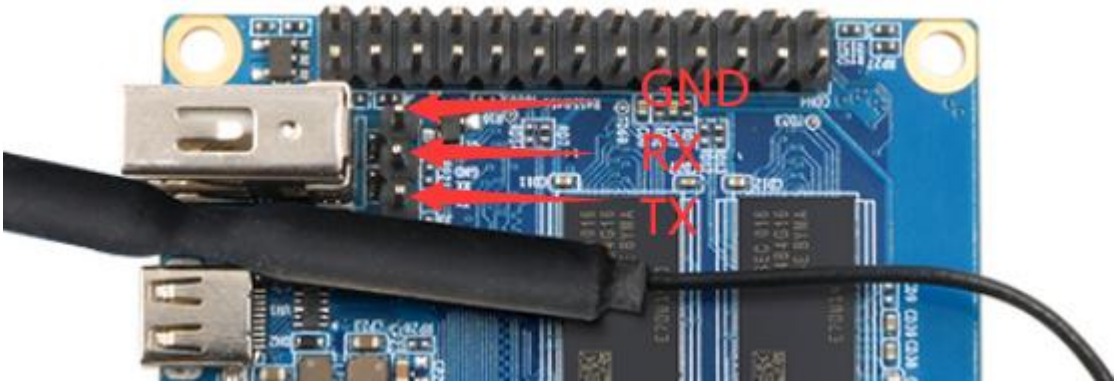
5) 如果以上都确定没问题后，请保存下系统启动过程中调试串口的输出 log（最好是以 txt 文本的形式，而不是拍照），然后向客服反馈问题。

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

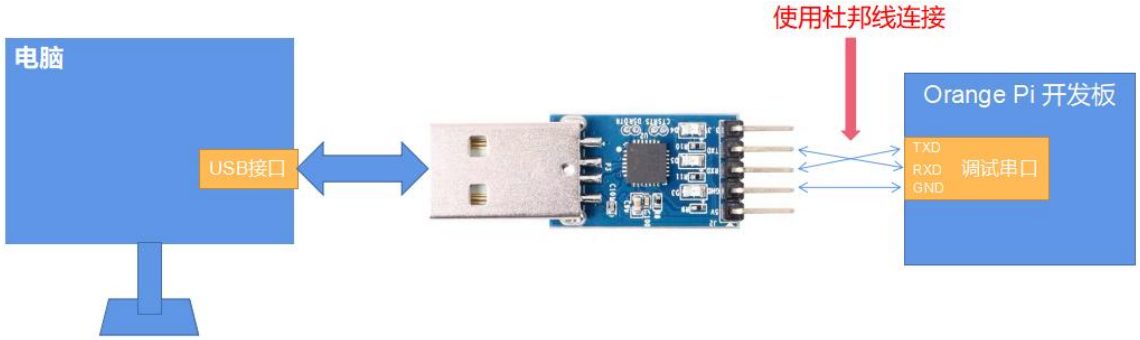
2) 开发板的调试串口 GND、TX 和 RX 引脚的对应关系如下图所示



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，如果不想仔细区分 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

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

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

```
test@test:~$ ls /dev/ttyUSB*
/dev/ttyUSB0
```

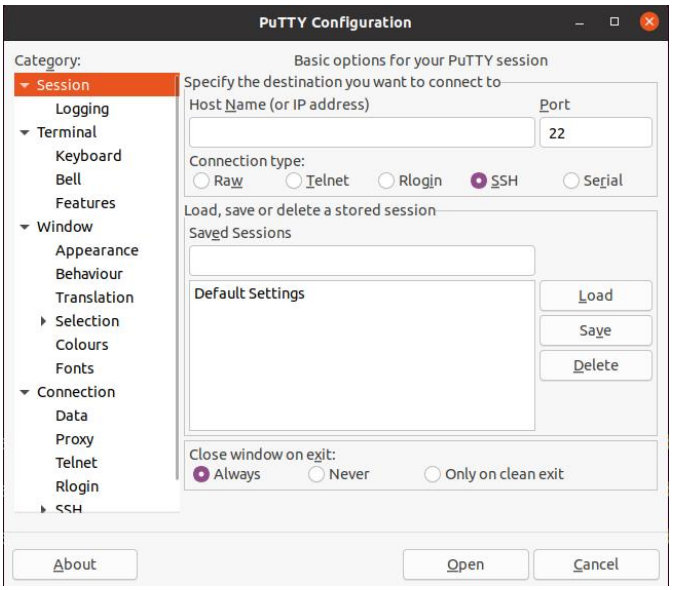
2) 然后使用下面的命令在 Ubuntu PC 上安装下 putty

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

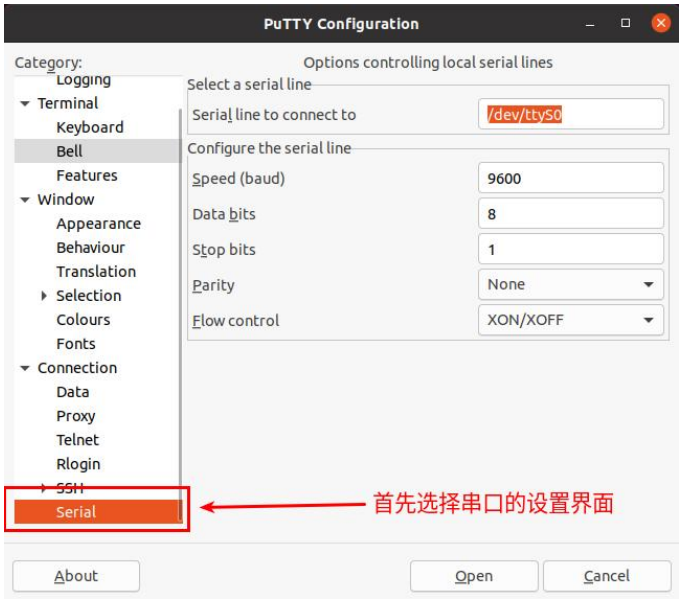
3) 然后运行 putty，**记得加 sudo 权限**

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

4) 执行 putty 命令后会弹出下面的界面



5) 首先选择串口的设置界面

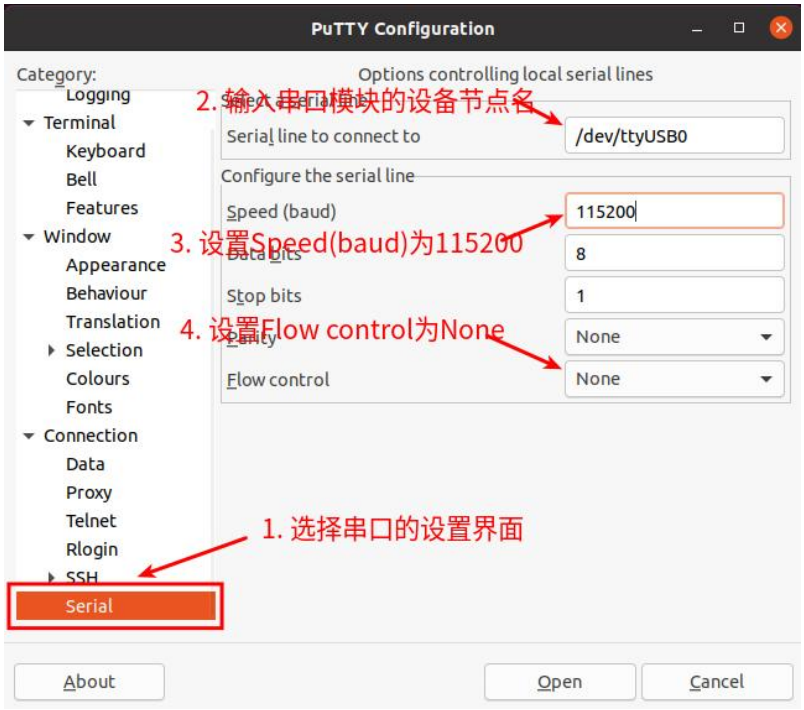


6) 然后设置串口的参数

a. 设置 **Serial line to connect to** 为**/dev/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）

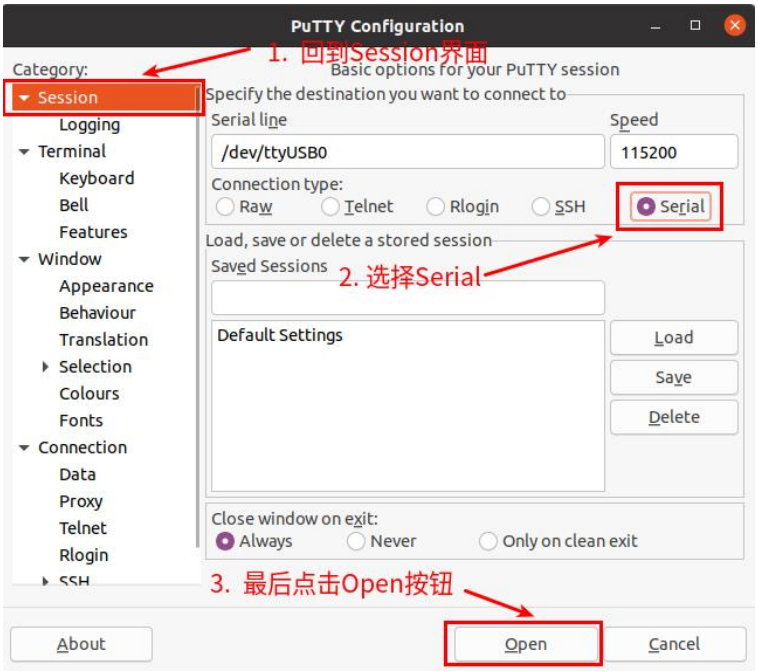
c. 设置 **Flow control** 为 **None**



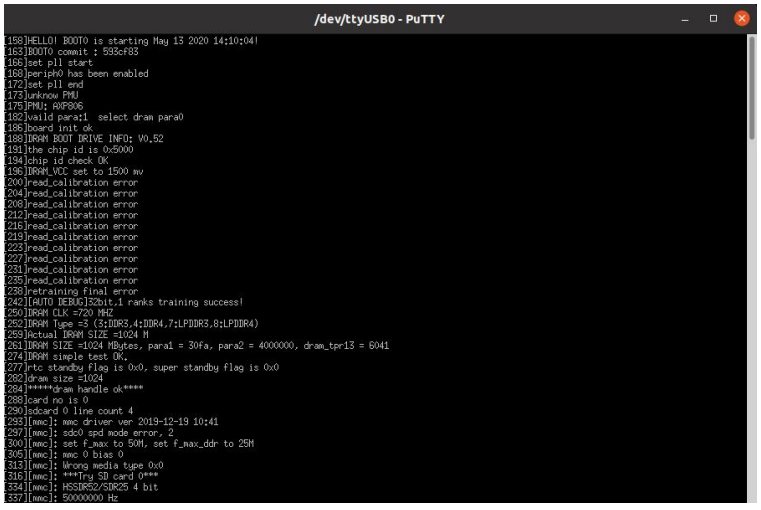
7) 在串口的设置界面设置完后，再回到 Session 界面

a. 首先选择 **Connection type** 为 **Serial**

b. 然后点击 **Open** 按钮连接串口



8) 然后启动开发板，就能从打开的串口终端中看到系统输出的 Log 信息了



2.7.3. Windows 平台调试串口的使用方法

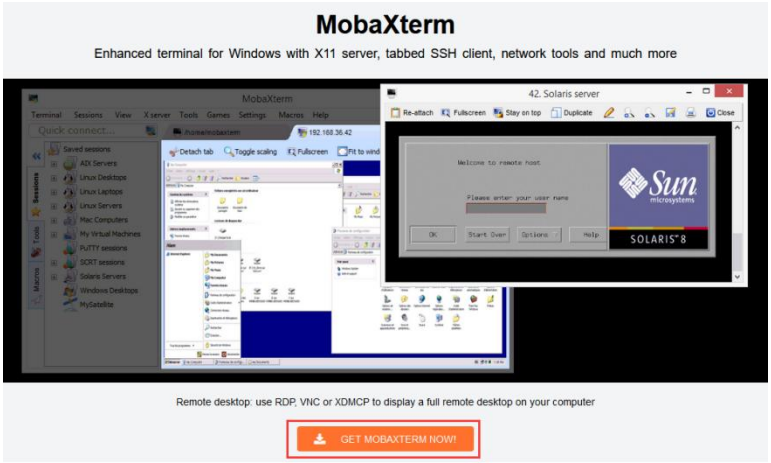
Windows 下可以使用的串口调试软件有很多，如 SecureCRT、MobaXterm 等，下面演示 MobaXterm 的使用方法，这款软件有免费版本，无需购买序列号即可使用。

1) 下载 MobaXterm

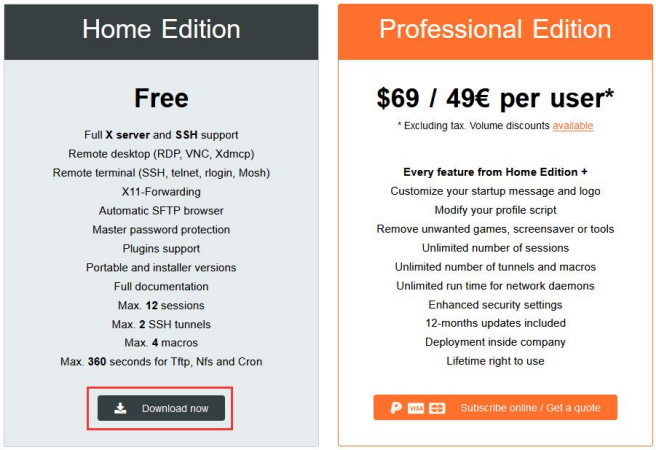
a. 下载 MobaXterm 网址如下

https://mobaxterm.mobatek.net/

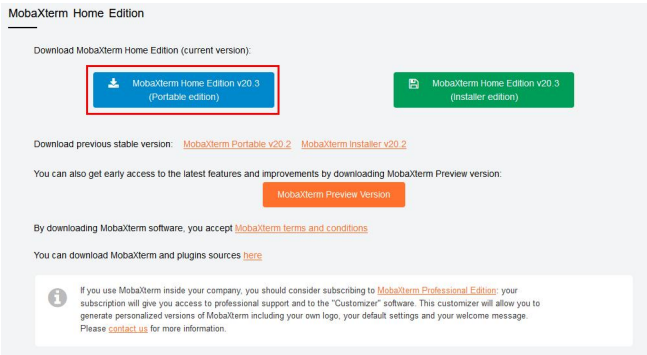
b. 进入 MobaXterm 下载网页后点击 **GET XOBATERM NOW!**



c. 然后选择下载 Home 版本



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

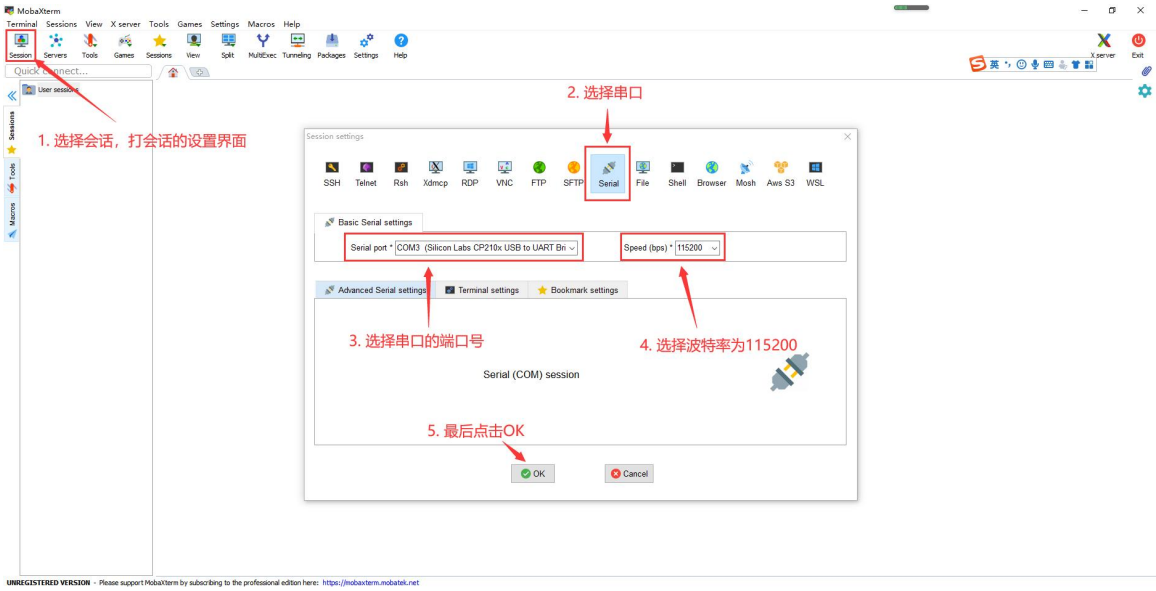


2) 下载完后使用解压缩软件解压下载的压缩包，即可得到 MobaXterm 的可执软件，

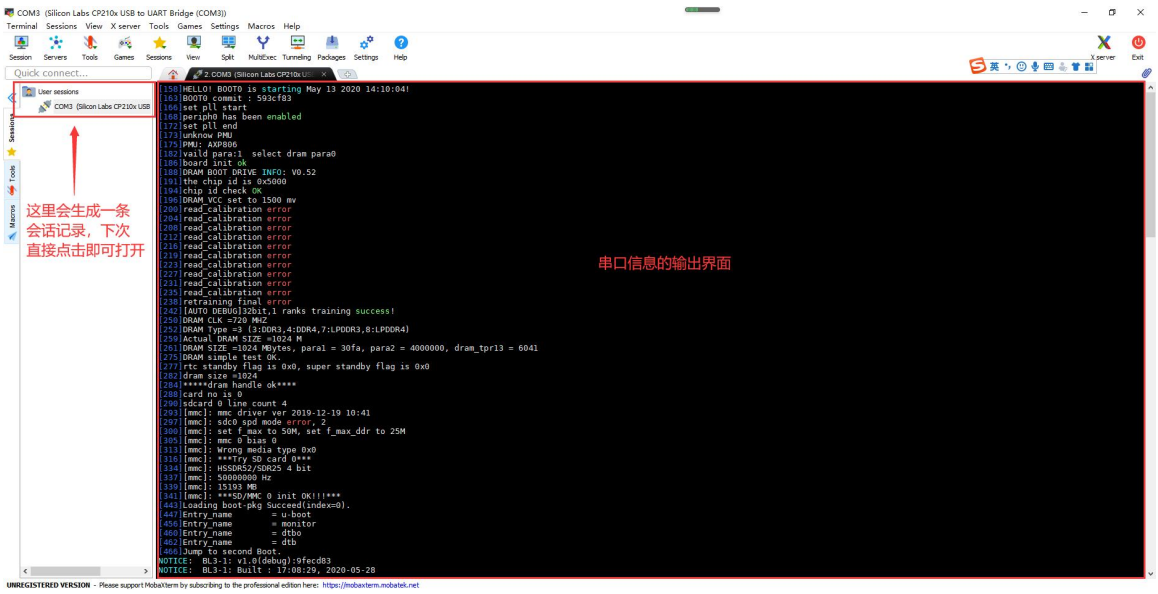
然后双击打开

3) 打开软件后，设置串口连接的步骤如下

a. 打开会话的设置界面
b. 选择串口类型
c. 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
d. 选择串口的波特率为 **115200**
e. 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


这里会生成一条
会话记录，下次
直接点击即可打开
串口信息的输出界面

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。

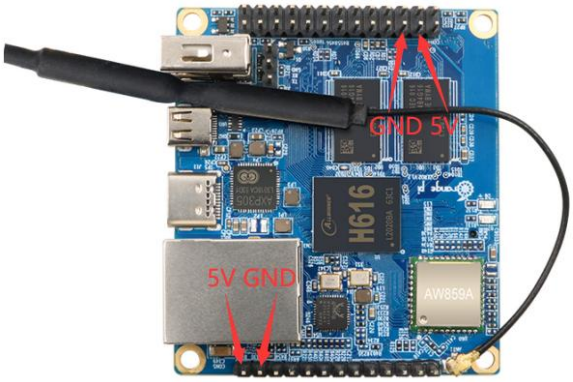
2) 使用 26pin 或者 13pin 接口中的 5V 引脚来给开发板供电，电源线的接法如下所示

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

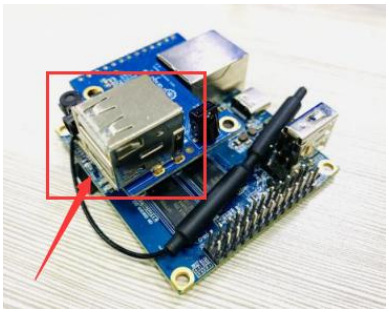
c. 黑色的杜邦线需要插到 26pin 或者 13pin 接口的 GND 引脚上

d. 26pin 和 13pin 接口 5V 引脚和 GND 引脚在开发板中的位置如下图所示，切记不要接反了



2.9. 使用开发板 13pin 接口扩展 USB 接口的方法

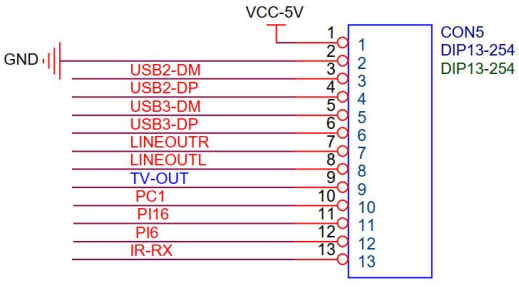
1) 如果有购买 Orange Pi 的 13pin 扩展板，将扩展板插入开发板的 13pin 接口中，就可以扩展 2 个 USB 接口

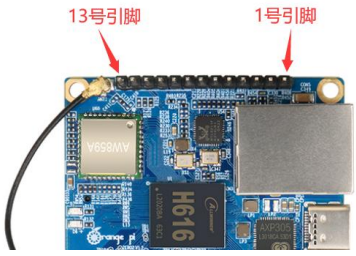


2) 如果没有 13pin 扩展板，可以使用 4pin 2.54mm 杜邦转 USB2.0 母头的线来扩展 USB 接口，具体方法如下所示：

a. 首先需要准备一根 4pin 2.54mm 杜邦转 USB2.0 母头的线（这种线在淘宝可以买到，请自行搜索购买），如下图所示：

b. 13pin 接口的原理图如下所示

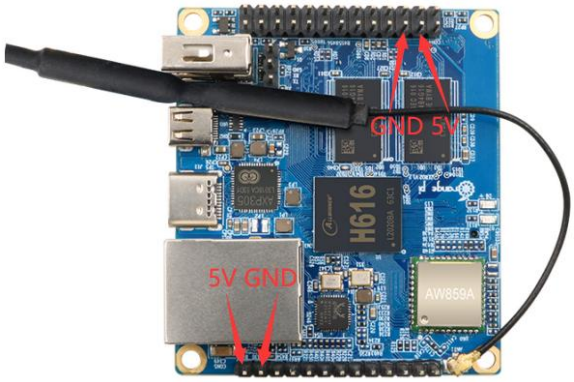




c. USB2 的接线如下所示

d. USB3 的接线如下所示

e. 如果需要在 13pin 接口上同时接两个 USB 设备，会发现 13pin 接口上的 5V 和 GND 引脚不够用，此时其中一个 USB 设备可以使用 26pin 接口中的 5V 和 GND 引脚，位置如下图所示：



3. Debian 和 Ubuntu 系统使用说明

Ubuntu 镜像和 Debian 镜像一般统称为 Linux 镜像（它们使用的都是 Linux 内核），所以当在手册中看到 Linux 镜像或者 Linux 系统时，指的就是 Ubuntu 或者 Debian 这样的镜像或者系统。

很多人都会有疑问能不能用纯 Ubuntu 或者纯 Debian 的系统（这里的纯可以理解为从 Ubuntu 或者 Debian 官网下载的系统）。答案是不行的，因为 Ubuntu 和 Debian 并没有提供针对 Orange Pi 的开发板适配的系统。

我们从 Ubuntu 和 Debian 的官网可以看到它们都是支持 arm64 架构的(开发板的 SOC 就是 arm64 架构），但是请注意这里说的支持指的仅仅是 Ubuntu 或者 Debian 提供了 arm64 版本的软件仓库(包含几万个软件包)或者说是 rootfs(Orange Pi 制作 Ubuntu 或者 Debian 系统时使用的正是这些软件包）。而制作一个针对某个开发板可以使用的 Ubuntu 或者 Debian 系统还需要移植 U-boot 和 Linux 内核等东西，并且还要修复遇到的 BUG，优化部分功能，这些都是 Orange Pi 来完成的。

由于 Orange Pi 只维护 Ubuntu 和 Debian 系统，所以 CentOS、Kali 或者 OpenWRT 等这些 Linux 发行版如果没有其他开发者移植或者自己移植适配的话，在 Orange Pi 的开发板上就是无法使用的（硬件跑这些系统是没问题的）。

另外，还有人经常会问其他开发板的系统能不能在 Orange Pi 开发板上使用。答案是不行的，因为不同的开发板使用的芯片，电路连接一般都是不同的。针对某款开发板开发的系统基本是无法在其他开发板上使用的。

3.1. 已支持的 linux 镜像类型和内核版本

Linux 镜像类型	内核版本	服务器版	桌面版
Ubuntu18.04 - Bionic	Linux4.9	支持	支持
Ubuntu 20.04 - Focal	Linux4.9	支持	支持
Debian 10 - Buster	Linux4.9	支持	支持
Ubuntu 20.04 - Focal	Linux5.16	支持	支持
Ubuntu 22.04 - Jammy	Linux5.16	支持	支持
Debian 11 - Bullseye	Linux5.16	支持	支持
Debian 12 - Bookworm	Linux5.16	支持	支持

注意，Debian 12 - Bookworm 是 Debian 官方正在开发中的系统，并不是目前已发布的稳定的 Debian 版本。主要供开发者体验使用。如果是做产品，考虑到稳定

性问题，请不要选择此系统。

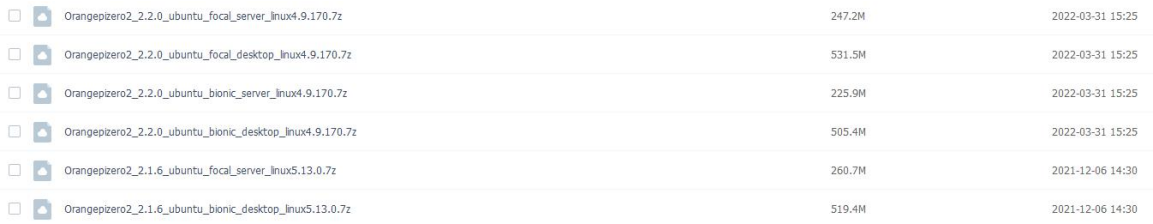
Debian 12 - Bookworm 随着 Debian 官方的开发进程，软件仓库中的软件包会持续的更新的（包括删除，新增软件包以及修复遇到的问题）。所以 Debian 12 - Bookworm 中的软件包比其他稳定版本的 Debian 系统中的软件包会要更新一些。这就意味着可以提前体验一些新的技术和功能，比如 H616 的 GPU 在 Debian12 Linux5.16 系统中就是可以使用的。

上面所有的镜像不一定都会编译出来放在百度云和谷歌网盘供大家下载使用。因为如果都放出来的话，数量会太多，很多刚接触 Linux 的同学可能会有选择困难症，另外 Ubuntu 和 Debian 一般使用其中的一个版本即可，在满足需求的情况下，没有必要每个不同的版本都去下载使用。

如果某些特殊需求（这种需求基本很少）一定要使用 Ubuntu 或者 Debian 的某个版本，但是没有现成的镜像可以下载的话，可以使用 Orange Pi 提供的源码自己编译想要的镜像，上面如果写了支持的话，那么在代码里就都是适配了的。Linux 镜像的编译方法请参考第 5 章和第 6 章的内容。

1) 在 **Orange Pi 的资料下载页面**可以看到下面的下载选项，**Ubuntu 镜像和 Debian 镜像一般统称为 Linux 镜像**

a. 点击 Ubuntu 镜像下载链接可以下载 Ubuntu 相关的镜像，比如打开百度网盘后可以看到下面几种 Ubuntu 镜像（**镜像版本号可能会更新**）



b. 点击 Debian 镜像下载链接可以下载 Debian 相关的镜像，比如打开百度网盘后可以看到下面几种 Debian 镜像（**镜像版本号可能会更新**）

2) Linux 镜像的命名规则

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

a. **开发板的型号**：都是 **Orangepizero2**。不同开发板的型号名一般都是不同的，烧录镜像前，请确保所选择镜像的这个型号名和开发板是匹配的。

b. **版本号**：如 **2.x.x 或者 3.x.x**，这个版本号会随着镜像功能的更新而递增，另外开发板 Linux 镜像的版本号最后一个数字都是偶数。

c. **Linux 发行版的类型**：目前支持 **Ubuntu** 和 **Debian**。由于 Ubuntu 源自 Debian，所以两个系统在使用上来说总体区别不大。但部分软件的默认配置和命令的使用上还是有些许区别的，另外 Ubuntu 和 Debian 都各自有维护所支持的软件仓库，在支持的可安装的软件包上也是有些许差异的。这些需要亲自去使用体验才会有比较深刻的认识。有关更多的细节，可以参考下 Ubuntu 和 Debian 官方提供的文档。

d. **发行版代号**：用来区分 Ubuntu 或者 Debian 这样具体的 Linux 发行版的不同版本。其中 **bionic** 和 **focal** 都是 Ubuntu 发行版，bionic 表示 Ubuntu18.04，focal 表示 Ubuntu20.04，jammy 表示 Ubuntu22.04，不同版本的最大的区别是新版本的 Ubuntu 系统维护的软件仓库的中的软件很多都比旧版本的 Ubuntu 系统中的要新，比如 Python 和 GCC 编译工具链等。**buster** 是 Debian 的具体版本代号，buster 表示 Debian10，**bullseye** 表示 Debian11，Debian11 是 Debian 官方最新发布的稳定版本。**bookworm** 表示 Debian12，为 Debian 官方下一个开发中的版本。

e. **服务器或桌面**：用来表示系统是否带桌面环境，如果为 **server** 就表示系统没有安装桌面环境，镜像占用的存储空间和资源比较小，主要使用命令行来操作控制系统。如果为 **desktop** 就表示系统默认安装有 XFCE4 桌面环境，镜像占用的存储空间和资源比较大，可以接显示器和鼠标键盘通过界面来操作系统。当然 desktop 版本的系统也可以像 server 版本的系统一样通过命令行来操作。

f. **内核版本**：用来表示 linux 内核的版本号，目前支持 **linux4.9.170** 和 **linux5.16.17**。

3.2. linux 内核驱动适配情况

功能	Linux4.9	Linux5.16
HDMI 视频	OK	OK
HDMI 音频	OK	OK
USB2.0 x 3	OK	OK
TF 卡启动	OK	OK
网卡	OK	OK
红外接收	OK	OK
WIFI	OK	OK
蓝牙	OK	OK
耳机音频	OK	OK
USB 摄像头	OK	OK
LED 灯	OK	OK
26pin GPIO	OK	OK
I2C3	OK	OK
SPI1	OK	OK
UART5	OK	OK
PWM	OK	OK
温度传感器	OK	OK
硬件看门狗	OK	OK
Mali GPU	NO	OK
视频编解码	NO	NO
TV-OUT	NO	NO

3.3. 本手册 linux 命令格式说明

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

```
```

如下所示，黄色方框里内容表示需要特别注意的内容，这里面的命令除外。

```
```

2) 命令前面的提示符类型说明

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

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

b. **root@orangepi:~$** 提示符表示这个命令是在**开发板的 linux 系统**中输入的，提示符最后的 **$** 表示系统当前用户为普通用户，当执行特权命令时，需要加上 **sudo**

c. **root@orangepi:~#** 提示符表示这个命令是在**开发板的 linux 系统**中输入的，提示符最后的 **#** 表示系统当前用户为 root 用户，可以执行任何想要执行的命令

d. **test@test:~$** 提示符表示这个命令是在 Ubuntu PC 或者 Ubuntu 虚拟机中输入的，而不是开发板的 linux 系统中。提示符最后的 **$** 表示系统当前用户为普通用户，当执行特权命令时，需要加上 **sudo**

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

3) 哪些是需要输入的命令？

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

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

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

```
orangepi@orangepi:~$ echo  \
"deb [arch=$(dpkg --print-architecture)  \
signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]  \
```

```
https://download.docker.com/linux/debian  \
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

3.4. linux 系统登录说明

3.4.1. linux 系统默认登录账号和密码

账号	密码
root	orangepi
orangepi	orangepi

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

当输入密码提示错误，或者 ssh 连接有问题，请注意，只要使用的是 Orange Pi 提供的 Linux 镜像，就请不要怀疑上面的密码不对，而是要找其它的原因。

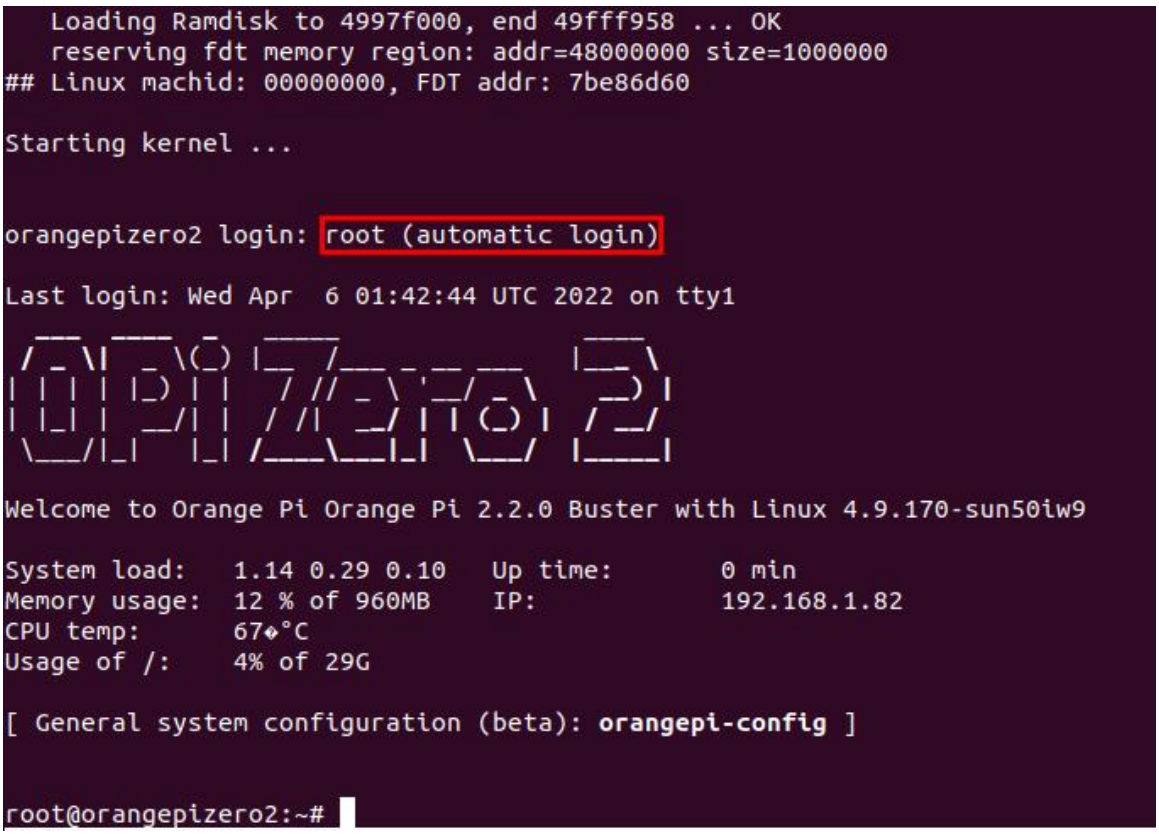
3.4.2. 设置 linux 系统终端自动登录的方法

1) root 用户自动登录终端的方法

a. 首先输入下面的命令创建终端自动登录的配置文件

```
root@orangepi:~# mkdir -p /etc/systemd/system/getty@.service.d/
root@orangepi:~# mkdir -p /etc/systemd/system/serial-getty@.service.d/
root@orangepi:~# cat <<-EOF >  \
/etc/systemd/system/serial-getty@.service.d/override.conf
[Service]
ExecStartPre=/bin/sh -c 'exec /bin/sleep 10'
ExecStart=
ExecStart=-/sbin/agetty --noissue --autologin root %I \$TERM
Type=idle
EOF
root@orangepi:~# cp /etc/systemd/system/serial-getty@.service.d/override.conf  \
/etc/systemd/system/getty@.service.d/override.conf
```

b. 然后重启系统就能看到终端会自动登录了（无需输入账号和密码），使用的用户为 root



2) orangepi 用户自动登录终端的方法

a. 首先输入下面的命令创建终端自动登录的配置文件

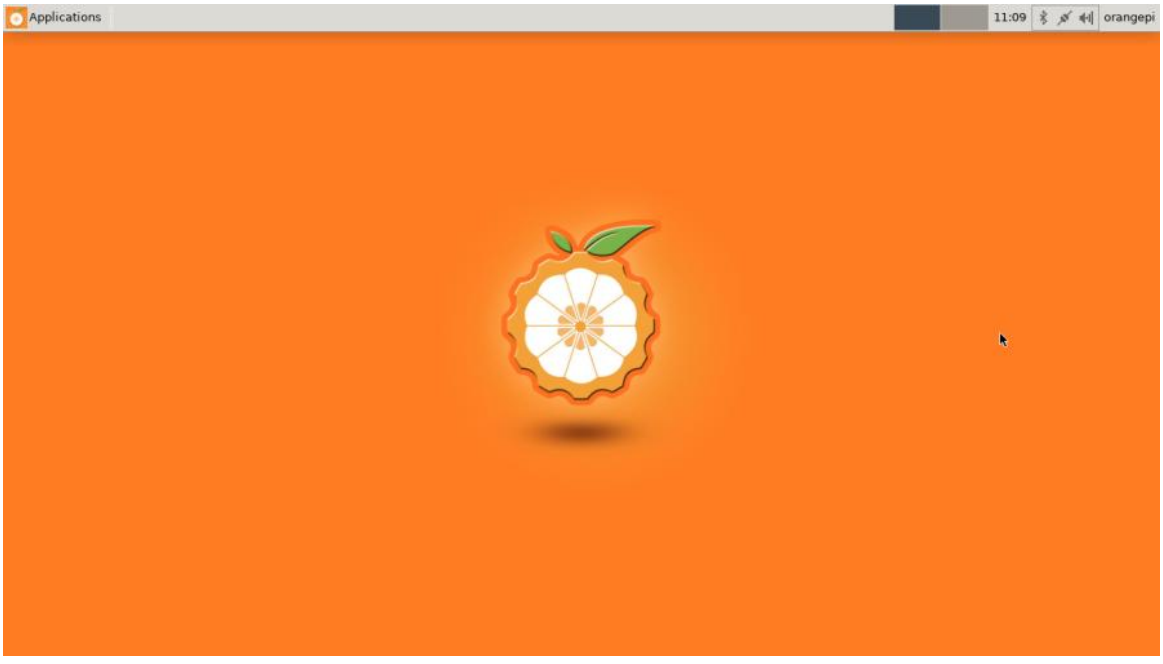
```
root@orangepi:~# mkdir -p /etc/systemd/system/getty@.service.d/
root@orangepi:~# mkdir -p /etc/systemd/system/serial-getty@.service.d/
root@orangepi:~# cat <<-EOF >    \
/etc/systemd/system/serial-getty@.service.d/override.conf
[Service]
ExecStartPre=/bin/sh -c 'exec /bin/sleep 10'
ExecStart=
ExecStart=-/sbin/agetty --noissue --autologin orangepi %I \$TERM
Type=idle
EOF
root@orangepi:~# cp /etc/systemd/system/serial-getty@.service.d/override.conf    \
/etc/systemd/system/getty@.service.d/override.conf
```

b. 然后重启系统就能看到终端会自动登录了（无需输入账号和密码），使用的用户为 **orangepi**

```
   Loading Ramdisk to 4997f000, end 49fff958 ... OK
   reserving fdt memory region: addr=48000000 size=1000000
## Linux machid: 00000000, FDT addr: 7be86d60

Starting kernel ...


orangepizero2 login: orangepi (automatic login)

Last login: Wed Apr  6 01:46:02 UTC 2022 from 192.168.1.179 on pts/0
  ___  ____  _   _____                  ____
 / _ \|  _ \(_) |__  /___ _ __ ___     |___ \
| | | | |_) | |   / // _ \ '__/ _ \      __) |
| |_| |  __/| |  / /|  __/ | | (_) |    / __/
 \___/|_|   |_| /____\___|_|  \___/    |_____|

Welcome to Orange Pi Orange Pi 2.2.0 Buster with Linux 4.9.170-sun50iw9

System load:   0.80 0.19 0.06   Up time:       0 min
Memory usage:  12 % of 960MB    IP:           192.168.1.82
CPU temp:      68°C
Usage of /:    4% of 29G

[ General system configuration (beta): orangepi-config ]


orangepi@orangepizero2:~$
```

3.4.3. linux 桌面版系统自动登录说明

1) 桌面版系统启动后会自动登录进入桌面，无需输入密码



2) 修改**/etc/lightdm/lightdm.conf.d/22-orangepi-autologin.conf**中的配置可以禁止桌

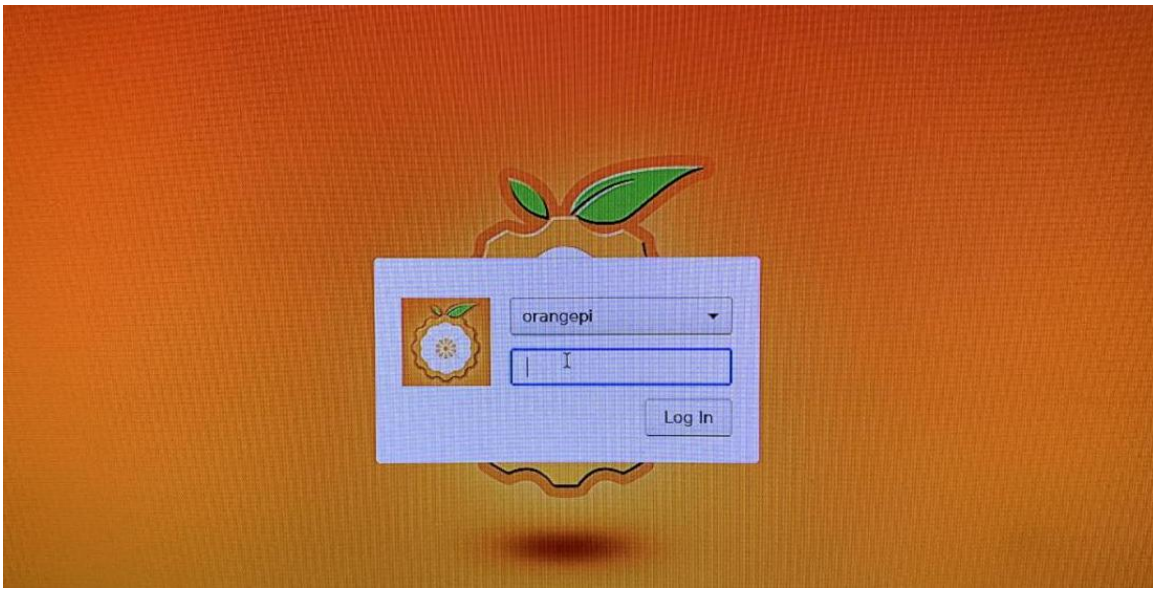
面版系统自动登录桌面，修改命令如下所示，也可以打开配置文件直接修改

```
orangepi@orangepi:~$ sudo sed -i   \
"s/autologin-user=orangepi/#autologin-user=orangepi/"   \
/etc/lightdm/lightdm.conf.d/22-orangepi-autologin.conf
```

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

```
orangepi@orangepi:~$ cat /etc/lightdm/lightdm.conf.d/22-orangepi-autologin.conf
[Seat:*]
#autologin-user=orangepi
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. 4. 4. Linux 桌面版系统 root 用户自动登录的设置方法

1) 首先在 **22-orangepi-autologin.conf** 中设置自动登录的用户为 **root**

```
orangepi@orangepi:~$ sudo vim /etc/lightdm/lightdm.conf.d/22-orangepi-autologin.conf
[Seat:*]
autologin-user=root
autologin-user-timeout=0
user-session=xfce
```

2) 然后修改下 **lightdm-autologin** 配置文件，将 root 修改为 anything，去掉 root 用户的限制

注意，Ubuntu22.04 不需要设置这一步。

```
orangepi@orangepi:~$ sudo vim /etc/pam.d/lightdm-autologin
# Allow access without authentication
#auth          required pam_succeed_if.so user != root quiet_success
auth          required pam_succeed_if.so user != anything quiet_success
```

3) 然后重启系统，就会自动使用 root 用户登录桌面了

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

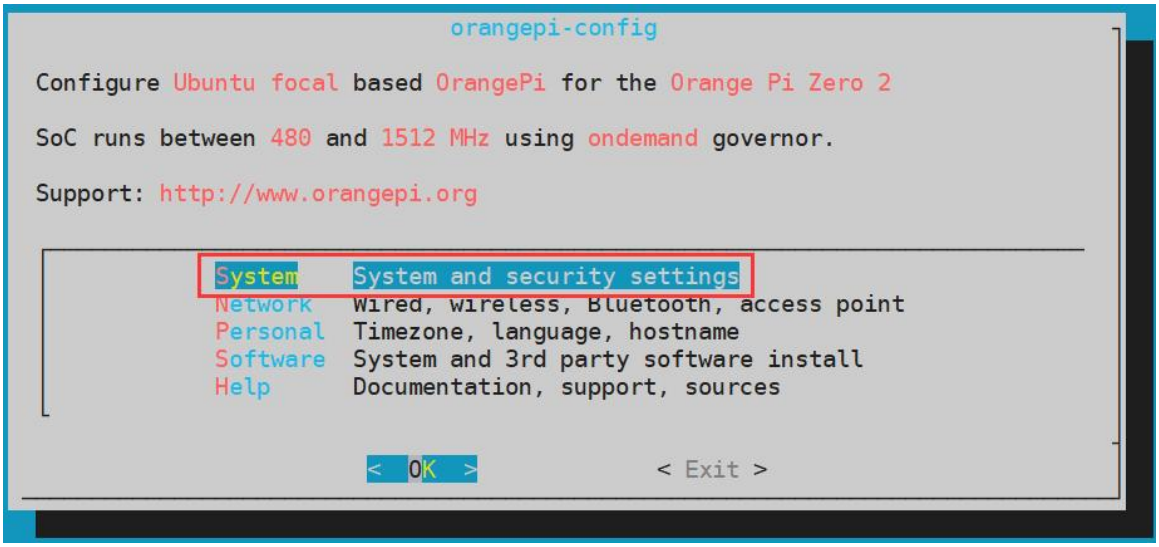
另外请注意这并不是一个 bug，因为 pulseaudio 本来就不允许在 root 用户下运行。

3.4.5. Linux 桌面版系统禁用桌面的方法

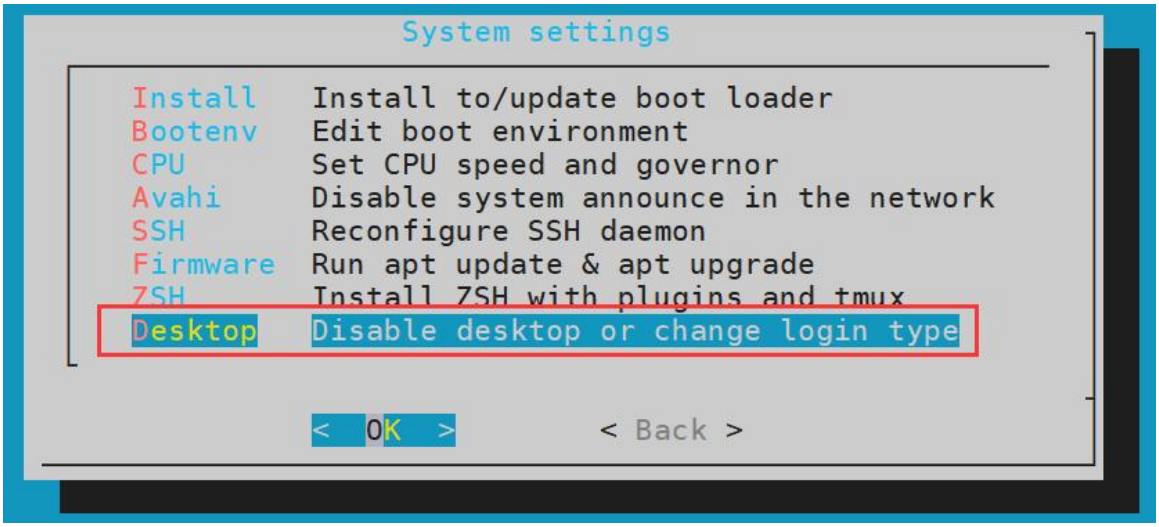
1) 首先在命令行中输入下面的命令，**请记得加 sudo 权限**

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

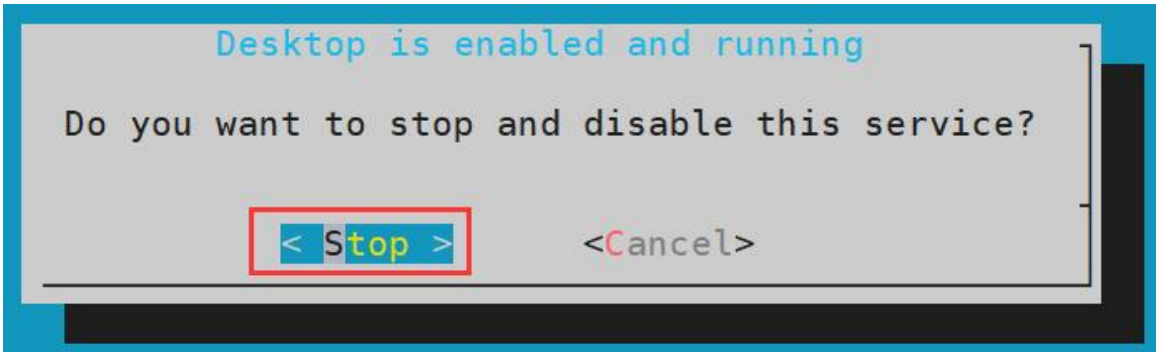
2) 然后选择 **System**



3) 然后选择 **Desktop**



4) 然后选择**<Stop>**



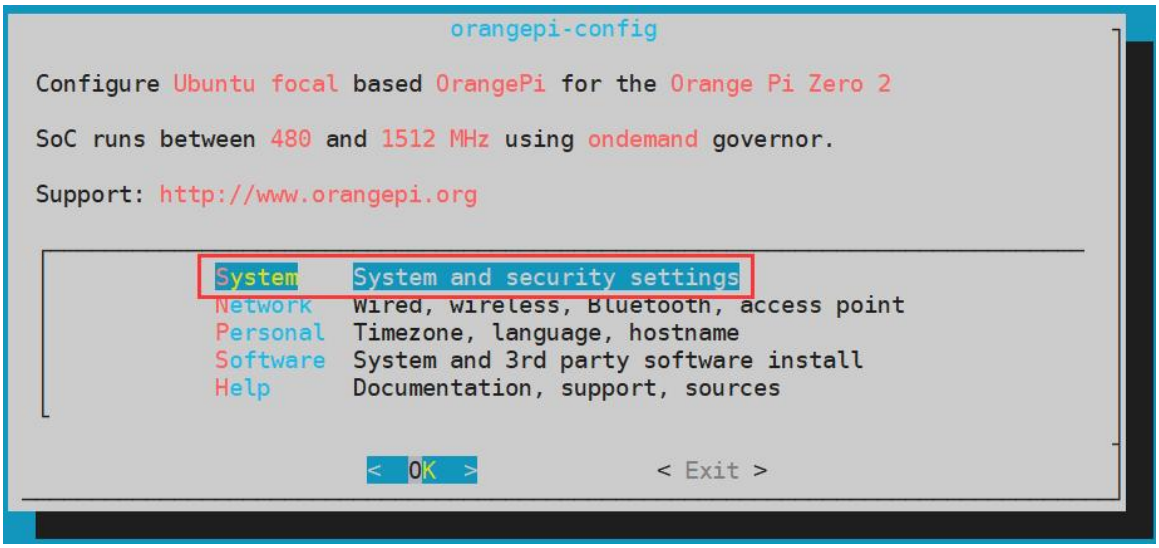
5) **然后重启 Linux 系统就会发现不会显示桌面了**

6) 重新打开桌面的步骤如下所示：

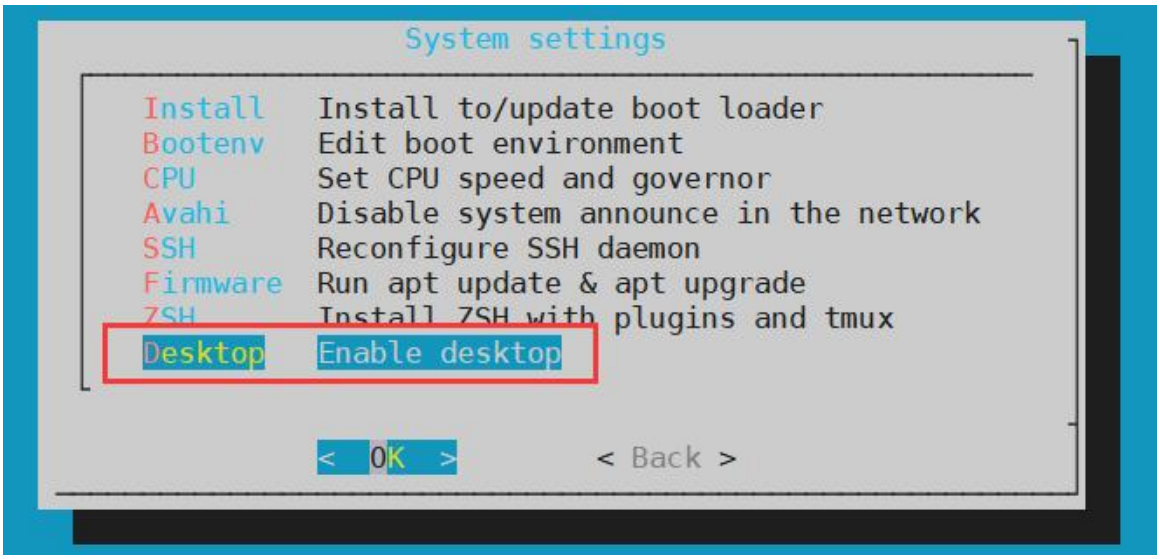
a. 首先在命令行中输入下面的命令，**请记得加 sudo 权限**

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

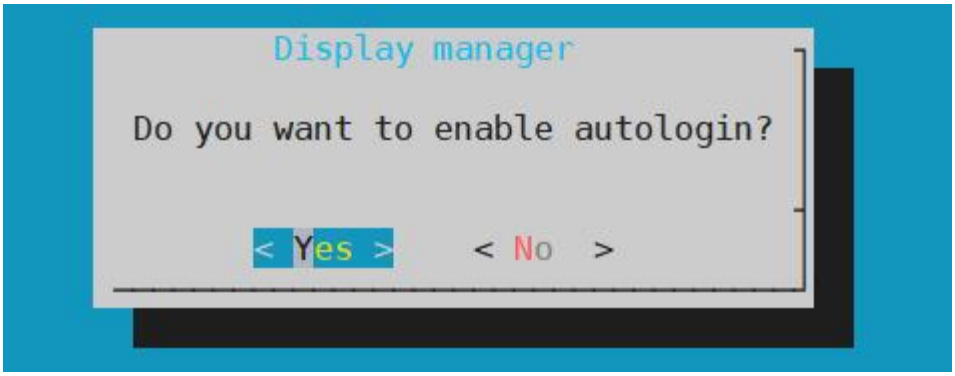
b. 然后选择 **System**



c. 然后选择 **Desktop Enable desktop**



d. 然后选择是否需要自动登录桌面，如果选择**<Yes>**就会自动登录进入桌面，如果选择**<No>**就会显示用户和密码的输入界面，需要输入密码才能进入桌面



e. 选择完后 HDMI 显示器就会显示桌面了

3.5. 板载 LED 灯测试说明

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

	绿灯	红灯
u-boot 启动阶段	灭	亮
内核启动到进入系统	亮	灭
GPIO 口	PC13	PC12

注意，下面的操作请在 root 用户下进行。

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

b. 设置绿灯熄灭的命令如下

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

c. 设置绿灯常亮的命令如下

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

d. 设置绿灯闪烁的命令如下

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

e. 设置绿灯停止闪烁的命令如下

```
root@orangepi:/sys/class/leds/orangepi: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
```

3.6. TF 卡中 linux 系统 rootfs 分区容量操作说明

3.6.1. 第一次启动会自动扩容 TF 卡中 rootfs 分区的容量

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

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

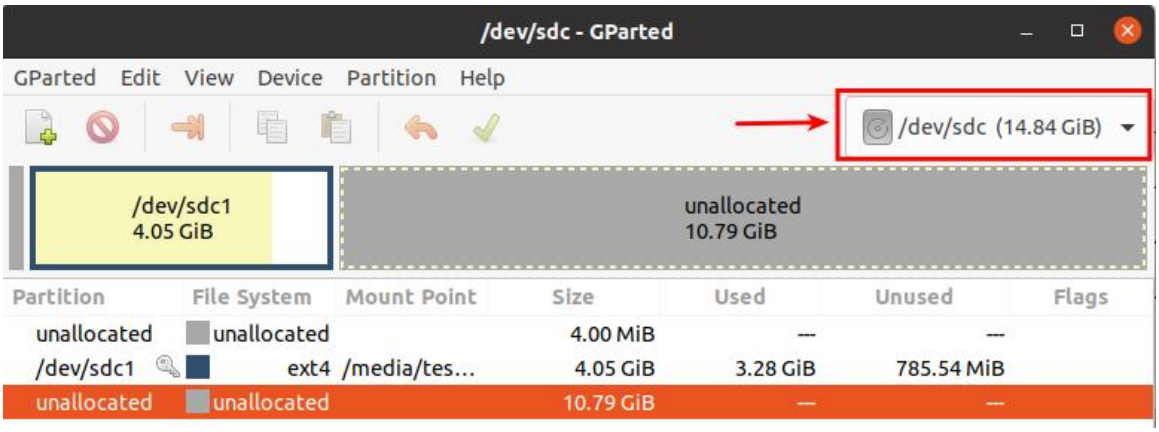
a. 首先在 Ubuntu 电脑中安装下 gparted 这个软件

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

b. 然后打开 gparted

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

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



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

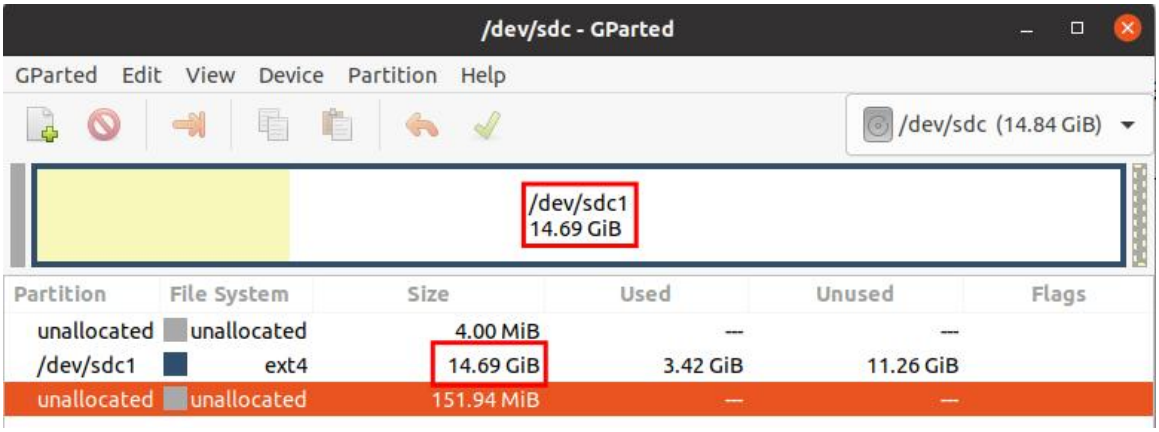
2) 然后可以将烧录好 Linux 系统的 TF 卡插入开发板中启动，TF 卡第一次启动 linux 系统时会通过 **orangepi-resize-filesystem.service** 这个 systemd 服务来调用 **orangepi-resize-filesystem** 脚本自动进行 rootfs 分区的扩容，所以**无需再手动扩容**

3) 登录系统后可以通过 **df -h** 命令来查看 rootfs 的大小，如果和 TF 卡的实际容量一

致，说明自动扩容运行正确

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

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



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

3.6.2. 禁止自动扩容 TF 卡中 rootfs 分区容量的方法

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

2) 然后 Ubuntu 电脑一般会自动挂载 TF 卡的分区，如果自动挂载正常，使用 ls 命令可以看到下面的输出，TF 卡的分区名和下面命令所示名字不一定相同

```
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
```

3) 然后在 Ubuntu 电脑中将当前用户切换成 root 用户

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

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

```
root@test:~# cd /media/test/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
```

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

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

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

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

注意，请在 root 用户下执行下面的命令。

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

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

```
root@orangepi:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
```

```
udev            925M     0  925M   0% /dev
tmpfs           199M  3.2M  196M   2% /run
/dev/mmcblk0p1   15G  3.2G   12G  23% /
```

3.6.3. 手动扩容 TF 卡中 rootfs 分区容量的方法

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

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

2) 然后 Ubuntu 电脑一般会自动挂载 TF 卡的分区，如果自动挂载正常，使用 **ls** 命令可以看到下面的输出，TF 卡的分区名和下面命令所示名字不一定相同

```
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
```

3) 然后在 Ubuntu 电脑中将当前用户切换成 root 用户

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

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

```
root@test:~# cd /media/test/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
```

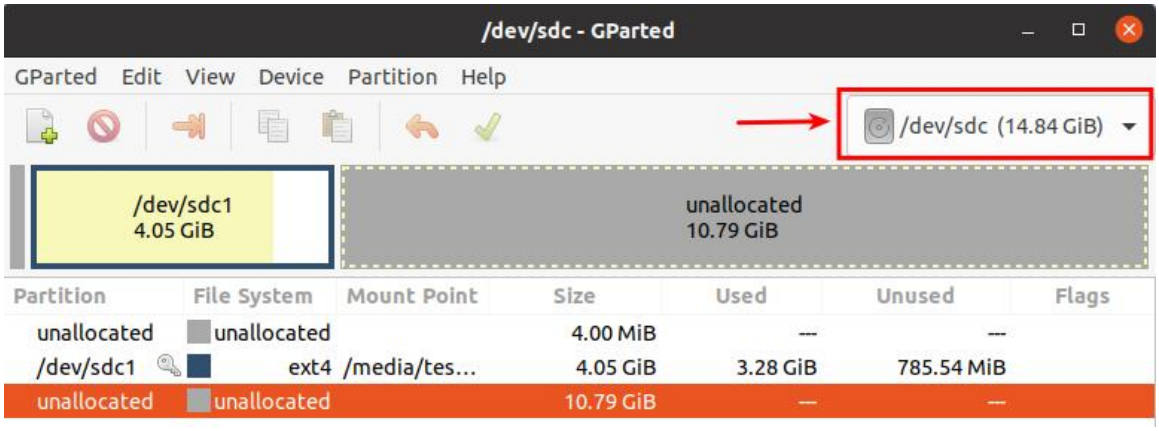
5) 然后在 Ubuntu 电脑中安装下 gparted 这个软件

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

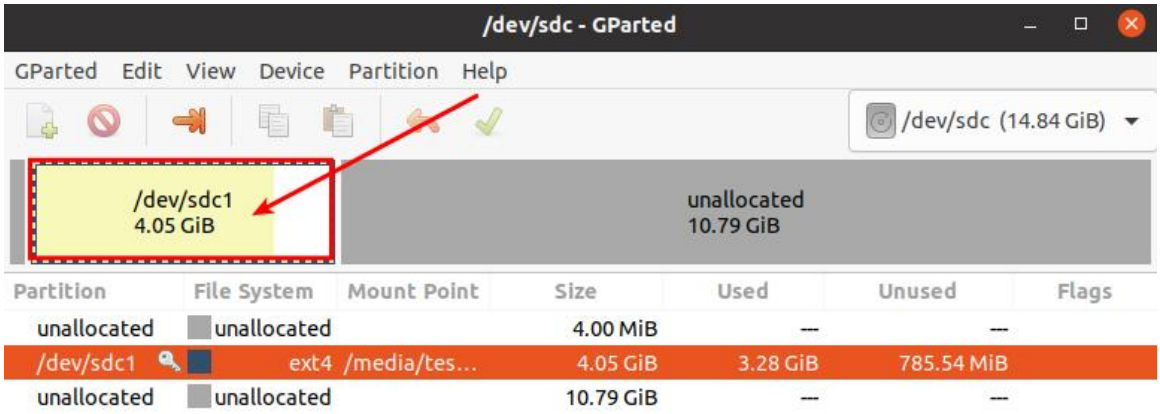
6) 然后打开 gparted

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

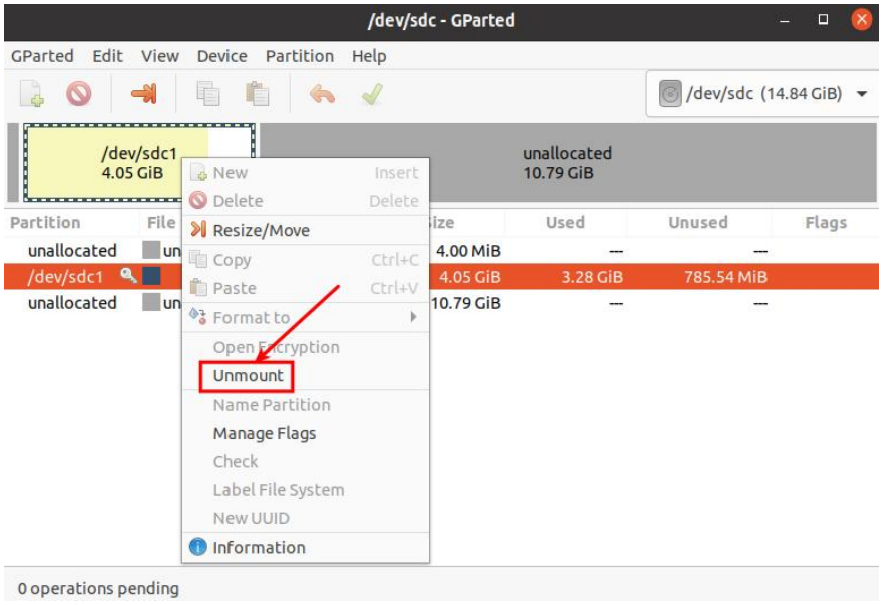
7) 打开 gparted 后在右上角可以选择 TF 卡，然后就可以看到 TF 卡容量的使用情况。下图显示的是烧录完 Linux 桌面版系统后 TF 卡的情况，可以看到，虽然 TF 卡的总容量是 16GB 的（在 GParted 中显示为 14.84GiB），但是 rootfs 分区（/dev/sdc1）实际只分配了 4.05GiB，还剩下 10.79GiB 未分配



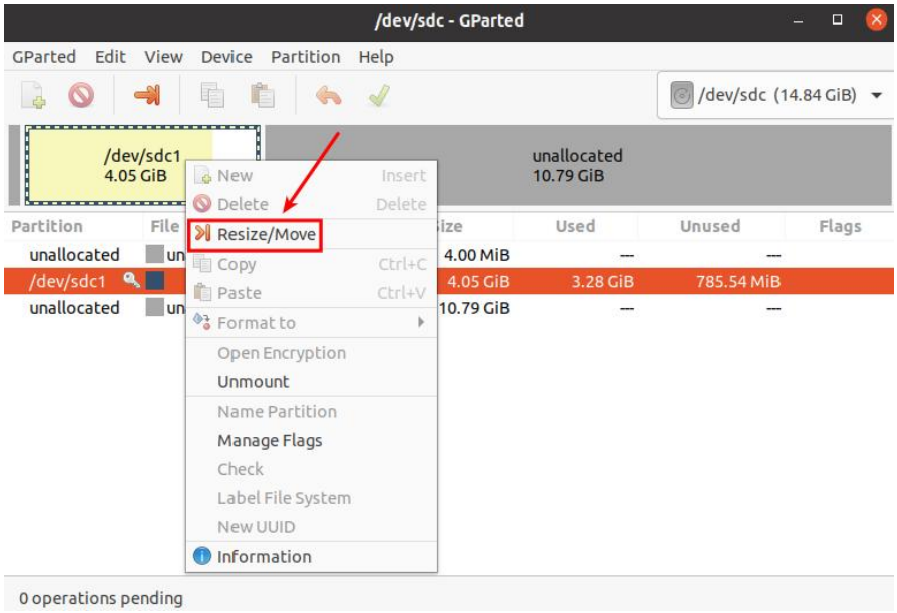
8) 然后选中 rootfs 分区（/dev/sdc1）



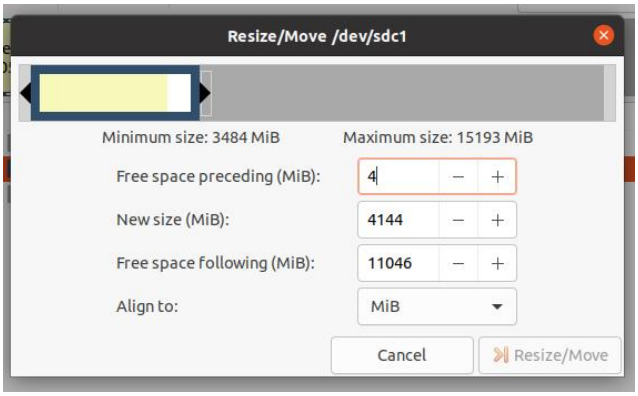
9) 再点击鼠标右键就可以看到下图所示的操作选项，如果 TF 卡已经挂载了，首先需要 Umount 掉 TF 卡的 rootfs 分区



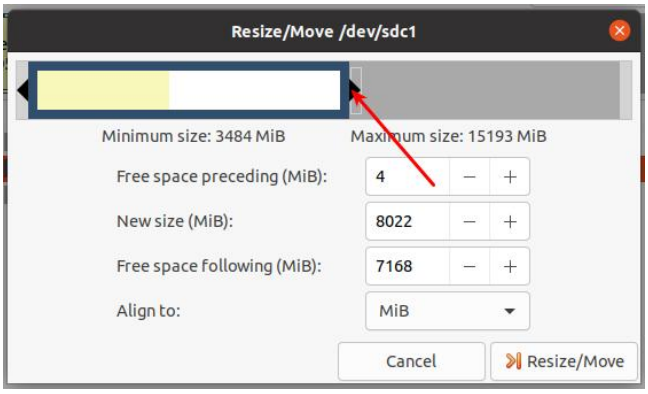
10) 然后再次选中 rootfs 分区，再点击鼠标右键，然后选择 **Resize/Move** 开始扩容 rootfs 分区的大小



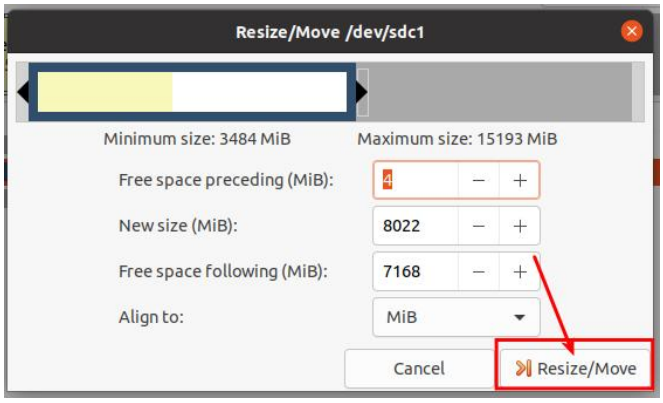
11) **Resize/Move** 选项打开后会弹出下面的设置界面



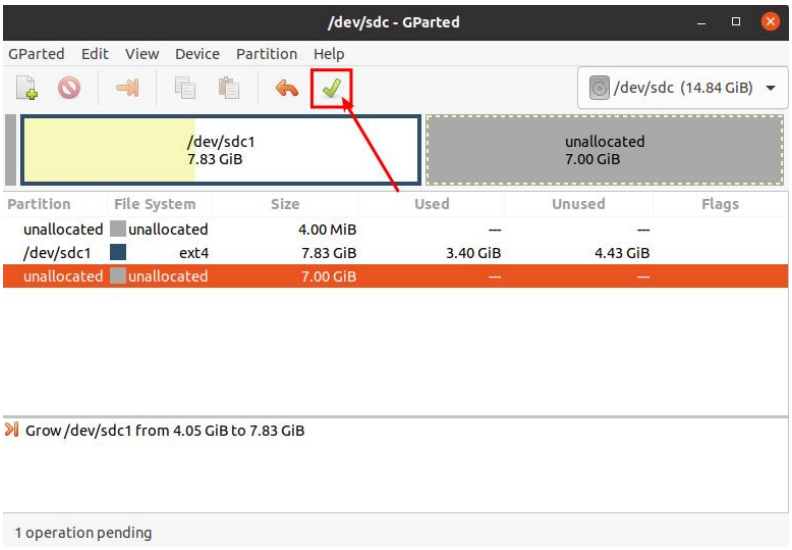
12) 然后可以直接拖动下图所示的位置来设置容量的大小，也可以通过设置 **New sieze(MiB)**中的数字来设置 rootfs 分区的大小



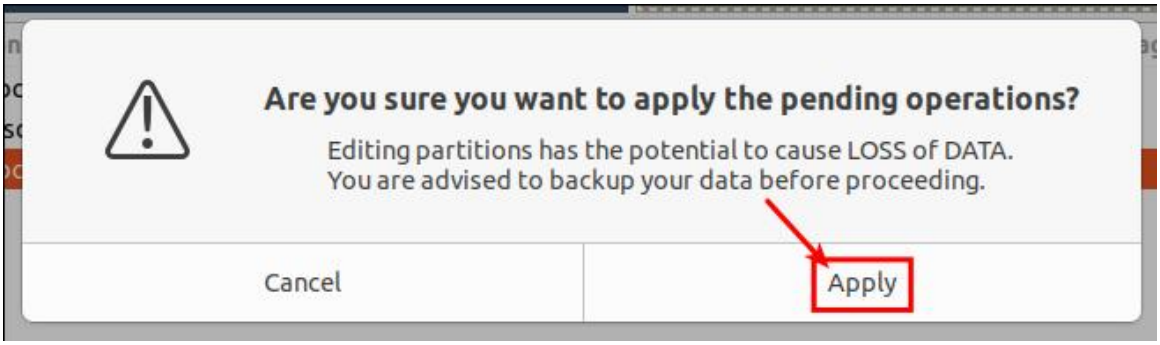
13) 设置好容量后，再点击右下角的 **Resize/Move** 即可



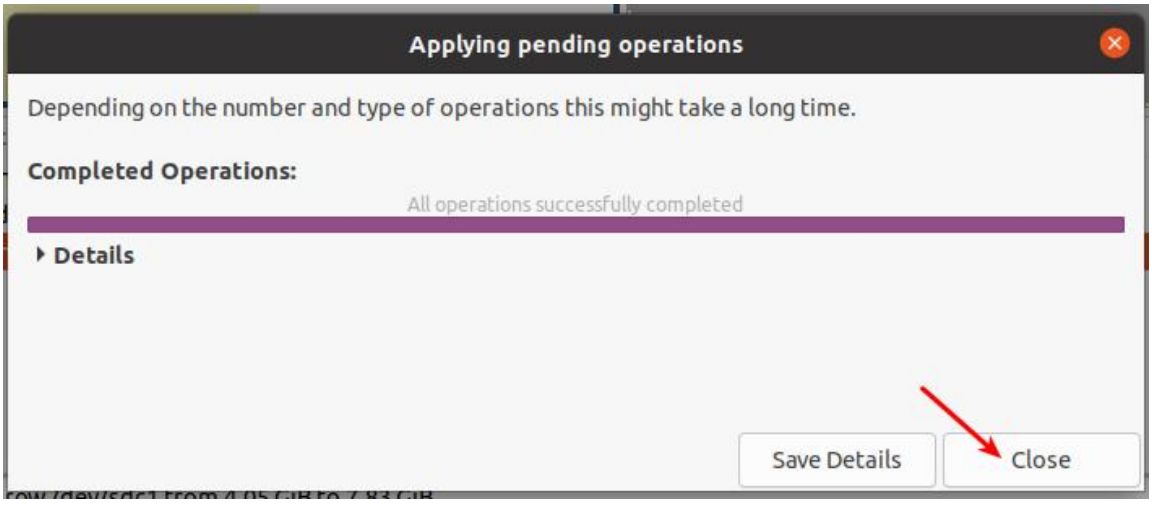
14) 最后确认无误后，再点击下图所示的**绿色** √



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



16) 扩容完成后点击 **Close** 关闭即可



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

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

3.6.4. 缩小 TF 卡中 rootfs 分区容量的方法

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

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

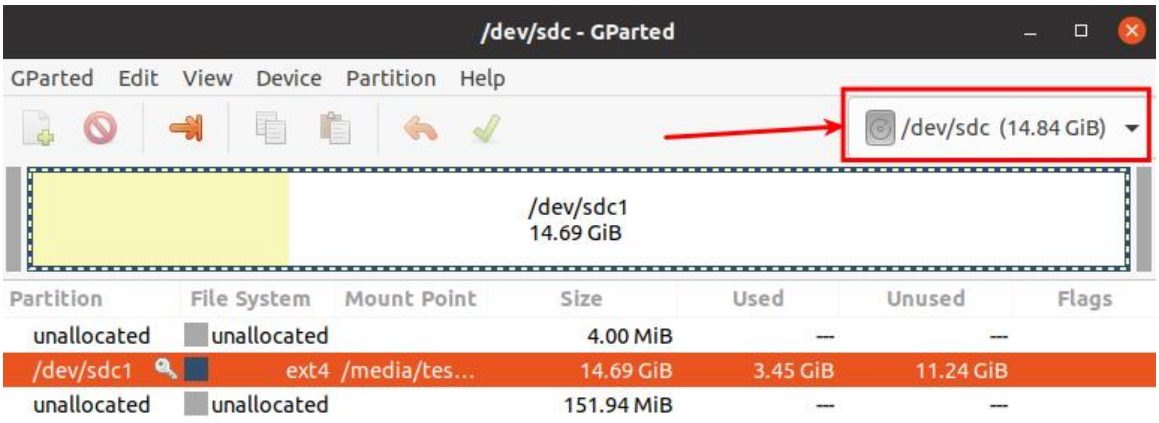
2) 然后在 Ubuntu 电脑中安装下 gparted 这个软件

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

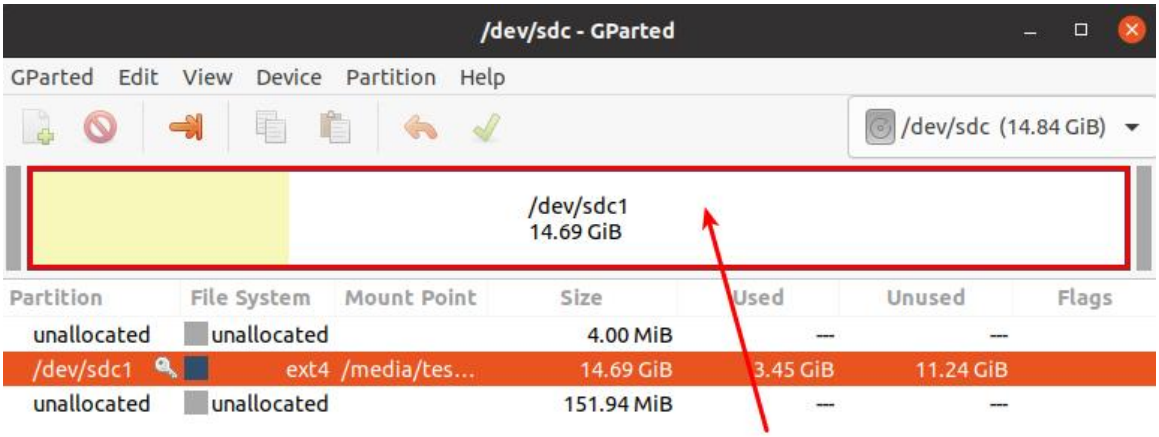
3) 然后打开 gparted

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

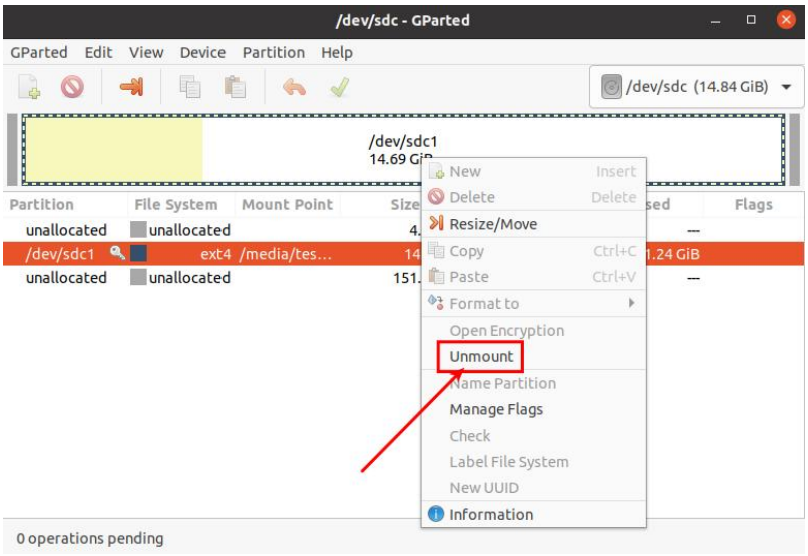
4) 打开 gparted 后在右上角可以选择 TF 卡，然后就可以看到 TF 卡容量的使用情况



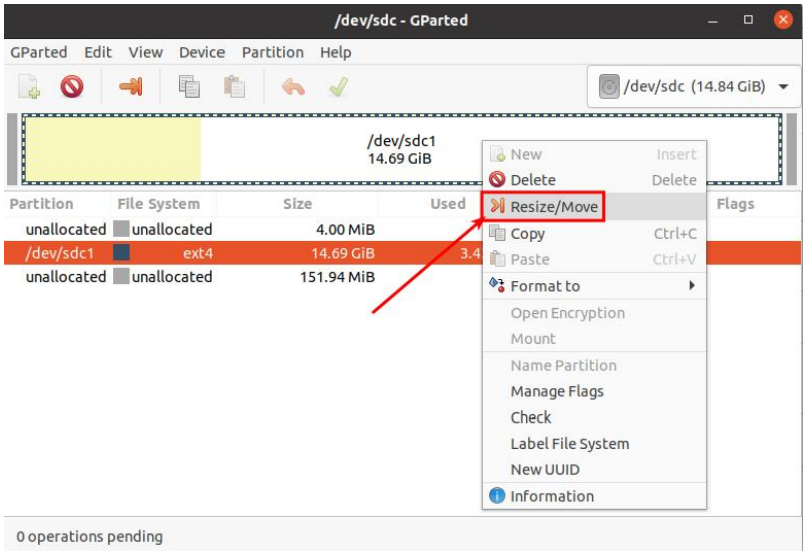
5) 然后选中 rootfs 分区（/dev/sdc1）



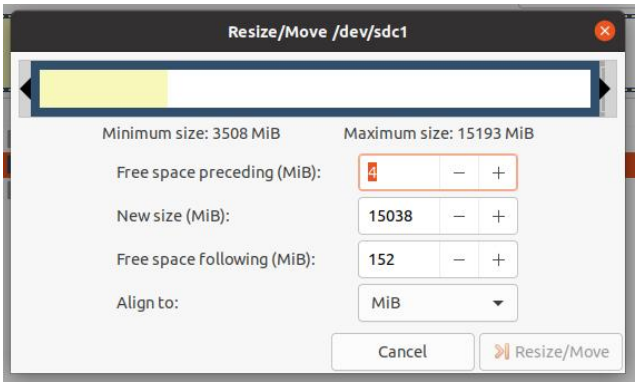
6) 再点击鼠标右键就可以看到下图所示的操作选项，如果 TF 卡已经挂载了，首先需要 Umount 掉 TF 卡的 rootfs 分区



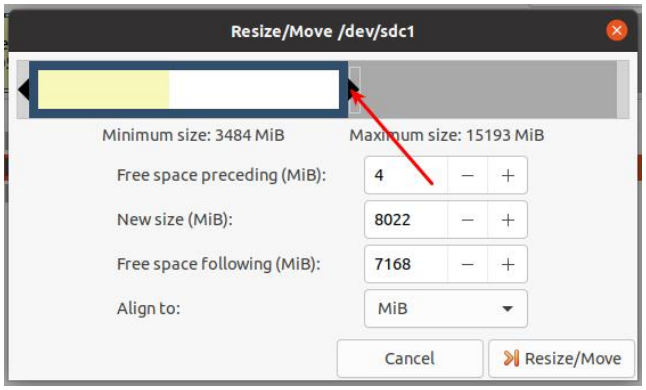
7) 然后再次选中rootfs分区，再点击鼠标右键，然后选择**Resize/Move**开始设置rootfs分区的大小



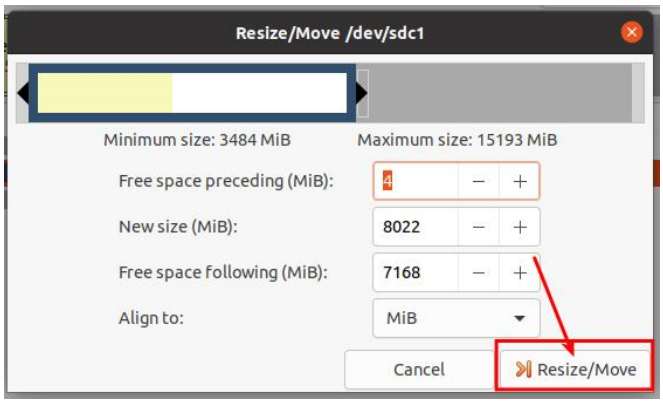
8) **Resize/Move** 选项打开后会弹出下面的设置界面



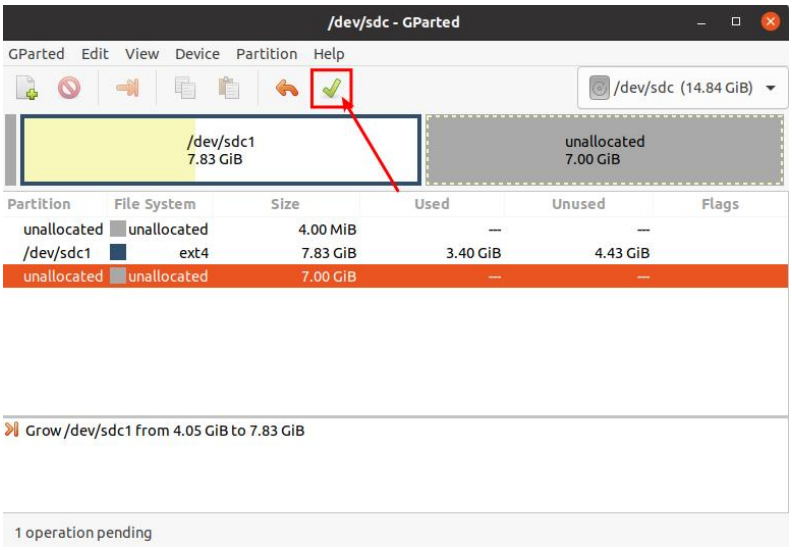
9) 然后可以直接拖动下图所示的位置来设置容量的大小，也可以通过设置 **New sieze(MiB)**中的数字来设置 rootfs 分区的大小



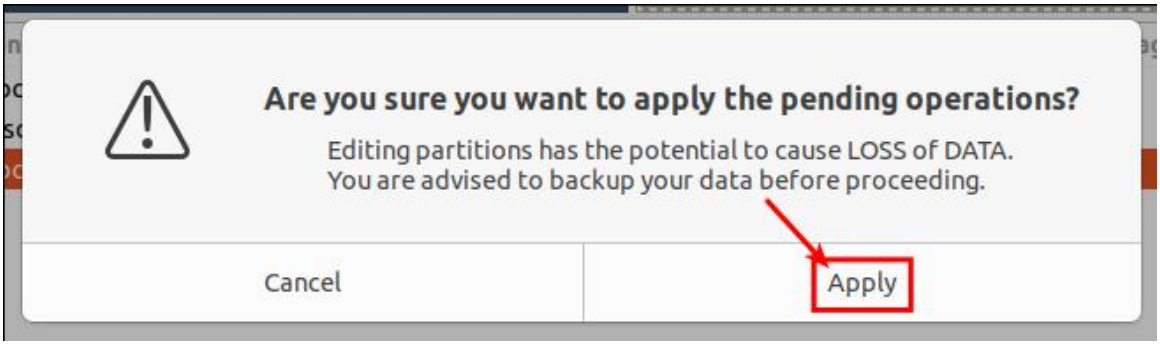
18) 设置好容量后，再点击右下角的 **Resize/Move** 即可



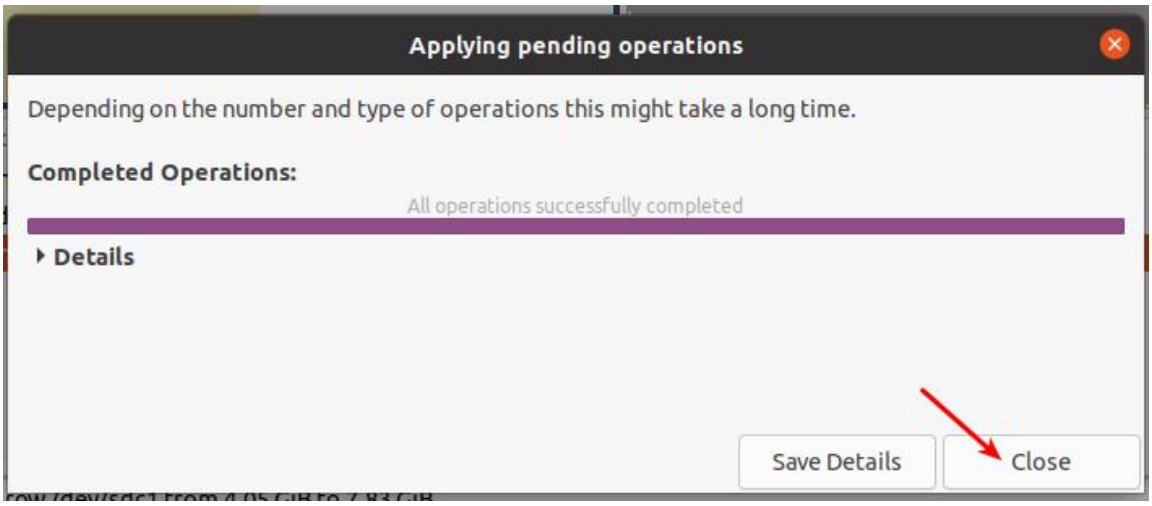
19) 最后确认无误后，再点击下图所示的**绿色** √



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



21) 扩容完成后点击 **Close** 关闭即可



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

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

3.7. 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 2.2.0 Focal ttyS0

orangepi 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

```
orangepi@orangepi:~$ sudo sed -i "s/verbosity=1/verbosity=7/" /boot/orangepiEnv.txt
orangepi@orangepi:~$ sudo sed -i "s/console=both/console=serial/" /boot/orangepiEnv.txt
```

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

```
orangepi@orangepi:~$ cat /boot/orangepiEnv.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

3.8. 网络连接测试

3.8.1. 以太网口测试

1) 首先将网线的一端插入开发板的以太网接口，网线的另一端接入路由器，并确保网络是畅通的

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

3) 在开发板的 Linux 系统中查看 IP 地址的命令如下所示

```
orangepi@orangepi:~$ ip addr show eth0
3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 5e:ac:14:a5:93:b3 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.16/24 brd 192.168.1.255 scope global dynamic noprefixroute eth0
       valid_lft 259174sec preferred_lft 259174sec
    inet6 240e:3b7:3240:c3a0:e269:8305:dc08:135e/64 scope global dynamic noprefixroute
       valid_lft 259176sec preferred_lft 172776sec
    inet6 fe80::957d:bbbd:4928:3604/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

开发板启动后查看 IP 地址有三种方法：

1. 接 HDMI 显示器，然后登录系统使用 **ip addr show eth0** 命令查看 IP 地址
2. 在调试串口终端输入 **ip addr show eth0** 命令来查看 IP 地址
3. 如果没有调试串口，也没有 HDMI 显示器，还可以通过路由器的管理界面来查看开发板网口的 IP 地址。不过这种方法经常有人会无法正常看到开发板的 IP 地址。如果看不到，调试方法如下所示：

A）首先检查 Linux 系统是否已经正常启动，如果开发板的绿灯亮了，一般是正常启动了，如果只亮红灯，或者红灯绿灯都没亮，说明系统都没正常启动；

B）检查网线有没有插紧，或者换根网线试下；

C）换个路由器试下（路由器的问题有遇到过很多，比如路由器无法正常分配 IP 地址，或者已正常分配 IP 地址但在路由器中看不到）；

D）如果没有路由器可换就只能连接 HDMI 显示器或者使用调试串口来查看 IP 地址。

另外需要注意的是开发板 DHCP 自动分配 IP 地址是不需要任何设置的。

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

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

3.8.2. WIFI 连接测试

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

3.8.2.1. 服务器版镜像通过命令连接 WIFI

当开发板没有连接以太网，没有连接 HDMI 显示器，只连接了串口时，推荐使用此小节演示的命令来连接 WIFI 网络。因为 nmtui 在某些串口软件（如 minicom）中只能显示字符，无法正常显示图形界面。当然，如果开发板连接了以太网或者 HDMI 显示屏，也可以使用此小节演示的命令来连接 WIFI 网络的。

1) 先登录 linux 系统，有下面三种方式
   a. 如果开发板连接了网线，可以通过 **ssh 远程登录 linux 系统**
   a. 如果开发板连接好了调试串口，可以使用串口终端登录 linux 系统
   b. 如果连接了开发板到HDMI显示器，可以通过HDMI显示的终端登录到linux系统

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

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

```
root@orangepi:~# nmcli dev wifi
IN-USE  BSSID              SSID                MODE   CHAN  RATE        SIGNAL  BARS  SECURITY
        28:6C:07:6E:87:2E  [redacted]_orangepi  Infra  9     260 Mbit/s  97            WPA1 WPA2
        D8:D8:66:A5:BD:D1  [redacted]           Infra  10    270 Mbit/s  90            WPA1 WPA2
        A0:40:A0:A1:72:20  [redacted]           Infra  4     405 Mbit/s  82            WPA2
        28:6C:07:6E:87:2F  [redacted]_orangepi_5G Infra  149   540 Mbit/s  80            WPA1 WPA2
        CA:50:E9:89:E2:44  [redacted]           Infra  1     130 Mbit/s  79            WPA1 WPA2
        A0:40:A0:A1:72:31  [redacted]           Infra  100   405 Mbit/s  67            WPA2
        D4:EE:07:08:A9:E0  [redacted]           Infra  4     130 Mbit/s  55            WPA1 WPA2
        88:C3:97:49:25:13  [redacted]           Infra  6     130 Mbit/s  52            WPA1 WPA2
        00:BD:82:51:53:C2  [redacted]           Infra  12    130 Mbit/s  49            WPA1 WPA2
        C0:61:18:FA:49:37  [redacted]           Infra  149   270 Mbit/s  47            WPA1 WPA2
        04:79:70:8D:0C:B8  [redacted]           Infra  153   270 Mbit/s  47            WPA2
        04:79:70:FD:0C:B8  [redacted]           Infra  153   270 Mbit/s  47            WPA2
        9C:A6:15:DD:E6:0C  [redacted]           Infra  10    270 Mbit/s  45            WPA1 WPA2
        B4:0F:3B:45:D1:F5  [redacted]           Infra  48    270 Mbit/s  45            WPA1 WPA2
        E8:CC:18:4F:7B:44  [redacted]           Infra  157   135 Mbit/s  45            WPA1 WPA2
        B0:95:8E:D8:2F:ED  [redacted]           Infra  11    405 Mbit/s  39            WPA1 WPA2
        C0:61:18:FA:49:36  [redacted]           Infra  11    270 Mbit/s  24            WPA1 WPA2
root@orangepi:~#
```

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

```
orangepi@orangepi:~$ nmcli dev wifi connect wifi_name password wifi_passwd
Device 'wlan0' successfully activated with 'cf937f88-ca1e-4411-bb50-61f402eef293'.
```

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

```
orangepi@orangepi:~$ ip addr show wlan0
11: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 23:8c:d6:ae:76:bb brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.11/24 brd 192.168.1.255 scope global dynamic noprefixroute wlan0
       valid_lft 259192sec preferred_lft 259192sec
    inet6 240e:3b7:3240:c3a0:c401:a445:5002:ccdd/64 scope global dynamic noprefixroute
       valid_lft 259192sec preferred_lft 172792sec
    inet6 fe80::42f1:6019:a80e:4c31/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

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

```
orangepi@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
```

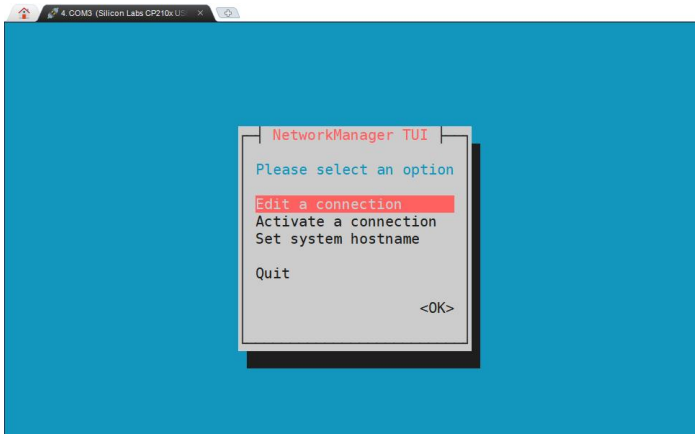
3.8.2.2. 服务器版镜像通过图形化方式连接 WIFI

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

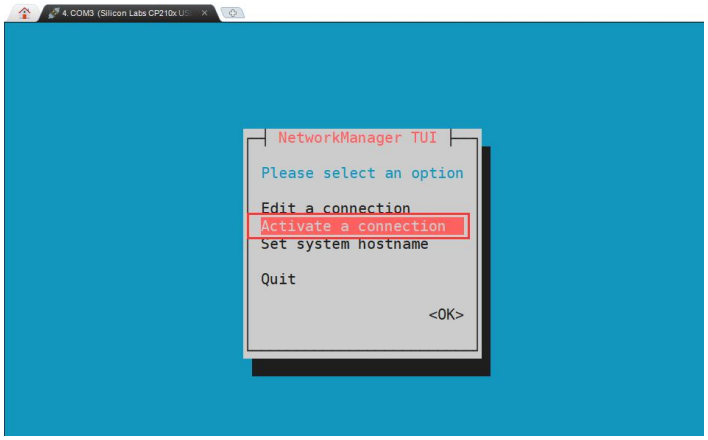
2) 然后在命令行中输入 nmtui 命令打开 wifi 连接的界面

```
orangepi@orangepi:~$ nmtui
```

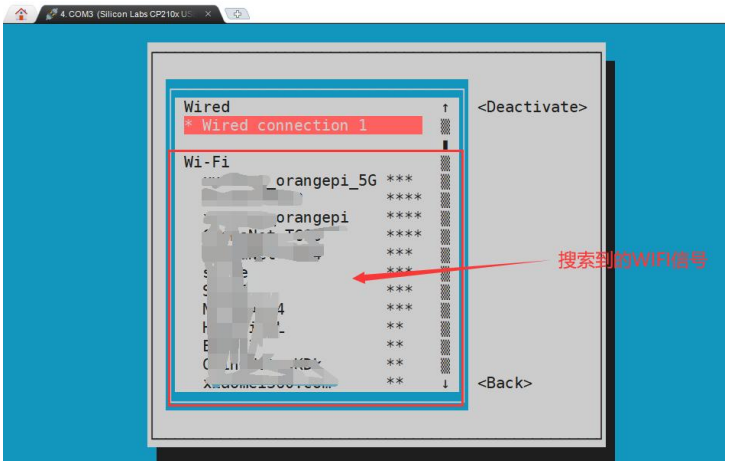
3) 输入 nmtui 命令打开的界面如下所示



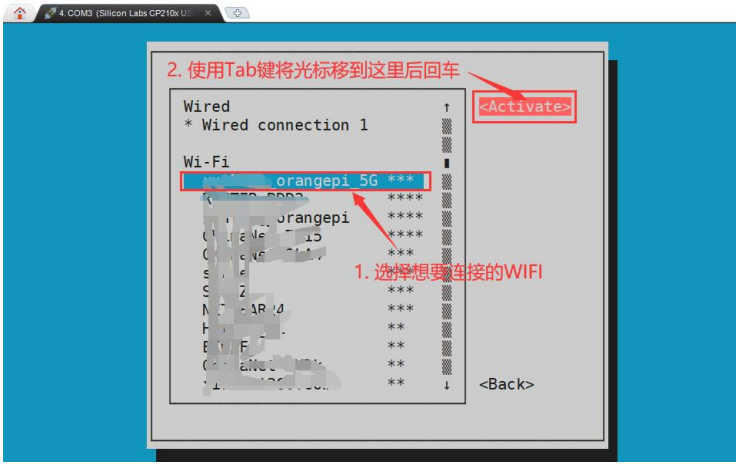
4) 选择 **Activate a connect** 后回车



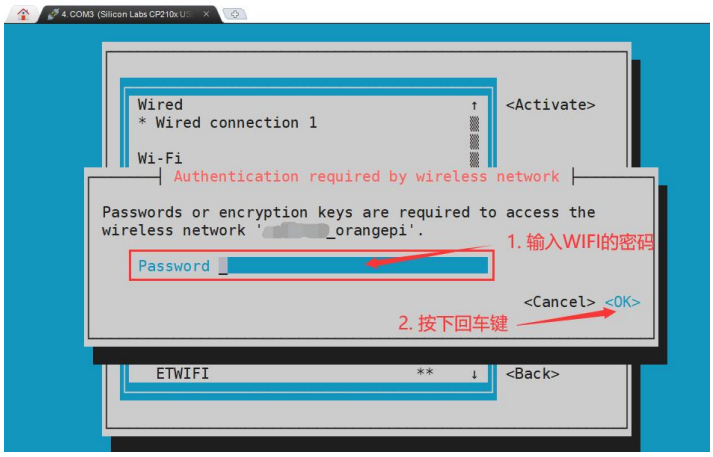
5) 然后就能看到所有搜索到的 WIFI 热点



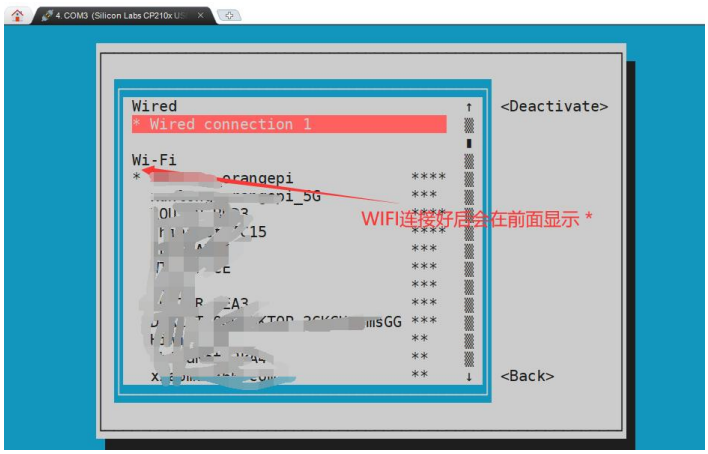
6) 选择想要连接的 WIFI 热点后再使用 Tab 键将光标定位到 **Activate** 后回车



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



8) WIFI 连接成功后会在已连接的 WIFI 名称前显示一个“*”



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

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

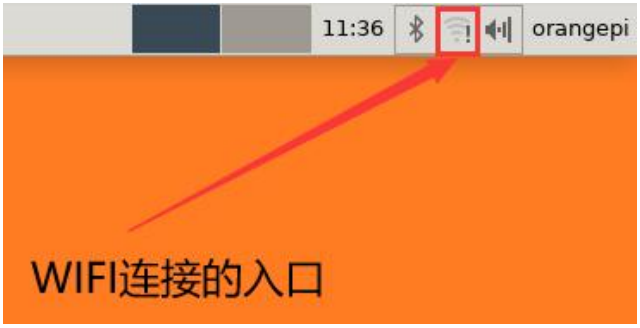
10) 使用 **ping** 命令可以测试 wifi 网络的连通性，**ping** 命令可以通过 **Ctrl+C** 快捷键来中断运行

```
orangepi@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 ---
```

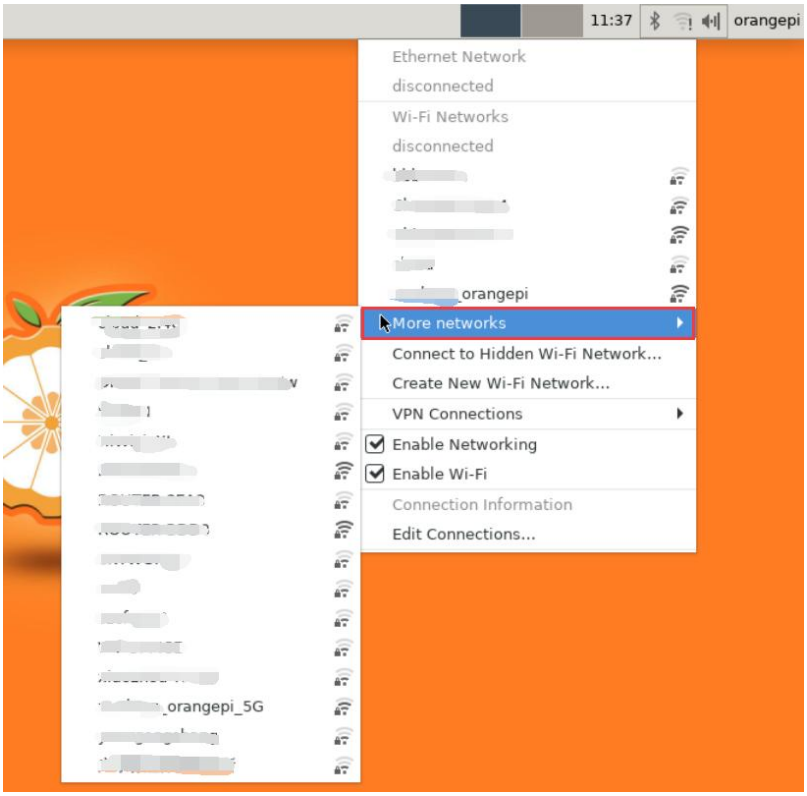
```
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 41.321/44.864/48.834/2.484 ms
```

3.8.2.3. 桌面版镜像的测试方法

1) 点击桌面右上角的网络配置图标（测试 WIFI 时请不要连接网线）

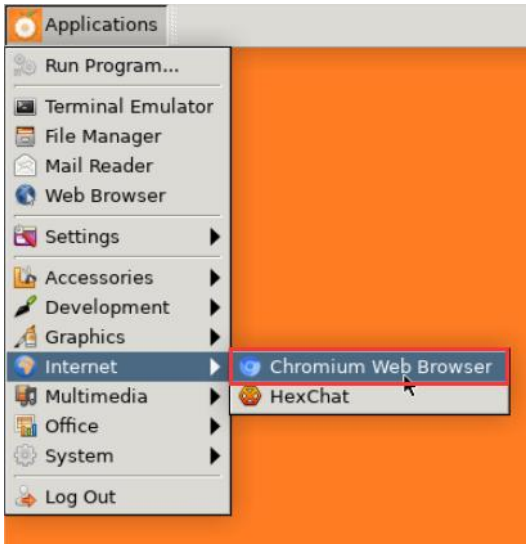


2) 在弹出的下拉框中点击 **More networks** 可以看到所有扫描到的 WIFI 热点，然后选择想要连接的 WIFI 热点

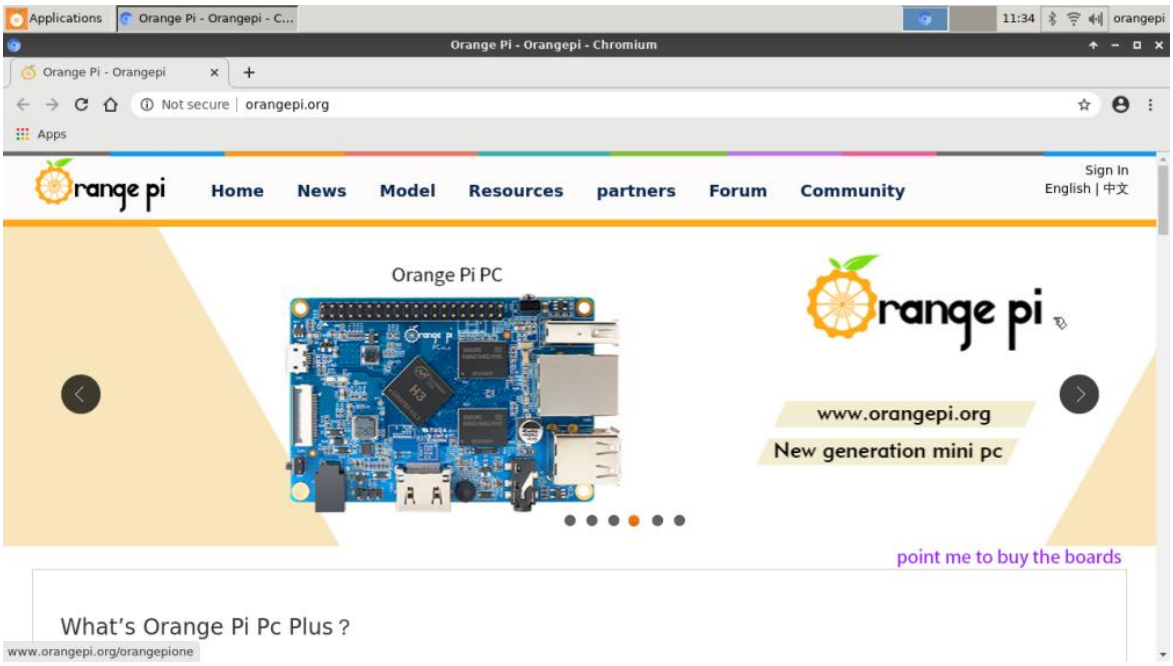


3) 然后输入 WIFI 热点的密码，再点击 **Connect** 就会开始连接 WIFI

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



5) 打开浏览器后如果能看到Orange Pi网站的页面 ，或者能打开其他网页说明 WIFI 连接正常



3. 8. 3. 使用 Hostapd 建立 WIFI 热点的方法

首先请确保开发板连接好了网线，并且能正常上网。因为设置 Hostapd 建立 WIFI 热点的过程中需要从网上下载一些软件包。如果开发板无法通过有线网口上网，会导致安装失败。

如果没有连接网线，当其它网络设备（如手机或电脑）连接开发板发射的 WIFI 热点后，是无法正常访问外部网络的（如手机浏览器打不开网页）。

使用 Hostapd 建立 WIFI 热点后（注意是设置完之后），如果不用访问外部网络，只是需要连接开发板发射的 WIFI 热点，那么不连接网线也是可以的。

另外设置 Hostapd 前，请确保 WIFI 是没有连接网络的，不然会提示 WIFI 正在使用导致无法正常设置 Hostapd。

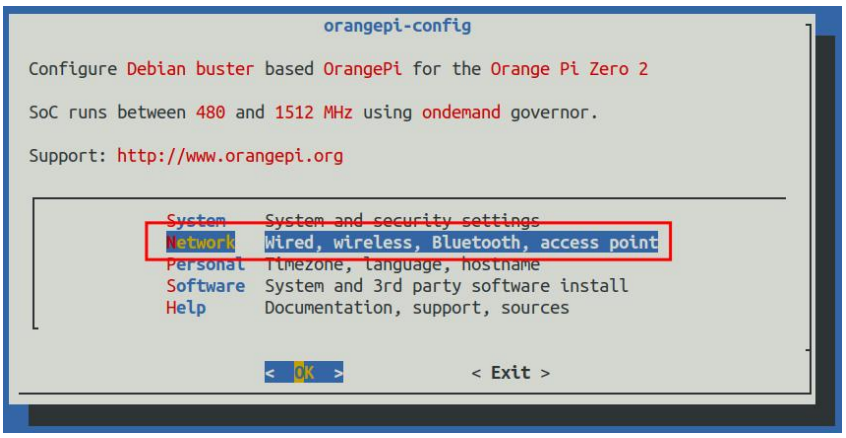
1) 首先更新下系统的软件源索引

```
root@orangepi:~$ sudo apt-get update
```

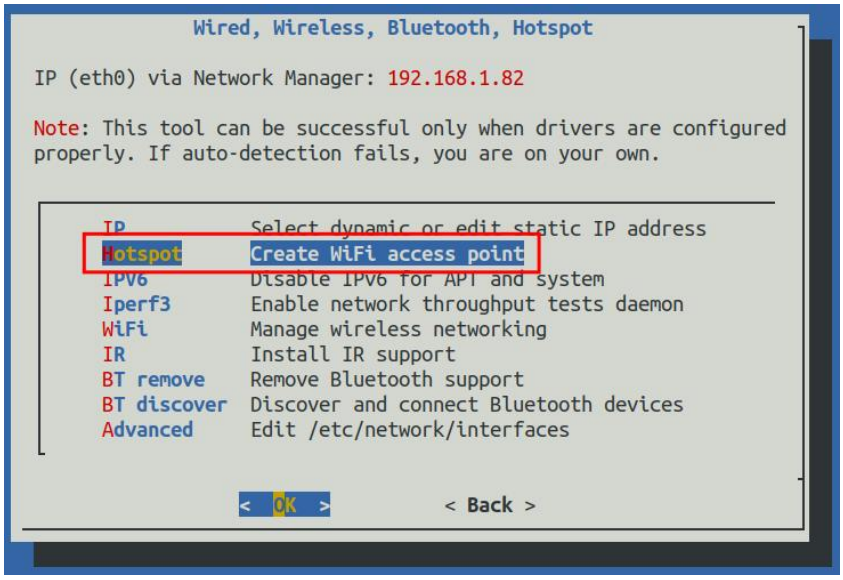
2) 然后在终端中输入 **orangepi-config** 命令，**记得加 sudo 权限**

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

3) orangepi-config 打开后的界面如下图所示，选择 **Network** 选项即可进入网络相关的设置界面

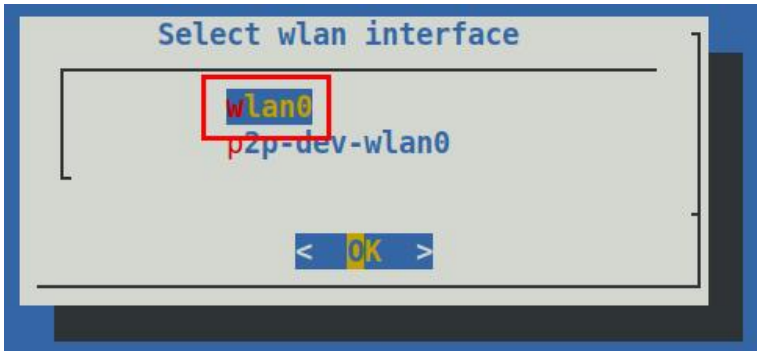


4) 然后选择 **Hotspot Create WiFi access point** 选项开始设置 Hotspot

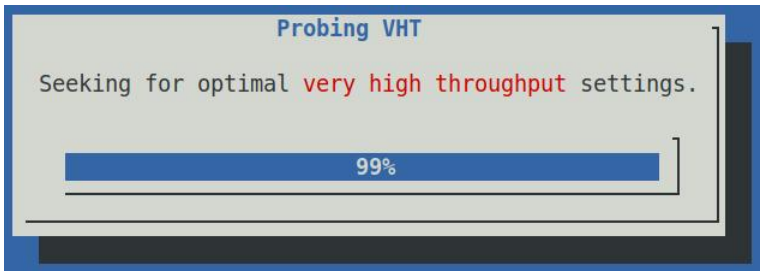


5) 当弹出下面的选择框时选择 **wlan0** 即可

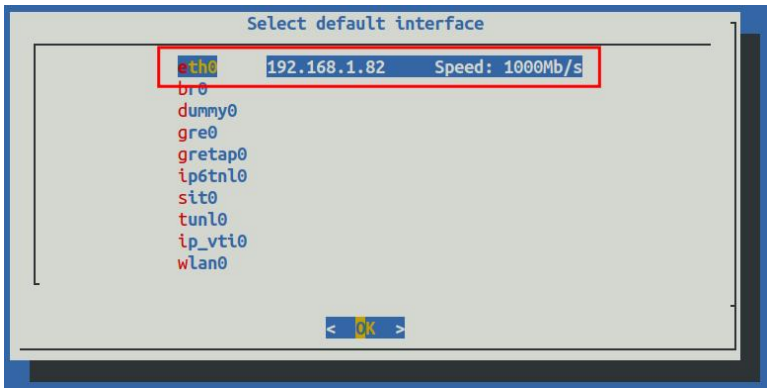
注意，此选择框在Ubuntu Bionic和Debian Buster中不会出现。



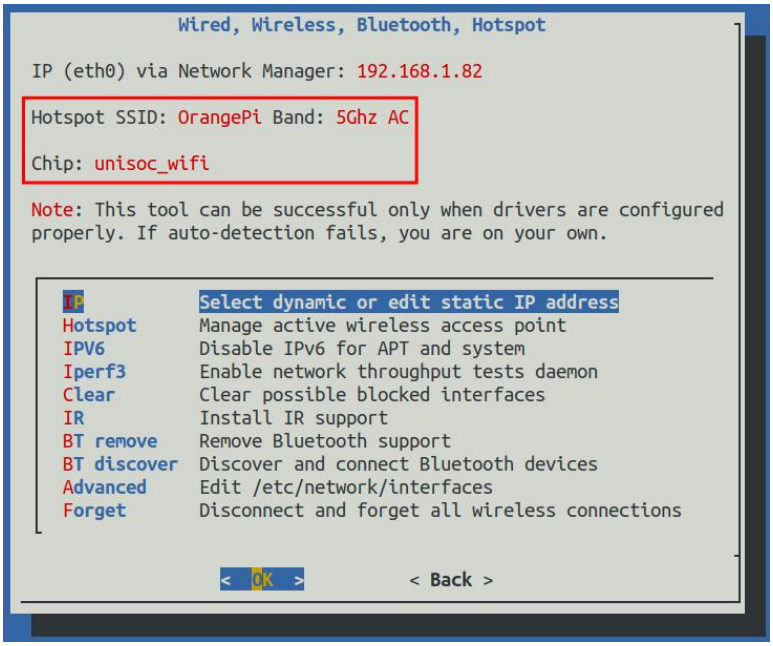
6) 然后 orangepi-config 会开始一系列的设置，此时耐心等待即可



7) 等待一段时间后会弹出下面的选择框，请选择第一个 **eth0**



8) 全部设置完成后，orangepi-config 如果显示下面的界面说明 Hostapd 设置正确

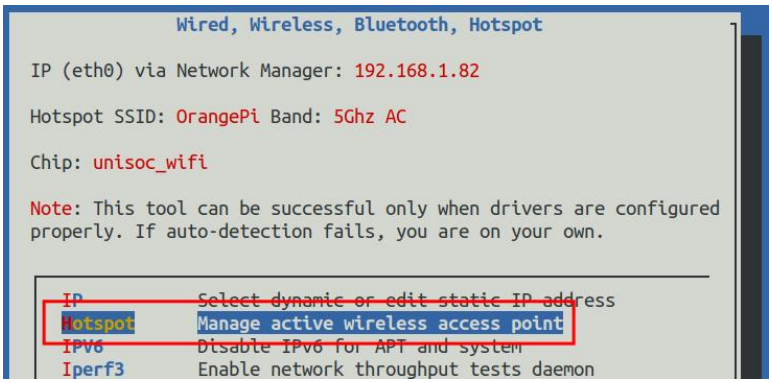


9) Hotspot 设置的 WIFI 热点的名称默认为：**OrangePi**，密码为：**12345678**，如果一切正常此时手机或者电脑就可以搜到名为 OrangePi 的 WIFI 热点了，连接上后如果

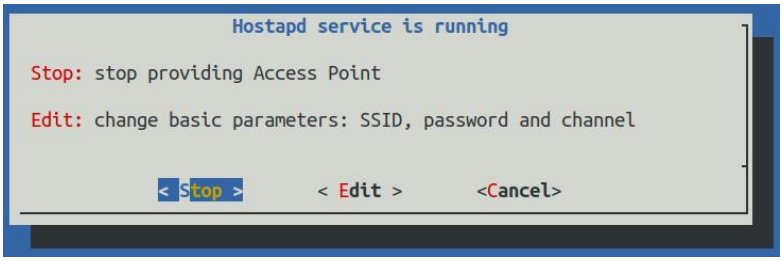
可以正常上网说明开发板的 Hostapd 设置正确。下图为手机连接开发板发射的 WIFI 热点示意图：

10) Hotspot 设置完成后，再在 orangepi-config 中打开 Hostapd 可以对其进行配置

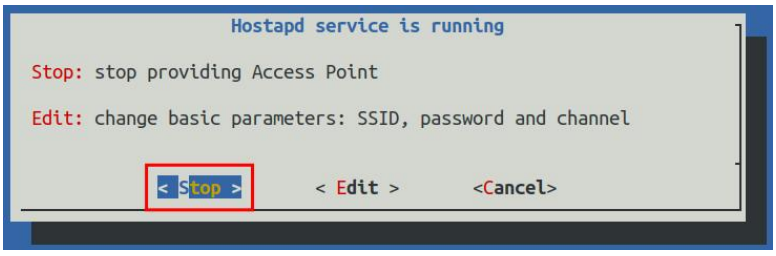
a. 首先在 config-config 中选择 **Hotspot**



b. 然后就可以看到下面的选择界面

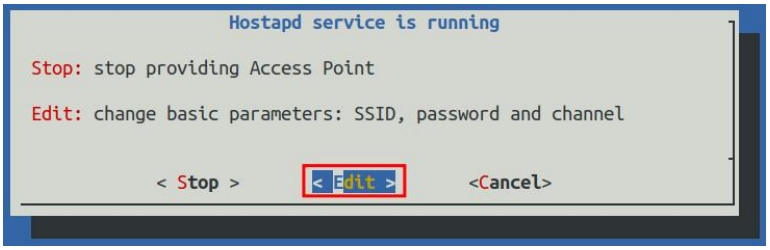


c. 选择 **Stop** 可以停止 Hostapd 服务

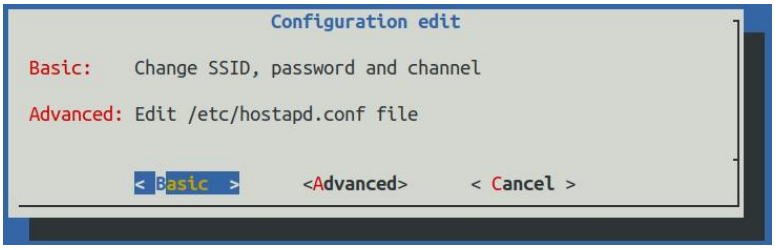


d. 选择 **Edit** 可以编辑 Hostapd 的配置

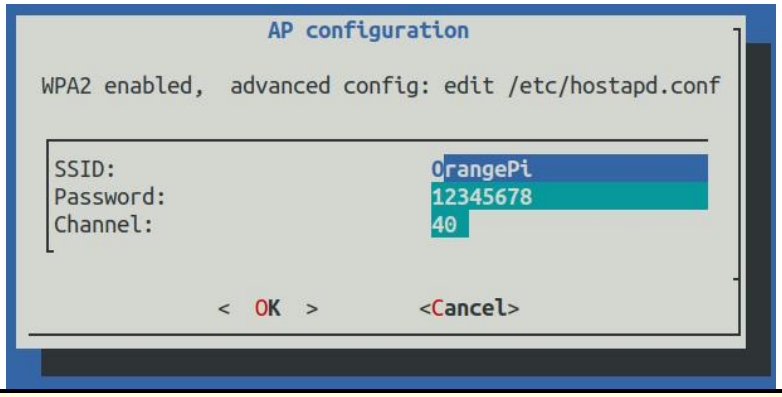
a) Hostapd 的 **Edit** 编辑选项位置如下图所示



b) **Edit** 选项打开后如下图所示

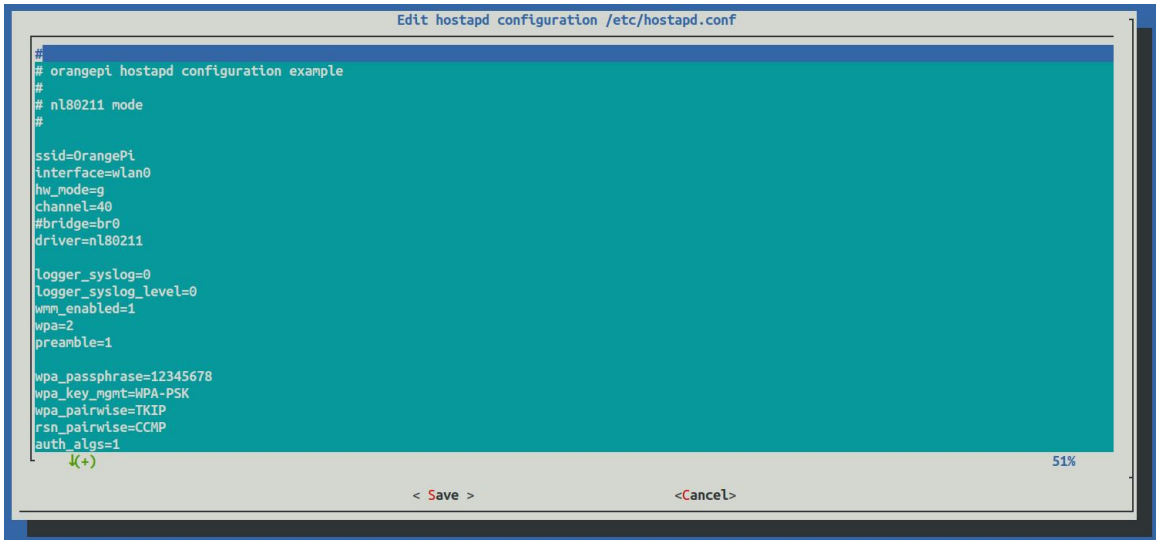


i. 选择 **Basic** 可以修改 WIFI 热点的名称和密码，修改完后选择**<OK>**保存即可



注意：如果要修改密码，修改后的密码不能少于 8 个字符，不然会导致 Hostapd 服务无法正常运行。

ii. 选择 **Advanced** 可以在**/etc/hostapd.conf** 这个 hostapd 的配置文件中直接修改 WIFI 热点的名称和密码以及其他的配置，修改完后选择**<Save>**保存即可



注意： 如果要修改密码，修改的密码不能少于 8 个字符，不然会导致 Hostapd 服务无法正常运行

3.8.4. 设置静态 IP 地址的方法

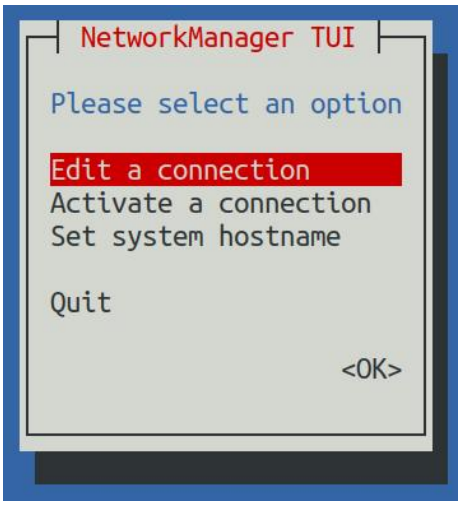
请不要通过修改/etc/network/interfaces 配置文件的方式来设置静态 IP 地址。

3.8.4.1. 使用 nmtui 命令来设置静态 IP 地址

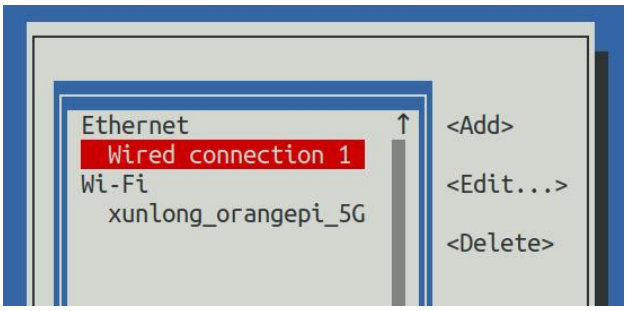
1) 首先运行 **nmtui** 命令

```
orangepi@orangepi:~$ nmtui
```

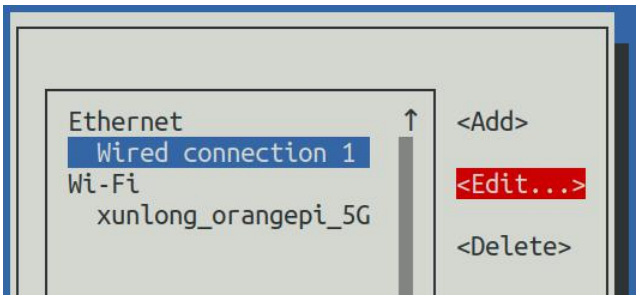
2) 然后选择 **Edit a connection** 并按下回车键



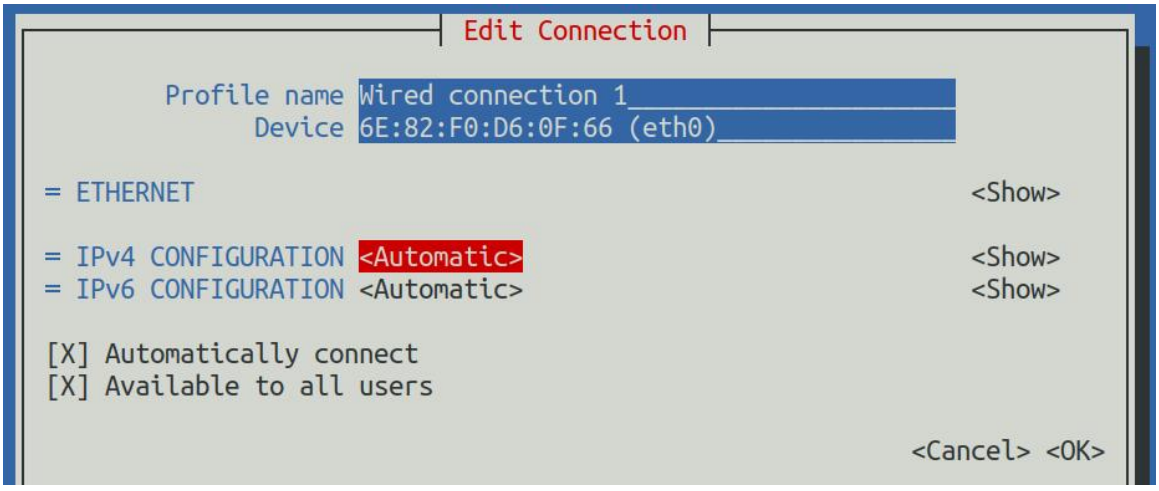
3) 然后选择需要设置静态 IP 地址的网络接口，比如设置 **Ethernet** 接口的静态 IP 地址选择 **Wired connection 1** 就可以了



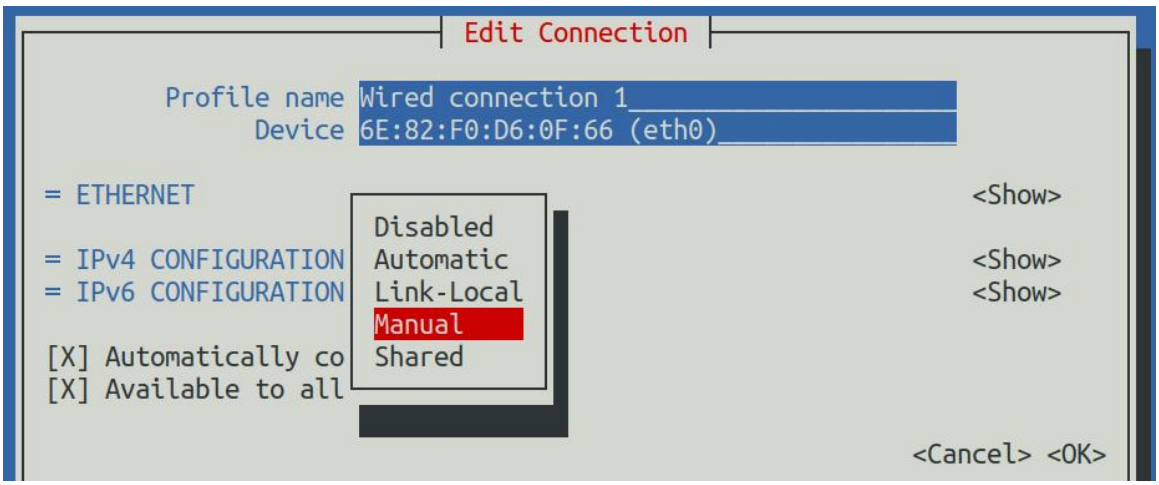
4) 然后通过 **Tab** 键选择 **Edit** 并按下回车键



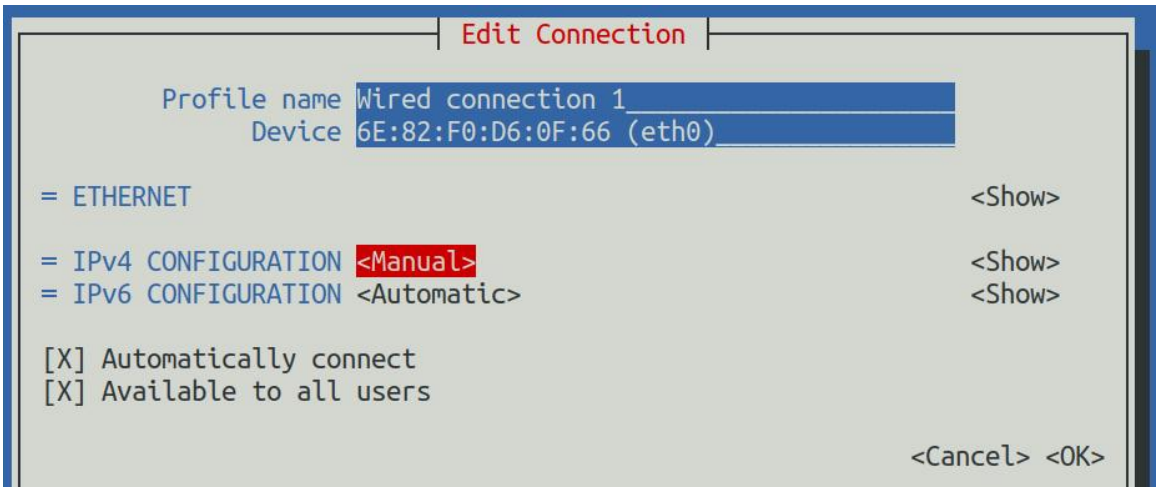
5) 然后通过 Tab 键将光标移动到下图所示的**<Automatic>**位置进行 IPv4 的配置



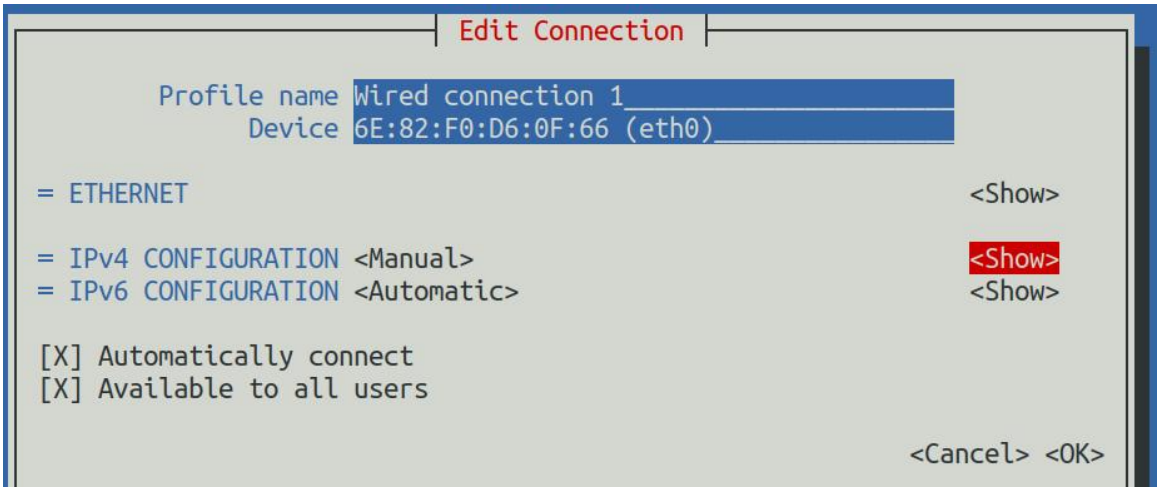
6) 然后回车，通过上下方向键选择 **Manual**，然后回车确定



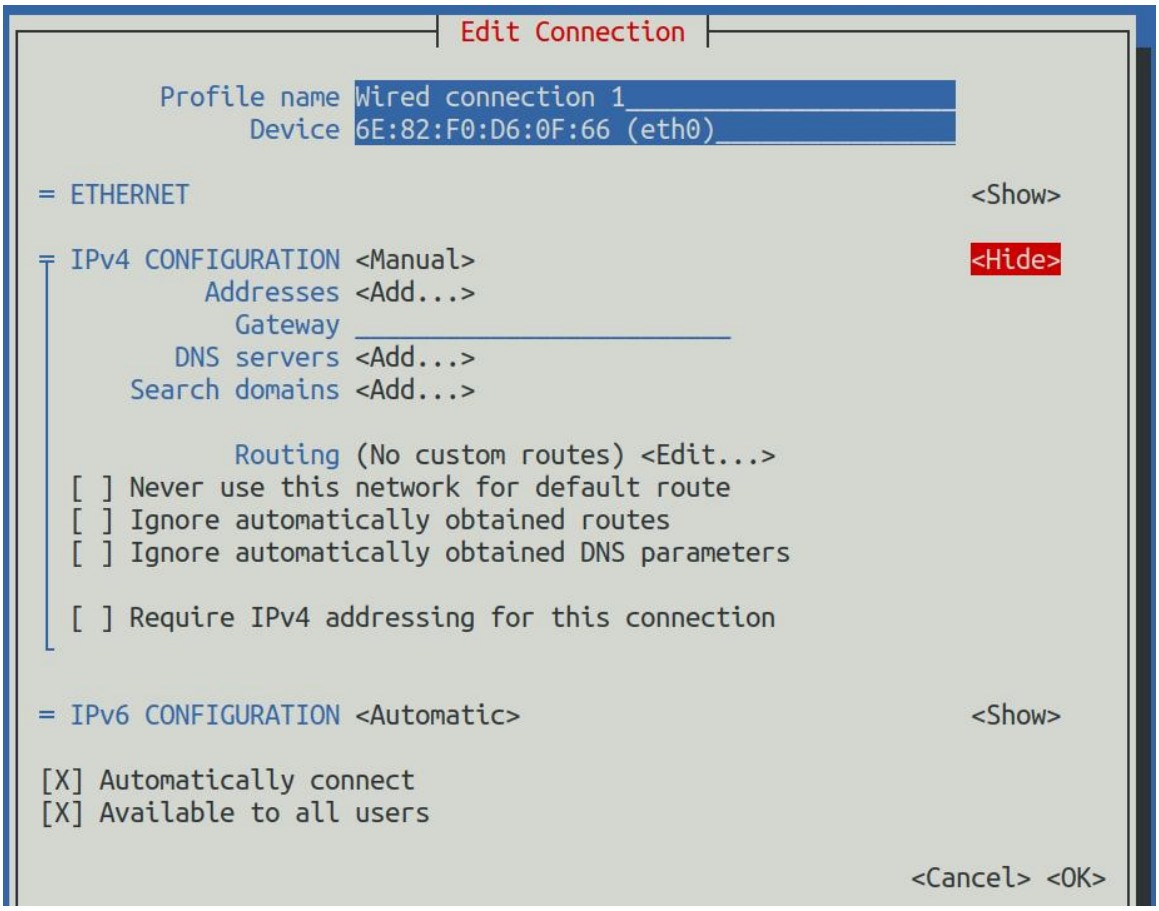
7) 选择完后的显示如下图所示



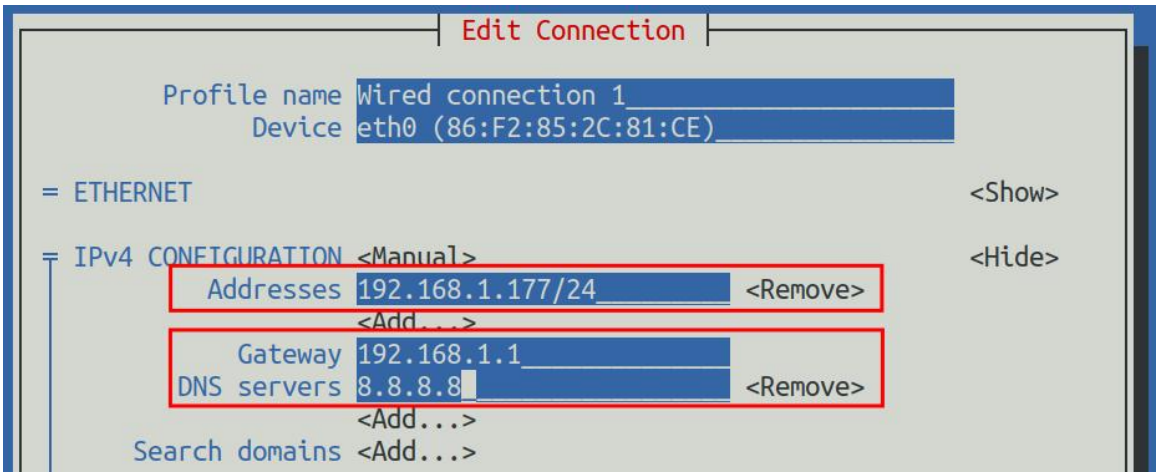
8) 然后通过 Tab 键将光标移动到**<Show>**



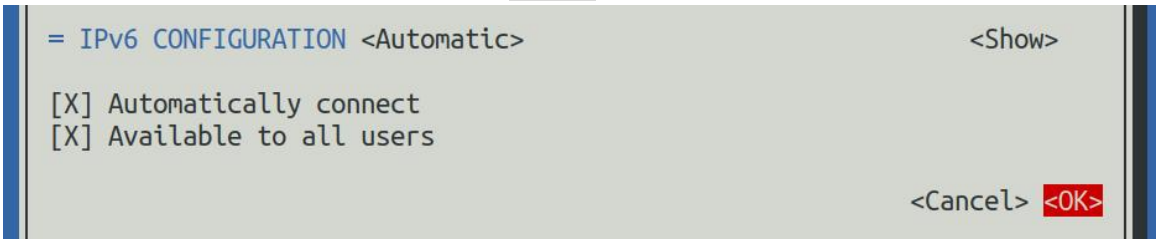
9) 然后回车，回车后会弹出下面的设置界面



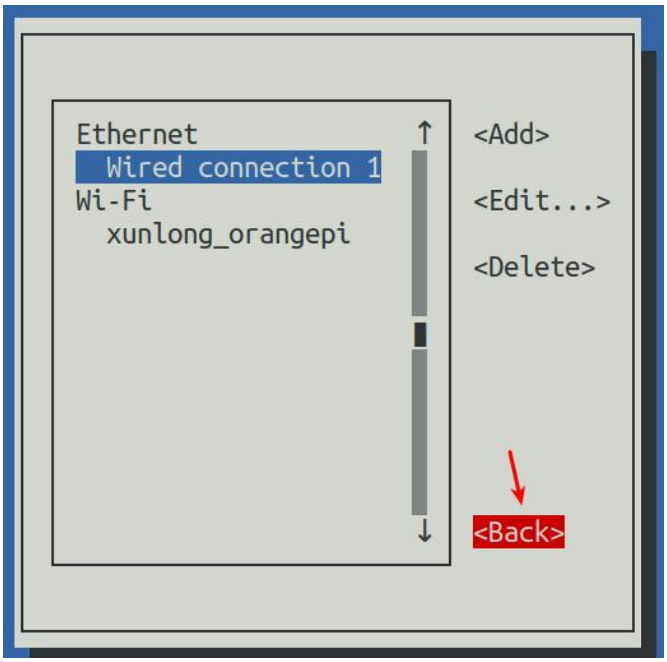
10) 然后就可以在下图所示的位置设置 IP 地址(Addresses)、网关(Gateway)和 DNS 服务器的地址（里面还有很多其他设置选项，请自行探索），**请根据自己的具体需求来设置，下图中设置的值只是一个示例**



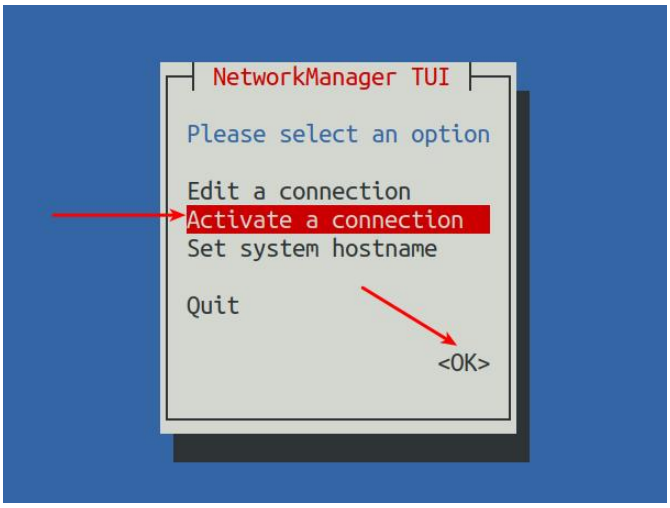
11) 设置完后将光标移动到右下角的**<OK>**，然后回车确认



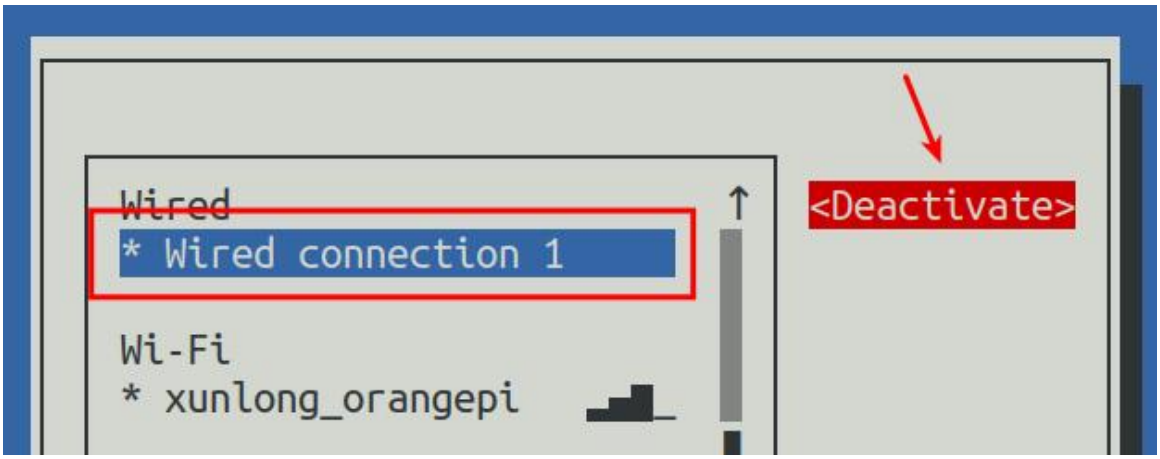
12) 然后点击**<Back>**回退到上一级选择界面



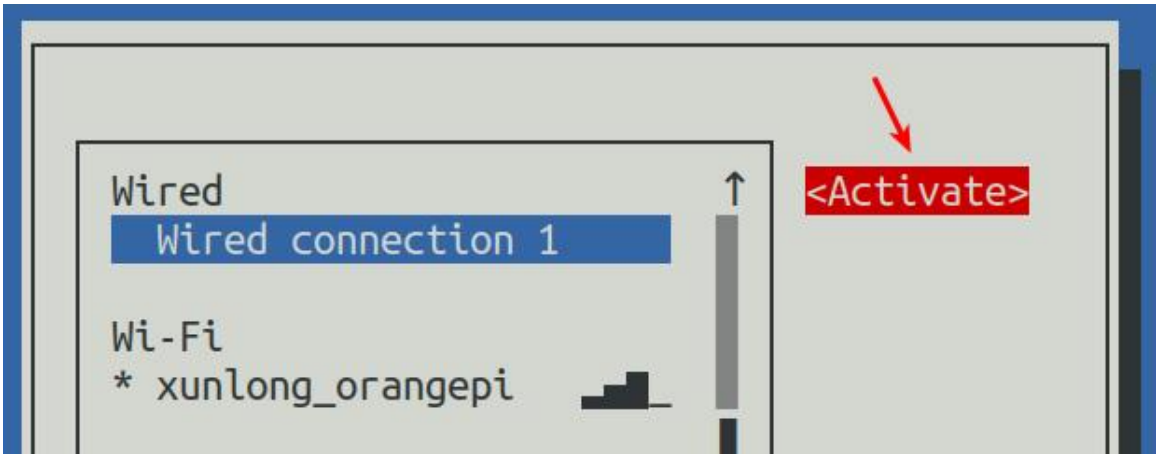
13) 然后选择 **Activate a connection**，再将光标移动到**<OK>**，最后点击回车



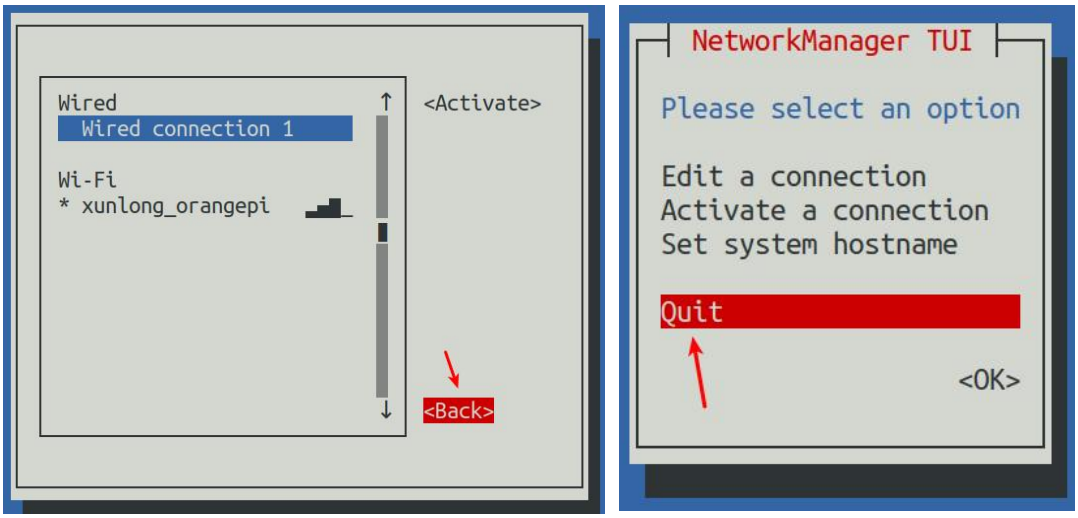
14) 然后选择需要设置的网络接口，比如 **Wired connection 1**，然后将光标移动到 **<Deactivate>**，再按下回车键禁用 **Wired connection 1**



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**ip addr show eth0**就能看到网口的IP地址已经变成前面设置的静态IP地址了

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

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

```
orangepi@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
```

3.8.4.2. 使用 nmcli 命令来设置静态 IP 地址

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

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

a. **orangepi** 为 WIFI 网络接口的名字（名字不一定相同）

b. **Wired connection 1** 为以太网接口的名字

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

3) 然后输入下面的命令，其中

a. **"Wired connection 1"** 表示设置以太网口的静态 IP 地址，如果需要设置 WIFI 的静态 IP 地址，请修改为 WIFI 网络接口对应的名字（通过 **nmcli con show** 命令可以获取到）

b. **ipv4.addresses** 后面是要设置的静态 IP 地址，可以修改为自己想要设置的值

c. **ipv4.gateway** 表示网关的地址

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

4) 然后重启 linux 系统

```
orangepi@orangepi:~$ sudo reboot
```

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

```
orangepi@orangepi:~$ ip addr show eth0
```

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

3.8.5. 设置 Linux 系统第一次启动自动连接网络的方法

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

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

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

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

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

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

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

1) 首先烧录想使用的开发板的 Linux 镜像到 TF 卡中，然后使用读卡器，将烧录好开发板 Linux 镜像的 TF 卡插入安装有 Linux 系统的机器中（比如安装有 Ubuntu 系统的电脑，下面都以 Ubuntu 电脑为例来演示）

2) 当 TF 卡插入 Ubuntu 电脑后，Ubuntu 电脑一般会自动挂载 TF 卡中的 Linux 根文件系统的分区，由下面的命令可以知道，
/media/test/d17d5e4f-ae41-4554-a727-bb5c9c94134f 即为 TF 卡中的 Linux 根文件系统挂载的路径（**路径名以实际看到的为准，不一定相同**）

```
test@test:~$ df -h | grep "media"
/dev/sdd1    1.4G    1.2G    167M    88%
/media/test/d17d5e4f-ae41-4554-a727-bb5c9c94134f
test@test:~$ ls /media/test/d17d5e4f-ae41-4554-a727-bb5c9c94134f
bin  boot  dev  etc  home  lib  lost+found  media  mnt  opt  proc  root  run
sbin  selinux  srv  sys  tmp  usr  var
```

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

```
test@test:~$ cd /media/test/d17d5e4f-ae41-4554-a727-bb5c9c94134f/boot
```

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

```
test@test:/media/test/d17d5e4f-ae41-4554-a727-bb5c9c94134f/boot$ sudo cp \
orangepi_first_run.txt.template orangepi_first_run.txt
```

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

```
test@test:/media/test/d17d5e4f-ae41-4554-a727-bb5c9c94134f/boot$ sudo vim \
orangepi_first_run.txt
```

6) orangepi_first_run.txt 文件中的变量使用说明

a. **FR_general_delete_this_file_after_completion** 变量用来设置第一次启动完后是否删除 orangepi_first_run.txt 这个文件，默认为 1，也就是删除，如果设置为 0，第一次启动后会将 orangepi_first_run.txt 重命名为 orangepi_first_run.txt.old，一般保持默认值即可

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

7) 下面演示几个具体的设置示例：
a. 比如想要开发板的 Linux 系统第一次启动后自动连接 WIFI 热点，可以这样设置：
    a) 设置 **FR_net_change_defaults** 为 **1**
    b) 设置 **FR_net_wifi_enabled** 为 **1**
    c) 设置 **FR_net_wifi_ssid** 为想要连接的 WIFI 热点的名字
    d) 设置 **FR_net_wifi_key** 为想要连接的 WIFI 热点的密码

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

c. 比如想要开发板的 Linux 系统第一次启动后自动设置以太网口的 IP 地址为

想要的静态 IP 地址，可以这样设置：

a) 设置 **FR_net_change_defaults** 为 **1**

b) 设置 **FR_net_ethernet_enabled** 为 **1**

c) 设置 **FR_net_use_static** 为 **1**

d) 设置 **FR_net_static_ip** 为想要的 IP 地址

e) 设置 **FR_net_static_gateway** 为对应的网关地址

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

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

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

3.9. 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 ip addr 命令或者通过查看路由器的方式获取开发板的 IP 地址。

3.9.1. Ubuntu 下 SSH 远程登录开发板

1) 获取开发板的 IP 地址

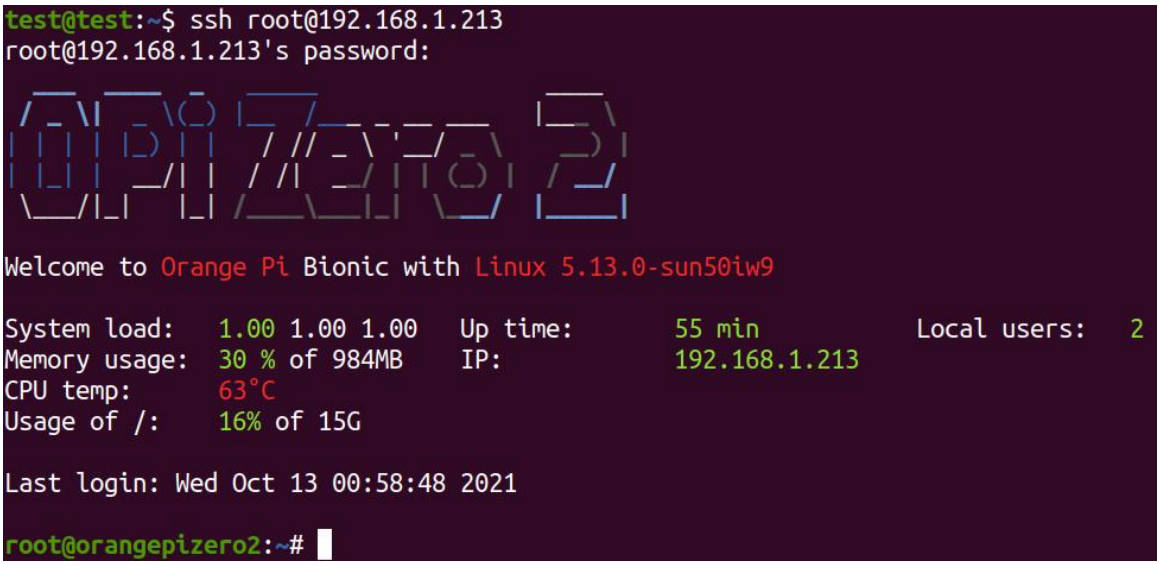
2) 然后就可以通过 ssh 命令远程登录 linux 系统

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

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

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

3) 成功登录系统后的显示如下图所示



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

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

如果还不行，请重烧系统试下。

3.9.2. Windows 下 SSH 远程登录开发板

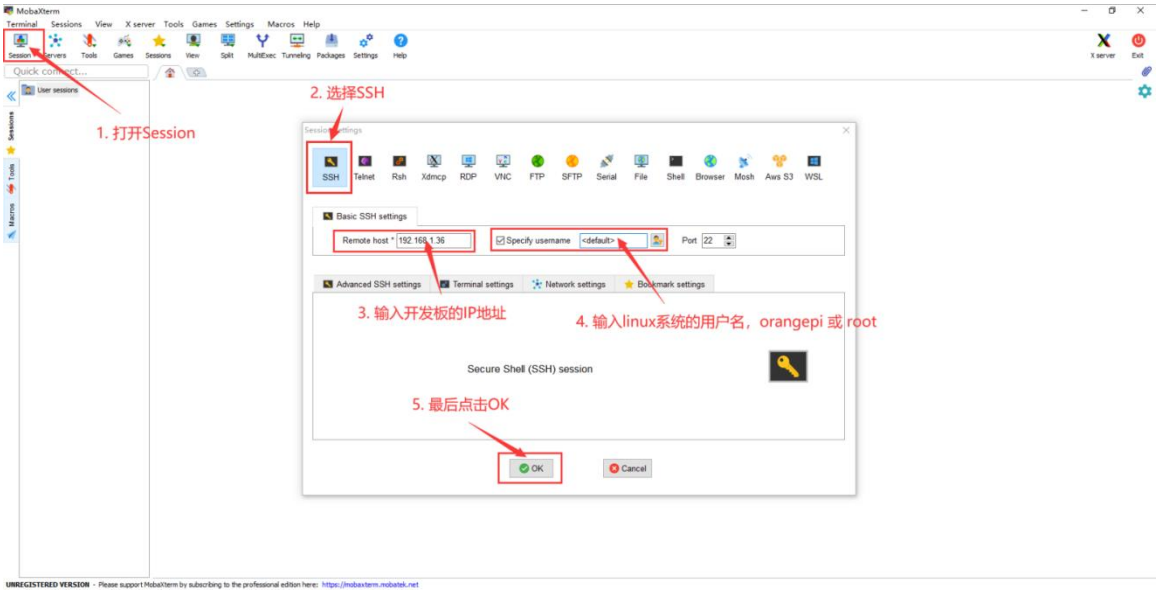
1) 首先获取开发板的 IP 地址

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. 然后在 **Specify username** 中输入 linux 系统的用户名 **root** 或 **orangepi**

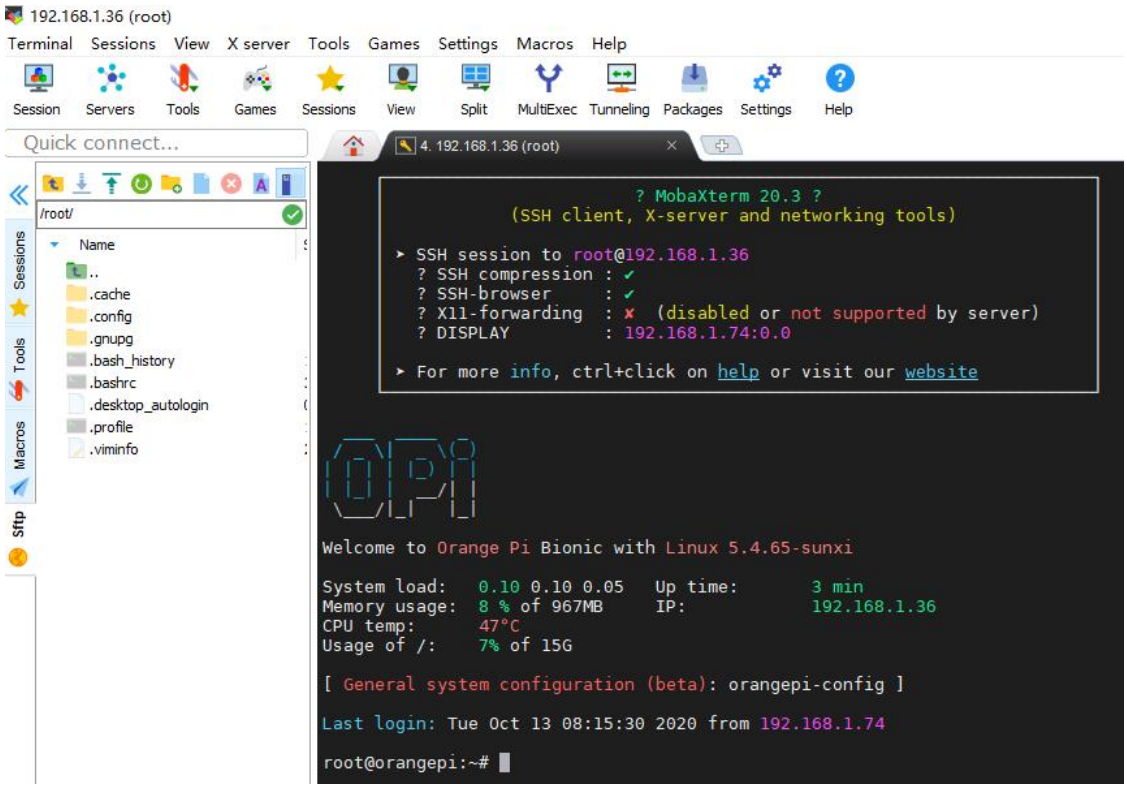
e. 最后点击 **OK** 即可



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

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

4) 成功登录系统后的显示如下图所示



3.10. HDMI 测试

3.10.1. 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

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

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

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

3.10.2. HDMI 转 VGA 显示测试

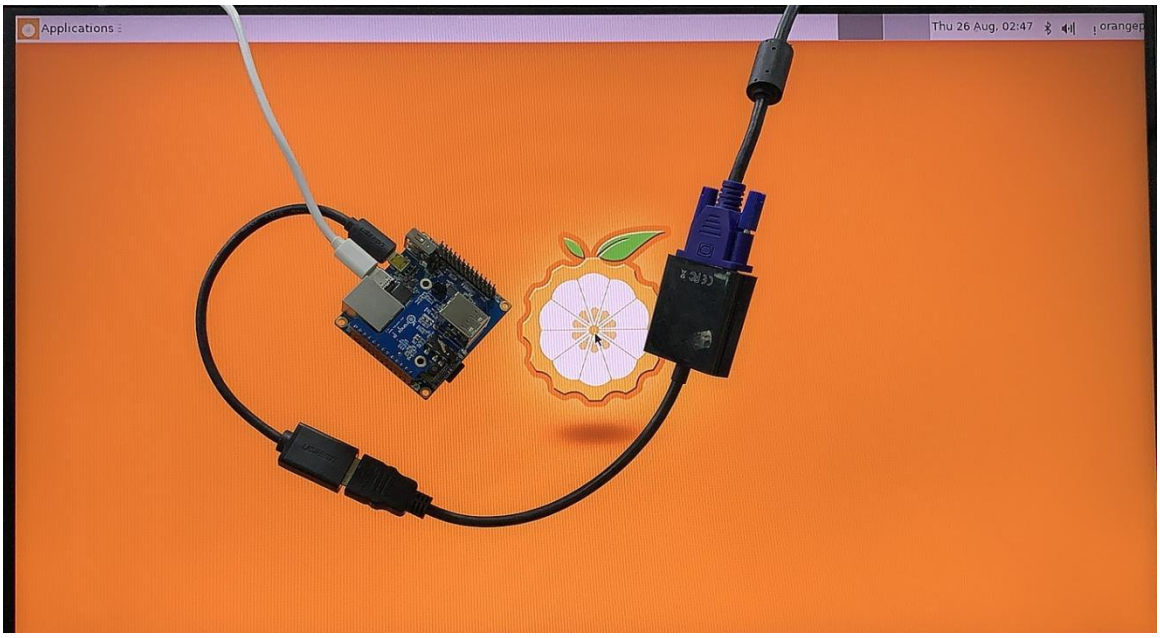
1) 首先需要准备下面的配件

a. HDMI 转 VGA 转换器

b. 一根 VGA 线和一根 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

3.10.3. Linux4.9 HDMI 分辨率设置的方法

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** 中有个 disp_mode 变量，可以通过它来设置 HDMI 输出的分辨率，linux 系统默认设置的分辨率为 1080p60

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

2) disp_mode 变量支持设置的值如下表所示

disp_mode 支持的值	HDMI 分辨率	HDMI 刷新率
480i	720x480	60
576i	720x480	50

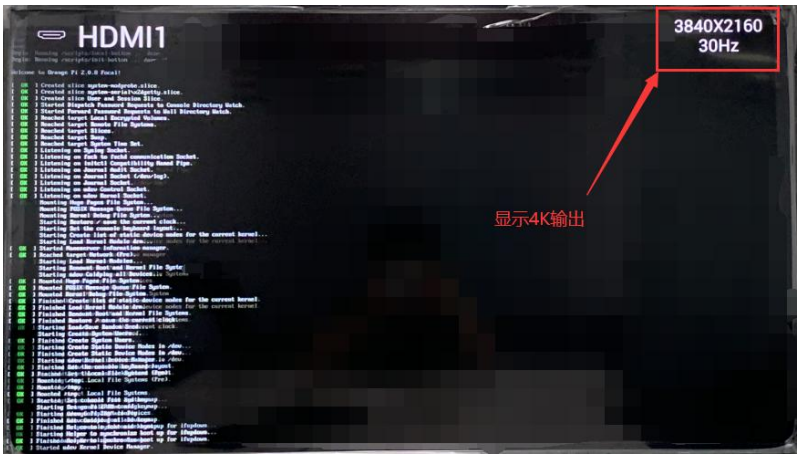
480p	**720x480**	**60**
576p	**720x576**	**60**
720p50	**1280x720**	**50**
720p60	**1280x720**	**60**
1080i50	**1920x1080**	**50**
1080i60	**1920x1080**	**60**
1080p24	**1920x1080**	**24**
1080p50	**1920x1080**	**50**
1080p60	**1920x1080**	**60**
2160p24	**3840x2160**	**24**
2160p25	**3840x2160**	**25**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

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。

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 HMDI 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

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

```
root@orangepi:/sys/class/disp/disp/attr# cat sys
screen 0:
de_rate 696000000 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] umap_skip[0] overflow[0]
       hdmi output mode(29)     fps:25.2        3840x2160
        err:0   skip:1  irq:69215       vsync:0 vsync_skip:0
   BUF    enable ch[1] lyr[0] z[0] prem[N] a[globl 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] transf[0]
```

3. 10. 4. 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=1920、fb0_height=1080

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

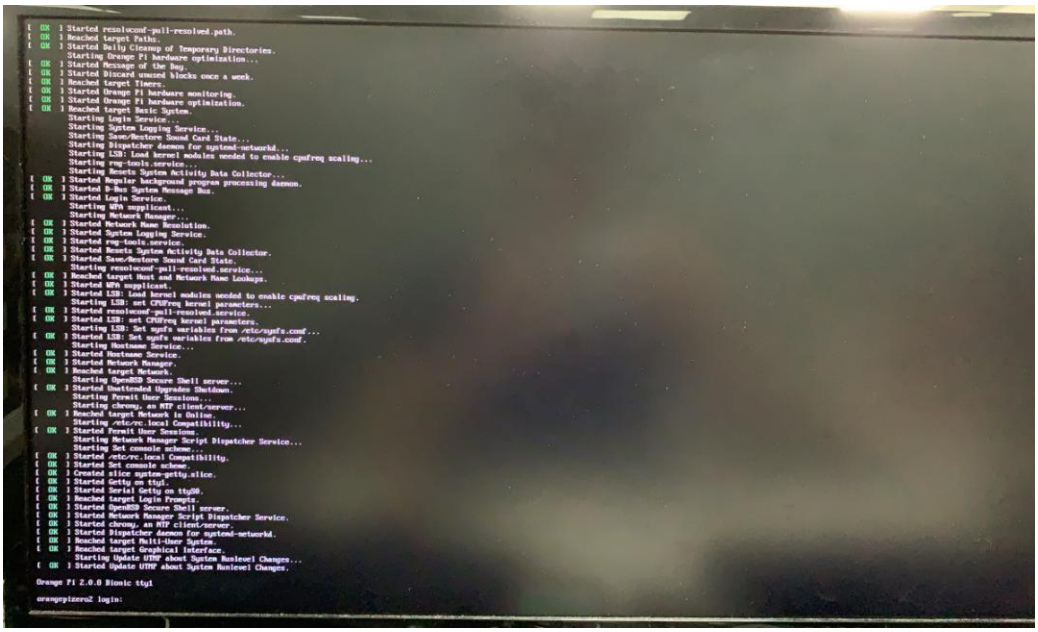
2) fb0_width 和 fb0_height 不同分辨率对应的**参考值**如下所示

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

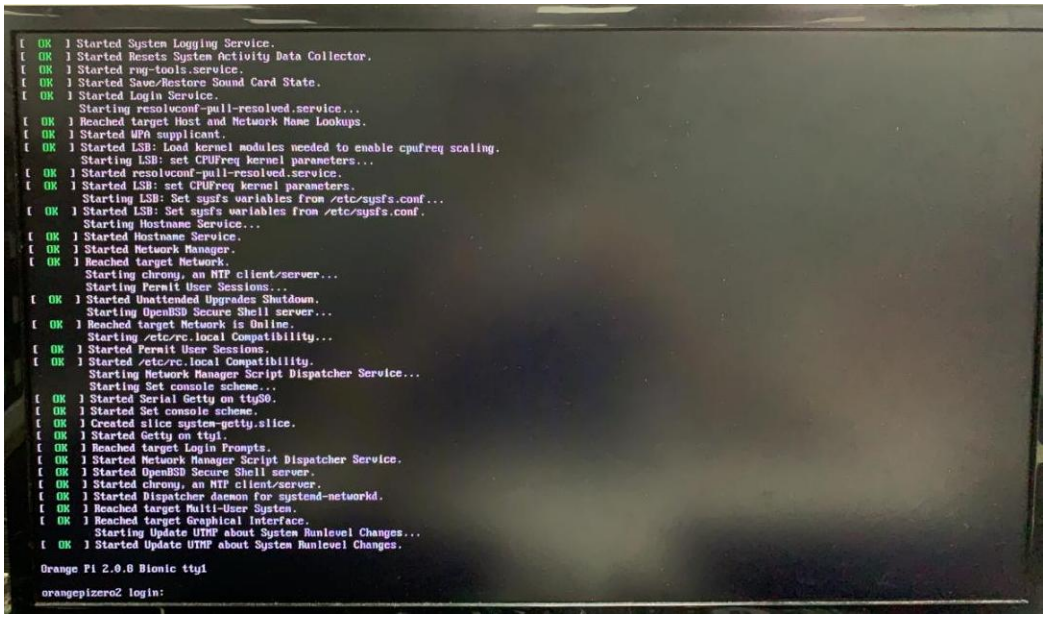
HDMI 分辨率	fb0_width	fb0_height
480p	720	480
576p	720	576
720p	1280	720
1080p	1920	1080
2160p	1920	1080

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

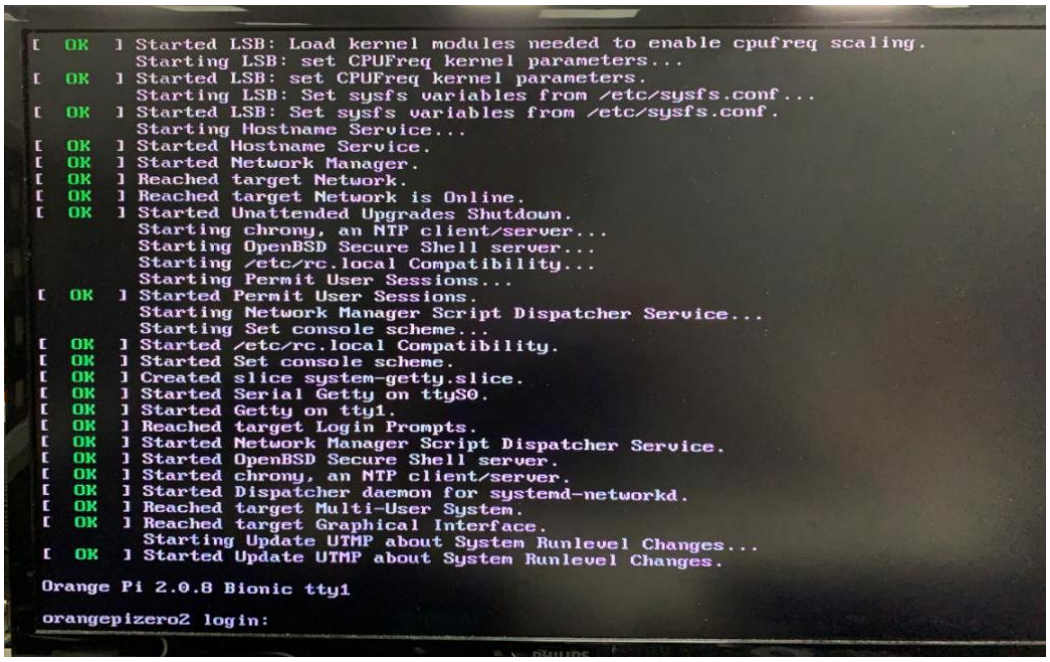
a. HDMI 分辨率为 1080p60，fb0_width 和 fb0_height 为 1920x1080 的显示情况



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



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



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

3.10.5. Framebuffer 光标设置

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

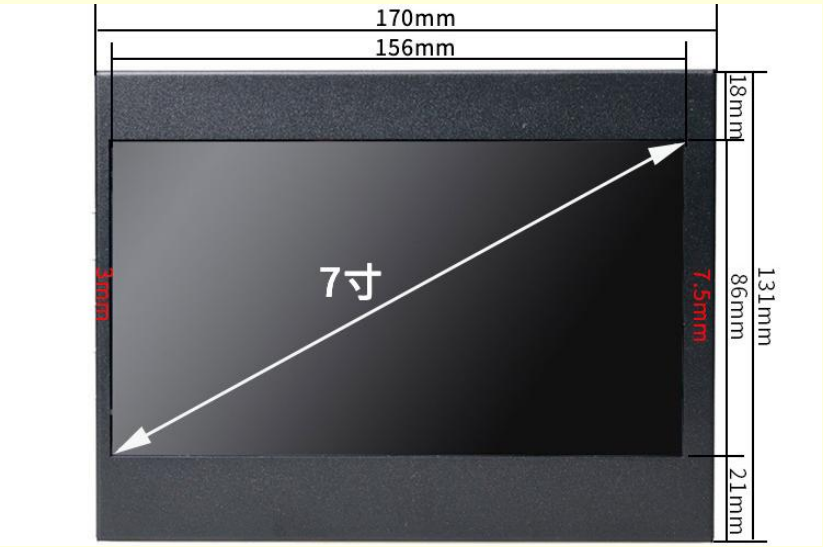
```
root@orangepi:~# echo 1 > /sys/class/graphics/fbcon/cursor_blink    #光标闪烁
root@orangepi:~# echo 0 > /sys/class/graphics/fbcon/cursor_blink    #光标不闪烁
```

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

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

3.10.6. USB 触摸屏的使用方法

此小节测试的触摸屏为在淘宝购买的 7 寸显示屏，分辨率为 1024x600。显示接口为 HDMI 接口，触摸屏控制接口为 USB 接口。



除了上面这款触摸屏外，其他类似的 USB 免驱的触摸屏应该也是可以用的。注意，我这里用的是应该两个字。如果用不了，那就需要自己去适配触摸屏的驱动。

1) 首先请按照购买的屏幕使用说明连接好 HDMI 线和触摸屏的 USB 线

2) 然后修改下 HDMI 的分辨率为 **1024x600**

a. Linux4.9 桌面版和服务器版系统，在**/boot/orangepiEnv.txt** 中修改下面的三个参数即可

注意下面的命令是在 root 用户下执行的。

```
root@orangepi:~# vim /boot/orangepiEnv.txt
disp_mode=720p60
fb0_width=1024
fb0_height=600
```

b. Linux5.16 服务器版本系统需要在**/boot/orangepiEnv.txt** 中添加下面的一行配置

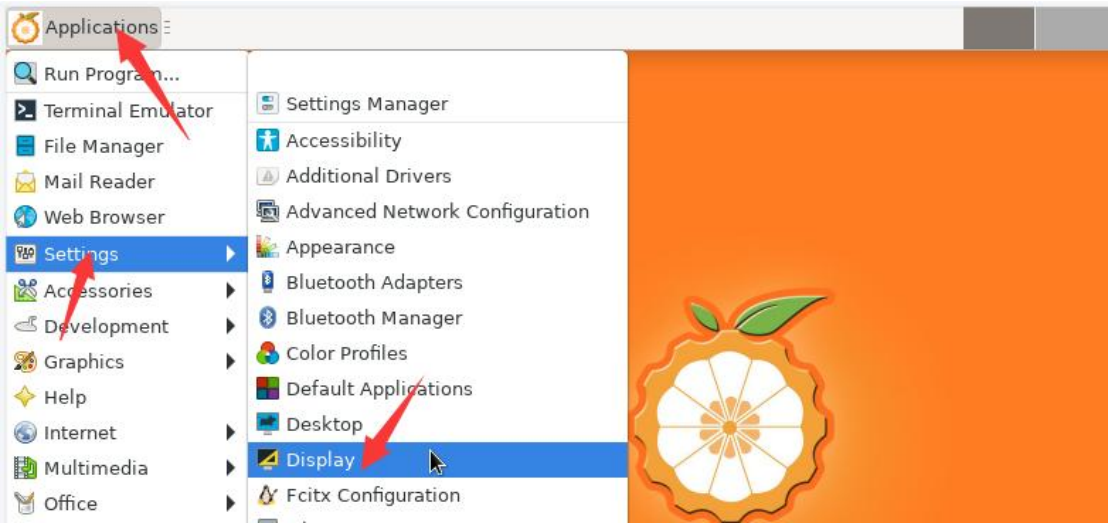
```
root@orangepi:~# vim /boot/orangepiEnv.txt
extraargs=video=HDMI-A-1:1024x600-24@60
```

如果想显示的字体大一点的话，可以设置分辨率为 720x480：

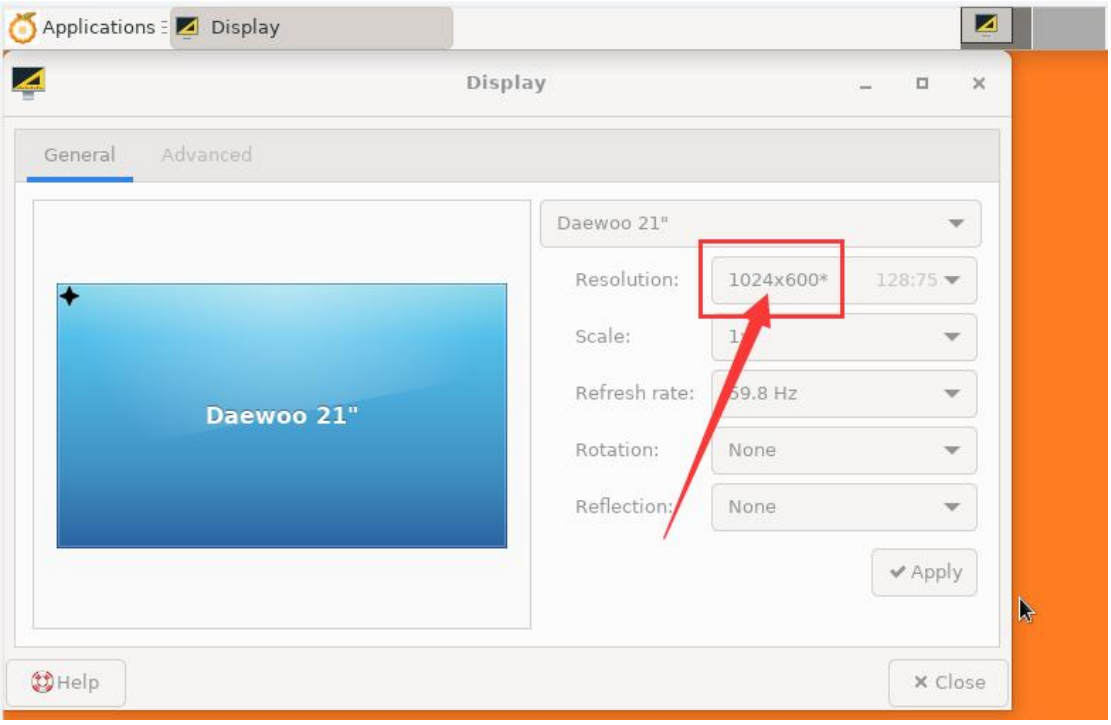
```
extraargs=video=HDMI-A-1:720x480-24@60
```

c. Linux5.16 桌面版本系统一般会自动设置好 **1024x600** 的分辨率，无需修改，使用下面的方法可以查看当前分辨率

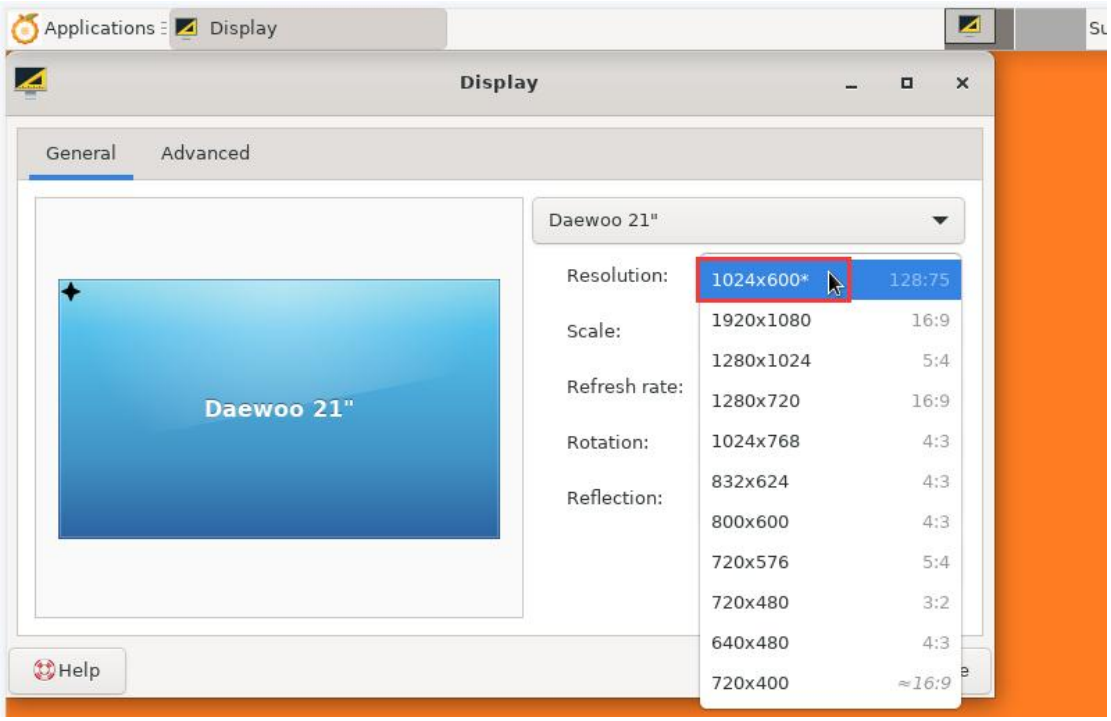
a) 首先打开 **Display**



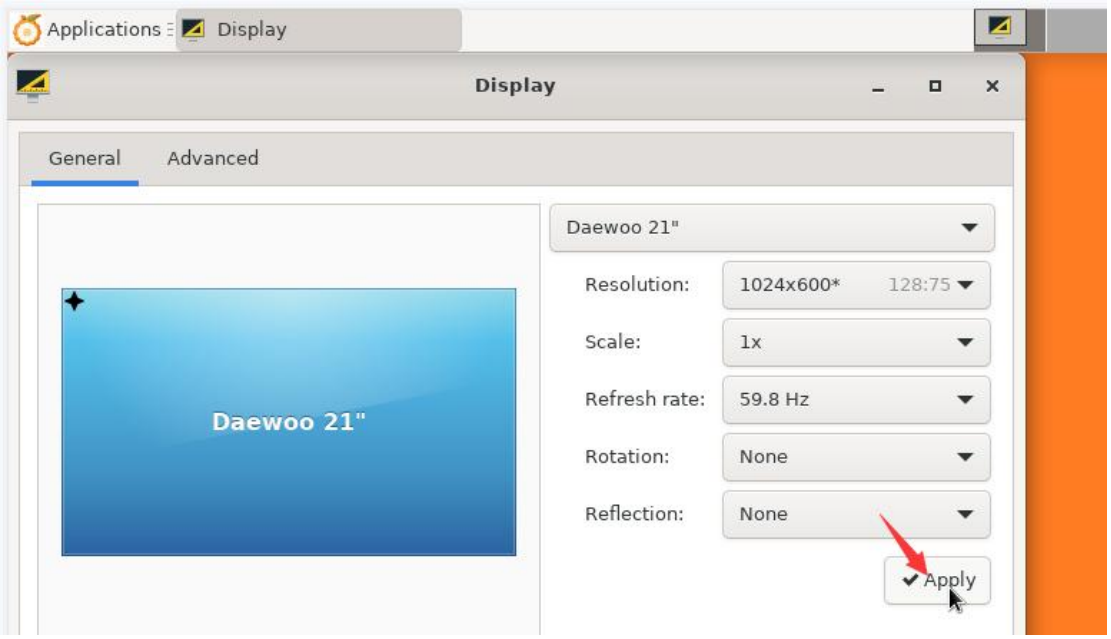
b) 然后可以看到 **Resolution** 中显示的分辨率为 **1024x600**



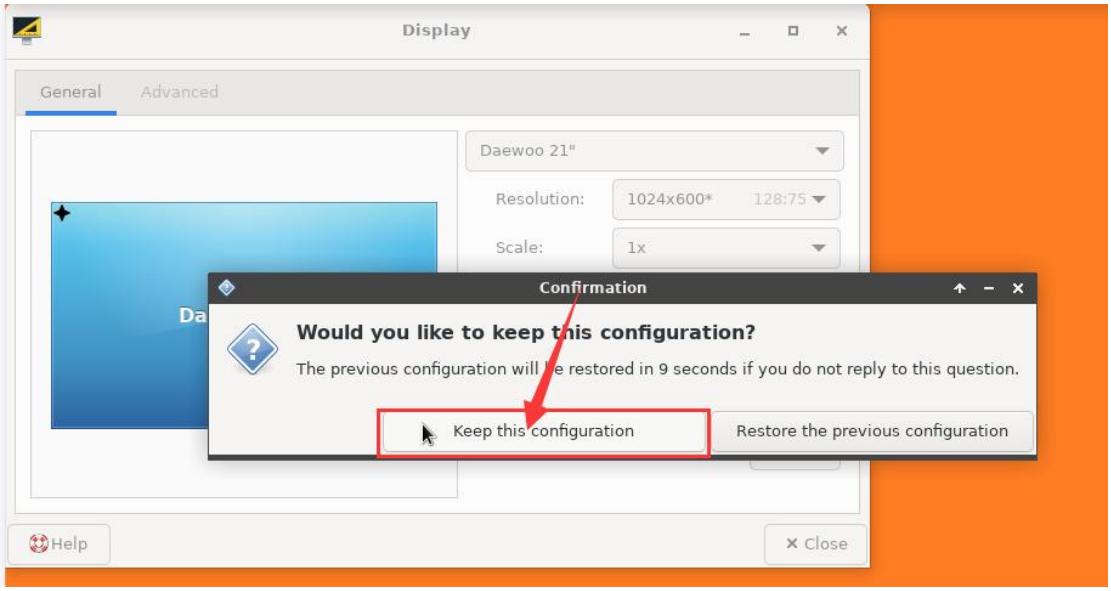
c) 如果显示的分辨率不对，可以在 **Resolution** 中选择 **1024x600** 分辨率



d) 然后点击 **Apply**

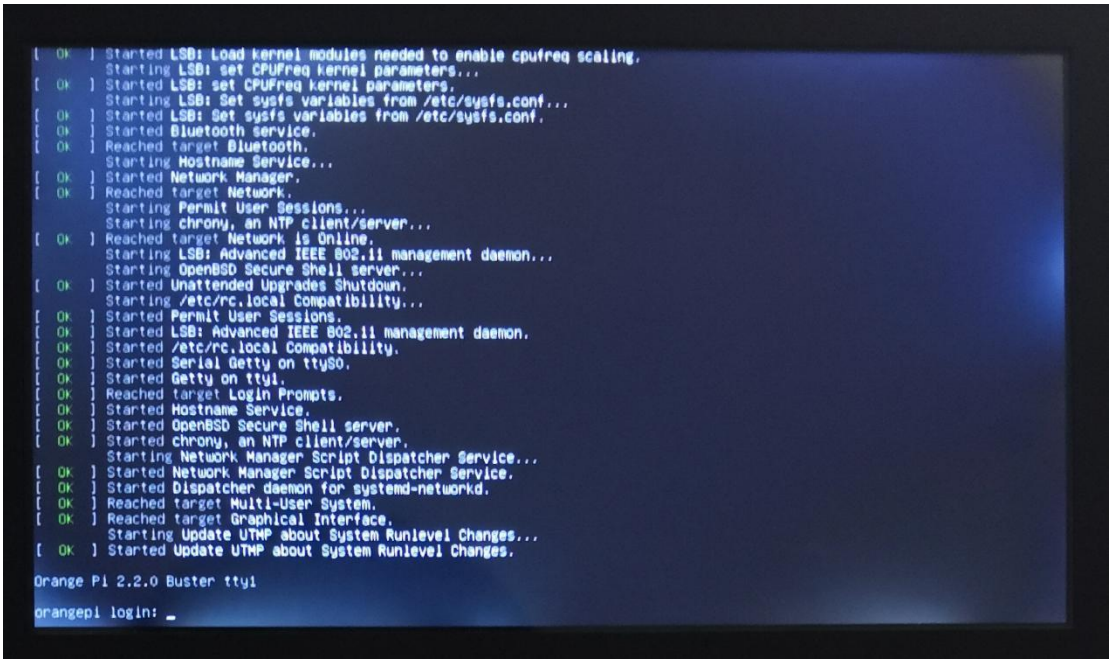


e) 然后选择 **Keep this configuration** 即可



3) HDMI 屏幕显示效果如下所示

a. 服务器版本的 Linux 系统 HDMI 显示效果如下所示



b. 桌面版本的 Linux 系统 HDMI 显示效果如下所示

4) 连接好触摸屏后，使用 **dmesg** 如果能看到类似下面的输出，说明触摸屏识别成功

```
root@orangepi:~# dmesg | grep "hid-multitouch"
[ 2314.985732] hid-multitouch 0003:1A86:E5E3.0009: input,hidraw0: USB HID v1.00 Device [wch.cn USB2IIC_CTP_CONTROL] on usb-sunxi-ohci-1/input0
```

5) 使用 **lsusb** 命令可以看到多了一个 USB 设备

```
root@orangepi:~# lsusb
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 004 Device 003: ID 1a86:e5e3 QinHeng Electronics
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

6) 使用 **evtest** 命令可以看到 USB 触摸屏的输入设备为**/dev/input/event3**

USB 触摸屏的输入设备号并不一定都是/dev/input/event3，请以实际看到的为准。

```
root@orangepi:~# evtest
No device specified, trying to scan all of /dev/input/event*
Available devices:
/dev/input/event0:          sunxi-keyboard
/dev/input/event1:          sunxi_ir_recv
/dev/input/event2:          MCE IR Keyboard/Mouse (sunxi-rc-recv)
/dev/input/event3:          wch.cn USB2IIC_CTP_CONTROL
Select the device event number [0-3]:
```

7) 然后选择 USB 触摸屏的输入设备号

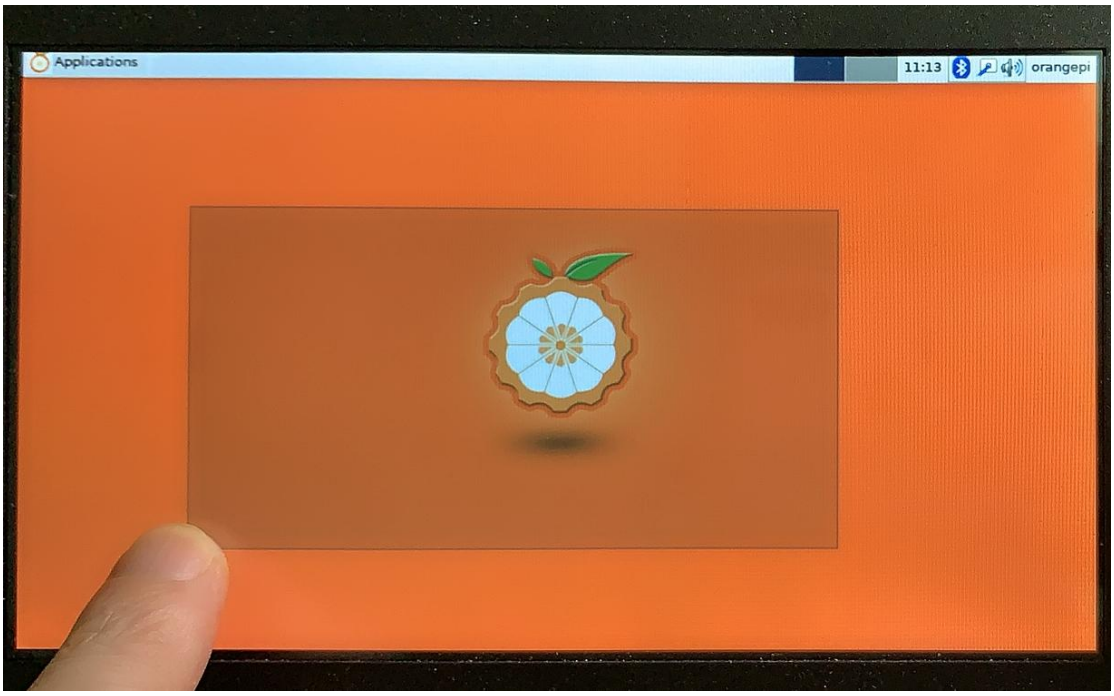
```
root@orangepi:~# evtest
No device specified, trying to scan all of /dev/input/event*
Available devices:
/dev/input/event0:          sunxi-keyboard
/dev/input/event1:          sunxi_ir_recv
/dev/input/event2:          MCE IR Keyboard/Mouse (sunxi-rc-recv)
/dev/input/event3:          wch.cn USB2IIC_CTP_CONTROL
Select the device event number [0-3]: 3        <-在这里输入 USB 触摸屏的输入设备号
```

USB 触摸屏的输入设备号并不一定都是 3，请以实际看到的为准。

8) 然后点击触摸屏就能看到类似下面的输出

```
Event: time 1655030305.127919, type 3 (EV_ABS), code 57 (ABS_MT_TRACKING_ID), value 3
Event: time 1655030305.127919, type 3 (EV_ABS), code 53 (ABS_MT_POSITION_X), value 2647
Event: time 1655030305.127919, type 3 (EV_ABS), code 54 (ABS_MT_POSITION_Y), value 1595
Event: time 1655030305.127919, type 3 (EV_ABS), code 48 (ABS_MT_TOUCH_MAJOR), value 24
Event: time 1655030305.127919, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 1
Event: time 1655030305.127919, type 3 (EV_ABS), code 0 (ABS_X), value 2647
Event: time 1655030305.127919, type 3 (EV_ABS), code 1 (ABS_Y), value 1595
Event: time 1655030305.127919, -------------- SYN_REPORT ------------
Event: time 1655030305.138924, type 3 (EV_ABS), code 57 (ABS_MT_TRACKING_ID), value 3
Event: time 1655030305.138924, type 3 (EV_ABS), code 53 (ABS_MT_POSITION_X), value 2647
Event: time 1655030305.138924, type 3 (EV_ABS), code 54 (ABS_MT_POSITION_Y), value 1595
Event: time 1655030305.138924, type 3 (EV_ABS), code 48 (ABS_MT_TOUCH_MAJOR), value 24
Event: time 1655030305.138924, type 3 (EV_ABS), code 0 (ABS_X), value 2647
Event: time 1655030305.138924, type 3 (EV_ABS), code 1 (ABS_Y), value 1595
Event: time 1655030305.138924, -------------- SYN_REPORT ------------
```

9) 桌面版本系统可以直接在触摸屏中操作

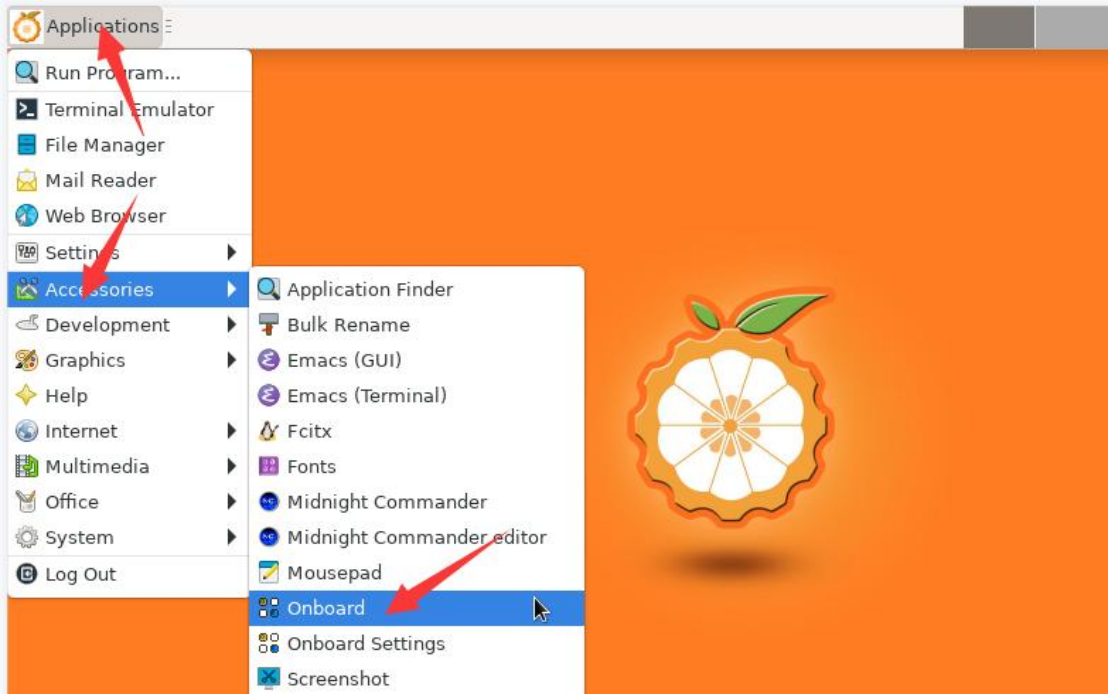


10) 桌面版本系统还可以安装 **Onboard** 这个屏幕键盘软件来辅助触摸屏的操作

a. 安装命令如下所示

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

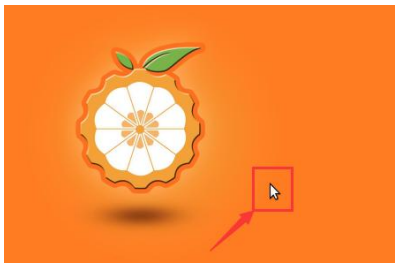
b. 然后就可以打开 **Onbaord**



c. **Onbaord** 打开后的显示如下所示，然后就可以通过触摸屏来输入数字字母等字符了

3.10.7. USB 触摸屏隐藏鼠标光标的方法

1) 使用触摸屏时，如果希望隐藏下图所示的鼠标光标，可以使用本小节介绍的方法



2) 首先打开下面的配置文件，然后在其中添加红色字体部分的配置

```
orangepi@orangepi:~$ sudo vim /etc/lightdm/lightdm.conf.d/11-orangepi.conf
[Seat:*]
user-session=xfce
greeter-show-manual-login=false
greeter-hide-users=false
allow-guest=false
xserver-command=X -bs -core -nocursor
```

3) **然后重启 Linux 系统**

4) 然后再使用触摸屏，就会发现鼠标光标已经不见了

3.11. 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

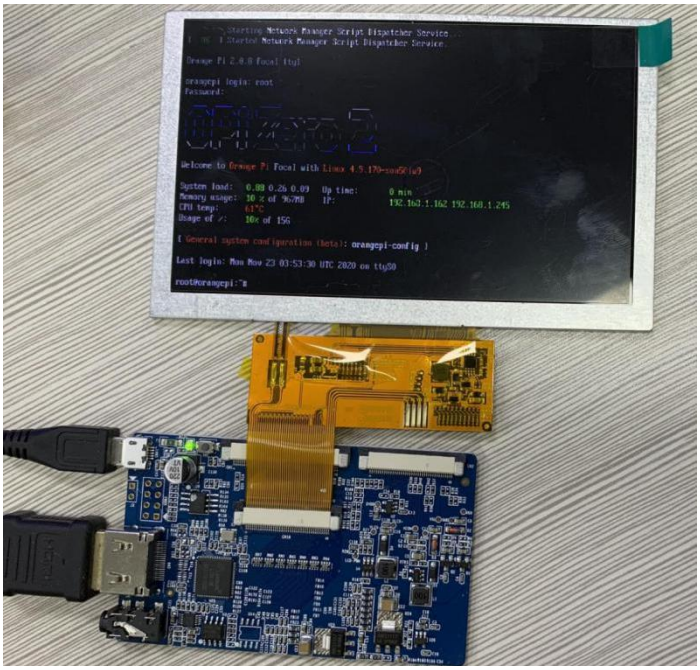
2) 然后使用 Micro HDMI 线将开发板的 Micro HDMI 接口连接到转接板的 HDMI 接口中，屏幕需要单独供电，请在转接板的 Micro USB 接口中插入 5V/2A 的电源

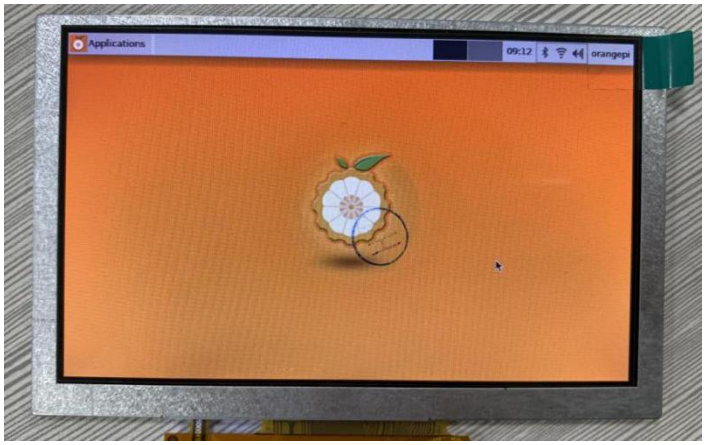
3) 在**/boot/orangepiEnv.txt** 中修改 HDMI 输出分辨率为 480p，Framebuffer 的长度和宽度可以设置为 800x480，如果设置为 1280x720，TFT 液晶屏显示的字体会很小

```
orangepi@orangepi:~$ sudo vim /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





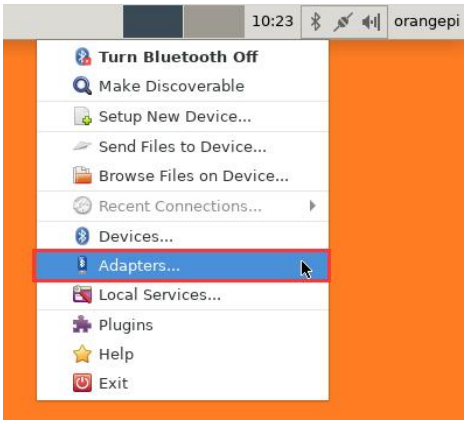
5) 关闭开发板的电源后，请记得同时关闭屏幕转接板的电源，下次启动开发板时再打开

3.12. 蓝牙使用方法

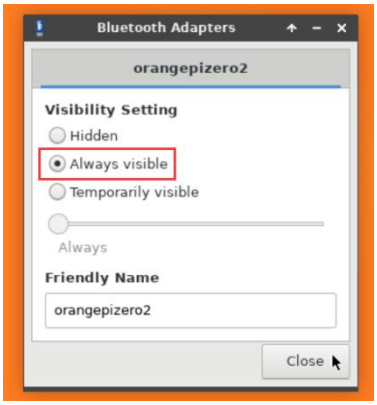
3.12.1. 桌面版镜像的测试方法

1) 点击桌面右上角的蓝牙图标

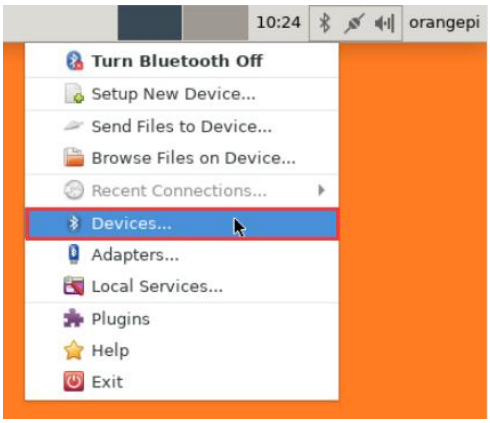
2) 然后选择适配器



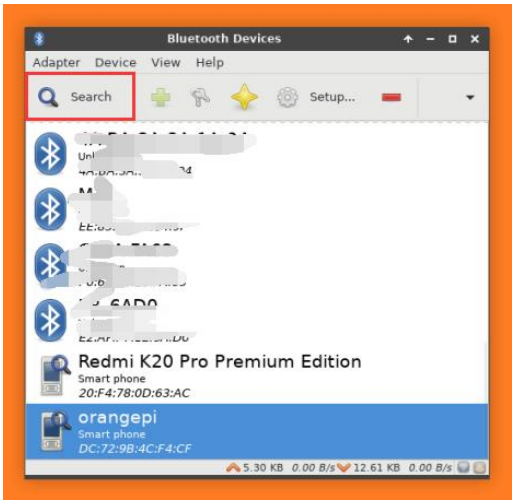
3) 在蓝牙的适配器设置界面中设置 **Visibility Setting** 为 **Always visible**，然后点击 **close** 关闭



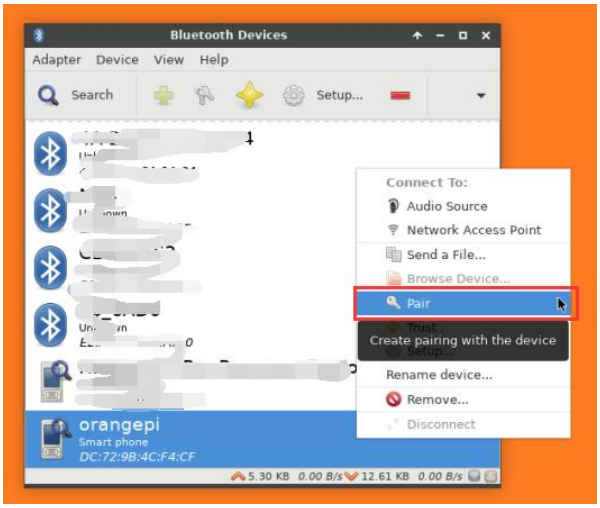
4) 然后打开蓝牙设备的配置界面



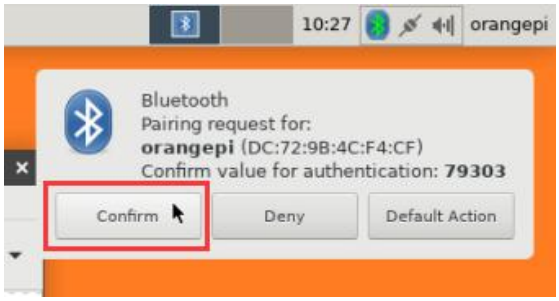
5) 点击 **Search** 即可开始扫描周围的蓝牙设备



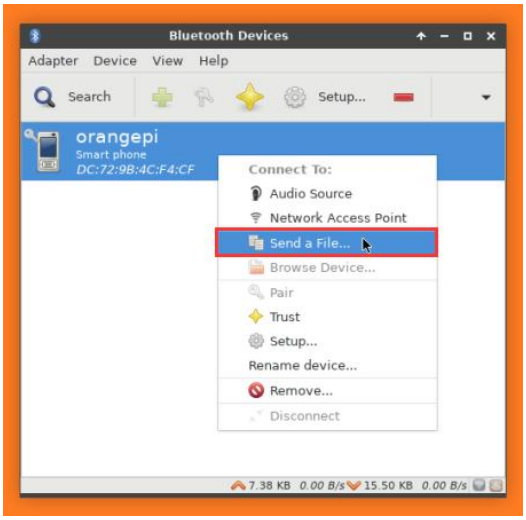
6) 然后选择想要连接的蓝牙设备，再点击鼠标右键就会弹出对此蓝牙设备的操作界面，选择 **Pair** 即可开始配对，这里演示的是和 Android 手机配对



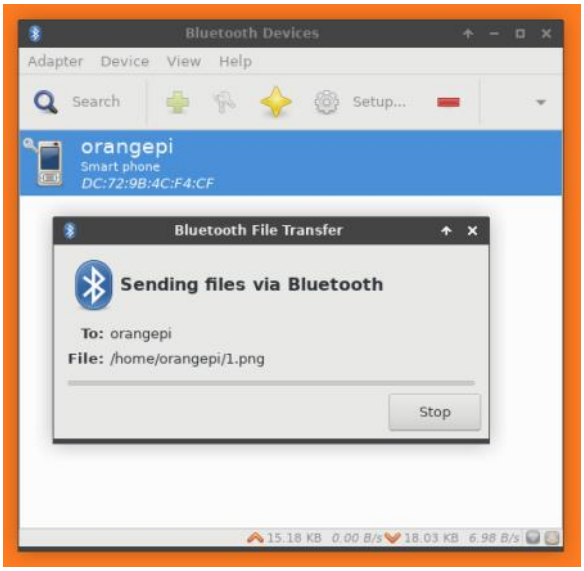
7) 配对时，桌面的右上角会弹出配对确认框，选择 **Confirm** 确认即可，此时手机上也同样需要进行确认



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



3.12.2. 服务器版镜像的使用方法

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

```
orangepi@orangepi:~$ sudo apt update && sudo apt install bluez
orangepi@orangepi:~$ 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: 0xbf 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: 'orangepizero2'
        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

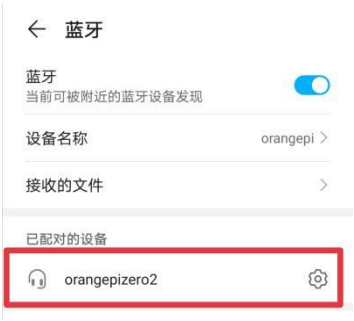
```
orangepi@orangepi:~$ sudo bluetoothctl
[NEW] Controller 10:11:12:13:14:15 orangepizero2 [default]
Agent registered
[bluetooth]# power on          #使能控制器
Changing power on succeeded
[bluetooth]# discoverable on      #设置控制器为可被发现的
Changing discoverable on succeeded
[CHG] Controller 10:11:12:13:14:15 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



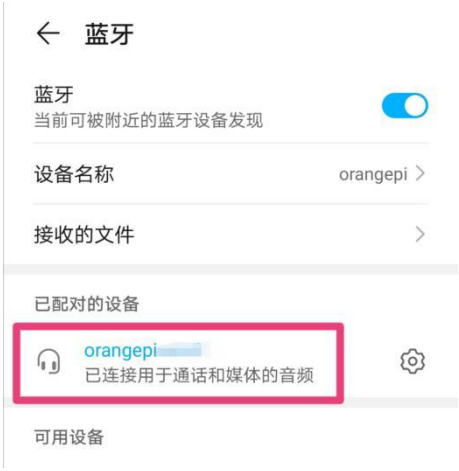
5) 连接蓝牙设备需要安装 **pulseaudio-module-bluetooth** 软件包，然后再启动 **pulseaudio** 服务

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

6) 连接蓝牙设备的方法

```
orangepi@orangepi:~$ sudo bluetoothctl
Agent registered
[bluetooth]# paired-devices          #查看已配对的蓝牙设备的 MAC 地址
Device DC:72:9B:4C:F4:CF orangepi
[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
[orangepi]#                    #出现这个提示符说明连接成功
```

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



3.13. USB 接口测试

USB 接口是可以接 USB hub 来扩展 USB 接口的数量的。

3.13.1. 连接 USB 鼠标或键盘测试

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

2) 连接 Orange Pi 开发板到 HDMI 显示器

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

3.13.2. 连接 USB 存储设备测试

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

2) 执行下面的命令如果能看到 sdX 的输出说明 U 盘识别成功

```
orangepi@orangepi:~$ cat /proc/partitions | grep "sd*"
major minor  #blocks  name
   8        0   30044160 sda
   8        1   30043119 sda1
```

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

```
orangepi@orangepi:~$ sudo mount /dev/sda1 /mnt/
```

```
orangepi@orangepi:~$ ls /mnt/
test.txt
```

Linux 系统挂载 **exfat** 格式的 U 盘可以使用下面的命令

```
orangepi@orangepi:~$ sudo apt-get install exfat-utils exfat-fuse
orangepi@orangepi:~$ sudo mount -t exfat /dev/sda1 /mnt/
```

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

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

3.13.3. USB 麦克风测试

1) 测试的 USB 麦克风如下所示

2) 首先将 USB 麦克风插入开发板的 USB 接口中

3) 然后用 **lsusb** 命令如果能看到下面的输出，说明 USB 麦克风识别正常

```
orangepi@orangepi:~$ lsusb
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 007: ID 08bb:2902 Texas Instruments PCM2902 Audio Codec
```

4) 使用 **arecord -l** 命令可以查看下是否有 USB 麦克风的声卡设备

```
orangepi@orangepi:~$ arecord -l
**** List of CAPTURE Hardware Devices ****
[...省略...]
```

```
card 3: Device [USB PnP Sound Device], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

5) 使用 USB 麦克风录音的命令如下所示，**此模块只支持单通道录音，所以需要指定通道数为 1**

```
orangepi@orangepi:~$ arecord -D hw:3,0 -d 10 -f cd -t wav -c 1 test.wav
```

6) 录完音后，如果播放 **test.wav** 文件有声音就说明 USB 麦克风可以正常使用

3.13.4. 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

3.13.4.1. 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

```
orangepi@orangepi:~$ 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

```
orangepi@orangepi:~$ 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

```
orangepi@orangepi:~$ sudo 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

```
orangepi@orangepi:~$ sudo apt update && sudo apt install bluez
orangepi@orangepi:~$ 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
```

3.13.5. USB 以太网卡测试

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

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

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

```
orangepi@orangepi:~$ 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

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

4) 测试网络连通性的命令如下

```
orangepi@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
```

3.13.6. USB 摄像头测试

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

2) 然后通过 lsmod 命令可以看到内核自动加载了下面的模块

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

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

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

注意 v4l2 中的 l 是小写字母 l，不是数字 1。

另外 video 的序号不一定都是 video0，请以实际看到的为准。

4) 使用 fswebcam 测试 USB 摄像头

a. 安装 fswebcam

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

b. 安装完 fswebcam 后可以使用下面的命令来拍照

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

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

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

```
orangepi@orangepi:~$ scp image.jpg test@192.168.1.55:/home/test（根据实际情况修
```

改 IP 地址和路径）

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

5) 使用 motion 测试 USB 摄像头

a. 安装摄像头测试软件 motion

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

注意，Ubuntu22.04 不用设置这一步。

```
orangepi@orangepi:~$ sudo sed -i  \
"s/start_motion_daemon=no/start_motion_daemon=yes/"  \
/etc/default/motion
```

c. 修改**/etc/motion/motion.conf** 的配置

```
orangepi@orangepi:~$ sudo sed -i  \
"s/stream_localhost on/stream_localhost off/"  \
/etc/motion/motion.conf
```

d. 另外还需确保**/etc/motion/motion.conf** 的 **videodevice** 设置为了 USB 摄像头对应的设备节点

注意，video 的序号不一定都是 video0，请以实际看到的为准。

```
orangepi@orangepi:~$ sudo vim /etc/motion/motion.conf
# Video device (e.g. /dev/video0) to be used for capturing.
videodevice /dev/video0
```

e. 然后运行 motion

```
orangepi@orangepi:~$ sudo motion -b
```

f. 使用 motion 前请先确保 Orange Pi 开发板能正常连接网络，然后通过 **ip addr show** 命令获取开发板的 IP 地址

g. 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

a) Github 的下载地址：

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

b) Gitee 的镜像下载地址为：

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

b. 安装依赖的软件包

a) Ubuntu 系统

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

b) Debian 系统

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

c. 编译安装 mjpg-streamer

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

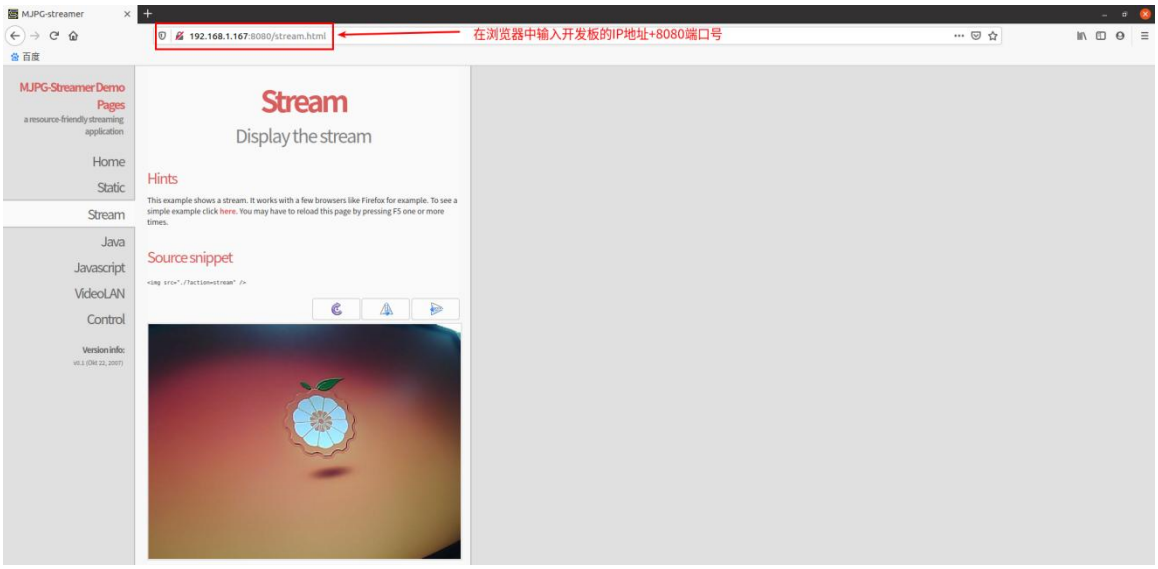
d. 然后输入下面的命令启动 mjpg_streamer

注意，video 的序号不一定都是 video0，请以实际看到的为准。

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

e. 然后在和开发板同一局域网的 Ubuntu PC 或者 Windows PC 或者手机的浏览

器中输入【开发板的 IP 地址:8080】就能看到摄像头输出的视频了



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

3.13.7. 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

```
orangepi@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

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

4) 然后加载 g_ether 内核模块

```
orangepi@orangepi:~$ sudo 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

```
orangepi@orangepi:~$ sudo ifconfig usb0 up
orangepi@orangepi:~$ sudo ifconfig usb0 192.168.10.10
orangepi@orangepi:~$ sudo 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 **enxaafd52849335**

```
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 enxaafd52849335: 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 enxaafd52849335 192.168.10.12
test@test:~$ ifconfig enxaafd52849335
enxaafd52849335:  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

```
orangepi@orangepi:~$ sudo 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
```

3.13.8. 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

```
orangepi@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

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

4) 然后加载 g_serial 内核模块

```
orangepi@orangepi:~$ sudo 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

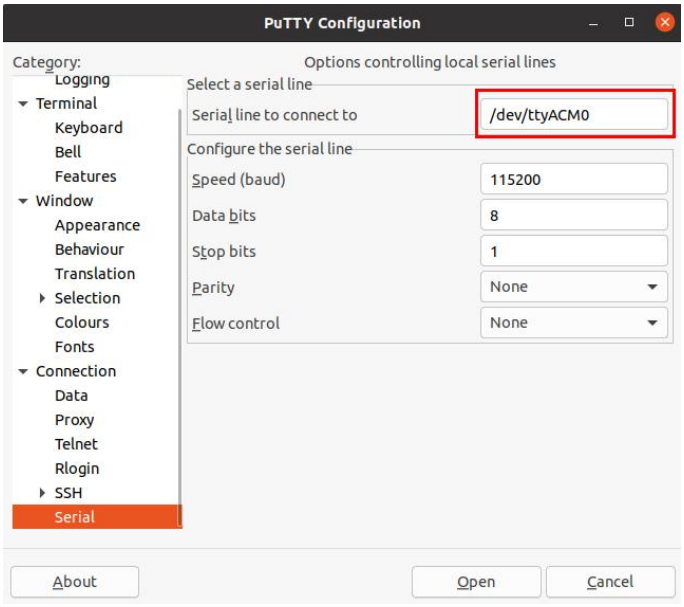
```
orangepi@orangepi:~$ ls /dev/ttyGS*
/dev/ttyGS0
```

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

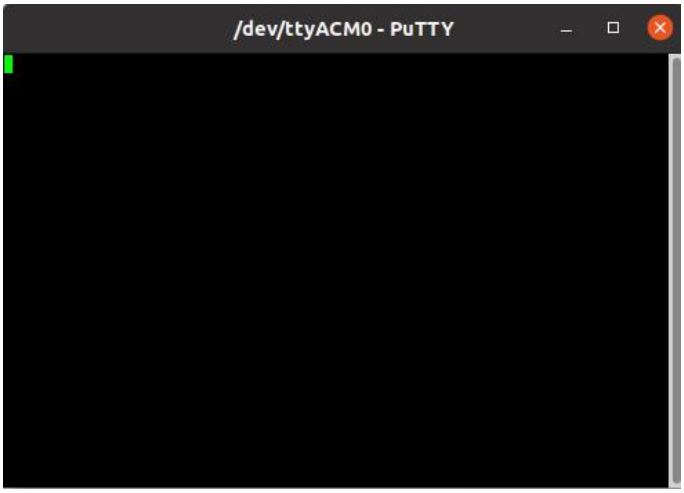
```
test@test:~$ ls /dev/ttyACM*
/dev/ttyACM0
```

7) 然后打开 Ubuntu PC 的 putty，连接 ttyACM0

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



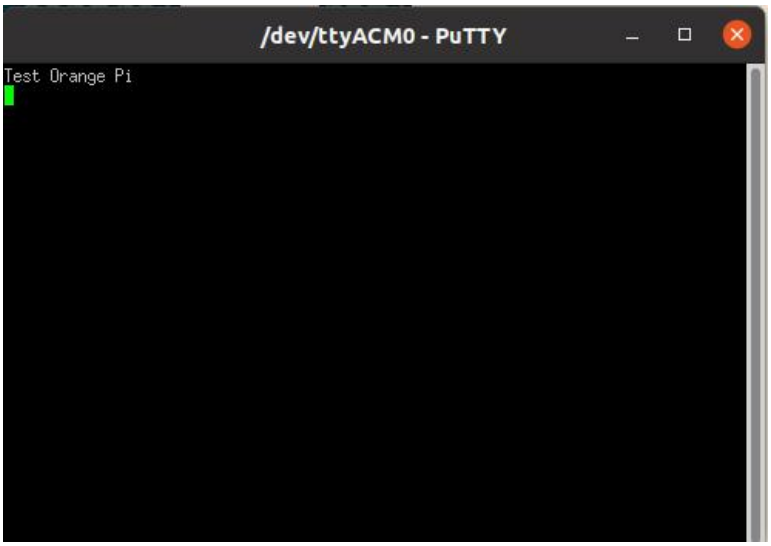
Putty 连接好 ttyACM0 后会打开下图所示的窗口



8) 然后回到开发板的 linux 系统，给/dev/ttyGS0 发送一串字符

```
orangepi@orangepi:~$ sudo echo "Test Orange Pi" > /dev/ttyGS0
```

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

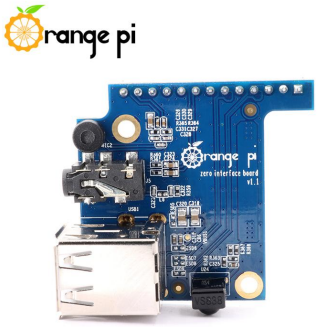


3.14. 音频测试

3.14.1. 使用命令行播放音频的方法

3.14.1.1. 耳机接口播放音频测试

1) 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. linux5.16 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) 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 /usr/share/sounds/alsa/audio.wav
Playing WAVE 'audio.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
```

b. Linux5.16 的播放命令如下所示

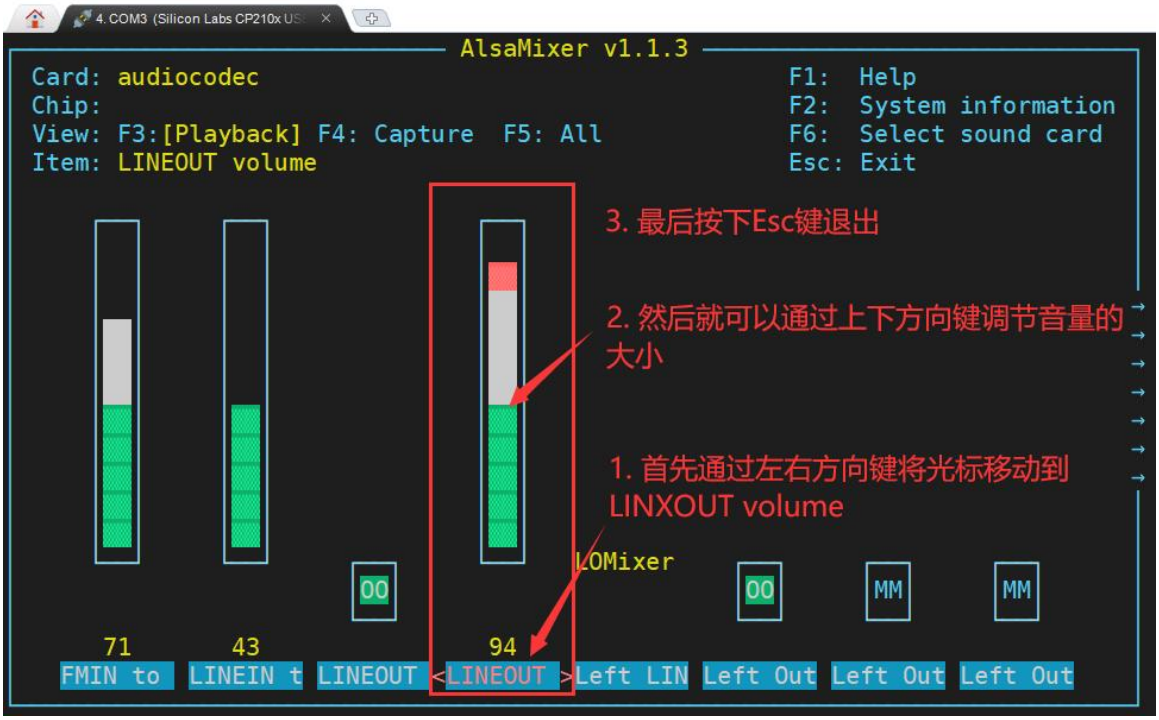
```
root@orangepi:~# aplay -D hw:2,0 /usr/share/sounds/alsa/audio.wav
Playing WAVE 'audio.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
```

4) 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



3.14.1.2. 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) 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@orangepi:~# aplay -D hw:1,0 /usr/share/sounds/alsa/audio.wav
```

b. Linux5.16 系统的播放命令如下所示

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

Linux5.16 系统烧录完第一次启动后，如果测试 HDMI 没有声音，可以使用下面的命令初始化下 HDMI 音频设备，然后再使用上面的命令测试：

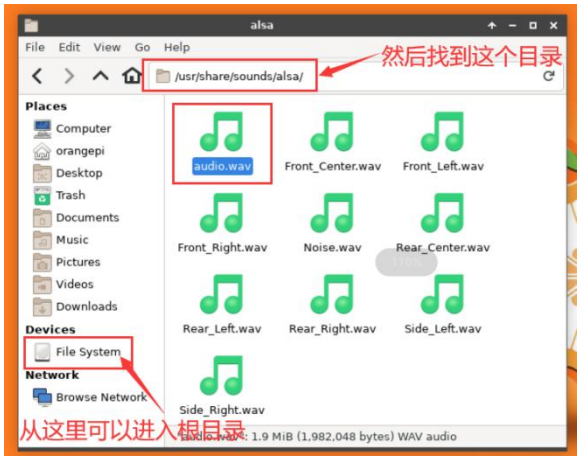
```
root@orangepi:~# aplay /usr/share/sounds/alsa/audio.wav -D hw:1,0 >/dev/null 2>&1
```

或者重启下系统再测试。

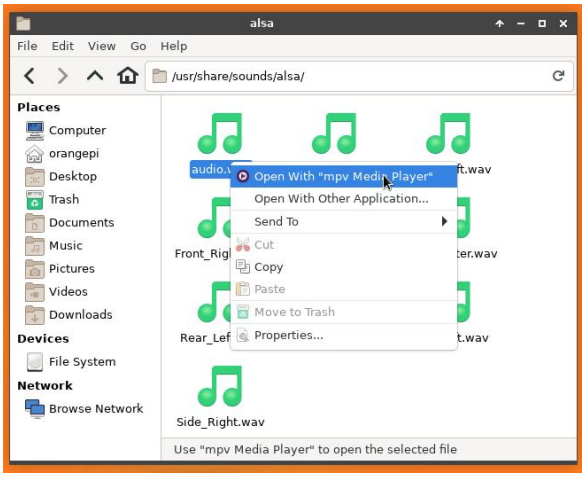
3.14.2. 在桌面系统中测试音频方法

1) 首先打开文件管理器

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

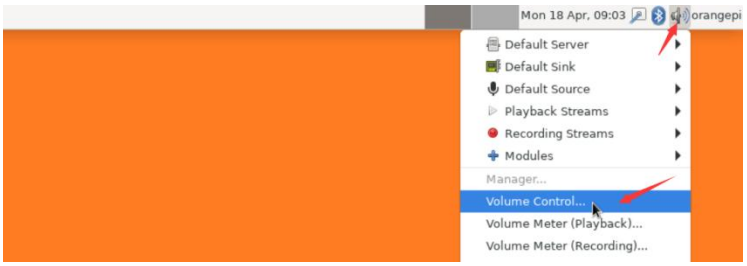


3) 然后选中 audio.wav 文件，右键选择使用 mpv 打开就可以开始播放

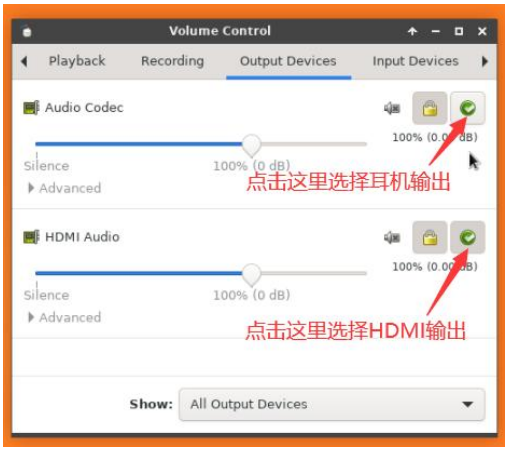


4) 切换 HDMI 播放和耳机播放的方法

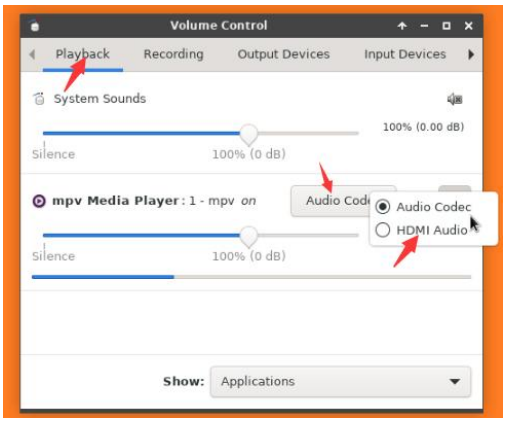
a. 首先打开音量控制界面



b. 选择 **Output Devices**，然后就可以选择输出的音频设备了

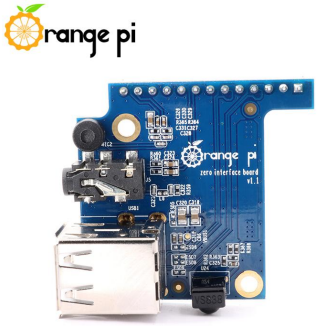


c. 另外，播放音频的时候，在 **Playback** 中会显示播放软件的音频设置选项，如下图所示，也可以在这里来设置需要播放到哪个音频设备



3.15. 红外接收测试

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



2) 安装 ir-keytable 红外测试软件

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

3) 然后执行 ir-keytable 可以查看红外设备的信息

a. linux4.9 系统输出如下所示

```
orangepi@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
```

b. linux5.16 系统的输出如下所示

```
orangepi@orangepi:~$ ir-keytable
Found /sys/class/rc/rc0/ with:
        Name: sunxi-ir
        Driver: sunxi-ir
        Default keymap: rc-empty
        Input device: /dev/input/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@orangepi:~# 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. linux5.16 系统输出如下所示

```
root@orangepi:~# ir-keytable -c -p NEC -t
Old keytable cleared
Protocols changed to nec
Testing events. Please, press CTRL-C to abort.
202.063219: lirc protocol(nec): scancode = 0x45c
202.063249: event type EV_MSC(0x04): scancode = 0x45c
202.063249: event type EV_SYN(0x00).
```

3.16. 温度传感器

1) H616 总共有 4 个温度传感器，查看温度的命令如下所示：

显示的温度值需要除以 1000，单位才是摄氏度。

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

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

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

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

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

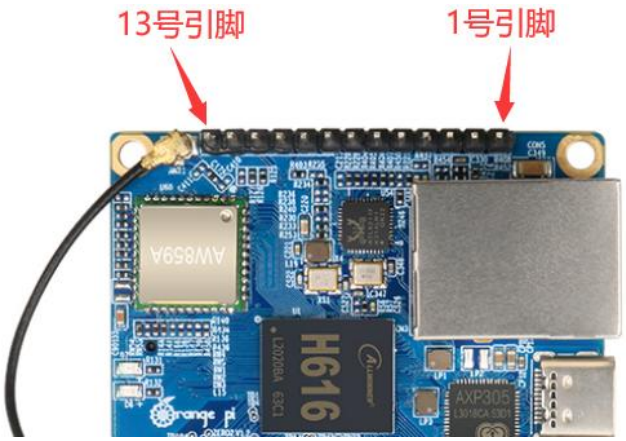
```
orangepi@orangepi:~$ cat /sys/class/thermal/thermal_zone2/type
ve_thermal_zone
orangepi@orangepi:~$ cat /sys/class/thermal/thermal_zone2/temp
59273
```

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

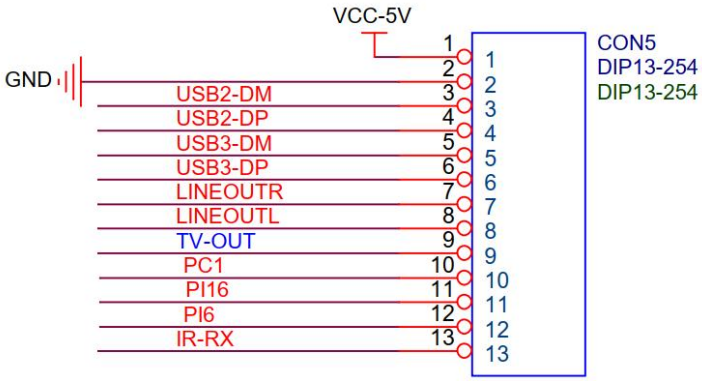
```
orangepi@orangepi:~$ cat /sys/class/thermal/thermal_zone3/type
ddr_thermal_zone
orangepi@orangepi:~$ cat /sys/class/thermal/thermal_zone3/temp
58949
```

3.17. 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

a. 13pin 引脚接扩展板时，可以额外提供

a) 2 个 USB 2.0 Host

b) 耳机左右声道音频输出

c) TV-OUT 视频输出

d) 红外接收功能

e) 接了扩展板后 13pin 接口的 10、11 和 12 号引脚就无法使用了

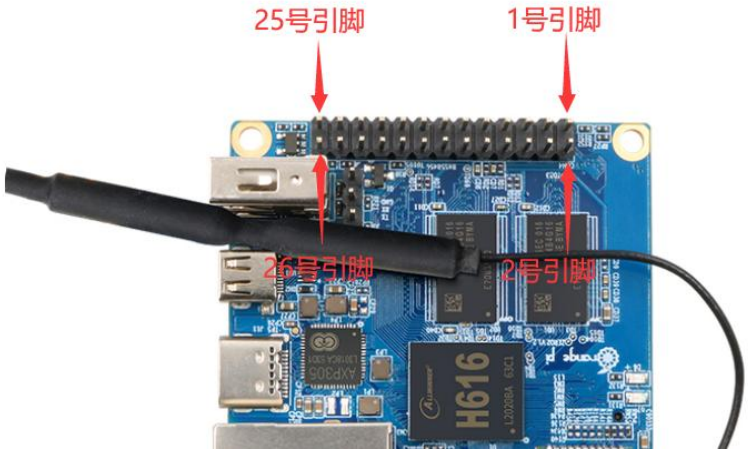
f) **另外需要注意 13pin 扩展板上的 MIC 在 Orange Pi Zero 2 上是无法使用的**

b. 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

3.18. 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
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

引脚	功能	GPIO	GPIO序号
2	5V		
4	5V		
6	GND		
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) 26pin 接口中总共有 **17** 个 GPIO 口，所有 GPIO 口的电压都是 **3.3v**

3.19. 安装 wiringOP 的方法

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。

1) 下载 wiringOP 的代码

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

2) 编译安装 wiringOP

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

3) 测试 gpio readall 命令的输出如下

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
 +------+-----+----------+------+---+  Zero 2  +---+------+----------+-----+------+
 | GPIO | wPi |   Name   | Mode | V | Physical | V | Mode | Name     | wPi | GPIO |
 +------+-----+----------+------+---+----++----+---+------+----------+-----+------+
 |      |     |     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  +---+------+----------+-----+------+
```

3.20. 26pin 接口 GPIO、I2C、UART、SPI 和 PWM 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。

注意，如果需要设置 overlays 同时打开多个配置，请像下面这样使用空格隔开写在一行即可。

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

overlays=spi-spidev i2c3 uart5 pwm12 pwm34

param_spidev_spi_bus=1

param_spidev_spi_cs=1

3.20.1. 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
 +------+-----+----------+------+---+  Zero 2  +---+------+----------+-----+------+
 | GPIO | wPi |   Name   | Mode | V | Physical | V | Mode | Name     | wPi | GPIO |
 +------+-----+----------+------+---++----+----++---+------+----------+-----+------+
 |      |     |     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
 +------+-----+----------+------+---+  Zero 2  +---+------+----------+-----+------+
 | GPIO | wPi |   Name   | Mode | V | Physical | V | Mode | Name     | wPi | GPIO |
 +------+-----+----------+------+---++----+----++---+------+----------+-----+------+
 |      |     |     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 | ALT2 | TXD.5    | 3   | 226  |
 |      |     |      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
```

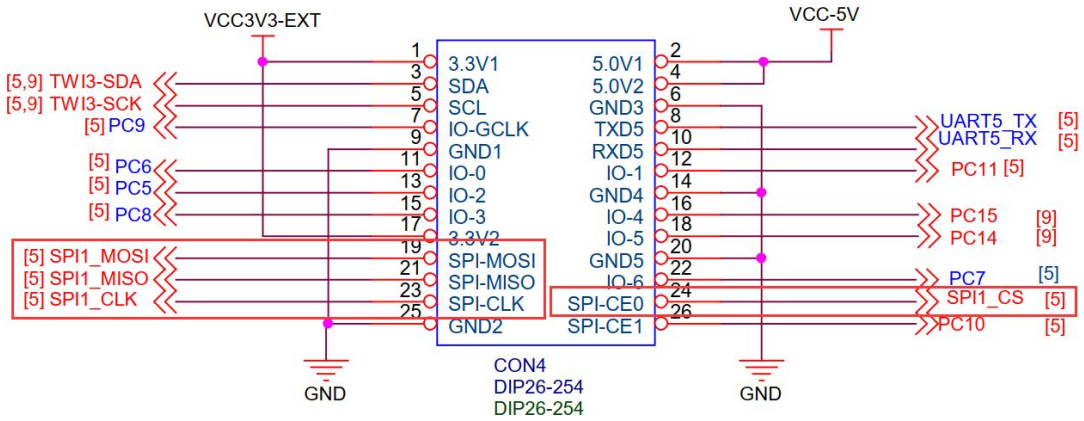
使用 gpio readall 可以看到 7 号引脚的值(V)变为了 1

```
root@orangepizero2:~# gpio readall
 +------+-----+----------+------+---+  Zero 2  +---+------+----------+-----+------+
 | GPIO | wPi |   Name   | Mode | V | Physical | V | Mode | Name     | wPi | GPIO |
 +------+-----+----------+------+---+----++----+---+------+----------+-----+------+
 |      |     |     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 | 1 |  7 || 8  | 0 | ALT2 | TXD.5    | 3   | 226  |
 |      |     |      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. 20. 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



如果使用的为 Linux5.16 内核的系统，spi1 默认是关闭的，需要手动打开才能使用。

在/boot/orangepiEnv.txt 中加入下面红色字体部分的配置，然后重启 Linux 系统就可以打开 spi1。

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

overlays=spi-spidev

param_spidev_spi_bus=1

param_spidev_spi_cs=1

2) 先查看下 linux 系统中是否存在 **spidev1.1** 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/spidev1.1
```

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

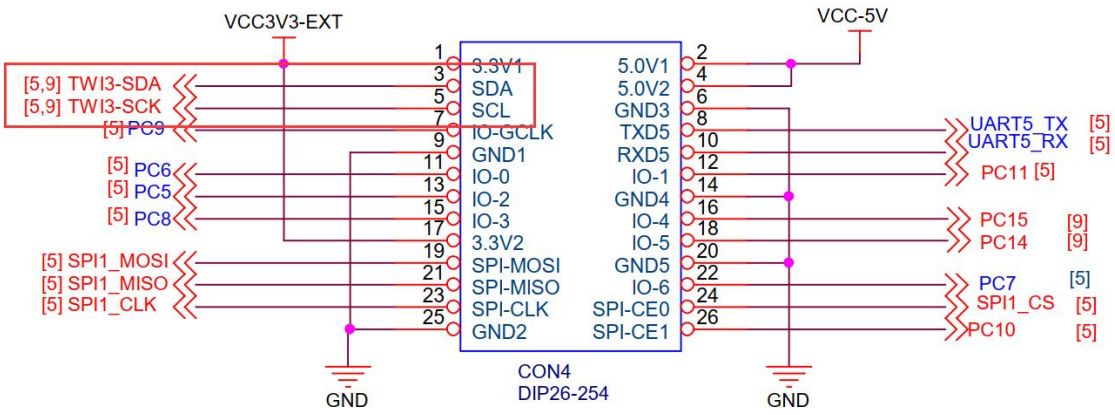
```
root@orangepi:~/wiringOP/examples# ./spidev_test -v -D /dev/spidev1.1
spi mode: 0x0
bits per word: 8
max speed: 500000 Hz (500 KHz)
TX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF F0 0D    | ......@....▒.............▒.
RX | FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF    | ................................
```

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/spidev1.1
spi mode: 0x0
bits per word: 8
max speed: 500000 Hz (500 KHz)
TX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF F0 0D    | ......@....▒.............▒.
RX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF F0 0D    | ......@....▒.............▒.
```

3.20.3. 26pin I2C 测试

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



如果使用的为 Linux5.16 内核的系统，i2c3 默认是关闭的，需要手动打开才能使用。

在/boot/orangepiEnv.txt 中加入下面红色字体部分的配置，然后重启 Linux 系统就可以打开 i2c3。

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

overlays=i2c3

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

```
root@orangepi:~# ls /dev/i2c-*
/dev/i2c-3    /dev/i2c-5
```

3) 然后开始测试 i2c，首先安装 i2c-tools

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

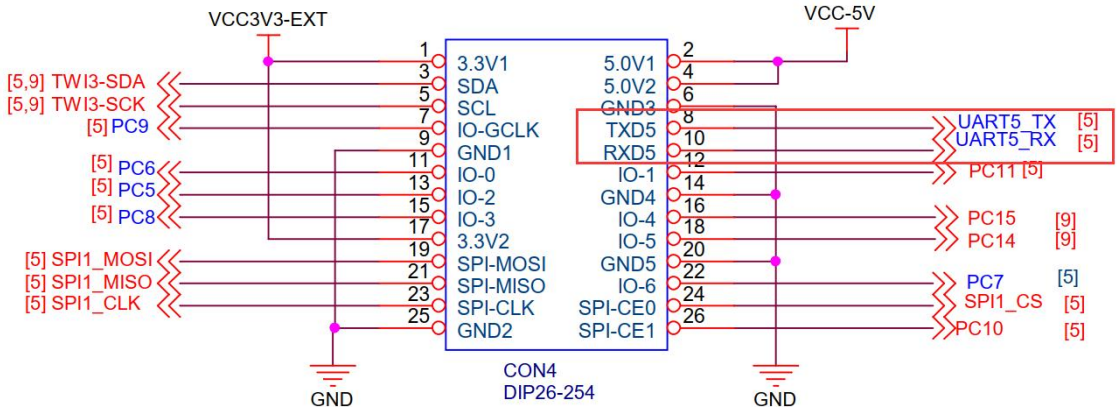
5) 然后使用 **i2cdetect -y 3** 命令如果能检测到连接的 i2c 设备的地址，就说明 i2c 能

正常使用

```
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. 20. 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



如果使用的为 Linux5.16 内核的系统，uart5 默认是关闭的，需要手动打开才能使用。

在/boot/orangepiEnv.txt 中加入下面红色字体部分的配置，然后重启 Linux 系统就可以打开 uart5。

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

overlays=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

	uart5
tx 引脚	对应 8 号引脚
rx 引脚	对应 10 号引脚

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

```
orangepi@orangepi:~$ gpio serial /dev/ttyS5

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

5) 也可以使用 wiringOP 中的 serialTest.c 程序来测试串口的回环功能，具体步骤如下所示：

a. 首先修改 wiringOP 中串口测试程序 serialTest 打开的串口设备节点名为 **/dev/ttyS5**

```
root@orangepi:~/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 ;
  }
```

b. 然后编译 wiringOP 中的串口测试程序 serialTest

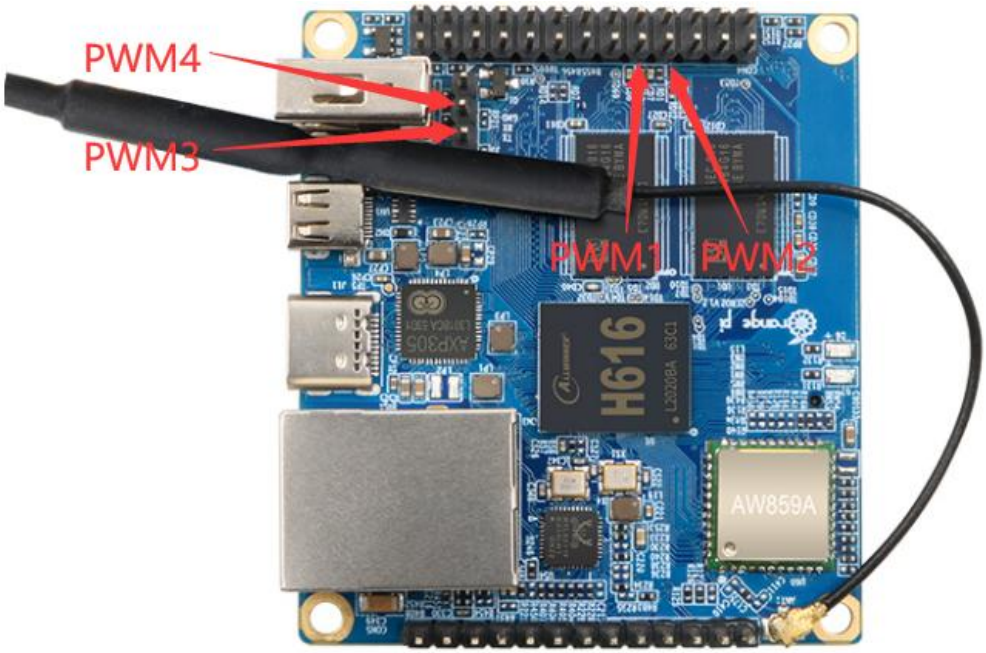
```
root@orangepi:~/wiringOP/examples# make serialTest
[CC] serialTest.c
[link]
root@orangepi:~/wiringOP/examples#
```

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

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

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

3.20.5. PWM 的测试方法

Orange Pi Zero 2 最多可以使用 4 通道 PWM，它们所在引脚的位置如下图所示:



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。

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。

3.20.5.1. Linux5.16 系统 PWM 测试方法

1) PWM1 和 PWM2 测试方法如下所示

a. 首先需要打开 PWM12 的配置，请在**/boot/orangepiEnv.txt** 中加入下面红色字体部分的配置

注意，如果打开了 pwm12，就不能同时打开 uart5，只能二选一。

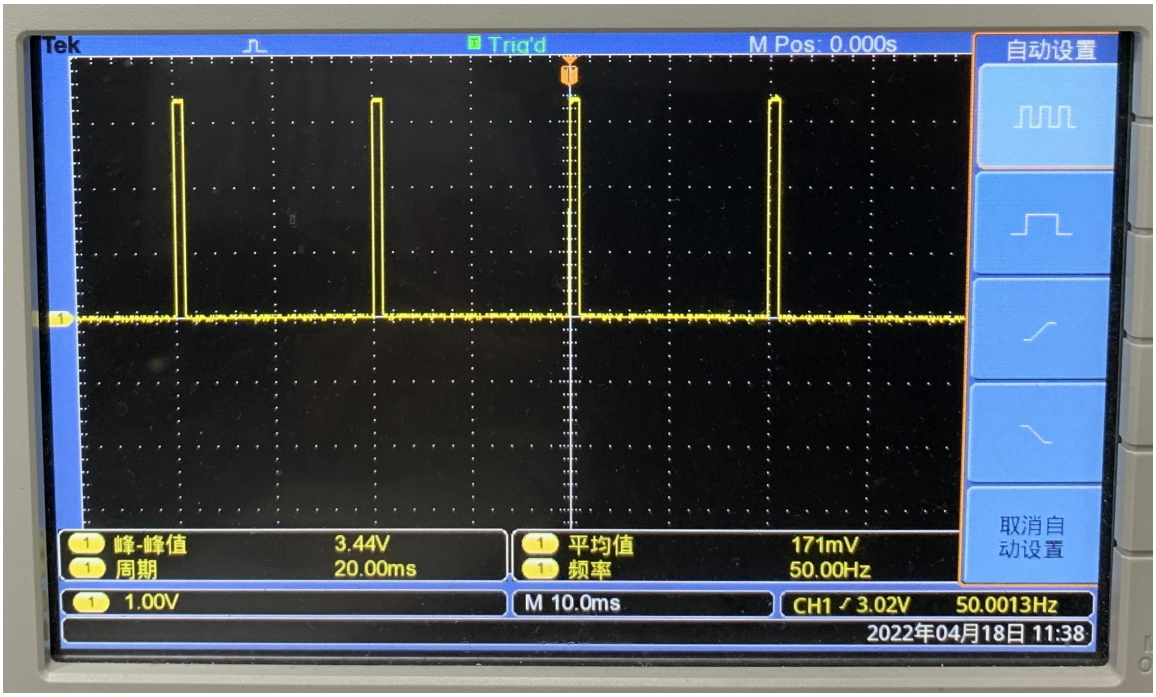
```
orangepi@orangepi:~$ sudo vim /boot/orangepiEnv.txt
overlays=pwm12
```

b. 然后重启 Linux 系统，在串口输出的 log 中如果可以看到下面的打印信息，说明 pwm12 打开成功

```
Applying kernel provided DT overlay sun50i-h616-pwm12.dtbo
4203 bytes read in 5 ms (820.3 KiB/s)
Applying kernel provided DT fixup script (sun50i-h616-fixup.scr)
```

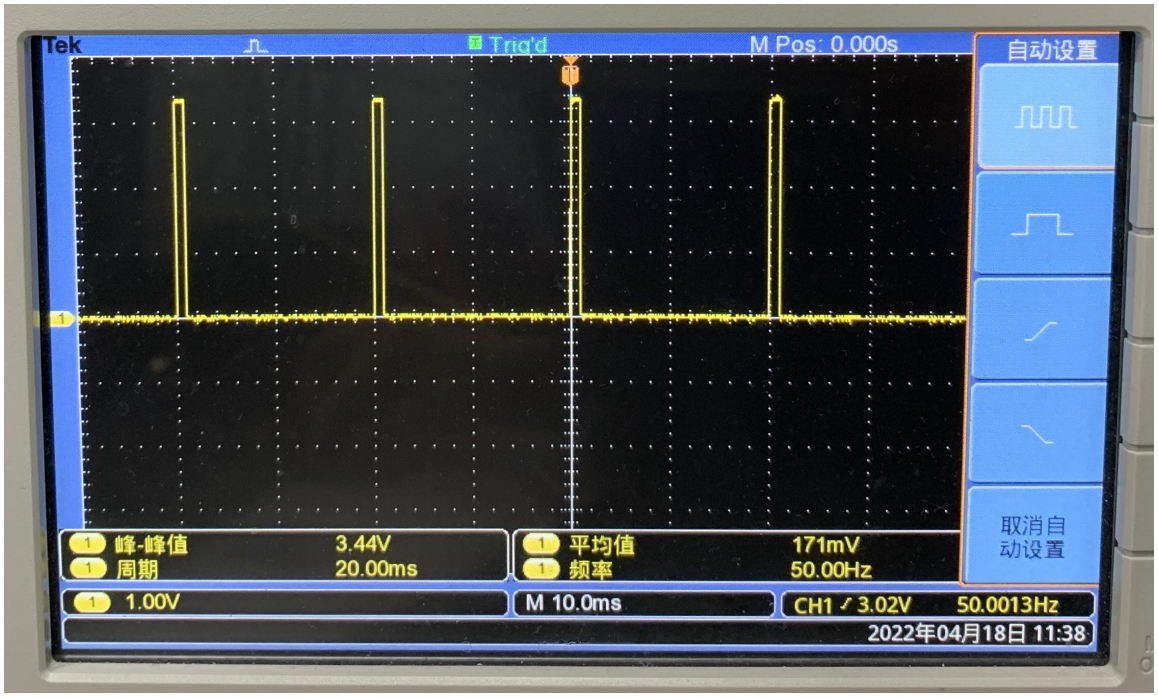
c. 然后就可以开始 PWM 的测试，在系统中输入下面的命令可以让 pwm1 输出一个 50Hz 的矩形波

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



d. 在系统中输入下面的命令可以让 pwm2 输出一个 50Hz 的方波

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



2) PWM3 和 PWM4 测试方法如下所示

a. 首先需要打开 PWM34 的配置，请在**/boot/orangepiEnv.txt** 中加入下面红色字体部分的配置

注意，如果打开了 pwm34，会同时关闭 uart0，此时调试串口就无法使用了。

```
orangepi@orangepi:~$ sudo vim /boot/orangepiEnv.txt
overlays=pwm34
```

b. 然后重启 Linux 系统，在串口输出的 log 中如果可以看到下面的打印信息，说明 pwm34 打开成功

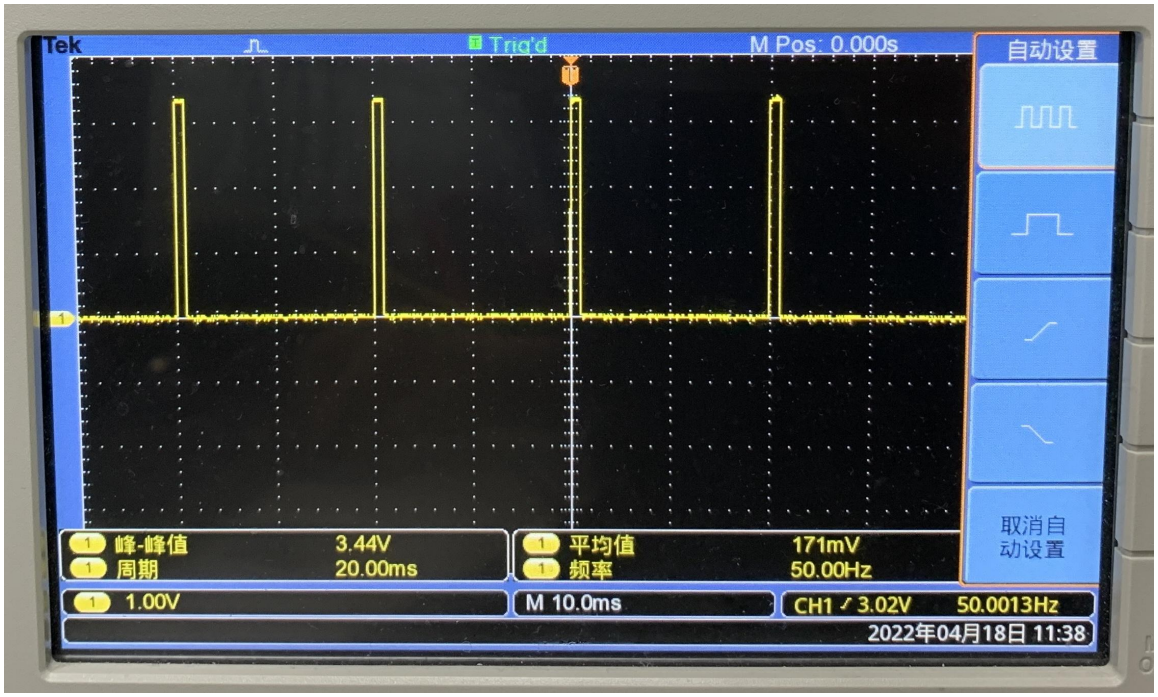
```
Applying kernel provided DT overlay sun50i-h616-pwm34.dtbo
4203 bytes read in 5 ms (820.3 KiB/s)
Applying kernel provided DT fixup script (sun50i-h616-fixup.scr)
## Executing script at 45000000
Warning: Disabling ttyS0 console due to enabled PWM3 and PWM4 overlay
10735700 bytes read in 446 ms (23 MiB/s)
22259720 bytes read in 922 ms (23 MiB/s)
```

c. 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

```
## Flattened Device Tree blob at 4fa00000
   Booting using the fdt blob at 0x4fa00000
   Loading Ramdisk to 495c2000, end 49fff014 ... OK
   Loading Device Tree to 0000000049550000, end 00000000495c1fff ... OK

Starting kernel ...
```

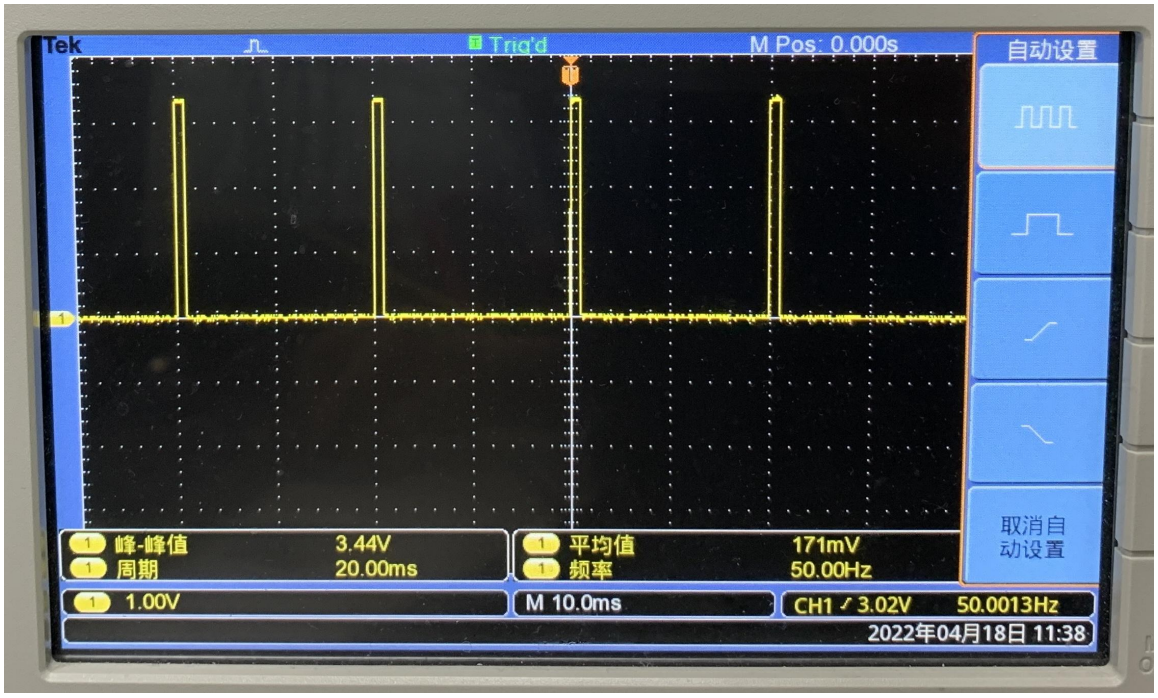
d. 然后就可以开始 PWM 的测试，在系统中输入下面的命令可以让 pwm3 输出一个 50Hz 的矩形波

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



e. 在系统中输入下面的命令可以让 pwm4 输出一个 50Hz 的方波

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



3) 同时打开 PWM12 和 PWM34 方法如下所示：

```
orangepi@orangepi:~$ sudo vim /boot/orangepiEnv.txt
overlays=pwm12 pwm34
```

3.20.5.2. Linux4.9 系统 PWM 测试方法

1) 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 **orangepi-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

```
orangepi@orangepi:~$ vim 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 **orangepi-add-overlay** 将 uart5_disable.dts 编译成 uart5_disable.dtbo

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

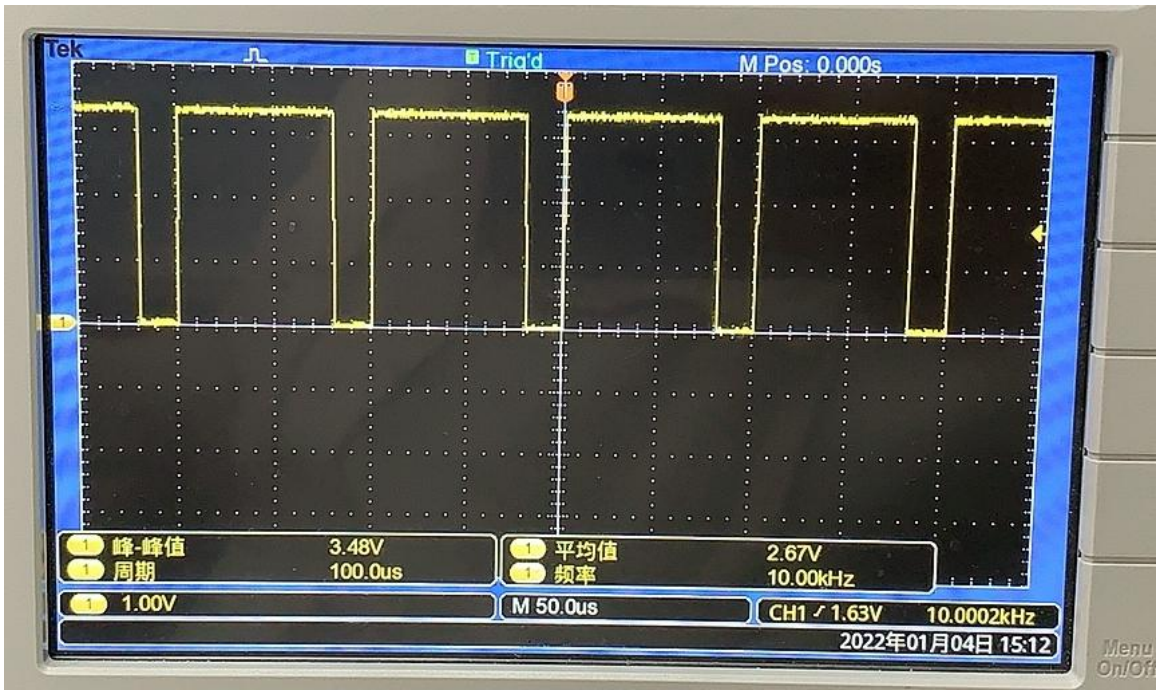
```
216 bytes read in 7 ms (29.3 KiB/s)
283 bytes read in 11 ms (24.4 KiB/s)
Applying user provided DT overlay uart5_disable.dtbo
8472451 bytes read in 367 ms (22 MiB/s)
24125512 bytes read in 1020 ms (22.6 MiB/s)
```

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

```
orangepi@orangepi:~$ ls /dev/ttyS*
/dev/ttyS0   /dev/ttyS1
```

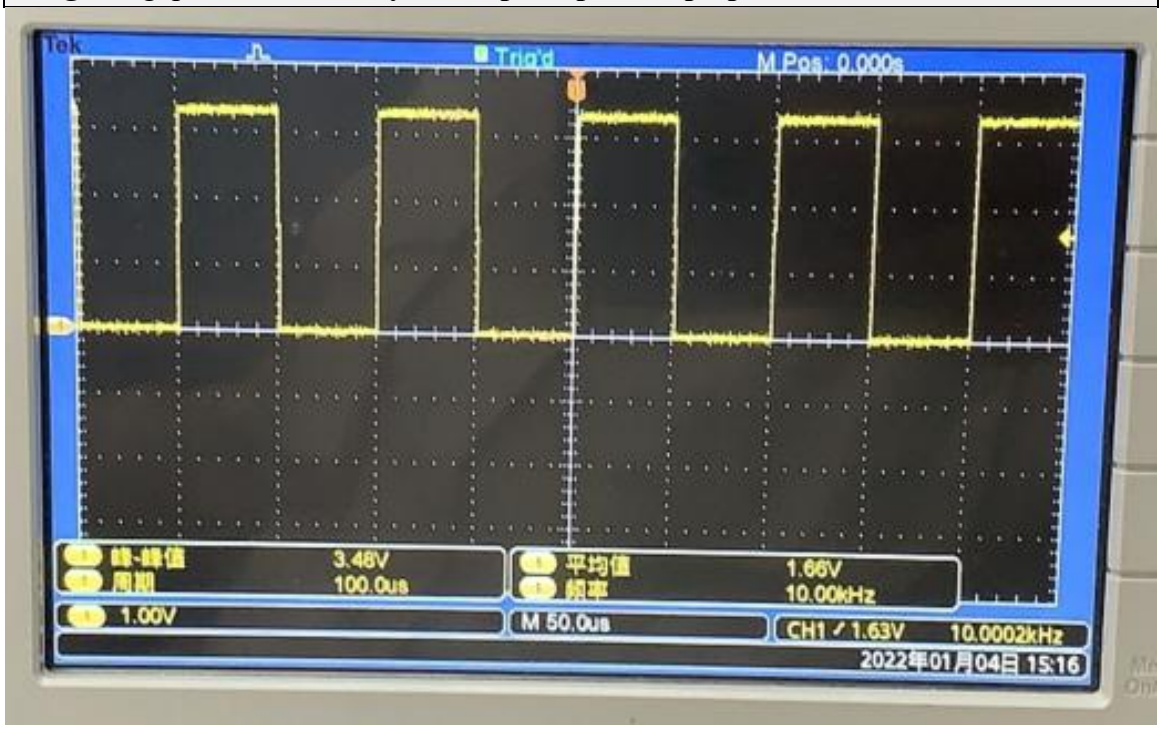
e. 然后就可以开始 PWM 的测试，在 Linux4.9 系统中输入下面的命令可以让 pwm1 输出一个 10kHz 的矩形波

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



2) PWM3 和 PWM4 测试方法如下所示

a. 首先需要在 dts 中关掉 UART0 的配置，linux 系统中预装了一个名为 **orangepi-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

```
orangepi@orangepi:~$ vim uart0_disable.dts
/dts-v1/;
/plugin/;

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

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

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

```
orangepi@orangepi:~$ sudo orangepi-add-overlay uart0_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 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)
```

d. 进入 Linux 系统后在**/dev** 下就看不到 ttyS0 这个设备节点了

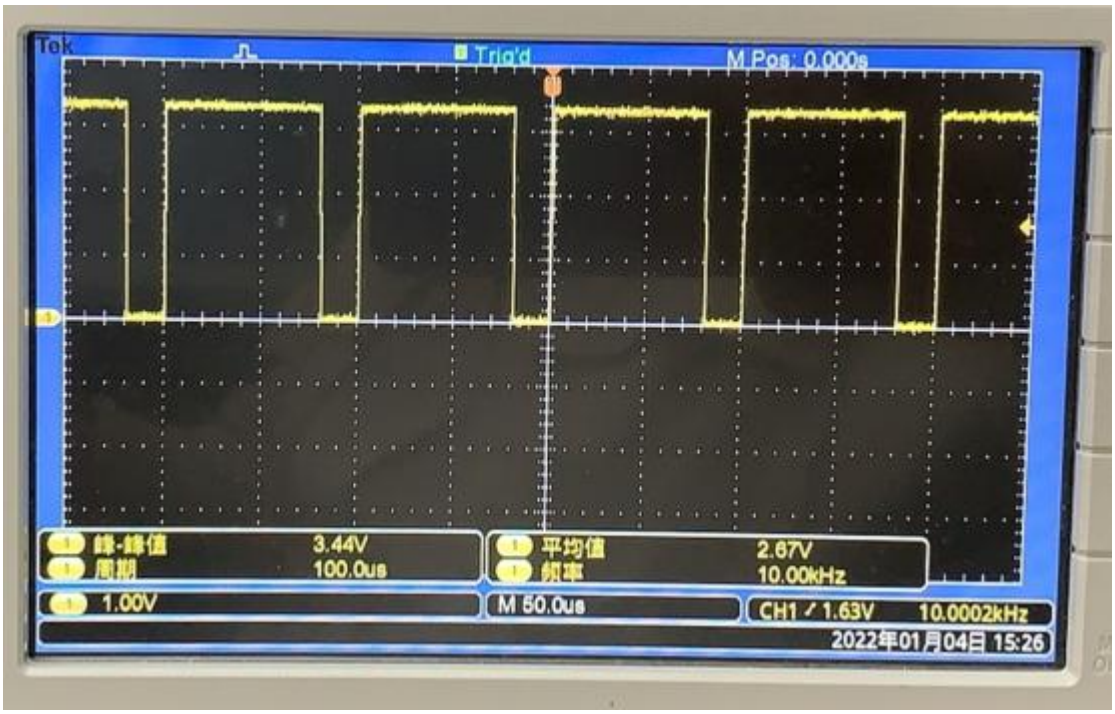
```
orangepi@orangepi:~$ ls /dev/ttyS*
/dev/ttyS1  /dev/ttyS5
```

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 49fff743 ... 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

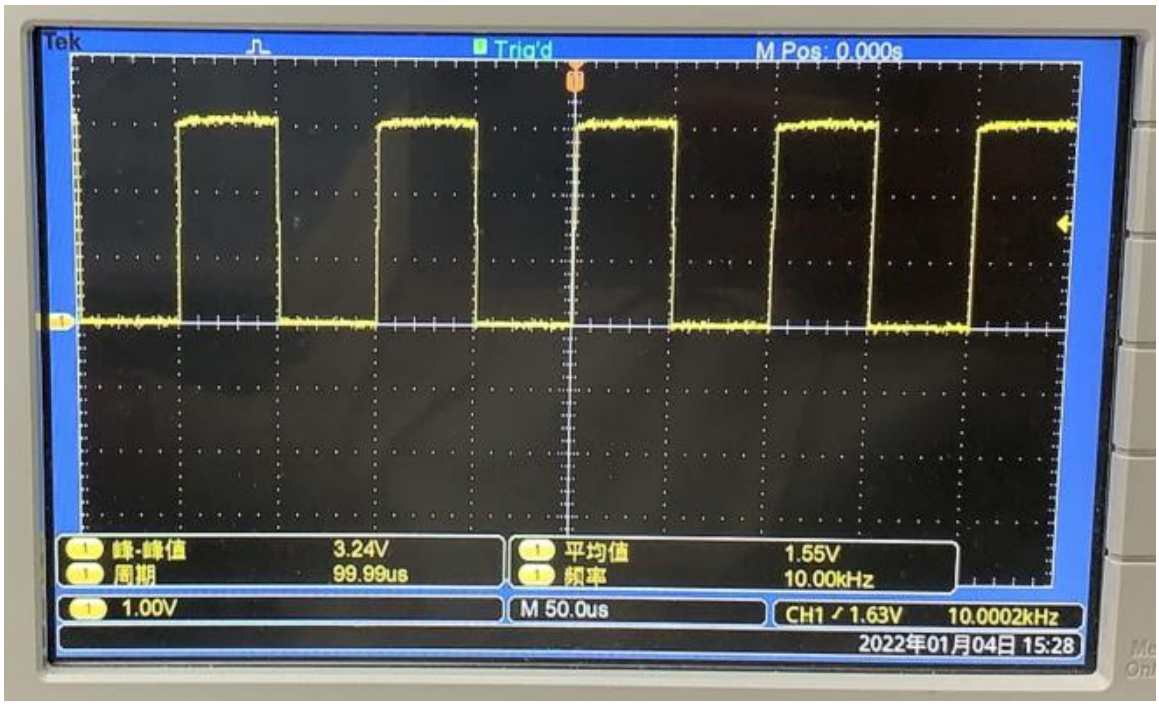
```
root@orangepi:~# echo 3 > /sys/class/pwm/pwmchip0/export
root@orangepi:~# echo 100000 > /sys/class/pwm/pwmchip0/pwm3/period
root@orangepi:~# echo 20000 > /sys/class/pwm/pwmchip0/pwm3/duty_cycle
root@orangepi:~# echo 1 > /sys/class/pwm/pwmchip0/pwm3/enable
```



g. 在 Linux4.9 系统中输入下面的命令可以让 pwm4 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
```



3.21. wiringOP-Python 的安装使用方法

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

3.21.1. wiringOP-Python 的安装方法

1) 首先安装依赖包

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

2) 然后使用下面的命令下载 wiringOP-Python 的源码

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

```
root@orangepi:~# git clone \
--recursive https://github.com/orangepi-xunlong/wiringOP-Python

Cloning into 'wiringOP-Python'...
```

```
remote: Enumerating objects: 581, done.
remote: Counting objects: 100% (29/29), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 581 (delta 17), reused 22 (delta 15), pack-reused 552
Receiving objects: 100% (581/581), 303.79 KiB | 1.78 MiB/s, done.
Resolving deltas: 100% (339/339), done.
Submodule 'wiringOP' (https://github.com/orangepi-xunlong/wiringOP.git) registered for path 'wiringOP'
Cloning into '/home/orangepi/wiringOP-Python/wiringOP'...
remote: Enumerating objects: 626, done.
remote: Counting objects: 100% (155/155), done.
remote: Compressing objects: 100% (39/39), done.
remote: Total 626 (delta 129), reused 136 (delta 116), pack-reused 471
Receiving objects: 100% (626/626), 365.55 KiB | 597.00 KiB/s, done.
Resolving deltas: 100% (409/409), done.
Submodule path 'wiringOP': checked out '0a7284942375ff68a9940c44234b6d6ec5f7aa59'
```

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

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

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

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

NAME
    wiringpi

DESCRIPTION
    # This file was automatically generated by SWIG (http://www.swig.org).
    # Version 4.0.2
    #
```

```
    # Do not make changes to this file unless you know what you are doing--modify
    # the SWIG interface file instead.
```

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

a. 首先使用 python3 命令进入 python3 的命令行模式

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

b. 然后导入 wiringpi 的 python 模块

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

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

```
>>> help(wiringpi)
Help on module wiringpi:

NAME
    wiringpi

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

CLASSES
    builtins.object
        GPIO
        I2C
        Serial
        nes

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

>>>
```

3.21.2. 26pin GPIO 口测试

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

```
root@orangepizero2:~# gpio readall
 +------+-----+----------+------+---+  Zero 2  +---+------+----------+-----+------+
 | GPIO | wPi |   Name   | Mode | V | Physical | V | Mode | Name     | wPi | GPIO |
 +------+-----+----------+------+---+----++----+---+------+----------+-----+------+
 |      |     |     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  +---+------+----------+-----+------+
```

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
 +------+-----+----------+------+---+  Zero 2  +---+------+----------+-----+------+
 | GPIO | wPi |   Name   | Mode | V | Physical | V | Mode | Name     | wPi | GPIO |
 +------+-----+----------+------+---+----++----+---+------+----------+-----+------+
 |      |     |     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) 直接用命令测试的步骤如下所示：

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

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

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

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

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

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

3) 在 python3 的命令行中测试的步骤如下所示：

a. 首先使用 python3 命令进入 python3 的命令行模式

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

b. 然后导入 wiringpi 的 python 模块

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

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

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

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

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

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

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

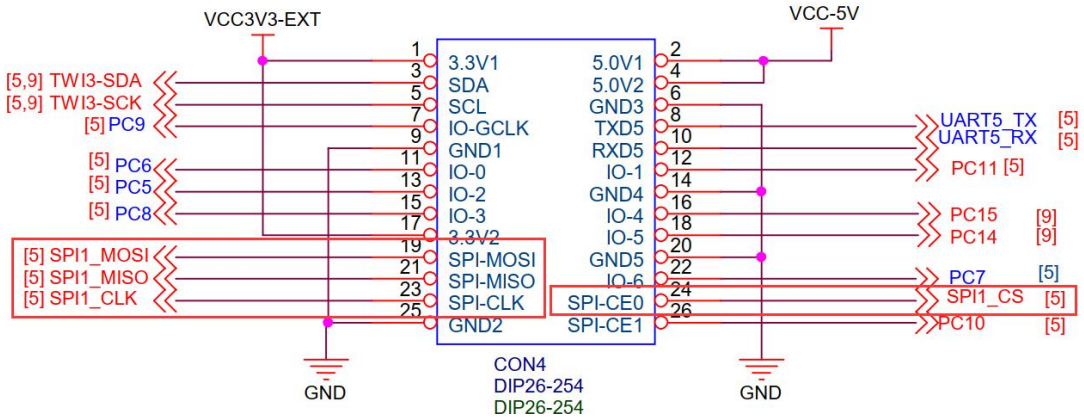
4) wiringOP-Python 在 python 代码中设置 GPIO 高低电平的方法可以参考下 examples 中的 **blink.py** 测试程序，**blink.py** 测试程序会设置开发板 26 pin 中所有的 GPIO 口的电压不断的高低变化

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

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

3.21.3. 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



如果使用的为 Linux5.16 内核的系统，spi1 默认是关闭的，需要手动打开才能使用。

在/boot/orangepiEnv.txt 中加入下面红色字体部分的配置，然后重启 Linux 系统就可以打开 spi1。

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

overlays=spi-spidev

param_spidev_spi_bus=1

param_spidev_spi_cs=1

2) 首先查看下 linux 系统中是否存在 **spidev1.1** 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-Python# ls /dev/spidev1*
/dev/spidev1.1
```

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

a. **--channel**：指定 SPI 的通道号

b. **--port**：指定 SPI 的端口号

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

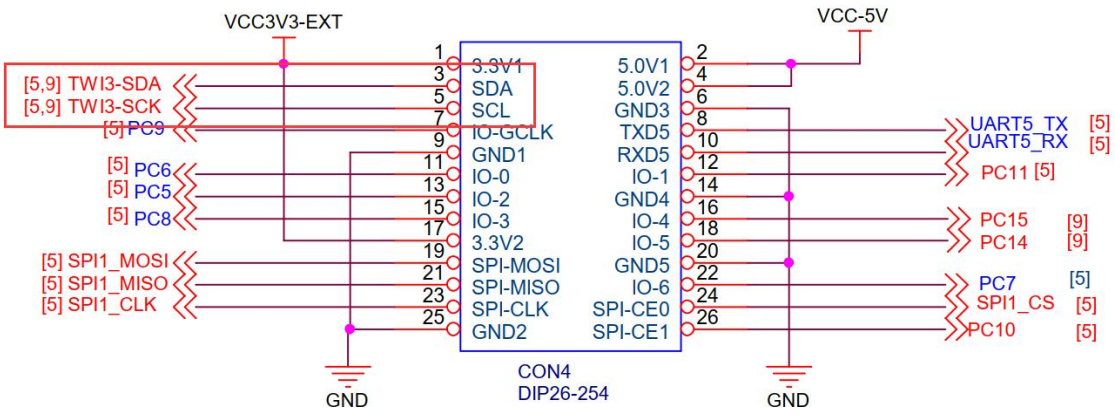
```
root@orangepi:~/wiringOP-Python# cd examples
root@orangepi:~/wiringOP-Python/examples# python3 spidev_test.py \
--channel 1 --port 1
spi mode: 0x0
max speed: 500000 Hz (500 KHz)
Opening device /dev/spidev1.1
TX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF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

5) 然后使用杜邦线短接 SPI1 的 txd（26pin 接口中的第 19 号引脚）和 rxd（26pin 接口中的第 21 号引脚）两个引脚再运行 spidev_test.py 的输出如下，可以看到发送和接收的数据一样，说明 SPI1 回环测试正常

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

3.21.4. 26pin I2C 测试

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



如果使用的为 Linux5.16 内核的系统，i2c3 默认是关闭的，需要手动打开才能使用。

在/boot/orangepiEnv.txt 中加入下面红色字体部分的配置，然后重启 Linux 系统就可以打开 i2c3。

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

overlays=i2c3

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

```
root@orangepi:~# ls /dev/i2c-*
/dev/i2c-3    /dev/i2c-5
```

3) 然后开始测试 i2c，首先安装下 i2c-tools

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

4) 然后在 26pin 接头的 i2c3 引脚上接一个 i2c 设备，这里以 DS1307 RTC 模块为例

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

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 设备连接正确

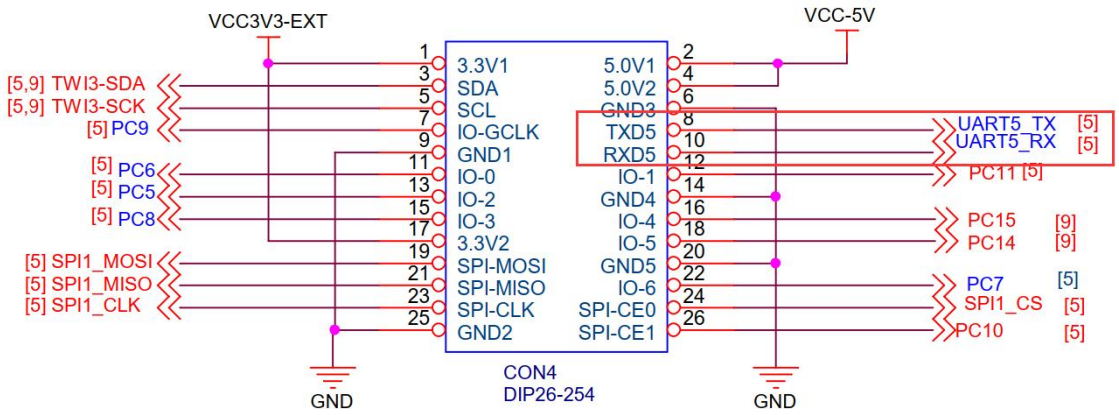
```
root@orangepizero2:~# i2cdetect -y 3
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

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

```
root@orangepi:~/wiringOP-Python# cd examples
root@orangepi:~/wiringOP-Python/examples# python3 ds1307.py --device \
"/dev/i2c-3"
Thu 2022-06-16    04:35:46
Thu 2022-06-16    04:35:47
Thu 2022-06-16    04:35:48
^C
exit
```

3.21.5. 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



如果使用的为 Linux5.16 内核的系统，uart5 默认是关闭的，需要手动打开才能使用。

在/boot/orangepiEnv.txt 中加入下面红色字体部分的配置，然后重启 Linux 系统就可以打开 uart5。

orangepi@orangepi:~$ **sudo vim /boot/orangepiEnv.txt**
overlays=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

	uart5
tx 引脚	对应 8 号引脚
rx 引脚	对应 10 号引脚

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

```
root@orangepi:~/wiringOP-Python# cd examples
root@orangepi:~/wiringOP-Python/examples# python3 serialTest.py --device \
"/dev/ttyS5"

Out:    0: ->    0
```

```
Out:    1: ->    1
Out:    2: ->    2
Out:    3: ->    3
Out:    4:^C
exit
```

3.22. 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

3.22.1. 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
```

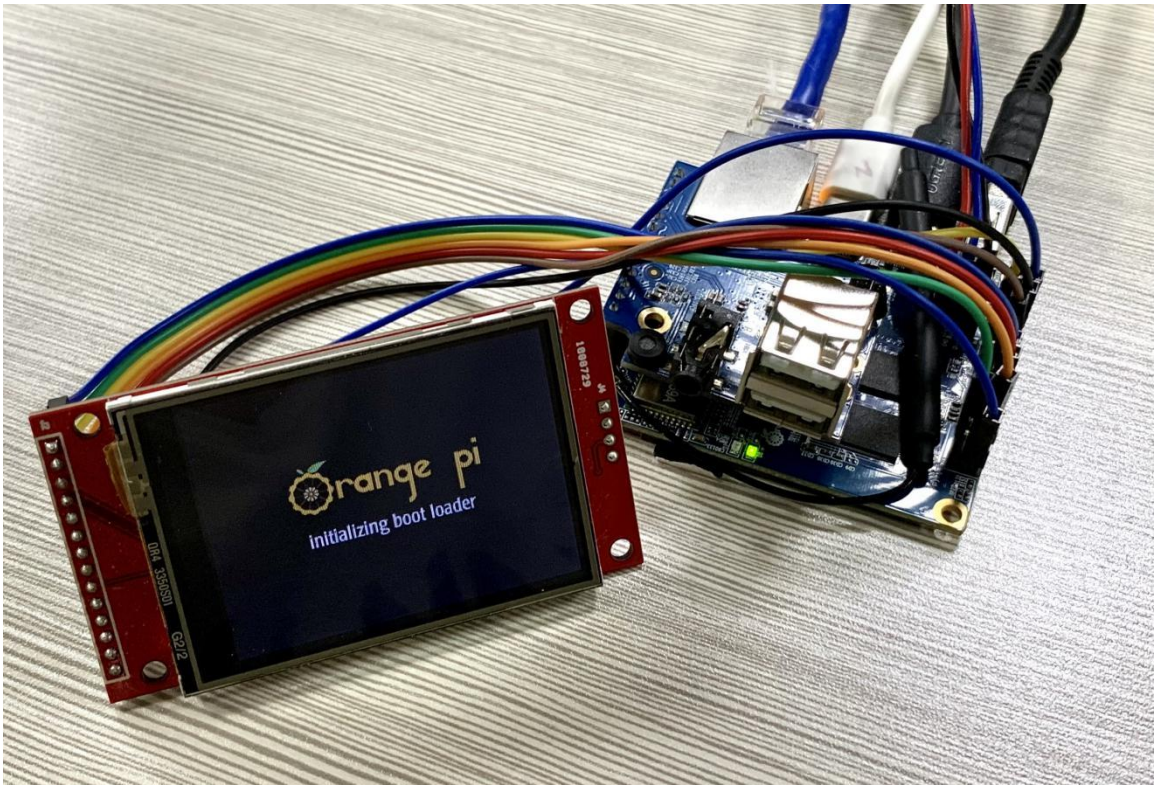
4) **fbtft_device** 内核模块加载时 dmesg 命令正确的输出 log 如下所示，而且由 log 可以知道 LCD 显示屏使用的 framebuffer 为 **fb1**

```
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

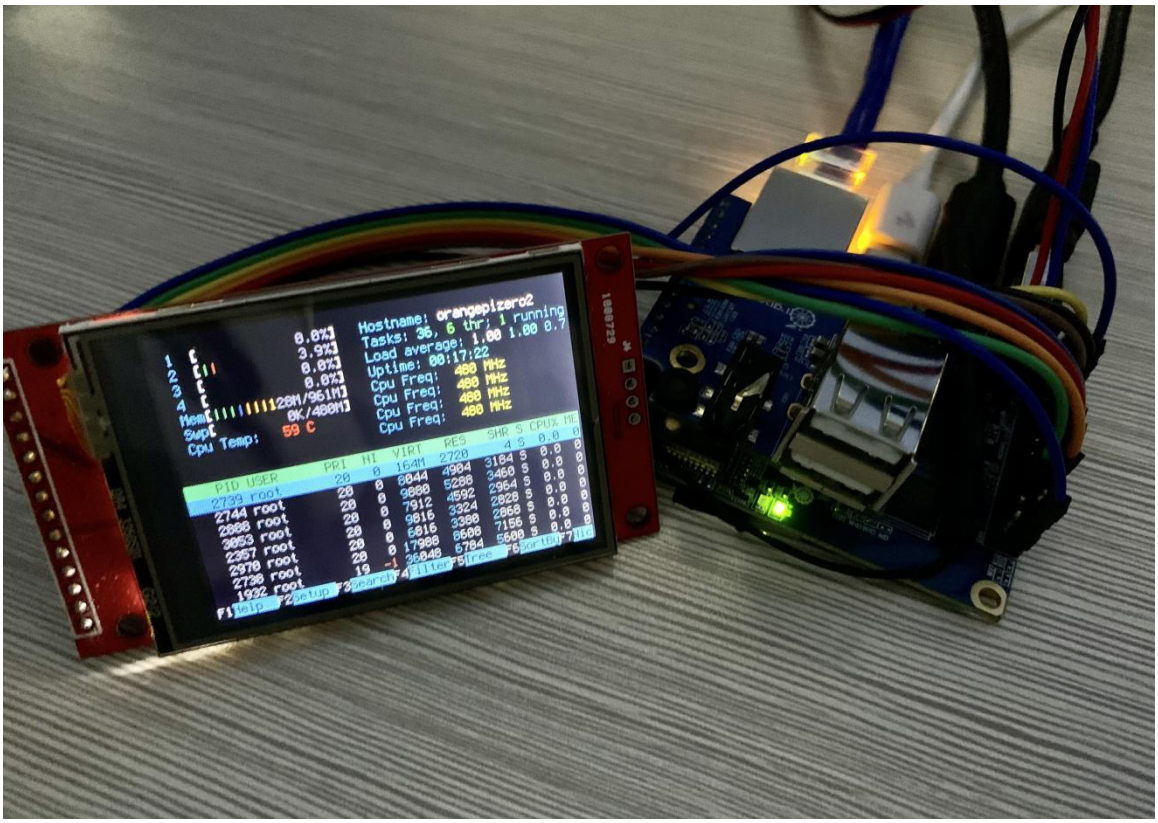
就不会再有图像输出了

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

如果要切换回 HDMI 显示，请使用下面的命令

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

下面是运行 htop 命令的输出

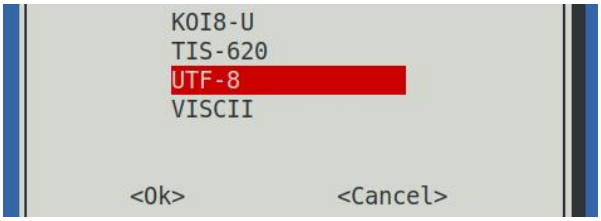


7) 由于默认的终端字体太大，导致显示屏无法显示太多的内容，可以通过下面的方法来缩小终端的字体

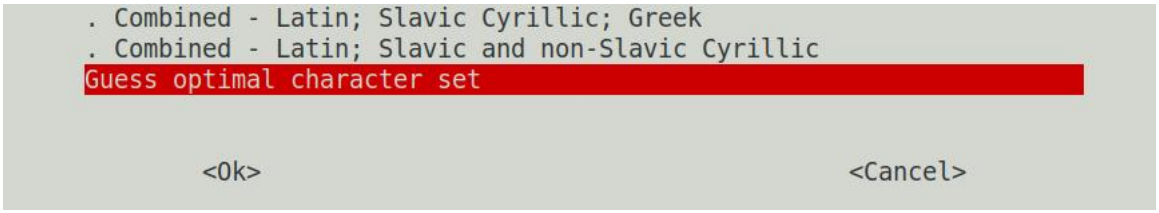
a. 首先运行 **dpkg-reconfigure console-setup**

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

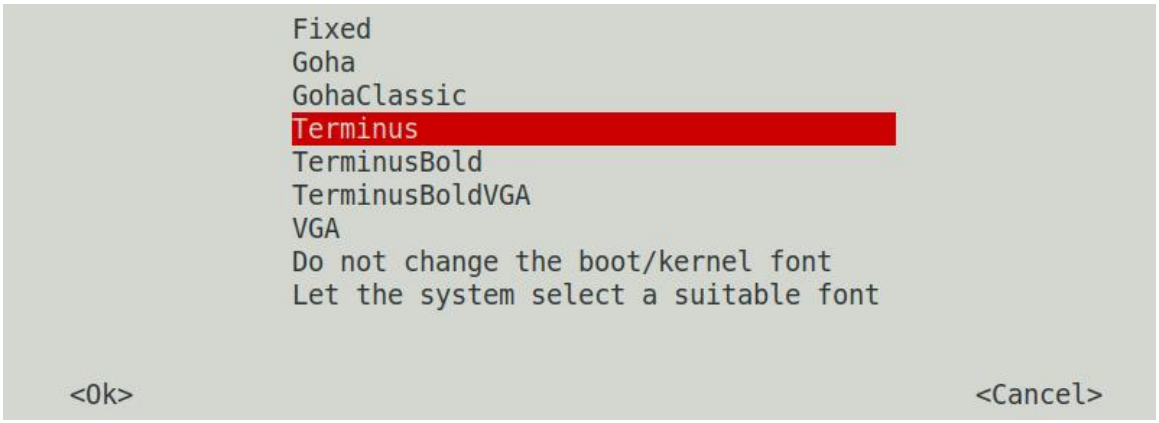
b. 终端编码选择 **UTF-8**



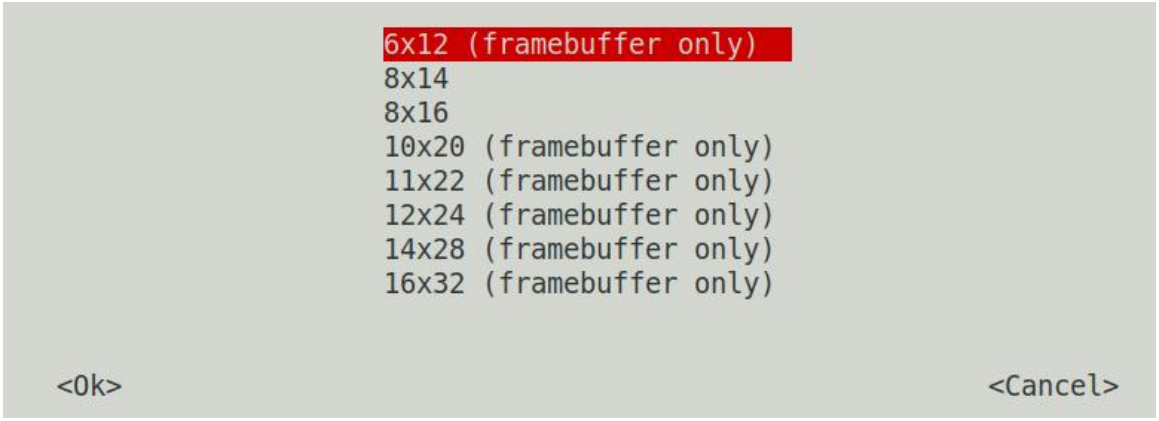
c. 然后选择 **Guess optimal character set**



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.con** 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
gpios=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
```

3.22.2. 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
```

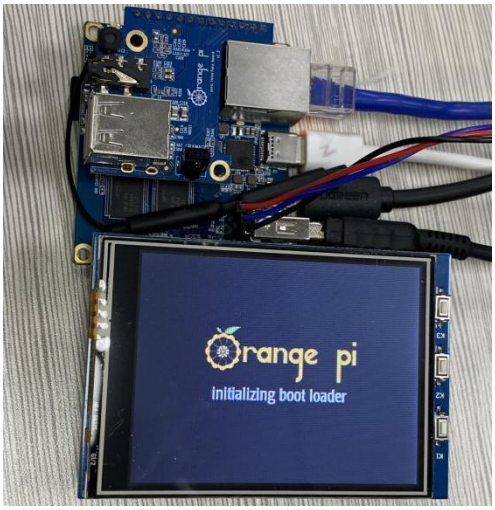
4) **fbtft_device** 内核模块加载时 dmesg 命令正确的输出 log 如下所示，而且由 log 可以知道 LCD 屏幕使用的 framebuffer 为 **fb1**

```
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@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 屏幕的 fb 设备——**fb1**，映射完后，HDMI 就不会再有图像输出了

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

如果要切换回 HDMI 显示，请使用下面的命令

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

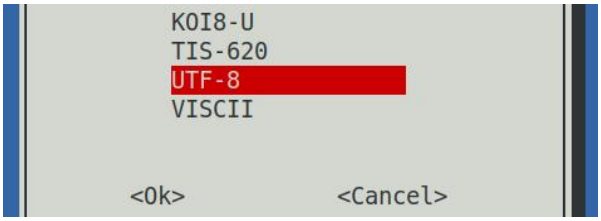
下面是运行 htop 命令的输出

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

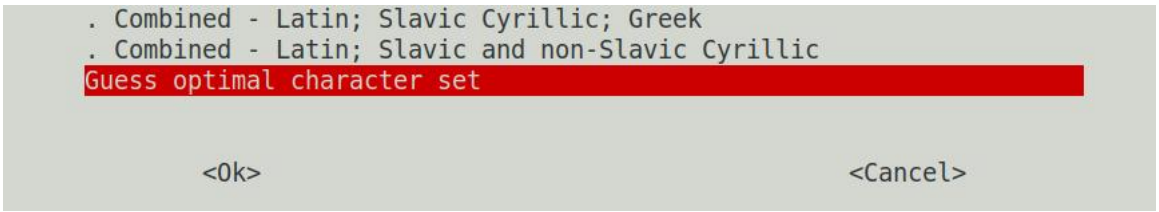
a. 首先运行 **dpkg-reconfigure console-setup**

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

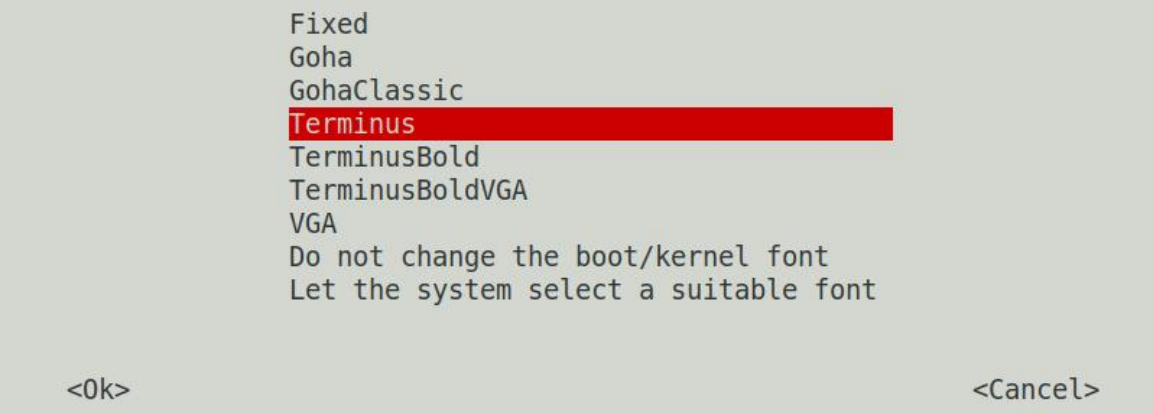
b. 终端编码选择 **UTF-8**



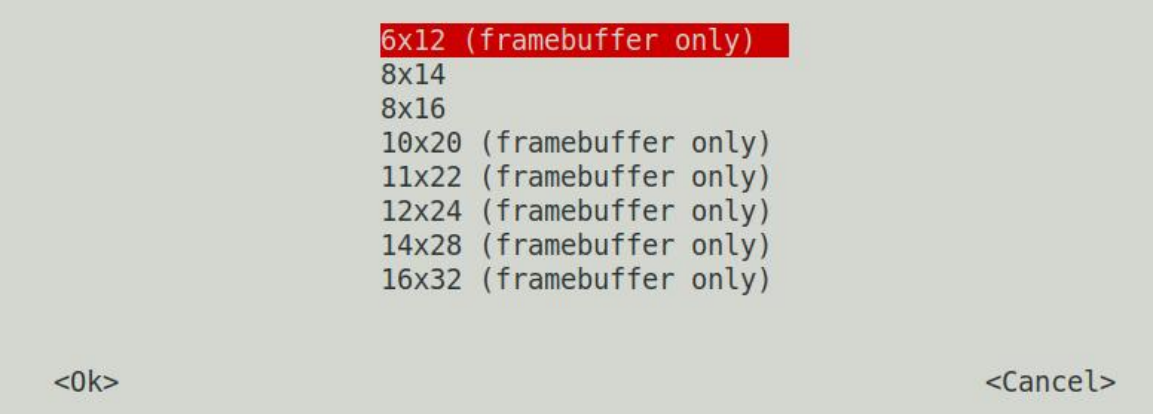
c. 然后选择 **Guess optimal character set**



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

3.22.3. 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_ili9488   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_ili9488':
[   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_ili9488 spi1.1 65000kHz 8 bits mode=0x00
[   378.971754] fb_ili9488: 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

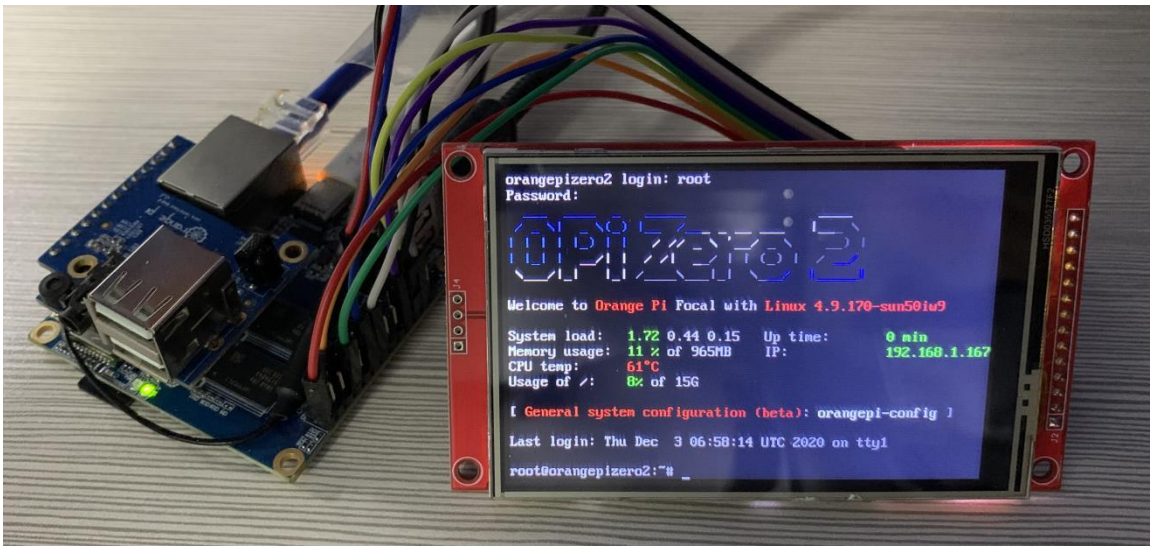
```
root@orangepi:~# apt update
root@orangepi:~# apt -y install fbi
root@orangepi:~# fbi -vt 1 -noverbose -d /dev/fb1 /boot/boot.bmp
```

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，LCD 屏幕将会显示终端的输出，HDMI 就不会再有图像输出了

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

如果要切换回 HDMI 显示，请使用下面的命令

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



7) 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.con** 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_ili9488  busnum=1 cs=1
gpios=reset:73,dc:70,led:69 rotate=90 speed=65000000 bgr=1 txbuflen=65536
```

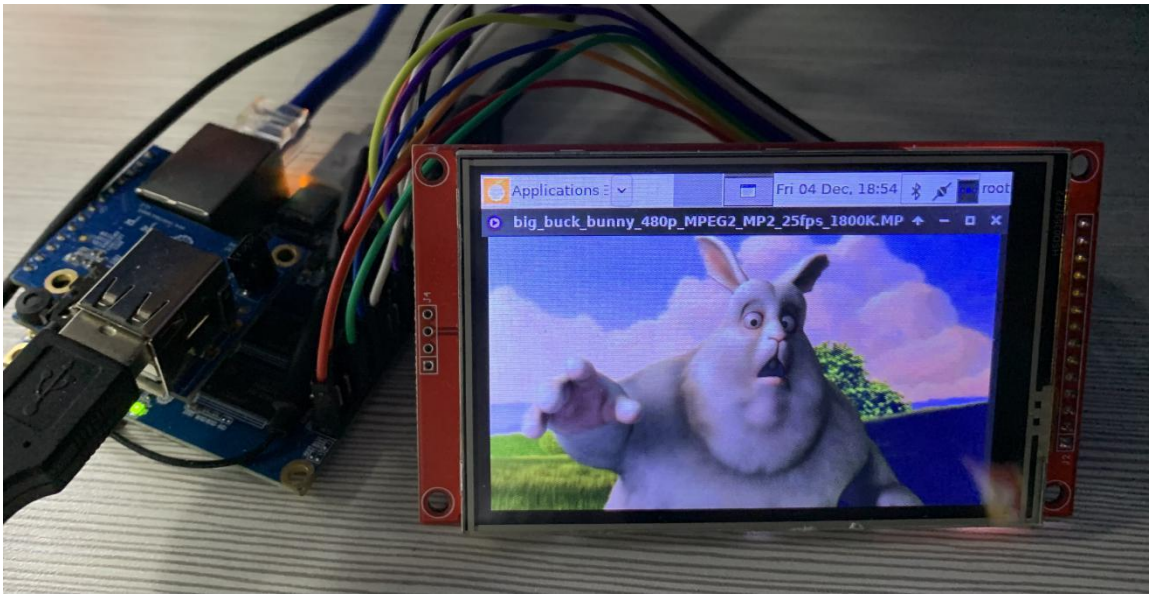
c. 然后重启 linux 系统就能看到 fbtft_device 相关的内核模块都已自动加载

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

3.23. 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。

如果使用的为 Linux5.16 内核的系统，uart5 默认是关闭的，需要手动打开才能使用。

在/boot/orangepiEnv.txt 中加入下面红色字体部分的配置，然后重启 Linux 系统就可以打开 uart5。

orangepi@orangepi:~$ **sudo vim /boot/orangepiEnv.txt**
overlays=uart5

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

rangepi@orangepi:~$ **sudo 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）

```
rangepi@orangepi:~$ sudo 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
```

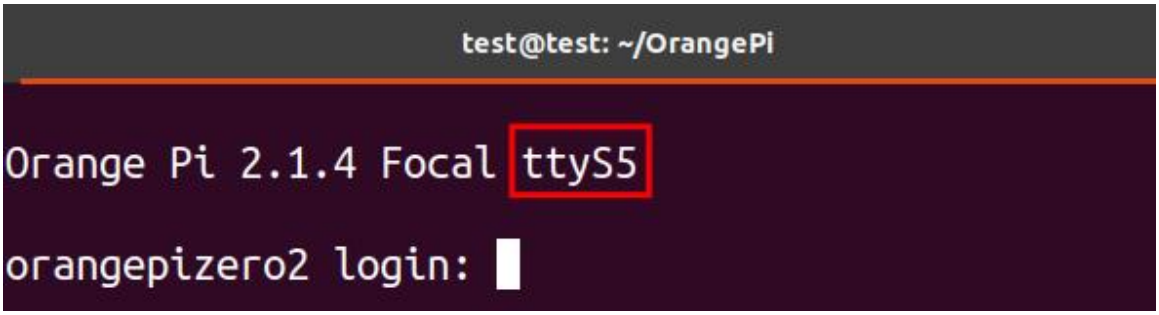
3) 然后将 USB 转 TTL 模块通过杜邦线接到 26pin 接口的 UART5 引脚上

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



3. 24. I2C 接口的 0.96 寸 OLED 模块使用方法

如果使用的为 Linux5.16 内核的系统，i2c3 默认是关闭的，需要手动打开才能使用。

在/boot/orangepiEnv.txt 中加入下面红色字体部分的配置，然后重启 Linux 系统就可以打开 i2c3。

orangepi@orangepi:~$ **sudo vim /boot/orangepiEnv.txt**
overlays=i2c3

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

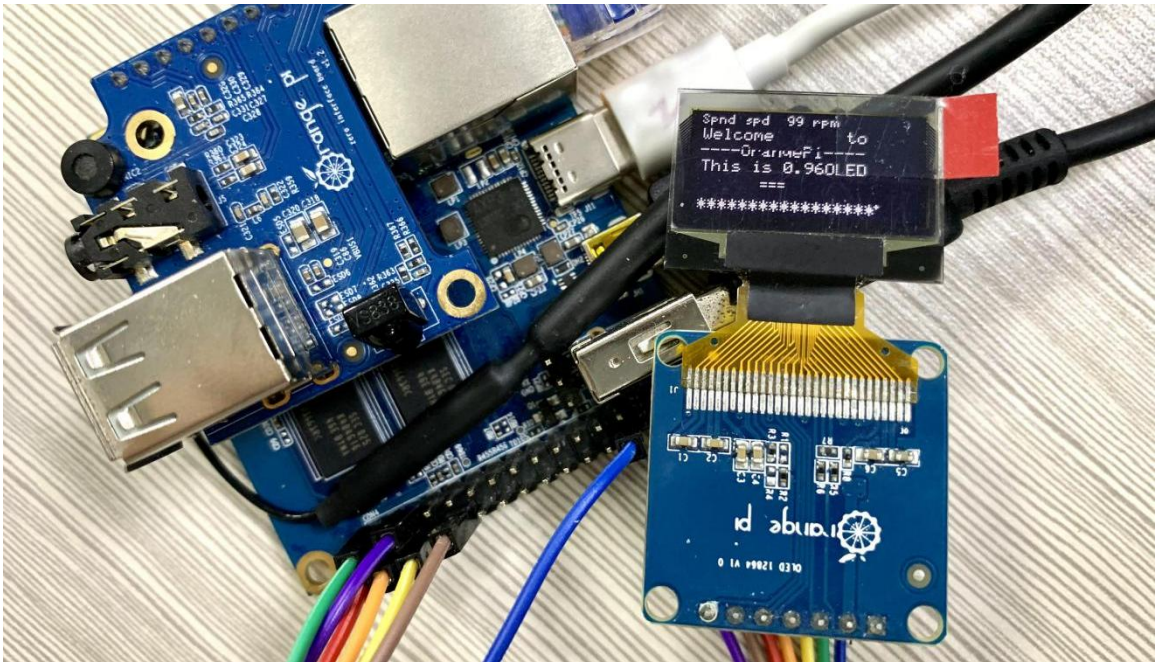
```
orangepi@orangepi:~$ sudo apt update
orangepi@orangepi:~$ sudo apt install -y i2c-tools
orangepi@orangepi:~$ sudo i2cdetect -y 3
```

```
root@orangepizero2:~# 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@orangepizero2:~#
```

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@orangepi:~# git clone https://github.com/orangepi-xunlong/wiringOP
root@orangepi:~# cd wiringOP
root@orangepi:~/wiringOP# ./build clean && ./build
root@orangepi:~/wiringOP# cd examples
root@orangepi:~/wiringOP/examples# make oled_demo
root@orangepi:~/wiringOP/examples# ./oled_demo /dev/i2c-3
---------start--------
----------end---------
```

5) 运行 oled_demo 后，在 OLED 屏幕上就能看到下面的输出



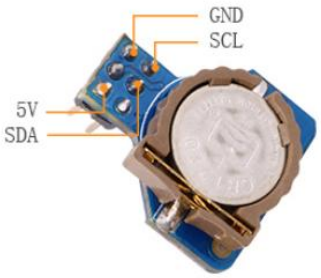
3.25. 香橙派 DS1307 RTC 时钟模块使用方法

如果使用的为 Linux5.16 内核的系统，i2c3 默认是关闭的，需要手动打开才能使用。

在/boot/orangepiEnv.txt 中加入下面红色字体部分的配置，然后重启 Linux 系统就可以打开 i2c3。

orangepi@orangepi:~$ **sudo vim /boot/orangepiEnv.txt**
overlays=i2c3

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

```
orangepi@orangepi:~$ sudo apt update
orangepi@orangepi:~$ sudo apt install -y i2c-tools
orangepi@orangepi:~$ sudo i2cdetect -y 3
```

```
root@orangepizero2:~# 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 **orangepi-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

```
orangepi@orangepi:~$ vim 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. linux5.16 系统 rtc-ds1307 模块 dts 文件的内容

```
orangepi@orangepi:~$ vim 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

```
orangepi@orangepi:~$ sudo 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 **orangepi-add-overlay** 后可以查看下**/boot/overlay-user** 中是否有 i2c-ds1307.dtbo 这个文件

```
orangepi@orangepi:~$ cd /boot/overlay-user/
orangepi@orangepi:/boot/overlay-user$ ls
i2c-ds1307.dtbo
```

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

```
orangepi@orangepi:~$ cat /boot/orangepiEnv.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（linux5.16 为 rtc1）**

```
orangepi@orangepi:~$ 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

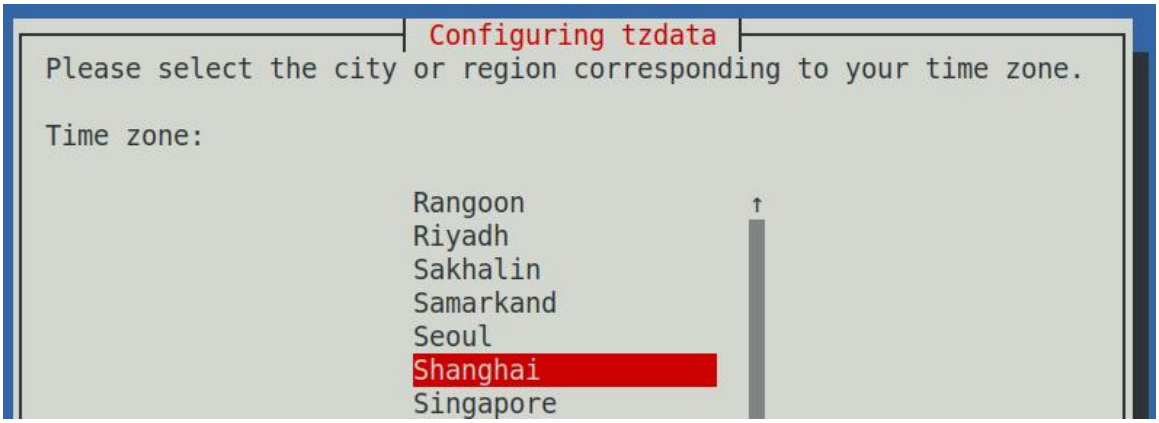
时间为正确的时间，linux 系统默认时间为世界标准时间 UTC，在中国，需要将时区修改为 Asia/Shanghai，使用 data 命令获取到的时间才正确，方法如下

a. 执行下面的命令

```
orangepi@orangepi:~$ sudo 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

```
orangepi@orangepi:~$ sudo 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

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

```
orangepi@orangepi:~$ sudo ntpdate 0.cn.pool.ntp.org
```

8) 查看 RTC 模块当前时间的命令如下所示

a. Linux4.9

```
orangepi@orangepi:~$ sudo hwclock -r
```

b. Linux5.16

```
orangepi@orangepi:~$ sudo hwclock -r -f /dev/rtc1
```

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

a. Linux4.9

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

b. Linux5.16

```
orangepi@orangepi:~$ date
orangepi@orangepi:~$ sudo hwclock -w -f /dev/rtc1
orangepi@orangepi:~$ sudo hwclock -r -f /dev/rtc1
```

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

3.26. 硬件看门狗测试

1) 下载 wiringOP 的代码

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

2) 编译 watchdog 测试程序

```
orangepi@orangepi:~$ cd wiringOP/examples/
orangepi@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

```
orangepi@orangepi:~/wiringOP/examples$ sudo ./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
```

3.27. 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。

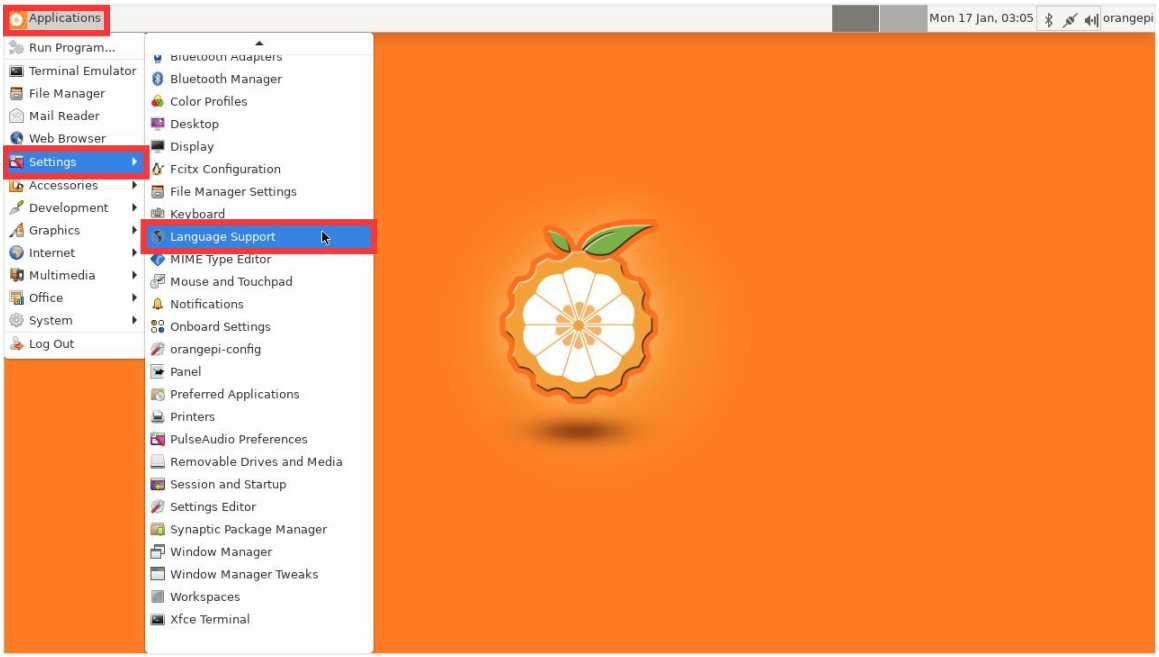
3.27.1. Ubuntu 系统的安装方法

1) 首先更新下系统的软件源

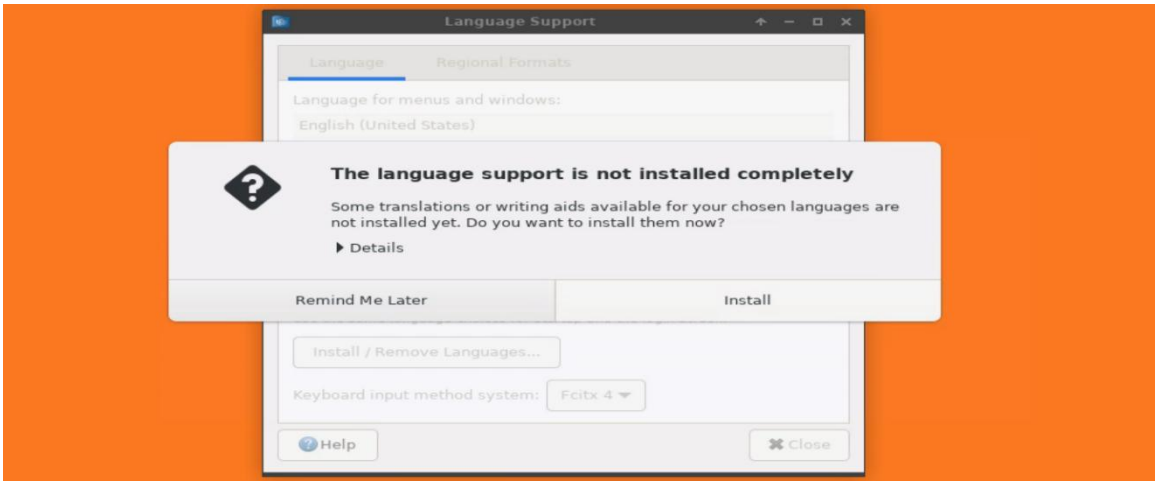
注意，不执行下面的命令后面安装过程可能会报错的，所以请不要忽略这一步。

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

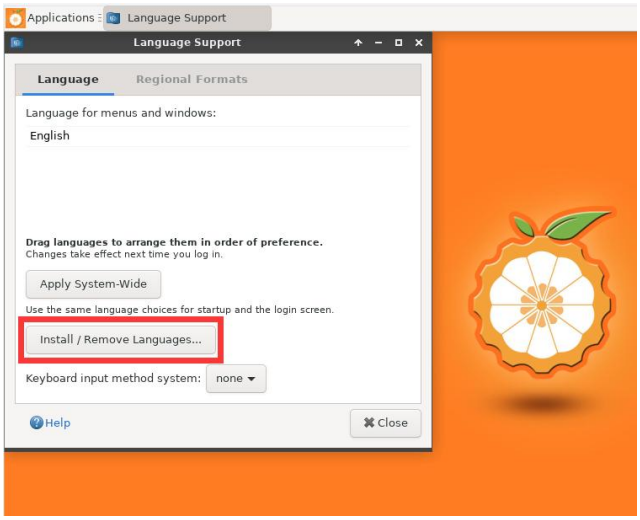
2) 然后打开 **Language Support**



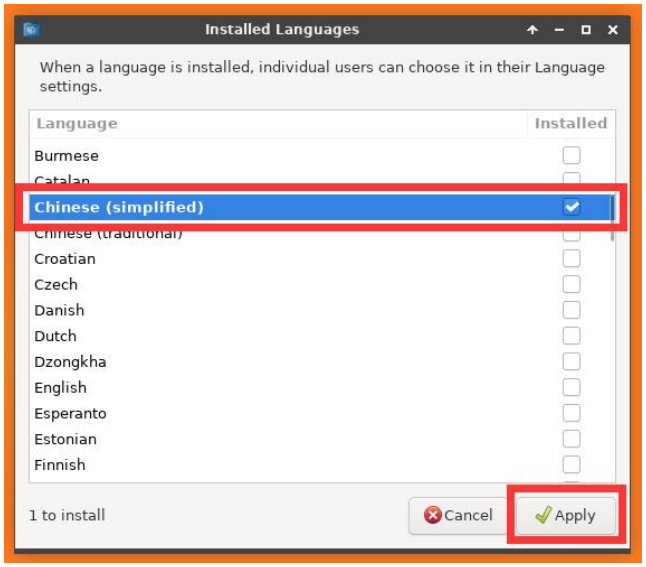
3) 如果有提示下面的信息，请点击 **Install** 修复一下，有些系统不会有下面的提示，则直接跳过即可



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

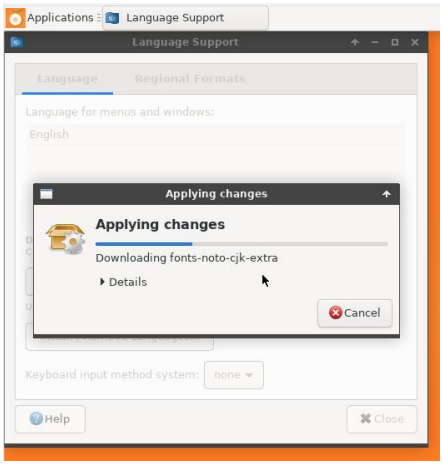


5) 然后找到 **Chinese (simplified)**，点击右边的方框选上，再点击右下角的 **Apply**



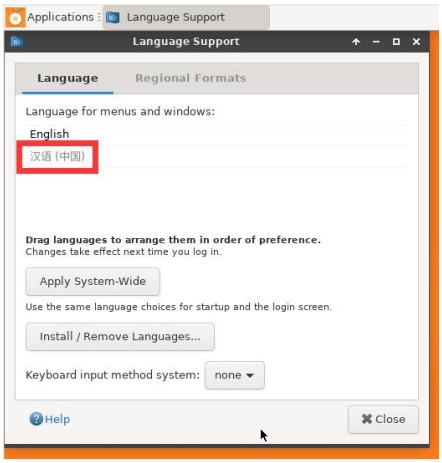
6) 然后在弹出的密码输入界面中输入 Linux 系统的密码，默认为 **orangepi**

7) 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

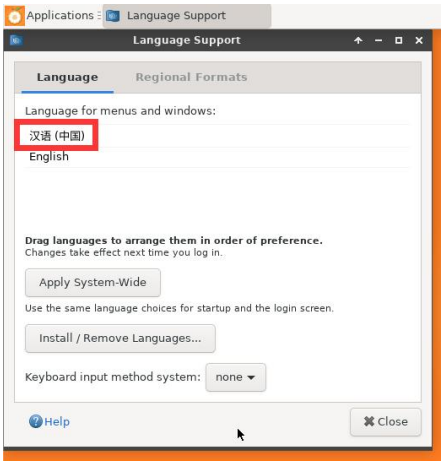


如果这里提示软件安装失败，一般是最开始有没有执行 apt update 命令。如果执行了 apt update 命令还是提示失败，一般是执行 apt update 命令的时候出错了，其实并没有执行成功。

8) 安装完成后就可以看到**汉语（中国）**的选项了

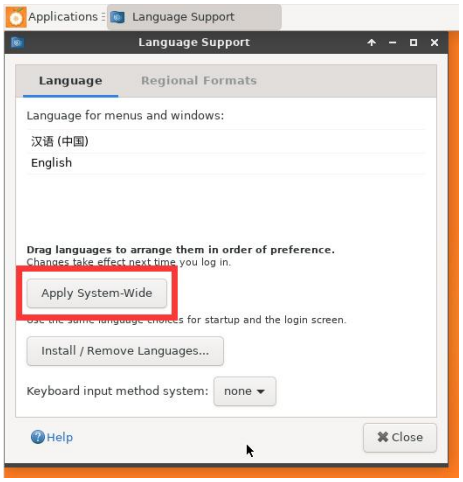


9) 然后请使用鼠标左键选中**汉语（中国）**并按住不动，然后往上将其拖到最开始的位置，拖完后的显示如下图所示：

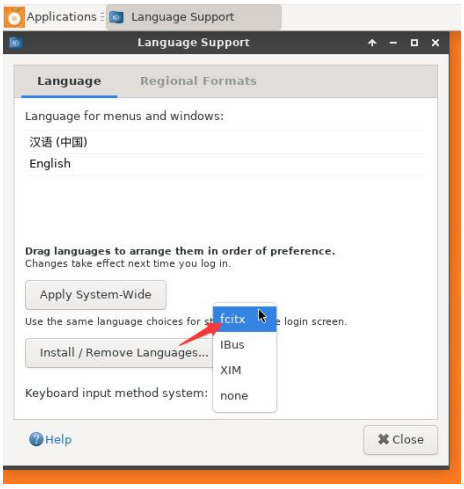


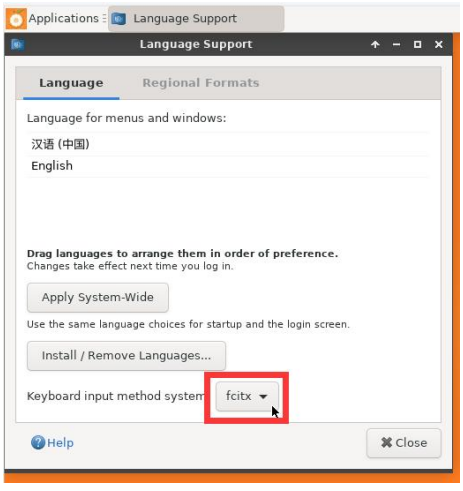
注意，这一步不是很好拖动的，请耐心多试几次。

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



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

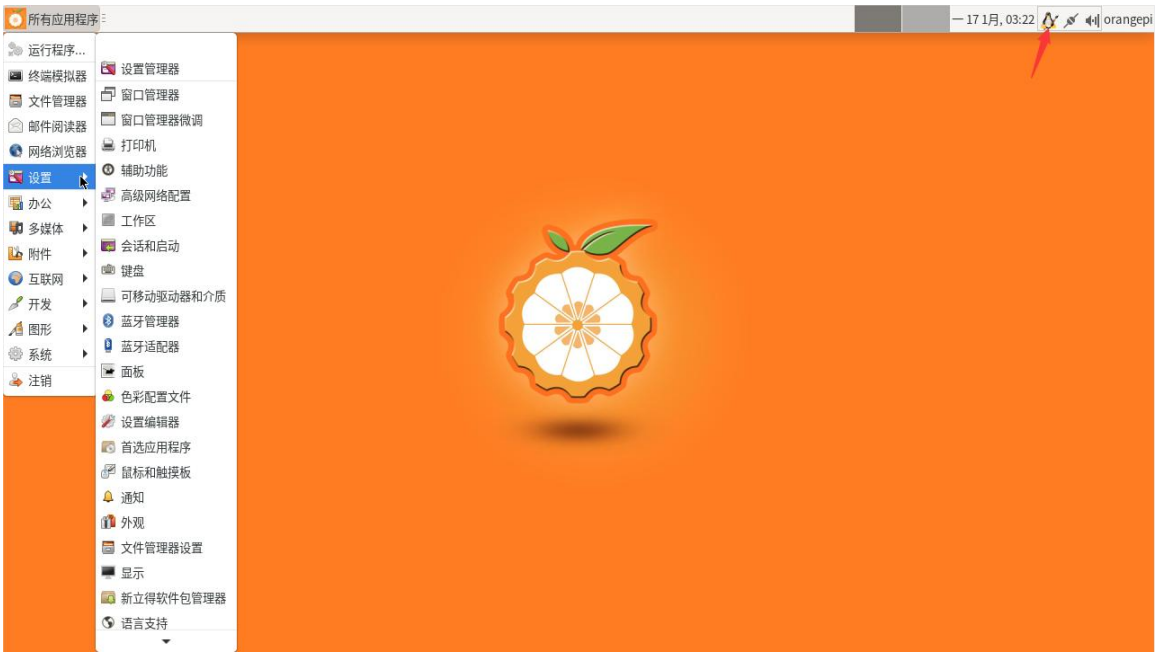




12) **然后重启 Linux 系统使配置生效**

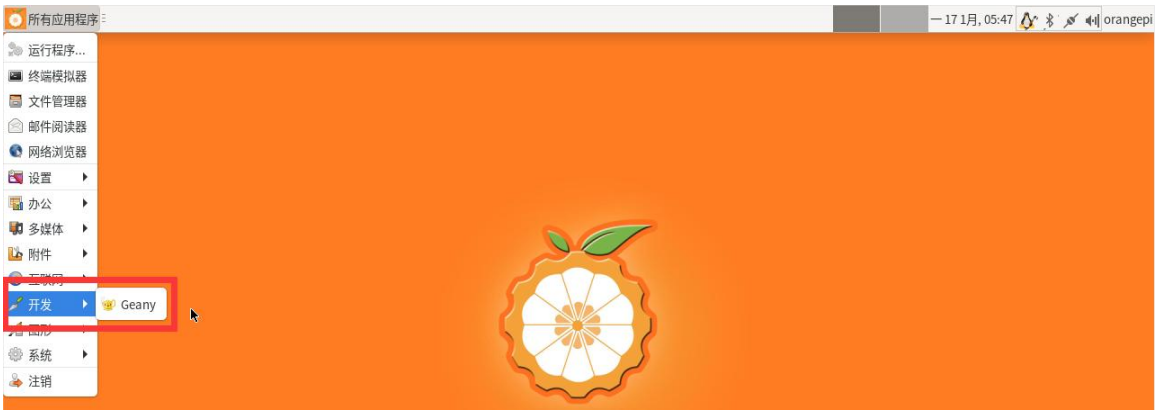
13) 重新进入系统后可以看到桌面都显示为中文了，在系统的右上角还能看到一只企鹅

注意，Ubuntu22.04 在右上角显示的是一个黑色的键盘图标。



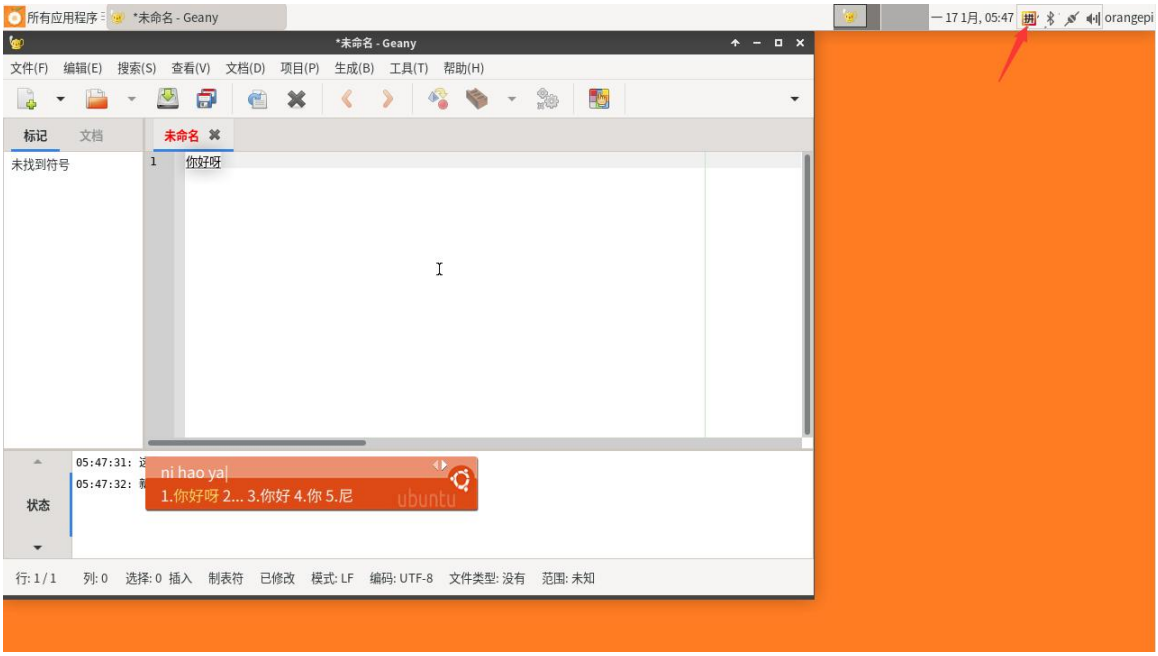
14) 然后我们可以打开 **Geany** 测试下中文输入法，打开方式如下图所示

注意，如果系统中没有安装 Geany，就请随意打开一个其他的应用程序测试即可。



15) 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



16) 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

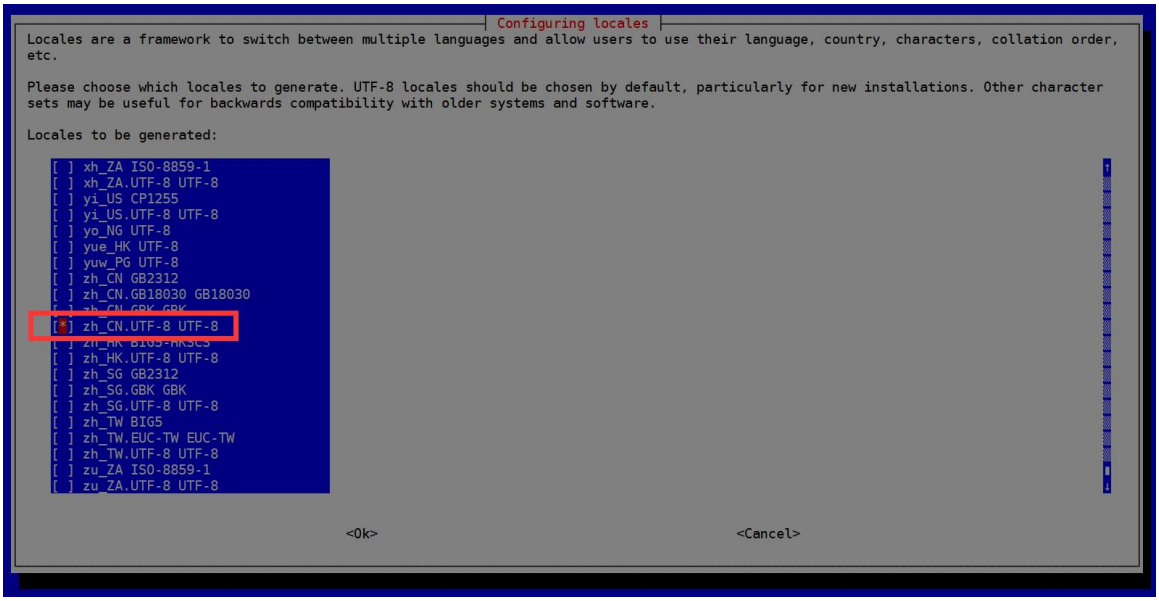
3.27.2. Debian 系统的安装方法

1) 首先设置默认 **locale** 为中文

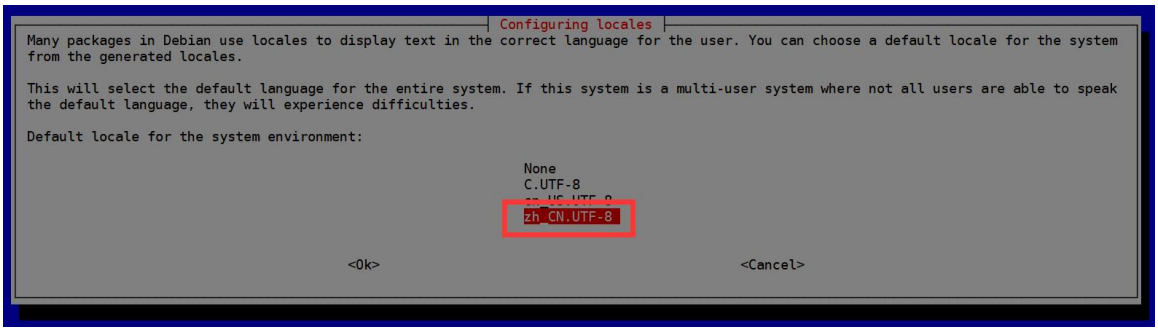
a. 输入下面的命令可以开始配置 **locale**

```
orangepi@orangepi:~$ sudo dpkg-reconfigure locales
```

b. 然后在弹出的界面中选择 **zh_CN.UTF-8 UTF-8**（通过键盘上的上下方向按键来上下移动，通过空格键来选择，最后通过 Tab 键可以将光标移动到 **<OK>**，然后回车即可）



c. 然后设置默认 **locale** 为 **zh_CN.UTF-8**



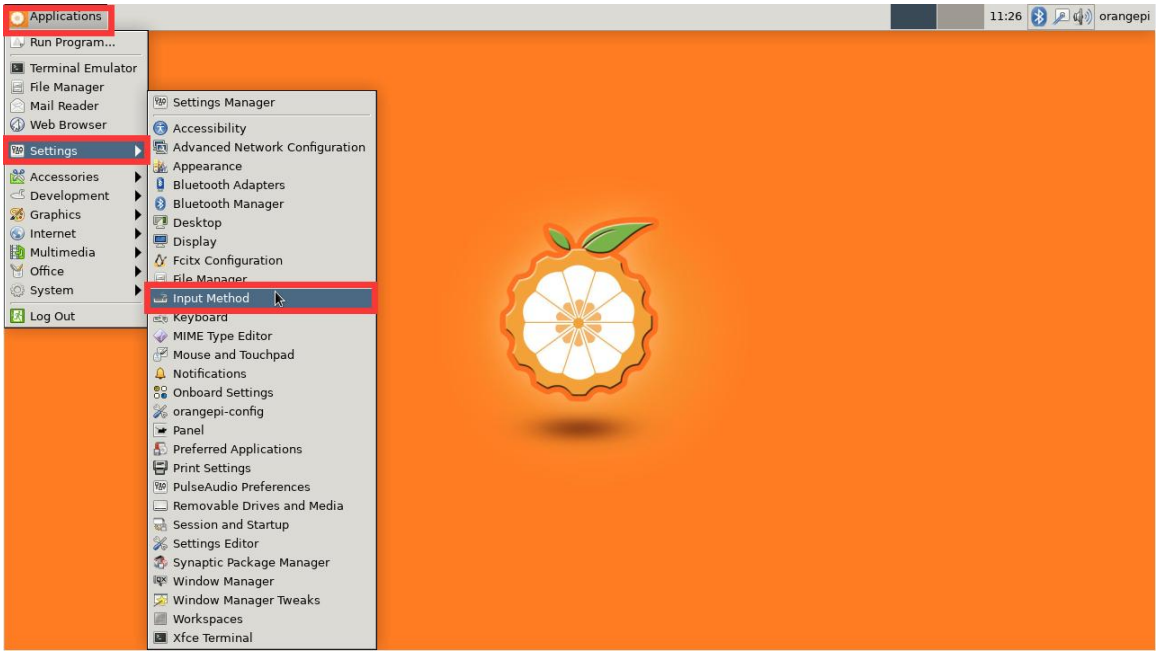
d. 退出界面后就会开始 **locale** 的设置，命令行显示的输出如下所示

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

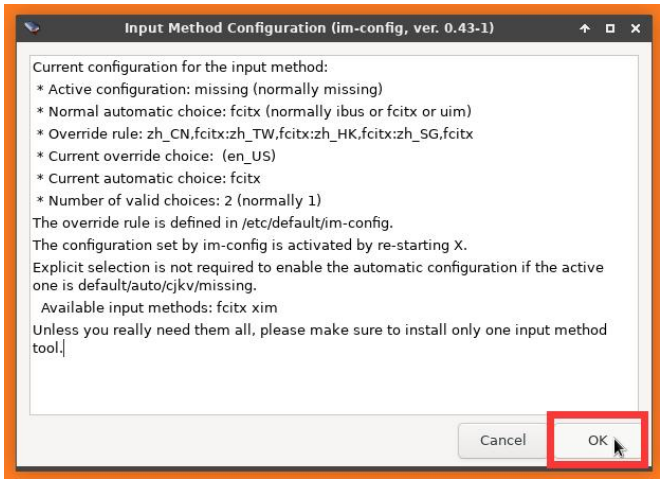
2) 然后安装下面的软件包

```
orangepi@orangepi:~$ sudo apt update
orangepi@orangepi:~$ sudo apt install -y fonts-arphic-bsmi00lp   \
fonts-arphic-gbsn00lp fonts-arphic-gkai00mp fcitx fcitx-table*    \
fcitx-frontend-gtk* fcitx-frontend-qt* fcitx-config-gtk* im-config   \
fcitx-googlepinyin fcitx-ui* zenity geany
```

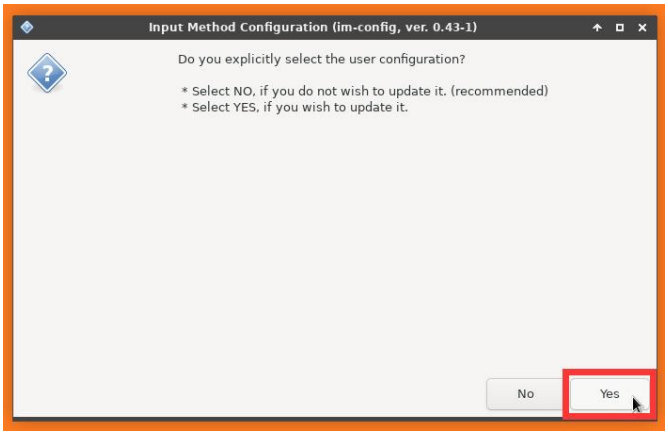
3) 然后打开 **Input Method**



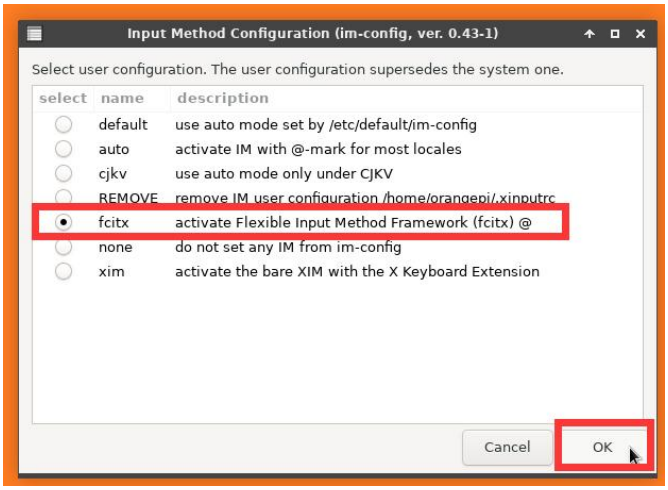
4) 然后选择 **OK**



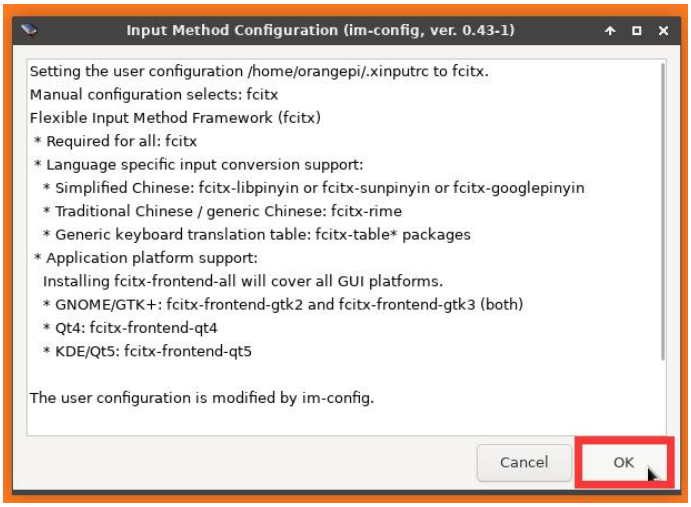
5) 然后选择 **Yes**



6) 然后选择 **fcitx**

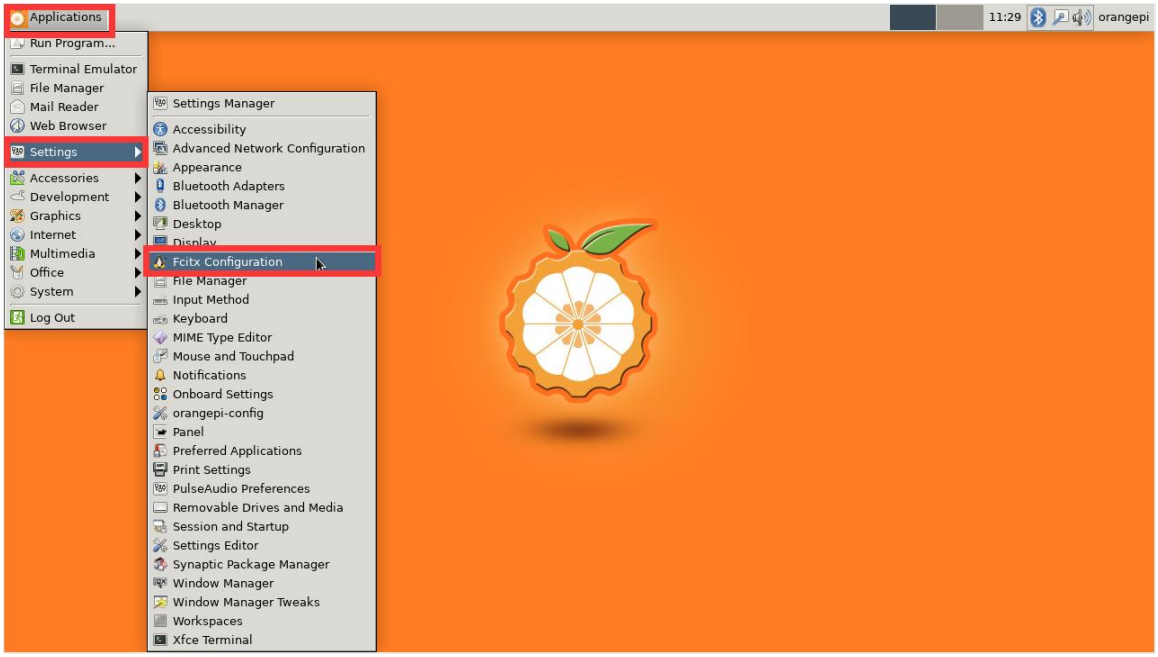


7) 然后选择 **OK**

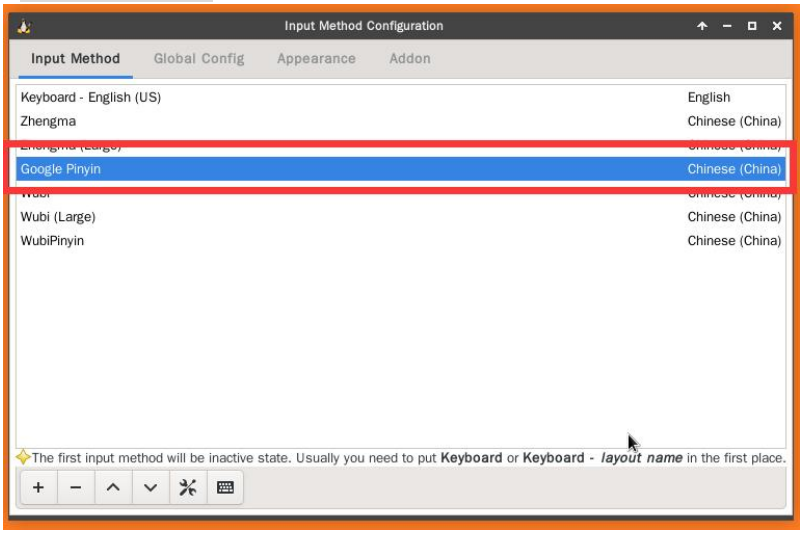


8) **然后重启 Linux 系统才能使配置生效**

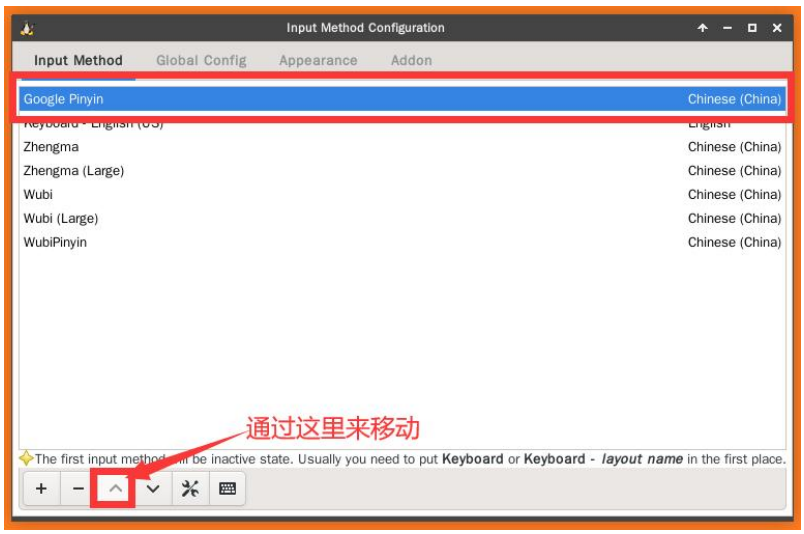
9) 然后打开 **Fcitx configuration**



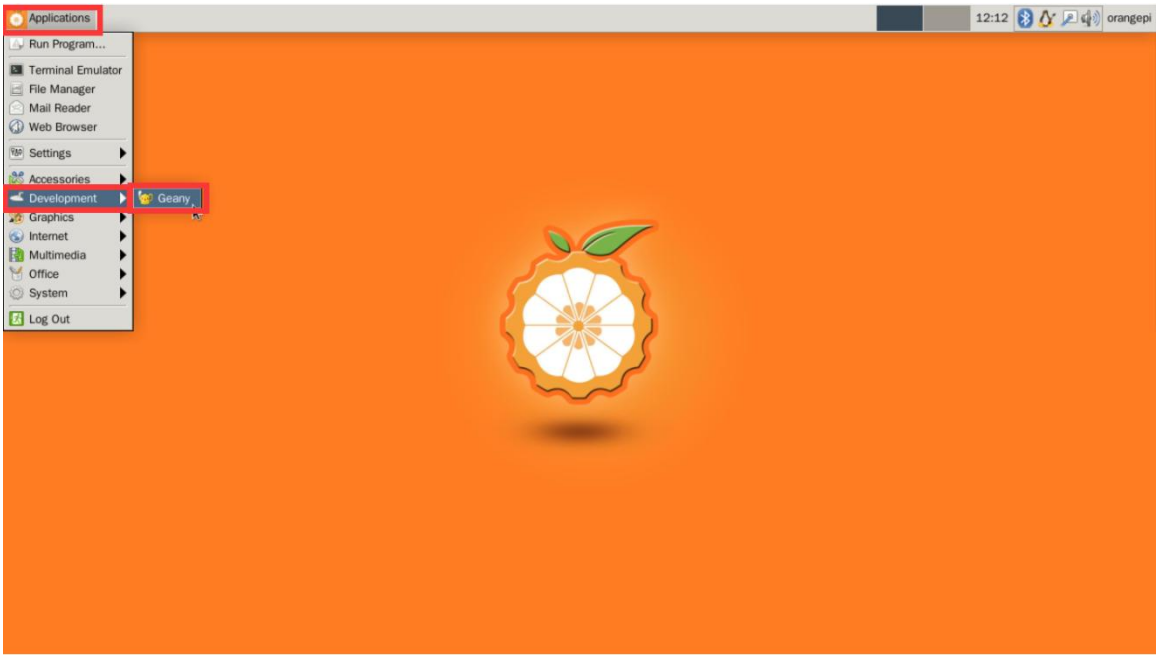
10) 然后选择 **Google Pinyin**



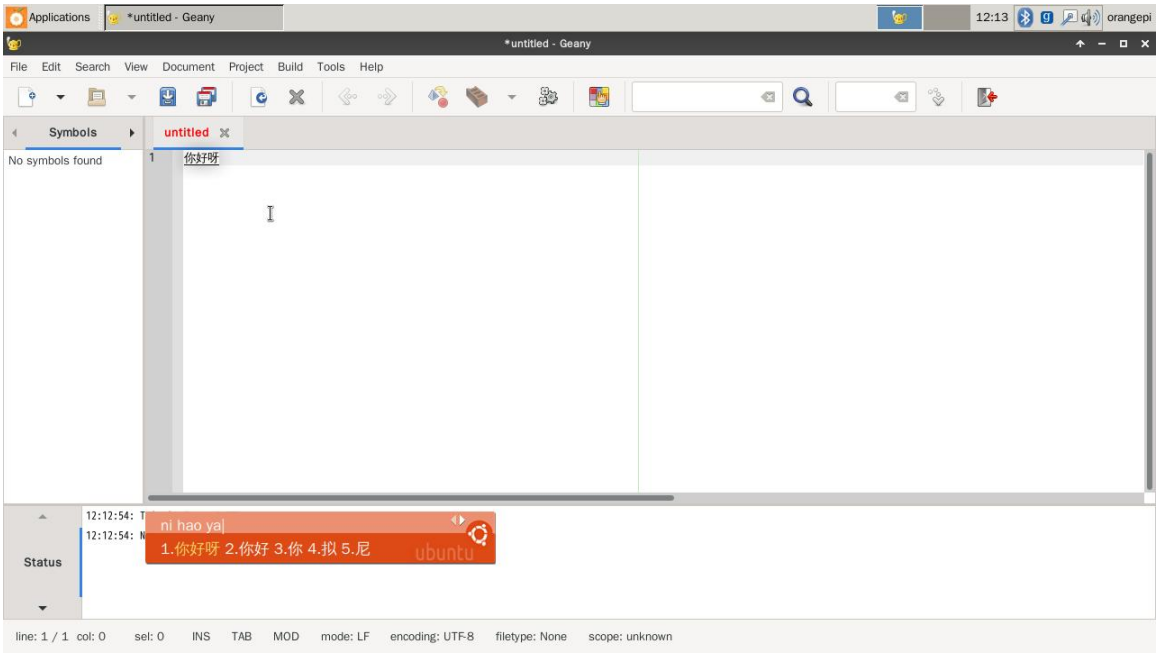
11) 然后将 **Google Pinyin** 放到最前面，再在右上角关闭这个配置界面



12) 然后打开 **Geany** 这个编辑器测试下中文输入法



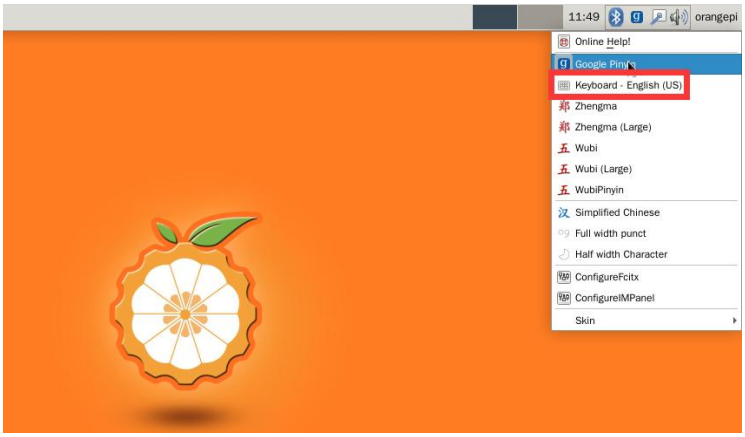
13) 中文输入测试如下所示



14) 在桌面的右上角可以切换回英文输入

a. 首先将鼠标光标放在下图所示的企鹅位置

b. 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

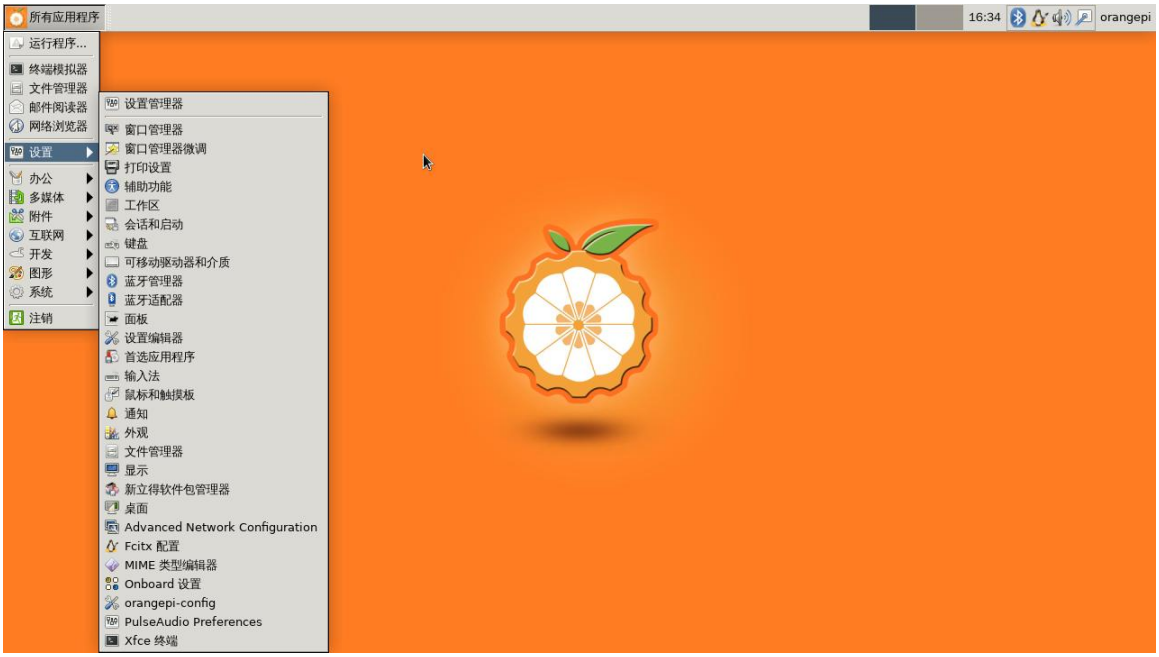


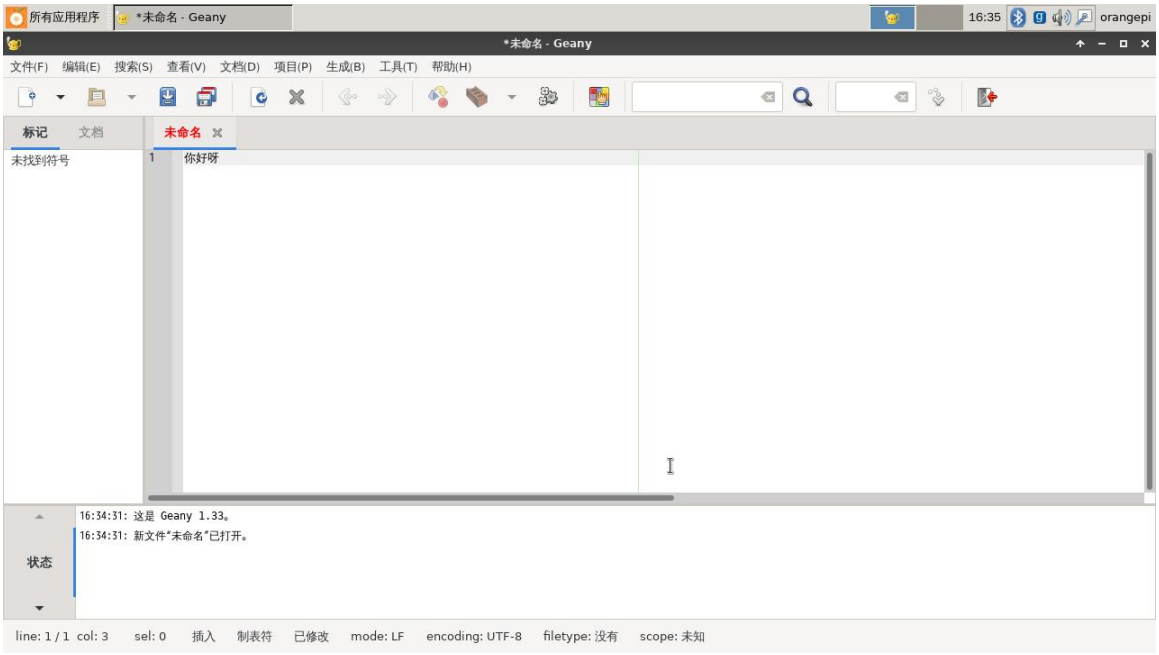
15) 此时就可以通过 **Ctrl+Space** 快捷键来切换中英文输入法了

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**

```
orangepi@orangepi:~$ sudo vim /etc/default/locale
#    File generated by update-locale
LC_MESSAGES=zh_CN.UTF-8
LANG=zh_CN.UTF-8
LANGUAGE=zh_CN.UTF-8
```

17) 然后**重启系统**就能看到系统显示为中文了





3.28. 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

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

3.29. 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

```
orangepi@orangepi:~$ 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

3.30. 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，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) 桌面版镜像，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  3.1M   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@orangepizero2:~#
root@orangepizero2:~# free -h
              total        used        free      shared  buff/cache   available
Mem:           984M        273M        436M       3.6M        273M        635M
Swap:          492M          0B        492M
root@orangepizero2:~#
root@orangepizero2:~# lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 18.04.5 LTS
Release:        18.04
Codename:       bionic
root@orangepizero2:~#
root@orangepizero2:~# uname -r
5.13.0-sun50iw9
root@orangepizero2:~#
```

3.31. 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。

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。

如果没有特殊需求，建议使用下面的镜像，遇到的问题会最少：

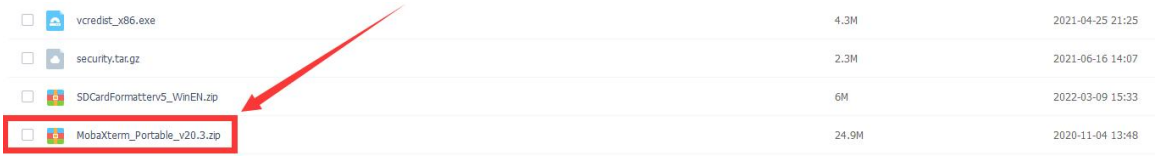
如果您从来没有使用过 Linux 系统，在开始安装 Klipper 前请先仔细阅读本手册 linux 命令格式说明一节的内容，搞清楚了哪些才是真正需要输入的命令部分再开始安装。

Orange Pi 提供的 Linux 系统默认的 apt 软件源都已设置为了清华源，所以无需替换软件源了。

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，然后通过 ssh 远程登录到 Linux 系统，在 Windows 系统中建议使用 MobaXterm 这个软件来远程登录开发板的 Linux 系统

a. 首先下载 MobaXterm，此软件在官方工具中可以下载到

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



b. 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

c. 然后在 MobaXterm 中新建一个 ssh 会话

a) 打开 **Session**

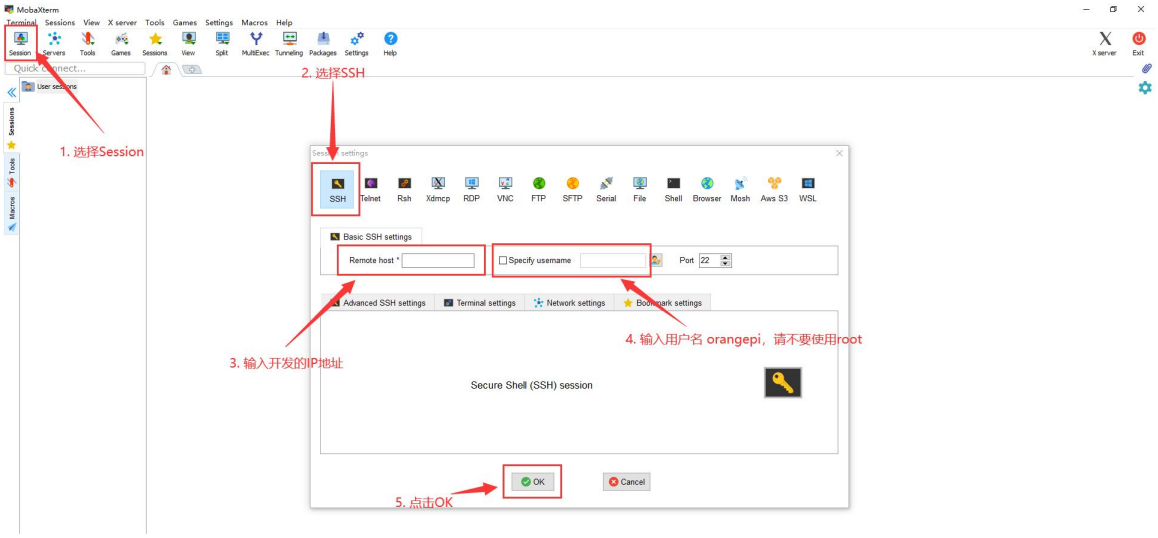
b) 然后在 **Session Setting** 中选择 **SSH**

c) 然后在 **Remote host** 中输入开发板的 IP 地址

d) 然后在 **Specify username** 中输入开发板 Linux 系统的用户名 **orangepi，请不要使用 root 用户**

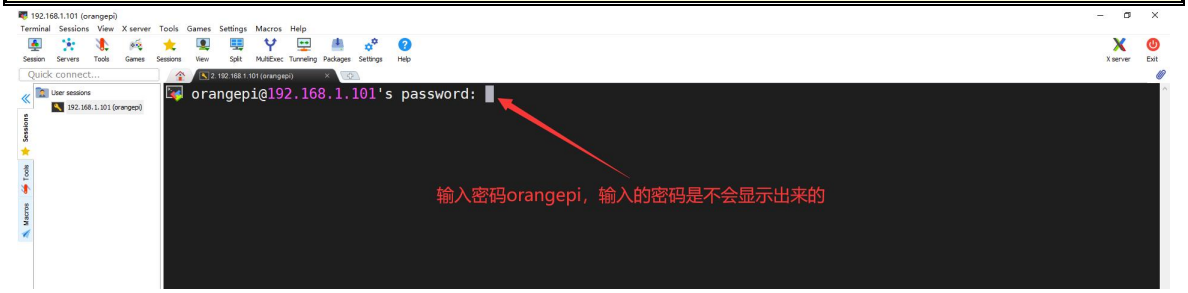
e) 最后点击 **OK** 即可

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

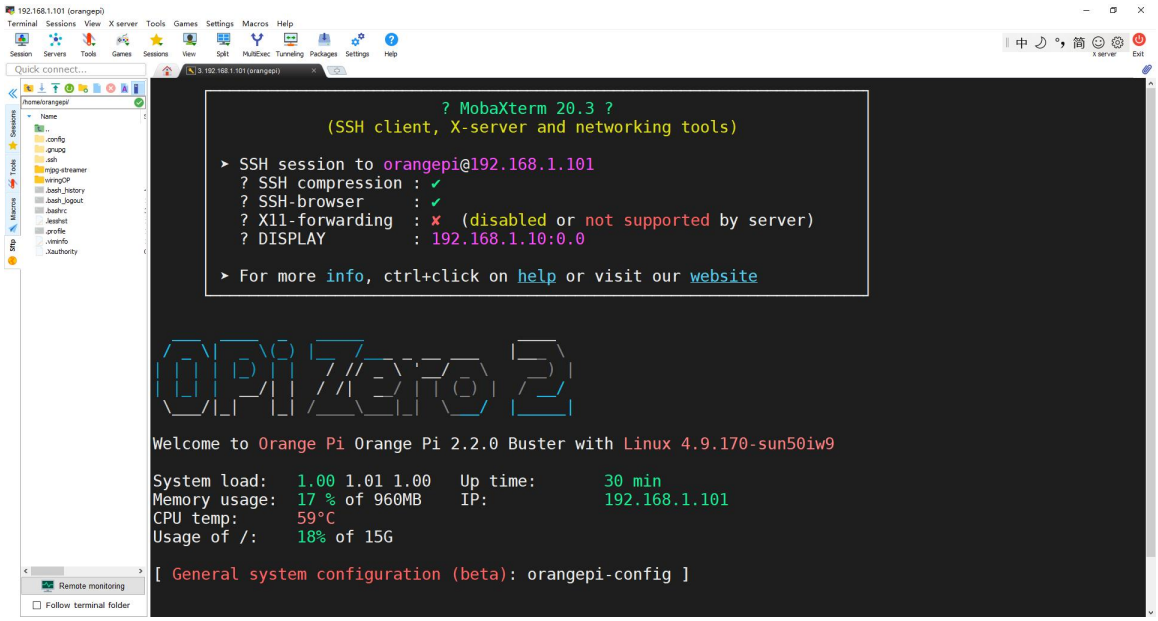


d. 然后会提示需要输入密码，开发板 Linux 系统 orangepi 用户的默认密码为 **orangepi**

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



e. 成功登录系统后的显示如下图所示



2) 然后下载 Kiauh 的源码

这里提供了两种下载 kiauh 源码的方法：

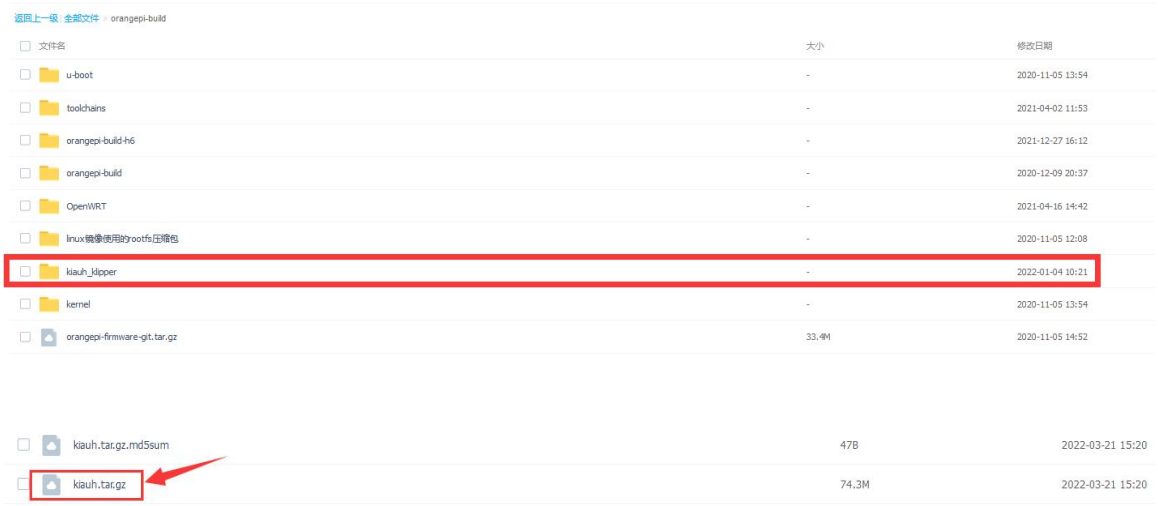
第一种是从百度云盘下载 Orange Pi 提供的 kiauh 源码压缩包进行测试。Orange Pi 提供的 kiauh 源码压缩包解决了由于 github 无法访问导致安装失败的问题，另外还修复了 KlipperScreen 安装失败的问题。

第二种是从 kiauh 的 github 仓库下载。但是使用这种方法下载源码，如果不会解决开发板 linux 系统访问 github 的问题，安装 klipper 基本是很难成功的，并且安装速度极慢。

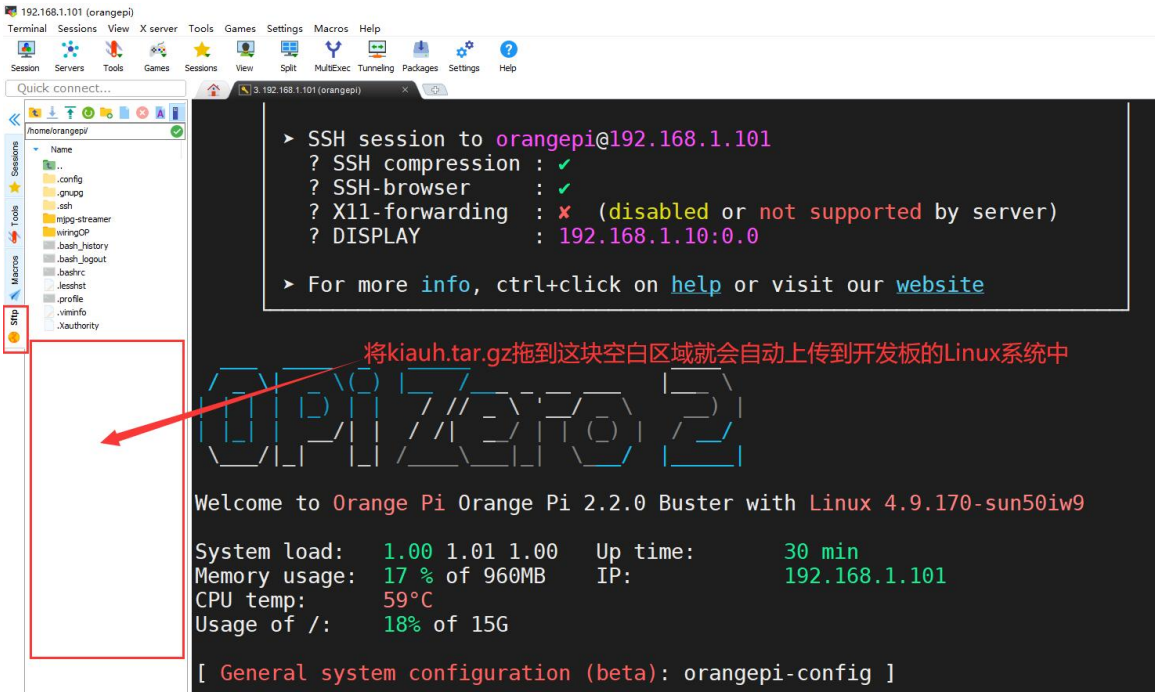
a. 第一种：从百度云盘下载 Orange Pi 提供的 kiauh 的源码压缩包的方法（**推荐方法**）

a) 从下面的百度云盘链接可以下载 Orange Pi 提供的 kiauh 源码压缩包。进入 **kiauh_klipper** 文件夹中就能看到

链接：https://pan.baidu.com/s/1vWQmCmSYdH7iCDFyKpJtVw
提取码：zero



b) 下载完 **kiauh.tar.gz** 压缩包后，首先将 **kiauh.tar.gz** 上传到开发板 linux 系统的**/home/orangepi** 目录中



c) 然后使用下面的命令解压下 **kiauh.tar.gz**

运行完 tar zxf kiauh.tar.gz 命令后，正常是不会有任何打印输出的（没有输出就是最好的结果），然后运行下 ls 命令可以看到当前目录下多了一个名为 kiauh 的文件夹。

ls 命令中的 l 是小写字母 L，不是数字 1，也不是小写字母 i。

```
orangepi@orangepi:~$ tar zxf kiauh.tar.gz
orangepi@orangepi:~$ ls
```

kiauh	kiauh.tar.gz

d) Orange Pi 提供的 Kiauh 源码的压缩包和从 github 下载的 Kiauh 源码的区别如下所示：

下面这一部分有关源码的修改说明如果看不懂在说什么请直接忽略，并不影响后面的安装。

i. Orange Pi 提供的 Kiauh 源码中有个 githhub_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

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

ii. 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

```
orangepi@orangepi:~/kiauh$ git status .
```

iii. 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

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

b. 第二种：从 github 下载 kiauh 的源码的命令如下所示（**如果没有 kiauh 安装 klipper 经验，不会解决 github 的网络问题，请不要使用这种方法来下载 kiauh 的源码，是很难安装成功的**）

```
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

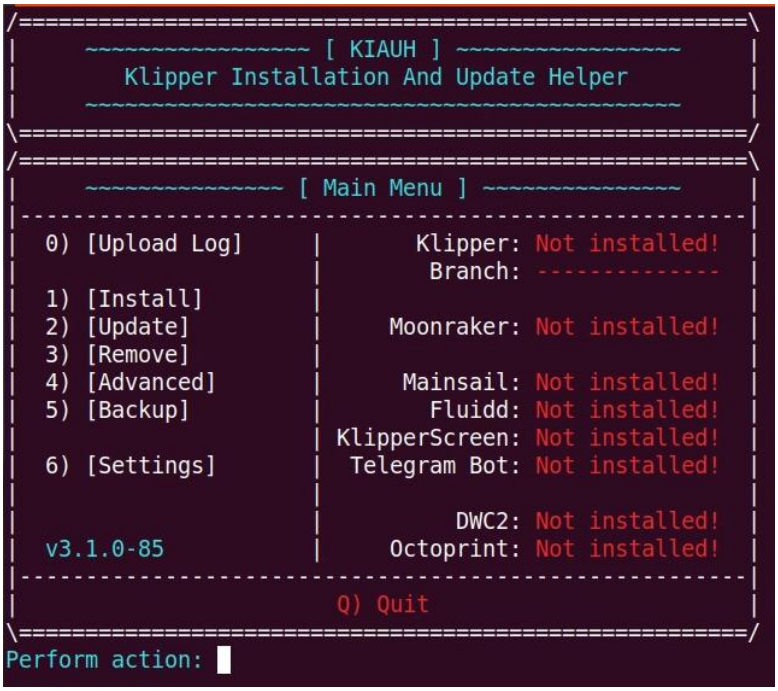
cd kiauh 命令是进入 kiauh 源码的意思，如果你已经在 kiauh 源码目录中了，就不需要再执行了。

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

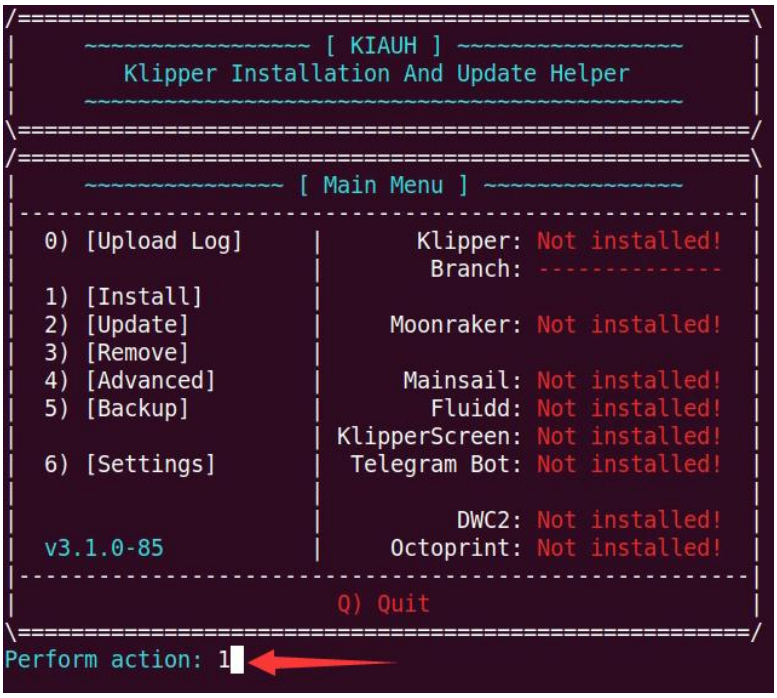
4) 然后就可以在 Kiauh 的源码目录下运行 kiauh.sh 脚本

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

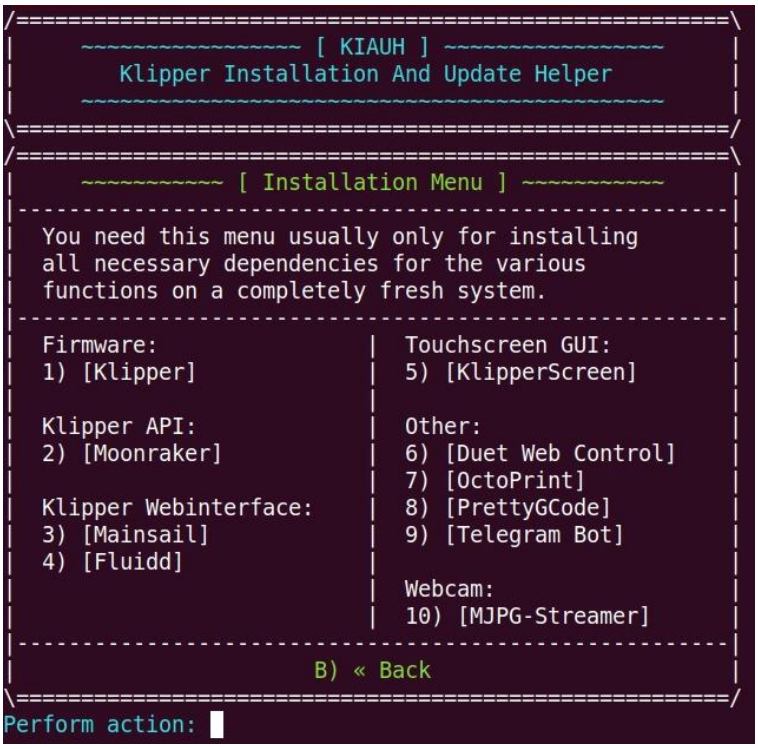
5) 运行 kiauh.sh 后会弹出下面的选择界面，可以看到 Kiauh 提供了很多的操作选项，还能显示 Klipper 等相关软件的安装状态



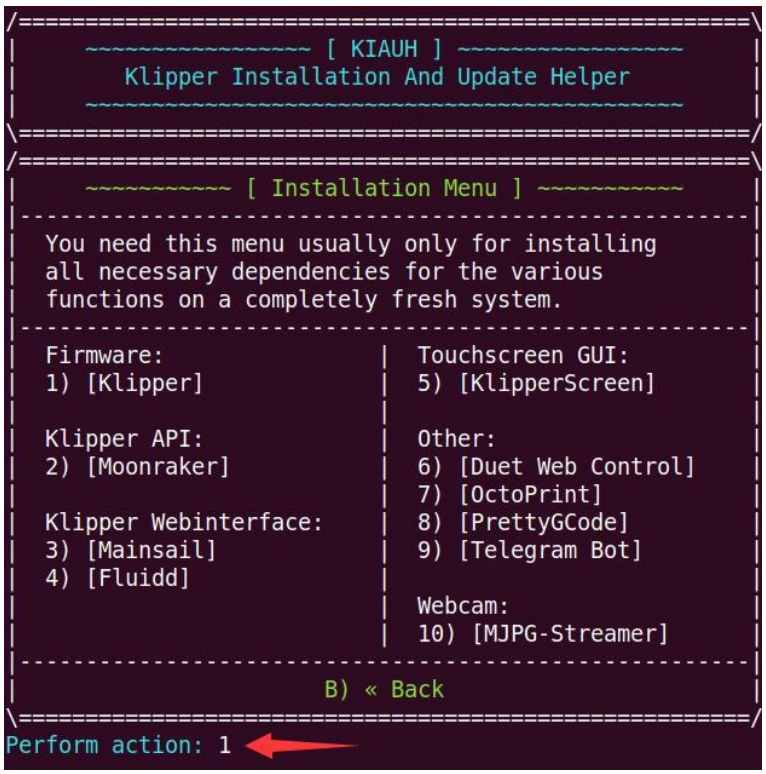
6) 然后就可以在 **Perform action:** 后面输入 **1** 后回车，选择 **1) [Install]**



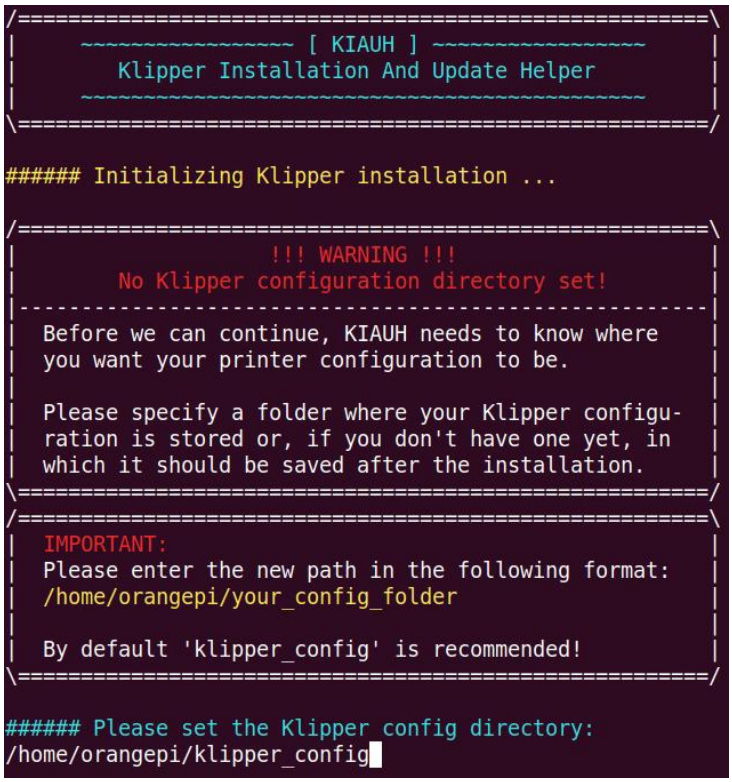
7) 然后会弹出下面的安装选择界面



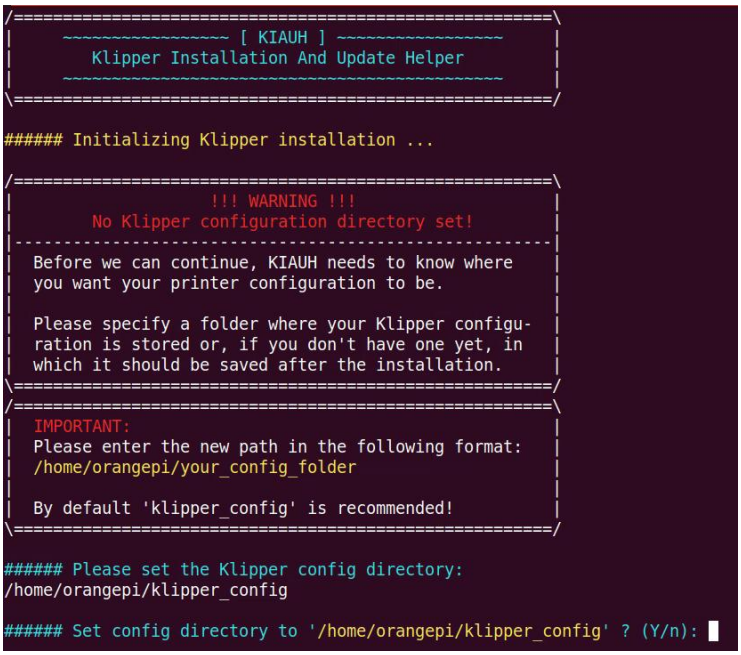
8) 我们首先在 **Perform action:** 后面输入 **1** 然后回车就会开始安装 Klipper 的过程



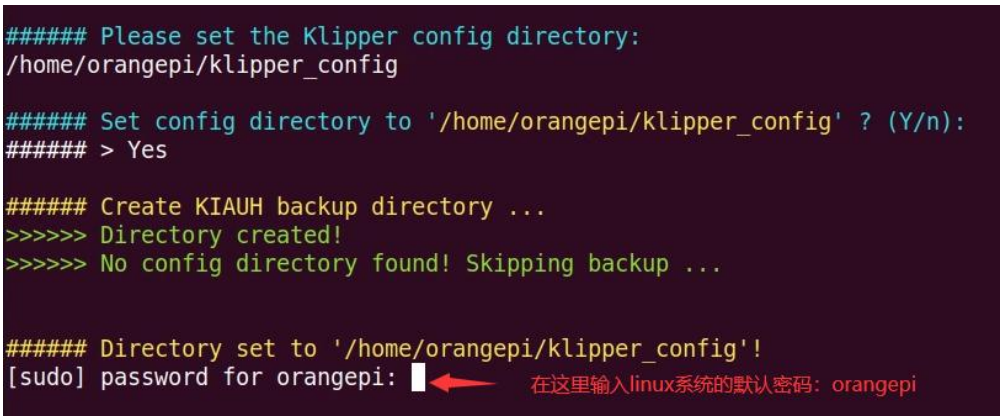
9) 开始安装 Klipper 后首先会提醒需要设置 Klipper 配置文件所在文件夹的路径，默认为**/home/orangepi/klipper_config**，如果不需要修改，直接回车即可



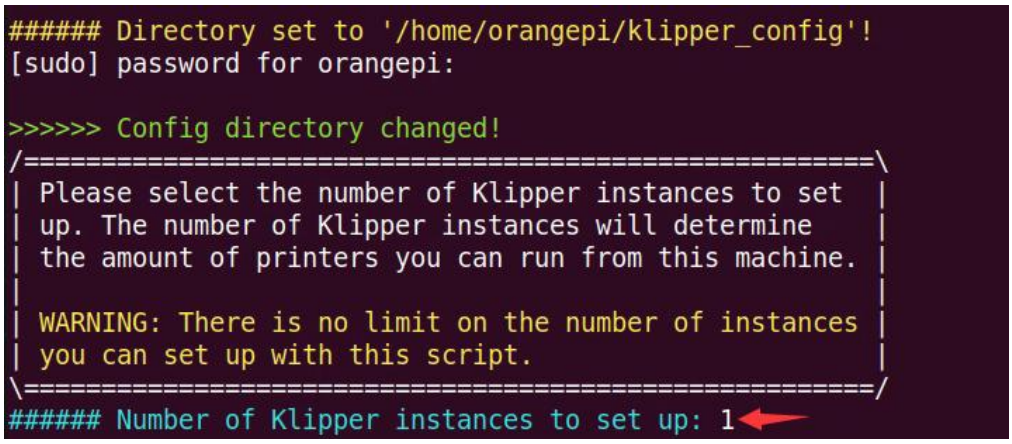
然后会弹出下面的提示信息，默认为 Y，直接回车即可



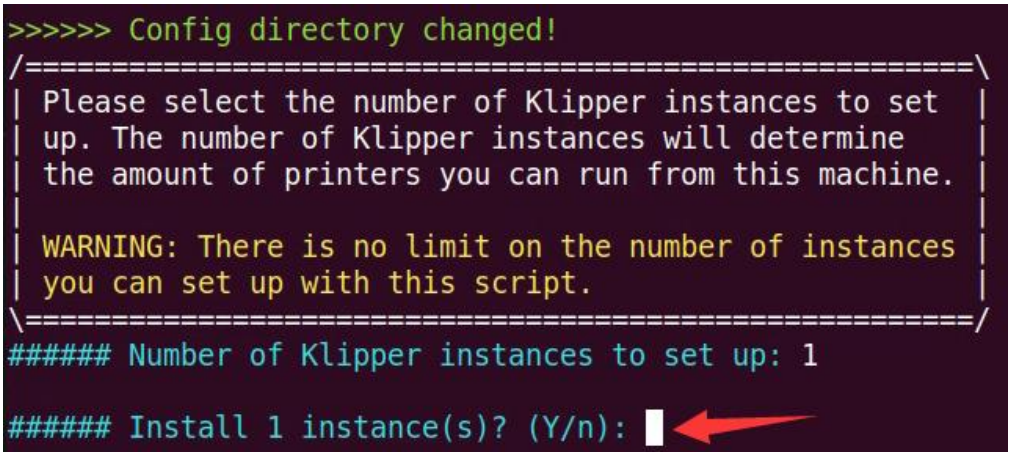
10) 然后会提示需要输入 Linux 系统的密码，默认为 orangepi



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



然后继续回车确认即可



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（**使用 Orange Pi 提供的 kiauh 源码不会有下载代码这个过程，所以也就不会由于网络问题出错**）

```
###### 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!
```

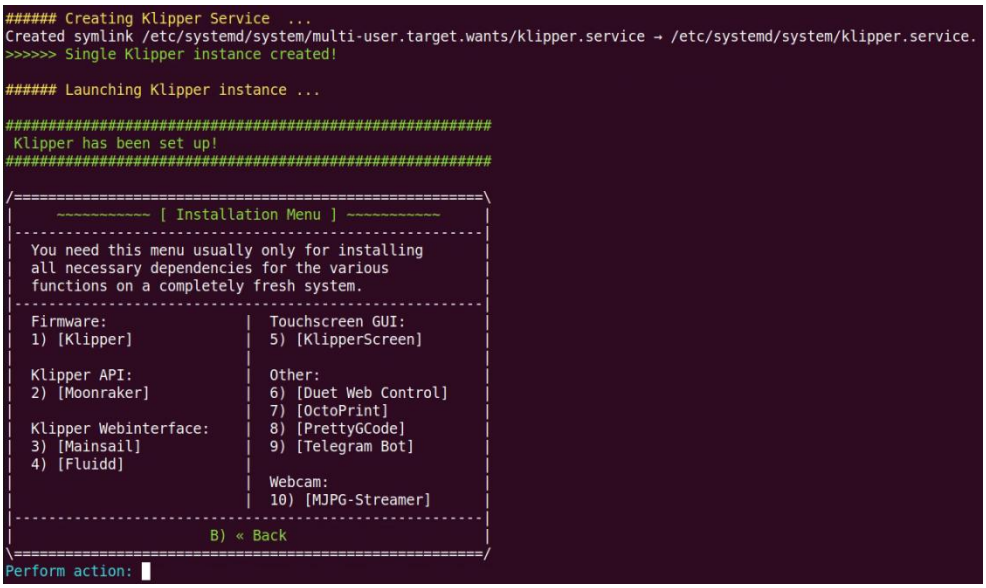
b. 然后会自动安装依赖包，请确保下载和安装的过程不会由于网络的原因出错

```
###### 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).
```

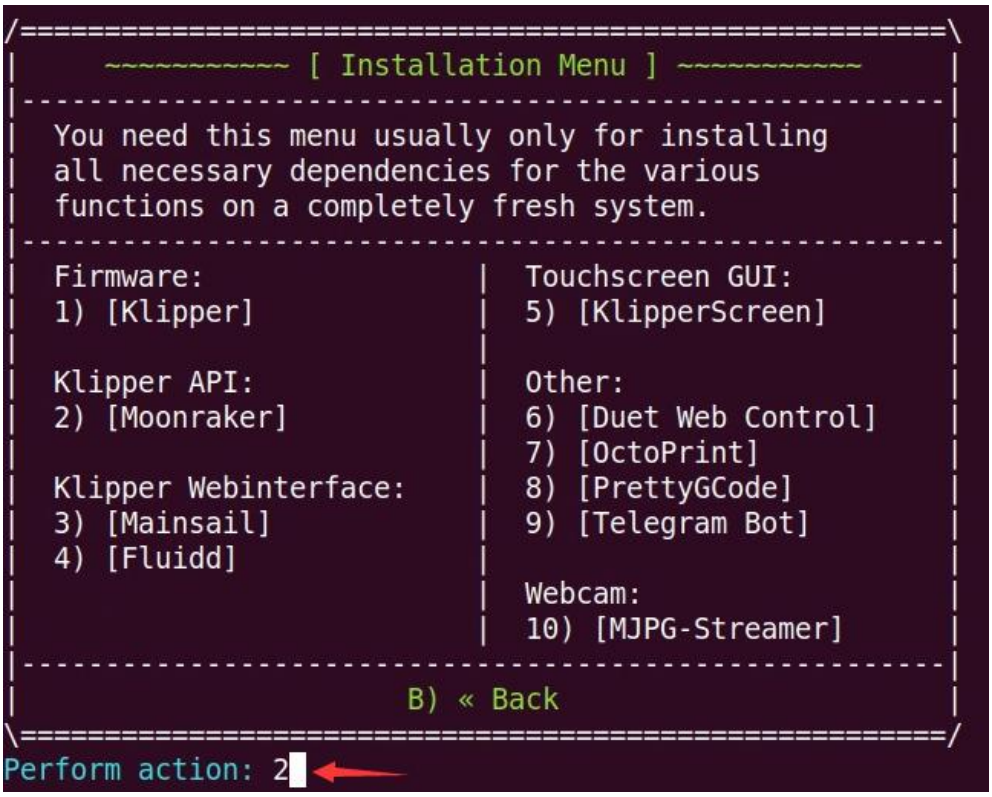
c. 然后自动会安装 Python 的虚拟环境，请确保下载和安装的过程不会由于网络的原因出错

```
###### 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
eed-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
```

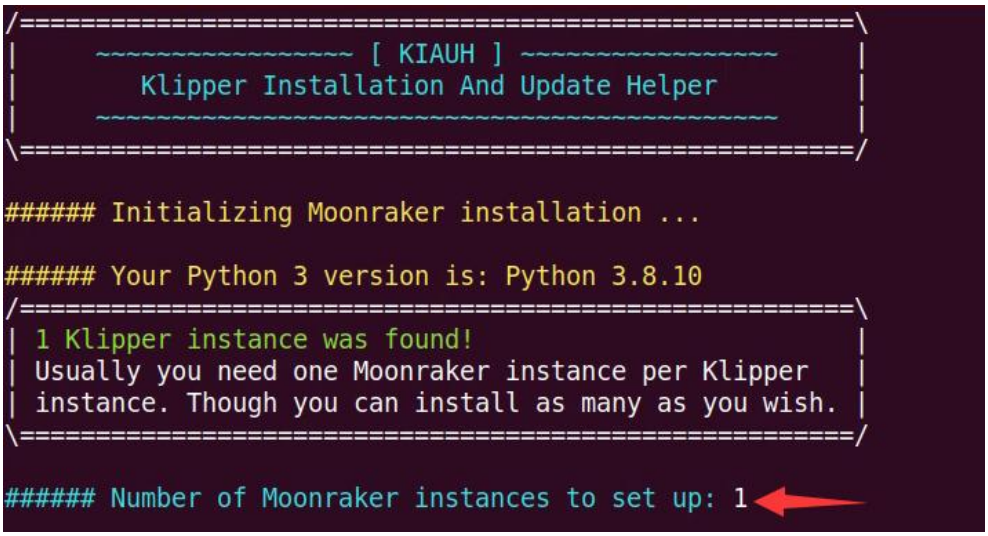
d. 安装完成后会显示下面的信息，并会自动返回 Kiauh 的安装界面



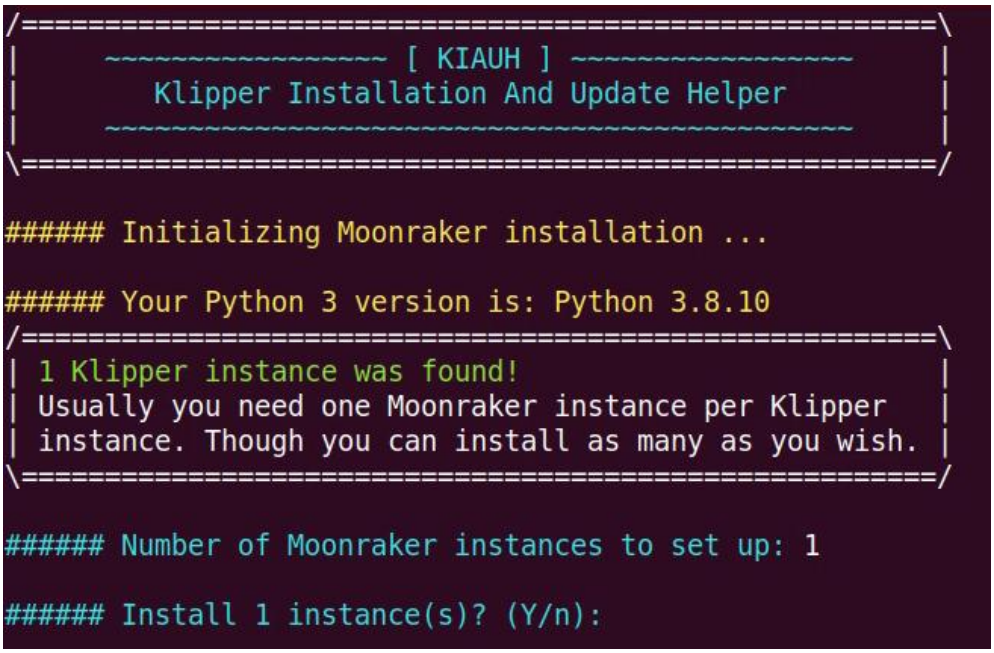
13) 然后开始安装 Klipper API——Moonraker，在 **Perform action:** 后面输入 **2** 后回车即可



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



然后继续回车确认即可



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

a. 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.
```

b. 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（**使用 Orange Pi 提供的 kiauh 源码不会有下载代码这个过程，所以也就不会由于网络问题出错**）

```
###### 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...
```

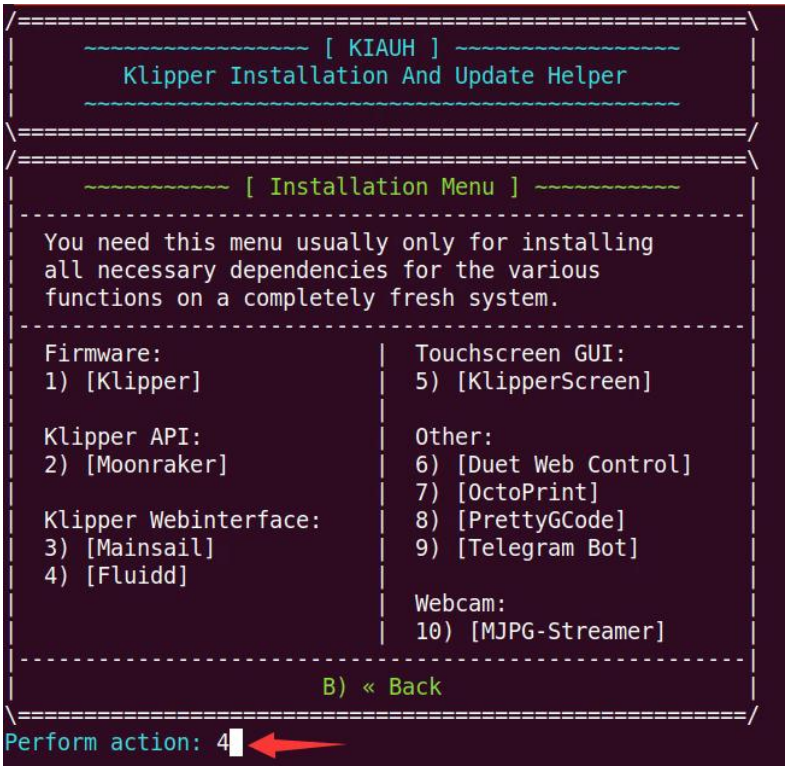
e. 然后会自动安装 Python 的虚拟环境，请确保下载和安装的过程不会由于网络的原因出错

```
###### 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,Xo
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-py2.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) [Fluidd]**等，这里我们测试下安装**4) [Fluidd]**的过程，要安装**4) [Fluidd]**在**Perform action:** 后面输入**4**，然后回车即可



17) Klipper 的 Web 操作界面——**Fluidd** 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（**使用 Orange Pi 提供的 kiauh 源码不会有下载 fluidd.zip 这个过程，所以也就不会由于网络问题出错**）

```
###### Downloading Fluidd ...
--[redacted]  13:33:33--  https://github.com/fluidd-core/fluidd/releases/download/v1.16.2/fluidd.zip
Resolving github.com (github.com)... 140.82.112.4
Connecting to github.com (github.com)|140.82.112.4|: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=AKIAIWNJYAX4CSVEH53A%2F20220101%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20220101T133334Z&X-Amz-Expires=300&X-Amz-Signature=76e8c06f5c7a375dc51fd37056c9b0f5c07b37e62665db2334b1a412361e2a74&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=295836951&response-content-disposition=attachment%3B%20filename%3Dfluidd.zip&response-content-type=application%2Foctet-stream [following]
--[redacted]  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=AKIAIWNJYAX4CSVEH53A%2F20220101%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20220101T133334Z&X-Amz-Expires=300&X-Amz-Signature=76e8c06f5c7a375dc51fd37056c9b0f5c07b37e62665db2334b1a412361e2a74&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=295836951&response-content-disposition=attachment%3B%20filename%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

[redacted] 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:
```

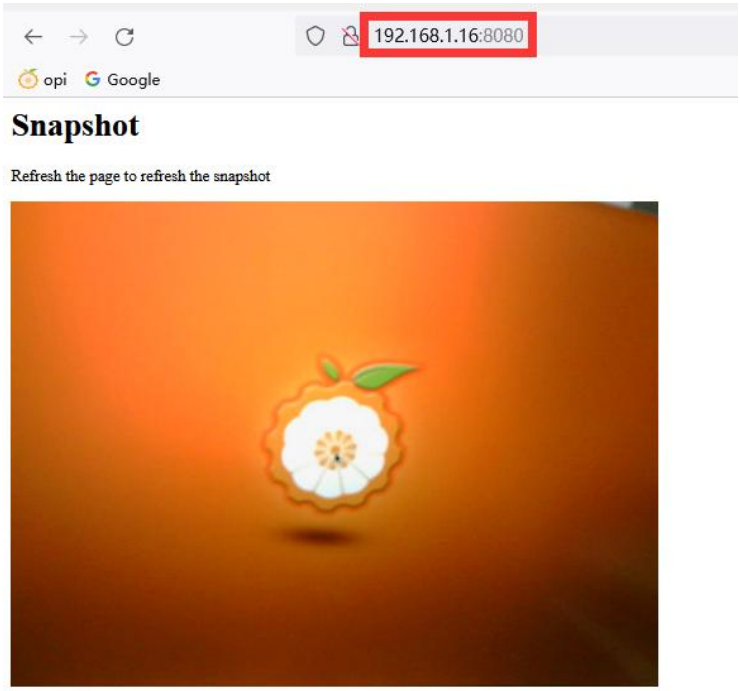
f. 然后会下载编译安装 MJPG-Streamer，这一步很容易由于网络问题下载失败，请特别注意下输出 log 和下图一致，没有报错（**使用 Orange Pi 提供的 kiauh 源码不会有下载代码这个过程，所以也就不会由于网络问题出错**）

```
###### 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
```

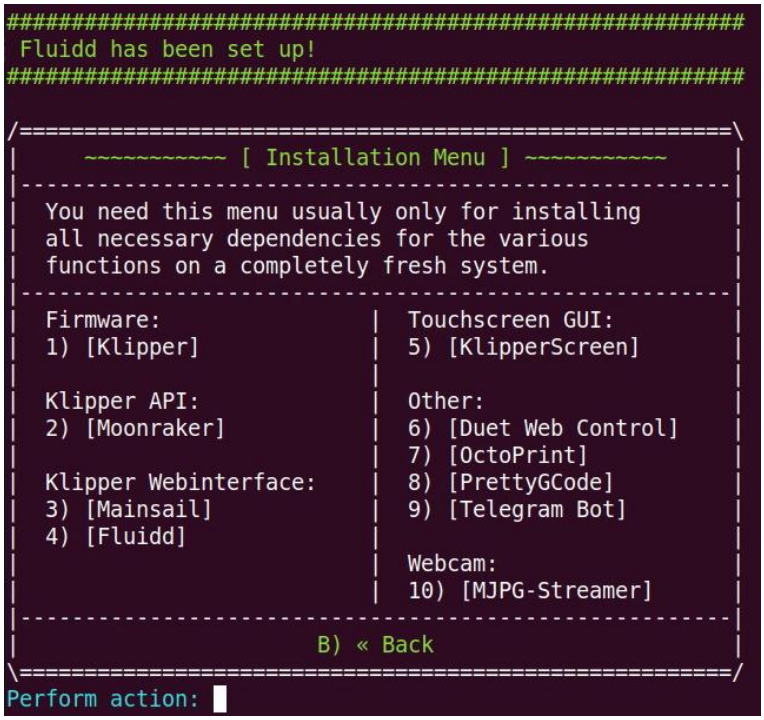
g. MJPG-Streamer 的访问地址如下所示，如果开发板有接 USB 摄像头，在浏览器中输入下面的地址就能看到 USB 摄像头的视频输出了

```
###########################################################
 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
```

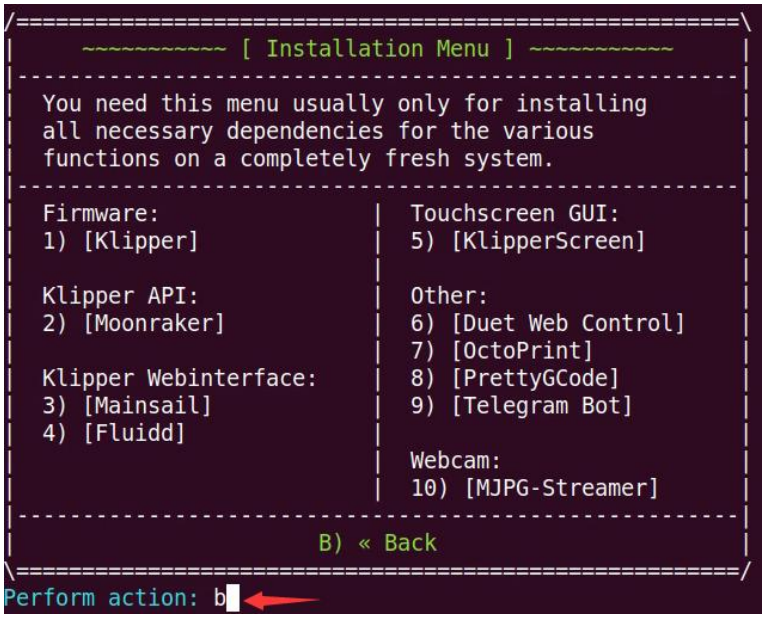
摄像头的输出如下所示

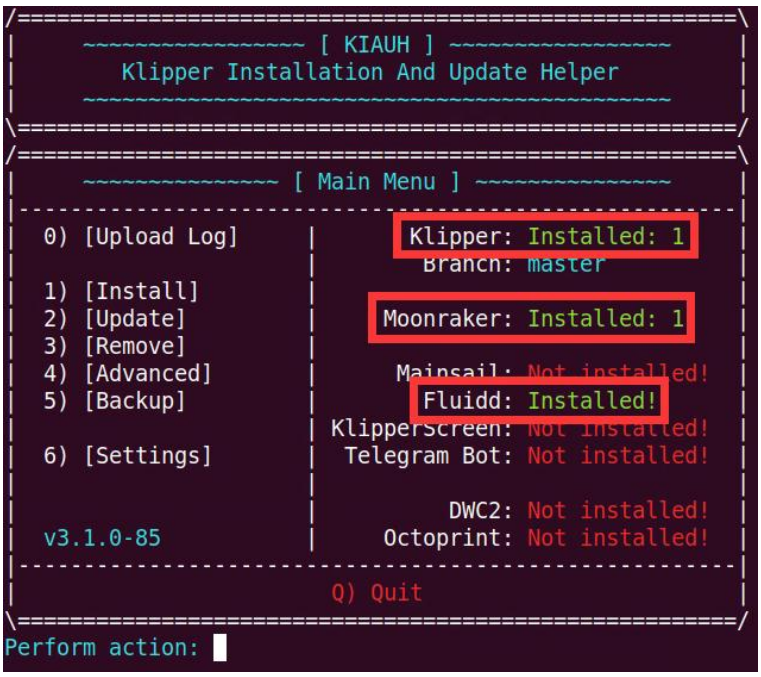


h. 安装完后的输出如下图所示



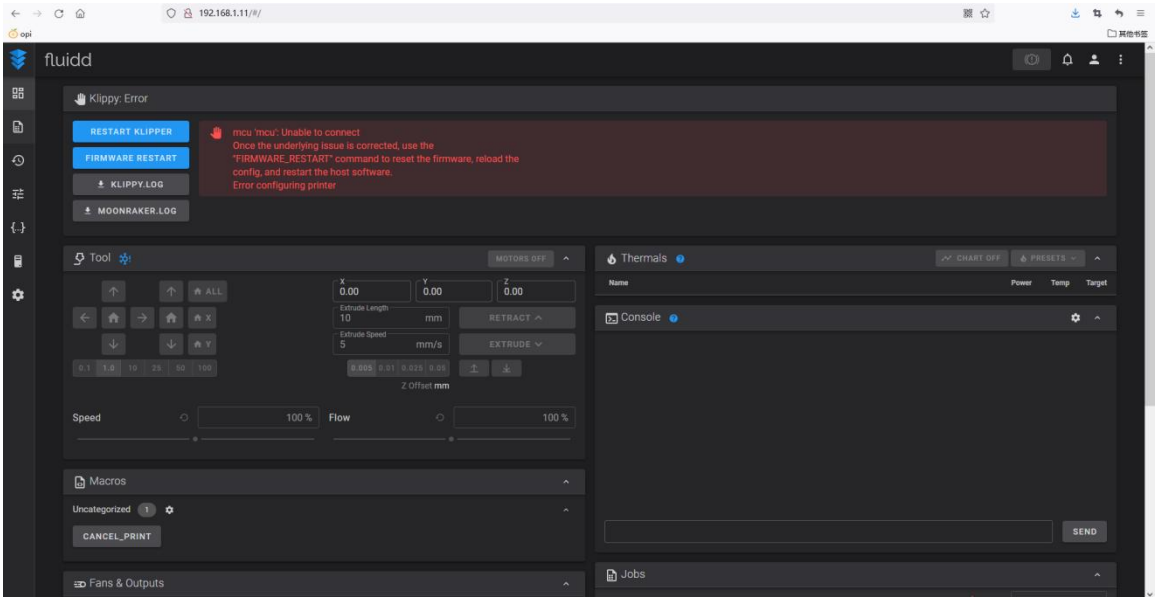
18) 然后在 **Perform action:** 后面输入 **B** 就能返回安装的主界面





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** 没有因为网络问题导致的源码下载失败的报错。

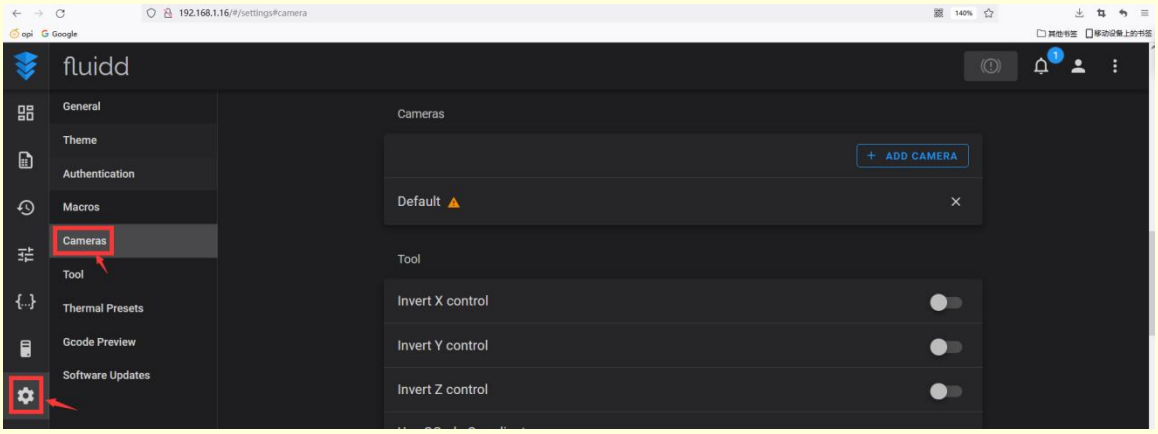
19) 最后在浏览器中输入开发板的 IP 地址就能看到 fluidd 的 Web 控制界面了



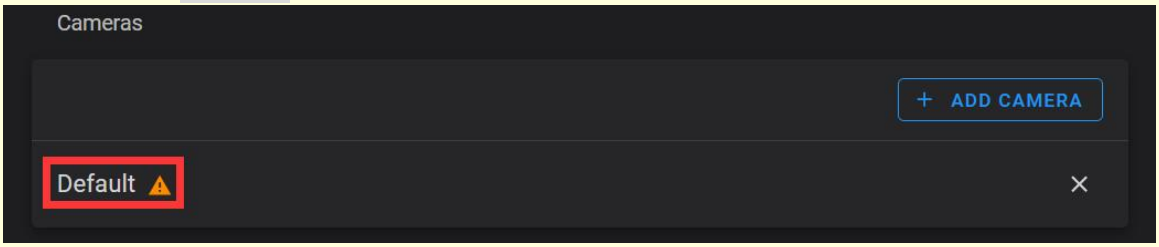
在 fluidd 的 Web 控制界面设置摄像头的步骤如下所示：

1. 首先请确保前面已正常安装了 MJPG-Streamer，并且能在浏览器中打开摄像头看到摄像头的输出画面

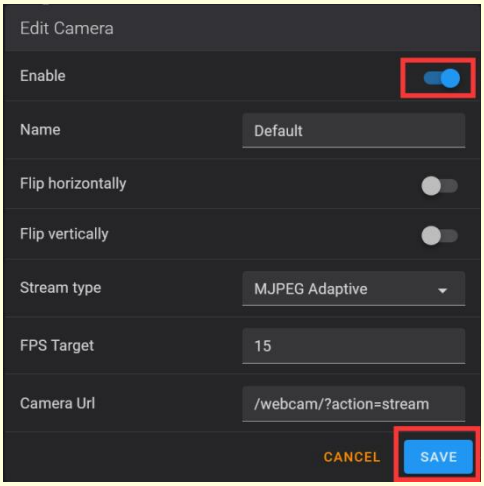
2. 然后在 Fluidd 中找到摄像头的设置界面



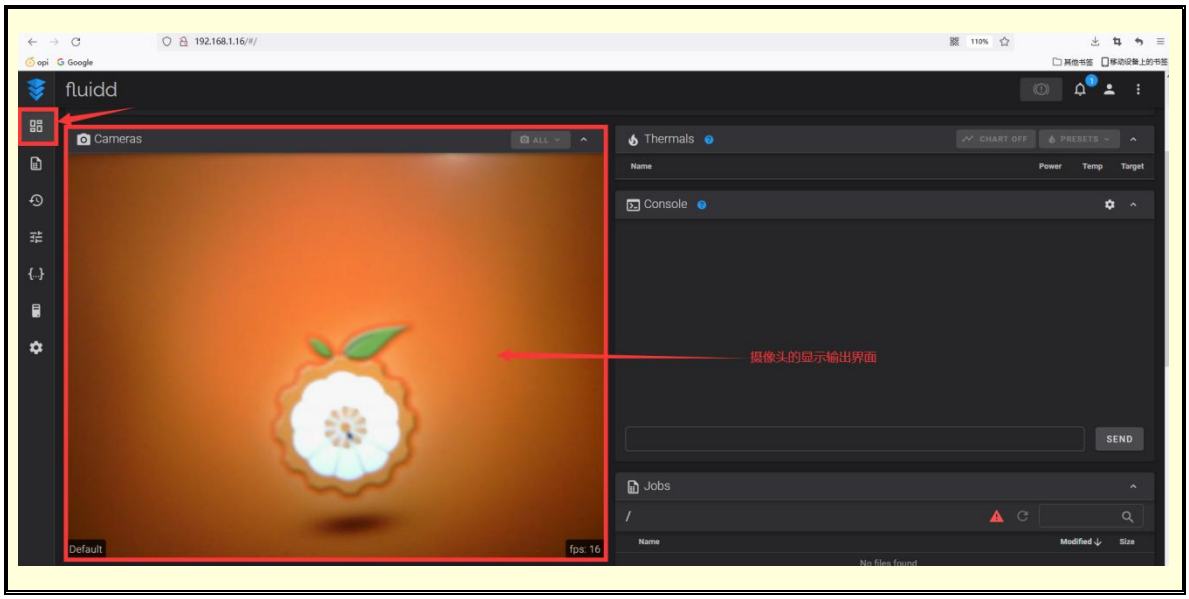
3. 然后点击 Default 打开摄像头的设置界面



4. 然后 Enable 摄像头，再点击 Save 保存



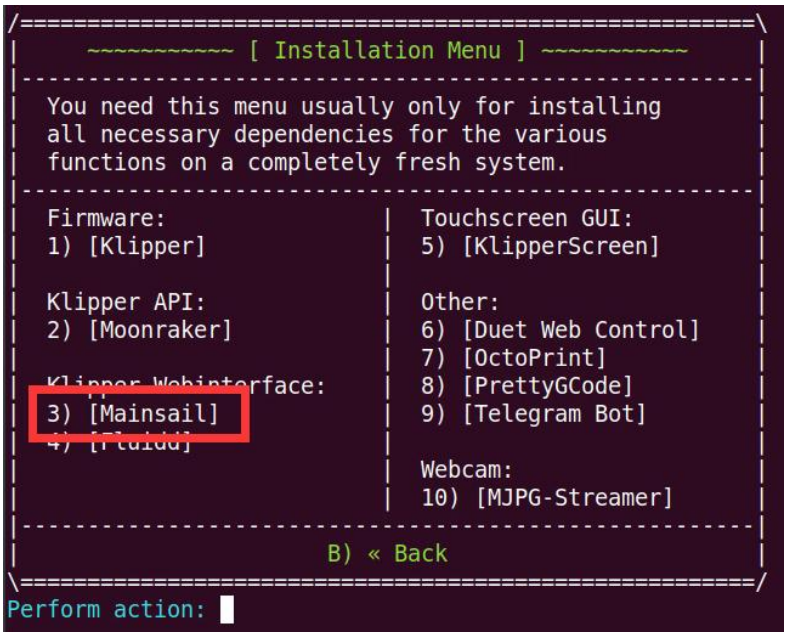
5. 然后回到主界面就能看到摄像头的输出画面了



20) 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]**

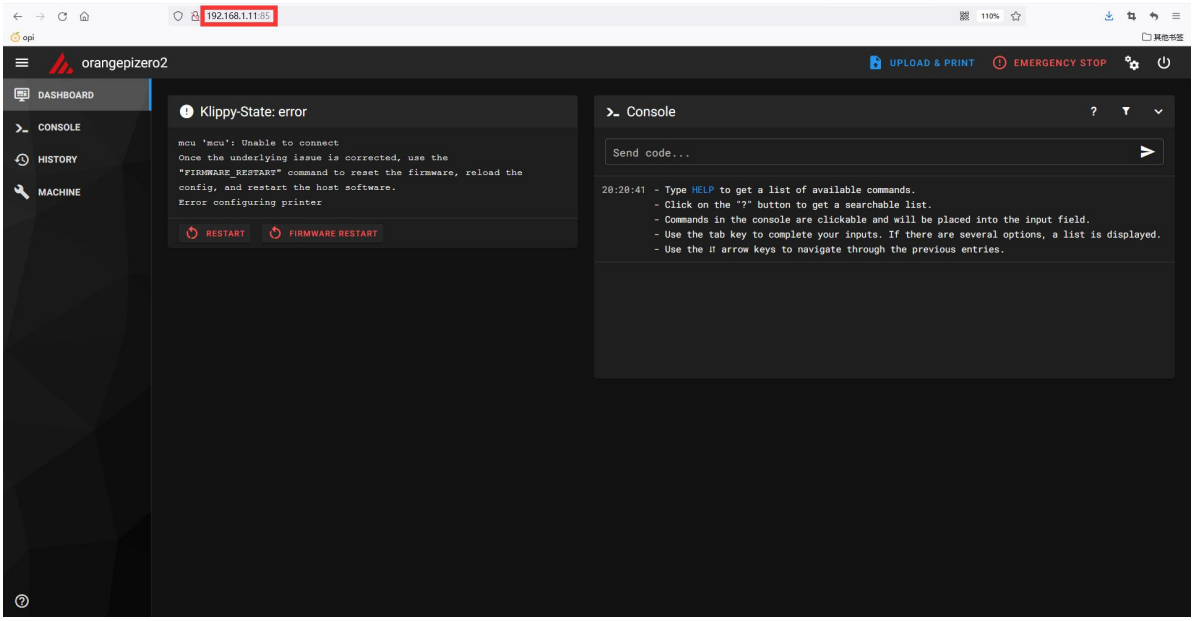
b. 安装 **3) [Mainsail]**选项如下所示



c. 如果前面已经安装了 Fluuid，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

```
###### Initializing Mainsail installation ...

###### Detected other enabled interfaces:
   ● Fluidd - Port: 80

/=======================================================\
|                    !!!WARNING!!!                      |
| You need to choose a different port for Mainsail!     |
| The following web interface is listening at port 80:  |
|                                                       |
|  ● Fluidd                                             |
|                                                       |
| Make sure you don't choose a port which was already   |
| assigned to one of the other webinterfaces and do NOT |
| use ports in the range of 4750 or above!              |
|                                                       |
| Be aware: there is NO sanity check for the following  |
| input. So make sure to choose a valid port!           |
\=======================================================/
Please enter a new Port:
```

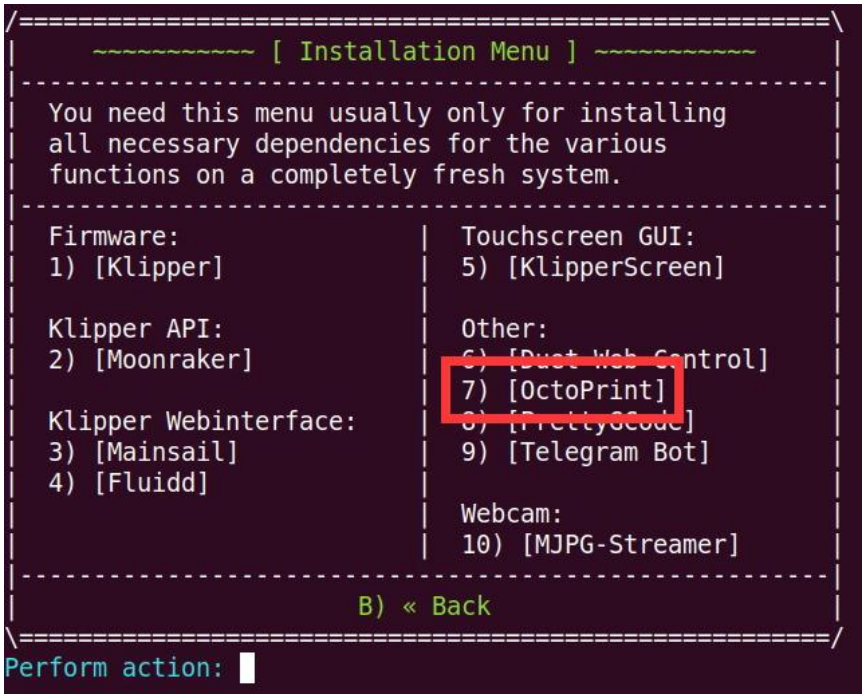
d. 安装完后在浏览器中输入开发板的 **IP 地址:端口号** 就能看到 Mainsail 的 Web 控制界面



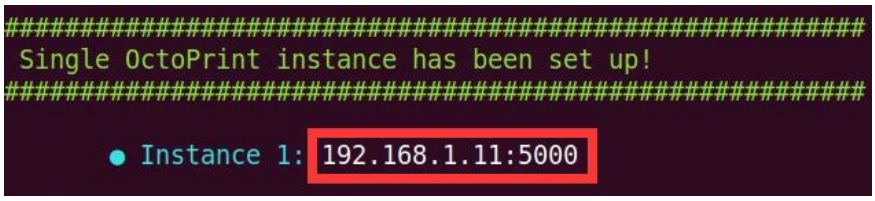
21) 使用 Orange Pi 提供的 Kiauh 源码压缩包安装 OctoPrint 示例

a. 首先请安装 **1) [Klipper]**和 **2) [Moonraker]**

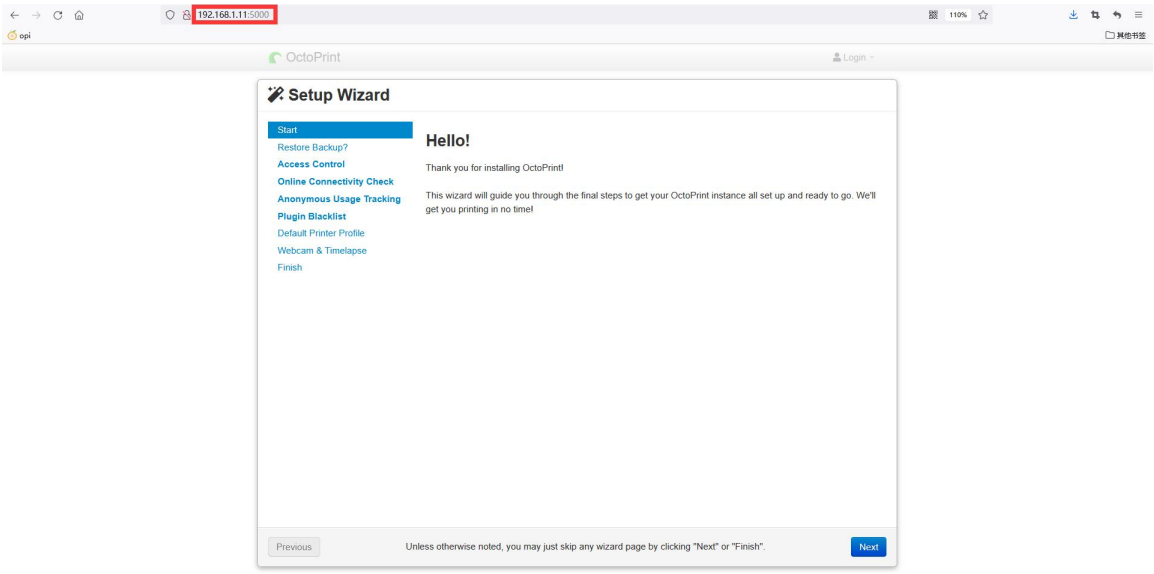
b. 安装 **7) [OctoPrint]**选项如下所示



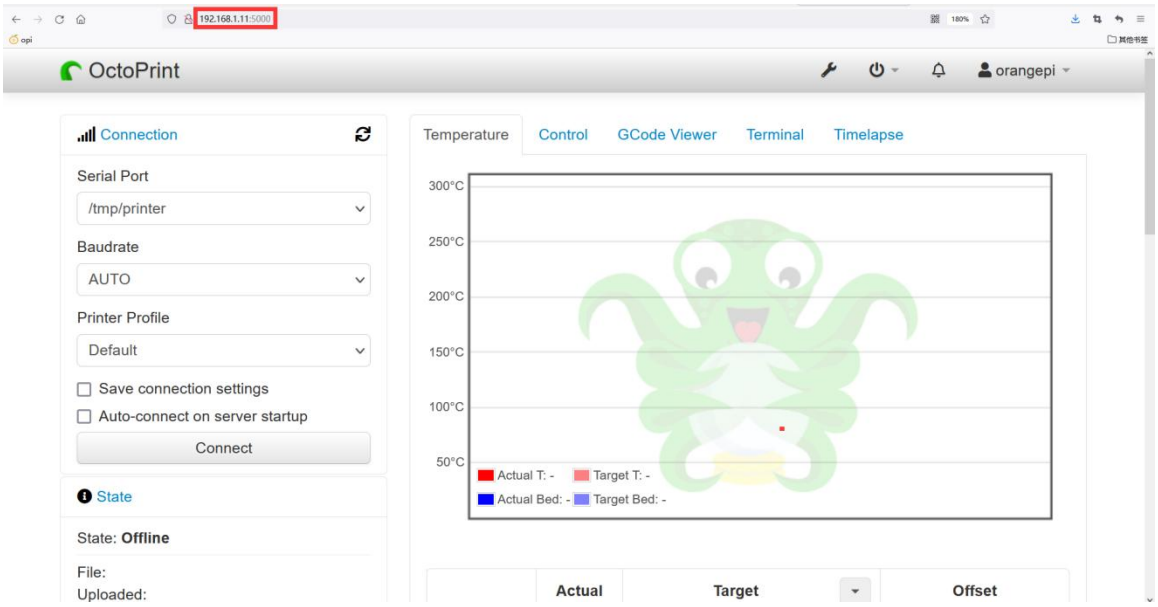
c. 安装完后就可以看到 OctoPrint 的 Web 界面访问地址，端口号为 5000



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



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

如果需要使用 kiauh 安装 KlipperScreen，请确保使用的 Ubuntu 或者 Debian 系统为 linux4.9 server 版本的系统，还有使用的 kiauh 源码为从 Orange Pi 的百度网盘下载的 Kiauh 源码压缩包。从 github 下载的 kiauh 源码安装 KlipperScreen 是无法正常使用的。

a. Orange Pi 对 KlipperScreen 源码的修改可以使用下面的命令来查看

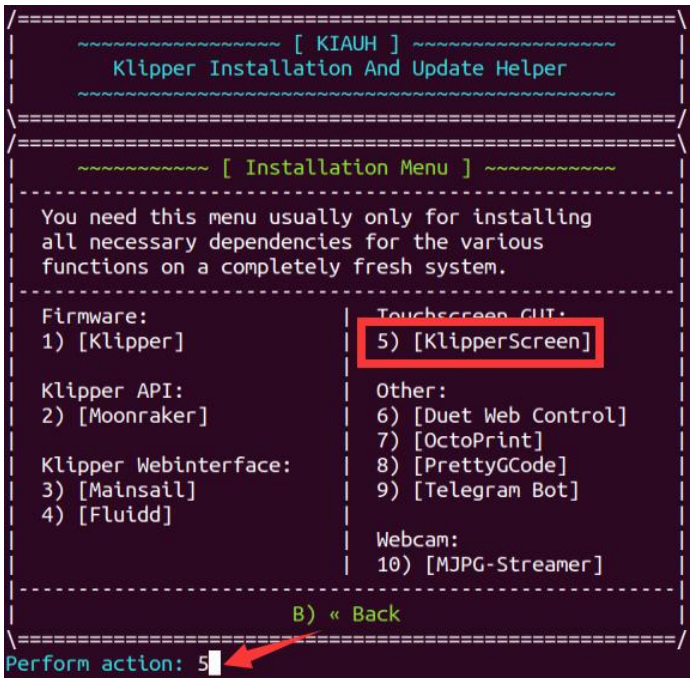
对源码不感兴趣的请直接跳过，不影响 KlipperScreen 的安装使用。

```
orangepi@orangepi:~$ cd kiauh/github_src/KlipperScreen
orangepi@orangepi:~/kiauh/github_src/KlipperScreen$ git status .
orangepi@orangepi:~/kiauh/github_src/KlipperScreen$ git diff .
```

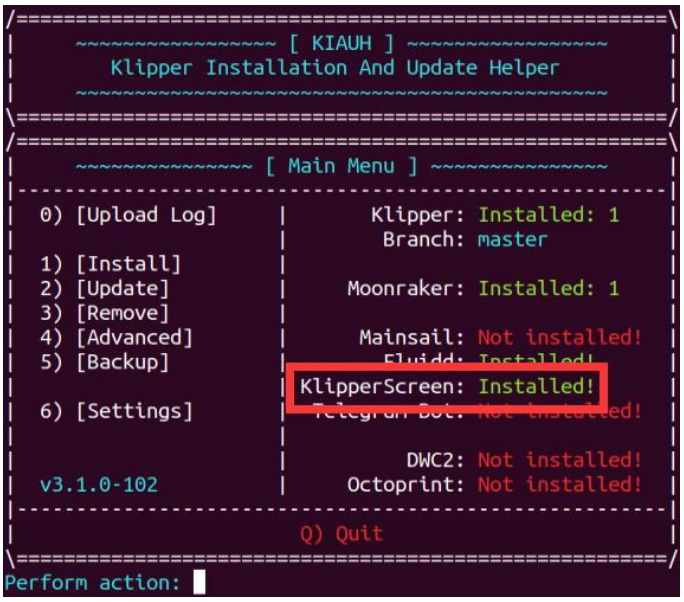
b. 安装 KlipperScreen 前请先将开发板接好 HDMI 显示器，并且确保 HDMI 能正常显示

c. 如果一定要用桌面版的 Ubuntu 或者 Debian 系统，首先请根据 **Linux 桌面版系统禁用桌面的方法**一节的说明关闭下桌面

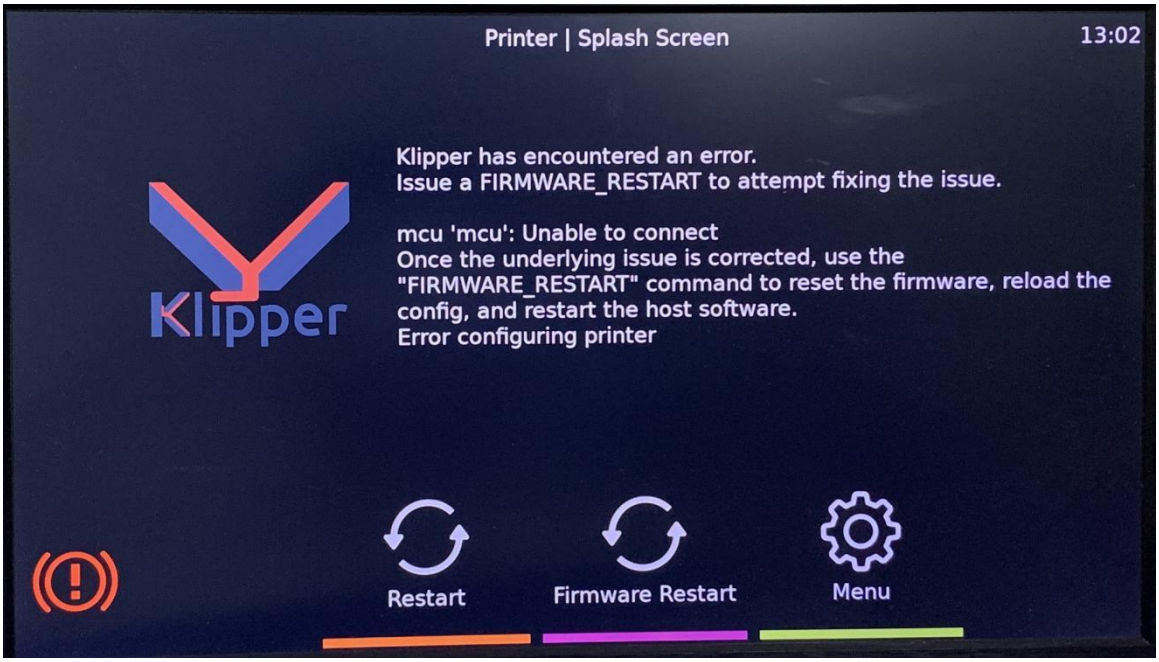
d. 然后在 Kiauh 中选择 **5) [KlipperScreen]**安装 KlipperScreen，**请仔细查看安装过程输出的打印信息，确保没有报错**



e. KlipperScreen 安装完后 Kiauh 显示如下所示。如果显示有问题，请卸载重装



f. KlipperScreen 安装完后，HDMI 显示器就能看到下图所示的界面了（如果看不到可以重启试下）



g. 设置 KlipperScreen 显示鼠标光标的方法为：

a) 首先运行下面的命令在**~/klipper_config/KlipperScreen.conf** 中设置 **show_cursor** 为 **True**

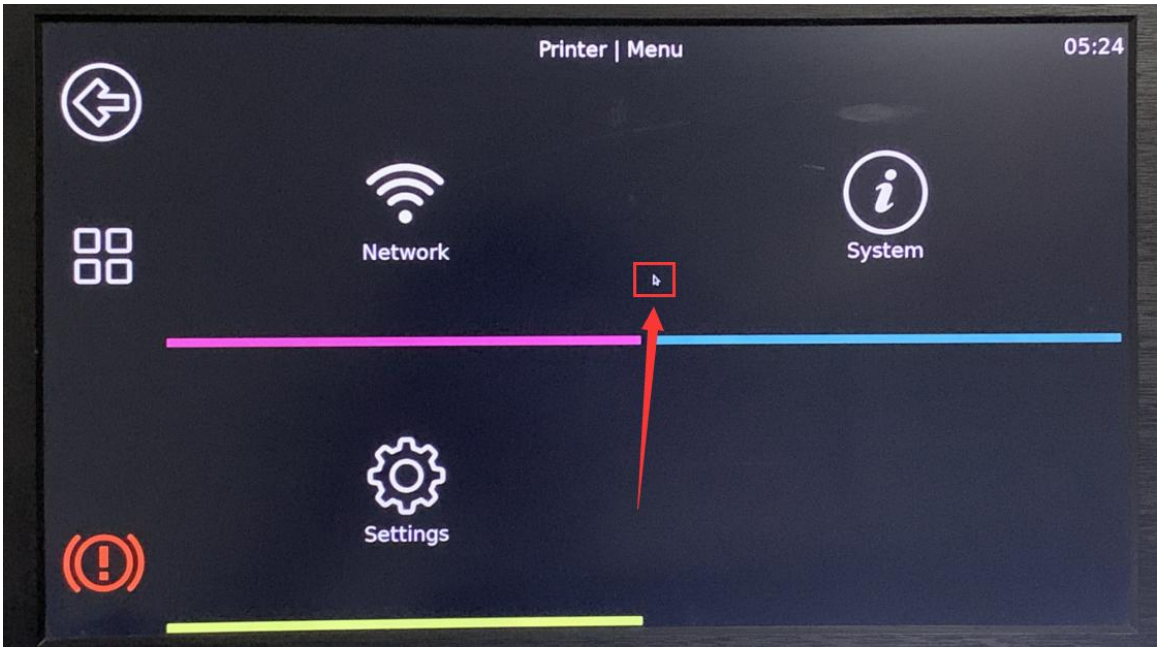
```
orangepi@orangepi:~$ cat <<EOF > ~/klipper_config/KlipperScreen.conf
[main]
show_cursor: True
EOF
```

b) 然后使用下面的命令检查下，确保配置没问题

```
orangepi@orangepi:~$ cat ~/klipper_config/KlipperScreen.conf
[main]
show_cursor: True
```

c) 然后重启下 **KlipperScreen** 服务，移动鼠标就能看到光标了

```
orangepi@orangepi:~$ systemctl restart KlipperScreen.service
```



23) 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**

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。

3.32. Python 相关说明

3.32.1. Python 源码编译安装的方法

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

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

1) 首先安装编译 Python 需要的依赖包

```
orangepi@orangepi:~$ sudo apt-get update
```

```
orangepi@orangepi:~$ sudo apt-get install -y build-essential zlib1g-dev  \
libncurses5-dev libgdbm-dev libnss3-dev libssl-dev libsqlite3-dev  \
libreadline-dev libffi-dev curl libbz2-dev
```

2) 然后下载最新版本的 Python3.9 源码并解压

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

3) 然后运行配置命令

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

4) 然后编译安装 Python3.9，编译时间大概需要半个小时左右

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

5) 安装完后可以使用下面的命令查看下刚安装的 Python 的版本号

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

6) 然后更新下 pip

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

3.32.2. Python 更换 pip 源的方法

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

1) 首先安装下 **python3-pip**

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

2) Linux 下永久更换 pip 源的方法

a. 先新建**~/.pip** 目录，然后添加 **pip.conf** 配置文件，并在其中设置 pip 的源为

清华源

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

b. 然后使用 pip3 安装 Python 库速度就会很快了

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

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

3.33. 安装 Docker 的方法

Docker 官方的提供的安装文档链接如下所示：

Debian 系统: https://docs.docker.com/engine/install/debian/
Ubuntu 系统: https://docs.docker.com/engine/install/ubuntu/

1) 老版本的 Docker 安装包叫做 docker、docker.io 或者 docker-engine，如果有安装这些包的话，需要先卸载它们，命令如下所示：

```
orangepi@orangepi:~$ sudo apt-get remove -y docker docker-engine docker.io  \
containerd runc
```

2) 然后添加 docker 官方的软件仓库

a. Debian 系统使用的命令如下所示

```
orangepi@orangepi:~$ sudo apt update
orangepi@orangepi:~$ sudo apt-get install -y ca-certificates curl gnupg lsb-release
orangepi@orangepi:~$ curl -fsSL https://download.docker.com/linux/debian/gpg |  \
sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
orangepi@orangepi:~$ echo "deb [arch=$(dpkg --print-architecture)  \
signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]  \
https://download.docker.com/linux/debian  \
```

```
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

b. Ubuntu 系统使用的命令如下所示

```
orangepi@orangepi:~$ sudo apt update
orangepi@orangepi:~$ sudo apt-get install -y ca-certificates curl gnupg lsb-release
orangepi@orangepi:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg |   \
sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
orangepi@orangepi:~$ echo "deb [arch=$(dpkg --print-architecture)   \
signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]   \
https://download.docker.com/linux/ubuntu   \
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

3) 然后安装 Docker Engine

```
orangepi@orangepi:~$ sudo apt update
orangepi@orangepi:~$ sudo apt install -y docker-ce docker-ce-cli containerd.io
```

注意：Debian Buster 安装完后如果报错请输入下面的命令来解决：

```
orangepi@orangepi:~$ echo 1 | update-alternatives --config iptables > /dev/null
orangepi@orangepi:~$ sudo systemctl restart docker
```

4) 然后可以将当前用户加入到 docker 用户组，这样不需要 sudo 就能运行 docker 命令

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

注意：需要退出重新登录系统才能生效，重启系统也可以

5) 验证 docker 的状态

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

```
orangepi@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.
```

7) 设置 docker 仓库为国内源的方法

a. 创建 **/etc/docker/daemon.json** 文件，在其中加入下面的配置

```
orangepi@orangepi:~$ sudo vim /etc/docker/daemon.json
{
  "registry-mirrors": [
    "https://docker.mirrors.ustc.edu.cn"
  ]
}
```

b. 然后输入下面的命令重启下 docker 服务（或者重启下系统）

```
orangepi@orangepi:~$ sudo systemctl restart docker
```

3. 34. Home Assistant 的安装方法

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

3. 34. 1. 通过 docker 安装

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

2) 然后可以搜索下 Home Assistant 的 docker 镜像

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

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

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

如果开发板连接的网络是比较快的，但是下载 docker 镜像时候特别慢，请检查下是不是忘了配置 docker 镜像的下载地址为国内源。配置方法在安装 Docker 的方法一节里面有讲。

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

```
orangepi@orangepi:~$ docker images homeassistant/home-assistant
REPOSITORY                    TAG       IMAGE ID       CREATED        SIZE
homeassistant/home-assistant  latest    bfa0ab9e1cf5   2 months ago   1.17GB
```

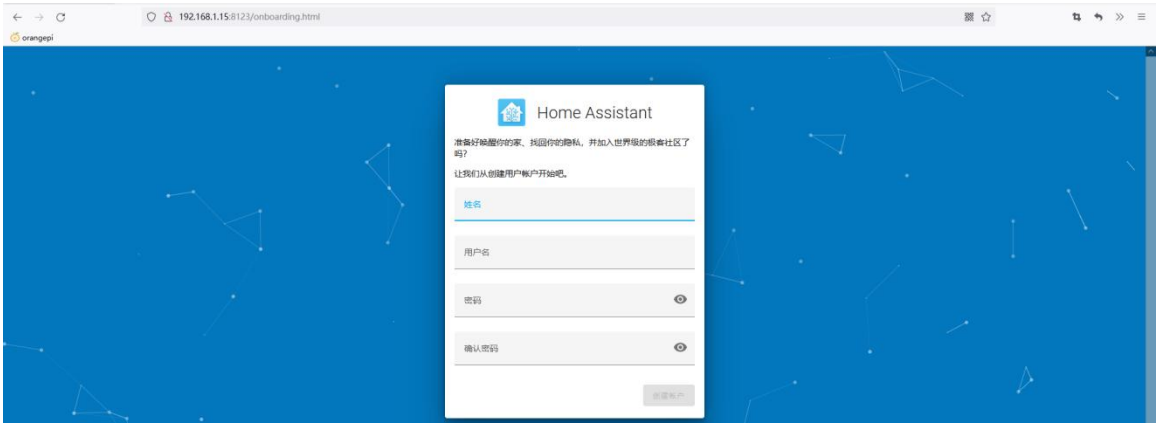
5) 此时就可以运行 Home Assistant 的 docker 容器了

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

```
--network=host \
homeassistant/home-assistant:latest
```

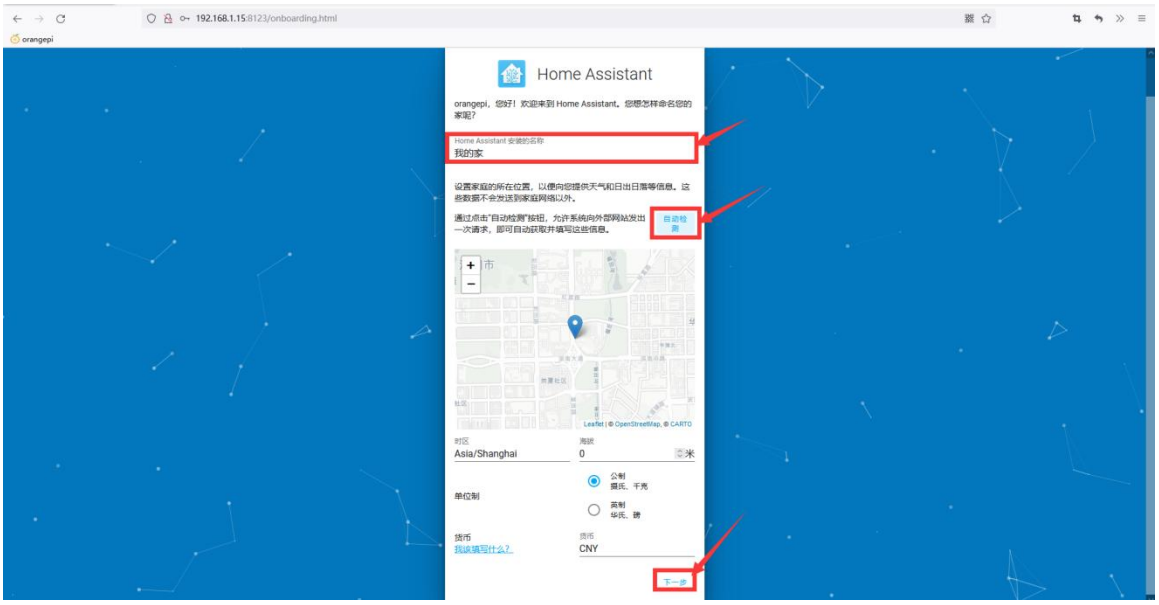
6) 然后在浏览器中输入【开发板的 IP 地址:8123】就能看到 Home Assistant 的界面

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

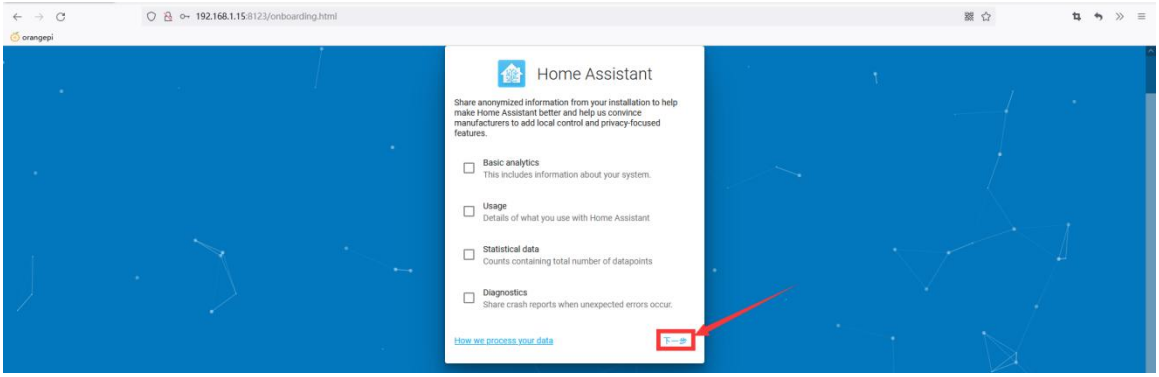


7) 然后输入**姓名、用户名**和**密码**再点击**创建账号**

8) 然后按照界面提示根据自己的喜好设置，再点击下一步

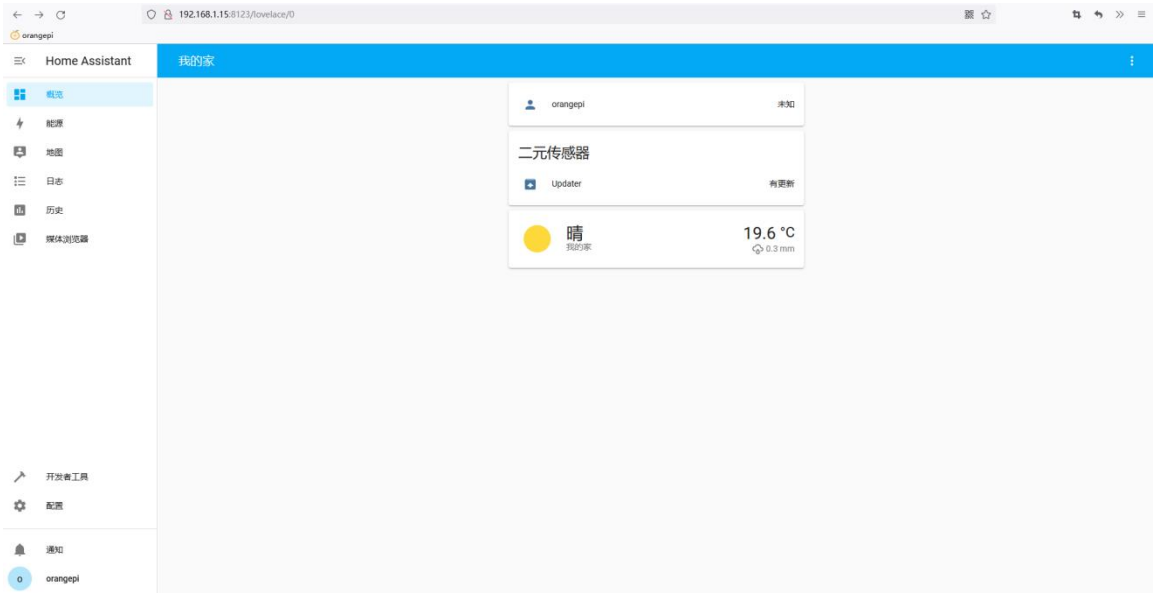


9) 然后点击下一步



10) 然后点击完成

11) Home Assistant 最终显示的主界面如下图所示



12) 停止 Home Assistant 容器的方法

a. 查看 docker 容器的命令如下所示

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

b. 停止 Home Assistant 容器的命令如下所示

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

c. 删除 Home Assistant 容器的命令如下所示

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

3.34.2. 通过 python 安装

安装前请先更换下 pip 的源为国内源，加快 Python 包的安装速度，配置方法见 Python 更换 pip 源的方法一节的说明。

1) 首先安装依赖包

```
orangepi@orangepi:~$ sudo apt-get update
orangepi@orangepi:~$ sudo apt-get install -y python3 python3-dev python3-venv   \
python3-pip libffi-dev libssl-dev libjpeg-dev zlib1g-dev autoconf build-essential   \
libopenjp2-7 libtiff5 libturbojpeg0-dev tzdata
```

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

Debian Bullseye 默认的 Python 版本就是 Python3.9，所以无需编译安装。
Ubuntu Jammy 默认的 Python 版本就是 Python3.10，所以也无需编译安装。

3) 然后创建 Python 虚拟环境

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

4) 然后安装需要的 Python 包

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

5) 然后就可以安装 Home Assistant Core

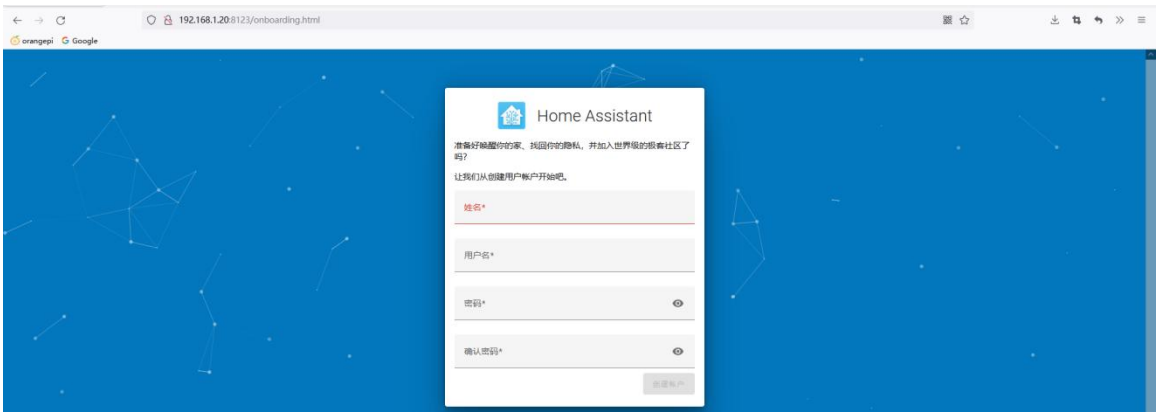
```
(homeassistant) orangepi@orangepi:/srv/homeassistant$ pip3 install homeassistant
```

6) 然后输入下面的命令就可以运行 Home Assistant Core

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

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

第一次运行 hass 命令时，会下载安装和缓存一些运行必须的库和依赖包。这个过程可能会花费几分钟的时间。注意，此时在浏览器中是无法看到 Home Assistant 的界面的，请等待一段时间后再刷新下。



3.35. OpenCV 的安装方法

3.35.1. 使用 apt 来安装 OpenCV

1) 安装命令如下所示

```
orangepi@orangepi:~$ sudo apt update
orangepi@orangepi:~$ sudo apt install -y libopencv-dev python3-opencv
```

2) 然后使用下面的命令打印 OpenCV 的版本号输出正常，说明 OpenCV 安装成功

a. Ubuntu22.04 中 OpenCV 的版本如下所示：

```
orangepi@orangepi:~$ python3 -c "import cv2; print(cv2.__version__)"
4.5.4
```

b. Ubuntu20.04 中 OpenCV 的版本如下所示：

```
orangepi@orangepi:~$ python3 -c "import cv2; print(cv2.__version__)"
4.2.0
```

c. Ubuntu18.04 中 OpenCV 的版本如下所示：

```
orangepi@orangepi:~$ python3 -c "import cv2; print(cv2.__version__)"
3.2.0
```

d. Debian10 中 OpenCV 的版本如下所示：

```
orangepi@orangepi:~$ python3 -c "import cv2; print(cv2.__version__)"
3.2.0
```

e. Debian11 中 OpenCV 的版本如下所示：

```
orangepi@orangepi:~$ python3 -c "import cv2; print(cv2.__version__)"
4.5.1
```

3.36. 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) 首先需要扩展下**/tmp** 空间的大小，设置完后需要**重启下开发板的 linux 系统**，命

令如下所示：

```
orangepi@orangepi:~$ sudo sed -i 's/nosuid/&,size=2G/' /etc/fstab
orangepi@orangepi:~$ sudo reboot
```

3) 重启后，可以看到/tmp 空间的大小变为 2G 了

```
orangepi@orangepi:~$ df -h | grep "/tmp"
tmpfs                   2.0G     12K  2.0G      1% /tmp
```

4) 然后在 linux 系统中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
```

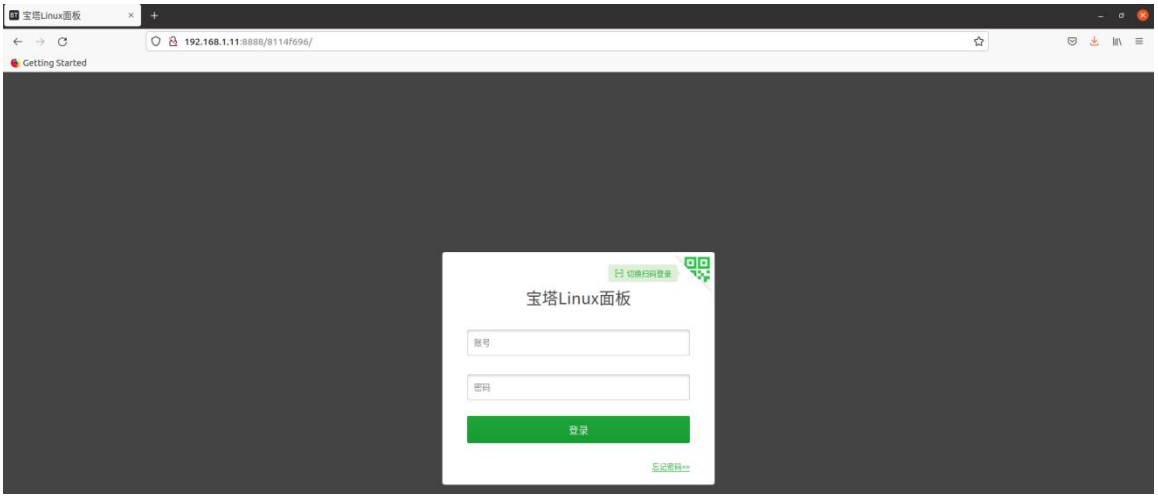
5) 然后宝塔安装程序会提醒是否安装 Bt-Panel 到/www 文件夹，此时输入 y 即可

```
+----------------------------------------------------------------------
| 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
```

6) 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

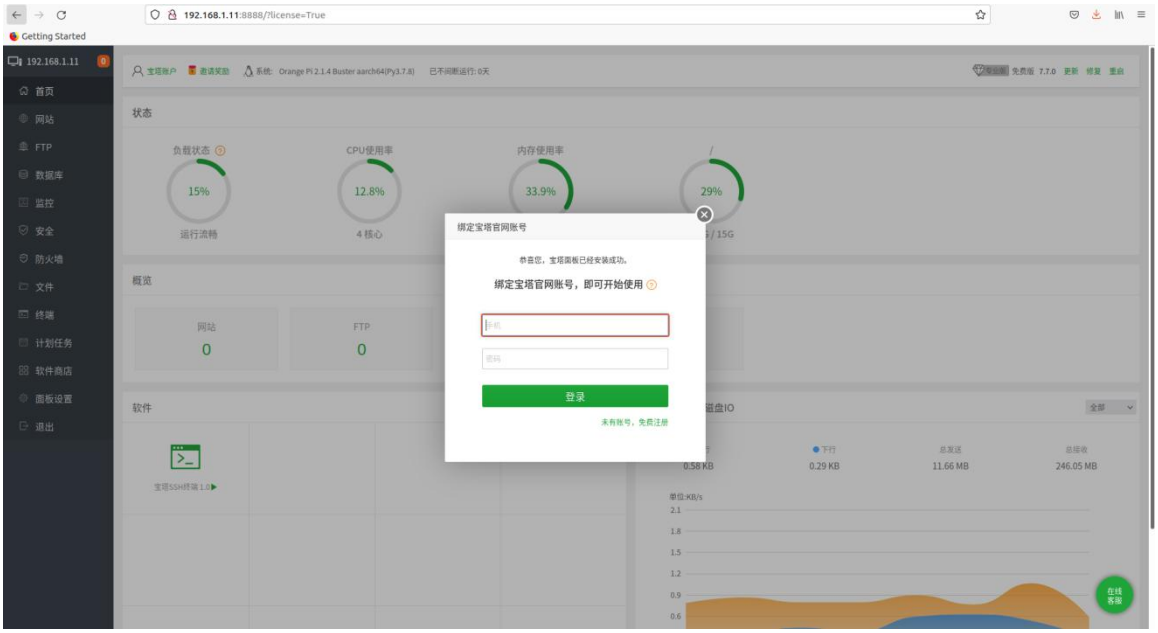
```
==================================================================
Congratulations! Installed successfully!
==================================================================
外网面板地址: http://[240e:3b7:3240:c3a0:d81f:613c:1739:3d6a]:8888/8114f696
内网面板地址: http://192.168.1.11:8888/8114f696
username: 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@orangepizero2:~$
```

7) 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

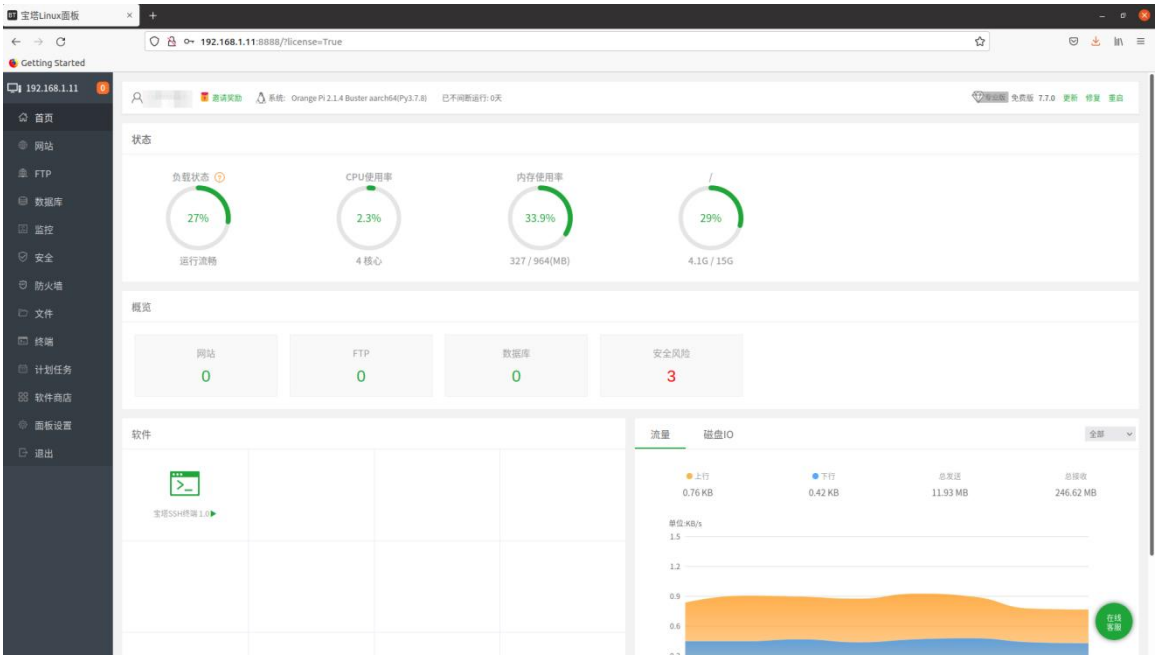


8) 成功登录宝塔后的会弹出下面的欢迎界面，首先请将中间的用户须知阅读完拖到最下面，然后就可以选择“我已同意并阅读《用户协议》”，接着点击“进入面板”就可以进入宝塔了

9) 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

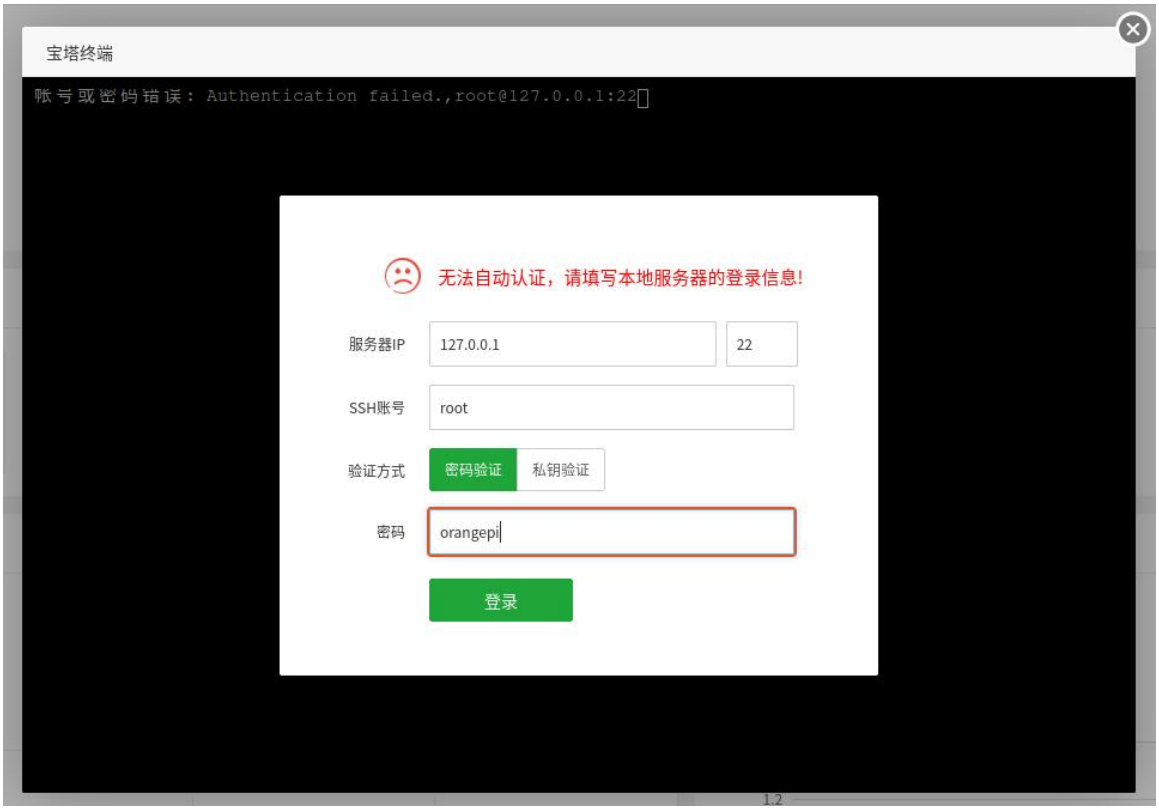


10) 最终显示的界面如下图所示，可以很直观的看到开发板 Linux 系统的一些状态信息，比如负载状态、CPU 的使用率、内存使用率和存储空间的使用情况等

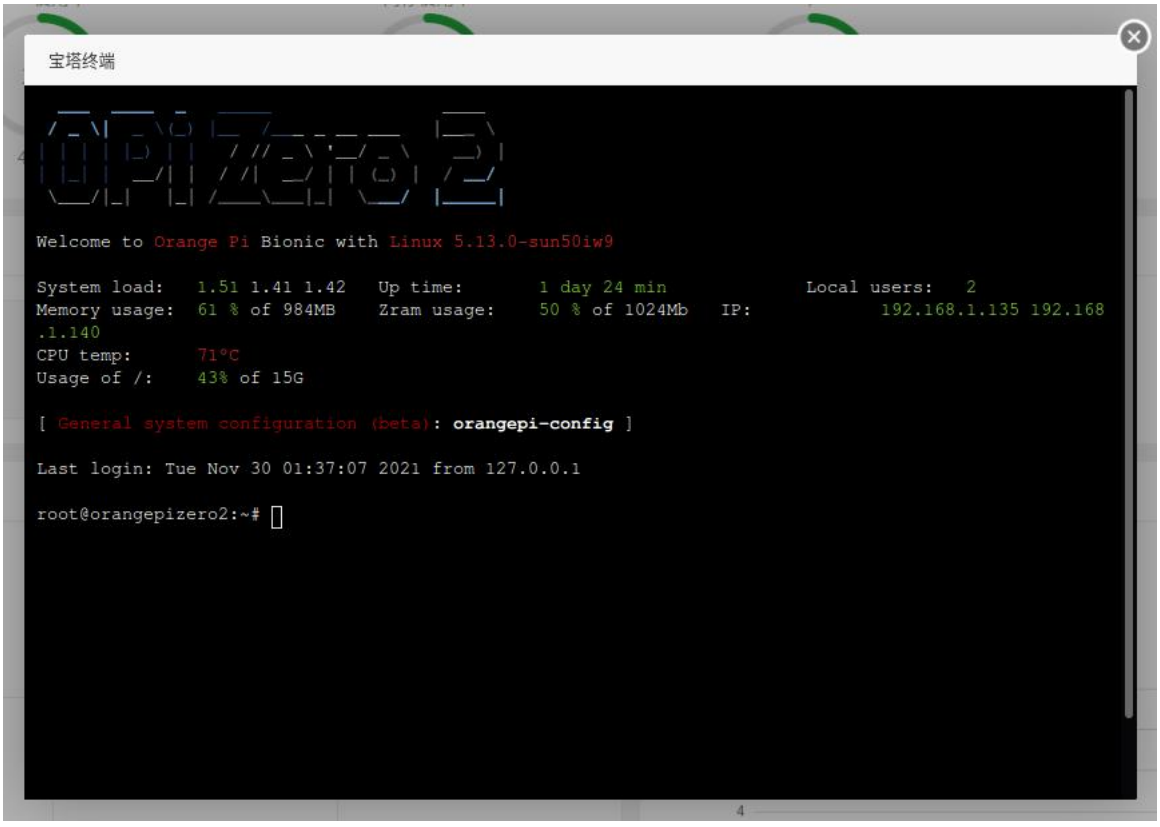


11) 测试宝塔的 SSH 终端登录

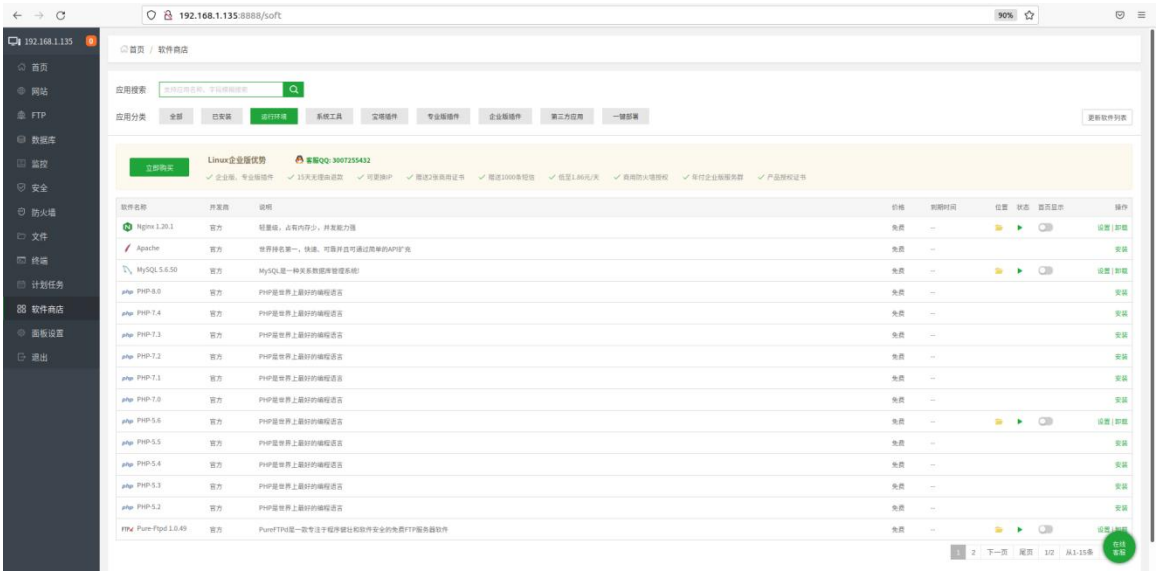
a. 打开宝塔的 SSH 终端后首先会提示需要输入开发板系统的密码，此时在密码框中输入 **orangepi**（默认密码，如果有修改请填写修改后的）即可



b. 成功登录后的显示如下图所示



12) 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



13) 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: 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
==================================================================
root@orangepizero2:~#
```

14) 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**

3.37. 远程登录 Linux 系统桌面的方法

相对于 VNC，更推荐使用 NoMachine 来远程登录 Linux 系统桌面

3.37.1. 使用 NoMachine 远程登录

请确保开发板安装的 Ubuntu 或者 Debian 系统为桌面版本的系统。另外 NoMachine 也提供了详细的使用文档，强烈建议通读此文档来熟悉 NoMachine 的

使用，文档链接如下所示：

https://knowledgebase.nomachine.com/DT10R00166

NoMachine 支持 Windows、Mac、Linux、iOS 和安卓平台，所以我们可以在多种设备上通过 NoMachine 来远程登录控制 Orange Pi 开发板。下面演示下在 Windows 中通过 NoMachine 来远程登录 Orange Pi 开发板的 Linux 系统桌面。其他平台的安装方法请参考下 NoMachine 的官方文档。

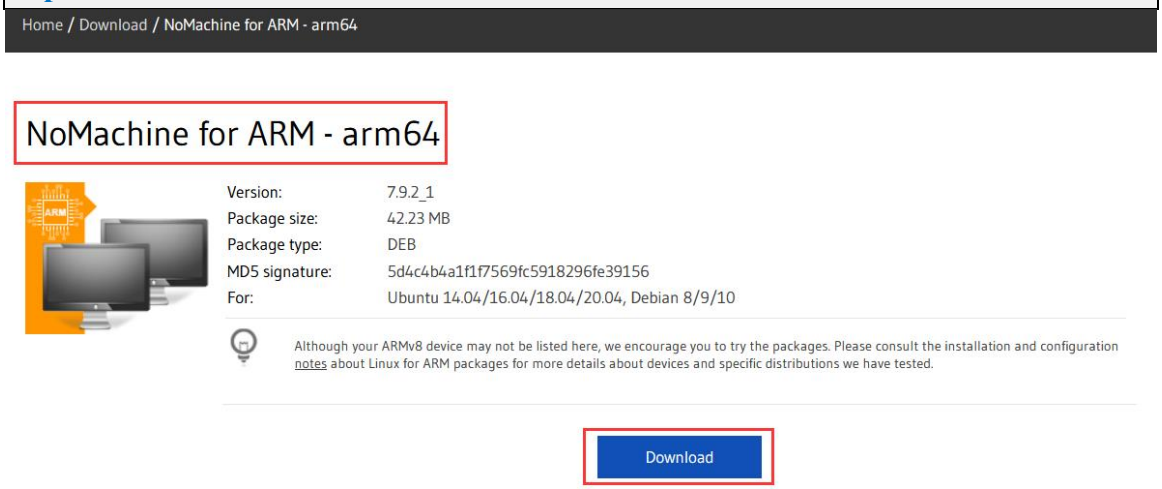
操作前请先确保 Windwos 电脑和开发板在同一局域网内，并且能正常 ssh 登录开发板的 Ubuntu 或者 Debian 系统。

1) 首先下载 NoMachine 软件 Linux **arm64** deb 版本的安装包，然后安装到开发板的 Linux 系统中

a. 由于 H616 是 ARMv8 架构的 SOC，我们使用的系统为 Ubuntu 或者 Debian，所以这里需要下载 **NoMachine for ARM ARMv8 DEB** 安装包，下载链接如下所示：

注意，这个下载链接可能会变，请认准 Armv8/Arm64 版本的 deb 包。

https://www.nomachine.com/download/download&id=116&s=ARM



b. 另外在**官方工具**中也可以下载到 **NoMachine** 的安装包

c. 然后将下载的 **nomachine_7.9.2_1_arm64.deb** 上传到开发板的 Linux 系统中

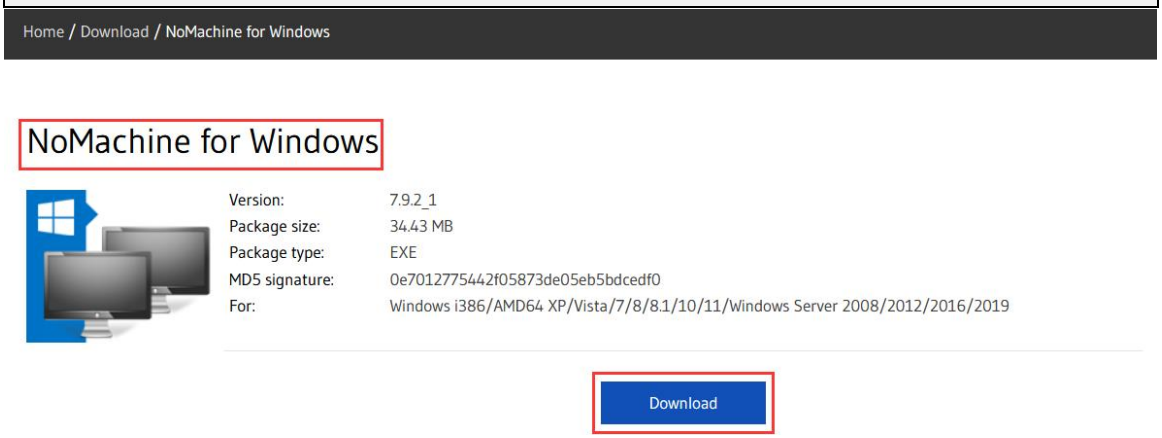
d. 然后使用下面的命令在开发板的 Linux 系统中安装 **NoMachine**

```
orangepi@orangepi:~$ sudo dpkg -i nomachine_7.9.2_1_arm64.deb
[sudo] password for orangepi:
Selecting previously unselected package nomachine.
(Reading database ... 182635 files and directories currently installed.)
Preparing to unpack nomachine_7.9.2_1_arm64.deb ...
Unpacking nomachine (7.9.2-1) ...
Setting up nomachine (7.9.2-1) ...
NX> 700 Starting install at: Sun Apr 17 10:52:07 2022.
NX> 700 Installing: nxclient version: 7.9.2.
NX> 700 Using installation profile: Debian.
NX> 700 Install log is: /usr/NX/var/log/nxinstall.log.
NX> 700 Compiling the USB module.
NX> 700 Installing: nxplayer version: 7.9.2.
NX> 700 Using installation profile: Debian.
NX> 700 Install log is: /usr/NX/var/log/nxinstall.log.
NX> 700 To connect the remote printer to the local desktop,
```

```
NX> 700 the user account must be a member of the CUPS System Group: lpadmin.
NX> 700 Installing: nxnode version: 7.9.2.
NX> 700 Using installation profile: Debian.
NX> 700 Install log is: /usr/NX/var/log/nxinstall.log.
NX> 700 Creating configuration in: /usr/NX/etc/node.cfg.
NX> 700 Installing: nxserver version: 7.9.2.
NX> 700 Using installation profile: Debian.
NX> 700 Install log is: /usr/NX/var/log/nxinstall.log.
NX> 700 Creating configuration in: /usr/NX/etc/server.cfg.
NX> 700 Install completed at: Sun Apr 17 10:53:00 2022.
NX> 700 NoMachine was configured to run the following services:
NX> 700 NX service on port: 4000
```

2) 然后下载 NoMachine 软件 Windows 版本的安装包，下载地址如下所示

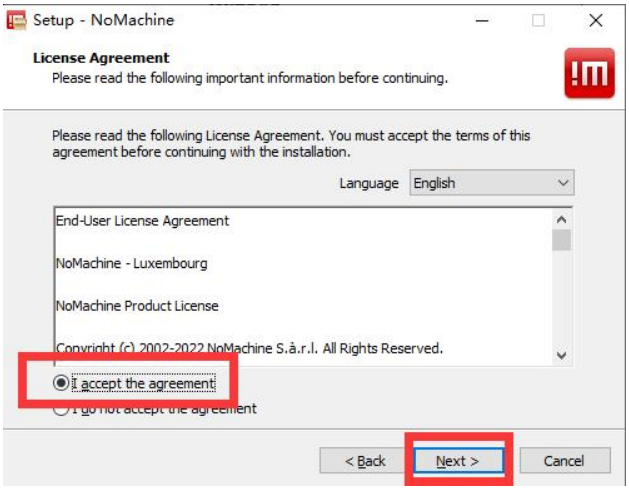
https://www.nomachine.com/download/download&id=8



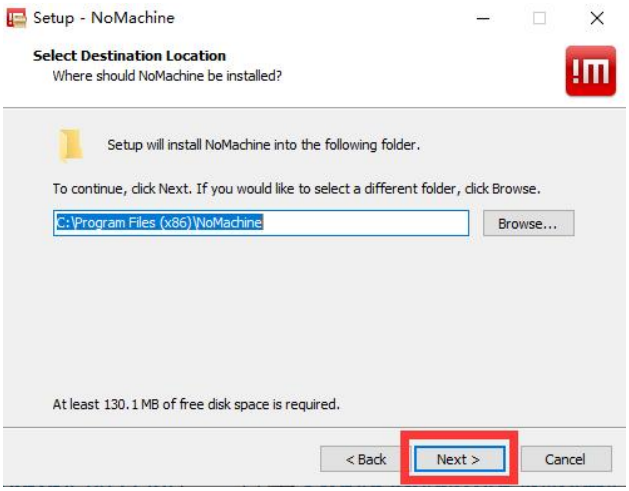
3) 然后在 Windows 中安装 NoMachine

a. 在 Windows 中双击 NoMachine 的安装包就可以开始 NoMachine 的安装，然后选择 **Next** 即可

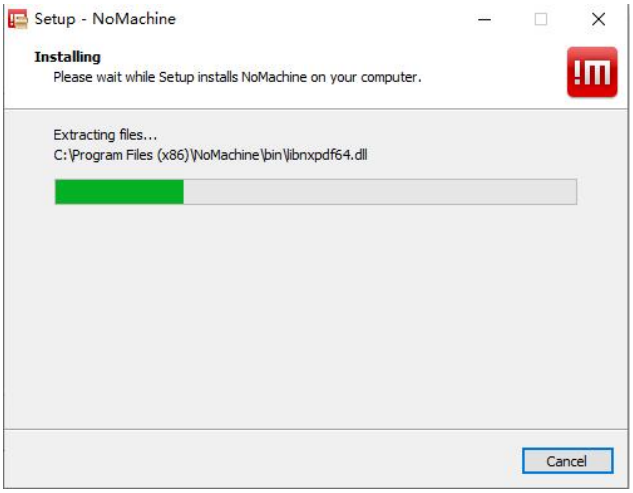
b. 然后选择 **I accept the agreement**，再选择 **Next**



c. 然后点击 **Next**

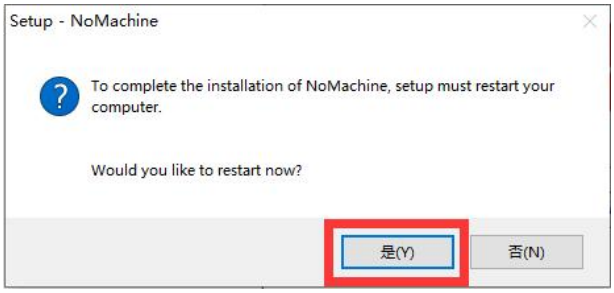


d. 然后就会开始安装过程



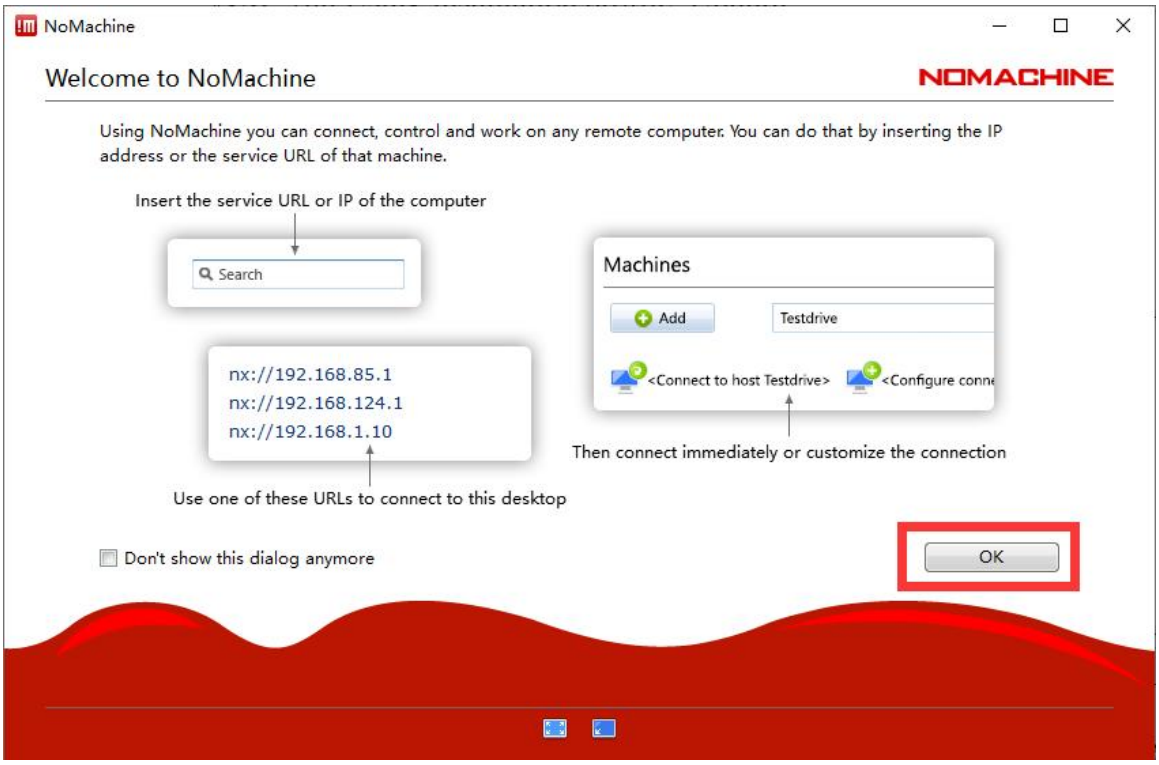
e. 安装完成后的显示如下图所示，然后点击 **Finish** 即可

f. 然后 NoMachine 会提示需要重启才能完成安装，这里我们选择**是(Y)**重启电脑即可

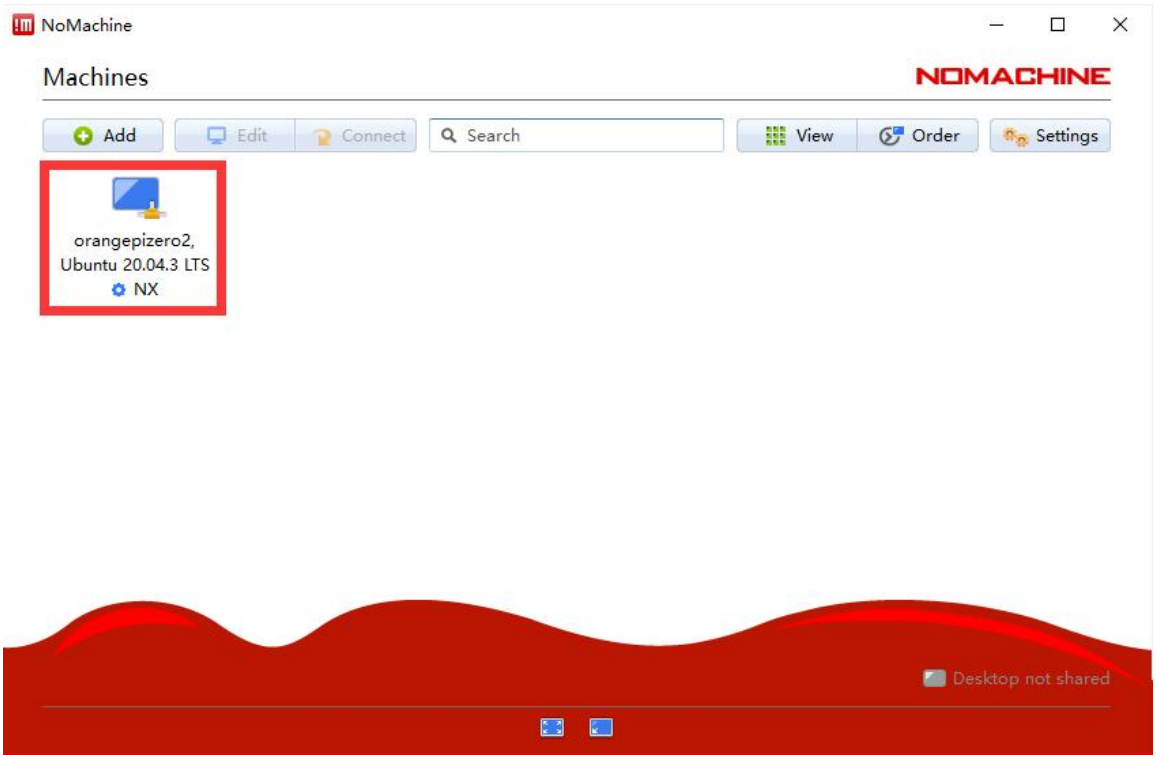


4) 然后在 Window 双击中打开 **NoMachine**

5) 然后点击 **OK**



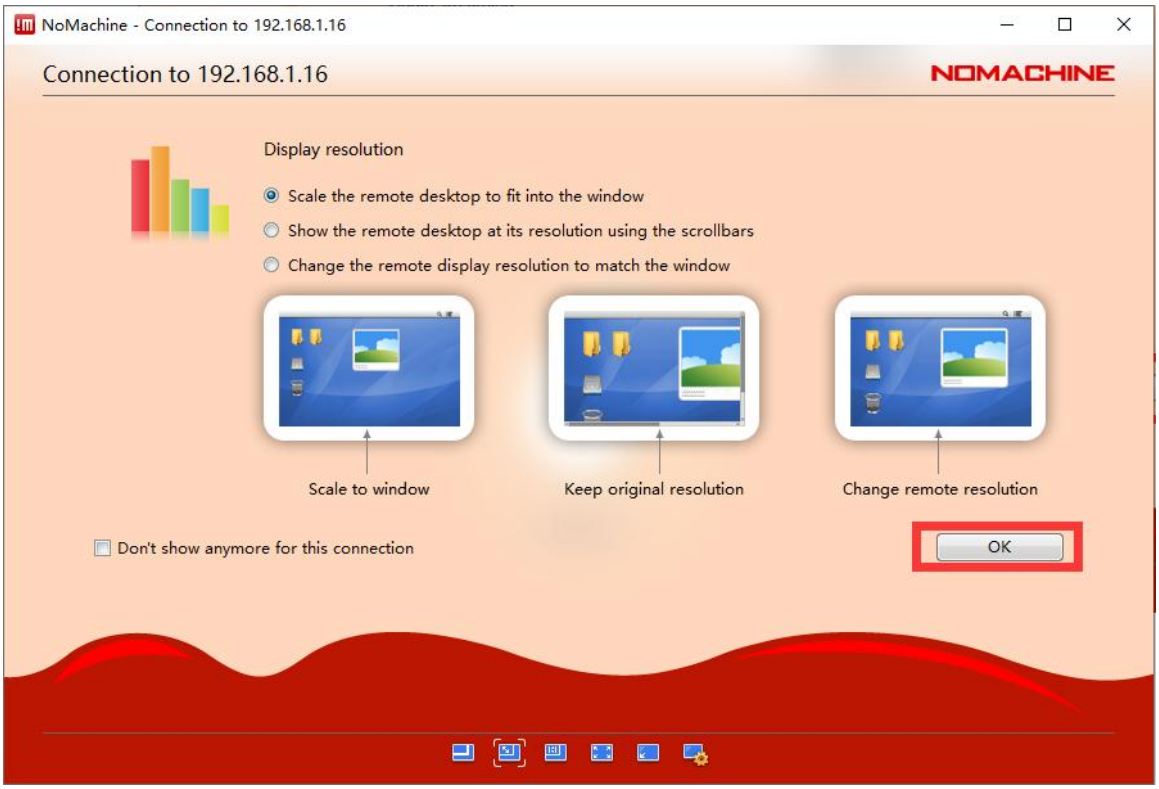
6) NoMachine 启动后会自动扫描局域网内其他安装有 NoMachine 的设备，进入 NoMachine 的主界面后就可以看到开发板已经在可连接的设备列表里了，然后点击下图红色方框所示的位置即可开始登录开发板的 Linux 系统桌面

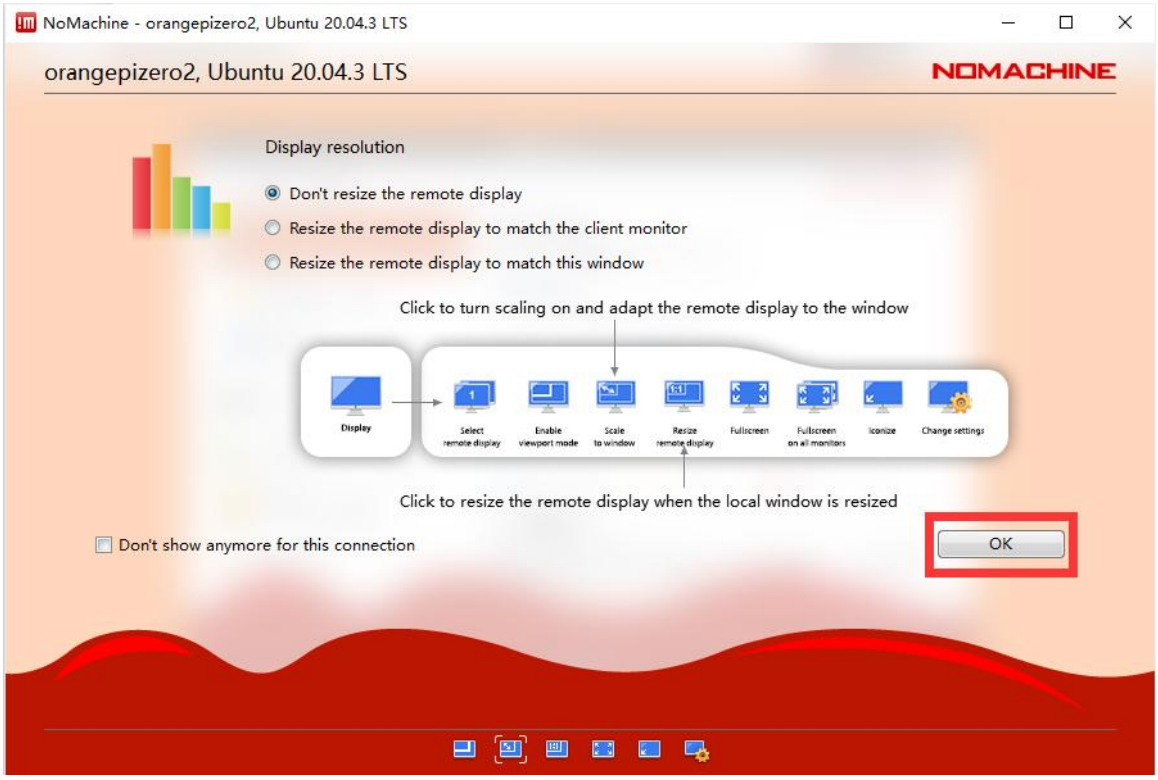


7) 然后在下图对应的位置输入开发板 Linux 系统的用户名和密码，再点击 **Login** 开始登陆

8) 然后点击 OK

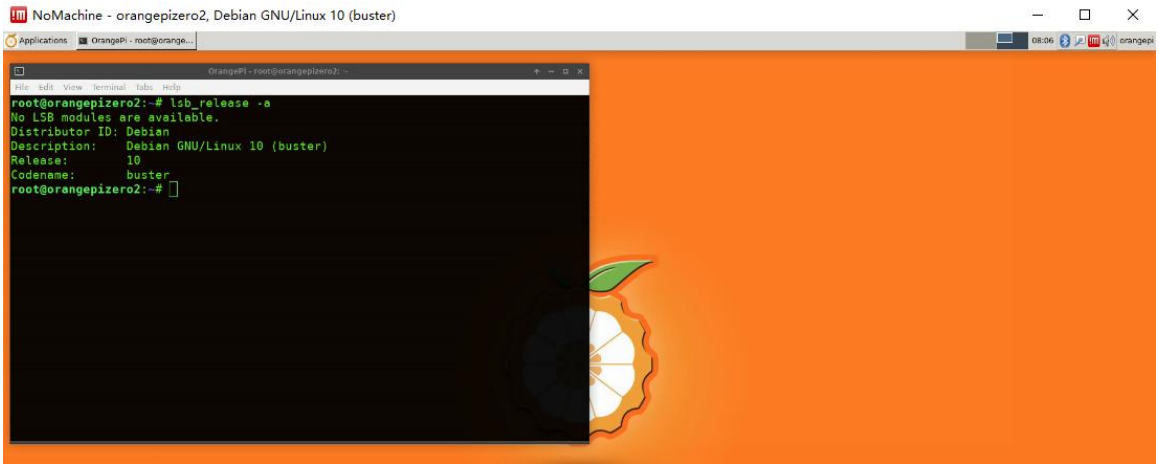
9) 然后可以对显示的分辨率进行设置，根据需要选择即可，然后点击 OK



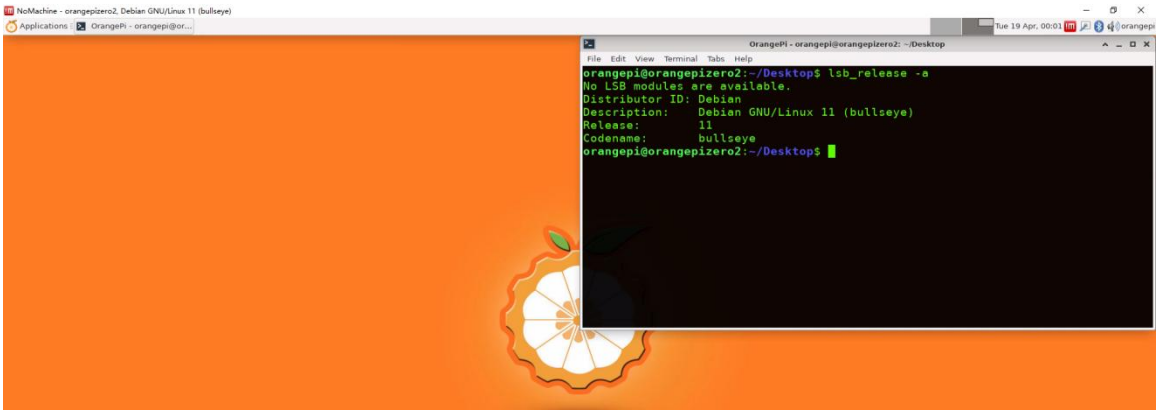


10) 最后就能看到开发板 Linux 系统的桌面了

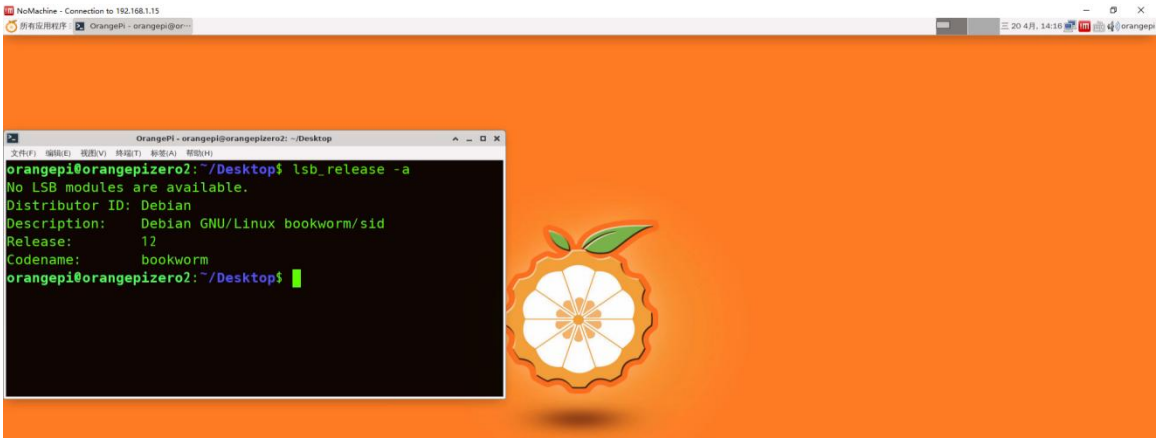
a. Debian10



b. Debian11



c. Debian12



3.37.2. 使用 VNC 远程登录

操作前请先确保 Windwos 电脑和开发板在同一局域网内，并且能正常 ssh 登录开发板的 Ubuntu 或者 Debian 系统

1) 首先在开发板的 Linux 系统中执行下面命令安装 **tightvncserver 和 xrdp**

```
orangepi@orangepi:~$ sudo apt update
orangepi@orangepi:~$ sudo apt -y install tightvncserver xrdp
```

Debian12 没有 xrdp 这个软件包，所以只用安装 tightvncserver

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

2) 然后在开发板的 Linux 系统中运行 **vncserver** 命令来设置密码（**密码为 6 到 8 位**

的字符）并创建初始的配置文件

```
orangepi@orangepi:~$ vncserver

You will require a password to access your desktops.

Password:
Verify:
Would you like to enter a view-only password (y/n)? n

New 'X' desktop is orangepi:1

Creating default startup script /home/orangepi/.vnc/xstartup
Starting applications specified in /home/orangepi/.vnc/xstartup
Log file is /home/orangepi/.vnc/orangepi:1.log
```

3) 然后设置 vnc 的 xstartup 配置文件

a. 首先停用 vnc 服务器实例

```
orangepi@orangepi:~$ vncserver -kill :1
Killing Xtightvnc process ID 5337
```

b. 然后备份下 xstartup 配置文件

```
orangepi@orangepi:~$ mv ~/.vnc/xstartup ~/.vnc/xstartup.bak
```

c. 然后新建一个 xstartup 配置文件，并在其中输入下面的内容

```
orangepi@orangepi:~$ vim ~/.vnc/xstartup
#!/bin/bash
xrdb $HOME/.Xresources
startxfce4 &
```

d. 然后给 xstartup 添加可执行权限

```
orangepi@orangepi:~$ chmod +x ~/.vnc/xstartup
```

e. 最后重启 vncserver 服务器，此时就可以通过 vnc 客户端来远程登录 Linux 桌面了

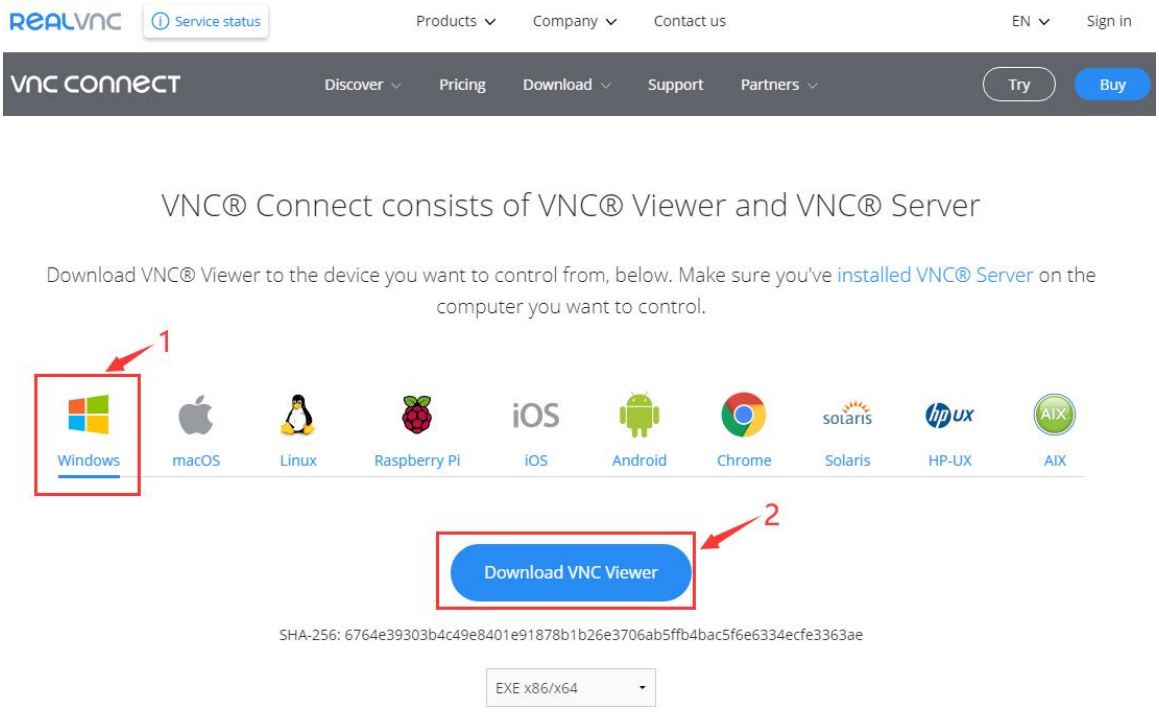
```
orangepi@orangepi:~$ vncserver

New 'X' desktop is orangepizero2:1

Starting applications specified in /home/orangepi/.vnc/xstartup
```

```
Log file is /home/orangepi/.vnc/orangepizero2:1.log
```

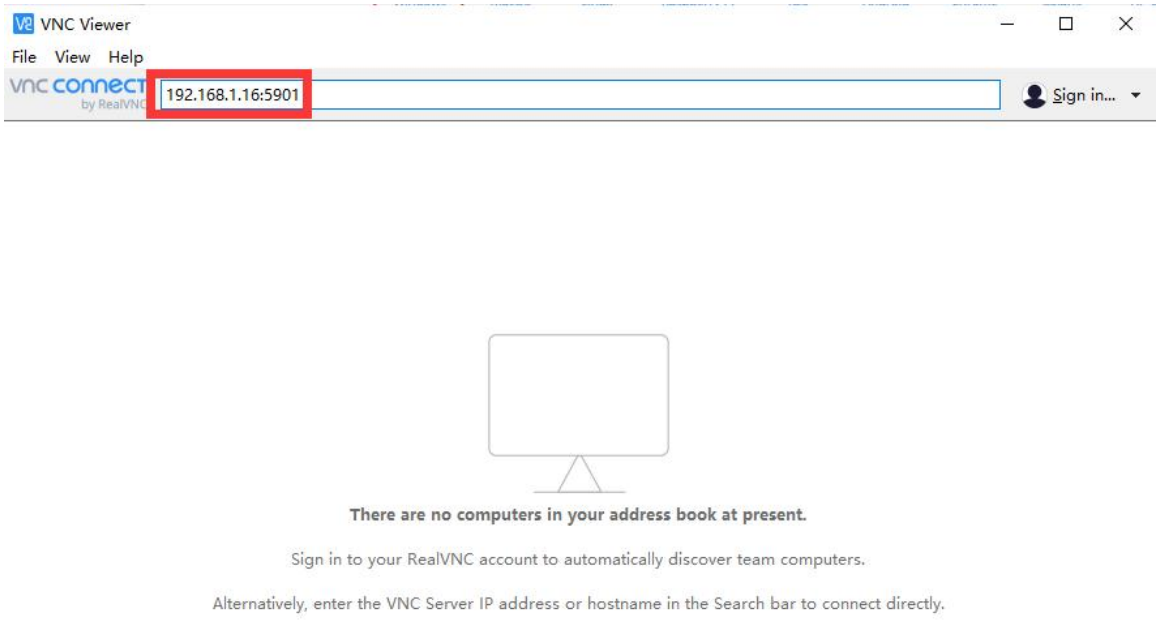
4) 使用 VNC Viewer 客户端连接开发板 Linux 系统桌面的步骤如下所示

a. 首先在 Windows PC 上下载安装 VNC Viewer 客户端，下载链接如下

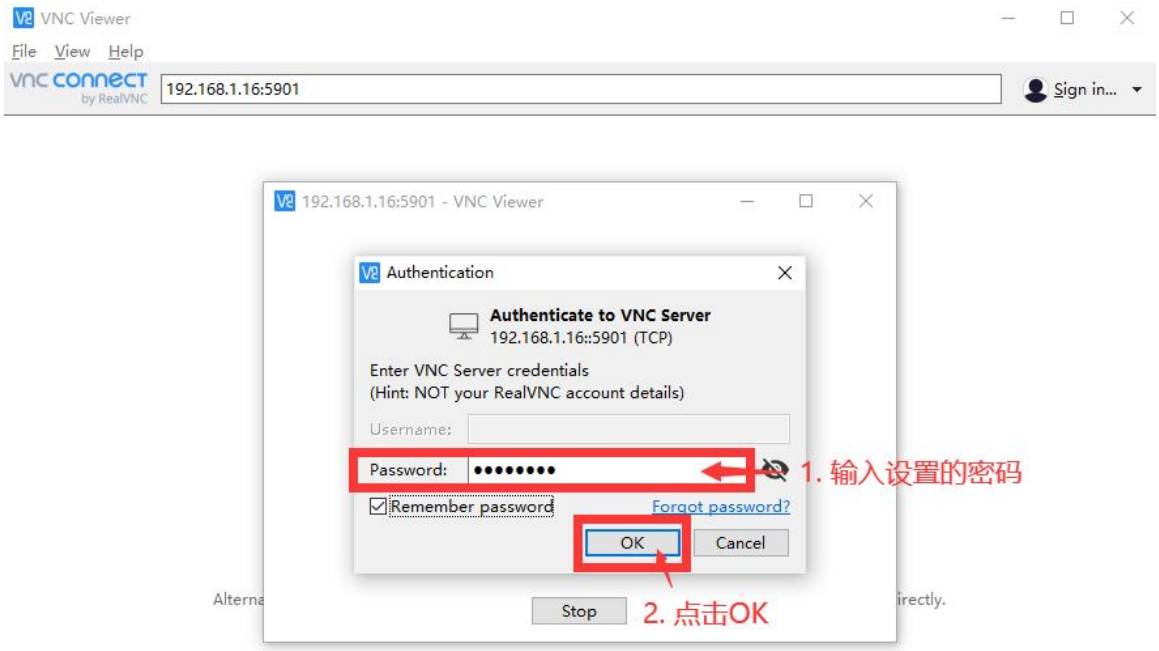
https://www.realvnc.com/en/connect/download/viewer/



b. 在 Windows PC 上安装好 VNC Viewer 客户端后，再打开 **VNC Viewer**，然后在 **VNC Viewer** 的搜索栏中输入**【开发板的 IP 地址:5901】**

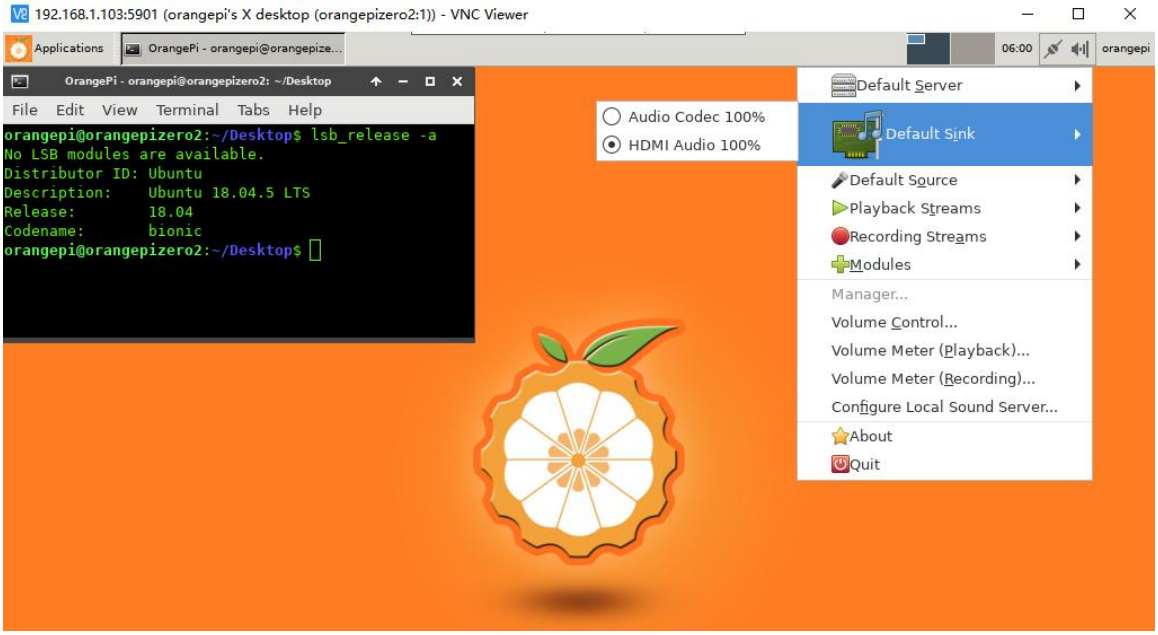


c. 然后在 **Passowrd** 一栏输入第 2 步运行 vncserver 命令时设置的密码，再点击 **OK** 即可远程登录到开发板的 Linux 系统桌面

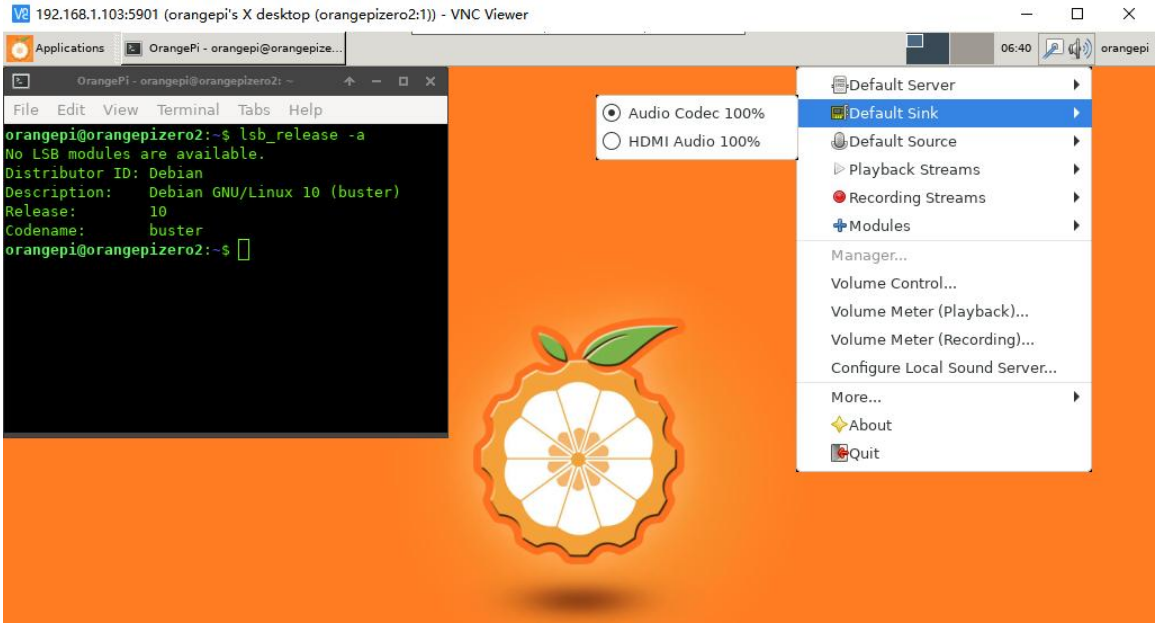


d. 登录成功后的界面显示如下图所示，然后就可以远程操作开发板 Linux 系统的桌面了

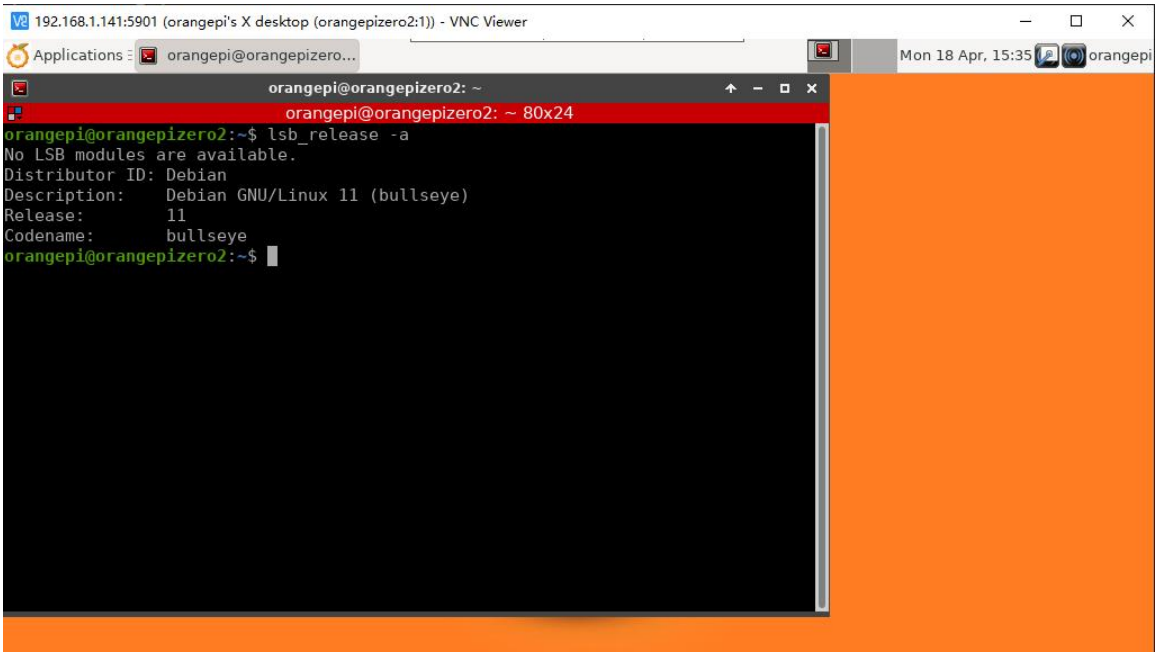
a) Ubuntu18.04 登录显示如下所示



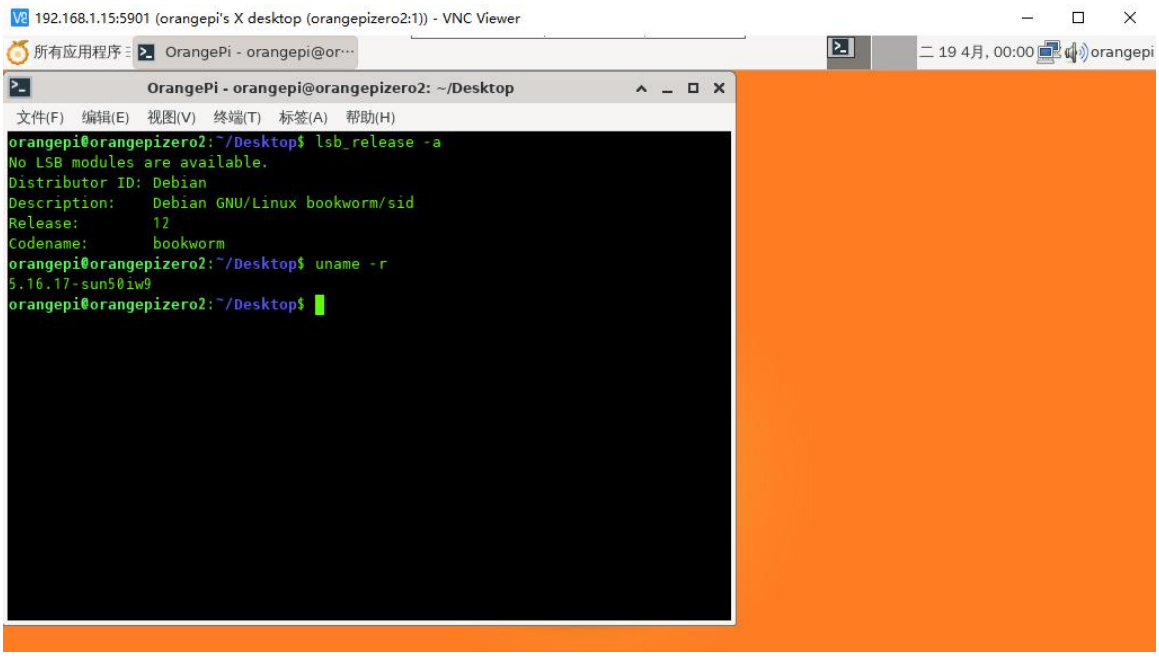
b) Debian10 登录显示如下所示



c) Debian11 登录显示如下所示



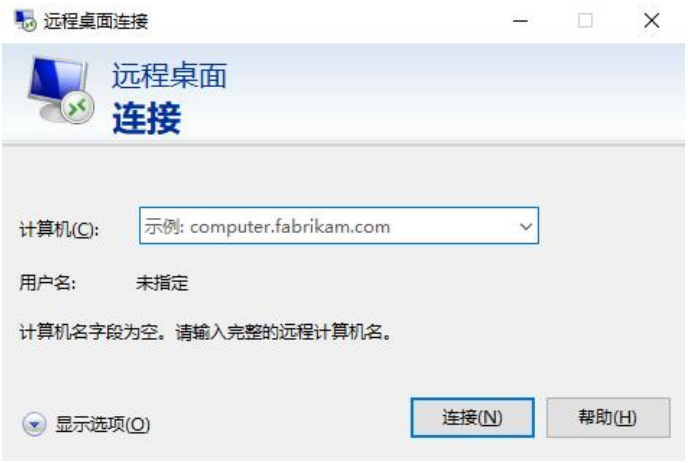
d) Debian12 登录显示如下所示



e) Ubuntu20.04 目前测试有些问题，请先使用 NoMachine

5) 使用 Windows 自带的**远程桌面连接**应用登录开发板 Linux 系统桌面的步骤为

a. 首先打开 Windows 自带的**远程桌面连接**



b. 然后输入开发板的 IP 地址

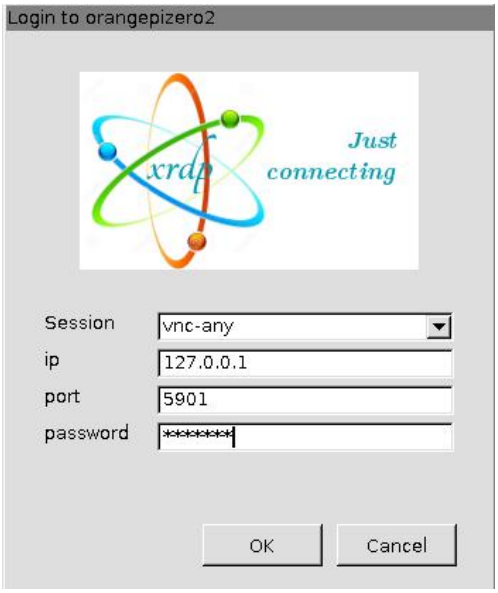
c. 然后根据下图说明设置连接信息

a) **Session：**需要选择 vnc-any

b) **ip：**可以输入 127.0.0.1 或者开发板的 IP 地址

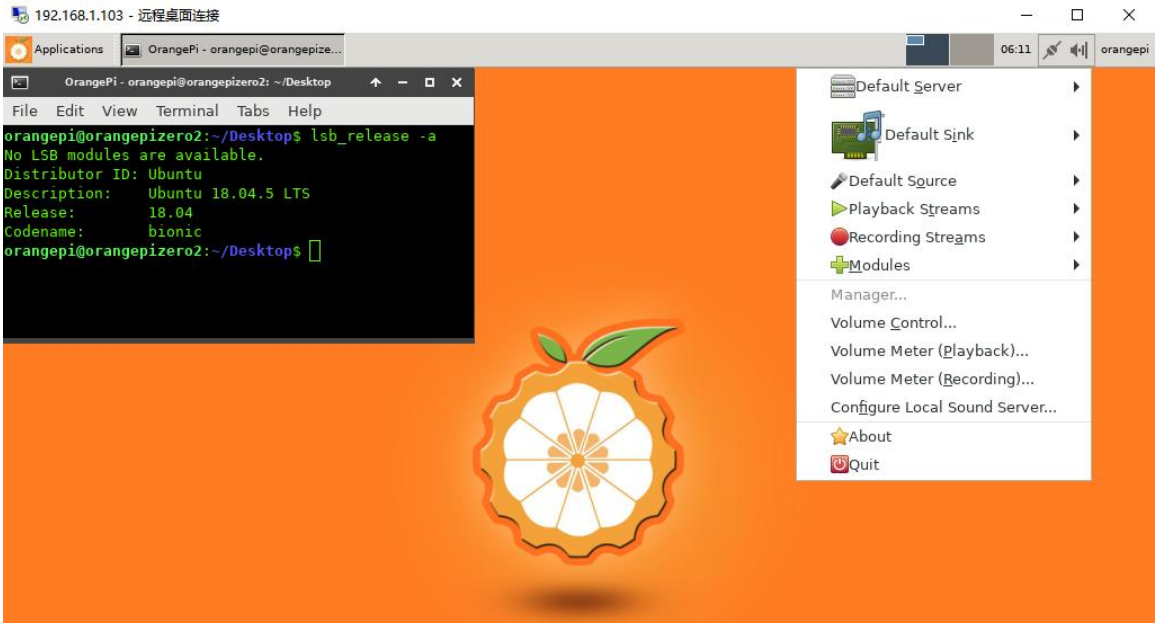
c) **port：**一般为 5901

d) **password：**需要输入第 2 步运行 vncserver 命令时设置的密码

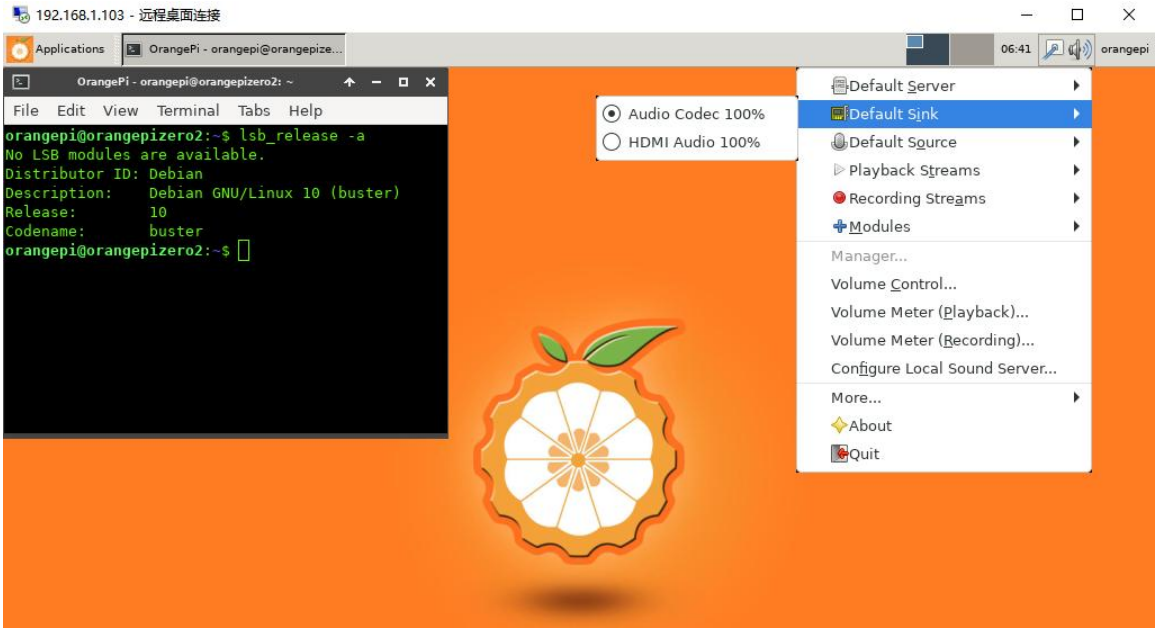


d. 成功登录开发板 Linux 系统桌面的显示如下图所示

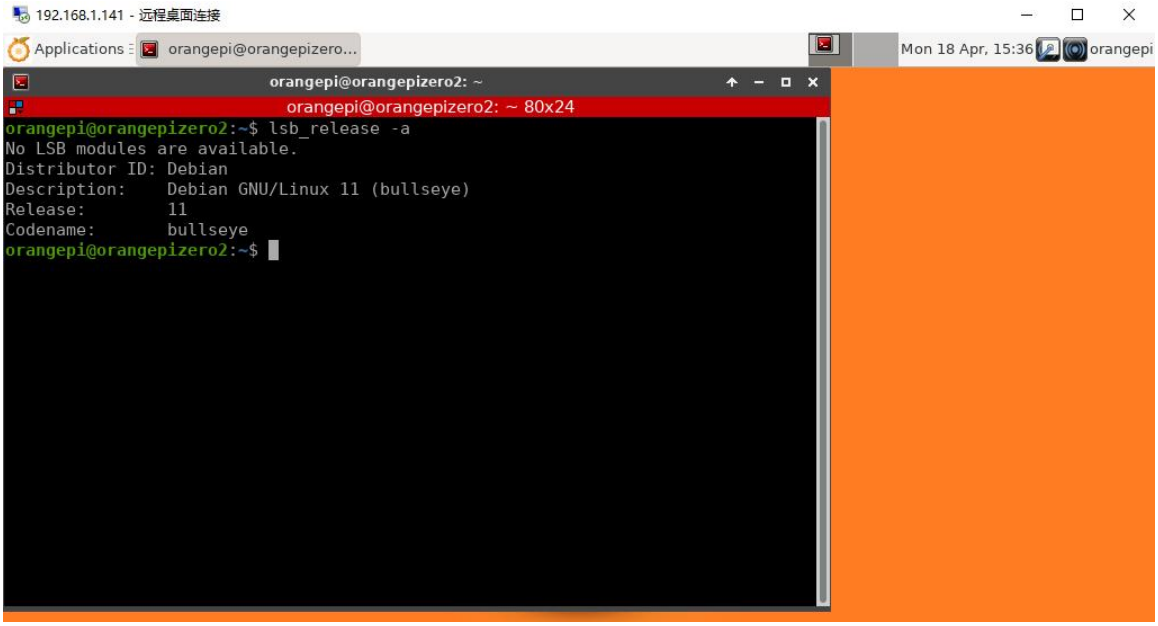
a) Ubuntu18.04 登录显示如下所示



b) Debian10 登录显示如下所示



c) Debian11 登录显示如下所示



3.38. 腾讯 ncnn 高性能神经网络前向计算框架测试

如果不知道 ncnn 是什么，可以看下 ncnn 在 github 上 README 中的介绍。

ncnn github 仓库地址为：https://github.com/Tencent/ncnn

ncnn

ncnn is a high-performance neural network inference computing framework optimized for mobile platforms. ncnn is deeply considerate about deployment and uses on mobile phones from the beginning of design. ncnn does not have third party dependencies. It is cross-platform, and runs faster than all known open source frameworks on mobile phone cpu. Developers can easily deploy deep learning algorithm models to the mobile platform by using efficient ncnn implementation, create intelligent APPs, and bring the artificial intelligence to your fingertips. ncnn is currently being used in many Tencent applications, such as QQ, Qzone, WeChat, Pitu and so on.

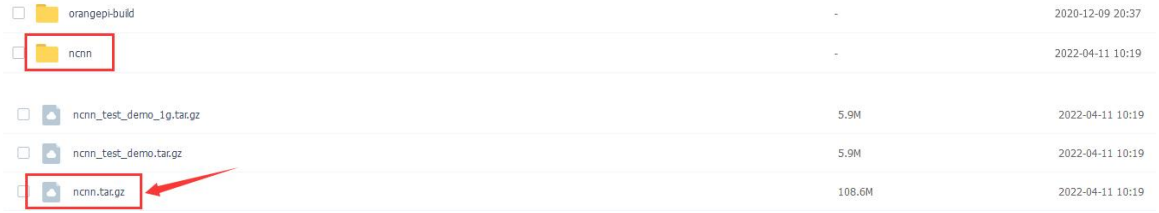
ncnn 是一个为手机端极致优化的高性能神经网络前向计算框架。ncnn 从设计之初深刻考虑手机端的部署和使用。无第三方依赖，跨平台，手机端 cpu 的速度快于目前所有已知的开源框架。基于 ncnn，开发者能够将深度学习算法轻松移植到手机端高效执行，开发出人工智能 APP，将 AI 带到你的指尖。ncnn 目前已在腾讯多款应用中使用，如 QQ，Qzone，微信，天天P图等。

1) 腾讯 ncnn 源码下载命令如下所示

a. 第一种方法：下载 Orang Pi 百度云盘中提供的 **ncnn.tar.gz** 压缩包

a) 从下面的百度云盘链接可以下载 **ncnn.tar.gz** 源码压缩包。进入 ncnn 文件夹中就能看到

链接：https://pan.baidu.com/s/1vWQmCmSYdH7iCDFyKpJtVw
提取码：zero



b) 下载完 **ncnn.tar.gz** 压缩包后，首先将 **ncnn.tar.gz** 上传到开发板 linux 系统中

c) 然后使用下面的命令解压 **ncnn.tar.gz**

```
orangepi@orangepi:~$ tar zxf ncnn.tar.gz
orangepi@orangepi:~$ ls
ncnn    ncnn.tar.gz
```

b. 第二种方法：使用 git 命令直接下载源码，但是如果不会解决开发板访问 github 的问题，是很难下载成功的。如果访问 github 没有问题，建议使用这种方法，因为这种方式可以保证代码是最新的。

```
orangepi@orangepi:~$ git clone https://github.com/Tencent/ncnn.git
orangepi@orangepi:~$ cd ncnn
```

```
orangepi@orangepi:~/ncnn$ git submodule update --init
```

2) 然后安装依赖包

```
orangepi@orangepi:~$ sudo apt update
orangepi@orangepi:~$ sudo apt install -y build-essential git cmake   \
libprotobuf-dev protobuf-compiler libopencv-dev
```

3) 然后在 ncnn 的源码中打开 **benchmark/benchncnn.cpp** 这个文件，在其中去掉 **vgg16** 这个测试项。由于开发板的内存只有 1GB，当测试到 vgg16 时，会由于内存不足导致 benchncnn 测试中断退出

```
orangepi@orangepi:~$ cd ncnn
orangepi@orangepi:~/ncnn$ vim benchmark/benchncnn.cpp

打开 benchmark/benchncnn.cpp 后，找到下面这行，然后注释掉

//benchmark("vgg16", ncnn::Mat(224, 224, 3), opt);
```

如果对 benchmark 测试不感兴趣，不用修改这里也是可以的。

这个问题只在 1G 内存的开发板中才有，在配有 2G 内存的开发板，如果 Orange Pi 3 LTS 中是正常的。

4) 然后开始编译，ncnn 编译命令如下所示

```
orangepi@orangepi:~/ncnn$ mkdir build
orangepi@orangepi:~/ncnn$ cd build
orangepi@orangepi:~/ncnn/build$ cmake   \
-DCMAKE_TOOLCHAIN_FILE=../toolchains/aarch64-linux-gnu.toolchain.cmake \
-DNCNN_SIMPLEOCV=ON -DNCNN_BUILD_EXAMPLES=ON ..
orangepi@orangepi:~/ncnn/build$ make -j$(nproc)
[  0%] Built target ncnn-generate-spirv
[  1%] Generating source absval_arm_arm82.h
[  1%] Generating source convolution1d_arm_arm82.cpp
......
[ 99%] Linking CXX executable squeezenetssd
[ 99%] Built target squeezenetssd
```

```
[100%] Linking CXX executable shufflenetv2
[100%] Built target shufflenetv2
```

在没有任何散热措施的情况下，在开发板上直接编译 ncnn 大约需要 20~25 分钟，请耐心等待编译完成。

5) ncnn 中有一些测试示例，比如 **squeezenet** 测试命令和结果如下所示

```
orangepi@orangepi:~/ncnn/build$ cd ../examples
orangepi@orangepi:~/ncnn/examples$ ../build/examples/squeezenet  \
../images/256-ncnn.png
532 = 0.165950
920 = 0.094098
716 = 0.062193
```

6) **benchncnn** 可以用来测试神经网络的推理性能，测试方法如下所示

a. 编译生成的 **benchncnn** 可执行文件在如下路径中，**注意下面的命令执行路径为 ncnn 源码的顶层目录**

```
orangepi@orangepi:~/ncnn$ ls build/benchmark/
benchncnn  CMakeFiles  cmake_install.cmake  Makefile
```

b. 首先需要将 **benchncnn** 复制到 **benchmark** 目录中

```
orangepi@orangepi:~/ncnn$ cp build/benchmark/benchncnn benchmark
```

c. **benchncnn** 的使用方法如下所示

```
./benchncnn [loop count] [num threads] [powersave] [gpu device] [cooling down]
```

Parameter

param	options	default
loop count	1~N	4
num threads	1~N	max_cpu_count
powersave	0=all cores, 1=little cores only, 2=big cores only	0
gpu device	-1=cpu-only, 0=gpu0, 1=gpu1 ...	-1
cooling down	0=disable, 1=enable	1

https://github.com/Tencent/ncnn/blob/9d0c36358cec2d1da471574064d4abd8787b45a8/benchmark/README.md

d. **benchncnn** 使用 cpu 测试命令和结果如下所示

```
orangepi@orangepi:~/ncnn$ cd benchmark
orangepi@orangepi:~/ncnn/benchmark$ ./benchncnn 4 $(nproc) 0 -1
```

a) Debian Bookworm Linux5.16 服务器版本系统测试结果

```
orangepi@orangepizero2:~/ncnn/benchmark$ ./benchncnn 4 $(nproc) 0 -1
loop_count = 4
num_threads = 4
powersave = 0
gpu_device = -1
cooling_down = 1
          squeezenet  min =   85.25  max =   88.18  avg =   86.30
     squeezenet_int8  min =   68.47  max =   68.75  avg =   68.62
          mobilenet  min =  109.89  max =  112.64  avg =  111.55
     mobilenet_int8  min =   60.99  max =   61.78  avg =   61.44
       mobilenet_v2  min =  105.97  max =  107.50  avg =  106.86
       mobilenet_v3  min =  101.99  max =  102.32  avg =  102.16
         shufflenet  min =   59.98  max =   83.20  avg =   66.64
      shufflenet_v2  min =   55.81  max =   57.12  avg =   56.29
             mnasnet  min =   93.07  max =   95.41  avg =   94.18
     proxylessnasnet  min =  100.65  max =  103.87  avg =  101.92
     efficientnet_b0  min =  156.15  max =  157.41  avg =  156.82
   efficientnetv2_b0  min =  162.43  max =  178.05  avg =  173.62
        regnety_400m  min =  111.62  max =  113.63  avg =  112.50
           blazeface  min =   19.83  max =   20.21  avg =   20.03
           googlenet  min =  252.25  max =  277.35  avg =  262.28
      googlenet_int8  min =  200.10  max =  200.59  avg =  200.32
           resnet18  min =  280.52  max =  282.21  avg =  281.05
      resnet18_int8  min =  164.09  max =  164.72  avg =  164.34
            alexnet  min =  231.92  max =  237.94  avg =  234.99
          vgg16_int8  min =  705.68  max =  714.29  avg =  708.92
           resnet50  min =  592.31  max =  595.78  avg =  593.96
      resnet50_int8  min =  411.82  max =  487.82  avg =  431.32
      squeezenet_ssd  min =  295.52  max =  298.63  avg =  297.11
 squeezenet_ssd_int8  min =  191.61  max =  263.06  avg =  214.73
       mobilenet_ssd  min =  246.86  max =  251.50  avg =  249.71
  mobilenet_ssd_int8  min =  145.98  max =  146.35  avg =  146.19
      mobilenet_yolo  min =  500.29  max =  665.17  avg =  559.03
  mobilenetv2_yolov3  min =  347.81  max =  356.19  avg =  350.40
        yolov4-tiny  min =  452.98  max =  458.24  avg =  455.48
          nanodet_m  min =  138.41  max =  146.72  avg =  144.06
    yolo-fastest-1.1  min =   77.42  max =   78.10  avg =   77.68
      yolo-fastestv2  min =   60.28  max =   60.83  avg =   60.57
orangepi@orangepizero2:~/ncnn/benchmark$ lsb_release -a
No LSB modules are available.
Distributor ID: Debian
Description:    Debian GNU/Linux bookworm/sid
Release:        12
Codename:       bookworm
orangepi@orangepizero2:~/ncnn/benchmark$ uname -r
5.16.17-sun50iw9
orangepi@orangepizero2:~/ncnn/benchmark$
```

b) Debian Bullseye Linux5.16 服务器版本系统测试结果

```
orangepi@orangepizero2:~/ncnn_test_demo_1g/benchncnn_demo$ ./benchncnn 4 $(nproc) 0 -1
loop_count = 4
num_threads = 4
powersave = 0
gpu_device = -1
cooling_down = 1
          squeezenet  min =   84.73  max =   85.78  avg =   85.05
     squeezenet_int8  min =   68.34  max =   68.71  avg =   68.54
           mobilenet  min =  111.59  max =  117.68  avg =  113.43
      mobilenet_int8  min =   62.66  max =   63.37  avg =   63.07
        mobilenet_v2  min =  105.77  max =  107.41  avg =  106.59
        mobilenet_v3  min =   87.12  max =   88.33  avg =   87.62
          shufflenet  min =   58.86  max =   59.51  avg =   59.20
       shufflenet_v2  min =   55.06  max =   55.81  avg =   55.43
             mnasnet  min =   92.46  max =   94.67  avg =   94.06
     proxylessnasnet  min =   98.56  max =  103.01  avg =  100.81
     efficientnet_b0  min =  156.78  max =  157.07  avg =  156.97
   efficientnetv2_b0  min =  177.06  max =  178.41  avg =  177.72
        regnety_400m  min =  119.82  max =  125.51  avg =  123.79
           blazeface  min =   19.50  max =   23.61  avg =   21.59
           googlenet  min =  257.38  max =  260.14  avg =  258.51
      googlenet_int8  min =  196.01  max =  248.73  avg =  209.44
            resnet18  min =  274.66  max =  286.09  avg =  278.35
       resnet18_int8  min =  167.24  max =  167.51  avg =  167.37
             alexnet  min =  245.09  max =  247.36  avg =  246.20
          vgg16_int8  min =  707.20  max =  709.31  avg =  708.32
            resnet50  min =  590.21  max =  615.26  avg =  602.39
       resnet50_int8  min =  407.66  max =  428.18  avg =  413.04
      squeezenet_ssd  min =  300.67  max =  327.00  avg =  309.63
 squeezenet_ssd_int8  min =  198.15  max =  221.46  avg =  204.54
       mobilenet_ssd  min =  250.40  max =  275.33  avg =  261.94
  mobilenet_ssd_int8  min =  166.05  max =  166.33  avg =  166.16
      mobilenet_yolo  min =  496.32  max =  524.59  avg =  506.88
  mobilenetv2_yolov3  min =  348.41  max =  376.18  avg =  356.31
         yolov4-tiny  min =  457.43  max =  491.69  avg =  466.93
           nanodet_m  min =  139.55  max =  168.32  avg =  152.27
    yolo-fastest-1.1  min =   80.87  max =   81.34  avg =   81.14
      yolo-fastestv2  min =   59.47  max =   62.13  avg =   60.89
orangepi@orangepizero2:~/ncnn_test_demo_1g/benchncnn_demo$ lsb_release -a
No LSB modules are available.
Distributor ID: Debian
Description:    Debian GNU/Linux 11 (bullseye)
Release:        11
Codename:       bullseye
orangepi@orangepizero2:~/ncnn_test_demo_1g/benchncnn_demo$ uname -r
5.16.17-sun50iw9
orangepi@orangepizero2:~/ncnn_test_demo_1g/benchncnn_demo$
```

c) Debian Buster Linux4.9 服务器版本系统测试结果

```
orangepi@orangepizero2:~/ncnn_test_demo_1g/benchncnn_demo$ ./benchncnn 4 $(nproc) 0 -1
loop_count = 4
num_threads = 4
powersave = 0
gpu_device = -1
cooling_down = 1
          squeezenet  min =  101.21  max =  102.54  avg =  101.86
     squeezenet_int8  min =   73.11  max =   73.34  avg =   73.23
           mobilenet  min =  129.88  max =  141.39  avg =  133.65
      mobilenet_int8  min =   65.91  max =   74.71  avg =   68.30
        mobilenet_v2  min =  119.89  max =  120.54  avg =  120.27
        mobilenet_v3  min =   98.65  max =   99.71  avg =   99.10
          shufflenet  min =   65.46  max =   66.61  avg =   66.07
       shufflenet_v2  min =   60.29  max =   62.68  avg =   61.16
             mnasnet  min =  107.85  max =  109.11  avg =  108.53
     proxylessnasnet  min =  110.94  max =  112.82  avg =  111.96
     efficientnet_b0  min =  159.31  max =  160.83  avg =  159.90
   efficientnetv2_b0  min =  185.99  max =  189.42  avg =  188.07
        regnety_400m  min =  124.11  max =  125.88  avg =  125.16
           blazeface  min =   22.18  max =   23.21  avg =   22.86
           googlenet  min =  273.08  max =  282.78  avg =  277.98
      googlenet_int8  min =  203.28  max =  209.00  avg =  205.70
            resnet18  min =  315.77  max =  318.84  avg =  317.48
       resnet18_int8  min =  183.04  max =  187.42  avg =  184.66
             alexnet  min =  270.63  max =  275.00  avg =  272.14
          vgg16_int8  min =  785.36  max =  818.18  avg =  795.80
            resnet50  min =  644.28  max =  659.93  avg =  650.18
       resnet50_int8  min =  412.37  max =  414.93  avg =  413.83
      squeezenet_ssd  min =  329.54  max =  336.00  avg =  331.96
 squeezenet_ssd_int8  min =  211.66  max =  214.66  avg =  212.80
       mobilenet_ssd  min =  264.88  max =  281.44  avg =  270.00
  mobilenet_ssd_int8  min =  141.64  max =  148.24  avg =  143.50
      mobilenet_yolo  min =  557.36  max =  581.72  avg =  571.21
  mobilenetv2_yolov3  min =  353.86  max =  369.12  avg =  361.83
         yolov4-tiny  min =  501.15  max =  515.92  avg =  506.44
           nanodet_m  min =  153.47  max =  156.90  avg =  154.92
    yolo-fastest-1.1  min =   82.01  max =   83.51  avg =   82.62
      yolo-fastestv2  min =   65.71  max =   66.68  avg =   66.40
orangepi@orangepizero2:~/ncnn_test_demo_1g/benchncnn_demo$ lsb_release -a
No LSB modules are available.
Distributor ID: Debian
Description:    Debian GNU/Linux 10 (buster)
Release:        10
Codename:       buster
orangepi@orangepizero2:~/ncnn_test_demo_1g/benchncnn_demo$ uname -r
4.9.170-sun50iw9
orangepi@orangepizero2:~/ncnn_test_demo_1g/benchncnn_demo$
```

d) Ubuntu Focal Linux4.9 服务器版本系统测试结果

```
orangepi@orangepizero2:~/ncnn_test_demo_1g/benchncnn_demo$ ./benchncnn 4 $(nproc) 0 -1
loop_count = 4
num_threads = 4
powersave = 0
gpu_device = -1
cooling_down = 1
          squeezenet  min =   97.04  max =   97.87  avg =   97.42
     squeezenet_int8  min =   74.87  max =   76.53  avg =   75.36
          mobilenet  min =  129.67  max =  134.08  avg =  131.46
     mobilenet_int8  min =   67.85  max =   69.84  avg =   68.45
       mobilenet_v2  min =  118.65  max =  119.99  avg =  119.47
       mobilenet_v3  min =   97.37  max =   98.85  avg =   98.07
         shufflenet  min =   65.06  max =   66.28  avg =   65.45
      shufflenet_v2  min =   60.81  max =   61.40  avg =   61.11
           mnasnet  min =  103.68  max =  107.60  avg =  105.52
     proxylessnasnet  min =  110.98  max =  115.12  avg =  113.02
     efficientnet_b0  min =  157.77  max =  159.13  avg =  158.55
   efficientnetv2_b0  min =  183.36  max =  188.11  avg =  184.91
        regnety_400m  min =  124.32  max =  124.85  avg =  124.60
           blazeface  min =   21.12  max =   22.68  avg =   21.95
           googlenet  min =  276.24  max =  281.58  avg =  278.66
      googlenet_int8  min =  199.85  max =  203.42  avg =  201.28
           resnet18  min =  311.15  max =  331.70  avg =  317.94
      resnet18_int8  min =  185.04  max =  188.46  avg =  186.23
           alexnet  min =  268.10  max =  272.91  avg =  269.84
         vgg16_int8  min =  794.69  max =  817.88  avg =  807.30
           resnet50  min =  657.25  max =  659.90  avg =  658.07
      resnet50_int8  min =  411.41  max =  414.83  avg =  413.04
     squeezenet_ssd  min =  329.91  max =  335.23  avg =  331.41
 squeezenet_ssd_int8  min =  211.84  max =  218.02  avg =  214.15
       mobilenet_ssd  min =  266.72  max =  282.67  avg =  271.74
  mobilenet_ssd_int8  min =  140.36  max =  140.97  avg =  140.64
      mobilenet_yolo  min =  529.34  max =  559.91  avg =  541.46
  mobilenetv2_yolov3  min =  371.21  max =  383.87  avg =  377.91
        yolov4-tiny  min =  501.76  max =  516.57  avg =  508.18
          nanodet_m  min =  155.52  max =  166.02  avg =  158.22
    yolo-fastest-1.1  min =   85.44  max =   89.75  avg =   86.63
      yolo-fastestv2  min =   66.37  max =   67.03  avg =   66.68
orangepi@orangepizero2:~/ncnn_test_demo_1g/benchncnn_demo$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 20.04.4 LTS
Release:        20.04
Codename:       focal
orangepi@orangepizero2:~/ncnn_test_demo_1g/benchncnn_demo$ uname -r
4.9.170-sun50iw9
orangepi@orangepizero2:~/ncnn_test_demo_1g/benchncnn_demo$
```

e) Ubuntu Bionic Linux4.9 服务器版本系统测试结果

```
orangepi@orangepizero2:~/ncnn_test_demo_1g/benchncnn_demo$ ./benchncnn 4 $(nproc) 0 -1
loop_count = 4
num_threads = 4
powersave = 0
gpu_device = -1
cooling_down = 1
          squeezenet  min =  100.80  max =  104.88  avg =  101.89
     squeezenet_int8  min =   73.56  max =   74.35  avg =   74.05
           mobilenet  min =  127.57  max =  141.65  avg =  131.51
      mobilenet_int8  min =   66.00  max =   66.70  avg =   66.32
        mobilenet_v2  min =  119.05  max =  125.64  avg =  121.20
        mobilenet_v3  min =   97.42  max =  100.12  avg =   98.53
          shufflenet  min =   65.89  max =   74.87  avg =   68.74
       shufflenet_v2  min =   60.22  max =   61.39  avg =   60.87
             mnasnet  min =  107.49  max =  111.98  avg =  109.83
     proxylessnasnet  min =  109.60  max =  110.65  avg =  110.16
     efficientnet_b0  min =  158.67  max =  167.94  avg =  161.66
   efficientnetv2_b0  min =  188.55  max =  196.77  avg =  190.78
        regnety_400m  min =  124.03  max =  133.10  avg =  126.77
           blazeface  min =   22.02  max =   22.79  avg =   22.46
           googlenet  min =  273.03  max =  284.48  avg =  277.67
      googlenet_int8  min =  199.92  max =  212.49  avg =  203.93
           resnet18  min =  313.82  max =  325.06  avg =  318.89
      resnet18_int8  min =  174.94  max =  180.87  avg =  177.25
            alexnet  min =  272.26  max =  279.92  avg =  274.40
          vgg16_int8  min =  773.62  max =  817.88  avg =  794.97
           resnet50  min =  641.58  max =  661.81  avg =  650.93
      resnet50_int8  min =  408.98  max =  415.24  avg =  411.01
     squeezenet_ssd  min =  329.31  max =  336.26  avg =  333.11
 squeezenet_ssd_int8  min =  210.64  max =  211.61  avg =  211.16
       mobilenet_ssd  min =  266.34  max =  278.02  avg =  271.36
  mobilenet_ssd_int8  min =  140.69  max =  147.47  avg =  142.61
      mobilenet_yolo  min =  566.12  max =  578.98  avg =  571.36
  mobilenetv2_yolov3  min =  350.73  max =  370.09  avg =  362.37
        yolov4-tiny  min =  499.14  max =  520.08  avg =  509.07
          nanodet_m  min =  154.07  max =  156.35  avg =  155.36
    yolo-fastest-1.1  min =   84.16  max =   84.74  avg =   84.45
      yolo-fastestv2  min =   66.01  max =   67.16  avg =   66.54
orangepi@orangepizero2:~/ncnn_test_demo_1g/benchncnn_demo$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 18.04.6 LTS
Release:        18.04
Codename:       bionic
orangepi@orangepizero2:~/ncnn_test_demo_1g/benchncnn_demo$ uname -r
4.9.170-sun50iw9
orangepi@orangepizero2:~/ncnn_test_demo_1g/benchncnn_demo$
```

7) NanoDet 是一个速度超快和轻量级的移动端 Anchor-free 目标检测模型。测试方法如下所示

a. 编译生成的 **nanodet** 可执行文件在如下路径中，**注意下面的命令执行路径为 ncnn 源码的上一级目录**

```
orangepi@orangepi:~$ ls ncnn/build/examples/nanodet
ncnn/build/examples/nanodet
```

b. 首先新建一个 **nanodet_demo** 的文件夹

```
orangepi@orangepi:~$ mkdir nanodet_demo
```

c. 然后将编译生成的 **nanodet** 可执行程序复制到 **nanodet_demo** 文件夹中

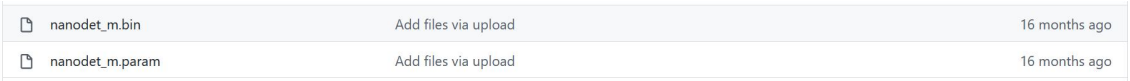
```
orangepi@orangepi:~$ cp ncnn/build/examples/nanodet nanodet_demo/
```

d. 然后需要下载 **nanodet** 模型文件并上传到 **nanodet_demo** 文件夹中

a) **nanodet** 模型文件下载地址如下所示

https://github.com/nihui/ncnn-assets/tree/master/models

b) 打开上面的链接，找 **nanodet_m.bin** 和 **nanodet_m.param** 这两个文件，然后下载下来，再上传到开发板 Linux 系统的 **nanodet_demo** 文件夹中



c) 此时 **nanodet_demo** 文件夹中应该有了下面的三个文件

```
orangepi@orangepi:~$ cd nanodet_demo
orangepi@orangepi:~/nanodet_demo$ ls
nanodet  nanodet_m.bin  nanodet_m.param
```

e. 然后还需要将想检测的图片放在 **nanodet_demo** 文件夹中，比如下面这张有很多车的图片(可以使用手机拍几张车流或者动物等的图片)

```
orangepi@orangepi:~/nanodet_demo$ ls
car.jpg  nanodet  nanodet_m.bin  nanodet_m.param
```

f. 然后运行下面的命令就可以使用 **nanodet** 进行目标检测，**car.jpg** 请替换成你的图片的名字

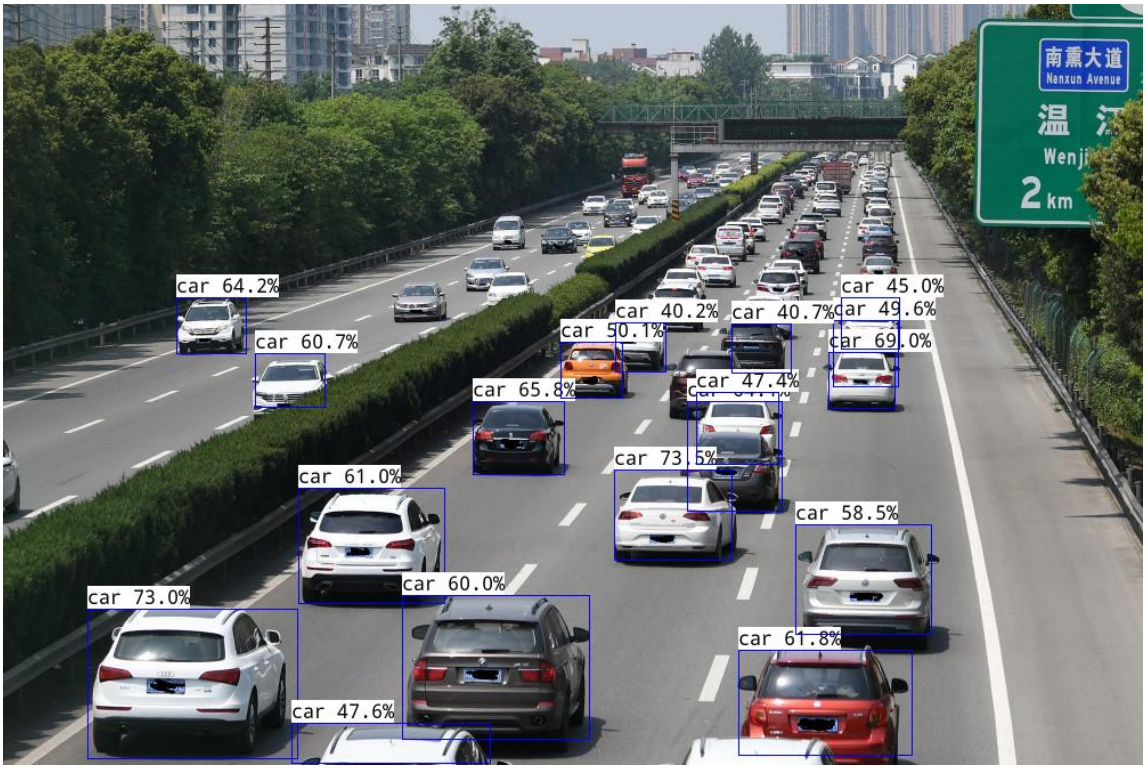
```
orangepi@orangepi:~/nanodet_demo$ ./nanodet car.jpg
2 = 0.73488 at 536.36 408.79 103.68 x 79.63
```

```
2 = 0.73003 at 74.47 530.85 184.61 x 131.70
2 = 0.68989 at 724.94 305.76 58.30 x 49.73
2 = 0.65828 at 412.10 348.38 80.33 x 64.65
2 = 0.64167 at 152.09 257.67 61.52 x 49.91
2 = 0.64124 at 600.63 348.06 83.82 x 96.93
2 = 0.61759 at 645.80 566.98 152.59 x 92.63
2 = 0.61004 at 259.78 424.55 128.62 x 101.88
2 = 0.60663 at 221.76 306.18 61.54 x 47.73
2 = 0.60043 at 350.11 518.50 164.37 x 126.59
2 = 0.58546 at 695.82 456.88 119.12 x 96.87
2 = 0.50075 at 489.94 296.66 54.39 x 49.01
2 = 0.49616 at 728.32 277.73 57.70 x 58.59
2 = 0.47553 at 253.21 630.12 174.18 x 35.88
2 = 0.47408 at 608.36 340.09 72.34 x 63.40
2 = 0.44989 at 735.15 257.97 51.55 x 45.79
2 = 0.40665 at 639.82 280.51 52.43 x 43.13
2 = 0.40169 at 537.50 279.67 44.98 x 43.54
imshow save image to image.png
waitKey stub
```

g. 检测完成的结果会保存在名为 **image.png** 的图片中

```
orangepi@orangepi:~/nanodet_demo$ ls
car.jpg   image.png   nanodet   nanodet_m.bin   nanodet_m.param
```

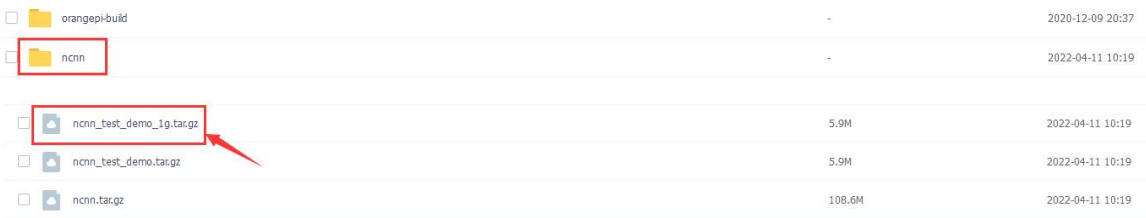
h. 如果使用的是桌面版的 Linux 系统，可以直接打开 **image.png** 查看，如果使用的是服务器版本的 Linux 系统，可以将 **image.png** 拷贝到电脑中查看，**image.png** 内容如下图所示，可以看到识别到的物体左上角会显示物体的种类和可信度的百分比



8) 为了方便测试 **benchncnn** 和 **nanodet**，我整理了一份只包含 **benchncnn** 和 **nanodet** 的可执行文件以及测试需要的模型文件打包成一个 **ncnn_test_demo_1g.tar.gz** 压缩包放在百度云盘上面了，不需要下载编译 **ncnn** 的源码，使用这份可执行程序就可以直接开始测试了

a. 从下面的百度云盘链接可以下载 **ncnn_test_demo_1g.tar.gz** 压缩包。进入 **ncnn** 文件夹中就能看到

链接：https://pan.baidu.com/s/1vWQmCmSYdH7iCDFyKpJtVw
提取码：zero



b. 下载完 **ncnn_test_demo_1g.tar.gz** 压缩包后，首先将 **ncnn_test_demo_1g.tar.gz** 压缩包上传到开发板 linux 系统中

c. 然后使用下面的命令解压 **ncnn_test_demo_1g.tar.gz**

```
orangepi@orangepi:~$ tar ncnn_test_demo_1g.tar.gz
```

d. 解压后进入 **ncnn_test_demo_1g** 目录可以看到其中包含 **benchncnn_demo**

和 **nanodet_demo** 两个子文件夹，它们分别用来测试 **benchncnn** 和 **nanodet**

```
orangepi@orangepi:~$ cd ncnn_test_demo_1g
orangepi@orangepi:~/ncnn_test_demo_1g$ ls
benchncnn_demo  nanodet_demo
```

e. 进入 **benchncnn_demo** 文件夹，然后运行**./benchncnn 4 $(nproc) 0 -1** 这个命令就可以直接测试神经网络的推理性能

```
orangepi@orangepi:~/ncnn_test_demo_1g$ cd benchncnn_demo
orangepi@orangepi:~/ncnn_test_demo_1g/benchncnn_demo$ ./benchncnn 4  \
$(nproc) 0 -1
```

f. 进入 **nanodet_demo** 文件夹，然后运行**./nanodet car.jpg** 这个命令就可以直接使用 **nanodet** 来检测 **car.jpg** 图片中的物体了，你也可以将想要检测的图片放在 **nanodet_demo** 文件夹中，然后使用 **nanodet** 来检测

```
orangepi@orangepi:~/ncnn_test_demo_1g$ cd nanodet_demo
orangepi@orangepi:~/ncnn_test_demo_1g/nanodet_demo$ ./nanodet car.jpg
```

注意，此小节演示的内容主要是证明下 ncnn 在 Orange Pi 的开发板和系统中是可以正常编译运行的，如果此小节演示的内容有问题，可以提供技术支持（比如 ncnn 源码下载失败、ncnn 编译有问题、benchncnn 和 nanodet 测试有问题），但是超过此小节的其他东西就无法提供技术支持了，请自行研究。

3.39. face_recognition 人脸识别库的安装和测试方法

注意，此小节的内容都是在桌面版本的 Linux 系统中测试的，所以请确保开发板使用的系统为桌面版本的系统。

另外下面的安装测试都是在 orangepi 用户下进行的，请保持环境一致。

face_recognition 源码仓库的地址为：

https://github.com/ageitgey/face_recognition

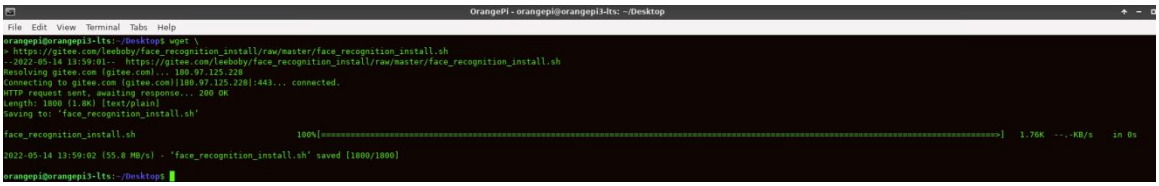
face_recognition 中文版本的说明文档为：

https://github.com/ageitgey/face_recognition/blob/master/README_Simplified_Chinese.md

3.39.1. 使用脚本自动安装 face_recognition 的方法

1) 首先在桌面中打开一个终端，然后下载 **face_recognition_install.sh**

```
orangepi@orangepi:~/Desktop$ wget \
https://gitee.com/leeboby/face_recognition_install/raw/master/face_recognition_install.sh
```



2) 然后执行下面的命令开始安装 **face_recognition**

```
orangepi@orangepi:~/Desktop$ bash face_recognition_install.sh
```

3) face_recognition 安装完后会自动下载 face_recognition 的源码，然后自动运行 face_recognition 中的一些示例，如果最后能看到桌面上弹出了下面的这些图片就说明 face_recognition 安装测试成功了

3.39.2. 手动安装 face_recognition 的方法

1) 首先新建**~/.pip** 目录，再添加 **pip.conf** 配置文件，并在其中设置 pip 的镜像源为清华源，需要执行的命令如下所示：

```
orangepi@orangepi:~$ mkdir -p ~/.pip
orangepi@orangepi:~$ cat <<EOF > ~/.pip/pip.conf
[global]
timeout = 6000
```

```
index-url = https://pypi.tuna.tsinghua.edu.cn/simple
trusted-host = pypi.tuna.tsinghua.edu.cn
EOF
```

2) 然后安装依赖包

```
orangepi@orangepi:~$ sudo apt update
orangepi@orangepi:~$ sudo apt install -y python3-pip libopencv-dev \
python3-opencv imagemagick python3-scipy python3-setuptools python3-wheel \
python3-dev cmake python3-testresources
```

3) 然后更新下 pip3

```
orangepi@orangepi:~$ python3 -m pip install -U pip setuptools wheel
```

4) 安装 **face_recognition** 前首先需要安装下 **dlib** 这个库，由于 dlib 这个库在开发板上编译安装比较慢，所以我在 **gitee** 上保存了一份编译好的 dlib whl 文件，下载后直接安装就可以了。dlib whl 文件下载地址如下所示：

```
https://gitee.com/leeboby/python_whl
```

a. 首先将 python_whl 仓库下载到开发板的 Linux 系统中

```
orangepi@orangepi:~$ git clone --depth=1 https://gitee.com/leeboby/python_whl
```

b. 在 python_whl 文件夹中可以看到有多个版本的 dlib 安装包，dlib 不同版本对应的 Linux 系统如下所示：

Ubuntu18.04	**dlib-19.24.0-cp36-cp36m-linux_aarch64.whl**
Ubuntu20.04	**dlib-19.24.0-cp38-cp38-linux_aarch64.whl**
Ubuntu22.04	**dlib-19.24.0-cp310-cp310-linux_aarch64.whl**
Debian10	**dlib-19.24.0-cp37-cp37m-linux_aarch64.whl**
Debian11	**dlib-19.24.0-cp39-cp39-linux_aarch64.whl**

c. 然后就可以开始安装 dlib，命令如下所示

a) Ubuntu18.04

```
orangepi@orangepi:~$ cd python_whl
orangepi@orangepi:~/python_whl$ python3 -m pip install
dlib-19.24.0-cp36-cp36m-linux_aarch64.whl
```

b) Ubuntu20.04

```
orangepi@orangepi:~$ cd python_whl
orangepi@orangepi:~/python_whl$ python3 -m pip install
```

```
dlib-19.24.0-cp38-cp38-linux_aarch64.whl
```

c) Ubuntu22.04

```
orangepi@orangepi:~$ cd python_whl
orangepi@orangepi:~/python_whl$ python3 -m pip install
dlib-19.24.0-cp310-cp310-linux_aarch64.whl
```

d) Debian10

```
orangepi@orangepi:~$ cd python_whl
orangepi@orangepi:~/python_whl$ python3 -m pip install
dlib-19.24.0-cp37-cp37m-linux_aarch64.whl
```

e) Debian11

```
orangepi@orangepi:~$ cd python_whl
orangepi@orangepi:~/python_whl$ python3 -m pip install
dlib-19.24.0-cp39-cp39-linux_aarch64.whl
```

d. 安装完后如果使用下面的命令能正常打印 dlib 的版本号，就说明 dlib 安装正确

```
orangepi@orangepi:~/python_whl$ python3 -c "import dlib; print(dlib.__version__)"
19.24.0
```

5) 然后安装下 **face_recognition_models-0.3.0-py2.py3-none-any.whl**

```
orangepi@orangepi:~/python_whl$ python3 -m pip install \
face_recognition_models-0.3.0-py2.py3-none-any.whl
```

6) 然后安装 **face_recognition**

```
orangepi@orangepi:~$ python3 -m pip install face_recognition
```

7) 然后**需要重新打开一个终端**，才能找到并运行 **face_detection** 和 **face_recognition** 这两个命令

a. face_recognition 命令用来在单张图片或一个图片文件夹中识别是谁的脸

b. face_detection 命令用来在单张图片或一个图片文件夹中定位人脸的位置

```
orangepi@orangepi:~$ which face_detection
/usr/local/bin/face_detection
orangepi@orangepi:~$ which face_recognition
/usr/local/bin/face_recognition
```

或者在终端中运行下面的命令，不用重新打开终端也能找到上面的两个命令

```
orangepi@orangepi:~$ export PATH=/home/orangepi/.local/bin:$PATH
```

3.39.3. face_recognition 的测试方法

注意，下面的操作都是在桌面中演示的，所以首先请连接好 HDMI 显示器，或者使用 NoMachine/VNC 远程登录 Linux 桌面来测试。

1) 在 **face_recognition** 的源码中有一些示例代码，我们可以直接用来测试，face_recognition 源码的下载地址如下所示：

a. GitHub 官方的下载地址

```
orangepi@orangepi:~$ git clone https://github.com/ageitgey/face_recognition.git
```

b. Gitee 镜像下载地址

```
orangepi@orangepi:~$ git clone https://gitee.com/leeboby/face_recognition.git
```

2) face_recognition 示例代码的路径如下所示

```
face_recognition/examples
```

3) face_recognition 的中文说明文档链接如下所示，使用 face_recognition 前请仔细阅读下

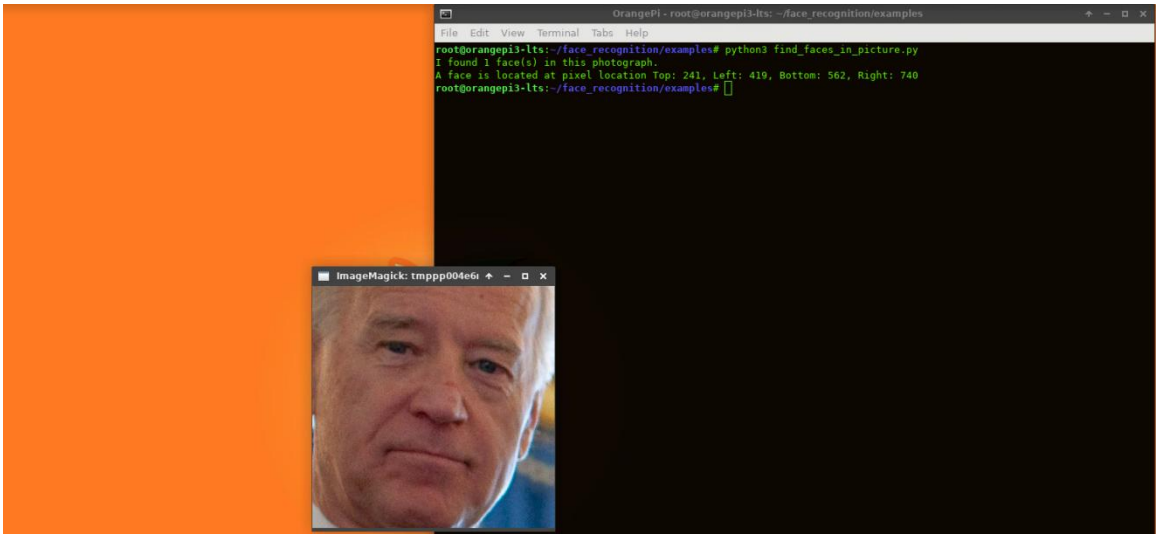
```
https://github.com/ageitgey/face_recognition/blob/master/README_Simplified_Chinese.md
```

4) **find_faces_in_picture.py** 用来在图片中定位人脸的位置，测试步骤如下所示

a. 在桌面中打开一个终端，然后进入 **face_recognition/examples** 目录，再执行下面的命令

```
orangepi@orangepi:~$ cd face_recognition/examples
orangepi@orangepi:~/face_recognition/examples$ python3 find_faces_in_picture.py
I found 1 face(s) in this photograph.
A face is located at pixel location Top: 241, Left: 419, Bottom: 562, Right: 740
```

b. 等待一段时间会弹出下面的图片，这就是在测试图片中定位到的人脸

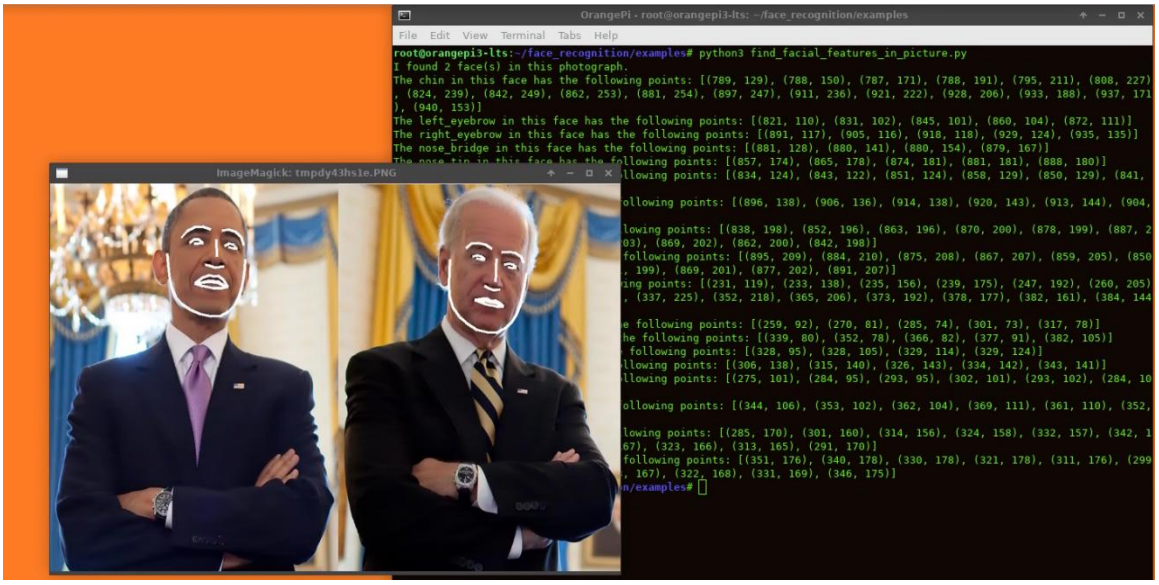


5) **find_facial_features_in_picture.py** 用来识别单张图片中人脸的关键点，测试步骤如下所示

a. 在桌面中打开一个终端，然后进入 **face_recognition/examples** 目录，再执行下面的命令

```
orangepi@orangepi:~$ cd face_recognition/examples
orangepi@orangepi:~/face_recognition/examples$ python3 \
find_facial_features_in_picture.py
```

b. 等待一段时间会弹出下面的图片，可以看到将人脸轮廓都标注出来了



6) **identify_and_draw_boxes_on_faces.py** 用来识别人脸并使用方框标注，测试步骤如下所示

a. 在桌面中打开一个终端，然后进入 **face_recognition/examples** 目录，再执行下面的命令

```
orangepi@orangepi:~$ cd face_recognition/examples
orangepi@orangepi:~/face_recognition/examples$ python3 \
identify_and_draw_boxes_on_faces.py
```

b. 等待一段时间会弹出下面的图片，可以看到将图片中的人脸都使用方框标注出来了，并且正确显示了人物的名字

7) **face_distance.py** 用来在不同精度上比较两个人脸是否属于一个人，首先打开一个终端，然后进入 **face_recognition/examples** 目录，再执行下面的命令就可以看到测试的输出结果

```
orangepi@orangepi:~$ cd face_recognition/examples
orangepi@orangepi:~/face_recognition/examples$ python3 face_distance.py
The test image has a distance of 0.35 from known image #0
- With a normal cutoff of 0.6, would the test image match the known image? True
- With a very strict cutoff of 0.5, would the test image match the known image? True

The test image has a distance of 0.82 from known image #1
- With a normal cutoff of 0.6, would the test image match the known image? False
- With a very strict cutoff of 0.5, would the test image match the known image?
  False
```

8) **recognize_faces_in_pictures.py** 用来识别未知图片中的人脸是谁。首先打开一个终端，然后进入 **face_recognition/examples** 目录，再执行下面的命令，等待一端时

间后就能看到测试结果

```
orangepi@orangepi:~$ cd face_recognition/examples
orangepi@orangepi:~/face_recognition/examples$ python3 \
recognize_faces_in_pictures.py
Is the unknown face a picture of Biden? False
Is the unknown face a picture of Obama? True
Is the unknown face a new person that we've never seen before? False
```

9) **facerec_from_webcam_faster.py** 用来识别 USB 摄像头中的人脸，测试步骤如下所示：

a. 首先请将 USB 摄像头插入开发板的 USB 接口中，然后通过 **v4l2-ctl**（**注意 v4l2 中的 l 是小写字母 l，不是数字 1**）命令查看下 USB 摄像头的设备节点的序号

```
orangepi@orangepi:~$ sudo apt update
orangepi@orangepi:~$ sudo apt install -y v4l-utils
orangepi@orangepi:~$ v4l2-ctl --list-devices
cedrus (platform:cedrus):
        /dev/video0

USB2.0 UVC PC Camera: USB2.0 UV (usb-5311000.usb-1):
        /dev/video1
        /dev/video2
```

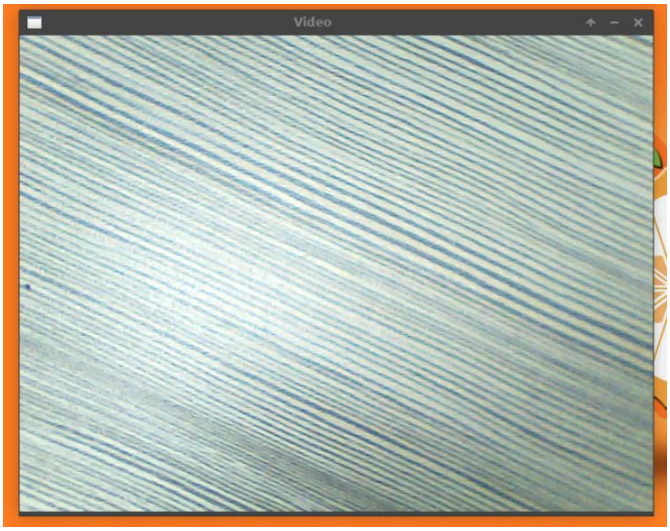
b. 然后在桌面中打开一个终端，进入 **face_recognition/examples** 目录后，首先修改下 **facerec_from_webcam_faster.py** 中使用的摄像头的设备序号。比如上面通过 **v4l2-ctl --list-devices** 命令查看到 USB 摄像头为**/dev/video1**，那就修改 **cv2.VideoCapture(0)**中的 **0** 为 **1**

```
orangepi@orangepi:~$ cd face_recognition/examples
orangepi@orangepi:~/face_recognition/examples$ vim \
facerec_from_webcam_faster.py
video_capture = cv2.VideoCapture(1)
```

c. 然后执行下面的命令运行 **facerec_from_webcam_faster.py**

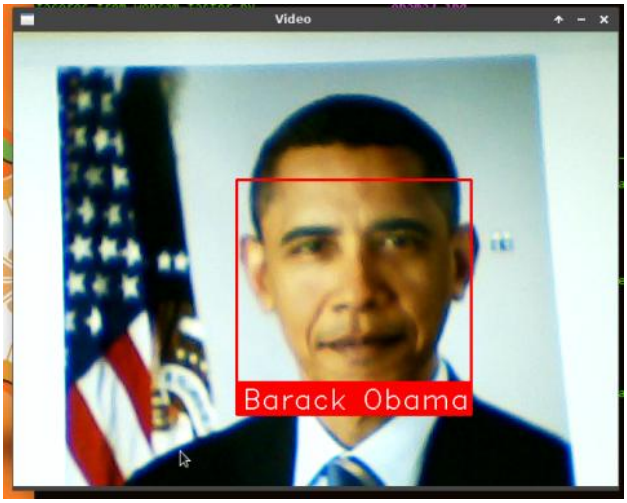
```
orangepi@orangepi:~/face_recognition/examples$ python3 \
facerec_from_webcam_faster.py
```

d. 等待一段时间会弹出摄像头的显示画面



e. 此时可以将摄像头对准自己，当摄像头检测到人脸时，会将检测到的人脸使用方框框起来。**注意，检测人脸时，摄像头显示的画面会比较卡顿，请不要移动过快**

f. 还可以打开一张奥巴马的图片，然后使用摄像头对准打开的图片，可以看到不仅能将人脸标注出来，还能正确显示检测到的人脸的名字。**注意，检测人脸时，摄像头显示的画面会比较卡顿，请不要移动过快**



10) **web_service_example.py** 是一个非常简单的使用 Web 服务上传图片运行人脸识别的案例，后端服务器会识别这张图片是不是奥巴马，并把识别结果以 json 键值对输出，测试步骤如下所示：

a. 在桌面中打开一个终端，然后进入 **face_recognition/examples** 目录，再执行下面的命令（**如果是使用脚本自动安装的 face_recognition，那么就不需要安装 flask 了**）

```
orangepi@orangepi:~$ python3 -m pip install flask
```

```
orangepi@orangepi:~$ cd face_recognition/examples
root@orangepi:~/face_recognition/examples$ python3 web_service_example.py
 * Serving Flask app 'web_service_example' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on all addresses (0.0.0.0)
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:5001
 * Running on http://192.168.1.79:5001 (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 500-161-390
```

b. 然后运行下面的命令就可以返回图片识别的结果（注意，下面的命令执行路径为 **face_recognition/examples**）

```
orangepi@orangepi:~/face_recognition/examples$ curl -XPOST -F \
"file=@obama2.jpg" http://127.0.0.1:5001
{
  "face_found_in_image": true,
  "is_picture_of_obama": true
}
```

c. 我们也可以将 **face_recognition/examples/obama2.jpg** 这张图片拷贝到其他的 Linux 电脑中，当然也可以自己准备一张名为 **obama2.jpg** 的图片，然后在 Linux 电脑中可以使用下面的命令远程通过开发板运行的服务来识别人脸（**注意命令中的 IP 地址需要替换为开发板的 IP 地址，file 后的文件名需要替换为想要测试的图片的名字**）

```
test@test:~$ curl -XPOST -F "file=@obama2.jpg" http://192.168.1.79:5001
{
  "face_found_in_image": true,
  "is_picture_of_obama": true
}
```

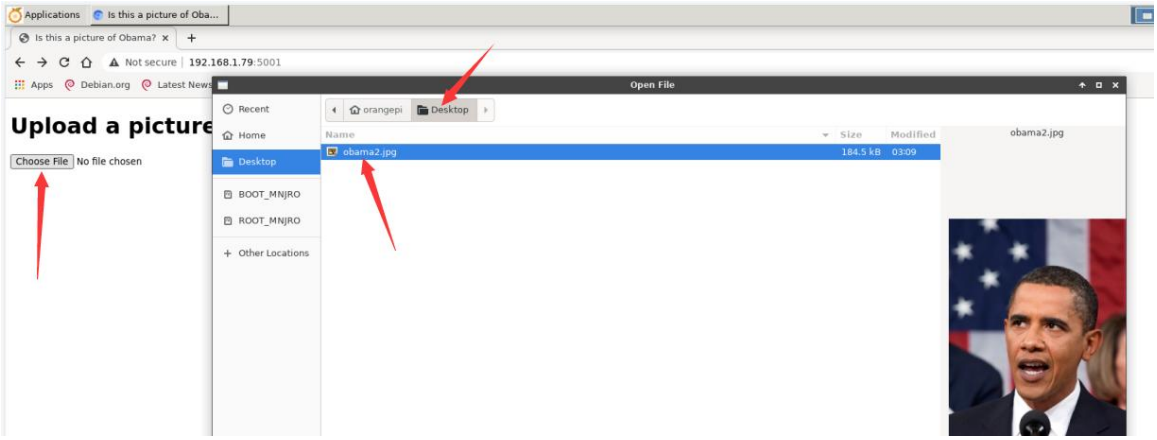
d. 使用浏览器测试的方法如下所示：

a) 首先打开浏览器，然后在浏览器的地址栏输入**开发板的 IP 地址:5001**，然后就能看到下面的页面

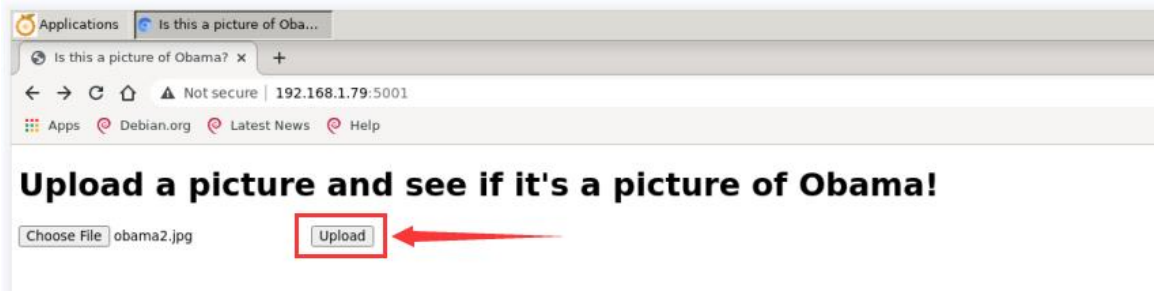
b) 然后将 obama2.jpg 复制到桌面

```
orangepi@orangepi:~/face_recognition/examples$ cp obama2.jpg \
/home/orangepi/Desktop/
```

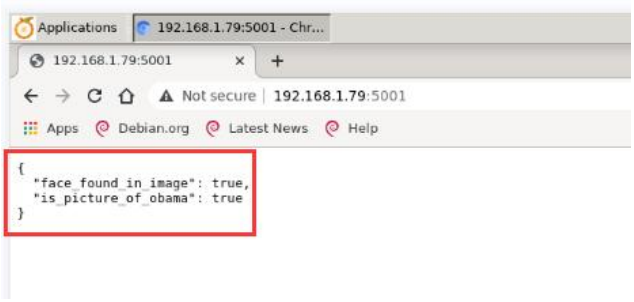
c) 然后在浏览器中选择刚才复制的图片



d) 然后点击 **Upload** 上传刚才选择的图片进行人脸识别



e) 等待一段时间后就会显示检测的结果



11) **face_detection** 命令测试示例

a. face_detection 命令行工具可以在单张图片或一个图片文件夹中定位人脸位置（输出像素点坐标）。使用 **face_detection --help** 可以查看下 face_detection 命令的帮助信息

```
orangepi@orangepi:~$ face_detection --help
Usage: face_detection [OPTIONS] IMAGE_TO_CHECK

Options:
  --cpus INTEGER    number of CPU cores to use in parallel. -1 means "use all in
                    system"
  --model TEXT      Which face detection model to use. Options are "hog" or
                    "cnn".
  --help            Show this message and exit.
```

b. 检测单张图片的示例如下所示：

```
orangepi@orangepi:~$ cd face_recognition/examples
orangepi@orangepi:~/face_recognition/examples$ face_detection obama2.jpg
obama2.jpg,302,474,611,164
```

c. 使用多核并行检测多张图片的示例如下所示：

a) 首先进入 **face_recognition/examples** 文件夹

b) 然后新建一个 test 文件夹

c) 然后将 jpg 图片都拷贝到 test 文件夹中

d) 然后使用所有的 cpu 并行运行 **face_detection** 来检查 test 文件夹中的图片，其中**--cpus -1** 表示使用所有的 cpu

```
orangepi@orangepi:~$ cd face_recognition/examples
orangepi@orangepi:~/face_recognition/examples$ mkdir test
orangepi@orangepi:~/face_recognition/examples$ cp *.jpg test
orangepi@orangepi:~/face_recognition/examples$ face_detection --cpus -1 test
```

```
test/obama-240p.jpg,29,261,101,189
test/obama_small.jpg,65,215,169,112
test/obama2.jpg,302,474,611,164
test/two_people.jpg,62,394,211,244
test/two_people.jpg,95,941,244,792
test/obama.jpg,136,624,394,366
test/obama-480p.jpg,65,507,189,383
test/obama-720p.jpg,94,751,273,572
test/obama-1080p.jpg,136,1140,394,882
test/biden.jpg,233,749,542,439
```

12) **face_recognition** 命令测试示例

a. **face_recognition** 命令行工具可以在单张图片或者一个图片文件夹中认出是谁的脸。使用 **face_recognition --help** 可以查看下 face_recognition 命令的帮助信息

```
orangepi@orangepi:~$ face_recognition --help
Usage: face_recognition [OPTIONS] KNOWN_PEOPLE_FOLDER
IMAGE_TO_CHECK

Options:
  --cpus INTEGER                  number of CPU cores to use in parallel (can speed
                                  up processing lots of images). -1 means "use all in
                                  system"
  --tolerance FLOAT               Tolerance for face comparisons. Default is 0.6.
                                  Lower this if you get multiple matches for the same
                                  person.
  --show-distance BOOLEAN         Output face distance. Useful for tweaking tolerance
                                  setting.
  --help                          Show this message and exit.
```

b. 首先新建一个已知名字的人脸图片文件夹 **known_people**，然后复制两张图片到 **known_people** 中，然后将 **obama2.jpg** 复制为 **unkown.jpg**，也就是我们要识别的图片

```
orangepi@orangepi:~$ cd face_recognition/examples
orangepi@orangepi:~/face_recognition/examples$ mkdir known_people
orangepi@orangepi:~/face_recognition/examples$ cp biden.jpg obama.jpg \
```

```
known_people
orangepi@orangepi:~/face_recognition/examples$ cp obama2.jpg unkown.jpg
```

c. 然后就可以使用下面的命令识别下 **unkown.jpg** 图片中人物的名字，可以看到识别到 unkown.jpg 图片为 obama

```
orangepi@orangepi:~/face_recognition/examples$ face_recognition known_people \
unkown.jpg
unkown.jpg,obama
```

d. 如果我们识别一张不相关的图片，就会显示 unknown_person

```
root@orangepi:~/face_recognition/examples$ face_recognition known_people \
alex-lacamoire.png
alex-lacamoire.png,unknown_person
```

e. 我们还可以新建一个 test 文件夹，然后在其中放入多张图片，然后就可以使用所有的 CPU 来并行识别所有的图片

```
orangepi@orangepi:~/face_recognition/examples$ mkdir test
orangepi@orangepi:~/face_recognition/examples$ cp *.jpg *.png test
orangepi@orangepi:~/face_recognition/examples$ face_recognition --cpus -1 \
known_people test
test/obama-240p.jpg,obama
test/alex-lacamoire.png,unknown_person
test/obama_small.jpg,obama
test/unkown.jpg,obama
test/obama2.jpg,obama
test/lin-manuel-miranda.png,unknown_person
test/two_people.jpg,biden
test/two_people.jpg,obama
test/obama-720p.jpg,obama
test/obama.jpg,obama
test/obama-480p.jpg,obama
test/biden.jpg,biden
test/obama-1080p.jpg,obama
```

3.40. Tensorflow 的安装方法

注意，安装 Tensorflow 前请确保使用的 Linux 系统为 Debian Buster。此小节演示的 Tensorflow 的安装方法不能保证在其他版本的 Linux 系统中能正常使用。

如果使用的是桌面版本的 Debian Buster 系统，安装 Tensorflow 前需要先关闭桌面，这样可以释放 200MB 左右的内存。如果不关闭桌面，会由于内存不足导致安装失败。关闭桌面的方法请查看 Linux 桌面版系统禁用桌面的方法一节的说明。安装完后再打开桌面即可。

安装过程中如果打印下面的错误可以不用管它，只要安装没有中断就没问题（此错误是内存不足造成的）：

Write-error on swap-device (253:1:304568)

3.40.1. 使用脚本自动安装 Tensorflow 的方法

1) 首先需要扩展下/tmp 空间的大小，设置完后需要重启下开发板的 linux 系统，命令如下所示：

```
orangepi@orangepi:~$ sudo sed -i 's/nosuid/&,size=2G/' /etc/fstab
orangepi@orangepi:~$ sudo reboot
```

2) 重启后，可以看到/tmp 空间的大小变为 2G 了

```
orangepi@orangepi:~$ df -h | grep "/tmp"
tmpfs                   2.0G     12K    2.0G     1% /tmp
```

3) 然后使用下面的命令下载 Orange Pi 提供的 tensorflow 安装脚本

```
orangepi@orangepi:~$ wget  \
https://gitee.com/leeboby/tensorflow/raw/master/install_tensorflow.sh
```

4) 然后运行 install_tensorflow.sh 脚本开始安装 tensorflow

```
orangepi@orangepi:~$ sudo bash install_tensorflow.sh
```

5) tensorflow 安装完成后会自动测试打印下 tensorflow 的版本号，如果最后能看到下面的输出，说明 tensorflow 安装成功

```
##### Start Test Tensorflow #####
Tensorflow version is : 2.4.0
###### End Test Tensorflow ######
```

3.40.2. 手动安装 Tensorflow 的步骤

1) 首先需要扩展下/tmp 空间的大小，设置完后需要重启下开发板的 linux 系统，命

令如下所示：

```
orangepi@orangepi:~$ sudo sed -i 's/nosuid/&,size=2G/' /etc/fstab
orangepi@orangepi:~$ sudo reboot
```

2) 重启后，可以看到/tmp 空间的大小变为 2G 了

```
orangepi@orangepi:~$ df -h | grep "/tmp"
tmpfs                   2.0G     12K  2.0G     1% /tmp
```

3) 然后使用下面的命令设置 pip 的源为清华源，加快 Python 包的下载速度

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

4) 然后安装依赖包

```
orangepi@orangepi:~$ sudo apt update
orangepi@orangepi:~$ sudo apt install -y python3-pip gfortran    \
libopenblas-dev liblapack-dev libatlas-base-dev libblas-dev    \
libhdf5-dev hdf5-tools libhdf5-dev zlib1g-dev zip libjpeg62-turbo-dev    \
python3-dev pkg-config python3-setuptools python3-wheel
```

5) 然后下载 tensorflow 相关的 **whl** 包

```
orangepi@orangepi:~$ git clone --depth=1 https://gitee.com/leeboby/tensorflow.git
```

6) 然后进入 **tensorflow** 目录安装 tensorflow 依赖的 whl 包

```
orangepi@orangepi:~$ cd tensorflow
orangepi@orangepi:~/tensorflow$ pip3 install    \
tensorflow/grpcio-1.32.0-cp37-cp37m-linux_aarch64.whl
orangepi@orangepi:~/tensorflow$ pip3 install    \
tensorflow/numpy-1.19.5-cp37-cp37m-linux_aarch64.whl
orangepi@orangepi:~/tensorflow$ pip3 install    \
tensorflow/h5py-2.10.0-cp37-cp37m-linux_aarch64.whl
```

7) 然后就可以使用下面的命令安装 tensorflow

```
orangepi@orangepi:~/tensorflow$ pip3 install  \
tensorflow-2.4.0-cp37-none-linux_aarch64.whl
```

8) 安装完 tensorflow 后，可以使用下面的命令打印下 tensorflow 的版本号，如果能正常打印出 tensorflow 的版本号 2.4.0，就说明 tensorflow 安装成功

```
orangepi@orangepi:~/tensorflow$ python3 -c  \
"import tensorflow; print(tensorflow.__version__)"
2.4.0
```

9) 参考资料

```
https://github.com/lhelontra/tensorflow-on-arm
https://tf.kmtea.eu/whl/stable.html
https://www.tensorflow.org
https://repo.rock-chips.com/pypi/simple
```

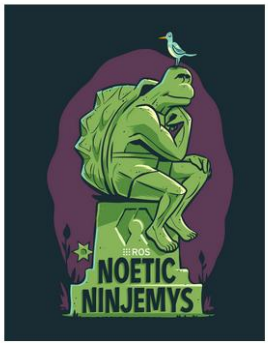
3.41. ROS 安装方法

3.41.1. ROS 1 Noetic 的安装方法

1) ROS 1 当前活跃的版本如下所示，推荐版本为 Noetic Ninjemys

Active ROS 1 distributions

Recommended



Distro	Release date	Poster	*Tuturtle*, turtle in tutorial	EOL date
ROS Noetic Ninjemys (**Recommended**)	May 23rd, 2020			May, 2025 (Focal EOL)
ROS Melodic Morenia	May 23rd, 2018			May, 2023 (Bionic EOL)

http://docs.ros.org
https://wiki.ros.org/Distributions

2) ROS 1 **Noetic Ninjemys** 官方安装文档链接如下所示：

a. Ubuntu

http://wiki.ros.org/noetic/Installation/Ubuntu

b. Debian

http://wiki.ros.org/noetic/Installation/Debian

3) ROS **Noetic Ninjemys** 官方安装文档中 Ubuntu Linux 推荐使用 Ubuntu20.04，所以请确保开发板使用的系统为 **Ubuntu20.04**，**如果想使用 Debian 系统，请自行测试，这里就不演示了**

http://wiki.ros.org/noetic/Installation

4) 首先添加 ros 的软件源

```
orangepi@orangepi:~$ sudo sh -c 'echo   \
"deb http://mirrors.ustc.edu.cn/ros/ubuntu $(lsb_release -sc) main"   \
> /etc/apt/sources.list.d/ros-latest.list'
```

5) 然后设置 Keys

```
orangepi@orangepi:~$ sudo apt-key adv    \
--keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key \
C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654

Executing: /tmp/apt-key-gpghome.0zbZ9ucMdb/gpg.1.sh --keyserver
hkp://keyserver.ubuntu.com:80 --recv-key
C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
gpg: key F42ED6FBAB17C654: public key "Open Robotics <info@osrfoundation.org>"
imported
gpg: Total number processed: 1
gpg:                    imported: 1
```

6) 然后更新下 apt 仓库缓存

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

7) 然后安装 ros

```
orangepi@orangepi:~$ sudo apt install -y ros-noetic-desktop-full
```

8) 运行 ros 中的命令前需要先设置下环境变量

```
orangepi@orangepi:~$ source /opt/ros/noetic/setup.bash
```

9) 如果不想每次使用 ros 中的命令前都手动设置环境变量，可以将设置环境变量的命令加入**~/.bashrc** 中，这样每次打开新的终端时都会自动设置好 ros 的环境变量

```
orangepi@orangepi:~$ echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
orangepi@orangepi:~$ source ~/.bashrc
```

10) 现在就已经安装好了运行核心 ros 包所需的东西了。如果要创建和管理自己的 ros 工作空间，还需要安装一些其他的工具。例如，rosinstall 是一个常用的命令行工具，使用它可以下载一些 ros 包的源码树。运行下面的命令可以安装此工具和构建 ros 包需要的其它依赖项

```
orangepi@orangepi:~$ sudo apt install -y python3-rosdep python3-rosinstall   \
python3-rosinstall-generator python3-wstool build-essential
```

11) 使用 ROS 工具前，首先需要初始化下 rosdep，然后编译源码时就能快速的安装

一些系统依赖和一些 ROS 中的核心组件

注意，运行下面的命令需要确保开发板能正常访问 github，否则会由于网络问题而报错。

```
orangepi@orangepi:~$ source /opt/ros/noetic/setup.bash
orangepi@orangepi:~$ sudo rosdep init
Wrote /etc/ros/rosdep/sources.list.d/20-default.list
Recommended: please run

        rosdep update
orangepi@orangepi:~$ rosdep update
reading in sources list data from /etc/ros/rosdep/sources.list.d
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/osx-homebrew.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/base.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/python.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/ruby.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/releases/fuerte.yaml
Query rosdistro index
https://raw.githubusercontent.com/ros/rosdistro/master/index-v4.yaml
Skip end-of-life distro "ardent"
Skip end-of-life distro "bouncy"
Skip end-of-life distro "crystal"
Skip end-of-life distro "dashing"
Skip end-of-life distro "eloquent"
Add distro "foxy"
Add distro "galactic"
Skip end-of-life distro "groovy"
Add distro "humble"
Skip end-of-life distro "hydro"
Skip end-of-life distro "indigo"
Skip end-of-life distro "jade"
Skip end-of-life distro "kinetic"
Skip end-of-life distro "lunar"
Add distro "melodic"
Add distro "noetic"
Add distro "rolling"
```

```
updated cache in /home/orangepi/.ros/rosdep/sources.cache
```

12) 验证 ROS 安装是否正确的方法

a. 首先运行下 **roscore**

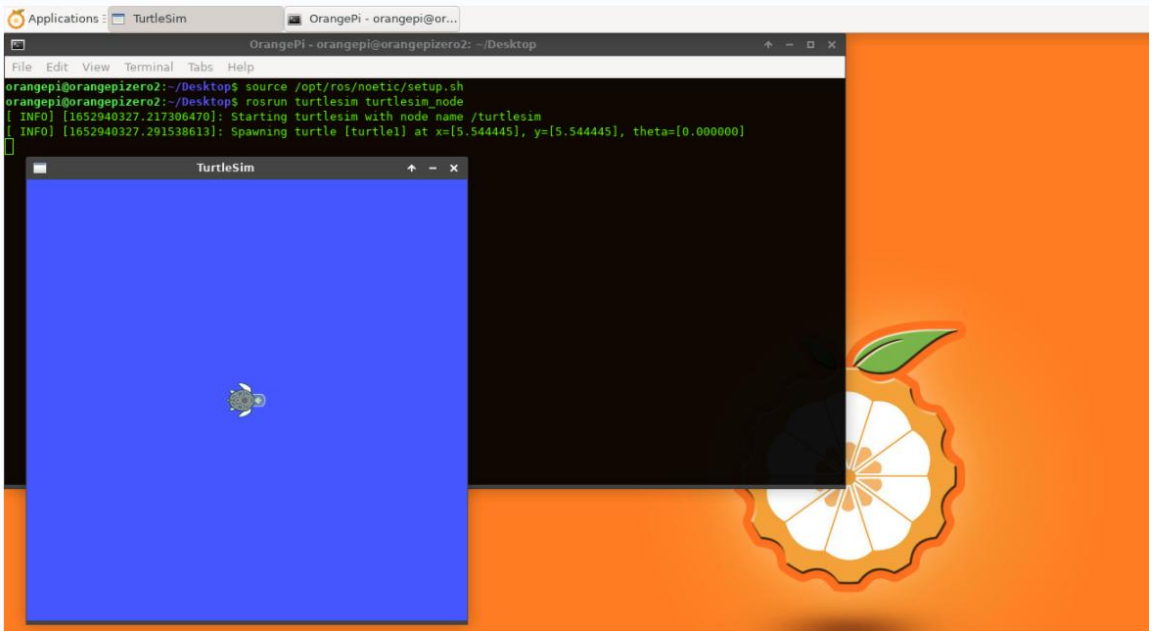
```
orangepi@orangepi:~$ source /opt/ros/noetic/setup.bash
orangepi@orangepi:~$ roscore
... logging to
/home/orangepi/.ros/log/132132c4-c873-11ec-9b13-5099013e1a05/roslaunch-orangepi-1
8381.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://orangepi:44425/
ros_comm version 1.15.14



SUMMARY
========

PARAMETERS
 * /rosdistro: noetic
 * /rosversion: 1.15.14

NODES

auto-starting new master
process[master]: started with pid [18389]
ROS_MASTER_URI=http://orangepi:11311/

setting /run_id to 132132c4-c873-11ec-9b13-5099013e1a05
process[rosout-1]: started with pid [18399]
started core service [/rosout]
```

b. 然后启动一个小海龟的例程来测试下 ROS 是否能正常使用

a) 首先在桌面中打开一个命令行终端窗口

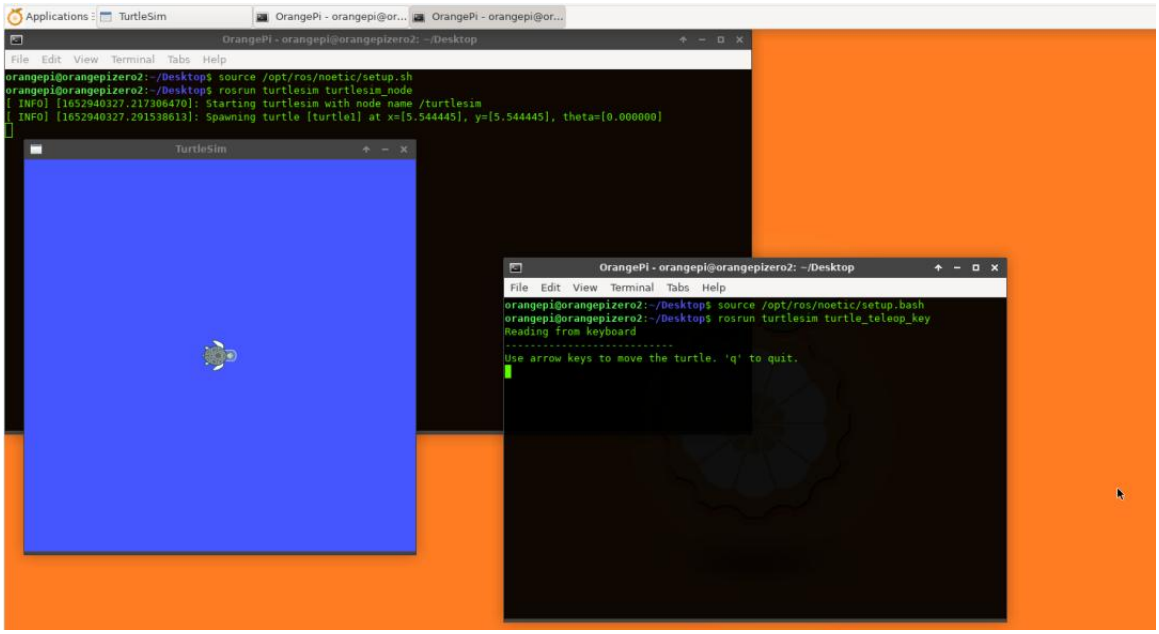
b) 然后输入下面的命令，就会弹出下图所示的一个小海龟

```
orangepi@orangepi:~$ source /opt/ros/noetic/setup.bash
orangepi@orangepi:~$ rosrun turtlesim turtlesim_node
```



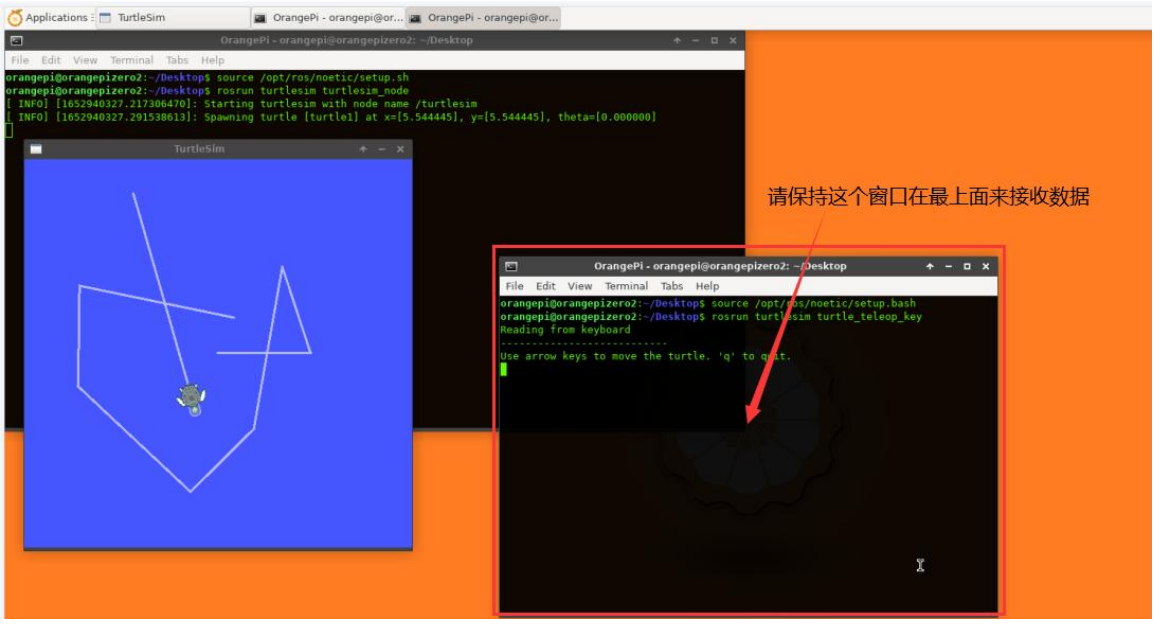
c) 然后再打开一个终端窗口输入下面的命令运行下海龟控制程序

```
orangepi@orangepi:~$ source /opt/ros/noetic/setup.bash
orangepi@orangepi:~$ rosrun turtlesim turtle_teleop_key
```



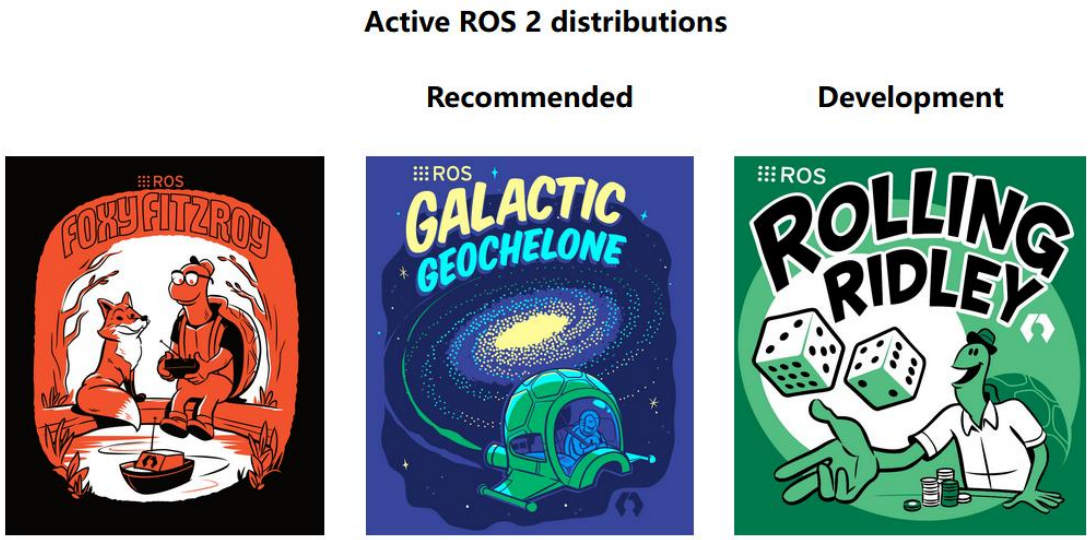
d) 然后请保持刚才打开的海龟控制程序的终端窗口在最上面，此时按下键

盘上的方向按键就可以控制小海龟上下左右移动了



3.41.2. ROS 2 Galactic 的安装方法

1) ROS 2 当前活跃的版本如下所示，推荐版本为 **Galactic Geochelone**



Distro	Release date	Logo	EOL date
Humble Hawksbill	May 23rd, 2022		May 2027
Galactic Geochelone	May 23rd, 2021		November 2022
Foxy Fitzroy	June 5th, 2020		May 2023

http://docs.ros.org
http://docs.ros.org/en/galactic/Releases.html

2) ROS 2 **Galactic Geochelone** 官方安装文档链接如下所示：

docs.ros.org/en/galactic/Installation.html
http://docs.ros.org/en/galactic/Installation/Ubuntu-Install-Debians.html

3) ROS 2 **Galactic Geochelone** 官方安装文档中 Ubuntu Linux 推荐使用 Ubuntu20.04，所以请确保开发板使用的系统为 **Ubuntu20.04**。安装 ROS 2 有几种方法，下面演示下通过 **Debian packages** 的方式来安装 ROS 2 **Galactic Geochelone**

4) 首先添加密钥

```
orangepi@orangepi:~$ sudo apt-key adv --keyserver    \
'hkp://keyserver.ubuntu.com:80'    \
--recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654

[sudo] password for orangepi:
Executing: /tmp/apt-key-gpghome.gNBSKx6Ums/gpg.1.sh --keyserver
hkp://keyserver.ubuntu.com:80 --recv-key
C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
gpg: key F42ED6FBAB17C654: public key "Open Robotics <info@osrfoundation.org>"
imported
gpg: Total number processed: 1
```

```
gpg:                    imported: 1
```

5) 然后添加 ROS 2 的仓库源到 Ubuntu 系统中

```
orangepi@orangepi:~$ echo "deb [arch=$(dpkg --print-architecture)]  \
http://mirrors.ustc.edu.cn/ros2/ubuntu $(source /etc/os-release && echo  \
$UBUNTU_CODENAME) main" | sudo tee /etc/apt/sources.list.d/ros2.list  \
> /dev/null
```

6) 然后更新下 apt 仓库缓存

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

7) 然后就可以安装 ROS 2 相关的软件包，下面的命令会安装 ROS, RViz, demos, tutorials 这些东西

```
orangepi@orangepi:~$ sudo apt install -y ros-galactic-desktop
```

8) 运行 **ros2** 命令前需要先设置下环境变量

```
orangepi@orangepi:~$ source /opt/ros/galactic/setup.bash
orangepi@orangepi:~$ ros2 -h
usage: ros2 [-h] Call `ros2 <command> -h` for more detailed usage. ...

ros2 is an extensible command-line tool for ROS 2.

optional arguments:
  -h, --help            show this help message and exit

Commands:
  action      Various action related sub-commands
  bag         Various rosbag related sub-commands
  component   Various component related sub-commands
  daemon      Various daemon related sub-commands
  doctor      Check ROS setup and other potential issues
  interface   Show information about ROS interfaces
  launch      Run a launch file
  lifecycle   Various lifecycle related sub-commands
  multicast   Various multicast related sub-commands
```

```
node          Various node related sub-commands
param         Various param related sub-commands
pkg           Various package related sub-commands
run           Run a package specific executable
security      Various security related sub-commands
service       Various service related sub-commands
topic         Various topic related sub-commands
wtf           Use `wtf` as alias to `doctor`

Call `ros2 <command> -h` for more detailed usage.
```

9) 可以使用下面的方法测试下 ROS 2 是否安装成功

a. 先打开一个终端，然后使用下面的两条命令运行一个 **C++ talker** 不断的发送 Hello World

```
orangepi@orangepi:~$ source /opt/ros/galactic/setup.bash
orangepi@orangepi:~$ ros2 run demo_nodes_cpp talker
[INFO] [1649599956.390893735] [talker]: Publishing: 'Hello World: 1'
[INFO] [1649599957.390812753] [talker]: Publishing: 'Hello World: 2'
[INFO] [1649599958.390890143] [talker]: Publishing: 'Hello World: 3'
.....
```

b. 然后再开一个终端，使用下面的两条命令运行一个 **Python listener**，如果能接收到上面发送的 Hello World 就说明 RSO 2 的 C++和 Python API 都能正常工作

```
orangepi@orangepi:~$ source /opt/ros/galactic/setup.bash
orangepi@orangepi:~$ ros2 run demo_nodes_py listener
[INFO] [1649600109.504678962] [listener]: I heard: [Hello World: 154]
[INFO] [1649600110.393777793] [listener]: I heard: [Hello World: 155]
[INFO] [1649600111.393769143] [listener]: I heard: [Hello World: 156]
.....
```

10) ROS 的使用方法请参考下 ROS 2 的文档

http://docs.ros.org/en/galactic/Tutorials.html

11) 运行下面的命令可以卸载 ROS 2

```
orangepi@orangepi:~$ sudo apt remove ~nros-galactic-* && sudo apt autoremove
```

```
orangepi@orangepi:~$ sudo rm /etc/apt/sources.list.d/ros2.list
orangepi@orangepi:~$ sudo apt update
orangepi@orangepi:~$ sudo apt autoremove
orangepi@orangepi:~$ sudo apt upgrade
```

3.42. OpenMediaVault 的安装方法

OpenMediaVault 是一款基于 Debian 的 NAS 操作系统。

由下表可以知道：
Debian10 只能安装 OpenMediaVault 5.x 版本;
Debian11 只能安装 OpenMediaVault 6.x 版本。

Table 1: openmediavault historical releases

Version	Codename	Base Distro	Status	Date Released
0.2	Ix	Debian 6	EOL	Oct 2011
0.3	Omnious	Debian 6	EOL	Jul 2012
0.4	Fedaykin	Debian 6	EOL	Sep 2012
0.5	Sardoukar	Debian 6	EOL	Aug 2013
1.0	Kralizec	Debian 7	EOL	Sept 2014
2.0	Stoneburner	Debian 7	EOL	Jun 2015
3.0	Erasmus	Debian 8	EOL	Jun 2016
4.0	Arrakis	Debian 9	EOL	Apr 2018
5.0	Usul	Debian 10	Stable	Mar 2020
6.0	Shaitan	Debian 11	In Development	est. Q2/2022

所以安装前请先选择想要安装的 OpenMediaVault 版本，然后确保开发板使用的 Debian 系统为匹配的系统。

另外 OpenMediaVault 官方推荐使用服务器版本的系统，所以请不要使用桌面版本的系统来安装 OpenMediaVault。

Can I install openmediavault on top a running Debian system? Yes, but it is recommended that the current running OS not to have a desktop environment installed.

3.42.1. Debian10 安装 OpenMediaVault 5.x

注意，Debian10 只能安装 OpenMediaVault 5.x。

1) OpenMediaVault 官方文档如下所示：

a. 5.x 版本的文档（**OpenMediaVault 的稳定版本**）

https://openmediavault.readthedocs.io/en/5.x/

2) 在 Debian10 中安装 OpenMediaVault 的官方文档如下所示：

https://openmediavault.readthedocs.io/en/5.x/installation/on_debian.html

3) 首先安装 OpenMediaVault 的 keyring，注意下面的命令都是在 **root** 用户下执行的

```
root@orangepi:~# apt-get install -y gnupg
root@orangepi:~# wget -O  \
"/etc/apt/trusted.gpg.d/openmediavault-archive-keyring.asc"  \
https://packages.openmediavault.org/public/archive.key
root@orangepi:~# apt-key add  \
"/etc/apt/trusted.gpg.d/openmediavault-archive-keyring.asc"
```

4) 然后添加 OpenMediaVault 的包仓库，注意切换到 **root** 用户再输入下面的命令，黑色字体部分是一条完整的命令，请直接复制

```
root@orangepi:~# cat <<EOF > /etc/apt/sources.list.d/openmediavault.list
deb https://mirrors.tuna.tsinghua.edu.cn/OpenMediaVault/public usul main
deb https://mirrors.tuna.tsinghua.edu.cn/OpenMediaVault/packages usul main
## Uncomment the following line to add software from the proposed repository.
# deb https://mirrors.tuna.tsinghua.edu.cn/OpenMediaVault/public usul-proposed main
# deb https://mirrors.tuna.tsinghua.edu.cn/OpenMediaVault/packages usul-proposed main
## This software is not part of OpenMediaVault, but is offered by third-party
## developers as a service to OpenMediaVault users.
# deb https://mirrors.tuna.tsinghua.edu.cn/OpenMediaVault/public usul partner
# deb https://mirrors.tuna.tsinghua.edu.cn/OpenMediaVault/packages usul partner
EOF
```

上面使用的是清华源，相关说明可以参考下面的链接

https://mirrors.tuna.tsinghua.edu.cn/help/openmediavault/

5) 然后使用下面的命令就可以安装 OpenMediaVault

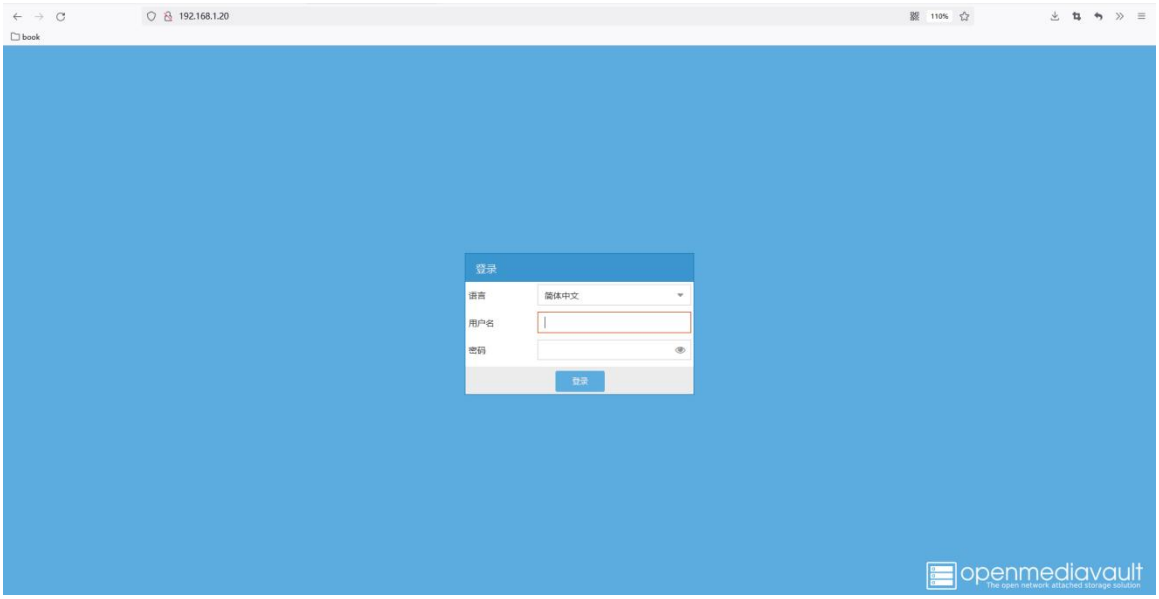
```
root@orangepi:~# export LANG=C.UTF-8
root@orangepi:~# export DEBIAN_FRONTEND=noninteractive
root@orangepi:~# export APT_LISTCHANGES_FRONTEND=none
root@orangepi:~# apt-get update
root@orangepi:~# apt-get --yes --auto-remove --show-upgraded \
--allow-downgrades --allow-change-held-packages \
```

```
--no-install-recommends \
--option DPkg::Options::="--force-confdef" \
--option DPkg::Options::="--force-confold" \
install openmediavault-keyring openmediavault
```

6) 然后运行下面的命令，运行结束后，再在浏览器中输入开发板的 IP 地址，即可打开 OpenMediaVault 的登录页面

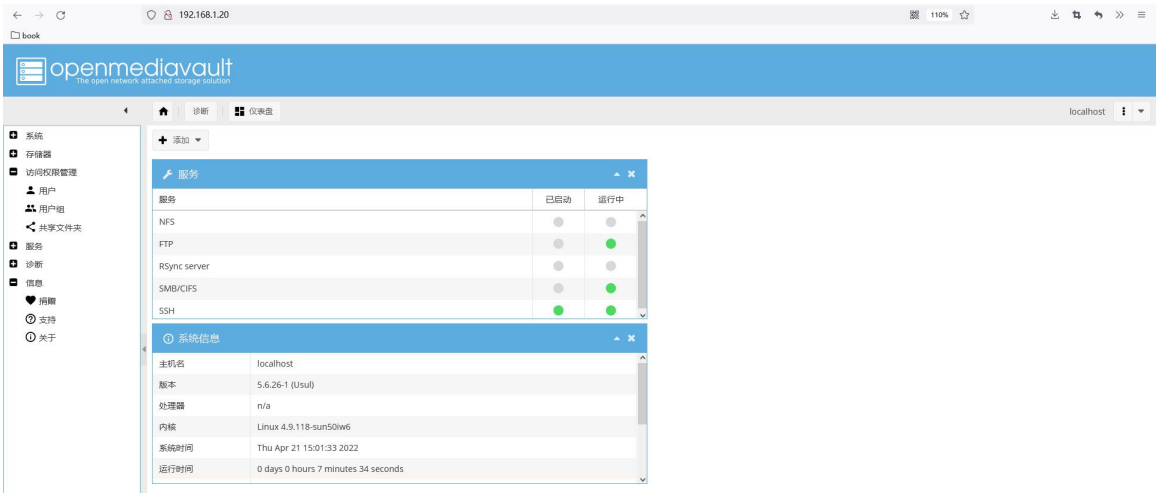
```
root@orangepi:~# omv-confdbadm populate
```

7) OpenMediaVault 的登录界面如下所示



8) 然后输入默认的用户名 **admin** 和密码 **openmediavault**

9) OpenMediaVault 登录进去显示的主界面如下所示



3. 42. 2. Debian11 安装 OpenMediaVault 6.x

注意，Debian11 只能安装 OpenMediaVault 6.x。

1) OpenMediaVault 官方文档如下所示：

https://openmediavault.readthedocs.io/en/latest/

2) 在 Debian 中安装 OpenMediaVault 的官方文档如下所示：

https://openmediavault.readthedocs.io/en/latest/installation/on_debian.html

3) 首先安装 OpenMediaVault 的 keyring，注意下面的命令都是在 **root** 用户下执行的

```
root@orangepi:~# apt-get install -y gnupg
root@orangepi:~# wget -O    \
"/etc/apt/trusted.gpg.d/openmediavault-archive-keyring.asc"    \
https://packages.openmediavault.org/public/archive.key
root@orangepi:~# apt-key add    \
"/etc/apt/trusted.gpg.d/openmediavault-archive-keyring.asc"
```

4) 然后添加 OpenMediaVault 的包仓库，注意切换到 **root** 用户再输入下面的命令，黑色字体部分是一条完整的命令，请直接复制

```
root@orangepi:~# cat <<EOF >> /etc/apt/sources.list.d/openmediavault.list
deb https://mirrors.tuna.tsinghua.edu.cn/OpenMediaVault/public shaitan main
```

```
deb https://mirrors.tuna.tsinghua.edu.cn/OpenMediaVault/packages shaitan main
## Uncomment the following line to add software from the proposed repository.
# deb https://mirrors.tuna.tsinghua.edu.cn/OpenMediaVault/public shaitan-proposed main
# deb https://mirrors.tuna.tsinghua.edu.cn/OpenMediaVault/packages shaitan-proposed main
## This software is not part of OpenMediaVault, but is offered by third-party
## developers as a service to OpenMediaVault users.
# https://mirrors.tuna.tsinghua.edu.cn/OpenMediaVault/public shaitan partner
# deb https://mirrors.tuna.tsinghua.edu.cn/OpenMediaVault/packages shaitan partner
EOF
```

上面使用的是清华源，相关说明可以参考下面的链接

```
https://mirrors.tuna.tsinghua.edu.cn/help/openmediavault/
```

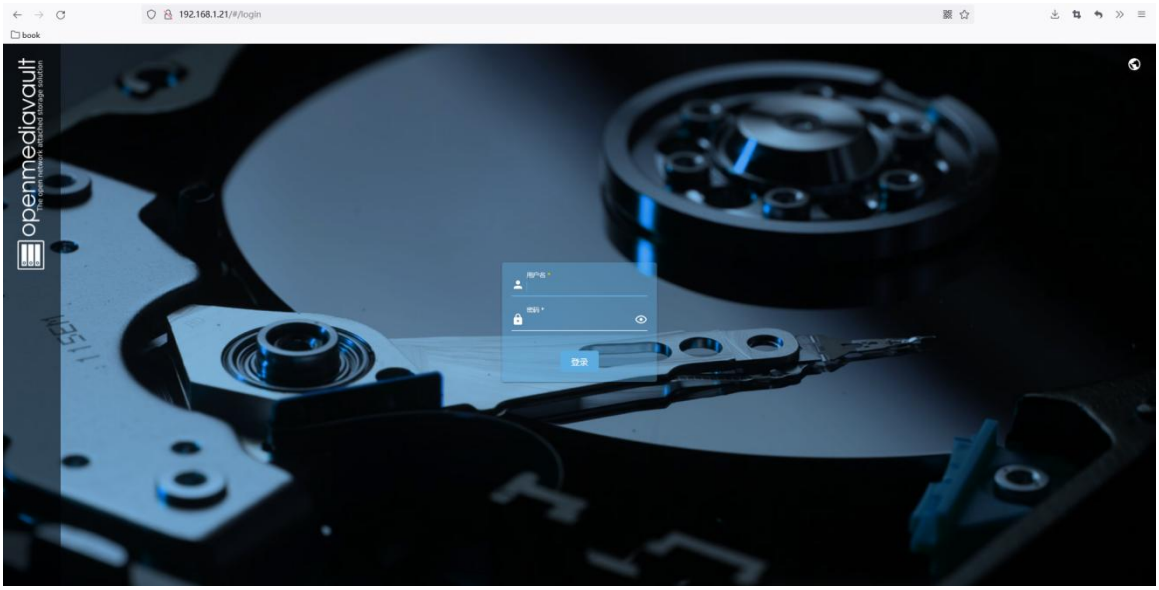
5) 然后使用下面的命令就可以安装 OpenMediaVault

```
root@orangepi:~# export LANG=C.UTF-8
root@orangepi:~# export DEBIAN_FRONTEND=noninteractive
root@orangepi:~# export APT_LISTCHANGES_FRONTEND=none
root@orangepi:~# apt-get update
root@orangepi:~# apt-get --yes --auto-remove --show-upgraded \
--allow-downgrades --allow-change-held-packages \
--no-install-recommends \
--option DPkg::Options::="--force-confdef" \
--option DPkg::Options::="--force-confold" \
install openmediavault-keyring openmediavault
```

6) 然后运行下面的命令，运行结束后，在浏览器中输入开发板的 IP 地址，即可打开 OpenMediaVault 的登录页面

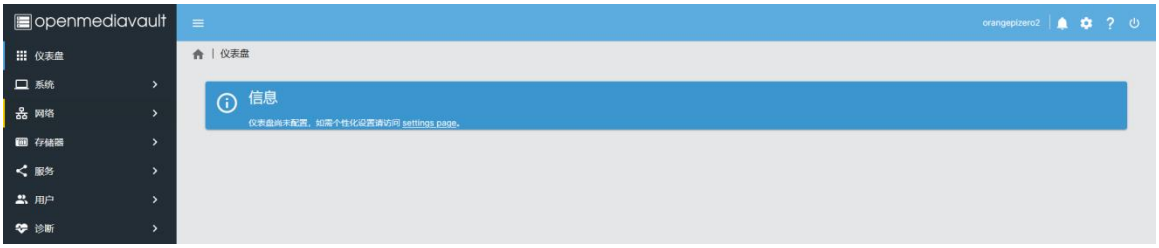
```
root@orangepi:~# omv-confdbadm populate
```

7) OpenMediaVault 的登录界面如下所示

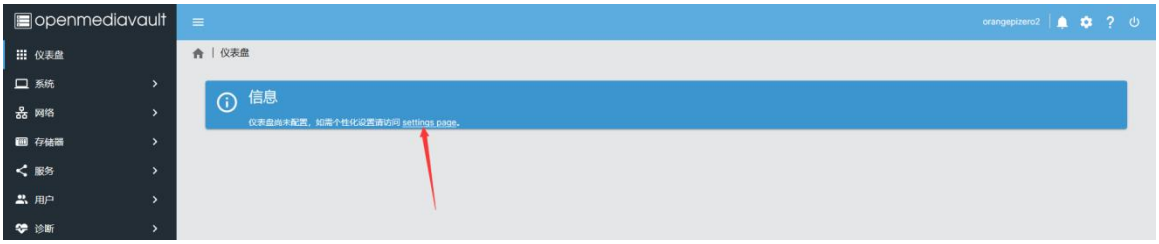


8) 然后输入默认的用户名 **admin** 和密码 **openmediavault**

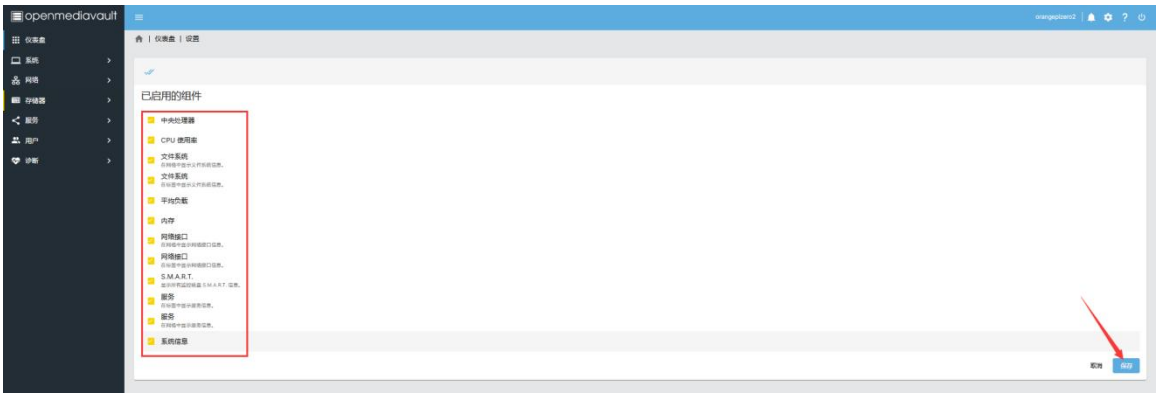
9) OpenMediaVault 登录进去显示的主界面如下所示



10) 然后点击下 **setting page**

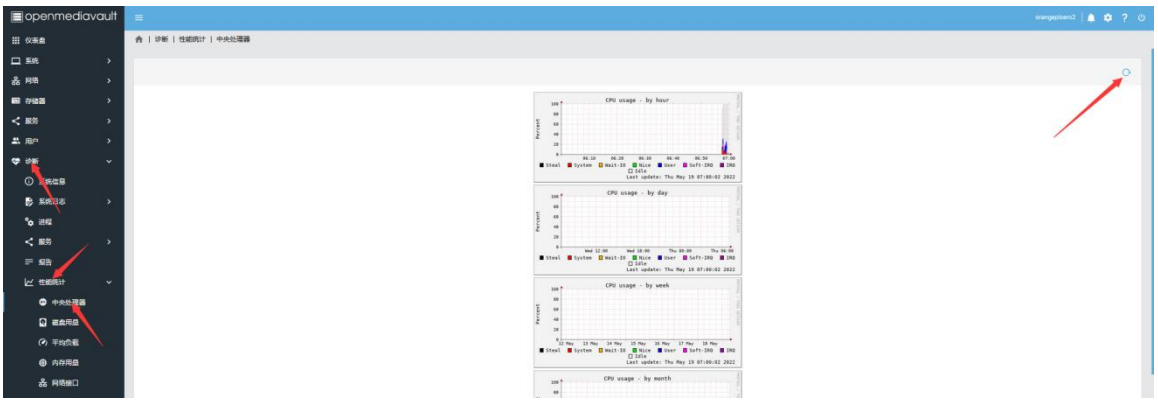


11) 然后把这些组件全部选上，再点击保存按钮保存下



12) 然后就能看到仪表盘显示的系统信息

13) 如果中央处理器显示的界面不正常，可以打开**诊断**->**性能统计**->**中央处理器**，然后点击右上角的刷新按钮刷新下



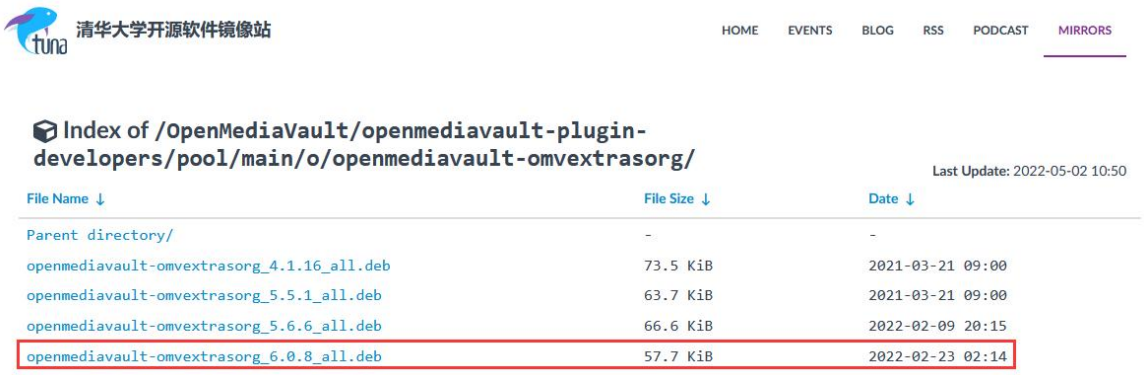
14) OMV 插件的安装方法

a. 首先打开下面的网址

https://mirrors.tuna.tsinghua.edu.cn/OpenMediaVault/openmediavault-plugin-devel

```
opers/pool/main/o/openmediavault-omvextrasorg/
```

b. 然后记录下最新的插件包的文件名



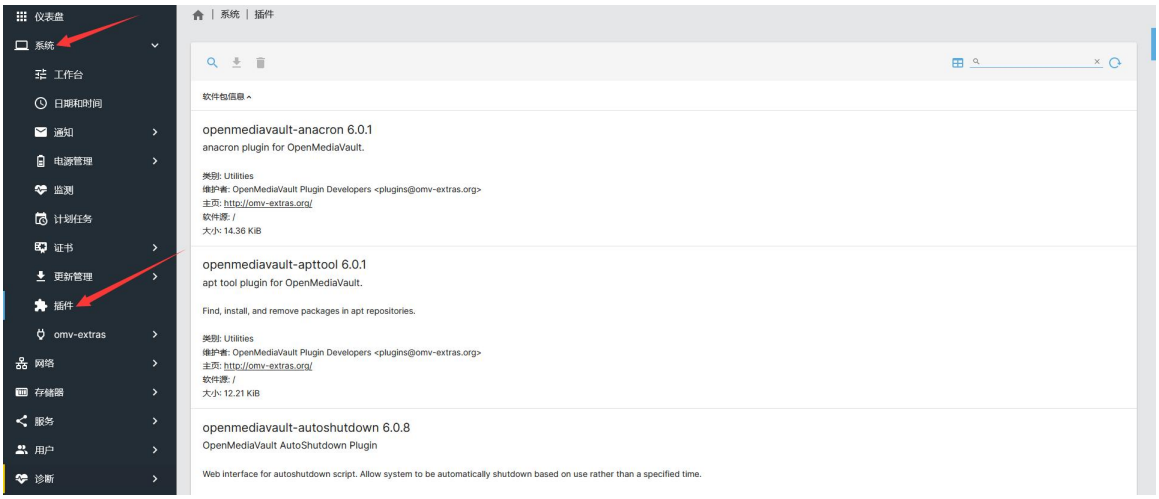
c. 然后在开发板的 Linux 系统中使用下面的命令下载上面显示的插件包（**如果下面的命令无法下载，可能是插件包的名字有所变化，请替换成最新的名字**）

```
orangepi@orangepi:~$ wget \
https://mirrors.tuna.tsinghua.edu.cn/OpenMediaVault/openmediavault-plugin-developers/pool/main/o/openmediavault-omvextrasorg/openmediavault-omvextrasorg_6.0.8_all.deb
```

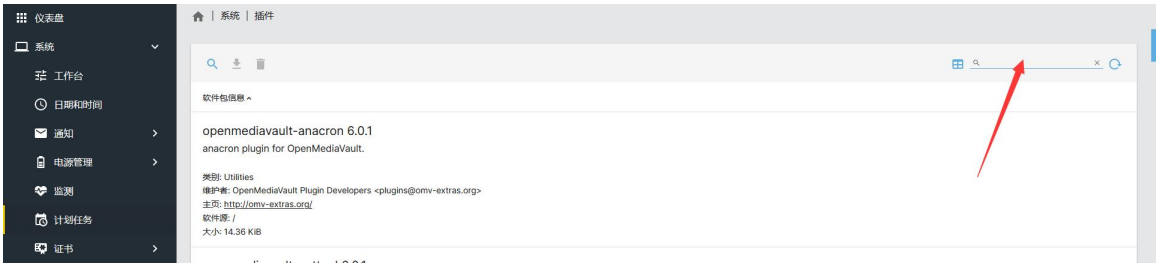
d. 然后在开发板的 Linux 系统中安装刚下载的 deb 包

```
orangepi@orangepi:~$ sudo dpkg -i openmediavault-omvextrasorg_6.0.8_all.deb
......
Summary for orangepizero2
------------
Succeeded: 6 (changed=4)
Failed:    0
------------
Total states run:     6
Total run time: 110.994 s
Processing triggers for openmediavault (6.0.27-1) ...
Updating workbench configuration files ...
Restarting engine daemon ...
```

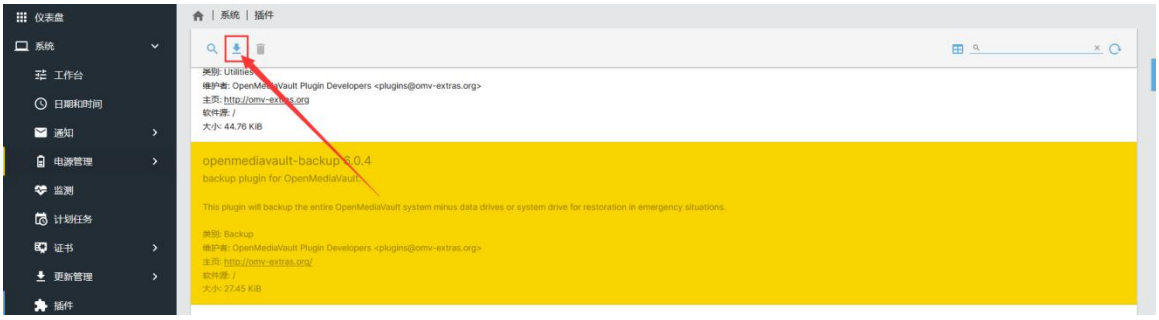
e. 然后点击**系统**->**插件**就能看到下面的界面



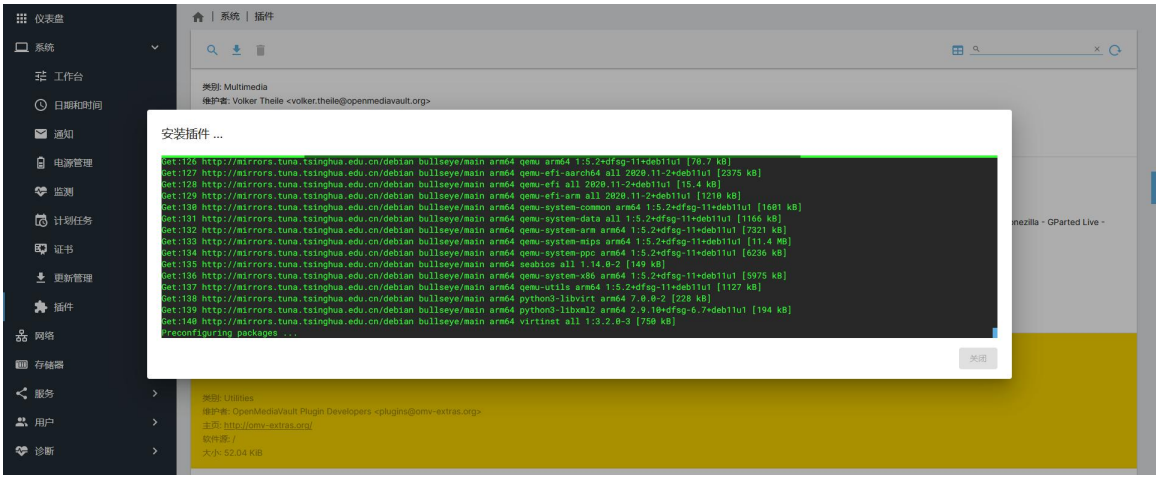
f. 然后可以选择或者搜索想要安装的插件



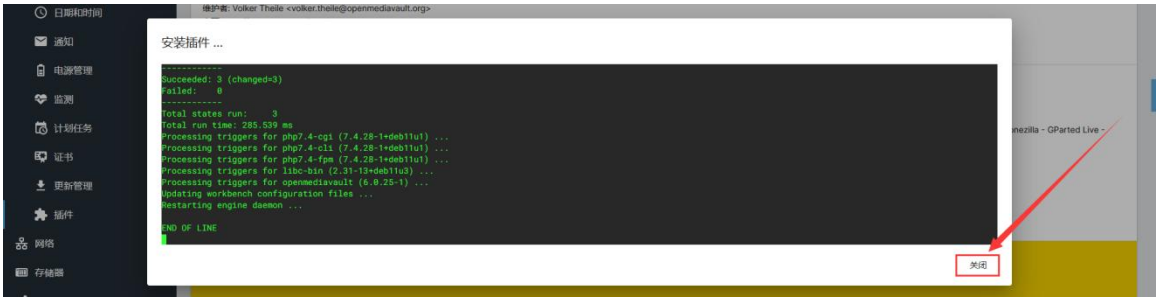
g. 然后点击下图所示的按钮就可以开始安装选中的插件



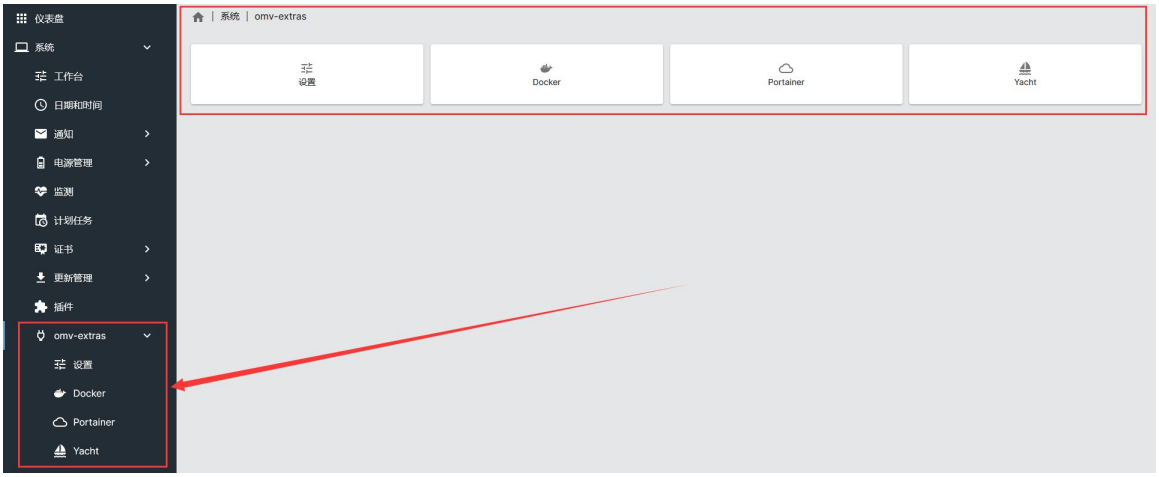
h. 插件安装过程如下所示



i. 安装完成后点击关闭即可



15) 插件包安装完成后在 web 界面的右边还会多出一个 **omv-extras** 选项，在 omv-extras 中可以安装 **Docker**、**Portainer** 和 **Yacht**



16) 安装 docker 前请先替换下 docker 的软件源，首先使用下面的命令打开 **omvextras.list**，然后将其中的内容替换为蓝色字体部分的内容。最后请记得使用 **sudo apt-get update** 更新下 Linux 系统的包索引缓存

```
orangepi@orangepi:~$ sudo vim /etc/apt/sources.list.d/omvextras.list
```

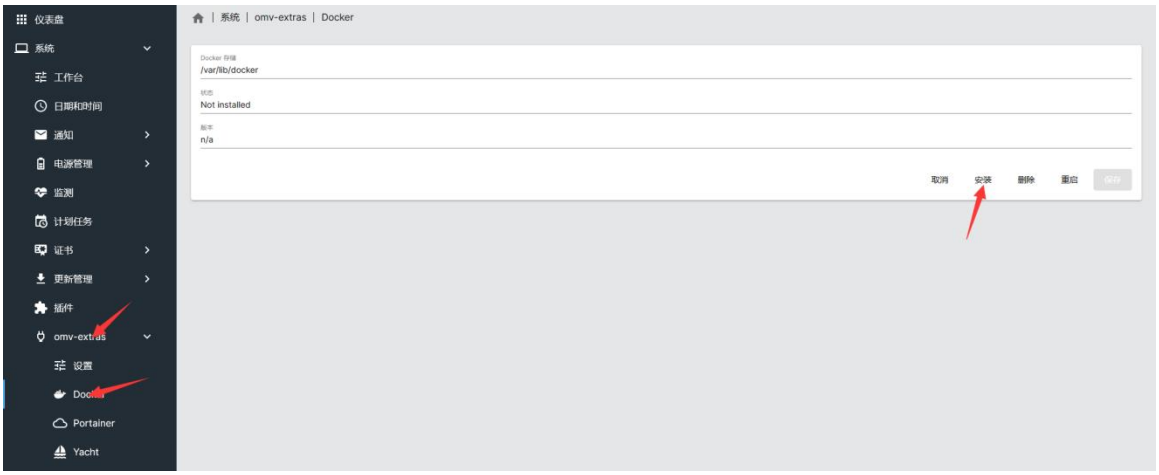
```
deb https://mirrors.tuna.tsinghua.edu.cn/OpenMediaVault/openmediavault-plugin-developers shaitan main
deb [arch=arm64] https://mirrors.tuna.tsinghua.edu.cn/docker-ce/linux/debian bullseye stable
orangepi@orangepi:~$ sudo apt-get update
```

17) 在 OMV 中安装 Docker 的方法如下所示

a. 首先安装下 **apparmor**

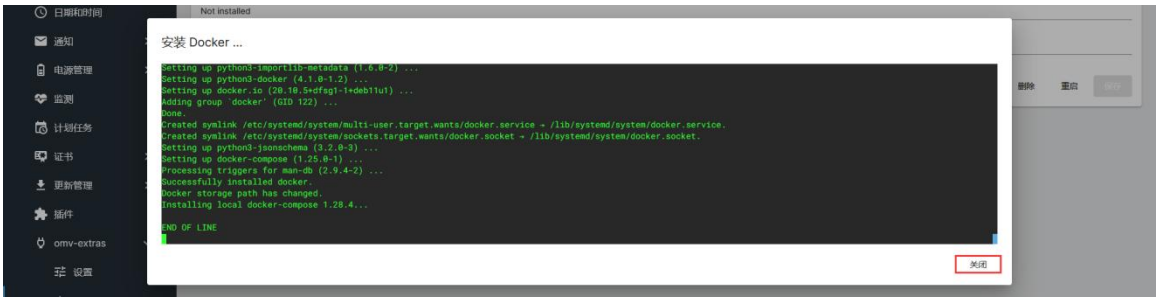
```
orangepi@orangepi:~$ sudo apt install -y apparmor
```

b. 然后打开 **Docker** 的控制界面，再点击**安装**按钮就可以开始安装 Docker



c. Docker 安装过程的显示输出如下所示

d. Docker 安装完成后的显示如下所示，然后点击**关闭**即可



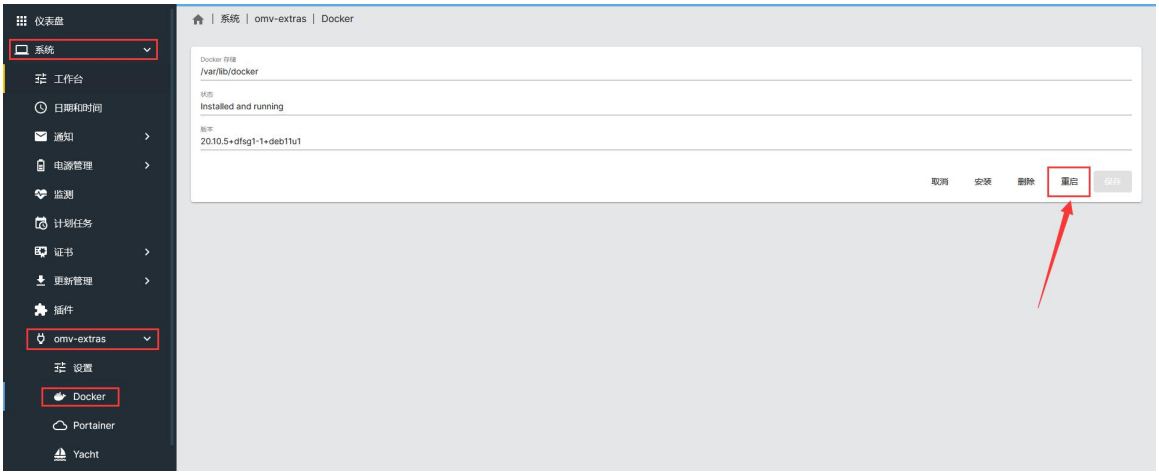
e. 最后需要设置下 Docker 仓库的地址为国内的地址，加快 Docker 容器的下载

速度。方法如下所示：

a) 首先打开**/etc/docker/daemon.json** 文件，然后在其中加入下面红色字体部分的配置（注意**"data-root": "/var/lib/docker"**后面要加一个**,**）

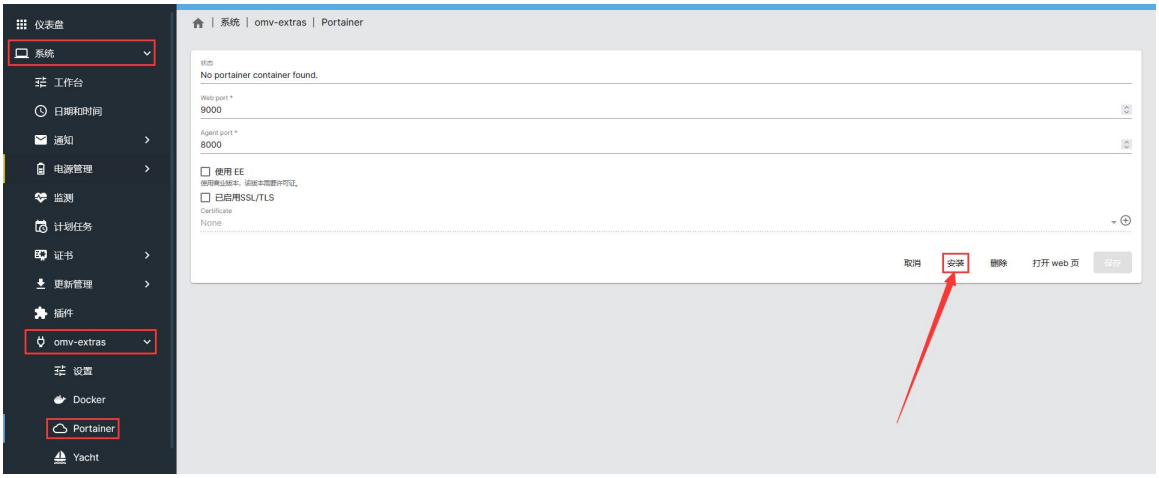
```
orangepi@orangepi:~$ sudo vim /etc/docker/daemon.json
{
  "data-root": "/var/lib/docker",
  "registry-mirrors": ["https://docker.mirrors.ustc.edu.cn"]
}
```

b) 然后在 Docker 控制界面点击下**重启**按钮来重启下 Docker 服务让配置生效（**如果有报错，首先请检查上面的配置是否正确，然后多试几次，或者重启下 Debian11 系统**）



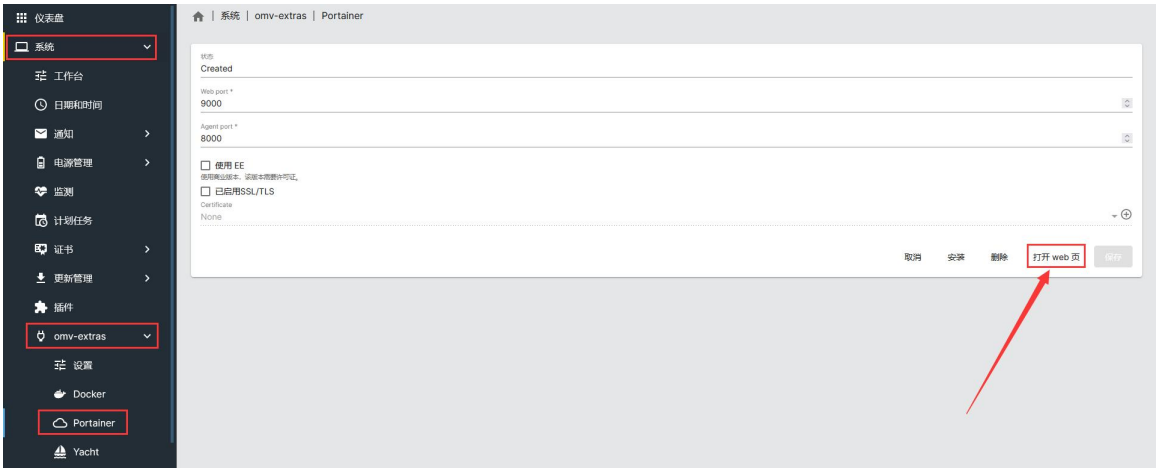
18) **Portainer** 是一个 docker 可视化管理工具，安装 Portainer 的步骤为：

a. 首先打开 Portainer 的控制界面，再点击**安装**按钮就可以开始安装 Portainer

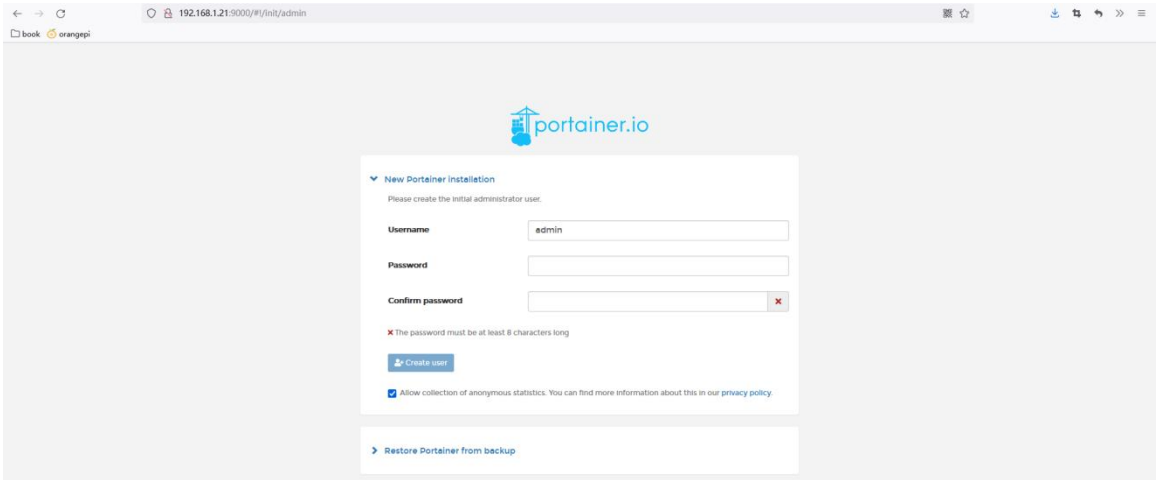


b. Portainer 安装完成后的显示如下图所示，然后点击**关闭**按钮关闭即可

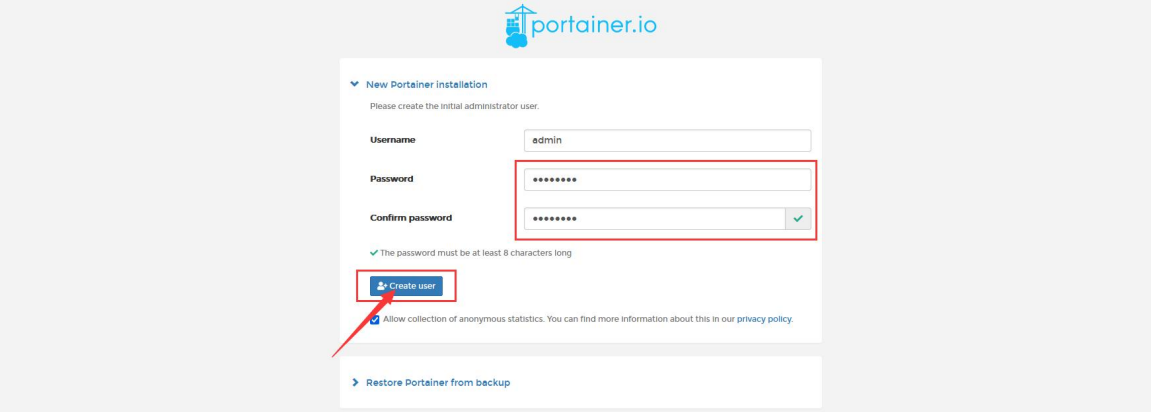
c. 然后再打开 Portainer 的控制界面，再点击**打开 web 页**就可以打开 Portainer 的 web 控制界面



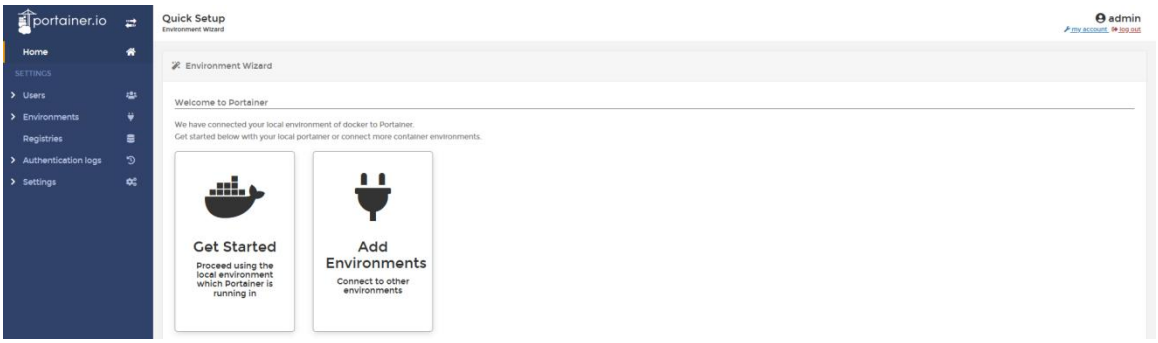
d. Portainer 的 web 控制界面打开后的显示如下所示



e. 然后设置下 Portainer 的密码，再点击 **Create user** 就可以进入 Portainer 的 web 控制界面



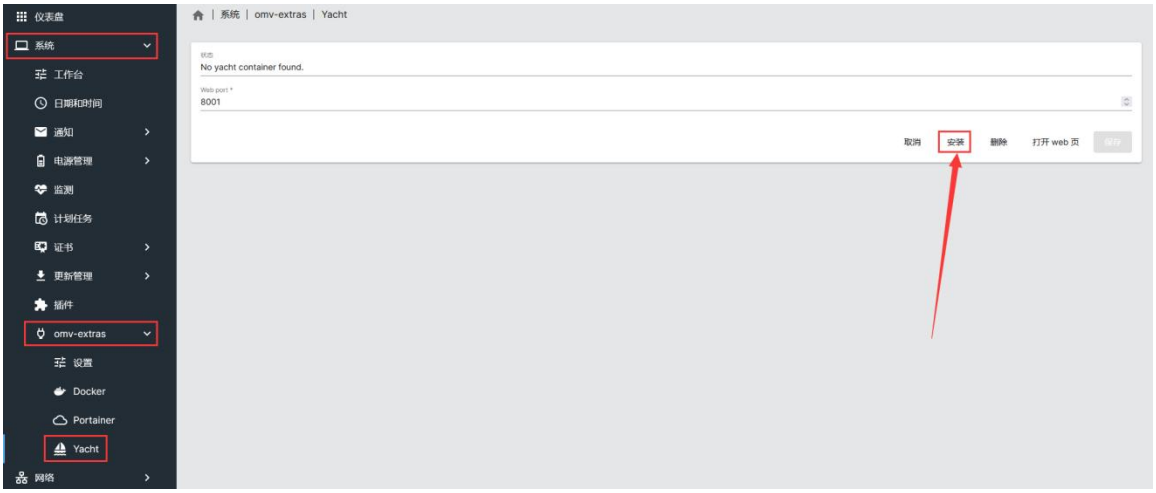
f. Portainer 登录后的主界面显示如下所示



g. Portainer 的使用方法请自行研究

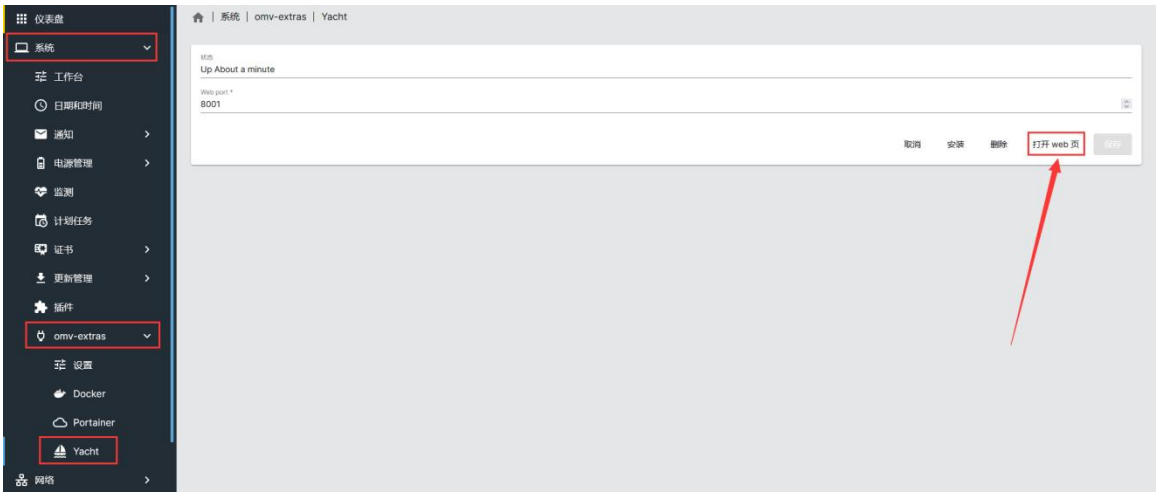
19) **Yacht** 和 Portainer 的功能类似，都是 docker 可视化管理工具，安装步骤如下所示：

a. 首先在 OMV 中打开 Yacht 的控制界面，然后点击**安装**按钮即可开始安装 Yacht

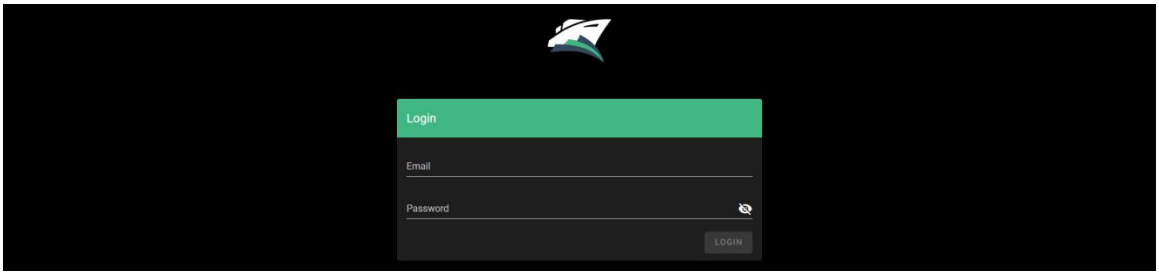


b. Yacht 安装后的显示如下图所示，然后点击**关闭**按钮关闭即可

c. 然后再在 OMV 中打开 Yacht 的控制界面，再点击**打开 web 页**就可以打开 Yacht 的 web 控制界面

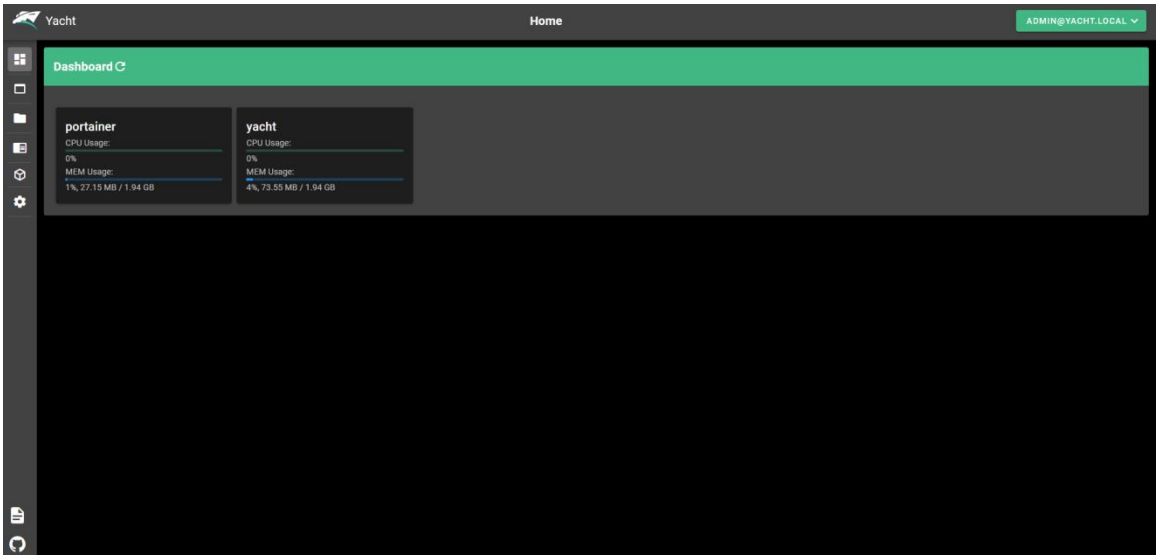


d. Yacht 的 web 控制界面打开后的显示如下所示



e. 然后在 **Email** 一栏中输入 Yacht 的默认账号 **admin@yacht.local**，并在 **Password** 一栏中输入默认密码 **pass**，再点击 **LOGIN** 就可以进入 Yacht 的 web 控制界面

f. Yacht 登录后的主界面显示如下所示



g. Yacht 的使用方法请自行研究

3.43. Pi-hole 的安装方法

注意，此小节只提供 Pi-hole 的安装方法，使用方法请参考 Pi-hole 的官方文档：

https://pi-hole.net
https://docs.pi-hole.net

注意，Pi-hole 还不支持 Ubuntu22.04，所以请不要使用这个系统来安装 Pi-hole。

1) 首先下载 Pi-hole 的安装脚本，然后运行

a. 使用 Orange Pi 提供 pi-hole 脚本安装（推荐）

注意，Orange Pi 提供的 pi-hole 安装脚本并没有修改任何 pi-hole 的功能，只是解决了由于无法正常访问 Github 导致的下载安装失败的问题。

a) 第一种方法：先下载 pi-hole 的仓库，然后运行安装脚本

```
orangepi@orangepi:~$ git clone --depth 1 https://gitee.com/leeboby/pi-hole.git   \
Pi-hole
orangepi@orangepi:~$ cd Pi-hole/automated\ install/
orangepi@orangepi:~$ sudo bash basic-install.sh
```

b) 第二种方法：先下载安装脚本，然后运行

```
orangepi@orangepi:~$ wget -O basic-install.sh   \
```

```
https://gitee.com/leeboby/pi-hole/raw/master/automated%20install/basic-install.sh
orangepi@orangepi:~$ sudo bash basic-install.sh
```

b. 使用 pi-hole 官方脚本安装（如果不会解决网络问题就不推荐）

a) 第一种方法：使用下面的命令可以下载 pi-hole 的安装脚本然后直接运行

```
orangepi@orangepi:~$ curl -sSL https://install.pi-hole.net | bash
```

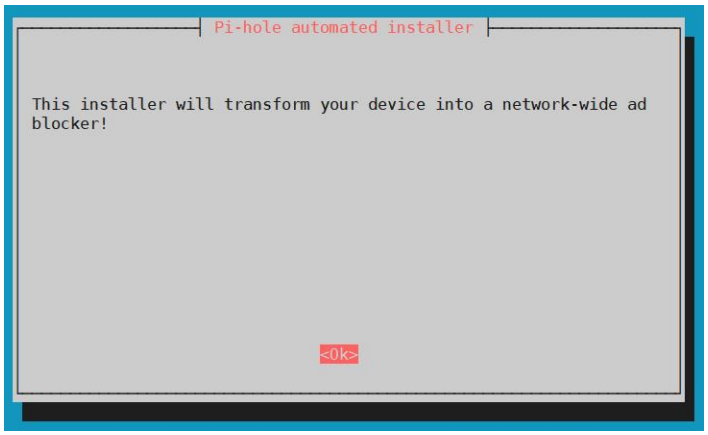
b) 第二种方法：先下载 pi-hole 的仓库，然后运行安装脚本

```
orangepi@orangepi:~$ git clone --depth 1 https://github.com/pi-hole/pi-hole.git  \
Pi-hole
orangepi@orangepi:~$ cd Pi-hole/automated\ install/
orangepi@orangepi:~$ sudo bash basic-install.sh
```

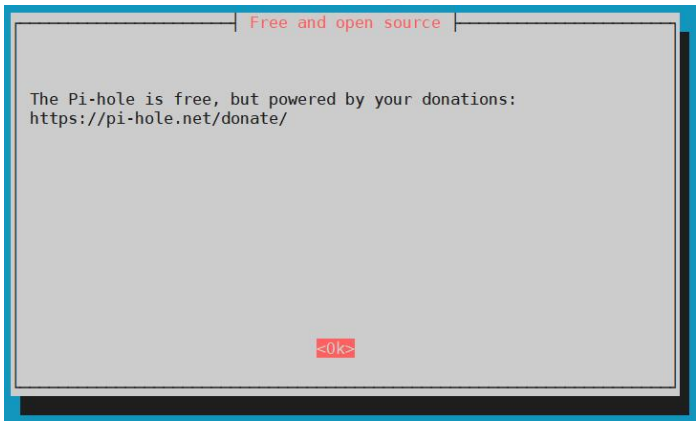
c) 第三种方法：先下载安装脚本，然后运行

```
orangepi@orangepi:~$ wget -O basic-install.sh https://install.pi-hole.net
orangepi@orangepi:~$ sudo bash basic-install.sh
```

2) 然后回车

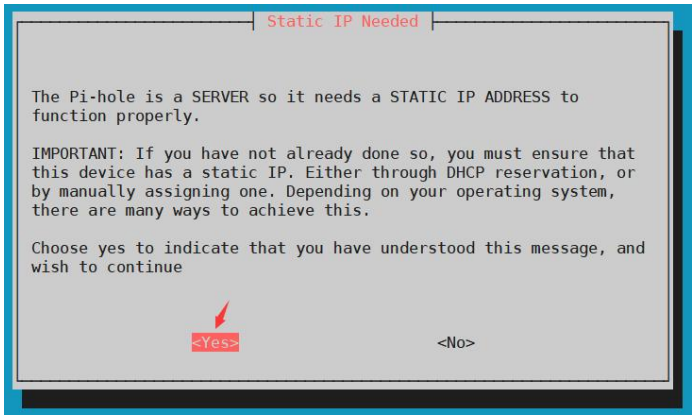


3) 继续回车



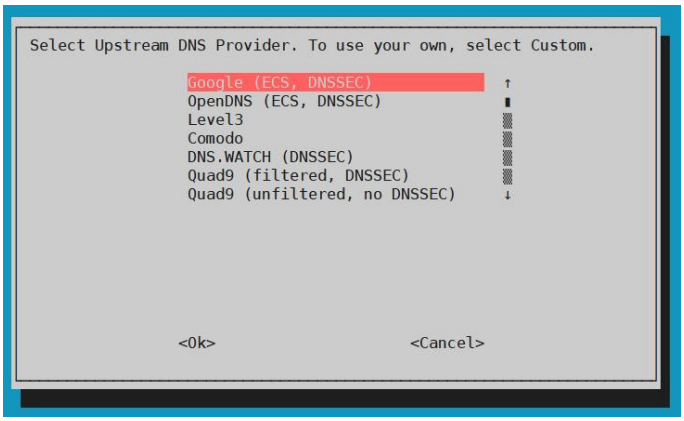
4) 然后 Pi-hole 会提示需要设置一个静态 IP 地址，请先选择 **Yes** 再回车

这里可以先不设置静态 IP 地址，等安装完后再设置，静态 IP 地址的设置方法可以参考设置静态 IP 地址的方法一节的说明。

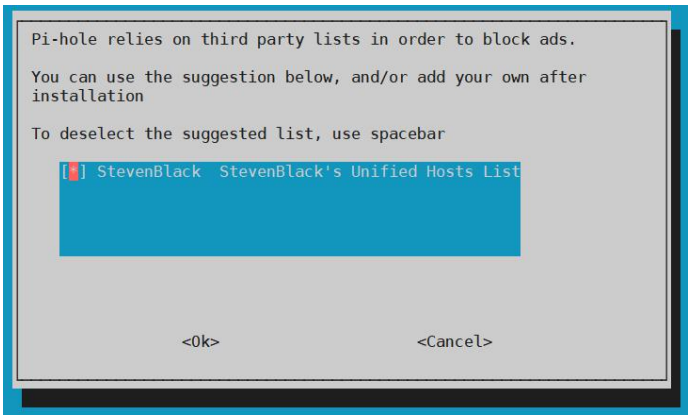


5) 然后选择网络接口，如果用的网线就选择 eth0，如果想用 WIFI 就选择 wlan0，选择完后回车即可

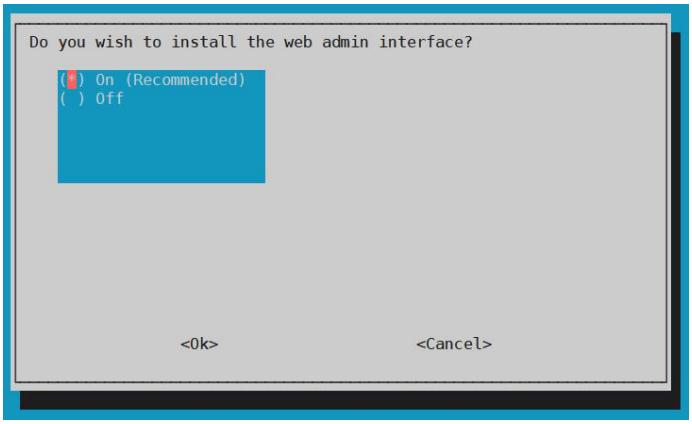
6) 然后选择 DNS 提供商，一般选择 Google 就可以了



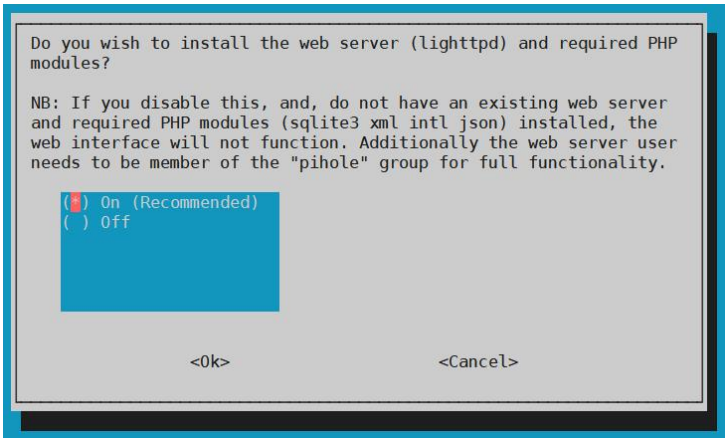
7) 然后选择规则列表，直接回车即可



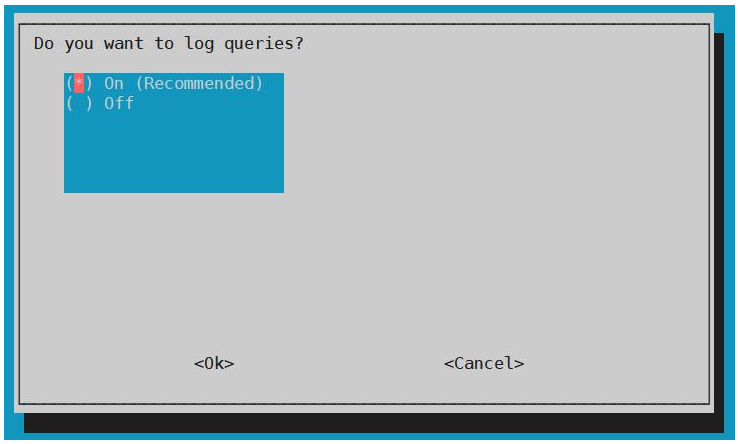
8) 然后选择是否安装 web 管理接口，直接回车选择安装



9) 然后回车选择安装 web 服务器和需要的 PHP 模块

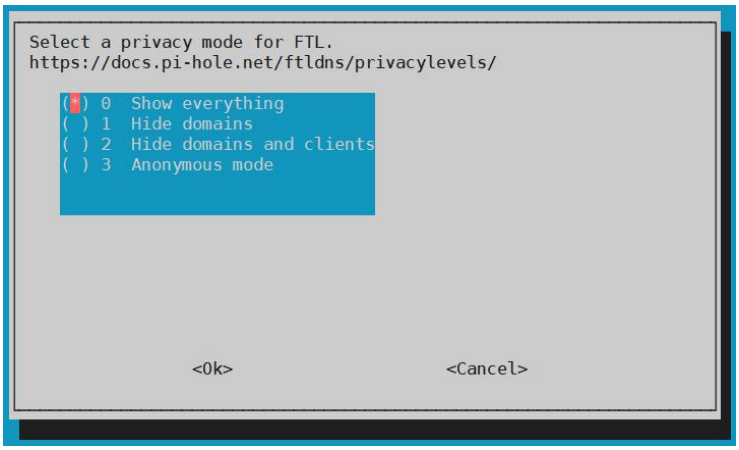


10) 然后回车选择开启日志

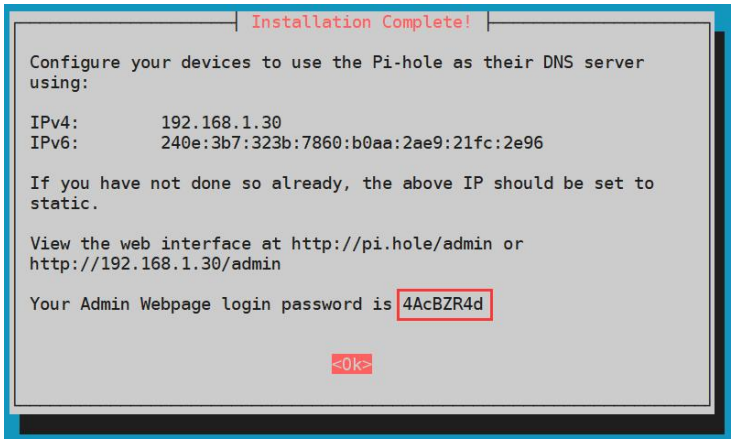


11) 然后选择隐私模式，安装时选择默认的 **0 Show everything** 就可以了，反正安装完成后都可以修改，隐私模式不同选项的区别的请查看下面链接：

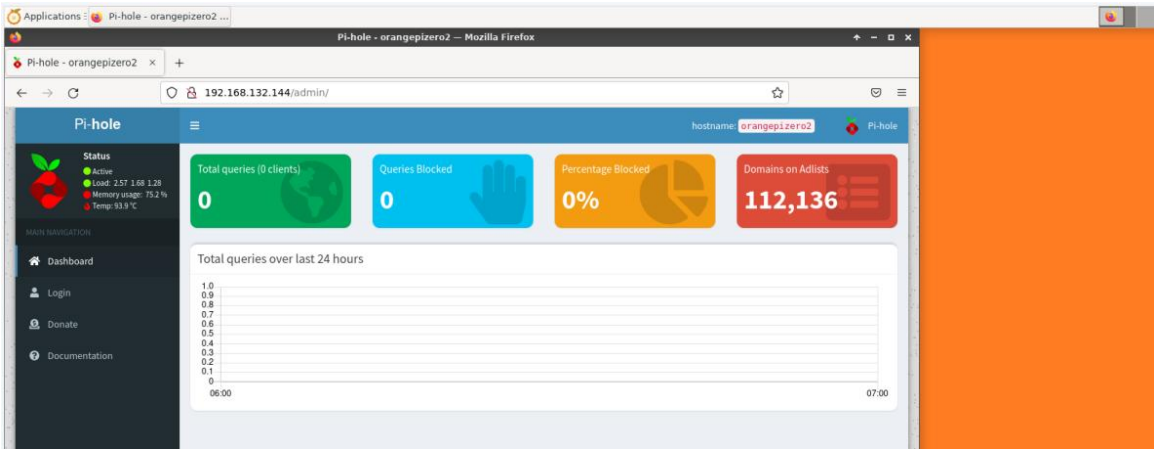
https://docs.pi-hole.net/ftldns/privacylevels/



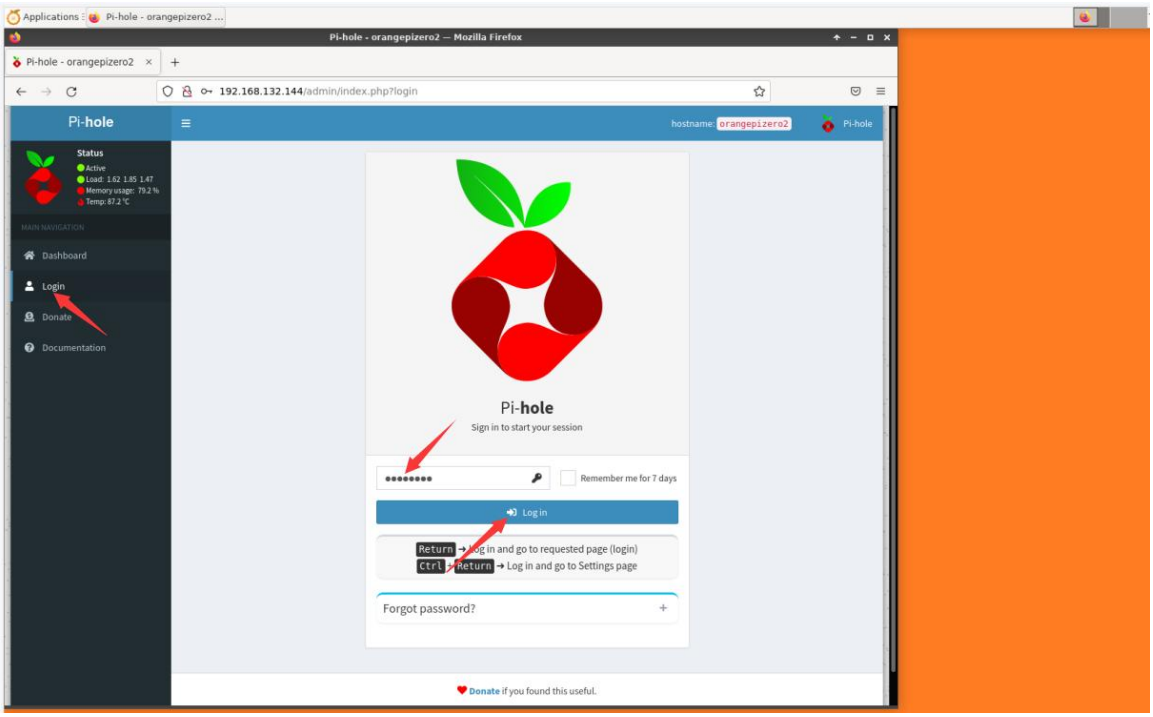
12) 然后等待 pi-hole 安装完成，最后会提示下面的信息，重点需要记住 web 界面登录的地址和登录密码



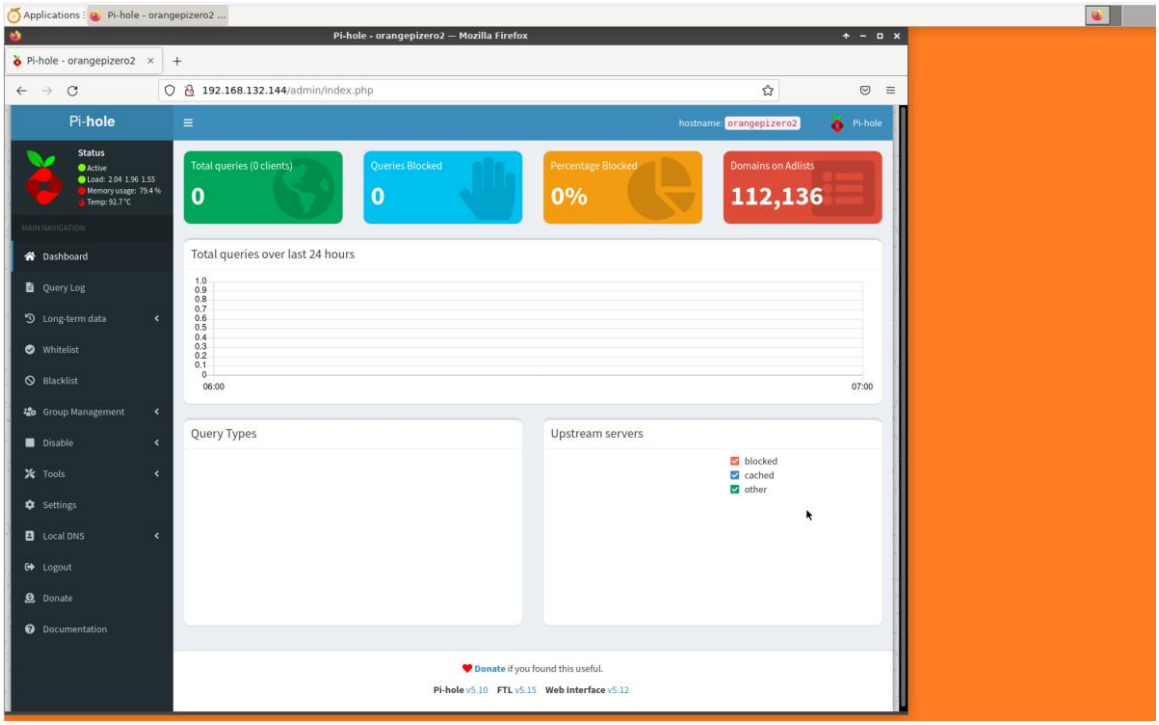
13) 然后在浏览器中输入**开发板的 IP 地址/admin** 就能看到 pi-hole 的 web 管理界面



14) 点击左边的 **Login**，然后输入上面显示的密码再点击下面的 **Log in** 就能登录进 pi-hole



15) 登录后的界面显示如下所示，可以看到比没登陆前多了很多选项



3. 44. GotoHTTP 使用介绍

1) GotoHTTP 可以用来在浏览器中远程控制开发板，安装比较简单，并且支持多种平台的设备，只要有浏览器可以上网，就能远程控制开发板。GotoHTTP 官网文档的链接如下所示，使用前请先仔细阅读一下

https://gotohttp.com/goto/help.12x

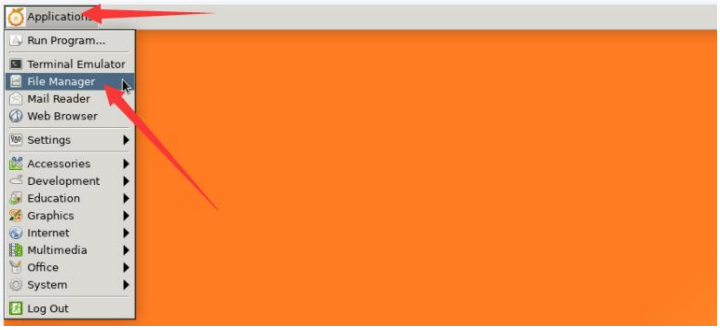
2) GotoHTTP 有提供图形界面和字符界面两种版本的压缩包。如果想使用图形界面，首先请确保开发板使用的 Linux 系统为桌面版系统

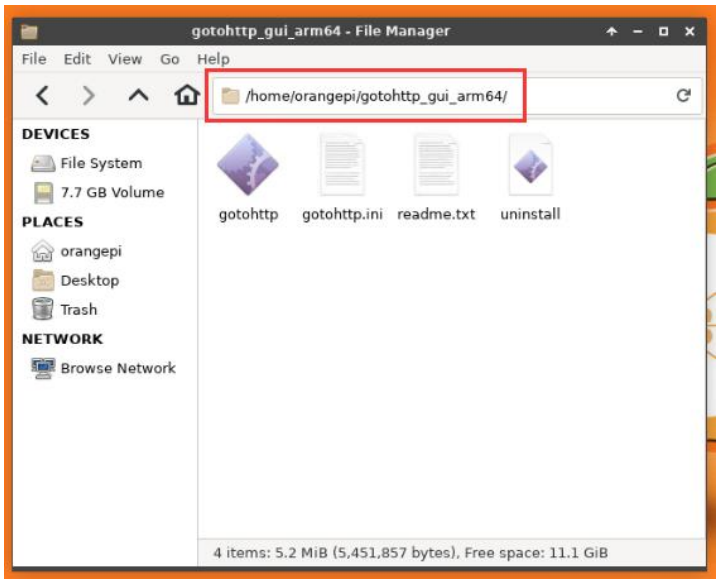
3) 图形界面版本的 GotoHTTP 下载运行命令如下所示

a. 首先下载解压 **gotohttp_gui_arm64.tar.gz**

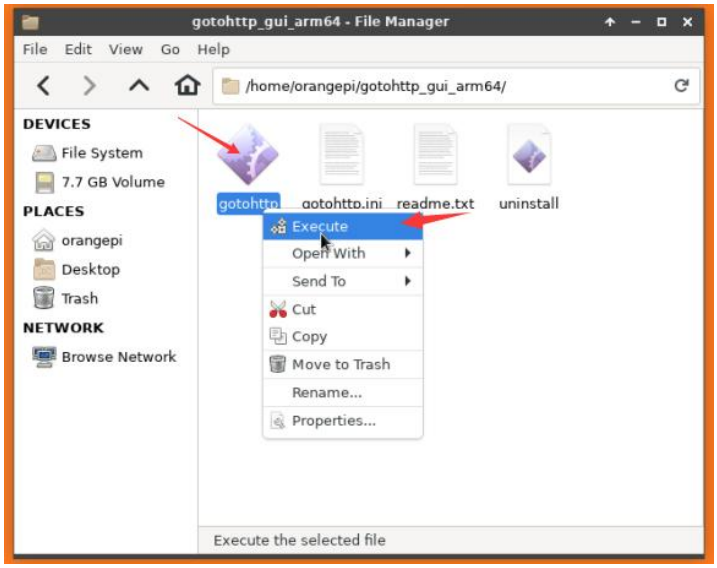
```
orangepi@orangepi:~$ wget http://gotohttp.com/gotohttp_gui_arm64.tar.gz
orangepi@orangepi:~$ tar -xvf gotohttp_gui_arm64.tar.gz
```

b. 然后在桌面中打开 **gotohttp_gui_arm64** 所在的文件夹





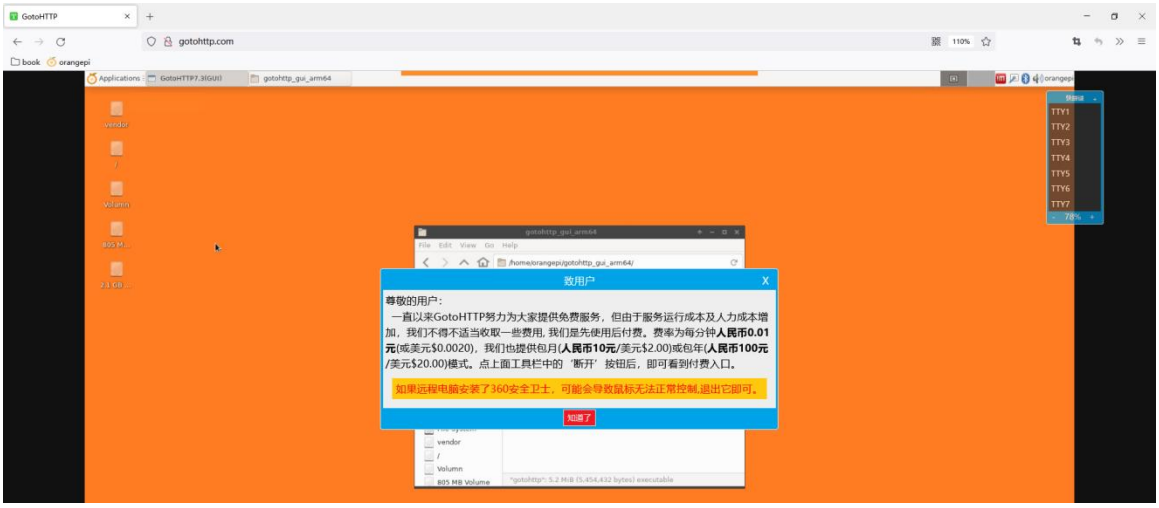
c. 然后选择 gotohttp 可执行程序，再点击右键，然后选择 **Execute** 打开 gotohttp



d. gotohttp 打开后的界面如下所示，重点需要记住 **Computer ID** 和 **Access Code**

e. 然后打开电脑或者手机的浏览器，在地址栏输入 **http://gotohttp.com**，然后在右边**电脑 ID** 一栏中输入上面显示的 **Computer Id**，在**控制码**中输入上面显示的 **Access Code**，最后点击开始控制

f. 然后在浏览器中就能看到开发板 Linux 系统的桌面了。gotohttp 打开后会弹出付费的提示，**使用完后是需要付费的**



4) 字符界面版本的 GotoHTTP 下载运行的命令如下所示

a. 首先下载解压 **gotohttp_cli_arm64.tar.gz**

```
orangepi@orangepi:~$ wget http://gotohttp.com/gotohttp_cli_arm64.tar.gz
orangepi@orangepi:~$ tar -xvf gotohttp_cli_arm64.tar.gz
```

b. 然后使用下面的命令就可以运行 GotoHTTP

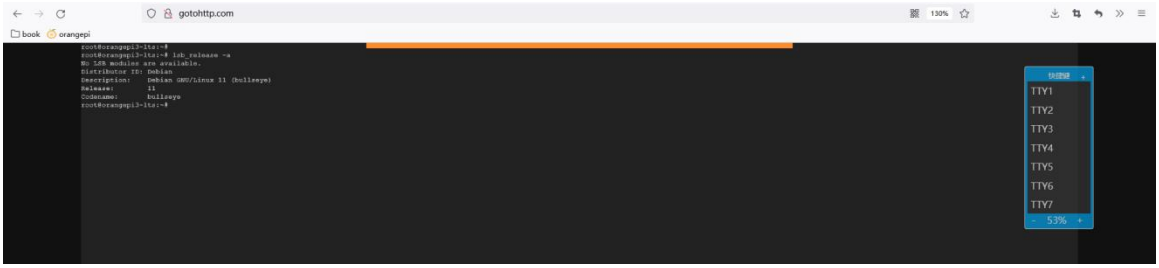
```
orangepi@orangepi:~$ sudo ./gotohttp_cli_arm64/gotohttp_cli
```

c. 运行 GotoHTTP 后如果可以看到下面的输出，说明启动正常。另外可以看到下面有输出 **Computer Id** 和 **Access Code**，请记住它们，后面会用到

```
Starting GotoHTTP...
orangepi@orangepi:~$ Registering to server.....
Connecting to server.....
Connected.(secure connection)

File Transfer: ON,   Try Framebuffer: OFF

This computer is ready for controlling remotely.
To control it, please open the URL http://gotohttp.com in web browser, then input
Computer Id: 137133946 and Access Code: 6103
```

d. 然后打开电脑或者手机的浏览器，在地址栏输入 **http://gotohttp.com**，然后在**电脑 ID** 一栏中输入上面显示的 **Computer Id**，在**控制码**中输入上面显示的 **Access Code**，最后点击开始控制

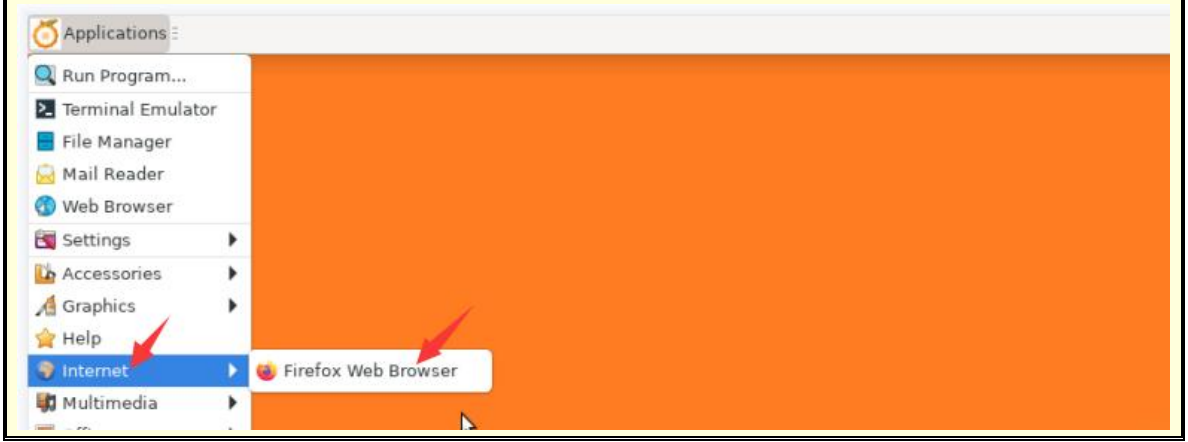
e. 然后在浏览器中就能看到开发板 Linux 系统的命令行输入界面了



3.45. Ubuntu22.04 安装浏览器的方法

3.45.1. Ubuntu22.04 火狐浏览器的安装方法

如果使用的是最新版本的 Ubuntu22.04 桌面版镜像，那么就已经预装了 deb 版本的火狐浏览器。安装前可以请先查看下面的位置是否已经有了火狐浏览器，如果有了，就不需要再安装了，直接使用即可。



1) Ubuntu22.04 默认只支持 snap 版本的火狐浏览器，安装命令为：

```
orangepi@orangepi:~$ sudo apt install firefox
```

2) firefox-esr 版本浏览器可以通过下面的方法来安装：

a. 首先添加 firefox-esr 版本 PPA 源

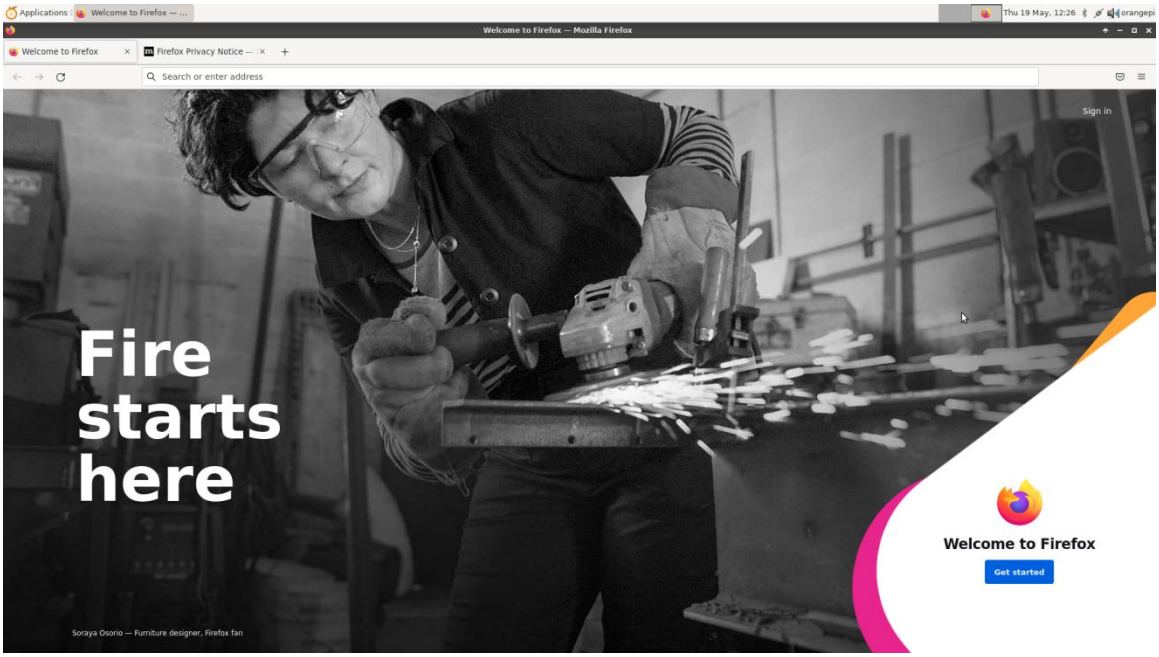
```
orangepi@orangepi:~$ sudo add-apt-repository ppa:mozillateam/ppa
PPA publishes dbgsym, you may need to include 'main/debug' component
Repository: 'deb https://ppa.launchpadcontent.net/mozillateam/ppa/ubuntu/ jammy main'
Description:
Mozilla Team's Firefox 91 ESR and Thunderbird 91 or 78 stable builds
More info: https://launchpad.net/~mozillateam/+archive/ubuntu/ppa
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.      <---  这里请回车确认
```

b. 然后使用下面的命令安装 firefox-esr

```
orangepi@orangepi:~$ sudo apt install -y firefox-esr
```

c. 安装完后在应用程序里面就可以看到 firefox-esr 浏览器的快捷方式

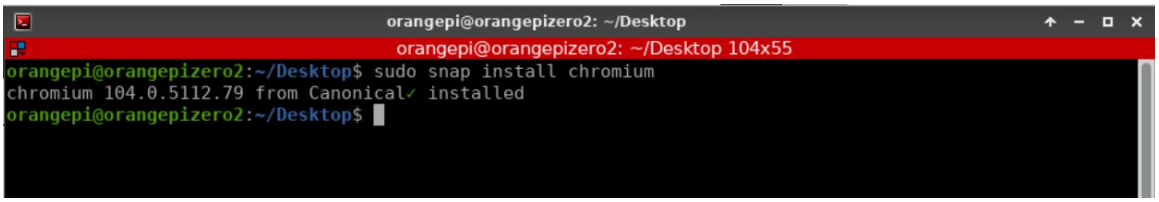
d. 打开 firefox-esr 浏览器后的显示如下所示



3.45.2. Ubuntu22.04 Chromium 浏览器的安装方法

1) 安装 Chromium 浏览器的命令如下所示

```
orangepi@orangepi:~$ sudo snap install chromium
```



3.46. GPU 测试说明

3.46.1. Ubuntu22.04 Linux5.16 系统 GPU 测试说明

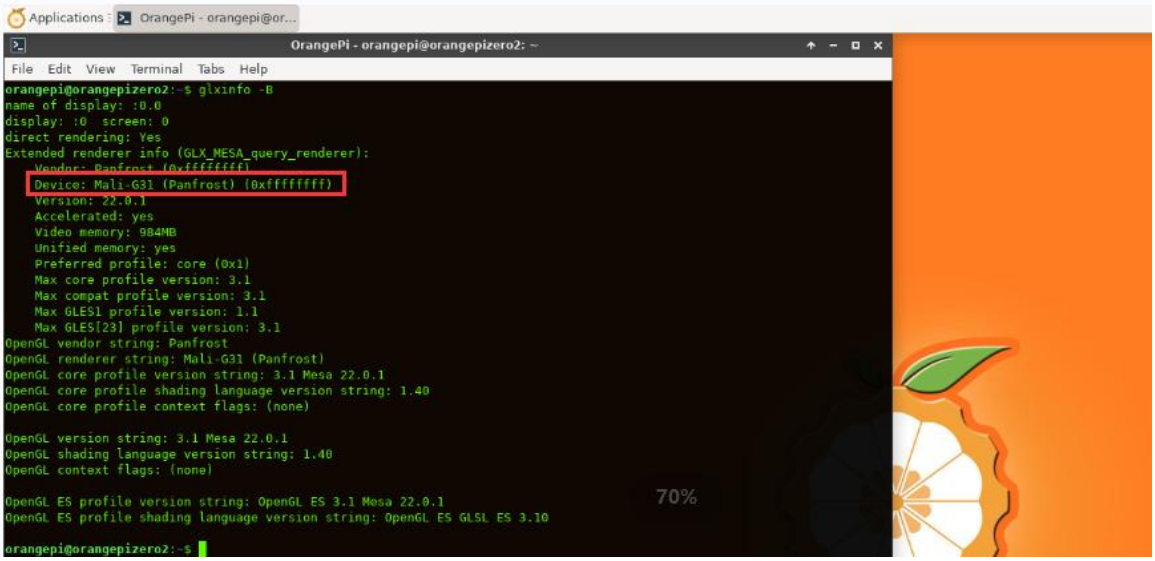
注意，下面的测试都是在桌面版本的系统中测试的，所以请确保开发板所使用的系统为桌面版本的系统。

1) 首先请连接好 HDMI 显示器，下面所有的命令都是在 HDMI 显示的桌面中操作的，请不要使用 ssh 远程登录或者使用串口来登录 Linux 系统

2) 进入桌面后，先打开一个终端，然后使用 **glxinfo -B** 命令可以看到使用的 GPU

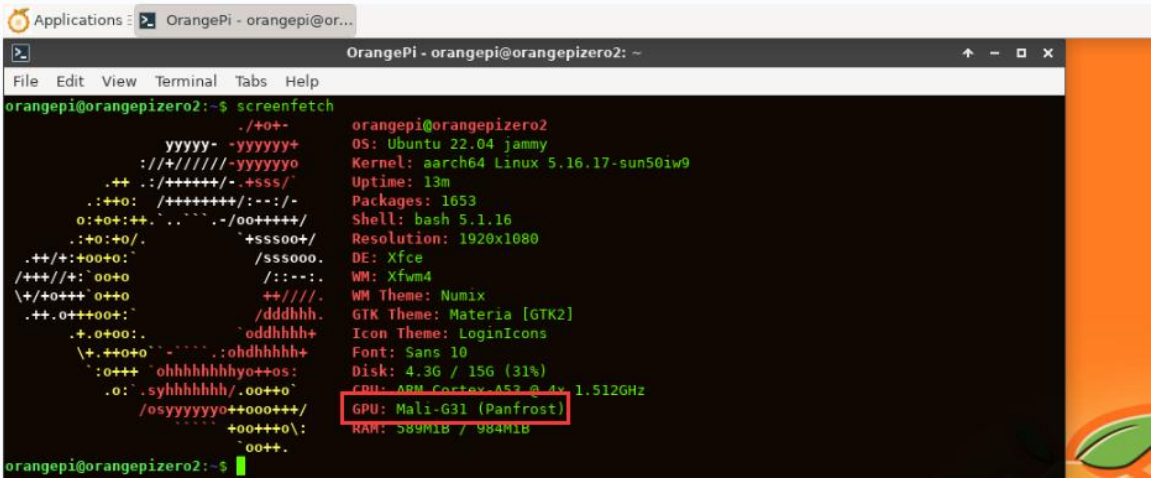
驱动为 **Mali-G31 (Panfrost)**而非 **llvmpipe**

```
orangepi@orangepi:~$ glxinfo -B
```



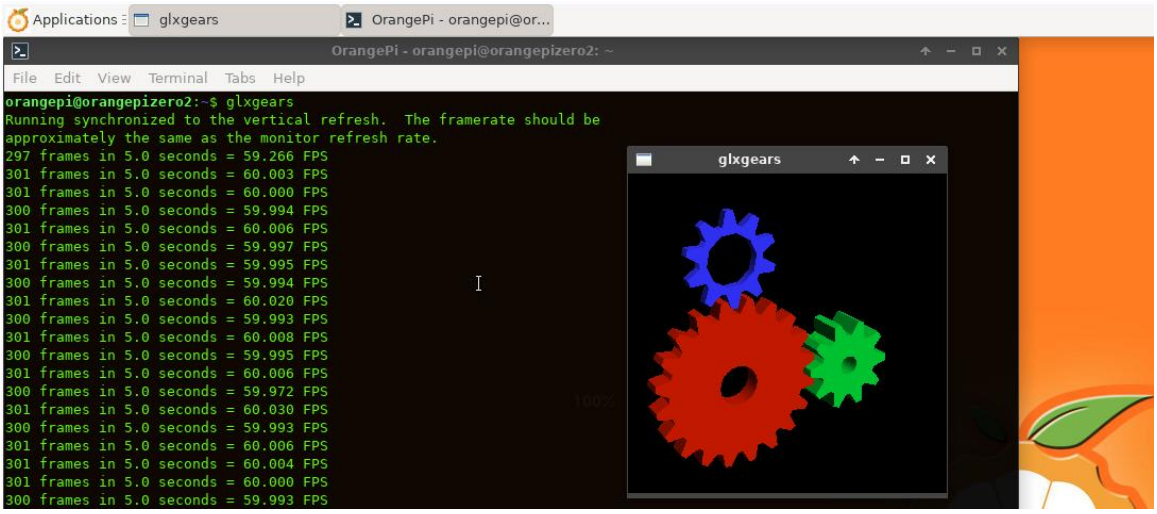
3) 使用 **screenfetch** 命令也可以看到 GPU 驱动使用的是 **Mali-G31 (Panfrost)**

```
orangepi@orangepi:~$ sudo apt install -y screenfetch
orangepi@orangepi:~$ screenfetch
```



4) **Glx-Gears** 测试如下所示

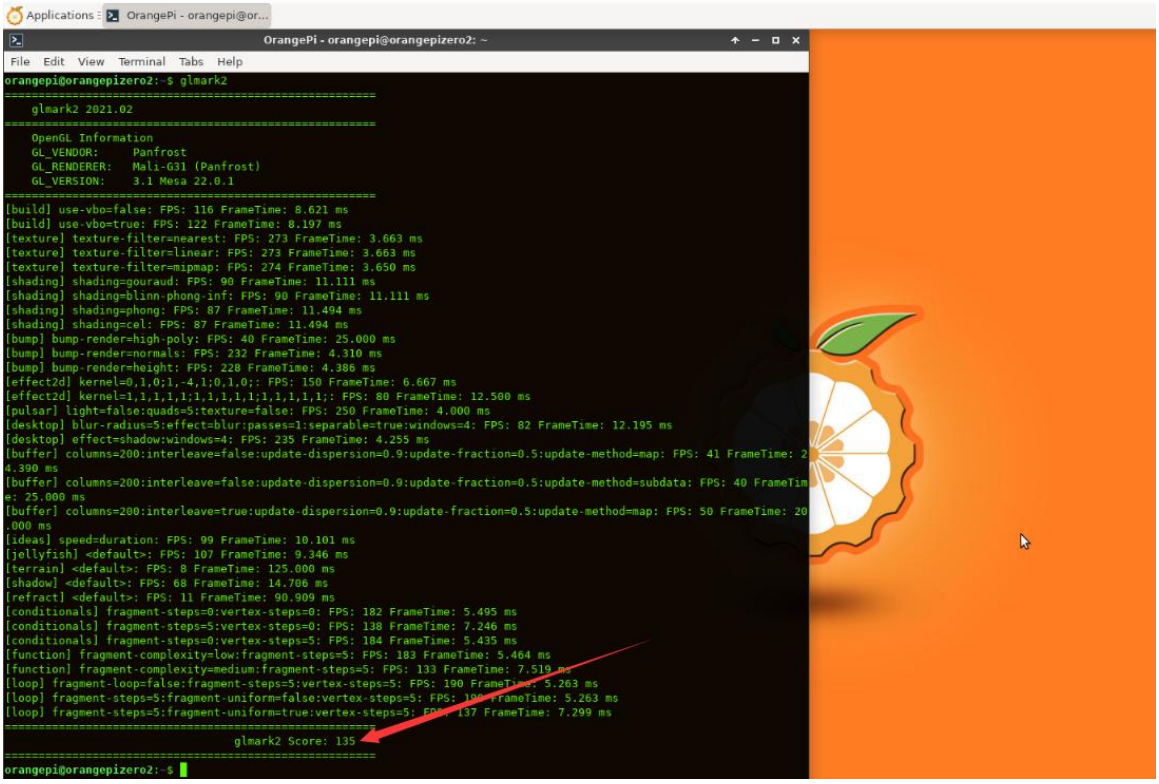
```
orangepi@orangepi:~$ glxgears
```



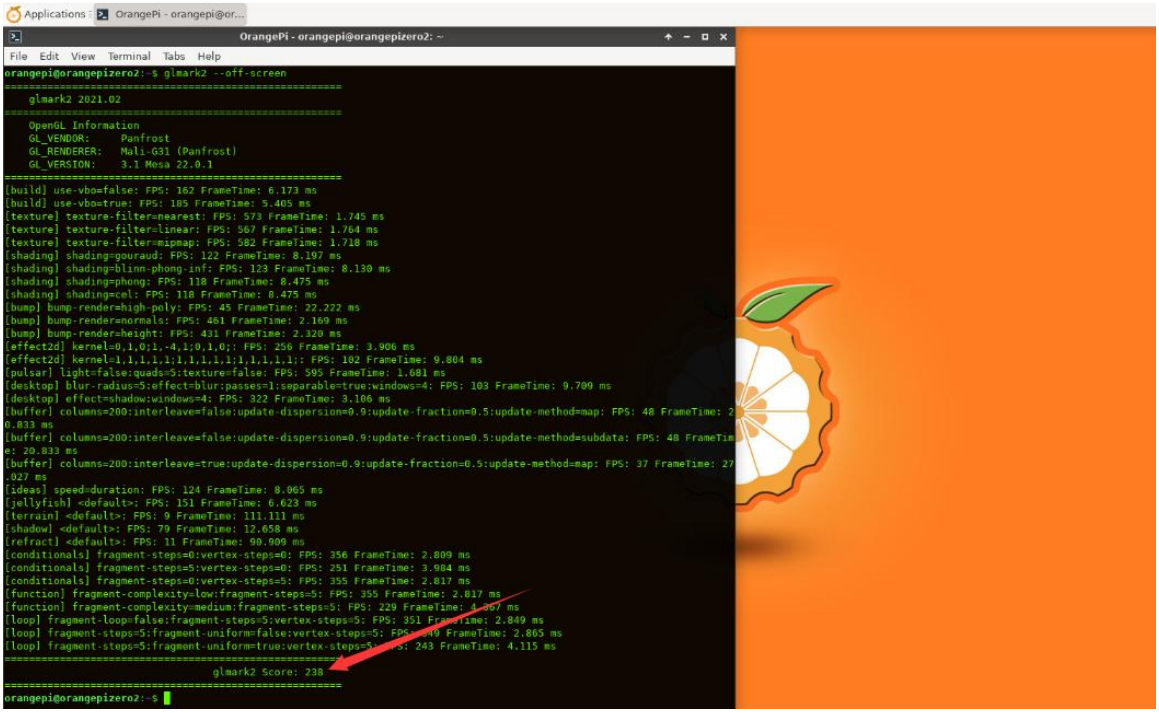
5) **glmark2-es2** 是 OpenGL (ES) 2.0 的基准测试工具，使用 glmark2 可以测试下 GPU OpenGL ES 2.0 的性能

```
orangepi@orangepi:~$ sudo apt install -y glmark2-es2
orangepi@orangepi:~$ glmark2-es2
orangepi@orangepi:~$ glmark2-es2 --off-screen
```

a. **glmark2-es2** 命令的测试分数如下所示

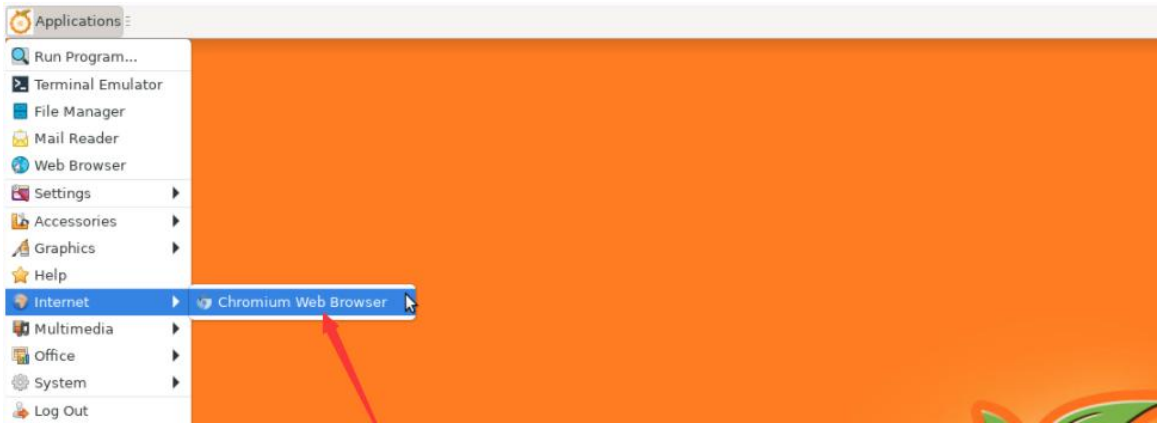


b. **glmark2-es2 --off-screen** 命令的测试分数如下所示

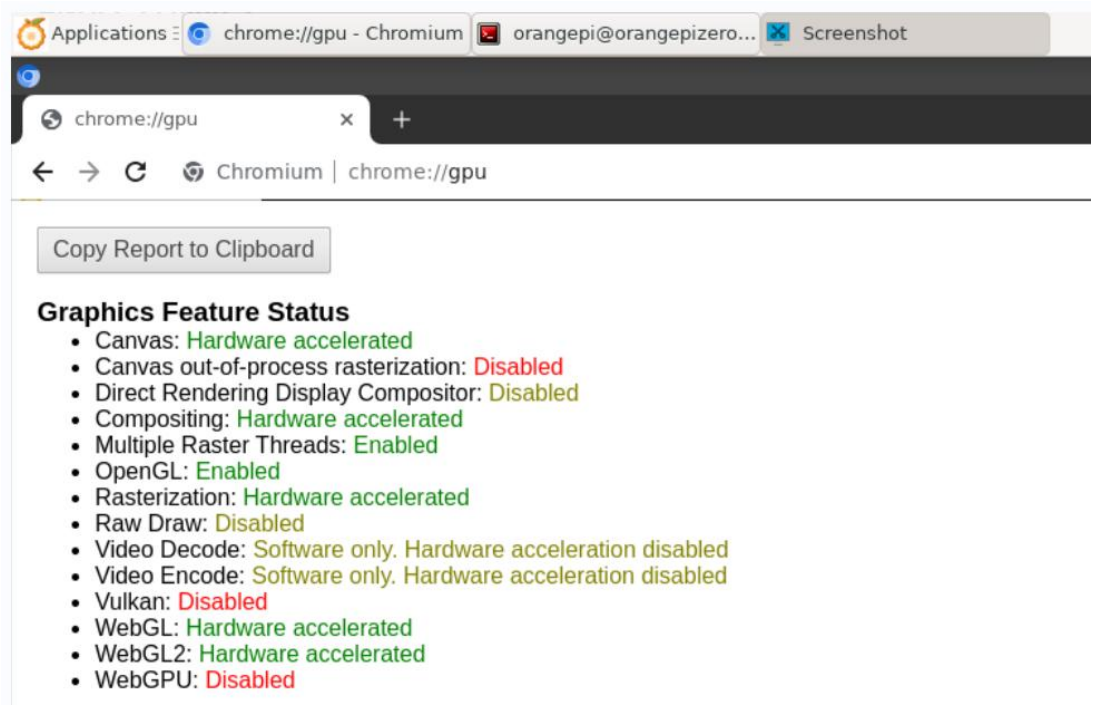


6) 然后可以安装下 snap 版本的 Chromium 浏览器，再打开 Chromium 浏览器

```
orangepi@orangepi:~$ sudo snap install chromium
```



7) 再在地址栏中输入 **chrome://gpu** 就可以查看到 GPU 的支持情况



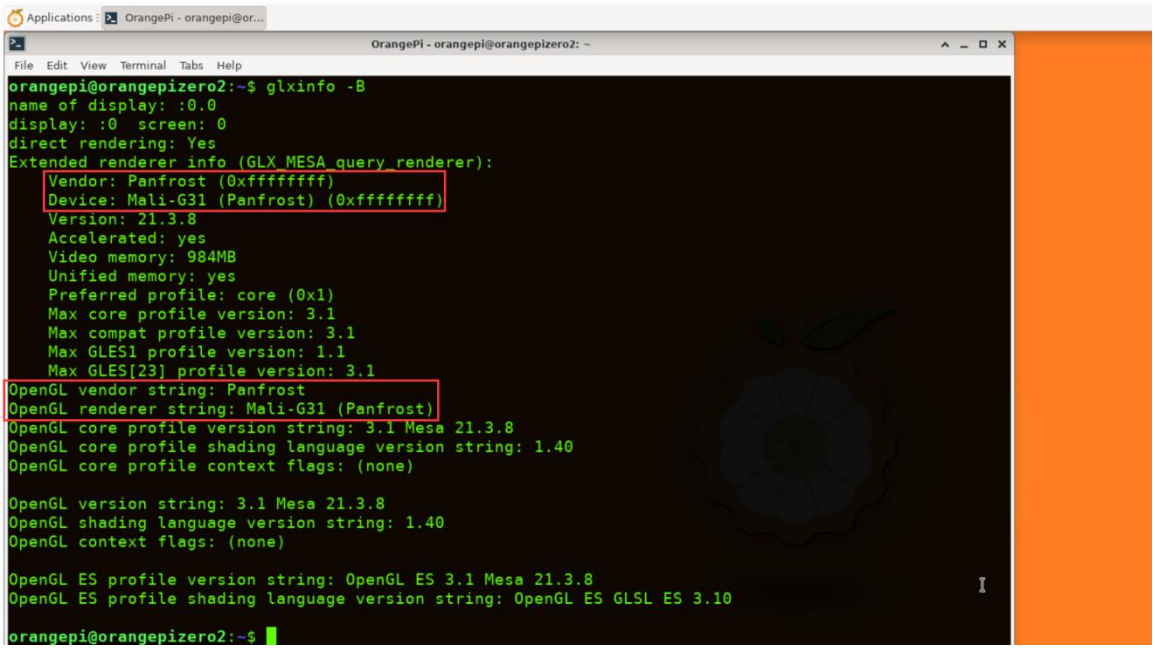
3. 46. 2. Debian12 Linux5.16 系统 GPU 测试说明

注意，下面的测试都是在桌面版本的系统中测试的，所以请确保开发板所使用的系统为桌面版本的系统。

1) 首先请连接好 HDMI 显示器，下面所有的命令都是在 HDMI 显示的桌面中操作的，请不要使用 ssh 远程登录或者使用串口来登录 Linux 系统

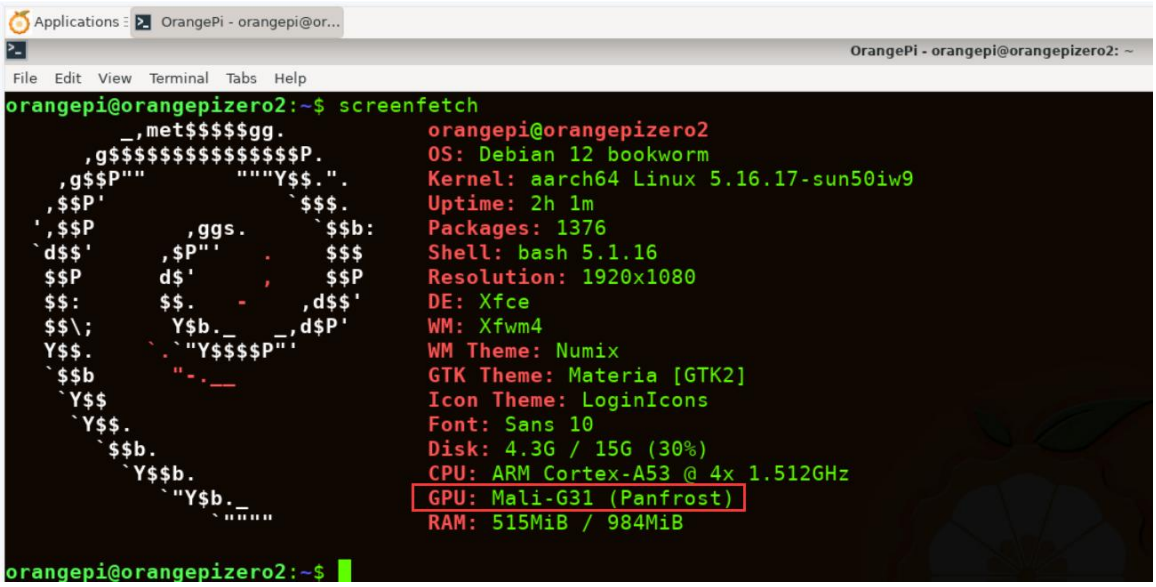
2) 进入桌面后，先打开一个终端，然后使用 **glxinfo -B** 命令可以看到使用的 GPU 驱动为 **Mali-G31 (Panfrost)**而非 **llvmpipe**

```
orangepi@orangepi:~$ glxinfo -B
```



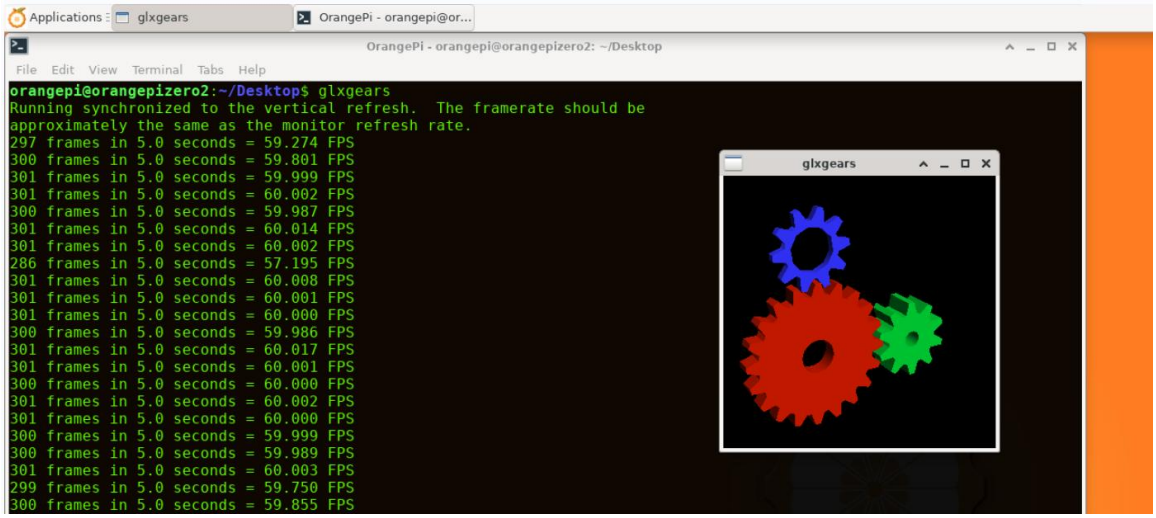
3) 使用 **screenfetch** 命令也可以看到 GPU 驱动使用的是 **Mali-G31 (Panfrost)**

```
orangepi@orangepi:~$ sudo apt install -y screenfetch
orangepi@orangepi:~$ screenfetch
```



4) **Glx-Gears** 测试如下所示

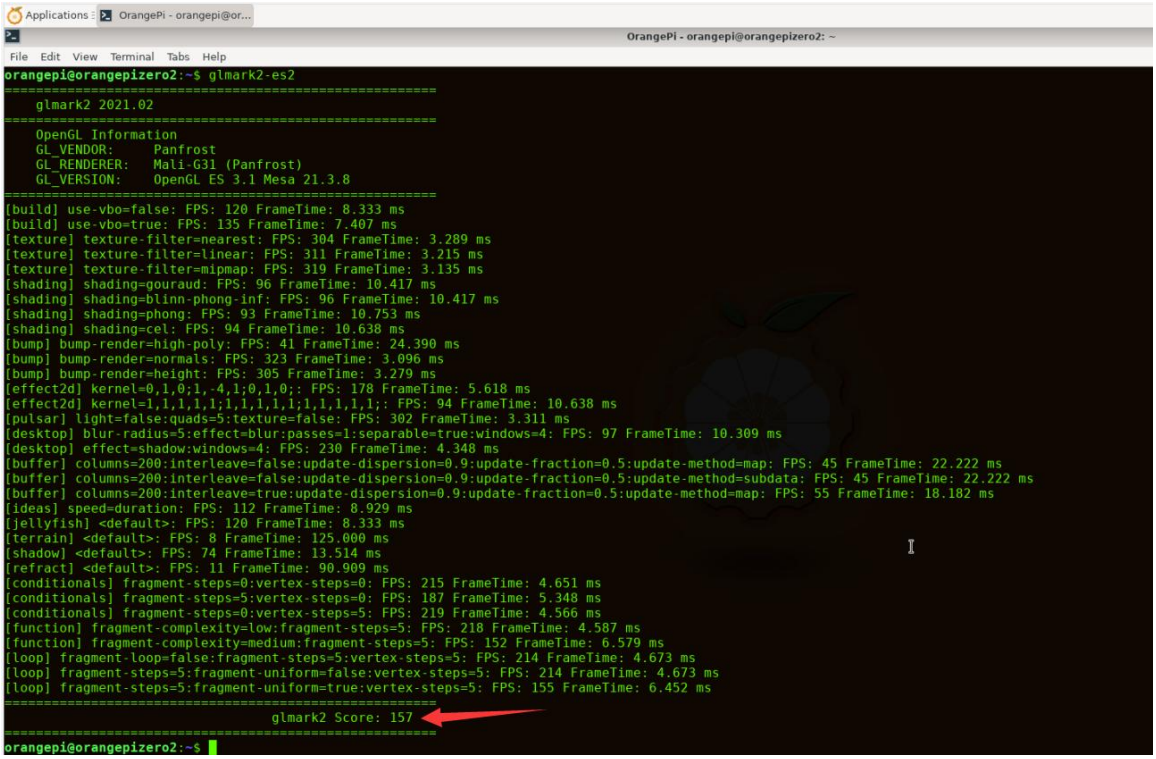
```
orangepi@orangepi:~$ glxgears
```



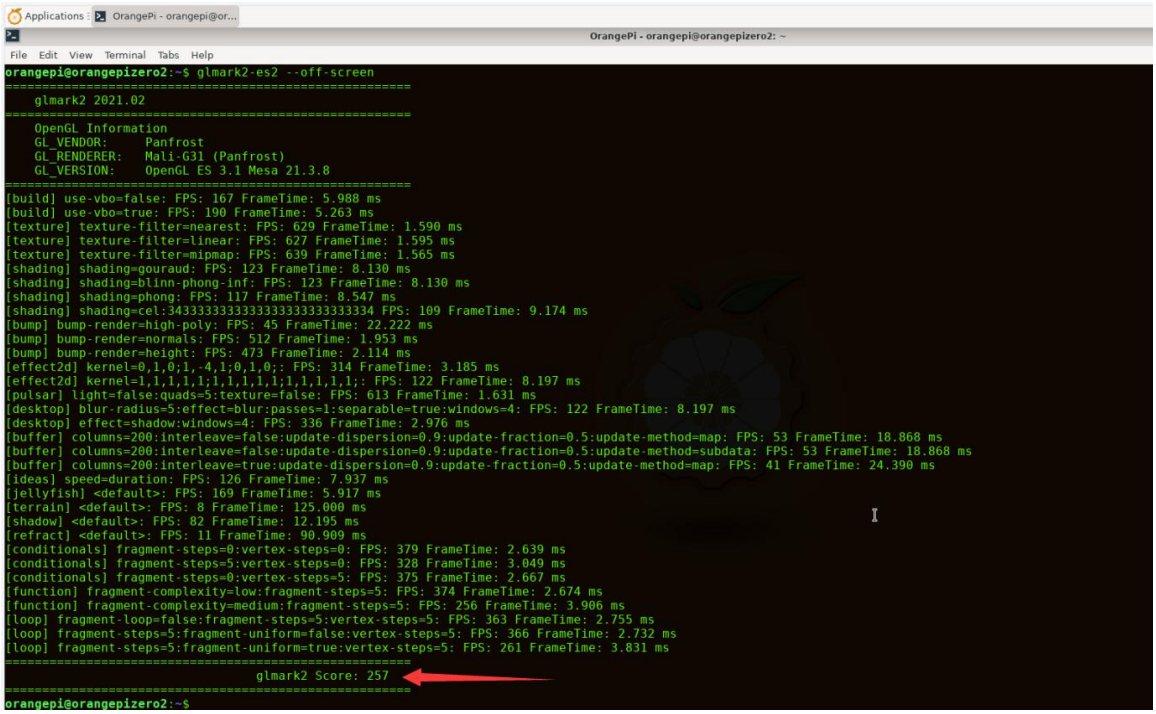
5) **glmark2-es2** 是 OpenGL (ES) 2.0 的基准测试工具，使用 glmark2 可以测试下 GPU OpenGL ES 2.0 的性能

```
orangepi@orangepi:~$ sudo apt install glmark2-es2-x11
orangepi@orangepi:~$ glmark2-es2
orangepi@orangepi:~$ glmark2-es2 --off-screen
```

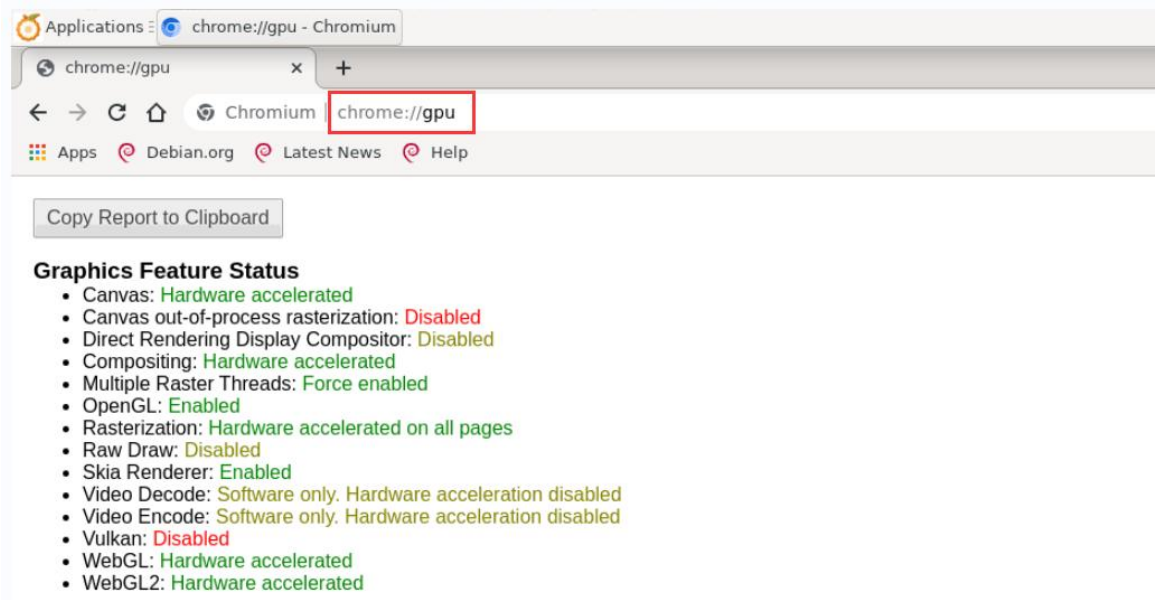
a. **glmark2-es2** 命令的测试分数如下所示



b. **glmark2-es2 --off-screen** 命令的测试分数如下所示



6) 另外可以打开 Chromium 浏览器，然后在地址栏中输入 **chrome://gpu** 就可以查看到 GPU 的支持情况



3.47. Linux 系统支持的部分编程语言测试

3.47.1. Debian Buster 系统

1) 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

```
orangepi@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

```
orangepi@orangepi:~$ vim hello_world.c
#include <stdio.h>

int main(void)
{
        printf("Hello World!\n");

        return 0;
}
```

c. 然后编译运行 **hello_world.c**

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

2) Debian Buster 默认安装有 Python2 和 Python3

a. Python 具体版本如下所示

```
orangepi@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.
>>>
```

```
orangepi@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

```
orangepi@orangepi:~$ vim hello_world.py
print('Hello World!')
```

c. 运行 **hello_world.py** 的结果如下所示

```
orangepi@orangepi:~$ python hello_world.py
Hello World!
orangepi@orangepi:~$ python3 hello_world.py
Hello World!
```

3) 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

```
orangepi@orangepi:~$ sudo apt install -y openjdk-11-jdk
```

b. 安装完后可以查看下 Java 的版本

```
orangepi@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**

```
orangepi@orangepi:~$ vim hello_world.java
public class hello_world
{
        public static void main(String[] args)
        {
                System.out.println("Hello World!");
        }
}
```

d. 然后编译运行 **hello_world.java**

```
orangepi@orangepi:~$ javac hello_world.java
orangepi@orangepi:~$ java hello_world
Hello World!
```

3.47.2. Debian Bullseye 系统

4) Debian Bullseye 默认安装有 gcc 编译工具链，可以直接在开发板的 Linux 系统中编译 C 语言的程序

a. gcc 的版本如下所示

```
orangepi@orangepi:~$ gcc --version
gcc (Debian 10.2.1-6) 10.2.1 20210110
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE.
```

b. 编写 C 语言的 **hello_world.c** 程序

```
orangepi@orangepi:~$ vim hello_world.c
#include <stdio.h>

int main(void)
{
        printf("Hello World!\n");

        return 0;
}
```

c. 然后编译运行 **hello_world.c**

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

5) Debian Bullseye 默认安装有 Python3

a. Python 具体版本如下所示

```
orangepi@orangepi:~$ python3
Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

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

```
orangepi@orangepi:~$ vim hello_world.py
print('Hello World!')
```

c. 运行 **hello_world.py** 的结果如下所示

```
orangepi@orangepi:~$ python3 hello_world.py
Hello World!
```

6) Debian Bullseye 默认没有安装 Java 的编译工具和运行环境

a. 可以使用下面的命令安装 openjdk，Debian Bullseye 中最新版本为 openjdk-17

```
orangepi@orangepi:~$ sudo apt install -y openjdk-17-jdk
```

b. 安装完后可以查看下 Java 的版本

```
orangepi@orangepi:~$ java --version
```

c. 编写 Java 版本的 **hello_world.java**

```
orangepi@orangepi:~$ vim hello_world.java
public class hello_world
{
        public static void main(String[] args)
        {
                System.out.println("Hello World!");
        }
}
```

d. 然后编译运行 **hello_world.java**

```
orangepi@orangepi:~$ javac hello_world.java
orangepi@orangepi:~$ java hello_world
Hello World!
```

3.47.3. Ubuntu Bionic 系统

1) Ubuntu Bionic 默认安装有 gcc 编译工具链，可以直接在开发板的 Linux 系统中编译 C 语言的程序

a. gcc 的版本如下所示

```
orangepi@orangepi:~$ gcc --version
gcc (Ubuntu/Linaro 7.5.0-3ubuntu1~18.04) 7.5.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR
```

```
PURPOSE.
```

b. 编写 C 语言的 **hello_world.c** 程序

```
orangepi@orangepi:~$ vim hello_world.c
#include <stdio.h>

int main(void)
{
        printf("Hello World!\n");

        return 0;
}
```

c. 然后编译运行 **hello_world.c**

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

2) Ubuntu Bionic 默认安装有 Python3

a. Python 具体版本如下所示

```
orangepi@orangepi:~$ python3
Python 3.6.9 (default, Oct  8 2020, 12:12:24)
[GCC 8.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

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

```
orangepi@orangepi:~$ vim hello_world.py
print('Hello World!')
```

c. 运行 **hello_world.py** 的结果如下所示

```
orangepi@orangepi:~$ python3 hello_world.py
Hello World!
```

3) Ubuntu Bionic 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-17

```
orangepi@orangepi:~$ sudo apt install -y openjdk-17-jdk
```

b. 安装完后可以查看下 Java 的版本

```
orangepi@orangepi:~$ java --version
```

```
openjdk 17.0.2 2022-01-18
OpenJDK Runtime Environment (build 17.0.2+8-Ubuntu-118.04)
OpenJDK 64-Bit Server VM (build 17.0.2+8-Ubuntu-118.04, mixed mode, sharing)
```

c. 编写 Java 版本的 **hello_world.java**

```
orangepi@orangepi:~$ vim hello_world.java
public class hello_world
{
        public static void main(String[] args)
        {
                System.out.println("Hello World!");
        }
}
```

d. 然后编译运行 **hello_world.java**

```
orangepi@orangepi:~$ javac hello_world.java
orangepi@orangepi:~$ java hello_world
Hello World!
```

3.47.4. Ubuntu Focal 系统

1) Ubuntu Focal 默认安装有 gcc 编译工具链，可以直接在开发板的 Linux 系统中编译 C 语言的程序

a. gcc 的版本如下所示

```
orangepi@orangepi:~$ gcc --version
gcc (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE.
```

b. 编写 C 语言的 **hello_world.c** 程序

```
orangepi@orangepi:~$ vim hello_world.c
#include <stdio.h>

int main(void)
{
        printf("Hello World!\n");
```

```
        return 0;
}
```

c. 然后编译运行 **hello_world.c**

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

2) Ubuntu Focal 默认安装有 Python3

a. Python3 具体版本如下所示

```
orangepi@orangepi:~$ python3
Python 3.8.10 (default, Sep 28 2021, 16:10:42)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

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

```
orangepi@orangepi:~$ vim hello_world.py
print('Hello World!')
```

c. 运行 **hello_world.py** 的结果如下所示

```
orangepi@orangepi:~$ python3 hello_world.py
Hello World!
```

3) Ubuntu Focal 默认没有安装 Java 的编译工具和运行环境

a. 可以使用下面的命令安装 openjdk-17

```
orangepi@orangepi:~$ sudo apt install -y openjdk-17-jdk
```

b. 安装完后可以查看下 Java 的版本

```
orangepi@orangepi:~$ java --version
openjdk 17.0.2 2022-01-18
OpenJDK Runtime Environment (build 17.0.2+8-Ubuntu-120.04)
OpenJDK 64-Bit Server VM (build 17.0.2+8-Ubuntu-120.04, mixed mode, sharing)
```

c. 编写 Java 版本的 **hello_world.java**

```
orangepi@orangepi:~$ vim hello_world.java
public class hello_world
{
        public static void main(String[] args)
        {
```

```
                System.out.println("Hello World!");
        }
}
```

d. 然后编译运行 **hello_world.java**

```
orangepi@orangepi:~$ javac hello_world.java
orangepi@orangepi:~$ java hello_world
Hello World!
```

3.47.5. Ubuntu Jammy 系统

4) Ubuntu Jammy 默认安装有 gcc 编译工具链，可以直接在开发板的 Linux 系统中编译 C 语言的程序

a. gcc 的版本如下所示

```
orangepi@orangepi:~$ gcc --version
gcc (Ubuntu 11.2.0-19ubuntu1) 11.2.0
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

b. 编写 C 语言的 **hello_world.c** 程序

```
orangepi@orangepi:~$ vim hello_world.c
#include <stdio.h>

int main(void)
{
        printf("Hello World!\n");

        return 0;
}
```

c. 然后编译运行 **hello_world.c**

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

5) Ubuntu Jammy 默认安装有 Python3

a. Python3 具体版本如下所示

```
orangepi@orangepi:~$ python3
Python 3.10.4 (main, Apr  2 2022, 09:04:19) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

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

```
orangepi@orangepi:~$ vim hello_world.py
print('Hello World!')
```

c. 运行 **hello_world.py** 的结果如下所示

```
orangepi@orangepi:~$ python3 hello_world.py
Hello World!
```

6) Ubuntu Jammy 默认没有安装 Java 的编译工具和运行环境

a. 可以使用下面的命令安装 openjdk-18

```
orangepi@orangepi:~$ sudo apt install -y openjdk-18-jdk
```

b. 安装完后可以查看下 Java 的版本

```
orangepi@orangepi:~$ java --version
openjdk 18-ea 2022-03-22
OpenJDK Runtime Environment (build 18-ea+36-Ubuntu-1)
OpenJDK 64-Bit Server VM (build 18-ea+36-Ubuntu-1, mixed mode, sharing)
```

c. 编写 Java 版本的 **hello_world.java**

```
orangepi@orangepi:~$ vim hello_world.java
public class hello_world
{
        public static void main(String[] args)
        {
                System.out.println("Hello World!");
        }
}
```

d. 然后编译运行 **hello_world.java**

```
orangepi@orangepi:~$ javac hello_world.java
orangepi@orangepi:~$ java hello_world
Hello World!
```

3.48. Linux 部分应用的安装方法

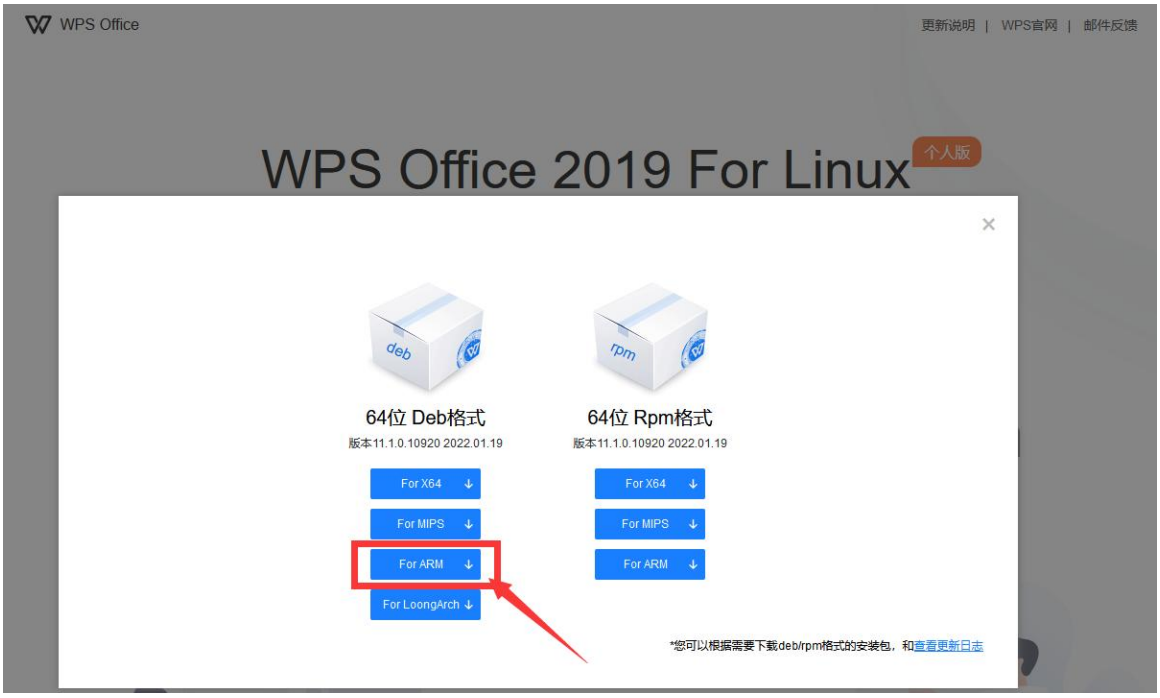
3.48.1. Linux 版本 WPS 的安装方法

请确保开发板安装的 Ubuntu 或者 Debian 系统为桌面版本的系统。

1) Linux 版本的 WPS 下载地址为

https://linux.wps.cn/

点击**立即下载**后，请选择 **64 位 deb 格式 ARM 版本**的软件包



2) 然后将下载的 **wps-office_11.1.0.10920_arm64.deb** 上传到开发板的 Linux 系统中

3) 然后安装 WPS 的依赖包

```
orangepi@orangepi:~$ sudo apt install -y libglu1-mesa xdg-utils
```

4) 安装 WPS 前请先设置好系统的中文环境，具体步骤请参考设置中文环境以及安装中文输入法一节

5) 然后使用下面的命令安装 WPS

```
orangepi@orangepi:~$ sudo apt install ./wps-office_11.1.0.10920_arm64.deb
```

6) 然后安装 WPS 需要的字体

a. 首先下载 **wps-fonts.zip** 字体压缩包

链接：https://pan.baidu.com/s/1vWQmCmSYdH7iCDFyKpJtVw
提取码：zero

b. 然后上传 **wps-fonts.zip** 到开发板的 Linux 系统中

c. 然后使用下面的命令解压

```
orangepi@orangepi:~$ unzip wps-fonts.zip -d wps-fonts
Archive:  wps-fonts.zip
  inflating: wps-fonts/mtextra.ttf
  inflating: wps-fonts/symbol.ttf
  inflating: wps-fonts/WEBDINGS.TTF
  inflating: wps-fonts/wingding.ttf
  inflating: wps-fonts/WINGDNG2.ttf
  inflating: wps-fonts/WINGDNG3.ttf
```

d. 然后将字体文件复制到**/usr/share/fonts/wps-office** 中

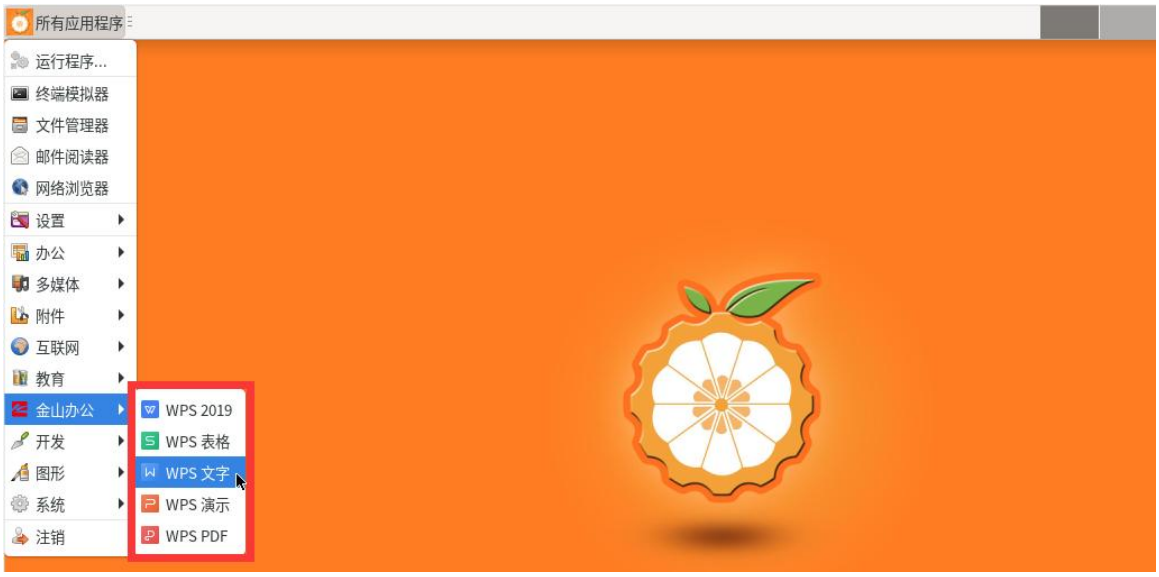
```
orangepi@orangepi:~$ sudo cp wps-fonts/* /usr/share/fonts/wps-office/
```

e. 然后生成索引信息并更新字体缓存

```
orangepi@orangepi:~$ sudo mkfontscale && sudo mkfontdir && sudo fc-cache
```

f. **然后重启系统让配置生效**

7) 然后在**所有应用程序**中就能看到 WPS 了

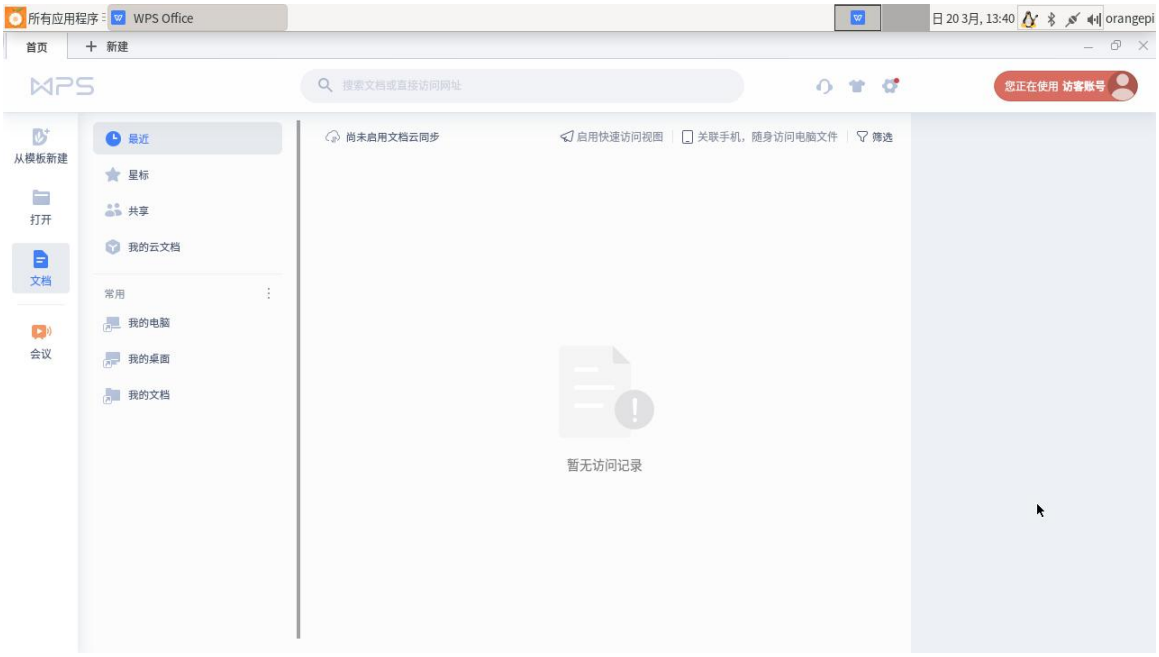


Ubuntu22.04 需要使用下面的命令才能打开 wps：

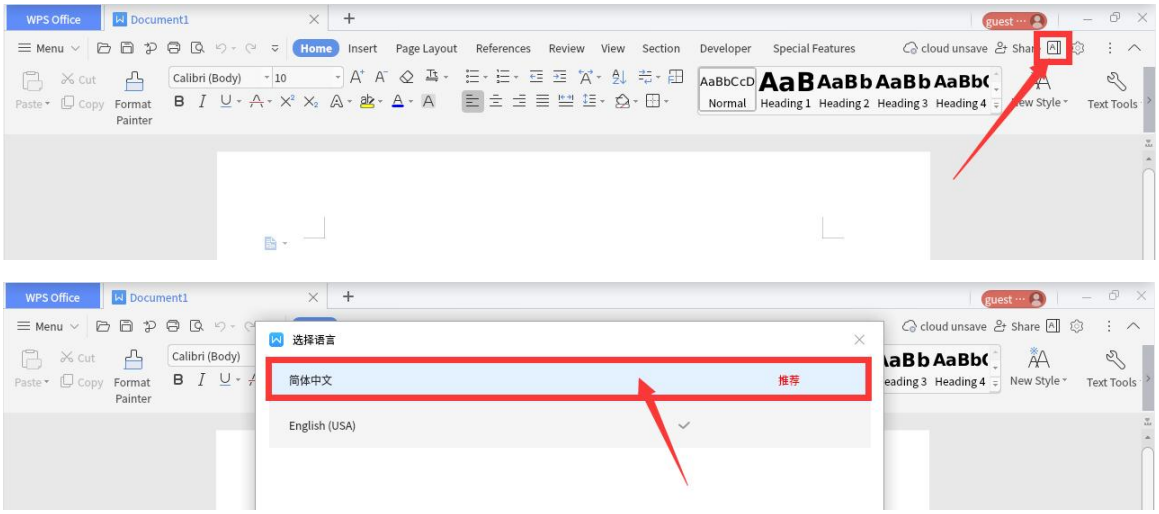
orangepi@orangepi:~$ **sudo wps**

8) 第一次打开 WPS，需要先同意 WPS 的许可协议和隐私协议，然后点击确定即可

9) WPS 主界面如下所示



10) Debian 系统安装完 WPS 后主界面如果没有显示中文的话可以在下图所示的地方进行设置（打开一个空白的文档就会显示下面的界面），设置完后需要重启 WPS 才能生效



3.48.2. Linux 版本 QQ 的安装方法

请确保开发板安装的 Ubuntu 或者 Debian 系统为桌面版本的系统。

1) Linux 版本的 QQ 下载地址如下所示，我们可以通过 shell 脚本来安装 QQ，也可以通过下载 deb 包的方式来安装 QQ

https://im.qq.com/linuxqq/download.html

2) 推荐使用 shell 脚本来安装 QQ，首先需要下载 shell 脚本，命令如下所示

```
orangepi@orangepi:~$ wget  \
https://down.qq.com/qqweb/LinuxQQ/linuxqq_2.0.0-b2-1089_arm64.sh

--2022-02-24 11:06:33--
https://down.qq.com/qqweb/LinuxQQ/linuxqq_2.0.0-b2-1089_arm64.sh
Resolving down.qq.com (down.qq.com)... 123.246.198.146, 42.249.219.120,
58.49.195.113, ...
Connecting to down.qq.com (down.qq.com)|123.246.198.146|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location:
https://4f4eefa7b6956d1ab658cbb09cc69794.rdt.tfogc.com:49156/down.qq.com/qqweb/L
inuxQQ/linuxqq_2.0.0-b2-1089_arm64.sh?mkey=62372c4c86c6d08e032308fc6f5d682b
&arrive_key=516777981279&cip=183.17.124.57&proto=https&access_type=
[following]
--2022-02-24 11:06:33--
https://4f4eefa7b6956d1ab658cbb09cc69794.rdt.tfogc.com:49156/down.qq.com/qqweb/L
inuxQQ/linuxqq_2.0.0-b2-1089_arm64.sh?mkey=62372c4c86c6d08e032308fc6f5d682b
&arrive_key=516777981279&cip=183.17.124.57&proto=https&access_type=
Resolving 4f4eefa7b6956d1ab658cbb09cc69794.rdt.tfogc.com
(4f4eefa7b6956d1ab658cbb09cc69794.rdt.tfogc.com)... 222.78.93.111
Connecting to 4f4eefa7b6956d1ab658cbb09cc69794.rdt.tfogc.com
(4f4eefa7b6956d1ab658cbb09cc69794.rdt.tfogc.com)|222.78.93.111|:49156... connected.
HTTP request sent, awaiting response... 200 OK
Length: 14006542 (13M) [application/x-sh]
Saving to: ‘linuxqq_2.0.0-b2-1089_arm64.sh’
```

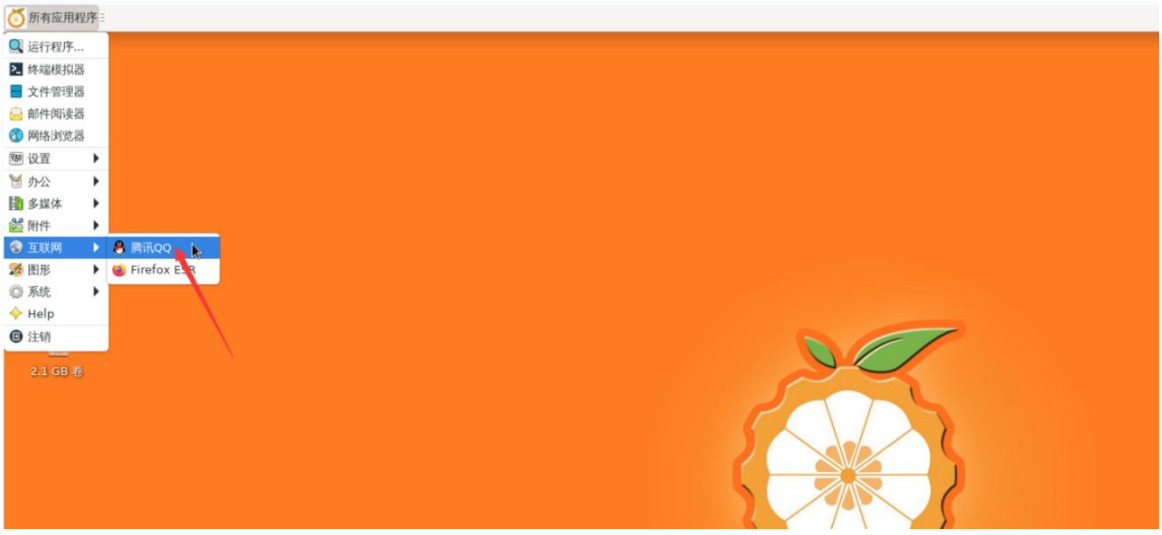
```
linuxqq_2.0.0-b2-1089_arm64.sh
100%[=============================================================>] 13.36M 598KB/s in 1m 48s

2022-02-24 11:08:21 (127 KB/s) - ‘linuxqq_2.0.0-b2-1089_arm64.sh’ saved [14006542/14006542]
```

3) 安装 QQ 前可以先设置好系统的中文环境，具体步骤请参考设置中文环境以及安装中文输入法一节

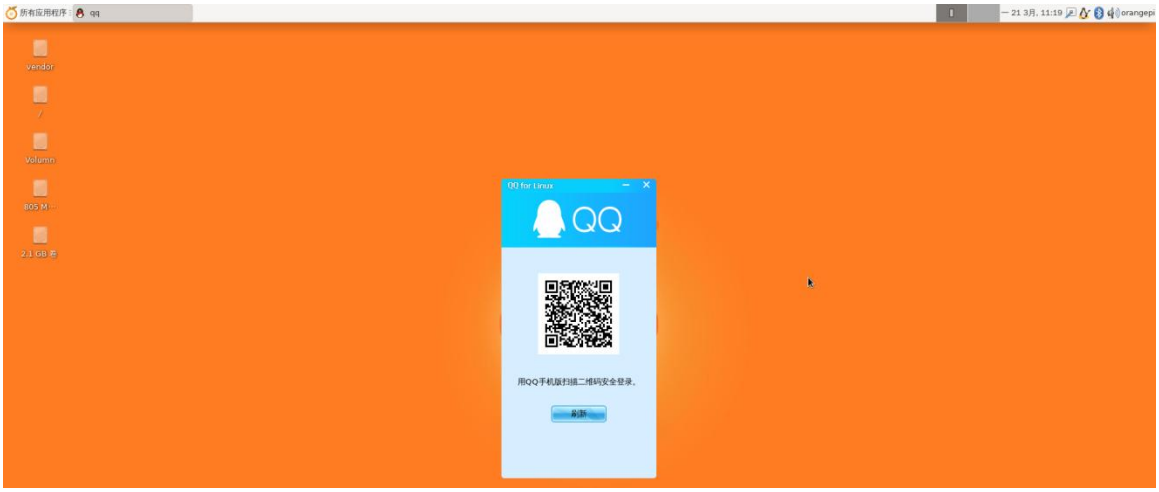
4) 然后运行刚下载的 shell 脚本就可安装 QQ 了，命令如下所示

```
orangepi@orangepi:~$ sudo bash linuxqq_2.0.0-b2-1089_arm64.sh
installing /usr/local/bin/qq
installing /usr/local/bin/crashpad_handler
installing /usr/local/share/tencent-qq/res.db
installing /usr/local/share/tencent-qq/qq.png
installing /usr/local/share/tencent-qq/credits.html
installing /usr/local/share/tencent-qq/CHANGELOG.txt
installing /usr/share/applications/qq.desktop
```

5) 然后在应用程序中就可以打开 QQ



6) 然后就可以拿出手机打开 QQ 扫描登录了



3.48.3. 向日葵远程控制软件的安装方法

请注意，向日葵官方是没有提供 arm64 版本的 Debian 和 Ubuntu 系统的安装包的。本小节测试的是麒麟 OS 软件仓库中的 sunloginclient_11.0.0.36662_arm64.deb 版本的安装包。

此安装包在 Ubuntu20.04 和 Ubuntu22.04 桌面版系统中测试是可以用的。

此安装包在 Ubuntu18.04、Debian10 和 Debian11 桌面版系统中测试是无法使用的。

1) 首先在开发板的 Linux 系统中使用下面的命令下载向日葵的 **arm64 deb** 版本的安装包

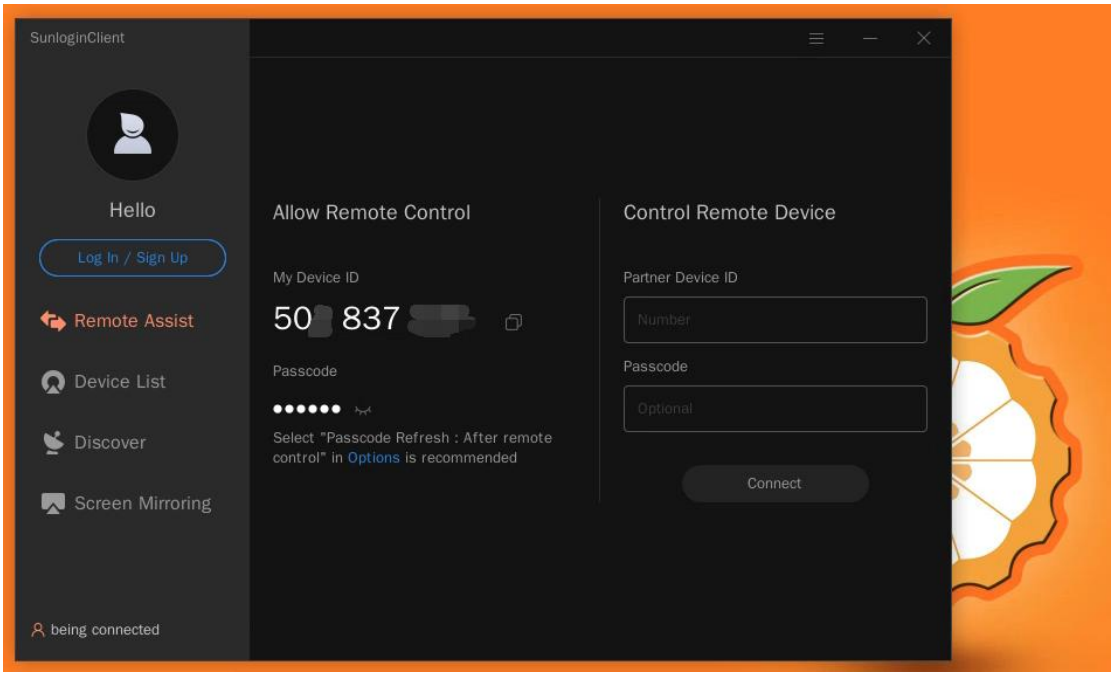
```
orangepi@orangepi:~$ wget \
http://archive.kylinos.cn/kylin/partner/pool/sunloginclient_11.0.0.36662_arm64.deb
```

2) 然后使用下面的命令安装 **sunloginclient_11.0.0.36662_arm64.deb**

```
orangepi@orangepi:~$ sudo apt-get update
orangepi@orangepi:~$ sudo apt-get install -y ./sunloginclient_11.0.0.36662_arm64.deb
```

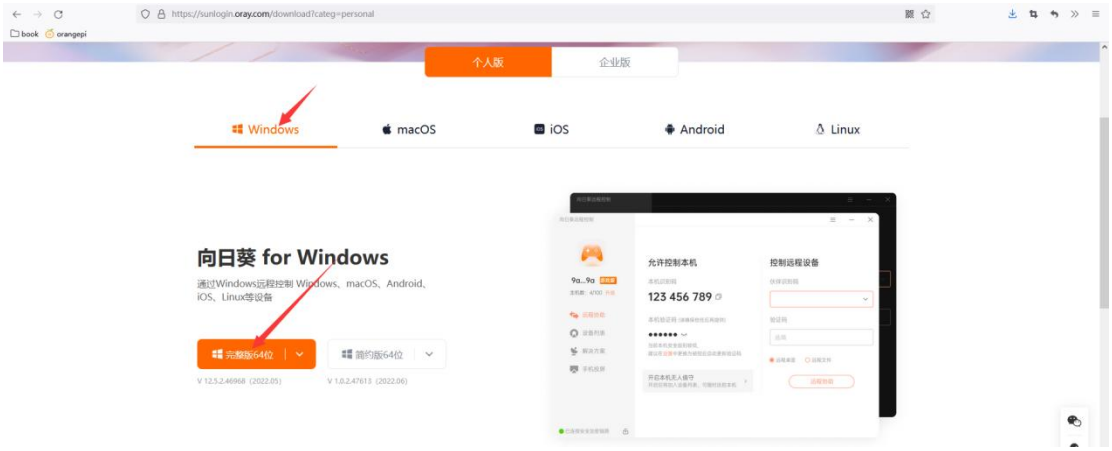
3) 安装完成后在 **Application->Internet** 中就能看到向日葵客户端的快捷方式

4) 在开发板的 Linux 系统中打开 **Sunlogin Client** 后的界面如下所示：

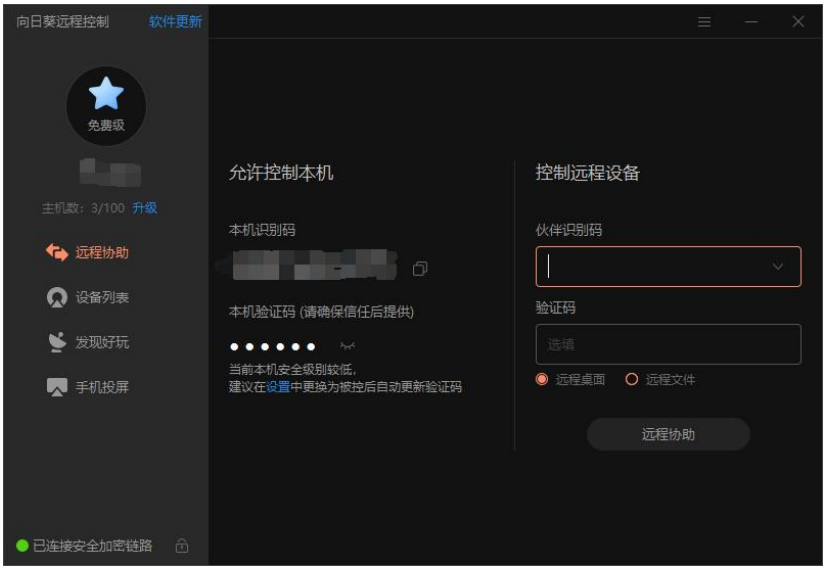


5) 然后在 Windows 电脑中安装下向日葵，向日葵也支持 macOS、iOS、Android 和 Linux 等平台，如果想使用的话，请下载对应的版本自行测试。向日葵 Windows 版本安装软件的下载链接如下所示：

https://sunlogin.oray.com/download?categ=personal



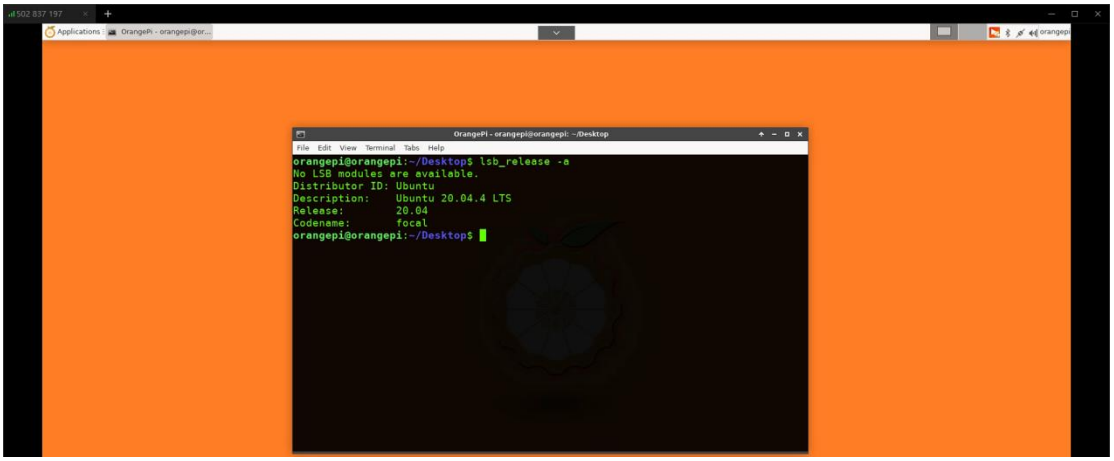
6) 在 Windows 系统中打开向日葵软件后的界面如下所示



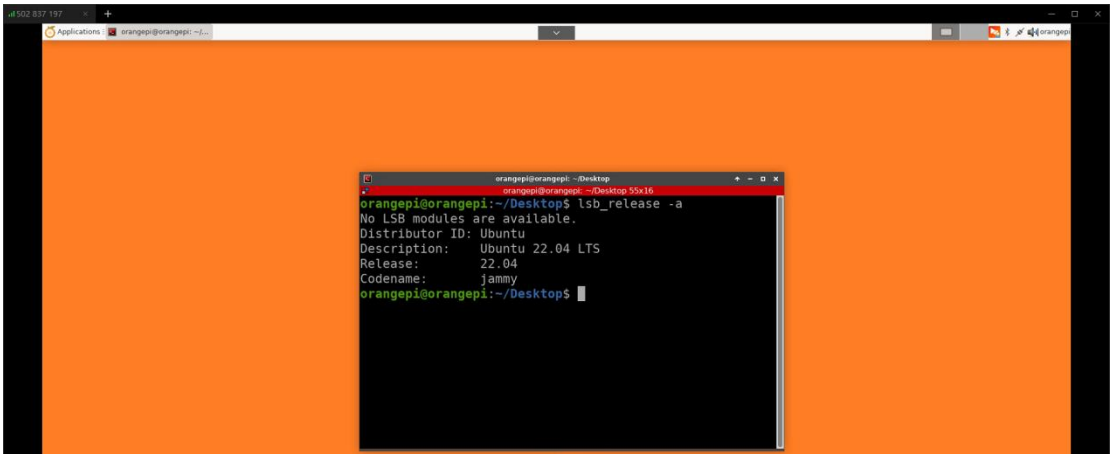
7) 然后在 Windows 系统中注册登录向日葵软件，再在向日葵**伙伴识别码**一栏中填入开发板 Linux 系统的向日葵软件中显示的 **My Device ID**，以及在**验证码**一栏中填入开发板 Linux 系统向日葵软件中显示的 **Passcode**，然后点击**远程协助**就能登录进开发板的 Linux 系统桌面

8) 在 Windows 中通过向日葵远程登录开发板 Linux 系统桌面的示意图如下所示

a. Ubuntu20.04 登录后的显示界面



b. Ubuntu22.04 登录后的显示界面



请注意，麒麟 OS 软件仓库中的 sunloginclient_11.0.0.36662_arm64.deb 虽然在 Ubuntu20.04 和 Ubuntu22.04 中基本使用没问题，但是还是有一些小问题的，比如开机无法自动启动，开机后第一次启动会有无法正常显示 My Device ID 和 Passcode 的情况出现，需要重新启动一次向日葵才行。

3.49. 安装内核头文件的方法

1) 内核头文件的获取方法有两种：

a. 方法一：从百度云盘下载编译好的头文件 deb 包

a) 百度云盘下载链接如下所示：

链接：**https://pan.baidu.com/s/1vWQmCmSYdH7iCDFyKpJtVw**
提取码：zero

b) 打开上面的链接后，首先找到名为 **linux-headers** 的文件夹

c) 然后进入 linux-headers 文件夹，下载对应内核的头文件的 deb 包

内核头文件 deb 包的版本号可能会更新，具体以实际看到的为准。	
内核版本	**内核头文件对应的 deb 包名**
Linux4.9	**linux-headers-legacy-sun50iw9_2.2.x_arm64.deb**
Linux5.16.17	**linux-headers-current-sun50iw9_3.0.x_arm64.deb**

b. 方法二：使用 orangepi-build 编译内核源码会自动生成内核头文件的 deb 包，具体方法请参考 **5.4 编译 linux 内核**和 **6.4 编译 linux 内核**两小节的说明

2) 然后将内核头文件 deb 包上传到开发板的 Linux 系统中

3) 然后使用下面的命令安装内核头文件 deb 包

内核头文件 deb 包的名字需要替换为实际的名字，请不要照抄。
安装过程中如果有打印警告信息，请直接忽略，不影响使用。

```
root@orangepi:~# dpkg -i linux-headers-legacy-sun50iw9_2.2.x_arm64.deb
```

4) 安装完后在**/usr/src** 下就能看到内核头文件所在的文件夹

```
root@orangepi:~# ls /usr/src
linux-headers-4.9.170-sun50iw9
```

5) 然后可以编写一个 hello 内核模块测试下内核头文件

a. 首先编写 hello 内核模块的代码，如下所示：

```
root@orangepi:~# vim hello.c
#include <linux/init.h>
#include <linux/module.h>

static int hello_init(void)
{
        printk("Hello Orange Pi -- init\n");

        return 0;
}
static void hello_exit(void)
```

```
{
        printk("Hello Orange Pi -- exit\n");

        return;
}

module_init(hello_init);
module_exit(hello_exit);

MODULE_LICENSE("GPL");
```

b. 然后编写编译 hello 内核模块的 Makefile 文件，如下所示：

```
root@orangepi:~# vim Makefile
ifneq ($(KERNELRELEASE),)
obj-m:=hello.o
else
KDIR :=/lib/modules/$(shell uname -r)/build
PWD  :=$(shell pwd)
all:
    make -C $(KDIR) M=$(PWD) modules
clean:
    rm -f *.ko *.o *.mod.o *.mod *.symvers *.cmd  *.mod.c *.order
endif
```

c. 然后使用 make 命令编译 hello 内核模块，编译过程的输出如下所示：

```
root@orangepi:~# make
make -C /lib/modules/5.16.17-sun50iw6/build M=/root modules
make[1]: Entering directory '/usr/src/linux-headers-5.16.17-sun50iw6'
  CC [M]  /root/hello.o
  MODPOST /root/Module.symvers
  CC [M]  /root/hello.mod.o
  LD [M]  /root/hello.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.16.17-sun50iw6'
```

d. 编译完后会生成 **hello.ko** 内核模块

```
root@orangepi:~# ls *.ko
hello.ko
```

e. 使用 **insmod** 命令可以将 **hello.ko** 内核模块插入内核中

```
root@orangepi:~# insmod hello.ko
```

f. 然后使用 **demsg** 命令可以查看下 **hello.ko** 内核模块的输出，如果能看到下面的输出说明 **hello.ko** 内核模块加载正确

```
root@orangepi:~# dmesg | grep "Hello"
[ 2871.893988] Hello Orange Pi -- init
```

g. 使用 **rmmod** 命令可以卸载 **hello.ko** 内核模块

```
root@orangepi:~# rmmod hello
root@orangepi:~# dmesg | grep "Hello"
[ 2871.893988] Hello Orange Pi -- init
[ 3173.800892] Hello Orange Pi -- exit
```

3.50. 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

```
orangepi@orangepi:~$ sudo poweroff
```

注意，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

```
orangepi@orangepi:~$ sudo reboot
```

4. Android TV 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 TV 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 hotsport	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

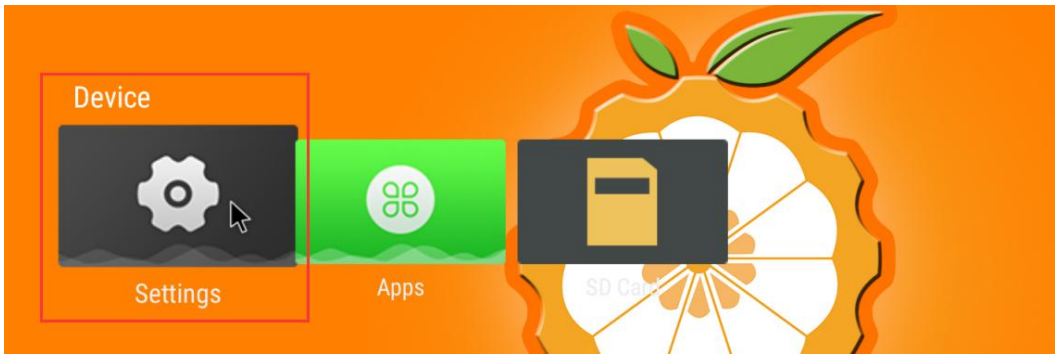
1) 我们一般都是使用鼠标和键盘来控制开发板的安卓系统，当进入某些界面，需要返回上一级界面或者桌面时，只能通过鼠标右键来返回，键盘是无法返回的

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

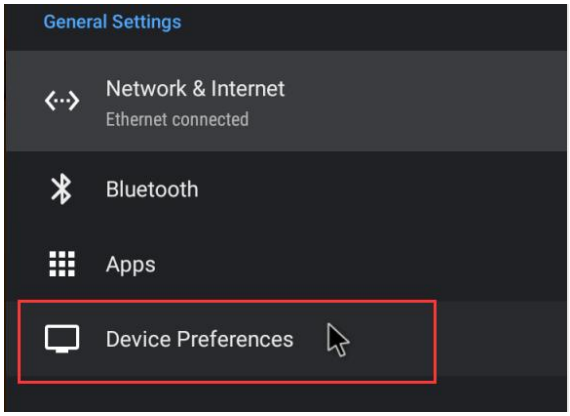
4.5. ADB 的使用方法

4.5.1. 打开 USB debugging 选项

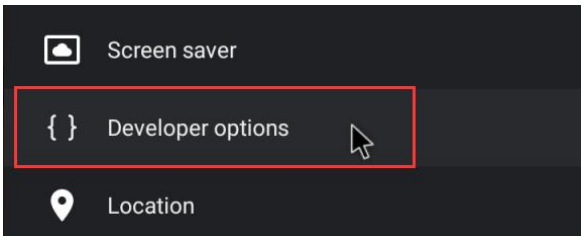
1) 选择 **Settings**



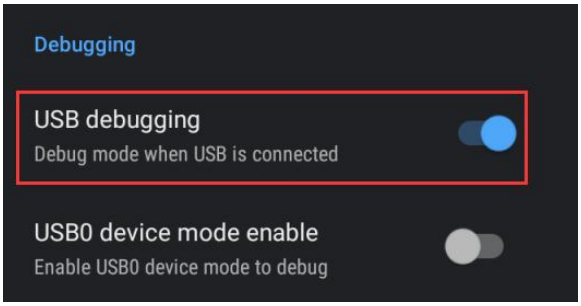
2) 然后选择 **Device Preferences**



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 Typc 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 Typc 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 接口能提供最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

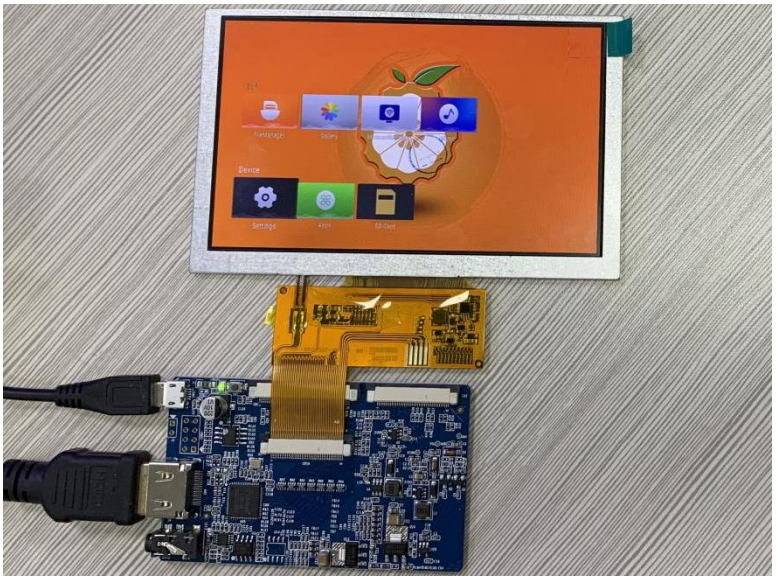
```
test@test:~$ adb shell
cupid-p2:/ #
```

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

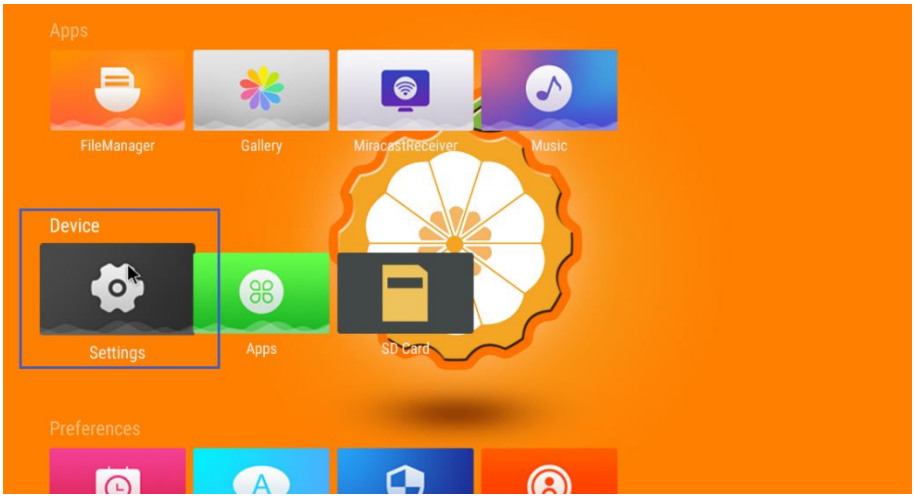
2) 然后使用 Micro HDMI 线将开发板的 Micro HDMI 接口连接到转接板的 HDMI 接口中，屏幕需要单独供电，请在转接板的 Micro USB 接口中插入 5V/2A 的电源

3) 开发板启动后，屏幕就可以看到 Android 系统的桌面了

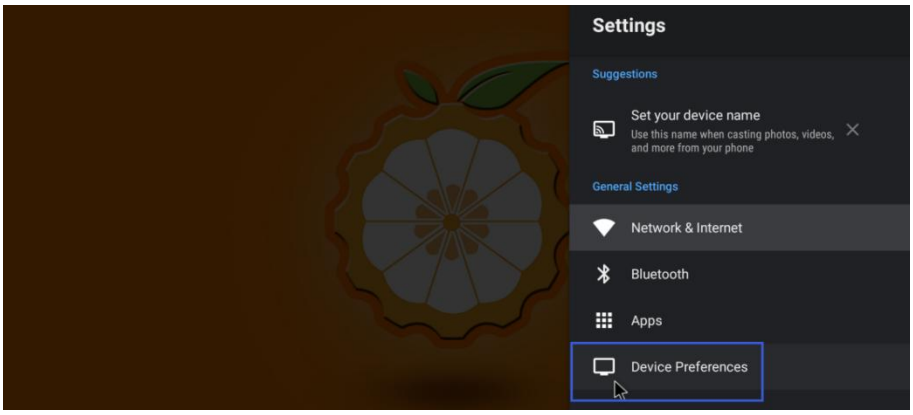


4) Android 系统第一次启动时，屏幕显示有抖动或者模糊不清是正常的，因为 Android 系统 HDMI 分辨率默认为 1080p，但是 TFT 液晶屏不支持这个分辨率，第一次启动进入 Android 系统后请首先修改 HDMI 输出的分辨率为 **480p**，然后屏幕显示就会很稳定了

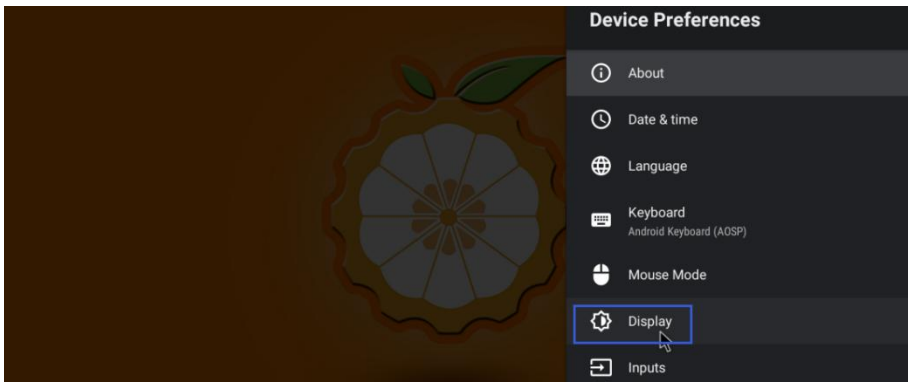
a. 首先选择 **Setting**



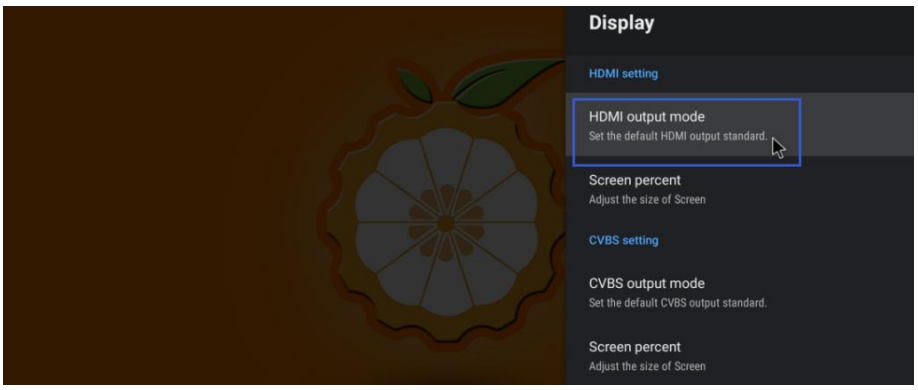
b. 然后选择 **Device Preferences**



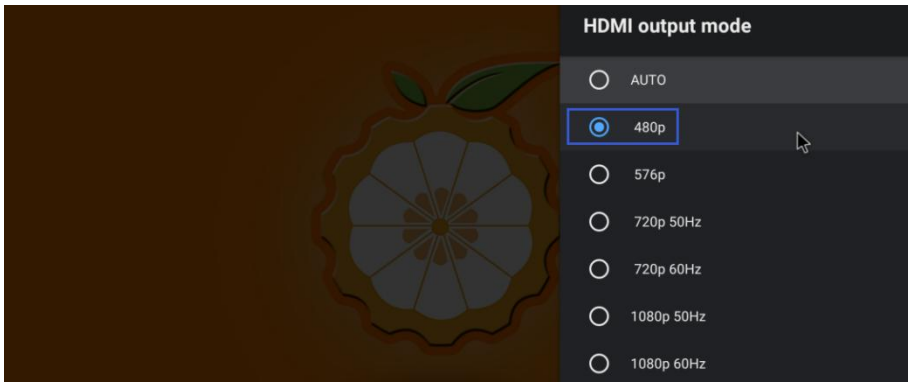
c. 然后选择 **Display**



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

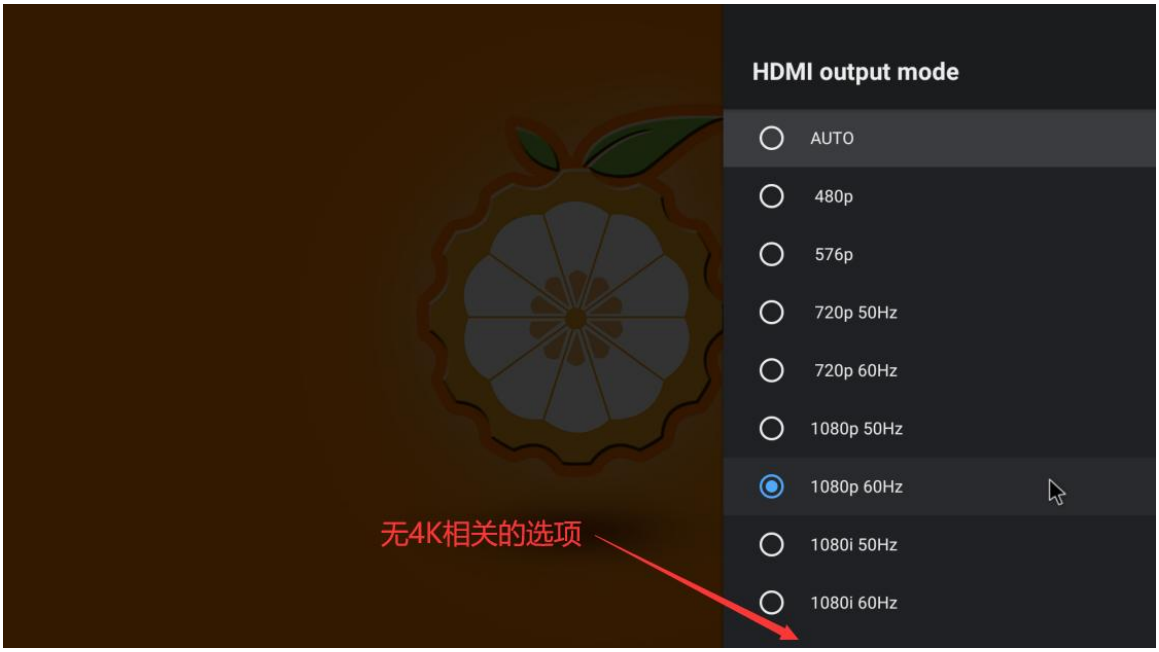


f. 此时再查看液晶屏的显示输出就会很稳定了

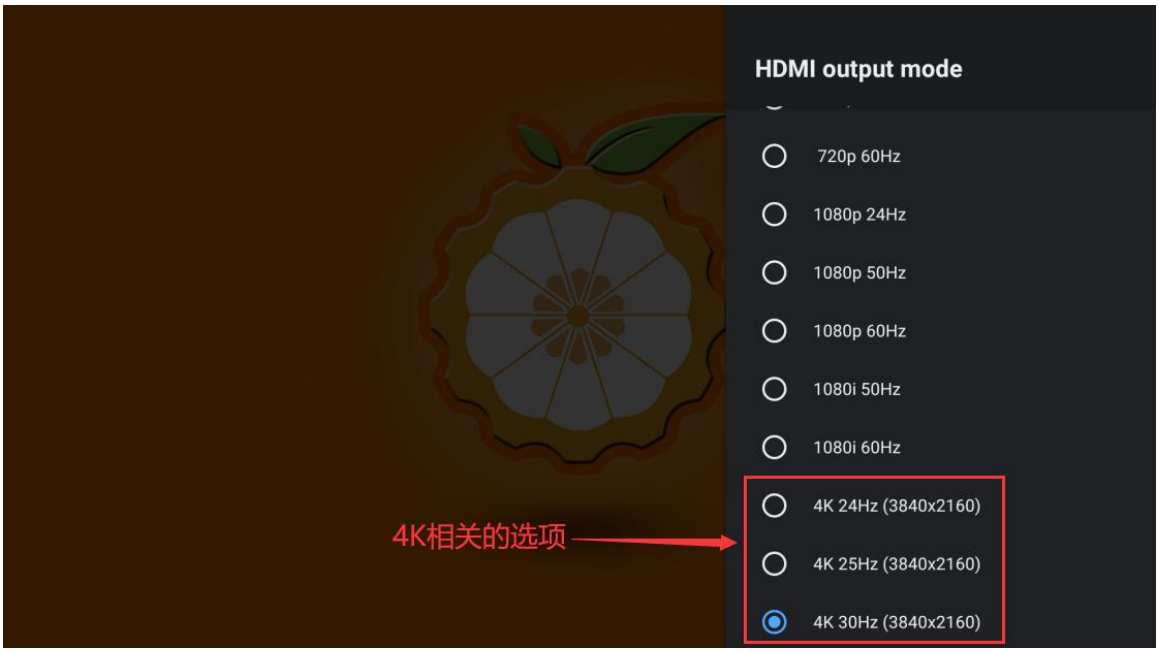
关闭开发板的电源后，请记得同时关闭屏幕转接板的电源，下次启动开发板前再打开。

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

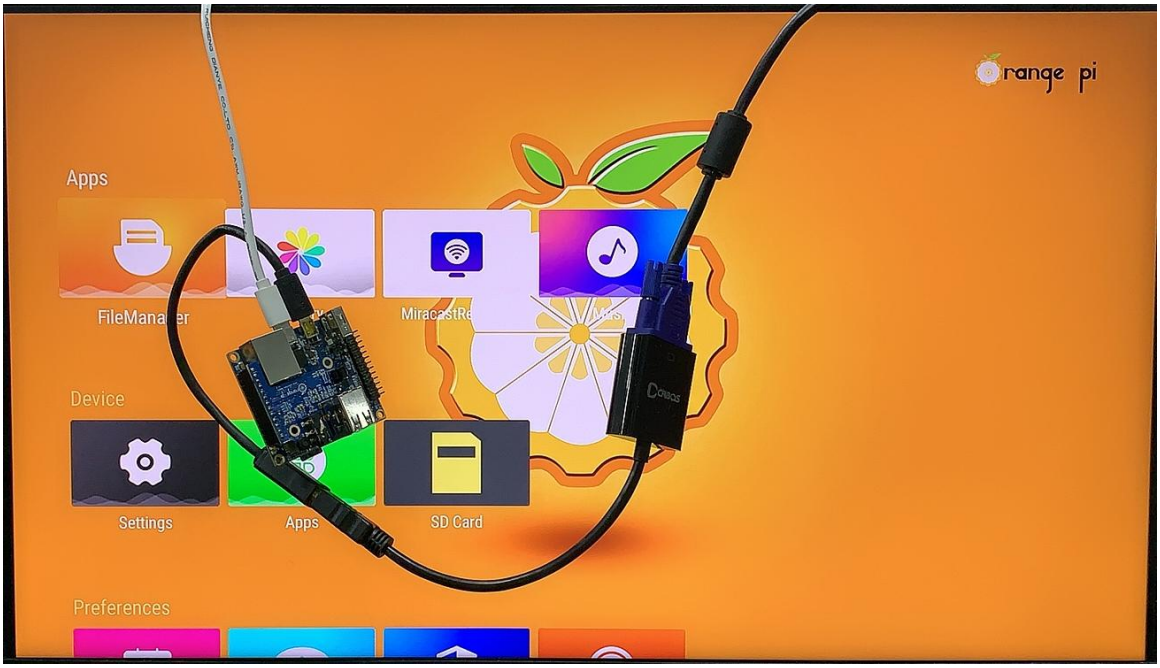
3) 首先需要准备下面的配件

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

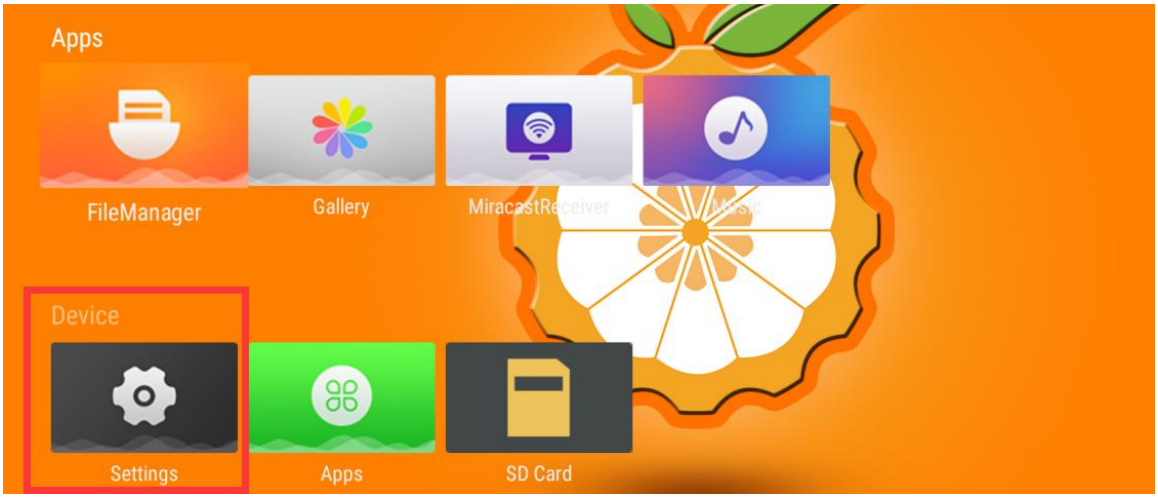
4) HDMI 转 VGA 显示测试如下所示



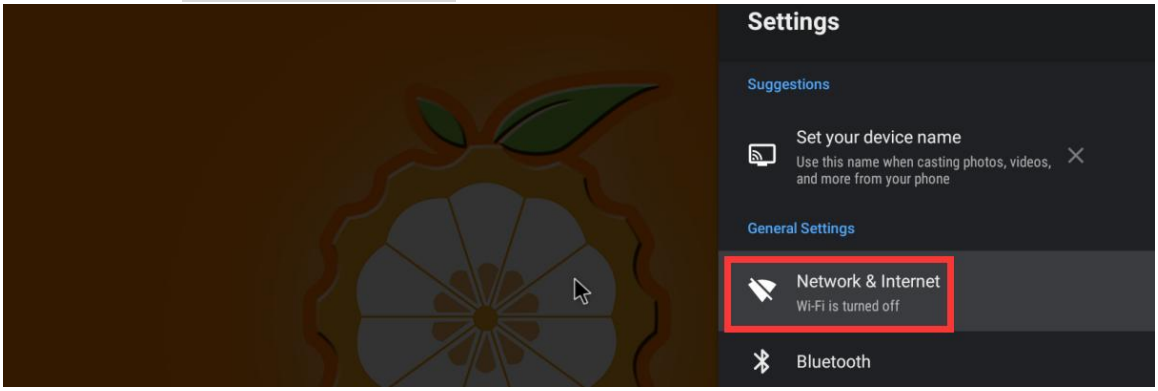
使用 HDMI 转 VGA 显示时，开发板以及开发板的 Android 系统是不需要做任何设置的，只需要开发板 Micro HDMI 接口能正常显示就可以了。所以如果测试有问题，请检查 HDMI 转 VGA 转换器、VGA 线以及显示器是否有问题。

4.9. WI-FI 的连接方法

1) 首先选择 **Settings**



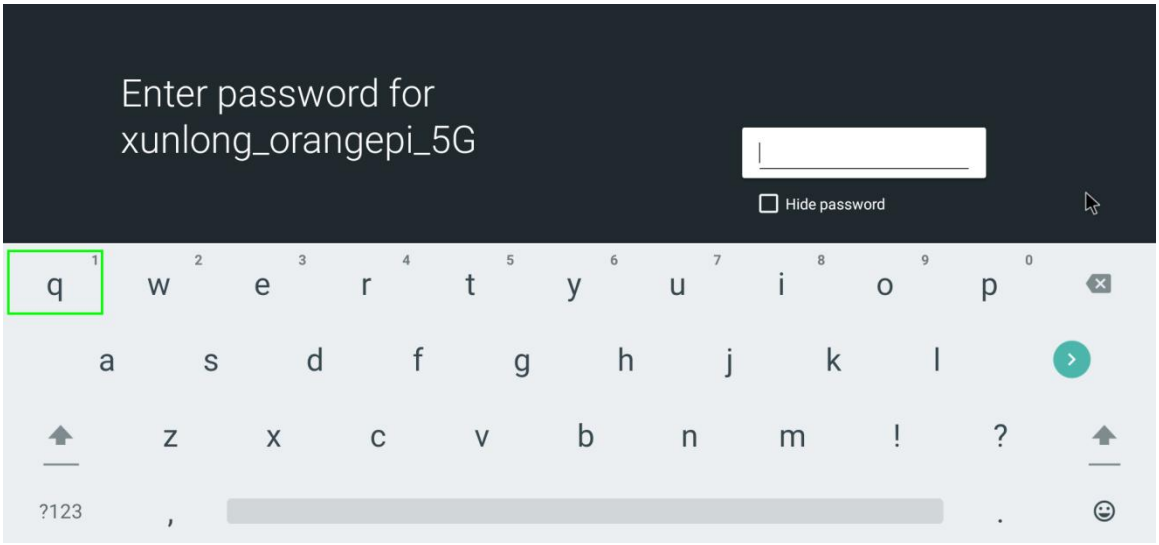
2) 然后选择 **Network & Internet**



3) 然后打开 WI-FI

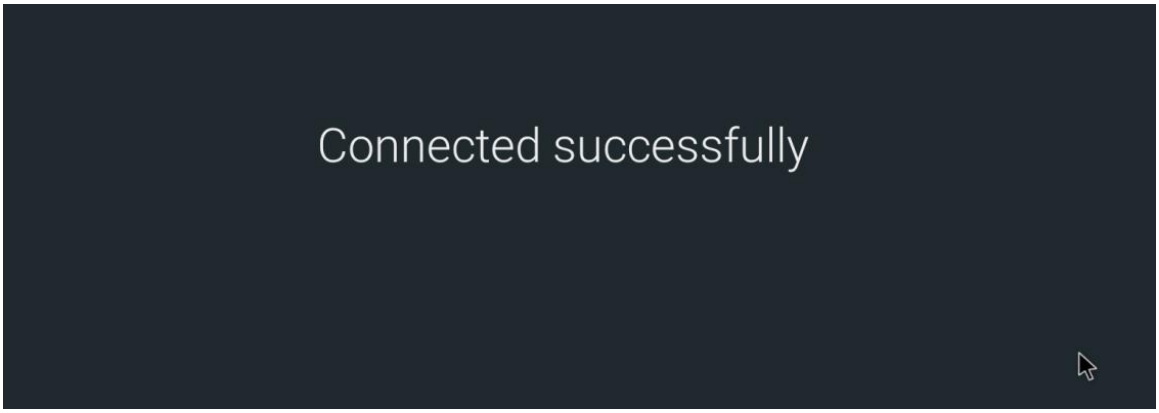
4) 打开 WI-FI 后在 **Available networks** 下面就可以看到搜索到的信号

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

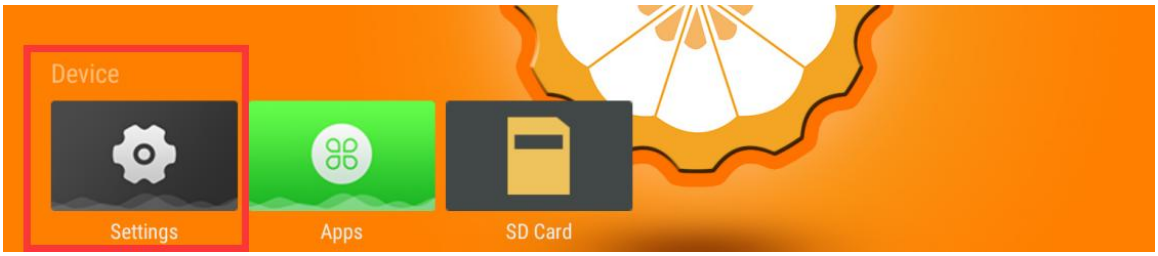
7) WI-FI 连接成功后的显示如下图所示



4. 10. WI-FI hotspot 的使用方法

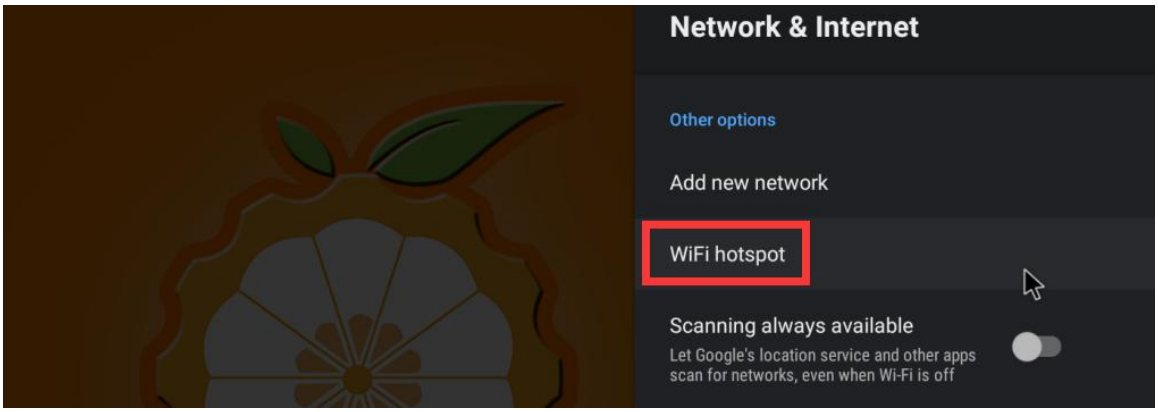
1) 首先请确保以太网口已连接网线，并且能正常上网

2) 然后选择 **Settings**



3) 然后选择 **Network & Internet**

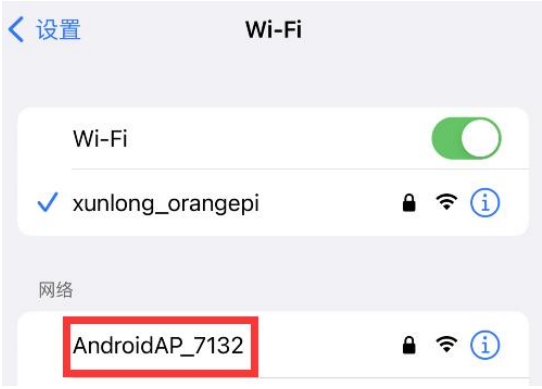
4) 然后选择 **WIFI hotspot**



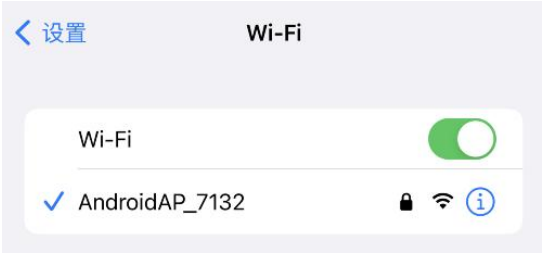
5) 然后打开 **Hotspot Enable**，下图中还可以看到生成的热点的名字和密码，记住它们，在连接热点的时候要用到（如果需要修改热点的名字和密码，需要先关闭 **Hotspot Enable**，然后才能修改）

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

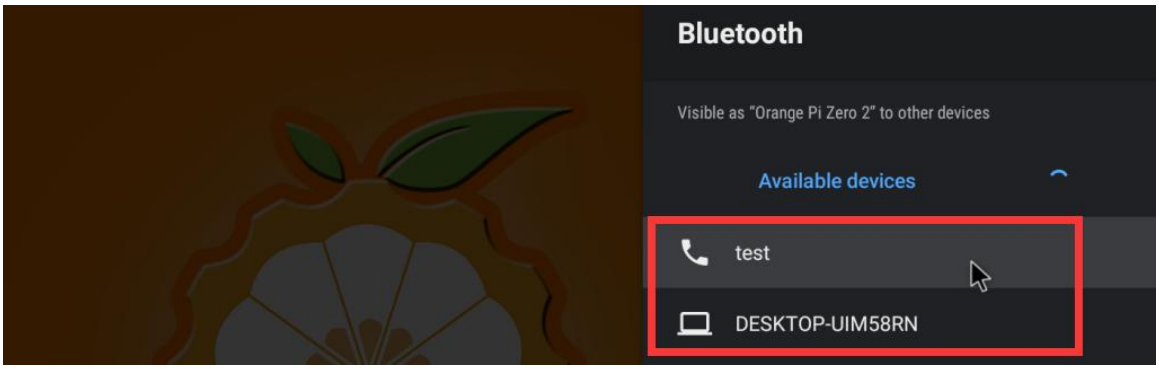
1) 首先选择 **Settings**

2) 然后选择 **Bluetooth**

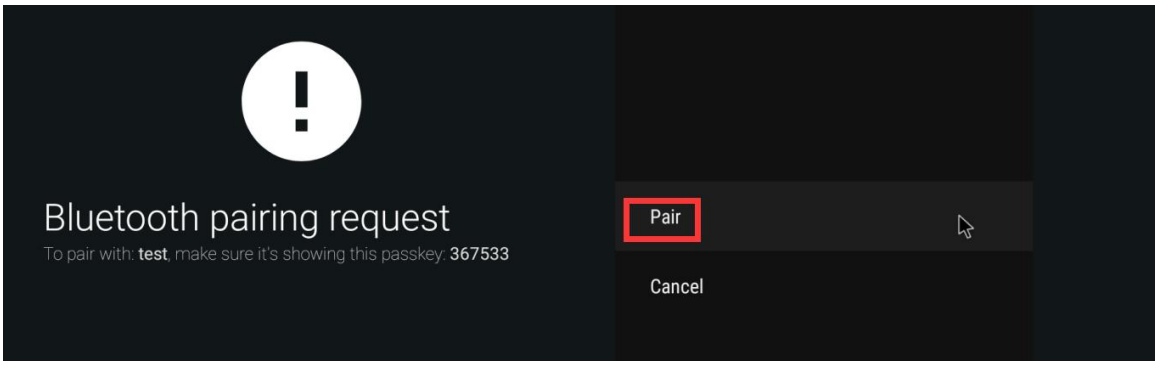
3) 然后打开 **Bluetooth Enable**

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

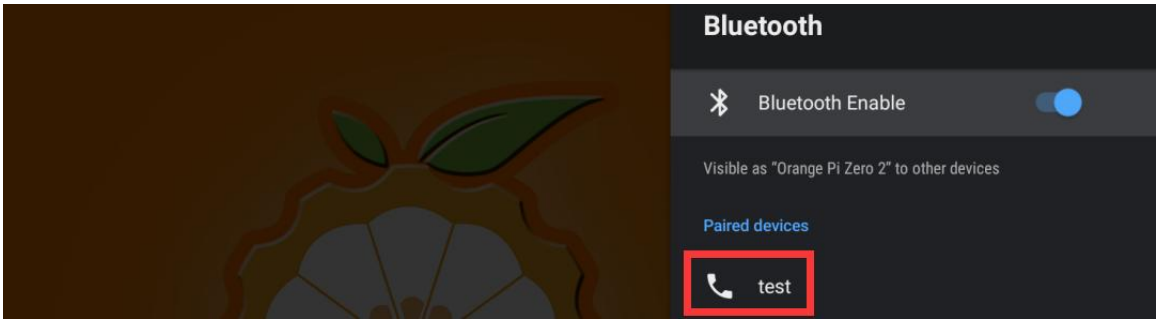


6) 然后点击想要连接的蓝牙设备就可以开始配对了，当弹出下面的界面时，请使用鼠标选择 **Pair** 选项

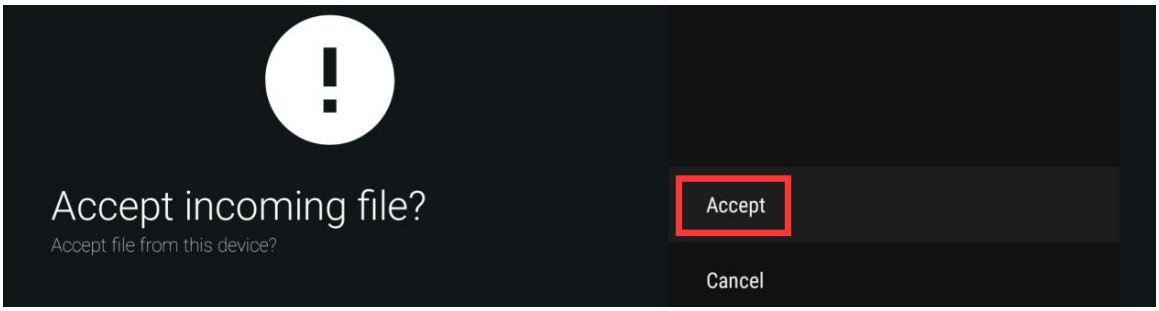


7) 这里测试的是开发板和安卓手机蓝牙的配置过程，此时在手机上会弹出下面的确认界面，在手机上也点击配对按钮后就会开始配对过程

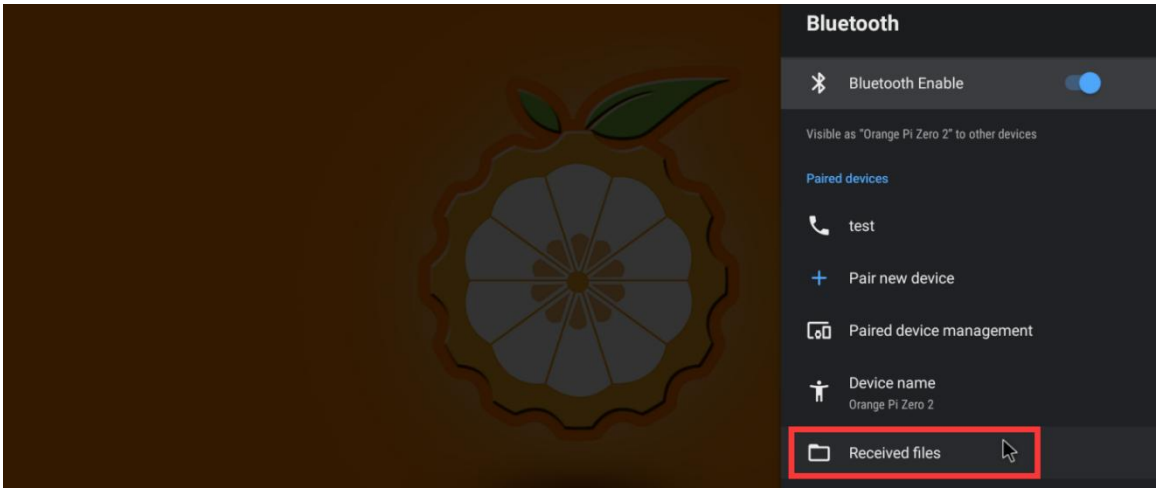
8) 配对完成后，再打开 **Paired devices** 下面就可以看到已配对的蓝牙设备



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

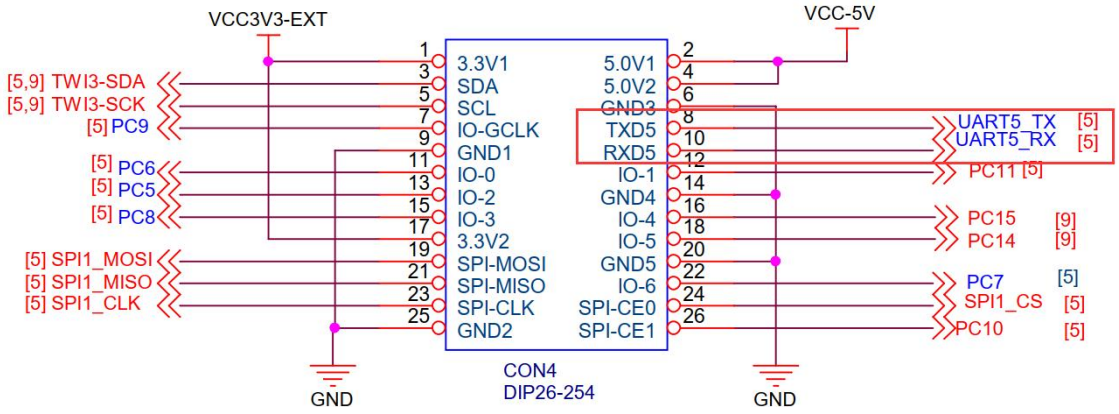


10) 开发板 Android 系统蓝牙接收到的图片可以打开 **Received files** 中查看



4. 12. 26pin 接口中串口的测试方法

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



2) 进入安卓系统后，请先确认下**/dev** 下是否存在 uart5 的设备节点

d. 使用 adb 查看的命令如下所示

```
test@test:~$ adb connect 192.168.1.82
connected to 192.168.1.82:5555
test@test:~$ adb shell ls /dev/ttyS5
/dev/ttyS5
```

e. 使用调试串口查看的命令如下所示

```
cupid-p2:/ # ls /dev/ttyS5
/dev/ttyS5
```

3) 然后开始测试 uart5 接口，先使用 2.54mm 的杜邦线短接要测试的 uart5 接口的 rx 和 tx

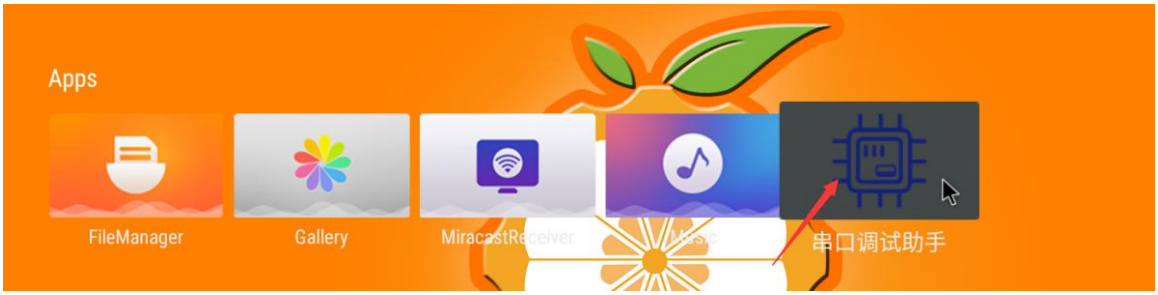
	uart5
tx 引脚	对应 8 号引脚
rx 引脚	对应 10 号引脚

4) 然后下载串口调试助手 APP——**SerialTestTool.apk**，此 APP 可以在 Orange Pi Zero 2 的官方工具里下载到

5) 然后安装 **SerialTestTool.apk**，使用 adb 安装的命令如下所示，如果不会用 adb 可以使用 U 盘拷贝安装

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

6) 然后打开串口调试助手 APP，串口调试助手安装好后的位置如下图所示



7) 然后设置串口调试助手

a. 串口号选择**/dev/ttyS5**

b. 波特率选择 **115200**

c. 最后记得点击右上角的**关闭按钮**打开串口

8) 然后可以发送一个字符串测试下串口的回环功能，看串口能不能接收到发送的字符串

9) 如下图所示，如果串口调试助手能接收到发送的字符串就说明串口能正常使用

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

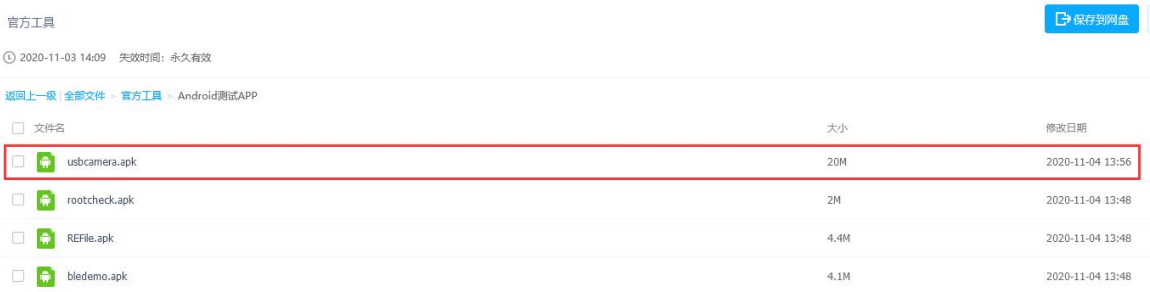
```
console:/ # lsmod
Module                  Size  Used by
sprdwl_ng             405504  0
sprdbt_tty             36864  2
uwe5622_bsp_sdio      274432  2 sprdwl_ng,sprdbt_tty
uvcvideo              102400  0
videobuf2_v4l2         28672  1 uvcvideo
videobuf2_vmalloc      16384  1 uvcvideo
videobuf2_memops       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/ -lh
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

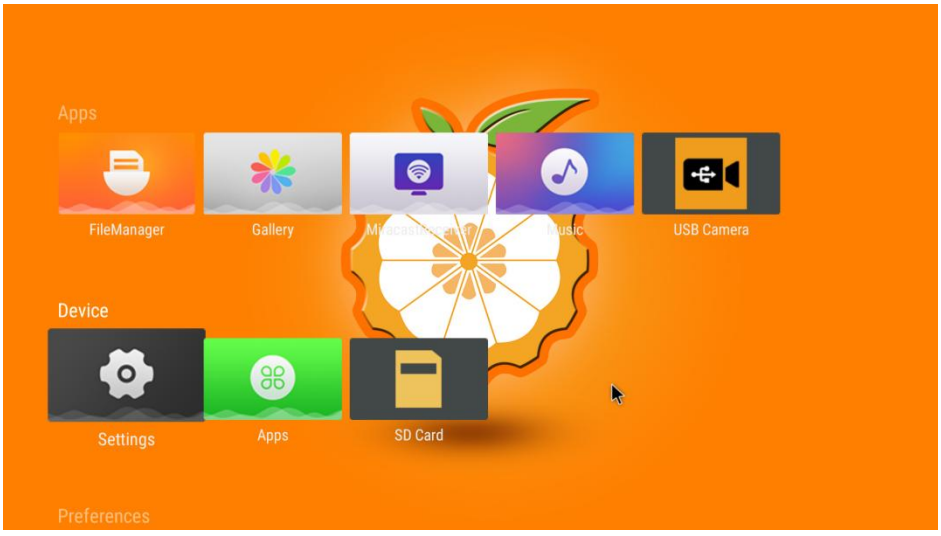
4) 在 Orange Pi Zero 2 资料下载页面的**官方工具**中下载 USB 摄像头测试 APP



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

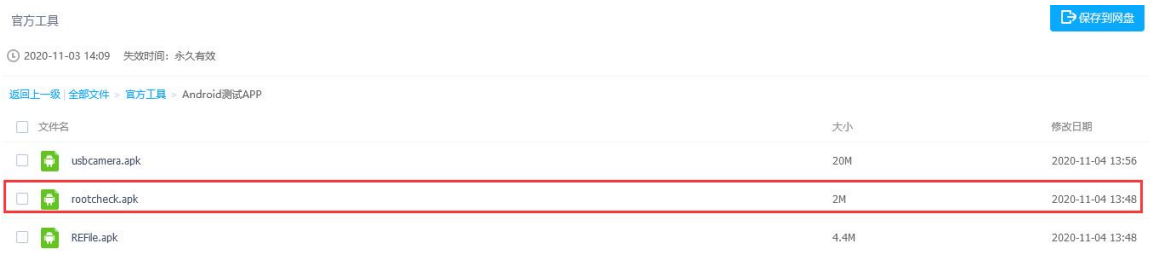


7) 然后双击打开 USB 摄像头 APP 就可以看到 USB 摄像头的输出视频了

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**



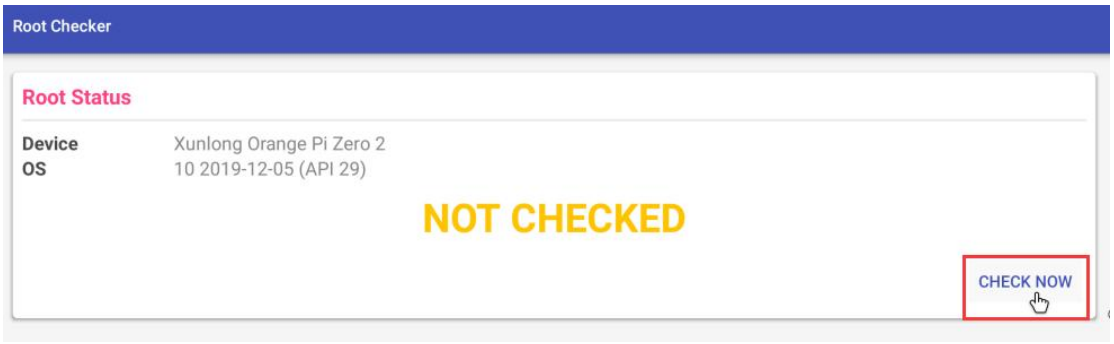
2) 然后确保 Ubuntu PC 和开发板的 adb 连接正常

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

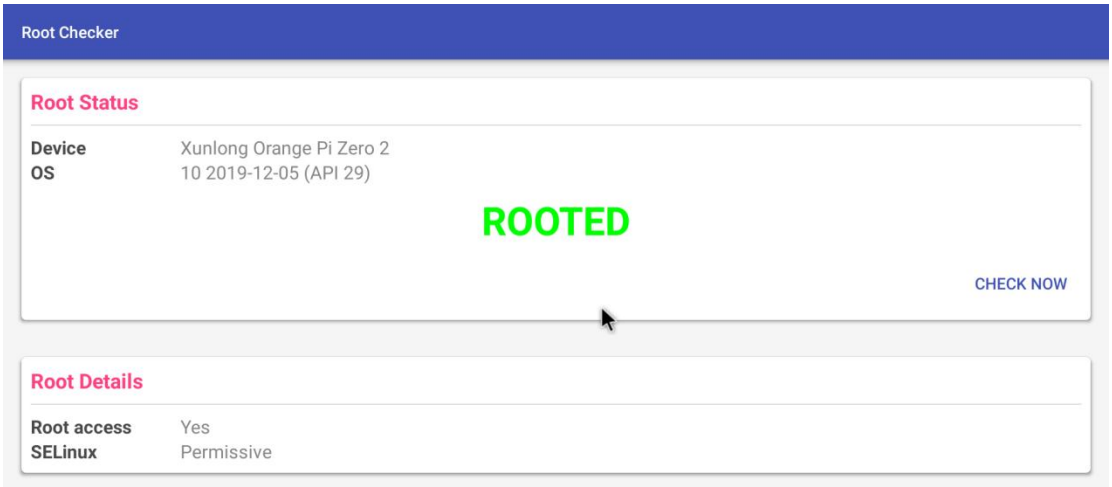
```
test@test:~$ adb install rootcheck.apk
```

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



4.15. 部分 Android APP 安装说明

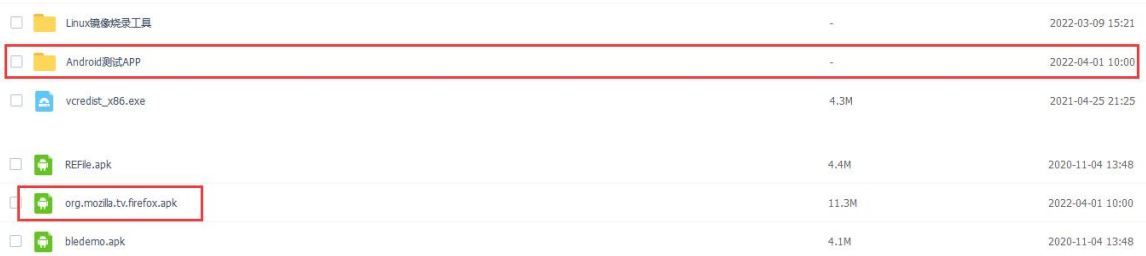
全志 H616 这款芯片主要是用在电视盒子上的，所以全志提供的安卓也是 TV 版本的安卓系统。如果需要安装 APP，可以先安装当贝这样的专为智能电视打造的应用市场，再通过当贝来安装需要的 APP。

另外需要注意的是，手机上使用的很多移动端 APP 在 Android TV 系统中是无法正常安装的，有些虽然能正常安装，但是使用起来也是很多问题。当安装软件前，最好看下软件的制造商有没有提供 TV 版本的安装包，比如腾讯视频、或者爱奇艺都有提供电视端 TV 版本的安装包。相对于移动端的 APP，TV 版本的体积要小很多，资源占用也会小很多，使用起来也会更流畅。

4.15.1. 浏览器安装说明

1) 首先要注意的是全志提供的 Android 系统是 TV 版本的安卓系统，所以安装火狐或者 Chrome 移动端的 APP 使用起来会非常卡顿的。如果需要安装浏览器请选择 TV 版本的安装包，不过现在市面上 TV 版本的浏览器非常少，目前能找到比较好用的是火狐以前开发的 TV 版本的浏览器，虽然已经停止更新了（打开 APP 后顶部会显示一个警告信息），但是使用起来没有问题，并且比较流畅。

2) 火狐 TV 版本 APP 可以在官方工具中下载到



3) 火狐 TV 版本浏览器的源码位置如下所示

https://github.com/mozilla-mobile/firefox-tv/releases

4.15.2. 腾讯视频安装说明

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

4.15.3. 优酷视频安装说明

1) 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

4.15.4. 爱奇艺视频安装说明

2) 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/

4.15.5. 乐播投屏安装说明

1) 下载地址如下所示

https://www.lebo.cn/Download.jsp

2) 在开发板的安卓 TV 系统中需要安装乐播投屏的电视版

3) 然后在手机中安装乐播投屏，就可以将手机屏幕投屏到开发板连接的 HDMI 显示器上

5. Linux SDK——旧版 orangepi-build 使用说明

5.1. 编译系统需求

1) Linux SDK，即 **orangepi-build**，只支持在安装有 **Ubuntu18.04** 的电脑上运行，所以下载 **orangepi-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 orangepi-build，因为 orangepi-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 orangepi-build，另外请不要在**开发板**的 Linux 系统中使用 orangepi-build

3) Ubuntu18.04 **amd64** 版本的安装镜像下载地址如下所示

```
https://repo.huaweicloud.com/ubuntu-releases/18.04.6/ubuntu-18.04.6-desktop-amd64.iso
或者
https://mirrors.tuna.tsinghua.edu.cn/ubuntu-releases/18.04.6/ubuntu-18.04.6-desktop-amd64.iso
```

4) 在电脑中或者虚拟机中安装完 Ubuntu18.04 后，请先设置 Ubuntu18.04 的软件源为清华源，不然后面安装软件的时候很容易由于网络原因而出错

a. 替换清华源的方法参考这个网页的说明即可

```
https://mirrors.tuna.tsinghua.edu.cn/help/ubuntu/
```

b. 注意 Ubuntu 版本需要切换到 18.04

Ubuntu 镜像使用帮助

Ubuntu 的软件源配置文件是 /etc/apt/sources.list。将系统自带的该文件做个备份，将该文件替换为下面内容，即可使用 TUNA 的软件源镜像。

选择你的ubuntu版本: 18.04 LTS

```
# 默认注释了源码镜像以提高 apt update 速度，如有需要可自行取消注释
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-updates main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-updates main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-backports main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-backports main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-security main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-security main restricted universe multiverse

# 预发布软件源，不建议启用
# deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-proposed main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-proposed main restricted universe multiverse
```

本镜像仅包含 32/64 位 x86 架构处理器的软件包，在 ARM(arm64, armhf)、PowerPC(ppc64el)、RISC-V(riscv64) 和 S390x 等架构的设备上（对应官方源为ports.ubuntu.com）请使用 ubuntu-ports 镜像。

c. 需要替换的**/etc/apt/sources.list** 文件的内容为

test@test:~$ **sudo mv /etc/apt/sources.list cat /etc/apt/sources.list.bak**
test@test:~$ **sudo vim /etc/apt/sources.list**

```
# 默认注释了源码镜像以提高 apt update 速度，如有需要可自行取消注释
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-updates main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-updates main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-backports main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-backports main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-security main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-security main restricted universe multiverse

# 预发布软件源，不建议启用
# deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-proposed main restricted universe multiverse
```

```
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ bionic-proposed main restricted universe multiverse
```

d. 替换完后需要更新下包信息，并确保没有报错

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

e. 另外，由于内核和 U-boot 等源码都是存放在 GitHub 上的，所以编译镜像的时候请确保电脑能正常从 GitHub 下载代码，这点是非常重要的

5.2. 获取 linux sdk 的源码

5.2.1. 从 github 下载 orangepi-build

1) linux sdk 其实指的就是 orangepi-build 这套代码，orangepi-build 是基于 armbian build 编译系统修改而来的，使用 orangepi-build 可以编译出多个版本的 linux 镜像。首先下载 orangepi-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

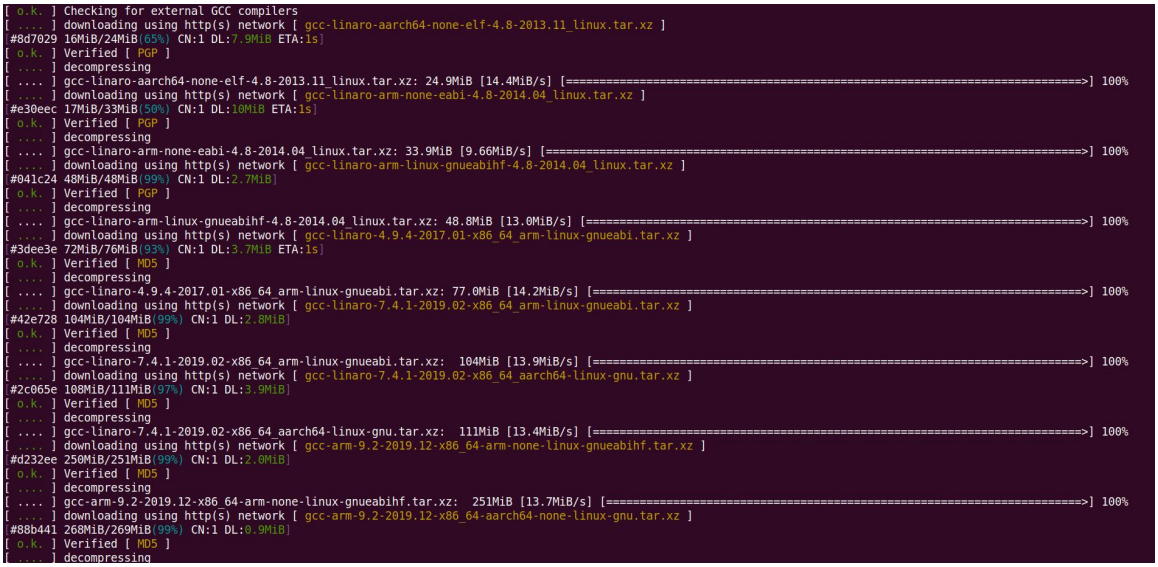
e. **scripts**: 编译 linux 镜像的通用脚本

```
test@test:~/orangepi-build$ ls
build.sh  external  LICENSE  README.md  scripts
```

如果是从 github 下载的 orangepi-build 的代码，下载完后可能会发现 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



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

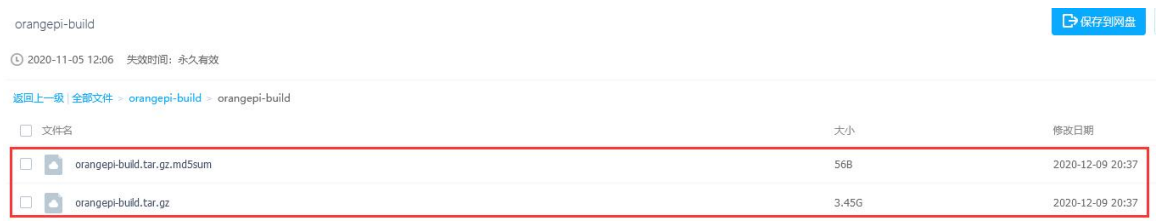
2) 百度云盘的 orangepi-build 文件夹下有两个文件

a. **orangepi-build.tar.gz** 为 orangepi-build 源码的压缩包

b. **orangepi-build.tar.gz.md5sum** 为 orangepi-build 源码的压缩包的 MD5 校验和文件

c. 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
```



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**

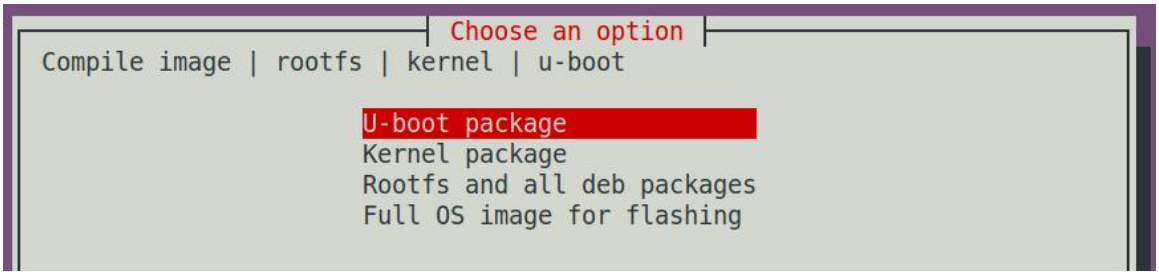
5.3. 编译 u-boot

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。

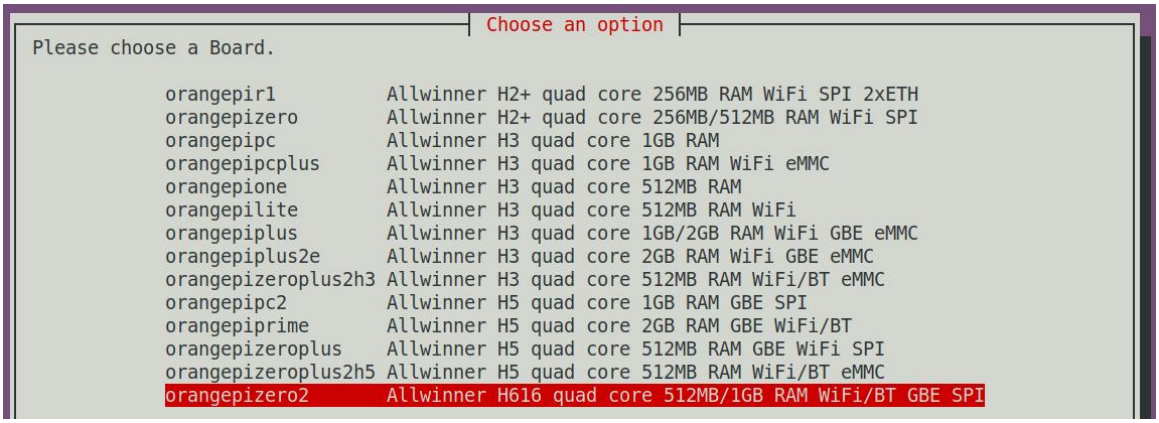
1) 运行 build.sh 脚本，记得加 sudo 权限

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

2) 选择 **U-boot package**，然后回车



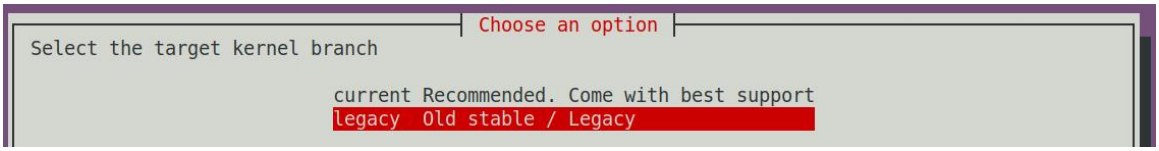
3) 接着选择开发板的型号



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_2.2.0_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=orangepizero2 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_2.2.0_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_2.2.0_arm64.deb .    (注意命令最后有个 “.”)
test@test:~/orangepi_build/output/debs/u-boot$ ls
linux-u-boot-legacy-orangepizero2_2.2.0_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_2.2.0_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-bulid 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_2.2.0_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
```

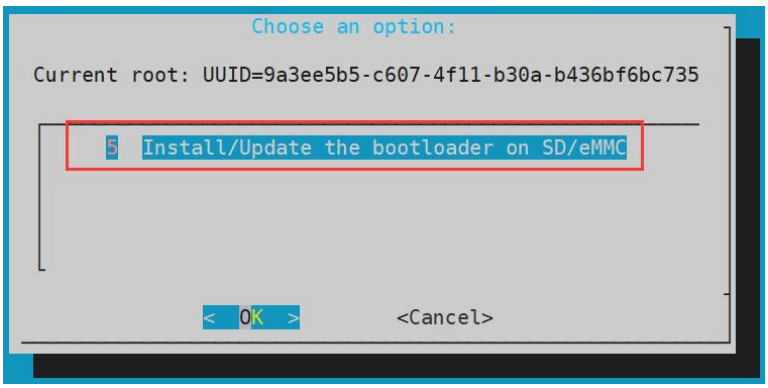
c. 再安装刚才上传的新的 u-boot 的 deb 包

```
root@orangepi:~# dpkg -i linux-u-boot-legacy-orangepizero2_2.2.0_arm64.deb
```

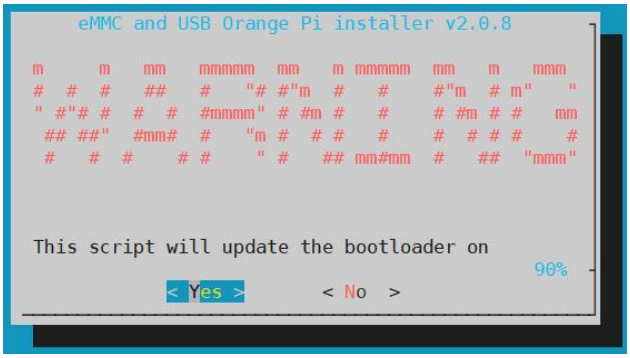
d. 然后运行 nand-sata-install 脚本

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

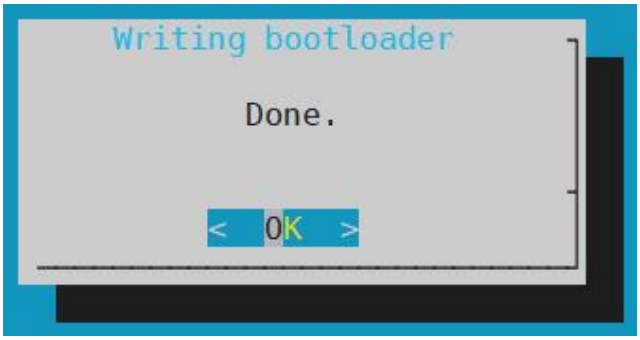
e. 然后选择 **5 Install/Update the bootloader on SD/eMMC**



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



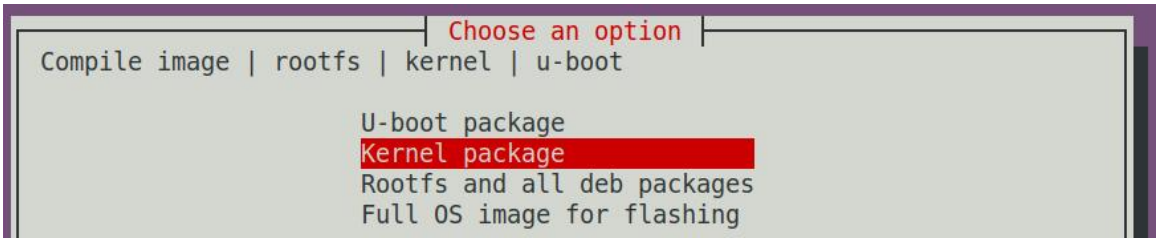
h. 然后就可以重启开发板来测试 u-boot 的修改是否生效了

5.4. 编译 linux 内核

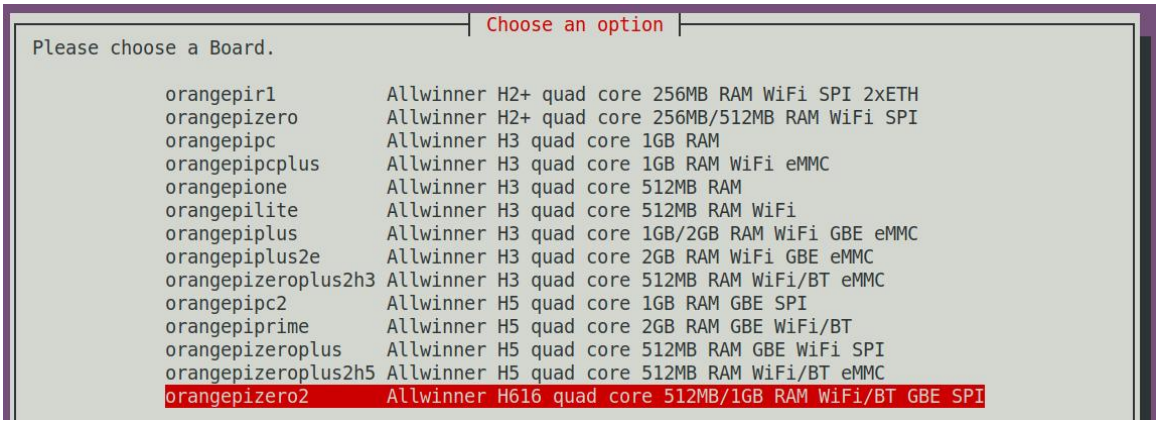
1) 运行 build.sh 脚本，记得加 sudo 权限

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

2) 选择 **Kernel package**，然后回车



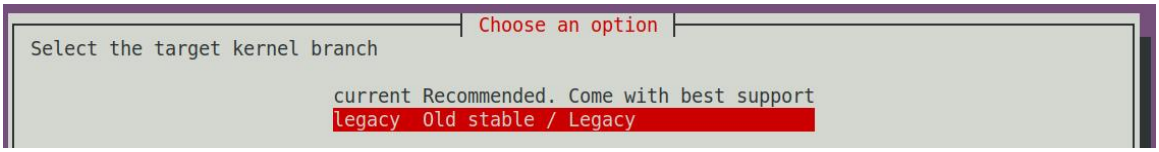
3) 接着选择开发板的型号



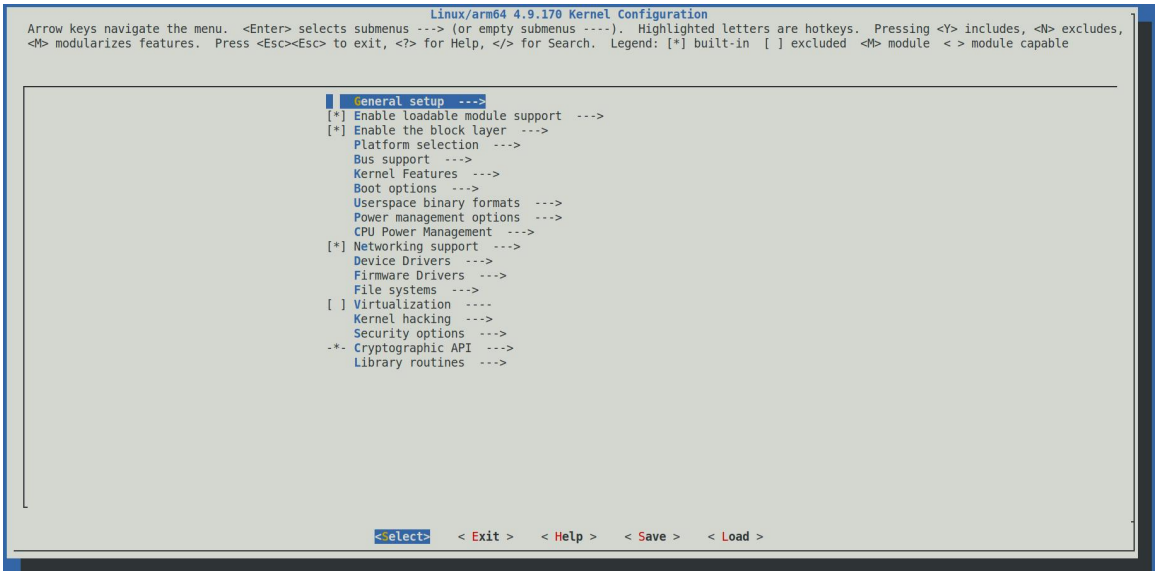
4) 然后选择分支

a. current 会编译 linux 5.13

b. 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



a. 如果不需要修改内核的配置选项，在运行 build.sh 脚本时，传入 **KERNEL_CONFIGURE=no** 就可临时屏蔽弹出内核的配置界面了

```
test@test:~/orangepi-build$ sudo ./build.sh KERNEL_CONFIGURE=no
```

b. 也可以设置 orangepi-build/userpatches/config-default.conf 配置文件中的 **KERNEL_CONFIGURE=no**，这样可以永久禁用这个功能

c. 编译内核的时候如果提示下面的错误，这是由于 Ubuntu PC 的终端界面太小，导致 make menuconfig 的界面无法显示，请把 Ubuntu PC 的终端调到最大，然后重新运行 build.sh 脚本

```
  HOSTCC  scripts/kconfig/mconf.o
  HOSTCC  scripts/kconfig/lxdialog/checklist.o
  HOSTCC  scripts/kconfig/lxdialog/util.o
  HOSTCC  scripts/kconfig/lxdialog/inputbox.o
  HOSTCC  scripts/kconfig/lxdialog/textbox.o
  HOSTCC  scripts/kconfig/lxdialog/yesno.o
  HOSTCC  scripts/kconfig/lxdialog/menubox.o
  HOSTLD  scripts/kconfig/mconf
scripts/kconfig/mconf  Kconfig
Your display is too small to run Menuconfig!
It must be at least 19 lines by 80 columns.
scripts/kconfig/Makefile:28: recipe for target 'menuconfig' failed
make[1]: *** [menuconfig] Error 1
Makefile:560: recipe for target 'menuconfig' failed
make: *** [menuconfig] Error 2
[ error ] ERROR in function compile_kernel [ compilation.sh:376 ]
[ error ] Error kernel menuconfig failed
[ o.k. ] Process terminated
```

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_2.2.0_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=orangepizero2
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_2.2.0_arm64.deb** 暂未使用，先不用关心

b. **linux-headers-legacy-sun50iw9_2.2.0_arm64.deb** 包含内核头文件

c. **linux-image-legacy-sun50iw9_2.2.0_arm64.deb** 包含内核镜像和内核模块

```
test@test:~/orangepi-build$ ls output/debs/linux-*
output/debs/linux-dtb-legacy-sun50iw9_2.2.0_arm64.deb
output/debs/linux-headers-legacy-sun50iw9_2.2.0_arm64.deb
output/debs/linux-image-legacy-sun50iw9_2.2.0_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:~/orangepi-build$ cd output/debs
test@test:~/orangepi_build/output/debs$ mkdir test
test@test:~/orangepi_build/output/debs$ cp \
linux-image-legacy-sun50iw9_2.2.0_arm64.deb test/
test@test:~/orangepi_build/output/debs$ cd test
test@test:~/orangepi_build/output/debs/test$ dpkg -x \
```

```
linux-image-legacy-sun50iw9_2.2.0_arm64.deb .
test@test:~/orangepi_build/output/debs/test$ ls
boot  etc  lib  linux-image-legacy-sun50iw9_2.2.0_arm64.deb  usr
```

b. 解压后的文件如下所示

```
test@test:~/orangepi_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_2.2.0_arm64.deb
└── usr
    ├── lib
    └── share

8 directories, 4 files
```

9) orangepi-bulid 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_2.2.0_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_2.2.0_arm64.deb
```

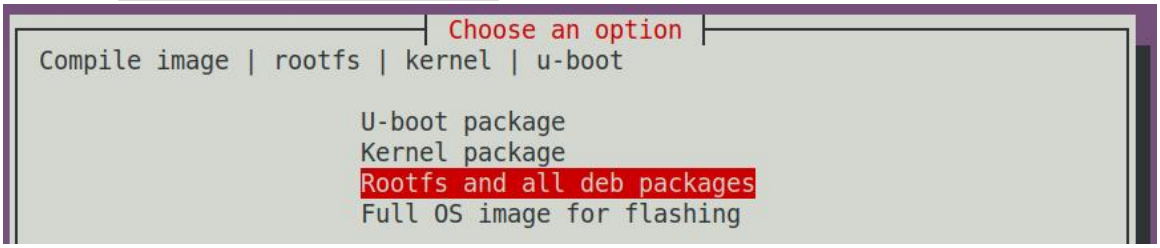
d. 然后重启开发板，再查看内核相关的修改是否已生效

5.5. 编译 rootfs

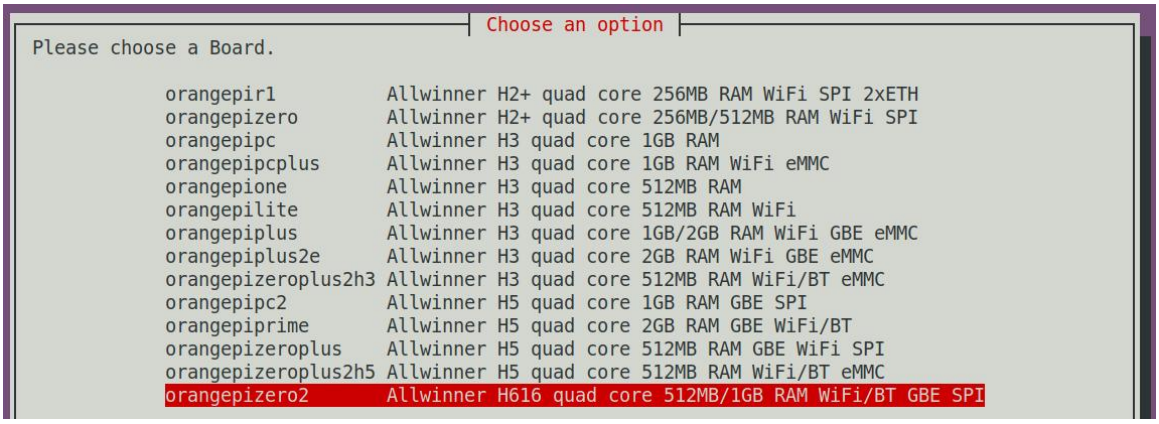
1) 运行 build.sh 脚本，记得加 sudo 权限

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

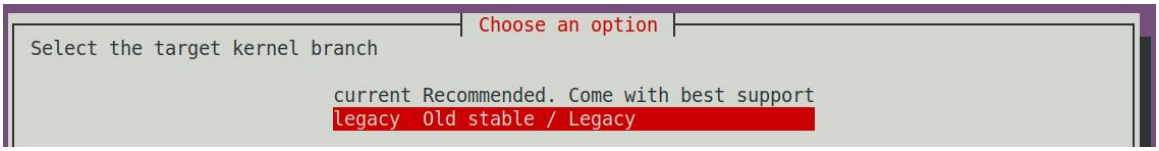
2) 选择 **Rootfs and all deb packages**，然后回车



3) 接着选择开发板的型号

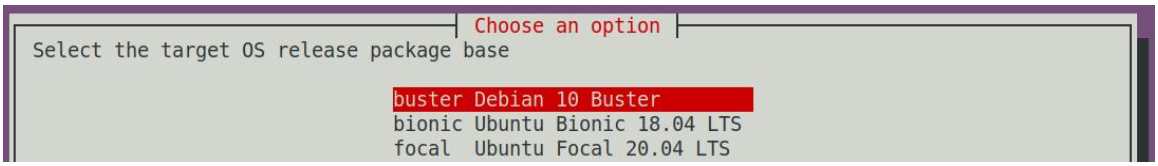


4) 然后选择分支类型

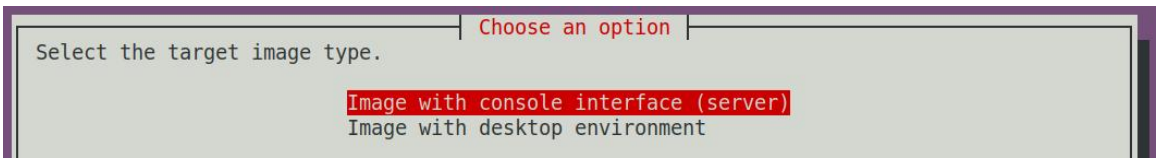


5) 然后选择 rootfs 的类型

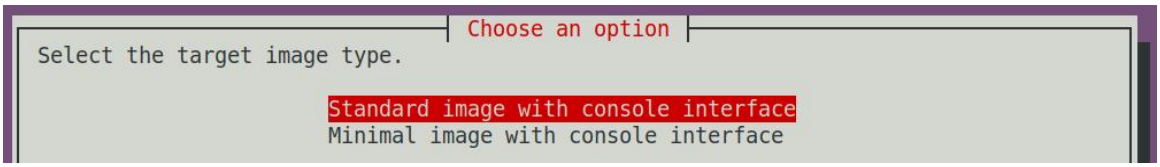
a. **buster** 表示 Debian 10
b. **bionic** 表示 Ubuntu 18.04
c. **focal** 表示 Ubuntu 20.04



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 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=orangepizero2**

```
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 **dekstop** 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

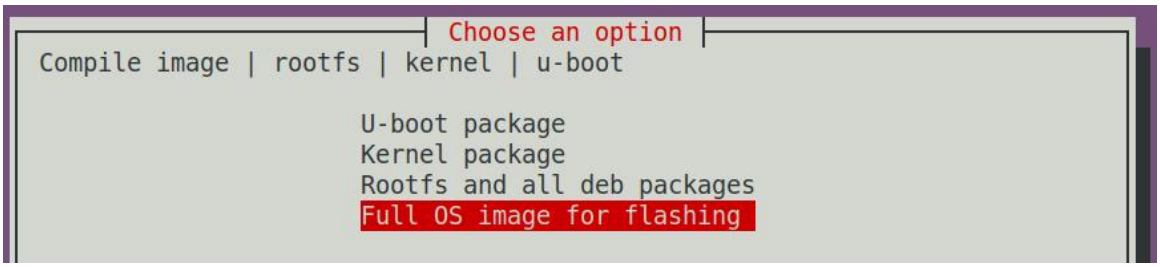
```
链接：https://pan.baidu.com/s/1vWQmCmSYdH7iCDFyKpJtVw
提取码：zero
```

5.6. 编译 linux 镜像

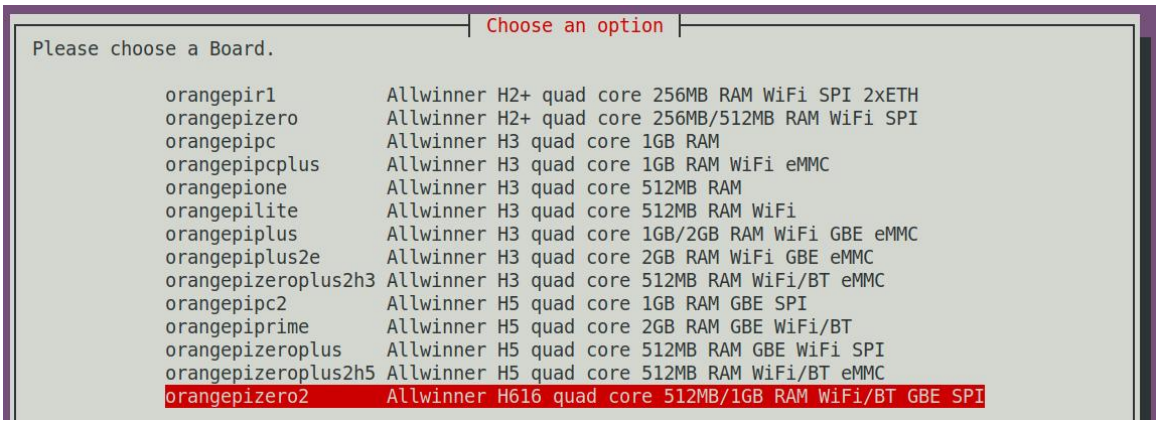
1) 运行 build.sh 脚本，记得加 sudo 权限

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

2) 选择 **Full OS image for flashing**，然后回车



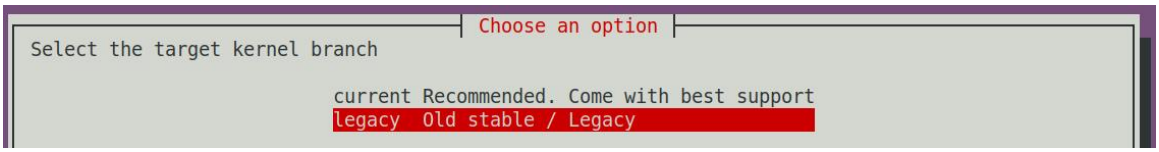
3) 然后选择开发板的型号



4) 然后选择分支

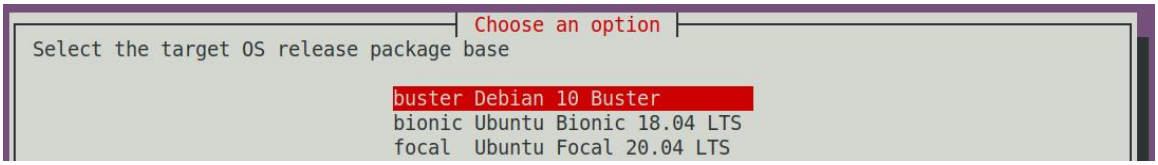
a. current 会编译 linux 5.13

b. legacy 会编译 linux4.9



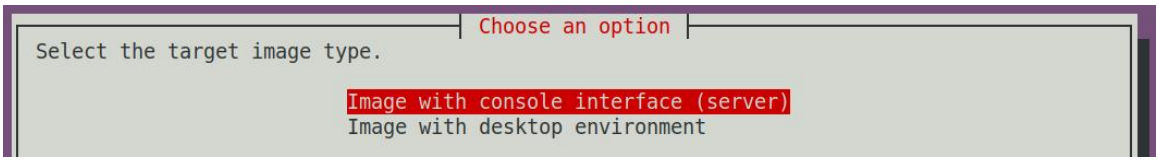
5) 然后选择 rootfs 的类型

a. **buster** 表示 Debian 10
b. **bionic** 表示 Ubuntu 18.04
c. **focal** 表示 Ubuntu 20.04

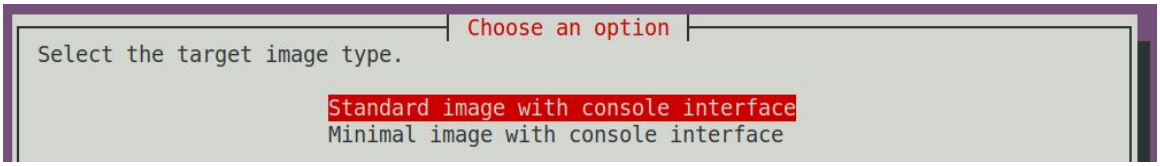


6) 然后选择镜像的类型

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

a. 初始化 Ubuntu PC 的编译环境，安装编译过程需要的软件包
b. 下载 u-boot 和 linux 内核的源码（如果已经缓存，则只更新代码）
c. 编译 u-boot 源码，生成 u-boot 的 deb 包
d. 编译 linux 源码，生成 linux 相关的 deb 包
e. 制作 linux firmware 的 deb 包
f. 制作 orangepi-config 工具的 deb 包
g. 制作板级支持的 deb 包

h. 如果是编译 desktop 版镜像，还会制作 desktop 相关的 deb 包
i. 检查 rootfs 是否已经缓存，如果没有缓存，则重新制作 rootfs，如果已经缓存，则直接解压使用
j. 安装前面生成的 deb 包到 rootfs 中
k. 对不同的开发板和不同类型镜像做一些特定的设置，如预装额外的软件包，修改系统配置等
l. 然后制作镜像文件，并格式化分区，默认类型为 ext4
m. 再将配置好的 rootfs 拷贝到镜像的分区中
n. 然后更新 initramfs
o. 最后将 u-boot 的 bin 文件通过 dd 命令写入到镜像中

9) 编译完镜像后会提示下面的信息

a. 编译生成的镜像的存放路径

```
[ o.k. ] Done building
[ output/images/orangepizero2_2.2.0_ubuntu_bionic_server_linux4.9.170/orangepizero2_2.2.0_ubuntu_bionic_server_linux4.9.170.img ]
```

b. 编译使用的时间

```
[ o.k. ] Runtime [ 19 min ]
```

c. 重复编译镜像的命令，使用下面的命令无需通过图形界面选择，可以直接开始编译镜像

```
[ o.k. ] Repeat Build Options [ sudo ./build.sh  BOARD=orangepizero2 BRANCH=legacy BUILD_OPT=image RELEASE=bionic BUILD_MINIMAL=no BUILD_DESKTOP=no KERNEL_CONFIGURE=yes ]
```

6. Linux SDK——新版 orangepi-build 使用说明

1. 新版本和旧版本的编译主机的区别？

旧版本的 orangepi-build 是在 Ubuntu18.04 的 x64 电脑或者虚拟机上运行的。
新版本的 orangepi-build 是在 Ubuntu22.04 的 x64 电脑或者虚拟机上运行的。

如果想使用本章所说的新版本 orangepi-build 来编译 Linux 镜像，首先需要准备一个安装有 Ubuntu22.04 的电脑或者虚拟机。

2. 新版本和旧版本源码存放地址的区别？

旧版本的 orangepi-build 源码存放在 orangepi-build 仓库的 main 分支：
https://github.com/orangepi-xunlong/orangepi-build/tree/main
新版本的 orangepi-build 源码存放在 orangepi-build 仓库的 next 分支：
https://github.com/orangepi-xunlong/orangepi-build/tree/next

3. 新版本和旧版本支持内核的区别？

旧版本的 orangepi-build 主要支持 Linux4.9 和 Linux5.13（5.13 不再更新）。
新版本的 orangepi-build 目前支持 Linux5.16。

4. 新版本和旧版本支持的 Linux 发行版类型的区别？

旧版本的 orangepi-build 主要支持 Debian10、Ubuntu18.04、Ubuntu20.04。
新版本的 orangepi-build 主要支持 Debian11/12、Ubuntu20.04、Ubuntu22.04。

5. 新版本和旧版本生成的镜像名字的区别？

旧版本的 orangepi-build 编译生成的镜像版本名为 2.xx。
新版本的 orangepi-build 编译生成的镜像版本名为 3.xx。桌面版镜像还会添加桌面类型的字段，如 xfce。

6.1. 编译系统需求

1) Linux SDK，即 **orangepi-build**，只支持在安装有 **Ubuntu 22.04** 的电脑上运行，所以下载 orangepi-build 前，请首先确保自己电脑已安装的 Ubuntu 版本是 Ubuntu 22.04。查看电脑已安装的 Ubuntu 版本的命令如下所示，如果 Release 字段显示的不是 **22.04**，说明当前使用的 Ubuntu 版本不符合要求，请更换系统后再进行下面的操作。

```
test@test:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04 LTS
Release:        22.04
Codename:       jammy
```

2) 如果电脑安装的是 Windows 系统，没有安装有 Ubuntu 22.04 的电脑，可以考虑使用 **VirtualBox** 或者 **VMware** 来在 Windows 系统中安装一个 Ubuntu 22.04 虚拟机。但是请注意，不要在 WSL 虚拟机上编译 orangepi-build，因为 orangepi-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 orangepi-build，另外请不要在**开发板**的 Linux 系统中使用 orangepi-build。

3) Ubuntu 22.04 **amd64** 版本的安装镜像下载地址为：

```
https://mirrors.tuna.tsinghua.edu.cn/ubuntu-releases/22.04/ubuntu-22.04-desktop-amd64.iso
```

4) 在电脑中或者虚拟机中安装完 Ubuntu 22.04 后，请先设置 Ubuntu 22.04 的软件源为清华源，不然后面安装软件的时候很容易由于网络原因而出错

a. 替换清华源的方法参考这个网页的说明即可

```
https://mirrors.tuna.tsinghua.edu.cn/help/ubuntu/
```

b. 注意 Ubuntu 版本需要切换到 22.04

Ubuntu 镜像使用帮助

Ubuntu 的软件源配置文件是 /etc/apt/sources.list。将系统自带的该文件做个备份，将该文件替换为下面内容，即可使用 TUNA 的软件源镜像。

选择你的ubuntu版本: 22.04 LTS

```
# 默认注释了源码镜像以提高 apt update 速度，如有需要可自行取消注释
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-updates main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-updates main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-backports main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-backports main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-security main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-security main restricted universe multiverse

# 预发布软件源，不建议启用
# deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-proposed main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-proposed main restricted universe multiverse
```

c. 需要替换的**/etc/apt/sources.list** 文件的内容为

```
test@test:~$ sudo mv /etc/apt/sources.list cat /etc/apt/sources.list.bak
test@test:~$ sudo vim /etc/apt/sources.list
# 默认注释了源码镜像以提高 apt update 速度，如有需要可自行取消注释
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-updates main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-updates main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-backports main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-backports main restricted universe multiverse
deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-security main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-security main restricted universe multiverse


# 预发布软件源，不建议启用
# deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-proposed main restricted universe multiverse
# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-proposed main restricted universe multiverse
```

d. 替换完后需要更新下包信息，并确保没有报错

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

e. 另外，由于内核和 U-boot 等源码都是存放在 GitHub 上的，所以编译镜像的时候请确保电脑能正常从 GitHub 下载代码，这点是非常重要的。

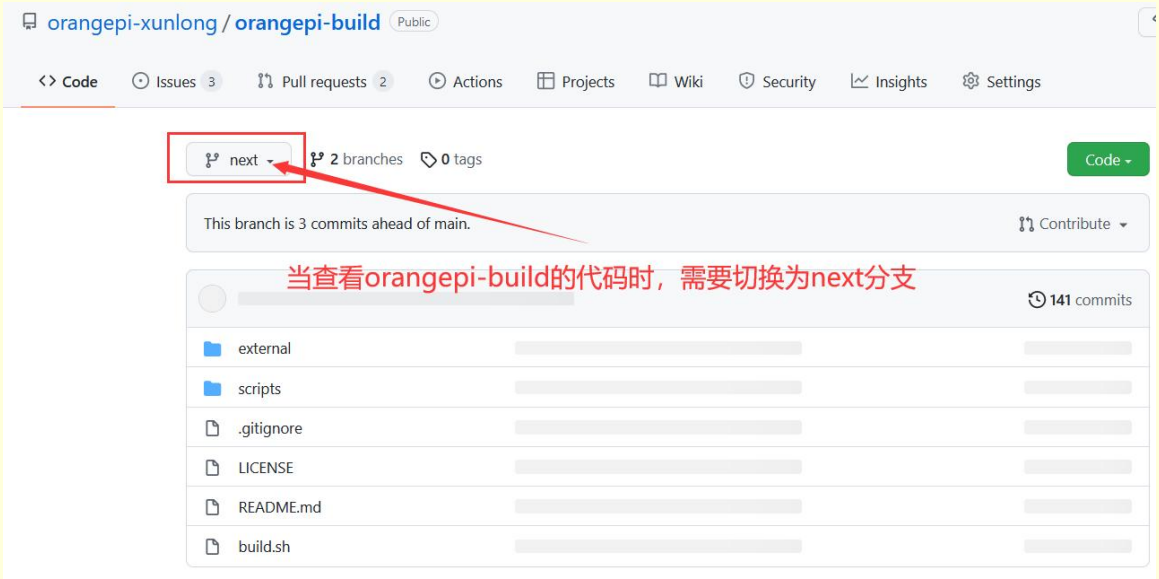
6.2. 获取 linux sdk 的源码

6.2.1. 从 github 下载 orangepi-build

1) linux sdk 其实指的就是 orangepi-build 这套代码，orangepi-build 是基于 armbian build 编译系统修改而来的，使用 orangepi-build 可以编译出多个版本的 linux 镜像。首先下载 orangepi-build 的代码，目前 Linux SDK 中的 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 install git
test@test:~$ git clone https://github.com/orangepi-xunlong/orangepi-build.git -b next
```

注意，此小节的说明是针对 orangepi-build 的 next 分支源码的，上面的 git clone 命令需要指定 orangepi-build 源码的分支为 next。



通过 git clone 命令下载 orangepi-build 的代码是不需要输入 github 账号的用户名和密码的（下载本手册中的其他代码也是一样的），如果如输入 git clone 命令后 Ubuntu PC 提示需要输入 github 账号的用户名和密码，一般都是 git clone 后面的 orangepi-build 仓库的地址输入错误了，请仔细检查命令拼写是否有错误，而不是以为我们这里忘了提供 github 账号的用户名和密码。

2) H616 系列开发板当前使用的 u-boot 和 linux 内核版本如下所示

分支	u-boot 版本	linux 内核版本
current	**u-boot 2021.10**	**linux5.16**

3) orangepi-build 下载完后会包含下面的文件和文件夹

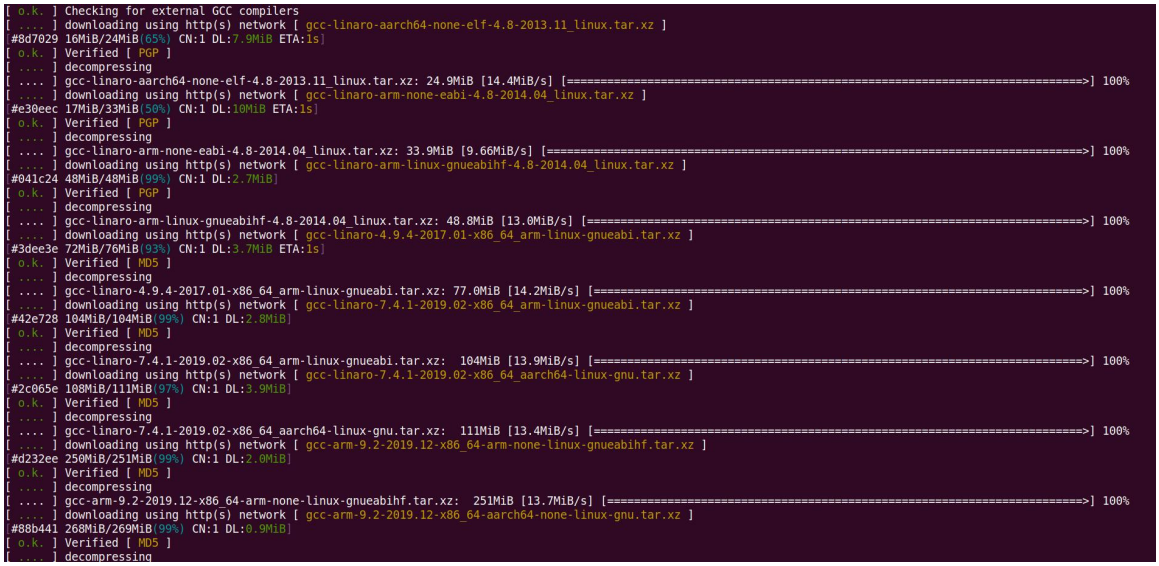
f. **build.sh**: 编译启动脚本
g. **external**: 包含编译镜像需要用的配置文件、特定的脚本以及部分程序的源码等
h. **LICENSE**: GPL 2 许可证文件
i. README.md: orangepi-build 说明文件
j. **scripts**: 编译 linux 镜像的通用脚本

```
test@test:~/orangepi-build$ ls
build.sh  external  LICENSE  README.md  scripts
```

如果是从 github 下载的 orangepi-build 的代码，下载完后你可能会发现 orangepi-build 中并没有包含 u-boot 和 linux 内核的源码，也没有编译 u-boot 和 linux 内核需要用到交叉编译工具链，这是正常的，因为这些东西都存放在其它单独的 github 仓库或者某些服务器上了（下文会详述其地址）。orangepi-build 在脚本和配置文件中会指定 u-boot、linux 内核和交叉编译工具链的地址，运行 orangepi-build 时，当其发现本地没有这些东西，会自动去相应的地方下载的。

6.2.2. 下载交叉编译工具链

1) orangepi-build 第一次运行的时候会自动下载交叉编译工具链放在 **toolchains** 文件夹中，每次运行 orangepi-build 的 build.sh 脚本后，都会检查 **toolchains** 中的交叉编译工具链是否都存在，如果不存在则会重新开始下载，如果存在则直接使用，不会重复下载



2) 交叉编译工具链在中国境内的镜像网址为清华大学的开源软件镜像站

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

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. linux5.16

```
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. v2021.10

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

6.2.3. orangepi-build 完整目录结构说明

1) orangepi-build 仓库下载完后并不包含 linux 内核、u-boot 的源码以及交叉编译工具链，linux 内核和 u-boot 的源码存放在独立的 git 仓库中

a. linux 内核源码存放的 git 仓库如下，注意切换 linux-orangepi 仓库的分支为
   a) Linux5.16

```
https://github.com/orangepi-xunlong/linux-orangepi/tree/orange-pi-5.16-sunxi64
```

b. u-boot 源码存放的 git 仓库如下，注意切换 u-boot-orangepi 仓库的分支为

```
https://github.com/orangepi-xunlong/u-boot-orangepi/tree/v2021.10-sunxi
```

如果对 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) orangepi-build 第一次运行的时候会去下载交叉编译工具链、u-boot 和 linux 内核源码，成功编译完一次 linux 镜像后在 orangepi-build 中可以看到的文件和文件夹有

a. **build.sh**: 编译启动脚本
b. **external**: 包含编译镜像需要用的配置文件、特定功能的脚本以及部分程序的源码，编译镜像过程中缓存的 rootfs 压缩包也存放在 external 中
c. **kernel**: 存放 linux 内核的源码，里面名为 **orange-pi-5.16-sunxi64** 的文件夹存放的就是 H616 系列开发板 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 **v2021.10-sunxi** 的文件夹存放的就是 H616 系列开发板的 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

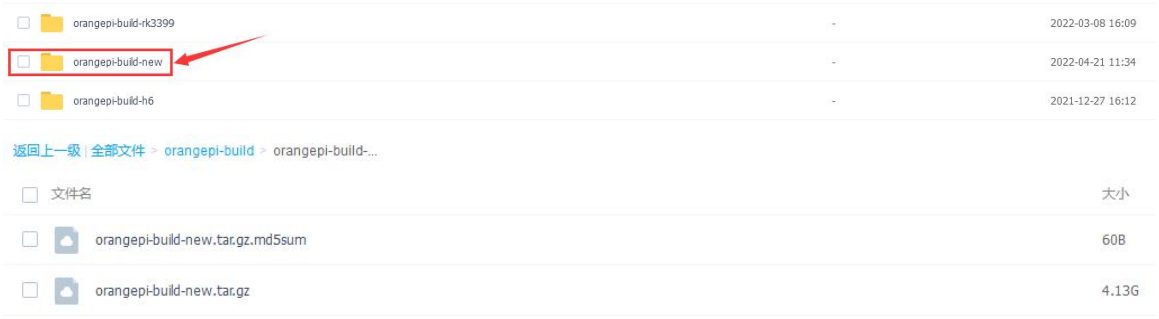
```
test@test:~/orangepi-build$ ls
```

```
build.sh  external  kernel  LICENSE  output  README.md  scripts  toolchains
u-boot  userpatches
```

6.2.4. 从百度云盘下载 orangepi-build

1) 如果从 github 下载 orangepi-build 的速度很慢，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为

链接：**https://pan.baidu.com/s/1vWQmCmSYdH7iCDFyKpJtVw**
提取码：zero



2) 百度云盘的 orangepi-build-new 文件夹下有两个文件

a. **orangepi-build-new.tar.gz** 为 orangepi-build 源码的压缩包
b. **orangepi-build-new.tar.gz.md5sum** 为 orangepi-build 源码的压缩包的 MD5 校验和文件
c. 下载完后，请首先检查下 orangepi-build-new.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-new.tar.gz.md5sum
orangepi-build-new.tar.gz: 成功
```

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

```
test@test:~$ tar -zxf orangepi-build-new.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
```

5) 百度云盘上的 **orangepi-build-new.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，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

6) 百度云盘上的 **orangepi-build-new.tar.gz** 解压后的 kernel 文件夹会包含内核源码，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。H616 系列开发板使用到的内核源码有：

orange-pi-5.16-sunxi64	**H616 系列开发板 current 分支使用**

7) 百度云盘上的 **orangepi-build-new.tar.gz** 解压后的 u-boot 文件夹中会包含 u-boot 源码，请不要修改 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。H616 系列开发板使用到的 u-boot 源码有：

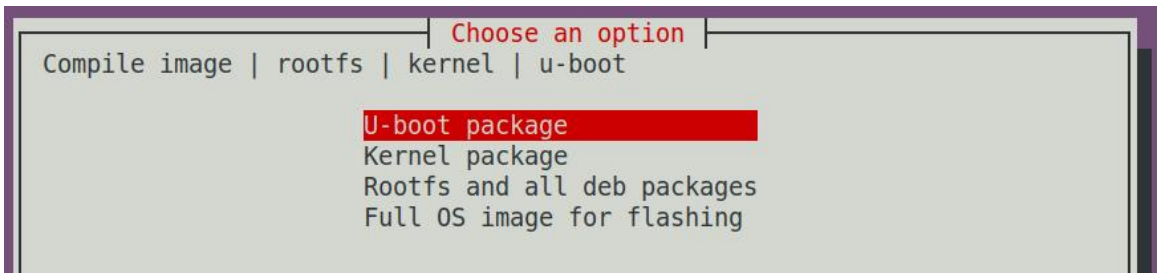
v2021.10-sunxi	**H616 系列开发板 current 分支使用**

6.3. 编译 u-boot

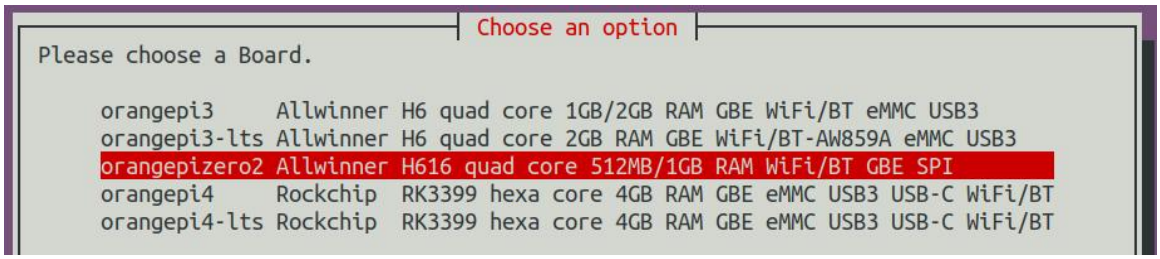
1) 运行 build.sh 脚本，记得加 sudo 权限

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

2) 选择 **U-boot package**，然后回车



3) 接着选择开发板的型号



4) 然后就会开始编译 u-boot，编译时提示的部分信息说明如下

a. u-boot 源码的版本

[o.k.] Compiling u-boot [**v2021.10**]

b. 交叉编译工具链的版本

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

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-current-orangepizero2_3.0.0_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=orangepizero2 BRANCH=current BUILD_OPT=u-boot KERNEL_CONFIGURE=yes**]

5) 查看编译生成的 u-boot deb 包

```
test@test:~/orangepi-build$ ls output/debs/u-boot/
linux-u-boot-current-orangepizero2_3.0.0_arm64.deb
```

6) 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-current-orangepizero2_3.0.0_arm64.deb .    (注意命令最后有个“.”)
test@test:~/orangepi_build/output/debs/u-boot$ ls
linux-u-boot-current-orangepizero2_3.0.0_arm64.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-current-orangepizero2_3.0.0_arm64
    │   └── u-boot-sunxi-with-spl.bin
    └── u-boot
        ├── LICENSE
        ├── orangepi_zero2_defconfig
        └── platform_install.sh
```

7) orangepi-bulid 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，如果是从百度云盘下载的源码压缩包，则没有这个问题，因为 u-boot 的源码都已缓存好了**），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"
```

8) 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-current-orangepizero2_3.0.0_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-current
```

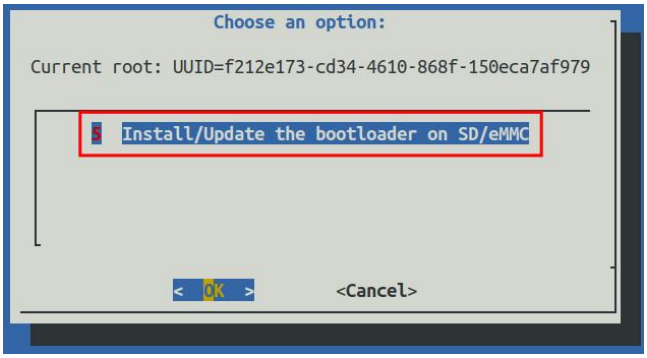
c. 再安装刚才上传的新的 u-boot 的 deb 包

```
root@orangepi:~# dpkg -i linux-u-boot-current-orangepizero2_3.0.0_arm64.deb
```

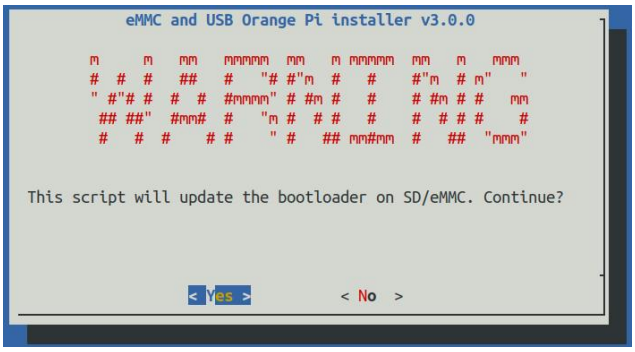
d. 然后运行 nand-sata-install 脚本

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

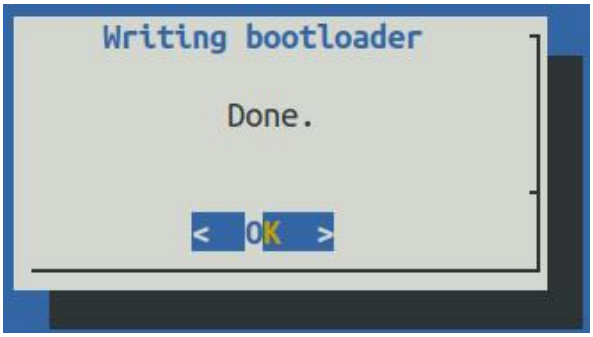
e. 然后选择 **5 Install/Update the bootloader on SD/eMMC**



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



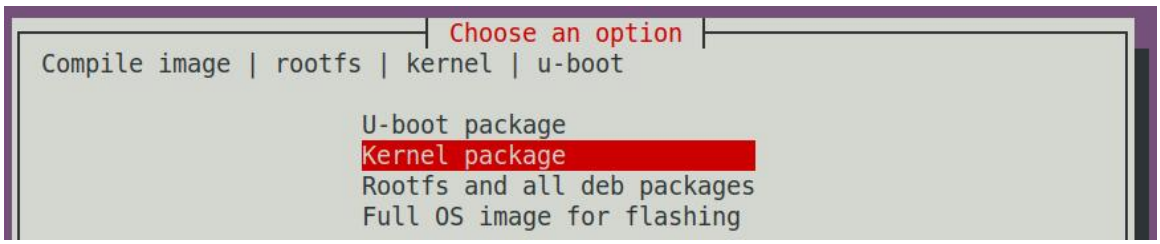
h. 然后就可以重启开发板来测试 u-boot 的修改是否生效了

6.4. 编译 linux 内核

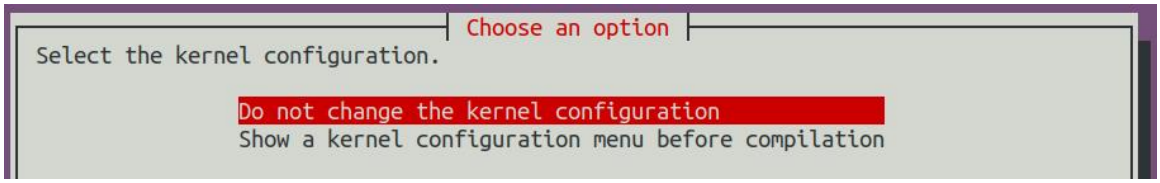
1) 运行 build.sh 脚本，记得加 sudo 权限

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

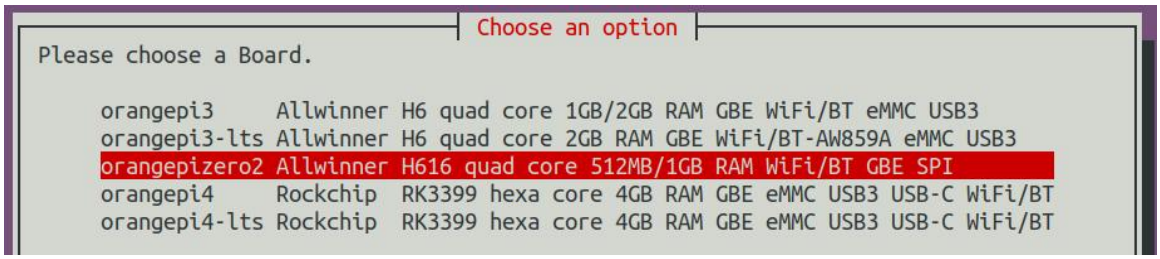
2) 选择 **Kernel package**，然后回车



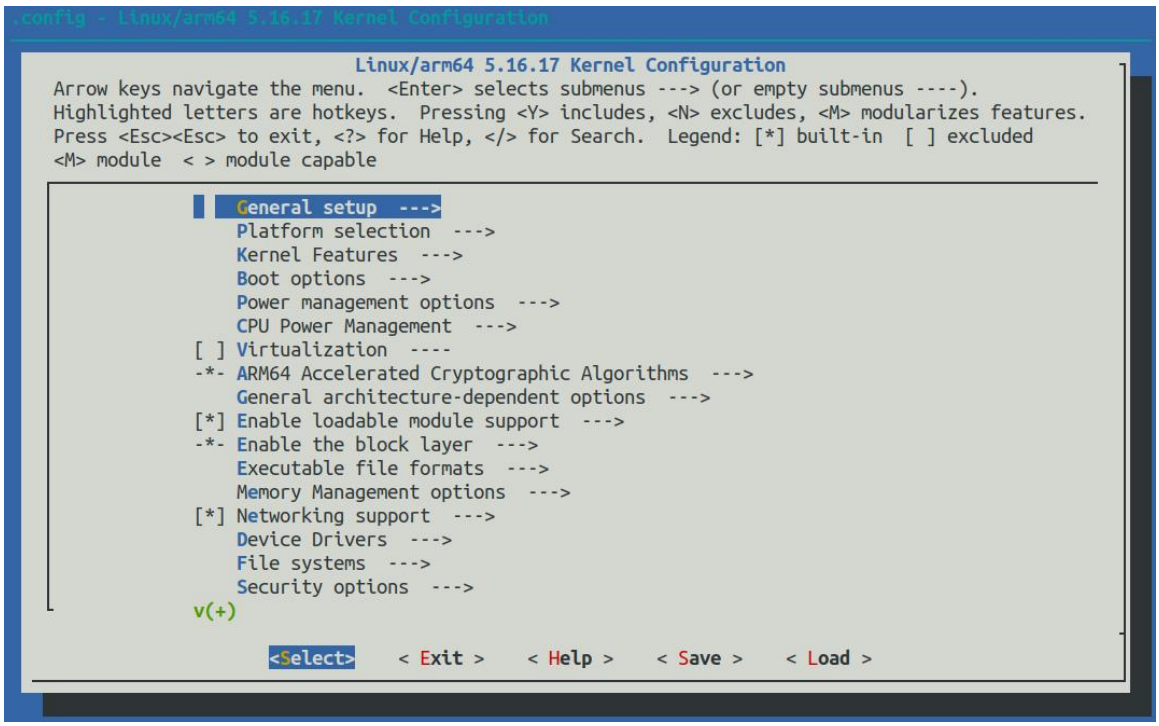
3) 然后会提示是否需要显示内核配置界面，如果不需要修改内核配置，则选择第一个即可，如果需要修改内核配置，则选择第二个



4) 接着选择开发板的型号



5) 如果第二步选择了需要显示内核配置菜单（第二个选项），则会弹出通过 **make menuconfig** 打开的内核配置的界面，此时可以直接修改内核的配置，修改完后再保存退出即可，退出后会开始编译内核源码



a. 如果不需要修改内核的配置选项，在运行 build.sh 脚本时，传入 **KERNEL_CONFIGURE=no** 就可临时屏蔽弹出内核的配置界面了

```
test@test:~/orangepi-build$ sudo ./build.sh KERNEL_CONFIGURE=no
```

b. 也可以设置 orangepi-build/userpatches/config-default.conf 配置文件中的 **KERNEL_CONFIGURE=no**，这样可以永久禁用这个功能

c. 编译内核的时候如果提示下面的错误，这是由于 Ubuntu PC 的终端界面太小，导致 make menuconfig 的界面无法显示，请把 Ubuntu PC 的终端调到最大，然后重新运行 build.sh 脚本

```
  HOSTCC  scripts/kconfig/mconf.o
  HOSTCC  scripts/kconfig/lxdialog/checklist.o
  HOSTCC  scripts/kconfig/lxdialog/util.o
  HOSTCC  scripts/kconfig/lxdialog/inputbox.o
  HOSTCC  scripts/kconfig/lxdialog/textbox.o
  HOSTCC  scripts/kconfig/lxdialog/yesno.o
  HOSTCC  scripts/kconfig/lxdialog/menubox.o
  HOSTLD  scripts/kconfig/mconf
scripts/kconfig/mconf  Kconfig
Your display is too small to run Menuconfig!
It must be at least 19 lines by 80 columns.
scripts/kconfig/Makefile:28: recipe for target 'menuconfig' failed
make[1]: *** [menuconfig] Error 1
Makefile:560: recipe for target 'menuconfig' failed
make: *** [menuconfig] Error 2
[ error ] ERROR in function compile_kernel [ compilation.sh:376 ]
[ error ] Error kernel menuconfig failed
[ o.k. ] Process terminated
```

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 current kernel [ 5.16.17 ]
```

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/linux-5.16-sun50iw9-current.config ]
```

d. 编译生成的内核相关的 deb 包的路径

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

e. 编译生成的内核镜像 deb 包的包名

```
[ o.k. ] File name [ linux-image-current-sun50iw9_3.0.0_arm64.deb ]
```

f. 编译使用的时间

```
[ o.k. ] Runtime [ 5 min ]
```

g. 最后会显示重复编译上一次选择的内核的编译命令，使用下面的命令无需通过图形界面选择，可以直接开始编译内核源码

```
[ o.k. ] Repeat Build Options [ sudo ./build.sh  BOARD=orangepizero2
BRANCH=current BUILD_OPT=kernel KERNEL_CONFIGURE=yes ]
```

7) 查看编译生成的内核相关的 deb 包

a. **linux-dtb-current-sun50iw9_3.0.0_arm64.deb** 包含有内核使用的 dtb 文件

b. **linux-headers-current-sun50iw9_3.0.0_arm64.deb** 包含内核头文件

c. **linux-image-current-sun50iw9_3.0.0_arm64.deb** 包含内核镜像和内核模块

```
test@test:~/orangepi-build$ ls output/debs/linux-*
output/debs/linux-dtb-current-sun50iw9_3.0.0_arm64.deb
output/debs/linux-image-current-sun50iw9_3.0.0_arm64.deb
output/debs/linux-headers-current-sun50iw9_3.0.0_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:~/orangepi-build$ cd output/debs
test@test:~/orangepi_build/output/debs$ mkdir test
test@test:~/orangepi_build/output/debs$ cp \
linux-image-current-sun50iw9_3.0.0_arm64.deb test/
test@test:~/orangepi_build/output/debs$ cd test
test@test:~/orangepi_build/output/debs/test$ dpkg -x \
linux-image-current-sun50iw9_3.0.0_arm64.deb .
test@test:~/orangepi_build/output/debs/test$ ls
```

```
boot  etc  lib  linux-image-current-sun50iw9_3.0.0_arm64.deb  usr
```

b. 解压后的文件如下所示

```
test@test:~/orangepi-build/output/debs/test$ tree -L 2
.
├── boot
│   ├── config-5.16.17-sun50iw9
│   ├── System.map-5.16.17-sun50iw9
│   └── vmlinuz-5.16.17-sun50iw9
├── etc
│   └── kernel
├── lib
│   └── modules
├── linux-image-current-sun50iw9_3.0.0_arm64.deb
└── usr
    ├── lib
    └── share

8 directories, 4 files
```

9) orangepi-bulid 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，如果是从百度云盘下载的源码压缩包，则没有这个问题，因为 linux 的源码都已缓存好了**），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-current-sun50iw9_3.0.0_arm64.deb root@192.168.1.xxx:/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-current-sun50iw9
```

c. 再安装刚才上传的新的 linux 内核的 deb 包

```
root@orangepi:~# dpkg -i linux-image-current-sun50iw9_3.0.0_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

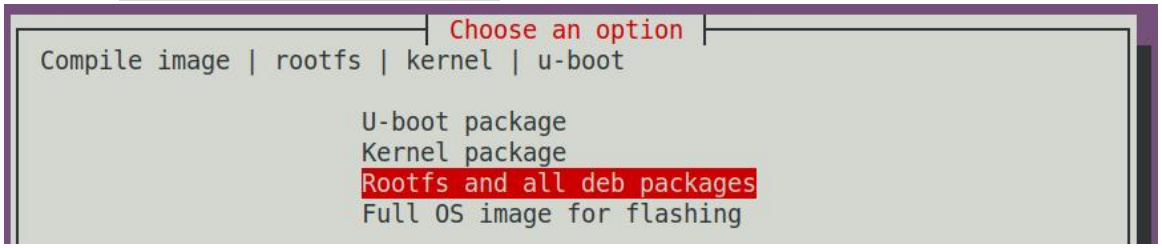
```
root@orangepi:~# reboot
```

6.5. 编译 rootfs

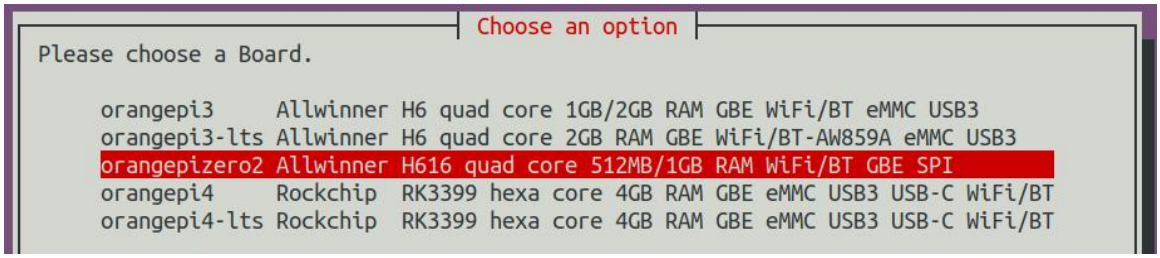
1) 运行 build.sh 脚本，记得加 sudo 权限

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

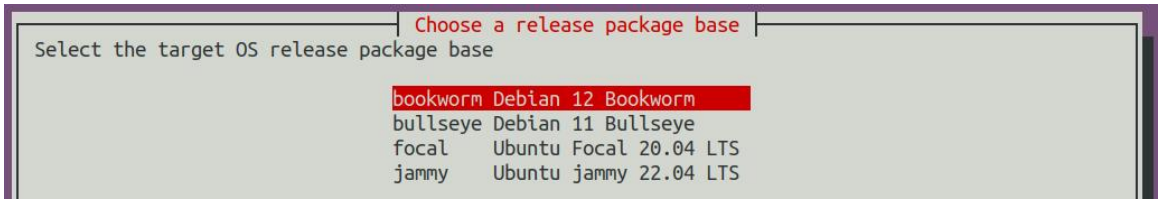
2) 选择 **Rootfs and all deb packages**，然后回车



3) 接着选择开发板的型号

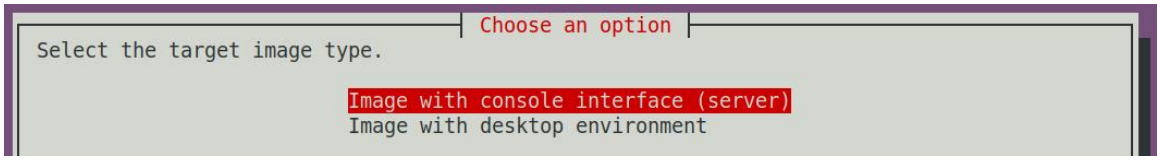


4) 然后选择 rootfs 的类型

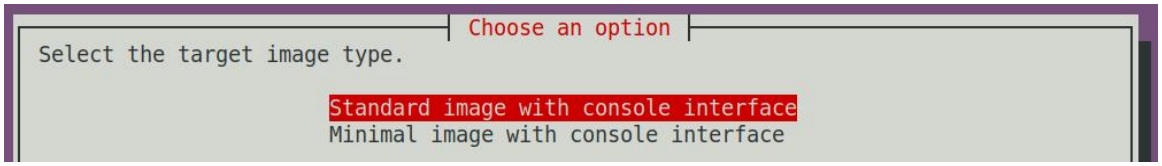


5) 然后选择镜像的类型

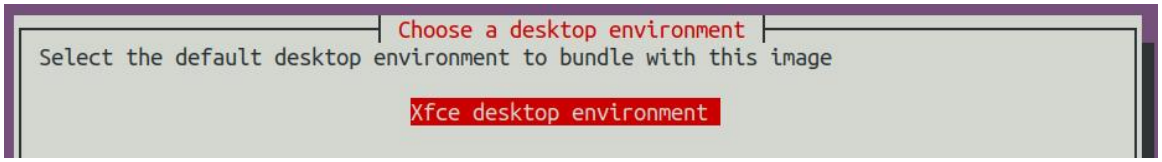
a. **Image with console interface (server)**表示服务器版的镜像，体积比较小
b. **Image with desktop environment** 表示带桌面的镜像，体积比较大

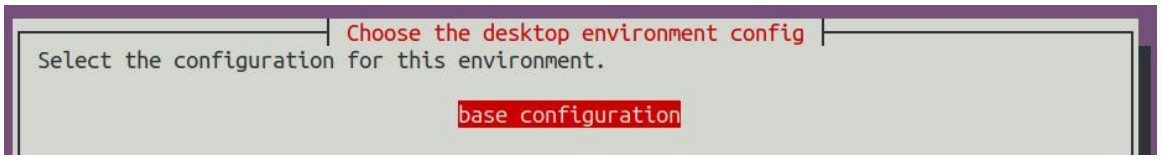


6) 如果是编译服务器版的镜像，还可以选择编译 Standard 版本或者 Minimal 版本，Minimal 版本预装的软件会比 Standard 版本少很多

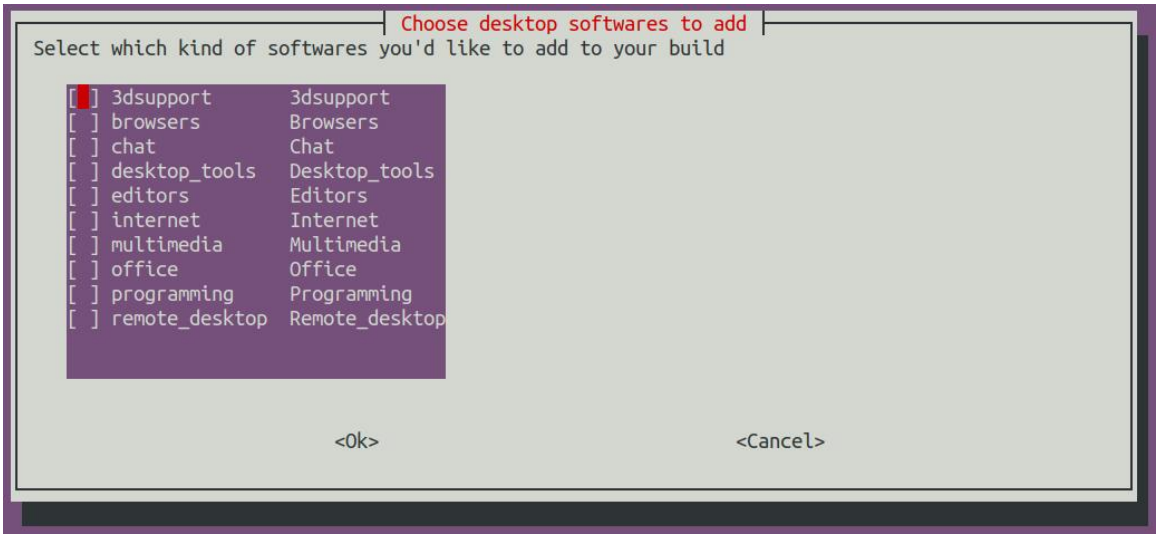


7) 如果是编译桌面版本的镜像还需要选择桌面环境的类型，不过目前只支持 XFCE，所以直接回车即可





然后可以选择需要安装的额外的软件包。比如，如果需要安装浏览器，可以选择 **browsers**。



每个选择具体包含哪些软件包，可以在 orangepi-build 的代码中看到，你也可以修改这些配置文件添加想安装的软件包：

a. 首先进入 **external/config/desktop** 目录可以看到不同 linux 发行版的桌面配置文件夹，注意不是代码中能看到的 Orange Pi 开发板都有支持测试过的

```
test@test:~/orangepi-build$ cd external/config/desktop
test@test:~/orangepi-build/external/config/desktop$ ls
bionic  bookworm  bullseye  buster  focal  jammy  README.md  sid
```

b. 然后选择需要查看或者修改的发行版的类型，进入对应的目录，比如**bullseys**

```
test@test:~/orangepi-build/external/config/desktop$ cd bullseye
test@test:~/orangepi-build/external/config/desktop/bullseye$ ls
appgroups  environments
```

c. 然后进入 **appgroups** 目录就能看到所有 app 分组

```
test@test:~/orangepi-build/external/config/desktop/bullseye$ cd appgroups
test@test:~/orangepi-build/external/config/desktop/bullseye/appgroups$ ls
3dsupport  browsers  chat  desktop_tools  editors  internet  multimedia
office  programming  remote_desktop
```

d. 打开不同组下面的 packages 文件就能查看该组所包含的软件

```
test@test:~/orangepi-build/external/config/desktop/bullseye/appgroups$ cat  \
programming/packages
build-essential
clang
meld
```

```
test@test:~/orangepi-build/external/config/desktop/bullseye/appgroups$ cat   \
office/packages
libreoffice
```

8) 然后就会开始编译 rootfs，编译时提示的部分信息说明如下

a. rootfs 的类型

```
[ o.k. ] local not found [ Creating new rootfs cache for bullseye ]
```

b. 编译生成的 rootfs 压缩包的存放路径

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

c. 编译生成的 rootfs 压缩包的名字

```
[ o.k. ] File name
 [ bullseye-xfce-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4 ]
```

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=orangepizero2
BRANCH=current BUILD_OPT=rootfs RELEASE=bullseye
BUILD_MINIMAL=no   BUILD_DESKTOP=no
KERNEL_CONFIGURE=yes   ]
```

9) 查看编译生成的 rootfs 压缩包

a. bullseye-xfce-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4 是 rootfs 的压缩包，名字各字段的含义为

a) **bullseye** 表示 rootfs 的 linux 发行版的类型

b) **xfce** 表示 rootfs 为桌面版的类型，如果为 **cli** 则表示服务器版类型

c) **arm64** 表示 rootfs 的架构类型

d) **25250ec7002de9e81a41de169f1f89721** 是由 rootfs 安装的所有软件包的包名生成的 MD5 哈希值，只要没有修改 rootfs 安装的软件包的列表，那么这个值就不会变，编译脚本会通过这个 MD5 哈希值来判断是否需要重新编译 rootfs

b. bullseye-xfce-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4.list 列出了 rootfs 安装的所有软件包的包名

```
test@test:~/orangepi-build$ ls external/cache/rootfs/
bullseye-xfce-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4
```

```
bullseye-xfce-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4.current
bullseye-xfce-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4.list
```

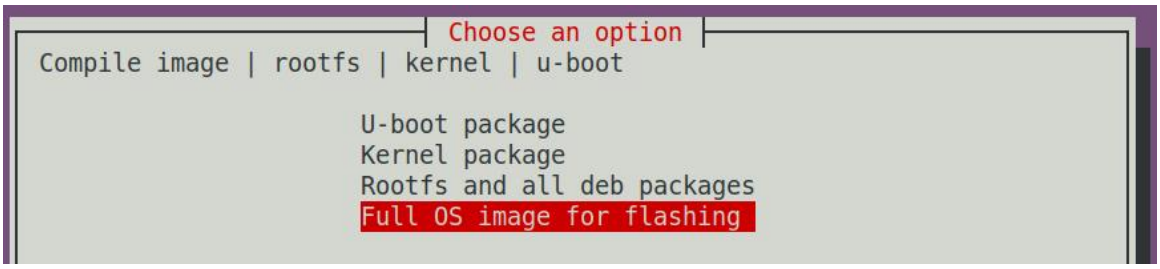
10) 如果需要的 rootfs 在 **external/cache/rootfs** 下已经存在，那么再次编译 rootfs 就会直接跳过编译过程，不会重新开始编译，编译镜像的时候也会去 **external/cache/rootfs** 下查找是否已经有缓存可用的 rootfs，如果有就直接使用，这样可以节省大量的下载编译时间

6.6. 编译 linux 镜像

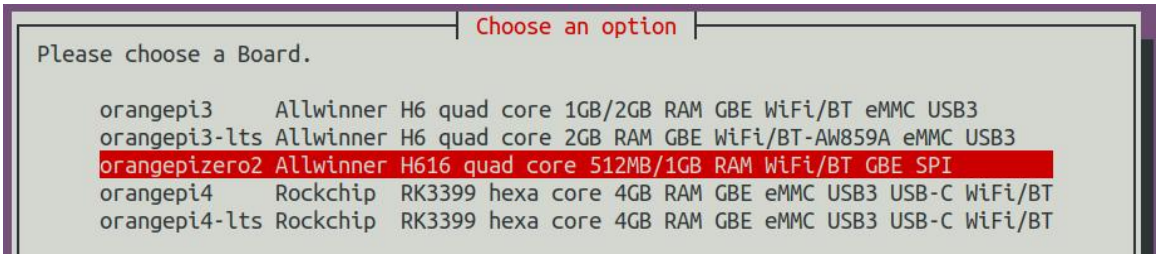
1) 运行 build.sh 脚本，记得加 sudo 权限

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

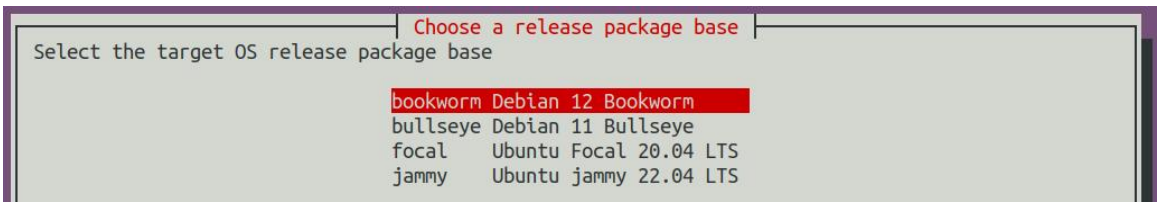
2) 选择 **Full OS image for flashing**，然后回车



3) 然后选择开发板的型号



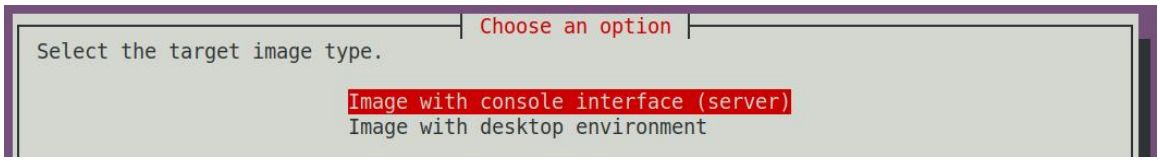
4) 然后选择 rootfs 的类型



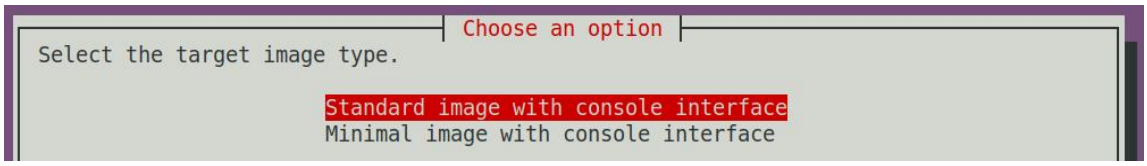
5) 然后选择镜像的类型

a. **Image with console interface (server)**表示服务器版的镜像，体积比较小

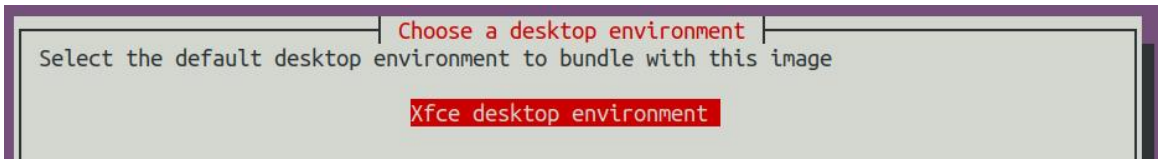
b. **Image with desktop environment** 表示带桌面的镜像，体积比较大

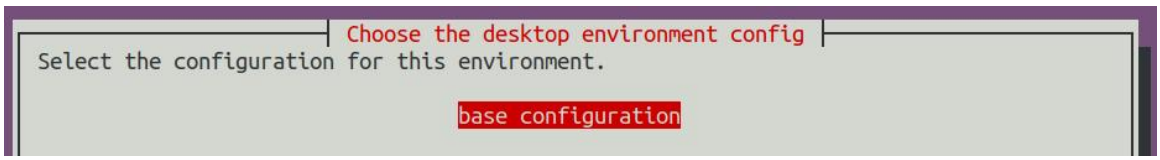


6) 如果是编译服务器版的镜像，还可以选择编译 Standard 版本或者 Minimal 版本，Minimal 版本预装的软件会比 Standard 版本少很多

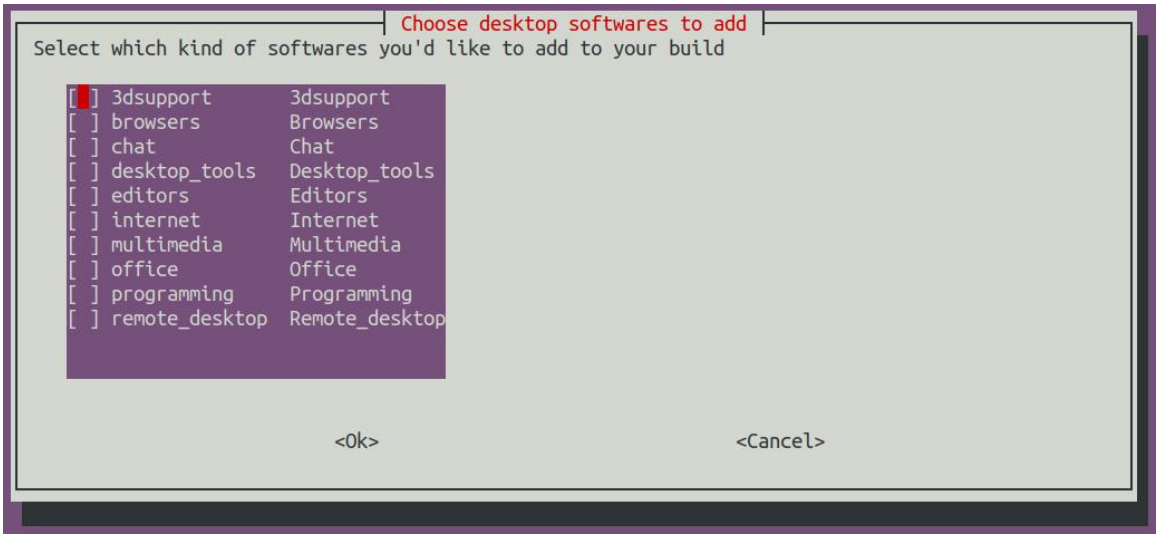


7) 如果是编译桌面版本的镜像还需要选择桌面环境的类型，不过目前只支持 XFCE，所以直接回车即可





然后可以选择需要安装的额外的软件包。比如，如果需要安装浏览器，可以选择 **browsers**。



每个选择具体包含哪些软件包，可以在 orangepi-build 的代码中看到，你也可以修改这些配置文件添加想安装的软件包：

a. 首先进入 **external/config/desktop** 目录可以看到不同 linux 发行版的桌面配置文件夹，注意不是代码中能看到的 Orange Pi 开发板都有支持测试过的

```
test@test:~/orangepi-build$ cd external/config/desktop
test@test:~/orangepi-build/external/config/desktop$ ls
bionic  bookworm  bullseye  buster  focal  jammy  README.md  sid
```

b. 然后选择需要查看或者修改的发行版的类型，进入对应的目录，比如**bullseys**

```
test@test:~/orangepi-build/external/config/desktop$ cd bullseye
test@test:~/orangepi-build/external/config/desktop/bullseye$ ls
appgroups  environments
```

c. 然后进入 **appgroups** 目录就能看到所有 app 分组

```
test@test:~/orangepi-build/external/config/desktop/bullseye$ cd appgroups
test@test:~/orangepi-build/external/config/desktop/bullseye/appgroups$ ls
3dsupport  browsers  chat  desktop_tools  editors  internet  multimedia
office  programming  remote_desktop
```

d. 打开不同组下面的 packages 文件就能查看该组所包含的软件

```
test@test:~/orangepi-build/external/config/desktop/bullseye/appgroups$ cat  \
programming/packages
build-essential
clang
meld
```

```
test@test:~/orangepi-build/external/config/desktop/bullseye/appgroups$ cat   \
office/packages
libreoffice
```

8) 然后就会开始编译 linux 镜像，编译的大致流程如下

a. 初始化 Ubuntu PC 的编译环境，安装编译过程需要的软件包
b. 下载 u-boot 和 linux 内核的源码（如果已经缓存，则只更新代码）
c. 编译 u-boot 源码，生成 u-boot 的 deb 包
d. 编译 linux 源码，生成 linux 相关的 deb 包
e. 制作 linux firmware 的 deb 包
f. 制作 orangepi-config 工具的 deb 包
g. 制作板级支持的 deb 包
h. 如果是编译 desktop 版镜像，还会制作 desktop 相关的 deb 包
i. 检查 rootfs 是否已经缓存，如果没有缓存，则重新制作 rootfs，如果已经缓存，则直接解压使用
j. 安装前面生成的 deb 包到 rootfs 中
k. 对不同的开发板和不同类型镜像做一些特定的设置，如预装额外的软件包，修改系统配置等
l. 然后制作镜像文件，并格式化分区，默认类型为 ext4
m. 再将配置好的 rootfs 拷贝到镜像的分区中
n. 然后更新 initramfs
o. 最后将 u-boot 的 bin 文件通过 dd 命令写入到镜像中

9) 编译完镜像后会提示下面的信息

a. 编译生成的镜像的存放路径

```
[ o.k. ] Done building
[ output/images/orangepizero2_3.0.0_debian_bullseye_linux5.10.43_xfce_desktop/orangepizero2_3.0.0_debian_bullseye_linux5.10.43_xfce_desktop.img ]
```

b. 编译使用的时间

```
[ o.k. ] Runtime [ 19 min ]
```

c. 重复编译镜像的命令，使用下面的命令无需通过图形界面选择，可以直接开始编译镜像

```
[ o.k. ] Repeat Build Options [ sudo ./build.sh  BOARD=orangepizero2
BRANCH=current BUILD_OPT=image RELEASE=bullseye
BUILD_MINIMAL=no BUILD_DESKTOP=no KERNEL_CONFIGURE=yes ]
```

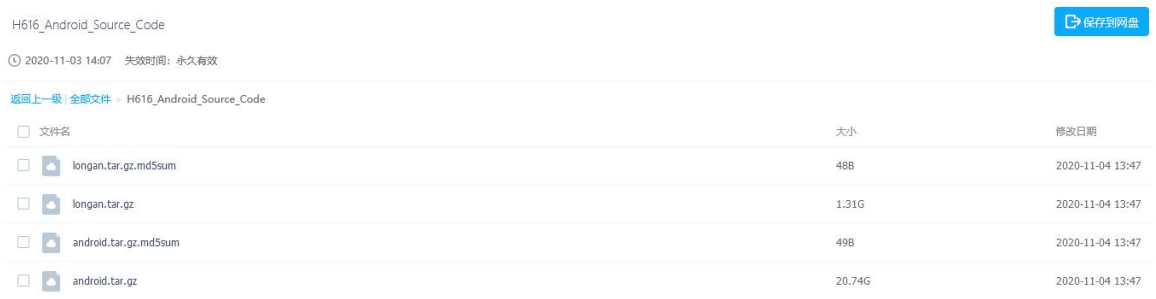
7. Android SDK 使用说明

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。

7.1. 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



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
```

7.2. 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
```

7.3. 编译 android 镜像

7.3.1. 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: ----------------------------------------
```

7.3.2. 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
```