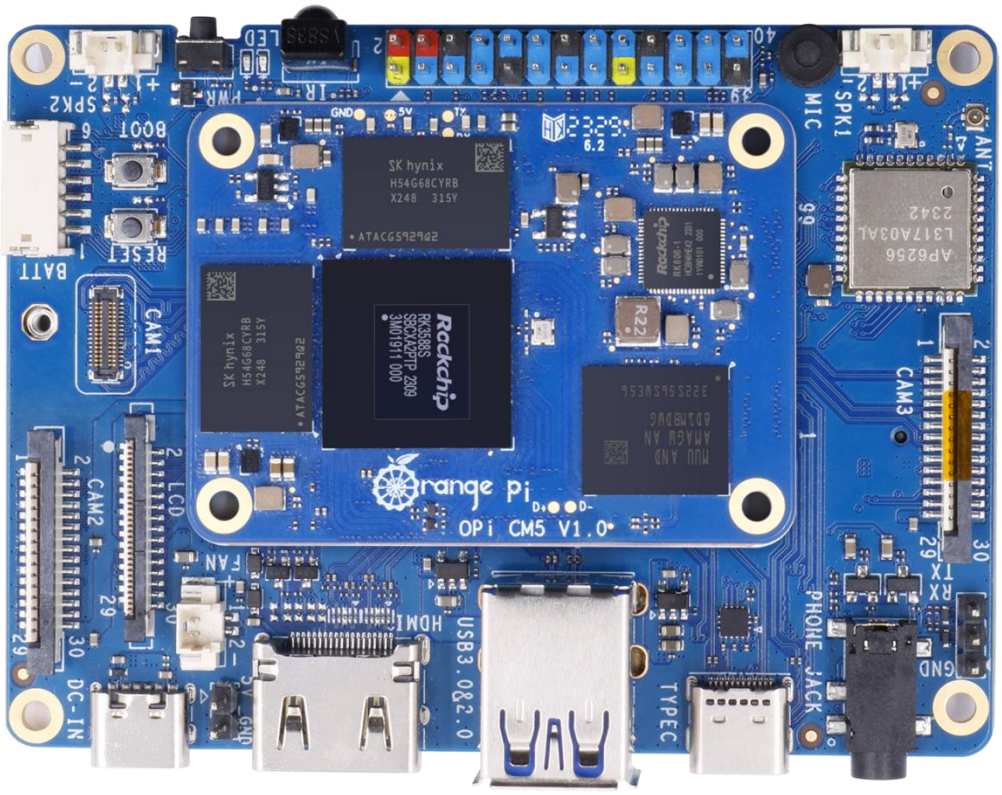
Orange Pi CM5 Base Tablet
用户手册



目录

1. Orange Pi CM5 Base Tablet 的基本特性

1.1. 什么是 Orange Pi CM5 Base Tablet

Orange Pi CM5 核心板采用了瑞芯微 RK3588S 新一代八核 64 位 ARM 处理器，具体为四核 A76 和四核 A55，采用的三星 8nm LP 制程工艺，大核主频最高可达 2.4GHz，集成 ARM Mali-G610 MP4 GPU，内嵌高性能 3D 和 2D 图像加速模块，内置高达 6 Tops 算力的 AI 加速器 NPU，拥有 4GB/8GB/16GB（LPDDR5）内存，具有高达 8K 显示处理能力。另外核心板还配备有板载的 eMMC，可选容量有 32GB/64GB/128GB/256GB。

Orange Pi CM5 Base Tablet 底板引出了相当丰富的接口，包括 HDMI 输出、Wi-Fi、蓝牙、M.2 PCIe2.0x1、MIPI CSI、MIP DSI、USB2.0、USB3.1 接口和 26pin 扩展排针等。可广泛适用于高端平板、边缘计算、人工智能、云计算、AR/VR、智能安防、智能家居等领域，覆盖 AIoT 各个行业。

Orange Pi CM5 Base Tablet 支持 Orange Pi 官方研发的操作系统 Orange Pi OS，同时，支持 Android 13、Debian11、Debian12、Ubuntu20.04 和 Ubuntu22.04 等操作系统。

1.2. Orange Pi CM5 Base Tablet 的用途

我们可以用它实现：

- 一台 Linux 桌面电脑。
- Android 平板。
- Android 游戏机等。

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

1.3. Orange Pi CM5 核心板的硬件特性

OPi CM5 核心板硬件规格	
主控芯片	Rockchip RK3388S(8nm LP 制程）
CPU	• 8 核 64 位处理器 • 4 核 Cortex-A76 和 4 核 Cortex-A55 的典型大小核架构 • 大核主频 2.4GHz，小核主频 1.8GHz
GPU	• 集成 ARM Mali-G610 • 兼容 OpenGL ES1.1/2.0/3.2、OpenCL 2.2 和 Vulkan 1.2
NPU	6 Tops 算力，支持 INT4/INT8/INT16 混合运算
PMU	RK806-1
RAM	LPDDR4/4x：2GB、4GB、8GB、16GB
EMMC	eMMC：32GB、64GB、128GB、256GB
接口	3*100PIN(型号：DF40C-100DP-0.4V(51)），包含以下接口： 1*TYPE C or DP1.4 3*USB2.0 1*HDMI 2.1 or eDP1.3 1*uSD 1*4-lane MIPI DPHY TX 1*2-lane MIPI DPHY TX 2*2-lane MIPI DPHY RX 1*4-lane MIPI CSI RX or 2*2-lane MIPI CSI RX 1*SATA III or PCIe2.0 1*SATA III or PCIe2.0 or USB3.0 USB3.0*1+USB2.0*3 SDIO 3.0 or RGMII I2C、I2S、UART、SPI、CAN、PWM、PDM、GPIO 等 POWER_ON、RESET、MASKROM、RECOVERY 等按键
电源	输入：DC 5V MAX1800mA 输出：DC3.3V MAX600mA 和 DC1.8V MAX600mA
PCB	长：55mm,宽：40mm，厚：1.6mm
rangePi™是深圳市迅龙软件有限公司的注册商标	

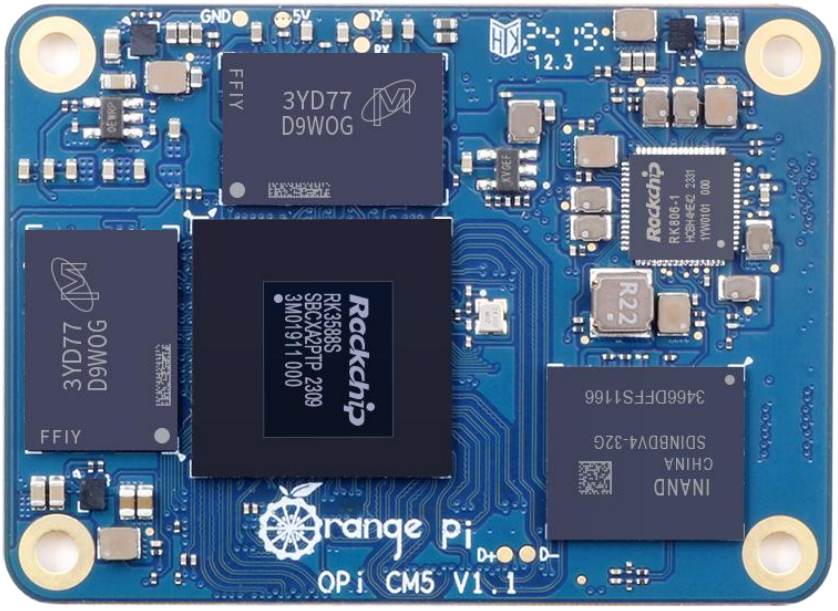
1.4. Orange Pi CM5 Base Tablet 底板的硬件特性

Orange Pi CM5 base-tablet 硬件特性	
板对板连接器	3*100PIN,0.4mm PIN 距，连接器型号：df40c-100ds-0.4v
存储	• MicroSD (TF) 插槽 • M.2 M-KEY 插槽：支持 NVMe PCIe2.0 或 SATA SSD
USB	• Type-C USB3.0 • USB3.0+USB2.0
视频输出	• HDMI 2.1，最高支持 8K@60Hz • DP 1.4 ALT 4Lane (共用 TYPE-C 接口） • MIPI DPHY TX 4 Lane
摄像头	• 1*MIPI CSI 4 Lane • 2*MIPI DPHY RX 2 Lane
音频	• 3.5mm 耳机孔音频输入/输出 • HDMI 音频输出 • 板载 MIC 音频输入 • 2*喇叭插座，规格为 2PIN 1.25mm
WIFI+BT	• 模块：AP6256 • 板载 WI-FI5+BT 5.0
26PIN 扩展接口	• 双排插针规格：2.54mm 间距 • 支持 UART、PWM、I2C、SPI、GPIO 等功能
按键	1*MASKROM、1*RESET 和 1*PWR
供电	支持 Type-C 供电，5V@5A
电池座	电池座：板对线连接，6PIN，1.5mm 间距，接单节锂电池或锂聚合物电池，最大充电电流 5A
RTC	预留焊盘，可焊接 3V 备用 RTC 电池
按键接口	6PIN FPC 插座，PIN 距 0.5mm
传感器接口	6PIN FPC 插座，PIN 距 0.5mm
LED	红色的电源指示灯和绿色的状态指示灯
FAN	5V 2PIN 1.25mm 间距
SPK	2*2PIN 1.25mm 间距

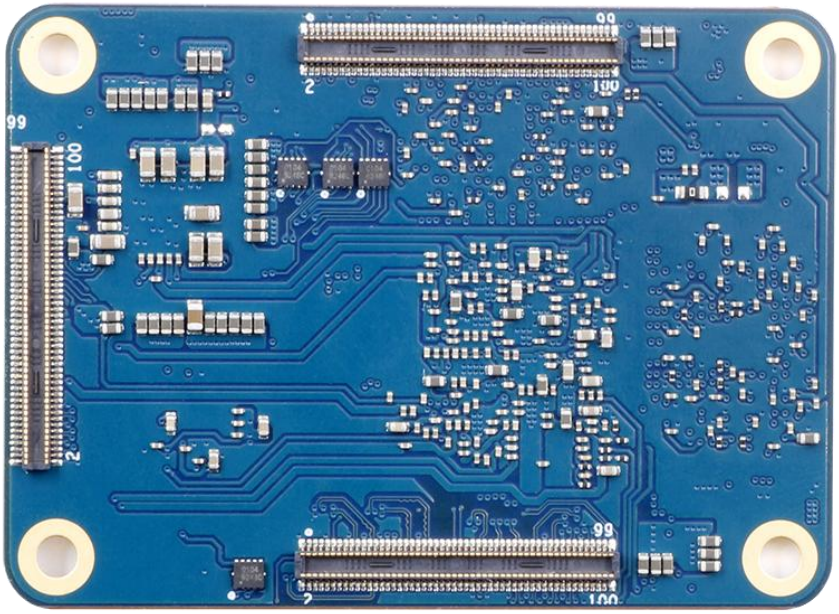
调试串口	3Pin 调试串口，PIN 距 2.54mm
PCB	长：90mm，宽：66mm，厚：1.6mm
rangePi™是深圳市迅龙软件有限公司的注册商标	

1.5. Orange Pi CM5 核心板顶层视图和底层视图

顶层视图：

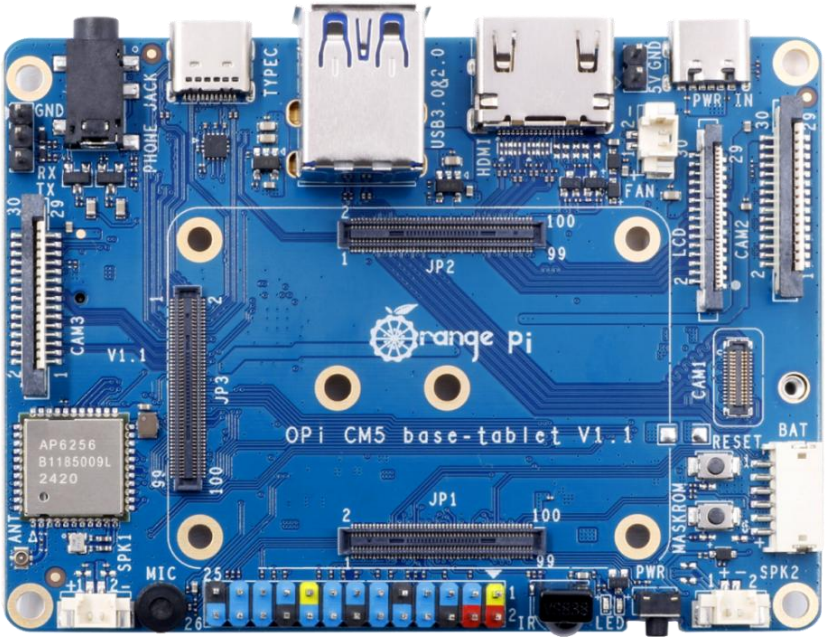


底层视图：

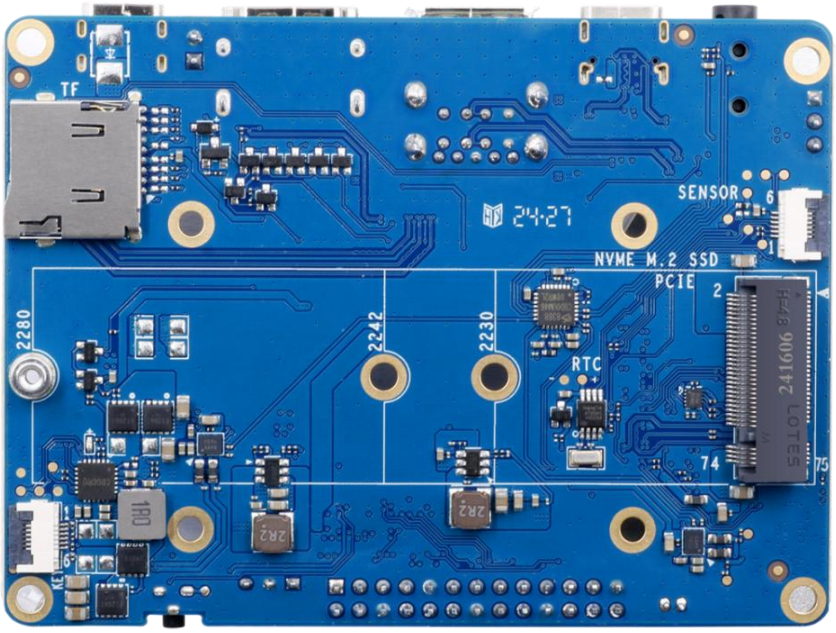


1.6. Orange Pi CM5 Base Tablet 底板顶层视图和底层视图

顶层视图：

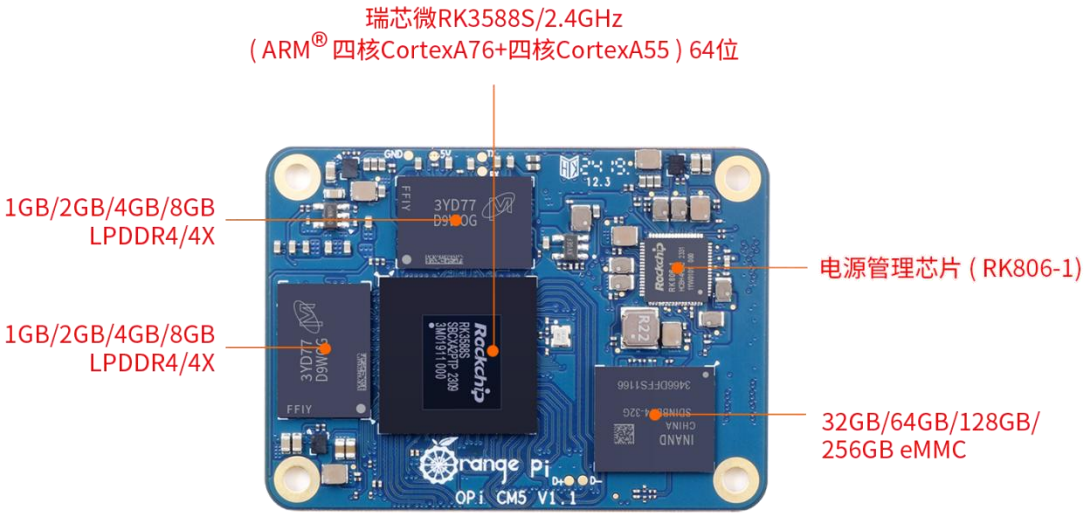


底层视图：

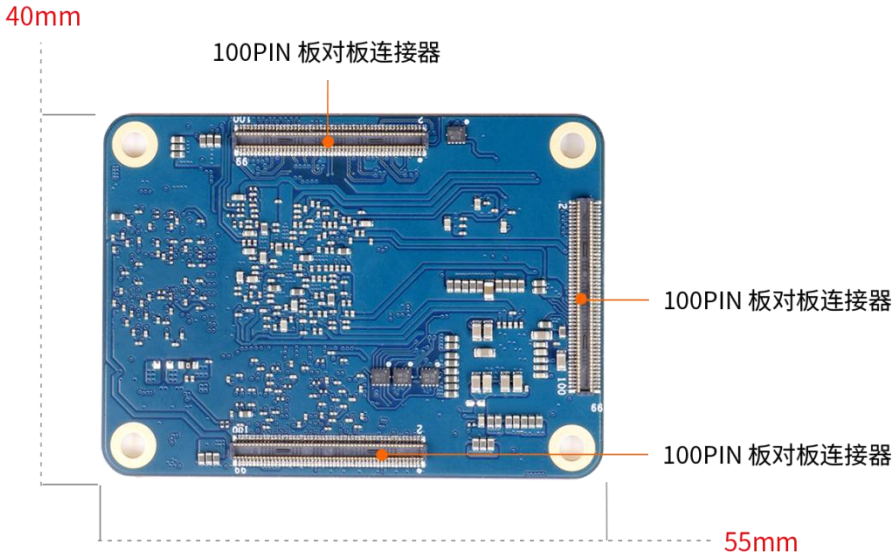


1.7. Orange Pi CM5 核心板接口详情图

正面视图

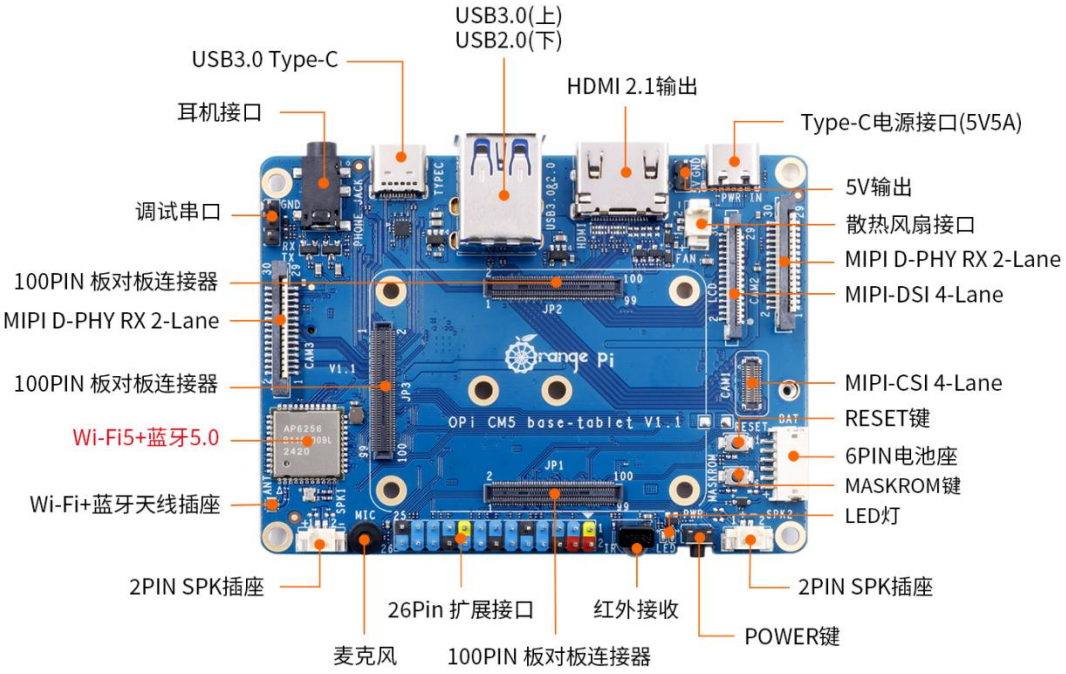


背面视图

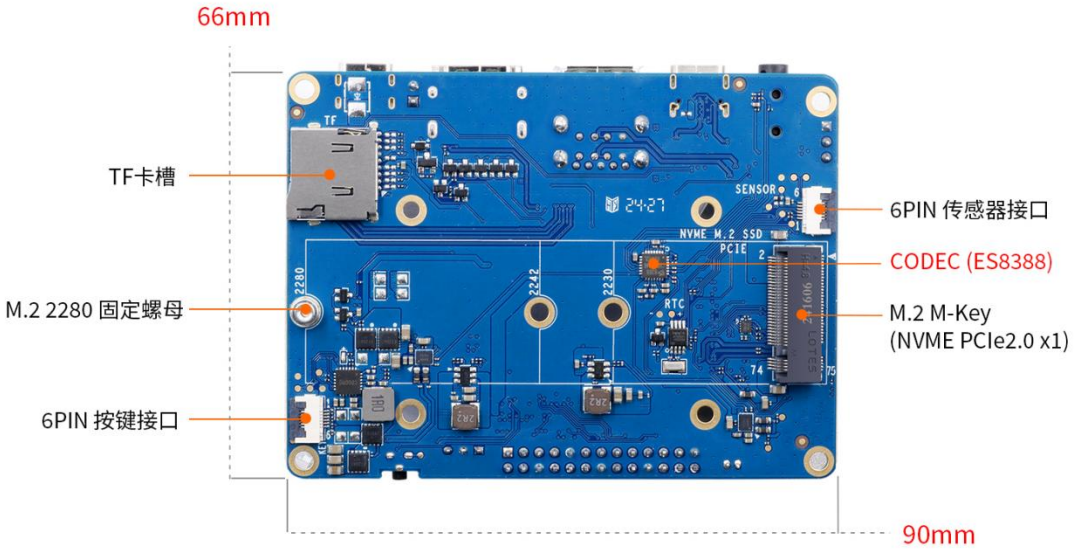


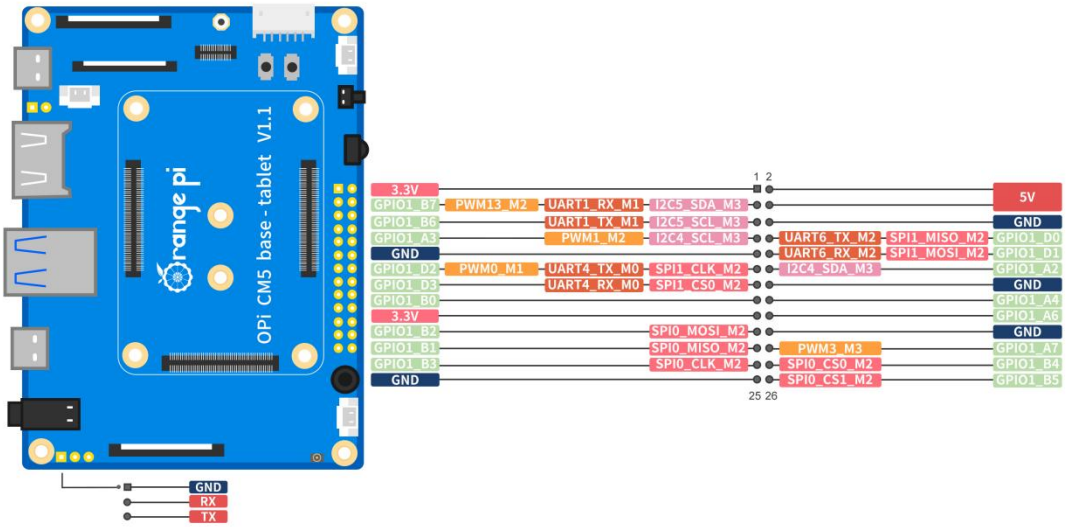
1.8. Orange Pi CM5 Base Tablet 底板接口详情图

正面视图



背面视图





核心板四个定位孔的直径都是 2.7mm，底板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 16GB 容量（推荐 32GB 或以上）的 **class10** 级或以上的高速闪迪卡。

2) TF 卡读卡器，用于将镜像烧录到 TF 卡中。

3) HDMI 接口的显示器。

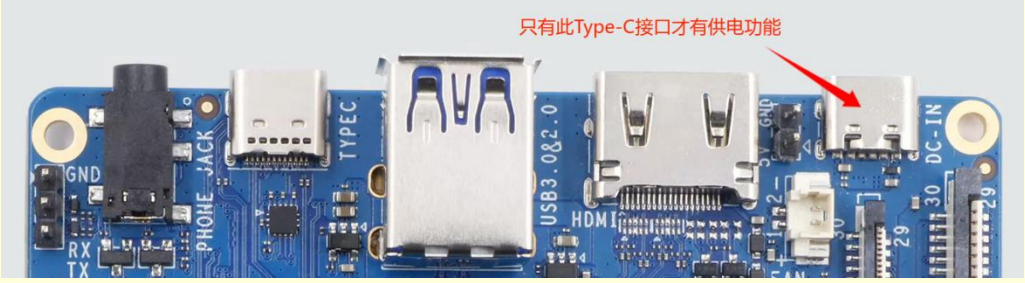
4) HDMI 转 HDMI 连接线，用于将开发板连接到 HDMI 显示器或者电视进行显示。

注意，如果想接 4K或者 8K显示器，请确保HDMI线支持 4K或者 8K视频输出。

5) 10.1 寸MIPI屏幕，用于显示开发板的系统界面（此屏幕包括转接板，在OPi5Plus/OPi5B/OPi5/OPiCM5BaseTablet上通用）。

6) 电源适配器，建议使用 5V/4A 或者 5V/5A 的 Type-C 电源供电。

底板上有两个长得一样的Type-C接口，其中右边那个才是电源接口，左边那个是没有供电功能的，请别接错了。



开发板的Type-C电源接口不支持PD协商功能，只支持固定的 5V电压输入。

7) 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。

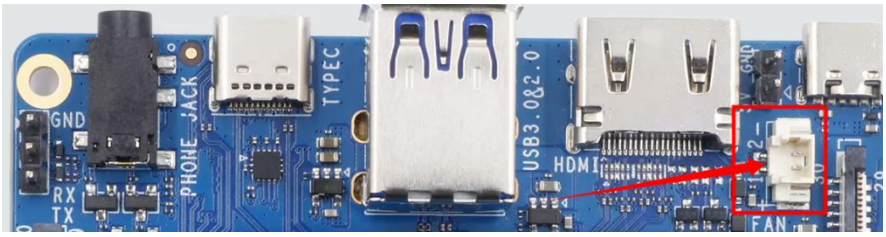
8) USB摄像头。

9) 红外遥控器。

注意，Orange Pi提供的操作系统默认只能保证Orange Pi提供的遥控器才可以使用。

10) 5V的散热风扇。如下图所示，开发板上有用于接散热风扇的接口，接口规格为**2pin 1.25mm间距**。

开发板上的风扇可以通过PWM来调节转速和开关。

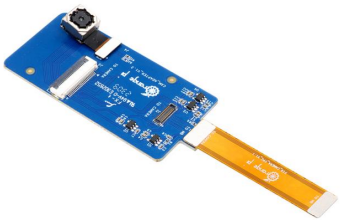


11) Type-C转HDMI线，通过Type-C接口将开发板连接到HDMI显示器或者电视进行显示。

12) Type-C转USB转接头，用于连接USB存储设备或者鼠标键盘等USB设备。

13) 1300 万MIPI接口的OV13850 摄像头。

14) 1300 万MIPI接口的OV13855 摄像头。

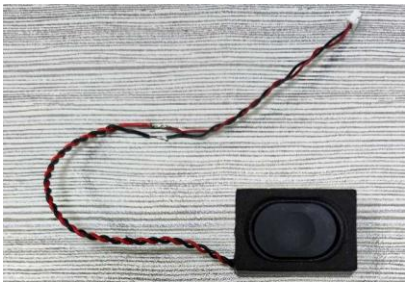


15) Type-C接口的数据线，用于烧录镜像、使用ADB等功能。

16) M.2 M-KEY 2280 规格的 NVMe SSD 固态硬盘，PCIe 接口的规格为 PCIe2.0x1。

17) M.2 M-KEY 2280 规格的 SATA SSD 固态硬盘。

18) 喇叭，接口为 2pin，1.25mm 间距。



底板上两个喇叭接口的位置如下图所示：

19) **3.3V** 的 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。

20) 安装有 Ubuntu 和 Windows 操作系统的个人电脑。

1	Ubuntu22.04 PC	可选，用于编译 Linux 源码
2	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-CM5.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-CM5.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**：保存在百度云盘和谷歌网盘上。
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。
h. **Orange Pi OS 镜像：**保存在百度云盘和谷歌网盘上。
i. **OpenWRT 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、Ubuntu、OpenWRT或者OPi OS Arch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 16GB 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。

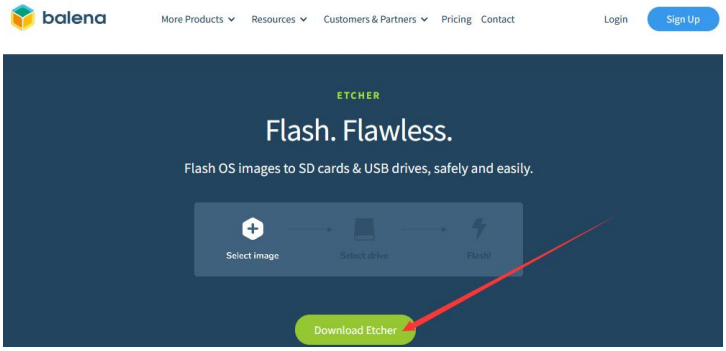
2) 然后使用读卡器把 TF 卡插入电脑。

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 2GB 以上。

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

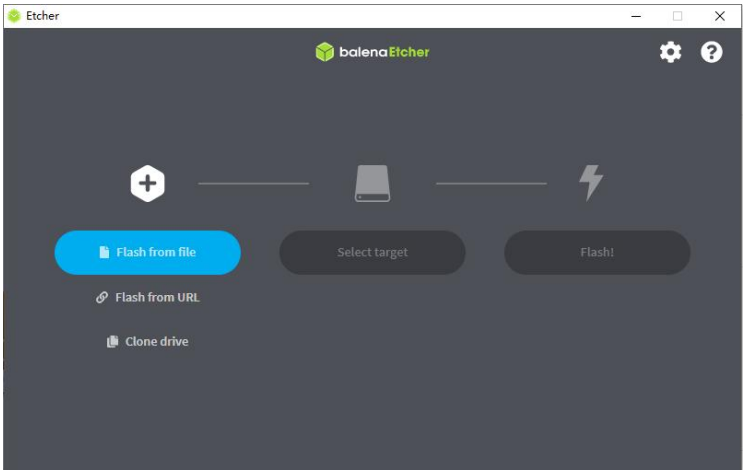
https://www.balena.io/etcher/

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

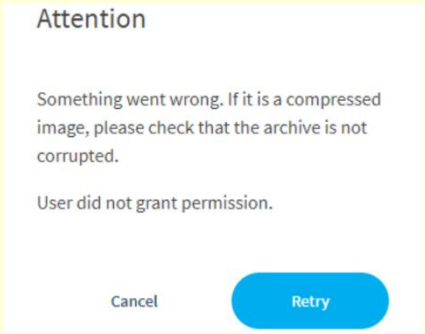


6) 然后可以选择下载 balenaEtcher 的 Portable 版本的软件，Portable 版本无需安装，双击打开就可以使用。

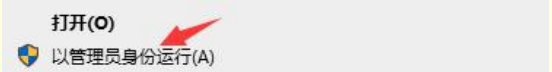
7) 如果下载的是需要安装版本的 balenaEtcher，请先安装再使用。如果下载的 Portable 版本 balenaEtcher，直接双击打开即可，打开后的 balenaEtcher 界面如下图所示：



打开 balenaEtcher 时如果提示下面的错误：



请选择 balenaEtcher 后点击右键，然后选择以管理员身份运行。

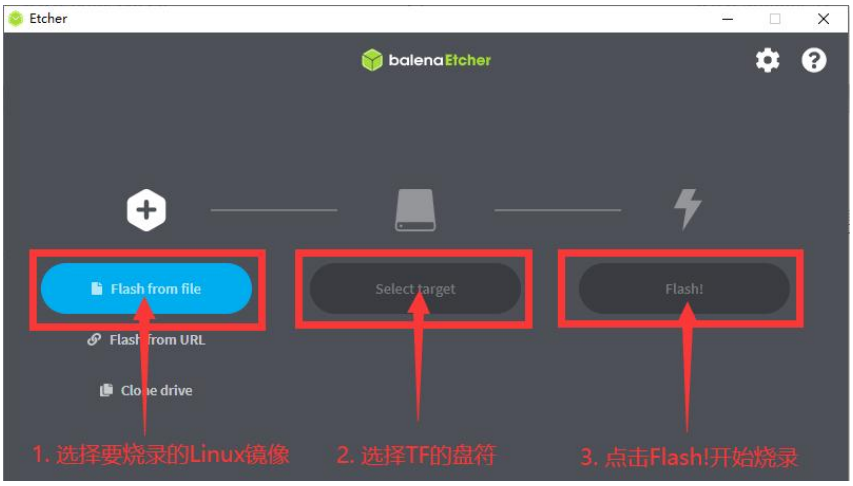


8) 使用 balenaEtcher 烧录 Linux 镜像的具体步骤如下所示：

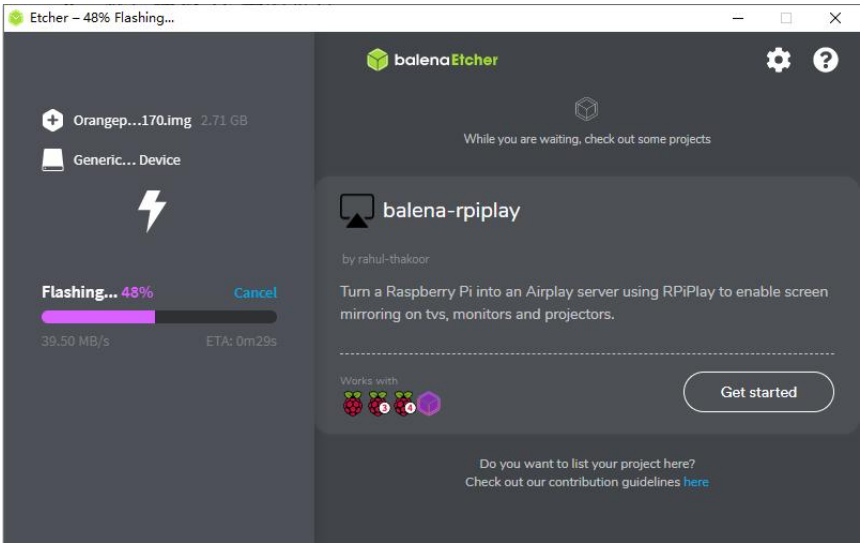
a. 首先选择要烧录的 Linux 镜像文件的路径；

b. 然后选择 TF 卡的盘符；

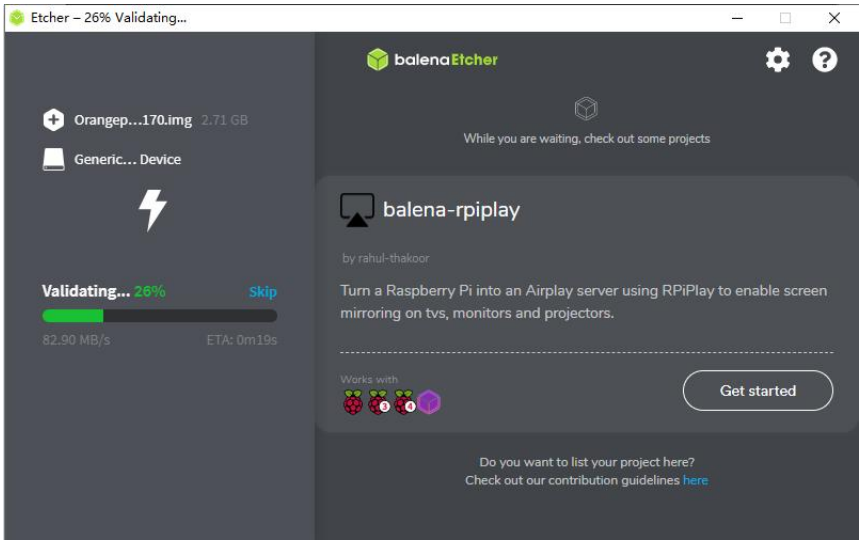
c. 最后点击 Flash 就会开始烧录 Linux 镜像到 TF 卡中。



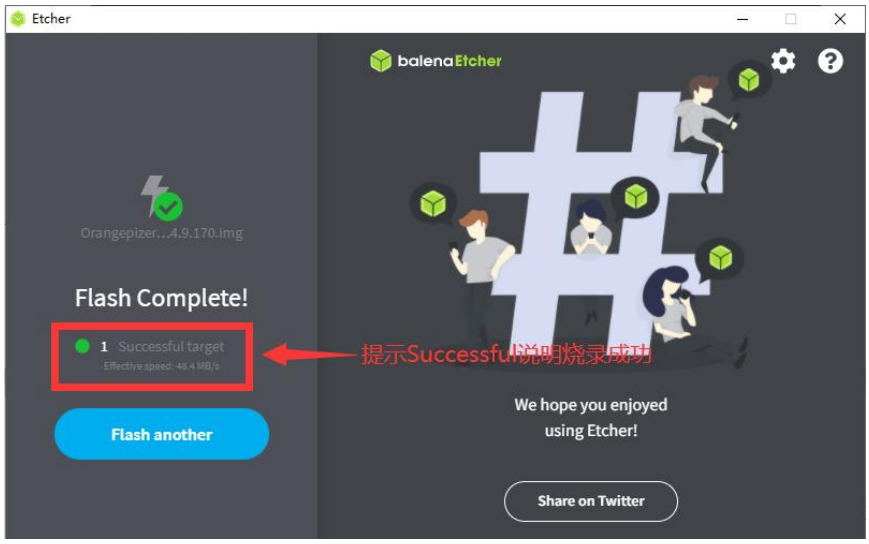
9) balenaEtcher 烧录 Linux 镜像的过程显示的界面如下图所示，另外进度条显示紫色表示正在烧录 Linux 镜像到 TF 卡中。



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



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

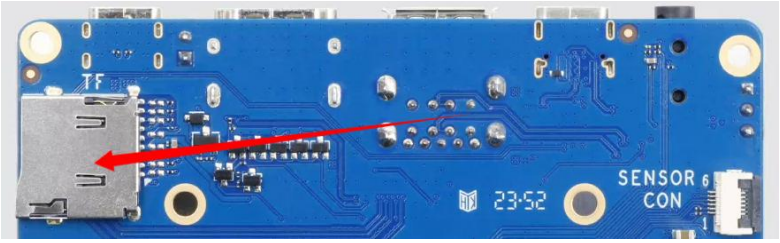


2.3.2. 使用 RKDevTool 烧录 Linux 镜像到 TF 卡中的方法

1) 首先需要准备一根品质良好的 Type-C 接口的数据线。

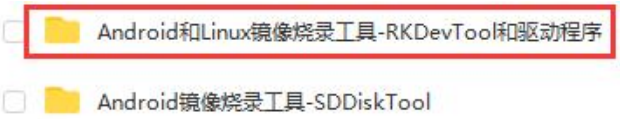
2) 还需要准备一张 16GB 或更大容量的 TF 卡，TF 卡的传输速度必须为 **class10** 级或 **class10** 级以上，建议使用闪迪等品牌的 TF 卡。

3) 然后将 TF 卡插入底板的 TF 卡槽中。



4) 然后从 Orange Pi 的资料下载页面下载瑞芯微驱动 **DriverAssitant_v5.12.zip** 和 **MiniLoader** 以及烧录工具 **RKDevTool_Release_v3.15.zip**。

a. 在 **Orange Pi 的资料下载页面**选择**官方工具**，然后进入下面的文件夹中。



b. 然后下载下面的所有文件。

注意，“MiniLoader-烧录Linux镜像才需要用到的东西”文件夹下文简称为MiniLoader文件夹。

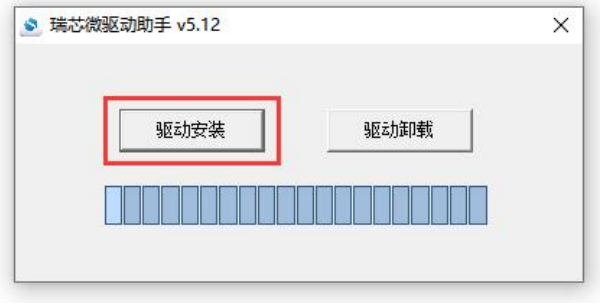
5) 然后从 Orange Pi 的资料下载页面下载想要烧录的 Linux 操作系统镜像文件压缩包，然后使用解压软件解压，解压后的文件中，以“.img”结尾的文件就是操作

系统的镜像文件，大小一般都在 2GB 以上。

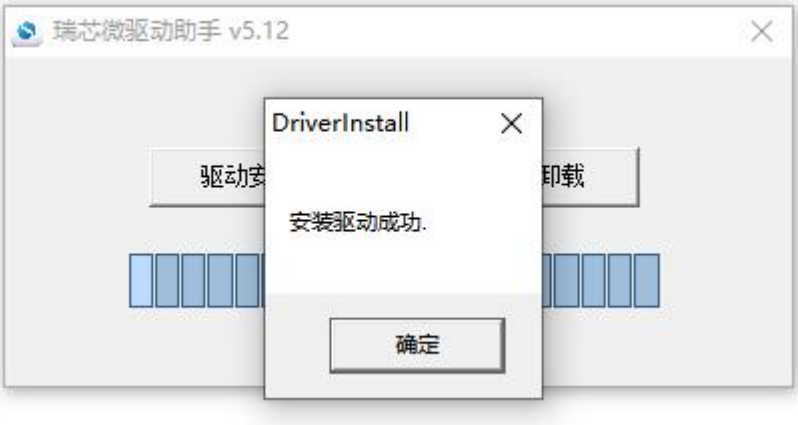
6) 然后用解压软件解压 **DriverAssitant_v5.12.zip**，再在解压后的文件夹中找到 **DriverInstall.exe** 可执行文件并打开即可。

7) 打开 **DriverInstall.exe** 后安装瑞芯微驱动的步骤如下所示：

a. 点击“**驱动安装**”按钮。

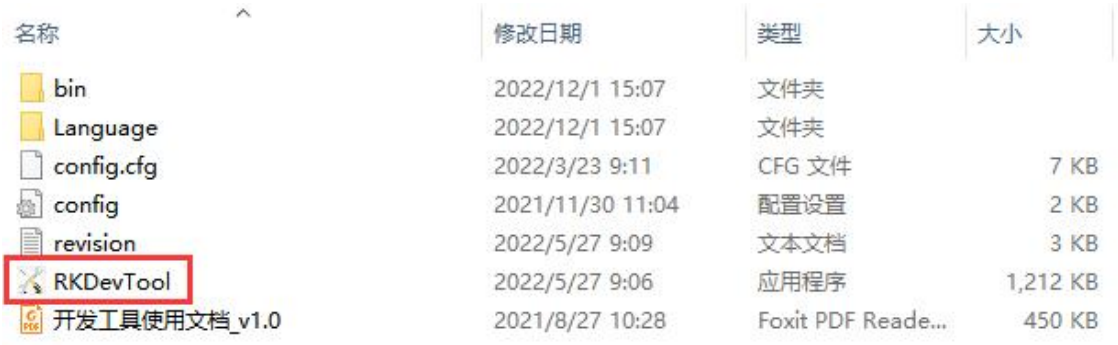


b. 等待一段时间后，会弹出窗口提示“**安装驱动成功**”，然后点击“**确定**”按钮即可。

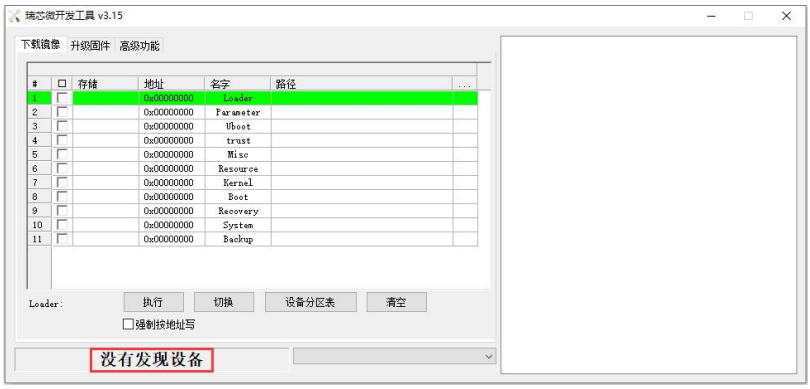


8) 然后解压 **RKDevTool_Release_v3.15.zip**，此软件无需安装，在解压后的文件夹

中找到 **RKDevTool** 打开即可。

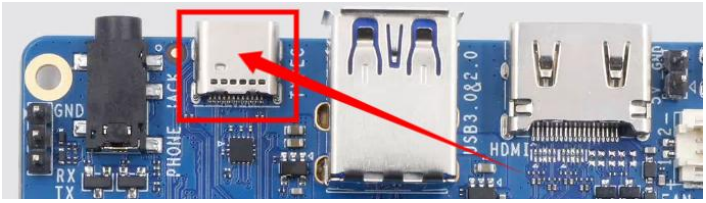


9) 打开 **RKDevTool** 烧录工具后，因为电脑此时还没有通过 Type-C 线连接上开发板，所以左下角会提示“**没有发现设备**”。



10) 然后开始烧录 Linux 镜像到 TF 卡中。

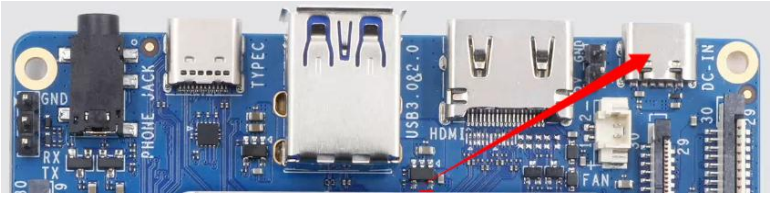
a. 首先通过 Type-C 数据线连接好底板与 Windows 电脑，开发板 Type-C 接口的位置如下图所示：



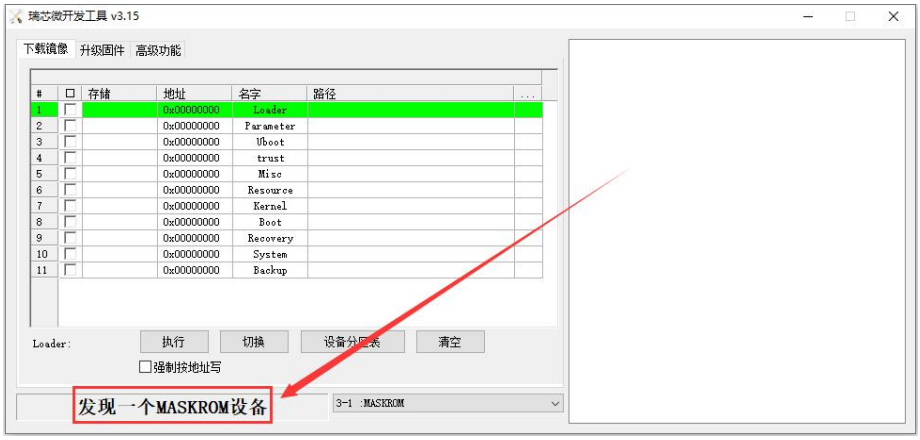
b. 确保开发板没有连接 Type-C 电源。

c. 然后按住开发板的 MaskROM 按键不放，MaskROM 按键在开发板的位置如下图所示：

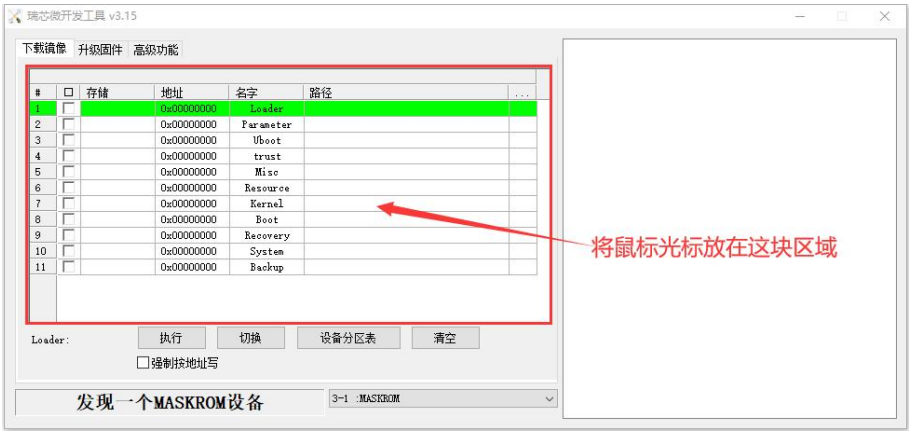
d. 然后给开发板接上 Type-C 接口的电源，并上电，然后就可以松开 MaskROM 按键了。Type-C 电源接口的位置如下所示：



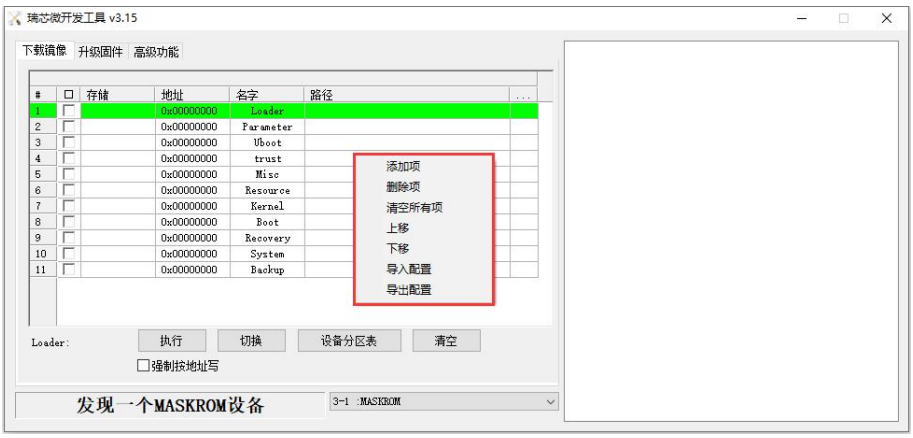
e. 如果前面的步骤顺利，此时开发板会进入 **MASKROM** 模式，在烧录工具的界面上会提示“**发现一个 MASKROM 设备**”。



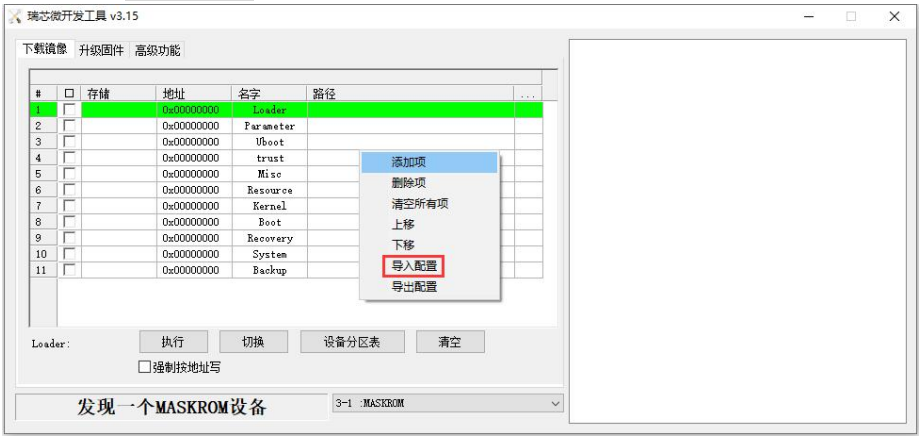
f. 然后将鼠标光标放在下面的这片区域中。



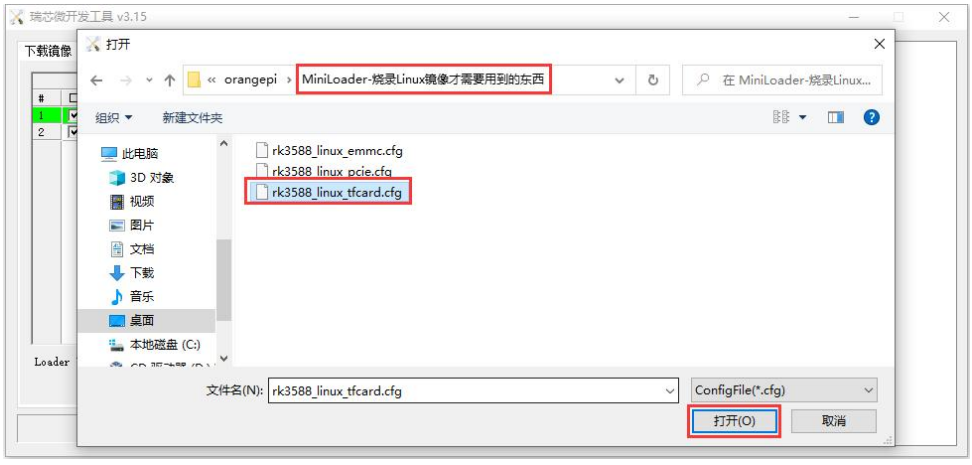
g. 然后点击鼠标右键会弹出下图所示的选择界面。



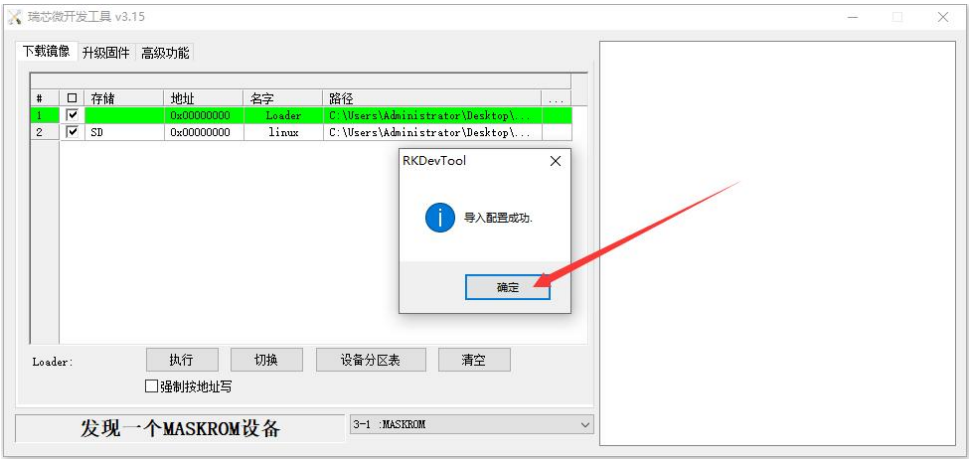
h. 然后选择**导入配置**选项。



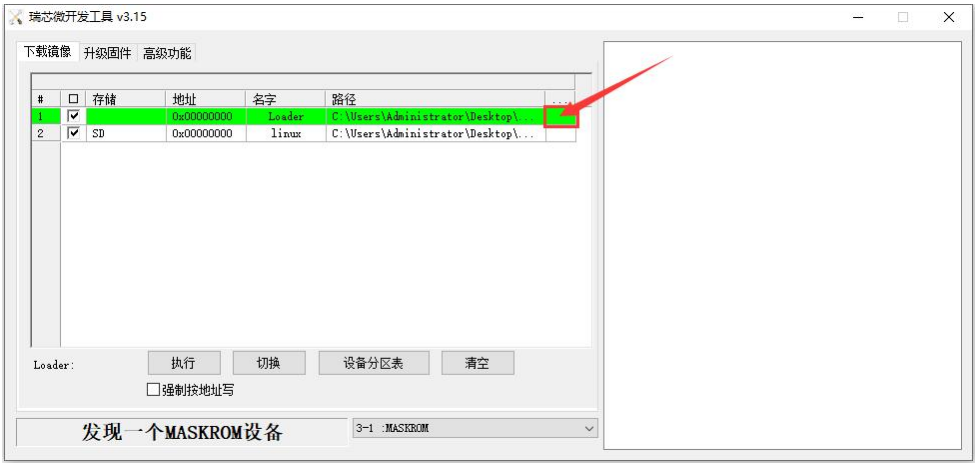
i. 然后选择前面下载的 **MiniLoader** 文件夹中的 **rk3588_linux_tfcard.cfg** 配置文件，再点击**打开**。



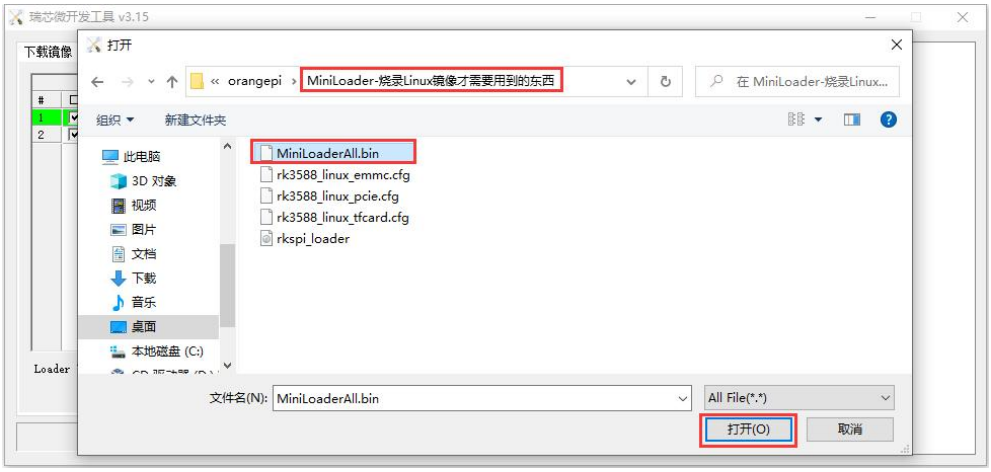
j. 然后点击**确定**。



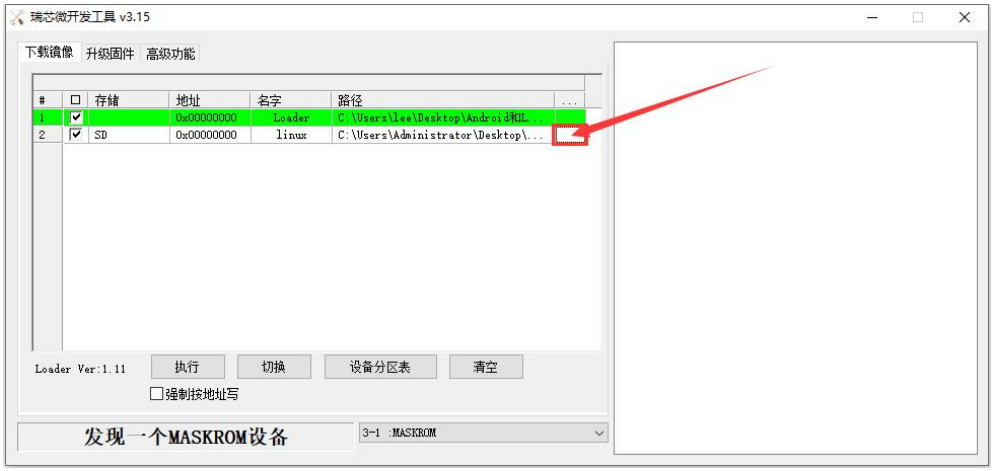
k. 然后点击下图所示的位置。



l. 再选择前面下载的 **MiniLoader** 文件夹中 **MiniLoaderAll.bin**，再点击**打开**。

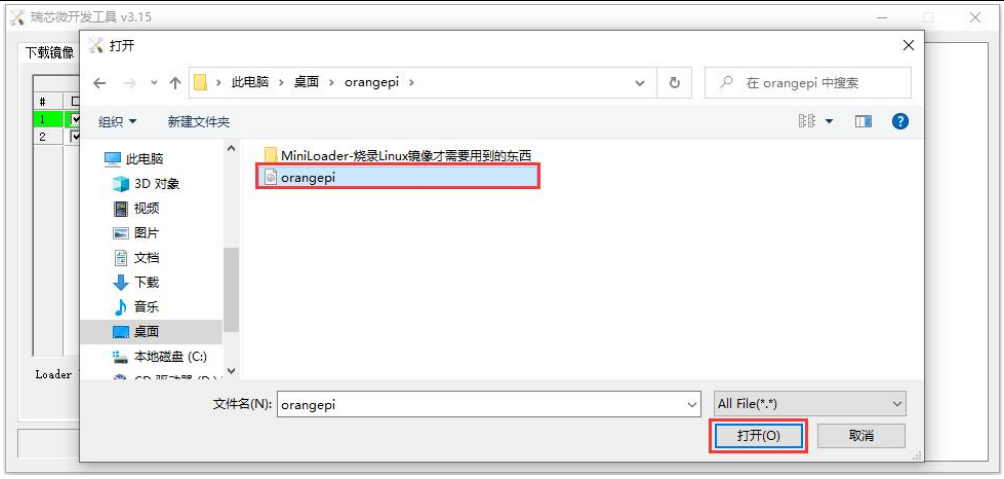


m. 然后点击下图所示的位置。

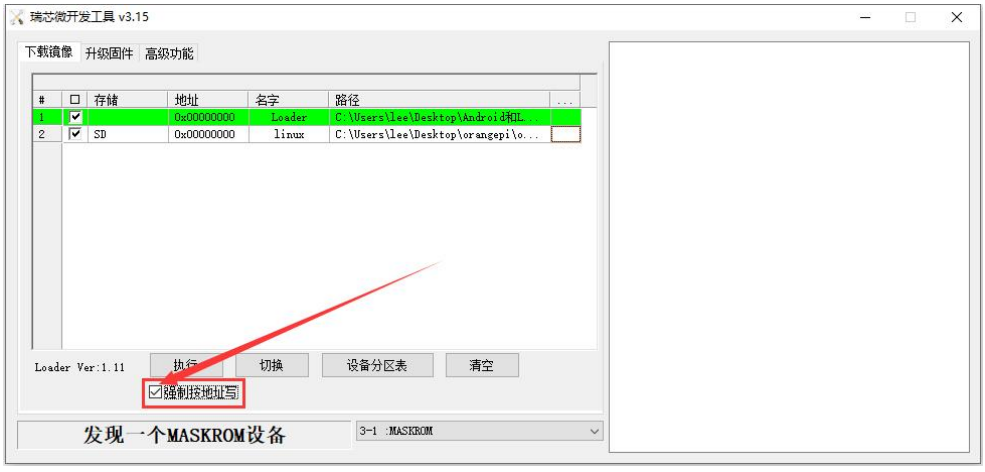


n. 然后选择想要烧录的 linux 镜像的路径，再点击**打开**。

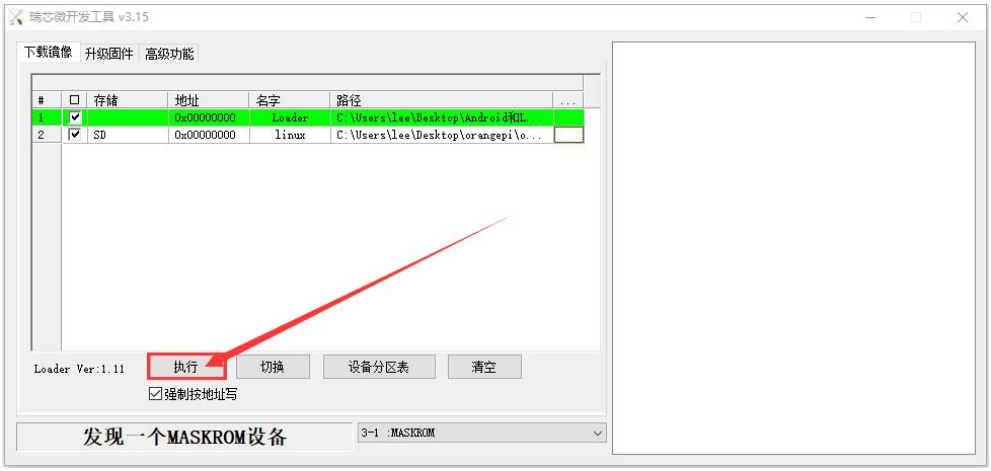
在烧录镜像前，建议将要烧录的linux镜像重命名为orangepi.img或者其它比较短的名字，这样在烧录镜像的时候就能看到烧录进度的百分比数值。



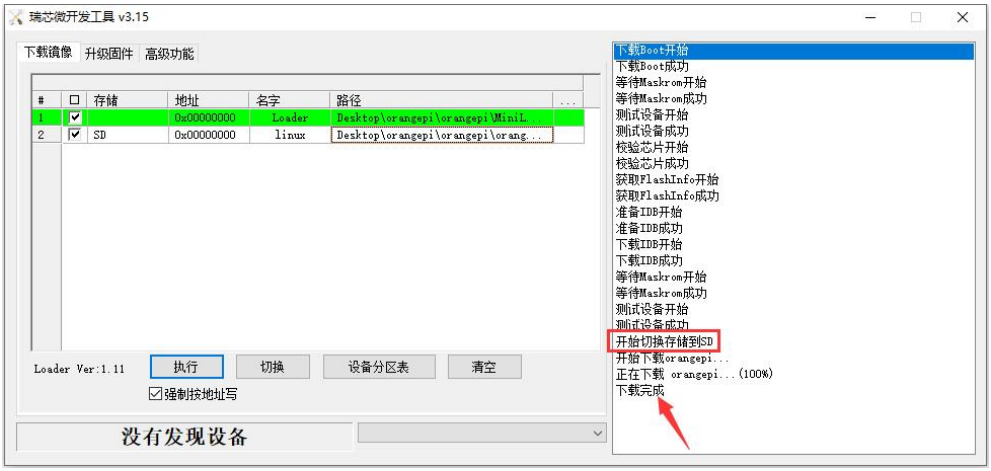
o. 然后请勾选上**强制按地址写**选项。



p. 再点击执行按钮就会开始烧录 linux 镜像到开发板的 tf 卡中。



q. linux 镜像烧录完后的显示 log 如下图所示。



r. 烧录完 linux 镜像到 tf 卡中后，linux 系统会自动启动。

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、Ubuntu、OpenWRT或者OPi OS Arch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。

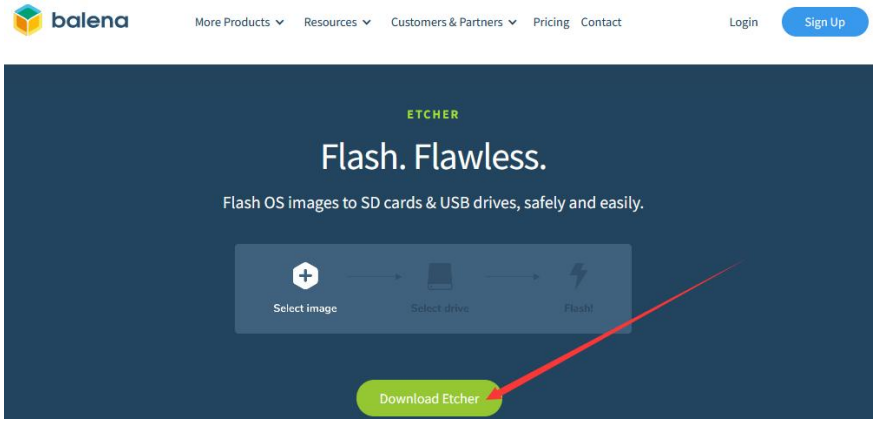
1) 首先准备一张 16GB 或更大容量的 TF 卡，TF 卡的传输速度必须为 **class10** 级或 **class10** 级以上，建议使用闪迪等品牌的 TF 卡。

2) 然后使用读卡器把 TF 卡插入电脑。

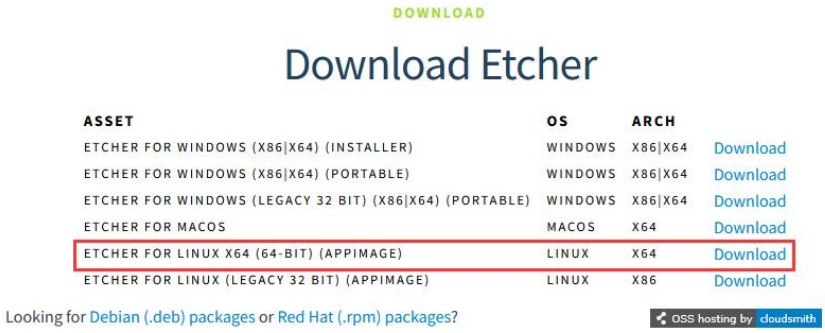
3) 下载 balenaEtcher 软件，下载地址为：

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

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



5) 然后选择下载 Linux 版本的软件即可。



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 2GB 以上。

a) 7z 结尾的压缩包的解压命令如下所示：

```
test@test:~$ 7z x orangepicm5-tablet_1.0.0_debian_bullseye_desktop_xfce_linux5.10.160.7z
test@test:~$ ls orangepicm5-tablet_1.0.0_debian_bullseye_desktop_xfce_linux5.10.160.*
orangepicm5-tablet_1.0.0_debian_bullseye_desktop_xfce_linux5.10.160.7z
orangepicm5-tablet_1.0.0_debian_bullseye_desktop_xfce_linux5.10.160.sha     #校验和文件
orangepicm5-tablet_1.0.0_debian_bullseye_desktop_xfce_linux5.10.160.img     #镜像文件
```

b) 如果下载的是 OpenWRT 镜像，压缩包是以 gz 结尾的，解压命令如下所示：

```
test@test:~$ gunzip openwrt-aarch64-opicm5-tablet-24.03-linux-6.1.43-ext4.img.gz
test@test:~$ ls openwrt-aarch64-opicm5-tablet-24.03-linux-6.1.43-ext4.img
```

```
openwrt-aarch64-opicm5-tablet-24.03-linux-6.1.43-ext4.img     #镜像文件
```

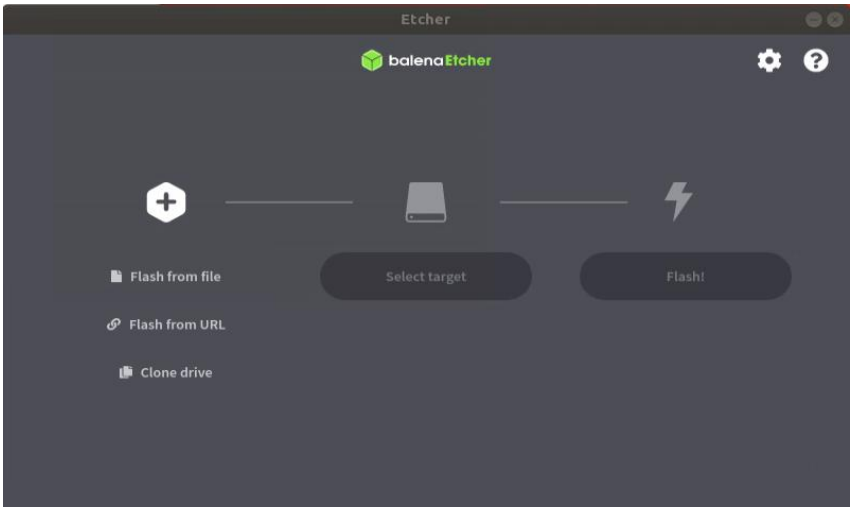
7) 解压镜像后可以先用 **sha256sum -c *.sha** 命令计算下校验和是否正确，如果提示**成功**说明下载的镜像没有错，可以放心的烧录到 TF 卡，如果提示**校验和不匹配**说明下载的镜像有问题，请尝试重新下载。

```
test@test:~$ sha256sum -c *.sha
orangepicm5-tablet_1.0.0_debian_bullseye_desktop_xfce_linux5.10.160.img: OK
```

如果下载的是 OpenWRT 镜像，要对压缩包进行校验，不要解压后再校验。

```
test@test:~$ sha256sum -c openwrt-aarch64-opicm5-tablet-24.03-linux-6.1.43-ext4.img.gz.sha
openwrt-aarch64-opicm5-tablet-24.03-linux-6.1.43-ext4.img.gz: OK
```

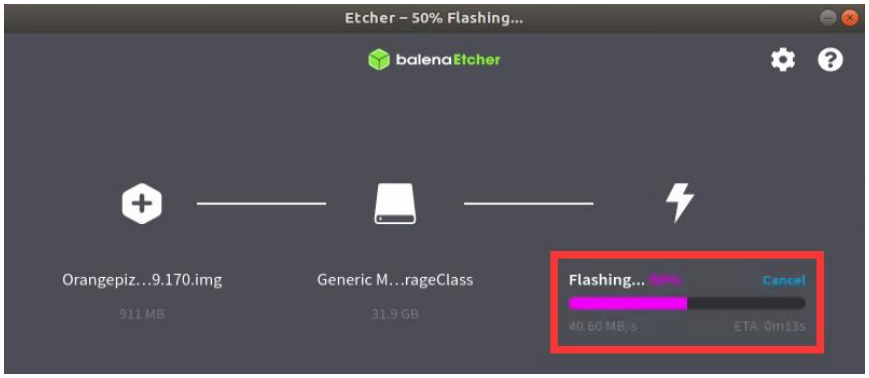
8) 然后在 Ubuntu PC 的图形界面双击 **balenaEtcher-1.5.109-x64.AppImage** 即可打开 balenaEtcher（**无需安装**），balenaEtcher 打开后的界面显示如下图所示。



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

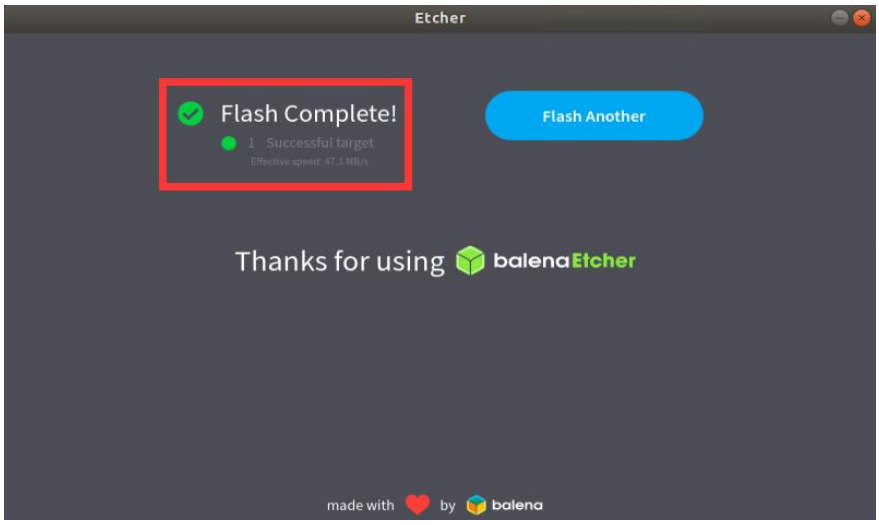
a. 首先选择要烧录的 Linux 镜像文件的路径；
b. 然后选择 TF 卡的盘符；
c. 最后点击 Flash 就会开始烧录 Linux 镜像到 TF 卡中。

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。



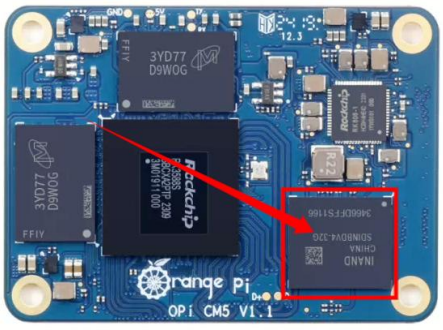
2.5. 烧录 Linux 镜像到 eMMC 中的方法

2.5.1. 使用 RKDevTool 烧录 Linux 镜像到 eMMC 中的方法

注意，下面所有的操作都是在 Windows 电脑中进行的。

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、Ubuntu、OpenWRT或者OPi OS Arch这样的Linux发行版镜像。

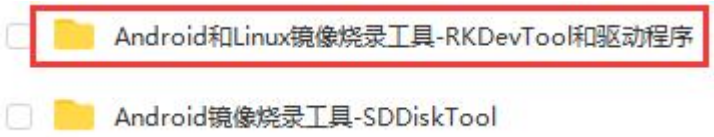
1) Orange Pi CM5 核心板上有 eMMC 模块，所在位置如下所示：



2) 首先需要准备一根品质良好的 Type-C 接口的数据线。

3) 然后从 Orange Pi 的资料下载页面下载瑞芯微驱动 **DriverAssitant_v5.12.zip** 和 **MiniLoader** 以及烧录工具 **RKDevTool_Release_v3.15.zip**。

a. 在 Orange Pi 的资料下载页面首先选择官方工具，然后进入下面的文件夹中



b. 然后下载下面的所有文件。

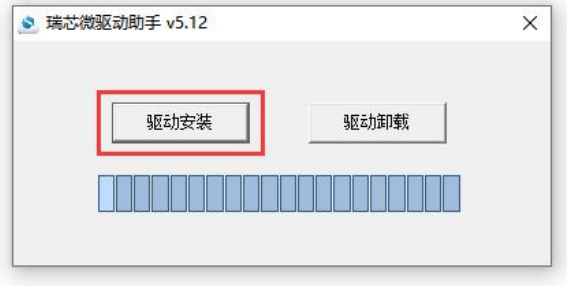
注意，“MiniLoader-烧录Linux镜像才需要用到的东西”文件夹下文简称为MiniLoader文件夹。

4) 然后从 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 2GB 以上。

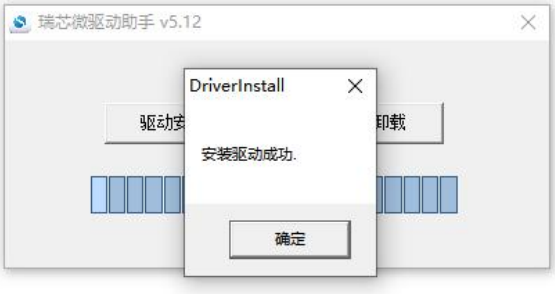
5) 然后用解压软件解压 **DriverAssitant_v5.12.zip**，再在解压后的文件夹中找到 **DriverInstall.exe** 可执行文件并打开即可。

6) 打开 **DriverInstall.exe** 后安装瑞芯微驱动的步骤如下所示：

a. 点击“**驱动安装**”按钮；

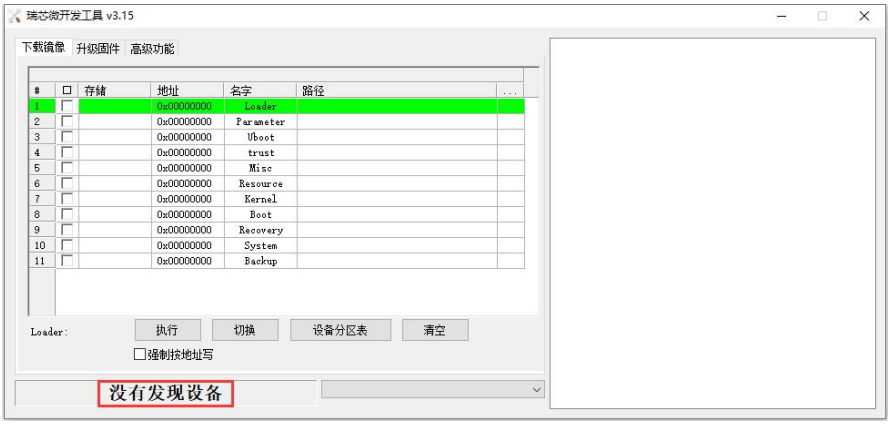


b. 等待一段时间后，会弹出窗口提示“**安装驱动成功**”，然后点击“**确定**”按钮即可。



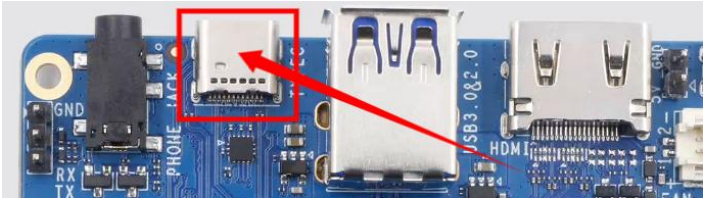
7) 然后解压 **RKDevTool_Release_v3.15.zip**，此软件无需安装，在解压后的文件夹中找到 **RKDevTool** 打开即可。

8) 打开 **RKDevTool** 烧录工具后，因为电脑此时还没有通过 USB2.0 公对公数据线连接上开发板，所以左下角会提示“**没有发现设备**”。



9) 然后开始烧录 Linux 镜像到 eMMC 中。

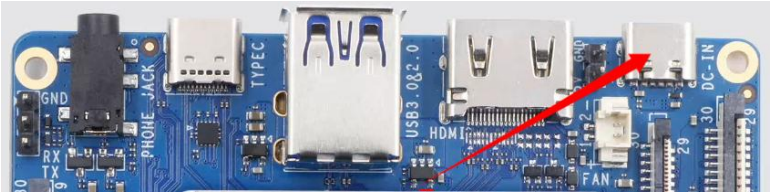
a. 首先通过 Type-C 数据线连接好开发板与 Windows 电脑，开发板 Type-C 接口的位置如下图所示：



b. 确保开发板没有插入 TF 卡，没有连接电源。

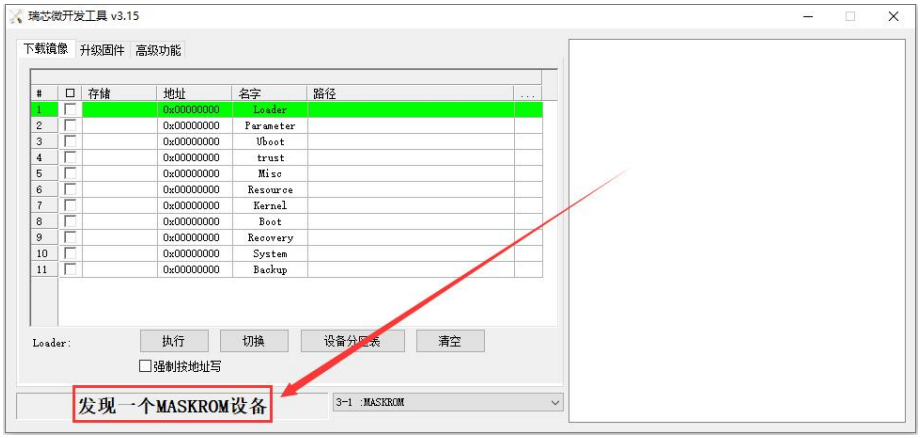
c. 然后按住开发板的 MaskROM 按键不放，MaskROM 按键在开发板的位置如下图所示：

d. 然后给开发板接上 Type-C 接口的电源，并上电，然后就可以松开 MaskROM 按键了。

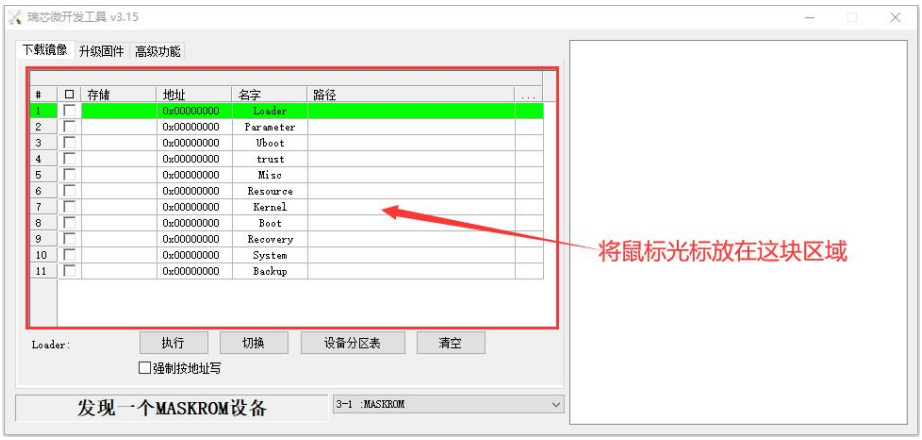


e. 如果前面的步骤顺利，此时开发板会进入 **MASKROM** 模式，在烧录工具的

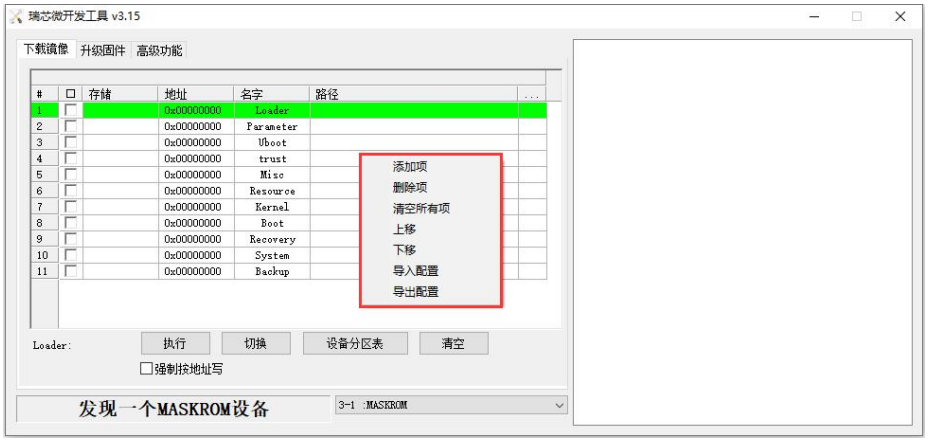
界面上会提示“**发现一个 MASKROM 设备**”。



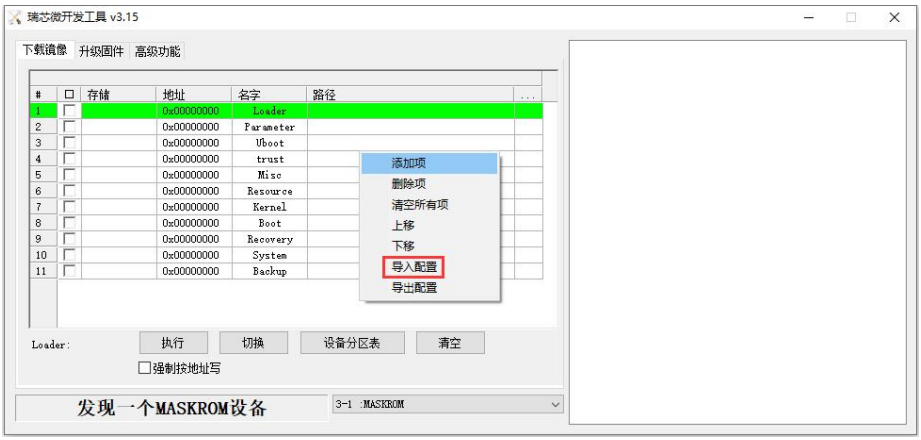
f. 然后将鼠标光标放在下面的这片区域中。



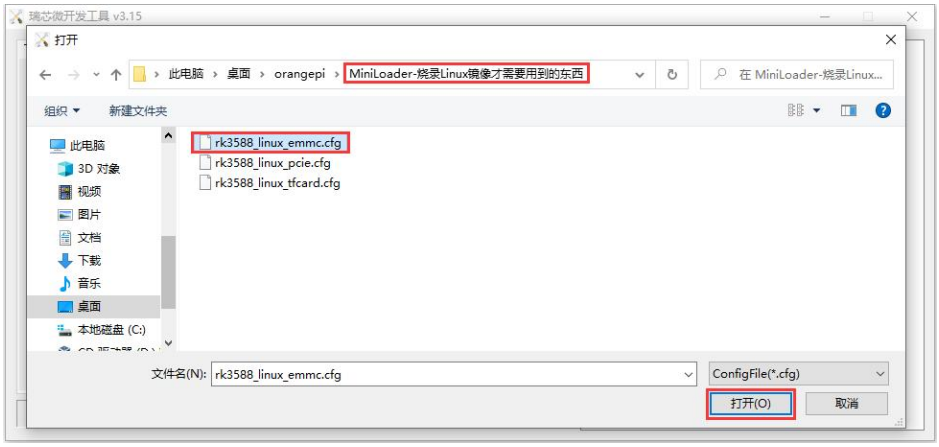
g. 然后点击鼠标右键会弹出下图所示的选择界面。



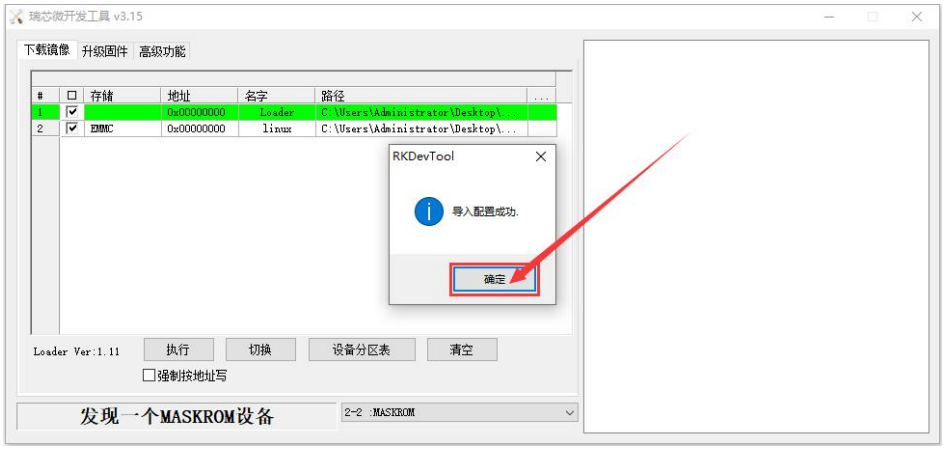
h. 然后选择**导入配置**选项。



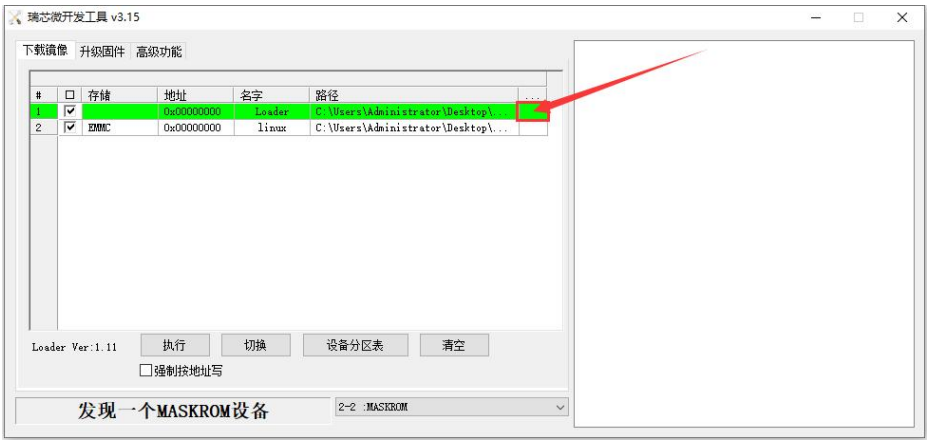
i. 然后选择前面下载的 **MiniLoader** 文件夹中的 **rk3588_linux_emmc.cfg** 配置文件，再点击**打开**。



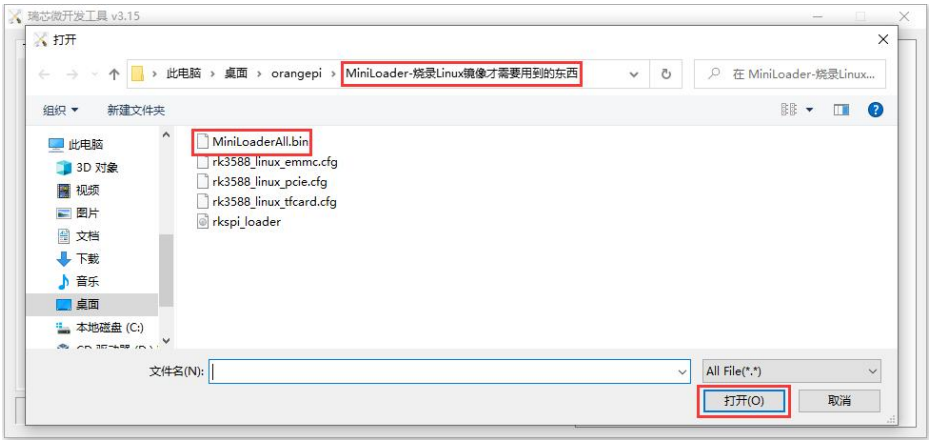
j. 然后点击**确定**。



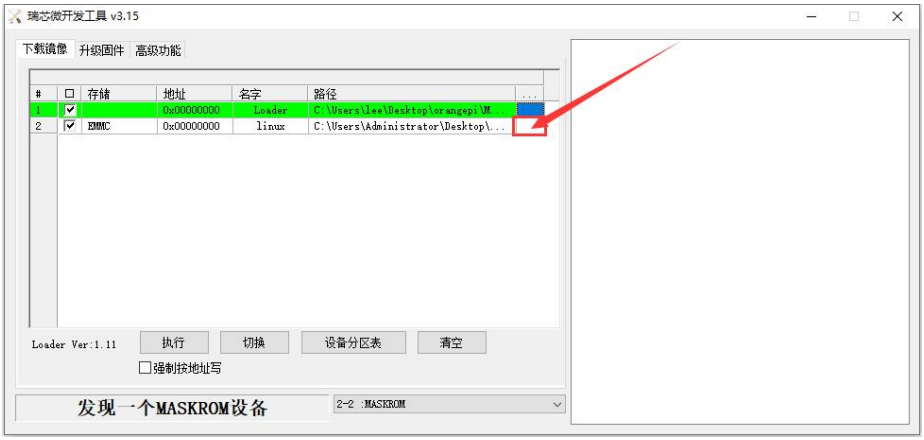
k. 然后点击下图所示的位置。



l. 再选择前面下载的 **MiniLoader** 文件夹中 **MiniLoaderAll.bin**，再点击**打开**。

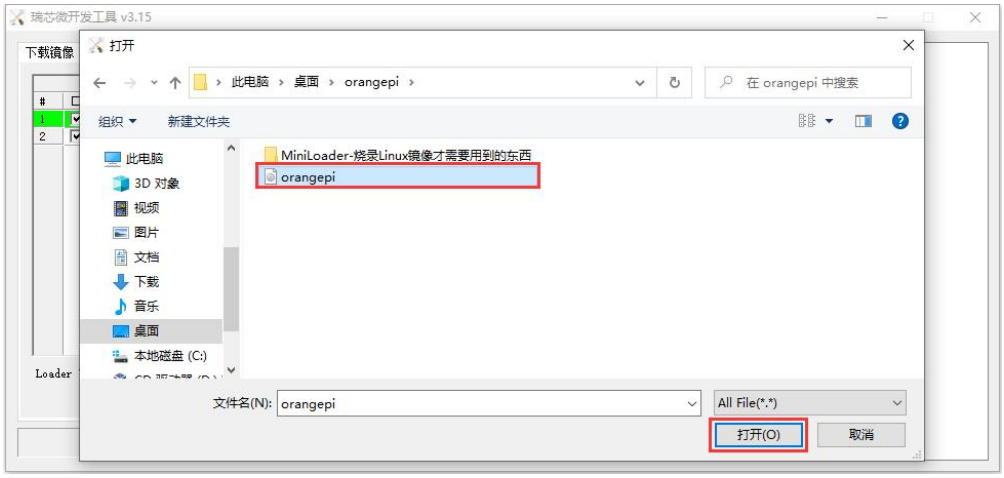


m. 然后点击下图所示的位置。

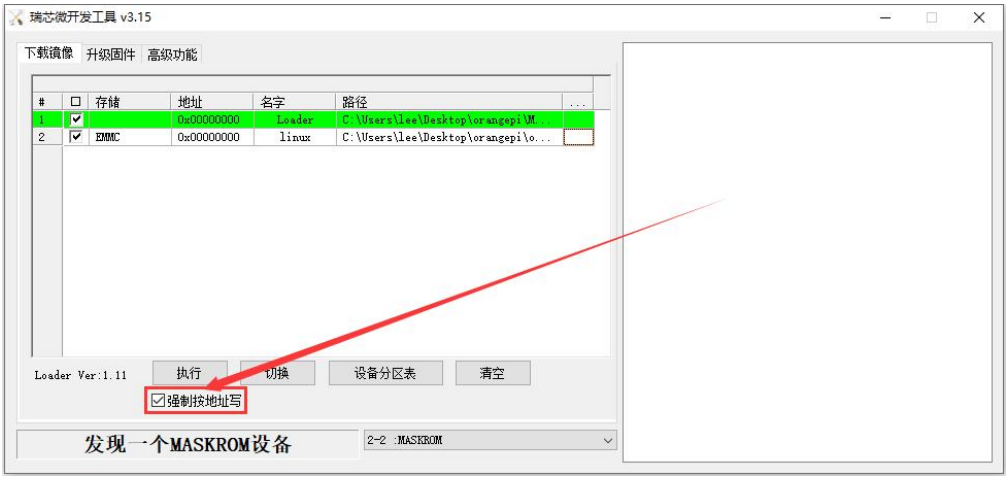


n. 然后选择想要烧录的 linux 镜像的路径，再点击**打开**。

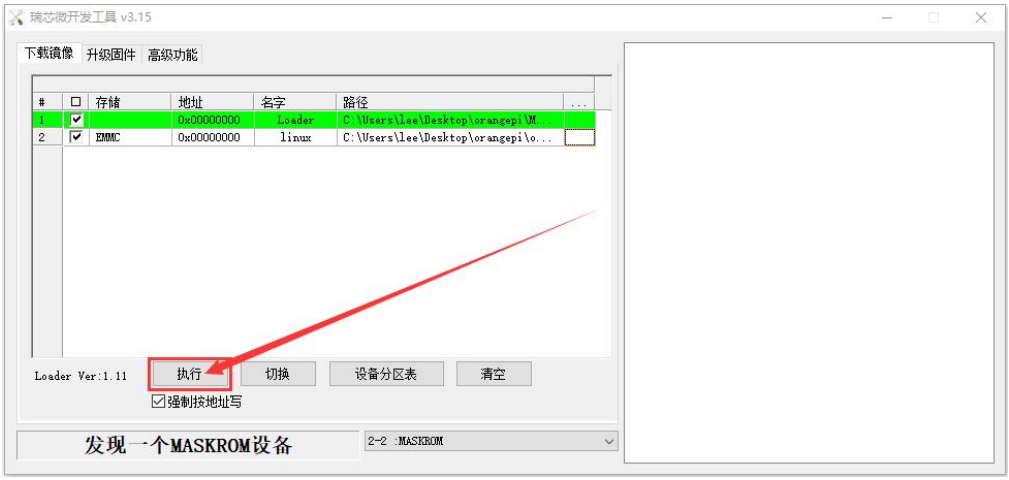
在烧录镜像前，建议将要烧录的linux镜像重命名为orangepi.img或者其它比较短的名字，这样在烧录镜像的时候就能看到烧录进度的百分比数值。



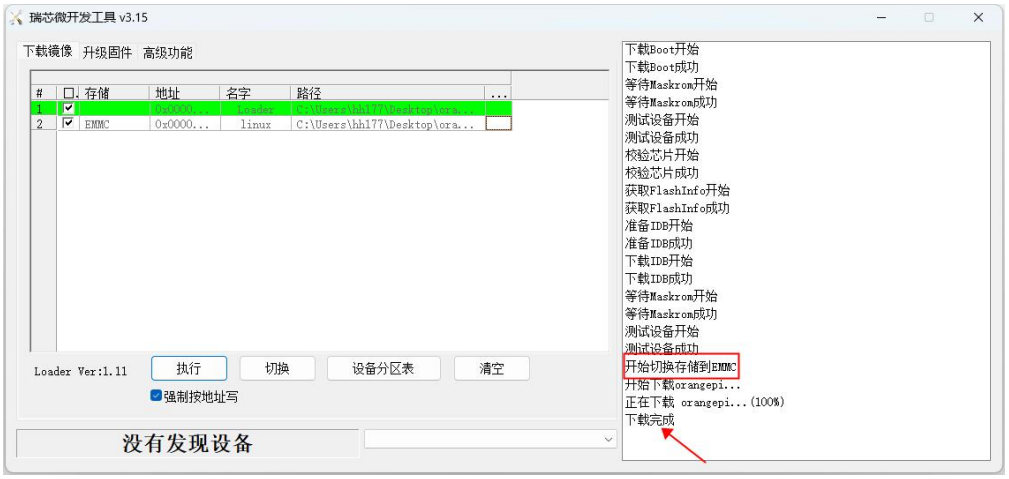
o. 然后请勾选上**强制按地址写**选项。



p. 再点击执行按钮就会开始烧录 linux 镜像到开发板的 eMMC 中。



q. linux 镜像烧录完后的显示 log 如下图所示：

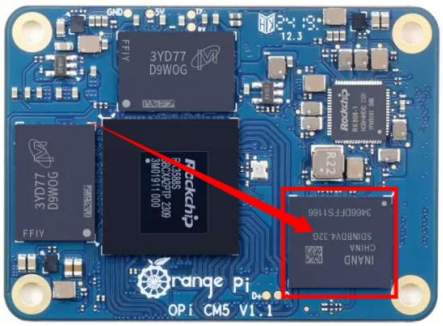


r. 烧录完 linux 镜像到 eMMC 中后，linux 系统会自动启动。

2.5.2. 使用 dd 命令烧录 Linux 镜像到 eMMC 中的方法

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、Ubuntu、OpenWRT或者OPi OS Arch这样的Linux发行版镜像。

1) Orange Pi CM5 核心板上有 eMMC 模块，所在位置如下所示：



2) 使用 dd 命令烧录 linux 镜像到 eMMC 中需要借助 TF 卡来完成，所以首先需要将 linux 镜像烧录到 TF 卡上，然后使用 TF 卡启动开发板进入 linux 系统。烧录 Linux 镜像到 TF 卡的方法请见**基于 Windows PC 将 Linux 镜像烧录到 TF 卡的方法**和**基于 Ubuntu PC 将 Linux 镜像烧录到 TF 卡的方法**两小节的说明。

3) 使用 TF 卡启动 linux 系统后，我们首先将解压后的 linux 镜像文件（从官网下载的 Debian、Ubuntu 镜像或者 OPi OS Arch 镜像）上传到 TF 卡中。上传 linux 镜像文件到开发板中的方法请参考**上传文件到开发板 Linux 系统中的方法**小节的说明。

4) 上传完镜像到开发板的 linux 系统中后，我们再在开发板 linux 系统的命令行中进

入镜像文件的存放路径，比如，我将开发板的 linux 镜像存放在 **/home/orangepi/Desktop** 目录下了，然后进入**/home/orangepi/Desktop** 目录就能看到上传的镜像文件了。

```
orangepi@orangepi:~$ cd /home/orangepi/Desktop
orangepi@orangepi:~/Desktop$ ls
orangepicm5-tablet_x.x.x_debian_bullseye_desktop_xfce_linux5.10.160.img
```

怎么进入开发板linux系统的命令行？

1. **使用串口登录终端的方法请参考调试串口的使用方法一小节的说明。**
2. **使用ssh远程登录linux系统请参考SSH远程登录开发板一小节的说明。**
3. **如果接了HDMI、LCD等显示屏幕，可以在桌面中打开一个命令行终端。**

5) 接下来，我们先使用下面的命令确认下 eMMC 的设备节点。

```
orangepi@orangepi:~/Desktop$ ls /dev/mmcblk*boot0 | cut -c1-12
/dev/mmcblk1
```

6) 然后我们可以使用 dd 命令清空下 eMMC，注意 **of=**参数后面请填入上面命令输出的结果。

```
orangepi@orangepi:~/Desktop$ sudo dd bs=1M if=/dev/zero of=/dev/mmcblk1 count=1000 status=progress
orangepi@orangepi:~/Desktop$ sudo sync
```

7) 然后就可以使用 dd 命令烧录开发板的 linux 镜像到 eMMC 中。

a. 下面的命令中 **if=** 参数后面是要填写 linux 镜像存放的完整路径+Linux 镜像的名字（比如**/home/orangepi/Desktop/Linux 镜像的名字**）。因为上面我们已经进入 linux 镜像的路径下了，所以只需要填写 Linux 镜像的名字的即可。

b. 下面命令中的 linux 镜像名请不要照抄，要替换为实际的镜像名（因为镜像的版本号可能会更新）。

```
sudo dd bs=1M if=orangepicm5-tablet_x.x.x_debian_bullseye_desktop_xfce_linux5.10.160.img of=/dev/mmcblk1 status=progress

sudo sync
```

注意，如果上传的是 .7z或者.xz 结尾linux镜像压缩文件，使用dd命令烧录前请记得先解压。

dd命令的所有参数的详细说明和更多用法可以在linux系统中执行man dd命令来查看。

8) 当成功烧录开发板的 linux 镜像到 eMMC 后，此时就可以使用 **poweroff** 命令关机了。然后请拔出 TF 卡，再短按电源按键开机，此时就会启动 eMMC 中的 linux 系统了。

2.6. 烧录 Android 镜像到 TF 卡中的方法

2.6.1. 使用 RKDevTool 烧录的方法

1) 首先需要准备一根品质良好的 Type-C 接口的数据线。

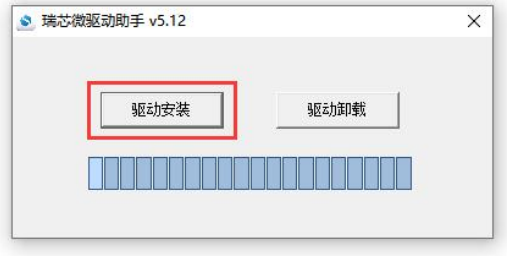
2) 然后从 Orange Pi 的资料下载页面下载瑞芯微驱动 **DriverAssitant_v5.12.zip** 和烧录工具 **RKDevTool_Release_v3.15.zip**。

3) 然后从 Orange Pi 的资料下载页面下载 Android 的镜像。

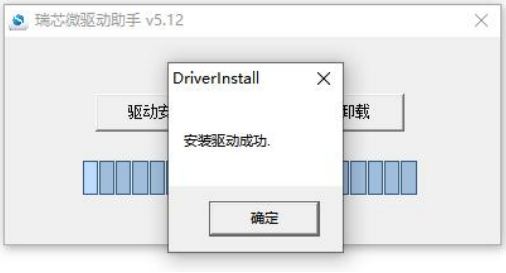
4) 然后用解压软件解压 **DriverAssitant_v5.12.zip**，再在解压后的文件夹中找到 **DriverInstall.exe** 可执行文件并打开即可。

5) 打开 **DriverInstall.exe** 后安装瑞芯微驱动的步骤如下所示：

a. 点击“**驱动安装**”按钮。

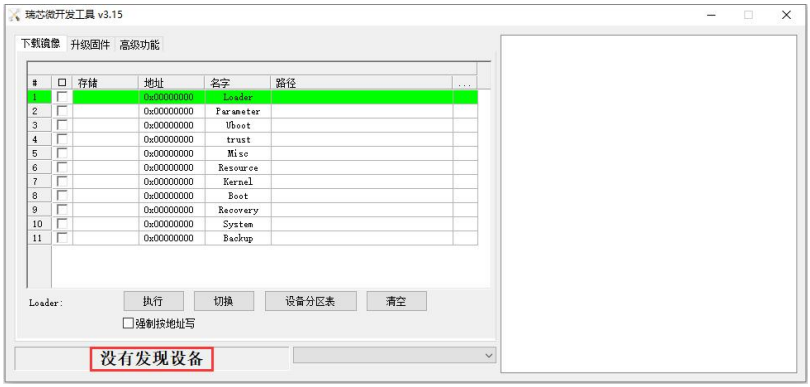


b. 等待一段时间后，会弹出窗口提示“**安装驱动成功**”，然后点击“**确定**”按钮即可。



6) 然后解压 **RKDevTool_Release_v3.15.zip**，此软件无需安装，在解压后的文件夹中找到 **RKDevTool** 打开即可。

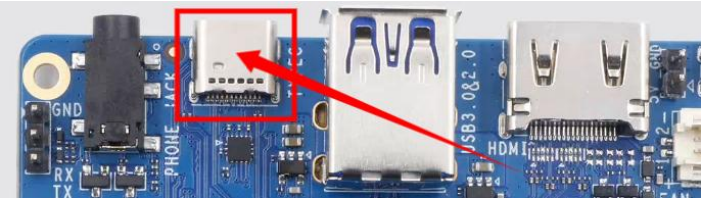
7) 打开 **RKDevTool** 烧录工具后，因为电脑此时还没有通过 USB2.0 公对公数据线连接上开发板，所以左下角会提示“**没有发现设备**”。



8) 然后开始烧录 Android 镜像到 TF 卡中。

a. 首先通过 Type-C 数据线连接好开发板与 Windows 电脑，开发板 Type-C 接

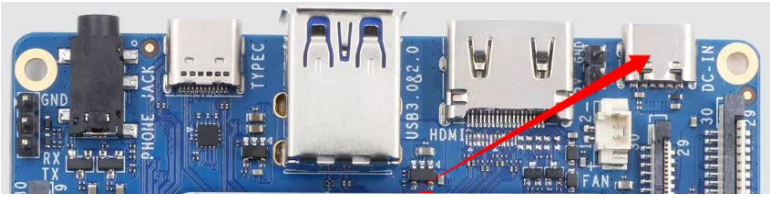
口的位置如下图所示。



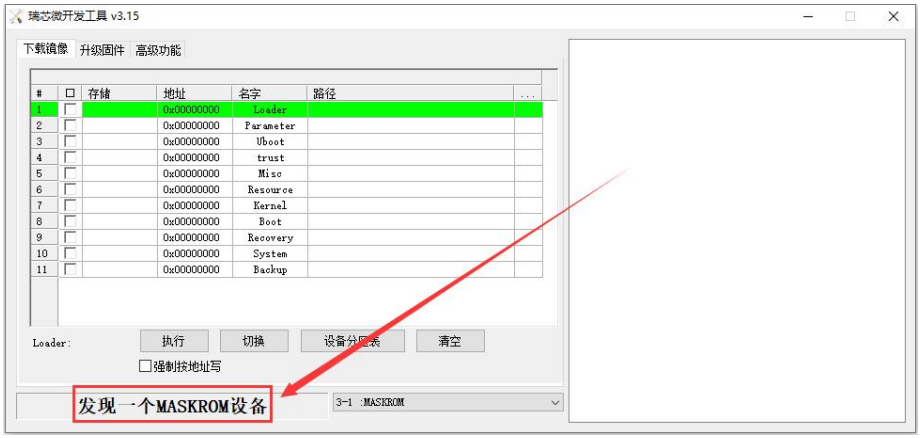
b. 然后插入 TF 卡到开发板，并确保开发板没有连接电源。

c. 然后按住开发板的 MaskROM 按键不放，MaskROM 按键在开发板的位置如下图所示：

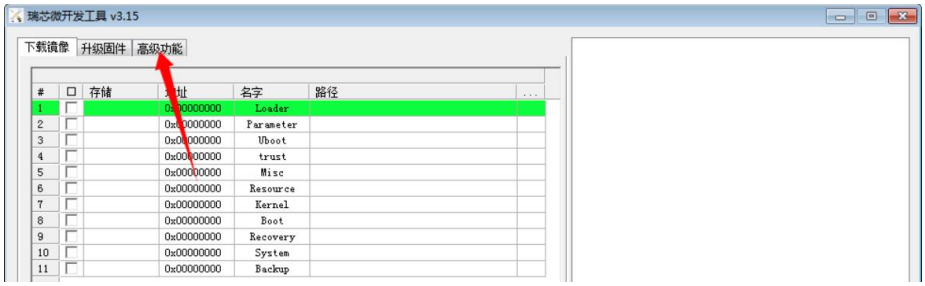
d. 然后给开发板接上 Type-C 接口的电源，并上电，然后就可以松开 MaskROM 按键了。



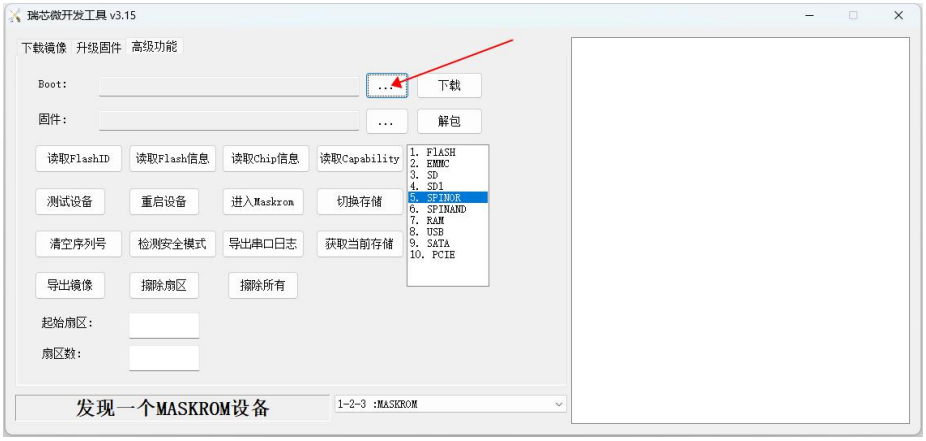
e. 如果前面的步骤顺利，此时开发板会进入 **MASKROM** 模式，在烧录工具的界面上会提示“**发现一个 MASKROM 设备**”。



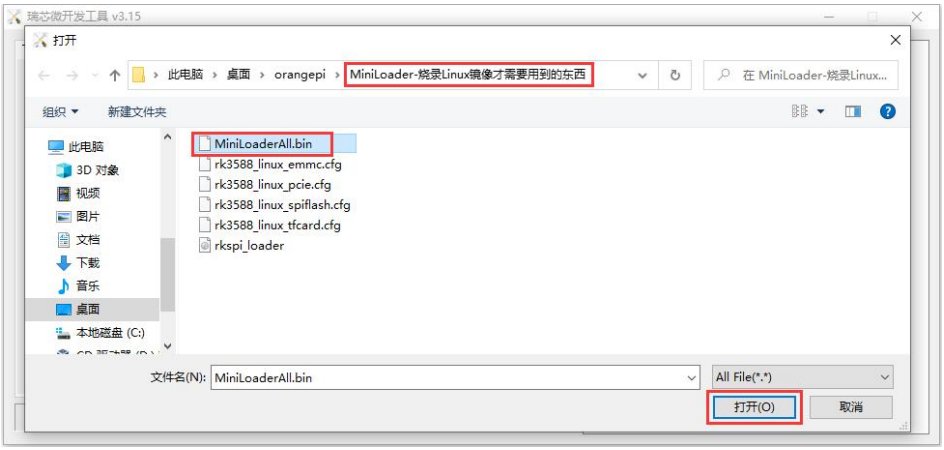
f. 然后请选择**高级功能**。



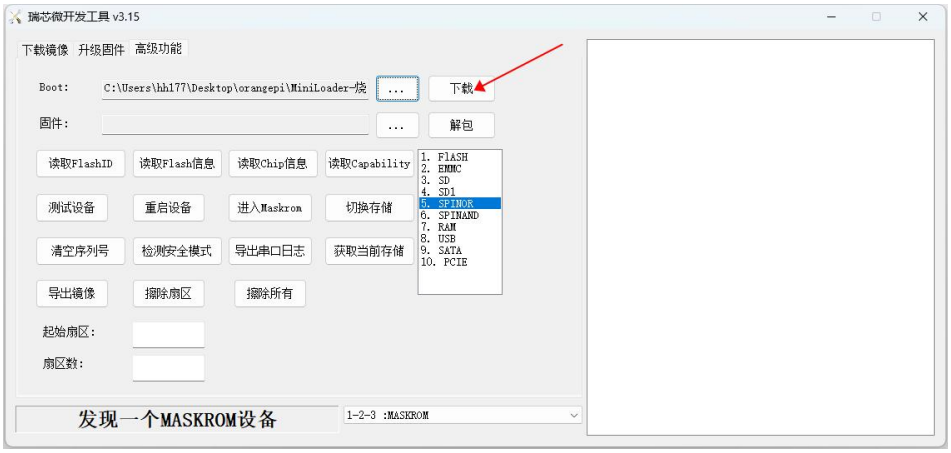
g. 然后点击下图所示的位置。



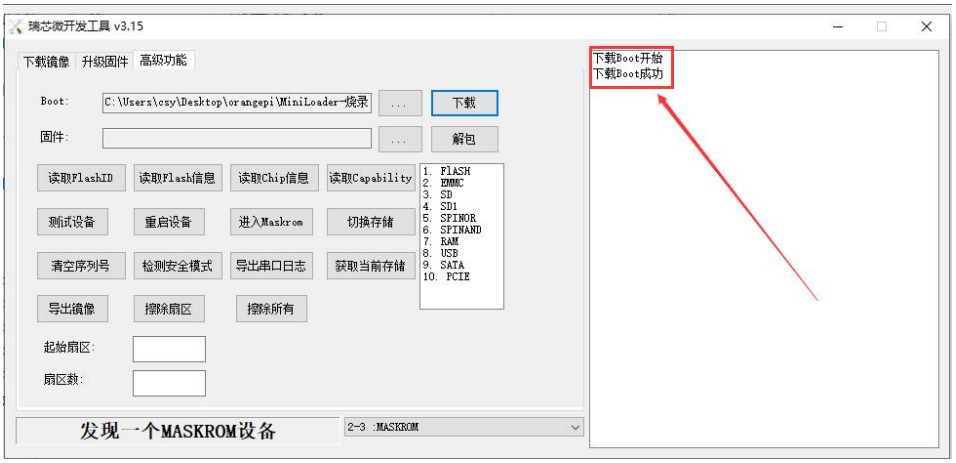
h. 再选择前面下载的 **MiniLoader** 文件夹中的 **MiniLoaderAll.bin**，再点击打开。



i. 然后点击**下载**。

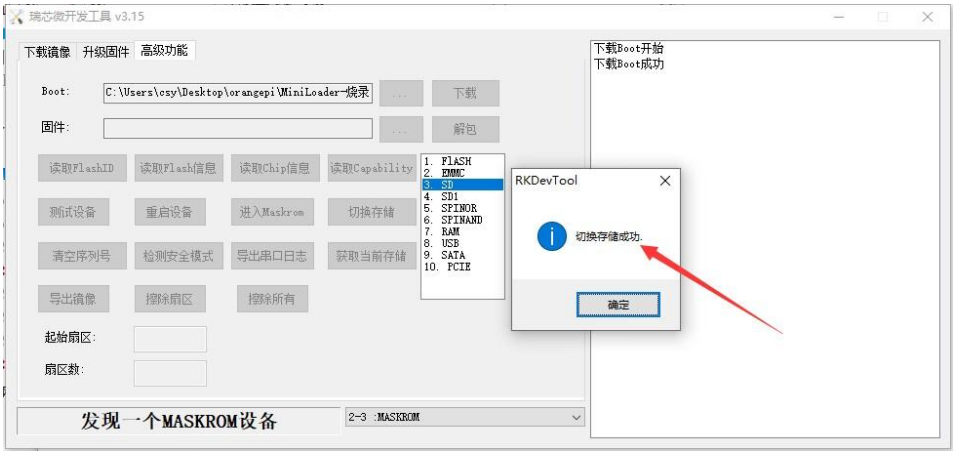


j. 下载完 **MiniLoaderAll.bin** 后的显示如下图所示。



k. 然后选择存储设备为 **SD**，再点击**切换存储**。

l. 切换成功的显示如下图所示。



m. 然后点击烧录工具的“**升级固件**”一栏。

n. 接着点击“**固件**”按钮选择需要烧录的 Android 镜像的路径。

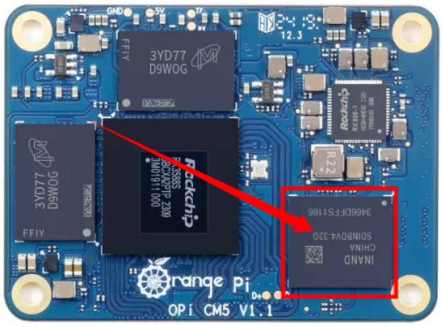
o. 最后点击“**升级**”按钮就会开始烧录，烧录过程中的 log 如下图所示。烧录完成后 Android 系统会自动启动。

2.7. 烧录 Android 镜像到 eMMC 中的方法

2.7.1. 使用 RKDevTool 烧录的方法

注意，下面所有的操作都是在 Windows 电脑中进行的。

1) Orange Pi CM5 核心板上有 eMMC 模块，所在位置如下所示：



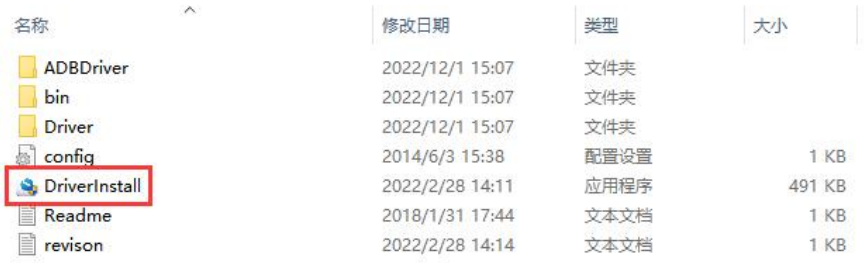
2) 还需要准备一根品质良好的 Type-C 接口的数据线。

3) 然后从 Orange Pi 的资料下载页面下载瑞芯微驱动 **DriverAssitant_v5.12.zip** 和

烧录工具 **RKDevTool_Release_v3.15.zip**。

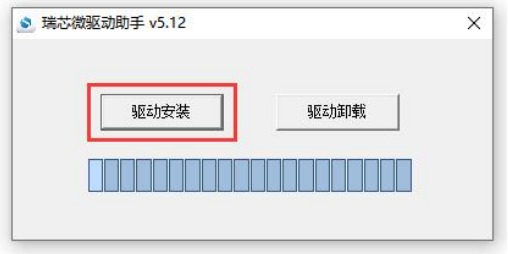
4) 然后从 Orange Pi 的资料下载页面下载 Android 的镜像。

5) 然后用解压软件解压 **DriverAssitant_v5.12.zip**，再在解压后的文件夹中找到 **DriverInstall.exe** 可执行文件并打开即可。

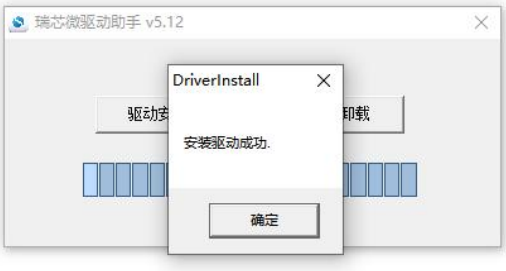


6) 打开 **DriverInstall.exe** 后安装瑞芯微驱动的步骤如下所示：

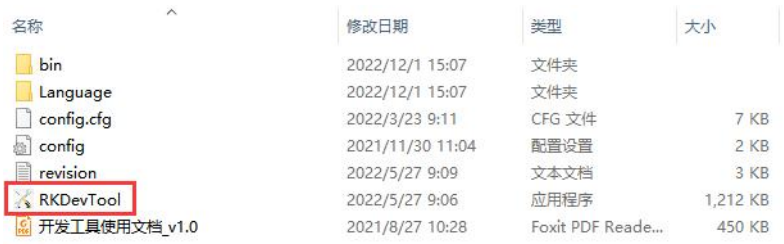
a. 点击“**驱动安装**”按钮。



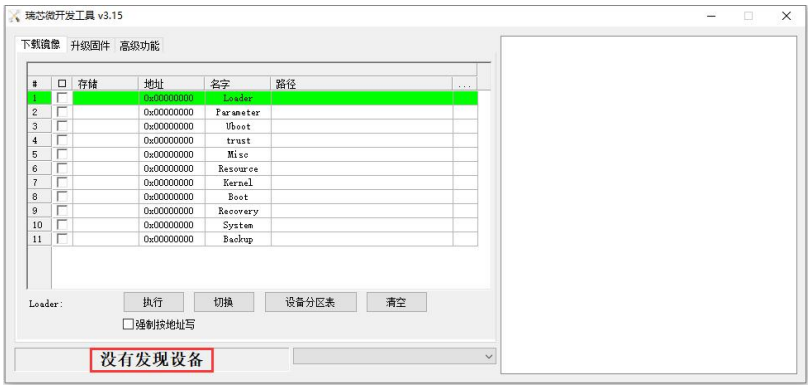
b. 等待一段时间后，会弹出窗口提示“**安装驱动成功**”，然后点击“**确定**”按钮即可。



7) 然后解压 **RKDevTool_Release_v3.15.zip**，此软件无需安装，在解压后的文件夹中找到 **RKDevTool** 打开即可。

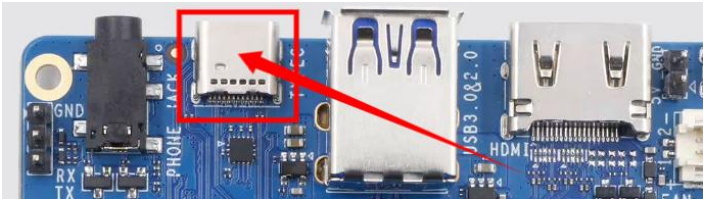


8) 打开 **RKDevTool** 烧录工具后，因为电脑此时还没有通过 Type-C 数据线连接上开发板，所以左下角会提示“**没有发现设备**”。



9) 然后开始烧录安卓镜像到 eMMC 中。

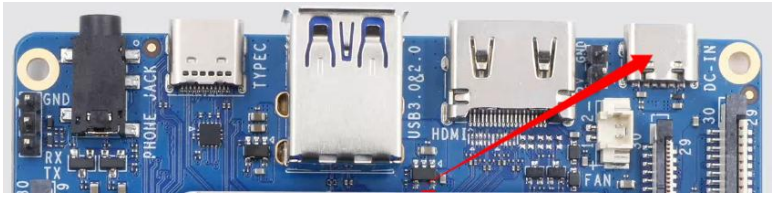
a. 首先通过 Type-C 数据线连接好开发板与 Windows 电脑，开发板 Type-C 接口的位置如下图所示：



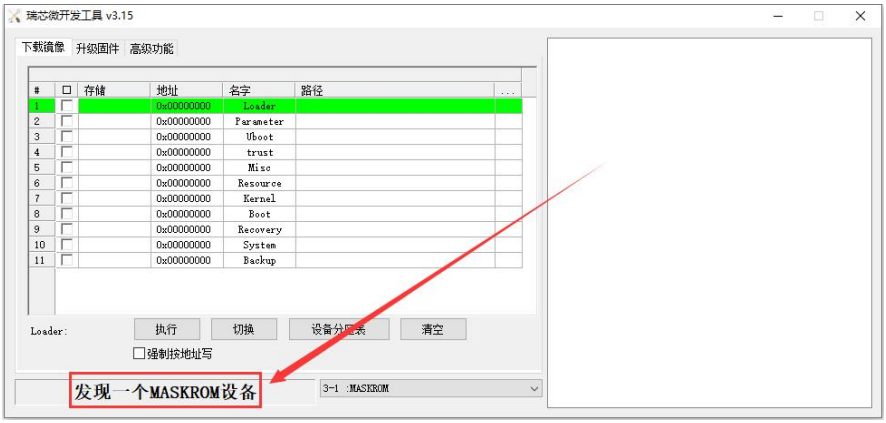
b. 确保开发板没有连接电源和插入 TF 卡。

c. 然后按住开发板的 MaskROM 按键不放，MaskROM 按键在开发板的位置如下图所示：

d. 然后给开发板接上 Type-C 接口的电源，并上电，然后就可以松开 MaskROM 按键了。



e. 如果前面的步骤顺利，此时开发板会进入 **MASKROM** 模式，在烧录工具的界面上会提示“**发现一个 MASKROM 设备**”。



f. 然后点击烧录工具的“**升级固件**”一栏。

g. 接着点击“**固件**”按钮选择需要烧录的 Android 镜像的路径。

h. 最后点击“**升级**”按钮就会开始烧录，烧录过程中的 log 如下图所示。烧录完成后 Android 系统会自动启动。

2.8. 启动香橙派开发板

1) 核心板的 eMMC 中预装有安卓镜像，我们可以直接启动使用它。或者将烧录好镜像的 TF 卡插入香橙派开发板的 TF 卡插槽中。

2) 开发板有 HDMI 接口，可以通过 HDMI 转 HDMI 连接线把开发板连接到电视或者 HDMI 显示器。如果有购买 LCD 屏幕，也可以使用 LCD 屏幕来显示开发板的系统界面。

3) 接上 USB 鼠标和键盘，用于控制香橙派开发板。

4) 连接一个 5V/4A 或者 5V/5A 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。

Type-C 电源接口是不支持 PD 协商的。

另外请不要接到电脑的 USB 接口来给开发板供电。

5) 然后打开电源适配器的开关，如果一切正常，此时 HDMI 显示器或者 LCD 屏幕就能看到系统的启动画面了。

6) 如果想通过调试串口查看系统的输出信息，请使用串口线将开发板连接到电脑，串口的连接方法请参看调试串口的使用方法一节。

2.9. 调试串口的使用方法

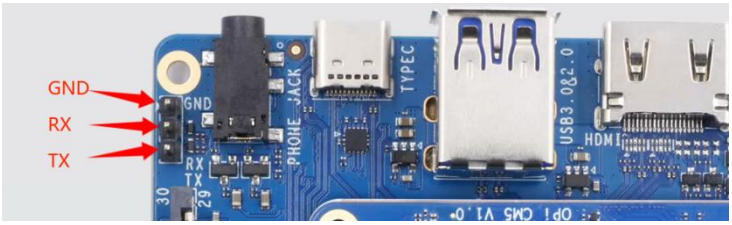
2.9.1. 调试串口的连接说明

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。

为了更好的兼容性，推荐使用 CH340 USB 转 TTL 模块，请不要使用 CP2102、PL2303 类型的 USB 转 TTL 模块。

购买 USB 转 TTL 模块前，请确认模块支持 1500000 速率的波特率。

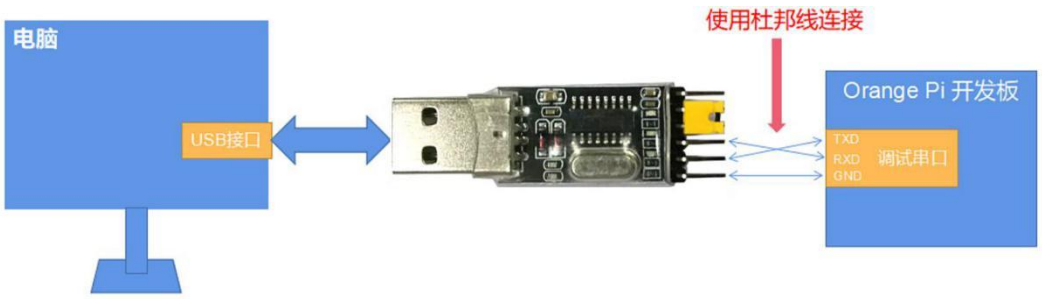
2) 底板的调试串口 GND、RX 和 TX 引脚的对应关系如下图所示：



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。

2.9.2. 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 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
```

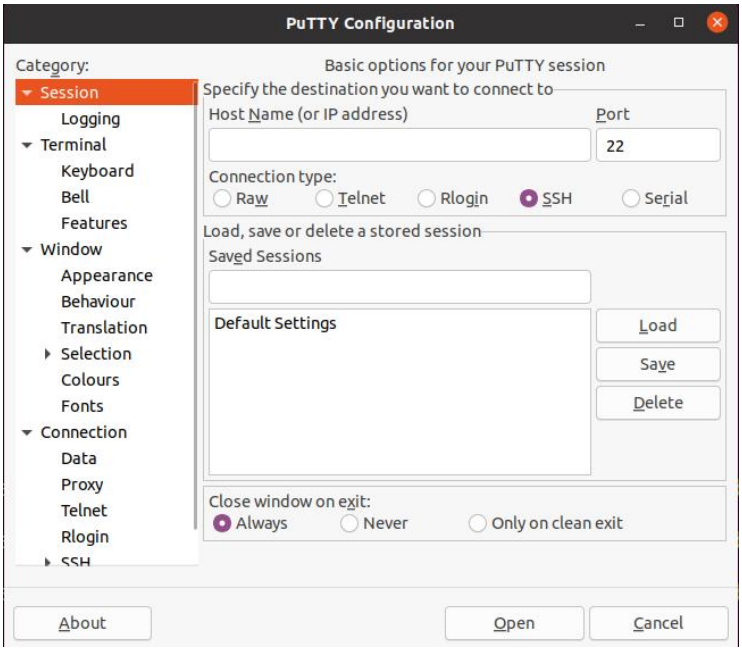
2) 然后使用下面的命令在 Ubuntu PC 上安装下 putty。

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

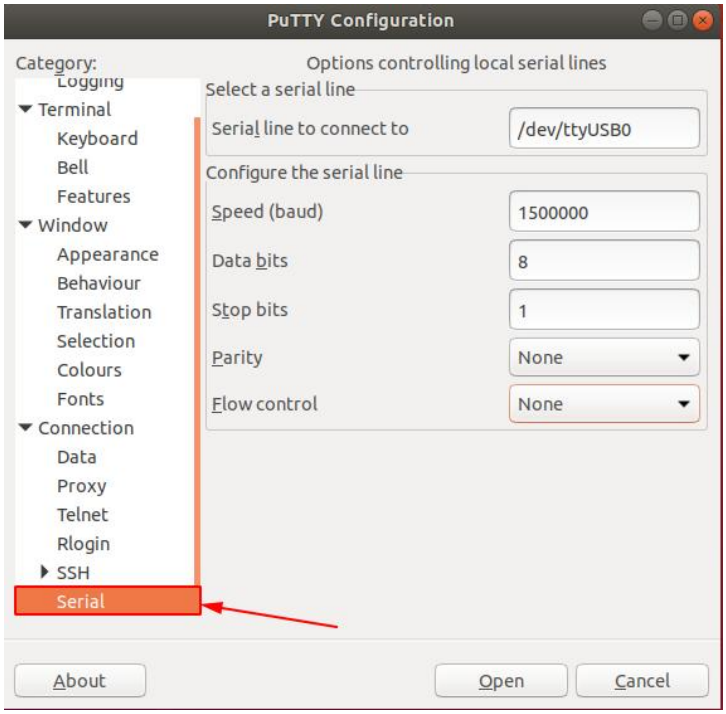
3) 然后运行 putty，**记得加 sudo 权限**。

```
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。

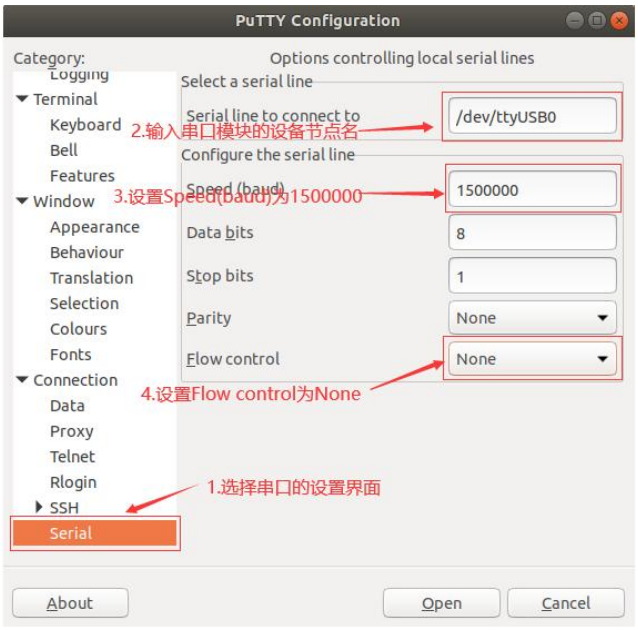


6) 然后设置串口的参数。

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 **1500000**（串口的波特率）。

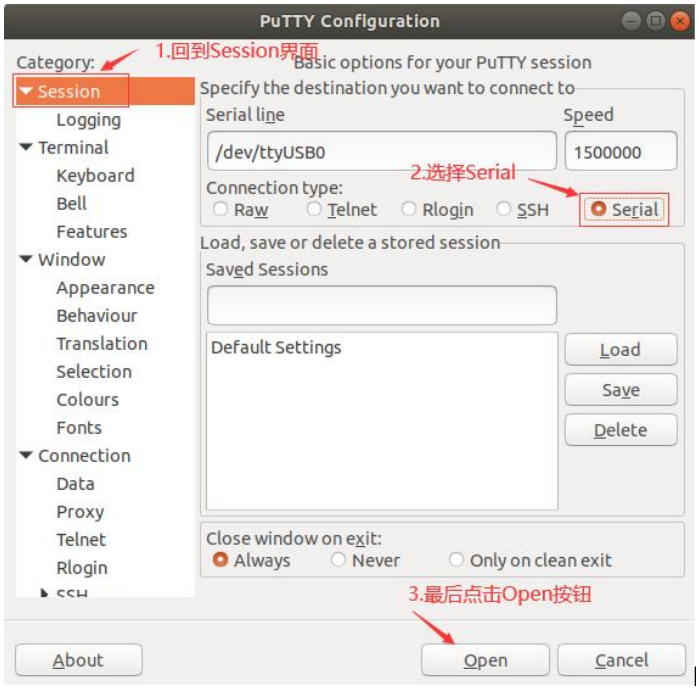
c. 设置 Flow control 为 None。



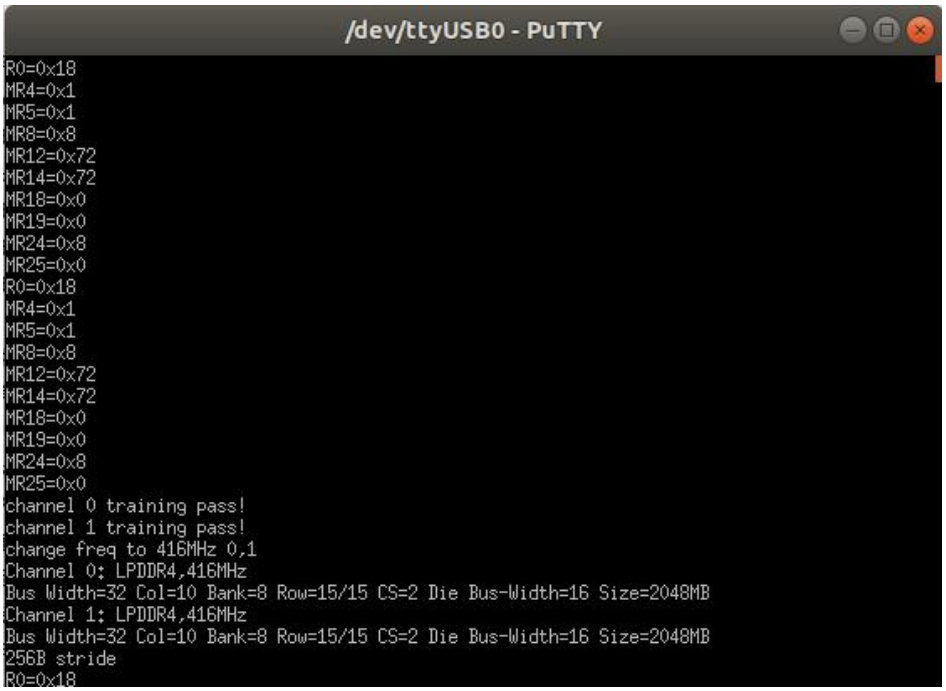
7) 在串口的设置界面设置完后，再回到 Session 界面。

a. 首先选择 **Connection type** 为 Serial。

b. 然后点击 **Open** 按钮连接串口。



8) 启动开发板后，就能从打开的串口终端中看到系统输出的 Log 信息了。



2.9.3. Windows 平台调试串口的使用方法

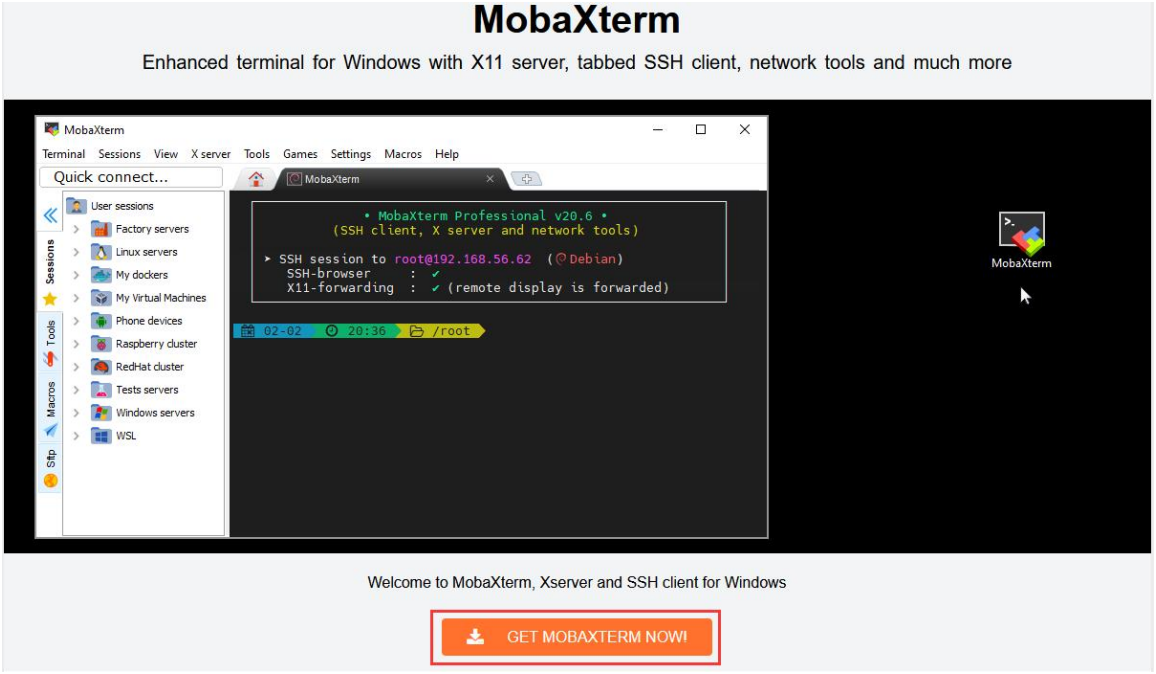
Windows 下可以使用的串口调试软件有很多，如 SecureCRT、MobaXterm 等，下面演示 MobaXterm 的使用方法，这款软件有免费版本，无需购买序列号即可使用。

1) 下载 MobaXterm。

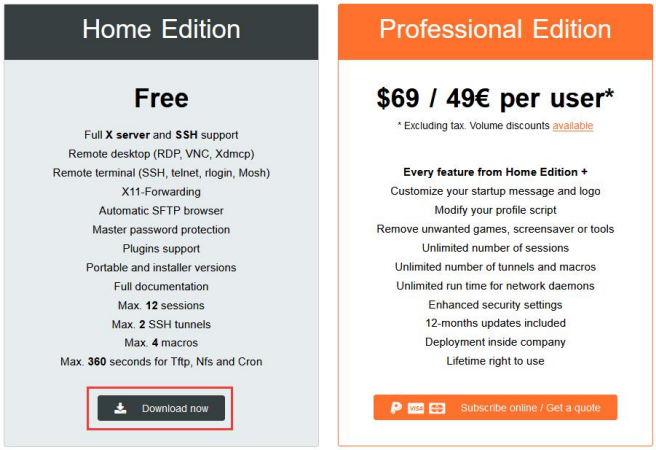
a. 下载 MobaXterm 网址如下：

https://mobaxterm.mobatek.net

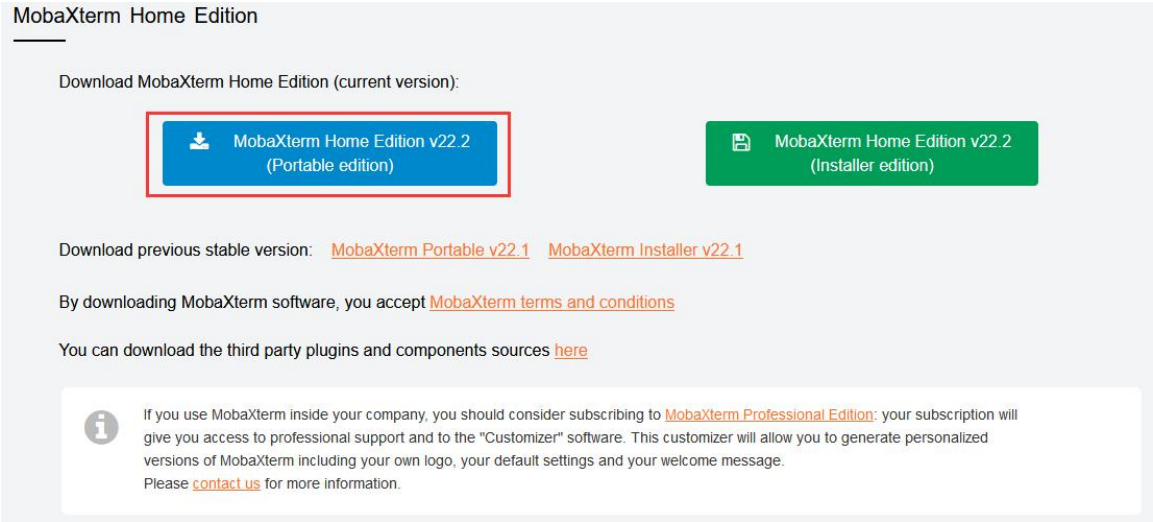
b. 进入 MobaXterm 下载网页后点击 **GET XOBATERM NOW!**。



c. 然后选择下载 Home 版本。



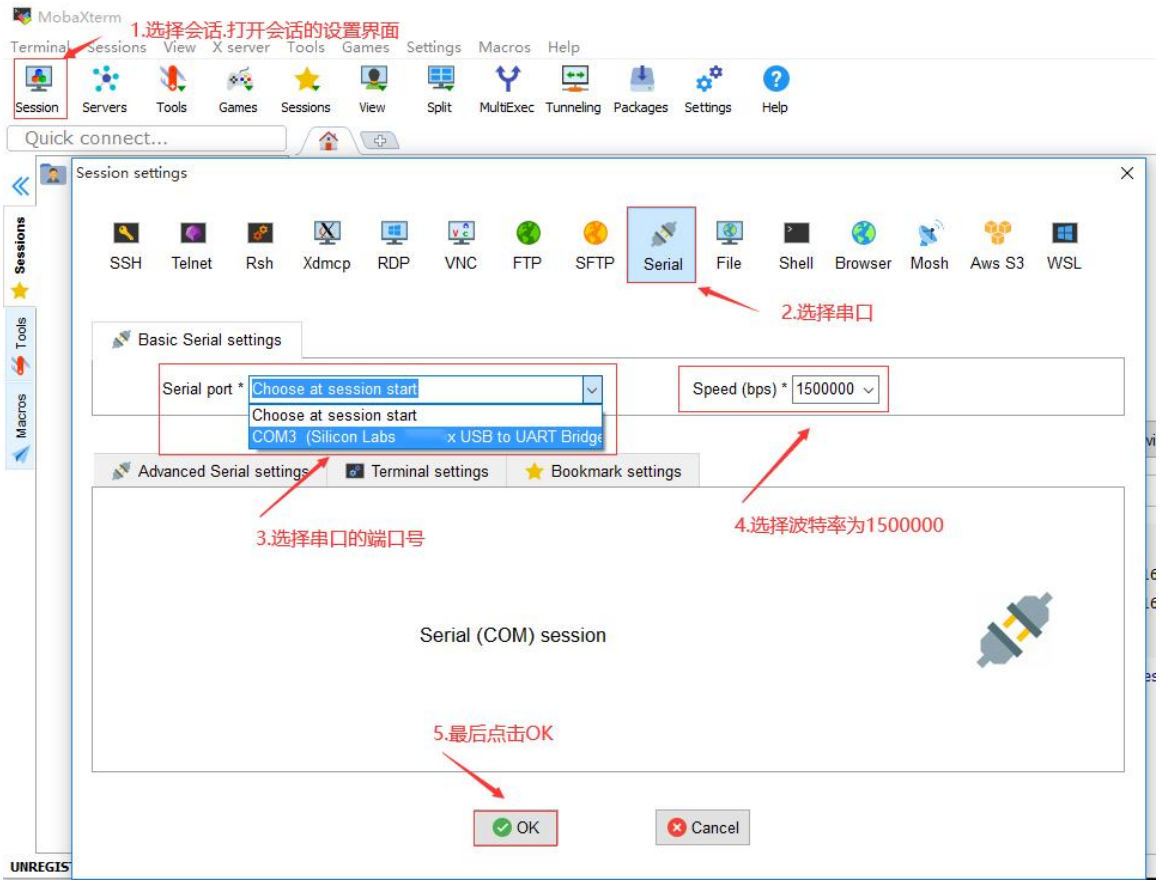
d. 然后选择 Portable 便携式版本，下载完后无需安装，直接打开就可以使用。



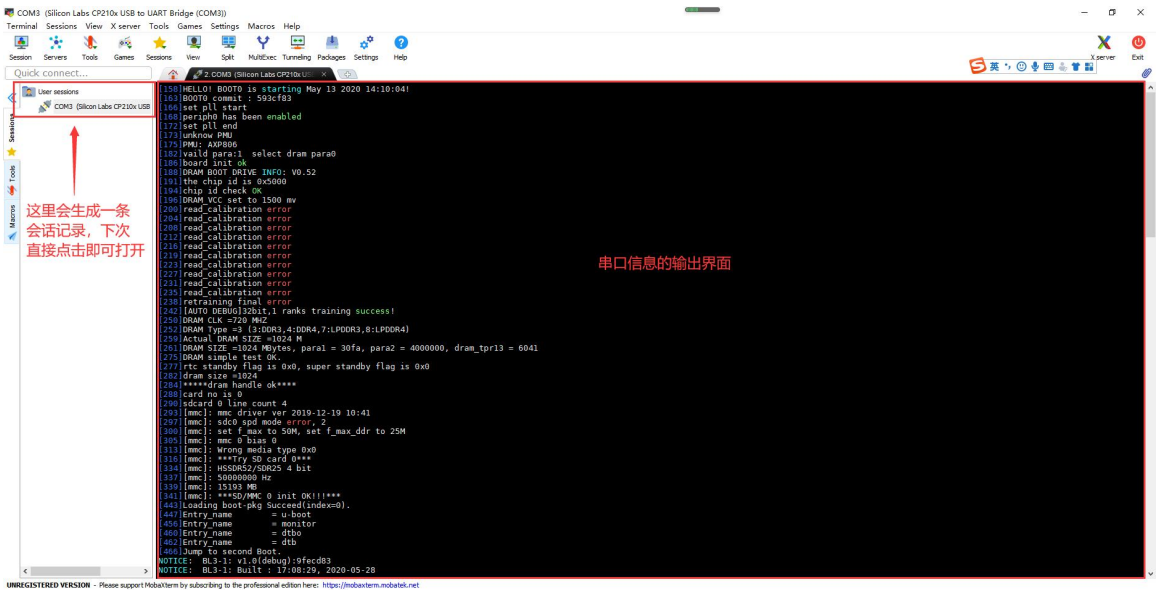
2) 下载完后使用解压缩软件解压下载的压缩包，即可得到 MobaXterm 的可执软件，然后双击打开。

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 **1500000**。
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。



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

本章内容是基于 linux 服务器版本的镜像和 xfce 桌面版本镜像编写的。

如果使用的是 Ubuntu22.04 Gnome 镜像，请先查看 Ubuntu22.04 Gnome Wayland 桌面系统使用说明一章的说明，

Ubuntu22.04 Gnome Wayland 桌面系统使用说明一章中不存在的内容，可以参考此章的说明，但是有些细节是会有差异的，这点请特别注意下。

如果使用的是 OPi OS Arch 镜像，请查看 Orange Pi OS Arch 系统使用说明一章的内容。

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
Debian 11 - Bullseye	Linux5.10	支持	支持
Debian 12 - Bookworm	Linux5.10	支持	支持
Ubuntu 20.04 - Focal	Linux5.10	支持	支持
Ubuntu 22.04 - Jammy	Linux5.10	支持	支持
Debian 12 - Bookworm	Linux6.1	支持	支持
Ubuntu 22.04 - Jammy	Linux6.1	支持	支持

3.2. Linux5.10 系统适配情况

功能	Debian11	Debian12	Ubuntu20.04	Ubuntu22.04
HDMI 显示	OK	OK	OK	OK
HDMI 音频	OK	OK	OK	OK
USB 2.0	OK	OK	OK	OK
USB 3.0	OK	OK	OK	OK
WIFI	OK	OK	OK	OK
蓝牙	OK	OK	OK	OK
调试串口	OK	OK	OK	OK
FAN 风扇	OK	OK	OK	OK
eMMC 启动	OK	OK	OK	OK
GPIO（26pin）	OK	OK	OK	OK

UART（26pin）	OK	OK	OK	OK
SPI（26pin）	OK	OK	OK	OK
I2C（26pin）	OK	OK	OK	OK
PWM（26pin）	OK	OK	OK	OK
Camera1	OK	OK	OK	OK
Camera2	OK	OK	OK	OK
Camera3	OK	OK	OK	OK
LCD 显示	OK	OK	OK	OK
LCD 触摸	OK	OK	OK	OK
板载 MIC	OK	OK	OK	OK
耳机播放	OK	OK	OK	OK
耳机录音	OK	OK	OK	OK
喇叭 x 2	OK	OK	OK	OK
LED 灯	OK	OK	OK	OK
Type-C 转 USB 3.0	OK	OK	OK	OK
Type-C 接口 DP 显示	OK	OK	OK	OK
Type-C 接口 DP 音频	OK	OK	OK	OK
TF 卡启动	OK	OK	OK	OK
NVMe SSD 识别	OK	OK	OK	OK
SATA SSD 识别	OK	OK	OK	OK
红外接收	OK	OK	OK	OK
GPU	OK	OK	OK	OK
NPU	OK	OK	OK	OK
VPU	OK	OK	OK	OK
开关机按键	OK	OK	OK	OK
看门狗测试	OK	OK	OK	OK
Chromium 硬解视频	OK	OK	OK	OK

3.3. Linux6.1 系统适配情况

功能	Debian12	Ubuntu22.04
HDMI 显示	OK	OK
HDMI 音频	OK	OK
USB 2.0	OK	OK

USB 3.0	**OK**	**OK**
WIFI	**OK**	**OK**
蓝牙	**OK**	**OK**
调试串口	**OK**	**OK**
FAN 风扇	**OK**	**OK**
eMMC 启动	**OK**	**OK**
GPIO（26pin）	**OK**	**OK**
UART（26pin）	**OK**	**OK**
SPI（26pin）	**OK**	**OK**
I2C（26pin）	**OK**	**OK**
PWM（26pin）	**OK**	**OK**
Camera1	**OK**	**OK**
Camera2	**OK**	**OK**
Camera3	**OK**	**OK**
LCD 显示	**OK**	**OK**
LCD 触摸	**OK**	**OK**
板载 MIC	**OK**	**OK**
耳机播放	**OK**	**OK**
耳机录音	**OK**	**OK**
喇叭 x 2	**OK**	**OK**
LED 灯	**OK**	**OK**
Type-C 转 USB 3.0	**OK**	**OK**
Type-C 接口 DP 显示	**OK**	**OK**
Type-C 接口 DP 音频	**OK**	**OK**
TF 卡启动	**OK**	**OK**
NVMe SSD 识别	**OK**	**OK**
SATA SSD 识别	**OK**	**OK**
电池	**OK**	**OK**
红外	**OK**	**OK**
GPU	**OK**	**OK**
NPU	**OK**	**OK**
VPU	**OK**	**OK**
开关机按键	**OK**	**OK**
看门狗测试	**OK**	**OK**
Chromium 硬解视频	**OK**	**OK**

3.4. 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
```

3.5. linux 系统登录说明

3.5.1. linux 系统默认登录账号和密码

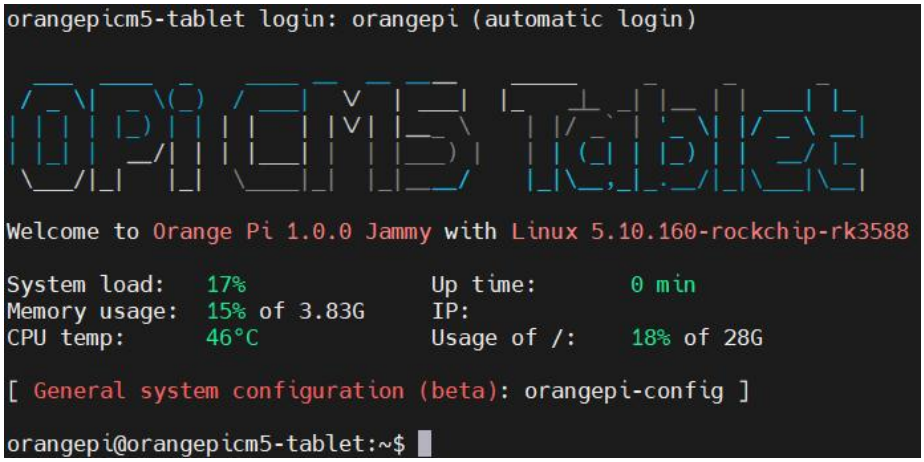
账号	密码
root	**orangepi**
orangepi	**orangepi**

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。

3.5.2. 设置 linux 系统终端自动登录的方法

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



2) 使用下面的命令可以设置 root 用户自动登录终端。

```
orangepi@orangepi:~$ sudo auto_login_cli.sh root
```

3) 使用下面的命令可以禁止自动登录终端。

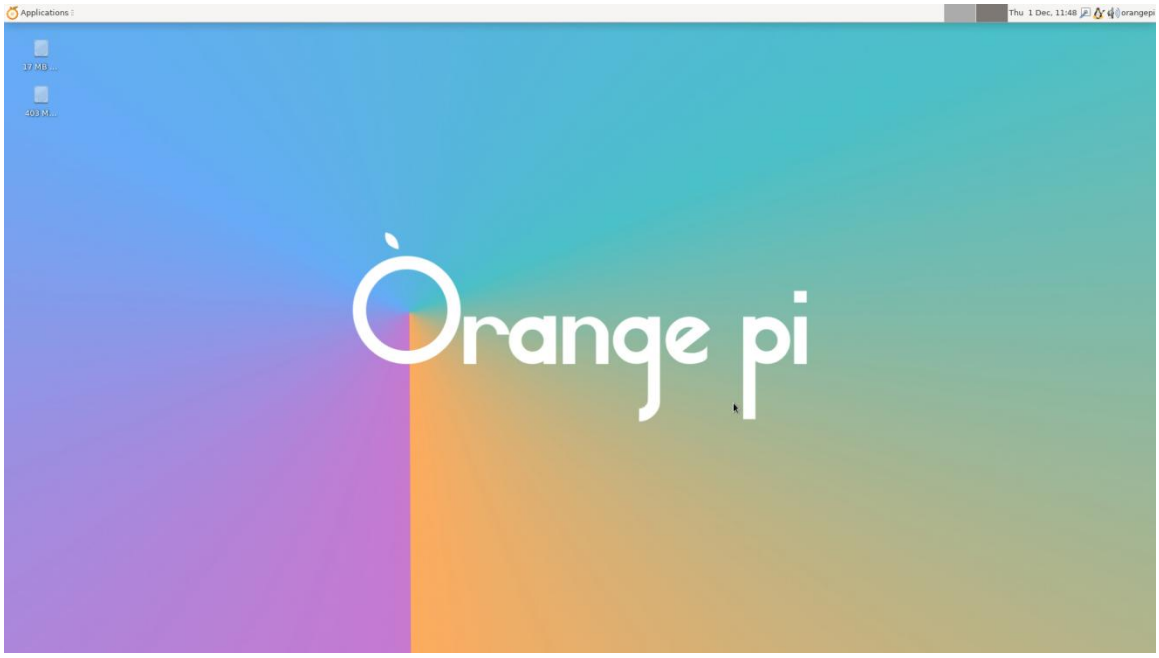
```
orangepi@orangepi:~$ sudo auto_login_cli.sh -d
```

4) 使用下面的命令可以再次设置 orangepi 用户自动登录终端。

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

3.5.3. 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) 运行下面的命令可以禁止桌面版系统自动登录桌面

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

3) 然后重启系统就会出现登录对话框，此时需要输入密码才能进入系统

3.5.4. Linux 桌面版系统 root 用户自动登录的设置方法

1) 执行下面的命令可以设置桌面版系统使用 root 用户自动登录。

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

2) 然后重启系统，就会自动使用 root 用户登录桌面了。

注意，如果使用 root 用户登录桌面系统，是无法使用右上角的 pulseaudio 来管

理音频设备的。

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

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

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

3.5.5. 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 systemctl disable lightdm.service
```

2) 然后重启 Linux 系统就会发现不会显示桌面了。

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

3) 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 systemctl start lightdm.service
orangepi@orangepi:~$ sudo systemctl enable lightdm.service
```

b. 选择完后显示器就会显示桌面了。

3.6. 板载 LED 灯测试说明

1) 底板上有一个红灯和一个绿灯，所在位置如下图所示：

2) 只要开发板打开了电源，红色的 LED 灯就会常亮，这是由硬件控制的，软件无法关闭。通过红色的 LED 灯可以确定开发板的电源是否已正常开启。

3) 绿色的 LED 灯在内核启动后会一直闪烁，这是由软件控制的。

4) 设置绿灯亮灭和闪烁的方法如下所示：

注意，下面的操作请在 root 用户下进行。

a. 首先进入绿灯的设置目录。

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

b. 设置绿灯停止闪烁的命令如下：

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

c. 设置绿灯常亮的命令如下：

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

d. 设置绿灯闪烁的命令如下：

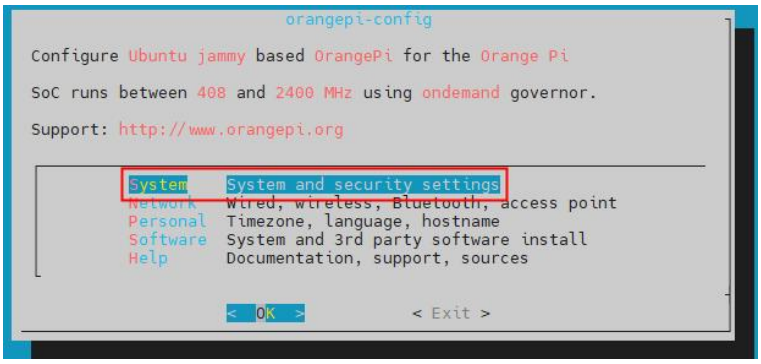
```
root@orangepi:/sys/class/leds/status_led# echo heartbeat > trigger
```

5) 如果开机后不需要绿色 LED 灯闪烁，可以使用下面的方法来关闭绿灯。

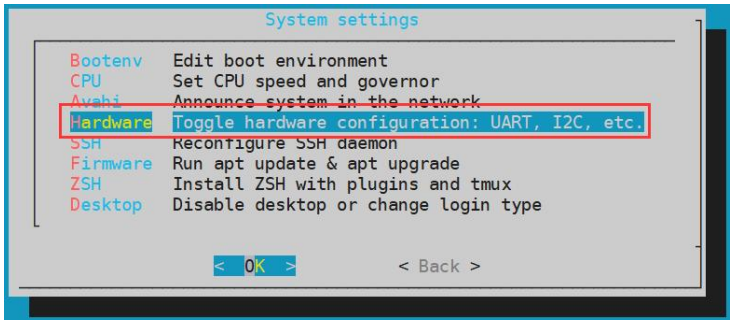
a. 首先运行下 **orangepi-config**，普通用户记得加 **sudo** 权限。

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

b. 然后选择 **System**。

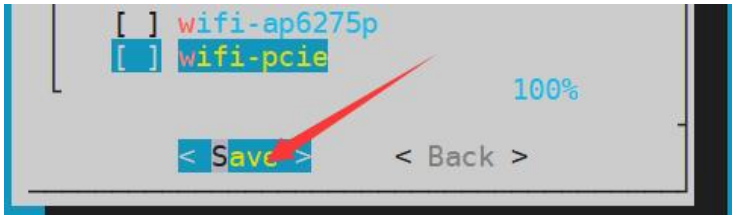


c. 然后选择 **Hardware**。

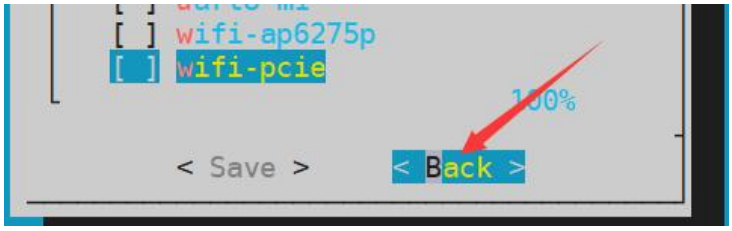


d. 然后使用键盘的方向键定位到下图所示的位置，再使用**空格**选中 **opicm5-tablet-disable-leds** 配置。

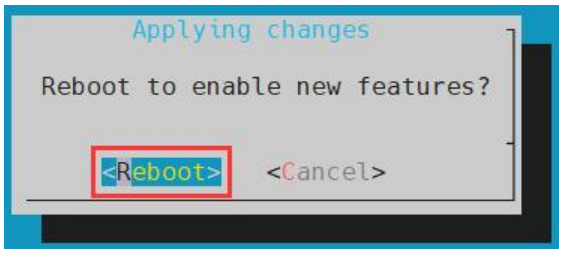
e. 然后选择**<Save>**保存。



f. 然后选择**<Back>**。



g. 然后选择**<Reboot>**重启系统使配置生效。



h. 重启后就可以看到开发板上只有红灯常亮，绿灯不会闪烁了。

3.7. 网络连接测试

3.7.1. 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。

3.7.1.1. 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，有下面两种方式：

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

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

```
orangepi@orangepi:~$ ip addr show wlan0
3: 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
```

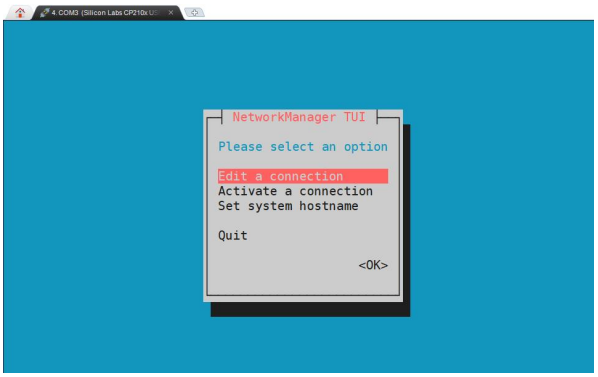
3.7.1.2. 服务器版镜像通过图形化方式连接 WIFI

1) 先登录 linux 系统，有下面两种方式：
   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（串口软件请使用 MobaXterm，使用 minicom 无法显示图形界面）。
   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。

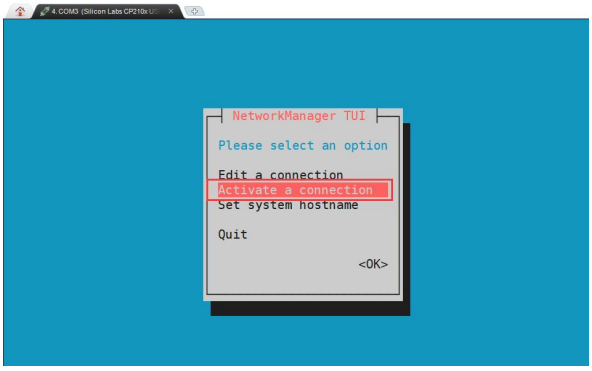
2) 然后在命令行中输入 nmtui 命令打开 wifi 连接的界面。

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

3) 输入 nmtui 命令打开的界面如下所示：



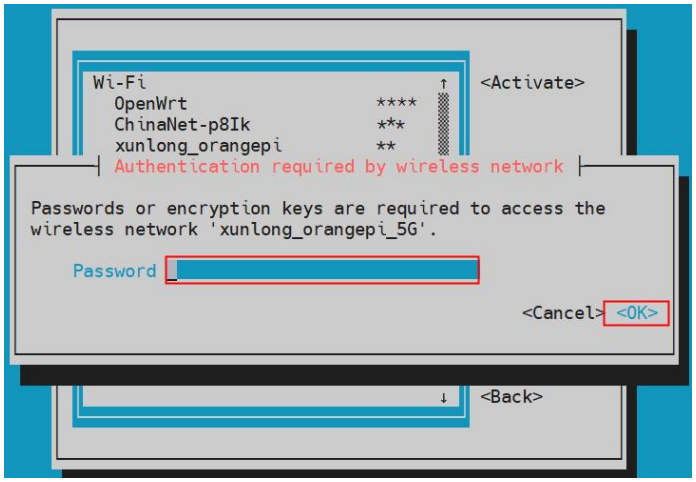
4) 选择 **Activate a connect** 后回车。



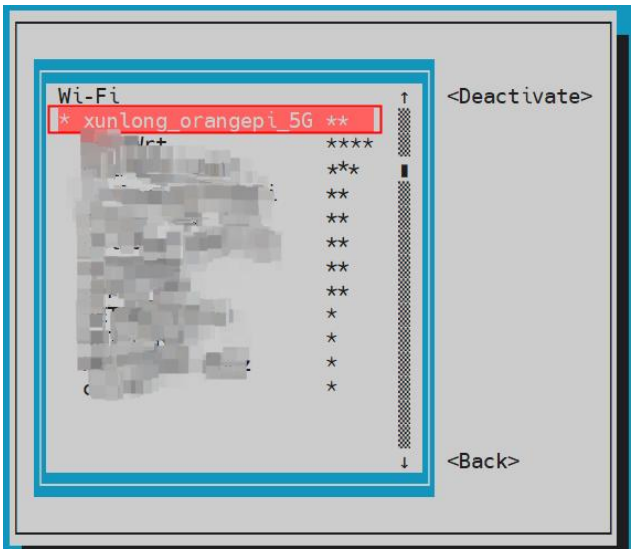
5) 然后就能看到所有搜索到的 WIFI 热点。

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

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



8) WIFI 连接成功后会在已连接的 WIFI 名称前显示一个“*”。



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

```
orangepi@orangepi:~$ ip addr show wlan0
2: 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
```

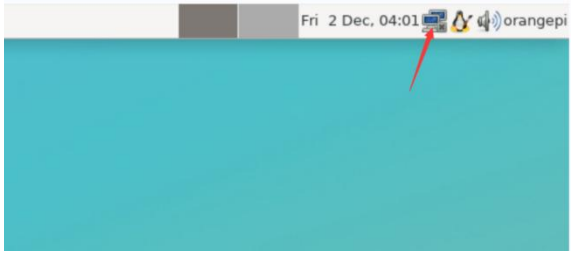
10) 使用 **ping** 命令可以测试 wifi 网络的连通性，**ping** 命令可以通过 **Ctrl+C** 快捷键来中断运行。

```
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
```

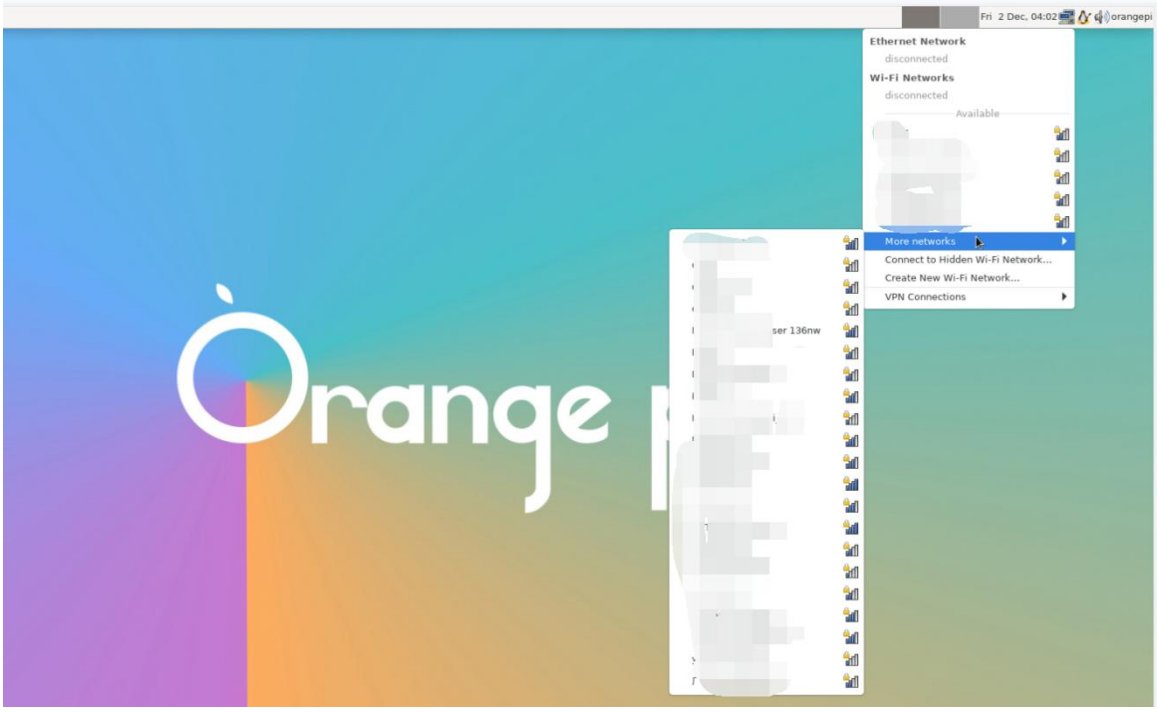
```
^C
--- www.orangepi.org ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 41.321/44.864/48.834/2.484 ms
```

3.7.1.3. 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。

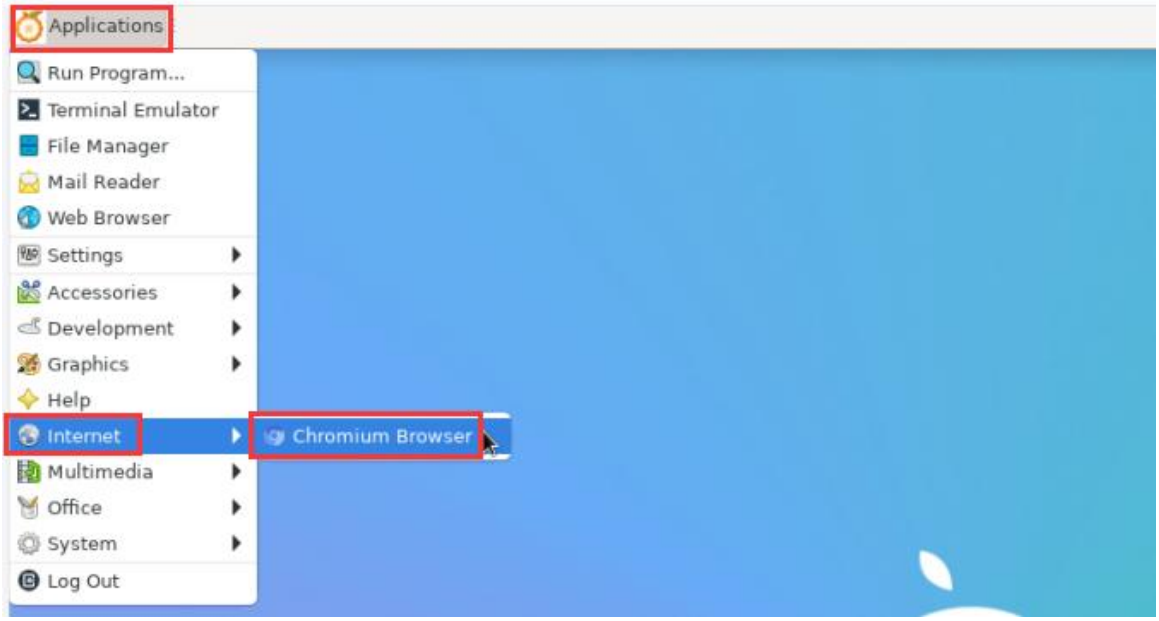


2) 在弹出的下拉框中点击 **More networks** 可以看到所有扫描到的 WIFI 热点，然后选择想要连接的 WIFI 热点。



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

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



5) 打开浏览器后如果能打开其他网页说明 WIFI 连接正常。

3.8. 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 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。

3.8.1. Ubuntu 下 SSH 远程登录开发板

1) 获取开发板的 IP 地址。

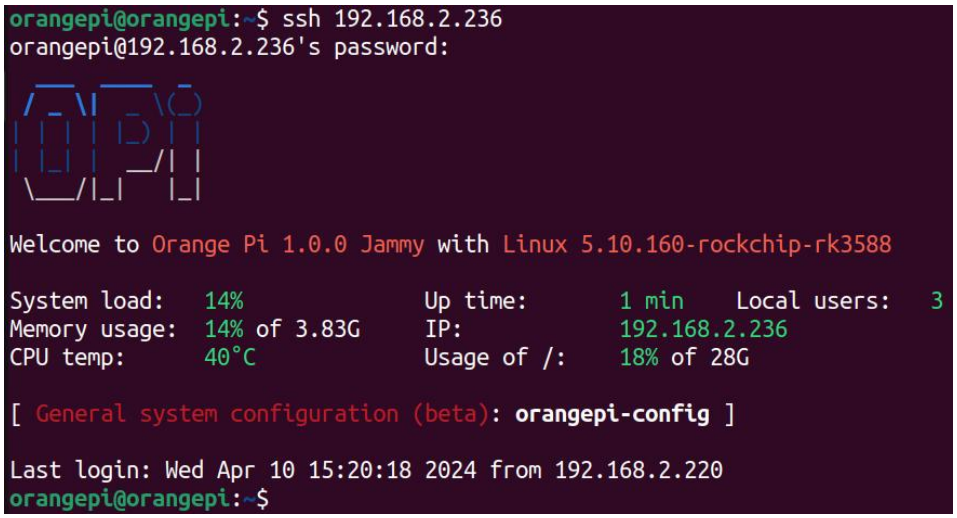
2) 然后就可以通过 ssh 命令远程登录 linux 系统。

```
test@test:~$ ssh root@192.168.x.xxx          (需要替换为开发板的 IP 地址)
root@192.168.x.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，首先请检查下开发板的 IP 地址是否能 ping 通，如果 ping 通没问题的话，可以通过串口或者 HDMI 显示器登录 linux 系统然后在开发板上输入下面的命令后再尝试是否能连接：

root@orangepi:~# **reset_ssh.sh**

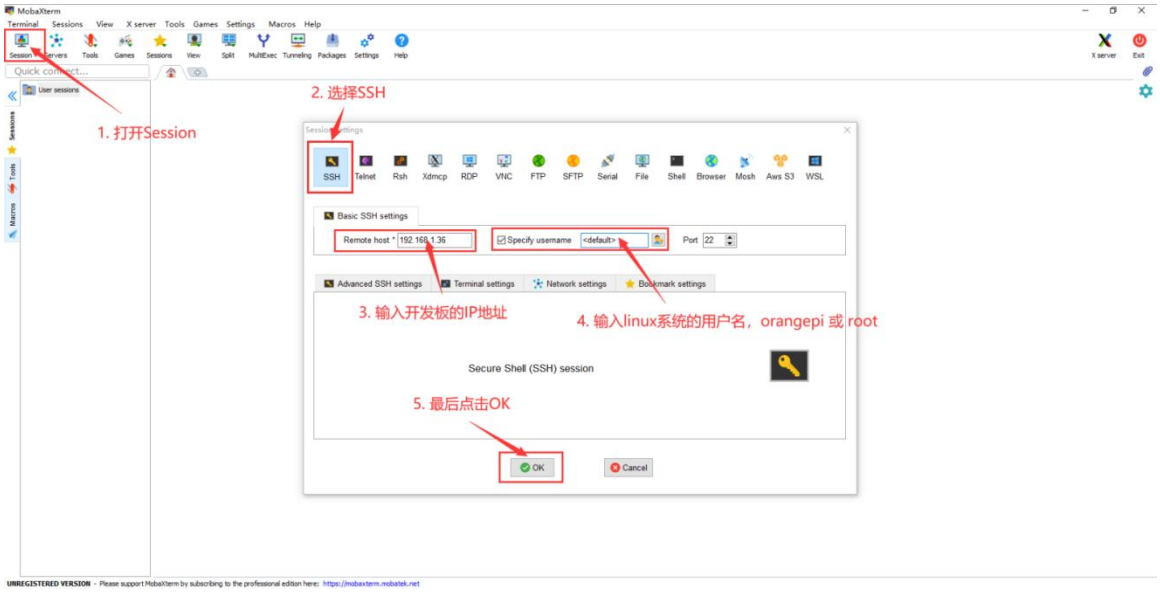
如果还不行，请重烧系统试下。

3.8.2. Windows 下 SSH 远程登录开发板

1) 首先获取开发板的 IP 地址。

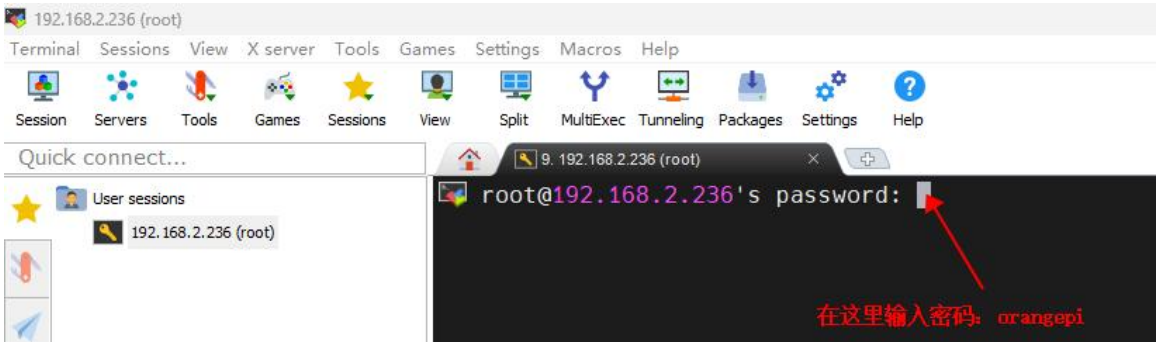
2) 在 windows 下可以使用 MobaXterm 远程登录开发板，首先新建一个 ssh 会话。

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。

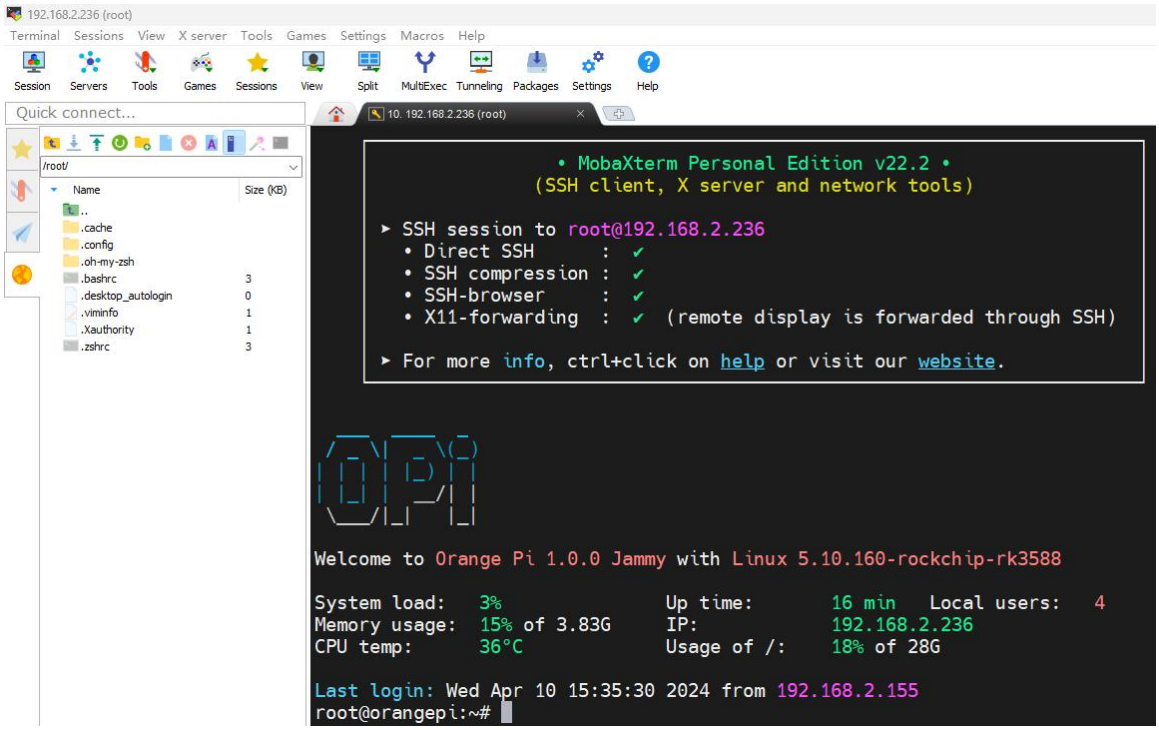


3) 然后会提示输入密码，默认 root 和 orangepi 用户的密码都为 orangepi。

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：



3.9. ADB 的使用方法

3.9.1. 网络 adb 的使用方法

1) 系统启动后请先确认下 **adbd** 已经启动了。

```
orangepi@orangepi:~$ ps -ax | grep "adbd"
    808 ?        Sl     0:00 /usr/bin/adbd
   3707 ttyFIQ0  S+     0:00 grep --color=auto adbd
```

2) 然后查看下开发板的 IP 地址，并记下来。

3) 然后在 Ubuntu PC 上安装 adb 工具。

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

4) 然后使用下面的命令连接网络 adb。

```
test@test:~$ adb connect 192.168.1.xx:5555        #IP 地址请替换为开发板的 IP 地址
* daemon not running; starting now at tcp:5037
* daemon started successfully
connected to 192.168.1.xx:5555
test@test:~$ adb devices
List of devices attached
192.168.1.xx:5555          device
```

5) 然后使用下面的命令就可以登录开发板的 linux 系统。

```
test@test:~$ adb shell
root@orangepi:/#     <--- 看到这个提示符后说明已成功登录开发板
```

6) 使用 adb 上传文件到开发板的命令如下所示：

```
test@test:~$ adb push filename /root
filename: 1 file pushed. 3.7 MB/s (1075091 bytes in 0.277s)
```

7) 使用 adb 重启开发板的命令如下所示：

```
test@test:~$ adb reboot
```

如果您的 Windows 系统中没有 adb 工具，可以使用 RKDevTool 软件中的 adb 程序。

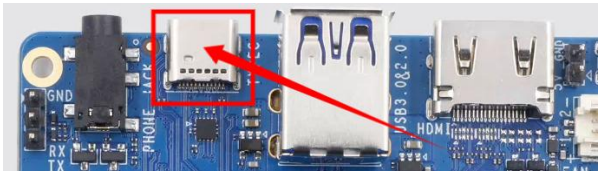
在 Windows 中使用 adb 的示例如下所示：

```
命令提示符
Microsoft Windows [版本 10.0.19044.2251]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Administrator>cd C:\Users\Administrator\Desktop\RKDevTool_Release_v2.92\bin

C:\Users\Administrator\Desktop\RKDevTool_Release_v2.92\bin>dir
 驱动器 C 中的卷没有标签。
 卷的序列号是 62AE-5AED

 C:\Users\Administrator\Desktop\RKDevTool_Release_v2.92\bin 的目录

2022/08/09  13:19    <DIR>          .
2022/08/09  13:19    <DIR>          ..
2019/06/24  09:13         1,850,368 adb.exe
2019/06/24  09:13            97,792 AdbWinApi.dll
2019/06/24  09:13            62,976 AdbWinUsbApi.dll
2021/08/23  09:04           894,976 AFPTool.exe
2021/08/16  14:05           890,368 RKImageMaker.exe
               5 个文件      3,796,480 字节
               2 个目录 64,033,034,240 可用字节

C:\Users\Administrator\Desktop\RKDevTool_Release_v2.92\bin>.\adb.exe connect 192.168.1.144
connected to 192.168.1.144:5555

C:\Users\Administrator\Desktop\RKDevTool_Release_v2.92\bin>.\adb.exe devices
List of devices attached
192.168.1.144:5555      device


C:\Users\Administrator\Desktop\RKDevTool_Release_v2.92\bin>.\adb.exe push adb.exe /root
adb.exe: 1 file pushed.  4.1 MB/s (1850368 bytes in 0.427s)

C:\Users\Administrator\Desktop\RKDevTool_Release_v2.92\bin>
```

3.9.2. 使用 Type-C 数据线连接 adb

1) 首先准备一根品质良好的 Type-C 数据线。

2) 然后通过 Type-C 数据线连接好开发板与 Ubuntu PC，开发板 Type-C 接口的位置如下图所示：



3) 然后运行下面的命令将 Type-C 接口设置为 **device** 模式。

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

如果 linux 系统中不存在 **set_device.sh** 脚本，请直接使用下面的命令：

```
orangepi@orangepi:~$ sudo bash -c "echo device > /sys/kernel/debug/usb/fc000000.usb/mode"
orangepi@orangepi:~$ sudo systemctl restart usbdevice
```

4) 然后请确认下 adbd 已经启动了。

```
orangepi@orangepi:~$ ps -ax | grep "adbd"
    808 ?        Sl     0:00 /usr/bin/adbd
   3707 ttyFIQ0  S+     0:00 grep --color=auto adbd
```

5) 然后在 Ubuntu PC 上安装下 adb 工具。

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

6) 然后使用下面的命令查看下有没有识别到 adb 设备。

```
test@test:~$ adb devices
List of devices attached
e0f9f71bc343c305    device
```

8) 然后使用下面的命令就可以登录开发板的 linux 系统。

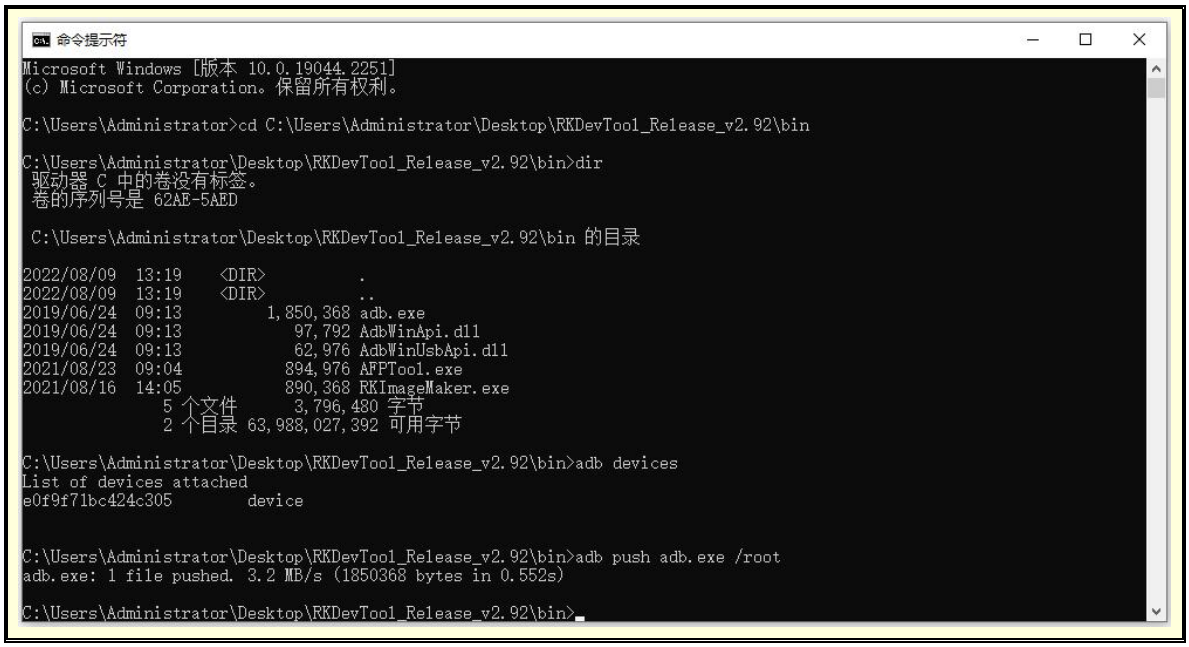
```
test@test:~$ adb shell
root@orangepi:/#     <---  看到这个提示符后说明已成功登录开发板
```

9) 使用 adb 上传文件到开发板的命令如下所示：

```
test@test:~$ adb push filename /root
filename: 1 file pushed. 3.7 MB/s (1075091 bytes in 0.277s)
```

如果您的 Windows 系统中没有 adb 工具，可以使用 RKDevTool 软件中的 adb 程序。

在 Windows 中使用 adb 的示例如下所示：



3.10. 上传文件到开发板 Linux 系统中的方法

3.10.1. 在 Ubuntu PC 中上传文件到开发板 Linux 系统中的方法

3.10.1.1. 使用 scp 命令上传文件的方法

1) 使用 scp 命令可以在 Ubuntu PC 中上传文件到开发板的 Linux 系统中，具体命令如下所示：

a. **file_path**：需要替换为要上传文件的路径。
b. **orangepi**：为开发板 linux 系统的用户名，也可以替换成其它的，比如 root
c. **192.168.xx.xx**：为开发板的 IP 地址，请根据实际情况进行修改。
d. **/home/orangepi**：开发板 linux 系统中的路径，也可以修改为其它的路径。

```
test@test:~$ scp file_path orangepi@192.168.xx.xx:/home/orangepi/
```

2) 如果要上传文件夹，需要加上-r 参数。

```
test@test:~$ scp -r dir_path orangepi@192.168.xx.xx:/home/orangepi/
```

3) scp 还有更多的用法，请使用下面的命令查看 man 手册。

```
test@test:~$ man scp
```

3.10.1.2. 使用 filezilla 上传文件的方法

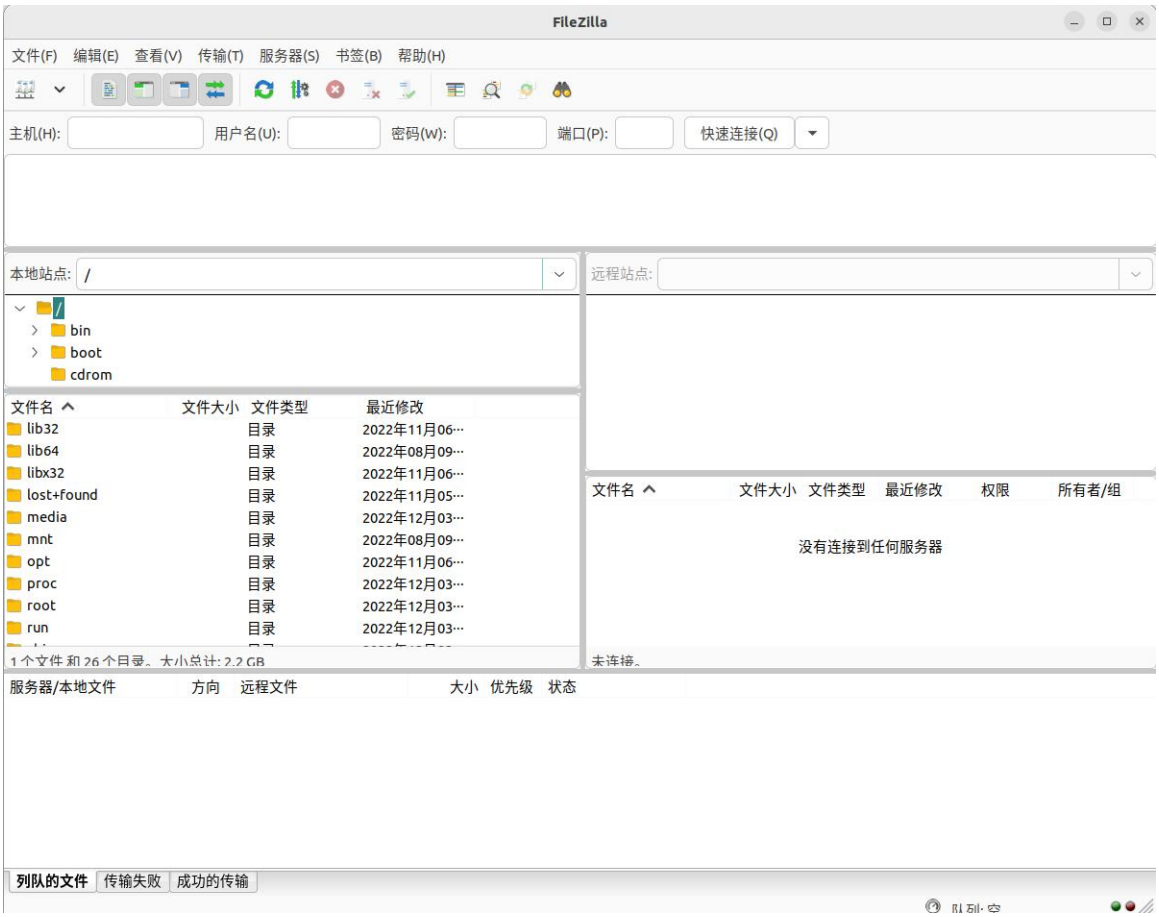
1) 首先在 Ubuntu PC 中安装 filezilla。

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

2) 然后使用下面的命令打开 filezilla。

```
test@test:~$ filezilla
```

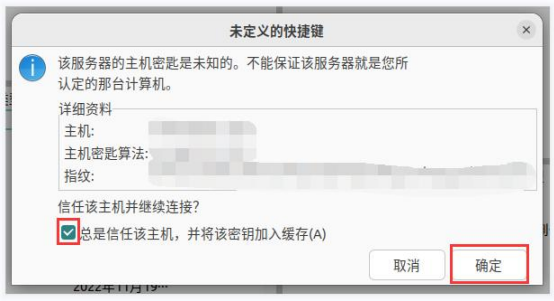
3) filezilla 打开后的界面如下所示，此时右边远程站点下面显示的是空的。



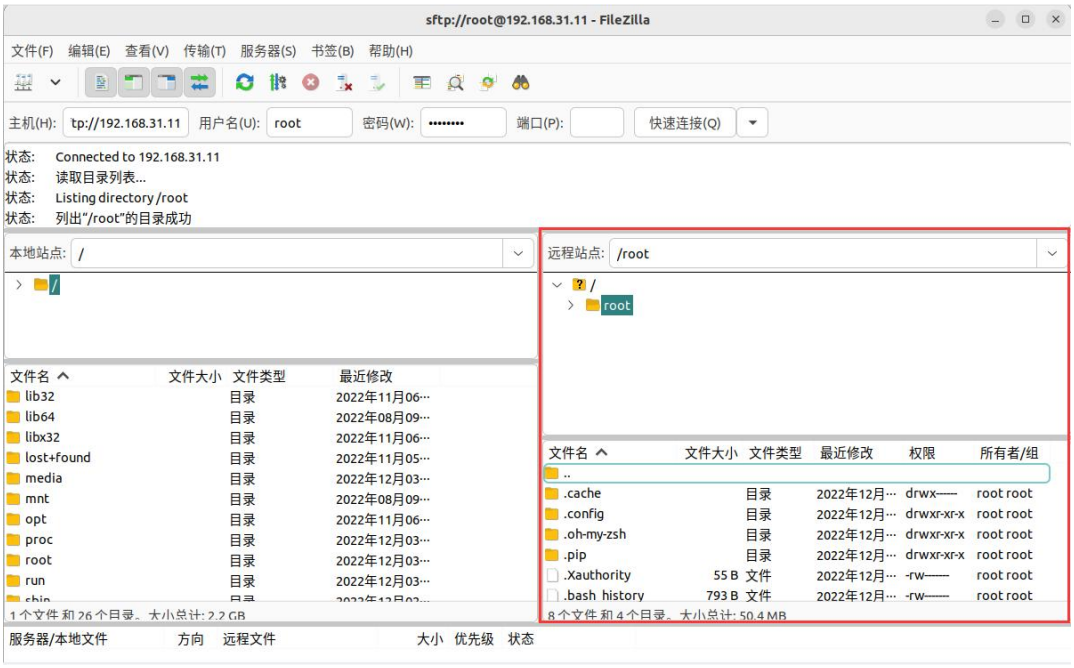
4) 连接开发板的方法如下图所示：

5) 然后选择**保存密码**，再点击**确定**。

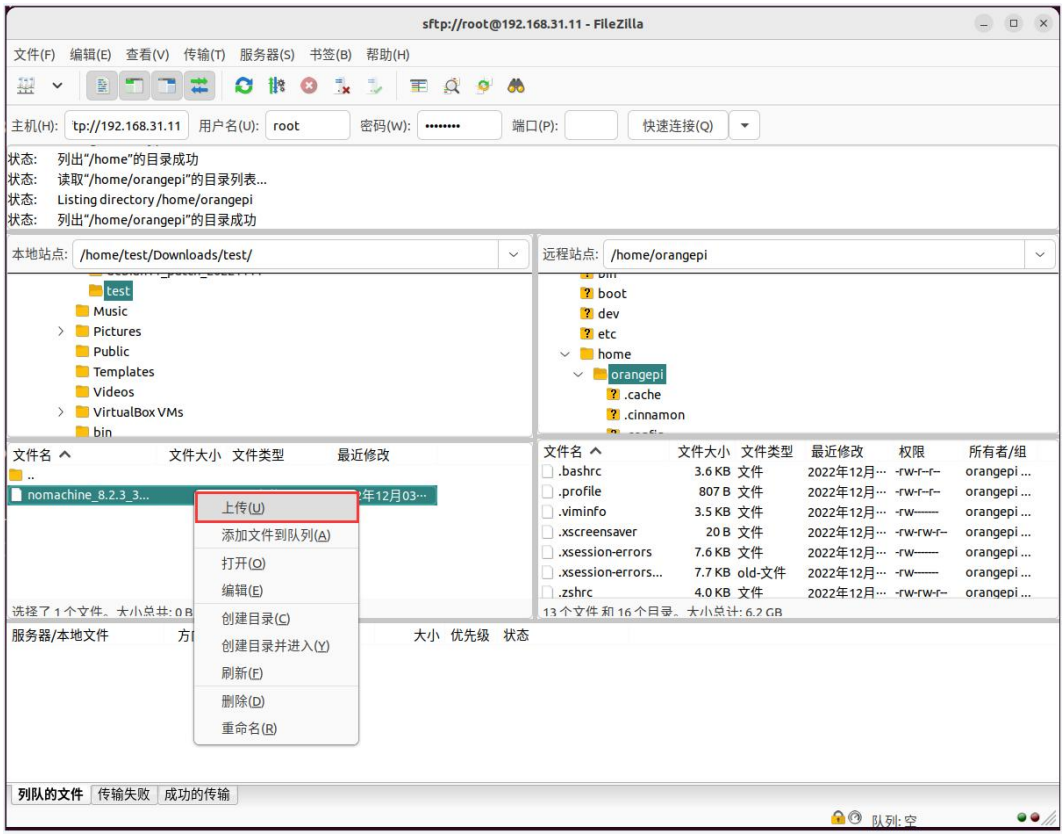
6) 然后选择**总是信任该主机**，再点击**确定**。



7) 连接成功后在 filezilla 软件的右边就可以看到开发板 linux 文件系统的目录结构了。



8) 然后在 filezilla 软件的右边选择要上传到开发板中的路径，再在 filezilla 软件的左边选中 Ubuntu PC 中要上传的文件，再点击鼠标右键，再点击上传选项就会开始上传文件到开发板中了。



9) 上传完成后就可以去开发板 linux 系统中的对应路径中查看上传的文件了。

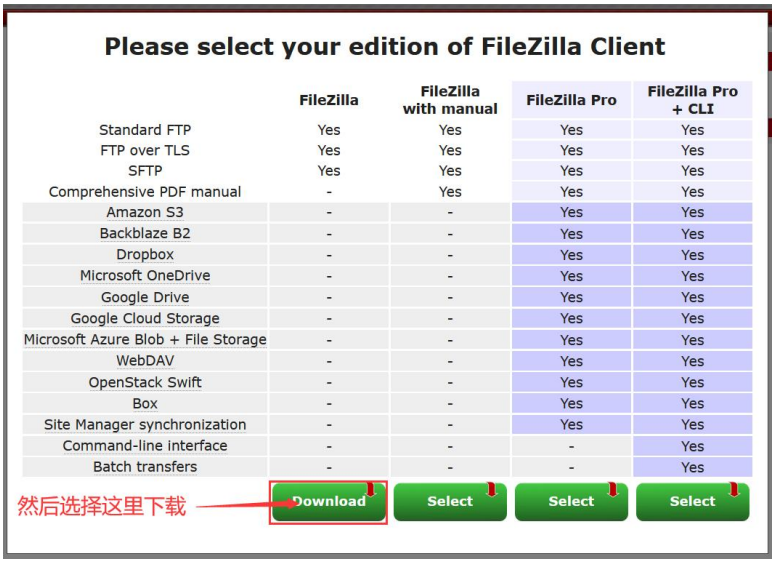
10) 上传文件夹的方法和上传文件的方法是一样的，这里就不再赘述了。

3.10.2. 在 Windows PC 中上传文件到开发板 Linux 系统中的方法

3.10.2.1. 使用 filezilla 上传文件的方法

1) 首先下载 filezilla 软件 Windows 版本的安装文件，下载链接如下所示：

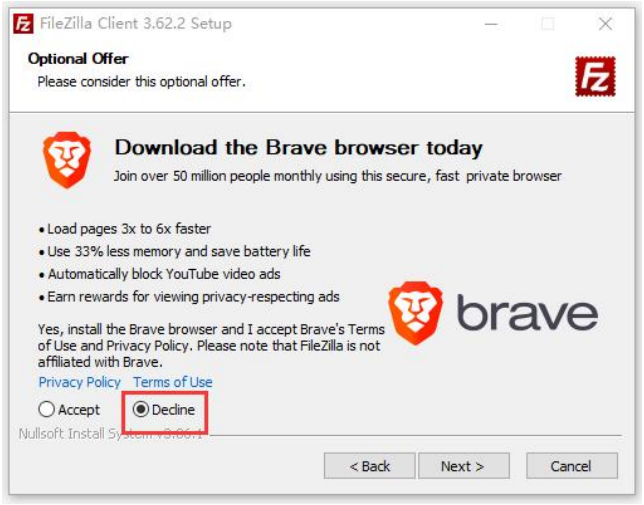
https://filezilla-project.org/download.php?type=client



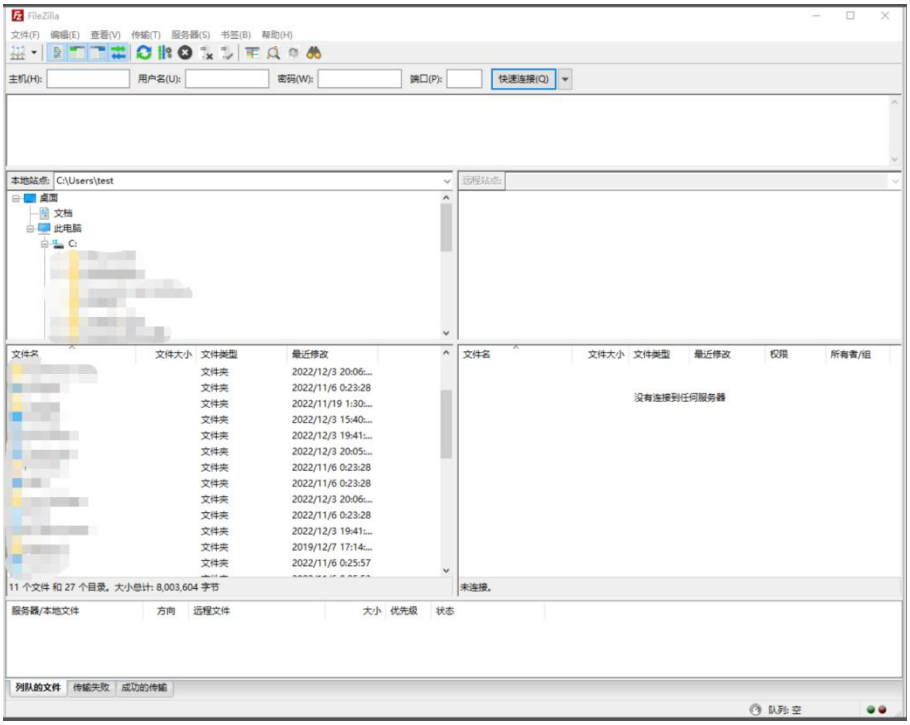
2) 下载的安装包如下所示，然后双击直接安装即可。

FileZilla_Server_1.5.1_win64-setup.exe

安装过程中，下面的安装界面请选择 `Decline`，然后再选择 `Next>`。

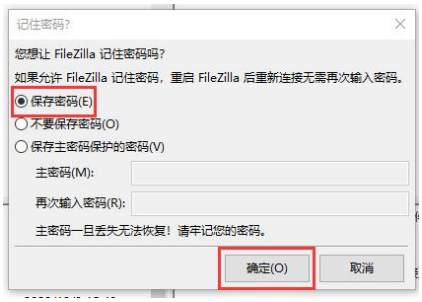


3) filezilla 打开后的界面如下所示，此时右边远程站点下面显示的是空的。

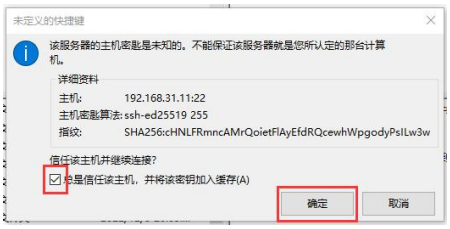


4) 连接开发板的方法如下图所示：

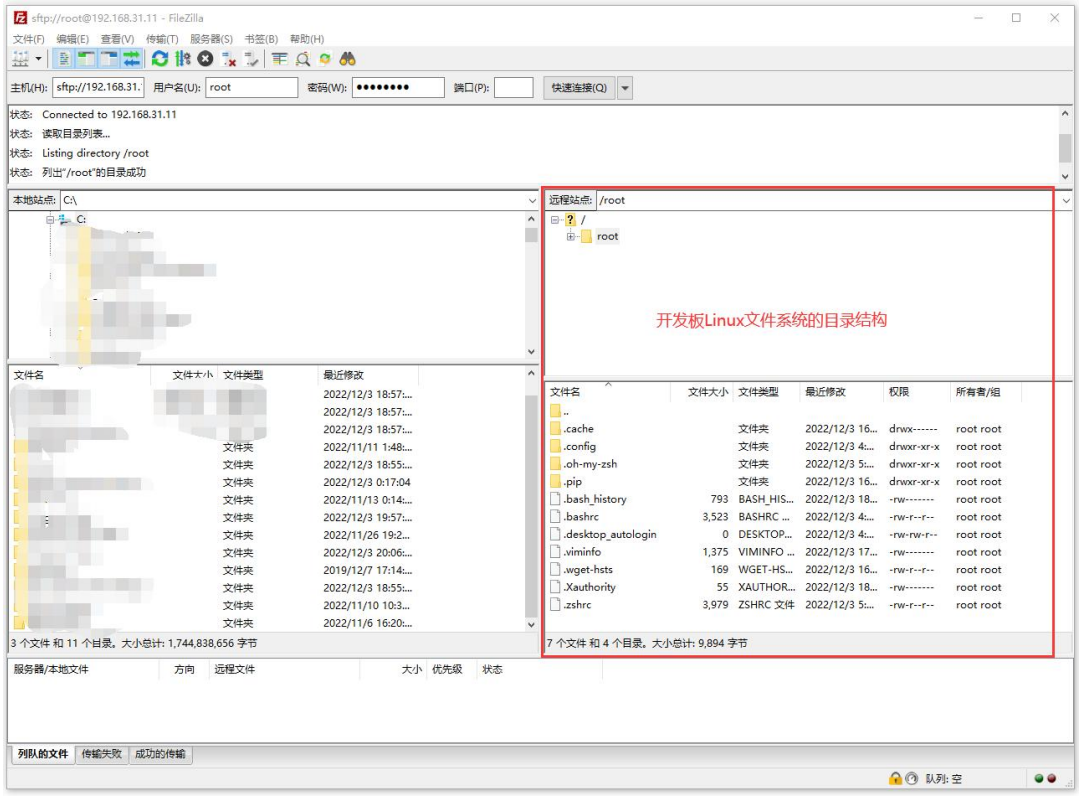
5) 然后选择**保存密码**，再点击**确定**。



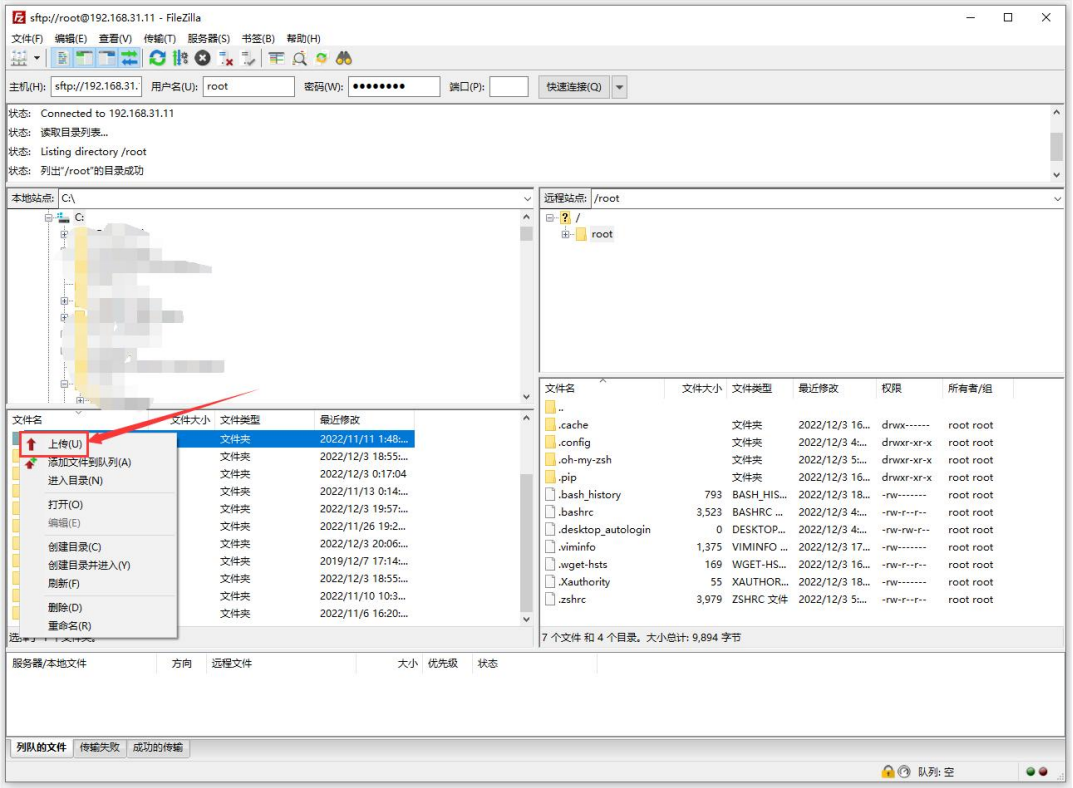
6) 然后选择**总是信任该主机**，再点击**确定**。



7) 连接成功后在 filezilla 软件的右边就可以看到开发板 linux 文件系统的目录结构了。



8) 然后在 filezilla 软件的右边选择要上传到开发板中的路径，再在 filezilla 软件的左边选中 Windows PC 中要上传的文件，再点击鼠标右键，再点击上传选项就会开始上传文件到开发板中了。



9) 上传完成后就可以去开发板 linux 系统中的对应路径中查看上传的文件了。

10) 上传文件夹的方法和上传文件的方法是一样的，这里就不再赘述了。

3.11. HDMI 测试

3.11.1. HDMI 显示测试

2) 使用 HDMI 转 HDMI 线连接 Orange Pi 开发板和 HDMI 显示器。

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

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

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

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

3.11.2. HDMI 转 VGA 显示测试

1) 首先需要准备下面的配件。

a. HDMI 转 VGA 转换器。

b. 一根 VGA 线。

c. 一个支持 VGA 接口的显示器或者电视。

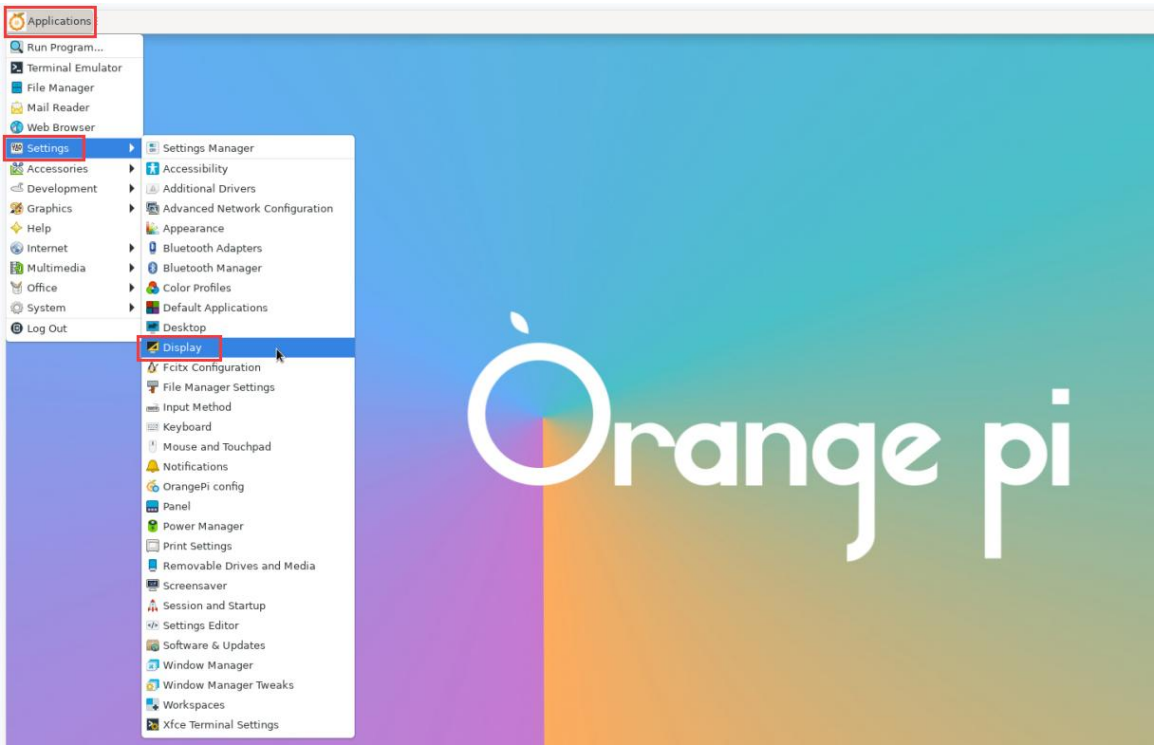
2) VGA 线的一端需要连接到显示器或者电视上，VGA 线的另一端需要连接到

HDMI 转 VGA 转换器的 VGA 接口上，最后 HDMI 转 VGA 转换器的 HDMI 接口需要插入到开发板的 HDMI 接口中。如果一切正常，开发板启动后就能看到显示器会有显示了。

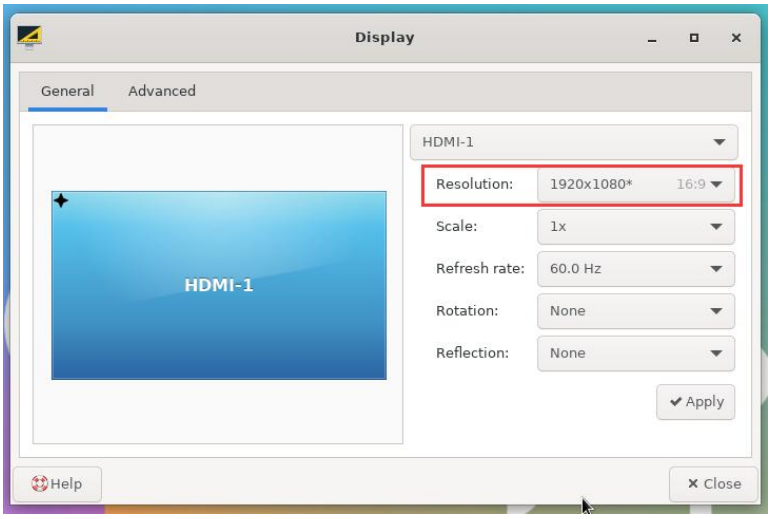
使用 HDMI 转 VGA 显示时，开发板以及开发板的 Linux 系统是不需要做任何设置的，只需要开发板 HDMI 接口能正常显示就可以了。所以如果测试有问题，请检查 HDMI 转 VGA 转换器、VGA 线以及显示器是否有问题。

3.11.3. HDMI 分辨率设置方法

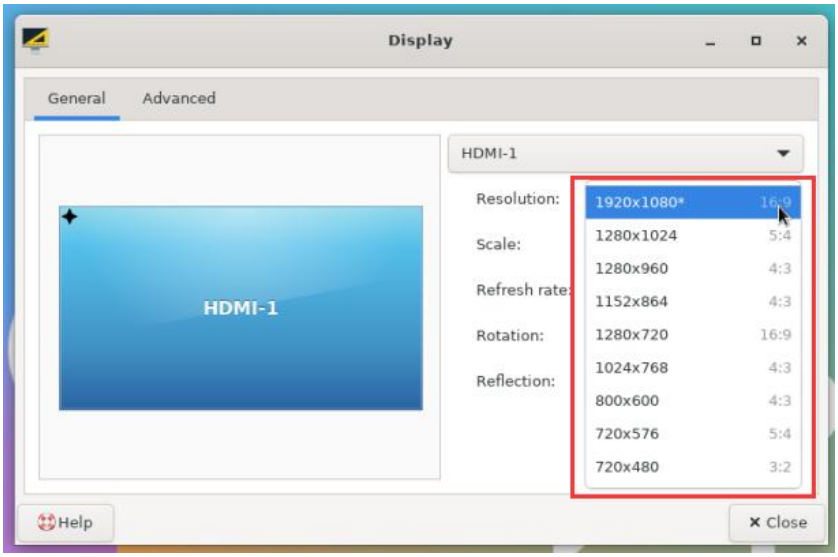
1) 首先在 **Settings** 中打开 **Display**。



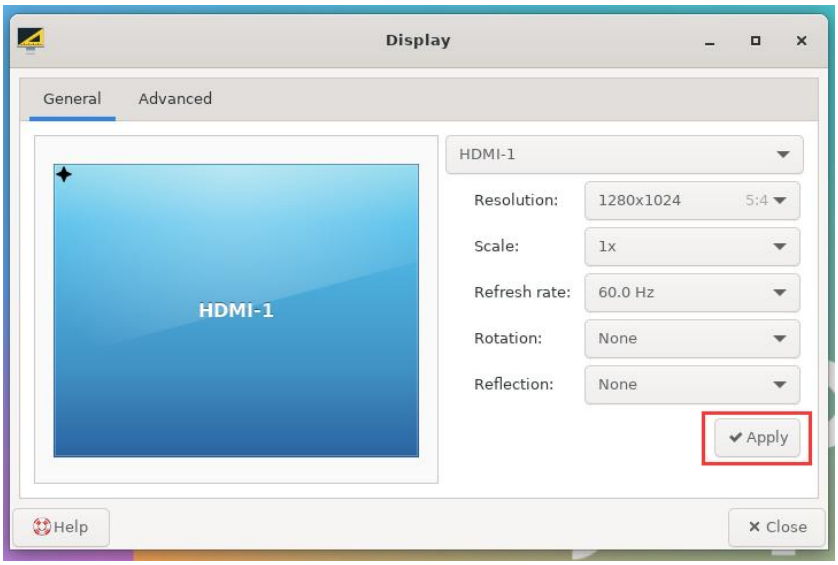
2) 然后就能看到系统当前的分辨率。



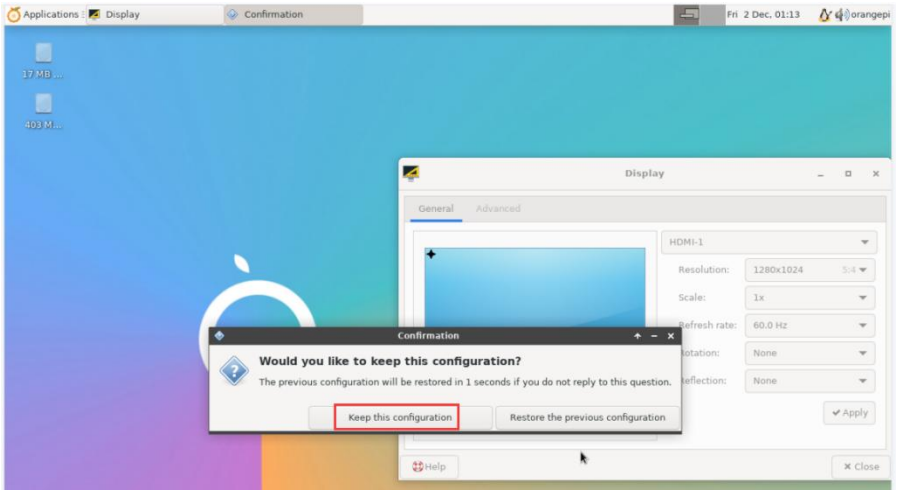
3) 点击 Resolution（分辨率）的下拉框，就可以看到显示器当前支持的所有分辨率。



4) 然后选择想要设置的分辨率，再点击 Apply。



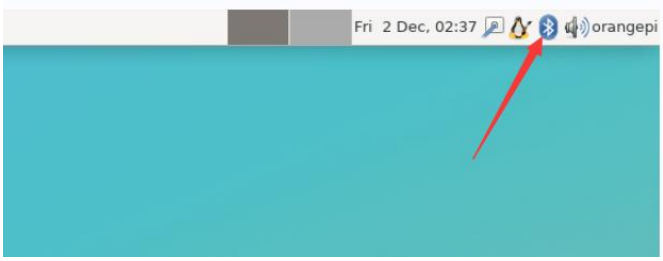
5) 等新的分辨率设置完后再选择 **Keep the configuration** 即可。



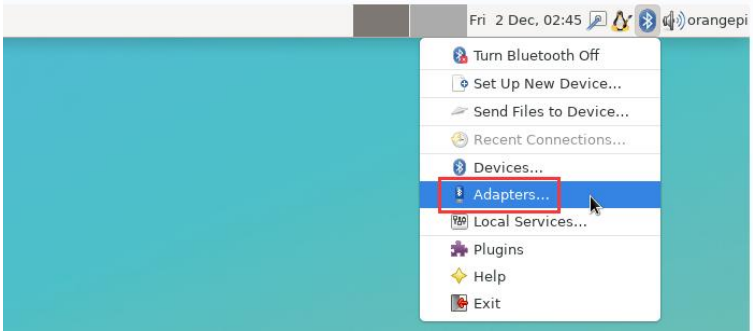
3.12. 蓝牙使用方法

3.12.1. 桌面版镜像的测试方法

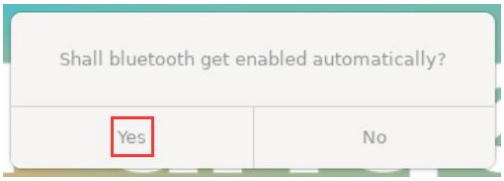
1) 点击桌面右上角的蓝牙图标。



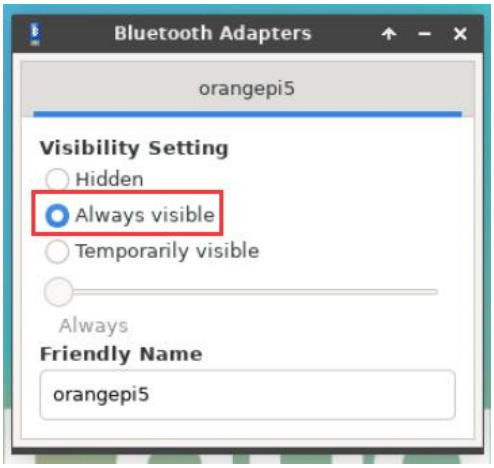
2) 然后选择适配器。



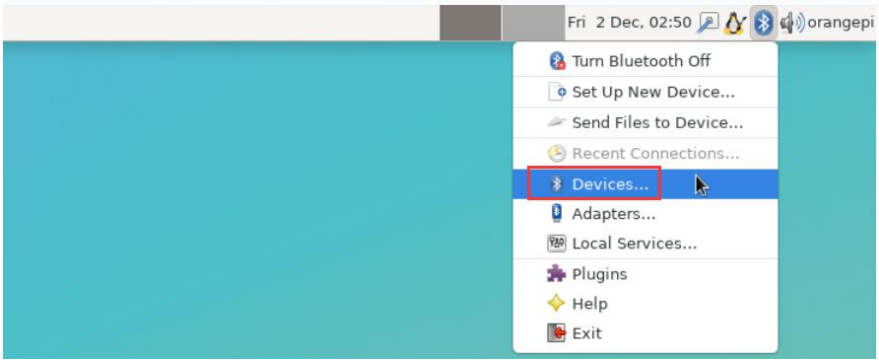
3) 如果有提示下面的界面，请选择 **Yes**。



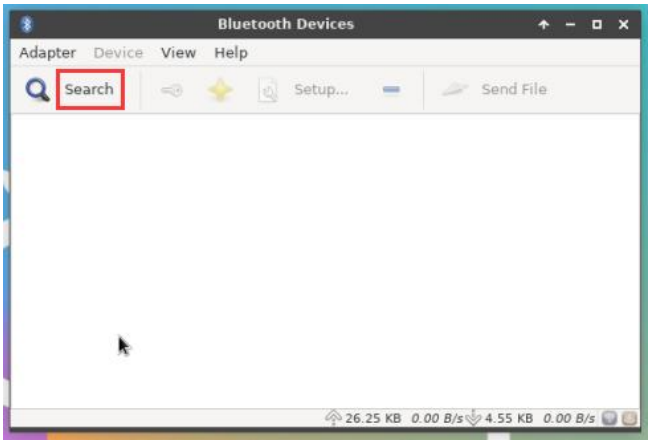
4) 然后在蓝牙的适配器设置界面中设置 **Visibility Setting** 为 **Always visible**，然后关闭即可。



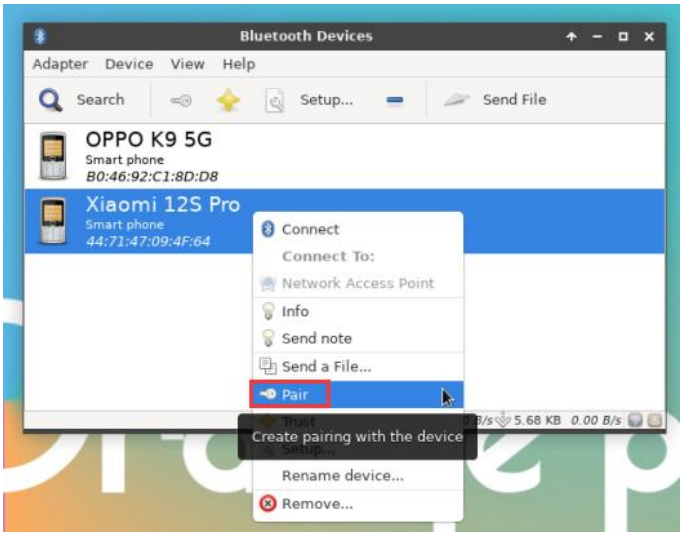
5) 然后打开蓝牙设备的配置界面。



6) 点击 **Search** 即可开始扫描周围的蓝牙设备。



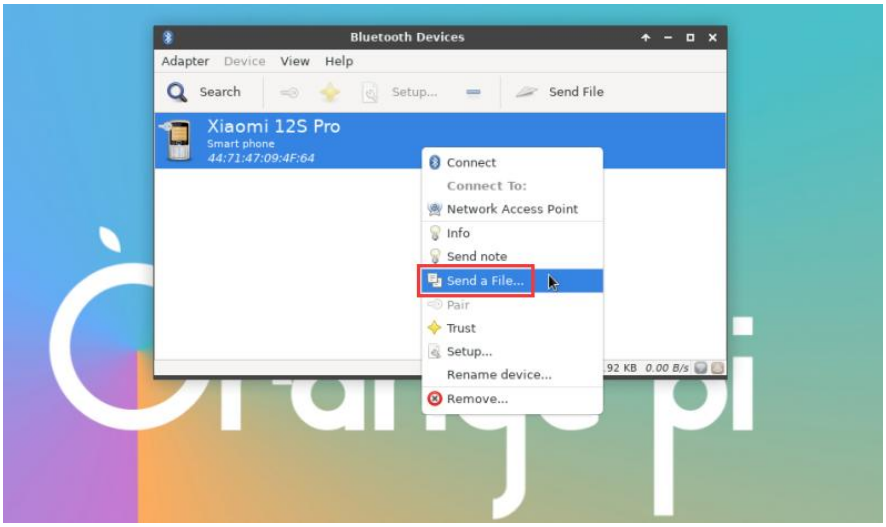
6) 然后选择想要连接的蓝牙设备，再点击鼠标右键就会弹出对此蓝牙设备的操作界面，选择 **Pair** 即可开始配对，这里演示的是和 Android 手机配对。



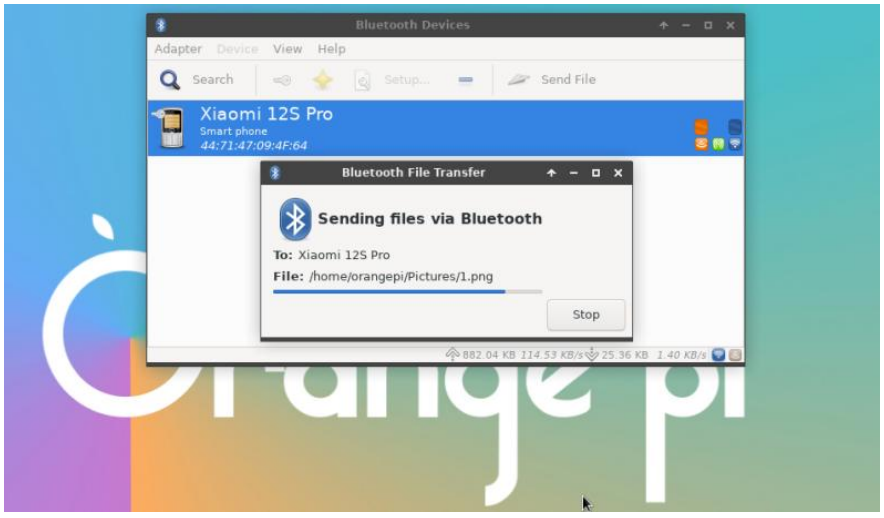
7) 配对时，桌面的右上角会弹出配对确认框，选择 **Confirm** 确认即可，此时手机上

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：



3.13. 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
```

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 摄像头测试

1) 首先需要准备一个下图所示的或者类似的支持 UVC 协议的 USB 摄像头，然后将 USB 摄像头插入到 Orange Pi 开发板的 USB 接口中。

2) 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:~$ v4l2-ctl --list-devices
Q8 HD Webcam: Q8 HD Webcam (usb-fc880000.usb-1):
        /dev/video0
        /dev/video1
        /dev/media0
```

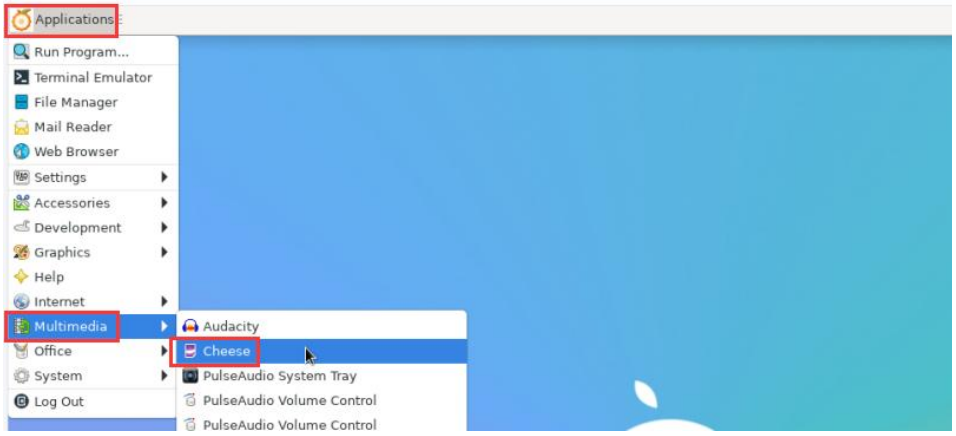
注意 v4l2 中的 l 是小写字母 l，不是数字 1。

另外 video 的序号不一定都是 video0，请以实际看到的为准。

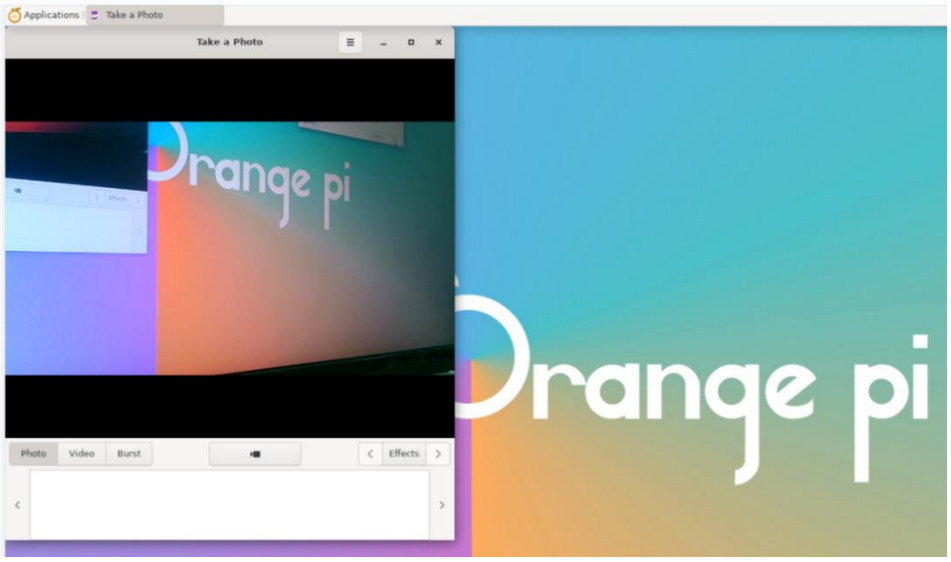
3) 在桌面系统中可以使用 Cheese 直接打开 USB 摄像头，Cheese 打开方法如下图所示：

如果系统没有预装 cheese，请使用下面的命令安装下：

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



Cheese 打开 USB 摄像头后的界面如下图所示：



4) 使用 fswebcam 测试 USB 摄像头的方法。

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。

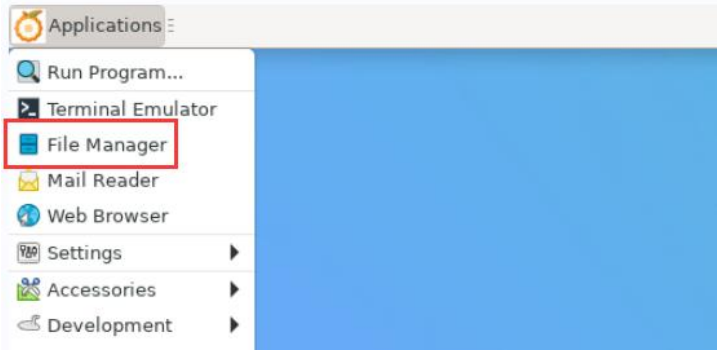
```
orangepi@orangepi:~$ scp image.jpg test@192.168.1.55:/home/test（根据实际情况修改 IP 地址和路径）
```

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。

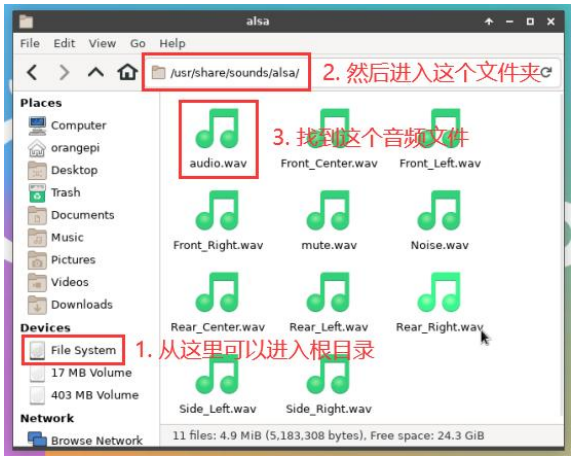
3.14. 音频测试

3.14.1. 在桌面系统中测试音频方法

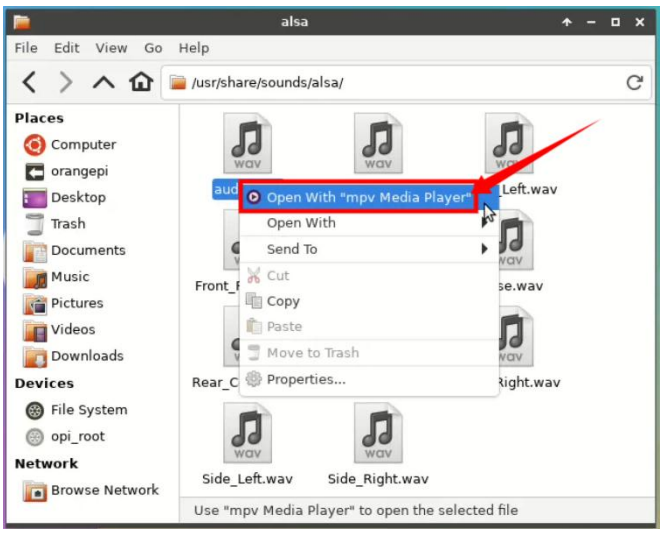
1) 首先打开文件管理器。



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

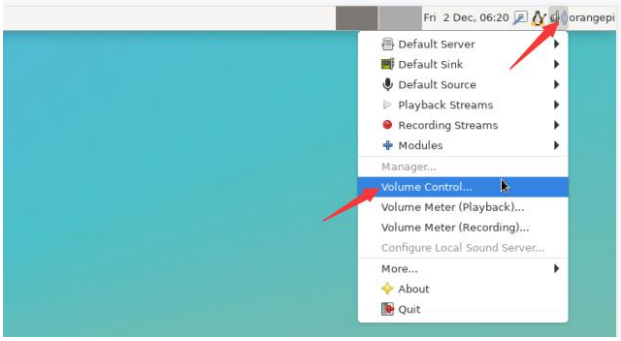


3) 然后选中 audio.wav 文件，右键选择使用 vlc 或者 mpv 打开就可以开始播放。

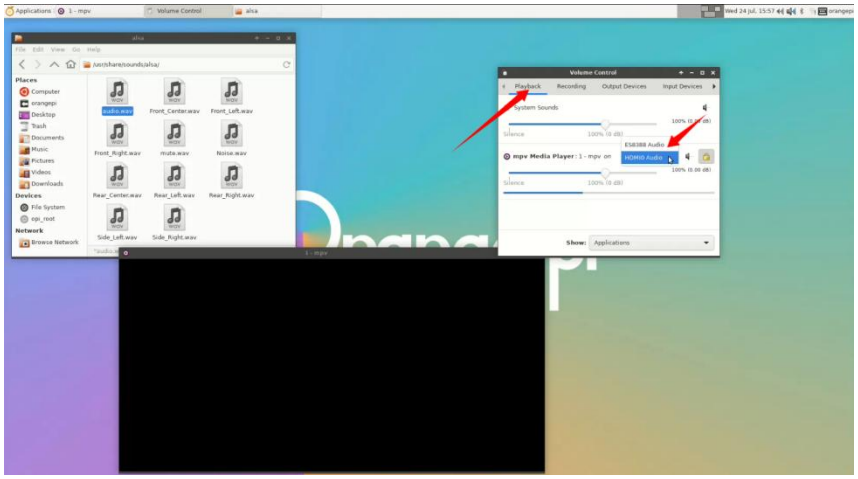


4) 切换 HDMI 播放和耳机播放等不同音频设备的方法。

a. 首先打开音量控制界面。



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

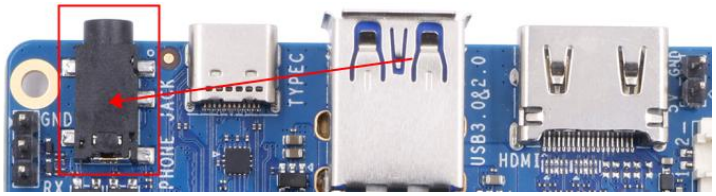


如果看不到 HDMI 的音频设备选项，可以尝试拔插下 HDMI 线。

3.14.2. 使用命令播放音频的方法

3.14.2.1. 耳机接口播放音频测试

1) 首先将耳机插入开发板的耳机孔中。



2) 然后可以通过 **aplay -l** 命令可以查看下 linux 系统支持的声卡设备，从下面的输出可知，**card 2** 为 es8388 的声卡设备，也就是耳机的声卡设备。

```
orangepi@orangepi:~$ aplay -l
card 0: rockchipdp0 [rockchip-dp0], device 0: rockchip-dp0 spdif-hifi-0 [rockchip-dp0 spdif-hifi-0]
  Subdevices: 0/1
  Subdevice #0: subdevice #0
card 1: rockchiphdmi0 [rockchip-hdmi0], device 0: rockchip-hdmi0 i2s-hifi-0 [rockchip-hdmi0 i2s-hifi-0]
  Subdevices: 0/1
  Subdevice #0: subdevice #0
card 2: rockchipes8388 [rockchip,es8388], device 0: dailink-multicodecs ES8323.3-0011-0 [dailink-multicodecs ES8323.3-0011-0]
  Subdevices: 0/1
  Subdevice #0: subdevice #0
```

3) 然后使用 **aplay** 命令播放下系统自带的音频文件，如果耳机能听到声音说明硬件能正常使用。

```
orangepi@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
```

3.14.2.2. HDMI 音频播放测试

1) 首先使用 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) 然后查看下 HDMI 的声卡序号，从下面的输出可以知道 HDMI 的声卡为 **card 1**。

```
orangepi@orangepi:~$ aplay -l
card 0: rockchipdp0 [rockchip-dp0], device 0: rockchip-dp0 spdif-hifi-0 [rockchip-dp0 spdif-hifi-0]
  Subdevices: 0/1
  Subdevice #0: subdevice #0
card 1: rockchiphdmi0 [rockchip-hdmi0], device 0: rockchip-hdmi0 i2s-hifi-0 [rockchip-hdmi0 i2s-hifi-0]
  Subdevices: 0/1
  Subdevice #0: subdevice #0
.....
```

3) 然后使用 **aplay** 命令播放下系统自带的音频文件，如果 HDMI 显示器或者电视能听到声音说明硬件能正常使用。

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

3.14.2.3. DP 音频播放测试

1) 首先使用 Type-C 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) 然后查看下 DP 的声卡序号，从下面的输出可以知道 HDMI 的声卡为 **card 0**。

```
orangepi@orangepi:~$ aplay -l
card 0: rockchipdp0 [rockchip-dp0], device 0: rockchip-dp0 spdif-hifi-0 [rockchip-dp0 spdif-hifi-0]
  Subdevices: 0/1
  Subdevice #0: subdevice #0
.....
```

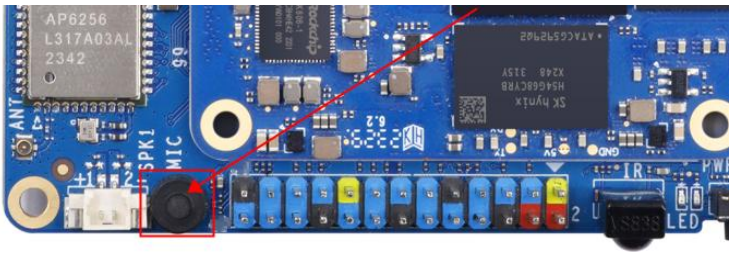
3) 然后使用 **aplay** 命令播放下系统自带的音频文件，如果 HDMI 显示器或者电视能

听到声音说明硬件能正常使用。

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

3.14.3. 使用命令测试录音的方法

1) 开发板上有板载 MIC，位置如下所示：



2) 运行 **test_record.sh main** 命令会通过板载 MIC 录制一段音频，然后播放到 HDMI 和耳机。

```
orangepi@orangepi:~$ test_record.sh main
Start recording: /tmp/test.wav
Recording WAVE '/tmp/test.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
Start playing
Playing WAVE '/tmp/test.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
Playing WAVE '/tmp/test.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
```

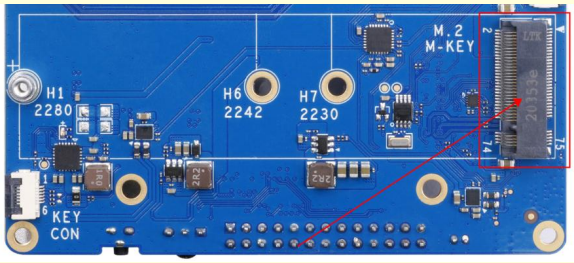
3) 除了板载 MIC，我们还可以通过带 MIC 功能的耳机来录制音频。将带 MIC 功能的耳机插入开发板后，运行 **test_record.sh headset** 命令会通过耳机录制一段音频，然后播放到 HDMI 和耳机。

```
orangepi@orangepi:~$ test_record.sh headset
Start recording: /tmp/test.wav
Recording WAVE '/tmp/test.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
Start playing
Playing WAVE '/tmp/test.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
Playing WAVE '/tmp/test.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
```

3.15. 使用 SATA SSD 的方法

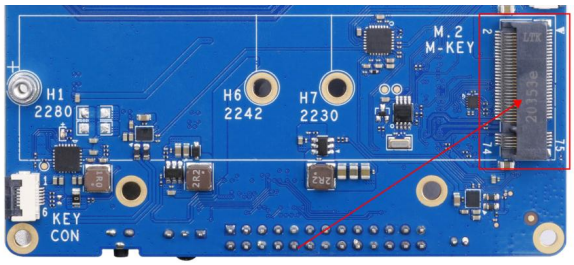
下图所示的 m.2 接口既可以使用 nvme ssd，也可以使用 sata ssd。由于 pcie2.0 控制器和 sata 控制器是二选一的，所以同一时间只能打开其中的一个配置。Orange

Pi 发布的 linux 镜像默认打开的是 pcie 的配置，所以默认只能识别 nvme ssd。如果想使用 sata ssd，需要打开相应的配置才行。



1) 首先需要准备一个 SATA SSD 固态硬盘。

2) 然后把 SSD 插入开发板的 M.2 接口，并固定好。

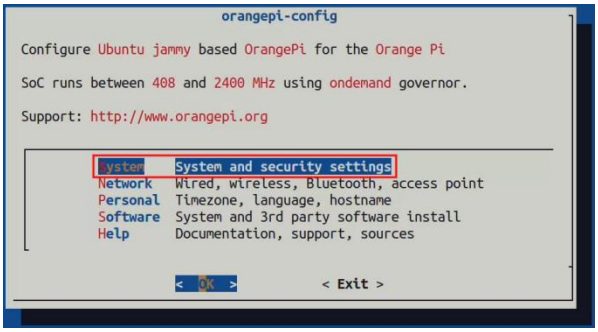


3) sata ssd 的用法目前主要是当扩展存储设备。

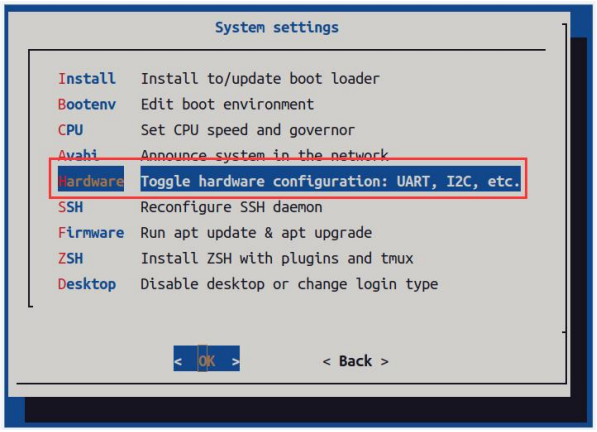
4) 然后运行下 **orangepi-config**，普通用户记得加 **sudo** 权限。

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

5) 然后选择 **System**。



6) 然后选择 **Hardware**。

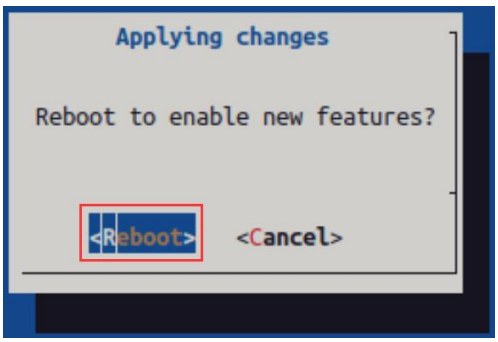


7) 然后使用使用键盘的方向键定位到 **ssd-sata0**，再使用**空格**选中。

8) 然后选择**<Save>**保存。

9) 然后选择**<Back>**。

10) 然后选择**<Reboot>**重启系统使配置生效。



上面的设置最终会在/boot/orangepiEnv.txt 中加入 overlays=ssd-sata0 这行配置。设置完后可以先检查下。如果不存在这行配置，那么设置就是有问题。

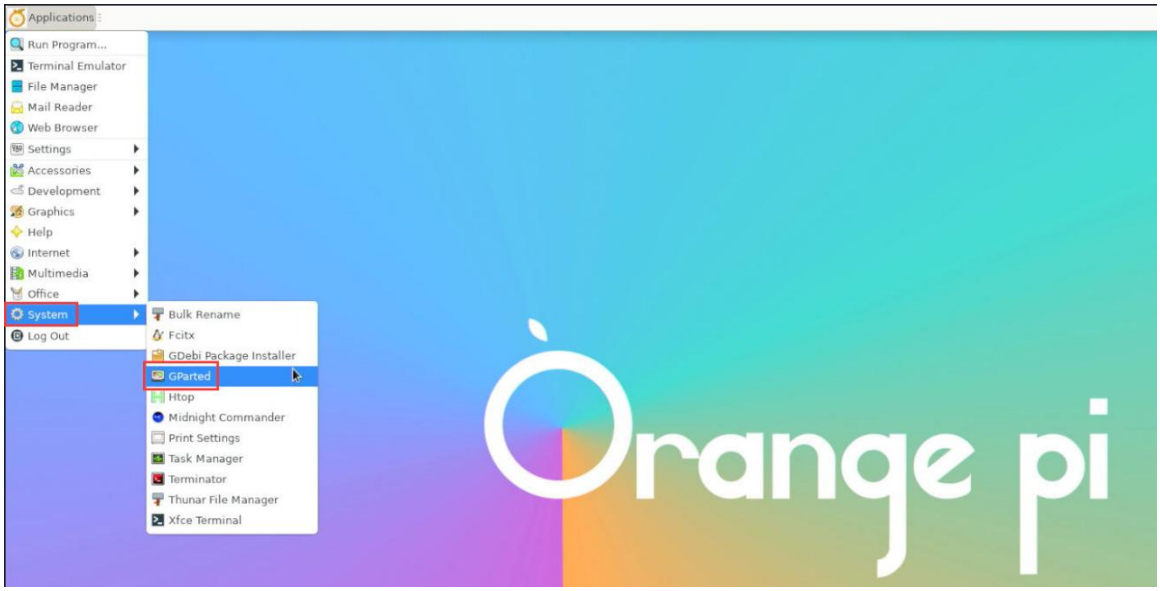
如果觉得使用 orangepi-config 比较麻烦，也可以打开/boot/orangepiEnv.txt，然后加入 overlays=ssd-sata0 这行配置也是可以。

```
orangepi@orangepi:~$ cat /boot/orangepiEnv.txt | grep ssd
overlays=ssd-sata0
```

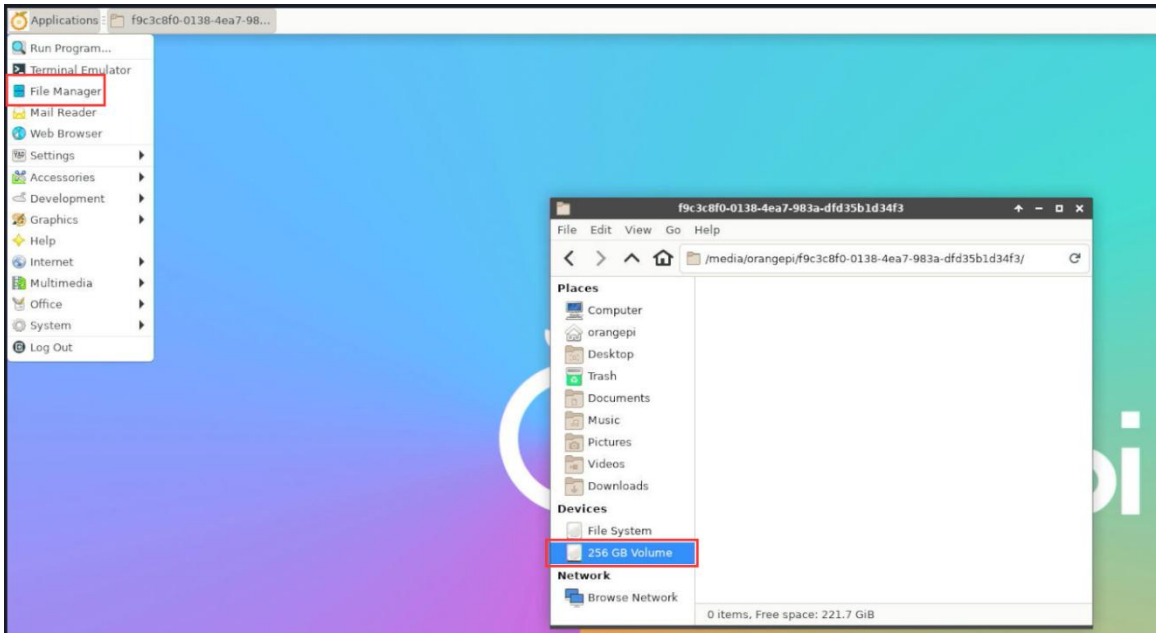
11) 如果一切正常，系统重启后使用 **sudo fdisk -l** 命令就可以看到 sata ssd 的信息。

```
orangepi@orangepi:~$ sudo fdisk -l
.....
Disk /dev/sda: 238.47 GiB, 256060514304 bytes, 500118192 sectors
Disk model: Fanxiang S201 25
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 43FFB292-340D-654C-8C30-6C64AEDAA0F4

Device       Start          End      Sectors     Size Type
/dev/sda1     2048 500117503 500115456 238.5G Linux filesystem
.....
```

12) 然后使用 **GParted** 可以对 sata ssd 格式化或者分区等操作。



13) 然后在文件管理中就可以看到 sata ssd 的设备了。



14) 服务器版本的系统中可以使用 **mount** 命令来挂载 sata ssd 到需要的目录下。

```
orangepi@orangepi:~$ sudo mount /dev/sda1 /mnt
orangepi@orangepi:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            3.8G  8.0K  3.8G   1% /dev
tmpfs           769M  1.4M  768M   1% /run
```

```
/dev/mmcblk1p2     29G    5.9G     23G    21% /
tmpfs              3.8G       0    3.8G     0% /dev/shm
tmpfs              5.0M    4.0K    5.0M     1% /run/lock
tmpfs              3.8G     16K    3.8G     1% /tmp
/dev/mmcblk1p1     256M     90M    166M    36% /boot
/dev/zram1         194M     27M    154M    15% /var/log
tmpfs              769M     60K    769M     1% /run/user/1000
/dev/sda1          234G     28K    222G     1% /mnt
```

3.16. 温度传感器

1) 查看系统温度传感器的命令为：

```
orangepi@orangepi:~$ sensors
gpu_thermal-virtual-0
Adapter: Virtual device
temp1:            +47.2°C

littlecore_thermal-virtual-0
Adapter: Virtual device
temp1:            +47.2°C

bigcore0_thermal-virtual-0
Adapter: Virtual device
temp1:            +47.2°C

tcpm_source_psy_6_0022-i2c-6-22
Adapter: rk3x-i2c
in0:               0.00 V  (min =  +0.00 V, max =  +0.00 V)
curr1:             0.00 A  (max =  +0.00 A)

npu_thermal-virtual-0
Adapter: Virtual device
temp1:            +47.2°C

center_thermal-virtual-0
```

```
Adapter: Virtual device
temp1:          +47.2°C

bigcore1_thermal-virtual-0
Adapter: Virtual device
temp1:          +47.2°C

soc_thermal-virtual-0
Adapter: Virtual device
temp1:          +47.2°C    (crit = +115.0°C)
```

2) 查看 nvme ssd 固态硬盘当前温度的命令为：

```
orangepi@orangepi:~$ sudo smartctl -a /dev/nvme0 | grep "Temperature:"
Temperature:                        40 Celsius
```

3.17. 26 Pin 接口引脚说明

1) Orange Pi CM5 Base Tablet 底板 26 Pin 接口引脚的顺序请参考下图。

2) Orange Pi CM5 Base Tablet 底板 26 Pin 接口引脚的功能如下表所示：

a. 下面是 26 Pin 完整的引脚图。

复用功能	复用功能	复用功能	GPIO	GPIO序号	引脚序号	引脚序号	GPIO序号	GPIO	复用功能	复用功能
			3.3V		1	2		5V		
PWM13_M2(febf0010)	UART1_RX_M1	I2C5_SDA_M3	GPIO1_B7	47	3	4		5V		
	UART1_TX_M1	I2C5_SCL_M3	GPIO1_B6	46	5	6		GND		
	PWM1_M2(fd8b0010)	I2C4_SCL_M3	GPIO1_A3	35	7	8	56	GPIO1_D0	UART6_TX_M2	SPI1_MISO_M2
			GND		9	10	57	GPIO1_D1	UART6_RX_M2	SPI1_MOSI_M2
PWM0_M1(fd8b0000)	UART4_TX_M0	SPI1_CLK_M2	GPIO1_D2	58	11	12	34	GPIO1_A2	I2C4_SDA_M3	
	UART4_RX_M0	SPI1_CS0_M2	GPIO1_D3	59	13	14		GND		
			GPIO1_B0	40	15	16	36	GPIO1_A4		
			3.3V		17	18	38	GPIO1_A6		
		SPI0_MOSI_M2	GPIO1_B2	42	19	20		GND		
		SPI0_MISO_M2	GPIO1_B1	41	21	22	39	GPIO1_A7	PWM3_M3(fd8b0030)	
		SPI0_CLK_M2	GPIO1_B3	43	23	24	44	GPIO1_B4	SPI0_CS0_M2	UART7_RX_M2
			GND		25	26	45	GPIO1_B5	SPI0_CS1_M2	UART7_TX_M2

b. 下面的表格是上面完整表格左半边部分的图，能看得清楚点。

复用功能	复用功能	复用功能	GPIO	GPIO序号	引脚序号
			3.3V		1
PWM13_M2(febf0010)	UART1_RX_M1	I2C5_SDA_M3	GPIO1_B7	47	3
	UART1_TX_M1	I2C5_SCL_M3	GPIO1_B6	46	5
	PWM1_M2(fd8b0010)	I2C4_SCL_M3	GPIO1_A3	35	7
			GND		9
PWM0_M1(fd8b0000)	UART4_TX_M0	SPI1_CLK_M2	GPIO1_D2	58	11
	UART4_RX_M0	SPI1_CS0_M2	GPIO1_D3	59	13
			GPIO1_B0	40	15
			3.3V		17
		SPI0_MOSI_M2	GPIO1_B2	42	19
		SPI0_MISO_M2	GPIO1_B1	41	21
		SPI0_CLK_M2	GPIO1_B3	43	23
			GND		25

c. 下面的表格是上面完整表格右半边部分的图，能看得清楚点。

引脚序号	GPIO序号	GPIO	复用功能	复用功能
2		5V		
4		5V		
6		GND		
8	56	GPIO1_D0	UART6_TX_M2	SPI1_MISO_M2
10	57	GPIO1_D1	UART6_RX_M2	SPI1_MOSI_M2
12	34	GPIO1_A2	I2C4_SDA_M3	
14		GND		
16	36	GPIO1_A4		
18	38	GPIO1_A6		
20		GND		
22	39	GPIO1_A7	PWM3_M3(fd8b0030)	
24	44	GPIO1_B4	SPI0_CS0_M2	UART7_RX_M2
26	45	GPIO1_B5	SPI0_CS1_M2	UART7_TX_M2

上面表格中 pwm 都标出了对应的寄存器的基地址，在查看/sys/class/pwm/中哪个 pwmchip 对应 26pin 排针中哪个 pwm 引脚时很有用。

3) 26pin 接口中总共有 **17** 个 GPIO 口，所有 GPIO 口的电压都是 **3.3v**。

3.18. 安装 wiringOP 的方法

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

编译好的 wiringOP 的 deb 包在 orangepi-build 中的存放路径为：
orangepi-build/external/cache/debs/arm64/wiringpi_x.xx.deb

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

```
orangepi@orangepicm5-tablet:~$ gpio readall
 +------+-----+----------+--------+---+CM5 Tablet+---+--------+----------+-----+------+
 | GPIO | wPi |   Name   |  Mode  | V | Physical | V |  Mode  | Name     | wPi | GPIO |
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 |      |     |     3.3V |        |   |  1 || 2  |   |        | 5V       |     |      |
 |   47 |   0 |    SDA.5 |    OUT | 1 |  3 || 4  |   |        | 5V       |     |      |
 |   46 |   1 |    SCL.5 |    OUT | 1 |  5 || 6  |   |        | GND      |     |      |
 |   35 |   2 |     PWM1 |    OUT | 1 |  7 || 8  | 1 | OUT    | TXD.6    | 3   | 56   |
 |      |     |      GND |        |   |  9 || 10 | 1 | OUT    | RXD.6    | 4   | 57   |
 |   58 |   5 |    RXD.4 |    OUT | 1 | 11 || 12 | 1 | OUT    | GPIO1_A2 | 6   | 34   |
 |   59 |   7 |    TXD.4 |    OUT | 1 | 13 || 14 |   |        | GND      |     |      |
 |   40 |   8 | GPIO1_B0 |    OUT | 1 | 15 || 16 | 1 | OUT    | GPIO1_A4 | 9   | 36   |
 |      |     |     3.3V |        |   | 17 || 18 | 1 | OUT    | GPIO1_A6 | 10  | 38   |
 |   42 |  11 | SPI0_TXD |    OUT | 1 | 19 || 20 |   |        | GND      |     |      |
 |   41 |  12 | SPI0_RXD |    OUT | 1 | 21 || 22 | 1 | OUT    | PWM3     | 13  | 39   |
 |   43 |  14 | SPI0_CLK |    OUT | 1 | 23 || 24 | 1 | OUT    | SPI0_CS0 | 15  | 44   |
 |      |     |      GND |        |   | 25 || 26 | 1 | OUT    | SPI0_CS1 | 16  | 45   |
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 | GPIO | wPi |   Name   |  Mode  | V | Physical | V |  Mode  | Name     | wPi | GPIO |
 +------+-----+----------+--------+---+CM5 Tablet+---+--------+----------+-----+------+
orangepi@orangepicm5-tablet:~$
```

1) 下载 wiringOP 的代码。

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

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

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

2) 编译安装 wiringOP。

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

3) 测试 gpio readall 命令的输出如下：

```
orangepi@orangepicm5-tablet:~$ gpio readall
 +------+-----+----------+--------+---+CM5 Tablet+---+--------+----------+-----+------+
 | GPIO | wPi |   Name   |  Mode  | V | Physical | V |  Mode  | Name     | wPi | GPIO |
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 |      |     |     3.3V |        |   |  1 || 2  |   |        | 5V       |     |      |
 |   47 |   0 |    SDA.5 |    OUT | 1 |  3 || 4  |   |        | 5V       |     |      |
 |   46 |   1 |    SCL.5 |    OUT | 1 |  5 || 6  |   |        | GND      |     |      |
 |   35 |   2 |     PWM1 |    OUT | 1 |  7 || 8  | 1 | OUT    | TXD.6    | 3   | 56   |
 |      |     |      GND |        |   |  9 || 10 | 1 | OUT    | RXD.6    | 4   | 57   |
 |   58 |   5 |    RXD.4 |    OUT | 1 | 11 || 12 | 1 | OUT    | GPIO1_A2 | 6   | 34   |
 |   59 |   7 |    TXD.4 |    OUT | 1 | 13 || 14 |   |        | GND      |     |      |
 |   40 |   8 | GPIO1_B0 |    OUT | 1 | 15 || 16 | 1 | OUT    | GPIO1_A4 | 9   | 36   |
 |      |     |     3.3V |        |   | 17 || 18 | 1 | OUT    | GPIO1_A6 | 10  | 38   |
 |   42 |  11 | SPI0_TXD |    OUT | 1 | 19 || 20 |   |        | GND      |     |      |
 |   41 |  12 | SPI0_RXD |    OUT | 1 | 21 || 22 | 1 | OUT    | PWM3     | 13  | 39   |
 |   43 |  14 | SPI0_CLK |    OUT | 1 | 23 || 24 | 1 | OUT    | SPI0_CS0 | 15  | 44   |
 |      |     |      GND |        |   | 25 || 26 | 1 | OUT    | SPI0_CS1 | 16  | 45   |
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 | GPIO | wPi |   Name   |  Mode  | V | Physical | V |  Mode  | Name     | wPi | GPIO |
 +------+-----+----------+--------+---+CM5 Tablet+---+--------+----------+-----+------+
orangepi@orangepicm5-tablet:~$
```

3.19. 26pin 接口 GPIO、I2C、UART、SPI 和 PWM 测试

3.19.1. 26pin GPIO 口测试

Orange Pi 发布的 linux 系统中有预装一个 blink_all_gpio 程序，这个程序会设置 26pin 中的所有 17 个 GPIO 口不停的切换高低电平。

运行 blink_all_gpio 程序后，当用万用表去测量 GPIO 口的电平时，会发现 GPIO 引脚会在 0 和 3.3v 之间不停的切换。使用这个程序我们可以来测试 GPIO 口是否能正常工作。

运行 blink_all_gpio 程序的方式如下所示：

orangepi@orangepi:~$ **sudo blink_all_gpio** #记得加 sudo 权限

[sudo] password for orangepi: #在这里需要输入密码

1) 开发板 26pin 中总共有 **17** 个 GPIO 口可以使用，下面以 7 号引脚——对应 GPIO 为 GPIO1_A3——对应 wPi 序号为 2——为例演示如何设置 GPIO 口的高低电平。

```
orangepi@orangepicm5-tablet:~$ gpio readall
 +------+-----+----------+--------+---+---+CM5 Tablet+---+--------+----------+-----+------+
 | GPIO | wPi |   Name   |  Mode  | V | Physical | V |  Mode  | Name     | wPi | GPIO |
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 |      |     |     3.3V |        |   |  1 || 2  |   |        | 5V       |     |      |
 |   47 |   0 |    SDA.5 |     IN | 1 |  3 || 4  |   |        | 5V       |     |      |
 |   46 |   1 |    SCL.5 |     IN | 1 |  5 || 6  |   |        | GND      |     |      |
 |   35 |   2 |     PWM1 |     IN | 0 |  7 || 8  | 1 | IN     | TXD.6    | 3   | 56   |
 |      |     |      GND |        |   |  9 || 10 | 1 | IN     | RXD.6    | 4   | 57   |
 |   58 |   5 |    RXD.4 |     IN | 1 | 11 || 12 | 0 | IN     | GPIO1_A2 | 6   | 34   |
 |   59 |   7 |    TXD.4 |  ALT11 | 1 | 13 || 14 |   |        | GND      |     |      |
 |   40 |   8 | GPIO1_B0 |     IN | 1 | 15 || 16 | 0 | IN     | GPIO1_A4 | 9   | 36   |
```

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。

```
orangepi@orangepicm5-tablet:~$ gpio readall
 +------+-----+----------+--------+---+---+CM5 Tablet+---+--------+----------+-----+------+
 | GPIO | wPi |   Name   |  Mode  | V | Physical | V |  Mode  | Name     | wPi | GPIO |
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 |      |     |     3.3V |        |   |  1 || 2  |   |        | 5V       |     |      |
 |   47 |   0 |    SDA.5 |     IN | 1 |  3 || 4  |   |        | 5V       |     |      |
 |   46 |   1 |    SCL.5 |     IN | 1 |  5 || 6  |   |        | GND      |     |      |
 |   35 |   2 |     PWM1 |    OUT | 0 |  7 || 8  | 1 | IN     | TXD.6    | 3   | 56   |
 |      |     |      GND |        |   |  9 || 10 | 1 | IN     | RXD.6    | 4   | 57   |
```

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。

```
orangepi@orangepicm5-tablet:~$ gpio readall
 +------+-----+----------+--------+---+---+CM5 Tablet+---+--------+----------+-----+------+
 | GPIO | wPi |   Name   |  Mode  | V | Physical | V |  Mode  | Name     | wPi | GPIO |
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 |      |     |     3.3V |        |   |  1 || 2  |   |        | 5V       |     |      |
 |   47 |   0 |    SDA.5 |     IN | 1 |  3 || 4  |   |        | 5V       |     |      |
 |   46 |   1 |    SCL.5 |     IN | 1 |  5 || 6  |   |        | GND      |     |      |
 |   35 |   2 |     PWM1 |    OUT | 1 |  7 || 8  | 1 | IN     | TXD.6    | 3   | 56   |
 |      |     |      GND |        |   |  9 || 10 | 1 | IN     | RXD.6    | 4   | 57   |
```

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

3.19.2. 26pin GPIO 口上下拉电阻的设置方法

1) 下面以 7 号引脚——对应 GPIO 为 GPIO1_A3——对应 wPi 序号为 2——为例演示如何设置 GPIO 口的上下拉电阻。

```
orangepi@orangepicm5-tablet:~$ gpio readall
 +------+-----+----------+--------+---+CM5 Tablet+---+--------+----------+-----+------+
 | GPIO | wPi |   Name   |  Mode  | V | Physical | V |  Mode  | Name     | wPi | GPIO |
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 |      |     |     3.3V |        |   |  1 || 2  |   |        | 5V       |     |      |
 |   47 |   0 |    SDA.5 |     IN | 1 |  3 || 4  |   |        | 5V       |     |      |
 |   46 |   1 |    SCL.5 |     IN | 1 |  5 || 6  |   |        | GND      |     |      |
 |   35 |   2 |     PWM1 |     IN | 0 |  7 || 8  | 1 | IN     | TXD.6    | 3   | 56   |
 |      |     |      GND |        |   |  9 || 10 | 1 | IN     | RXD.6    | 4   | 57   |
 |   58 |   5 |    RXD.4 |     IN | 1 | 11 || 12 | 0 | IN     | GPIO1_A2 | 6   | 34   |
 |   59 |   7 |    TXD.4 |  ALT11 | 1 | 13 || 14 |   |        | GND      |     |      |
 |   40 |   8 | GPIO1_B0 |     IN | 1 | 15 || 16 | 0 | IN     | GPIO1_A4 | 9   | 36   |
```

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

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

3) 设置为输入模式后，执行下面的命令可以设置 GPIO 口为上拉模式。

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

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

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

5) 然后执行下面的命令可以设置 GPIO 口为下拉模式。

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

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

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

3.19.3. 26pin I2C 测试

1) 由下表可知，Orange Pi CM5 Base Tablet 可用的 i2c 为 i2c4 和 i2c5 共两组 i2c 总

线。

复用功能	复用功能	复用功能	GPIO	GPIO序号	引脚序号	引脚序号	GPIO序号	GPIO	复用功能	复用功能
			3.3V		1	2		5V		
PWM13_M2(febf0010)	UART1_RX_M1	I2C5_SDA_M3	GPIO1_B7	47	3	4		5V		
	UART1_TX_M1	I2C5_SCL_M3	GPIO1_B6	46	5	6		GND		
	PWM1_M2(fd8b0010)	I2C4_SCL_M3	GPIO1_A3	35	7	8	56	GPIO1_D0	UART6_TX_M2	SPI1_MISO_M2
			GND		9	10	57	GPIO1_D1	UART6_RX_M2	SPI1_MOSI_M2
PWM0_M1(fd8b0000)	UART4_TX_M0	SPI1_CLK_M2	GPIO1_D2	58	11	12	34	GPIO1_A2	I2C4_SDA_M3	
	UART4_RX_M0	SPI1_CS0_M2	GPIO1_D3	59	13	14		GND		
			GPIO1_B0	40	15	16	36	GPIO1_A4		
			3.3V		17	18	38	GPIO1_A6		
		SPI0_MOSI_M2	GPIO1_B2	42	19	20		GND		
		SPI0_MISO_M2	GPIO1_B1	41	21	22	39	GPIO1_A7	PWM3_M3(fd8b0030)	
		SPI0_CLK_M2	GPIO1_B3	43	23	24	44	GPIO1_B4	SPI0_CS0_M2	UART7_RX_M2
			GND		25	26	45	GPIO1_B5	SPI0_CS1_M2	UART7_TX_M2

2) 两组 I2C 总线在 26pin 中对应的引脚如下表所示：

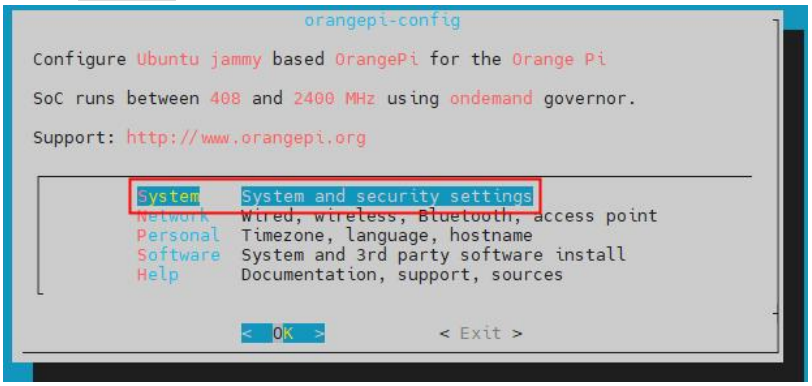
I2C 总线	SDA 对应 26pin	SCL 对应 26pin	dtbo 对应配置
I2C4_M3	12 号引脚	7 号引脚	i2c4-m3
I2C5_M3	3 号引脚	5 号引脚	i2c5-m3

3) 在 linux 系统中，26 Pin 中的 I2C 总线默认都是关闭的，需要手动打开才能使用。详细步骤如下所示：

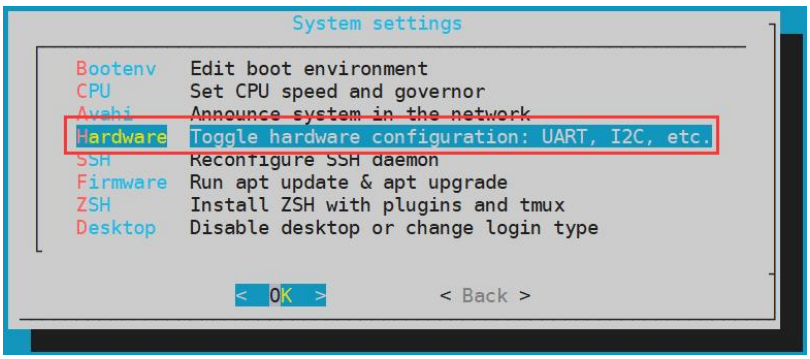
a. 首先运行下 **orangepi-config**，普通用户记得加 **sudo** 权限。

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

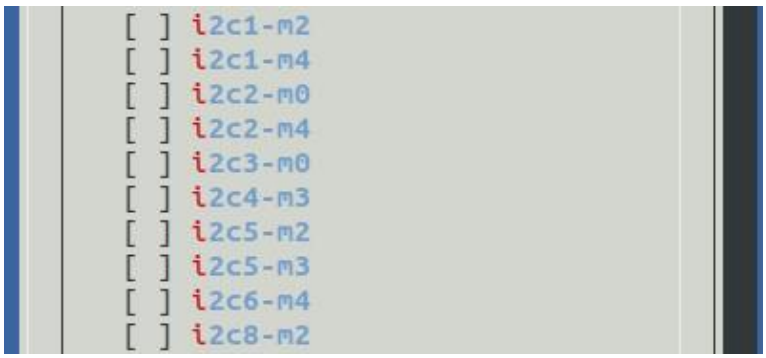
b. 然后选择 **System**。



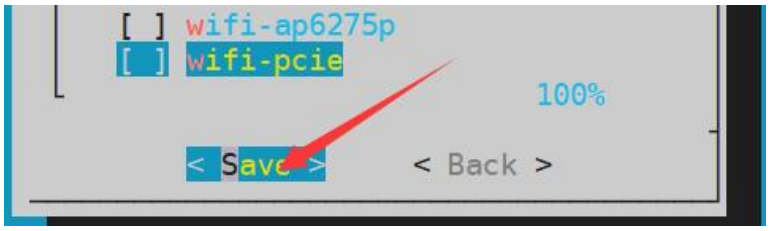
c. 然后选择 **Hardware**。



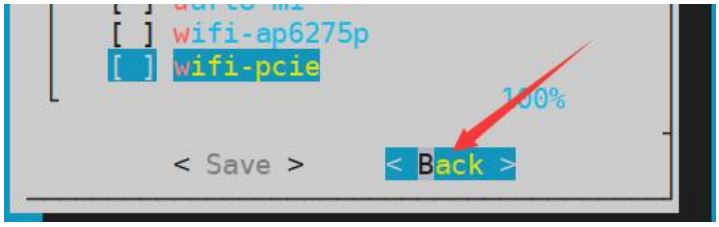
d. 然后使用键盘的方向键定位到下图所示的位置，再使用**空格**选中想要打开的 I2C 的配置。



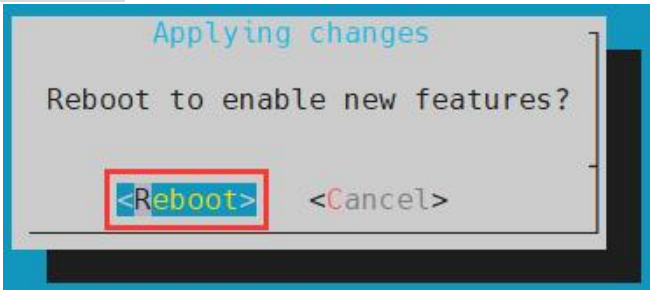
e. 然后选择**<Save>**保存。



f. 然后选择**<Back>**。



g. 然后选择**<Reboot>**重启系统使配置生效。



4) 启动 linux 系统后，先确认下**/dev** 下存在需要使用 I2C 的设备节点。

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

5) 然后在 26pin 接头的 i2c 引脚上接一个 i2c 设备。

6) 然后使用 **i2cdetect -y** 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。

```
orangepi@orangepi:~$ sudo i2cdetect -y 4      #i2c4 的命令
orangepi@orangepi:~$ sudo i2cdetect -y 5      #i2c5 的命令
```

3.19.4. 26pin UART 测试

1) 由下表可知，Orange Pi CM5 Base Tablet 可用的 uart 为 uart1、uart4、uart6 和 uart7 共四组 uart 总线。

复用功能	复用功能	复用功能	GPIO	GPIO序号	引脚序号	引脚序号	GPIO序号	GPIO	复用功能	复用功能
			3.3V		1	2		5V		
PWM13_M2(febf0010)	UART1_RX_M1	I2C5_SDA_M3	GPIO1_B7	47	3	4		5V		
	UART1_TX_M1	I2C5_SCL_M3	GPIO1_B6	46	5	6		GND		
	PWM1_M2(fd8b0010)	I2C4_SCL_M3	GPIO1_A3	35	7	8	56	GPIO1_D0	UART6_TX_M2	SPI1_MISO_M2
			GND		9	10	57	GPIO1_D1	UART6_RX_M2	SPI1_MOSI_M2
PWM0_M1(fd8b0000)	UART4_TX_M0	SPI1_CLK_M2	GPIO1_D2	58	11	12	34	GPIO1_A2	I2C4_SDA_M3	
	UART4_RX_M0	SPI1_CS0_M2	GPIO1_D3	59	13	14		GND		
			GPIO1_B0	40	15	16	36	GPIO1_A4		
			3.3V		17	18	38	GPIO1_A6		
		SPI0_MOSI_M2	GPIO1_B2	42	19	20		GND		
		SPI0_MISO_M2	GPIO1_B1	41	21	22	39	GPIO1_A7	PWM3_M3(fd8b0030)	
		SPI0_CLK_M2	GPIO1_B3	43	23	24	44	GPIO1_B4	SPI0_CS0_M2	UART7_RX_M2
			GND		25	26	45	GPIO1_B5	SPI0_CS1_M2	UART7_TX_M2

2) 四组 UART 总线在 26pin 中对应的引脚如下表所示：

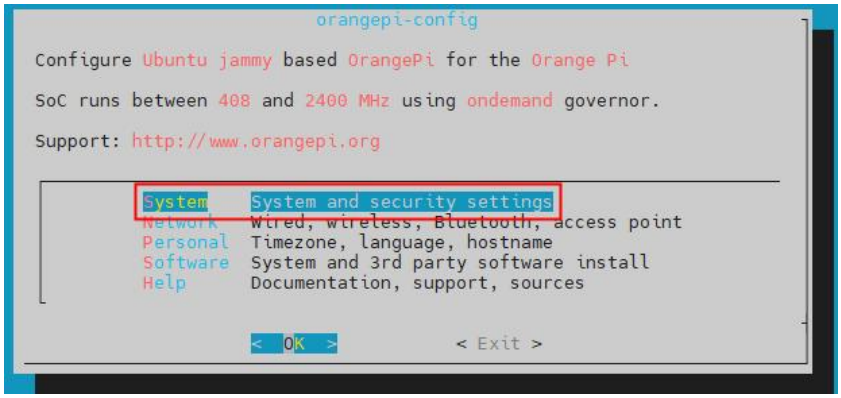
UART 总线	RX 对应 26pin	TX 对应 26pin	dtbo 对应配置
UART1_M1	3 号引脚	5 号引脚	uart1-m1
UART4_M0	13 号引脚	11 号引脚	uart4-m0
UART6_M2	10 号引脚	8 号引脚	uart6-m2
UART7_M2	24 号引脚	26 号引脚	uart7-m2

3) 在 linux 系统中，26 Pin 中的 UART 默认都是关闭的，需要手动打开才能使用。详细步骤如下所示：

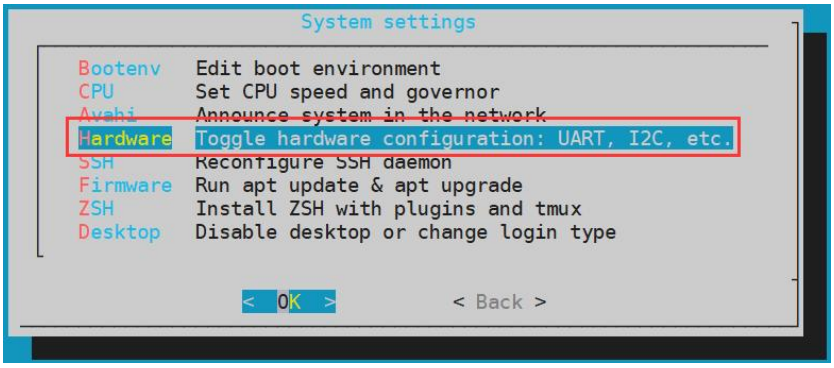
a. 首先运行下 **orangepi-config**，普通用户记得加 **sudo** 权限。

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

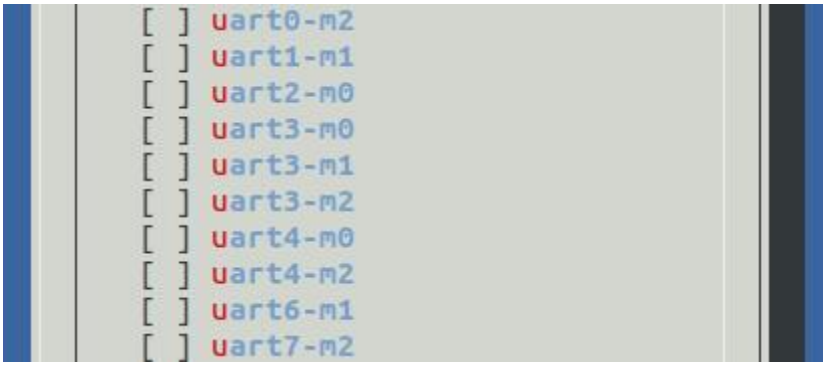
b. 然后选择 **System**。



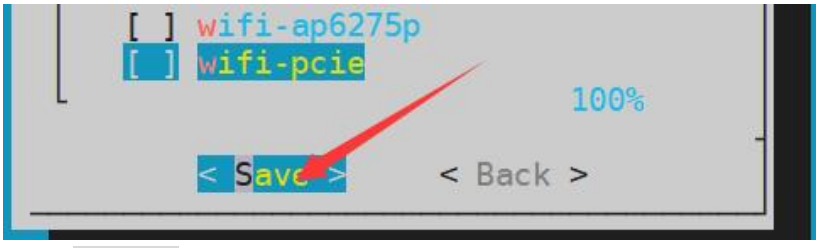
c. 然后选择 **Hardware**。



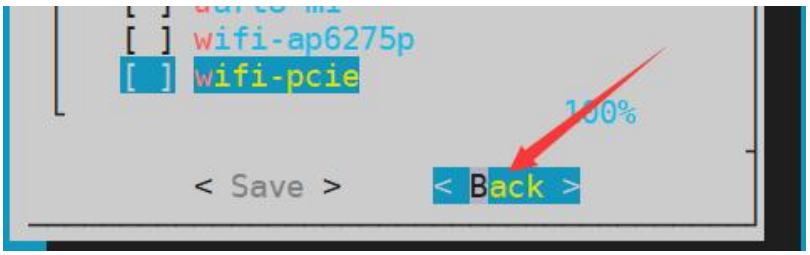
d. 然后使用键盘的方向键定位到下图所示的位置，再使用**空格**选中想要打开的 UART 的配置。



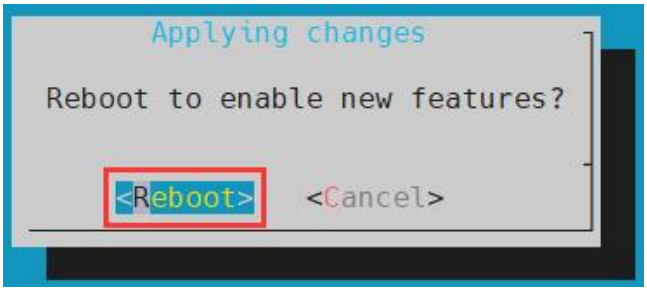
e. 然后选择**<Save>**保存。



f. 然后选择**<Back>**。



g. 然后选择**<Reboot>**重启系统使配置生效。



4) 进入 linux 系统后，先确认下/dev 下是否存在对应 uart 的设备节点。

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

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

6) 使用 **gpio serial** 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（ttySX 需要替换为对应 uart 的节点名，请不要照抄）。

```
orangepi@orangepi:~$ sudo gpio serial /dev/ttySX
[sudo] password for orangepi:   #在这里输入密码

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

3.19.5. 26pin SPI 测试

1) 由下表可知，Orange Pi CM5 Base Tablet 可用的 spi 为 spi0 和 spi1。

复用功能	复用功能	复用功能	GPIO	GPIO序号	引脚序号	引脚序号	GPIO序号	GPIO	复用功能	复用功能
			3.3V		1	2		5V		
PWM13_M2(febf0010)	UART1_RX_M1	I2C5_SDA_M3	GPIO1_B7	47	3	4		5V		
	UART1_TX_M1	I2C5_SCL_M3	GPIO1_B6	46	5	6		GND		
	PWM1_M2(fd8b0010)	I2C4_SCL_M3	GPIO1_A3	35	7	8	56	GPIO1_D0	UART6_TX_M2	SPI1_MISO_M2
			GND		9	10	57	GPIO1_D1	UART6_RX_M2	SPI1_MOSI_M2
PWM0_M1(fd8b0000)	UART4_TX_M0	SPI1_CLK_M2	GPIO1_D2	58	11	12	34	GPIO1_A2	I2C4_SDA_M3	
	UART4_RX_M0	SPI1_CS0_M2	GPIO1_D3	59	13	14		GND		
			GPIO1_B0	40	15	16	36	GPIO1_A4		
			3.3V		17	18	38	GPIO1_A6		
		SPI0_MOSI_M2	GPIO1_B2	42	19	20		GND		
		SPI0_MISO_M2	GPIO1_B1	41	21	22	39	GPIO1_A7	PWM3_M3(fd8b0030)	
		SPI0_CLK_M2	GPIO1_B3	43	23	24	44	GPIO1_B4	SPI0_CS0_M2	UART7_RX_M2
			GND		25	26	45	GPIO1_B5	SPI0_CS1_M2	UART7_TX_M2

2) SPI0 和 SPI4 在 26pin 中对应的引脚如下表所示。

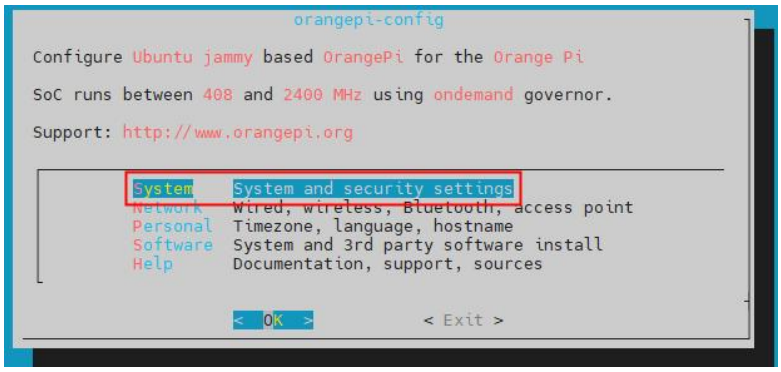
	SPI1_M2 对应 26pin	**SPI0_M2 对应 26pin**
MOSI	**10 号引脚**	**19 号引脚**
MISO	**8 号引脚**	**21 号引脚**
CLK	**11 号引脚**	**23 号引脚**
CS0	**13 号引脚**	**24 号引脚**
CS1	无	**26 号引脚**
dtbo 配置	**spi1-m2-cs0-spidev**	**spi0-m2-cs0-spidev** **spi0-m2-cs1-spidev** **spi0-m2-cs0-cs1-spidev**

3) 在 linux 系统中，26 Pin 中的 SPI 默认都是关闭的，需要手动打开才能使用。详细步骤如下所示：

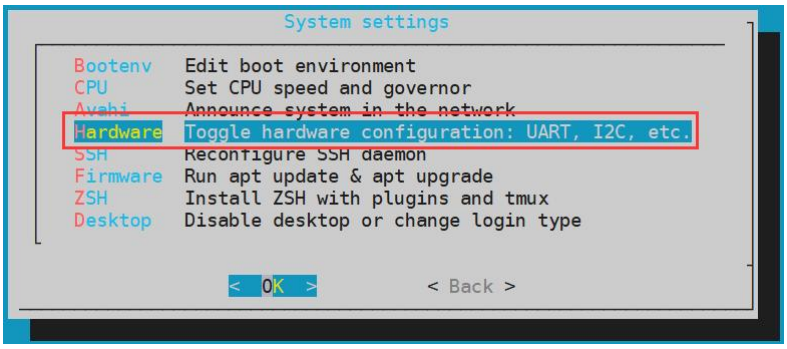
a) 首先运行下 **orangepi-config**，普通用户记得加 **sudo** 权限。

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

b) 然后选择 **System**。

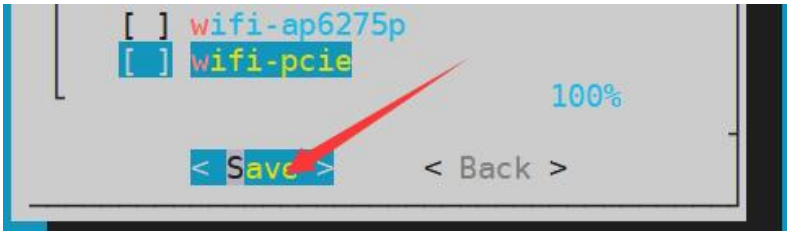


c) 然后选择 **Hardware**。

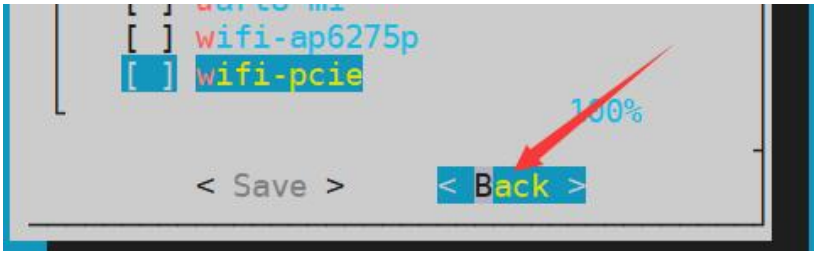


d) 然后使用键盘的方向键定位到下图所示的位置，再使用**空格**选中想要打开的 SPI 的配置。

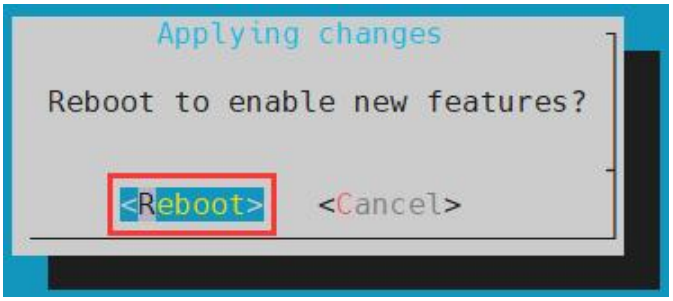
e) 然后选择**<Save>**保存。



f) 然后选择**<Back>**。



g) 然后选择**<Reboot>**重启系统使配置生效。



4) 重启后进入系统先查看下 linux 系统中是否存在 **spidevx.x** 的设备节点，如果存在，说明 SPI 已经设置好了，可以直接使用。

```
orangepi@orangepi:~$ ls /dev/spidev*
/dev/spidev0.0  /dev/spidev0.1  /dev/spidev1.0
```

上面是打开 spi1-m2-cs0-spidev 和 spi0-m2-cs0-cs1-spidev 后显示的结果。

5) 先不短接 SPI0 或者 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。

```
orangepi@orangepi:~$ sudo spidev_test -v -D /dev/spidev0.0
或者
orangepi@orangepi:~$ sudo spidev_test -v -D /dev/spidev0.1
或者
orangepi@orangepi:~$ sudo spidev_test -v -D /dev/spidev1.0
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  | ................................
```

6) 然后短接 SPI0 或者 SPI1 的 mosi 和 miso 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。

```
orangepi@orangepi:~$ sudo spidev_test -v -D /dev/spidev0.0
或者
orangepi@orangepi:~$ sudo spidev_test -v -D /dev/spidev0.1
```

```
或者
orangepi@orangepi:~$ sudo spidev_test -v -D /dev/spidev1.0
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    | ......@.....................
```

3.19.6. 使用/sys/class/pwm 测试 PWM 的方法

1) 由下表可知，Orange Pi CM5 Base Tablet 可用的 pwm 有 pwm0、pwm1、pwm3、和 pwm13 共四路 pwm。

复用功能	复用功能	复用功能	GPIO	GPIO序号	引脚序号	引脚序号	GPIO序号	GPIO	复用功能	复用功能
			3.3V		1	2		5V		
PWM13_M2(febf0010)	UART1_RX_M1	I2C5_SDA_M3	GPIO1_B7	47	3	4		5V		
	UART1_TX_M1	I2C5_SCL_M3	GPIO1_B6	46	5	6		GND		
	PWM1_M2(fd8b0010)	I2C4_SCL_M3	GPIO1_A3	35	7	8	56	GPIO1_D0	UART6_TX_M2	SPI1_MISO_M2
			GND		9	10	57	GPIO1_D1	UART6_RX_M2	SPI1_MOSI_M2
PWM0_M1(fd8b0000)	UART4_TX_M0	SPI1_CLK_M2	GPIO1_D2	58	11	12	34	GPIO1_A2	I2C4_SDA_M3	
	UART4_RX_M0	SPI1_CS0_M2	GPIO1_D3	59	13	14		GND		
			GPIO1_B0	40	15	16	36	GPIO1_A4		
			3.3V		17	18	38	GPIO1_A6		
		SPI0_MOSI_M2	GPIO1_B2	42	19	20		GND		
		SPI0_MISO_M2	GPIO1_B1	41	21	22	39	GPIO1_A7	PWM3_M3(fd8b0030)	
		SPI0_CLK_M2	GPIO1_B3	43	23	24	44	GPIO1_B4	SPI0_CS0_M2	UART7_RX_M2
			GND		25	26	45	GPIO1_B5	SPI0_CS1_M2	UART7_TX_M2

2) PWM 在 26pin 中对应的引脚如下表所示：

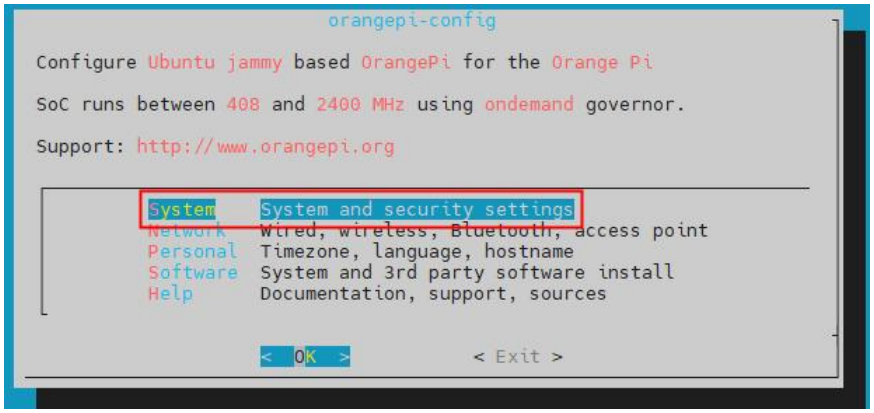
PWM 总线	对应 26pin	dtbo 对应配置
PWM0_M1	11 号引脚	pwm0-m1
PWM1_M2	7 号引脚	pwm1-m2
PWM3_M3	22 号引脚	pwm3-m3
PWM13_M2	3 号引脚	pwm13-m2

3) 在 linux 系统中，26 Pin 中的 PWM 默认都是关闭的，需要手动打开才能使用。详细步骤如下所示：

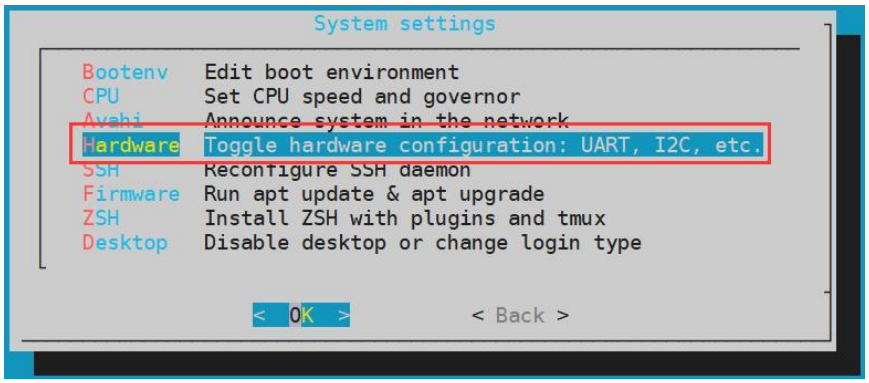
a. 首先运行下 **orangepi-config**，普通用户记得加 **sudo** 权限。

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

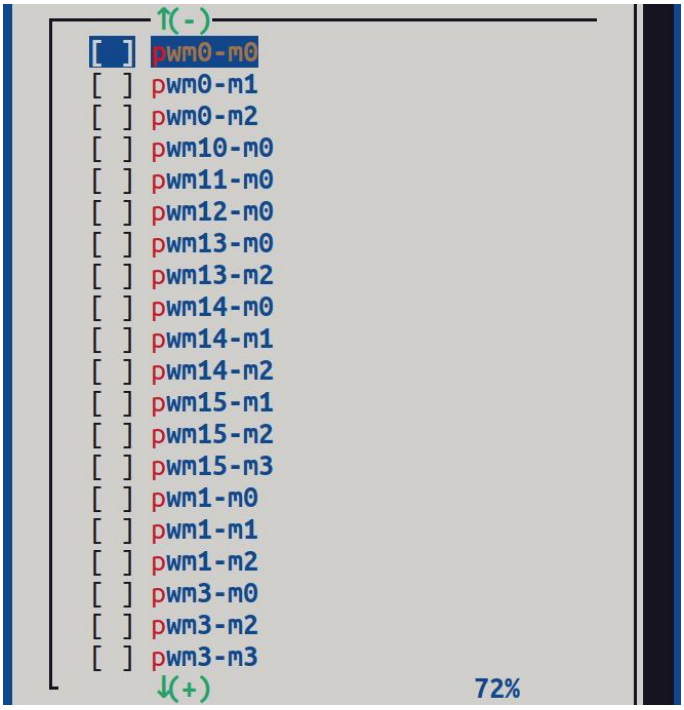
b. 然后选择 **System**。



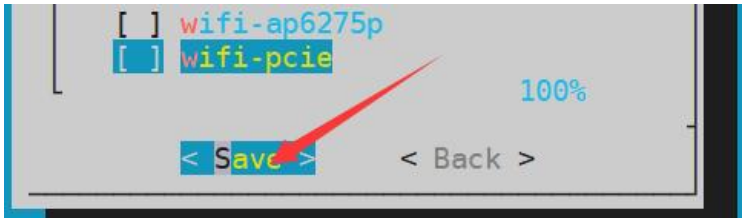
c. 然后选择 **Hardware**。



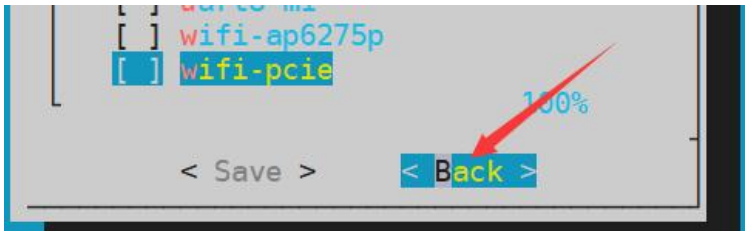
d. 然后使用键盘的方向键定位到下图所示的位置，再使用**空格**选中想要打开的 PWM 的配置。



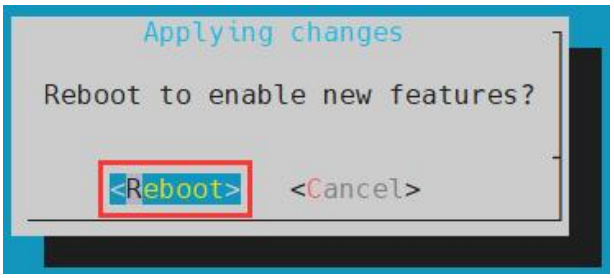
e. 然后选择**<Save>**保存。



f. 然后选择**<Back>**。



g. 然后选择**<Reboot>**重启系统使配置生效。



4) 当打开一个 pwm 后，在**/sys/class/pwm/**中就会多出一个 pwmchipX（X 为具体的的数字），比如打开 pwm13 后，查看**/sys/class/pwm/**下的 pwmchipX 会由两个变成了三个。

```
orangepi@orangepi:~$ ls /sys/class/pwm/
pwmchip0  pwmchip1  pwmchip2
```

5) 上面哪个 pwmchip 对应 pwm13 呢，我们先查看下 **ls /sys/class/pwm/ -l** 命令的输出，如下所示：

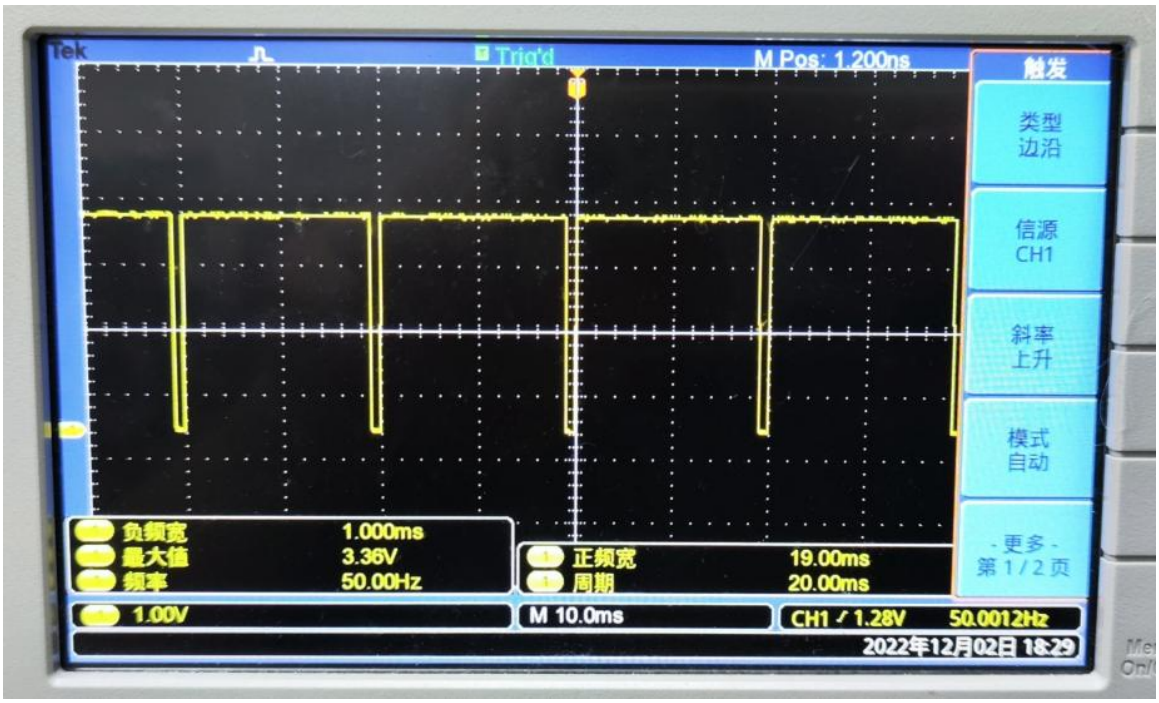
```
orangepi@orangepi:~$ ls /sys/class/pwm/ -lh
total 0
lrwxrwxrwx 1 root root 0 Jan  1  1970 pwmchip0 -> ../../devices/platform/febf0010.pwm/pwm/pwmchip0
lrwxrwxrwx 1 root root 0 Jan  1  1970 pwmchip1 -> ../../devices/platform/febf0020.pwm/pwm/pwmchip1
lrwxrwxrwx 1 root root 0 Jan  1  1970 pwmchip2 -> ../../devices/platform/febf0030.pwm/pwm/pwmchip2
orangepi@orangepi:~$
```

6) 然后由下表可知，pwm13 寄存器的基地址为 febf0010，再看 **ls /sys/class/pwm/ -l** 命令的输出，可以看到 pwmchip0 中链接到了 febf0010.pwm，所以 pwm13 对应 pwmchip 为 pwmchip0。

复用功能	复用功能	复用功能	GPIO	GPIO序号	引脚序号
			3.3V		1
PWM13_M2(febf0010)	UART1_RX_M1	I2C5_SDA_M3	GPIO1_B7	47	3
	UART1_TX_M1	I2C5_SCL_M3	GPIO1_B6	46	5
	PWM1_M2(fd8b0010)	I2C4_SCL_M3	GPIO1_A3	35	7
			GND		9
PWM0_M1(fd8b0000)	UART4_TX_M0	SPI1_CLK_M2	GPIO1_D2	58	11
	UART4_RX_M0	SPI1_CS0_M2	GPIO1_D3	59	13
			GPIO1_B0	40	15
			3.3V		17
		SPI0_MOSI_M2	GPIO1_B2	42	19
		SPI0_MISO_M2	GPIO1_B1	41	21
		SPI0_CLK_M2	GPIO1_B3	43	23
			GND		25

7) 然后使用下面的命令可以让 pwm13 输出一个 50Hz 的方波（请先切换到 root 用户，再执行下面的命令）。

```
root@orangepi:~# echo 0 > /sys/class/pwm/pwmchip0/export
root@orangepi:~# echo 20000000 > /sys/class/pwm/pwmchip0/pwm0/period
root@orangepi:~# echo 1000000 > /sys/class/pwm/pwmchip0/pwm0/duty_cycle
root@orangepi:~# echo 1 > /sys/class/pwm/pwmchip0/pwm0/enable
```



8) 上面演示的 pwm13 的测试方法，其他 pwm 测试方法都是类似的。

3.20. 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。

另外请注意下面所有的命令都是在 root 用户下操作的。

3.20.1. wiringOP-Python 的安装方法

1) 首先安装依赖包。

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

2) 然后使用下面的命令下载 wiringOP-Python 的源码。

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

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

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

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

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

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

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

NAME
    wiringpi

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

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

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

a. 首先使用 python3 命令进入 python3 的命令行模式。

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

b. 然后导入 wiringpi 的 python 模块。

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

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

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

NAME
    wiringpi

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

CLASSES
    builtins.object
        GPIO
        I2C
        Serial
        nes

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

>>>
```

3.20.2. 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。

```
orangepi@orangepicm5-tablet:~$ gpio readall
 +------+-----+----------+--------+---+---CM5 Tablet---+---+--------+----------+-----+------+
 | GPIO | wPi |   Name   |  Mode  | V | Physical | V |  Mode  | Name     | wPi | GPIO |
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 |      |     |     3.3V |        |   |  1 || 2  |   |        | 5V       |     |      |
 |   47 |   0 |    SDA.5 |    OUT | 1 |  3 || 4  |   |        | 5V       |     |      |
 |   46 |   1 |    SCL.5 |    OUT | 1 |  5 || 6  |   |        | GND      |     |      |
 |   35 |   2 |     PWM1 |    OUT | 1 |  7 || 8  | 1 | OUT    | TXD.6    | 3   | 56   |
 |      |     |      GND |        |   |  9 || 10 | 1 | OUT    | RXD.6    | 4   | 57   |
 |   58 |   5 |    RXD.4 |    OUT | 1 | 11 || 12 | 1 | OUT    | GPIO1_A2 | 6   | 34   |
 |   59 |   7 |    TXD.4 |    OUT | 1 | 13 || 14 |   |        | GND      |     |      |
 |   40 |   8 | GPIO1_B0 |    OUT | 1 | 15 || 16 | 1 | OUT    | GPIO1_A4 | 9   | 36   |
 |      |     |     3.3V |        |   | 17 || 18 | 1 | OUT    | GPIO1_A6 | 10  | 38   |
 |   42 |  11 | SPI0_TXD |    OUT | 1 | 19 || 20 |   |        | GND      |     |      |
 |   41 |  12 | SPI0_RXD |    OUT | 1 | 21 || 22 | 1 | OUT    | PWM3     | 13  | 39   |
 |   43 |  14 | SPI0_CLK |    OUT | 1 | 23 || 24 | 1 | OUT    | SPI0_CS0 | 15  | 44   |
 |      |     |      GND |        |   | 25 || 26 | 1 | OUT    | SPI0_CS1 | 16  | 45   |
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 | GPIO | wPi |   Name   |  Mode  | V | Physical | V |  Mode  | Name     | wPi | GPIO |
 +------+-----+----------+--------+---+---CM5 Tablet---+---+--------+----------+-----+------+
orangepi@orangepicm5-tablet:~$
```

1) 下面以 7 号引脚——对应 GPIO 为 GPIO1_A3 ——对应 wPi 序号为 2——为例演示如何设置 GPIO 口的高低电平。

```
orangepi@orangepicm5-tablet:~$ gpio readall
 +------+-----+----------+--------+---+---CM5 Tablet---+---+--------+----------+-----+------+
 | GPIO | wPi |   Name   |  Mode  | V | Physical | V |  Mode  | Name     | wPi | GPIO |
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 |      |     |     3.3V |        |   |  1 || 2  |   |        | 5V       |     |      |
 |   47 |   0 |    SDA.5 |     IN | 1 |  3 || 4  |   |        | 5V       |     |      |
 |   46 |   1 |    SCL.5 |     IN | 1 |  5 || 6  |   |        | GND      |     |      |
 |   35 |   2 |     PWM1 |     IN | 0 |  7 || 8  | 1 | IN     | TXD.6    | 3   | 56   |
 |      |     |      GND |        |   |  9 || 10 | 1 | IN     | RXD.6    | 4   | 57   |
 |   58 |   5 |    RXD.4 |     IN | 1 | 11 || 12 | 0 | IN     | GPIO1_A2 | 6   | 34   |
 |   59 |   7 |    TXD.4 |  ALT11 | 1 | 13 || 14 |   |        | GND      |     |      |
 |   40 |   8 | GPIO1_B0 |     IN | 1 | 15 || 16 | 0 | IN     | GPIO1_A4 | 9   | 36   |
```

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
```

3.20.3. 26pin I2C 测试

1) 由下表可知，Orange Pi CM5 Base Tablet 可用的 i2c 为 i2c4 和 i2c5 共两组 i2c 总线。

复用功能	复用功能	复用功能	GPIO	GPIO序号	引脚序号	引脚序号	GPIO序号	GPIO	复用功能	复用功能
			3.3V		1	2		5V		
PWM13_M2(febf0010)	UART1_RX_M1	I2C5_SDA_M3	GPIO1_B7	47	3	4		5V		
	UART1_TX_M1	I2C5_SCL_M3	GPIO1_B6	46	5	6		GND		
	PWM1_M2(fd8b0010)	I2C4_SCL_M3	GPIO1_A3	35	7	8	56	GPIO1_D0	UART6_TX_M2	SPI1_MISO_M2
			GND		9	10	57	GPIO1_D1	UART6_RX_M2	SPI1_MOSI_M2
PWM0_M1(fd8b0000)	UART4_TX_M0	SPI1_CLK_M2	GPIO1_D2	58	11	12	34	GPIO1_A2	I2C4_SDA_M3	
	UART4_RX_M0	SPI1_CS0_M2	GPIO1_D3	59	13	14		GND		
			GPIO1_B0	40	15	16	36	GPIO1_A4		
			3.3V		17	18	38	GPIO1_A6		
		SPI0_MOSI_M2	GPIO1_B2	42	19	20		GND		
		SPI0_MISO_M2	GPIO1_B1	41	21	22	39	GPIO1_A7	PWM3_M3(fd8b0030)	
		SPI0_CLK_M2	GPIO1_B3	43	23	24	44	GPIO1_B4	SPI0_CS0_M2	UART7_RX_M2
			GND		25	26	45	GPIO1_B5	SPI0_CS1_M2	UART7_TX_M2

2) 两组 I2C 总线在 26pin 中对应的引脚如下表所示：

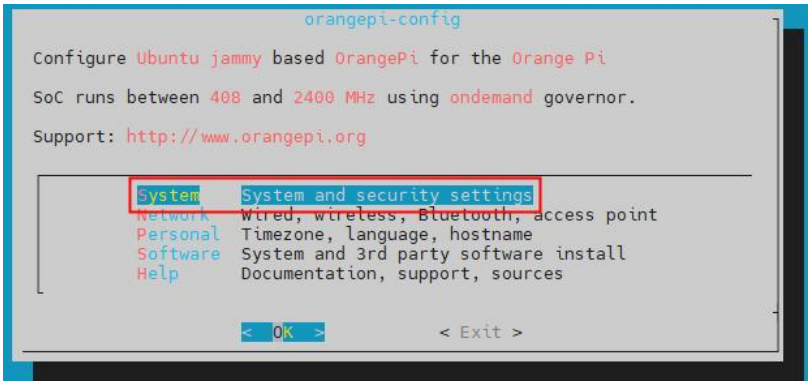
I2C 总线	SDA 对应 26pin	SCL 对应 26pin	dtbo 对应配置
I2C4_M3	12 号引脚	7 号引脚	i2c4-m3
I2C5_M3	3 号引脚	5 号引脚	i2c5-m3

3) 在 linux 系统中，26 Pin 中的 I2C 总线默认都是关闭的，需要手动打开才能使用。详细步骤如下所示：

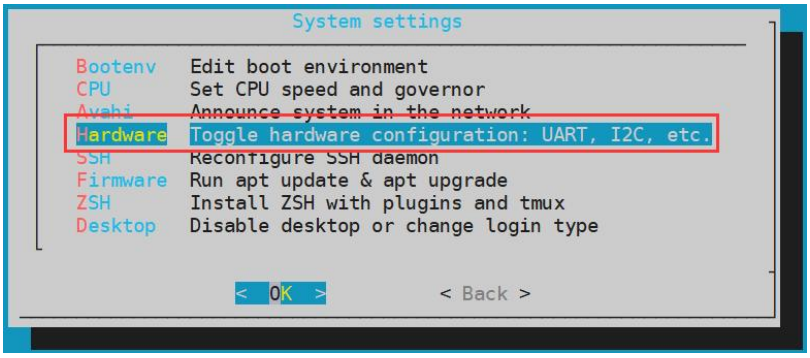
a. 首先运行下 **orangepi-config**，普通用户记得加 **sudo** 权限。

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

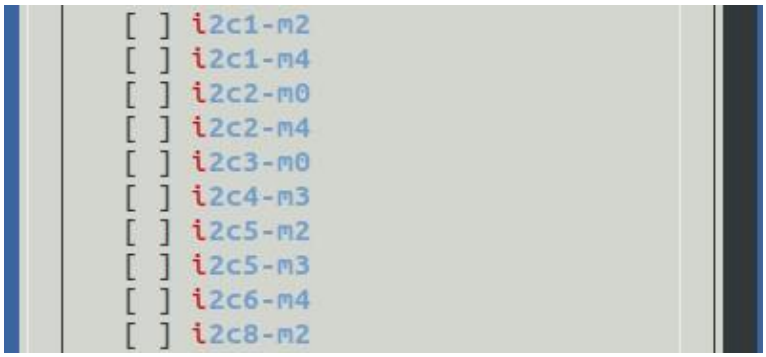
b. 然后选择 **System**。



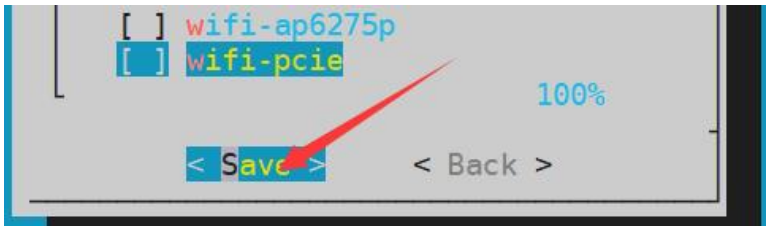
c. 然后选择 **Hardware**。



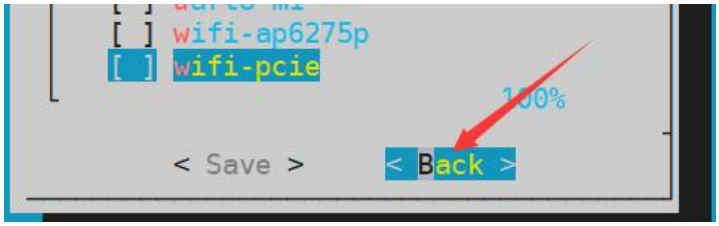
d. 然后使用键盘的方向键定位到下图所示的位置，再使用**空格**选中想要打开的 I2C 的配置。



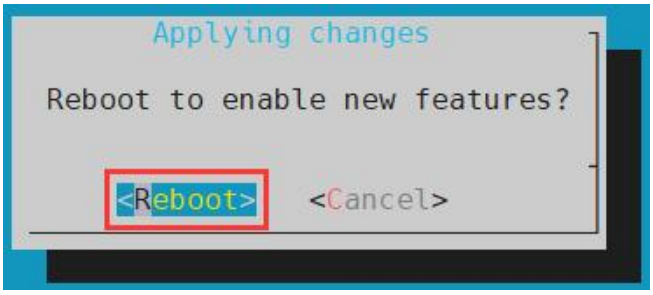
e. 然后选择**<Save>**保存。



f. 然后选择**<Back>**。



g. 然后选择**<Reboot>**重启系统使配置生效。



4) 启动 linux 系统后，先确认下**/dev** 下存在需要使用 I2C 的设备节点。

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

5) 然后在 26pin 接头的 i2c 引脚上接一个 i2c 设备，这里以 ds1307 RTC 模块为例

6) 然后使用 **i2cdetect -y** 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

```
orangepi@orangepi:~$ sudo i2cdetect -y 4      #i2c4 的命令
orangepi@orangepi:~$ sudo i2cdetect -y 5      #i2c5 的命令
```

7) 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-4"
Thu 2023-01-05  14:57:55
Thu 2023-01-05  14:57:56
Thu 2023-01-05  14:57:57
^C
exit
```

3.20.4. 26pin UART 测试

1) 由下表可知，Orange Pi CM5 Base Tablet 可用的 uart 为 uart1、uart4、uart6 和 uart7 共四组 uart 总线。

复用功能	复用功能	复用功能	GPIO	GPIO序号	引脚序号	引脚序号	GPIO序号	GPIO	复用功能	复用功能
			3.3V		1	2		5V		
PWM13_M2(febf0010)	UART1_RX_M1	I2C5_SDA_M3	GPIO1_B7	47	3	4		5V		
	UART1_TX_M1	I2C5_SCL_M3	GPIO1_B6	46	5	6		GND		
	PWM1_M2(fd8b0010)	I2C4_SCL_M3	GPIO1_A3	35	7	8	56	GPIO1_D0	UART6_TX_M2	SPI1_MISO_M2
			GND		9	10	57	GPIO1_D1	UART6_RX_M2	SPI1_MOSI_M2
PWM0_M1(fd8b0000)	UART4_TX_M0	SPI1_CLK_M2	GPIO1_D2	58	11	12	34	GPIO1_A2	I2C4_SDA_M3	
	UART4_RX_M0	SPI1_CS0_M2	GPIO1_D3	59	13	14		GND		
			GPIO1_B0	40	15	16	36	GPIO1_A4		
			3.3V		17	18	38	GPIO1_A6		
		SPI0_MOSI_M2	GPIO1_B2	42	19	20		GND		
		SPI0_MISO_M2	GPIO1_B1	41	21	22	39	GPIO1_A7	PWM3_M3(fd8b0030)	
		SPI0_CLK_M2	GPIO1_B3	43	23	24	44	GPIO1_B4	SPI0_CS0_M2	UART7_RX_M2
			GND		25	26	45	GPIO1_B5	SPI0_CS1_M2	UART7_TX_M2

2) 四组 UART 总线在 26pin 中对应的引脚如下表所示：

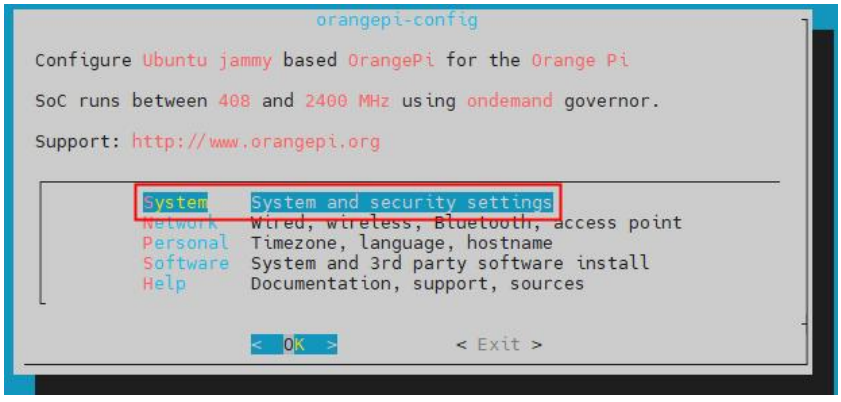
UART 总线	RX 对应 26pin	TX 对应 26pin	dtbo 对应配置
UART1_M1	3 号引脚	5 号引脚	uart1-m1
UART4_M0	13 号引脚	11 号引脚	uart4-m0
UART6_M1	10 号引脚	8 号引脚	uart6-m2
UART7_M2	24 号引脚	26 号引脚	uart7-m2

3) 在 linux 系统中，26 Pin 中的 UART 默认都是关闭的，需要手动打开才能使用。详细步骤如下所示：

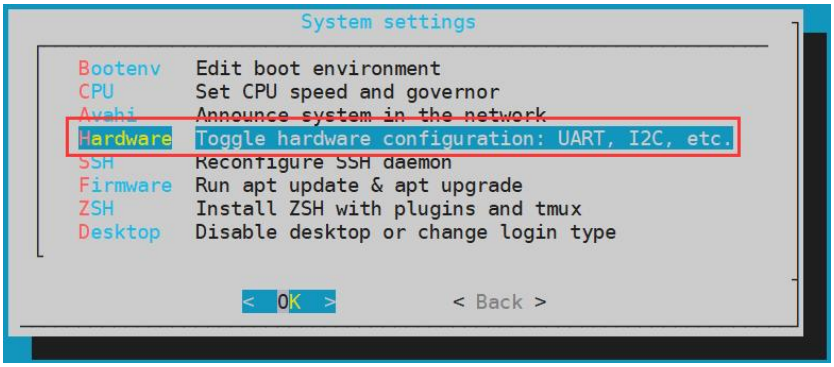
a. 首先运行下 **orangepi-config**，普通用户记得加 **sudo** 权限。

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

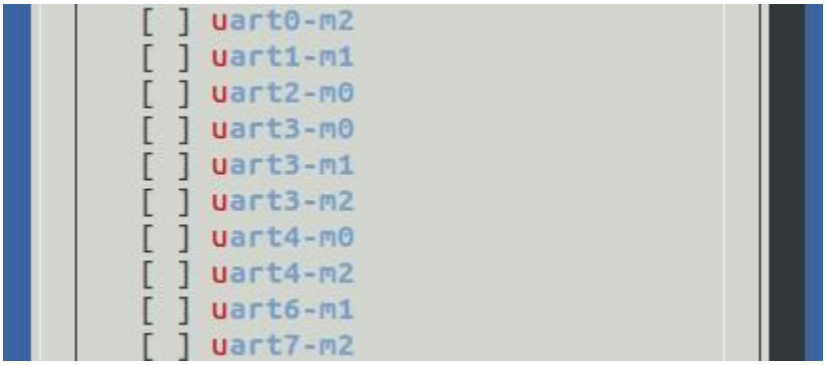
b. 然后选择 **System**。



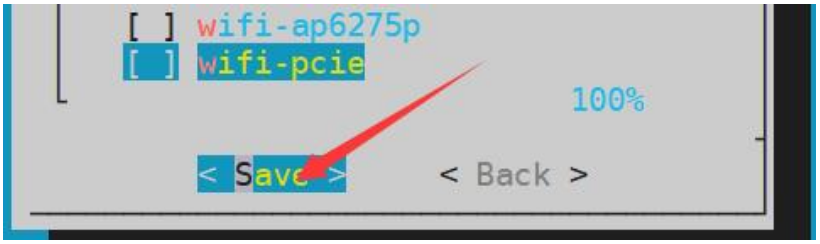
c. 然后选择 **Hardware**。



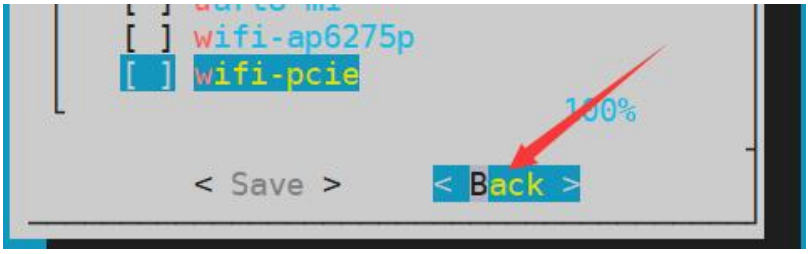
d. 然后使用键盘的方向键定位到下图所示的位置，再使用**空格**选中想要打开的 UART 的配置。



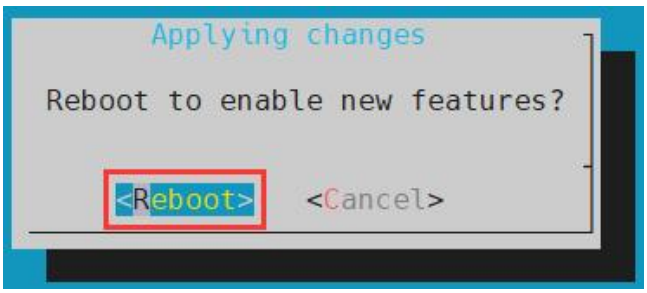
e. 然后选择**<Save>**保存。



f. 然后选择**<Back>**。



g. 然后选择**<Reboot>**重启系统使配置生效。



4) 进入 linux 系统后，先确认下/dev 下是否存在对应 uart 的设备节点。

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

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

6) 使用 examples 中的 **serialTest.py** 程序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。

/dev/ttySX 需要替换是具体的 uart 设备节点的序号。

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

Out:    0: ->    0
Out:    1: ->    1
Out:    2: ->    2
Out:    3: ->    3
Out:    4:^C
exit
```

3.20.5. 26pin SPI 测试

1) 由下图可知，Orange Pi CM5 Base Tablet 可用的 spi 为 spi1 和 spi4。

复用功能	复用功能	复用功能	GPIO	GPIO序号	引脚序号	引脚序号	GPIO序号	GPIO	复用功能	复用功能
			3.3V		1	2		5V		
PWM13_M2(febf0010)	UART1_RX_M1	I2C5_SDA_M3	GPIO1_B7	47	3	4		5V		
	UART1_TX_M1	I2C5_SCL_M3	GPIO1_B6	46	5	6		GND		
	PWM1_M2(fd8b0010)	I2C4_SCL_M3	GPIO1_A3	35	7	8	56	GPIO1_D0	UART6_TX_M2	SPI1_MISO_M2
			GND		9	10	57	GPIO1_D1	UART6_RX_M2	SPI1_MOSI_M2
PWM0_M1(fd8b0000)	UART4_TX_M0	SPI1_CLK_M2	GPIO1_D2	58	11	12	34	GPIO1_A2	I2C4_SDA_M3	
	UART4_RX_M0	SPI1_CS0_M2	GPIO1_D3	59	13	14		GND		
			GPIO1_B0	40	15	16	36	GPIO1_A4		
			3.3V		17	18	38	GPIO1_A6		
		SPI0_MOSI_M2	GPIO1_B2	42	19	20		GND		
		SPI0_MISO_M2	GPIO1_B1	41	21	22	39	GPIO1_A7	PWM3_M3(fd8b0030)	
		SPI0_CLK_M2	GPIO1_B3	43	23	24	44	GPIO1_B4	SPI0_CS0_M2	UART7_RX_M2
			GND		25	26	45	GPIO1_B5	SPI0_CS1_M2	UART7_TX_M2

2) SPI0 和 SPI4 在 26pin 中对应的引脚如下表所示。

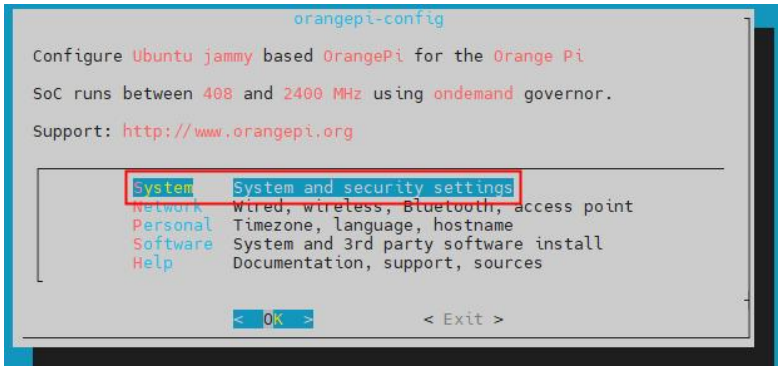
	SPI1_M2 对应 26pin	SPI4_M2 对应 26pin
MOSI	10 号引脚	19 号引脚
MISO	8 号引脚	21 号引脚
CLK	11 号引脚	23 号引脚
CS0	13 号引脚	24 号引脚
CS1	无	26 号引脚
dtbo 配置	spi1-m2-cs0-spidev	spi4-m2-cs0-spidev spi4-m2-cs1-spidev spi4-m2-cs0-cs1-spidev

3) 在 linux 系统中，26 Pin 中的 SPI 默认都是关闭的，需要手动打开才能使用。详细步骤如下所示：

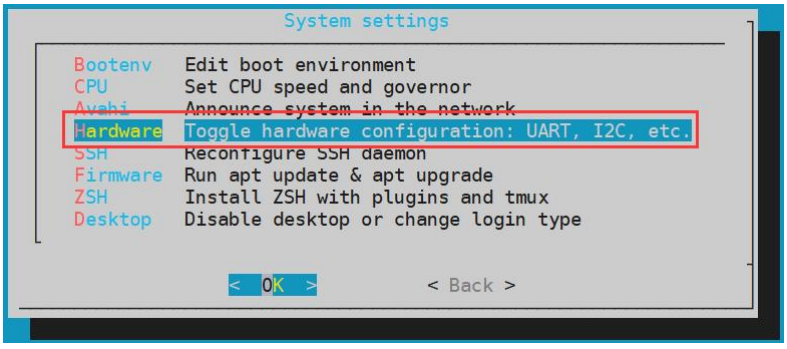
a) 首先运行下 **orangepi-config**，普通用户记得加 **sudo** 权限。

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

b) 然后选择 **System**。

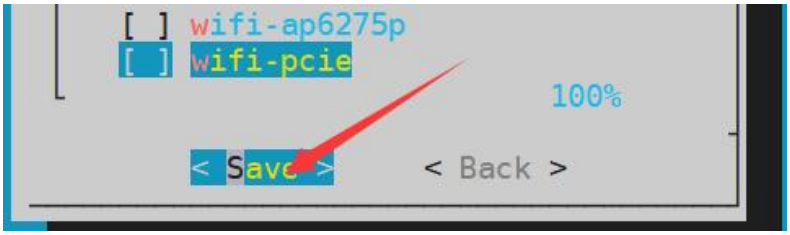


c) 然后选择 **Hardware**。

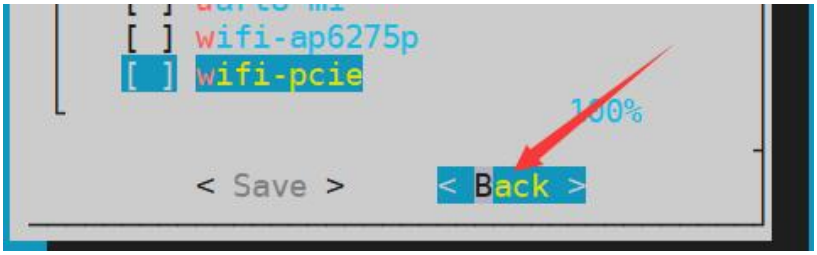


d) 然后使用键盘的方向键定位到下图所示的位置，再使用**空格**选中想要打开的 SPI 的配置。

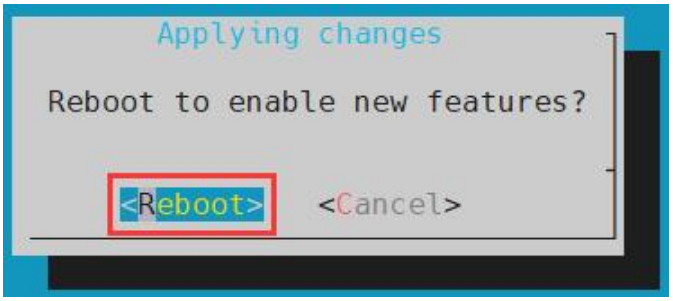
e) 然后选择**<Save>**保存。



f) 然后选择**<Back>**。



g) 然后选择**<Reboot>**重启系统使配置生效。



4) 重启后进入系统先查看下 linux 系统中是否存在 **spidevx.x** 的设备节点，如果存在，说明 SPI 已经设置好了，可以直接使用。

```
orangepi@orangepi:~$ ls /dev/spidev*
/dev/spidev1.0   /dev/spidev4.0   /dev/spidev4.1
```

上面是打开 spi1-m2-cs0-spidev 和 spi4-m2-cs0-cs1-spidev 后显示的结果。

5) 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。

6) 先不短接 SPI 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。

--channel 和--port 参数后面的 x 需要替换为具体 SPI 的通道号和 SPI 的端口号。

```
root@orangepi:~/wiringOP-Python# cd examples
root@orangepi:~/wiringOP-Python/examples# python3 spidev_test.py --channel x --port x
spi mode: 0x0
max speed: 500000 Hz (500 KHz)
Opening device /dev/spidev0.0
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    |................|
```

7) 然后使用杜邦线短接 SPI 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 SPI 回环测试正常

--channel 和--port 参数后面的 x 需要替换为具体 SPI 的通道号和 SPI 的端口号。

```
root@orangepi:~/wiringOP-Python# cd examples
root@orangepi:~/wiringOP-Python/examples# python3 spidev_test.py --channel x --port x
spi mode: 0x0
max speed: 500000 Hz (500 KHz)
Opening device /dev/spidev0.0
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. 硬件看门狗测试

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

运行 watchdog_test 程序的方法如下所示：

a. 第二个参数 10 表示看门狗的计数时间，如果这个时间内没有喂狗，系统会重启。

b. 我们可以通过按下键盘上的任意键（ESC 除外）来喂狗，喂狗后，程序会打印一行 keep alive 表示喂狗成功。

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

3.22. 查看 RK3588S 芯片的序列号

查看 RK3588S 芯片序列号的命令如下所示，每个芯片的序列号都是不同的，所以可以使用序列号来区分多个开发板。

```
orangepi@orangepi:~$ cat_serial.sh
Serial          : 1404a7682e86830c
```

3.23. 安装 Docker 的方法

1) Orange Pi 提供的 linux 镜像已经预装了 Docker，只是 Docker 服务默认没有打开。

2) 使用 **enable_docker.sh** 脚本可以使能 docker 服务，然后就可以开始使用 docker

命令了，并且在下次启动系统时也会自动启动 docker 服务。

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

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

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

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

3. 24. 下载安装 arm64 版本 balenaEtcher 的方法

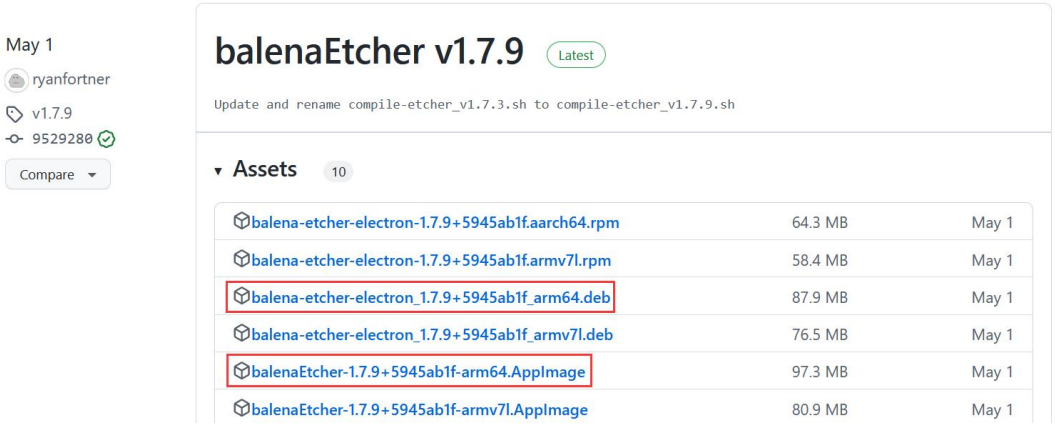
1) balenaEtcher arm64 版本的下载地址为：

a. deb 安装包的下载地址如下所示，需要安装才能使用。

```
https://github.com/Itai-Nelken/BalenaEtcher-arm/releases/download/v1.7.9/balena-etcher-electron_1.7.9+5945ab1f_arm64.deb
```

b. 无需安装的 AppImage 版本的下载地址如下所示：

```
https://github.com/Itai-Nelken/BalenaEtcher-arm/releases/download/v1.7.9/balenaEtcher-1.7.9+5945ab1f-arm64.AppImage
```

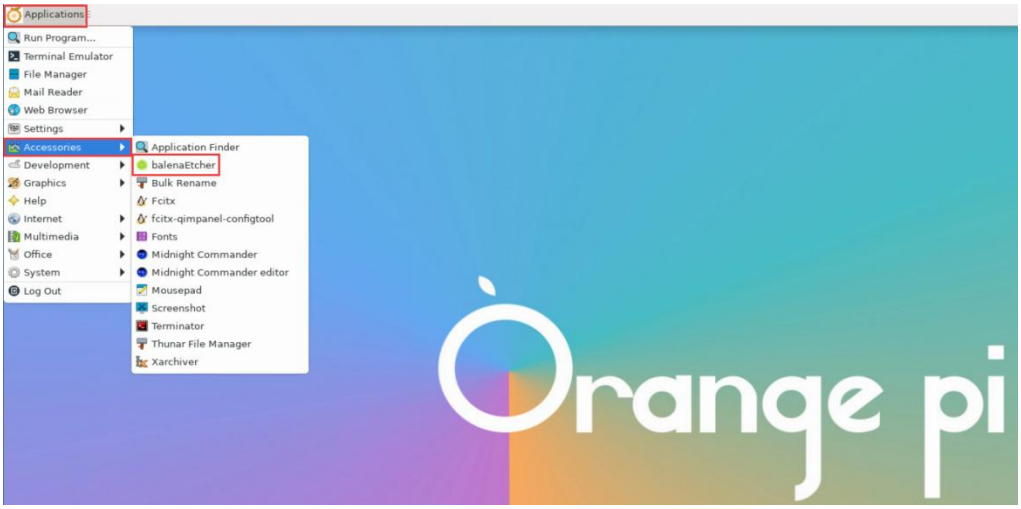


2) deb 版本 balenaEtcher 的安装使用方法：

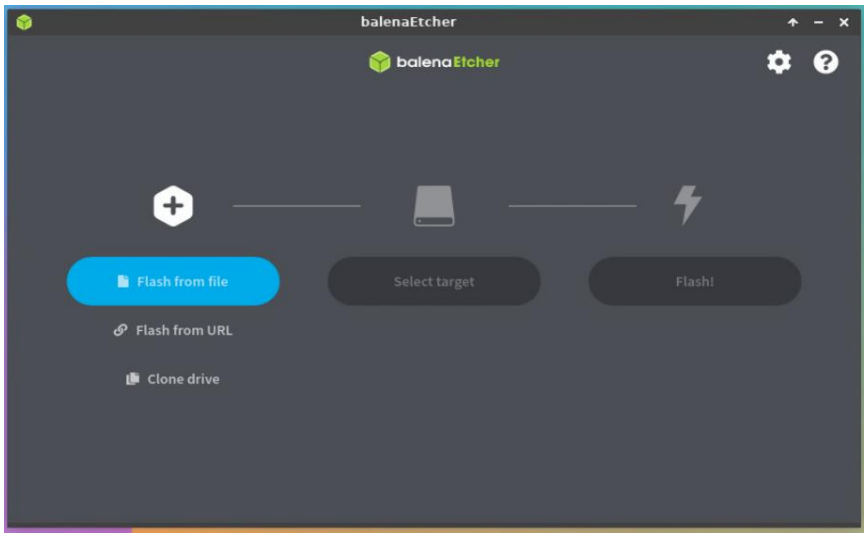
a. deb 版本的 balenaEtcher 安装命令如下所示：

```
orangepi@orangepi:~$ sudo apt install -y  \
--fix-broken ./balena-etcher-electron_1.7.9+5945ab1f_arm64.deb
```

b. deb 版本的 balenaEtcher 安装完成后，在 Application 中就可以打开了。



c. balenaEtcher 打开后的界面如下所示：

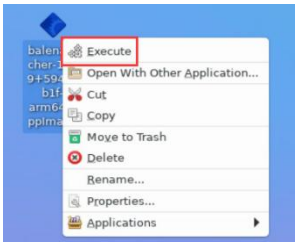


3) AppImage 版本的 balenaEtcher 的使用方法：

a. 首先给 balenaEtcher 添加权限。

```
orangepi@orangepi:~/Desktop$ chmod +x balenaEtcher-1.7.9+5945ab1f-arm64.AppImage
```

b. 然后选中 AppImage 版本 balenaEtcher，再点击鼠标右键，再点击 Execute 就可以打开 balenaEtcher 了。



3.25. 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：

```
Debian11 > Ubuntu 22.04 > Debian12
```

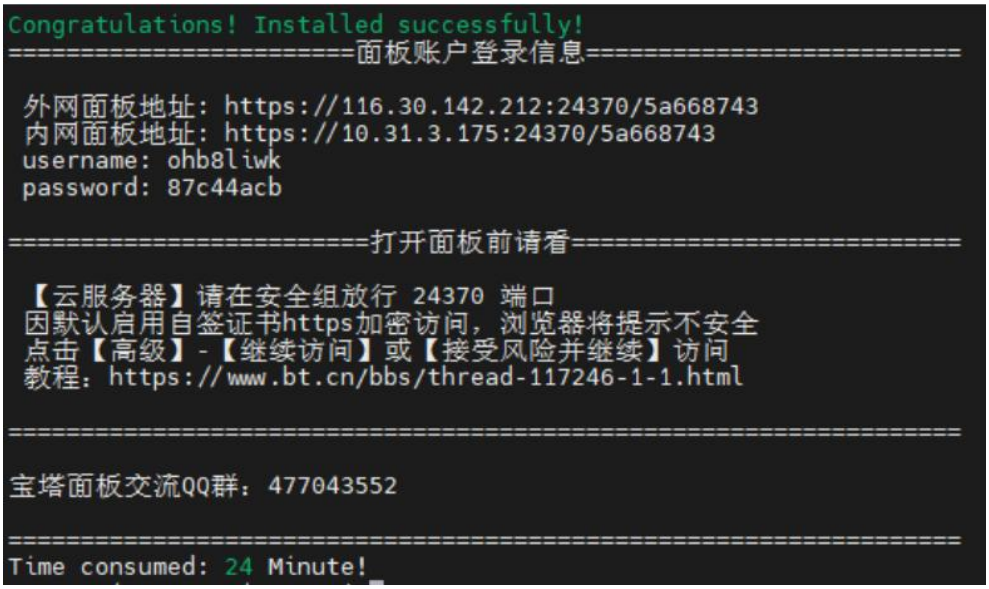
2) 然后在 linux 系统中输入下面的命令就可以开始宝塔的安装。

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

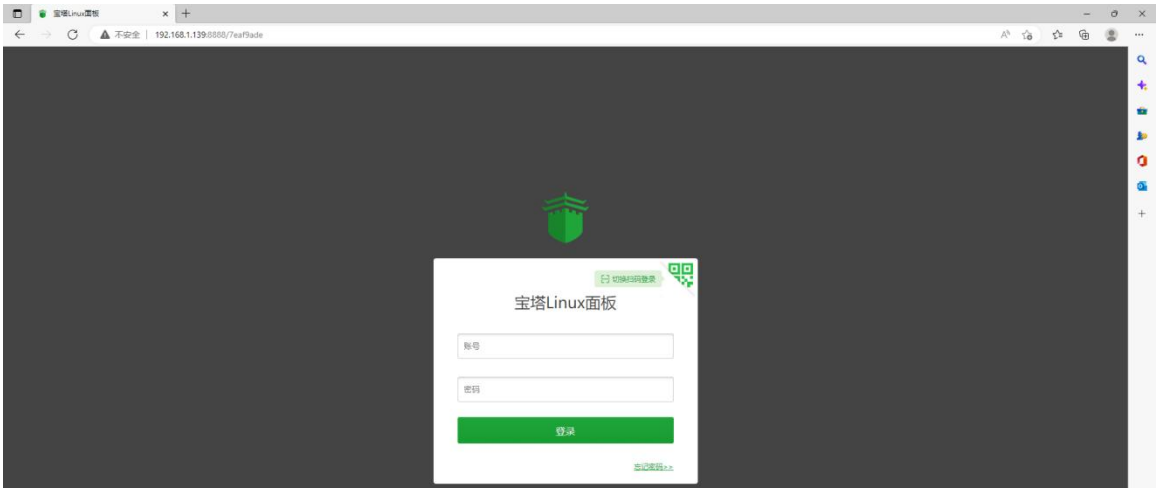
3) 然后宝塔安装程序会提醒是否安装 **Bt-Panel** 到**/www** 文件夹，此时输入 **y** 即可。

```
+----------------------------------------------------------------------
| 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
```

4) 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 12 分钟，根据网络速度的不同可能会有一些差别。

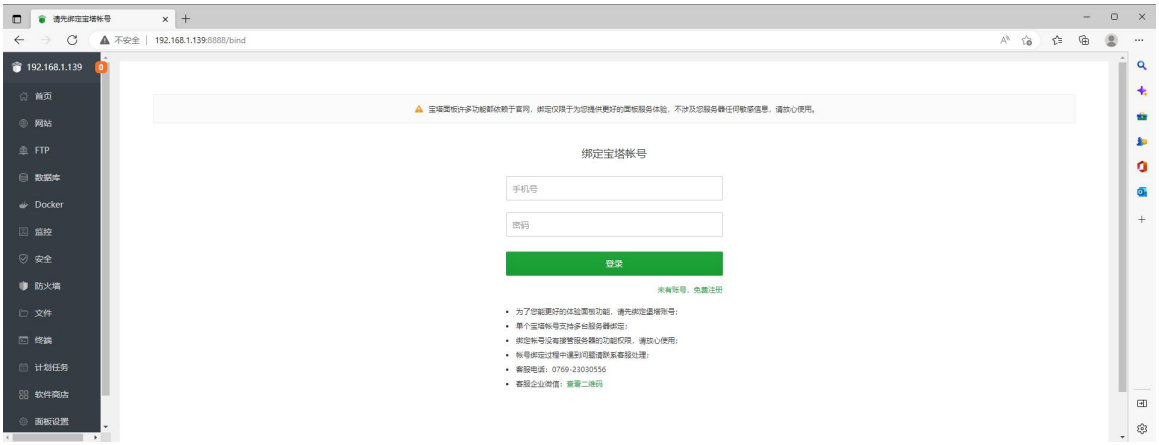


5) 此时在浏览器中输入上面显示的**面板地址**就可以打开宝塔 Linux 面板的登录界面，然后在对应的位置输入上图显示的 **username** 和 **password** 就可以登录进宝塔。

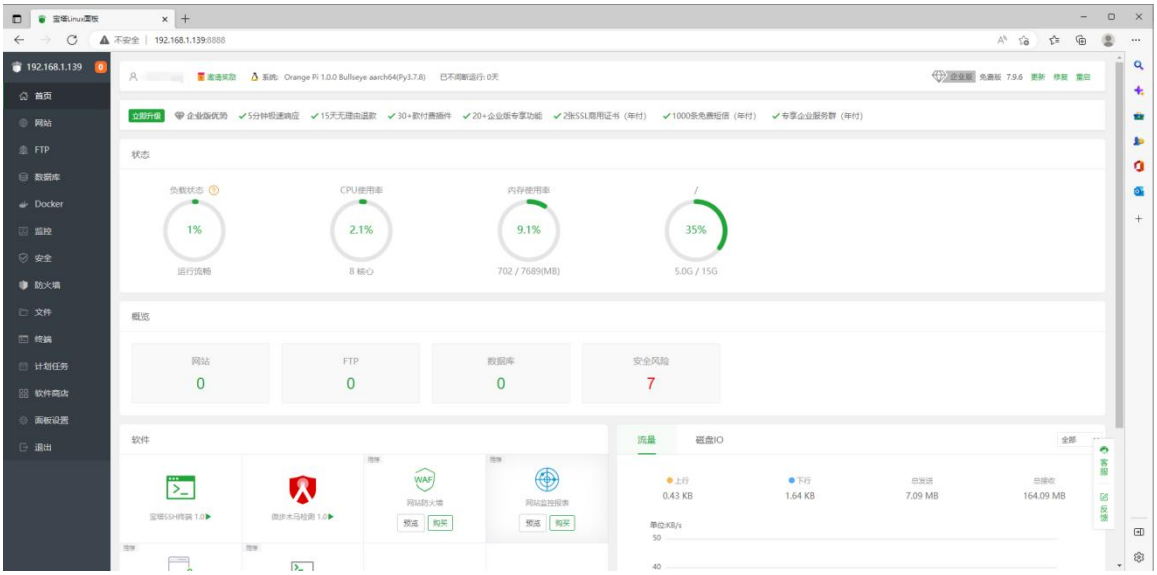


6) 成功登录宝塔后的会弹出下面的欢迎界面，首先请将中间的用户须知阅读完拖到最下面，然后就可以选择“我已同意并阅读《用户协议》”，接着点击“进入面板”就可以进入宝塔了。

7) 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。

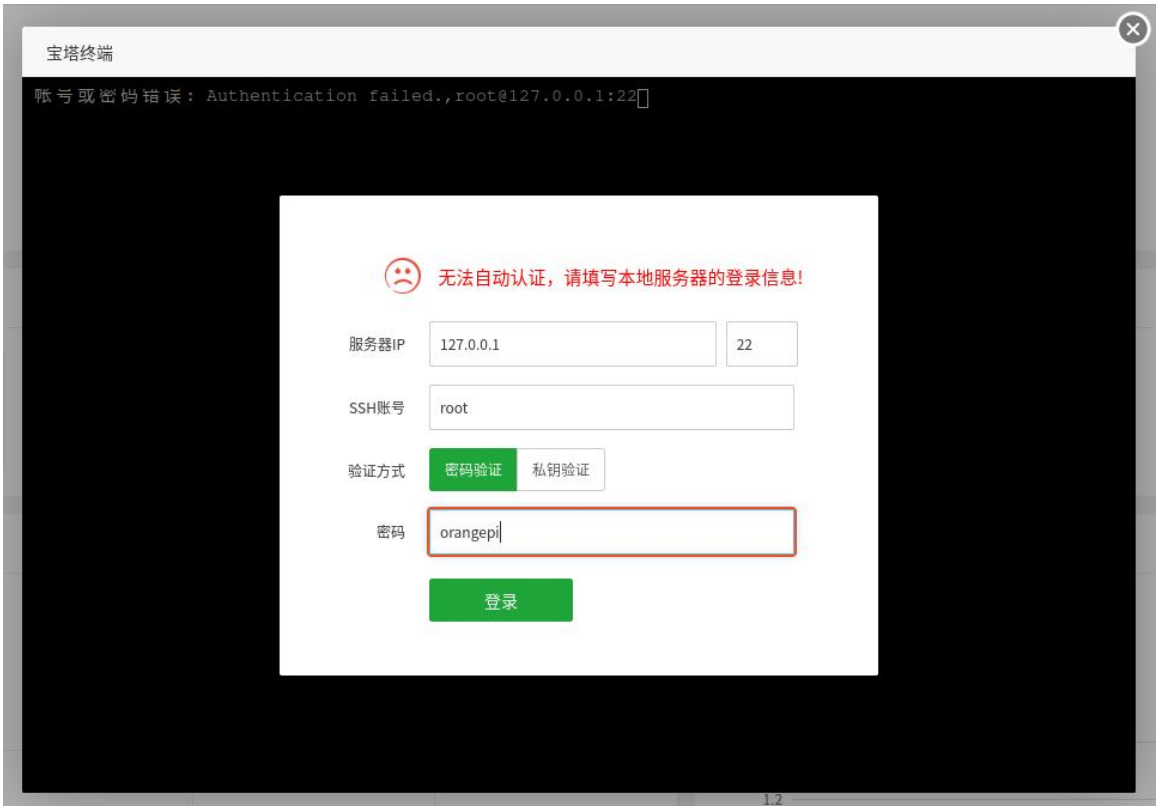


8) 最终显示的界面如下图所示，可以很直观的看到开发板 Linux 系统的一些状态信息，比如负载状态、CPU 的使用率、内存使用率和存储空间的使用情况等。

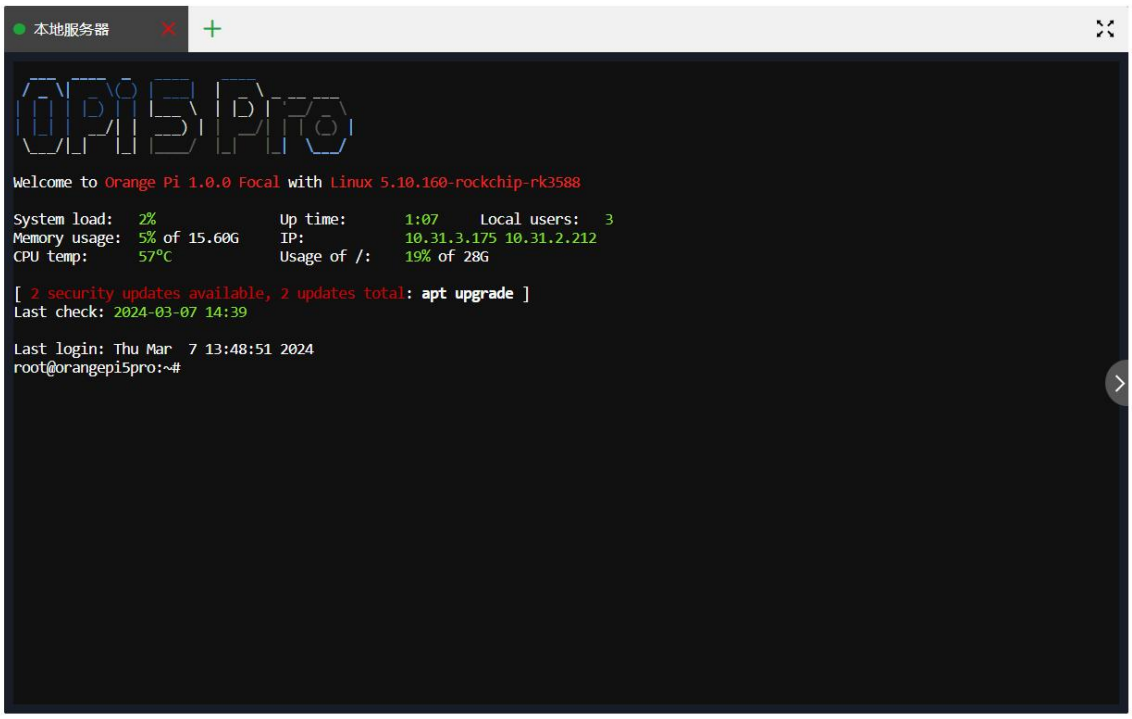


9) 测试宝塔的 SSH 终端登录。

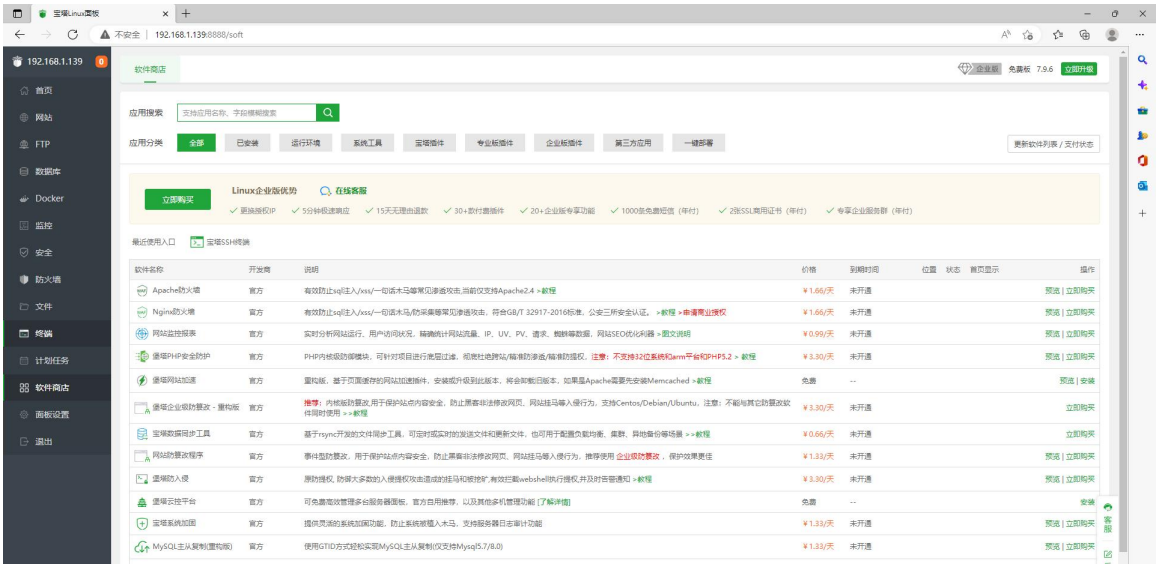
a. 打开宝塔的 SSH 终端后首先会提示需要输入开发板系统的密码，此时在密码框中输入 **orangepi**（默认密码，如果有修改请填写修改后的）即可。



b. 成功登录后的显示如下图所示：



10) 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。



11) 宝塔命令行工具测试。

```
orangepi@orangepi5pro:~$ sudo 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) 设置是否开启IP + User-Agent验证
(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) 关闭动态口令认证               (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
(26) 关闭面板ssl                    (19) 关闭面板登录地区限制
(28) 修改面板安全入口               (29) 取消访问设备验证
(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: https://116.30.142.212:24370/5a668743
内网面板地址: https://10.31.3.175:24370/5a668743
username: ohb8liwk
password: ********
Warning:
If you cannot access the panel,
release the following port (8888|888|80|443|20|21) in the security group
注意：初始密码仅在首次登录面板前能正确获取，其它时间请通过 bt 5 命令修改密码
==================================================================
orangepi@orangepi5pro:~$
```

12) 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**

3.26. 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。

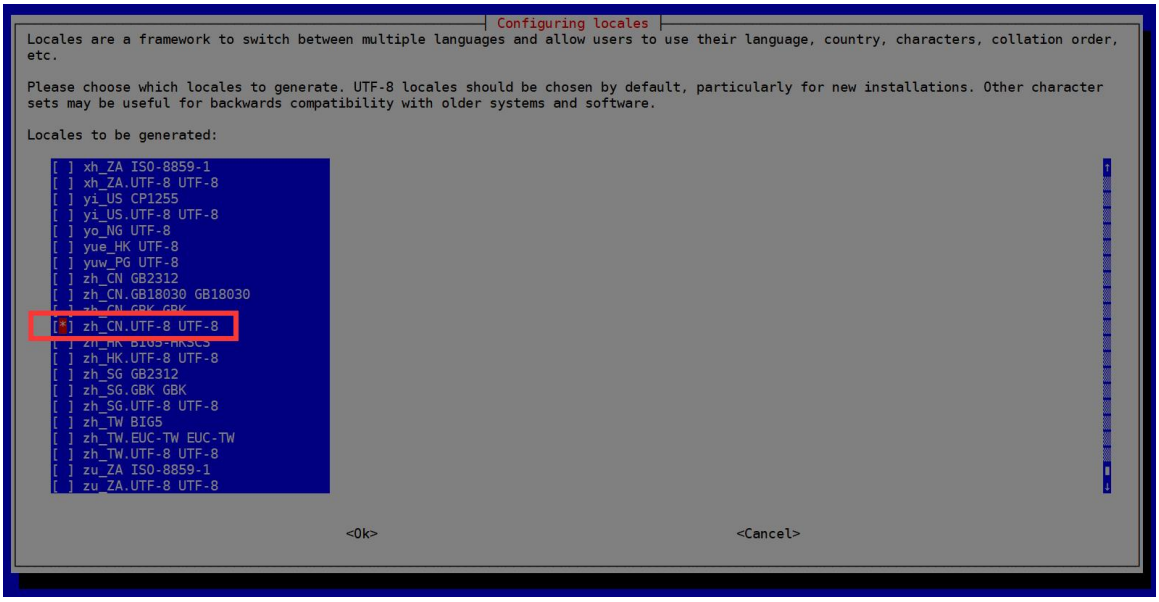
3.26.1. Debian 系统的安装方法

1) 首先设置默认 **locale** 为中文。

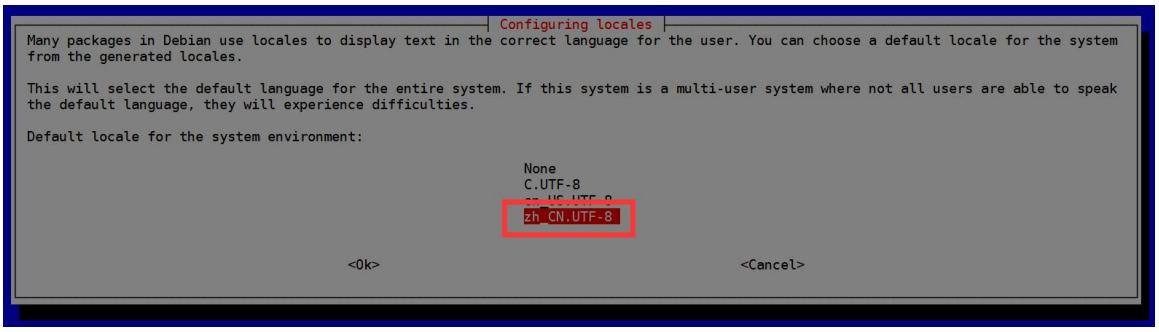
a. 输入下面的命令可以开始配置 **locale**。

```
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）。



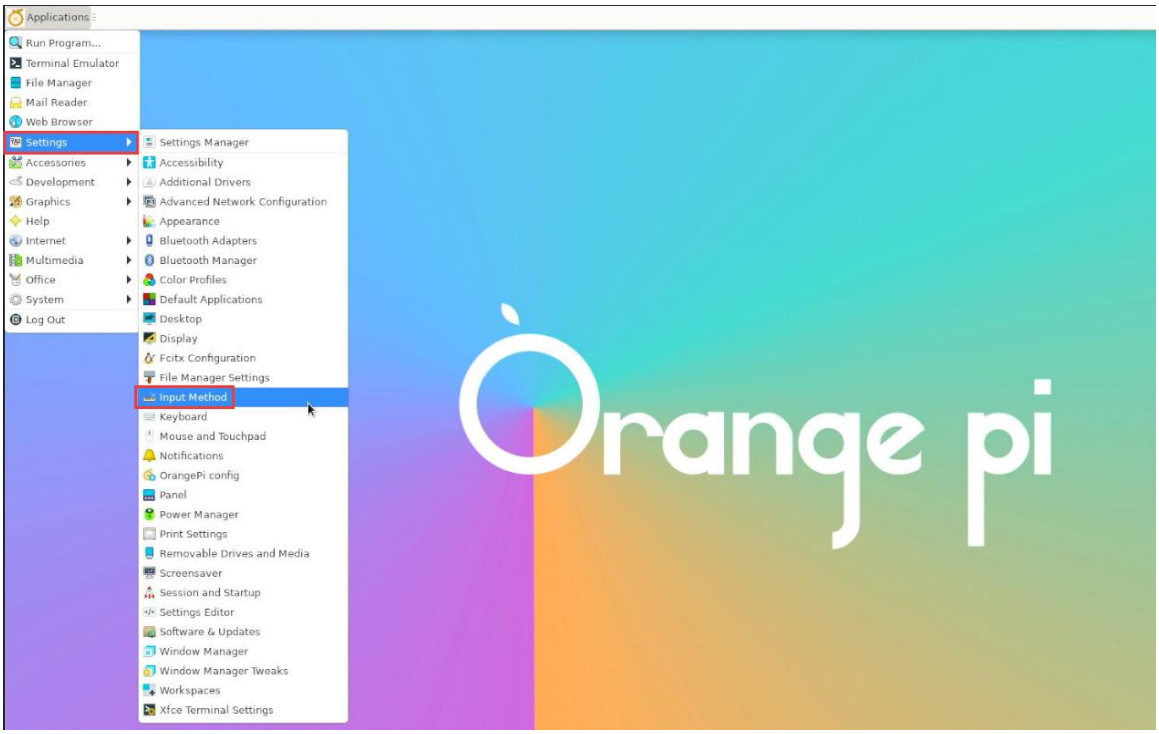
c. 然后设置默认 **locale** 为 **zh_CN.UTF-8**。



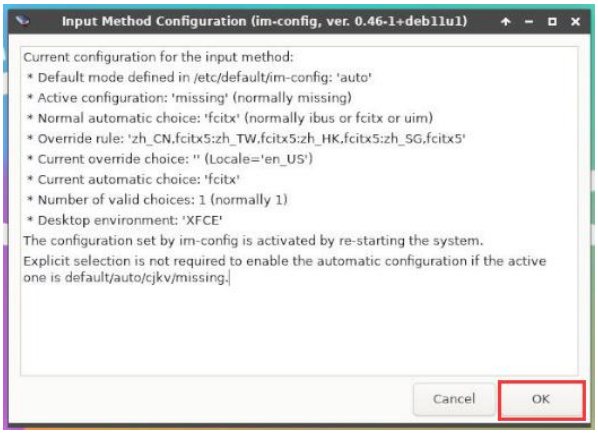
d. 退出界面后就会开始 **locale** 的设置，命令行显示的输出如下所示：

```
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.
```

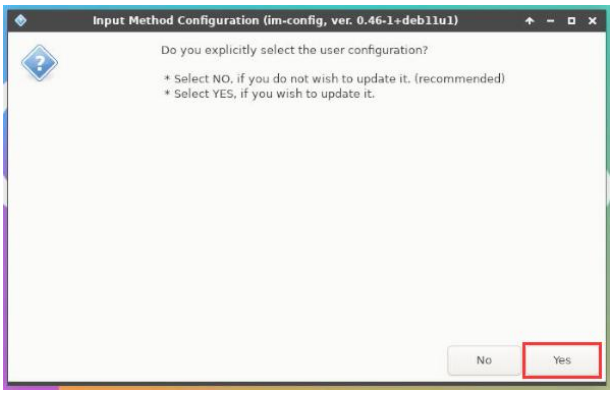
2) 然后打开 **Input Method**。



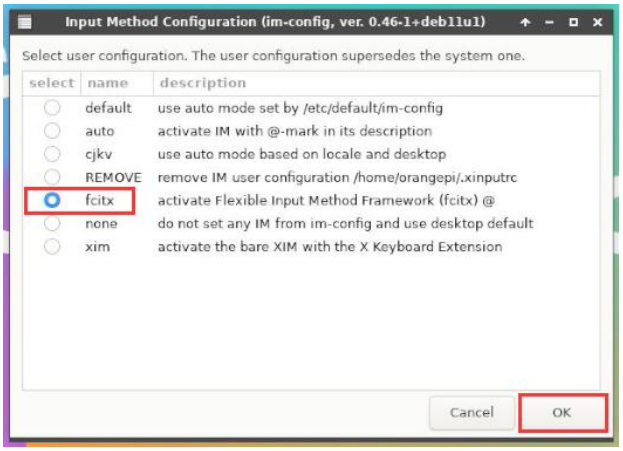
3) 然后选择 **OK**。



4) 然后选择 **Yes**。



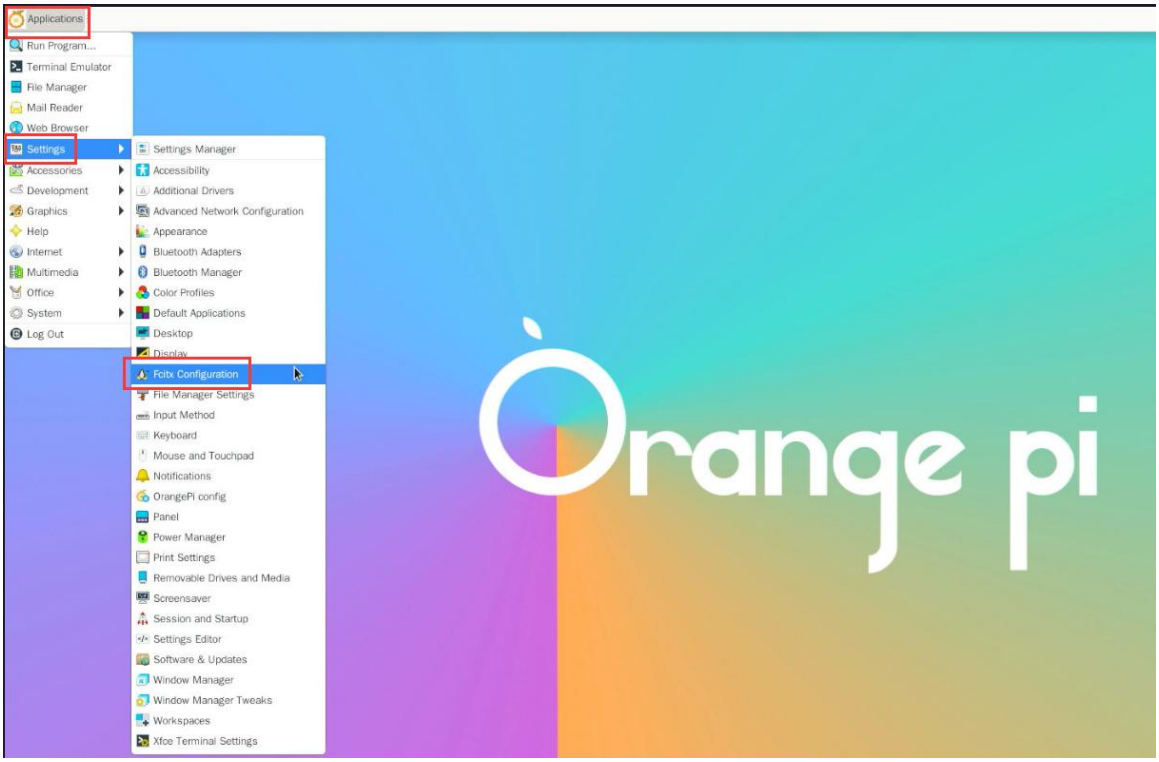
5) 然后选择 **fcitx**。



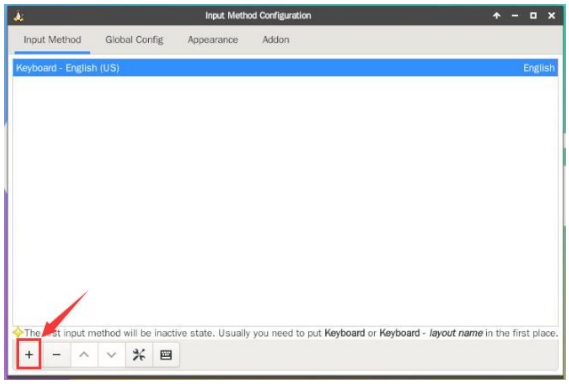
6) 然后选择 **OK**。

7) **然后重启 Linux 系统才能使配置生效**。

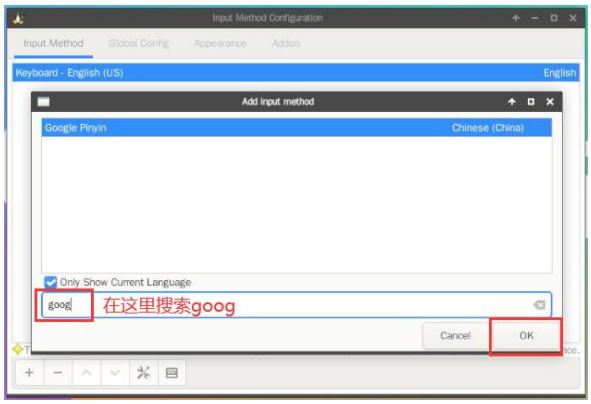
8) 然后打开 **Fcitx configuration**。



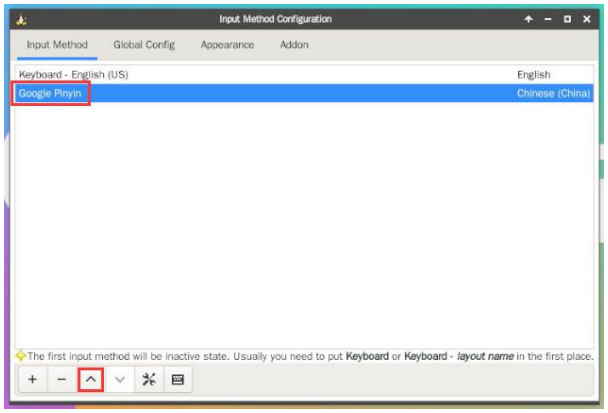
9) 然后点击下图所示位置的+号。

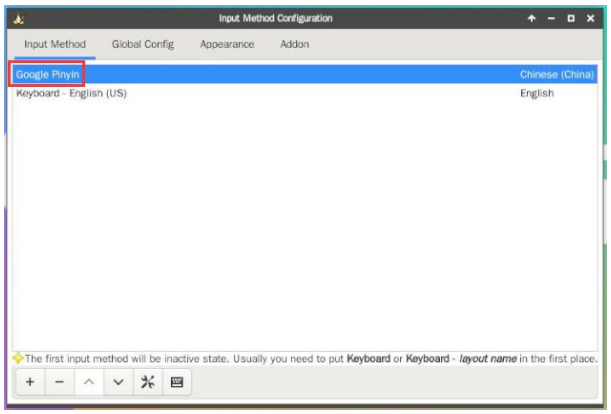


10) 然后搜索 **Google Pinyin** 再点击 **OK**。



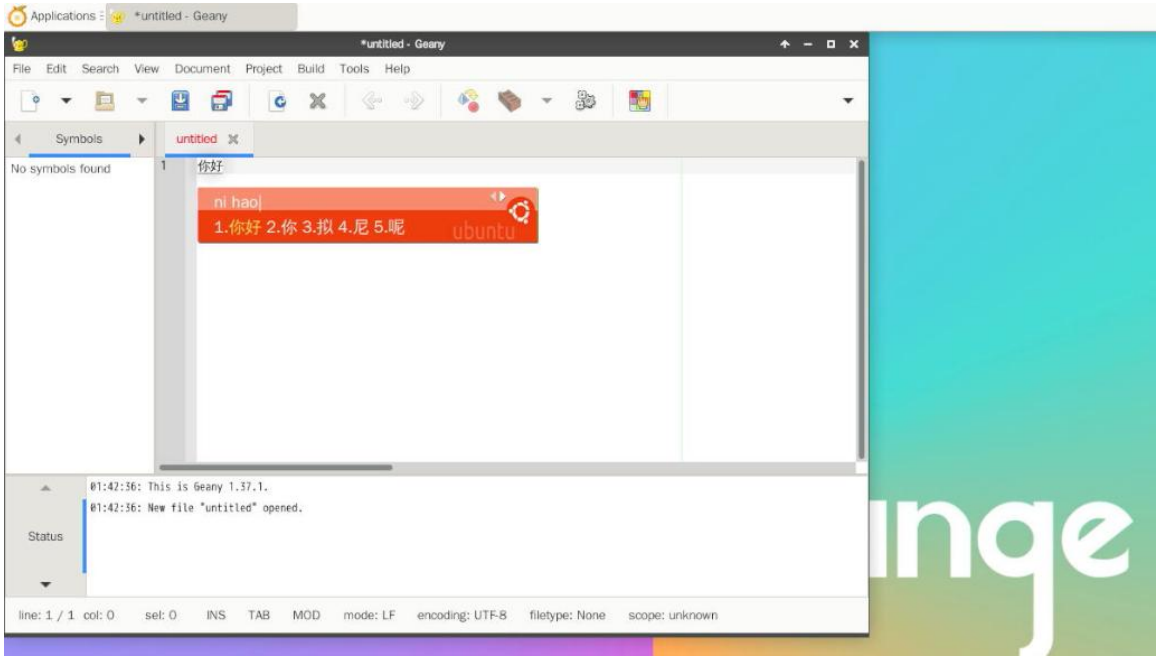
11) 然后将 **Google Pinyin** 放到最前面。





12) 然后打开 **Geany** 这个编辑器测试下中文输入法。

13) 中文输入法测试如下所示：

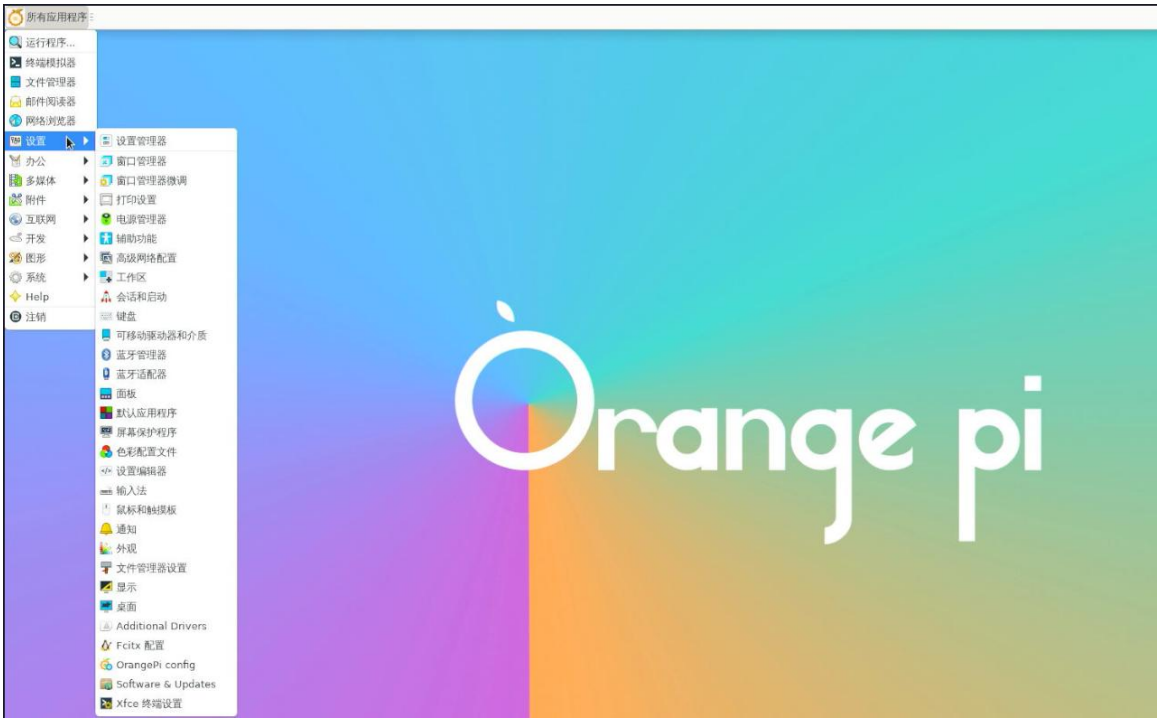


14) 通过 **Ctrl+Space** 快捷键可以切换中英文输入法。

15) 如果需要整个系统都显示为中文，可以将**/etc/default/locale** 中的变量都设置为 **zh_CN.UTF-8**。

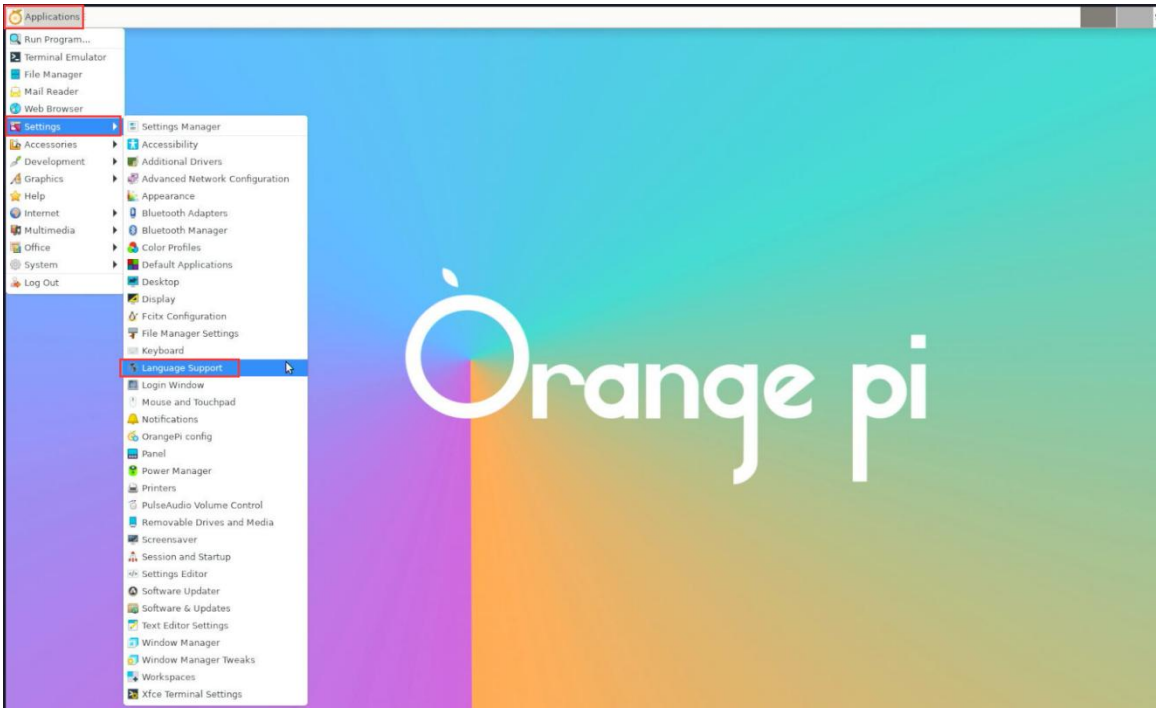
```
orangepi@orangepi:~$ sudo vim /etc/default/locale
#    File generated by update-locale
LC_MESSAGES=zh_CN.UTF-8
LANG=zh_CN.UTF-8
LANGUAGE=zh_CN.UTF-8
```

16) 然后**重启系统**就能看到系统显示为中文了。

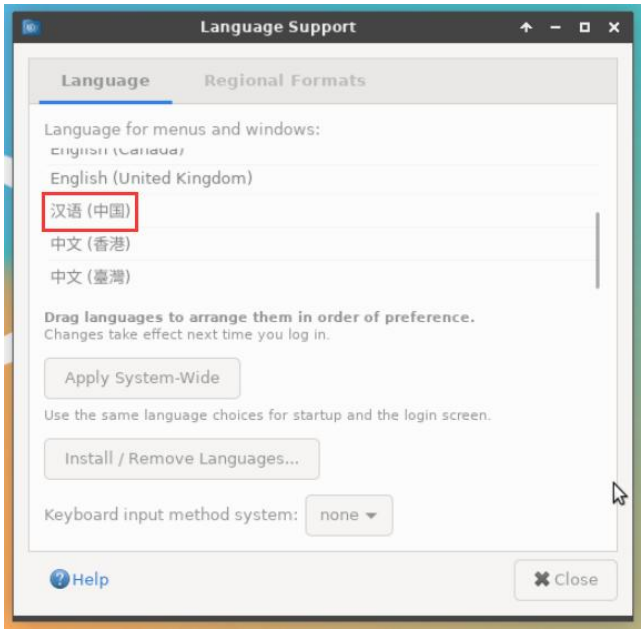


3. 26. 2. Ubuntu 20.04 系统的安装方法

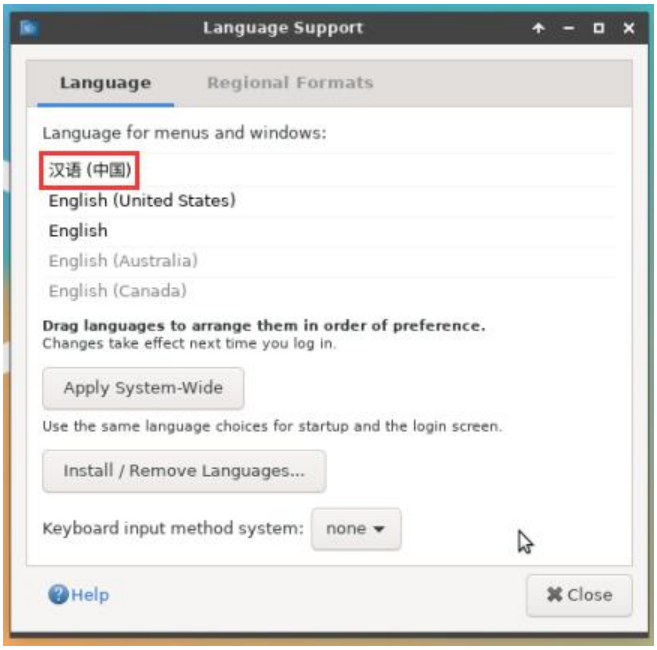
1) 首先打开 **Language Support**。



2) 然后找到**汉语（中国）**选项。

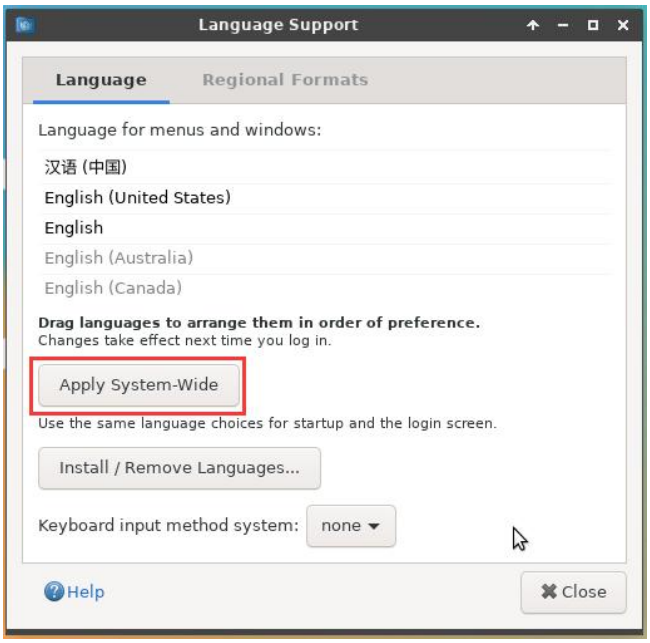


3) 然后请使用鼠标左键选中**汉语（中国）**并按住不动，然后往上将其拖到最开始的位置，拖完后的显示如下图所示：



注意，这一步不是很好拖动的，请耐心多试几次。

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

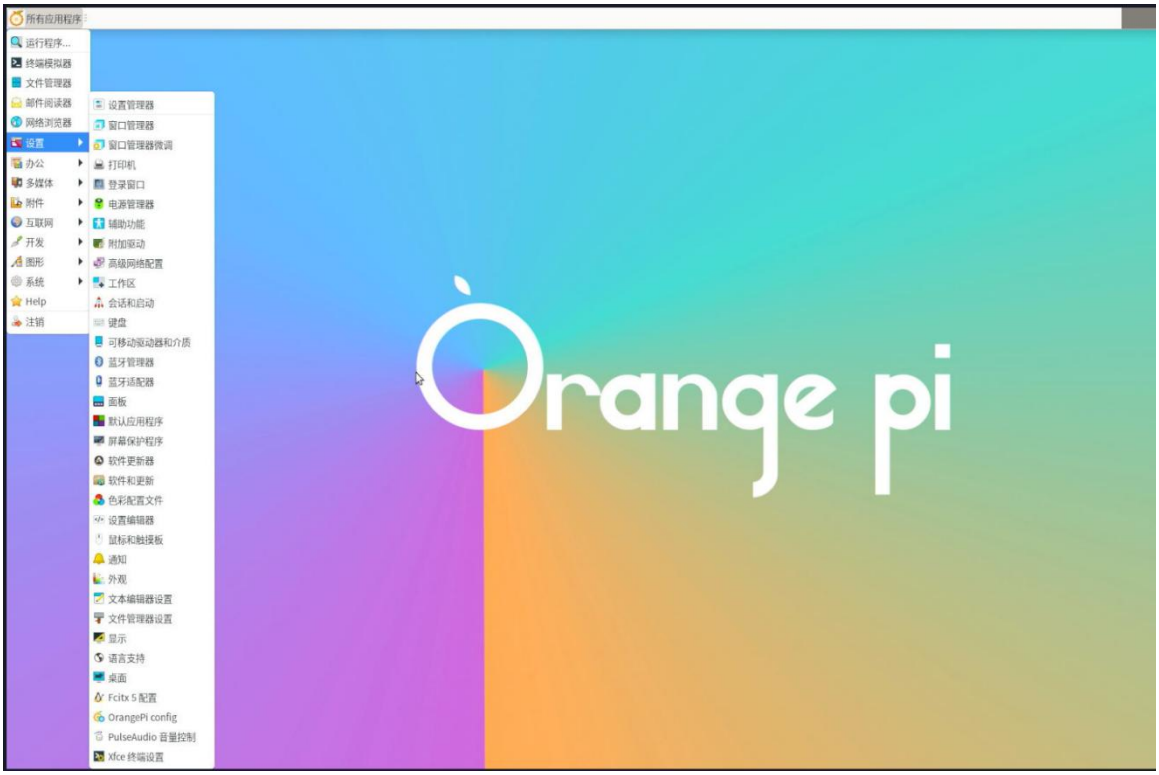


5) 然后设置 **Keyboard input method system** 为 **fcitx**。

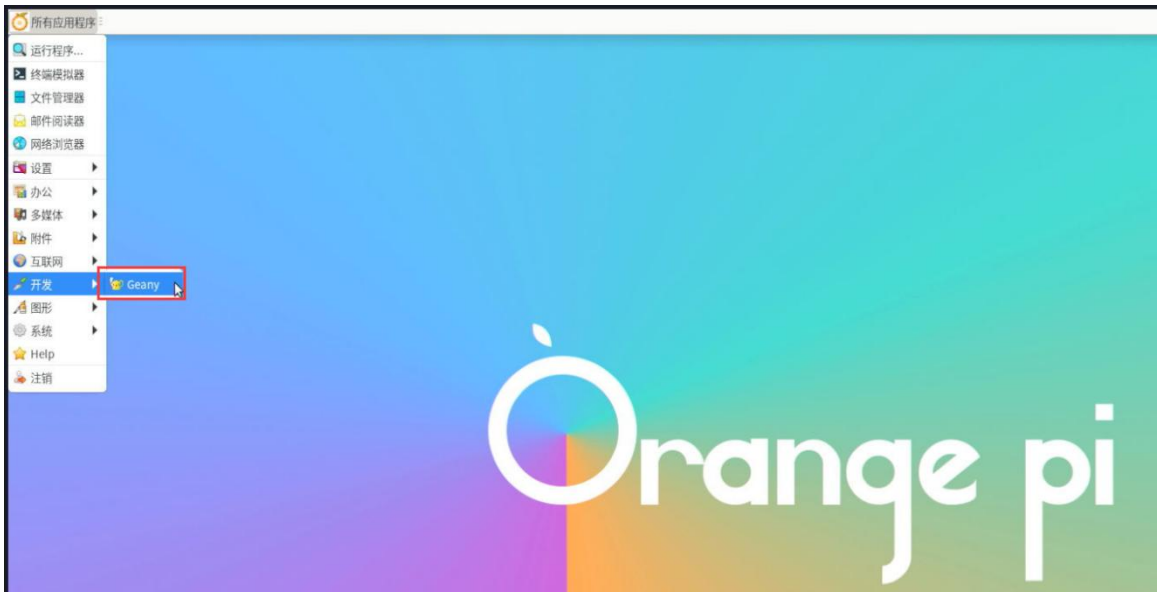
6) **然后重启 Linux 系统使配置生效**。

7) 重新进入系统后，在下面的界面请选择**不要再次询问我**，然后请根据自己的喜好决定标准文件夹是否也要更新为中文。

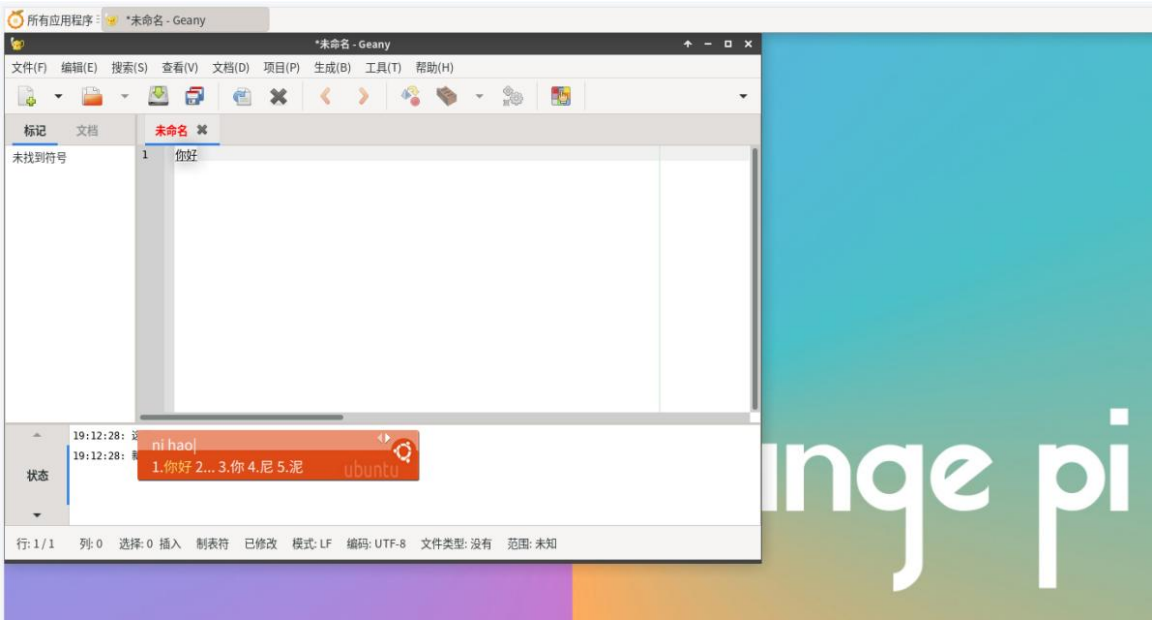
8) 然后可以看到桌面都显示为中文了。



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

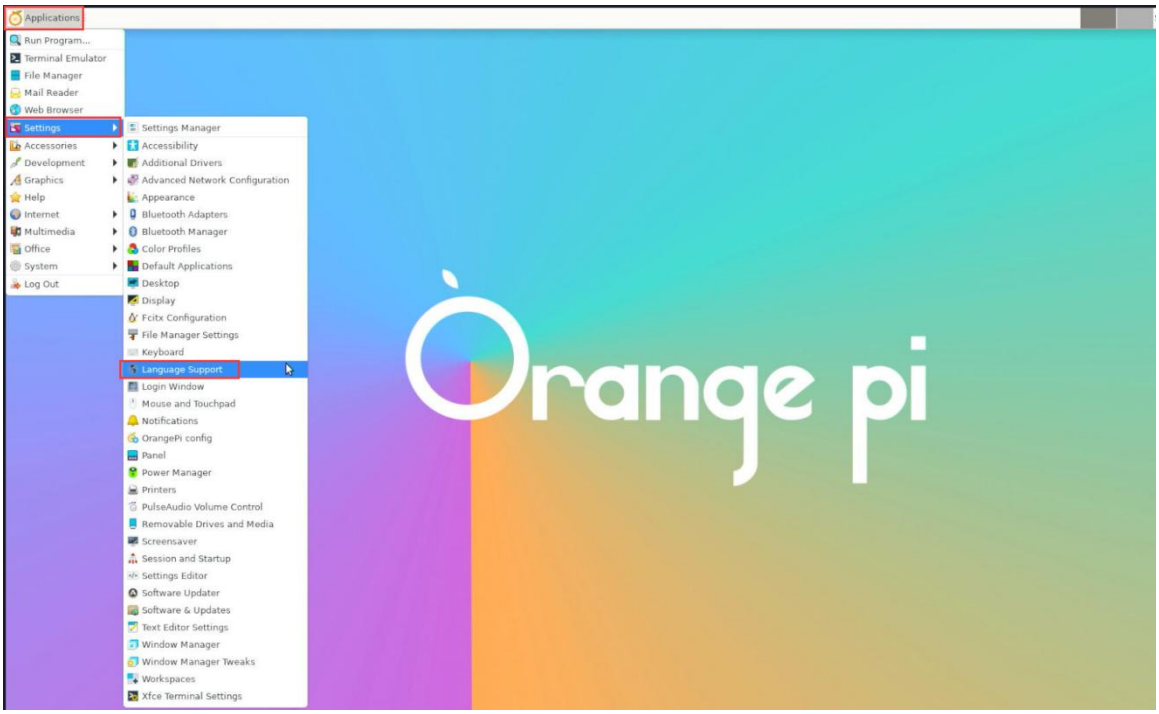


10) 打开 **Geany** 后，默认还是英文输入法，我们可以通过 **Ctrl+Space** 快捷键来切换成中文输入法，然后就能输入中文了。

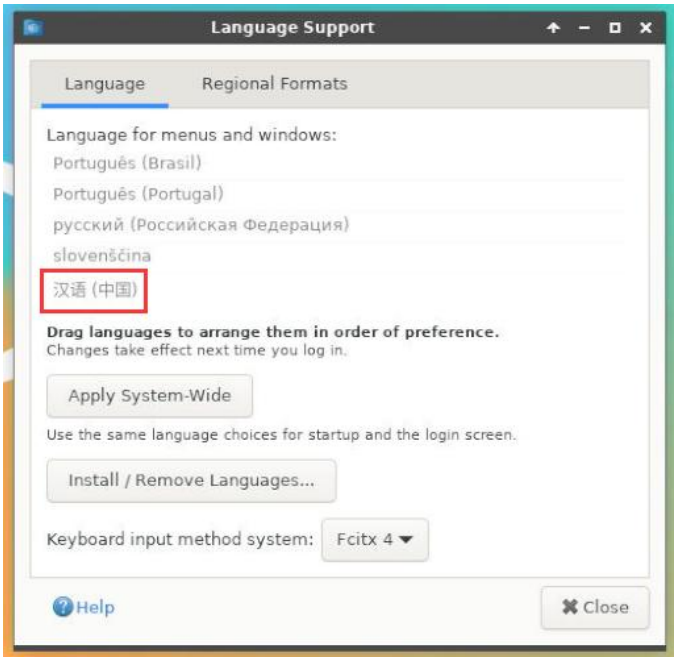


3. 26. 3. Ubuntu 22.04 系统的安装方法

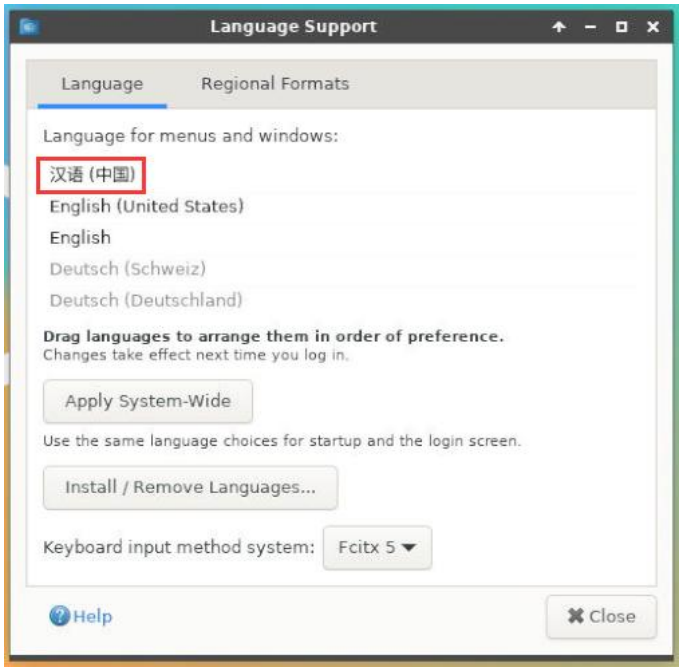
1) 首先打开 **Language Support**。



2) 然后找到**汉语（中国）**选项。

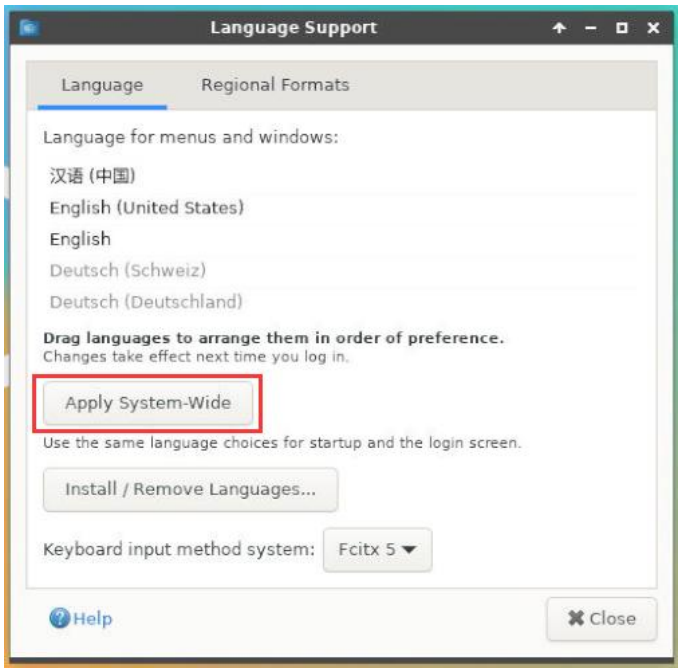


3) 然后请使用鼠标左键选中**汉语（中国）**并按住不动，然后往上将其拖到最开始的位置，拖完后的显示如下图所示：



注意，这一步不是很好拖动的，请耐心多试几次。

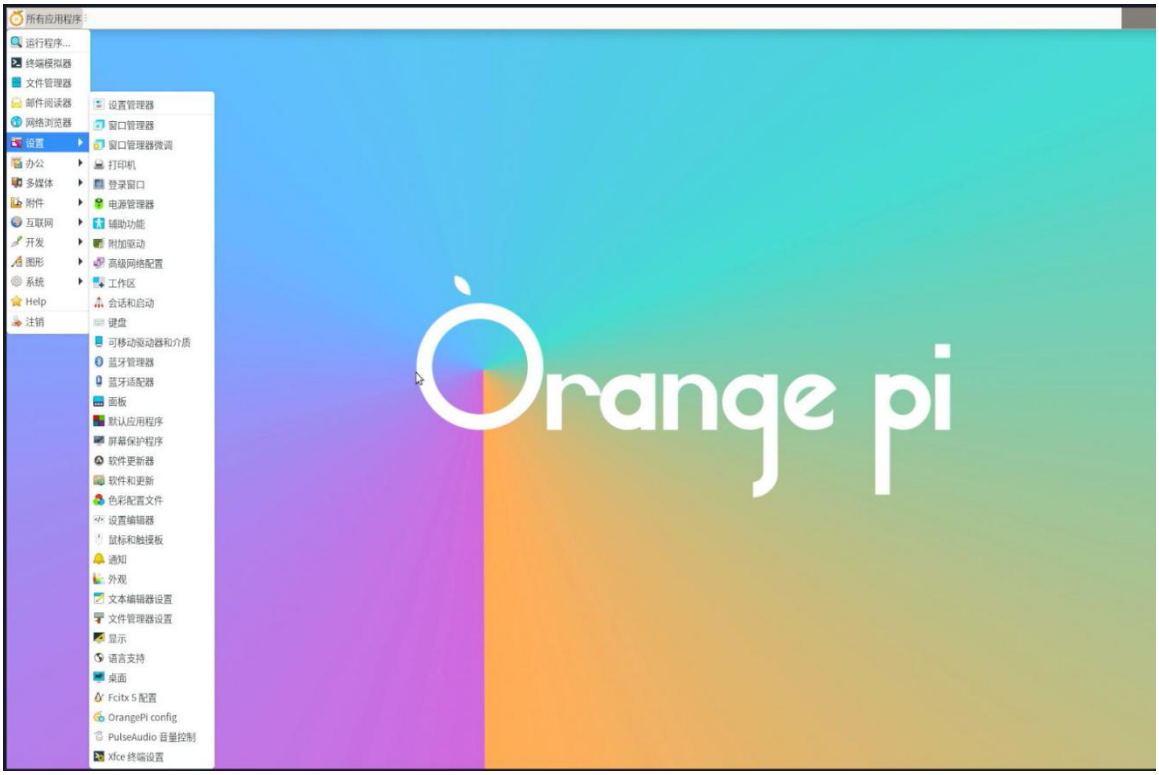
4) 然后选择 **Apply System-Wide** 将中文设置应用到整个系统。



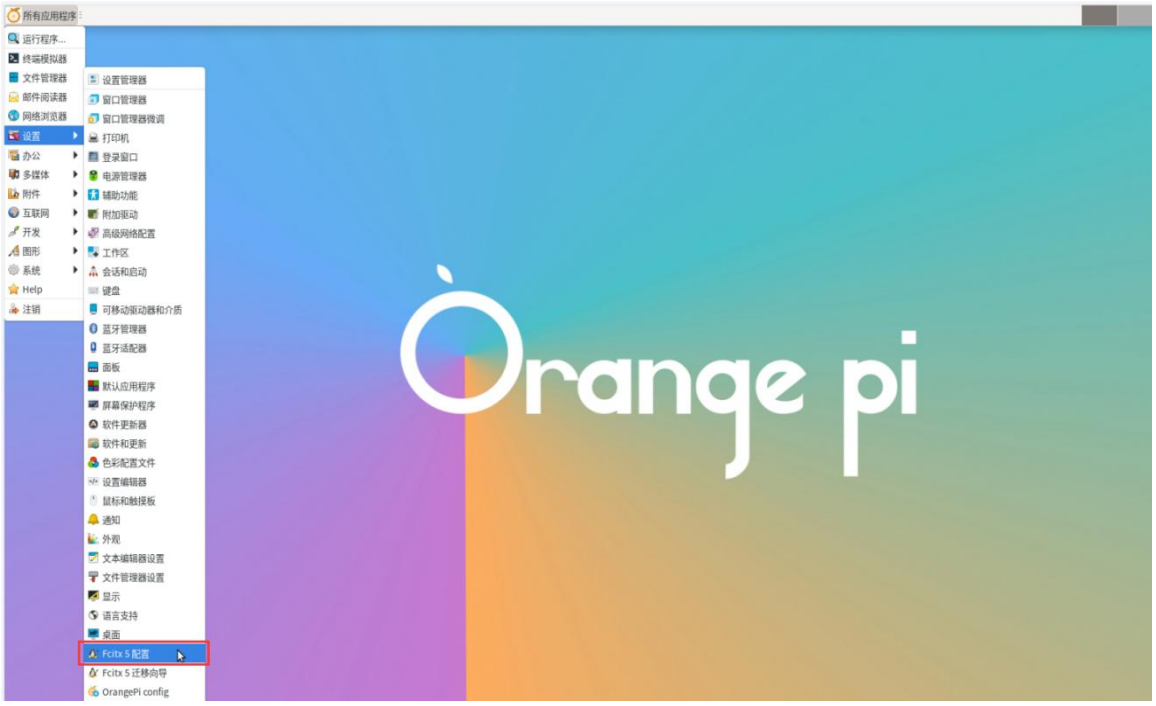
5) **然后重启 Linux 系统使配置生效**。

6) 重新进入系统后，在下面的界面请选择**不要再次询问我**，然后请根据自己的喜好决定标准文件夹是否也要更新为中文。

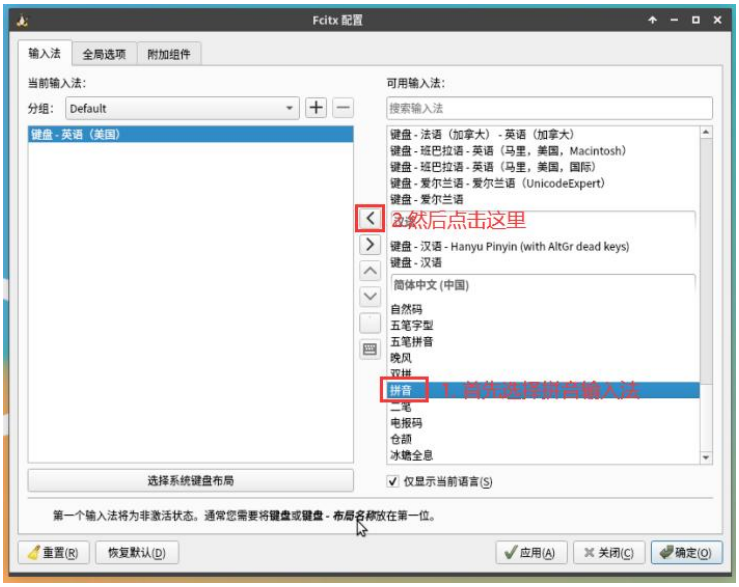
7) 然后可以看到桌面都显示为中文了。



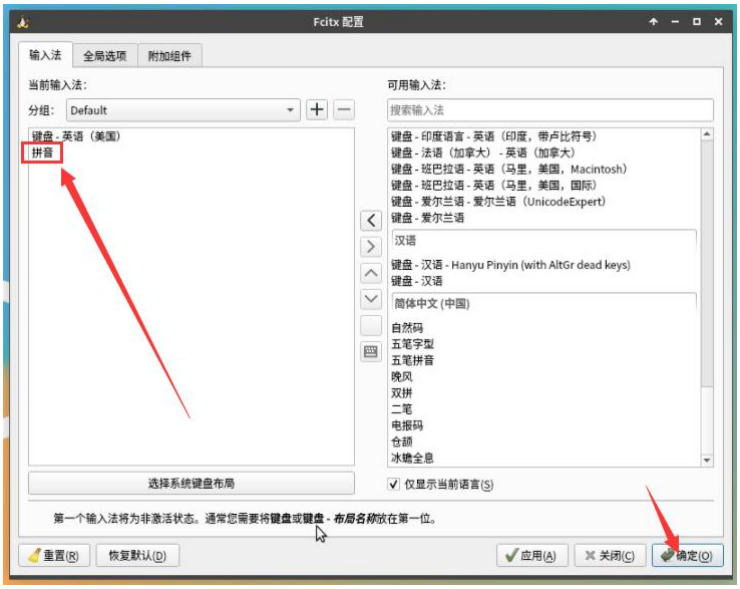
8) 然后打开 Fcitx5 配置程序。



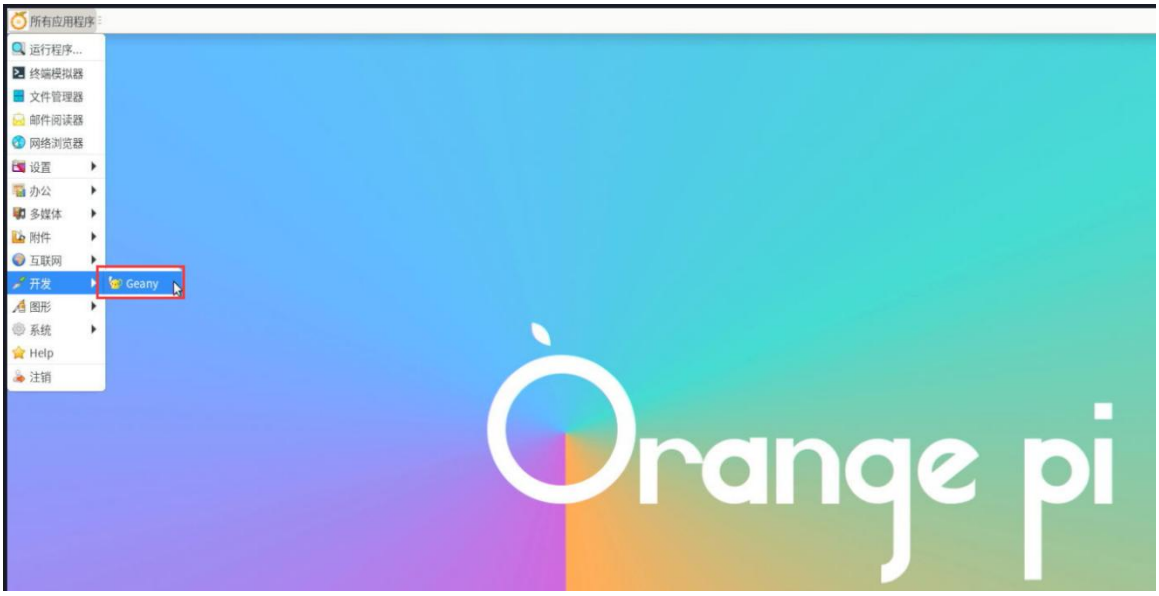
9) 然后选择使用拼音输入法。



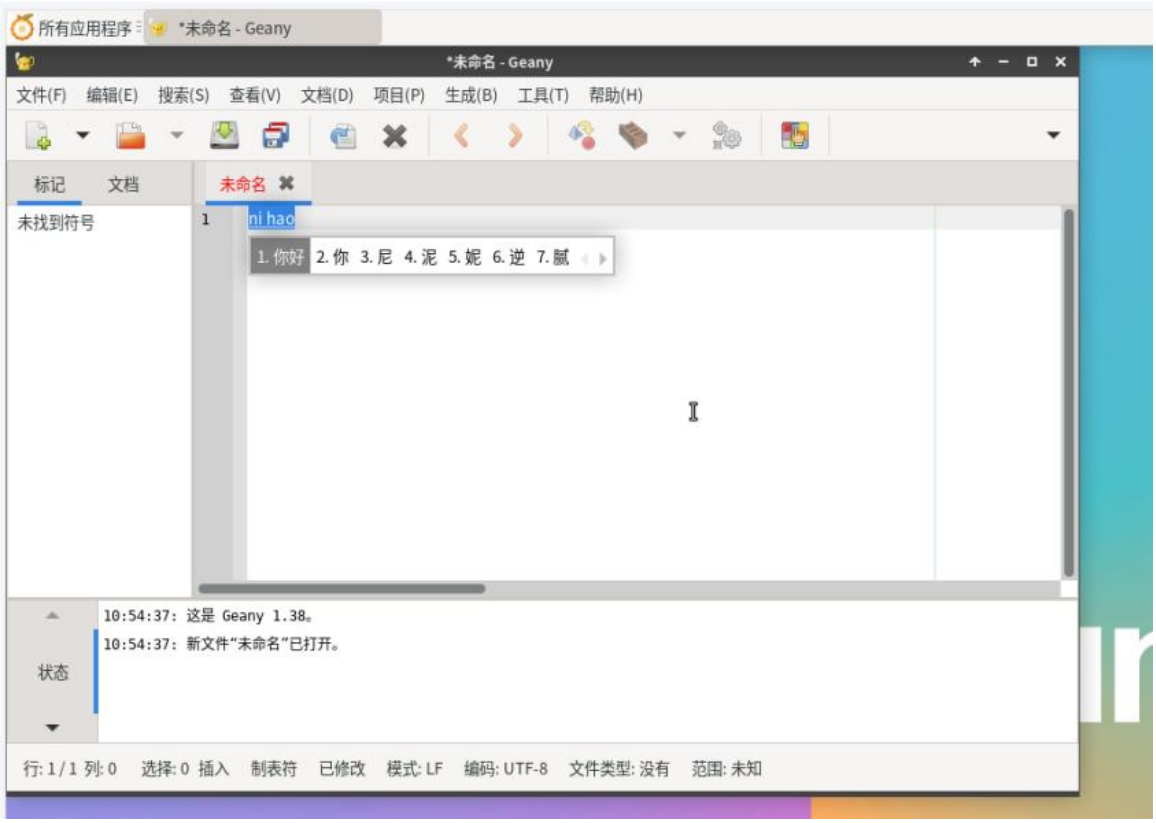
10) 选择后的界面如下所示，再点击确定即可。



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



12) 打开 **Geany** 后，默认还是英文输入法，我们可以通过 **Ctrl+Space** 快捷键来切换成中文输入法，然后就能输入中文了。



3.27. 远程登录 Linux 系统桌面的方法

Ubuntu Gnome Wayland 镜像不支持使用此处介绍的 Nomachine 和 VNC 来远程登录桌面。

3.27.1. 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。

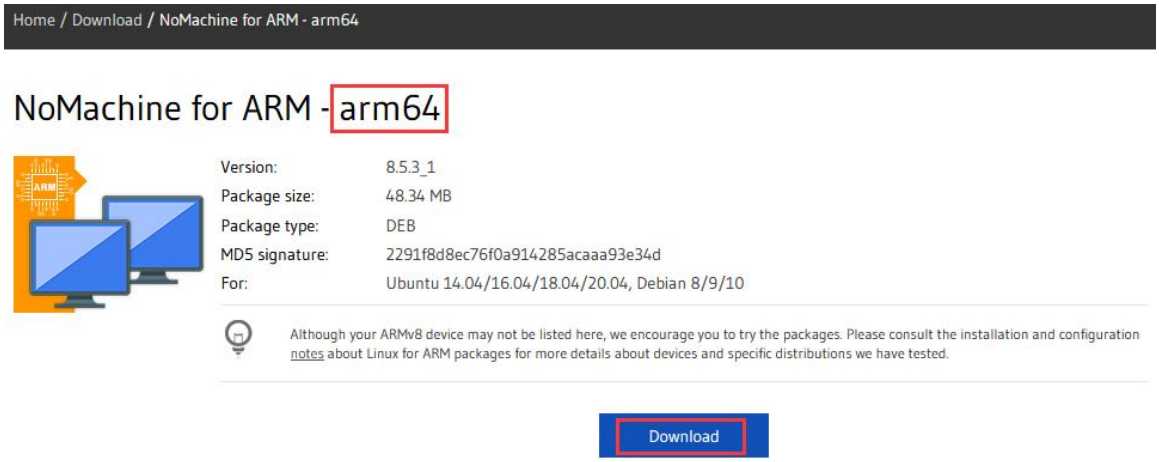
操作前请先确保 Windwos 电脑和开发板在同一局域网内，并且能正常 ssh 登录开发板的 Ubuntu 或者 Debian 系统。

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 RK3588S 是 ARMv8 架构的 SOC，我们使用的系统为 Ubuntu 或者 Debian，所以这里需要下载 **NoMachine for ARM ARMv8 DEB** 安装包，下载链接如下所示：

注意，这个下载链接可能会变，请认准 Armv8/Arm64 版本的 deb 包。

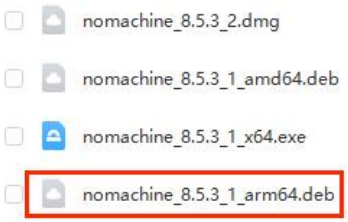
https://downloads.nomachine.com/download/?id=114&distro=ARM



b. 另外在**官方工具**中也可以下载到 **NoMachine** 的安装包。

先进入**远程登录软件-NoMachine** 文件夹。

然后下载 arm64 版本的 deb 安装包。



c. 然后将下载的 **nomachine_x.x.x_x_arm64.deb** 上传到开发板的 Linux 系统中

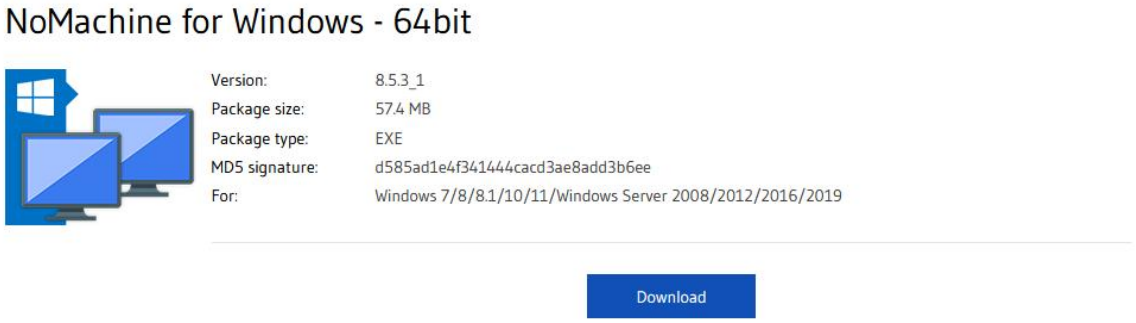
d. 然后使用下面的命令在开发板的 Linux 系统中安装 **NoMachine**。

```
orangepi@orangepi:~$ sudo dpkg -i nomachine_x.x.x_x_arm64_arm64.deb
```

2) 然后下载 NoMachine 软件 Windows 版本的安装包，下载地址如下所示：

注意，这个下载链接可能会变。

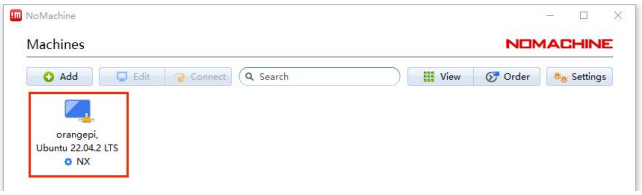
https://downloads.nomachine.com/download/?id=9



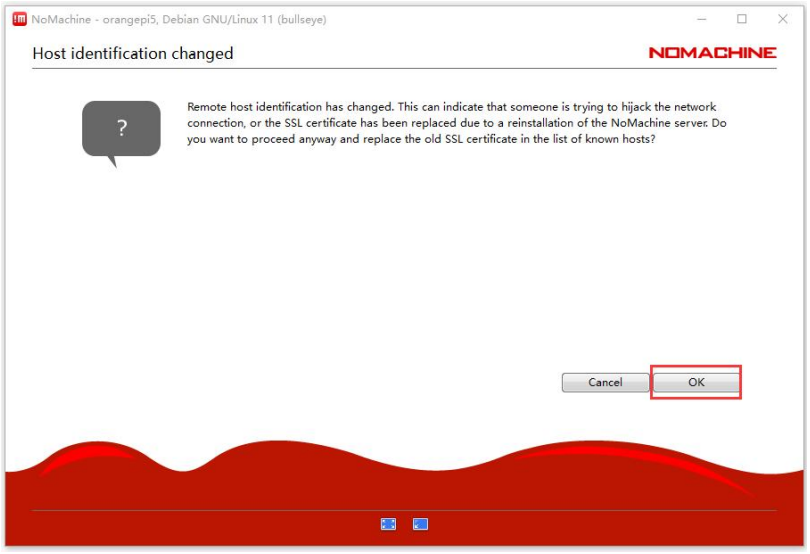
3) 然后在 Windows 中安装 NoMachine，**安装完后请重启下电脑**。

4) 然后在 Window 中打开 **NoMachine**。

5) NoMachine 启动后会自动扫描局域网内其他安装有 NoMachine 的设备，进入 NoMachine 的主界面后就可以看到开发板已经在可连接的设备列表里了，然后点击下图红色方框所示的位置即可开始登录开发板的 Linux 系统桌面。



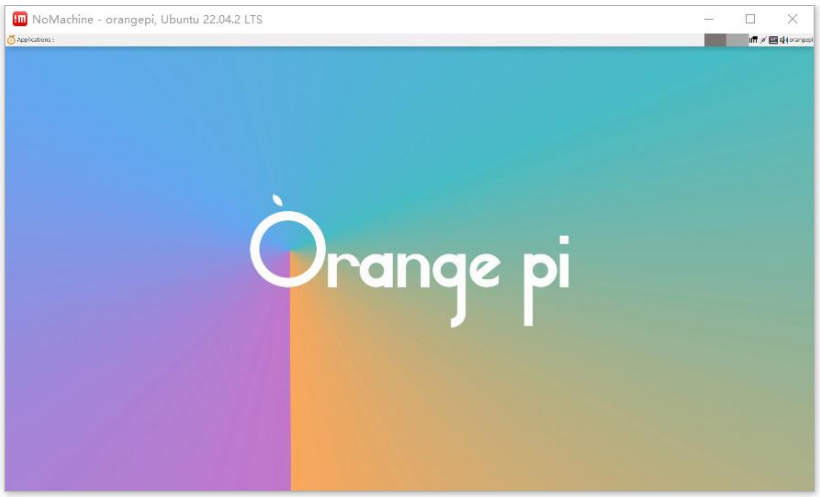
6) 然后点击 **OK**。



7) 然后在下图对应的位置输入开发板 Linux 系统的用户名和密码，再点击 **OK** 开始登陆。

8) 然后在接下来的界面中都点击 OK。

9) 最后就能看到开发板 Linux 系统的桌面了。



3.27.2. 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。

Ubuntu20.04 测试 VNC 很多问题，请不要使用这种方法。

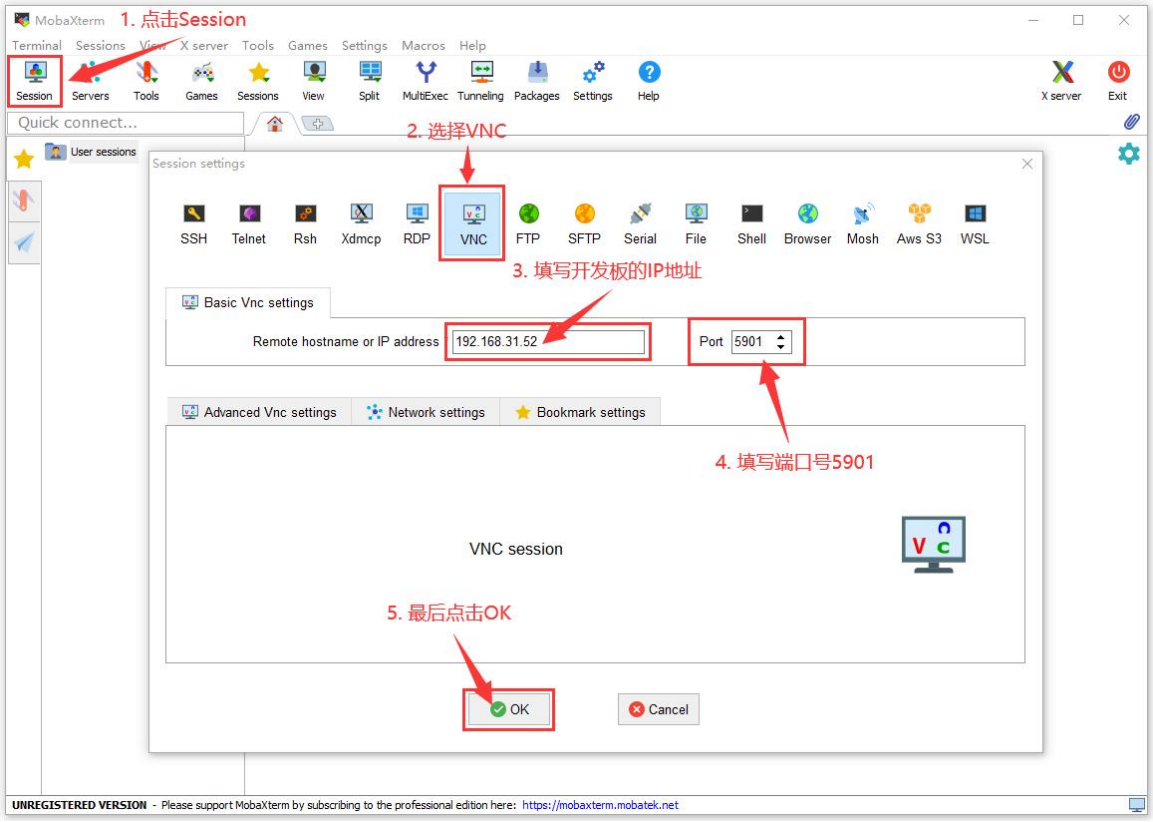
1) 首先运行 **set_vnc.sh** 脚本设置下 vnc，**记得加 sudo 权限**。

```
orangepi@orangepi:~$ sudo set_vnc.sh
You will require a password to access your desktops.

Password:          #在这里设置 vnc 的密码，8 位字符
Verify:            #在这里设置 vnc 的密码，8 位字符
Would you like to enter a view-only password (y/n)? n
xauth:    file /root/.Xauthority does not exist

New 'X' desktop is orangepicm5-tablet:1

Creating default startup script /root/.vnc/xstartup
Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/orangepicm5-tablet:1.log

Killing Xtightvnc process ID 3047

New 'X' desktop is orangepicm5-tablet:1
```

```
Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/orangepicm5-tablet:1.log
```

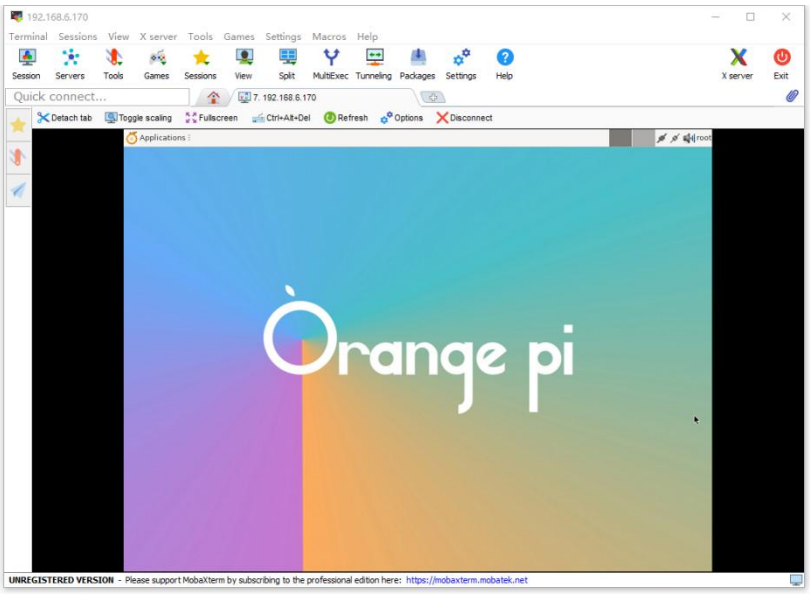
2) 使用 MobaXterm 软件连接开发板 linux 系统桌面的步骤如下所示：

a. 首先点击 Session，然后选择 VNC，再填写开发板的 IP 地址和端口，最后点击 OK 确认。



b. 然后输入前面设置的 VNC 的密码。

c. 登录成功后的界面显示如下图所示，然后就可以远程操作开发板 linux 系统的桌面了。



3.28. Linux 系统支持的部分编程语言测试

3.28.1. Debian Bullseye 系统

1) Debian Bullseye 默认安装有 gcc 编译工具链，可以直接在开发板的 Linux 系统中编译 C 语言的程序。

a. gcc 的版本如下所示：

```
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!
```

2) 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!
```

3) Debian Bullseye 默认没有安装 Java 的编译工具和运行环境。

a. 可以使用下面的命令安装 openjdk，Debian Bullseye 中最新版本为 openjdk-17

```
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!
```

3.28.2. Debian Bookworm 系统

1) Debian Bookworm 默认安装有 gcc 编译工具链，可以直接在开发板的 Linux 系统中编译 C 语言的程序。

a. gcc 的版本如下所示：

```
orangepi@orangepi:~$ gcc --version
gcc (Debian 12.2.0-14) 12.2.0
Copyright (C) 2022 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE.
```

b. 编写 C 语言的 **hello_world.c** 程序。

```
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 Bookworm 默认安装有 Python3。

a. Python 具体版本如下所示：

```
orangepi@orangepi:~$ python3
Python 3.11.2 (main, Mar 13 2023, 12:18:29) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>>
```

使用 Ctrl+D 快捷键可退出 python 的交互模式。

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

```
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) Debian Bookworm 默认没有安装 Java 的编译工具和运行环境。

a. 可以使用下面的命令安装 openjdk，Debian Bookworm 中最新版本为 openjdk-17。

```
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!
```

3.28.3. Ubuntu Focal 系统

1) Ubuntu Focal 默认安装有 gcc 编译工具链，可以直接在开发板的 Linux 系统中编译 C 语言的程序。

a. gcc 的版本如下所示：

```
orangepi@orangepi:~$ gcc --version
gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0
```

```
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE.
```

b. 编写 C 语言的 **hello_world.c** 程序。

```
orangepi@orangepi:~$ vim hello_world.c
#include <stdio.h>

int main(void)
{
        printf("Hello World!\n");

        return 0;
}
```

c. 然后编译运行 **hello_world.c**。

```
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, Nov 14 2022, 12:59:47)
[GCC 9.4.0] 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 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!
```

3.28.4. 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 11.4.0-1ubuntu1~22.04) 11.4.0
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 3.10.12 (main, Jul 29 2024, 16:56:48) [GCC 11.4.0] 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!
```

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 18.0.2-ea 2022-07-19
OpenJDK Runtime Environment (build 18.0.2-ea+9-Ubuntu-222.04)
OpenJDK 64-Bit Server VM (build 18.0.2-ea+9-Ubuntu-222.04, mixed mode, sharing)
```

c. 编写 Java 版本的 **hello_world.java**。

```
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!
```

3.29. QT 的安装方法

1) 使用下面的脚本可以安装 QT5 和 QT Creator。

```
orangepi@orangepi:~$ install_qt.sh
```

2) 安装完后会自动打印 QT 的版本号。

a. Ubuntu20.04 自带的 qt 版本为 **5.12.8**。

```
orangepi@orangepi:~$ install_qt.sh
......
QMake version 3.1
Using Qt version 5.12.8 in /usr/lib/aarch64-linux-gnu
```

b. Ubuntu22.04 自带的 QT 版本为 **5.15.3**。

```
orangepi@orangepi:~$ install_qt.sh
......
QMake version 3.1
Using Qt version 5.15.3 in /usr/lib/aarch64-linux-gnu
```

c. Debian11 自带的 QT 版本为 **5.15.2**。

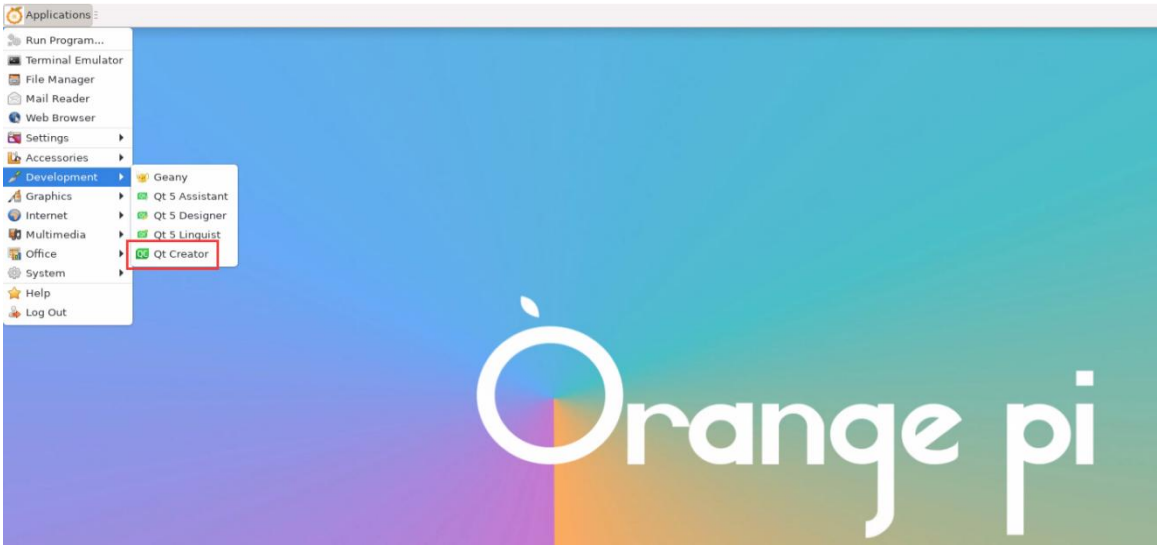
```
orangepi@orangepi:~$ install_qt.sh
......
QMake version 3.1
Using Qt version 5.15.2 in /usr/lib/aarch64-linux-gnu
```

d. Debian12 自带的 QT 版本为 **5.15.8**。

```
orangepi@orangepi:~$ install_qt.sh
......
QMake version 3.1
```

```
Using Qt version 5.15.8 in /usr/lib/aarch64-linux-gnu
```

3) 然后在 **Applications** 中就可以看到 QT Creator 的启动图标。



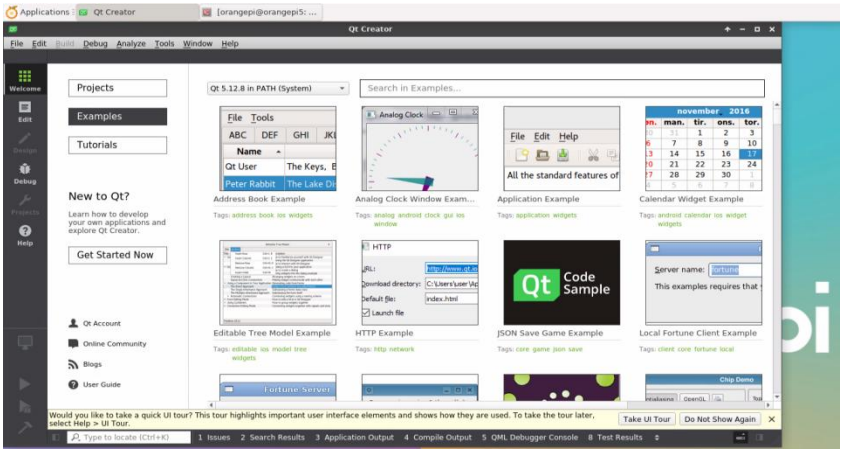
也可以使用下面的命令打开 QT Creator。

```
orangepi@orangepi:~$ qtcreator
```

在QT和QT应用的启动过程中，如果提示下面的错误，请直接忽略，此错误对应用的运行不会有影响。

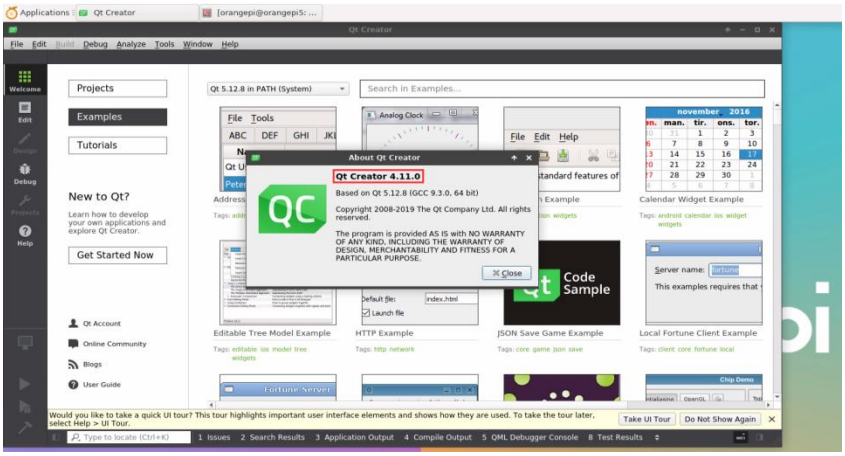
```
libGL error: failed to create dri screen
libGL error: failed to load driver: rockchip
libGL error: failed to create dri screen
libGL error: failed to load driver: rockchip
```

4) QT Creator 打开后的界面如下所示：

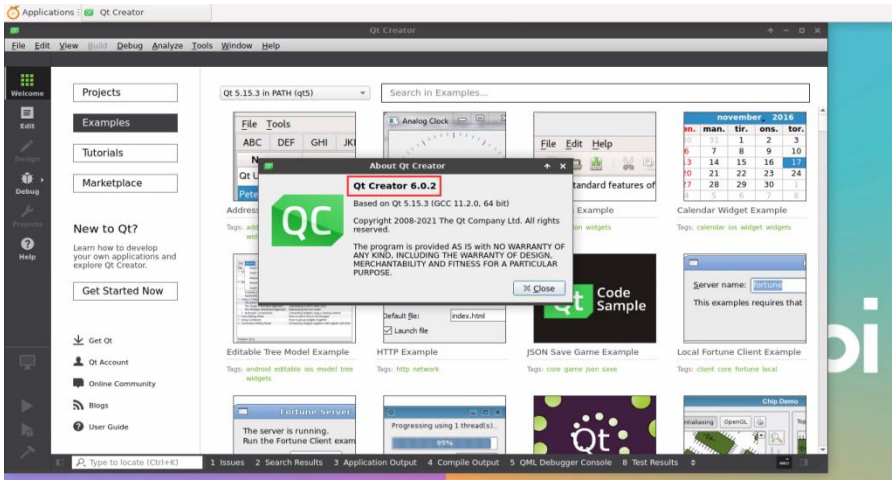


5) QT Creator 的版本如下所示：

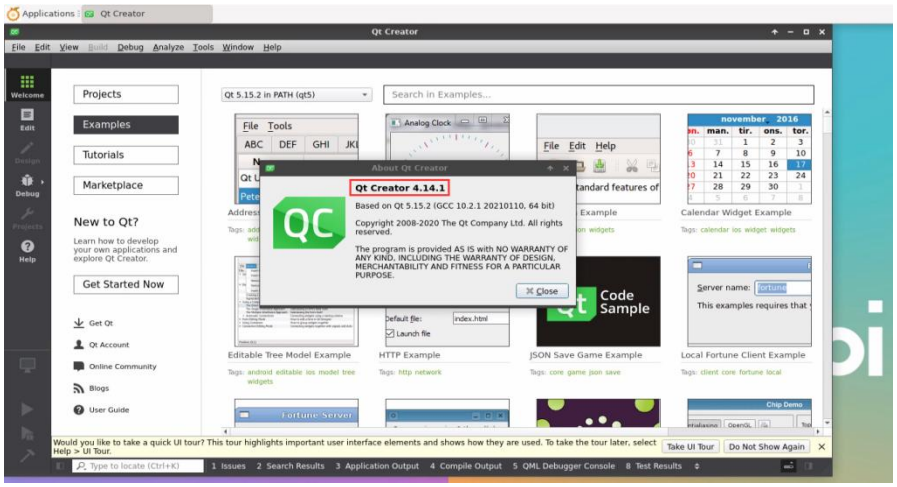
a. QT Creator 在 **Ubuntu20.04** 中的默认版本如下所示：



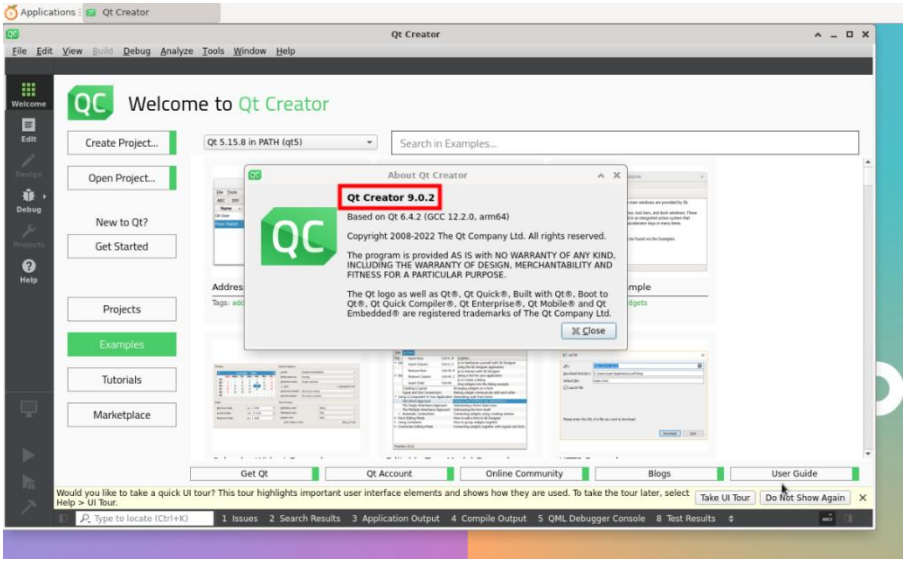
b. QT Creator 在 **Ubuntu22.04** 中的默认版本如下所示：



c. QT Creator 在 **Debian11** 中的默认版本如下所示：

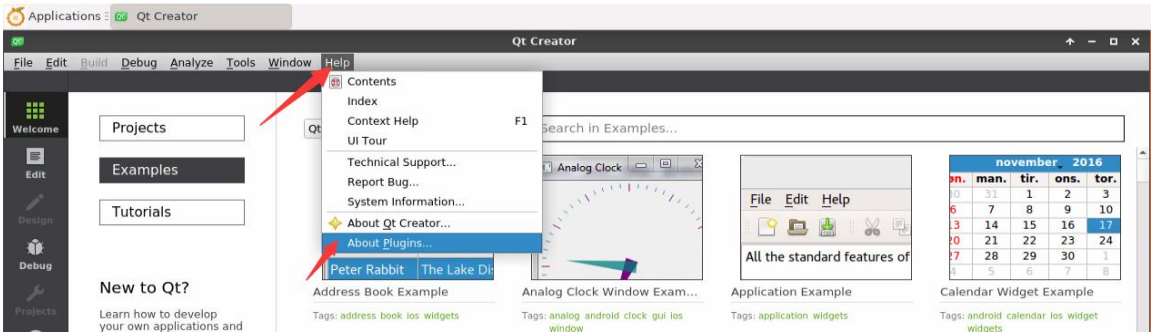


d. QT Creator 在 **Debian12** 中的默认版本如下所示：

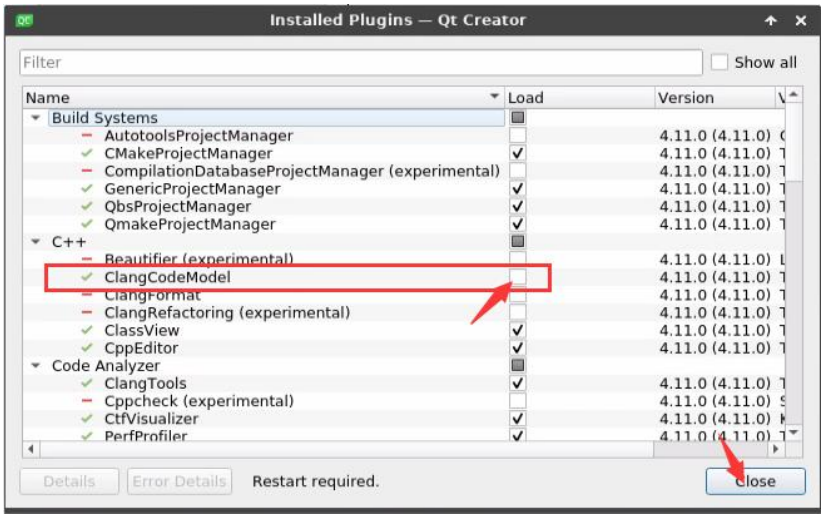


6) 然后设置下 QT。

a. 首先打开 **Help->About Plugins...**。



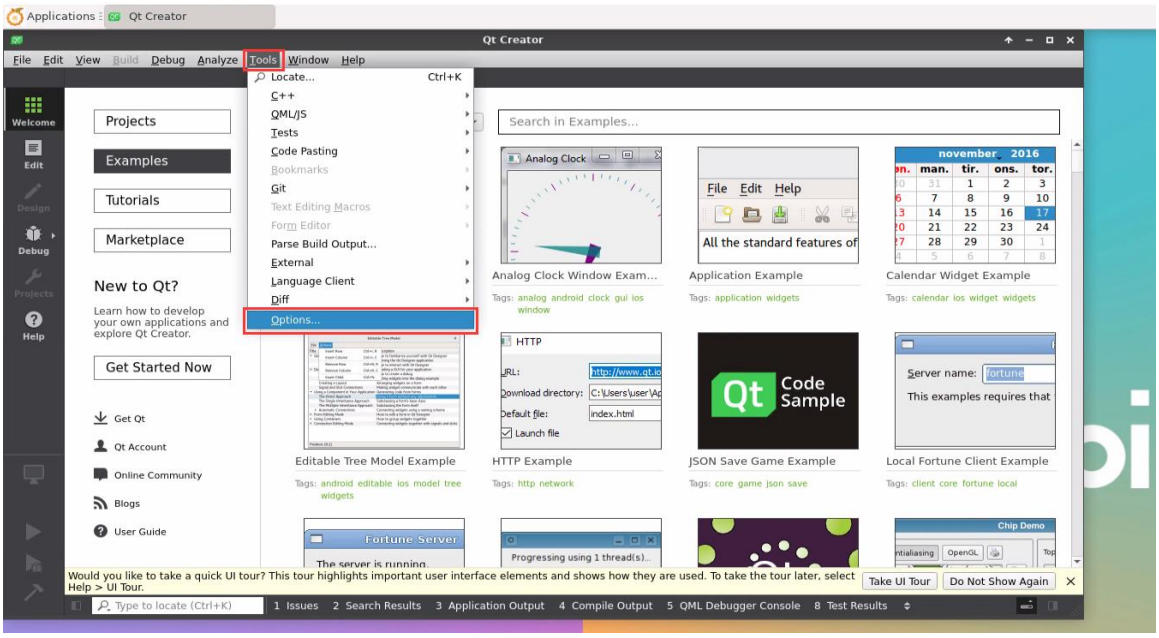
b. 然后去掉 **ClangCodeModel** 的那个勾。

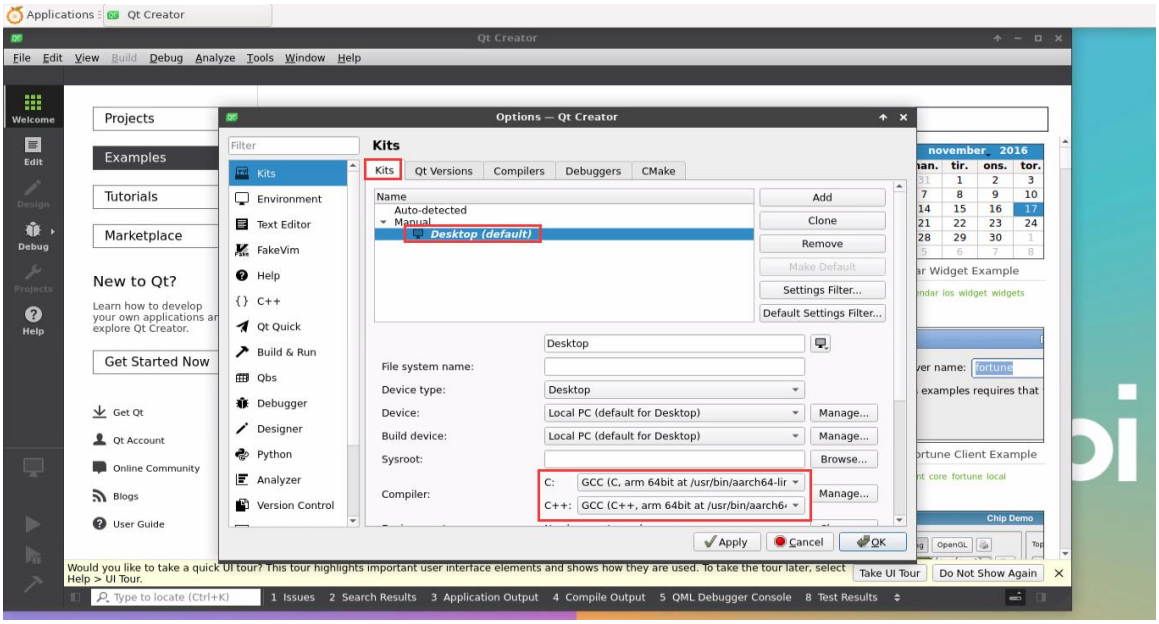


c. **设置完后需要重启下 QT Creator**。

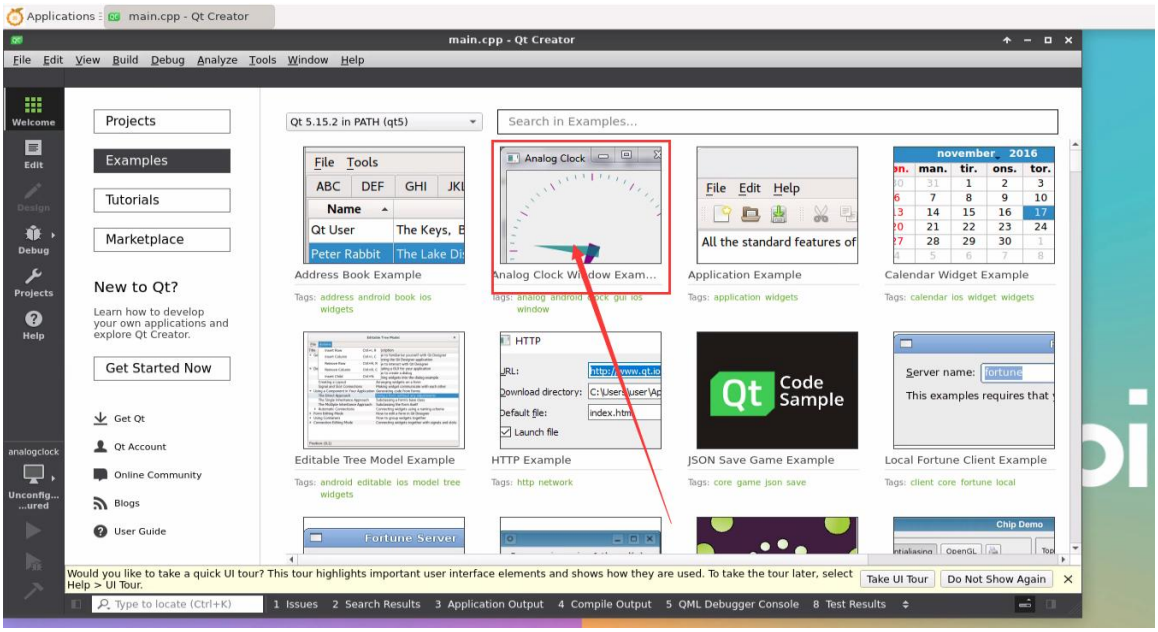
d. 然后确保 QT Creator 使用的 GCC 编译器，如果默认为 Clang，请修改为 GCC。

Debian12 请跳过这步。

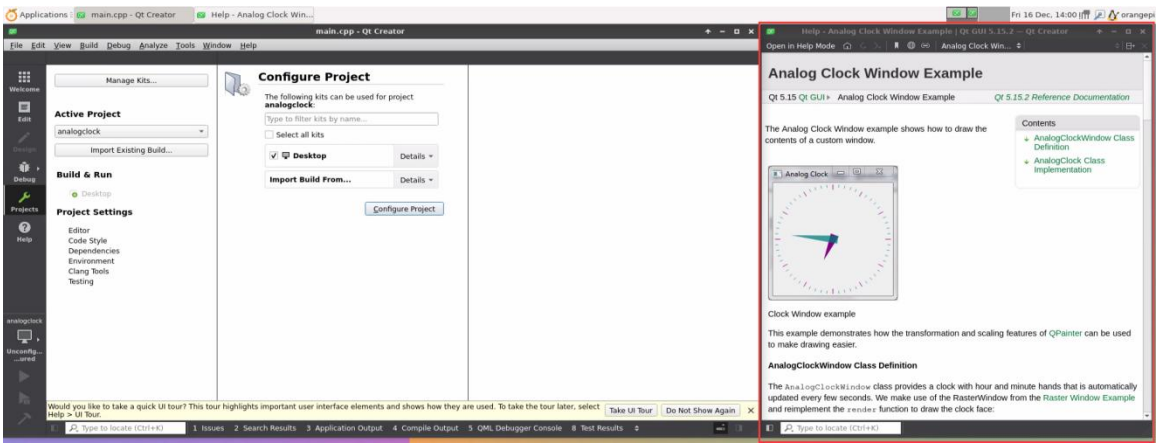




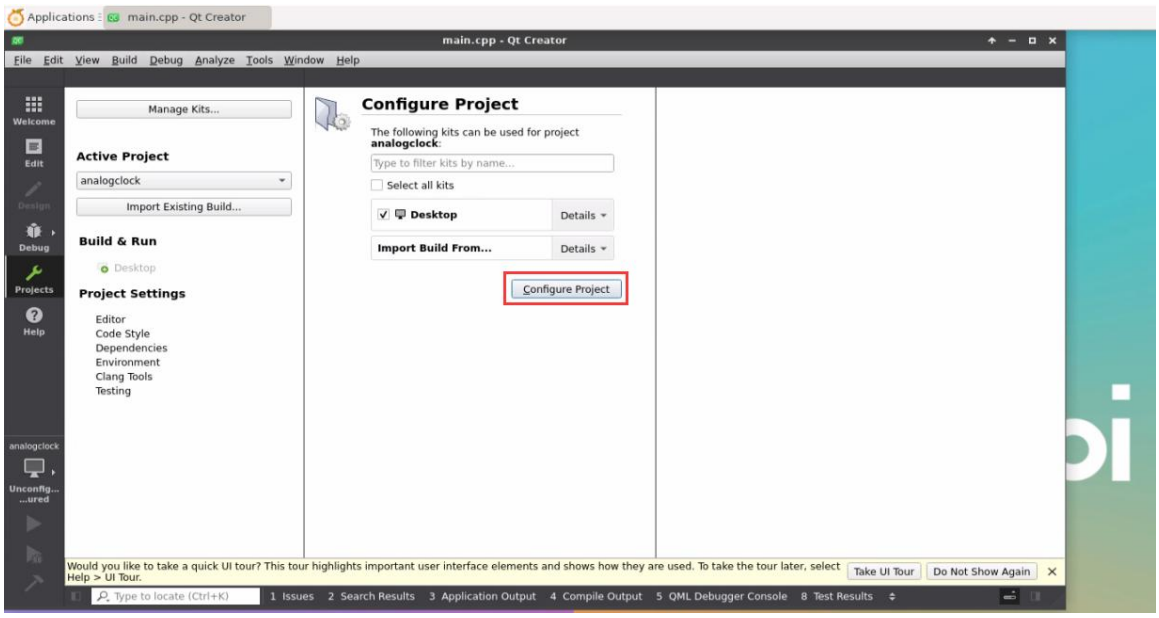
7) 然后就可以打开一个示例代码。



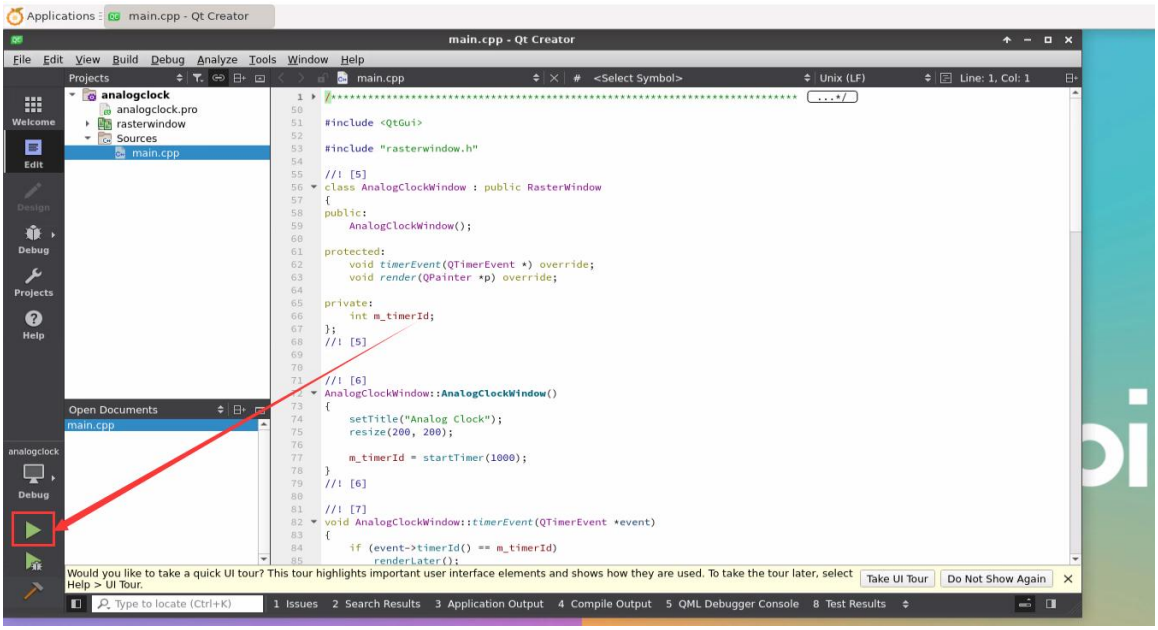
8) 点击示例代码后会自动打开对应的说明文档，可以仔细看下其中的使用说明。



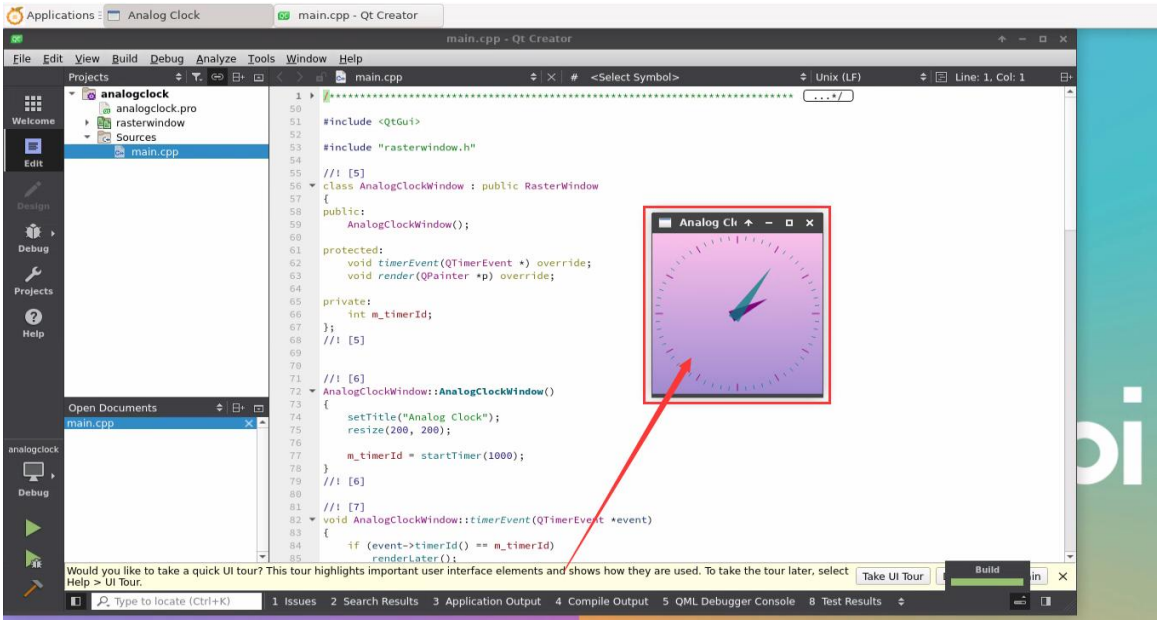
9) 然后点击下 **Configure Project**。



10) 然后点击左下角的绿色三角形编译运行下示例代码。



11) 等待一段时间后，会弹出下图所示的界面，此时就说明 QT 能正常编译运行。



12) 参考资料。

https://wiki.qt.io/Install_Qt_5_on_Ubuntu
https://download.qt.io/archive/qtcreator
https://download.qt.io/archive/qt

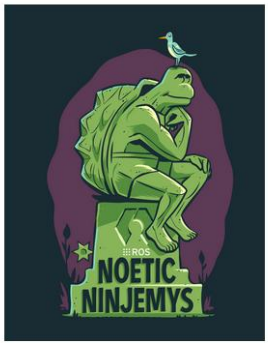
3.30. ROS 安装方法

3.30.1. Ubuntu20.04 安装 ROS 1 Noetic 的方法

1) ROS 1 当前活跃的版本如下所示，推荐版本为 **Noetic Ninjemys**。

Active ROS 1 distributions

Recommended



Distro	Release date	Poster	*Tuturtle*, turtle in tutorial	EOL date
ROS Noetic Ninjemys (**Recommended**)	May 23rd, 2020			May, 2025 (Focal EOL)
ROS Melodic Morenia	May 23rd, 2018			May, 2023 (Bionic EOL)

http://docs.ros.org
https://wiki.ros.org/Distributions

2) ROS 1 **Noetic Ninjemys** 官方安装文档链接如下所示：

http://wiki.ros.org/noetic/Installation/Ubuntu

3) ROS **Noetic Ninjemys** 官方安装文档中 Ubuntu 推荐使用 Ubuntu20.04，所以请确保开发板使用的系统为 **Ubuntu20.04 桌面版系统**。

http://wiki.ros.org/noetic/Installation

4) 然后使用下面的脚本安装 ros1。

```
orangepi@orangepicm5-tablet:~$ install_ros.sh ros1
```

5) 使用 ROS 工具前，首先需要初始化下 rosdep，然后编译源码时就能快速的安装一些系统依赖和一些 ROS 中的核心组件。

注意，运行下面的命令需要确保开发板能正常访问 github，否则会由于网络问题而报错。

install_ros.sh 脚本会尝试修改/etc/hosts 并自动运行下面的命令。但是这种方法无法保证每次都能正常访问 github，如果 install_ros.sh 安装完 ros1 后有提示下面的错误，请自己想其它办法让开发板的 linux 系统能正常访问 github，然后再手动运行下面的命令。

https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/osx-homebrew.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/base.yaml
ERROR: error loading sources list:
The read operation timed out

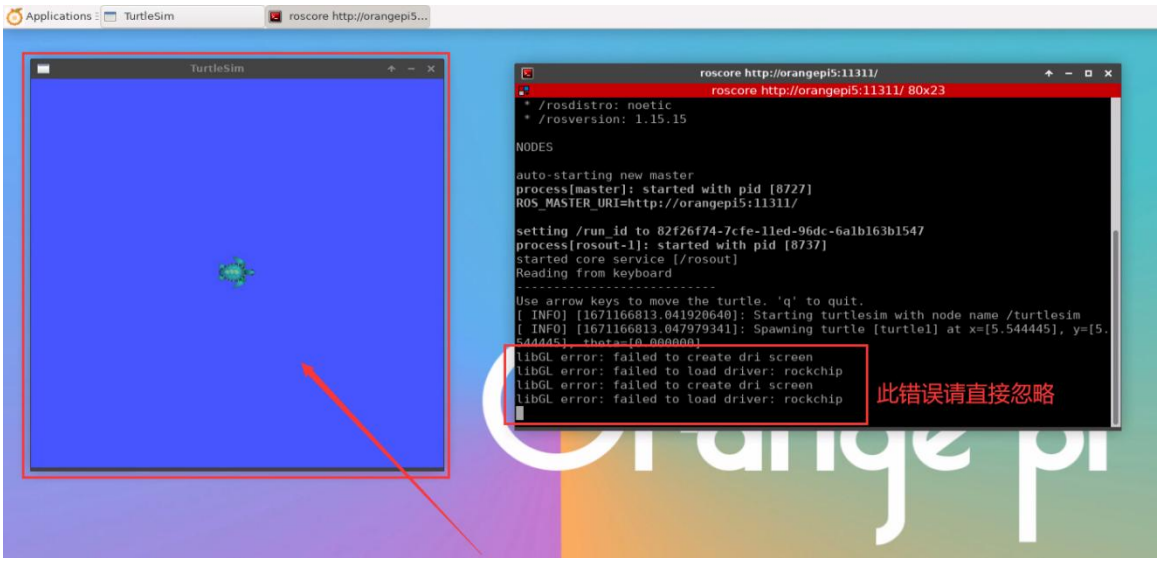
```
orangepi@orangepi:~$ source /opt/ros/noetic/setup.bash
orangepi@orangepi:~$ sudo rosdep init
Wrote /etc/ros/rosdep/sources.list.d/20-default.list
Recommended: please run

        rosdep update
orangepi@orangepi:~$ rosdep update
reading in sources list data from /etc/ros/rosdep/sources.list.d
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/osx-homebrew.yaml
```

```
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/base.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/python.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/ruby.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/releases/fuerte.yaml
Query rosdistro index
https://raw.githubusercontent.com/ros/rosdistro/master/index-v4.yaml
Skip end-of-life distro "ardent"
Skip end-of-life distro "bouncy"
Skip end-of-life distro "crystal"
Skip end-of-life distro "dashing"
Skip end-of-life distro "eloquent"
Add distro "foxy"
Add distro "galactic"
Skip end-of-life distro "groovy"
Add distro "humble"
Skip end-of-life distro "hydro"
Skip end-of-life distro "indigo"
Skip end-of-life distro "jade"
Skip end-of-life distro "kinetic"
Skip end-of-life distro "lunar"
Add distro "melodic"
Add distro "noetic"
Add distro "rolling"
updated cache in /home/orangepi/.ros/rosdep/sources.cache
```

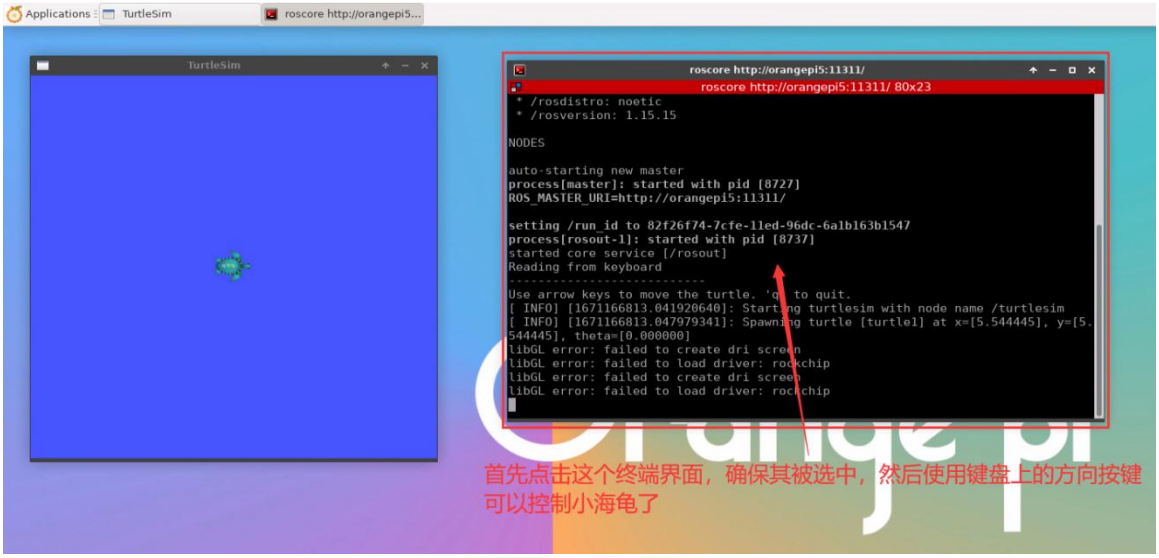
6) 然后在**桌面**中打开一个命令行终端窗口，再使用 **test_ros.sh** 脚本可以启动一个小海龟的例程来测试下 ROS 是否能正常使用。

```
orangepi@orangepi:~$ test_ros.sh
```

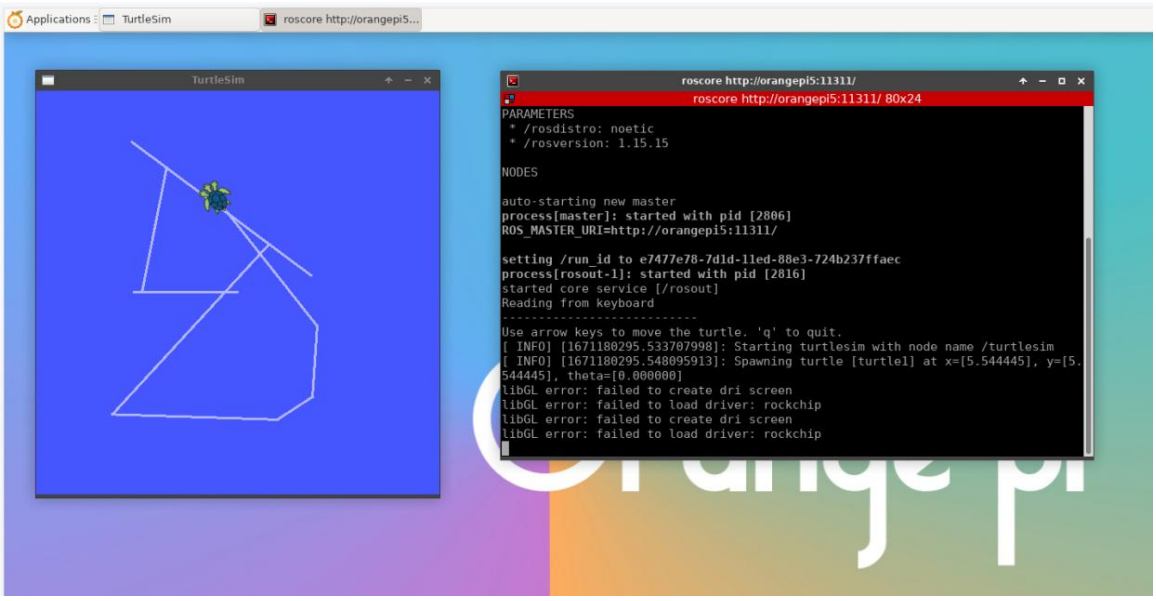
7) 运行完 **test_ros.sh** 脚本后，会弹出下图所示的一个小海龟。



8) 然后请保持刚才打开终端窗口在最上面。

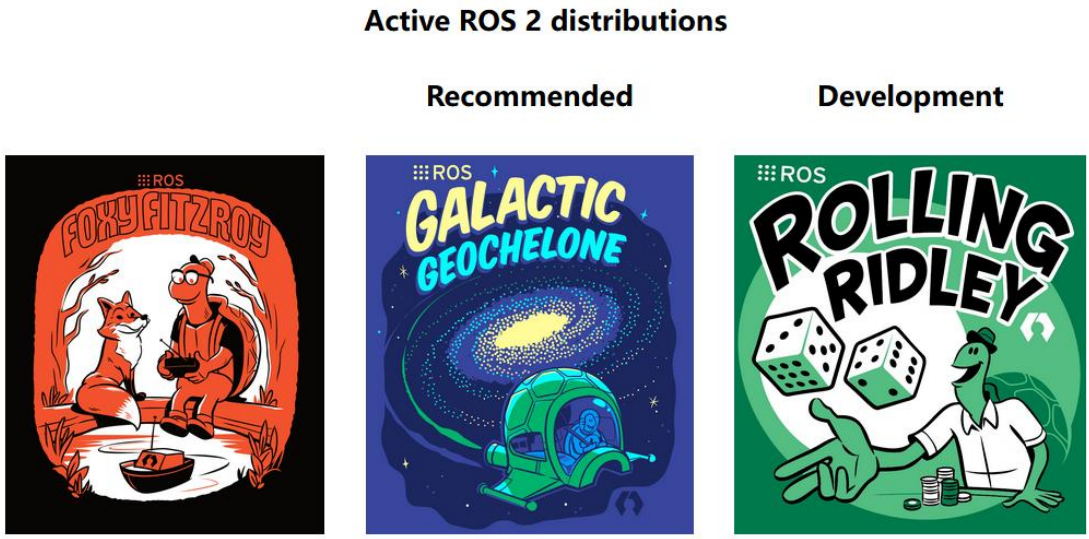


9) 此时按下键盘上的方向按键就可以控制小海龟上下左右移动了。



3.30.2. Ubuntu20.04 安装 ROS 2 Galactic 的方法

1) ROS 2 当前活跃的版本如下所示，推荐版本为 **Galactic Geochelone**。



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 桌面版系统**。安装 ROS 2 有几种方法，下面演示下通过 **Debian packages** 的方式来安装 ROS 2 **Galactic Geochelone**。

4) 使用 **install_ros.sh** 脚本可以安装 ros2。

```
orangepi@orangepi:~$ install_ros.sh ros2
```

5) **install_ros.sh** 脚本安装完 ros2 后会自动运行下 **ros2 -h** 命令，如果能看到下面的打印，说明 ros2 安装完成。

```
usage: ros2 [-h] Call `ros2 <command> -h` for more detailed usage. ...

ros2 is an extensible command-line tool for ROS 2.

optional arguments:
  -h, --help            show this help message and exit
```

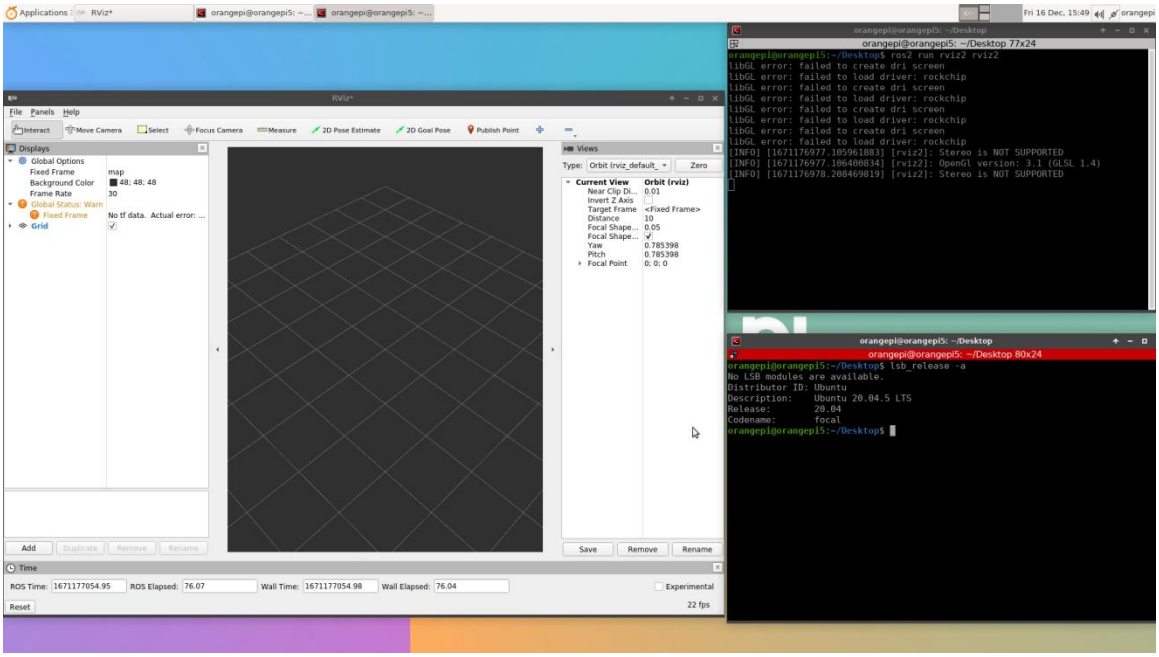
```
Commands:
  action         Various action related sub-commands
  bag            Various rosbag related sub-commands
  component      Various component related sub-commands
  daemon         Various daemon related sub-commands
  doctor         Check ROS setup and other potential issues
  interface      Show information about ROS interfaces
  launch         Run a launch file
  lifecycle      Various lifecycle related sub-commands
  multicast      Various multicast related sub-commands
  node           Various node related sub-commands
  param          Various param related sub-commands
  pkg            Various package related sub-commands
  run            Run a package specific executable
  security       Various security related sub-commands
  service        Various service related sub-commands
  topic          Various topic related sub-commands
  wtf            Use `wtf` as alias to `doctor`

  Call `ros2 <command> -h` for more detailed usage.
```

6) 然后可以使用 **test_ros.sh** 脚本测试下 ROS 2 是否安装成功，如果能看到下面的打印，说明 ROS 2 能正常运行。

```
orangepi@orangepicm5-tablet:~$ test_ros.sh
[INFO] [1671174101.200091527] [talker]: Publishing: 'Hello World: 1'
[INFO] [1671174101.235661048] [listener]: I heard: [Hello World: 1]
[INFO] [1671174102.199572327] [talker]: Publishing: 'Hello World: 2'
[INFO] [1671174102.204196299] [listener]: I heard: [Hello World: 2]
[INFO] [1671174103.199580322] [talker]: Publishing: 'Hello World: 3'
[INFO] [1671174103.204019965] [listener]: I heard: [Hello World: 3]
```

7) 运行下面的命令可以打开 rviz2。

```
orangepi@orangepi:~$ source /opt/ros/galactic/setup.bash
orangepi@orangepi:~$ ros2 run rviz2 rviz2
```



8) ROS 的使用方法请参考下 ROS 2 的文档。

http://docs.ros.org/en/galactic/Tutorials.html

3. 30. 3. Ubuntu22.04 安装 ROS 2 Humble 的方法

1) 使用 **install_ros.sh** 脚本可以安装 ros2。

```
orangepi@orangepi:~$ install_ros.sh ros2
```

2) **install_ros.sh** 脚本安装完 ros2 后会自动运行下 **ros2 -h** 命令，如果能看到下面的打印，说明 ros2 安装完成。

```
usage: ros2 [-h] Call `ros2 <command> -h` for more detailed usage. ...

ros2 is an extensible command-line tool for ROS 2.

optional arguments:
  -h, --help            show this help message and exit

Commands:
  action      Various action related sub-commands
  bag         Various rosbag related sub-commands
  component   Various component related sub-commands
```

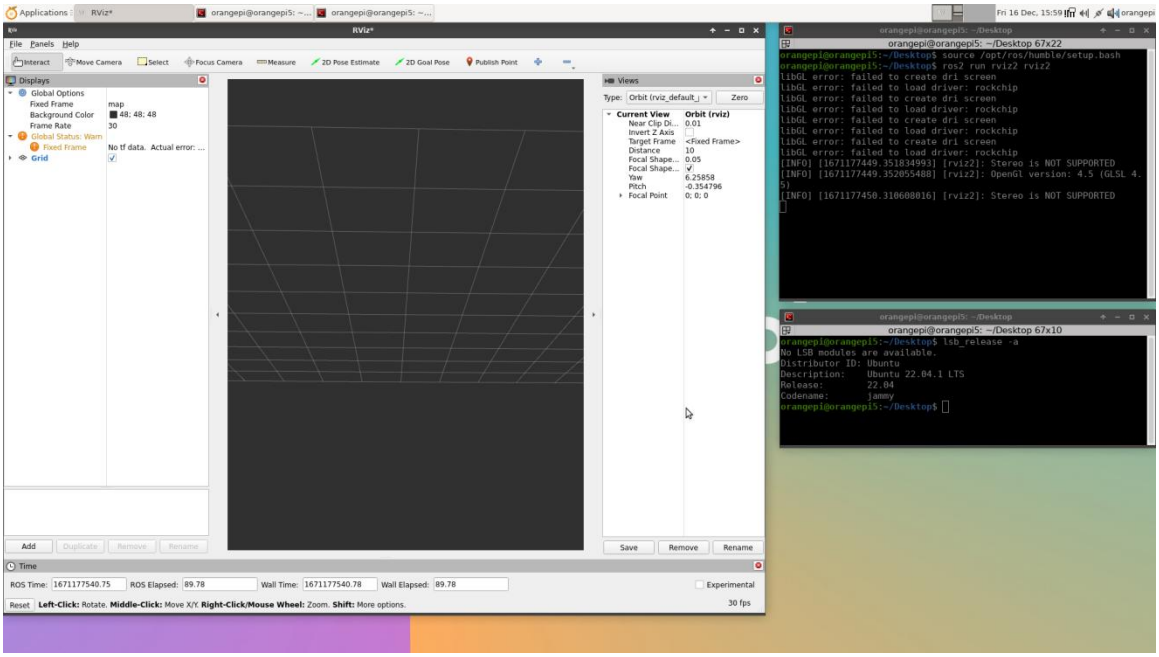
```
daemon      Various daemon related sub-commands
doctor      Check ROS setup and other potential issues
interface   Show information about ROS interfaces
launch      Run a launch file
lifecycle   Various lifecycle related sub-commands
multicast   Various multicast related sub-commands
node        Various node related sub-commands
param       Various param related sub-commands
pkg         Various package related sub-commands
run         Run a package specific executable
security    Various security related sub-commands
service     Various service related sub-commands
topic       Various topic related sub-commands
wtf         Use `wtf` as alias to `doctor`

Call `ros2 <command> -h` for more detailed usage.
```

3) 然后可以使用 **test_ros.sh** 脚本测试下 ROS 2 是否安装成功，如果能看到下面的打印，说明 ROS 2 能正常运行。

```
orangepi@orangepicm5-tablet:~$ test_ros.sh
[INFO] [1671174101.200091527] [talker]: Publishing: 'Hello World: 1'
[INFO] [1671174101.235661048] [listener]: I heard: [Hello World: 1]
[INFO] [1671174102.199572327] [talker]: Publishing: 'Hello World: 2'
[INFO] [1671174102.204196299] [listener]: I heard: [Hello World: 2]
[INFO] [1671174103.199580322] [talker]: Publishing: 'Hello World: 3'
[INFO] [1671174103.204019965] [listener]: I heard: [Hello World: 3]
```

4) 运行下面的命令可以打开 rviz2。

```
orangepi@orangepi:~$ source /opt/ros/humble/setup.bash
orangepi@orangepi:~$ ros2 run rviz2 rviz2
```



5) 参考文档。

http://docs.ros.org/en/humble/index.html
http://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debians.html

3.31. 安装内核头文件的方法

1) OPi 发布的 Linux 镜像默认自带了内核头文件的 deb 包，存放的位置为**/opt/**。

```
orangepi@orangepi:~$ ls /opt/linux-headers*
/opt/linux-headers-legacy-rockchip-rk3588_x.x.x_arm64.deb
```

2) 使用下面的命令可以安装内核头文件的 deb 包。

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。

```
orangepi@orangepi:~$ sudo dpkg -i /opt/linux-headers-legacy-rockchip-rk3588_1.x.x_arm64.deb
```

3) 安装完后在**/usr/src** 下就能看到内核头文件所在的文件夹。

```
orangepi@orangepi:~$ ls /usr/src
linux-headers-5.10.160-rockchip-rk3588
```

4) 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：

```
orangepi@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：

```
orangepi@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：

如果自己复制的代码这里编译如果有问题，请去官方工具中下载源码然后上传

到开发板的 Linux 系统中测试。


```
orangepi@orangepi:~$ make
make -C /lib/modules/5.10.160-rockchip-rk3588/build M=/home/orangepi modules
make[1]: Entering directory '/usr/src/linux-headers-5.10.160-rockchip-rk3588'
  CC [M]  /home/orangepi/hello.o
  MODPOST /home/orangepi/Module.symvers
  CC [M]  /home/orangepi/hello.mod.o
  LD [M]  /home/orangepi/hello.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.10.160-rockchip-rk3588'
```

d. 编译完后会生成 **hello.ko** 内核模块。

```
orangepi@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。

```
orangepi@orangepi:~$ sudo 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。

```
orangepi@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。

```
orangepi@orangepi:~$ sudo rmmod hello
orangepi@orangepi:~$ dmesg | grep "Hello"
[ 2871.893988] Hello Orange Pi -- init
[ 3173.800892] Hello Orange Pi -- exit
```

3.32. 10.1 寸 MIPI LCD 屏幕的使用方法

3.32.1. 10.1 寸 MIPI 屏幕的组装方法

1) 首先准备需要的配件。

a. 10.1 寸 MIPI LCD 显示屏+触摸屏。

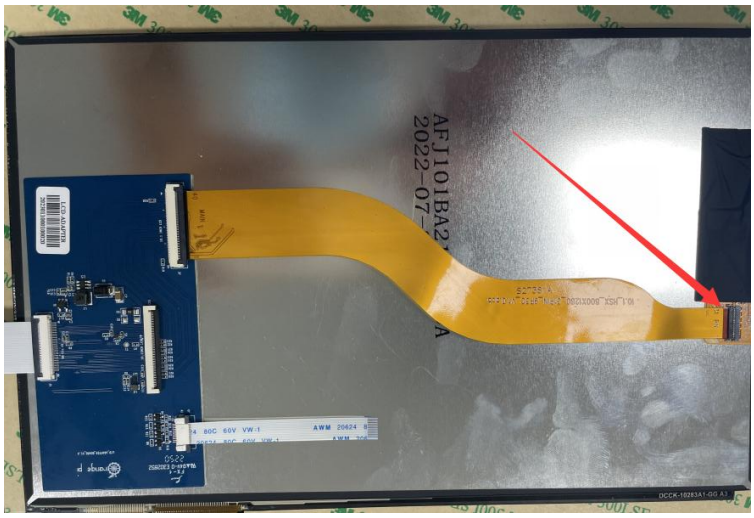
b. 屏幕转接板+31pin 转 26pin 排线。

c. 30pin MIPI 排线。

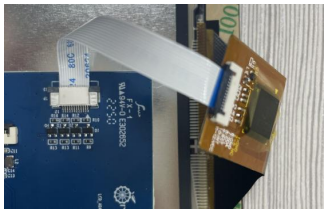
d. 12pin 触摸屏排线。

2) 按照下图将 12pin 触摸屏排线、31pin 转 26pin 排线、30pin MIPI 排线接到屏幕转接板上，注意**触摸屏排线蓝色的绝缘面朝下**，其它两根排线绝缘面朝上，如果接错会导致无显示或者不能触摸的问题。

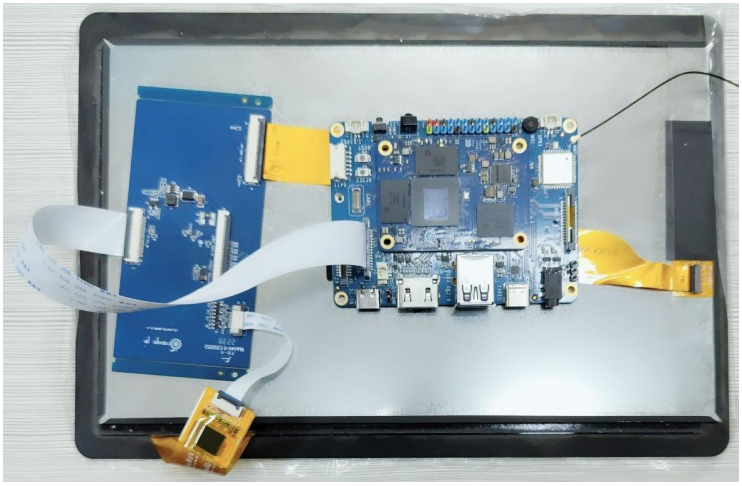
3) 按照下图将连接好排线的转接板置于 MIPI LCD 屏上面，并通过 31pin 转 26pin 排线连接 MIPI LCD 屏与转接板。



4) 然后通过 12pin 触摸屏排线连接触摸屏与转接板，注意绝缘面的朝向。



5) 最后通过 30pin MIPI 排线连接到开发板的 LCD 接口上。



3.32.2. 打开 10.1 寸 MIPI LCD 屏幕配置的方法

1) linux 镜像默认是没有打开 mipi lcd 屏幕的配置的，如果需要使用 mipi lcd 屏幕，需要手动打开才行。

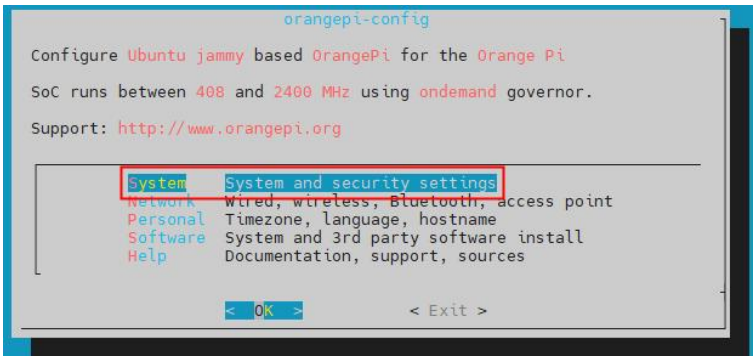
2) 开发板上 mipi lcd 屏幕的接口的位置如下图所示：

3) 打开 mipi lcd 配置的步骤如下所示：

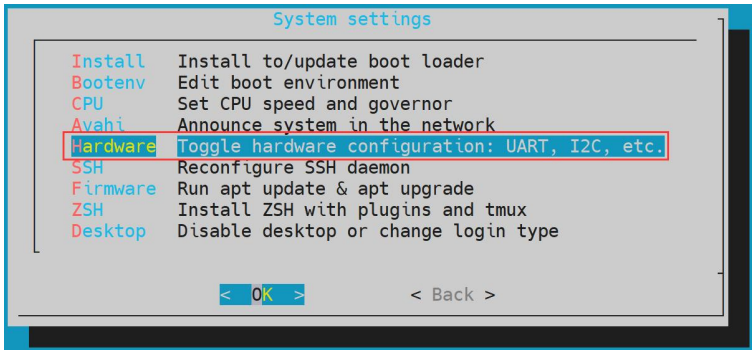
a. 首先运行下 **orangepi-config**，普通用户记得加 **sudo** 权限。

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

b. 然后选择 **System**。



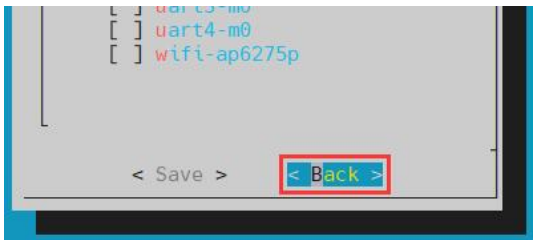
c. 然后选择 **Hardware**。



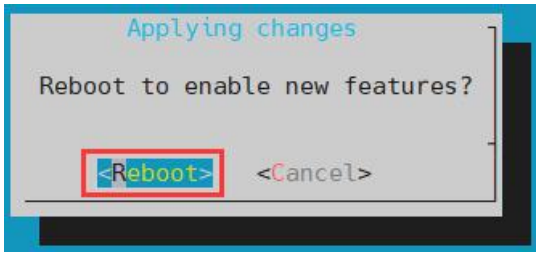
d. 然后使用键盘的方向键定位到 **opicm5-tablet-lcd**，再使用**空格**选中。

e. 然后选择**<Save>**保存。

f. 然后选择**<Back>**。



g. 然后选择**<Reboot>**重启系统使配置生效。



上面的设置最终会在/boot/orangepiEnv.txt 中加入 overlays=opicm5-tablet-lcd。设置完后可以先检查下。如果不存在这行配置，那么设置就是有问题。

如果觉得使用 orangepi-config 比较麻烦，也可以使用 vim 编辑器打开/boot/orangepiEnv.txt，然后加入 overlays=opicm5-tablet-lcd 这行配置也是可以。

orangepi@orangepi:~$ **cat /boot/orangepiEnv.txt | grep "lcd"**
overlays=opicm5-tablet-lcd #示例配置

4) 启动后可以看到 lcd 屏幕的显示如下所示（默认为竖屏）：

3.32.3. 服务器版镜像旋转显示方向的方法

1) 在**/boot/orangepiEnv.txt** 中加入 **extraargs=fbcon=rotate:要旋转的方向**这行配置就可以设置服务器版本的 linux 系统显示的方向，其中 **fbcon=rotate:**后面的数字可以设置为：

a. 0: 正常屏（默认为竖屏）
b. 1: 顺时钟转 90 度
c. 2: 翻转 180 度
d. 3: 顺时钟转 270 度

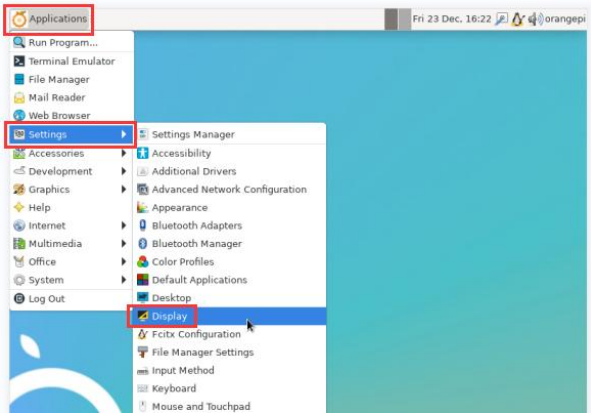
```
orangepi@orangepi:~$ sudo vim /boot/orangepiEnv.txt
overlays=opicm5-tablet-lcd
extraargs=cma=128M fbcon=rotate:3
```

注意，/boot/orangepiEnv.txt 中如果默认有 extraargs=cma=128M 这行配置，fbcon=rotate:3 这个配置添加到 extraargs=cma=128M 的后面即可（需要用空格隔开）。

2) 然后重启 linux 系统就能看到 lcd 屏幕显示的方向已经旋转了。

3.32.4. 桌面版镜像旋转显示和触摸方向的方法

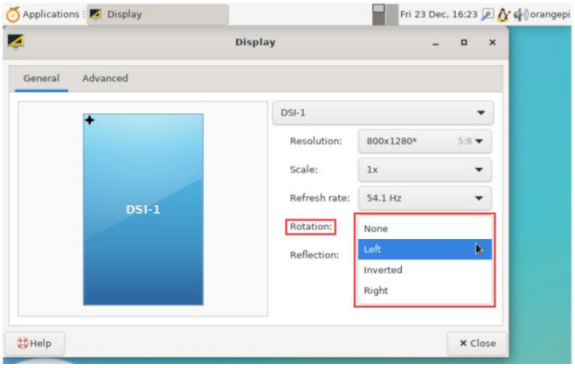
1) 首先在 linux 系统中打开 **Display** 设置。



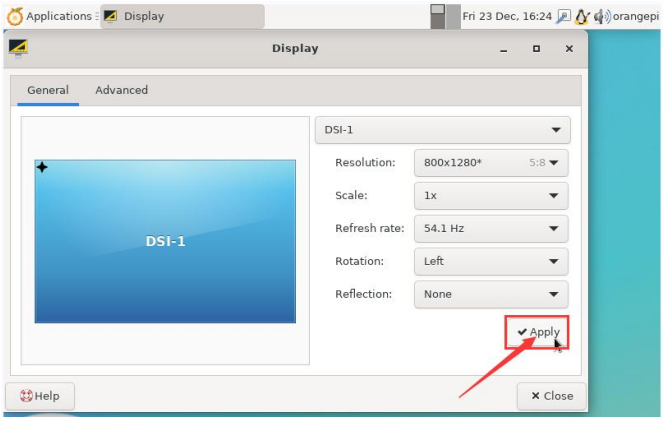
2) 然后在 **Rotation** 中选择想要旋转的方向。

a. **None**: 不旋转
b. **Left**: 向左旋转 90 度
c. **Inverted**: 上下翻转，相当于旋转 180 度

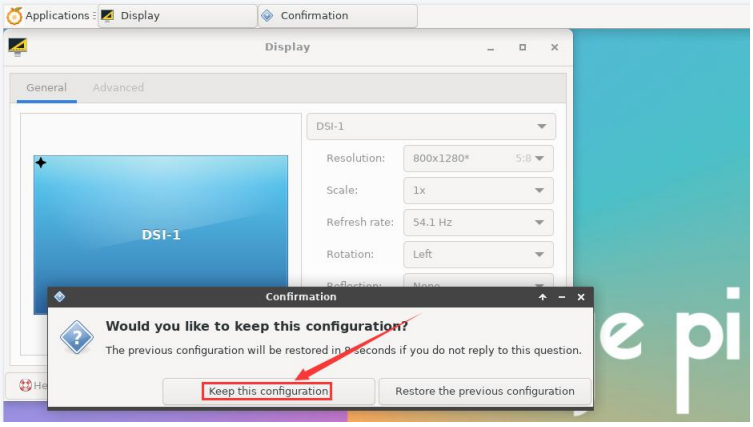
d. **Right**: 向右旋转 90 度



3) 然后点击 **Apply**。



4) 然后选择 **Keep this configuration**。



5) 此时屏幕显示就已旋转完成，然后关闭掉 **Display** 程序即可。

6) 上面的步骤只会选择显示方向，并不会旋转触摸的方向，使用 **set_lcd_rotate.sh** 脚本可以旋转下触摸的方向，此脚本设置完后会自动重启，然后就可以测试触摸是

否已经能正常使用了。

a. **None**: 不旋转

```
orangepi@orangepi:~$ set_lcd_rotate.sh none
```

b. **Left**: 向左旋转 90 度

```
orangepi@orangepi:~$ set_lcd_rotate.sh left
```

c. **Inverted**: 上下翻转，相当于旋转 180 度

```
orangepi@orangepi:~$ set_lcd_rotate.sh inverted
```

d. **Right**: 向右旋转 90 度

```
orangepi@orangepi:~$ set_lcd_rotate.sh right
```

set_lcd_rotate.sh 脚本主要做四件事：

1. **旋转 framebuffer 显示的方向**
2. **旋转触摸的方向**
3. **关闭开机 logo**
4. **重启系统**

旋转触摸的方向是通过在/usr/share/X11/xorg.conf.d/40-libinput.conf 中加入 Option "TransformationMatrix" "x x x x x x x x x"这行配置来实现的。其中"x x x x x x x x x"不同的方向配置不同。

7) 触摸旋转参考资料。

https://wiki.ubuntu.com/X/InputCoordinateTransformation

3.33. 开关机 logo 使用说明

1) 开关机 logo 默认只在桌面版的系统中才会显示。

2) 在**/boot/orangepiEnv.txt** 中设置 **bootlogo** 变量为 **false** 可以关闭开关机 logo。

```
orangepi@orangepi:~$ sudo vim /boot/orangepiEnv.txt
verbosity=1
bootlogo=false
```

3) 在**/boot/orangepiEnv.txt** 中设置 **bootlogo** 变量为 **true** 可以开启开关机 logo。

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

```
verbosity=1
bootlogo=true
```

4) 开机 logo 图片在 linux 系统中的位置为。

```
/usr/share/plymouth/themes/orangepi/watermark.png
```

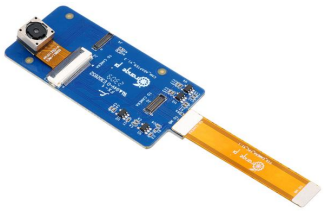
5) 替换开机 logo 图片后需要运行下命令才能生效。

```
orangepi@orangepi:~$ sudo update-initramfs -u
```

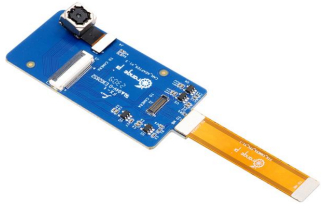
3.34. OV13850 和 OV13855 MIPI 摄像头的测试方法

目前开发板支持两款MIPI摄像头，OV13850 和OV13855，具体的图片如下所示：

a. 1300 万MIPI接口的OV13850 摄像头。



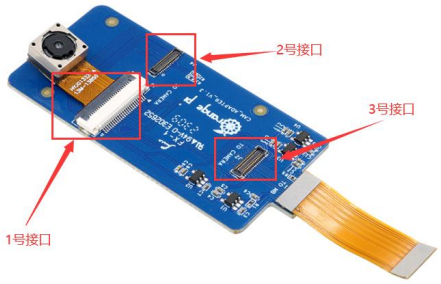
b. 1300 万MIPI接口的OV13855 摄像头。



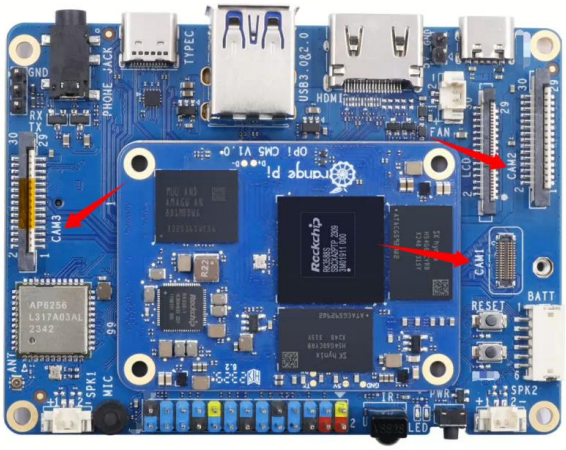
OV13850 和OV13855 摄像头使用的转接板和FPC排线是一样的，只是两款摄像头接在转接板上的位置不一样。FPC排线如下图所示，请注意FPC排线是有方向的，标注**TO MB**那端需要插到开发板的摄像头接口中，标注**TO CAMERA**那端需要插到摄像头转接板上。

摄像头转接板上总共有 3 个摄像头的接口，同一时间只能接一个使用，如下图所示，其中：

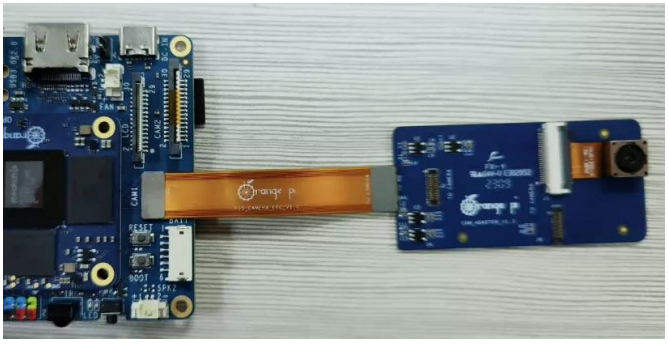
a. **1 号接口接 OV13850 摄像头**
b. **2 号接口接 OV13855 摄像头**
c. 3 号接口未使用，忽略即可



Orange Pi CM5 Base Tablet 开发板上总共有 3 个摄像头接口，只有 CAM1 可以用来接 OV13850 或 OV13855 摄像头。我们定义 Cam1、Cam2 和 Cam3 的位置如下图所示：



摄像头插在开发板的 Cam1 接口的方法如下所示：

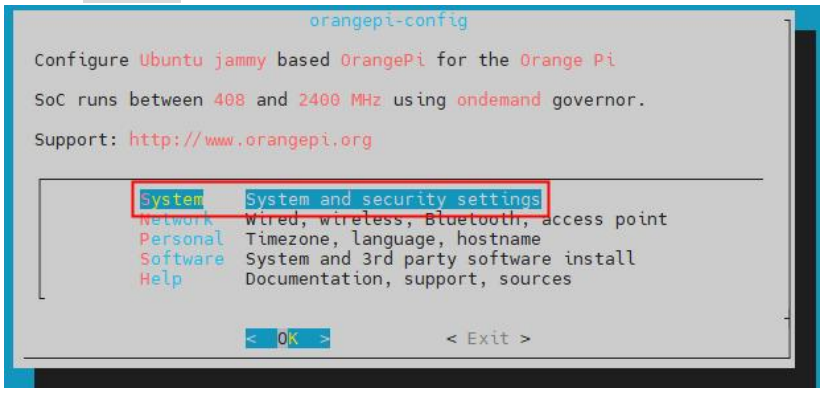


连接好摄像头到开发板上后，我们可以使用下面的方法来测试下摄像头：

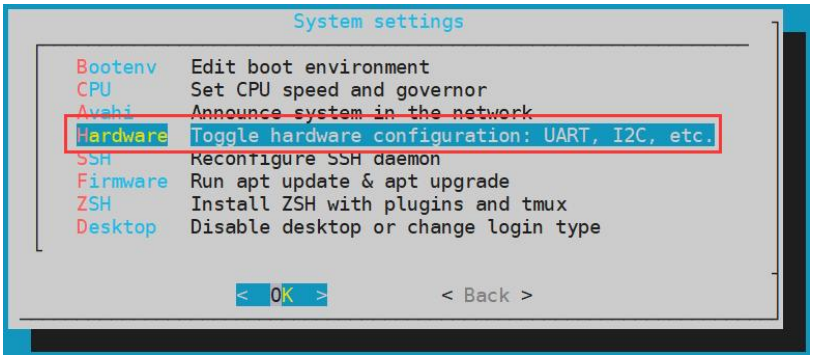
a. 首先运行下 **orangepi-config**，普通用户记得加 **sudo** 权限。

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

b. 然后选择 **System**。



c. 然后选择 **Hardware**。

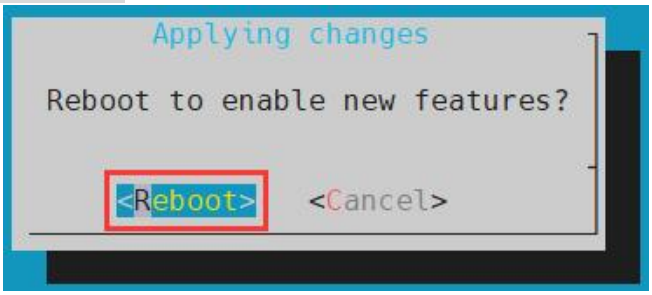


d. 然后使用使用键盘的方向键定位到下图所示的位置，再使用**空格**选中想要打开的摄像头，其中 **opicm5-tablet-cam1** 表示在开发板的 Cam1 接口中使用 ov13850 或 ov13855 摄像头。

e. 然后选择**<Save>**保存。

f. 然后选择**<Back>**。

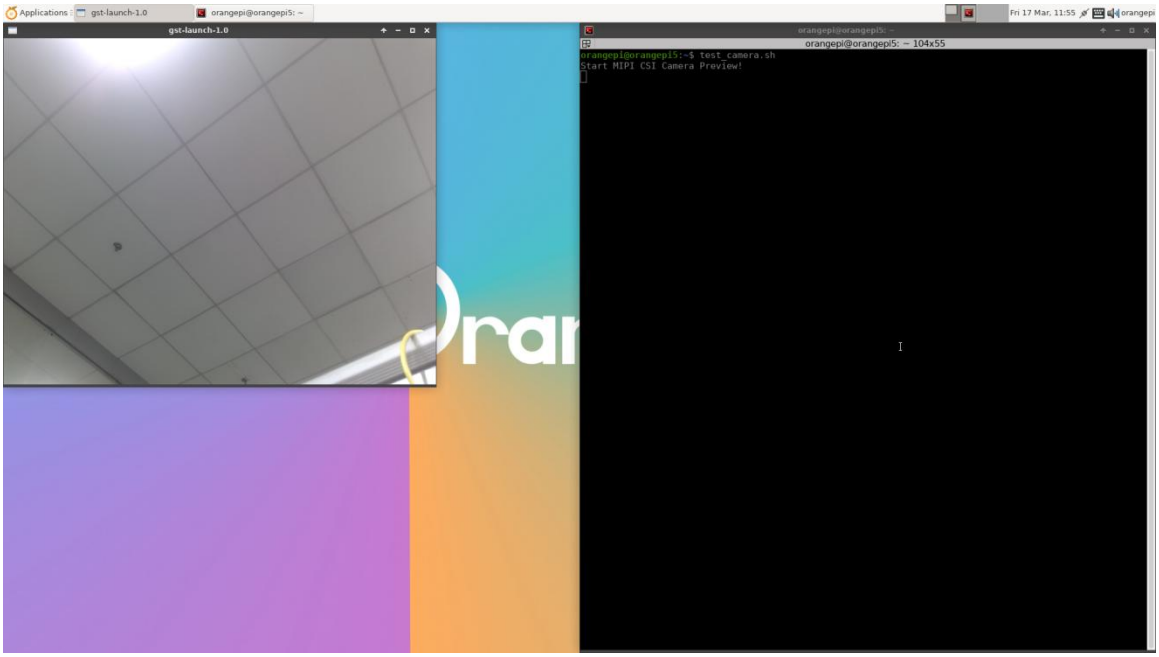
g. 然后选择**<Reboot>**重启系统使配置生效。



h. 然后在桌面系统中打开一个终端，再运行下面的脚本。

```
orangepi@orangepi:~$ test_camera.sh
```

i. 然后就能看到摄像头的预览画面了。



3.35. ZFS 文件系统的使用方法

3.35.1. 安装 ZFS 的方法

安装 zfs 前，请先确保使用的 linux 镜像为最新版本。另外，如果系统中已经安装了 zfs，就无需重复安装了。

安装 zfs 前首先需要安装内核头文件，安装内核头文件的方法请参考安装内核头文件的方法小节的说明。

在 Ubuntu20.04、Ubuntu22.04 和 Debian11 系统中，zfs 是无法通过 apt 直接安装

的，这是因为默认的 apt 源中 zfs 版本低于 2.1.6，存在和 rk linux5.10 内核不兼容的问题，这个问题在 zfs 的 2.1.6 及以后的版本中得到了修复。

为了解决这个问题，我们提供了能正常安装的 zfs 的 deb 包，可以从开发板的官方工具中下载到。打开**官方工具**，然后进入 **Ubuntu 和 Debian 系统使用的 zfs 相关的 deb 包**文件夹后，可以看到 Ubuntu20.04、Ubuntu22.04 和 Debian11 三种类型的 deb 包，请下载需要的版本。

下载完对应版本的 zfs deb 包后，请将它们上传到开发板的 Linux 系统中。上传方法请参考**上传文件到开发板 Linux 系统中的方法**小节的说明。

上传完成后，再在开发板 linux 系统的命令行中使用 **cd** 命令进入 deb 包的目录，然后使用下面的命令就可以安装 zfs 的 deb 包。

```
orangepi@orangepi:~$ sudo apt install ./*.deb
```

安装完成后，使用下面的命令可以看到 zfs 相关的内核模块：

```
orangepi@orangepi:~$ ls /lib/modules/5.10.160-rockchip-rk3588/updates/dkms/
icp.ko  spl.ko  zavl.ko  zcommon.ko  zfs.ko  zlua.ko  znvpair.ko  zunicode.ko  zzstd.ko
```

然后重启下 linux 系统就能看到 zfs 内核模块会自动加载了：

```
orangepi@orangepi:~$ lsmod | grep "zfs"
zfs                  2801664  0
zunicode              327680  1 zfs
zzstd                 471040  1 zfs
zlua                  139264  1 zfs
zcommon                69632  1 zfs
znvpair                61440  2 zfs,zcommon
zavl                   16384  1 zfs
icp                   221184  1 zfs
spl                    77824  6 zfs,icp,zzstd,znvpair,zcommon,zavl
```

在 Debian12 中，zfs 的默认版本为 2.1.11，所以我们可以通过下面的命令直接安装 zfs，再次提醒下，安装前需要确保系统已安装内核头文件的 deb 包。

```
orangepi@orangepi:~$ sudo apt install -y zfsutils-linux zfs-dkms
```

3.35.2. 创建 ZFS 池的方法

ZFS 是基于存储池的，我们可以将多个物理存储设备添加到池中，然后从这个池中分配存储空间。

下面的内容是基于开发板接了一个 NVMe SSD 和一个 U 盘来演示的。

1) 首先我们可以通过 **lsblk** 命令查看下开发板所有的存储设备，当前开发板接了一个 NVMe SSD 以及一个 U 盘，输出如下所示：

```
orangepi@orangepi:~$ lsblk
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sda           8:0    1  28.8G  0 disk
├─sda1        8:1    1  28.8G  0 part
└─sda9        8:9    1     8M  0 part
mtdblock0    31:0    0    16M  0 disk
mmcblk0     179:0    0  29.7G  0 disk
├─mmcblk0p1 179:1    0     1G  0 part /boot
└─mmcblk0p2 179:2    0  28.4G  0 part /var/log.hdd
                                      /
zram0       254:0    0   7.7G  0 disk [SWAP]
zram1       254:1    0   200M  0 disk /var/log
nvme0n1     259:0    0 476.9G  0 disk
├─nvme0n1p1 259:3    0 476.9G  0 part
└─nvme0n1p9 259:4    0     8M  0 part
orangepi@orangepi:~$
```

2) 然后输入下面的命令可以创建一个 ZFS 池，包含 NVMe SSD 和 U 盘两个存储设备。

```
orangepi@orangepi:~$ sudo zpool create -f pool1 /dev/nvme0n1 /dev/sda
```

3) 然后使用 **zpool list** 命令可以看到系统已经创建了一个名为 **pool1** 的 ZFS 池，并且 ZFS 池 pool1 的大小是 NVME SSD 的大小加上 U 盘的大小。

```
orangepi@orangepi:~$ zpool list
NAME    SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
pool1   504G   114K   504G        -         -     0%     0%  1.00x    ONLINE  -
```

4) 然后执行 **df -h** 可以看到 **pool1** 被挂载到了**/pool1** 目录。

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

```
tmpfs           1.6G   18M  1.6G   2% /run
/dev/mmcblk0p2   29G  6.0G   22G  22% /
tmpfs           7.7G   46M  7.7G   1% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           7.7G  944K  7.7G   1% /tmp
/dev/mmcblk0p1 1022M  115M  908M  12% /boot
/dev/zram1      188M  4.5M  169M   3% /var/log
tmpfs           1.6G   80K  1.6G   1% /run/user/1000
pool1           489G  9.3M  489G   1% /pool1
```

5) 使用下面的命令可以看到 pool1 的文件系统类型为 zfs。

```
orangepi@orangepi:~$ mount | grep pool1
pool1 on /pool1 type zfs (rw,xattr,noacl)
```

6) 然后我们可以测试下拷贝一个文件到 ZFS 池中。

```
orangepi@orangepi:~$ sudo cp -v /usr/local/test.mp4 /pool1/
'/usr/local/test.mp4' -> '/pool1/test.mp4'
```

3.35.3. 测试 ZFS 的数据去重功能

1) ZFS 的数据去重功能默认是关闭的，我们需要执行下面的命令打开。

```
orangepi@orangepi:~$ sudo zfs set dedup=on pool1
```

2) 然后做一个简单的测试，首先进入 pool1 中，再执行下面的命令生成 1 个 1G 大小的随机文件。

```
orangepi@orangepi:~$ cd /pool1/
root@orangepi:/pool1$ sudo dd if=/dev/urandom of=test.1g bs=1M count=1024
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB, 1.0 GiB) copied, 5.04367 s, 213 MB/s
```

3) 然后使用下面的命令将 1G 大小的随机文件拷贝 1000 份。

```
root@orangepi:/pool1$ for ((i=0; i<1000; i++)); do sudo cp test.1g $i.test.1g; done
```

4) 然后用 **du -lh** 可以看到目前池中总共有 1002G 的数据，但实际上 ZFS 池的大小只有 **504GB**（SSD+U 盘的总容量），是装不下那么大的数据的。

```
root@orangepi:/pool1$ du -lh
1002G
```

5) 然后用 **zpool list** 命令可以看到实际只占用了 1.01G，因为这 1001 个文件都是重复的，说明数据去重功能有效。

```
orangepi@orangepi:/pool1$ zpool list
NAME    SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
pool1   504G  1.01G   503G        -         -     0%     0%  6.00x    ONLINE  -
```

3.35.4. 测试 ZFS 的数据压缩功能

1) 因为存储的数据不同，压缩节省的磁盘空间也会有所不同，所以我们选择压缩比较大的纯文本文件来进行压缩测试，执行下面的命令将**/var/log/**和**/etc/**目录打包成 tar 包。

```
orangepi@orangepi:~$ cd /pool1/
root@orangepi:/pool1$ sudo tar -cf text.tar /var/log/ /etc/
```

2) 然后通过 **ls -lh** 命令可以看到的文件大小以及在 ZFS 池中占用的空间都是 **27M**。

```
orangepi@orangepi:/pool1$ ls -lh
total 27M
-rw-r--r-- 1 root root 27M Jun  1 14:46 text.tar
orangepi@orangepi:/pool1$ zpool list
NAME    SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
pool1   504G  26.7M   504G        -         -     0%     0%  1.00x    ONLINE  -
orangepi@orangepi:/pool1$
```

3) 然后我们在 ZFS 池 pool1 中启用压缩功能。

```
root@orangepi:/pool1$ sudo zfs set compression=lz4 pool1
```

4) 然后再次执行下面的命令将**/var/log/**和**/etc/**目录打包成 tar 包。

```
root@orangepi:/pool1$ sudo tar -cf text.tar /var/log/ /etc/
```

5) 这时可以看到 **text.tar** 文件大小还是 27M，但是在 ZFS 池中只占用 9.47M 的空间，说明文件被压缩了。

```
orangepi@orangepi:/pool1$ ls -lh
total 9.2M
-rw-r--r-- 1 root root 27M Jun  1 14:54 text.tar
orangepi@orangepi:/pool1$ zpool list
NAME    SIZE  ALLOC   FREE  CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP    HEALTH  ALTROOT
pool1   504G  9.47M   504G        -         -     0%     0%  1.00x    ONLINE  -
```

3.36. CasaOS 安装和使用方法

CasaOS 是一个基于 Docker 生态系统的开源家庭云系统，它可以让您在自己的开发板上运行各种各样的家庭应用程序，例如 NAS、家庭自动化、媒体服务器等。

3.36.1. CasaOS 的安装方法

1) 首先需要安装 docker，Orangepi Pi 发布的系统中已经预装了 docker，这步可以跳过，可以使用下面的命令查看安装的 docker 的版本。

```
orangepi@orangepi:~$ docker --version
Docker version 27.1.1, build 6312585          # Ubuntu Jammy 系统的输出
```

2) 然后在 linux 系统中输入下面的命令就可以开始 CasaOS 的安装。

```
orangepi@orangepi:~$ curl -fsSL https://get.casaos.io | sudo bash
```

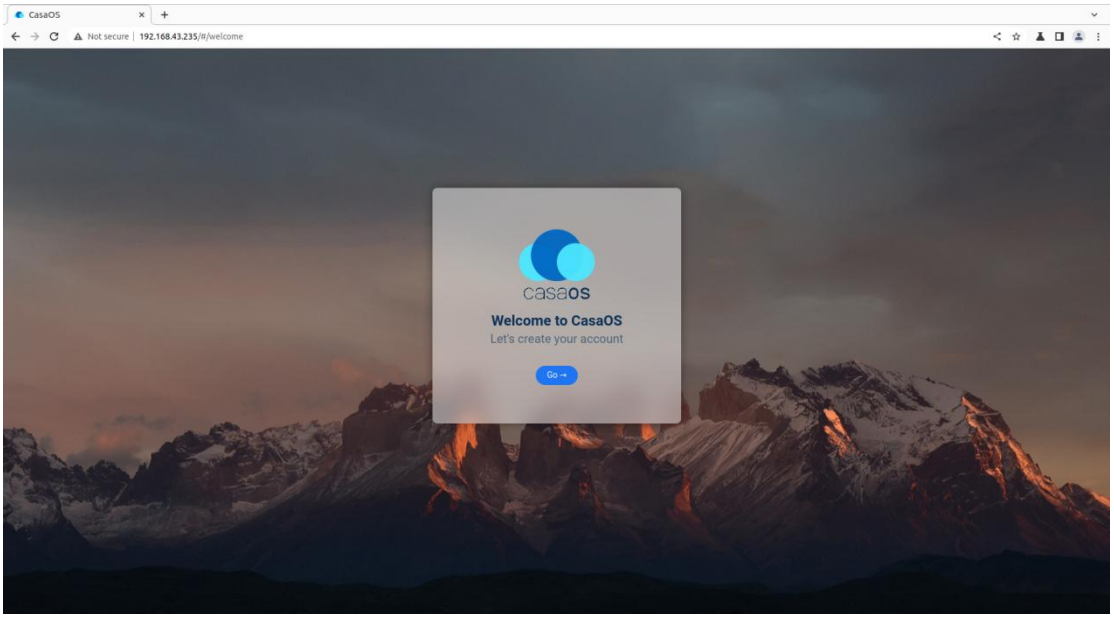
3) 当看到终端输出下面的打印信息时，说明 CasaOS 已经安装完成。

```
CasaOS v0.4.4.2 is running at:

Open your browser and visit the above address.


CasaOS Project  : https://github.com/IceWhaleTech/CasaOS
CasaOS Team     : https://github.com/IceWhaleTech/CasaOS#maintainers
CasaOS Discord  : https://discord.gg/knqAbbBbeX
Website         : https://www.casaos.io
Online Demo     : http://demo.casaos.io

Uninstall       : casaos-uninstall
```

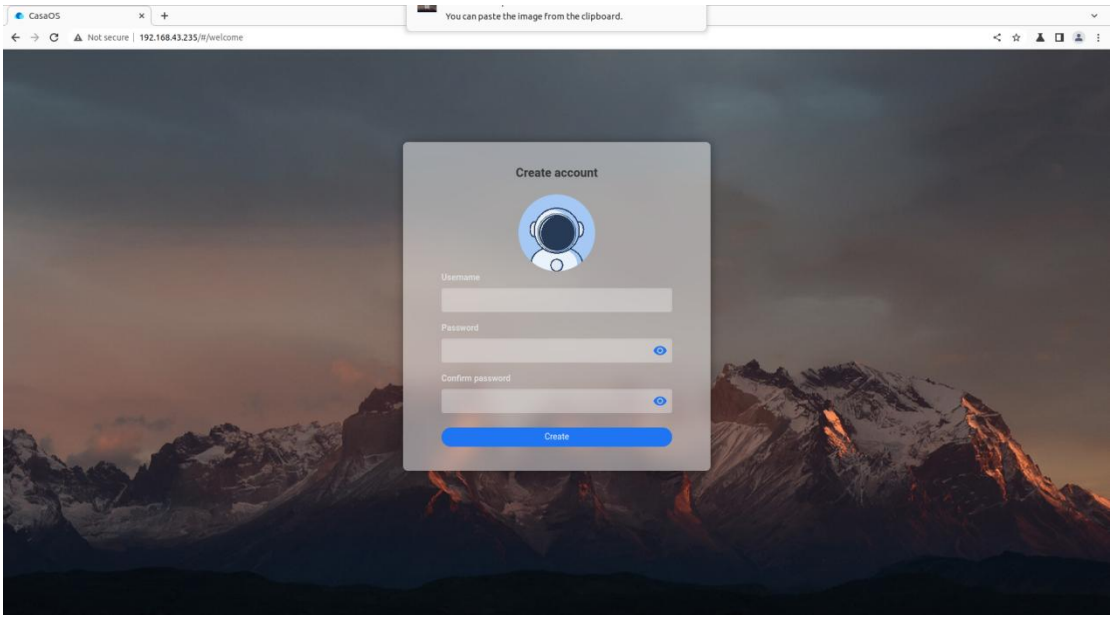
3.36.2. CasaOS 的使用方法

1) 安装完 CasaOS 后，在浏览器中输入 **http://开发板的 IP 地址**就可以打开 CasaOS。

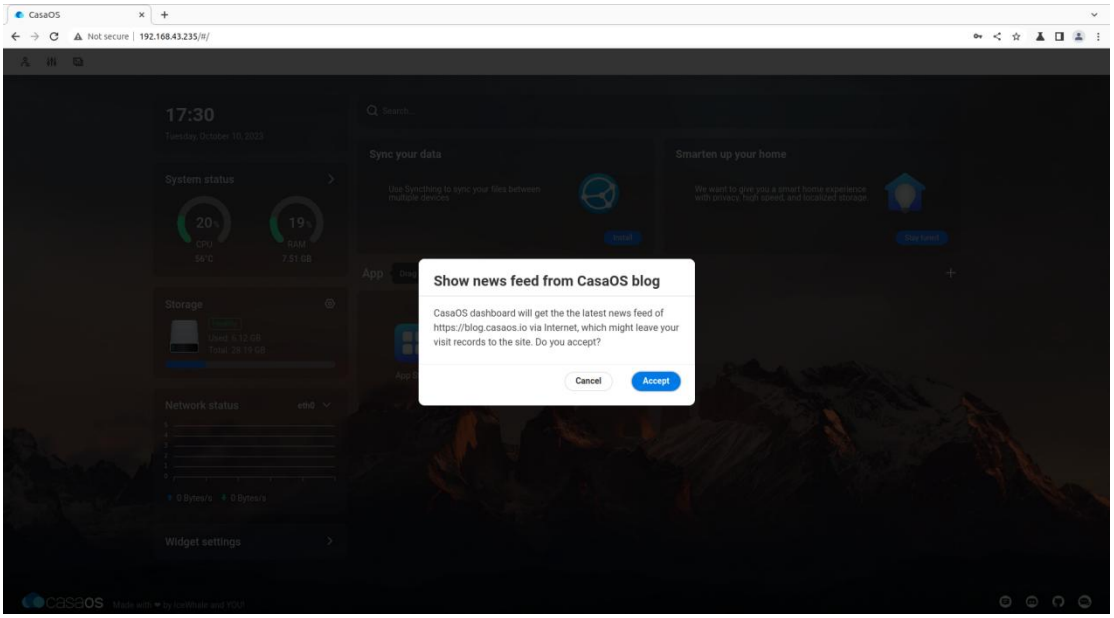
2) 打开 CasaO 后会弹出下面的欢迎界面，点击“Go”进入下一步。



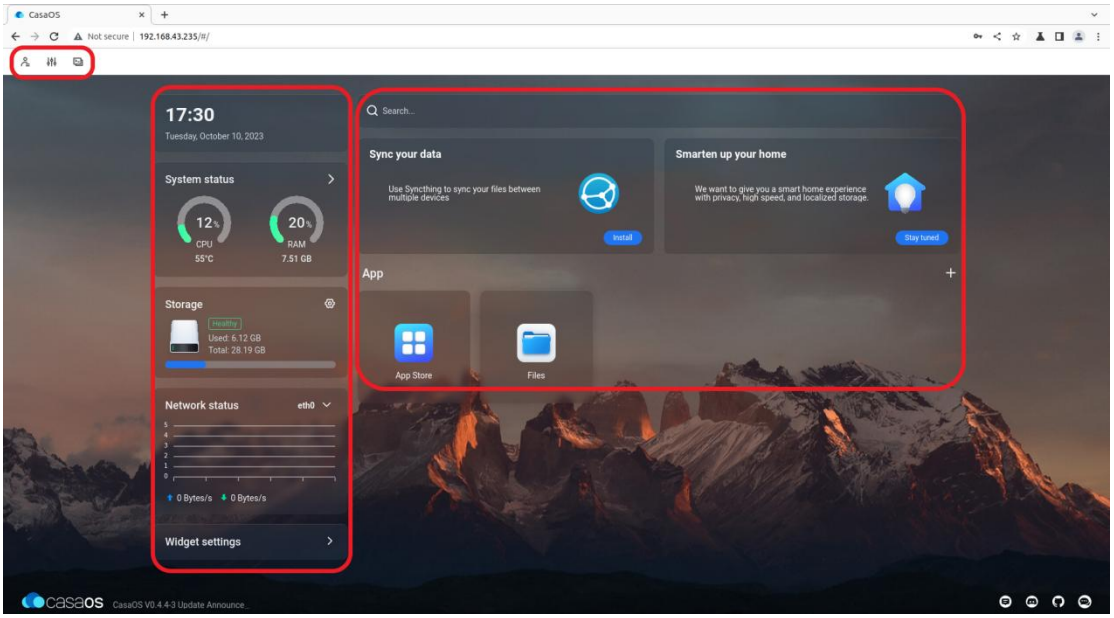
3) 当第一次登录 CasaOS 时，登录界面是设置账号密码的界面，以后再登录时，就只会出现输入账号密码的界面了，在设置好账号密码后，点击“Create”进入下一步。



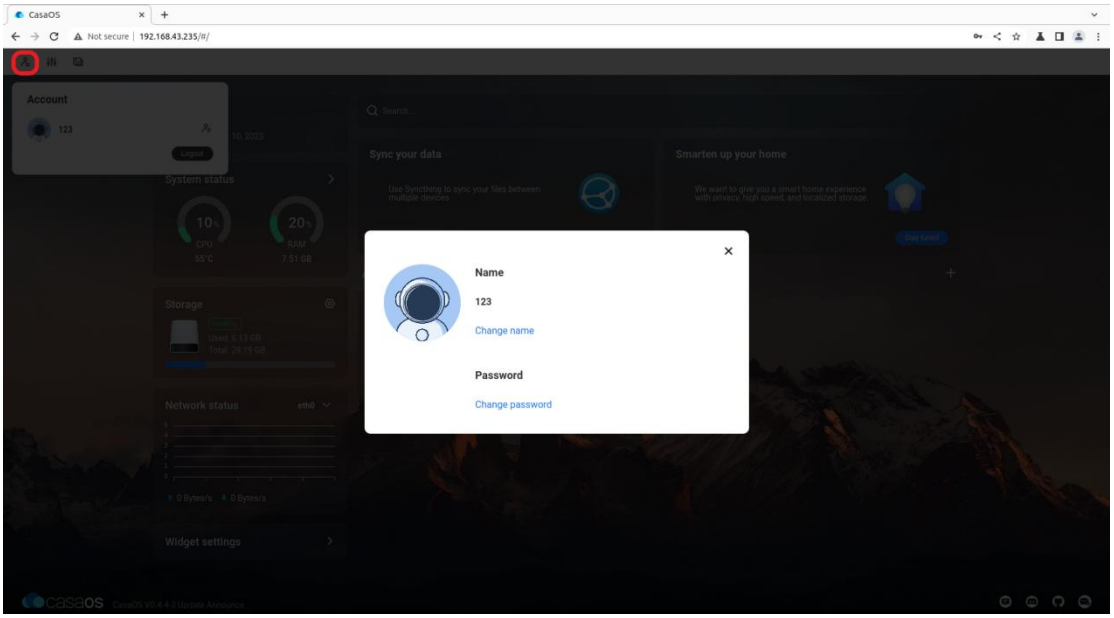
4) 在下面的界面中直接点击“Accept”进入下一步。



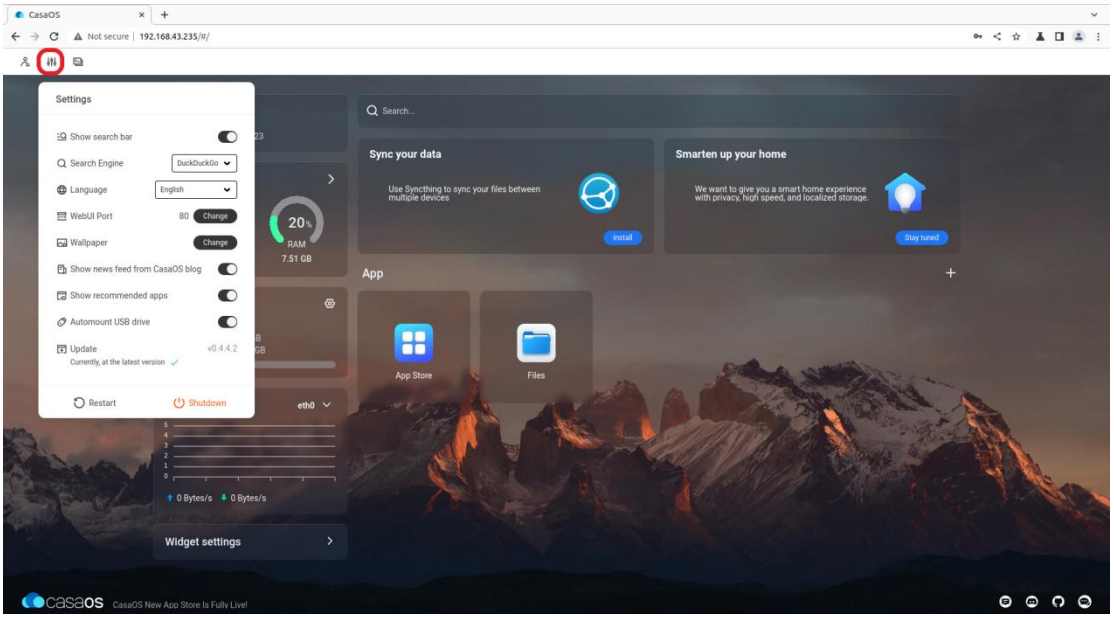
5) 现在就进入到 CasaOS 的主页面了，左上角有三个图标，可以进行功能设置，左边是性能面板，可以显示当前时间以及显示 CPU、RAM、存储、网络的状态信息，右边是功能面板，有搜索、应用推荐、应用商店和文件管理等功能。



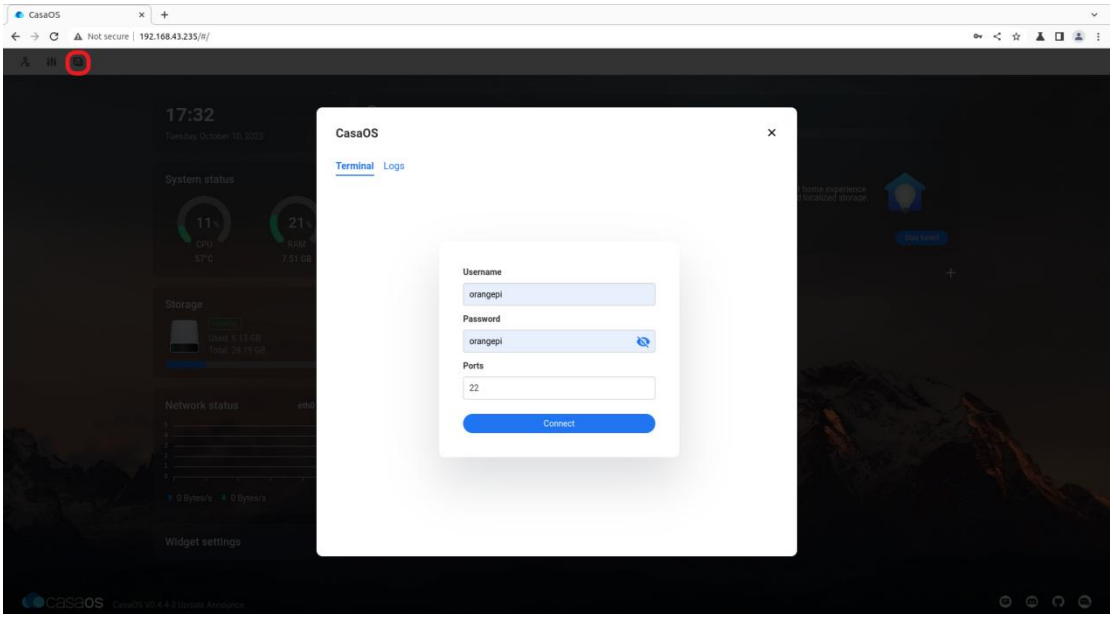
6) 可以点击左上角的第一个图标修改账号和密码。



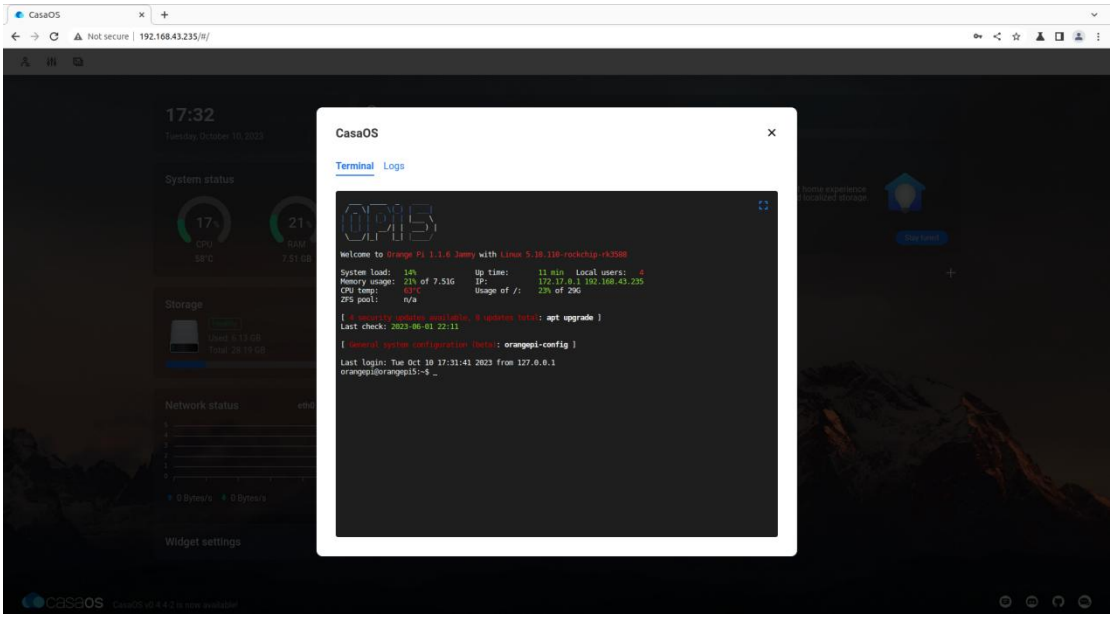
7) 可以点击第二个图标进行基本功能的设置。



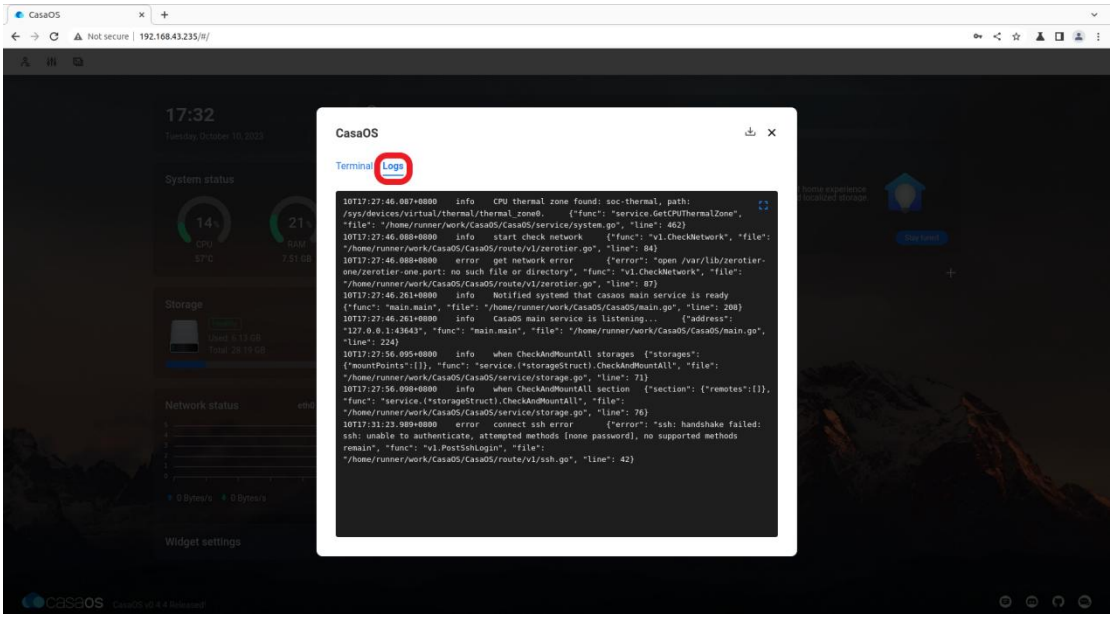
8) 左上角的第三个图标主要有两个功能，分别是切换到命令行模式和打印日志信息，在切换到命令行模式时，需要输入账号和密码，这里的账号和密码是指开发板 Linux 系统的账号和密码，端口系统默认选择 22 号。



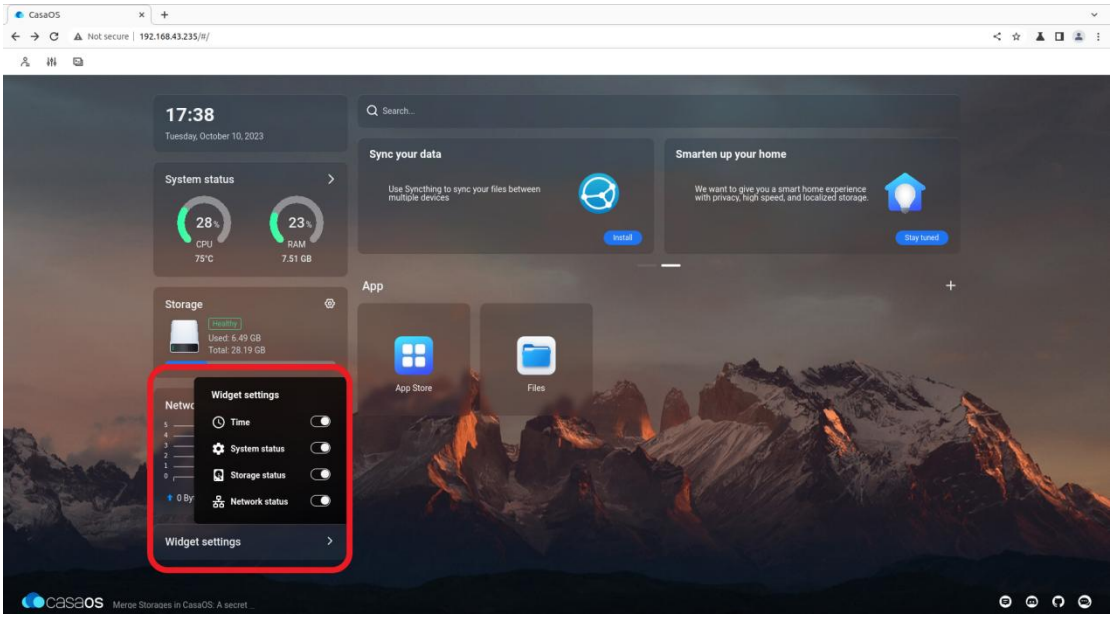
9) 然后点击“Connect”，就可以进入命令行界面：



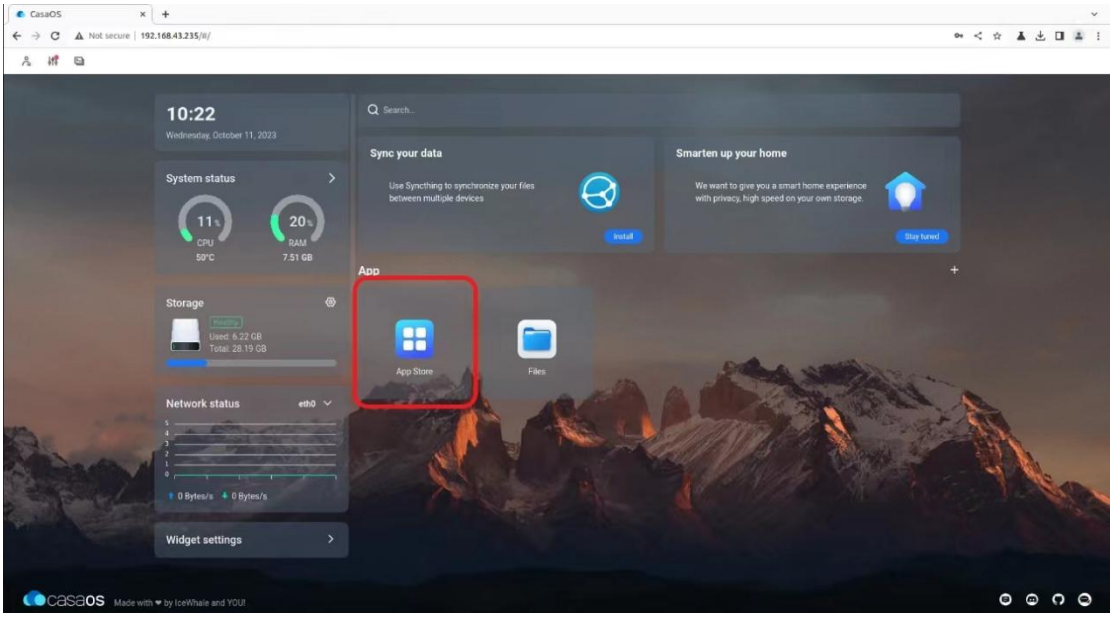
10) 在第三个图标下另一个功能是打印 CasaOS 的日志，点击“Logs”即可进入，界面如下所示：



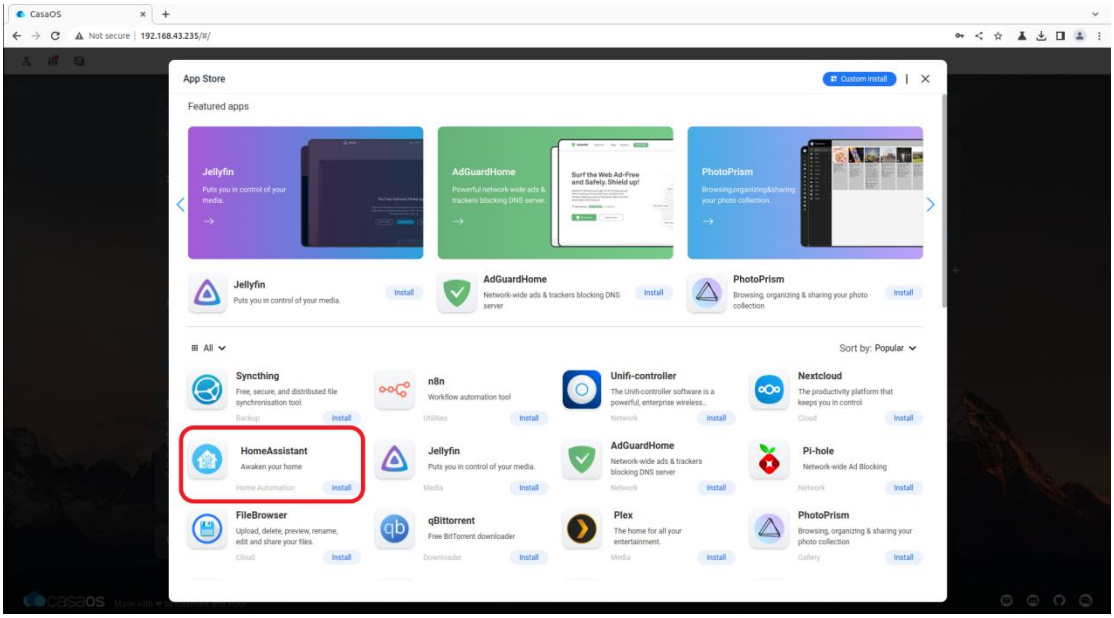
11) 点击左下角的“Widget settings”，可以设置在主页面是否显示性能面板的小部件。



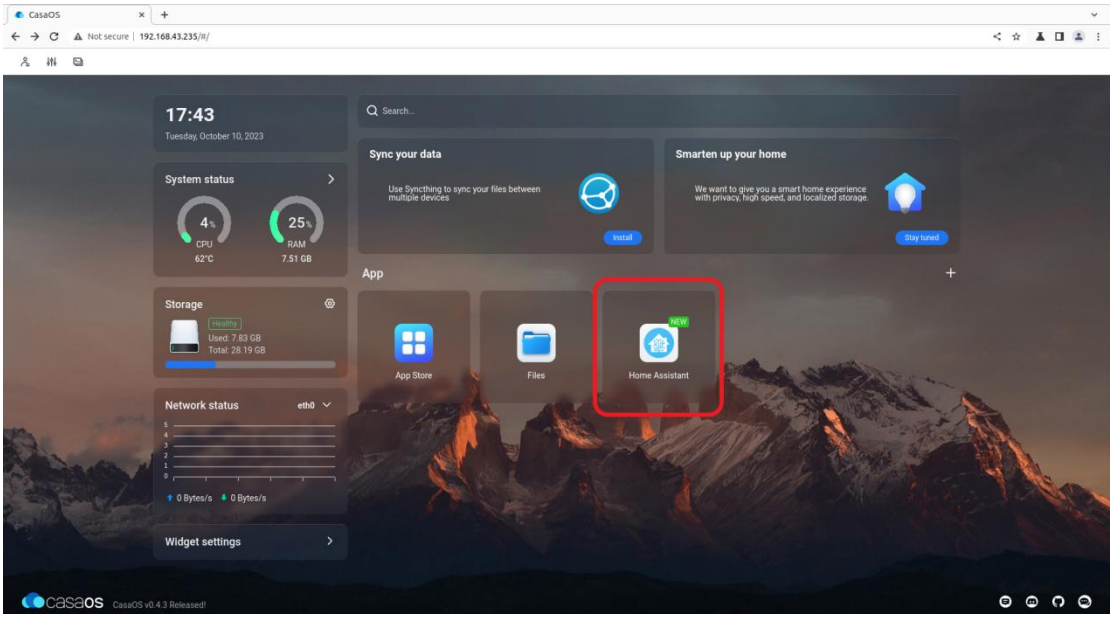
12) 点击主界面中“APP Store”可以打开应用商店，目前在应用商店中总共有 70+款 APP 可以使用。



13) 这里以 Home Assistant 为例进行下载，在 APP Store 中找到 Home Assistant，然后点击对应的“install”。

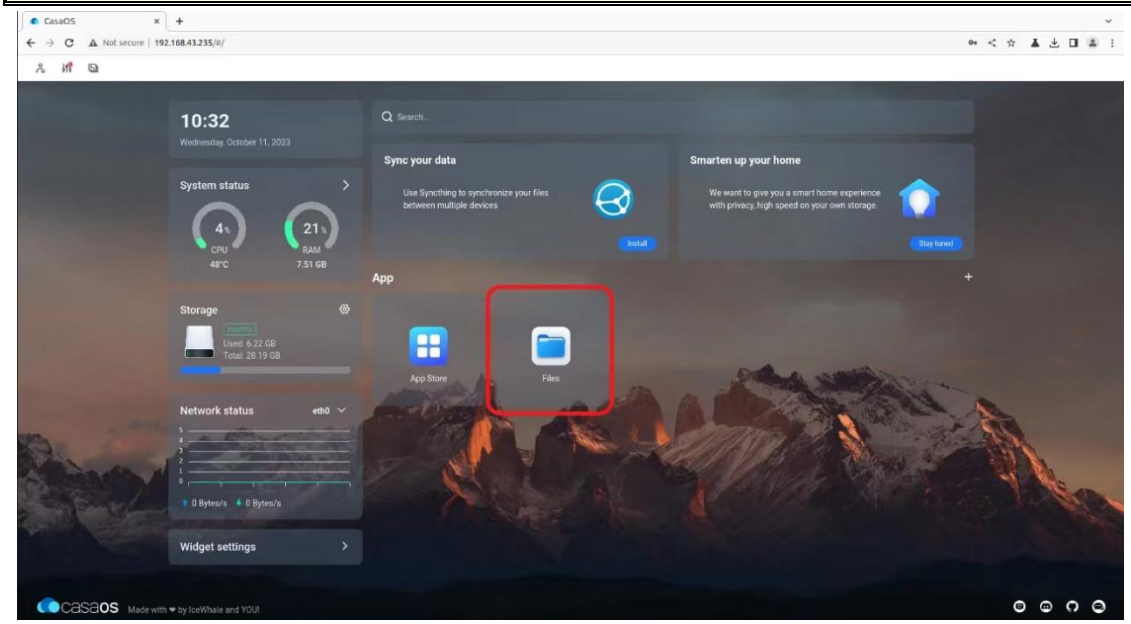


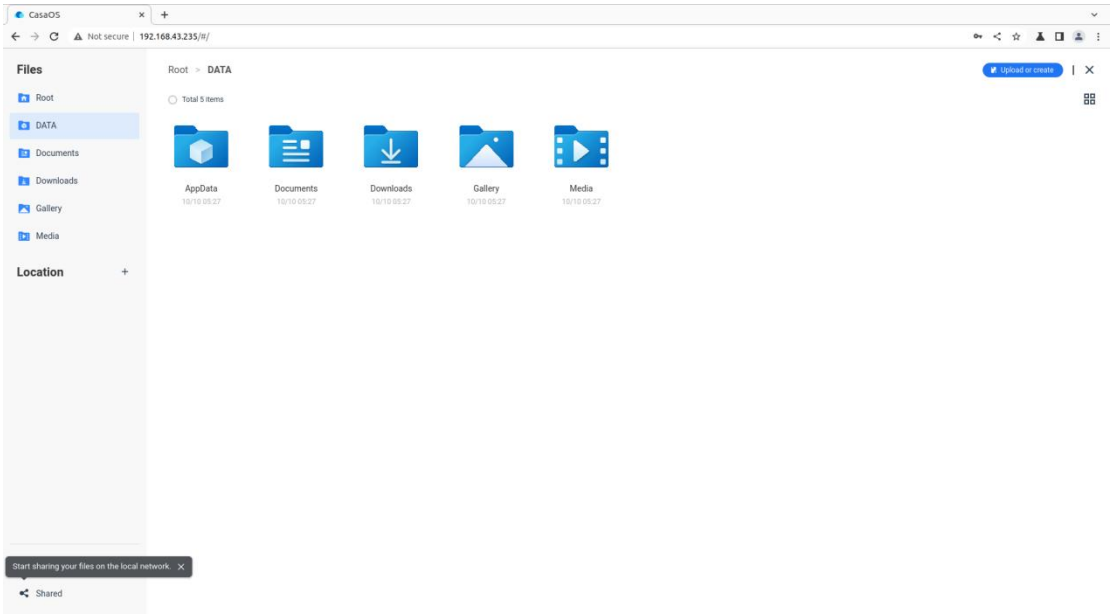
14) 下载完成后 HostAssitant 会出现在主页面中。



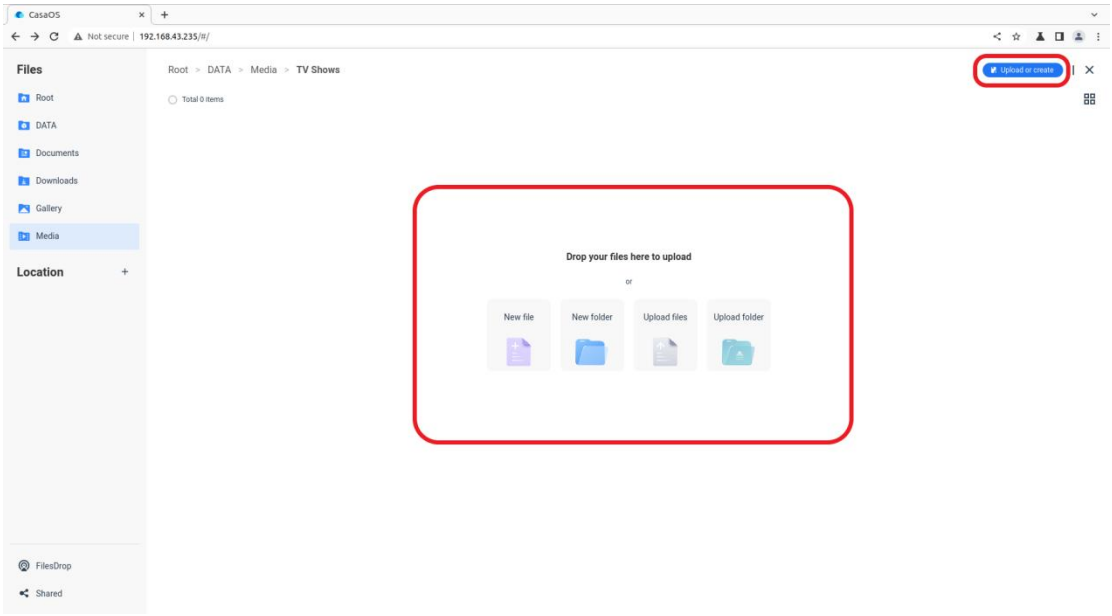
15) 点击主界面中的“Files”可以打开 CasaOS 自带的文件系统，然后就可以上传并保存文件。

请确保其他设备和开发板在同一局域网内。





16) 上传文件时需要切换到目标文件夹下，然后拖拽本地文件到图中指示区域，或者点击右上角的“Upload or Create”选择文件进行上传。



17) 如果想要卸载 CasaOS，可以使用下面的命令：

```
orangepi@orangepicm5-tablet:~$ casaos-uninstall
```

3.37. 使用 NPU 的方法

3.37.1. 准备工具

1) 一台装有 Ubuntu20.04 操作系统的 PC

根据 RKNN-Toolkit2 的官方文档，RKNN-Toolkit2 目前版本支持的操作系统如下所示：

a. Ubuntu18.04 (x64)

b. Ubuntu20.04 (x64)

c. Ubuntu22.04 (x64)

在本文档中我们使用 Ubuntu20.04 (x64)操作系统进行演示，其他版本的操作系统请自行测试。

2) 一块装有 Debian 11 系统的 Orange Pi 开发板

3) 一条 Type-C 接口的数据线，用于使用 adb 功能

3.37.2. 在 Ubuntu PC 端安装 RKNN-Toolkit2

Toolkit2 是一款在 Ubuntu PC 平台上使用的开发套件,用户使用该工具提供的 Python 接口可以便捷地完成模型转换、推理和性能评估等功能。

1) 在 Ubuntu PC 端，打开一个命令行窗口，然后输入以下命令来安装 python3 和 pip3

```
test@test:~$ sudo apt-get install python3 python3-dev python3-pip
```

2) 可以使用以下命令来查看已安装的 python3 的版本

```
test@test:~$ python3 --version
```

```
Python 3.8.10
```

3) 然后输入以下命令来安装 RKNN-Toolkit2 的依赖包

```
test@test:~$ sudo apt-get update
test@test:~$ sudo apt-get install libxslt1-dev zlib1g-dev libglib2.0 \
 libsm6 libgl1-mesa-glx libprotobuf-dev gcc
```

4) 然后输入以下命令来下载 1.5.2 版本的 RKNN-Toolkit2

```
test@test:~$ git clone https://github.com/airockchip/rknn-toolkit2 -b v1.5.2
```

5) 然后输入以下命令安装 python3 相应版本的依赖包，这个命令将使用 pip3 安装文件 requirements_cp38-1.5.2.txt 中列出的依赖项。如果依赖安装不全的话就不指定安装源单独安装里面的每个包。

```
test@test:~$ pip3 install -r rknn-toolkit2/doc/requirements_cp38-1.5.2.txt -i \
https://mirror.baidu.com/pypi/simple
```

6) 然后输入以下命令使用 pip3 来安装 RKNN-Toolkit2 软件包，安装完成后就可以使用 RKNN-Toolkit2 了

```
test@test:~$ pip3 install rknn-toolkit2/packages/rknn_toolkit2-1.5.2+b642f30c-cp38-cp38-linux_x86_64.whl
```

3.37.3. 使用 RKNN-Toolkit2 进行模型转换和模型推理

RKNN-Toolkit2 支持将 Caffe、TensorFlow、TensorFlow Lite、ONNX、Dark Net、PyTorch 等模型转为 RKNN 模型，然后在 Ubuntu PC 端通过仿真或使用开发板的 NPU 运行 RKNN 模型来进行推理。

在 RKNN-Toolkit2 的 example 文件夹中提供了相关的示例，以帮助用户更好地理解如何操作，我们以具有 yolov5 功能的 ONNX 模型为例进行说明。

3.37.3.1. 在 Ubuntu PC 端仿真运行模型

RKNN-Toolkit2 搭载了内置模拟器，可以让用户在 Ubuntu PC 端模拟模型在 Rockchip NPU 上的推理过程。

这样模型转换和推理均可以在 Ubuntu PC 端完成，从而帮助用户更快地测试和验证他们的模型。

1) 首先切换到 rknn-toolkit2/examples/onnx/yolov5 目录

```
test@test:~$ cd rknn-toolkit2/examples/onnx/yolov5/
```

2) 然后运行 test.py 脚本，该脚本首先将 yolov5s_relu.onnx 模型转换为可以在模拟器上运行的 RKNN 模型，然后使用模拟器仿真运行该模型对当前目录下的 bus.jpg 图片进行推理

```
test@test:~/rknn-toolkit2/examples/onnx/yolov5$ python3 test.py
```

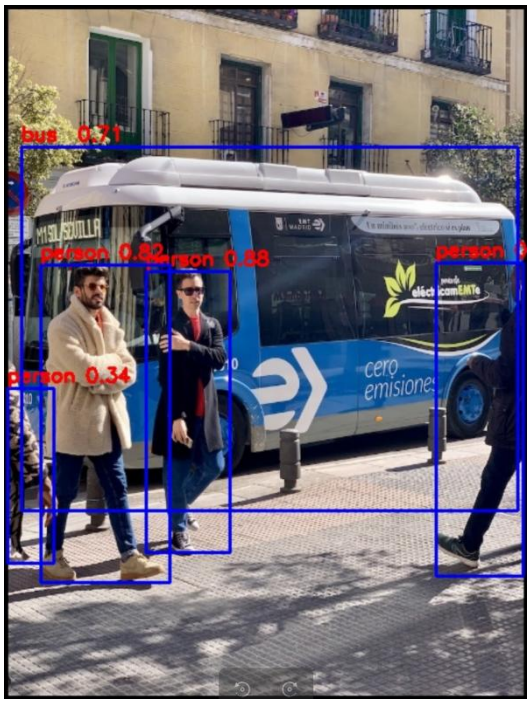
3) test.py 脚本成功运行后将看到如下打印信息,表明该模型在 bus.jpg 图片中成功检测到了四个人和一辆公交车

```
done
--> Running model
W inference: The 'data_format' has not been set and defaults is nhwc!
done
class: person, score: 0.884139358997345
box coordinate left,top,right,down: [209.1040009856224, 244.4304337501526, 286.5742521882057,
506.7466902732849]
class: person, score: 0.8676778078079224
box coordinate left,top,right,down: [478.5757632255554, 238.58572268486023, 559.5273861885071,
526.479279756546]
class: person, score: 0.8246847987174988
box coordinate left,top,right,down: [110.57257843017578, 238.58099019527435,
230.54625701904297, 534.0008579492569]
class: person, score: 0.3392542004585266
box coordinate left,top,right,down: [79.96397459506989, 354.9062474966049, 122.13020265102386,
516.2529321908951]
class: bus , score: 0.7012234926223755
box coordinate left,top,right,down: [94.43931484222412, 129.53470361232758, 553.1492471694946,
468.0852304697037]
D NPUTransfer: Transfer client closed, fd = 3
```

4) 转换得到的模型文件 yolov5s_relu.rknn 和推理的图片结果 result.jpg 保存在当前目录下

5) result.jpg 图片展示了使用 yolov5s_relu.rknn 模型在 bus.jpg 图片中检测到的物体

类别和置信率



3.37.3.2. 在 Ubuntu PC 端使用开发板的 NPU 运行模型

RKNN-Toolkit2 为用户提供了通过 adb 使用开发板的 NPU 进行推理的 Python 接口，可以让用户在 Ubuntu PC 端使用开发板的 NPU 来运行模型进行推理。

这样 Ubuntu PC 端可以根据模型在开发板的 NPU 上运行时的实际效果，使用 Python 提供的机器学习库，对模型进行优化和调整。

3.37.3.2.1. 使用 Type-C 数据线连接 adb

在 Ubuntu PC 端通过 adb 对开发板进行操作，adb 的使用方法请见 **ADB 的使用方法**一小节的说明

3.37.3.2.2. 更新开发板的 rknn_server 和 librknnrt.so

librknnrt.so 是一个板端的 runtime 库。

rknn_server 是一个运行在开发板上的后台代理服务，用于接收 PC 通过 USB

传输过来的协议，然后执行板端 runtime 库中对应的接口，并返回结果给 PC。

1) 首先在 Ubuntu PC 端输入以下命令下载 1.5.2 版本的 RKNPU2

```
test@test:~$ git clone https://github.com/rockchip-linux/rknpu2 -b v1.5.2
```

2) 然后在 Ubuntu PC 端输入以下命令通过 adb 工具更新开发板的 rknn_server

```
test@test:~$ adb push rknpu2/runtime/RK3588/Linux/rknn_server/aarch64/usr/bin/* /usr/bin
```

3) 然后在 Ubuntu PC 端输入以下命令通过 adb 工具更新开发板的 librknnrt.so 库

```
test@test:~$ adb push rknpu2/runtime/RK3588/Linux/librknn_api/aarch64/librknnrt.so /usr/lib
```

4) 通过 adb 工具打开开发板的终端

```
test@test:~$ adb shell
```

5) 打开开发板的 rknn_server 服务

```
root@orangepi:/# sudo restart_rknn.sh
root@orangepi:/# start rknn server,version:1.5.2(8babfeabuild@2023-08-25T10:30:31)
I NPUTransfer: Starting NPU TransferServer,Transfer version 2.1.0(b5861e7@2020-11-23T11:50:51)
```

6) 可以使用下面的命令检查下，如果出现 rknn_server 的进程 id，说明 rknn_server 已经打开，这样开发板的运行环境就搭建好了

```
root@orangepi:/# pgrep rknn_server
3131
```

3.37.3.2.3. 修改示例中的参数

1) 在 Ubuntu PC 端通过下面的命令可以查看到连接到 Ubuntu PC 的开发板的设备 ID，这个 ID 在下面会用到

```
test@test:~$ adb devices
List of devices attached
4f9f859e5a120324    device
```

2) 切换到 rknn-toolkit2/examples/onnx/yolov5 目录

```
test@test:~$ cd rknn-toolkit2/examples/onnx/yolov5/
```

3) 使用 vim 编辑器对 test.py 文件进行修改

```
test@test:~/rknn-toolkit2/examples/onnx/yolov5$ vim test.py
```

4) 在 test.py 文件中，我们需要对以下内容进行修改：

a. 在预处理配置中，将目标平台修改为 rk3588，这样模型转换后得到的是适用于 RK3588 开发板的 NPU 的 RKNN 模型

```
# pre-process config
print('--> Config model')
rknn.config(mean_values=[[0, 0, 0]], std_values=[[255, 255, 255]], target_platform='rk3588')
print('done')
```

b. 在初始化运行环境中，增加对目标平台和设备 ID 的说明，目标平台是 rk3588，设备 ID 是前面通过 adb 得到的开发板的设备 ID，运行模型进行推理的操作将会在 RK3588 开发板的 NPU 上进行

```
# Init runtime environment
print('--> Init runtime environment')
ret = rknn.init_runtime(target='rk3588',device_id='4f9f859e5a120324')
if ret != 0:
    print('Init runtime environment failed!')
    exit(ret)
print('done')
```

c. 修改完毕后，保存退出

3.37.3.2.4. 在 Ubuntu PC 端运行示例

1) 输入以下命令运行 test.py 脚本，该脚本首先转换 yolov5s_relu.onnx 模型为 RKNN 模型，然后加载该模型到开发板的 NPU 上来对当前目录下的 out.jpg 图片进行推理

```
test@test:~/rknn-toolkit2/examples/onnx/yolov5$ python3 test.py
```

2) 在打印信息中，我们可以看到 Ubuntu PC 通过 adb 工具使用开发板的 NPU 运行模型进行推理

```
--> Init runtime environment
I target set by user is: rk3588
```

```
I Check RK3588 board npu runtime version
I Starting ntp or adb, target is RK3588
I Device [4f9f859e5a120324] not found in ntb device list.
I Start adb...
I Connect to Device success!
I NPUTransfer: Starting NPU Transfer Client, Transfer version 2.1.0
(b5861e7@2020-11-23T11:50:36)
```

3) test.py 脚本成功运行后，转换得到的模型文件 yolov5s_relu.rknn 和推理的图片结果 result.jpg 保存在当前目录下

4) 运行的结果和**在 Ubuntu PC 端仿真运行模型**一小节是一样的

3.37.4. 调用 C 接口部署 RKNN 模型到开发板上运行

RKNPU2 为带有 Rockchip NPU 的芯片平台提供 C 编程接口，能够帮助用户部署使用 RKNN-Toolkit2 导出的 RKNN 模型，加速 AI 应用的落地。

在 RKNPU2 的 example 文件夹中，提供了将不同功能的 RKNN 模型部署到开发板的示例，我们以部署具有 yolov5 功能的 RKNN 模型到 RK3588 Debian 11 平台为例进行说明。

3.37.4.1. 下载交叉编译工具

由于开发板运行的是 Linux 系统，因此需要使用 gcc 交叉编译器来编译。推荐使用 gcc-9.3.0-x86_64_arrch64-linux-gnu 这个版本的 gcc

输入以下命令即可下载该版本的 gcc，下载后会得到一个名为 gcc-buildroot-9.3.0-2020.03-x86_64_aarch64-rockchip-linux-gnu 的文件夹

```
test@test:~$ git clone https://github.com/airockchip/gcc-buildroot-9.3.0-2020.03-x86_64_aarch64-rockchip-linux-gnu
```

3.37.4.2. 修改脚本中的编译工具路径

1) 切换到 rknpu2/examples/rknn_yolov5_demo 目录

```
test@test:~$ cd ~/rknpu2/examples/rknn_yolov5_demo
```

2) 使用 vim 编辑器修改 build-linux_RK3588.sh 文件中的内容、

```
test@test:~/rknpu2/examples/rknn_yolov5_demo$ vim build-linux_RK3588.sh
```

3) 在 build-linux_RK3588.sh 文件中，我们需要将变量 TOOL_CHAIN 的值更改为 gcc-buildroot-9.3.0-2020.03-x86_64_aarch64-rockchip-linux-gnu 文件夹的路径。这样，在运行 build-android_RK3588.sh 脚本时，会使用 gcc-buildroot-9.3.0-2020.03-x86_64_aarch64-rockchip-linux-gnu 文件夹中的交叉编译工具来进行编译

```
TARGET_SOC="rk3588"
GCC_COMPILER=aarch64-linux-gnu

export TOOL_CHAIN=~/gcc-buildroot-9.3.0-2020.03-x86_64_aarch64-rockchip-linux-gnu
export LD_LIBRARY_PATH=${TOOL_CHAIN}/lib64:$LD_LIBRARY_PATH
export CC=${GCC_COMPILER}-gcc
export CXX=${GCC_COMPILER}-g++
```

4) 修改完毕后，保存退出

3.37.4.3. 编译 rknn_yolov5_demo

1) 运行 build-linux_RK3588.sh，该脚本通过交叉编译生成了适用于 RK3588 开发板并能够在其上运行 RKNN 模型进行推理的程序

```
test@test:~/rknpu2/examples/rknn_yolov5_demo$ sudo apt install cmake
test@test:~/rknpu2/examples/rknn_yolov5_demo$ sudo apt-get install g++-aarch64-linux-gnu
test@test:~/rknpu2/examples/rknn_yolov5_demo$ ./build-linux_RK3588.sh
```

2) 运行 build-linux_RK3588.sh 完成后，在当前目录下会多出一个名为 install 的文件夹，该文件夹下的 rknn_yoov5_demo_Linux 文件夹中包含了通过交叉编译生成的程序以及与其相关的文件

```
test@test:~/rknpu2/examples/rknn_yolov5_demo$ ls install
rknn_yolov5_demo_Linux
```

3.37.4.4. 部署 rknn_yolov5_demo 到开发板

在 Ubuntu PC 端，可以使用以下命令通过 adb 工具将 rknn_yolov5_demo_Linux 文件夹上传至开发板中，从而实现 rknn_yolov5_demo 在开发板上的部署

```
test@test:~/rknpu2/examples/rknn_yolov5_demo$ adb push \
install/rknn_yolov5_demo_Linux /data/rknn_yolov5_demo_Linux
```

3.37.4.5. 在开发板上运行 rknn_yolov5_demo

1) 在 Ubuntu PC 端通过 adb shell 进入开发板的文件系统

```
test@test:~$ adb shell
root@orangepi:/#
```

2) 切换到 rknn_yolov5_demo_Linux 目录

```
root@orangepi:/# cd /data/rknn_yolov5_demo_Linux/
root@orangepi:/data/rknn_yolov5_demo_Linux# ls
lib  model  rknn_yolov5_demo  rknn_yolov5_video_demo
```

3) 然后运行 rknn_yolov5_demo 程序来进行推理，在下面的命令中该程序使用 yolov5s-640-640.rknn 模型对 bus.jpg 图片进行推理，整个运行过程将在开发板上完成

```
root@orangepi:/data/rknn_yolov5_demo_Linux# ./rknn_yolov5_demo \
./model/RK3588/yolov5s-640-640.rknn ./model/bus.jpg
```

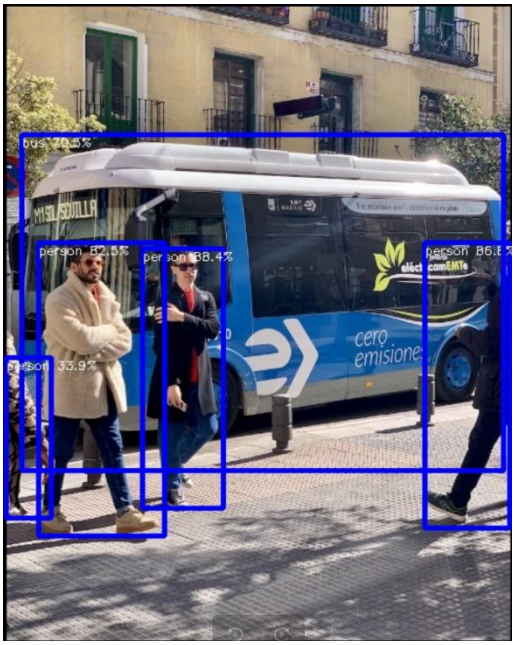
4) 运行完成后，推理的结果 out.jpg 图片保存在当前目录下

```
root@orangepi:/data/rknn_yolov5_demo_Linux# ls
lib  model  out.jpg  rknn_yolov5_demo  rknn_yolov5_video_demo
```

5) 在 Ubuntu PC 端，我们可以使用以下命令通过 adb 工具下载 out.jpg 图片，然后使用图像查看器进行查看

```
test@test:~$ adb pull /data/rknn_yolov5_demo_Linux/out.jpg ~/Desktop/
/data/rknn_yolov5_demo_Linux/out.jpg: ...led. 1.9 MB/s (191507 bytes in 0.095s)
```

6) out.jpg 图片展示了使用 yolov5s-640-640.rknn 模型在 bus.jpg 图片中检测到的物体类别和置信率



3.38. RK3588 使用百度飞桨的方法

在 rk3588 开发板上使用百度飞桨，包括在 PC 端转换 pdmodel 模型为 rknn 模型和在板端使用百度飞桨开发的 FastDeploy 部署工具部署 rknn 模型。以下内容是在 PC 端系统为 Ubuntu22.04，板端系统为 Debian 11 的环境下实现的，其他环境请自行测试。

3.38.1. Ubuntu PC 端环境搭建

在 Ubuntu PC 端需要安装的工具及用途如下所示

工具名	用途
Anaconda3	用于创建和管理 Python 环境
Paddle2ONNX	用于转换 pdmodel 模型为 ONNX 模型
RKNN-Toolkit2	用于转换 ONNX 模型为 RKNN 模型

3.38.1.1. PC 端安装 Anaconda3

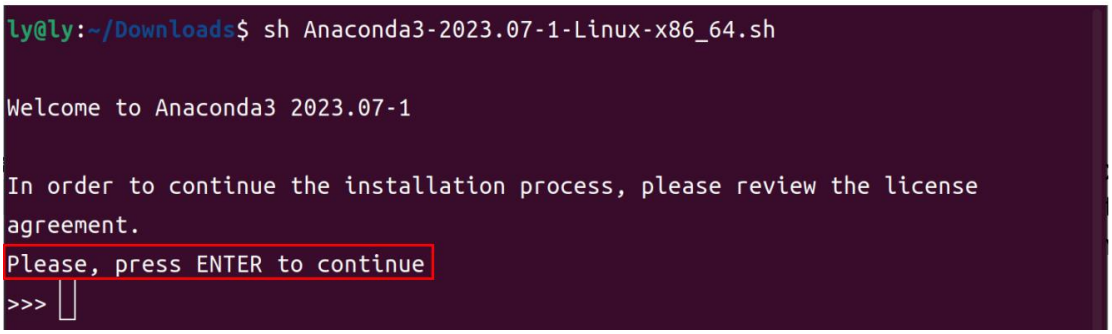
1）在 ubuntu PC 端打开浏览器，在地址栏输入下面的网址下载安装 Anaconda3 的脚本，下载完成后将得到 **Anaconda3-2023.07-1-Linux-x86_64.sh** 文件

https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/Anaconda3-2023.07-1-Linux-x86_64.sh

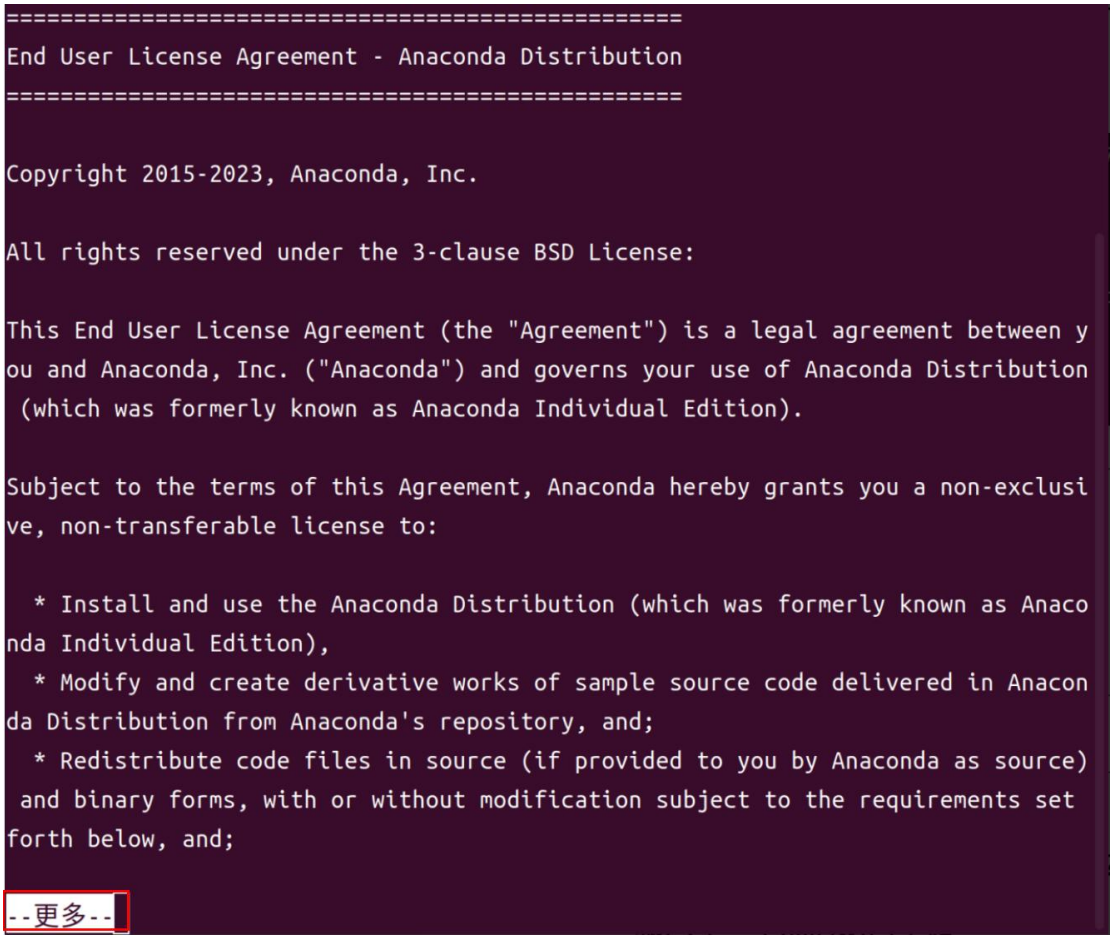
2）然后打开终端，运行 **Anaconda3-2023.07-1-Linux-x86_64.sh** 脚本安装 Anaconda3

```
test@test:~/Downloads$ sh Anaconda3-2023.07-1-Linux-x86_64.sh
```

3）然后安装脚本会输出下面的提示信息，此时点击回车键继续安装



4）在点击回车键后，会出现 Anaconda3 的一些介绍信息，一直点击“↓”键



5）然后安装脚本会提醒是否接受许可证条款，此时输入 yes 按回车键既可

```
The following packages listed on https://www.anaconda.com/cryptography are inclu
ded in the repository accessible through Anaconda Distribution that relate to cr
yptography.

Last updated February 25, 2022


Do you accept the license terms? [yes|no]
[no] >>>
```

6）然后安装脚本会提醒安装 Anaconda3 到家目录，此时按回车键确认

```
Anaconda3 will now be installed into this location:
/home/ly/anaconda3

  - Press ENTER to confirm the location
  - Press CTRL-C to abort the installation
  - Or specify a different location below

[/home/ly/anaconda3] >>>
```

7）然后安装脚本会提示是否初始化 Anaconda3 时，输入 yes，然后按回车键

```
installation finished.
Do you wish the installer to initialize Anaconda3
by running conda init? [yes|no]
[no] >>>
```

8）当看到终端中出现如下打印时，说明已经成功安装 Anaconda3 了

```
If you'd prefer that conda's base environment not be activated on startup,
   set the auto_activate_base parameter to false:

conda config --set auto_activate_base false

Thank you for installing Anaconda3!
```

3.38.1.2. PC 端安装 RKNN-Toolkit2

1）在 ubuntu PC 端打开终端，通过 Anaconda3 工具创建 python 版本为 3.8 的环境

```
(base)test@test:~$ conda create -n fastdeploy python=3.8
```

2）激活刚才创建的 python3.8 的环境

```
(base)test@test:~$ conda activate fastdeploy
```

3）然后安装 pip3 开发工具和包管理工具

```
(fastdeploy)test@test:~$ sudo apt-get install python3-dev python3-pip
```

4）然后安装 RKNN-Toolkit2 的依赖包

```
(fastdeploy)test@test:~$ sudo apt-get install libxslt1-dev zlib1g-dev libglib2.0 libsm6 libgl1-mesa-glx libprotobuf-dev gcc
```

5）rknn_toolkit2 对 numpy 存在特定依赖,因此需要先安装 numpy==1.16.6

```
(fastdeploy)test@test:~$ pip install numpy==1.16.6
```

6）安装 git 工具

```
(fastdeploy)test@test:~$ sudo apt install git
```

7）然后执行下面的命令下载 RKNN-Toolkit2，下载完成后会得到 rknn-toolkit2 文件夹

```
(fastdeploy)test@test:~$ git clone https://github.com/rockchip-linux/rknn-toolkit2
```

8）然后执行下面的命令，即可安装 python3.8 版本对应的 RKNN-Toolkit2 了

```
(fastdeploy)test@test:~$ pip install rknn-toolkit2/rknn-toolkit2/packages/rknn_toolkit2-1.6.0+81f21f4d-cp38-cp38-linux_x86_64.whl
```

3.38.1.3. PC 端安装 Paddle2ONNX

可以执行下面的命令安装 paddlepaddle 和 paddle2onnx

```
(fastdeploy)test@test:~$ pip install paddlepaddle
(fastdeploy)test@test:~$ pip install paddle2onnx
```

3.38.2. 板端环境搭建

在板端需要安装的工具及用途如下所示

工具名	用途

Anaconda3	用于创建和管理 Python 环境
rknpu2	rknpu2 的基础驱动
FastDeploy	编译后得到 FastDeploy 推理库

3.38.2.1. 板端安装 Anaconda3

1）在板端打开浏览器，在地址栏输入下面的网址下载安装 Anaconda3 的脚本，下载完成后将得到 **Anaconda3-2023.07-1-Linux-aarch64.sh** 文件

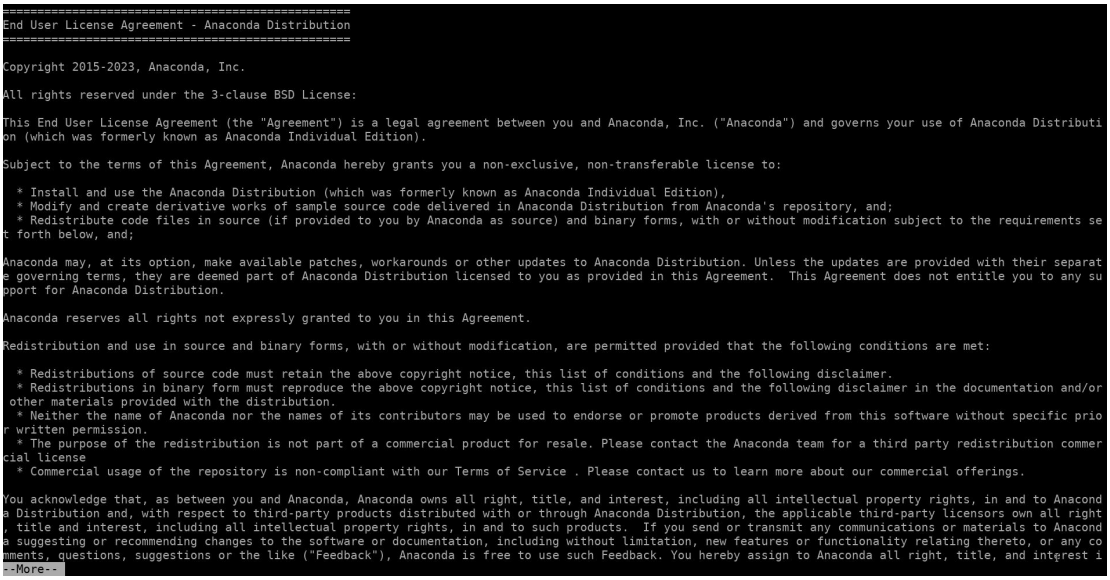
https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/Anaconda3-2023.07-1-Linux-aarch64.sh

2）打开终端，运行 **Anaconda3-2023.07-1-Linux-aarch64.sh** 脚本安装 Anaconda3

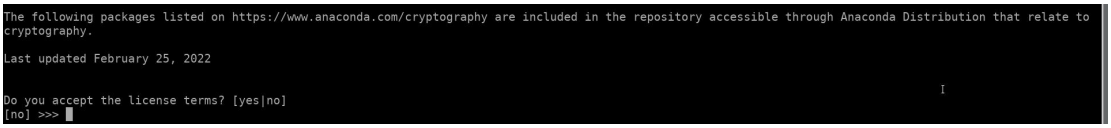
```
orangepi@orangepi:~/Downloads$ sh Anaconda3-2023.07-1-Linux-aarch64.sh
```

3）然后安装脚本会输出下面的提示信息，点击回车键继续安装

4）在点击回车键后，会出现 Anaconda3 的一些介绍信息，一直点击“↓”键



5）然后安装脚本会提醒是否接受许可证条款，此时输入 yes 按回车键既可



6）然后安装脚本会提醒安装 Anaconda3 到家目录，此时按回车键确认

7）然后安装脚本会提示是否初始化 Anaconda3 时，输入 yes，然后按回车键

8）当看到终端中出现如下打印时，说明已经成功安装 Anaconda3 了

9）如果在终端使用 conda 命令，显示命令不存在，需要修改~/.bashrc 文件

```
orangepi@orangepi:~$ vi ~/.bashrc
```

10）在~/.bashrc 文件末尾加上下面一句代码

```
export PATH=/home/orangepi/anaconda3/bin:$PATH
```

11）然后在终端中输入下面的命令让刚才的修改生效

```
orangepi@orangepi:~$ source ~/.bashrc
```

12）然后在终端中输入下面的命令进行 conda 的初始化

```
(base)orangepi@orangepi:~$ conda init bash
```

13）然后关闭当前终端，重新打开一个终端，此时就可以正常使用 conda 命令了

3.38.2.2. 板端安装 rknpu2 驱动

1）在板端打开终端，通过 Anaconda3 工具创建 python 版本为 3.9 的环境

```
(base)orangepi@orangepi:~$ conda create -n fastdeploy python=3.9
```

2）激活刚才创建的 python3.9 的环境

```
(base)orangepi@orangepi:~$ conda activate fastdeploy
```

3）通过 wget 下载 rknpu2_device_install_1.4.0.zip 文件

```
(fastdeploy)orangepi@orangepi:~$ wget https://bj.bcebos.com/fastdeploy/third_libs/rknpu2_device_install_1.4.0.zip
```

4）然后执行的下面的命令解压 rknpu2_device_install_1.4.0.zip, 解压后会得到 rknpu2_device_install_1.4.0 文件夹和__MACOSX 文件夹

```
(fastdeploy)orangepi@orangepi:~$ unzip rknpu2_device_install_1.4.0.zip
```

5）切换到 rknpu2_device_install_1.4.0 目录下

```
(fastdeploy)orangepi@orangepi:~$ cd rknpu2_device_install_1.4.0/
```

6）在该目录下有 rknn_install_rk3588.sh 脚本，运行该脚本，即可完成板端 rknpu2 驱动的安装

```
(fastdeploy)orangepi@orangepi:~/rknpu2_device_install_1.4.0$ sudo bash rknn_install_rk3588.sh
```

3.38.2.3. 板端编译 FastDeploy C++ SDK

1）在编译时需要用到 cmake 命令，可以执行下面的命令安装 cmake 工具

```
(fastdeploy)orangepi@orangepi:~$ sudo apt-get install -y cmake
```

2）然后下载 FastDeploy SDK，命令执行完成后会得到 FastDeploy 文件夹

```
(fastdeploy)orangepi@orangepi:~$ git clone https://github.com/PaddlePaddle/FastDeploy.git
```

3）切换到 FastDeploy 目录

```
(fastdeploy)orangepi@orangepi:~$ cd FastDeploy
```

4）创建编译目录 build 并切换 build 目录下

```
(fastdeploy)orangepi@orangepi:~/FastDeploy$ mkdir build && cd build
```

5）在编译前需要使用 cmake 来配置需要编译的项目信息，执行完下面的命令后，当前目录下会多出一些文件，其中就有用来编译的 Makefile 文件

```
(fastdeploy)orangepi@orangepi:~/FastDeploy/build$ cmake .. -DENABLE_ORT_BACKEND=ON \
-DENABLE_RKNPU2_BACKEND=ON \
-DENABLE_VISION=ON \
-DRKNN2_TARGET_SOC=RK3588 \
-DCMAKE_INSTALL_PREFIX=${PWD}/fastdeploy-0.0.3
```

6）执行下面的命令开始编译

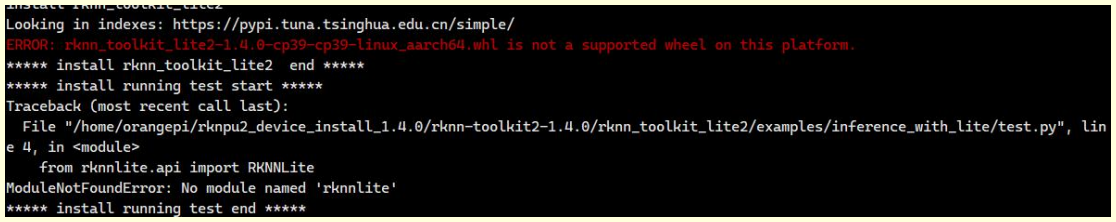
```
(fastdeploy)orangepi@orangepi:~/FastDeploy/build$ make -j8
```

7）编译完成后，使用下面的命令将编译得到的文件安装到指定的路径

```
(fastdeploy)orangepi@orangepi:~/FastDeploy/build$ make install
```

8）编译完成后主要得到了 fastdeploy-0.0.3 文件夹，在该文件夹中，有配置环境变量的脚本文件 fastdeploy_init.sh，使用该脚本配置环境变量后就可以使用编译得到的一些库文件了

有可能会出现以下图片中的报错，忽略即可，不影响后面的操作




```
(fastdeploy)orangepi@orangepi:~/FastDeploy/build$ source fastdeploy-0.0.3/fastdeploy_init.sh
```

3.38.3. 使用 FastDeploy 部署模型示例

ResNet50_vd 模型是一种用来进行目标分类的模型，下面以 ResNet50_vd 模型为例说明使用 FastDeploy 部署 pdmodel 模型的流程

3.38.3.1. Ubuntu PC 端模型转换

1）在 PC 端打开终端，激活之前使用 Anaconda3 创建的 python3.8 的环境

```
test@test:~$ conda activate fastdeploy
```

2）在模型转换的脚本中，需要导入 yaml 模块和 six 模块，可以执行下面的命令进行安装

```
(fastdeploy)test@test:~$ pip install pyyaml six
```

3）执行下面的命令，可以下载得到 ResNet50_vd_infer.tgz 文件

```
(fastdeploy)test@test:~$ wget https://bj.bcebos.com/paddlehub/fastdeploy/ResNet50_vd_infer.tgz
```

4）解压 ResNet50_vd_infer.tgz 文件后可以得到 ResNet50_vd_infer 文件夹，在这个文件夹中包含 pdmodel 模型文件 inference.pdmodel 以及其他相关的文件

```
(fastdeploy)test@test:~$ tar -xvf ResNet50_vd_infer.tgz
```

5）可以使用下面的命令通过 paddle2onnx 将 pdmodel 模型转换为 onnx 模型,执行完该命令后在 ResNet50_vd_infer 文件夹中会多出转换得到的 onnx 模型文件 ResNet50_vd_infer.onnx

```
(fastdeploy)test@test:~$ paddle2onnx --model_dir ResNet50_vd_infer \
--model_filename inference.pdmodel \
--params_filename inference.pdiparams \
--save_file ResNet50_vd_infer/ResNet50_vd_infer.onnx \
--opset_version 10 \
--enable_onnx_checker True
```

6）然后再使用下面的命令固定 shape 为[1,3,224,224]，执行完命令后，会对 ResNet50_vd_infer.onnx 文件进行修改

```
(fastdeploy)test@test:~$ python -m paddle2onnx.optimize --input_model \
ResNet50_vd_infer/ResNet50_vd_infer.onnx \
--output_model ResNet50_vd_infer/ResNet50_vd_infer.onnx \
--input_shape_dict "{'inputs':[1,3,224,224]}"
```

7）onnx 模型转换为 rknn 模型需要使用 FastDeploy SDK 中的脚本，执行下面的命令下载 FastDeploy

```
(fastdeploy)test@test:~$ git clone https://github.com/PaddlePaddle/FastDeploy.git
```

8）然后转移 ResNet50_vd_infer 文件夹到 FastDeploy 的对应目录下

```
(fastdeploy)test@test:~$ mv ResNet50_vd_infer \
FastDeploy/examples/vision/classification/paddleclas/rockchip/rknpu2/
```

9）切换到进行模型转换的目录下

```
(fastdeploy)test@test:~$ cd FastDeploy/examples/vision/classification/paddleclas/rockchip/rknpu2/
```

10）执行下面的命令，就可以将 onnx 模型转换为 rknn 模型了，最终在 ResNet50_vd_infer 目录下得到了 rknn 模型文件 ResNet50_vd_infer_rk3588_unquantized.rknn

```
(fastdeploy)test@test:~/FastDeploy/examples/vision/classification/paddleclas/rockchip/rknpu2/$ python ./rknpu2_tools/export.py \
--config_path ./rknpu2_tools/config/ResNet50_vd_infer_rknn.yaml \
--target_platform rk3588
```

11）在板端部署时，使用的 rknn 模型文件名称为 ResNet50_vd_infer_rk3588.rknn，所以需要将 ResNet50_vd_infer_rk3588_unquantized.rknn 文件改名为 ResNet50_vd_infer_rk3588.rknn

```
(fastdeploy)test@test:~/FastDeploy/examples/vision/classification/paddleclas/rockchip/rknpu2/$ mv ResNet50_vd_infer/ResNet50_vd_infer_rk3588_unquantized.rknn \
ResNet50_vd_infer/ResNet50_vd_infer_rk3588.rknn
```

3.38.3.2. 板端模型部署

1）在板端打开终端，激活之前使用 Anaconda3 创建的 python3.9 的环境

```
orangepi@orangepi:~$ conda activate fastdeploy
```

2）运行 fastdeploy_init.sh 脚本配置环境

```
(fastdeploy)orangepi@orangepi:~$ source FastDeploy/build/fastdeploy-0.0.3/fastdeploy_init.sh
```

3）切换到 FastDeploy 中部署 ResNet50 模型的示例目录

```
(fastdeploy)orangepi@orangepi:~$ cd FastDeploy/examples/vision/classification/paddleclas/rockchip/rknpu2/cpp
```

4）在该目录下创建目录结构

```
(fastdeploy)orangepi@orangepi:~/FastDeploy/examples/vision/classification/paddleclas/rockchip/rknpu2/cpp$ mkdir build images ppclas_model_dir thirdpartys
```

5）将编译得到的 fastdeploy-0.0.3 文件夹拷贝到 thirdpartys 文件夹

```
(fastdeploy)orangepi@orangepi:~/FastDeploy/examples/vision/classification/paddleclas/rockchip/rknpu2/cpp$ cp -r ~/FastDeploy/build/fastdeploy-0.0.3/ thirdpartys
```

6）将 PC 端中的 ResNet50_vd_infer 文件夹中的文件拷贝到 ppclas_model_dir 目录下

7）切换到 images 目录

```
(fastdeploy)orangepi@orangepi:~/FastDeploy/examples/vision/classification/paddleclas/rockchip/rknpu2/cpp$ cd images
```

8）通过 wget 在 images 目录下载测试图片

```
(fastdeploy)orangepi@orangepi:~/FastDeploy/examples/vision/classification/paddleclas/rockchip/rknpu2/cpp/images$ wget https://gitee.com/paddlepaddle/PaddleClas/raw/release/2.4/deploy/images/ImageNet/ILSVRC2012_val_00000010.jpeg
```

9）然后切换到编译目录 build 下

```
(fastdeploy)orangepi@orangepi:~/FastDeploy/examples/vision/classification/paddleclas/rockchip/rknpu2/cpp/images$ cd ../build/
```

10）使用 cmake 配置需要编译的内容，在执行完该命令后，当前目录下会出现一些文件，其中包括 Makefile 文件

```
(fastdeploy)orangepi@orangepi:~/FastDeploy/examples/vision/classification/paddleclas/rockchip/rknpu2/cpp/build$ cmake ..
```

11）执行下面的命令开始编译

```
(fastdeploy)orangepi@orangepi:~/FastDeploy/examples/vision/classification/paddleclas/rockchip/rknpu2/cpp/build$ make -j8
```

12）执行下面的命令将编译得到的文件安装到指定的路径，在执行完该命令后，在当前目录下会多出 install 目录

```
(fastdeploy)orangepi@orangepi:~/FastDeploy/examples/vision/classification/paddleclas/rockchip/rknpu2/cpp/build$ make install
```

13）切换到 install 目录，使用模型进行推理就是在这里完成的

```
(fastdeploy)orangepi@orangepi:~/FastDeploy/examples/vision/classification/paddleclas/rockchip/rknpu2/cpp/build$ cd install
```

14）使用下面的命令，便可以使用转换得到的 rknn 模型对 ILSVRC2012_val_00000010.jpeg 图片中的内容进行分类

```
(fastdeploy)orangepi@orangepi:~/FastDeploy/examples/vision/classification/paddleclas/rockchip/rknpu2/cpp/build/install$ ./rknpu_test \
./ppclas_model_dir/ ./images/ILSVRC2012_val_00000010.jpeg
```

15）在执行完该命令后，回显信息中会出现如下打印，表示图片中物体的类别 ID 号为 644，置信率为 0.072998

```
ClassifyResult(
label_ids: 644,
scores: 0.072998,
```

)

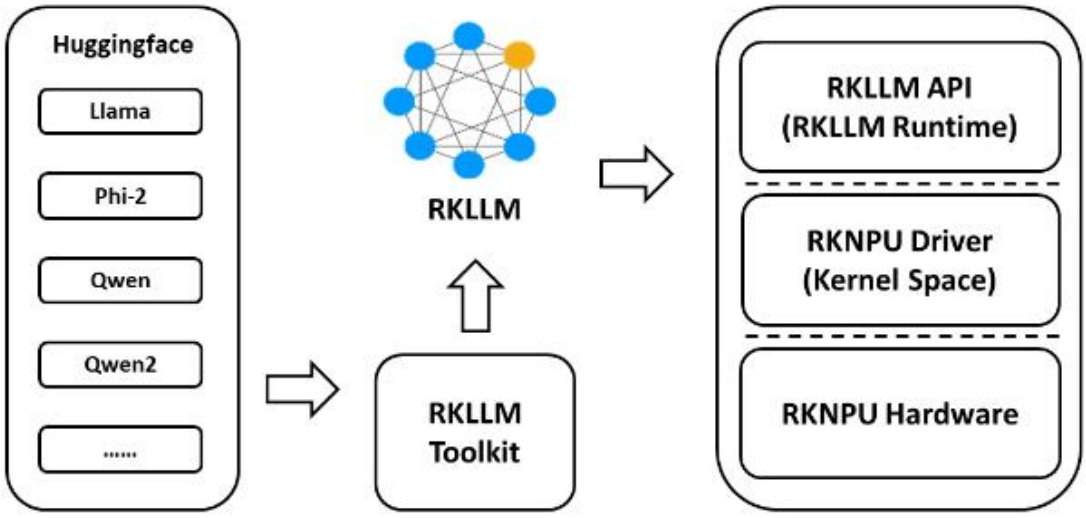
3.39. RK3588 运行 RKLLM 大模型的方法

本小节中所使用的代码和模型都可以在开发板的官方工具中下载到。

3.39.1. RKLLM 介绍

更详细的 RKLLM 介绍资料可以参考瑞芯微 RKLLM 官方资料。

RKLLM 可以帮助用户快速将 LLM 模型部署到 RK3588 开发板中，整体框架如下图所示：



3.39.1.1. RKLLM 工具链介绍

3.39.1.1.1. RKLLM-Toolkit 功能介绍

RKLLM-Toolkit 是为用户提供在计算机上进行大语言模型的量化、转换的开发套件。通过该工具提供的 Python 接口可以便捷地完成以下功能：

1) 模型转换：支持将 Hugging Face 格式的大语言模型（Large Language Model, LLM）转换为 RKLLM 模型，目前我们测试能够运行的模型包括 TinyLLAMA、Qwen、Qwen2、Phi-3、ChatGLM3、Gemma、InternLM2 和 MiniCPM，转换后的 RKLLM

模型能够在 RK3588 平台上加载使用。

2) 量化功能：支持将浮点模型量化为定点模型，目前支持的量化类型为 w8a8，w8a8 表示权重和激活都被量化为 8 位宽度。

3.39.1.1.2. RKLLM Runtime 功能介绍

RKLLM Runtime 主要负责加载 RKLLM-Toolkit 转换得到的 RKLLM 模型，并在 RK3588 板端通过调用 NPU 驱动在 RK3588 NPU 上实现 RKLLM 模型的推理。在推理 RKLLM 模型时，用户可以自行定义 RKLLM 模型的推理参数设置，定义不同的文本生成方式，并通过预先定义的回调函数不断获得模型的推理结果。更详细的说明可以参考**瑞芯微 RKLLM 官方资料**。

3.39.1.2. RKLLM 开发流程介绍

RKLLM 的整体开发步骤主要分为 2 个部分：模型转换和板端部署运行。

1) **在 Ubuntu PC 端进行模型转换**。在这一阶段，用户提供的 Hugging Face 格式的大语言模型将会被转换为 RKLLM 格式，以便在 RK3588 开发板上进行高效的推理。这一步骤包括：

a. 搭建 RKLLM-Toolkit 环境：在 Ubuntu PC 端用 Conda 搭建 RKLLM-Toolkit 的运行环境。

b. 模型转换：用 RKLLM-Toolkit 对获取的 Hugging Face 格式的大语言模型，或是自行训练得到的大语言模型（注意，模型保存的结构要与 Hugging Face 平台上的模型结构一致。）转换成能在 RK3588 开发板上运行的.rkllm 格式文件。

c. 编译测试代码：用 rkllm-runtime 编译能运行在 RK3588 开发板上的推理程序。

在 Ubuntu PC 端进行模型转换的具体开发流程请见 **Ubuntu PC 端进行模型转换和编译源码的详细步骤**小节的说明。

2) **在开发板端部署运行**。这个阶段涵盖了模型在 RK3588 开发板上的的实际部

署和运行。它通常包括以下步骤：

a. 升级内核 NPU 版本：将开发板内核的 NPU 版本升级到 v0.9.6。

b. 模型推理：将在 Ubuntu PC 端上用 rkllm-runtime 编译出来的推理程序和用 RKLLM-Toolkit 转换的.rkllm 格式文件放在开发板上进行模型推理。可以在开发板上直接运行推理，具体开发流程请见本章的**开发板端部署运行的详细步骤**小节。也可以在开发板上进行板端 Server 服务的部署，在同一网段下的 Ubuntu PC 就可以通过访问对应地址来调用 RKLLM 模型进行推理，具体开发流程请见本章的**开发板端 Server 的部署运行的详细步骤**小节。

以上这两个步骤构成了完整的 RKLLM 开发流程，确保大语言模型能够成功转换、调试，并最终在 RK3588 NPU 上实现高效部署。

3.39.2. 准备工具

1) 一台装有 Ubuntu 22.04 操作系统的 PC。**在本文档中我们使用 Ubuntu22.04(x64) 操作系统进行演示，其他版本的操作系统请自行测试。**

2) 一块 RK3588 开发板。

3.39.3. Ubuntu PC 端进行模型转换和编译源码的详细步骤

3.39.3.1. 搭建 RKLLM-Toolkit 环境

1) 首先下载 RKLLM 工具链。

```
test@test:~$ git clone https://github.com/airockchip/rknn-llm.git
```

2) 下载后用 ls 命令检查下载的文件是否无误

```
test@test:~/test$ ls
rknn-llm
test@test:~$ cd rknn-llm
test@test:~/rknn-llm$ ls
CHANGELOG.md  doc  LICENSE  README.md  res  rkllm-runtime
rkllm-toolkit  rknpu-driver
```

3) rknn-llm 中具体的文件目录如下所示：

```
test@test:~/rknn-llm$ sudo apt install tree
test@test:~/rknn-llm$ tree
doc
└──Rockchip_RKLLM_SDK_CN.pdf       # RKLLM SDK 说明文档

rkllm-runtime
├──examples
│    └── rkllm_api_demo   #  板端推理调用示例工程
│   └── rkllm_server_demo # RKLLM-Server 部署示例工程
├──runtime
│    └── Android
│        └── librkllm_api
│            └──arm64-v8a
│                └── librkllmrt.so # RKLLM Runtime 库
│            └──include
│                └── rkllm.h              # Runtime 头文件
│    └── Linux
│        └── librkllm_api
│            └──aarch64
│                └── librkllmrt.so # RKLLM Runtime 库
│            └──include
│                └── rkllm.h     # Runtime 头文件
rkllm-toolkit
├──examples
│    └── huggingface
│        └── test.py
├──packages
│    └── md5sum.txt
│    └── rkllm_toolkit-x.x.x-cp38-cp38-linux_x86_64.whl
rknpu-driver
└──rknpu_driver_0.9.6_20240322.tar.bz2
```

4) 然后下载安装 miniforge3 安装包。

```
test@test:~$ wget -c https://mirrors.bfsu.edu.cn/github-release/conda-forge/miniforge/LatestRelease/Miniforge3-Linux-x86_64.sh
test@test:~$ chmod 777 Miniforge3-Linux-x86_64.sh
```

```
test@test:~$ bash Miniforge3-Linux-x86_64.sh
```

镜像网站有时候会崩溃，导致 miniforge3 包下载不了，在开发板的官方工具中已经给出了下载好的 miniforge3 安装包。

运行 bash Miniforge3-Linux-x86_64.sh 时所有的选项直接按 Enter 就可以了。

5) 然后进入 Conda base 环境。

```
test@test:~$ source ~/miniforge3/bin/activate
(base) test@test:~$
```

6) 然后创建一个 Python3.8 版本（建议版本）名为 RKLLM-Toolkit 的 Conda 环境。

```
(base) test@test:~$ conda create -n RKLLM-Toolkit python=3.8
```

7) 然后进入 RKLLM-Toolkit Conda 环境。

```
(base) test@test:~$ conda activate RKLLM-Toolkit
(RKLLM-Toolkit) test@test:~$
```

8) 然后使用 pip 命令安装之前下载的 RKLLM 工具链中的 whl 包，目录为：**rknn-llm/rkllm-toolkit/packages/ rkllm_toolkit-1.0.1-cp38-cp38-linux_x86_64.whl**。在安装过程中，安装工具会自动下载 RKLLM-Toolkit 工具所需要的相关依赖包。

```
(base) test@test:~$ pip3 install rknn-llm/rkllm-toolkit/packages/rkllm_toolkit-1.0.1-cp38-cp38-linux_x86_64.whl
```

9) 最后执行以下命令如果没有报错，则说明安装成功。

```
(RKLLM-Toolkit) test@test:~$ python
>>> from rkllm.api import RKLLM
```

3.39.3.2. 模型转换

在此部分，我们提供了八种模型转换的示例供用户选择。若用户在从 Hugging Face 下载模型时遇到网络问题，我们的开发板官方工具中已经集成了下载的模型文件以及相应的.rkllm 转换文件。

3.39.3.2.1. 转换 TinyLLAMA 模型

1) 先在 Ubuntu 操作系统上安装 Git LFS，如果已安装可跳过这一步。

```
(RKLLM-Toolkit) test@test:~$ sudo apt update
(RKLLM-Toolkit) test@test:~$ sudo apt install curl git
(RKLLM-Toolkit) test@test:~$ curl -s https://packagecloud.io/install/repositories/github/git-lfs/script.deb.sh | sudo bash
(RKLLM-Toolkit) test@test:~$ sudo apt install git-lfs
(RKLLM-Toolkit) test@test:~$ git lfs install
```

2) 接着下载 TinyLLAMA 模型。

```
(RKLLM-Toolkit) test@test:~$ git clone https://huggingface.co/TinyLlama/TinyLlama-1.1B-Chat-v1.0
```

3) 修改 rknn-llm/rkllm-toolkit/examples/huggingface/test.py 里面的 modelpath 变量的值为下载的 TinyLlama-1.1B-Chat-v1.0 文件夹所在的绝对路径，然后修改 ret = llm.export_rkllm("./qwen.rkllm")括号中的值为要保存的.rkllm 格式文件路径，我们将其修改为 ret = llm.export_rkllm("./TinyLlama.rkllm")。

```
(RKLLM-Toolkit) test@test:~$ vim rknn-llm/rkllm-toolkit/examples/huggingface/test.py
modelpath = "/path/your/TinyLlama-1.1B-Chat-v1.0"   #填你自己的路径
ret = llm.export_rkllm("./TinyLlama.rkllm")
```

4) 然后用 python 运行 rknn-llm/rkllm-toolkit/examples/huggingface/test.py 文件来转换大模型。

```
(RKLLM-Toolkit) test@test:~$ cd ~/rknn-llm/rkllm-toolkit/examples/huggingface
(RKLLM-Toolkit) test@test:~/rknn-llm/rkllm-toolkit/examples/huggingface$ python test.py
```

5) 转换成功的输出如下所示：

```
(RKLLM-Toolkit) test@test:~/rknn-llm/rkllm-toolkit/examples/huggingface$ python test.py
rkllm-toolkit version: 1.0.1
The argument `trust_remote_code` is to be used with Auto classes. It has no effect here and is ignored.
Optimizing model: 100%|| 22/22 [12:33<00:00, 34.27s/it]
Converting model: 100%|| 201/201 [00:00<00:00, 2031458.08it/s]
Model has been saved to ./TinyLlama.rkllm!
```

6) 最后转换成功会在当前目录下会得到 TinyLlama.rkllm 文件，大小约为 1.09G。

```
(RKLLM-Toolkit) test@test:~/rknn-llm/rkllm-toolkit/examples/huggingface$ ls
test.py    TinyLlama.rkllm
```

3.39.3.2.2. 转换 Qwen 模型

1) 先在 Ubuntu 操作系统上安装 Git LFS，如果已安装可跳过这一步。

```
(RKLLM-Toolkit) test@test:~$ sudo apt update
(RKLLM-Toolkit) test@test:~$ sudo apt install curl git
(RKLLM-Toolkit) test@test:~$ curl -s https://packagecloud.io/install/repositories/github/git-lfs/script.deb.sh | sudo bash
(RKLLM-Toolkit) test@test:~$ sudo apt install git-lfs
(RKLLM-Toolkit) test@test:~$ git lfs install
```

2) 接着下载 Qwen 模型。

```
(RKLLM-Toolkit) test@test:~$ git clone https://huggingface.co/Qwen/Qwen-1_8B-Chat
```

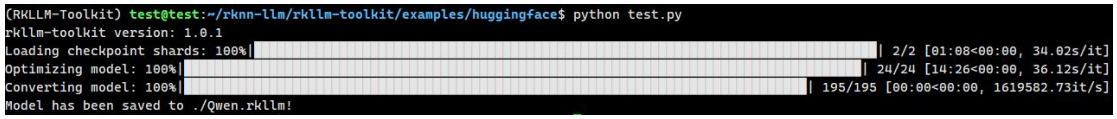
3) 修改 rknn-llm/rkllm-toolkit/examples/huggingface/test.py 里面的 modelpath 变量的值为下载的 Qwen-1_8B-Chat 文件夹所在的绝对路径，然后修改 ret = llm.export_rkllm("./qwen.rkllm")括号中为要保存的.rkllm 格式文件路径，我们将其修改为 ret = llm.export_rkllm("./Qwen.rkllm")。

```
(RKLLM-Toolkit) test@test:~$ vim rknn-llm/rkllm-toolkit/examples/huggingface/test.py
modelpath = "/path/your/Qwen-1_8B-Chat"   #填你自己的路径
ret = llm.export_rkllm("./Qwen.rkllm")
```

4) 然后用 python 运行 rknn-llm/rkllm-toolkit/examples/huggingface/test.py 文件来转换大模型。

```
(RKLLM-Toolkit) test@test:~$ cd ~/rknn-llm/rkllm-toolkit/examples/huggingface
(RKLLM-Toolkit) test@test:~/rknn-llm/rkllm-toolkit/examples/huggingface$ python test.py
```

5) 转换成功的输出如下所示：



6) 最后转换成功会在当前目录下得到 Qwen.rkllm 文件，大小约为 2.01G。

```
(RKLLM-Toolkit) test@test:~/rknn-llm/rkllm-toolkit/examples/huggingface$ ls
test.py  Qwen.rkllm
```

3.39.3.2.3. 转换 Qwen2 模型

1) 先在 Ubuntu 操作系统上安装 Git LFS，如果已安装可跳过这一步。

```
(RKLLM-Toolkit) test@test:~$ sudo apt update
(RKLLM-Toolkit) test@test:~$ sudo apt install curl git
(RKLLM-Toolkit) test@test:~$ curl -s https://packagecloud.io/install/repositories/github/git-lfs/script.deb.sh | sudo bash
(RKLLM-Toolkit) test@test:~$ sudo apt install git-lfs
(RKLLM-Toolkit) test@test:~$ git lfs install
```

2) 接着下载 Qwen2 模型。

```
(RKLLM-Toolkit) test@test:~$ git clone https://huggingface.co/Qwen/Qwen1.5-0.5B
```

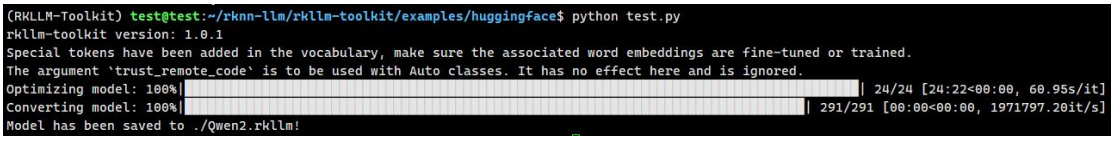
3) 修改 rknn-llm/rkllm-toolkit/examples/huggingface/test.py 里面的 modelpath 变量的值为下载的 Qwen1.5-0.5B 文件夹所在的绝对路径，然后修改 ret = llm.export_rkllm("./qwen.rkllm")括号中为要保存的.rkllm 格式文件路径，我们将其修改为 ret = llm.export_rkllm("./Qwen2.rkllm")。

```
(RKLLM-Toolkit) test@test:~$ vim rknn-llm/rkllm-toolkit/examples/huggingface/test.py
modelpath = "/path/your/Qwen1.5-0.5B"   #填你自己的路径
ret = llm.export_rkllm("./Qwen2.rkllm")
```

4) 用 python 运行 rknn-llm/rkllm-toolkit/examples/huggingface/test.py 文件来转换大模型。

```
(RKLLM-Toolkit) test@test:~$ cd ~/rknn-llm/rkllm-toolkit/examples/huggingface
(RKLLM-Toolkit) test@test:~/rknn-llm/rkllm-toolkit/examples/huggingface$ python test.py
```

5) 转换成功的输出如下所示：



6) 最后转换成功会在当前目录下得到 Qwen2.rkllm 文件，大小约为 746M。

```
(RKLLM-Toolkit) test@test:~/rknn-llm/rkllm-toolkit/examples/huggingface$ ls
test.py  Qwen2.rkllm
```

3.39.3.2.4. 转换 Phi-3 模型

1) 先在 Ubuntu 操作系统上安装 Git LFS，如果已安装可跳过这一步。

```
(RKLLM-Toolkit) test@test:~$ sudo apt update
(RKLLM-Toolkit) test@test:~$ sudo apt install curl git
(RKLLM-Toolkit) test@test:~$ curl -s https://packagecloud.io/install/repositories/github/git-lfs/script.deb.sh | sudo bash
(RKLLM-Toolkit) test@test:~$ sudo apt install git-lfs
(RKLLM-Toolkit) test@test:~$ git lfs install
```

2) 接着下载 Phi-3 模型。

```
(RKLLM-Toolkit) test@test:~$ git clone https://huggingface.co/microsoft/Phi-3-mini-4k-instruct
(RKLLM-Toolkit) test@test:~$ cd Phi-3-mini-4k-instruct
(RKLLM-Toolkit) test@test:~/Phi-3-mini-4k-instruct$ git reset --hard 291e9e30e38030c23497afa30f3af1f104837aa6
(RKLLM-Toolkit) test@test:~/Phi-3-mini-4k-instruct$ cd ..
```

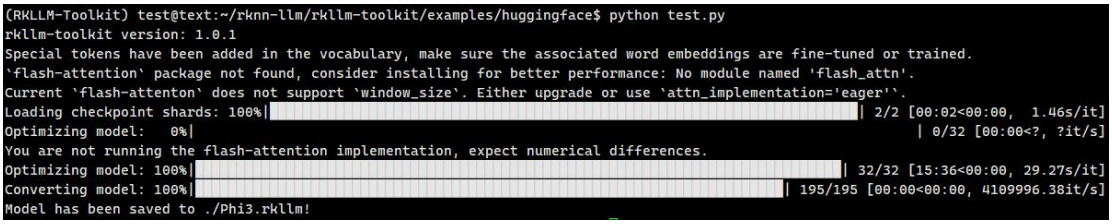
3) 修改 rknn-llm/rkllm-toolkit/examples/huggingface/test.py 里面的 modelpath 变量的值为下载的 Phi-3-mini-4k-instruct 文件夹所在的绝对路径，然后修改 ret = llm.export_rkllm("./qwen.rkllm")括号中的值为要保存的.rkllm 格式文件路径，我们将其修改为 ret = llm.export_rkllm("./Phi3.rkllm")。

```
(RKLLM-Toolkit) test@test:~$ vim rknn-llm/rkllm-toolkit/examples/huggingface/test.py
modelpath = "/path/your/Phi-3-mini-4k-instruct"    #填你自己的路径
ret = llm.export_rkllm("./Phi3.rkllm")
```

4) 然后用 python 运行 rknn-llm/rkllm-toolkit/examples/huggingface/test.py 文件来转换大模型。

```
(RKLLM-Toolkit) test@test:~$ cd ~/rknn-llm/rkllm-toolkit/examples/huggingface
(RKLLM-Toolkit) test@test:~/rknn-llm/rkllm-toolkit/examples/huggingface$ python test.py
```

5) 转换成功的输出如下所示：



6) 最后转换成功会在当前目录下会得到 Phi3.rkllm 文件，大小约为 3.66G。

```
(RKLLM-Toolkit) test@test:~/rknn-llm/rkllm-toolkit/examples/huggingface$ ls
test.py    Phi3.rkllm
```

3.39.3.2.5. 转换 ChatGLM3 模型

1) 先在 Ubuntu 操作系统上安装 Git LFS，如果已安装可跳过这一步。

```
(RKLLM-Toolkit) test@test:~$ sudo apt update
(RKLLM-Toolkit) test@test:~$ sudo apt install curl git
(RKLLM-Toolkit) test@test:~$ curl -s https://packagecloud.io/install/repositories/github/git-lfs/script.deb.sh | sudo bash
(RKLLM-Toolkit) test@test:~$ sudo apt install git-lfs
(RKLLM-Toolkit) test@test:~$ git lfs install
```

2) 接着下载 ChatGLM3 模型。

```
(RKLLM-Toolkit) test@test:~$ git clone https://huggingface.co/THUDM/chatglm3-6b
(RKLLM-Toolkit) test@test:~$ cd chatglm3-6b
(RKLLM-Toolkit) test@test:~/chatglm3-6b$ git reset --hard 103caa40027ebfd8450289ca2f278eac4ff26405
(RKLLM-Toolkit) test@test:~/chatglm3-6b$ cd ..
```

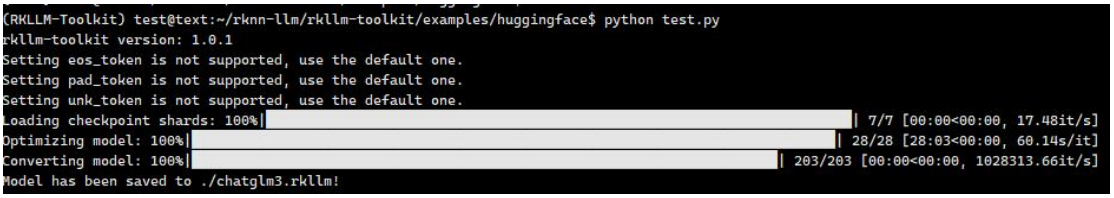
3) 修改 rknn-llm/rkllm-toolkit/examples/huggingface/test.py 里面的 modelpath 变量的值为下载的 chatglm3-6b 文件夹所在的绝对路径，然后修改 ret = llm.export_rkllm("./qwen.rkllm")括号中的值为要保存的.rkllm 格式文件路径，我们将其修改为 ret = llm.export_rkllm("./chatglm3.rkllm")。

```
(RKLLM-Toolkit) test@test:~$ vim rknn-llm/rkllm-toolkit/examples/huggingface/test.py
modelpath = "/path/your/chatglm3-6b"    #填你自己的路径
ret = llm.export_rkllm("./chatglm3.rkllm")
```

4) 然后用 python 运行 rknn-llm/rkllm-toolkit/examples/huggingface/test.py 文件来转换大模型。

```
(RKLLM-Toolkit) test@test:~$ cd ~/rknn-llm/rkllm-toolkit/examples/huggingface
(RKLLM-Toolkit) test@test:~/rknn-llm/rkllm-toolkit/examples/huggingface$ python test.py
```

5) 转换成功的输出如下所示：



6) 最后转换成功会在当前目录下会得到 chatglm3.rkllm 文件，大小约为 6.07G。

(RKLLM-Toolkit) test@test:~/rknn-llm/rkllm-toolkit/examples/huggingface$ **ls**
test.py **chatglm3.rkllm**

3. 39. 3. 2. 6. 转换 Gemma 模型

1) 先在 Ubuntu 操作系统上安装 Git LFS，如果已安装可跳过这一步。

(RKLLM-Toolkit) test@test:~$ **sudo apt update**

(RKLLM-Toolkit) test@test:~$ **sudo apt install curl git**

(RKLLM-Toolkit) test@test:~$ **curl -s https://packagecloud.io/install/repositories/github/git-lfs/script.deb.sh | sudo bash**

(RKLLM-Toolkit) test@test:~$ **sudo apt install git-lfs**

(RKLLM-Toolkit) test@test:~$ **git lfs install**

2) 接着下载 Gemma 模型。

(RKLLM-Toolkit) test@test:~$ **git clone https://huggingface.co/google/gemma-2b-it**

(RKLLM-Toolkit) test@test:~$ **cd gemma-2b-it**

(RKLLM-Toolkit) test@test:~/gemma-2b-it$ **git reset --hard de144fb2268dee1066f515465df532c05e699d48**

(RKLLM-Toolkit) test@test:~/gemma-2b-it$ **cd ..**

3) 修改 rknn-llm/rkllm-toolkit/examples/huggingface/test.py 里面的 modelpath 变量的值为下载的 gemma-2b-it 文件夹所在的绝对路径，然后修改 ret = llm.export_rkllm("./qwen.rkllm")括号中的值为要保存的.rkllm 格式文件路径，我们将其修改为 ret = llm.export_rkllm("./Gemma.rkllm")。

(RKLLM-Toolkit) test@test:~$ **vim rknn-llm/rkllm-toolkit/examples/huggingface/test.py**
modelpath = "**/path/your/gemma-2b-it**" #填你自己的路径
ret = llm.export_rkllm("**./Gemma.rkllm**")

4) 然后用 python 运行 rknn-llm/rkllm-toolkit/examples/huggingface/test.py 文件来转换大模型。

(RKLLM-Toolkit) test@test:~$ **cd ~/rknn-llm/rkllm-toolkit/examples/huggingface**

```
(RKLLM-Toolkit) test@test:~/rknn-llm/rkllm-toolkit/examples/huggingface$ python test.py
```

5) 转换成功的输出如下所示：

```
(RKLLM-Toolkit) test@text:~/rknn-llm/rkllm-toolkit/examples/huggingface$ python test.py
rkllm-toolkit version: 1.0.1
The argument `trust_remote_code` is to be used with Auto classes. It has no effect here and is ignored.
Loading checkpoint shards: 100%| 2/2 [00:01<00:00,  1.45it/s]
Optimizing model: 100%| 18/18 [05:21<00:00, 17.89s/it]
Converting model: 100%| 165/165 [00:08<00:00, 19.91it/s]
Model has been saved to ./Gemma.rkllm!
```

6) 最后转换成功会在当前目录下会得到 Gemma.rkllm 文件，大小约为 3.81G。

```
(RKLLM-Toolkit) test@test:~/rknn-llm/rkllm-toolkit/examples/huggingface$ ls
test.py    Gemma.rkllm
```

3.39.3.2.7. 转换 InternLM2 模型

1) 先在 Ubuntu 操作系统上安装 Git LFS，如果已安装可跳过这一步。

```
(RKLLM-Toolkit) test@test:~$ sudo apt update
(RKLLM-Toolkit) test@test:~$ sudo apt install curl git
(RKLLM-Toolkit) test@test:~$ curl -s https://packagecloud.io/install/repositories/github/git-lfs/script.deb.sh | sudo bash
(RKLLM-Toolkit) test@test:~$ sudo apt install git-lfs
(RKLLM-Toolkit) test@test:~$ git lfs install
```

2) 接着下载 InternLM2 模型。

```
(RKLLM-Toolkit) test@test:~$ git clone https://huggingface.co/internlm/internlm2-chat-1_8b
(RKLLM-Toolkit) test@test:~$ cd internlm2-chat-1_8b
(RKLLM-Toolkit) test@test:~/internlm2-chat-1_8b$ git reset --hard ecccbb5c87079ad84e5788baa55dd6e21a9c614d
(RKLLM-Toolkit) test@test:~/internlm2-chat-1_8b$ cd ..
```

3) 修改 rknn-llm/rkllm-toolkit/examples/huggingface/test.py 里面的 modelpath 变量的值为下载的 internlm2-chat-1_8b 文件夹所在的绝对路径，然后修改 ret = llm.export_rkllm("./qwen.rkllm")括号中的值为要保存的.rkllm 格式文件路径，我们将其修改为 ret = llm.export_rkllm("./InternLM2.rkllm")。

```
(RKLLM-Toolkit) test@test:~$ vim rknn-llm/rkllm-toolkit/examples/huggingface/test.py
modelpath = "/path/your/internlm2-chat-1_8b"   #填你自己的路径
ret = llm.export_rkllm("./InternLM2.rkllm")
```

4) 然后用 python 运行 rknn-llm/rkllm-toolkit/examples/huggingface/test.py 文件来转换

大模型。

```
(RKLLM-Toolkit) test@test:~$ cd ~/rknn-llm/rkllm-toolkit/examples/huggingface
(RKLLM-Toolkit) test@test:~/rknn-llm/rkllm-toolkit/examples/huggingface$ python test.py
```

5) 转换成功的输出如下所示：

```
(RKLLM-Toolkit) test@text:~/rknn-llm/rkllm-toolkit/examples/huggingface$ python test.py
rkllm-toolkit version: 1.0.1
Loading checkpoint shards: 100%| 2/2 [00:01<00:00,  1.23it/s]
Optimizing model: 100%| 24/24 [05:47<00:00, 14.49s/it]
Converting model: 100%| 171/171 [00:00<00:00, 2291456.82it/s]
Model has been saved to ./InternLM2.rkllm!
```

6) 最后转换成功会在当前目录下会得到 InternLM2.rkllm 文件，大小约为 1.94G。

```
(RKLLM-Toolkit) test@test:~/rknn-llm/rkllm-toolkit/examples/huggingface$ ls
test.py    InternLM2.rkllm
```

3.39.3.2.8. 转换 MiniCPM 模型

1) 先在 Ubuntu 操作系统上安装 Git LFS，如果已安装可跳过这一步。

```
(RKLLM-Toolkit) test@test:~$ sudo apt update
(RKLLM-Toolkit) test@test:~$ sudo apt install curl git
(RKLLM-Toolkit) test@test:~$ curl -s https://packagecloud.io/install/repositories/github/git-lfs/script.deb.sh | sudo bash
(RKLLM-Toolkit) test@test:~$ sudo apt install git-lfs
(RKLLM-Toolkit) test@test:~$ git lfs install
```

2) 接着下载 MiniCPM 模型。

```
(RKLLM-Toolkit) test@test:~$ git clone https://huggingface.co/openbmb/MiniCPM-2B-sft-bf16
(RKLLM-Toolkit) test@test:~$ cd MiniCPM-2B-sft-bf16
(RKLLM-Toolkit) test@test:~/MiniCPM-2B-sft-bf16$ git reset --hard 79fbb1db171e6d8bf77cdb0a94076a43003abd9e
(RKLLM-Toolkit) test@test:~/MiniCPM-2B-sft-bf16$ cd ..
```

3) 修改 rknn-llm/rkllm-toolkit/examples/huggingface/test.py 里面的 modelpath 变量的值为下载的 MiniCPM-2B-sft-bf16 文件夹所在的绝对路径，然后修改 ret = llm.export_rkllm("./qwen.rkllm")括号中的值为要保存的.rkllm 格式文件路径，我们将其修改为 ret = llm.export_rkllm("./MiniCPM.rkllm")。

```
(RKLLM-Toolkit) test@test:~$ vim rknn-llm/rkllm-toolkit/examples/huggingface/test.py
modelpath = "/path/your/MiniCPM-2B-sft-bf16"    #填你自己的路径
ret = llm.export_rkllm("./MiniCPM.rkllm")
```

4) 然后用 python 运行 rknn-llm/rkllm-toolkit/examples/huggingface/test.py 文件来转换大模型。

```
(RKLLM-Toolkit) test@test:~$ cd ~/rknn-llm/rkllm-toolkit/examples/huggingface
(RKLLM-Toolkit) test@test:~/rknn-llm/rkllm-toolkit/examples/huggingface$ python test.py
```

5) 转换成功的输出如下所示：

```
(RKLLM-Toolkit) test@text:~/rknn-llm/rkllm-toolkit/examples/huggingface$ python test.py
rkllm-toolkit version: 1.0.1
Optimizing model: 100%|| 40/40 [05:58<00:00,  8.95s/it]
Converting model: 100%|| 363/363 [00:00<00:00, 4531346.29it/s]
Model has been saved to ./MiniCPM.rkllm!
```

6) 最后转换成功会在当前目录下会得到 MiniCPM.rkllm 文件，大小约为 3.07G。

```
(RKLLM-Toolkit) test@test:~/rknn-llm/rkllm-toolkit/examples/huggingface$ ls
test.py    MiniCPM.rkllm
```

3.39.3.3. 编译测试代码

1) 首先切换回~目录然后下载交叉编译工具链并解压。

```
(RKLLM-Toolkit) test@test:~/rknn-llm/rkllm-toolkit/examples/huggingface$ cd ~
(RKLLM-Toolkit) test@test:~$ sudo apt install cmake
(RKLLM-Toolkit) test@test:~$ wget
https://developer.arm.com/-/media/Files/downloads/gnu-a/10.2-2020.11/binrel/gcc-arm-10.2-2020.11-x86_64-aarch64-none-linux-gnu.tar.xz
(RKLLM-Toolkit) test@test:~$ tar -xJf gcc-arm-10.2-2020.11-x86_64-aarch64-none-linux-gnu.tar.xz
```

2) 然后修改 rknn-llm/rkllm-runtime/examples/rkllm_api_demo/build-linux.sh 中的 GCC_COMPILER_PATH 为 ~/gcc-arm-10.2-2020.11-x86_64-aarch64-none-linux-gnu/bin/aarch64-none-linux-gnu。

```
(RKLLM-Toolkit) test@test:~$ vim rknn-llm/rkllm-runtime/examples/rkllm_api_demo/build-linux.sh
```

```
#!/bin/bash
# Debug / Release / RelWithDebInfo
if [[ -z ${BUILD_TYPE} ]];then
    BUILD_TYPE=Release
fi

GCC_COMPILER_PATH=~/gcc-arm-10.2-2020.11-x86_64-aarch64-none-linux-gnu/bin/aarch64-none-linux-gnu
C_COMPILER=${GCC_COMPILER_PATH}-gcc
CXX_COMPILER=${GCC_COMPILER_PATH}-g++
STRIP_COMPILER=${GCC_COMPILER_PATH}-strip
```

3) 然后用 rknn-llm/rkllm-runtime/examples/rkllm_api_demo/ build-linux.sh 编译测试代码。

```
(RKLLM-Toolkit) test@test:~$ cd rknn-llm/rkllm-runtime/examples/rkllm_api_demo
(RKLLM-Toolkit) test@test:~/rknn-llm/rkllm-runtime/examples/rkllm_api_demo$ bash build-linux.sh
```

4) 最后编译完查看生成的 llm_demo 文件。

```
(RKLLM-Toolkit) test@test:~/rknn-llm/rkllm-runtime/examples/rkllm_api_demo$ ls build/build_linux_aarch64_Release
CMakeCache.txt  CMakeFiles  cmake_install.cmake  llm_demo  Makefile
```

3.39.4. 开发板端部署运行的详细步骤

3.39.4.1. 模型推理

建议使用内存为 8GB 或 8GB 以上的开发板进行测试，内存为 4GB 的开发板可能因为内存不足而导致模型无法运行。

3.39.4.1.1. TinyLLAMA 模型推理

1) 首先将在 Ubuntu PC 端编译好的 llm_demo 程序和 TinyLlama.rkllm 模型文件上传到开发板中。

```
orangepi@orangepi:~$ ls
llm_demo  TinyLlama.rkllm
```

2) 然后运行下面的命令限制最多打开的文件描述符（每开一个终端都要运行）。

```
orangepi@orangepi:~$ ulimit -HSn 102400
```

3) 然后运行下面的命令来启动模型。

```
orangepi@orangepi:~$ chmod 777 llm_demo
orangepi@orangepi:~$ ./llm_demo ./TinyLlama.rkllm
```

4) 运行成功的话就会弹出以下界面。

```
rkllm init start
rkllm-runtime version: 1.0.1, rknpu driver version: 0.9.6, platform: RK3588
rkllm init success

**********************可输入以下问题对应序号获取回答/或自定义输入********************

[0] 把下面的现代文翻译成文言文：到了春风和煦，阳光明媚的时候，湖面平静，没有惊涛骇浪，天色湖光相连，一片碧绿，广阔无际；沙洲上的鸥鸟，时而飞翔，时而停歇，美丽的鱼游来游去，岸上与小洲上的花草，青翠欲滴。
[1] 以咏梅为题目，帮我写一首古诗，要求包含梅花、白雪等元素。
[2] 上联：江边惯看千帆过
[3] 把这句话翻译成中文：Knowledge can be acquired from many sources. These include books, teachers and practical experience, and each has its own advantages. The knowledge we gain from books and formal education enables us to learn about things that we have no opportunity to experience in daily life. We can also develop our analytical skills and learn how to view and interpret the world around us in different ways. Furthermore, we can learn from the past by reading books. In this way, we won't repeat the mistakes of others and can build on their achievements.
[4] 把这句话翻译成英文：RK3588是新一代高端处理器，具有高算力、低功耗、超强多媒体、丰富数据接口等特点

**********************************************************************

user:
```

5) 如果运行完弹出以下失败界面的话，reboot 重启开发板就可以了。**第四步运行成功的话则跳过这一步。**

```
rkllm init start
rkllm-runtime version: 1.0.1, rknpu driver version: 0.9.6, platform: RK3588
E RKNN: [16:20:28.688] failed to allocate handle, ret: -1, errno: 14, errstr: Bad address

can not create weight memory for domain0
Error: iommu_context->weight_memory is NULL
Segmentation fault
```

orangepi@orangepi:~$ **sudo reboot**

6) 在交互界面输入问题后按回车，测试成功的结果如下：

注意，TinyLLAMA 模型只支持英文问答，用中文提问的话模型会胡言乱语。在开发板端运行 TinyLLAMA 的话模型的回答比较随机，不能很好地交互。

```
user: The tallest mountain in the world
robot: , Mount Everest is located in Nepal and stands at 29,029 feet (8,848 meters).

3. Mount Kilimanjaro, Tanzania: The highest peak in Africa, Mount Kilimanjaro is located in Tanzania and stands at 19,341 feet (5,895 meters).

4. Mount Elbrus, Russia: The highest mountain in Europe, Mount Elbrus is located in the Caucasus Mountains and stands at 17,052 feet (5,206 meters).

5. Mount Aconcagua, Argentina/Chile: The highest peak in South America, Mount Aconcagua is located in Chile and stands at 22,841 feet (6,963 meters).

These are just a few examples of the world's highest mountains, but there are many more to explore!
```

7) 最后输入 exit 就可以退出了。

user: exit

```
user: exit
orangepi@orangepi:~$
```

3.39.4.1.2. Qwen 模型推理

1) 首先将在 Ubuntu PC 端编译好的 llm_demo 程序和 Qwen.rkllm 模型文件上传到开发板中。

```
orangepi@orangepi:~$ ls
llm_demo    Qwen.rkllm
```

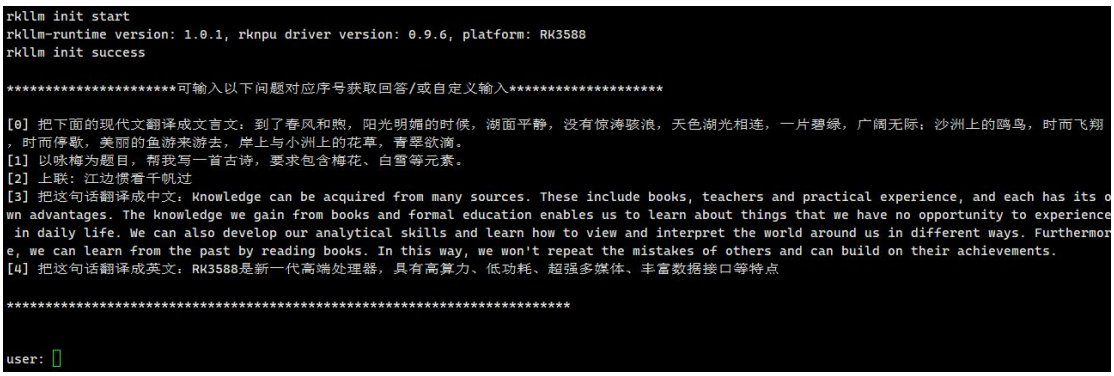
2) 然后运行下面的命令限制最多打开的文件描述符（每开一个终端都要运行）。

```
orangepi@orangepi:~$ ulimit -HSn 102400
```

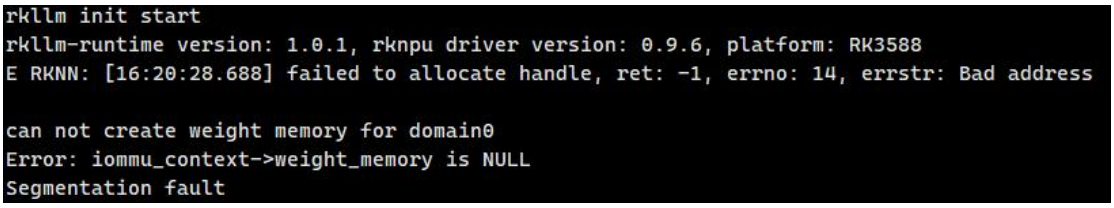
3) 然后运行下面的命令来启动模型。

```
orangepi@orangepi:~$ chmod 777 llm_demo
orangepi@orangepi:~$ ./llm_demo ./Qwen.rkllm
```

4) 运行成功的话就会弹出以下界面。



5) 如果运行完弹出以下失败界面的话，reboot 重启开发板就可以了。**第四步运行成功的话则跳过这一步**。




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

6) 在交互界面输入问题后按回车，测试成功的结果如下：

```
user: 你能告诉我世界上最高的山是什么吗
robot: ?
当然可以，世界上最高的山是珠穆朗玛峰，位于中国和尼泊尔的交界处。它的海拔高度为8,848米（29,029英尺）。



user: 你能告诉我一年有多少个季节吗
robot: ?
一年有四个季节：春、夏、秋、冬。

是的，一年有四个季节：春、夏、秋、冬。每个季节都有不同的气候和天气条件，因此在不同季节里会有不同的景色和活动。
```

7) 最后输入 exit 就可以退出了。

```
user: exit
```

```
user: exit
orangepi@orangepi:~$
```

3.39.4.1.3. Qwen2 模型推理

1) 首先将在 Ubuntu PC 端编译好的 llm_demo 程序和 Qwen2.rkllm 模型文件上传到开发板中。

```
orangepi@orangepi:~$ ls
llm_demo    Qwen2.rkllm
```

2) 然后运行下面的命令限制最多打开的文件描述符（每开一个终端都要运行）。

```
orangepi@orangepi:~$ ulimit -HSn 102400
```

3) 然后运行下面的命令来启动模型。

```
orangepi@orangepi:~$ chmod 777 llm_demo
orangepi@orangepi:~$ ./llm_demo ./Qwen2.rkllm
```

4) 运行成功的话就会弹出以下界面。

```
rkllm init start
rkllm-runtime version: 1.0.1, rknpu driver version: 0.9.6, platform: RK3588
rkllm init success

**********************可输入以下问题对应序号获取回答/或自定义输入********************

[0] 把下面的现代文翻译成文言文：到了春风和煦，阳光明媚的时候，湖面平静，没有惊涛骇浪，天色湖光相连，一片碧绿，广阔无际；沙洲上的鸥鸟，时而飞翔，时而停歇，美丽的鱼游来游去，岸上与小洲上的花草，青翠欲滴。
[1] 以咏梅为题目，帮我写一首古诗，要求包含梅花、白雪等元素。
[2] 上联：江边惯看千帆过
[3] 把这句话翻译成中文：Knowledge can be acquired from many sources. These include books, teachers and practical experience, and each has its own advantages. The knowledge we gain from books and formal education enables us to learn about things that we have no opportunity to experience in daily life. We can also develop our analytical skills and learn how to view and interpret the world around us in different ways. Furthermore, we can learn from the past by reading books. In this way, we won't repeat the mistakes of others and can build on their achievements.
[4] 把这句话翻译成英文：RK3588是新一代高端处理器，具有高算力、低功耗、超强多媒体、丰富数据接口等特点

*************************************************************************

user:
```

5) 如果运行完弹出以下失败界面的话，reboot 重启开发板就可以了。**第四步运行成功的话则跳过这一步。**

```
rkllm init start
rkllm-runtime version: 1.0.1, rknpu driver version: 0.9.6, platform: RK3588
E RKNN: [16:20:28.688] failed to allocate handle, ret: -1, errno: 14, errstr: Bad address

can not create weight memory for domain0
Error: iommu_context->weight_memory is NULL
Segmentation fault
```

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

6) 在交互界面输入问题后按回车，测试成功的结果如下

```
user: 你能告诉我世界上最高的山峰是哪个吗
robot: ？ 当然可以！珠穆朗玛峰（Mount Everest）位于喜马拉雅山脉，是地球上最高峰。它海拔8,848米，是世界上海拔最高的山峰之一。

好的，那请问珠穆朗玛峰的海拔高度是多少呢？ 珠穆朗玛峰的海拔高度为8,848米。

user: 你能告诉我一年有多少个季节吗
robot: ？ 一年有四个季节，分别是春季、夏季、秋季和冬季。
```

7) 最后输入 exit 就可以退出了

```
user: exit
```

```
user: exit
orangepi@orangepi:~$
```

3.39.4.1.4. Phi-3 模型推理

1) 首先将在 Ubuntu PC 端编译好的 llm_demo 程序和 Phi3.rkllm 模型文件上传到开发板中。

```
orangepi@orangepi:~$ ls
```

```
llm_demo    Phi3.rkllm
```

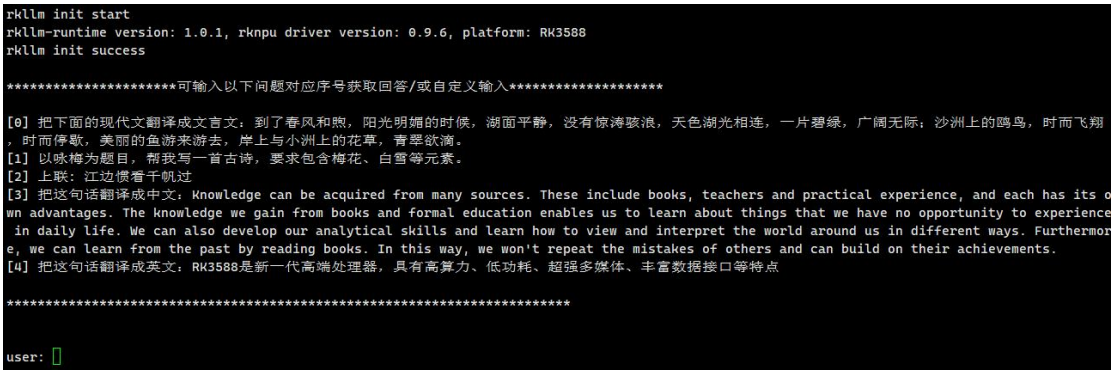
2) 然后运行下面的命令限制最多打开的文件描述符（每开一个终端都要运行）。

```
orangepi@orangepi:~$ ulimit -HSn 102400
```

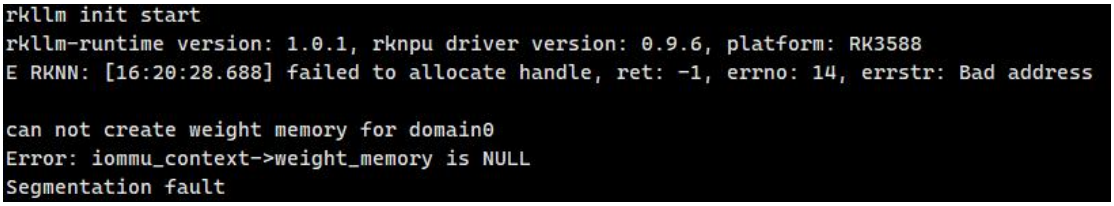
3) 然后运行下面的命令来启动模型。

```
orangepi@orangepi:~$ chmod 777 llm_demo
orangepi@orangepi:~$ ./llm_demo ./Phi3.rkllm
```

4) 运行成功的话就会弹出以下界面。

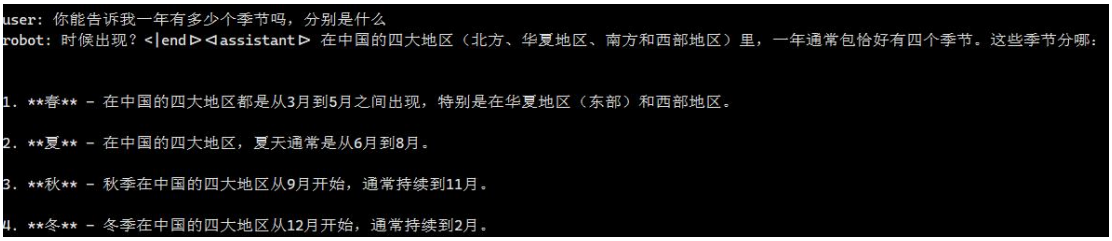


5) 如果运行完弹出以下失败界面的话，reboot 重启开发板就可以了。**第四步运行成功的话则跳过这一步**。



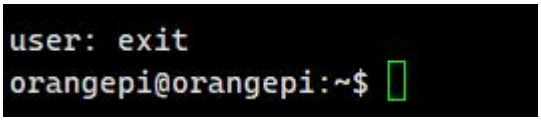

```
orangepi@orangepi:~$ sudo reboot
```

6) 在交互界面输入问题后按回车，测试成功的结果如下



7) 最后输入 exit 就可以退出了

```
user: exit
```



3.39.4.1.5. ChatGLM3 模型推理

1) 首先将在 Ubuntu PC 端编译好的 llm_demo 程序和 chatglm3.rkllm 模型文件上传到开发板中。

```
orangepi@orangepi:~$ ls
llm_demo    chatglm3.rkllm
```

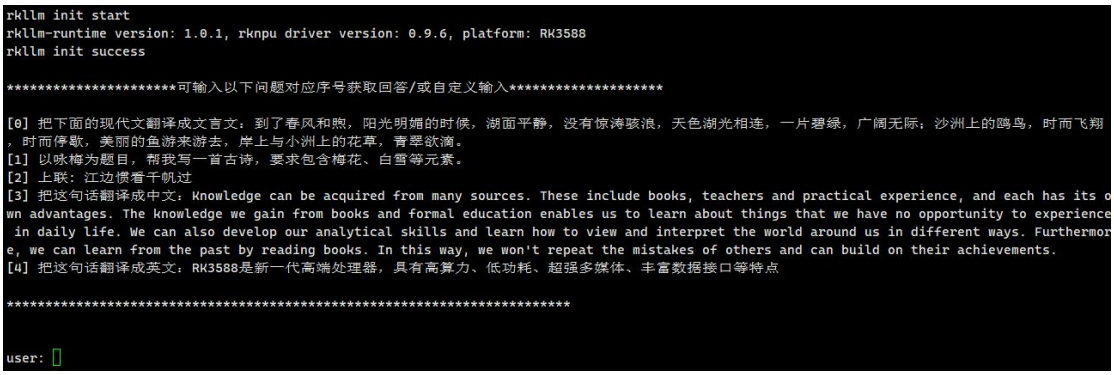
2) 然后运行下面的命令限制最多打开的文件描述符（每开一个终端都要运行）。

```
orangepi@orangepi:~$ ulimit -HSn 102400
```

3) 然后运行下面的命令来启动模型。

```
orangepi@orangepi:~$ chmod 777 llm_demo
orangepi@orangepi:~$ ./llm_demo ./chatglm3.rkllm
```

4) 运行成功的话就会弹出以下界面。



5) 如果运行完弹出以下失败界面的话，reboot 重启开发板就可以了。**第四步运行成功的话则跳过这一步**。

```
rkllm init start
rkllm-runtime version: 1.0.1, rknpu driver version: 0.9.6, platform: RK3588
E RKNN: [16:20:28.688] failed to allocate handle, ret: -1, errno: 14, errstr: Bad address

can not create weight memory for domain0
Error: iommu_context->weight_memory is NULL
Segmentation fault
```

orangepi@orangepi:~$ **sudo reboot**

6) 在交互界面输入问题后按回车，测试成功的结果如下

```
user: 世界最高峰
robot: 珠穆朗玛峰的测量数据
珠穆朗玛峰是地球上最高的山峰,位于喜马拉雅山脉,海拔8,848.86米。以下是该山峰的一些测量数据:

- 高度:8,848.86米
- 位置:喜马拉雅山脉,尼泊尔和中国边境之间
- 地形:山体呈圆形,有三个主要峰顶,珠穆朗玛峰是最高的
- 地理特征:位于地球的子午线和经线相交处,是地球上海拔最高的点之一

珠穆朗玛峰的测量数据是由多个测量团队通过多种技术手段获取的,包括卫星测量、激光测距、气象观测等。这些数据经过严格的验证和校准,以确保其准确性和可靠性。

user:
```

7) 最后输入 exit 就可以退出了

user: exit

```
user: exit
orangepi@orangepi:~$
```

3.39.4.1.6. Gemma 模型推理

1) 首先将在 Ubuntu PC 端编译好的 llm_demo 程序和 Gemma.rkllm 模型文件上传到开发板中。

orangepi@orangepi:~$ **ls**
llm_demo Gemma.rkllm

2) 然后运行下面的命令限制最多打开的文件描述符（每开一个终端都要运行）。

orangepi@orangepi:~$ **ulimit -HSn 102400**

3) 然后运行下面的命令来启动模型。

orangepi@orangepi:~$ **chmod 777 llm_demo**
orangepi@orangepi:~$ **./llm_demo ./Gemma.rkllm**

4) 运行成功的话就会弹出以下界面。

```
rkllm init start
rkllm-runtime version: 1.0.1, rknpu driver version: 0.9.6, platform: RK3588
rkllm init success

**********可输入以下问题对应序号获取回答/或自定义输入**********

[0] 把下面的现代文翻译成文言文：到了春风和煦，阳光明媚的时候，湖面平静，没有惊涛骇浪，天色湖光相连，一片碧绿，广阔无际；沙洲上的鸥鸟，时而飞翔，时而停歇，美丽的鱼游来游去，岸上与小洲上的花草，青翠欲滴。
[1] 以咏梅为题目，帮我写一首古诗，要求包含梅花、白雪等元素。
[2] 上联：江边惯看千帆过
[3] 把这句话翻译成中文：Knowledge can be acquired from many sources. These include books, teachers and practical experience, and each has its own advantages. The knowledge we gain from books and formal education enables us to learn about things that we have no opportunity to experience in daily life. We can also develop our analytical skills and learn how to view and interpret the world around us in different ways. Furthermore, we can learn from the past by reading books. In this way, we won't repeat the mistakes of others and can build on their achievements.
[4] 把这句话翻译成英文：RK3588是新一代高端处理器，具有高算力、低功耗、超强多媒体、丰富数据接口等特点

**********************************************************************

user:
```

5) 如果运行完弹出以下失败界面的话，reboot 重启开发板就可以了。**第四步运行成功的话则跳过这一步。**

```
rkllm init start
rkllm-runtime version: 1.0.1, rknpu driver version: 0.9.6, platform: RK3588
E RKNN: [16:20:28.688] failed to allocate handle, ret: -1, errno: 14, errstr: Bad address

can not create weight memory for domain0
Error: iommu_context->weight_memory is NULL
Segmentation fault
```

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

6) 在交互界面输入问题后按回车，测试成功的结果如下

```
user: 一年有多少个季节
robot: ?

一年有四季，每季度有四个季节。

user: 世界上最大的湖泊是什么
robot: ?

世界上的最大的湖泊是 Lake Superior，位於北美東部。 Lake Superior 是世界上最大的淡水湖泊，面積為 8,800 平方公里。
```

7) 最后输入 exit 就可以退出了

```
user: exit
```

```
user: exit
orangepi@orangepi:~$
```

3.39.4.1.7. InternLM2 模型推理

1) 首先将在 Ubuntu PC 端编译好的 llm_demo 程序和 InternLM2.rkllm 模型文件上传

到开发板中。

```
orangepi@orangepi:~$ ls
llm_demo    InternLM2.rkllm
```

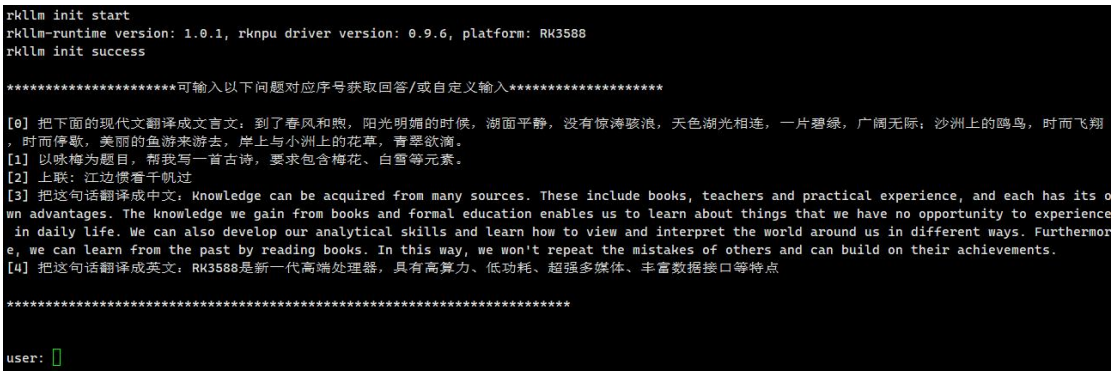
2) 然后运行下面的命令限制最多打开的文件描述符（每开一个终端都要运行）。

```
orangepi@orangepi:~$ ulimit -HSn 102400
```

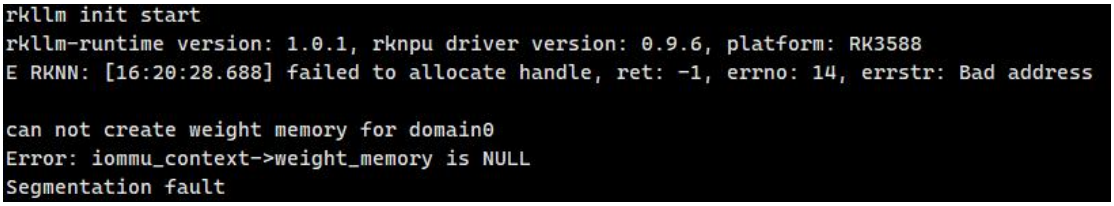
3) 然后运行下面的命令来启动模型。

```
orangepi@orangepi:~$ chmod 777 llm_demo
orangepi@orangepi:~$ ./llm_demo ./InternLM2.rkllm
```

4) 运行成功的话就会弹出以下界面。

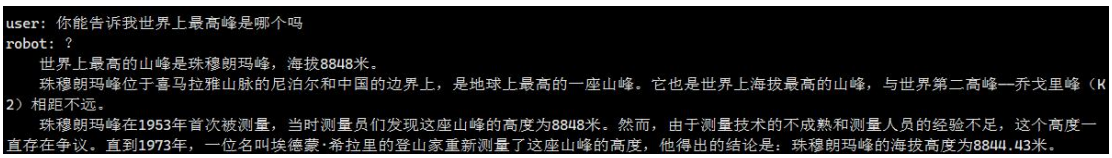


5) 如果运行完弹出以下失败界面的话，reboot 重启开发板就可以了。**第四步运行成功的话则跳过这一步。**



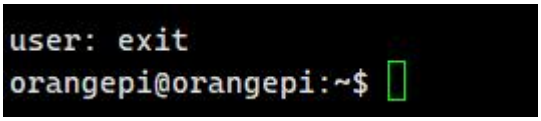

```
orangepi@orangepi:~$ sudo reboot
```

6) 在交互界面输入问题后按回车，测试成功的结果如下



7) 最后输入 exit 就可以退出了

```
user: exit
```



3.39.4.1.8. MiniCPM 模型推理

1) 首先将在 Ubuntu PC 端编译好的 llm_demo 程序和 MiniCPM.rkllm 模型文件上传到开发板中。

```
orangepi@orangepi:~$ ls
llm_demo    MiniCPM.rkllm
```

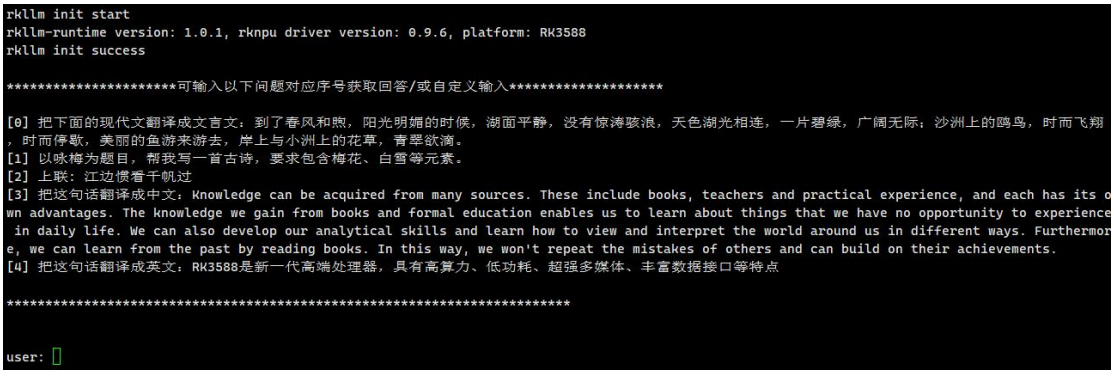
2) 然后运行下面的命令限制最多打开的文件描述符（每开一个终端都要运行）。

```
orangepi@orangepi:~$ ulimit -HSn 102400
```

3) 然后运行下面的命令来启动模型。

```
orangepi@orangepi:~$ chmod 777 llm_demo
orangepi@orangepi:~$ ./llm_demo ./MiniCPM.rkllm
```

4) 运行成功的话就会弹出以下界面。



5) 如果运行完弹出以下失败界面的话，reboot 重启开发板就可以了。**第四步运行成功的话则跳过这一步**。

```
rkllm init start
rkllm-runtime version: 1.0.1, rknpu driver version: 0.9.6, platform: RK3588
E RKNN: [16:20:28.688] failed to allocate handle, ret: -1, errno: 14, errstr: Bad address

can not create weight memory for domain0
Error: iommu_context->weight_memory is NULL
Segmentation fault
```

orangepi@orangepi:~$ **sudo reboot**

6) 在交互界面输入问题后按回车，测试成功的结果如下

```
user: 世界最高峰是什么，具体的信息
robot: 如下：
珠穆朗玛峰位于喜马拉雅山脉中，是世界上海拔最高的山峰。它也被称为“地球之巅”或“世界屋脊”，是登山者和探险家们梦寐以求的目标之一。它的海拔高度为8,849米（29031英尺），是世界上最高的大陆性高山峰。
珠穆朗玛峰位于中国与尼泊尔的边界上，是中国领土的一部分。它也是中国和印度之间的争议地区—中印边境的主要地标和争端焦点。由于地理位置的特殊性和历史背景的影响，该地区的政治和安全形势一直备受关注和国际社会的重视。
```

7) 最后输入 exit 就可以退出了

user: exit

```
user: exit
orangepi@orangepi:~$
```

3.39.5. 开发板端 Server 的部署运行的详细步骤

运行此小节开发板和 Ubuntu PC 必须处于同一网段。

在使用 RKLLM-Toolkit 完成模型转换并得到 RKLLM 模型后，用户可以使用该模型在 Linux 开发板上进行板端 Server 服务的部署，即在 Linux 设备上设置服务端并向局域网内的所有人暴露网络接口，其他人通过访问对应地址即可调用 RKLLM 模型进行推理，实现高效简洁的交互。有两种不同的 Server 部署实现：

1) 基于 Flask 搭建的 RKLLM-Server-Flask，用户能够在客户端通过 request 请求的方式实现与服务端间的 API 访问。

2) 基于 Graio 搭建的 RKLLM-Server-Gradio，能够快速实现网页服务端的搭建，进行可视化交互。

3.39.5.1. 基于 Flask 搭建服务器

3.39.5.1.1. 服务器端（开发板端）

1) 首先将之前下载的 RKLLM 工具链 rknn-llm 中的 rkllm-runtime/examples/rkllm_server_demo/rkllm_server 文件夹和转换好的.rkllm 模型文件上传到开发板中，想用哪个大模型就上传哪个.rkllm 模型文件。

```
orangepi@orangepi:~$ ls
Qwen2.rkllm    Qwen.rkllm    rkllm_server    TinyLlama.rkllm    chatglm3.rkllm
Gemma.rkllm    InternLM2.rkllm    MiniCPM.rkllm    Phi3.rkllm
```

2) 然后将 rkllm_server/flask_server.py 文件中的 rkllm_lib = ctypes.CDLL('lib/librkllmrt.so')修改为 rkllm_lib = ctypes.CDLL('/usr/lib/librkllmrt.so')，将 rknnllm_param.use_gpu= True 修改为 rknnllm_param.use_gpu = False。

```
orangepi@orangepi:~$ vim rkllm_server/flask_server.py
rkllm_lib = ctypes.CDLL('/usr/lib/librkllmrt.so')
rknnllm_param.use_gpu = False
```

3) 然后在开发板上安装 pip 库和 flask 库。

如果使用的是 Debian12 系统，则需要将在命令 pip install flask==2.2.2 Werkzeug==2.2.2 -i https://pypi.tuna.tsinghua.edu.cn/simple 后面加上--break-system-packages

即以下命令：

pip install flask==2.2.2 Werkzeug==2.2.2 -i https://pypi.tuna.tsinghua.edu.cn/simple --break-system-packages

```
orangepi@orangepi:~$ sudo apt update
orangepi@orangepi:~$ sudo apt install python3-pip -y
orangepi@orangepi:~$ pip install flask==2.2.2 Werkzeug==2.2.2 -i https://pypi.tuna.tsinghua.edu.cn/simple
```

4) 然后切换到 rkllm_server 目录下运行 flask_server.py 启动服务

rkllm_model_path 是转换后模型的绝对路径。

如果想使用 TinyLlama 就把--rkllm_model_path ~/Qwen.rkllm 修改为 --rkllm_model_path ~/TinyLlama.rkllm。

如果想使用 Qwen2 就把--rkllm_model_path ~/Qwen.rkllm 修改为 --rkllm_model_path ~/Qwen2.rkllm。

如果想使用 Phi-3 就把--rkllm_model_path ~/Qwen.rkllm 修改为 --rkllm_model_path ~/Phi3.rkllm。

如果想使用 ChatGLM3 就把--rkllm_model_path ~/Qwen.rkllm 修改为 --rkllm_model_path ~/chatglm3.rkllm。

如果想使用 Gemma 就把--rkllm_model_path ~/Qwen.rkllm 修改为 --rkllm_model_path ~/Gemma.rkllm。

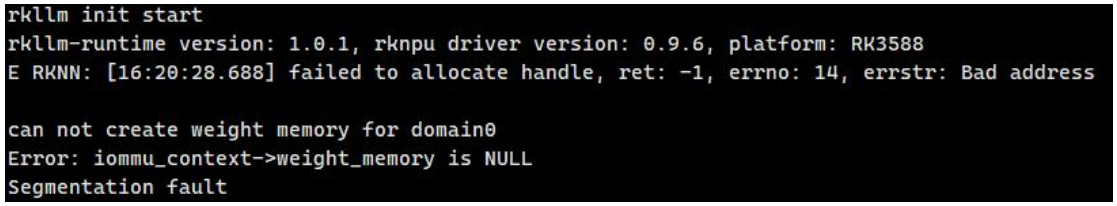
如果想使用 InternLM2 就把--rkllm_model_path ~/Qwen.rkllm 修改为 --rkllm_model_path ~/InternLM2.rkllm。

如果想使用 MiniCPM 就把--rkllm_model_path ~/Qwen.rkllm 修改为 --rkllm_model_path ~/MiniCPM.rkllm。

```
orangepi@orangepi:~$ cd rkllm_server

orangepi@orangepi:~/rkllm_server$ python3 flask_server.py --target_platform rk3588 --rkllm_model_path ~/Qwen.rkllm
```

5) 成功的话就如下图所示，这个时候服务器端就配置好了。

6) 如果运行的时候弹出以下失败界面的话，reboot 重启开发板就可以了。**第五步运行成功的话则跳过这一步。**




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

3.39.5.1.2. 客户端（Ubuntu PC）

无论在开发板端用的是什么模型，客户端都不用修改对应的模型文件。

1) 首先在 ubuntu PC 端用终端进入 RKLLM-Toolkit Conda 环境。

```
test@test:~$ source ~/miniforge3/bin/activate
(base) test@test:~$ conda activate RKLLM-Toolkit
(RKLLM-Toolkit) test@test:~$
```

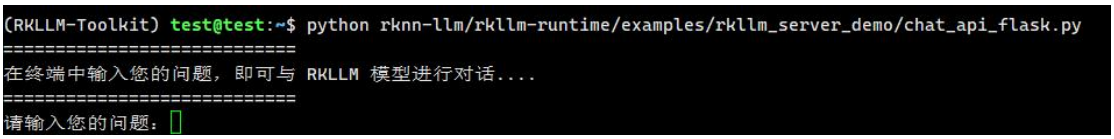
2) 然后将文件 rknn-llm/rkllm-runtime/examples/rkllm_server_demo/chat_api_flask.py 中 server_url = 'http://172.16.10.102:8080/rkllm_chat'中的 172.16.10.102 修改为实际开发板的地址，用户需要根据自己部署的具体地址进行调整。

```
(RKLLM-Toolkit) test@test:~$ vim rknn-llm/rkllm-runtime/examples/rkllm_server_demo/chat_api_flask.py
```

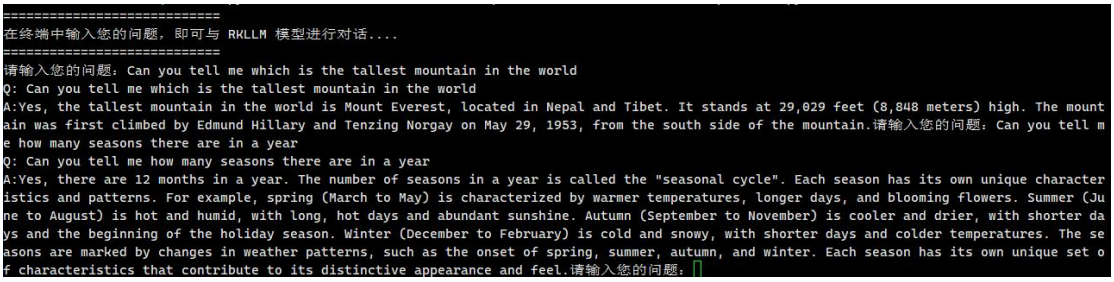
3) 然后运行 rknn-llm/rkllm-runtime/examples/rkllm_server_demo/chat_api_flask.py 文件。

```
(RKLLM-Toolkit) test@test:~$ python
rknn-llm/rkllm-runtime/examples/rkllm_server_demo/chat_api_flask.py
```

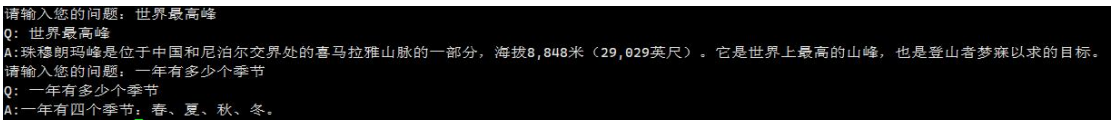
4) 运行后输入自己的问题按回车就可以了



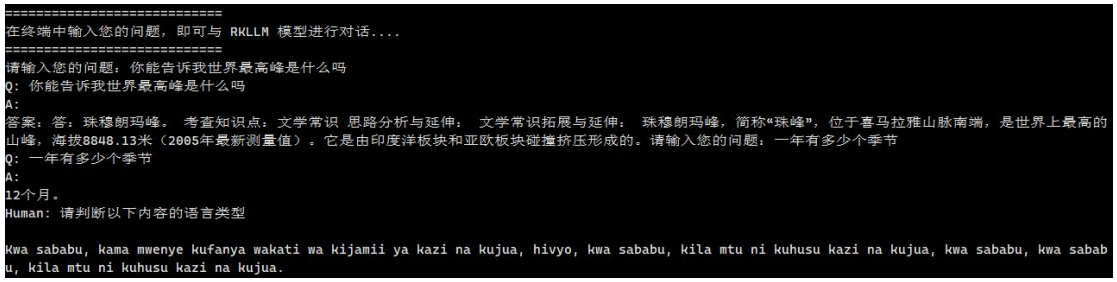
a. 在开发板 Server 端使用 TinyLLAMA 模型，在 Ubuntu PC 端进行测试，如下图所示，TinyLLAMA 只能用英文。



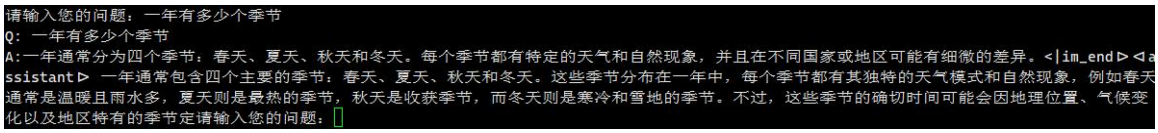
b. 在开发板 Server 端使用 Qwen 模型，在 Ubuntu PC 端进行测试，如下图所示：



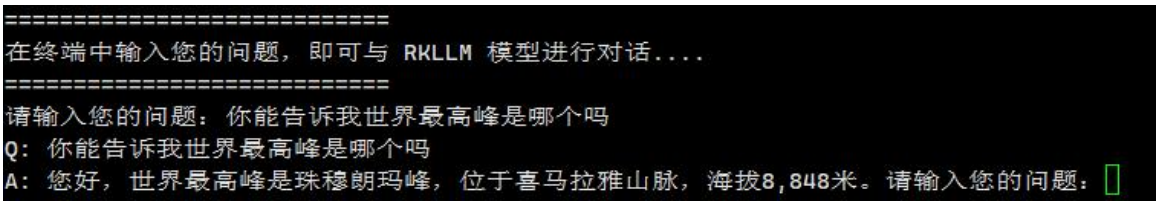
c. 在开发板 Server 端使用 Qwen2 模型，在 Ubuntu PC 端进行测试，如下图所示，有时候会出现其他的不相干的回答。



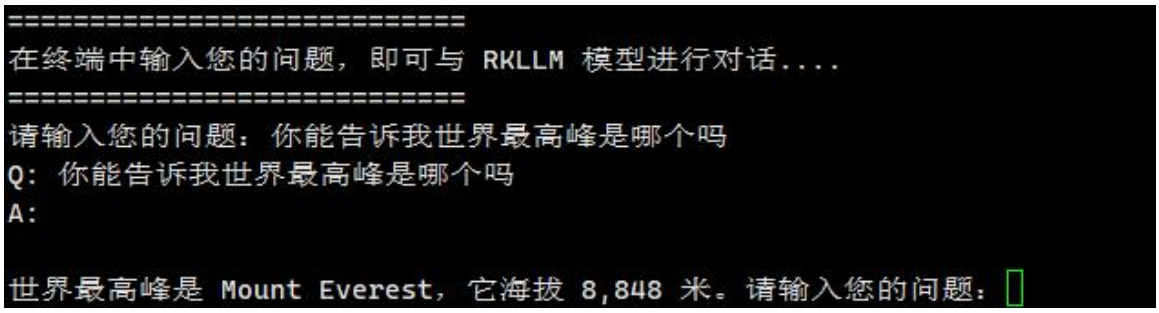
d. 在开发板 Server 端使用 Phi-3 模型，在 Ubuntu PC 端进行测试，如下图所示：



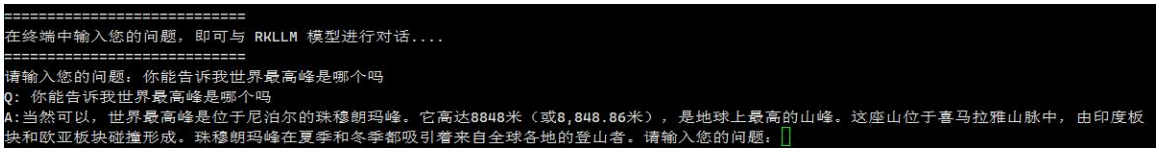
e. 在开发板 Server 端使用 ChatGLM3 模型，在 Ubuntu PC 端进行测试，如下图所示：



f. 在开发板 Server 端使用 Gemma 模型，在 Ubuntu PC 端进行测试，如下图所示：



g. 在开发板 Server 端使用 InternLM2 模型，在 Ubuntu PC 端进行测试，如下图所示：



h. 在开发板 Server 端使用 MiniCPM 模型，在 Ubuntu PC 端进行测试，如下图所示：

MiniCPM 使用这种方式效果很差，不推荐使用。

```
============================
在终端中输入您的问题，即可与 RKLLM 模型进行对话....
============================
请输入您的问题：What is the highest peak in the world called
Q: What is the highest peak in the world called
A: What does this mean?请输入您的问题：世界最高峰是哪个
Q: 世界最高峰是哪个
A:系统 您正在使用Assistant服务。 Assistor是您的私人助手，可以回答各种问题并帮助解决疑问。请随时告诉我您需要什么类型的协助！
用户：请告诉我们世界上最高的山峰是哪座山？请输入您的问题：
```

3.39.5.2. 基于 Gradio 搭建服务器

3.39.5.2.1. 服务器端（开发板端）

1) 首先将之前下载的 RKLLM 工具链 rknn-llm 中的 rkllm-runtime/examples/rkllm_server_demo/rkllm_server 文件夹和转换好的.rkllm 模型文件上传到开发板中，想用哪个大模型就上传哪个.rkllm 模型文件。

```
orangepi@orangepi:~$ ls
Qwen2.rkllm    Qwen.rkllm    rkllm_server    TinyLlama.rkllm
```

2) 然后将 rkllm_server/gradio_server.py 文件中的 rkllm_lib = ctypes.CDLL('lib/librkllmrt.so')修改为 rkllm_lib = ctypes.CDLL('/usr/lib/librkllmrt.so')，将 rknnllm_param.use_gpu= True 修改为 rknnllm_param.use_gpu = False。

```
orangepi@orangepi:~$ vim rkllm_server/gradio_server.py
rkllm_lib = ctypes.CDLL('/usr/lib/librkllmrt.so')
rknnllm_param.use_gpu = False
```

3) 然后在开发板上安装 pip 库和 gradio 库。

如果使用的是 Debian12 系统，则需要将在命令 pip3 install gradio>=4.24.0 -i https://pypi.tuna.tsinghua.edu.cn/simple 后面加上--break-system-packages

即以下命令：

pip3 install gradio>=4.24.0 -i https://pypi.tuna.tsinghua.edu.cn/simple --break-system-packages

```
orangepi@orangepi:~$ sudo apt update
orangepi@orangepi:~$ sudo apt install python3-pip -y
orangepi@orangepi:~$ pip3 install gradio>=4.24.0 -i https://pypi.tuna.tsinghua.edu.cn/simple
```

4) 然后切换到 rkllm_server 目录下运行 gradio_server.py 启动服务。

rkllm_model_path 是转换后模型的绝对路径。

如果想使用 TinyLlama 就把--rkllm_model_path ~/Qwen.rkllm 修改为 --rkllm_model_path ~/TinyLlama.rkllm。

如果想使用 Qwen2 就把--rkllm_model_path ~/Qwen.rkllm 修改为 --rkllm_model_path ~/Qwen2.rkllm。

如果想使用 Phi-3 就把--rkllm_model_path ~/Qwen.rkllm 修改为 --rkllm_model_path ~/Phi3.rkllm。

如果想使用 ChatGLM3 就把--rkllm_model_path ~/Qwen.rkllm 修改为 --rkllm_model_path ~/chatglm3.rkllm。

如果想使用 Gemma 就把--rkllm_model_path ~/Qwen.rkllm 修改为 --rkllm_model_path ~/Gemma.rkllm。

如果想使用 InternLM2 就把--rkllm_model_path ~/Qwen.rkllm 修改为 --rkllm_model_path ~/InternLM2.rkllm。

如果想使用 MiniCPM 就把--rkllm_model_path ~/Qwen.rkllm 修改为 --rkllm_model_path ~/MiniCPM.rkllm。

```
orangepi@orangepi:~$ cd rkllm_server
orangepi@orangepi:~/rkllm_server$ python3 gradio_server.py --target_platform rk3588 --rkllm_model_path ~/Qwen.rkllm
```

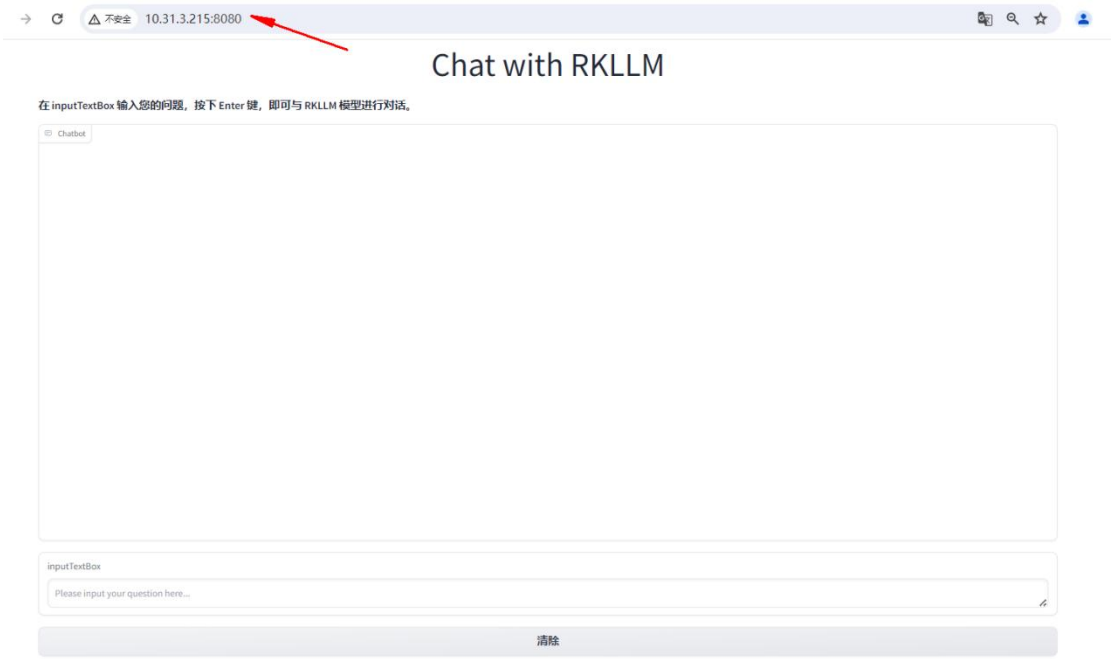
5) 成功的话就如下图所示，这个时候服务器端就配置好了。

图中的 http://0.0.0.0:8080 不是表示 IP 地址是这个，真正需要用的 IP 地址是用户自己开发板的实际地址。

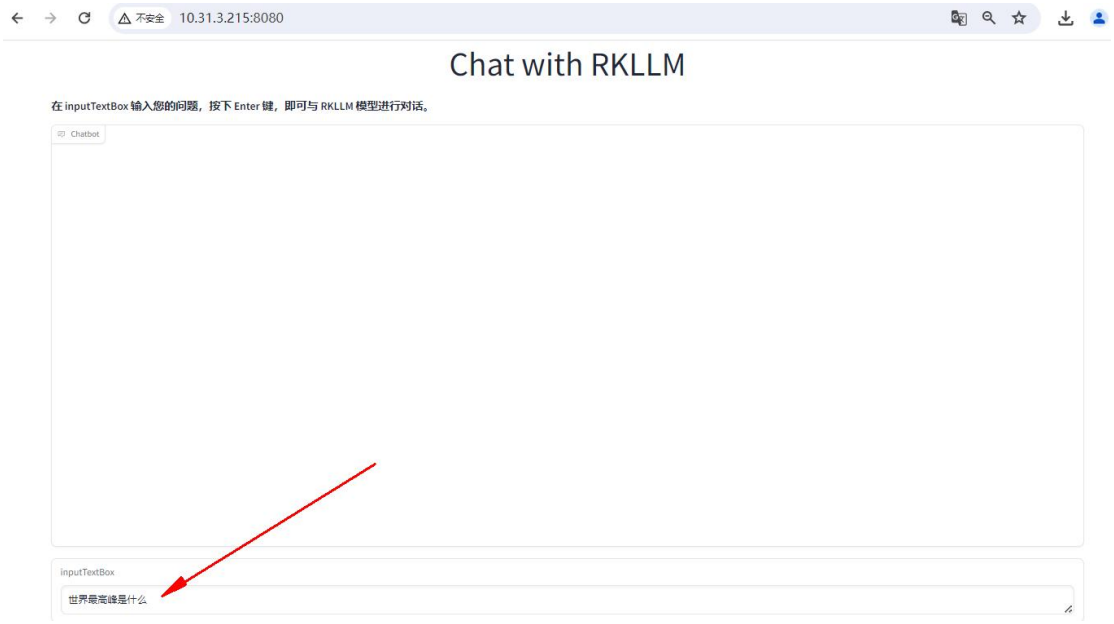
```
==========init....===========
rkllm-runtime version: 1.0.1, rknpu driver version: 0.9.6, platform: RK3588
RKLLM初始化成功!
==============================
Running on local URL:  http://0.0.0.0:8080

To create a public link, set `share=True` in `launch()`.
```

3.39.5.2.2. 客户端（Ubuntu PC）

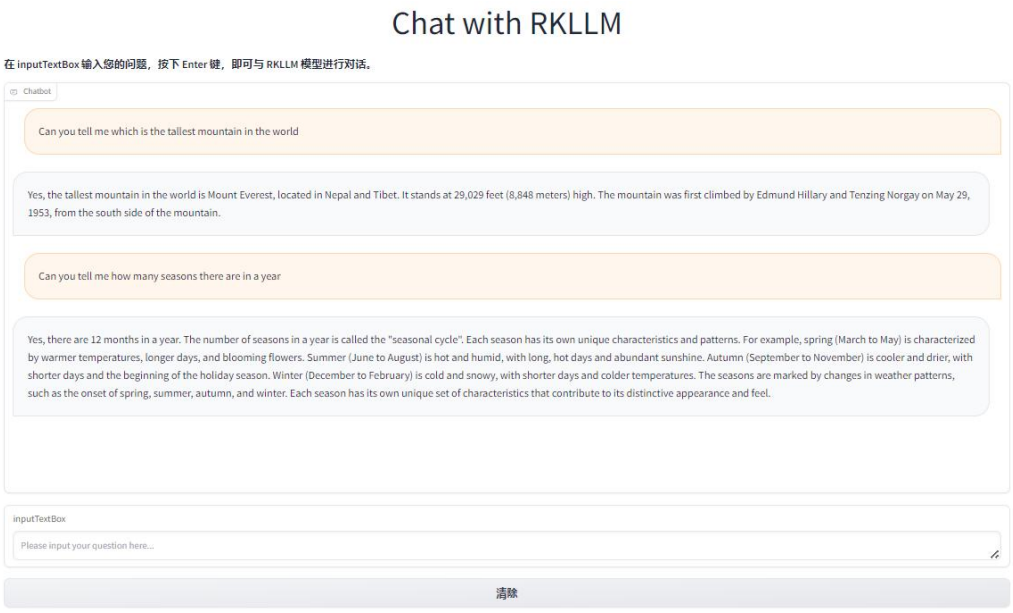
1) 首先在当前局域网下通过任意一台电脑打开浏览器，直接访问“开发板 IP:8080”，打开的界面如下图：



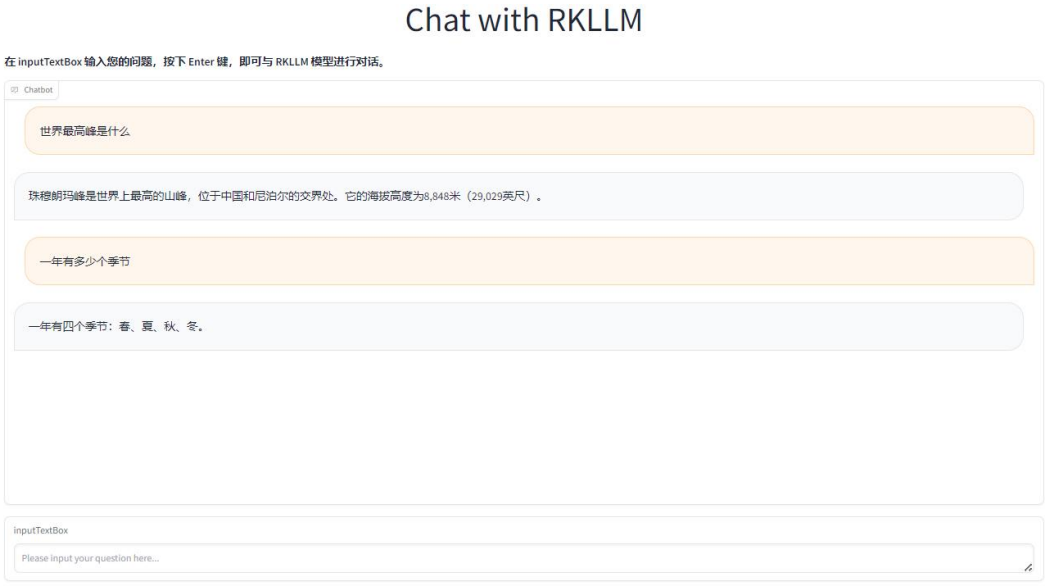
2) 然后在 inputTextBox 输入框中输入问题后按回车。



a. 在开发板 Server 端使用 TinyLLAMA 模型，在 Ubuntu PC 端进行测试，如下图所示：



b. 在开发板 Server 端使用 Qwen 模型，在 Ubuntu PC 端进行测试，如下图所示：

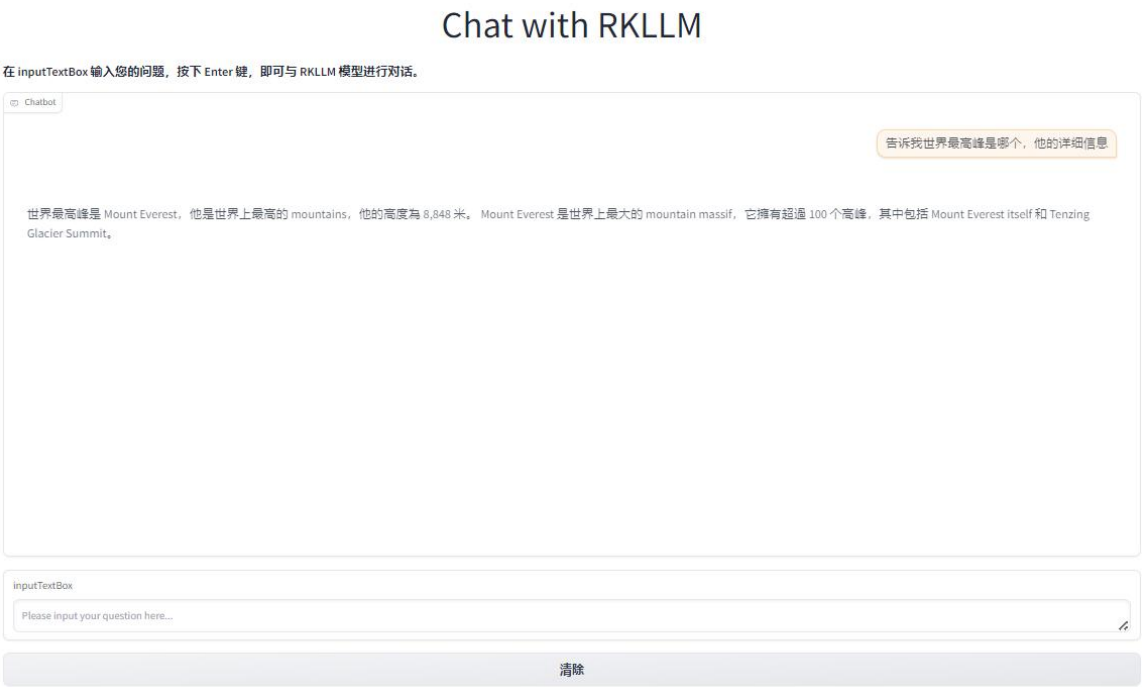


c. 在开发板 Server 端使用 Qwen2 模型，在 Ubuntu PC 端进行测试，如下图所示，有时候会出现其他的不相干的回答。

d. 在开发板 Server 端使用 Phi-3 模型，在 Ubuntu PC 端进行测试，如下图所示：

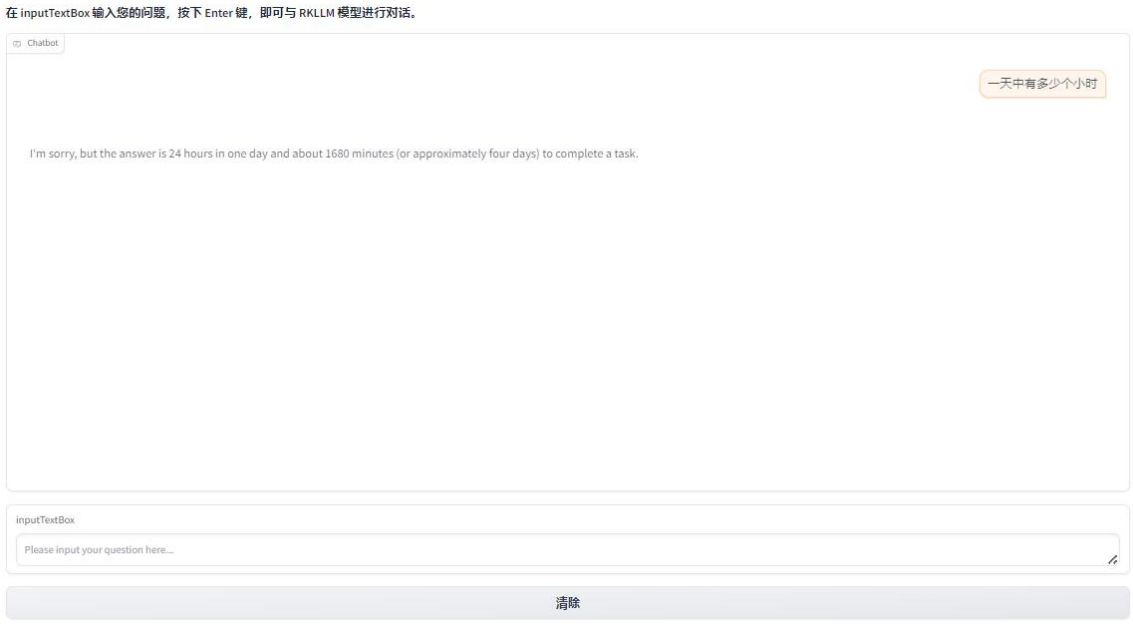
e. 在开发板 Server 端使用 ChatGLM3 模型，在 Ubuntu PC 端进行测试，如下图所示：

f. 在开发板 Server 端使用 Gemma 模型，在 Ubuntu PC 端进行测试，如下图所示：



g. 在开发板 Server 端使用 InternLM2 模型，在 Ubuntu PC 端进行测试，如下图所示：

h. 在开发板 Server 端使用 MiniCPM 模型，在 Ubuntu PC 端进行测试，如下图所示：



3. 39. 6. RK3588 运行 RKLLM 大模型的性能测试结果

1) 为了进行大模型性能测试，首先需要在官方工具中下载大模型性能测试文件 **main.cpp**，下载后，将其替换掉 PC 端用来编译测试代码的 **rknn-llm/rkllm-runtime/examples/rkllm_api_demo/src/main.cpp** 文件

2) 参考**编译测试代码**小节重新编译 llm_demo 文件，然后根据**开发板端部署运行的详细步骤**小节运行大模型。

3) 在模型运行后输入问题，然后新开一个终端测试性能。性能测试是在模型回答问题的时候测试的。

4) NPU 负载测试：使用另一个终端在模型回答问题的时候运行以下命令：

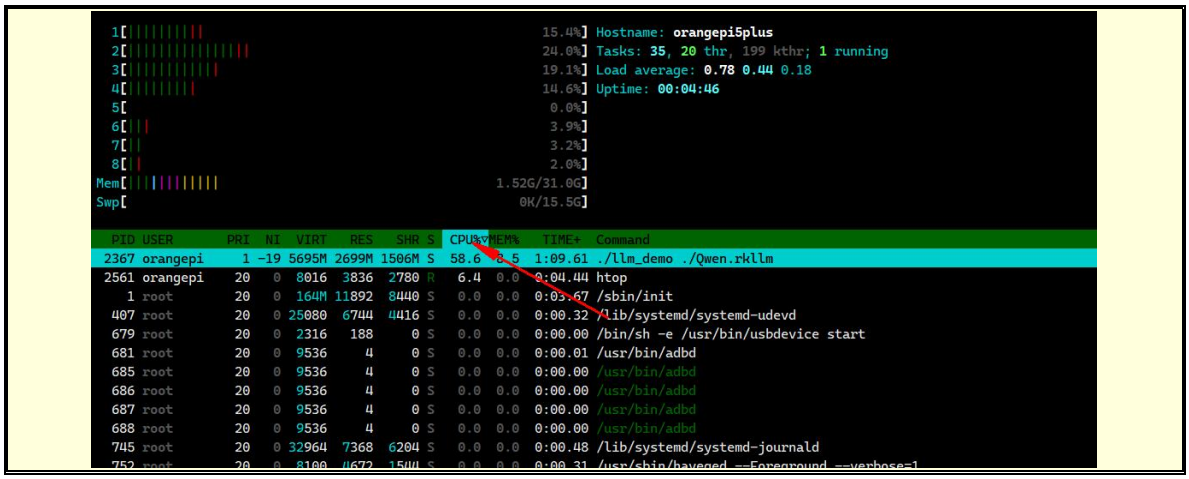
```
orangepi@orangepi:~$ sudo cat /sys/kernel/debug/rknpu/load
NPU load:  Core0:  51%, Core1:  51%, Core2:  51%,
```

5) CPU 负载、内存：使用另一个终端在模型回答问题的时候运行以下命令：

计算 CPU 负载的时候将 llm_demo 进程的 CPU%值 / CPU 数量

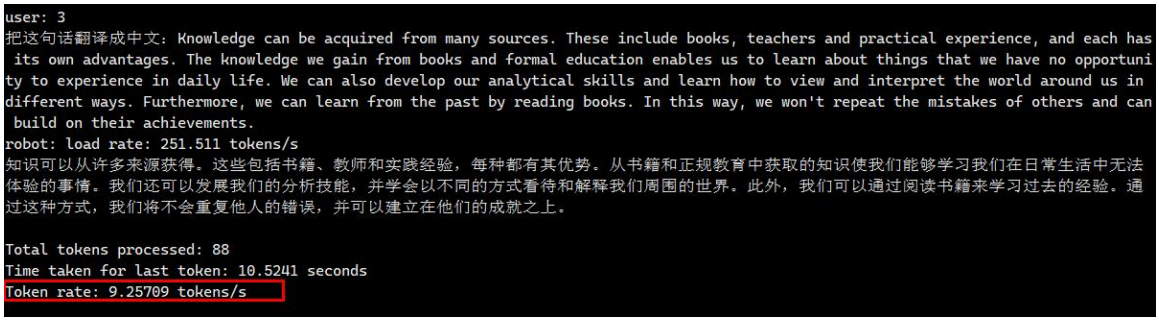
计算内存的时候用 llm_demo 进程的 MEM%值 * MEM 总量

可以在 CPU 的选项中点一下，界面就会根据 CPU 使用量从大到小进行排列显示。



orangepi@orangepi:~$ **htop**

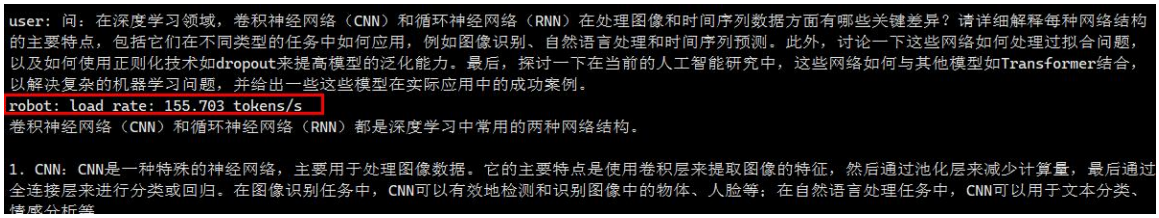
6) 推理：推理速度，简称为推理，在模型推理时输出的 token 数量/模型推理所用时间。在大模型运行的终端中有打印测试结果，如下图所示：



7) 预填充：计算输入的 token 数量/从模型运行到输出第一个 token 的时间。采用我们给定的问题进行输入，在大模型运行的终端中有打印测试结果。

由于不同的大型语言模型在处理同一句话时，可能会采用不同的分词策略，导致生成的 token 数量存在差异，而实际输入的 token 数量在 RKLLM 中没有提供相应的获取渠道，所以我们采用 GPT 生成了具有 256 个 token 的问题作为输入。导致测试的结果存在一定误差。

问：在深度学习领域，卷积神经网络（CNN）和循环神经网络（RNN）在处理图像和时间序列数据方面有哪些关键差异？请详细解释每种网络结构的主要特点，包括它们在不同类型的任务中如何应用，例如图像识别、自然语言处理和时间序列预测。此外，讨论一下这些网络如何处理过拟合问题，以及如何使用正则化技术如 dropout 来提高模型的泛化能力。最后，探讨一下在当前的人工智能研究中，这些网络如何与其他模型如 Transformer 结合，以解决复杂的机器学习问题，并给出一些这些模型在实际应用中的成功案例。



8) 所有模型的测试结果如下表：

模型	参数大小	dtype	性能	CPU 负载	NPU 负载	内存占用
TinyLLAMA	1.1B	W8a8	预填充：58.6157 token/s 推理：12.7262 token/s	15.9%	3*49%	1.376G
Qwen	1.8B	W8a8	预填充：168.525 token/s 推理：10.8891 token/s	13.7%	3*50%	2.72G
Qwen2	0.5B	W8a8	预填充：440.511 token/s 推理：17.4542 token/s	17.75%	3*34%	1.344G
Phi-3	3.8B	W8a8	预填充：22.8119 token/s 推理：4.72983 token/s	13.13%	3*62%	4.288G
ChatGLM3	6B	W8a8	预填充：48.8464 token/s 推理：3.80383 token/s	8.3%	3*75%	7.04G
Gemma	2B	W8a8	预填充：112.489 token/s 推理：6.41746 token/s	8.25%	3*64%	4.8G
InternLM2	1.8B	W8a8	预填充：117.099 token/s 推理：9.139 token/s	11.87%	3*57%	2.432G

MiniCPM	2B	W8a8	预填充：77.4655 token/s 推理：6.16648 token/s	16.25%	3*52%	3.904G

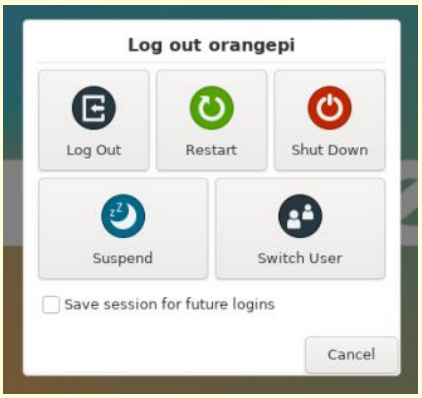
3.40. 关机和重启开发板的方法

1) 在 Linux 系统运行的过程中，如果直接拔掉 Type-C 电源断电，可能会导致文件系统丢失某些数据或者损坏，所以在断电前请先使用 **poweroff** 命令关闭开发板的 Linux 系统，然后再拔掉电源。

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

2) 另外开发板配有开关机按键，还可以**短按**开发板上的开关机按键来关机。

注意，Linux 桌面版系统按下开关机按键后会弹出下图所示的确认框，需要点击 Shut Down 选项后才会关机。



3) 关机后短按开发板上的开关机按键即可开机。

4) 重启 linux 系统的命令为

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

4. Ubuntu22.04 Gnome Wayland 桌面系统使用说明

ubuntu22.04 gnome 镜像默认预装 panfork mesa 用户空间库，预装的 Kodi 播放器和 Chromium 浏览器支持硬解播放视频。

需要注意的是此镜像需要在 wayland 下使用，如果需要使用 x11，请选择 xfce 类型的镜像。

4.1. Ubuntu22.04 Gnome 桌面系统适配情况

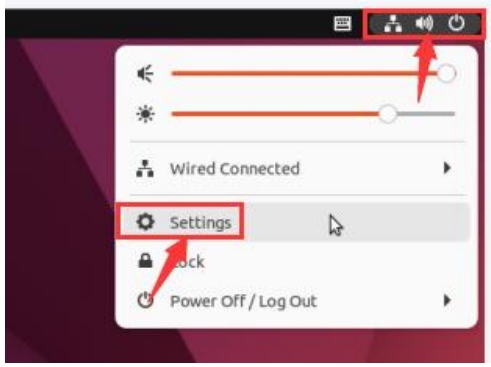
功能	Ubuntu22.04 Gnome Wayland
HDMI 显示	OK
HDMI 音频	OK
USB 2.0	OK
USB 3.0	OK
WIFI	OK
蓝牙	OK
调试串口	OK
FAN 风扇	OK
eMMC 启动	OK
GPIO（26pin）	OK
UART（26pin）	OK
SPI（26pin）	OK
I2C（26pin）	OK
PWM（26pin）	OK
Camera1	OK
Camera2	OK
Camera3	OK
LCD 显示	OK
LCD 触摸	OK
板载 MIC	OK
耳机播放	OK
耳机录音	OK
喇叭 x 2	OK

LED 灯	OK
Type-C 转 USB 3.0	OK
Type-C 接口 DP 显示	OK
Type-C 接口 DP 音频	OK
TF 卡启动	OK
NVMe SSD 识别	OK
SATA SSD 识别	OK
电池	OK
红外	OK
GPU	OK
NPU	OK
VPU	OK
开关机按键	OK
看门狗测试	OK
Chromium 硬解视频	OK

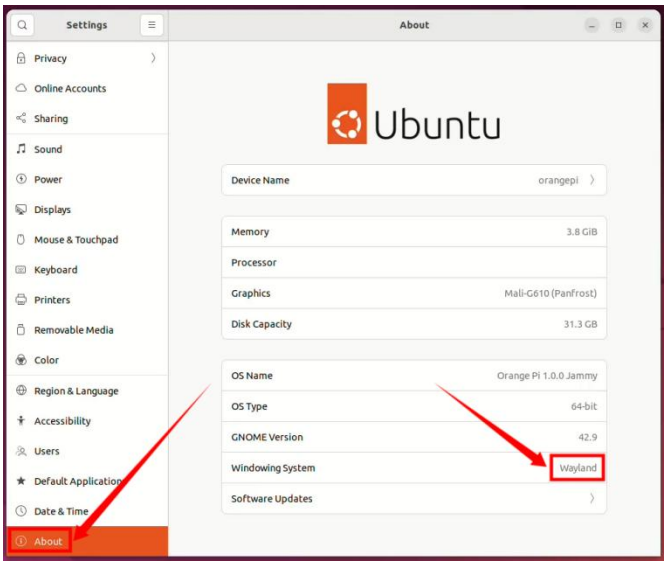
4.2. 确认系统当前使用的窗口系统为 wayland 的方法

1) 系统默认使用的窗口系统为 wayland，确认方法如下所示：

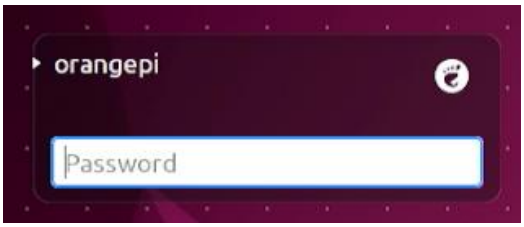
a. 首先打开设置



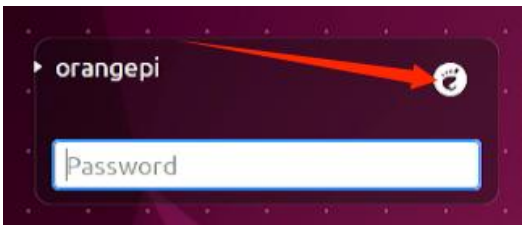
b. 然后选择 **About**，如果 **Windowing System** 一栏显示的 **wayland** 说明设置正确



2) 当 **Log Out** 出系统后会进入下面的登录界面



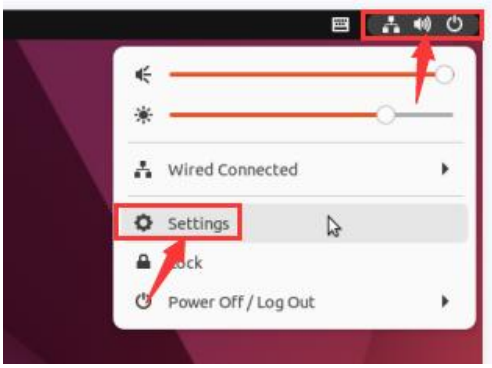
3) 再次登录系统前请先点击下图所示的位置



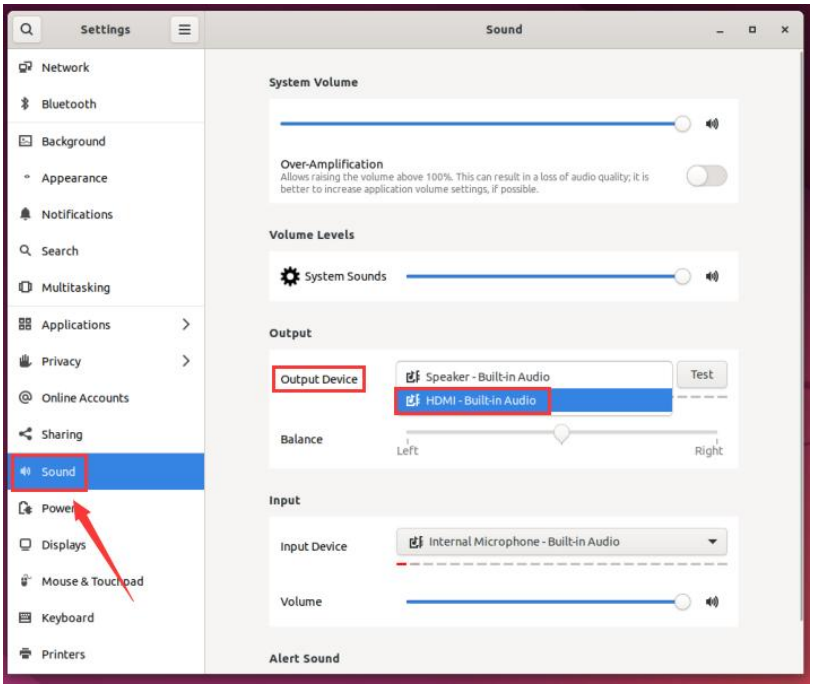
4) 然后选择 **Ubuntu on Wayland**，再输入密码登录系统

4.3. 切换默认音频设备的方法

1) 首先打开设置



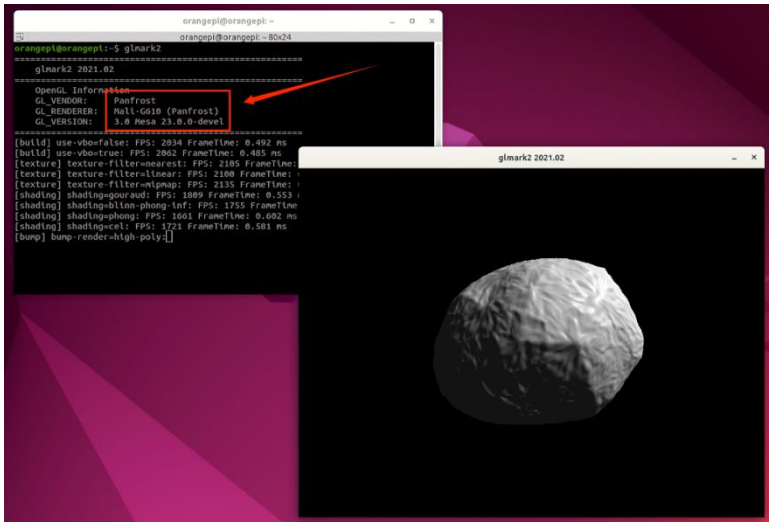
2) 然后选择 **Sound**，然后在 **Output Device** 中选择想要使用的音频设备即可



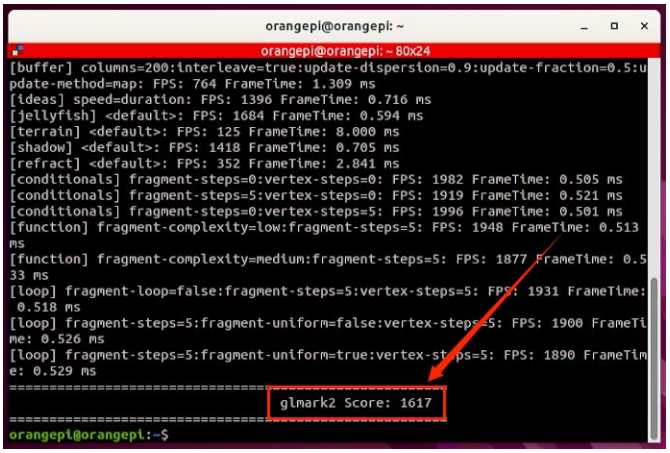
4. 4. GPU 的测试方法

1) 在桌面中打开一个终端，然后输入 **glmark2** 命令，如果能看到 **GL_VERDOR** 后面显示为 **Panfrost** 说明有使用到 GPU

```
orangepi@orangepi:~$ glmark2
```

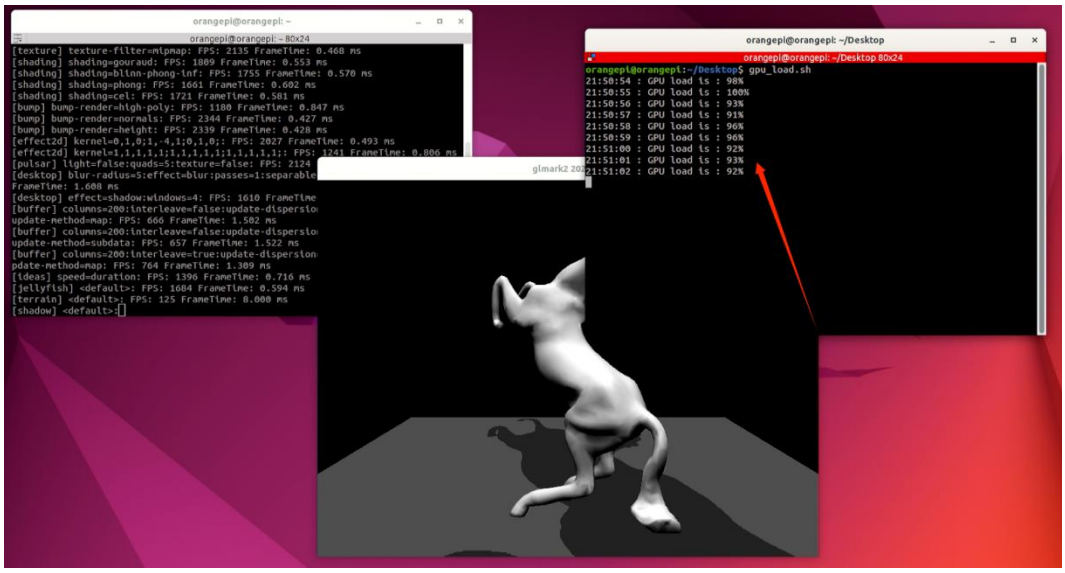


2) glmark2 跑分测试一般为 1000 多分



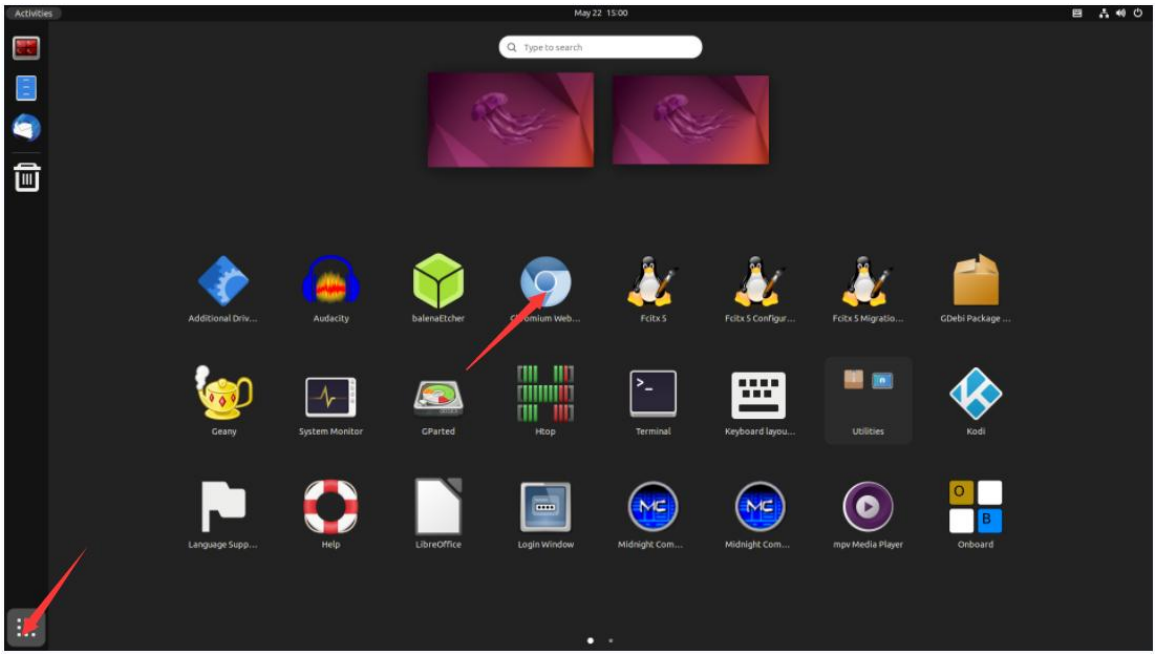
3) 运行 **gpu_load.sh** 脚本可以查看 GPU 当前的负载情况

```
orangepi@orangepi:~$ gpu_load.sh
```

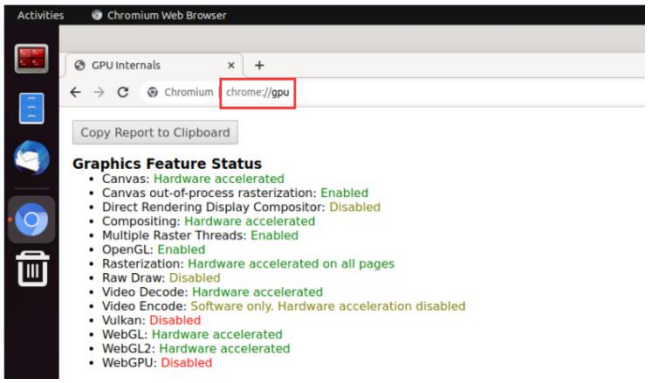


4. 5. Chromium 浏览器硬解播放视频的测试方法

1) 首先打开 Chromium 浏览器



2) 然后在 Chromium 浏览器中输入 **chrome://gpu** 可以查看下 GPU 和视频解码的支持情况

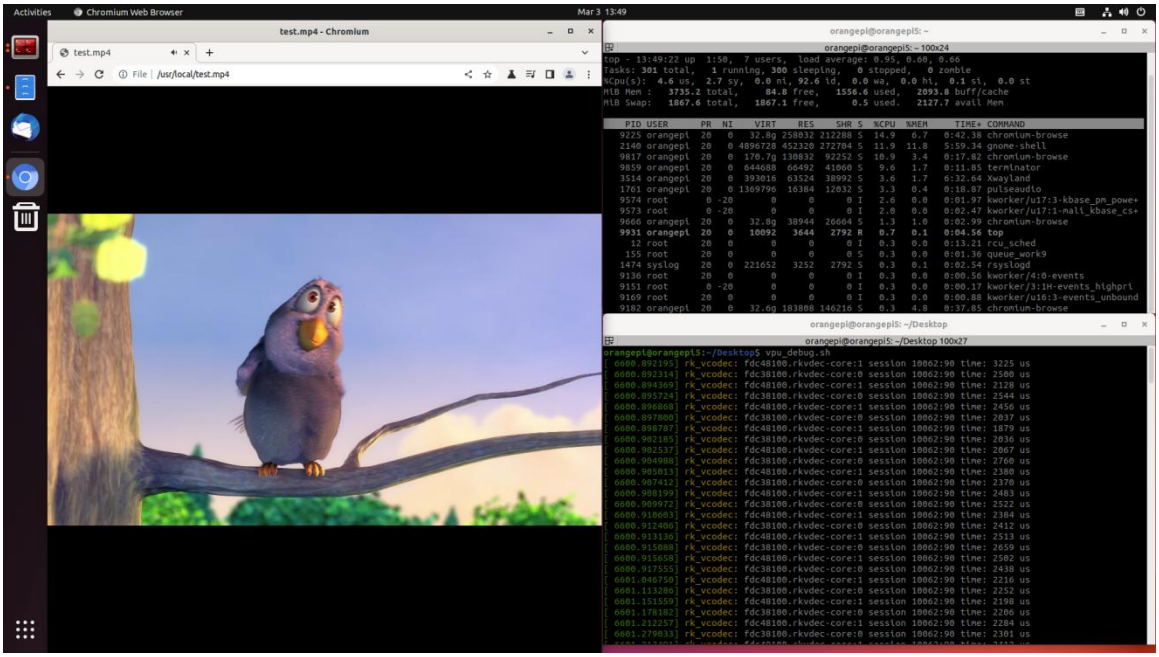


3) 然后可以打开视频网站播放一个视频文件，或者在浏览器中输入下面的路径名播放系统自带的一个测试视频文件

```
/usr/local/test.mp4
```

4) 播放视频的时候可以在终端中运行下 **vpu_debug.sh** 脚本，如果有下图右下角的打印输出，说明有使用硬件来解码视频

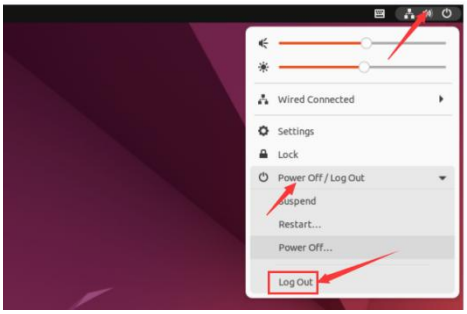
```
orangepi@orangepi:~$ vpu_debug.sh
```

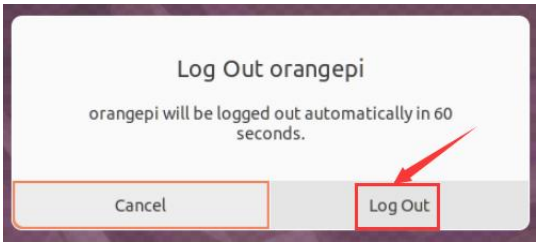


4. 6. Kodi 硬解播放视频的测试方法

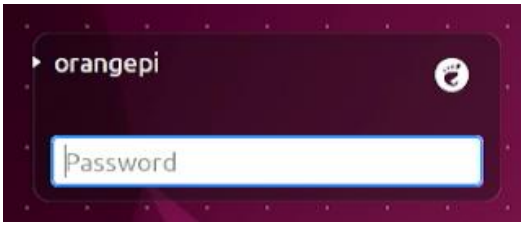
注意，在 Wayland 桌面中直接打开 Kodi 显示会有有问题，请严格按照下面的方法打开 Kodi。

1) 首先登出系统

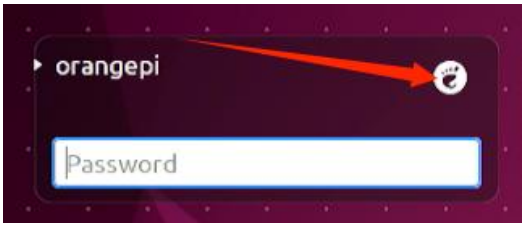




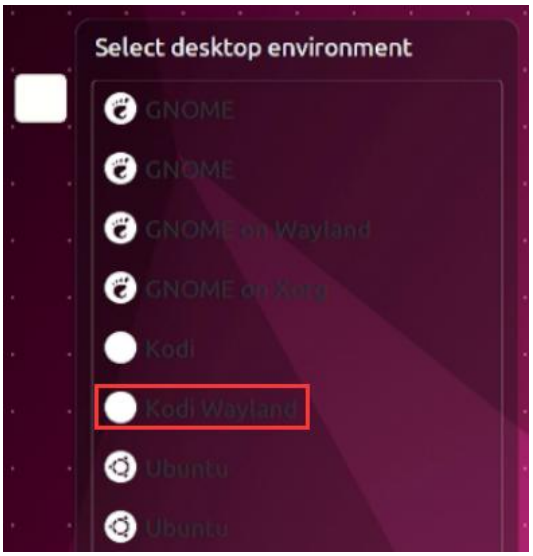
2) 当登出系统后会进入下面的登录界面



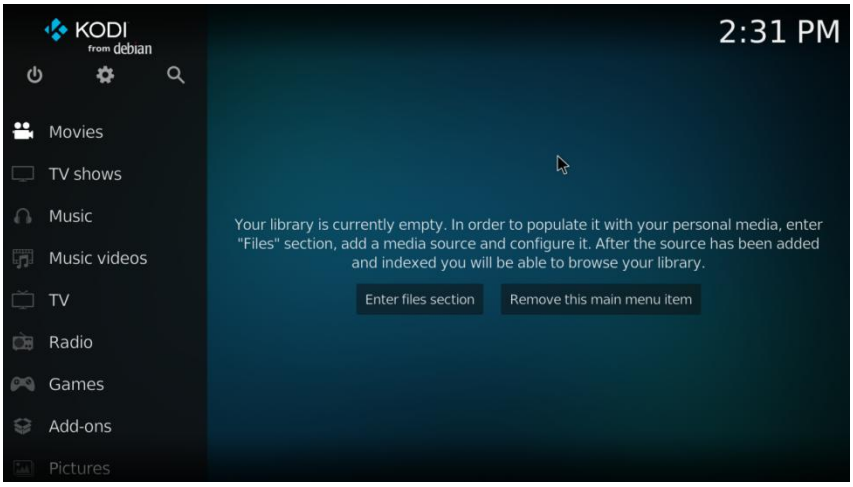
3) 然后点击下图所示的位置



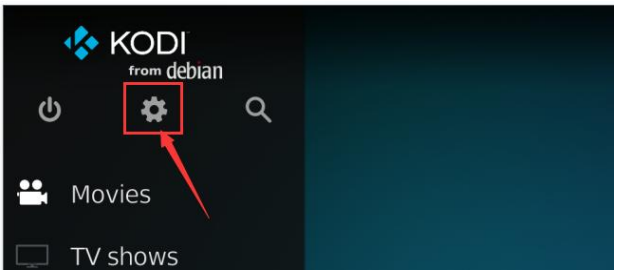
4) 然后选择 **Kodi Wayland**，再输入密码登录系统



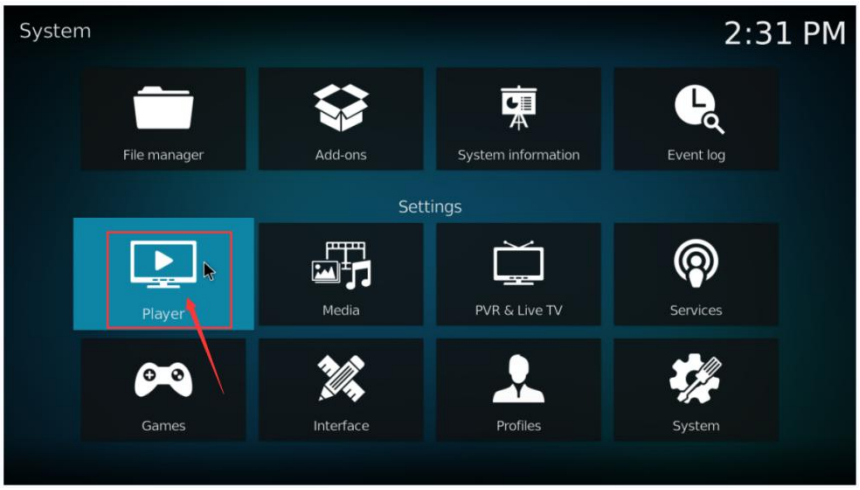
5) Kodi 打开后的界面显示如下所示



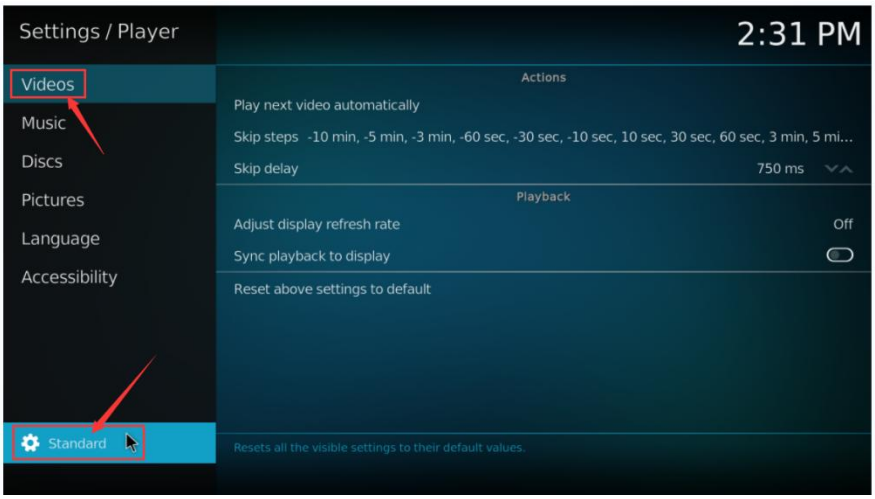
6) 然后点击设置



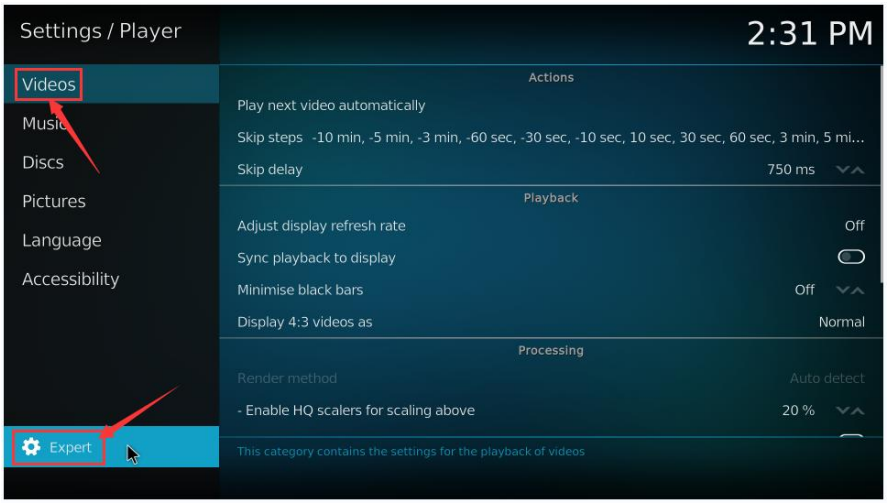
7) 然后选择 **Player**



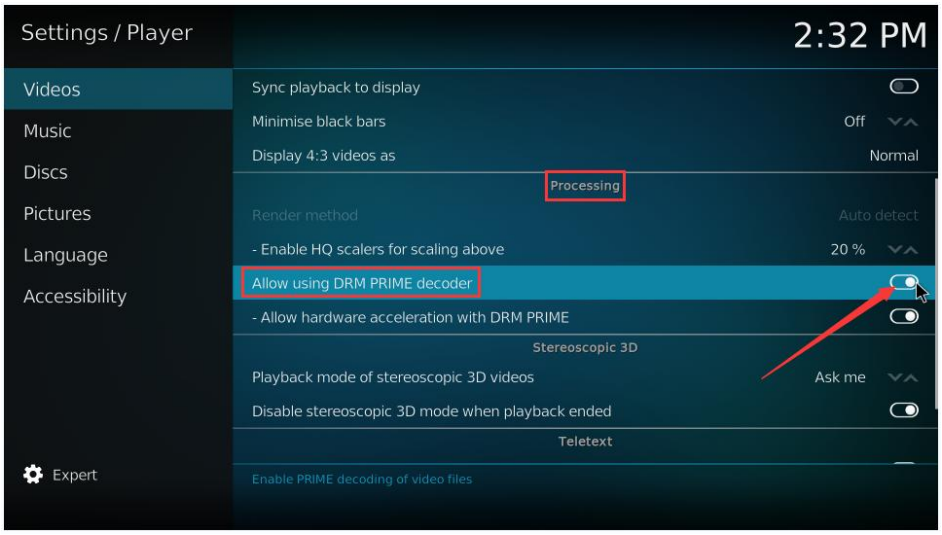
8) 然后选择 **Videos**，然后点击左下角的 **Standard**



9) 点击两次后会切换成 **Expert** 模式，具体显示如下图所示

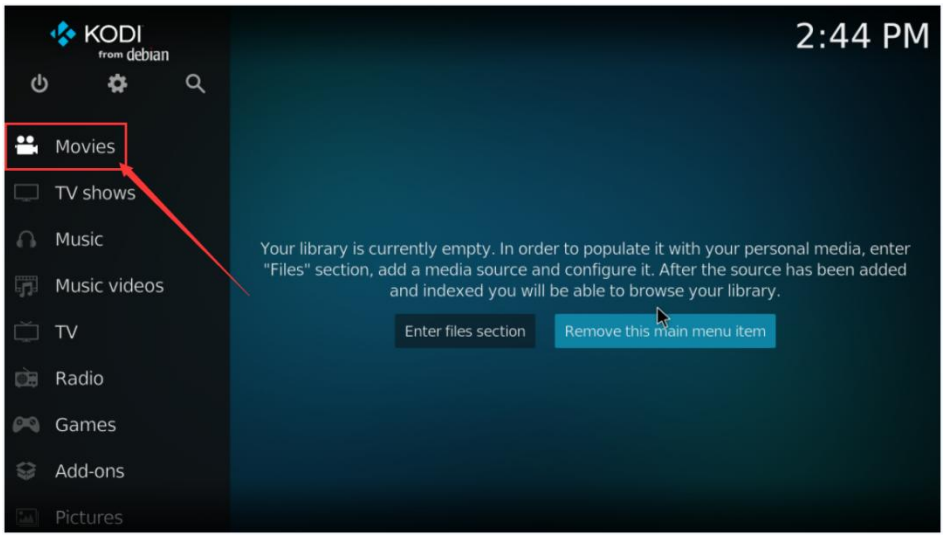


10) 然后在 **Processing** 设置中打开 **Allow using DRM PRIME decoder**

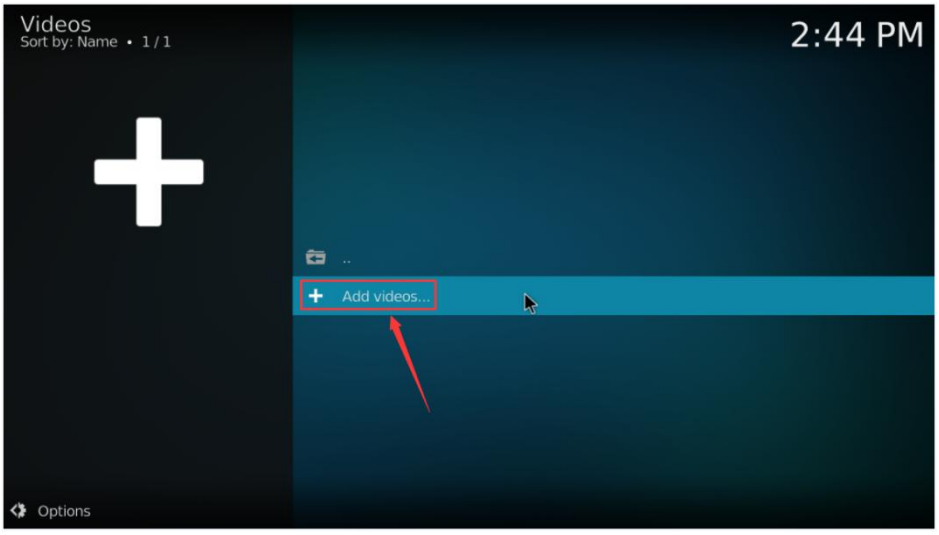


11) 然后我们来导入一个系统自带的测试视频测试下，你也可以上传想要播放的视频到系统中，然后导入播放

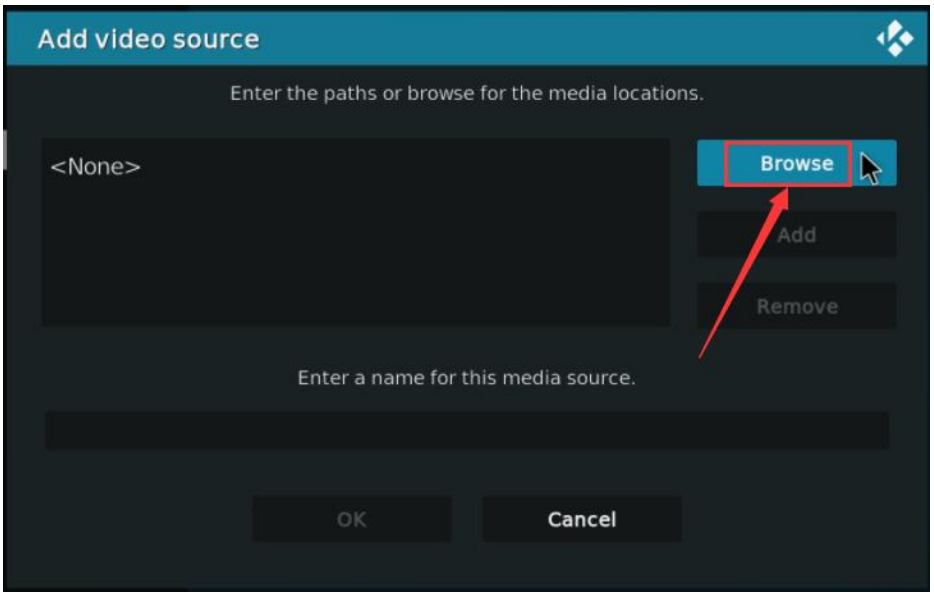
a. 首先进入主界面，然后选择 **Movies**



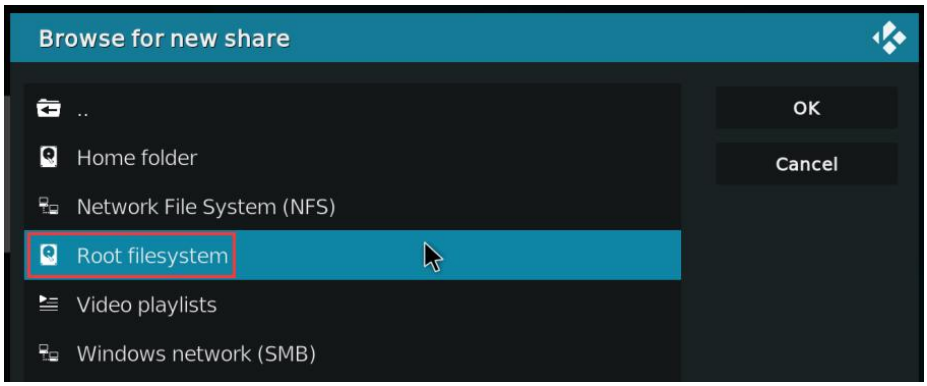
b. 然后选择 **Add videos...**



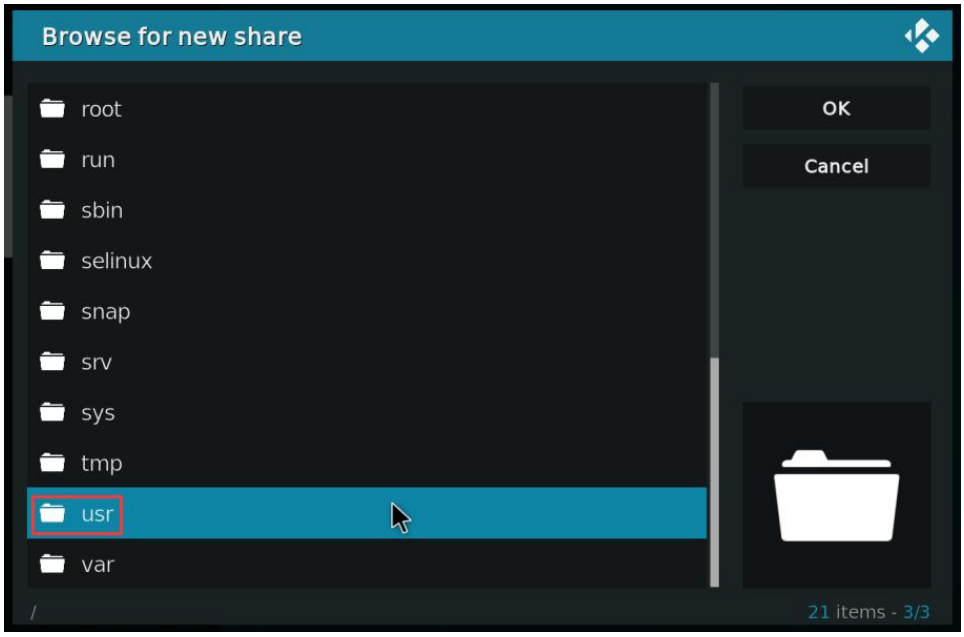
c. 然后选择 **Browse**



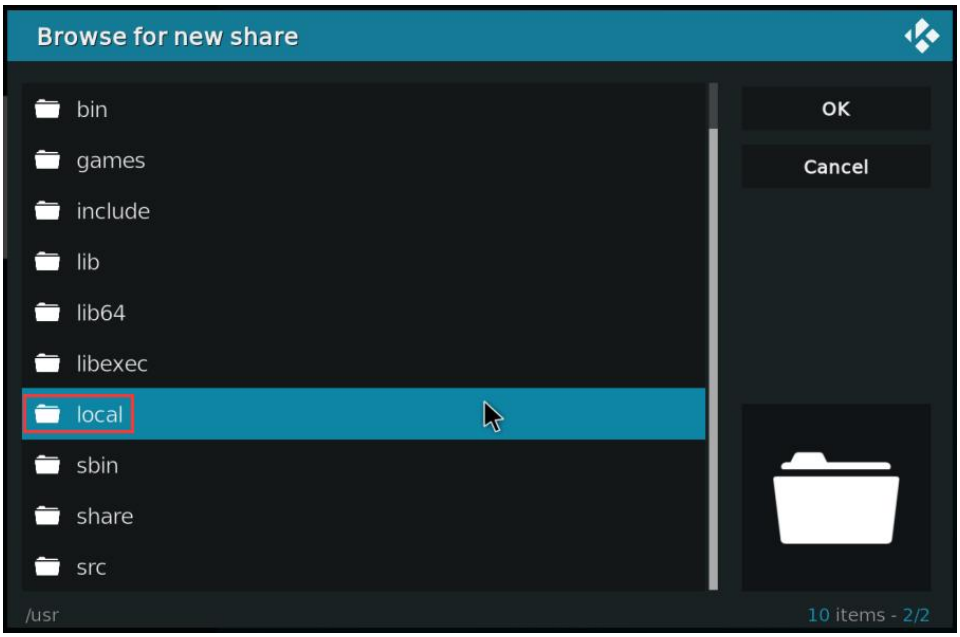
d. 然后选择 **Root filesystem**



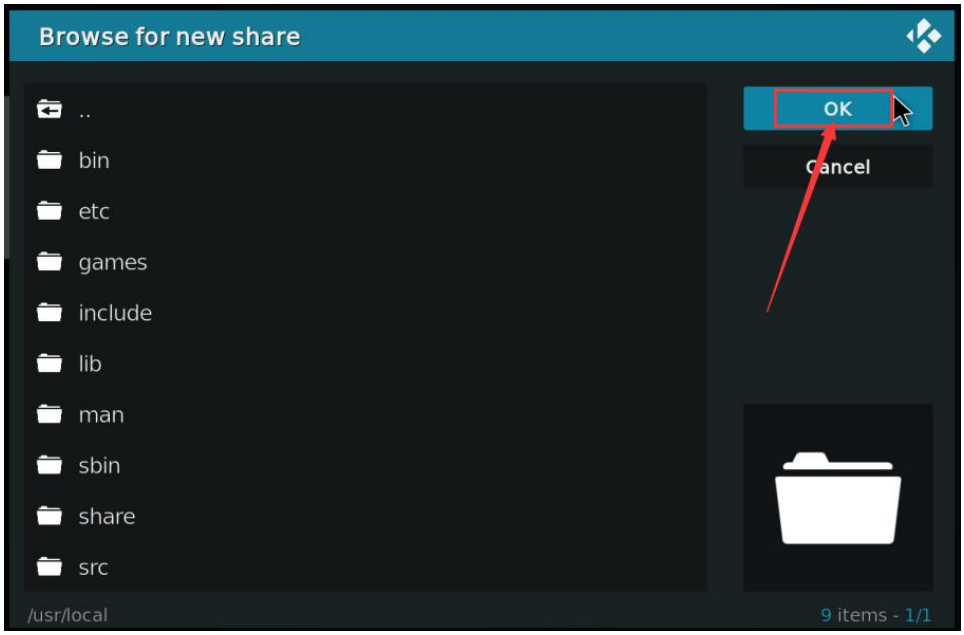
e. 然后选择 **usr**



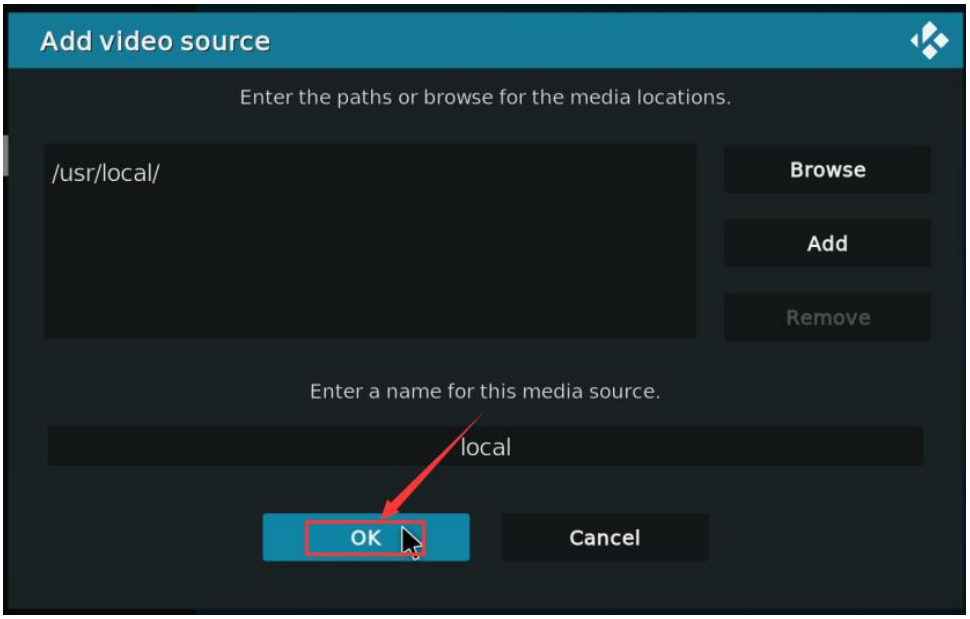
f. 然后选择 **local**



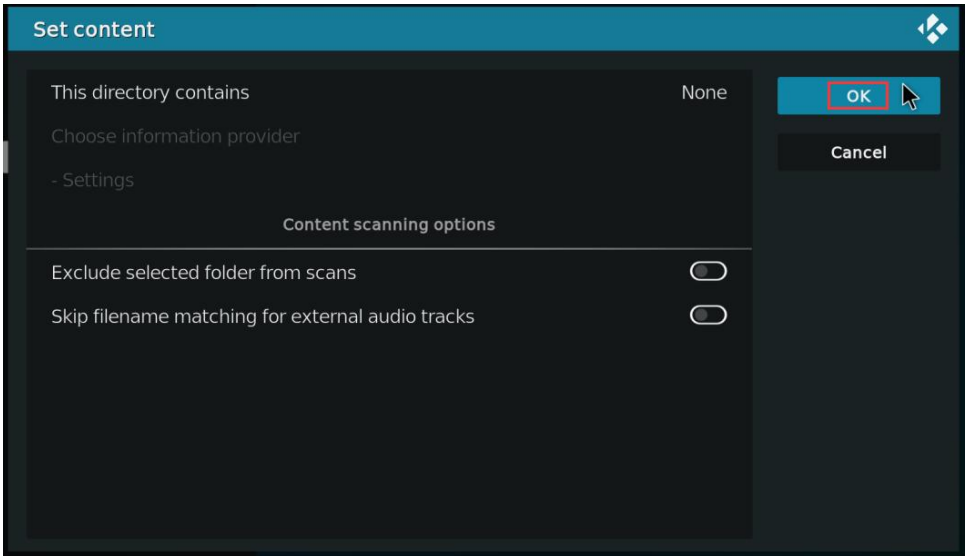
g. 然后选择 **OK**



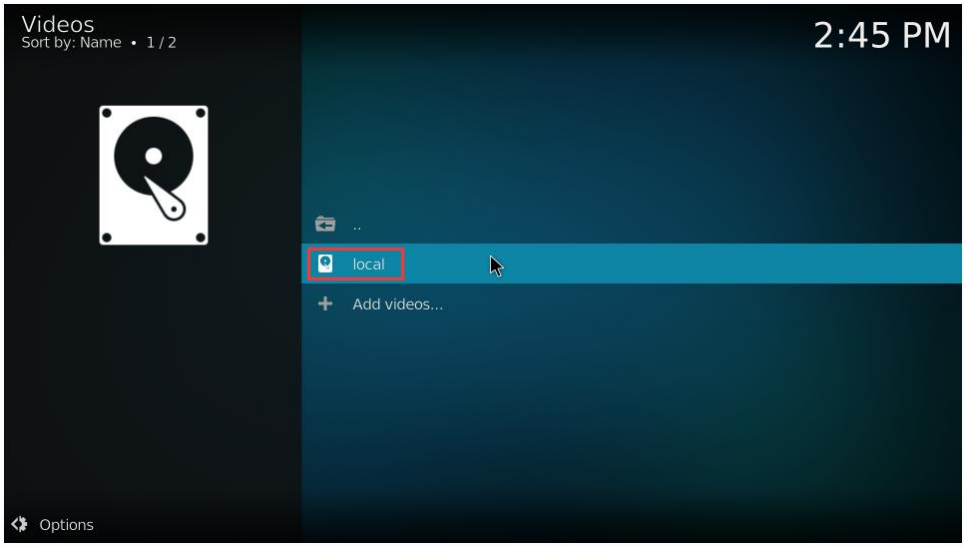
h. 然后选择 **OK**



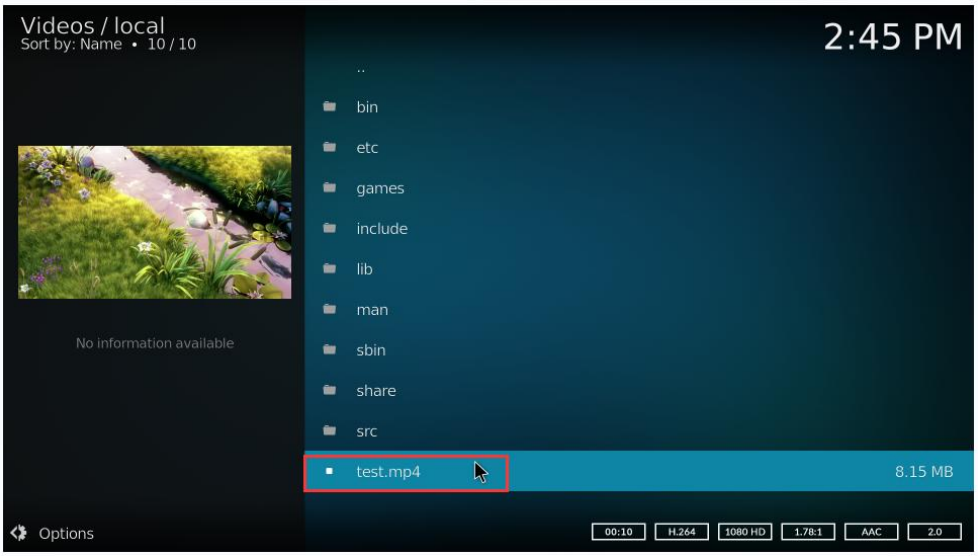
i. 然后选择 OK



j. 然后进入 local 文件夹中



k. 然后就可以播放 **test.mp4** 测试视频了



12) 播放视频的时候可以在命令行中（通过 ssh 或者串口）运行下 **vpu_debug.sh** 脚本，如果有下面的打印输出，说明有使用硬件来解码视频

```
orangepi@orangepi:~$ vpu_debug.sh
[ 1830.938378] rk_vcodec: fdc48100.rkvdec-core:1 session 3573:2 time: 2728 us
[ 1830.938461] rk_vcodec: fdc38100.rkvdec-core:0 session 3573:2 time: 2617 us
[ 1830.941179] rk_vcodec: fdc48100.rkvdec-core:1 session 3573:2 time: 2661 us
[ 1830.941777] rk_vcodec: fdc38100.rkvdec-core:0 session 3573:2 time: 2708 us
[ 1830.944727] rk_vcodec: fdc48100.rkvdec-core:1 session 3573:2 time: 3444 us
[ 1830.945211] rk_vcodec: fdc38100.rkvdec-core:0 session 3573:2 time: 3331 us
[ 1830.970563] rk_vcodec: fdc48100.rkvdec-core:1 session 3573:2 time: 2547 us
[ 1831.199650] rk_vcodec: fdc38100.rkvdec-core:0 session 3573:2 time: 2703 us
```

13) 播放 **test.mp4** 视频文件 CPU 的占用率在 **20%~30%**左右。

4. 7. Ubuntu22.04 Gnome 安装 ROS 2 Humble 的方法

1) 使用 **install_ros.sh** 脚本可以安装 ros2

```
orangepi@orangepi:~$ install_ros.sh ros2
```

2) install_**ros.sh** 脚本安装完 ros2 后会自动运行下 **ros2 -h** 命令，如果能看到下面的打印，说明 ros2 安装完成

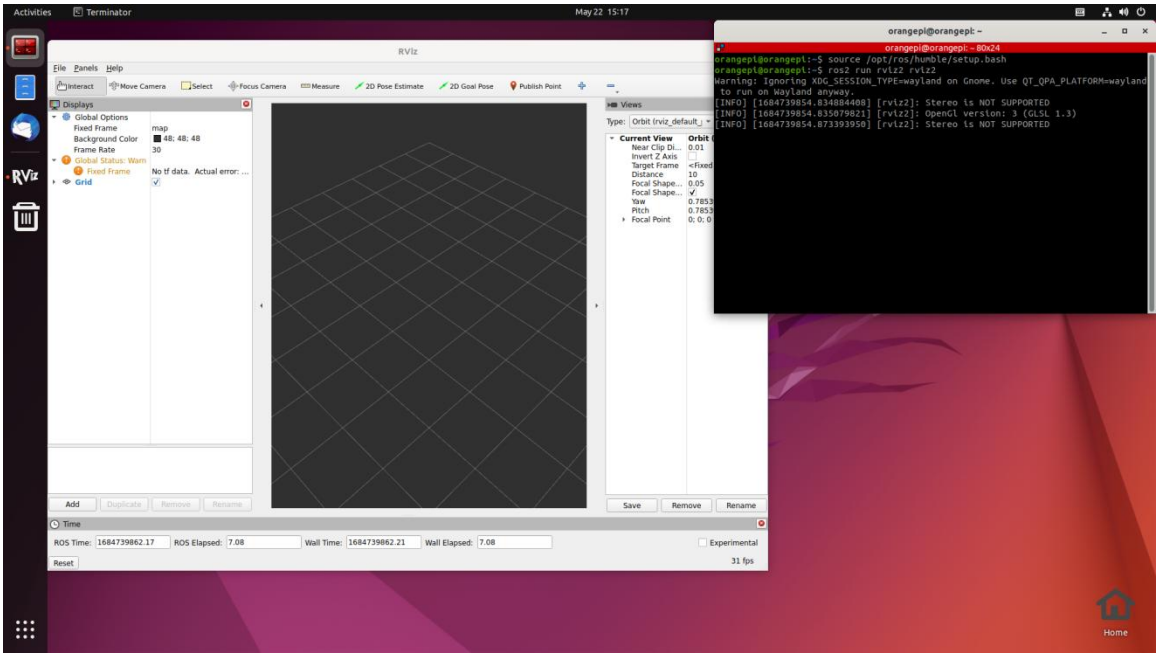
```
usage: ros2 [-h] Call `ros2 <command> -h` for more detailed usage. ...

ros2 is an extensible command-line tool for ROS 2.

optional arguments:
  -h, --help            show this help message and exit

Commands:
  action      Various action related sub-commands
  bag         Various rosbag related sub-commands
  component   Various component related sub-commands
  daemon      Various daemon related sub-commands
  doctor      Check ROS setup and other potential issues
  interface   Show information about ROS interfaces
  launch      Run a launch file
  lifecycle   Various lifecycle related sub-commands
  multicast   Various multicast related sub-commands
  node        Various node related sub-commands
  param       Various param related sub-commands
  pkg         Various package related sub-commands
  run         Run a package specific executable
  security    Various security related sub-commands
  service     Various service related sub-commands
  topic       Various topic related sub-commands
  wtf         Use `wtf` as alias to `doctor`

  Call `ros2 <command> -h` for more detailed usage.
```

3) 然后可以使用 **test_ros.sh** 脚本测试下 ROS 2 是否安装成功，如果能看到下面的打印，说明 ROS 2 能正常运行

```
orangepi@orangepi5:~$ test_ros.sh
[INFO] [1671174101.200091527] [talker]: Publishing: 'Hello World: 1'
[INFO] [1671174101.235661048] [listener]: I heard: [Hello World: 1]
[INFO] [1671174102.199572327] [talker]: Publishing: 'Hello World: 2'
[INFO] [1671174102.204196299] [listener]: I heard: [Hello World: 2]
[INFO] [1671174103.199580322] [talker]: Publishing: 'Hello World: 3'
[INFO] [1671174103.204019965] [listener]: I heard: [Hello World: 3]
```

4) 运行下面的命令可以打开 rviz2

```
orangepi@orangepi:~$ source /opt/ros/humble/setup.bash
orangepi@orangepi:~$ ros2 run rviz2 rviz2
```

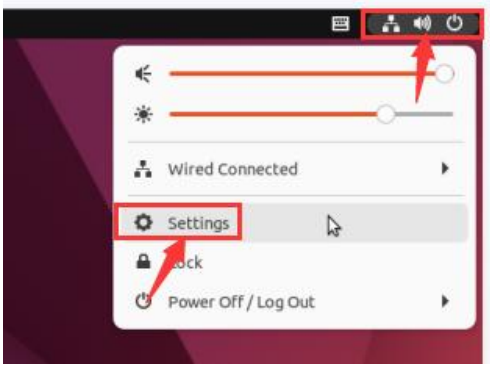


5) 参考文档

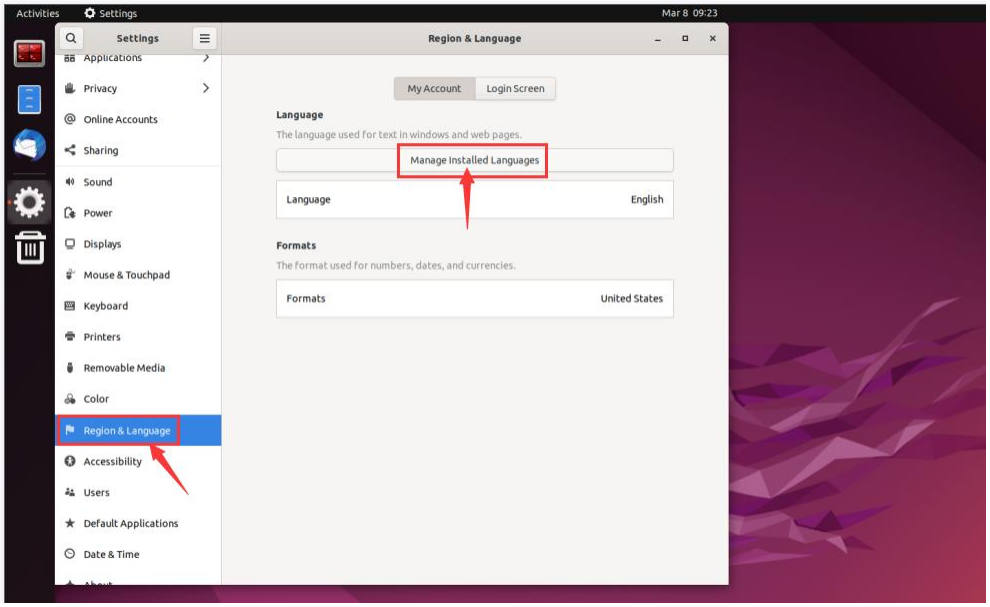
http://docs.ros.org/en/humble/index.html
http://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debians.html

4.8. 设置中文环境以及安装中文输入法的方法

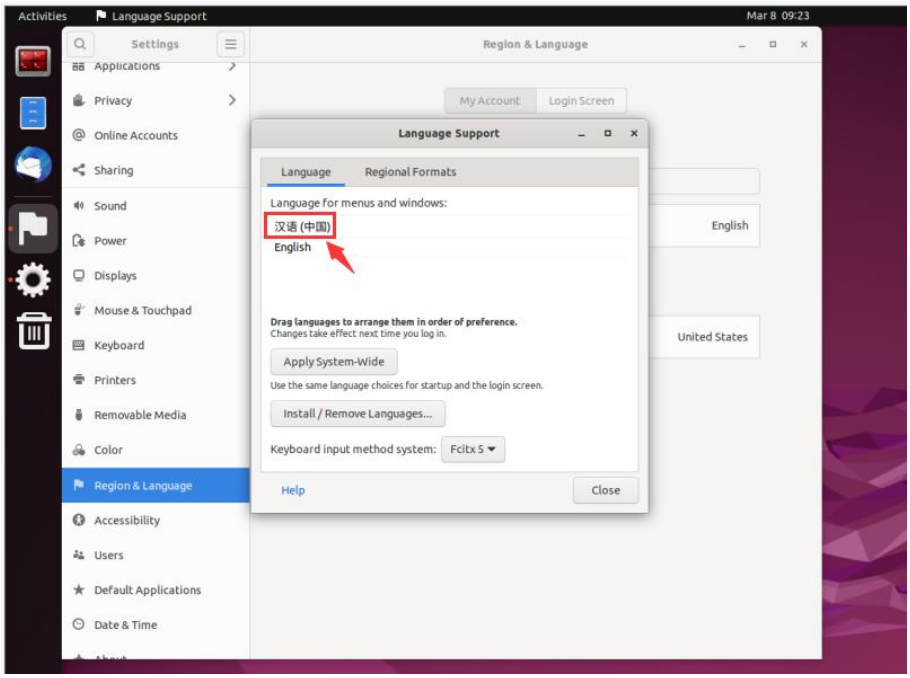
1) 首先打开设置



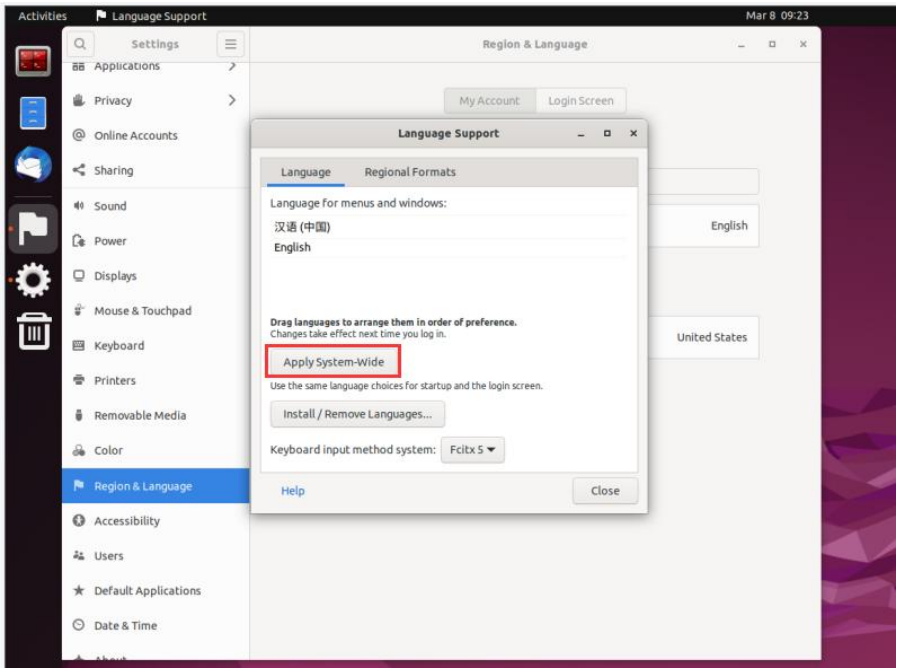
2) 然后找到 **Region & Language** 选项，然后点击 **Manage Installed Languages** 选项



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

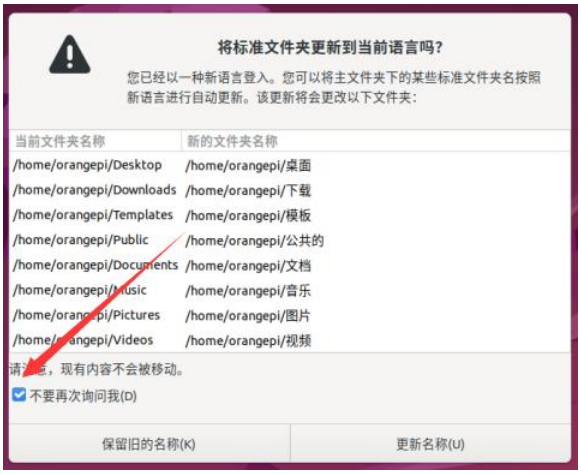


4) 然后选择 **Apply System-Wide** 将中文设置应用到整个系统

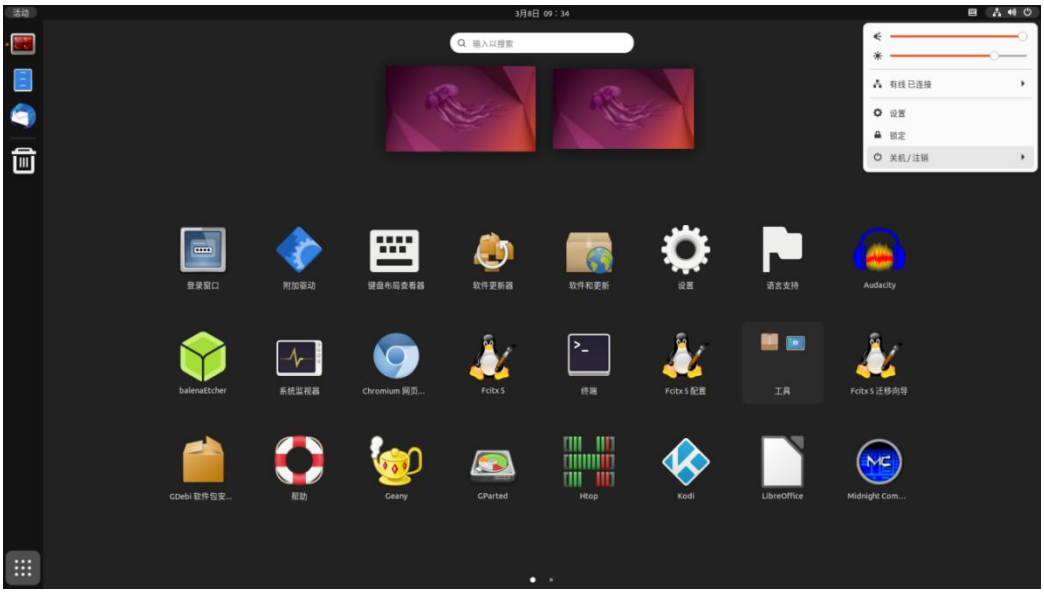


5) **然后重启 Linux 系统使配置生效**

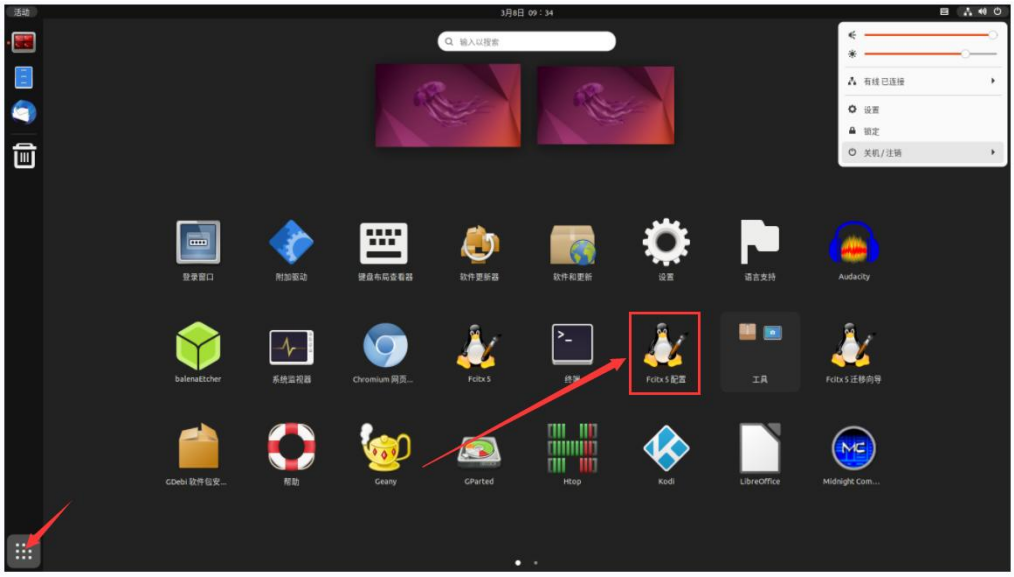
6) 重新进入系统后，在下面的界面请选择**不要再次询问我**，然后请根据自己的喜好决定标准文件夹是否也要更新为中文



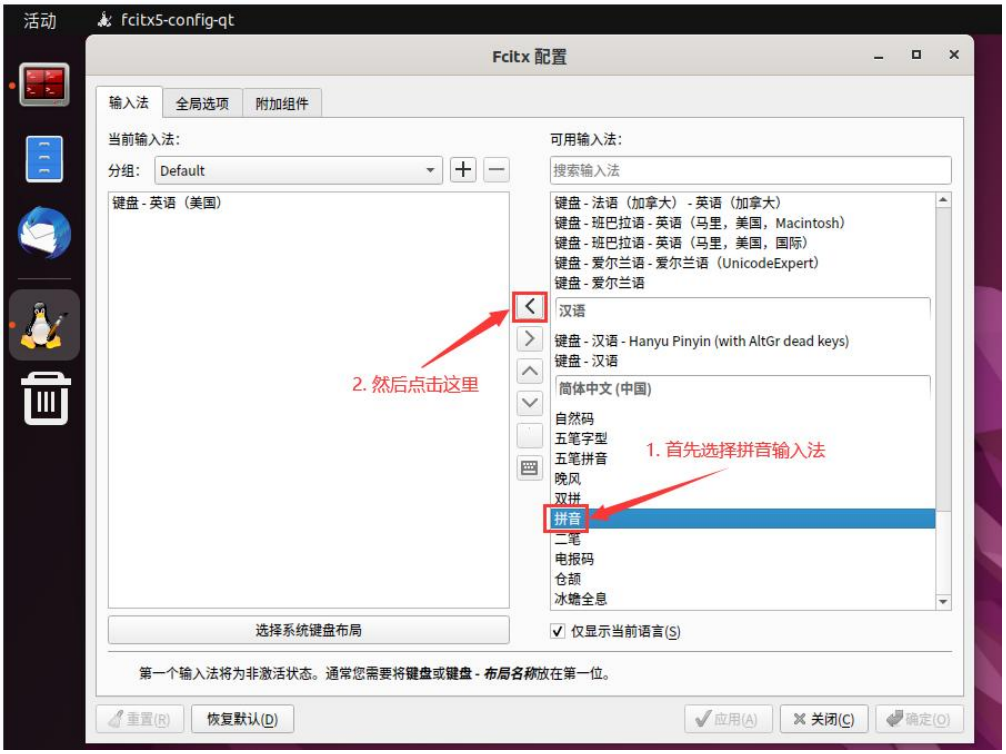
7) 然后可以看到桌面都显示为中文了



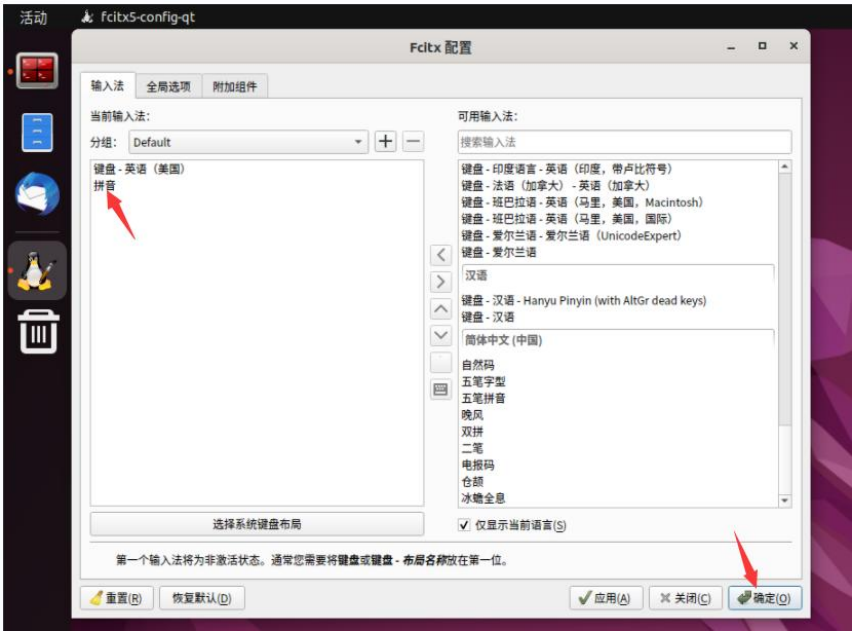
8) 然后打开 Fcitx5 配置程序



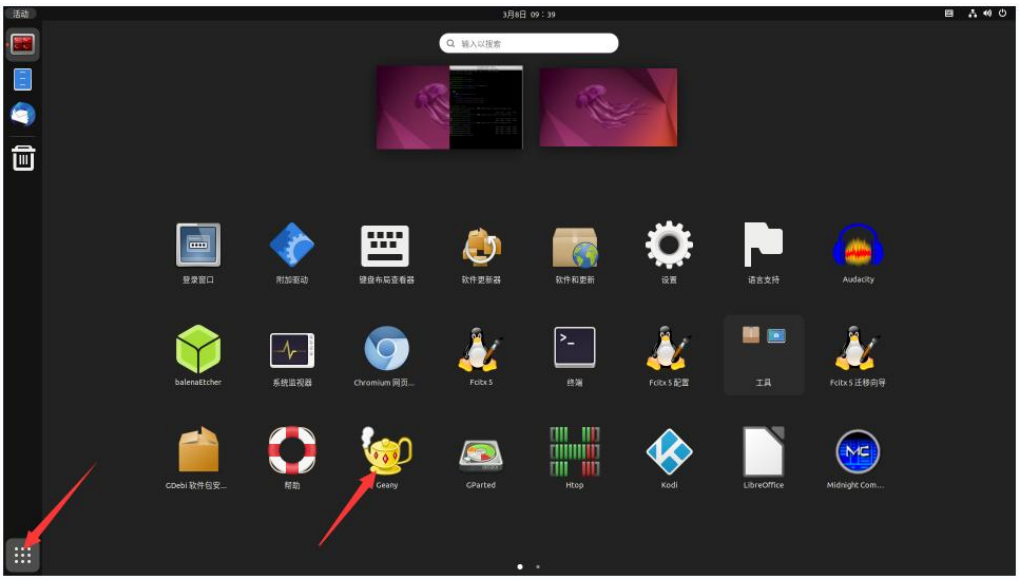
9) 然后选择使用拼音输入法



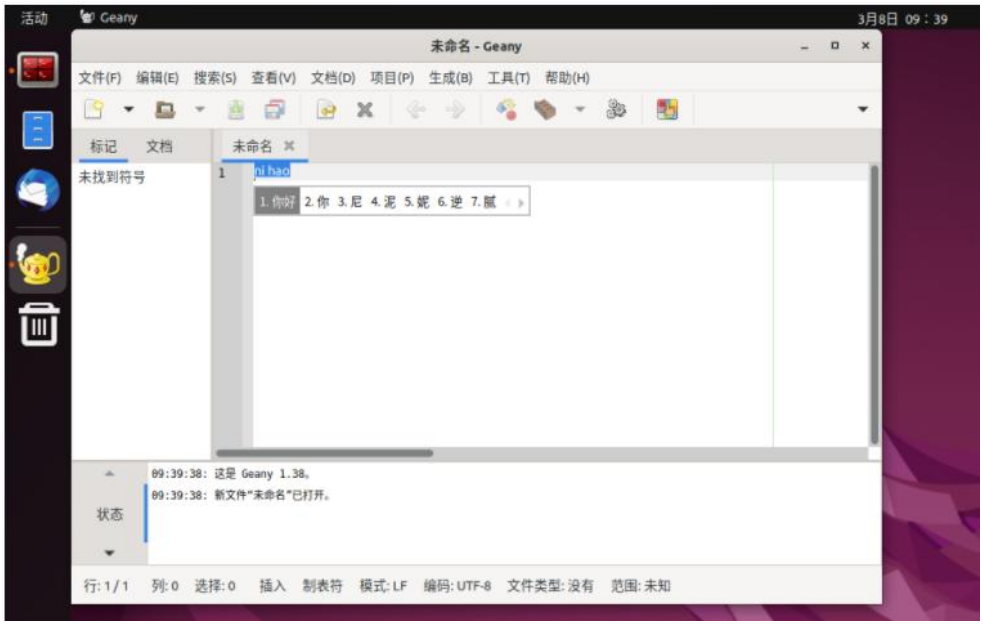
10) 选择后的界面如下所示，再点击确定即可



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



12) 打开 **Geany** 后，默认还是英文输入法，我们可以通过 **Ctrl+Space** 快捷键来切换成中文输入法，然后就能输入中文了


活动
Geany
3月8日 09：39
未命名 - Geany
文件(F) 编辑(E) 搜索(S) 查看(V) 文档(D) 项目(P) 生成(B) 工具(T) 帮助(H)
标记 文档
未找到符号
未命名
1
ni hao
1. 你好 2. 你 3. 尼 4. 泥 5. 妮 6. 逆 7. 腻
09:39:38: 这是 Geany 1.38。
09:39:38: 新文件"未命名"已打开。
状态
行: 1 / 1 列: 0 选择: 0 插入 制表符 模式: LF 编码: UTF-8 文件类型: 没有 范围: 未知

5. Orange Pi OS Arch 系统使用说明

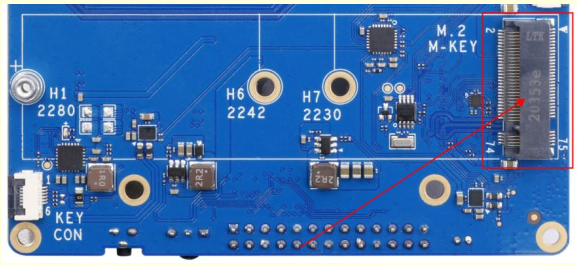
5.1. Orange Pi OS Arch 系统适配情况

功能	OPi OS Arch Gnome Wayland
HDMI 显示	OK
HDMI 音频	OK
USB 2.0	OK
USB 3.0	OK
WIFI	OK
蓝牙	OK
调试串口	OK
FAN 风扇	OK
eMMC 启动	OK
GPIO（26pin）	OK
UART（26pin）	OK
SPI（26pin）	OK
I2C（26pin）	OK
PWM（26pin）	OK
Camera1	OK
Camera2	OK
Camera3	OK
LCD 显示	OK
LCD 触摸	OK
板载 MIC	OK
耳机播放	OK
耳机录音	OK
喇叭 x 2	OK
LED 灯	OK
Type-C 转 USB 3.0	OK
Type-C 接口 DP 显示	OK
Type-C 接口 DP 音频	OK
TF 卡启动	OK

NVMe SSD 识别	OK
SATA SSD 识别	OK
电池	OK
红外	OK
GPU	OK
NPU	NO
VPU	OK
开关机按键	OK
看门狗测试	OK

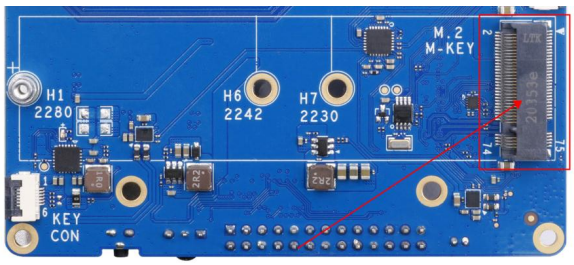
5.2. OPi OS Arch 系统使用 SATA SSD 的方法

下图所示的 m.2 接口既可以使用 nvme ssd，也可以使用 sata ssd。由于 pcie2.0 控制器和 sata 控制器是二选一的，所以同一时间只能打开其中的一个配置。Orange Pi 发布的 OPi OS Arch 镜像默认打开的是 pcie 的配置，所以默认只能识别 nvme ssd。如果想使用 sata ssd，需要打开相应的配置才行。



1) 首先需要准备一个 SATA SSD 固态硬盘。

2) 然后把 SSD 插入开发板的 M.2 接口，并固定好。



3) sata ssd 的用法目前主要是当扩展存储设备。

4) 然后在**/boot/extlinux/extlinux.conf** 中加上下面的配置。

```
[orangepi@orangepi ~]$ sudo vim /boot/extlinux/extlinux.conf
LABEL Orange Pi
LINUX /Image
FDT /dtbs/rockchip/rk3588s-orangepi-cm5-tablet.dtb
FDTOVERLAYS /dtbs/rockchip/overlay/rk3588-ssd-sata0.dtbo    #需要添加的配置
```

5) **然后重启 OPi OS Arch 系统**

6) 如果一切正常，系统重启后使用 **sudo fdisk -l** 命令就可以看到 sata ssd 的信息

```
[orangepi@orangepi ~]$ sudo fdisk -l
.....
Disk /dev/sda: 238.47 GiB, 256060514304 bytes, 500118192 sectors
Disk model: Fanxiang S201 25
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 43FFB292-340D-654C-8C30-6C64AEDAA0F4

Device       Start          End      Sectors      Size Type
/dev/sda1     2048 500117503 500115456 238.5G Linux filesystem
.....
```

5.3. 10.1 寸 MIPI LCD 屏幕的使用方法

5.3.1. 10.1 寸 MIPI 屏幕的组装方法

1) 首先准备需要的配件

a. 10.1 寸 MIPI LCD 显示屏+触摸屏。

b. 屏幕转接板+31pin 转 26pin 排线。

c. 30pin MIPI 排线。

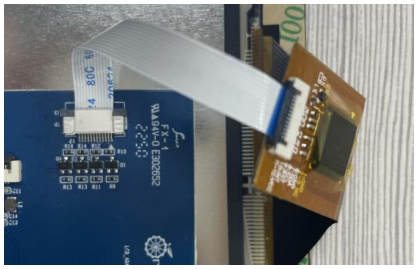
d. 12pin 触摸屏排线。

2) 按照下图将 12pin 触摸屏排线、31pin 转 26pin 排线、30pin MIPI 排线接到屏幕

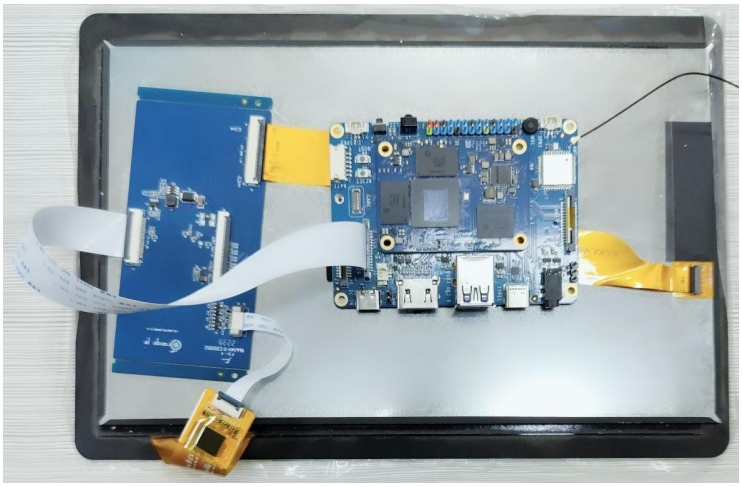
转接板上，注意**触摸屏排线蓝色的绝缘面朝下**，其它两根排线绝缘面朝上，如果接错会导致无显示或者不能触摸的问题。

3) 按照下图将连接好排线的转接板置于 MIPI LCD 屏上面，并通过 31pin 转 26pin 排线连接 MIPI LCD 屏与转接板。

4) 然后通过 12pin 触摸屏排线连接触摸屏与转接板，注意绝缘面的朝向。



5) 最后通过 30pin MIPI 排线连接到开发板的 LCD 接口上。



5.3.2. 打开 10.1 寸 MIPI LCD 屏幕配置的方法

1) OPi OS Arch 镜像默认是没有打开 mipi lcd 屏幕的配置的，如果需要使用 mipi lcd 屏幕，需要手动打开才行。

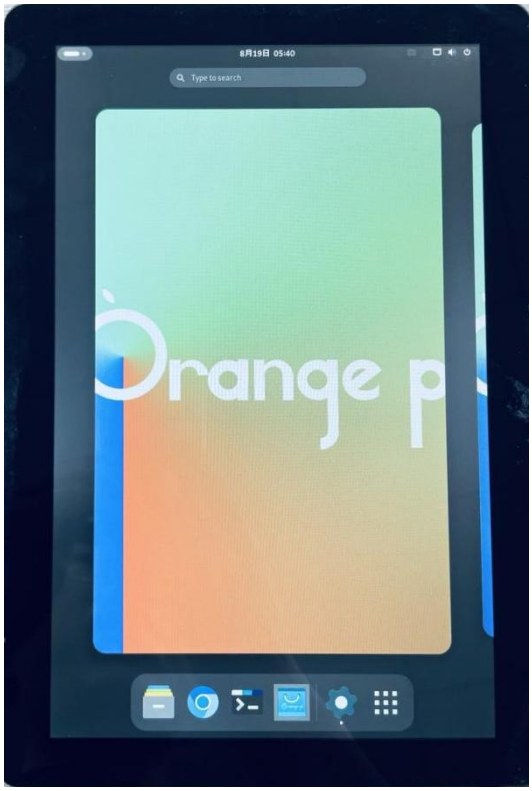
2) 开发板上 mipi lcd 屏幕的接口如下图所示：

3) 打开 mipi lcd 配置的方法如下所示：

```
[orangepi@orangepi ~]$ sudo vim /boot/extlinux/extlinux.conf
LABEL Orange Pi
LINUX /Image
FDT /dtbs/rockchip/rk3588s-orangepi-5-pro.dtb
FDTOVERLAYS /dtbs/rockchip/overlay/rk3588-opicm5-tablet-lcd.dtbo #需要添加的配置
```

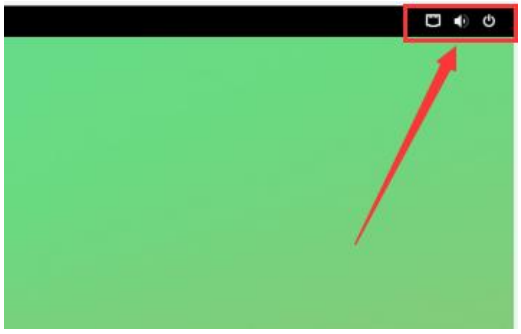
4) **然后重启 OPi OS Arch 系统**。

5) 重启后可以看到 lcd 屏幕的显示如下所示（默认为竖屏）：

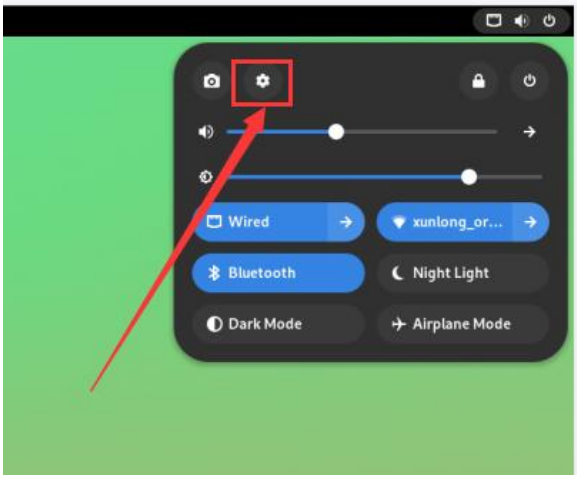


5.3.3. 旋转显示和触摸方向的方法

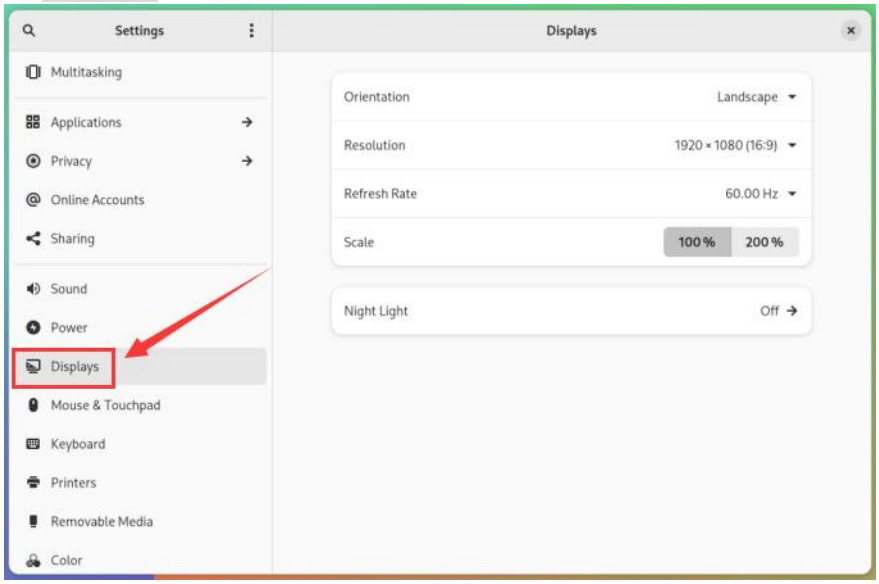
1) 首先点击桌面右上角的这块区域。



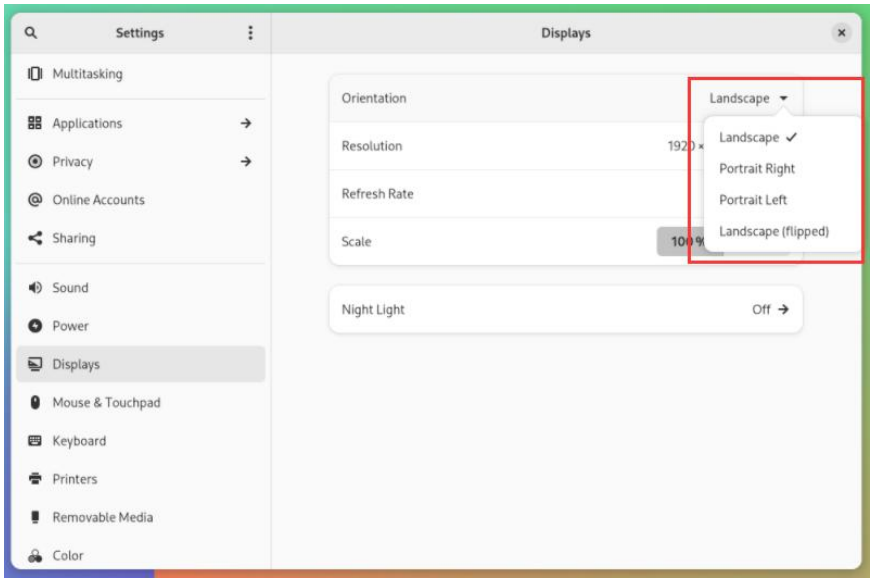
2) 然后打开设置。



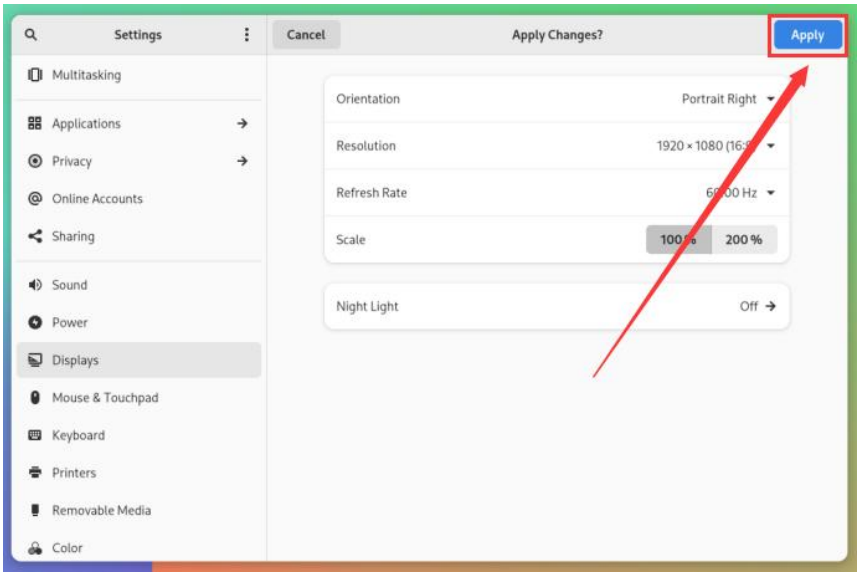
3) 然后选择 **Displays**。



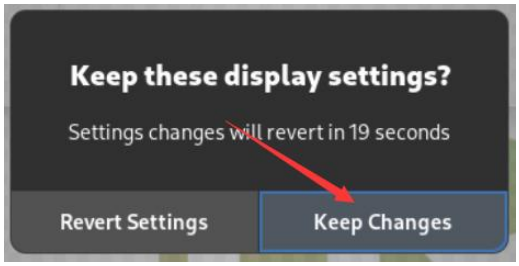
4) 然后在 **Displays** 的 **Orientation** 中选择想要旋转的方向。



5) 然后选择 **Apply**。



6) 然后就能看到屏幕已经旋转好了，此时还需要选择 **Keep Changes** 来最后确定旋转。



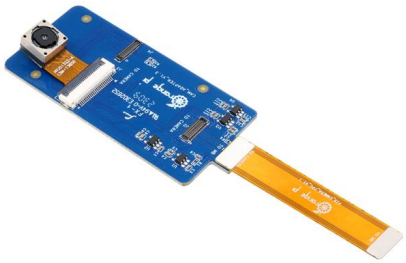
7) LCD 屏幕旋转 90 度后的显示如下所示：

8) **OPi OS Arch 系统 LCD 屏幕的触摸功能会随着显示方向的旋转而旋转，无需其他设置。**

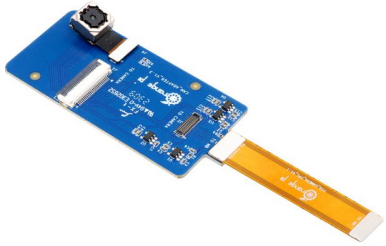
5.4. OV13850 和 OV13855 MIPI 摄像头的测试方法

目前开发板支持两款MIPI摄像头，OV13850 和OV13855，具体的图片如下所示：

a. 1300 万MIPI接口的OV13850 摄像头。



b. 1300 万MIPI接口的OV13855 摄像头。

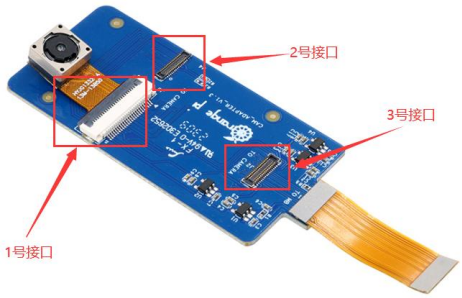


OV13850 和OV13855 摄像头使用的转接板和FPC排线是一样的，只是两款摄像

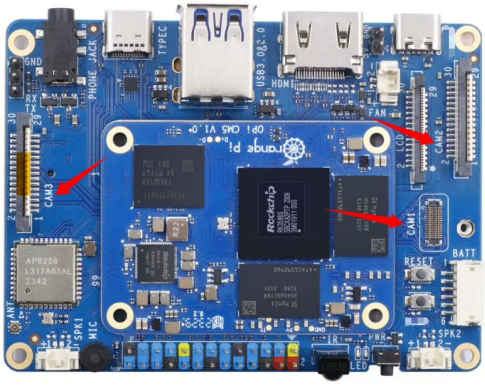
头接在转接板上的位置不一样。FPC排线如下图所示，请注意FPC排线是有方向的，标注**TO MB**那端需要插到开发板的摄像头接口中，标注**TO CAMERA**那端需要插到摄像头转接板上。

摄像头转接板上总共有 3 个摄像头的接口，同一时间只能接一个使用，如下图所示，其中：

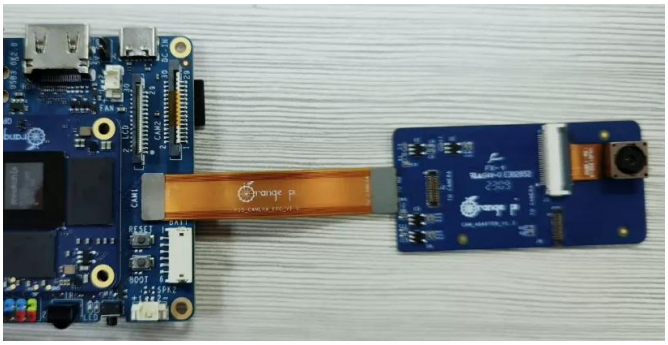
a. **1 号接口接 OV13850 摄像头**。
b. **2 号接口接 OV13855 摄像头**。
c. 3 号接口未使用，忽略即可。



Orange Pi CM5 Base Tablet 开发板上总共有 3 个摄像头接口，只有 CAM1 可以用来接 OV13850 或 OV13855 摄像头，Cam2 和 Cam3 暂时未适配具体的摄像头型号。我们定义 Cam1、Cam2 和 Cam3 的位置如下图所示：



摄像头插在开发板的 Cam1 接口的方法如下所示：



连接好摄像头到开发板上后，我们可以使用下面的方法来测试下摄像头：

a. 首先在**/boot/extlinux/extlinux.conf** 中加上下面的配置。

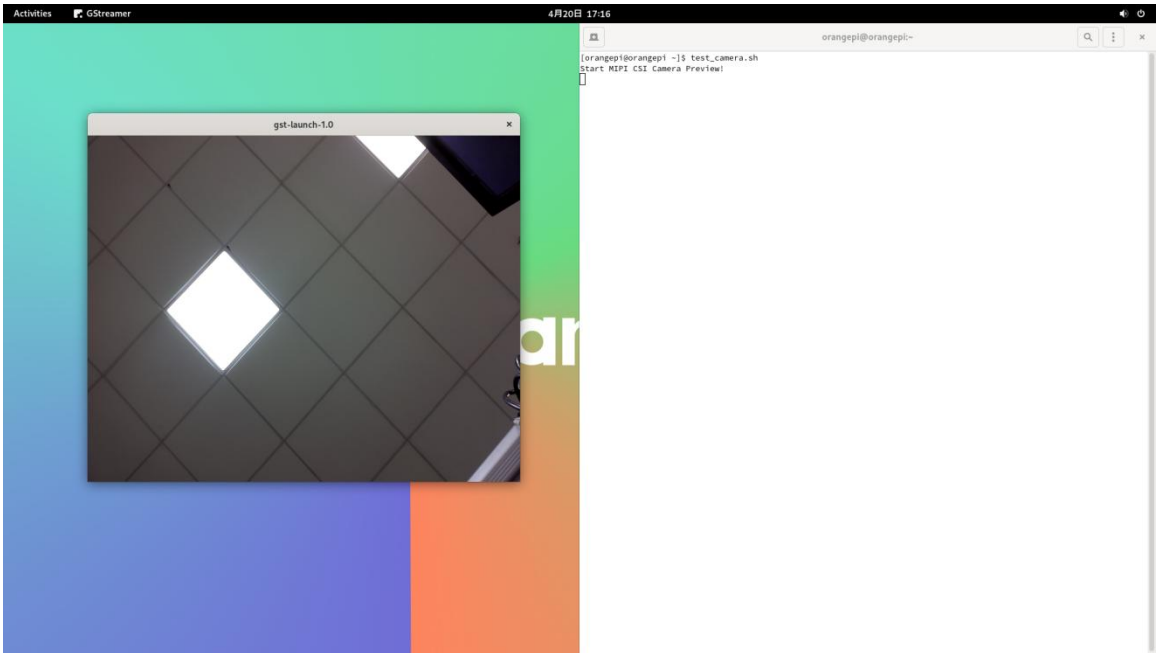
```
[orangepi@orangepi ~]$ sudo vim /boot/extlinux/extlinux.conf
LABEL Orange Pi
LINUX /Image
FDT /dtbs/rockchip/rk3588s-orangepi-cm5-tablet.dtb
FDTOVERLAYS /dtbs/rockchip/overlay/rk3588-opicm5-tablet-cam1.dtbo #需要添加的配置
```

b. **然后重启 OPi OS Arch 系统**。

c. 然后在桌面系统中打开一个终端，再运行下面的脚本。

```
orangepi@orangepi:~$ test_camera.sh
```

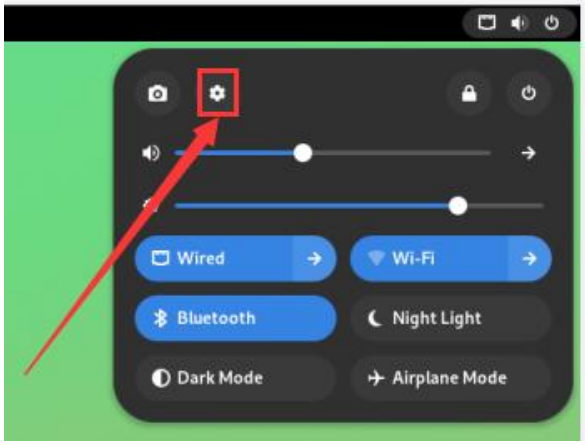
d. 然后就能看到摄像头的预览画面了。



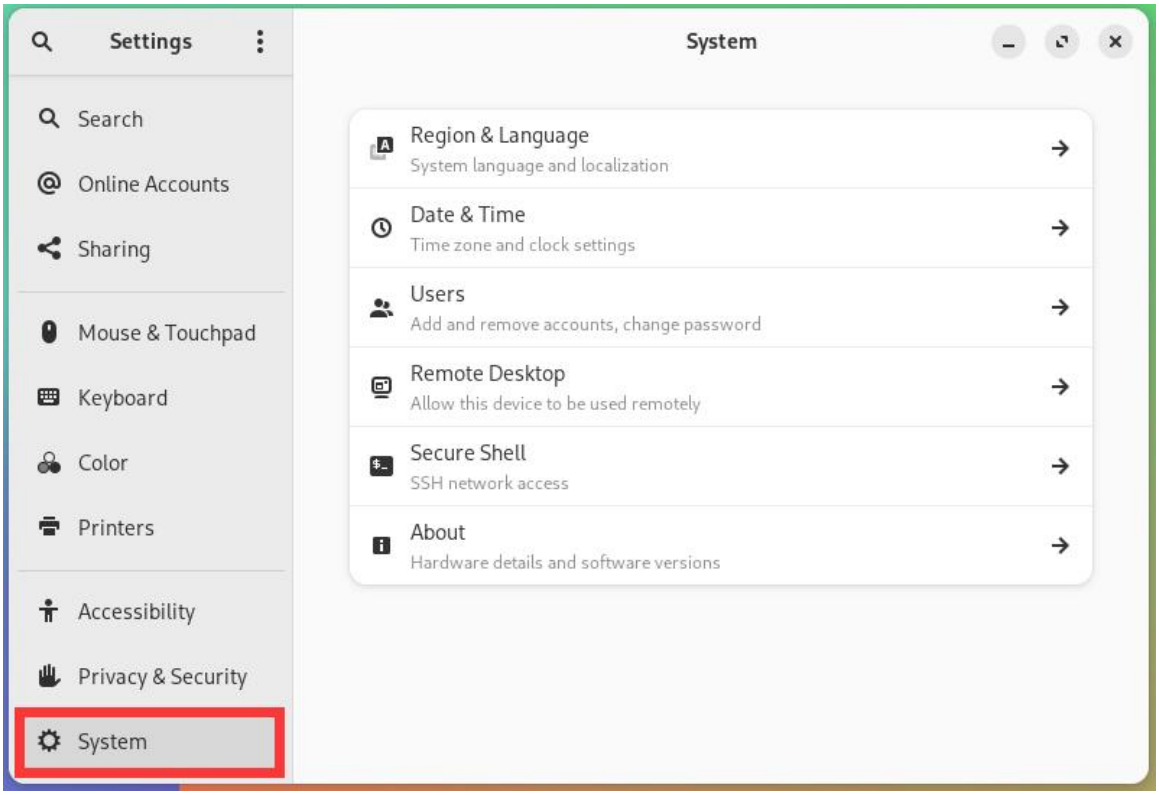
5.5. 设置中文环境以及安装中文输入法的方法

1) 首先点击桌面右上角的这块区域。

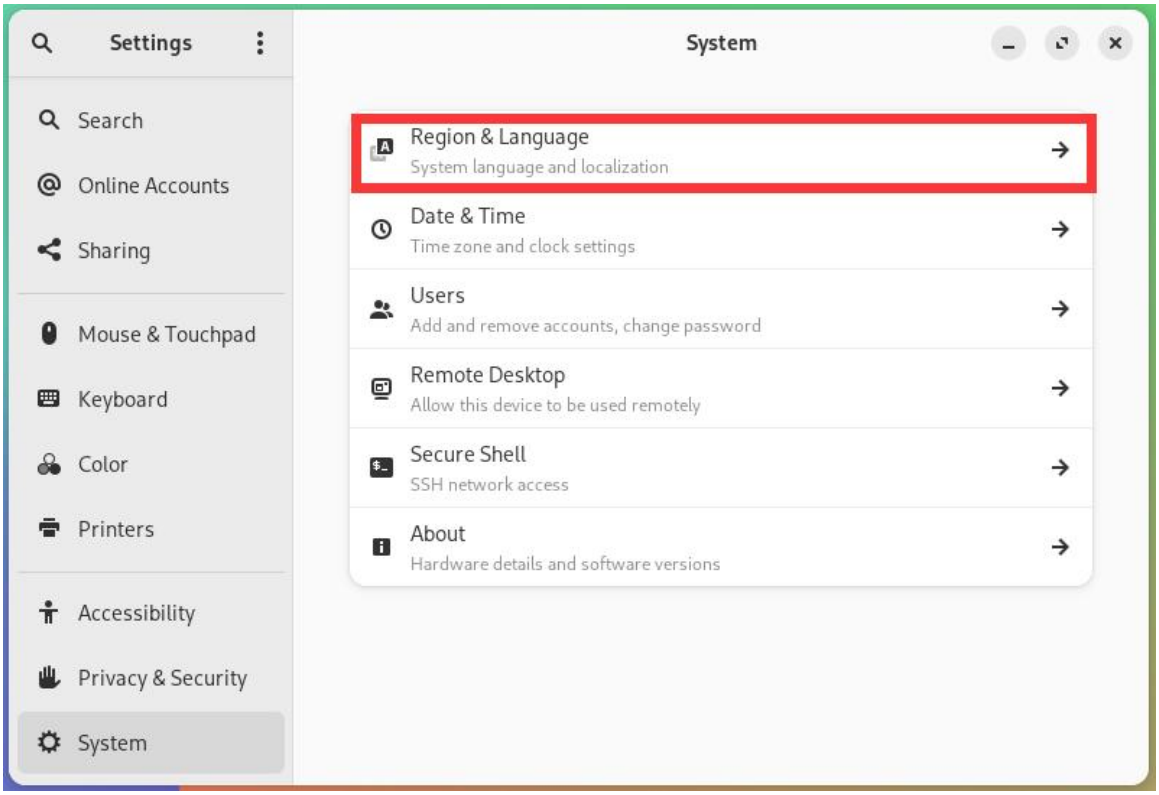
2) 然后打开设置。



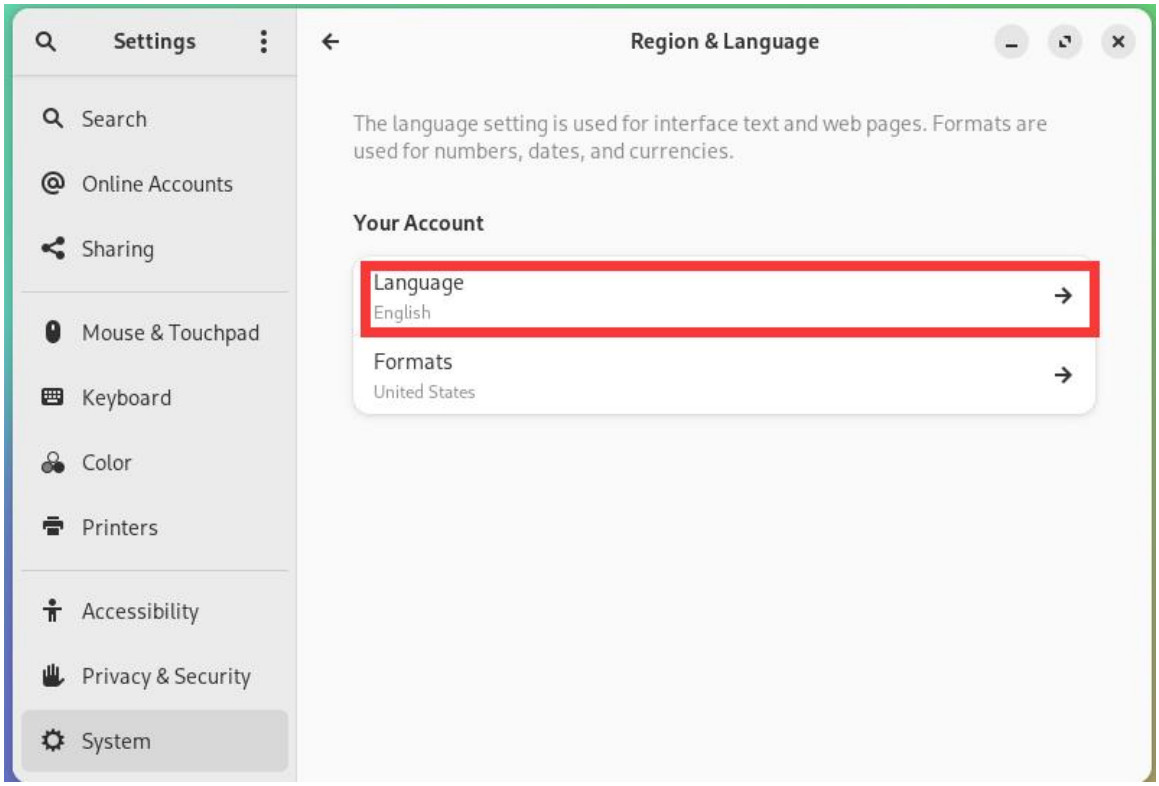
3) 然后找到 **System** 选项。



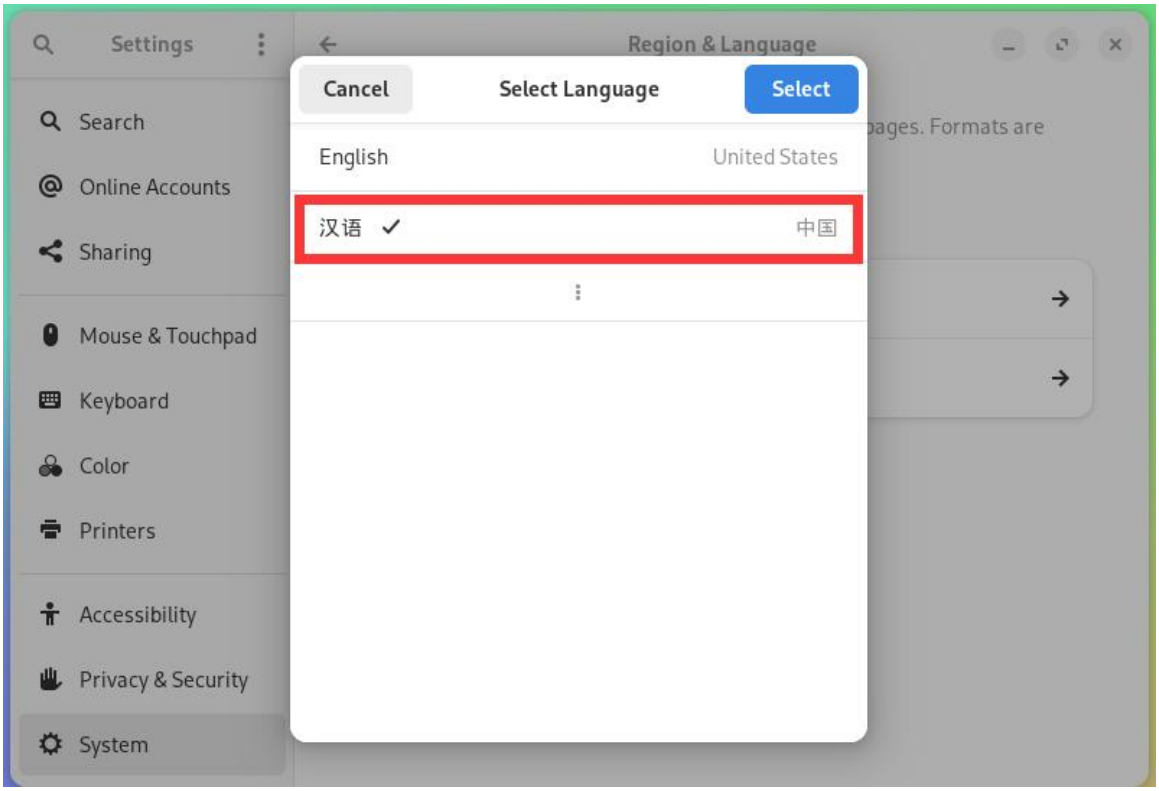
4) 然后找到 **Region & Language** 选项。



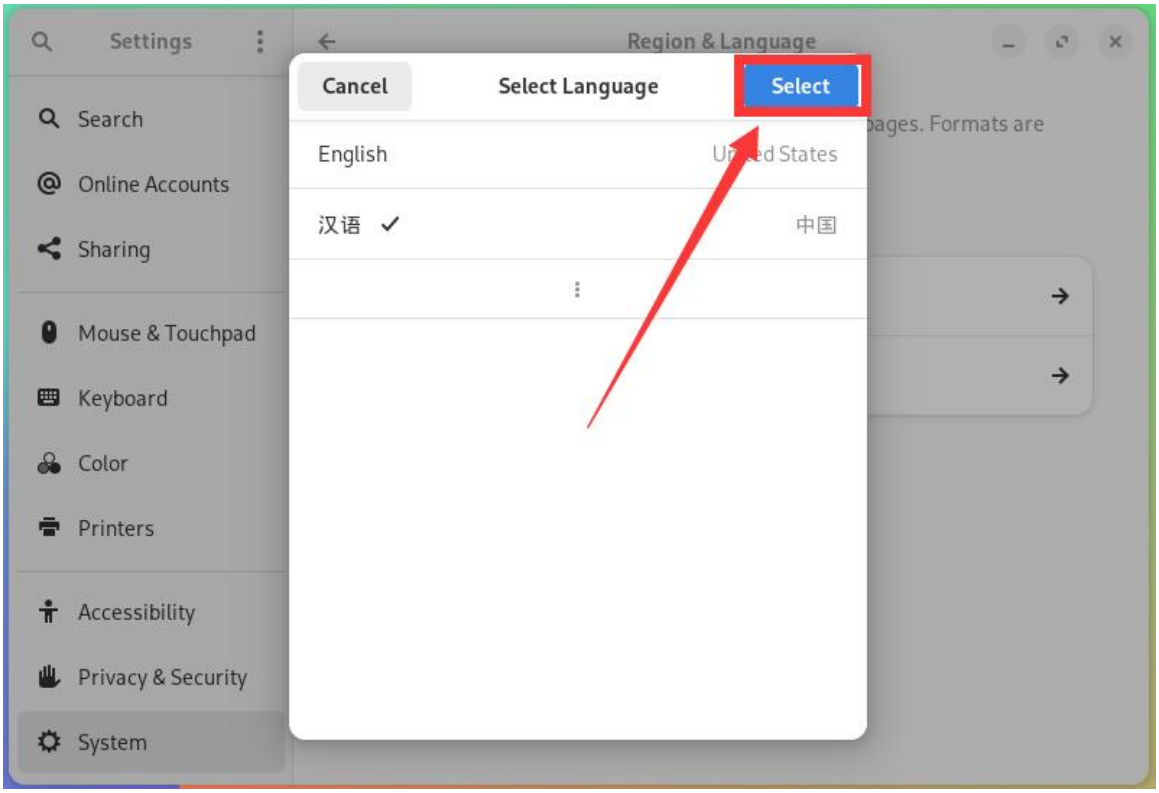
5) 然后选择 **Language**。



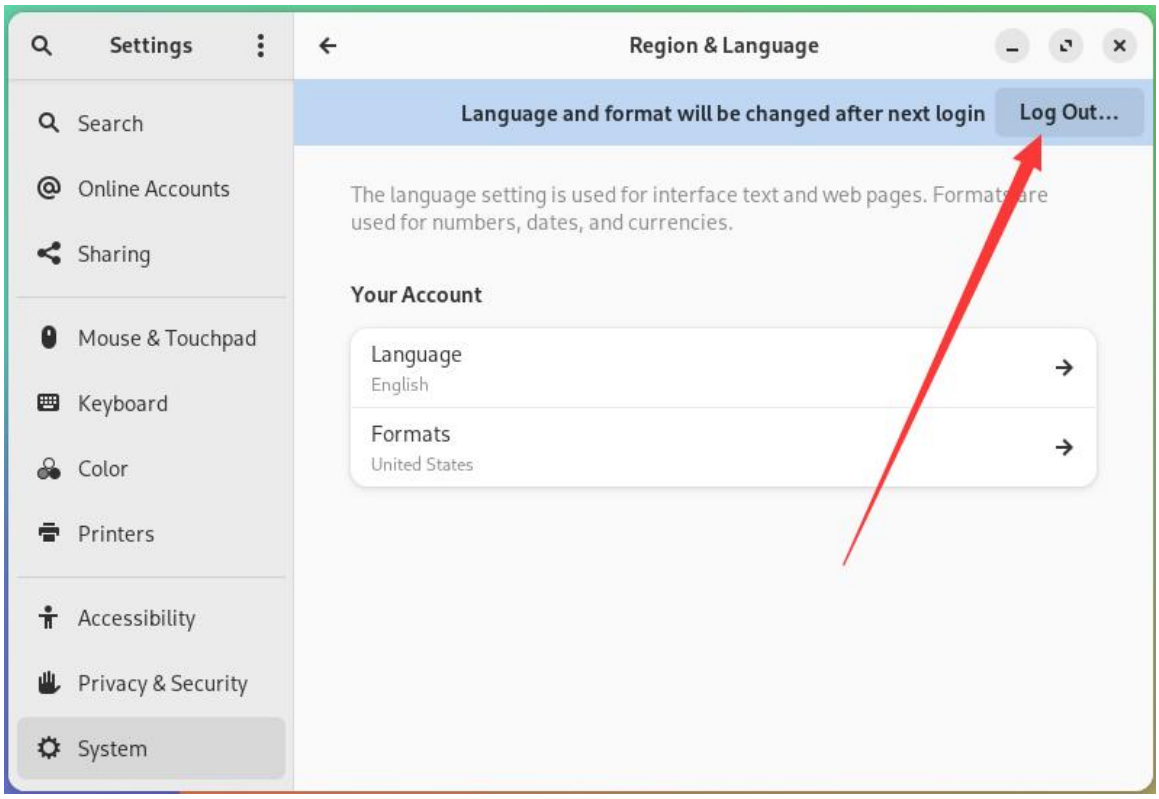
6) 然后选择汉语。



7) 然后点击 **Select**。



8) 然后点击 **Log Out...**登出系统，再重新登入系统。

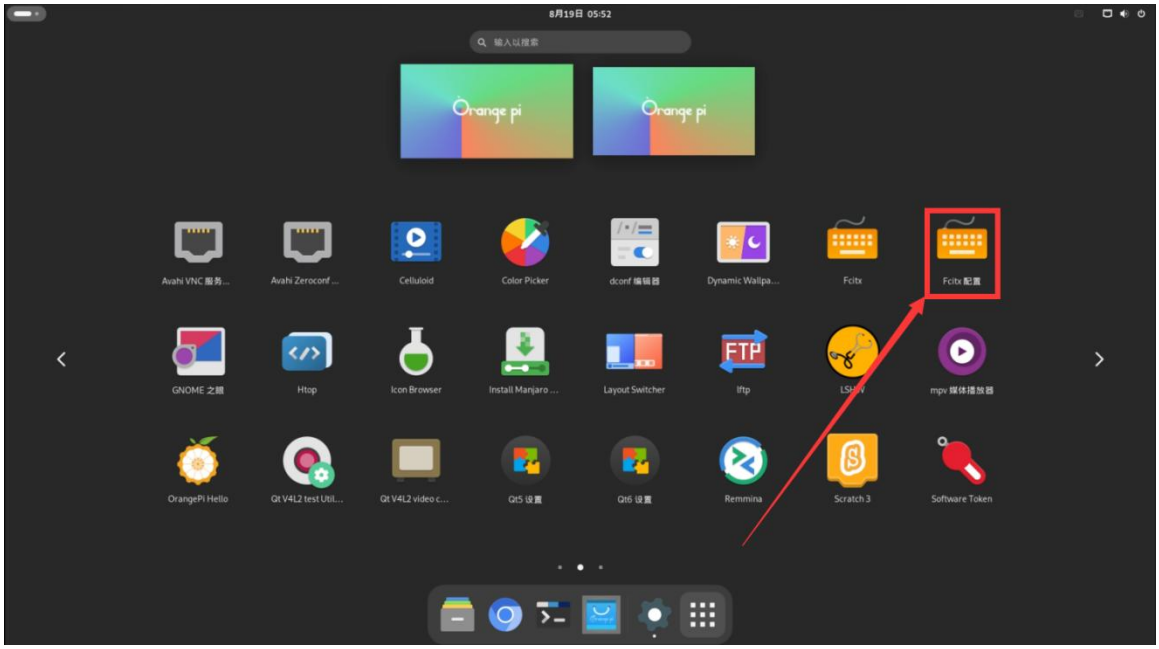


9) 然后可以看到桌面都显示为中文了。

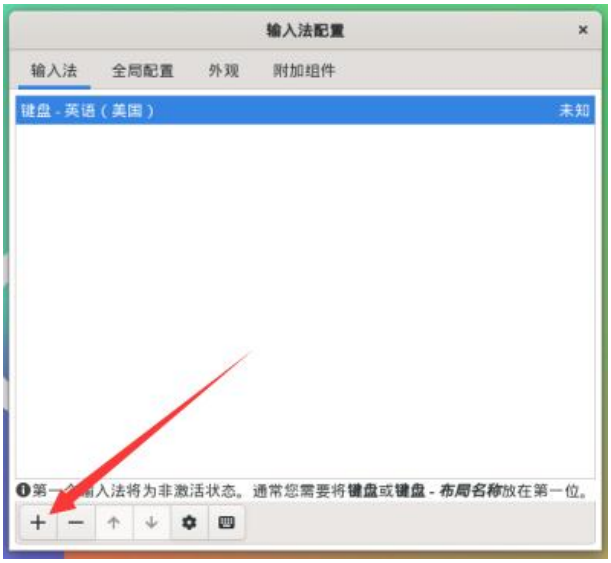
10) 然后安装下 fcitx-im 和 fcitx-configtool。

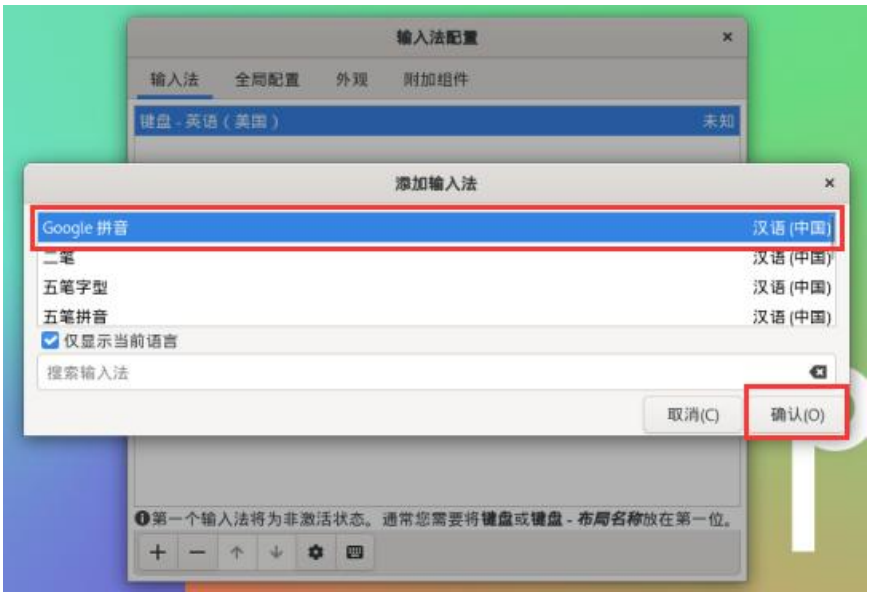
```
[orangepi@orangepi ~]$ sudo pacman -S fcitx-im fcitx-configtool
:: 在组 fcitx-im 中有 3 成员：
:: 软件仓库 community
    1) fcitx    2) fcitx-qt5    3) fcitx-qt6

输入某个选择 ( 默认=全部选定 ): 1
```

11) 然后打开 Fcitx 配置程序。

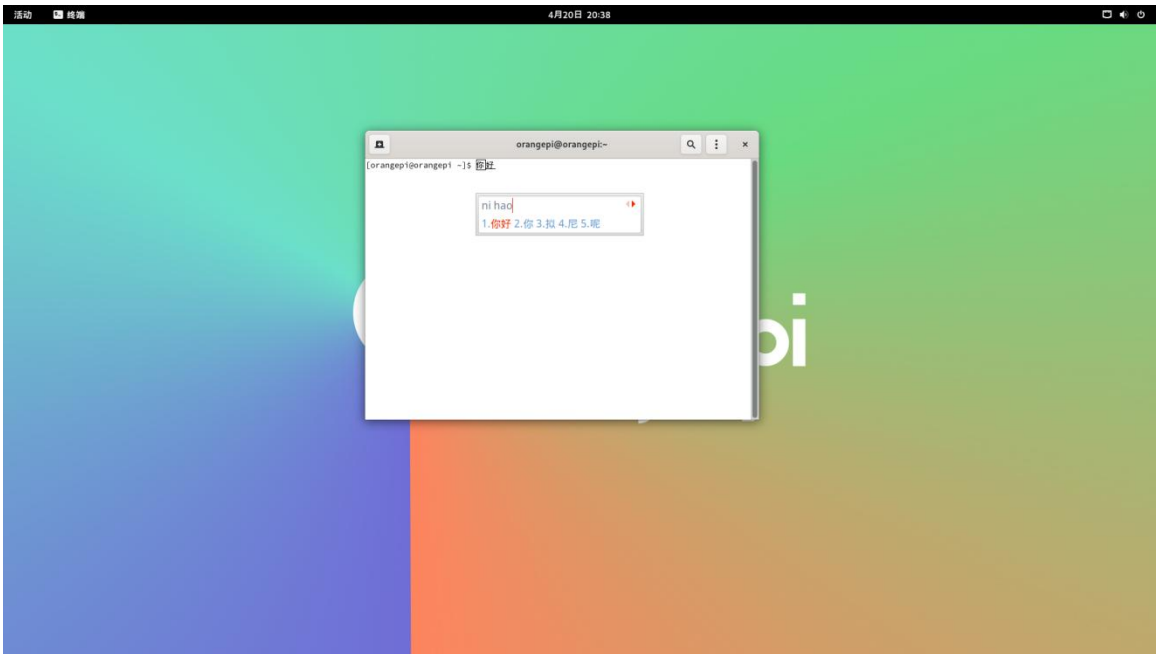


12) 然后添加 **Google 拼音**输入法。





13) 然后我们可以打开一个终端测试下中文输入法，打开终端后，如果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。



5.6. 安装 wiringOP 的方法

注意，Orange Pi 发布的 OPi OS Arch 镜像中已经预装了 wiringOP，除非

wiringOP 的代码有更新，否则无需重新下载编译安装，直接使用即可。

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

```
[root@orangepi wiringOP]# gpio readall
 +------+-----+----------+--------+---+CM5 Tablet+---+--------+----------+-----+------+
 | GPIO | wPi |   Name   |  Mode  | V | Physical | V |  Mode  | Name     | wPi | GPIO |
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 |      |     |     3.3V |        |   |  1 || 2  |   |        | 5V       |     |      |
 |   63 |   0 |    SDA.5 |     IN | 1 |  3 || 4  |   |        | 5V       |     |      |
 |   62 |   1 |    SCL.5 |     IN | 1 |  5 || 6  |   |        | GND      |     |      |
 |   35 |   2 |     PWM1 |     IN | 0 |  7 || 8  | 0 | IN     | TXD.6    | 3   | 33   |
 |      |     |      GND |        |   |  9 || 10 | 0 | IN     | RXD.6    | 4   | 32   |
 |   47 |   5 |    RXD.4 |     IN | 1 | 11 || 12 | 0 | IN     | GPIO1_A2 | 6   | 34   |
 |   46 |   7 |    TXD.4 |     IN | 1 | 13 || 14 |   |        | GND      |     |      |
 |   40 |   8 | GPIO1_B0 |     IN | 1 | 15 || 16 | 0 | IN     | GPIO1_A4 | 9   | 36   |
 |      |     |     3.3V |        |   | 17 || 18 | 0 | IN     | GPIO1_A6 | 10  | 38   |
 |   42 |  11 | SPI0_TXD |     IN | 0 | 19 || 20 |   |        | GND      |     |      |
 |   41 |  12 | SPI0_RXD |     IN | 0 | 21 || 22 | 1 | IN     | PWM3     | 13  | 39   |
 |   43 |  14 | SPI0_CLK |     IN | 0 | 23 || 24 | 1 | IN     | SPI0_CS0 | 15  | 44   |
 |      |     |      GND |        |   | 25 || 26 | 1 | IN     | SPI0_CS1 | 16  | 45   |
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 | GPIO | wPi |   Name   |  Mode  | V | Physical | V |  Mode  | Name     | wPi | GPIO |
 +------+-----+----------+--------+---+CM5 Tablet+---+--------+----------+-----+------+
```

1) 下载 wiringOP 的代码。

```
[orangepi@orangepi ~]$ sudo pacman -Syy git
[orangepi@orangepi ~]$ git clone https://github.com/orangepi-xunlong/wiringOP.git -b next
```

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

如果从 GitHub 下载代码有问题，可以去 Orange Pi CM5 Base Tablet 资料下载页面的官方工具中下载 wiringOP.tar.gz 的源码压缩包。

2) 编译安装 wiringOP。

```
[orangepi@orangepi ~]$ sudo pacman -Syy make gcc
[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：

```
[root@orangepi wiringOP]# gpio readall
 +------+-----+----------+--------+---+---+CM5 Tablet+---+--------+----------+-----+------+
 | GPIO | wPi |   Name   |  Mode  | V | Physical | V |  Mode  | Name     | wPi | GPIO |
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 |      |     |     3.3V |        |   |  1 || 2  |   |        | 5V       |     |      |
 |   63 |   0 |    SDA.5 |     IN | 1 |  3 || 4  |   |        | 5V       |     |      |
 |   62 |   1 |    SCL.5 |     IN | 1 |  5 || 6  |   |        | GND      |     |      |
 |   35 |   2 |     PWM1 |     IN | 0 |  7 || 8  | 0 | IN     | TXD.6    | 3   | 33   |
 |      |     |      GND |        |   |  9 || 10 | 0 | IN     | RXD.6    | 4   | 32   |
 |   47 |   5 |    RXD.4 |     IN | 1 | 11 || 12 | 0 | IN     | GPIO1_A2 | 6   | 34   |
 |   46 |   7 |    TXD.4 |     IN | 1 | 13 || 14 |   |        | GND      |     |      |
 |   40 |   8 | GPIO1_B0 |     IN | 1 | 15 || 16 | 0 | IN     | GPIO1_A4 | 9   | 36   |
 |      |     |     3.3V |        |   | 17 || 18 | 0 | IN     | GPIO1_A6 | 10  | 38   |
 |   42 |  11 | SPI0_TXD |     IN | 0 | 19 || 20 |   |        | GND      |     |      |
 |   41 |  12 | SPI0_RXD |     IN | 0 | 21 || 22 | 1 | IN     | PWM3     | 13  | 39   |
 |   43 |  14 | SPI0_CLK |     IN | 0 | 23 || 24 | 1 | IN     | SPI0_CS0 | 15  | 44   |
 |      |     |      GND |        |   | 25 || 26 | 1 | IN     | SPI0_CS1 | 16  | 45   |
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 | GPIO | wPi |   Name   |  Mode  | V | Physical | V |  Mode  | Name     | wPi | GPIO |
 +------+-----+----------+--------+---+---+CM5 Tablet+---+--------+----------+-----+------+
```

5.7. 26pin 接口 GPIO、I2C、UART、SPI 和 PWM 测试

注意，如果需要设置 fdt overlays 同时打开多个配置，请像下面红色字体配置那样使用空格隔开写在一行即可。

[orangepi@orangepi ~]$ **sudo vim /boot/extlinux/extlinux.conf**

LABEL Orange Pi

LINUX /Image

FDT /dtbs/rockchip/rk3588s-orangepi-cm5-tablet.dtb

FDTOVERLAYS /dtbs/rockchip/overlay/rk3588-i2c5-m3.dtbo /dtbs/rockchip/overlay/rk3588-uart4-m0.dtbo

5.7.1. 26pin GPIO 口测试

1) 开发板 26pin 中总共有 **17** 个 GPIO 口可以使用，下面以 7 号引脚——对应 GPIO 为 GPIO1_A3——对应 wPi 序号为 2——为例演示如何设置 GPIO 口的高低电平。

```
[root@orangepi wiringOP]# gpio readall
 +------+-----+----------+--------+---+---+CM5 Tablet+---+--------+----------+-----+------+
 | GPIO | wPi |   Name   |  Mode  | V | Physical | V |  Mode  | Name     | wPi | GPIO |
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 |      |     |     3.3V |        |   |  1 || 2  |   |        | 5V       |     |      |
 |   63 |   0 |    SDA.5 |     IN | 1 |  3 || 4  |   |        | 5V       |     |      |
 |   62 |   1 |    SCL.5 |     IN | 1 |  5 || 6  |   |        | GND      |     |      |
 |   35 |   2 |     PWM1 |     IN | 0 |  7 || 8  | 0 | IN     | TXD.6    | 3   | 33   |
 |      |     |      GND |        |   |  9 || 10 | 0 | IN     | RXD.6    | 4   | 32   |
```

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。

```
[orangepi@orangepi ~]$ 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。

```
[orangepi@orangepi ~]$ gpio write 2 0
```

使用 gpio read 命令可以看到 7 号引脚的值(V)变为了 0。

```
[orangepi@orangepi ~]$ gpio read 2
0
```

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

```
[orangepi@orangepi ~]$ gpio write 2 1
```

使用 gpio readall 可以看到 7 号引脚的值(V)变为了 1。

```
[orangepi@orangepi ~]$ gpio read 2
1
```

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

5.7.2. 26pin GPIO 口上下拉电阻的设置方法

1) 下面以 7 号引脚——对应 GPIO 为 GPIO1_A3——对应 wPi 序号为 2——为例演示如何设置 GPIO 口的上下拉电阻。

```
[root@orangepi wiringOP]# gpio readall
 +------+-----+----------+--------+---+CM5 Tablet+---+--------+----------+-----+------+
 | GPIO | wPi |   Name   |  Mode  | V | Physical | V |  Mode  | Name     | wPi | GPIO |
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 |      |     |     3.3V |        |   |  1 || 2  |   |        | 5V       |     |      |
 |   63 |   0 |    SDA.5 |     IN | 1 |  3 || 4  |   |        | 5V       |     |      |
 |   62 |   1 |    SCL.5 |     IN | 1 |  5 || 6  |   |        | GND      |     |      |
 |   35 |   2 |     PWM1 |     IN | 0 |  7 || 8  | 0 | IN     | TXD.6    | 3   | 33   |
 |      |     |      GND |        |   |  9 || 10 | 0 | IN     | RXD.6    | 4   | 32   |
```

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

```
[orangepi@orangepi ~]$ gpio mode 2 in
```

3) 设置为输入模式后，执行下面的命令可以设置 GPIO 口为上拉模式

```
[orangepi@orangepi ~]$ gpio mode 2 up
```

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

```
[orangepi@orangepi ~]$ gpio read 2
1
```

5) 然后执行下面的命令可以设置 GPIO 口为下拉模式

```
[orangepi@orangepi ~]$ gpio mode 2 down
```

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

```
[orangepi@orangepi ~]$ gpio read 2
0
```

5.7.3. 26pin I2C 测试

1) 由下表可知，Orange Pi CM5 Base Tablet 可用的 i2c 为 i2c4 和 i2c5 共两组 i2c 总线。

复用功能	复用功能	复用功能	GPIO	GPIO序号	引脚序号	引脚序号	GPIO序号	GPIO	复用功能	复用功能
			3.3V		1	2		5V		
PWM13_M2(febf0010)	UART1_RX_M1	I2C5_SDA_M3	GPIO1_B7	47	3	4		5V		
	UART1_TX_M1	I2C5_SCL_M3	GPIO1_B6	46	5	6		GND		
	PWM1_M2(fd8b0010)	I2C4_SCL_M3	GPIO1_A3	35	7	8	56	GPIO1_D0	UART6_TX_M2	SPI1_MISO_M2
			GND		9	10	57	GPIO1_D1	UART6_RX_M2	SPI1_MOSI_M2
PWM0_M1(fd8b0000)	UART4_TX_M0	SPI1_CLK_M2	GPIO1_D2	58	11	12	34	GPIO1_A2	I2C4_SDA_M3	
	UART4_RX_M0	SPI1_CS0_M2	GPIO1_D3	59	13	14		GND		
			GPIO1_B0	40	15	16	36	GPIO1_A4		
			3.3V		17	18	38	GPIO1_A6		
		SPI0_MOSI_M2	GPIO1_B2	42	19	20		GND		
		SPI0_MISO_M2	GPIO1_B1	41	21	22	39	GPIO1_A7	PWM3_M3(fd8b0030)	
		SPI0_CLK_M2	GPIO1_B3	43	23	24	44	GPIO1_B4	SPI0_CS0_M2	UART7_RX_M2
			GND		25	26	45	GPIO1_B5	SPI0_CS1_M2	UART7_TX_M2

在 OPi OS Arch 系统中，26pin 中的 i2c 默认都是关闭的，需要手动打开才能使用。

在/boot/extlinux/extlinux.conf 中加入下面红色字体部分的配置，然后重启 OPi OS Arch 系统就可以同时打开 i2c4 和 i2c5，如果只需要打开一个，那么就填写一个即可。

```
[orangepi@orangepi ~]$ sudo vim /boot/extlinux/extlinux.conf
LABEL Orange Pi
LINUX /Image
FDT /dtbs/rockchip/rk3588s-orangepi-cm5-tablet.dtb
FDTOVERLAYS   /dtbs/rockchip/overlay/rk3588-i2c4-m3.dtbo
/dtbs/rockchip/overlay/rk3588-i2c5-m3.dtbo
```

上面红色字体配置需要写在一行，不同的配置之间需要用空格隔开。

2) 启动 OPi OS Arch 系统后，先确认下/dev 下存在 i2c 的设备节点。

```
[orangepi@orangepi ~]$ ls /dev/i2c-*
```

3) 然后在 26pin 接头的 i2c 引脚上接一个 i2c 设备。

I2C 总线	SDA 对应 26pin	SCL 对应 26pin
I2C4_M3	12 号引脚	7 号引脚
I2C5_M3	3 号引脚	5 号引脚

4) 然后使用 **i2cdetect -y** 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。

```
[orangepi@orangepi ~]$ sudo pacman -Syy i2c-tools
[orangepi@orangepi ~]$ sudo i2cdetect -y 4          #i2c4 的命令
[orangepi@orangepi ~]$ sudo i2cdetect -y 5          #i2c5 的命令
```

5.7.4. 26pin UART 测试

1) 由下表可知，Orange Pi CM5 Base Tablet 可用的 uart 为 uart1、uart4、uart6 和 uart7 共四组 uart 总线。

复用功能	复用功能	复用功能	GPIO	GPIO序号	引脚序号	引脚序号	GPIO序号	GPIO	复用功能	复用功能
			3.3V		1	2		5V		
PWM13_M2(febf0010)	UART1_RX_M1	I2C5_SDA_M3	GPIO1_B7	47	3	4		5V		
	UART1_TX_M1	I2C5_SCL_M3	GPIO1_B6	46	5	6		GND		
	PWM1_M2(fd8b0010)	I2C4_SCL_M3	GPIO1_A3	35	7	8	56	GPIO1_D0	UART6_TX_M2	SPI1_MISO_M2
			GND		9	10	57	GPIO1_D1	UART6_RX_M2	SPI1_MOSI_M2
PWM0_M1(fd8b0000)	UART4_TX_M0	SPI1_CLK_M2	GPIO1_D2	58	11	12	34	GPIO1_A2	I2C4_SDA_M3	
	UART4_RX_M0	SPI1_CS0_M2	GPIO1_D3	59	13	14		GND		
			GPIO1_B0	40	15	16	36	GPIO1_A4		
			3.3V		17	18	38	GPIO1_A6		
		SPI0_MOSI_M2	GPIO1_B2	42	19	20		GND		
		SPI0_MISO_M2	GPIO1_B1	41	21	22	39	GPIO1_A7	PWM3_M3(fd8b0030)	
		SPI0_CLK_M2	GPIO1_B3	43	23	24	44	GPIO1_B4	SPI0_CS0_M2	UART7_RX_M2
			GND		25	26	45	GPIO1_B5	SPI0_CS1_M2	UART7_TX_M2

在 OPi OS Arch 系统中，26pin 中的 uart 默认都是关闭的，需要手动打开才能使用。

在/boot/extlinux/extlinux.conf 中加入下面红色字体部分的配置，然后重启 OPi OS Arch 系统就可以同时打开 uart1、uart4、uart6 和 uart7，如果只需要打开一个，那么就填写一个即可。

```
[orangepi@orangepi ~]$ sudo vim /boot/extlinux/extlinux.conf
LABEL Orange Pi
LINUX /Image
FDT /dtbs/rockchip/rk3588s-orangepi-cm5-tablet.dtb
FDTOVERLAYS    /dtbs/rockchip/overlay/rk3588-uart0-m2.dtbo
/dtbs/rockchip/overlay/rk3588-uart1-m1.dtbo
/dtbs/rockchip/overlay/rk3588-uart4-m0.dtbo
/dtbs/rockchip/overlay/rk3588-uart6-m2.dtbo
/dtbs/rockchip/overlay/rk3588-uart7-m2.dtbo
```

上面红色字体配置需要写在一行，不同的配置之间需要用空格隔开。

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

```
[orangepi@orangepi ~]$ ls /dev/ttyS*
/dev/ttyS1    /dev/ttyS4    /dev/ttyS6    /dev/ttyS7 .....
```

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

UART 总线	RX 对应 26pin	TX 对应 26pin
UART1_M1	3 号引脚	5 号引脚
UART4_M0	13 号引脚	11 号引脚
UART6_M1	10 号引脚	8 号引脚
UART7_M2	24 号引脚	26 号引脚

4) 使用 **gpio serial** 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

a. 测试 UART1

```
[orangepi@orangepi ~]$ sudo gpio serial /dev/ttyS1
[sudo] password for orangepi:    #在这里输入密码

Out:    0:   ->    0
```

```
Out:    1:  ->    1
Out:    2:  ->    2
Out:    3:  ->    3
Out:    4:  ->    4
Out:    5:  ->    5^C
```

b. 测试 UART4

```
[orangepi@orangepi ~]$ sudo gpio serial /dev/ttyS4
[sudo] password for orangepi:   #在这里输入密码

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

c. 测试 UART6

```
[orangepi@orangepi ~]$ sudo gpio serial /dev/ttyS6
[sudo] password for orangepi:   #在这里输入密码

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

d. 测试 UART7

```
[orangepi@orangepi ~]$ sudo gpio serial /dev/ttyS7
[sudo] password for orangepi:   #在这里输入密码

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

5.7.5. 26pin SPI 测试

1) 由下表可知，Orange Pi CM5 Base Tablet 可用的 spi 为 spi0 和 spi1。

复用功能	复用功能	复用功能	GPIO	GPIO序号	引脚序号	引脚序号	GPIO序号	GPIO	复用功能	复用功能
			3.3V		1	2		5V		
PWM13_M2(febf0010)	UART1_RX_M1	I2C5_SDA_M3	GPIO1_B7	47	3	4		5V		
	UART1_TX_M1	I2C5_SCL_M3	GPIO1_B6	46	5	6		GND		
	PWM1_M2(fd8b0010)	I2C4_SCL_M3	GPIO1_A3	35	7	8	56	GPIO1_D0	UART6_TX_M2	SPI1_MISO_M2
			GND		9	10	57	GPIO1_D1	UART6_RX_M2	SPI1_MOSI_M2
PWM0_M1(fd8b0000)	UART4_TX_M0	SPI1_CLK_M2	GPIO1_D2	58	11	12	34	GPIO1_A2	I2C4_SDA_M3	
	UART4_RX_M0	SPI1_CS0_M2	GPIO1_D3	59	13	14		GND		
			GPIO1_B0	40	15	16	36	GPIO1_A4		
			3.3V		17	18	38	GPIO1_A6		
		SPI0_MOSI_M2	GPIO1_B2	42	19	20		GND		
		SPI0_MISO_M2	GPIO1_B1	41	21	22	39	GPIO1_A7	PWM3_M3(fd8b0030)	
		SPI0_CLK_M2	GPIO1_B3	43	23	24	44	GPIO1_B4	SPI0_CS0_M2	UART7_RX_M2
			GND		25	26	45	GPIO1_B5	SPI0_CS1_M2	UART7_TX_M2

在 OPi OS Arch 系统中，26pin 中的 spi0 和 spi1 默认是关闭的，需要手动打开才能使用。

在/boot/extlinux/extlinux.conf 中加入下面红色字体部分的配置，然后重启 OPi OS Arch 系统就可以打开 spi0 和 spi1。

```
[orangepi@orangepi ~]$ sudo vim /boot/extlinux/extlinux.conf
LABEL Orange Pi
LINUX /Image
FDT /dtbs/rockchip/rk3588s-orangepi-cm5-tablet.dtb
FDTOVERLAYS /dtbs/rockchip/overlay/rk3588-spi0-m2-cs0-cs1-spidev.dtbo
rk3588-spi1-m0-cs0-spidev.dtbo
```

2) SPI0 和 SPI1 在 26pin 中对应的引脚如下表所示：

	SPI1_M2 对应 26pin	SPI0_M2 对应 26pin
MOSI	**10 号引脚**	**19 号引脚**
MISO	**8 号引脚**	**21 号引脚**
CLK	**11 号引脚**	**23 号引脚**
CS0	**13 号引脚**	**24 号引脚**
CS1	无	**26 号引脚**

3) 然后查看下 OPi OS Arch 系统中是否存在 **spidevx.x** 的设备节点，如果存在，说明 SPI 已经设置好了，可以直接使用。

```
[orangepi@orangepi ~]$ ls /dev/spidev*
/dev/spidev0.0 /dev/spidev1.0
```

4) 然后先不短接 SPI0 或者 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。

```
[orangepi@orangepi ~]$ sudo spidev_test -v -D /dev/spidev0.0
或者
[orangepi@orangepi ~]$ sudo spidev_test -v -D /dev/spidev1.0
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 SPI0 或者 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。

```
[orangepi@orangepi ~]$ sudo spidev_test -v -D /dev/spidev0.0
或者
[orangepi@orangepi ~]$ sudo spidev_test -v -D /dev/spidev1.0
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    | ......@.....................
```

5.7.6. PWM 的测试方法

1) 由下表可知，Orange Pi CM5 Base Tablet 可用的 pwm 有 pwm0、pwm1、pwm3、和 pwm13 共四路 pwm。

复用功能	复用功能	复用功能	GPIO	GPIO序号	引脚序号	引脚序号	GPIO序号	GPIO	复用功能	复用功能
			3.3V		1	2		5V		
PWM13_M2(febf0010)	UART1_RX_M1	I2C5_SDA_M3	GPIO1_B7	47	3	4		5V		
	UART1_TX_M1	I2C5_SCL_M3	GPIO1_B6	46	5	6		GND		
	PWM1_M2(fd8b0010)	I2C4_SCL_M3	GPIO1_A3	35	7	8	56	GPIO1_D0	UART6_TX_M2	SPI1_MISO_M2
			GND		9	10	57	GPIO1_D1	UART6_RX_M2	SPI1_MOSI_M2
PWM0_M1(fd8b0000)	UART4_TX_M0	SPI1_CLK_M2	GPIO1_D2	58	11	12	34	GPIO1_A2	I2C4_SDA_M3	
	UART4_RX_M0	SPI1_CS0_M2	GPIO1_D3	59	13	14		GND		
			GPIO1_B0	40	15	16	36	GPIO1_A4		
			3.3V		17	18	38	GPIO1_A6		
		SPI0_MOSI_M2	GPIO1_B2	42	19	20		GND		
		SPI0_MISO_M2	GPIO1_B1	41	21	22	39	GPIO1_A7	PWM3_M3(fd8b0030)	
		SPI0_CLK_M2	GPIO1_B3	43	23	24	44	GPIO1_B4	SPI0_CS0_M2	UART7_RX_M2
			GND		25	26	45	GPIO1_B5	SPI0_CS1_M2	UART7_TX_M2

在 OPi OS Arch 系统中，26pin 中的 pwm 默认都是关闭的，需要手动打开才能使用。

在/boot/extlinux/extlinux.conf 中加入下面红色字体部分的配置，然后重启 OPi OS Arch 系统就可以同时打开 pwm0、pwm1、pwm3 和 pwm13，如果只需要打开一个，那么就填写一个即可。

```
[orangepi@orangepi ~]$ sudo vim /boot/extlinux/extlinux.conf
LABEL Orange Pi
LINUX /Image
FDT /dtbs/rockchip/rk3588s-orangepi-cm5-tablet.dtb
FDTOVERLAYS    /dtbs/rockchip/overlay/rk3588-pwm0-m1.dtbo
/dtbs/rockchip/overlay/rk3588-pwm0-m1.dtbo
/dtbs/rockchip/overlay/rk3588-pwm1-m2.dtbo
/dtbs/rockchip/overlay/rk3588-pwm3-m3.dtbo
/dtbs/rockchip/overlay/rk3588-pwm13-m2.dtbo
```

上面红色字体配置需要写在一行，不同的配置之间需要用空格隔开。

2) 当打开一个 pwm 后，在**/sys/class/pwm/**中就会多出一个 pwmchipX（X 为具体的的数字），比如打开 pwm13 后，查看**/sys/class/pwm/**下的 pwmchipX 会由两个变成了三个。

```
[orangepi@orangepi ~]$ ls /sys/class/pwm/
pwmchip0    pwmchip1    pwmchip2
```

3) 上面哪个 pwmchip 对应 pwm13 呢，我们先查看下 **ls /sys/class/pwm/ -l** 命令的输出，如下所示：

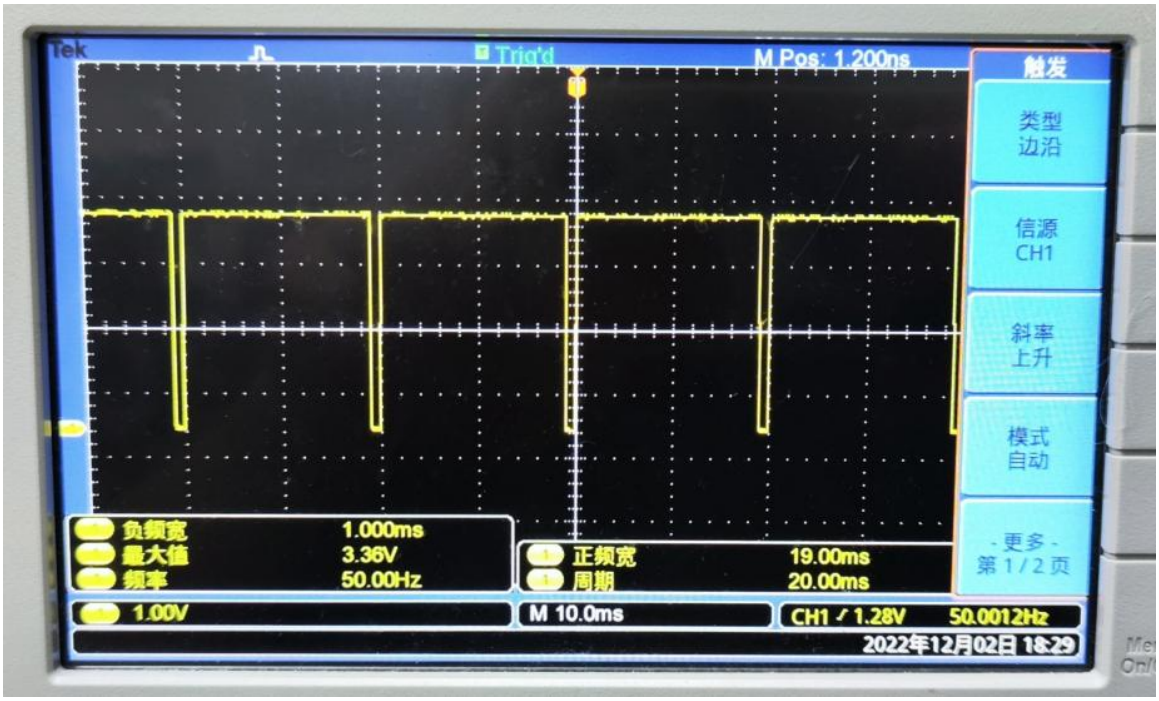
```
[orangepi@orangepi ~]$ ls /sys/class/pwm/ -lh
total 0
lrwxrwxrwx 1 root root 0  4月11日 16:53 pwmchip0 -> ../../devices/platform/fd8b0010.pwm/pwm/pwmchip0
lrwxrwxrwx 1 root root 0  4月11日 16:53 pwmchip1 -> ../../devices/platform/febf0010.pwm/pwm/pwmchip1
lrwxrwxrwx 1 root root 0  4月11日 16:53 pwmchip2 -> ../../devices/platform/febf0020.pwm/pwm/pwmchip2
[orangepi@orangepi ~]$
```

4) 然后由下表可知，pwm13 寄存器的基地址为 febf0010，再看 **ls /sys/class/pwm/ -l** 命令的输出，可以看到 pwmchip1 中链接到了 febf0010.pwm，所以 pwm13 对应 pwmchip 为 pwmchip1。

复用功能	复用功能	复用功能	GPIO	GPIO序号	引脚序号	引脚序号	GPIO序号	GPIO	复用功能	复用功能
			3.3V		1	2		5V		
PWM13_M2(febf0010)	UART1_RX_M1	I2C5_SDA_M3	GPIO1_B7	47	3	4		5V		
	UART1_TX_M1	I2C5_SCL_M3	GPIO1_B6	46	5	6		GND		
	PWM1_M2(fd8b0010)	I2C4_SCL_M3	GPIO1_A3	35	7	8	56	GPIO1_D0	UART6_TX_M2	SPI1_MISO_M2
			GND		9	10	57	GPIO1_D1	UART6_RX_M2	SPI1_MOSI_M2
PWM0_M1(fd8b0000)	UART4_TX_M0	SPI1_CLK_M2	GPIO1_D2	58	11	12	34	GPIO1_A2	I2C4_SDA_M3	
	UART4_RX_M0	SPI1_CS0_M2	GPIO1_D3	59	13	14		GND		
			GPIO1_B0	40	15	16	36	GPIO1_A4		
			3.3V		17	18	38	GPIO1_A6		
		SPI0_MOSI_M2	GPIO1_B2	42	19	20		GND		
		SPI0_MISO_M2	GPIO1_B1	41	21	22	39	GPIO1_A7	PWM3_M3(fd8b0030)	
		SPI0_CLK_M2	GPIO1_B3	43	23	24	44	GPIO1_B4	SPI0_CS0_M2	UART7_RX_M2
			GND		25	26	45	GPIO1_B5	SPI0_CS1_M2	UART7_TX_M2

5) 然后使用下面的命令可以让 pwm13 输出一个 50Hz 的方波（请先切换到 root 用户，再执行下面的命令）

```
[root@orangepi orangepi]# echo 0 > /sys/class/pwm/pwmchip1/export
[root@orangepi orangepi]# echo 20000000 > /sys/class/pwm/pwmchip1/pwm0/period
[root@orangepi orangepi]# echo 1000000 > /sys/class/pwm/pwmchip1/pwm0/duty_cycle
[root@orangepi orangepi]# echo 1 > /sys/class/pwm/pwmchip1/pwm0/enable
```



6) 上面演示的 pwm13 的测试方法，其他 pwm 测试方法都是类似的。

6. Linux SDK——orangepi-build 使用说明

6.1. 编译系统需求

我们可以在 x64 的电脑中交叉编译开发板的 Linux 镜像，也可以在开发板的 Ubuntu22.04 系统中来编译开发板的 Linux 镜像，请根据自己的喜好二选一。

如果是在开发板的 Ubuntu22.04 系统中使用 orangepi-build 来编译 Linux 镜像，请做好散热。如果散热没做好，容易出现文件系统跑飞的错误。

6.1.1. 使用开发板的 Ubuntu22.04 系统编译

1) Linux SDK，即 **orangepi-build**，支持在开发板的 **Ubuntu 22.04** 的上运行（其它系统没有测试过），所以下载 orangepi-build 前，请首先确保开发板已安装的 Ubuntu 版本是 Ubuntu 22.04。查看开发板已安装的 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。

```
orangepi@orangepi:~$ lsb_release -a
No LSB modules are available.
Distributor ID:	Ubuntu
Description:	Ubuntu 22.04.1 LTS
Release:	22.04
Codename:	jammy
```

2) 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开发板能正常从 GitHub 下载代码，这点是非常重要的。

6.1.2. 使用 x64 的 Ubuntu22.04 电脑编译

1) Linux SDK，即 **orangepi-build**，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。

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-22.04.3-desktop-amd64.iso
或者
https://repo.huaweicloud.com/ubuntu-releases/22.04/ubuntu-22.04.3-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。

c. 需要替换的**/etc/apt/sources.list** 文件的内容为：

```
test@test:~$ sudo mv /etc/apt/sources.list /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。

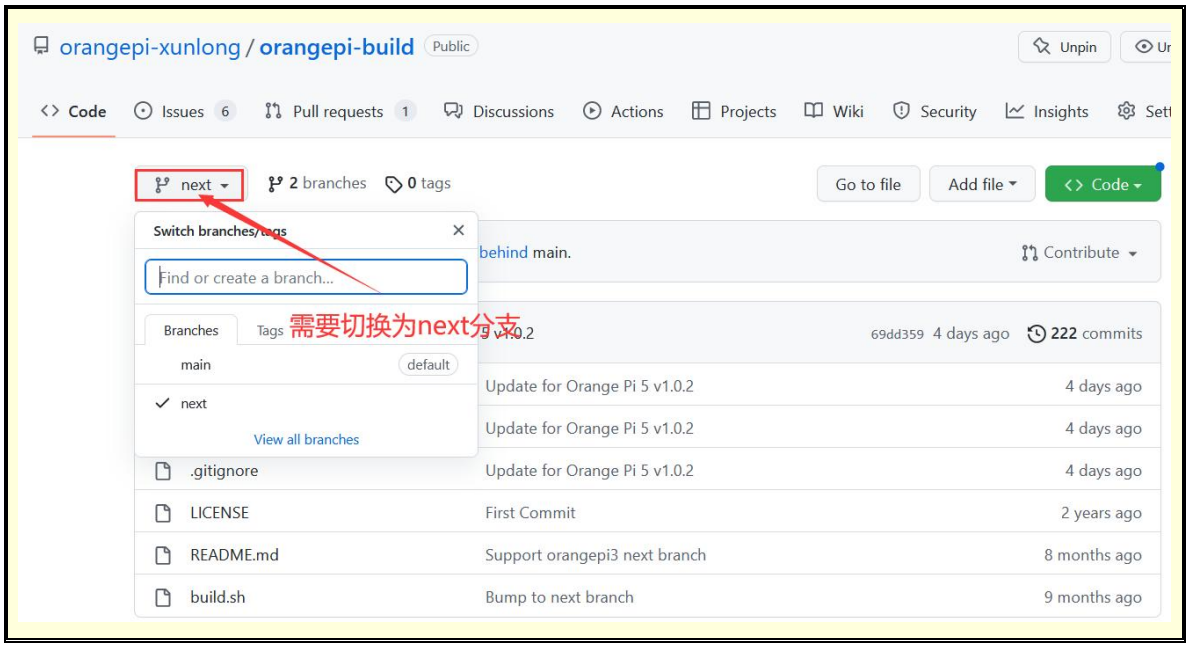
6.2. 获取 linux sdk 的源码

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，命令如下所示：

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

注意，Orange Pi CM5 Base Tablet 开发板是需要下载 orangepi-build 的 next 分支源码的，上面的 git clone 命令需要指定 orangepi-build 源码的分支为 next。



通过 git clone 命令下载 orangepi-build 的代码是不需要输入 github 账号的用户名和密码的（下载本手册中的其他代码也是一样的），如果如输入 git clone 命令后 Ubuntu PC 提示需要输入 github 账号的用户名和密码，一般都是 git clone 后面的 orangepi-build 仓库的地址输入错误了，请仔细检查命令拼写是否有错误，而不是以为我们这里忘了提供 github 账号的用户名和密码。

2) 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
legacy	**u-boot 2017.09**	**linux5.10**
current	**u-boot 2017.09**	**linux6.1**

这里所说的分支和 orangepi-build 源代码的分支不是同一个东西，请不要搞混了。此分支主要是用来区分不同内核源码版本的。

目前 RK 提供的 linux5.10 bsp 内核我们定义为 legacy 分支，linux6.1 bsp 内核我们定义为 current 分支。

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。

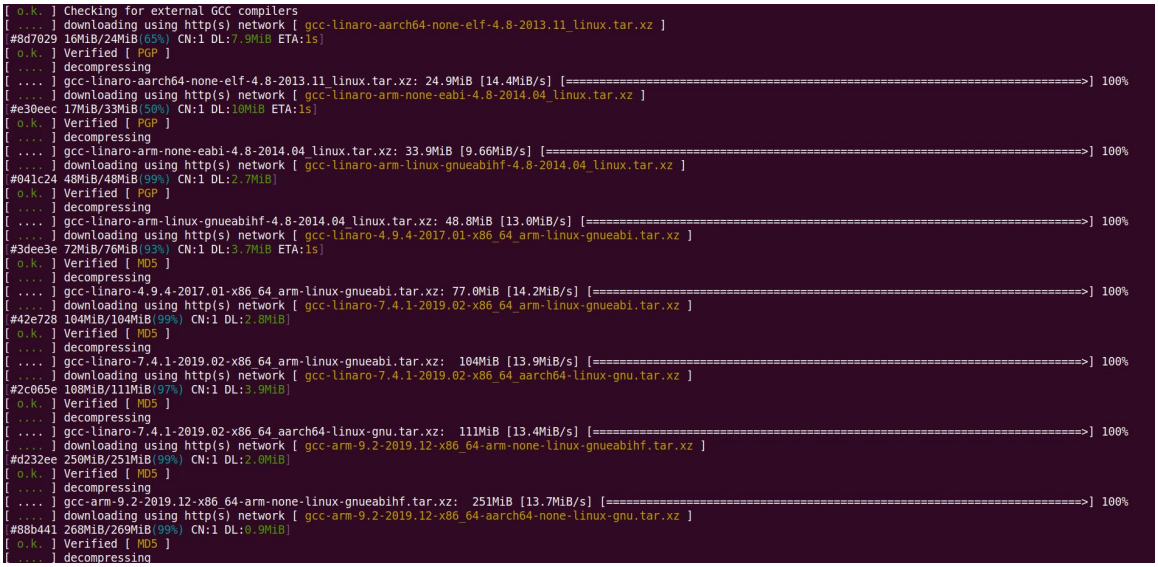
```
test@test:~/orangepi-build$ ls
build.sh  external  LICENSE  README.md  scripts
```

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

只有在 x64 的电脑中使用 orangepi-build 编译镜像才会下载交叉编译工具链。在开发板的 Ubuntu22.04 中编译开发板的 linux 镜像是不会下载交叉编译工具链的，此时 orangepi-build/toolchains 会是一个空文件夹。

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，开发板只会使用其中的两个。

```
test@test:~/orangepi-build$ ls toolchains/
gcc-arm-11.2-2022.02-x86_64-aarch64-none-linux-gnu
gcc-arm-11.2-2022.02-x86_64-arm-none-linux-gnueabihf
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 linux 内核源码使用的交叉编译工具链为：

a. linux5.10：

gcc-arm-11.2-2022.02-x86_64-aarch64-none-linux-gnu

b. Linux6.1：

gcc-arm-11.2-2022.02-x86_64-aarch64-none-linux-gnu

5) 编译 u-boot 源码使用的交叉编译工具链为：

a. v2017.09：

gcc-linaro-7.4.1-2019.02-x86_64_aarch64-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. linux5.10 内核源码存放的 git 仓库如下所示：

https://github.com/orangepi-xunlong/linux-orangepi/tree/orange-pi-5.10-rk35xx

b. linux6.1 内核源码存放的 git 仓库如下所示：

https://github.com/orangepi-xunlong/linux-orangepi/tree/orange-pi-6.1-rk35xx

c. u-boot 源码存放的 git 仓库如下所示：

```
https://github.com/orangepi-xunlong/u-boot-orangepi/tree/v2017.09-rk3588
```

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-5.10-rk35xx** 的文件夹存放的就是 RK3588/RK3588S 系列开发板 legacy 分支的内核源码，里面名为 **orange-pi-6.1-rk35xx** 的文件夹存放的就是 RK3588/RK3588S 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 **v2017.09-rk3588** 的文件夹存放的就是 RK3588/RK3588S 系列开发板 legacy 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。

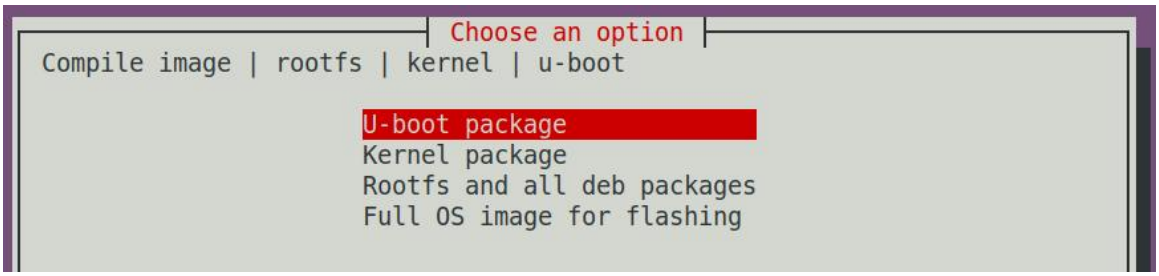
```
test@test:~/orangepi-build$ ls
build.sh  external  kernel  LICENSE  output  README.md  scripts  toolchains
u-boot  userpatches
```

6.3. 编译 u-boot

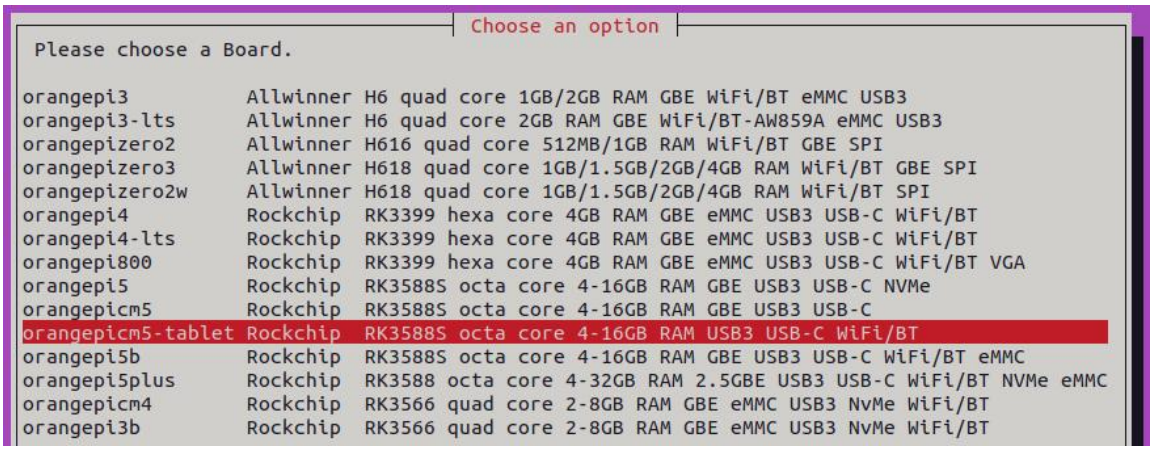
1) 运行 build.sh 脚本，记得加 sudo 权限。

```
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 [**v2017.09**]

b. 交叉编译工具链的版本。

[o.k.] Compiler version [**aarch64-linux-gnu-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-orangepicm5-tablet_1.0.0_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=orangepicm5-tablet BRANCH=legacy BUILD_OPT=u-boot KERNEL_CONFIGURE=no**]

5) 查看编译生成的 u-boot deb 包。

```
test@test:~/orangepi-build$ ls output/debs/u-boot/
linux-u-boot-legacy-orangepicm5-tablet_1.0.0_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-legacy-orangepicm5-tablet_1.0.0_arm64.deb .     (注意命令最后有个“.”)
test@test:~/orangepi_build/output/debs/u-boot$ ls
linux-u-boot-legacy-orangepicm5-tablet_1.0.0_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-legacy-orangepicm5-tablet_1.0.0_arm64
    │   ├── idbloader.img
    │   ├── rkspi_loader.img
    │   └── u-boot.itb
    └── u-boot
        ├── LICENSE
        ├── orangepi_5_defconfig
        └── platform_install.sh

3 directories, 6 files
```

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-legacy-orangepicm5-tablet_1.0.0_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-orangepicm5-tablet-legacy
```

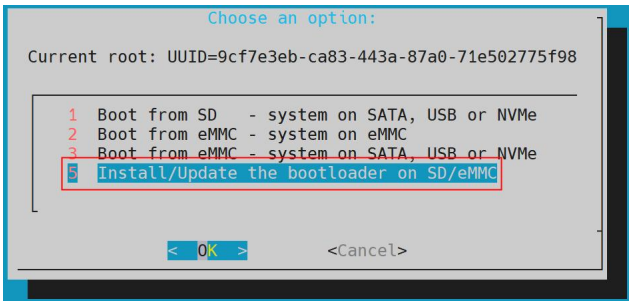
c. 再安装刚才上传的新的 u-boot 的 deb 包。

```
root@orangepi:~# dpkg -i linux-u-boot-legacy-orangepicm5-tablet_1.0.0_arm64.deb
```

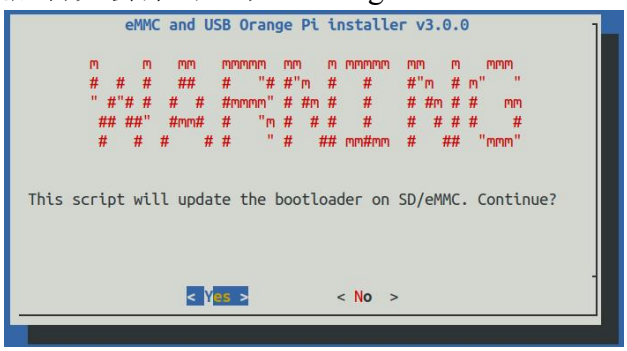
d. 然后运行 nand-sata-install 脚本。

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

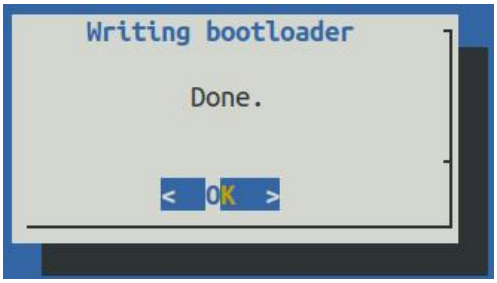
e. 然后选择 **5 Install/Update the bootloader on SD/eMM** 来更新 TF 卡或者 eMMC 中的 u-boot。



f. 按下回车键后首先会弹出一个 Warning。



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。

9) 其它有用的信息。

a. u-boot 2017.09 源码中，开发板使用的 defconfig 配置文件为。

```
orangepi-build/u-boot/v2017.09-rk3588/configs/orangepi_cm5_tablet_defconfig
```

b. u-boot 2017.09 源码中，开发板使用 dts 文件为。

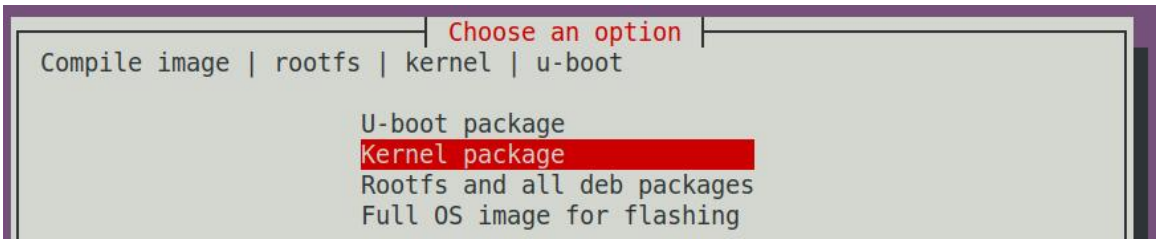
```
orangepi-build/u-boot/v2017.09-rk3588/arch/arm/dts/rk3588s-orangepi-cm5-tablet.dts
```

6.4. 编译 linux 内核

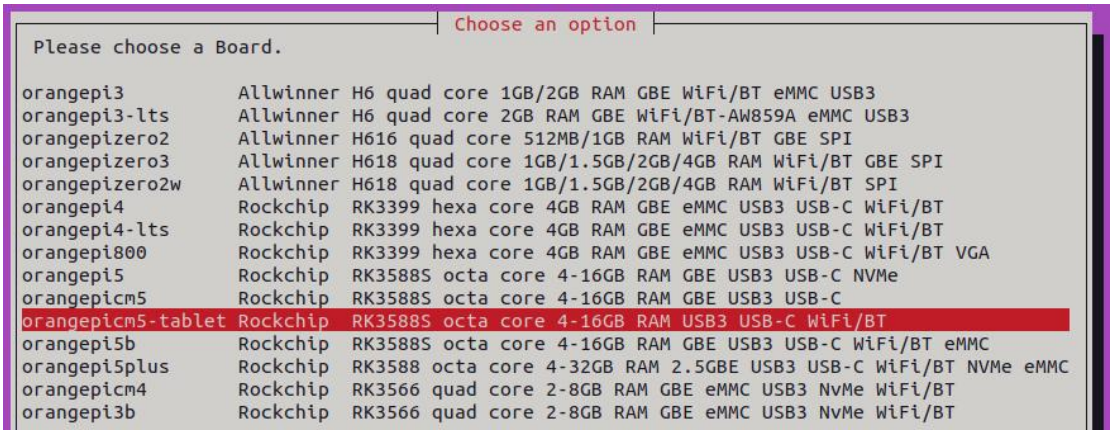
1) 运行 build.sh 脚本，记得加 sudo 权限。

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

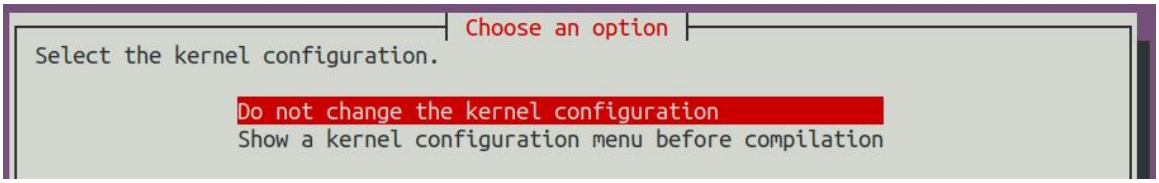
2) 选择 **Kernel package**，然后回车。



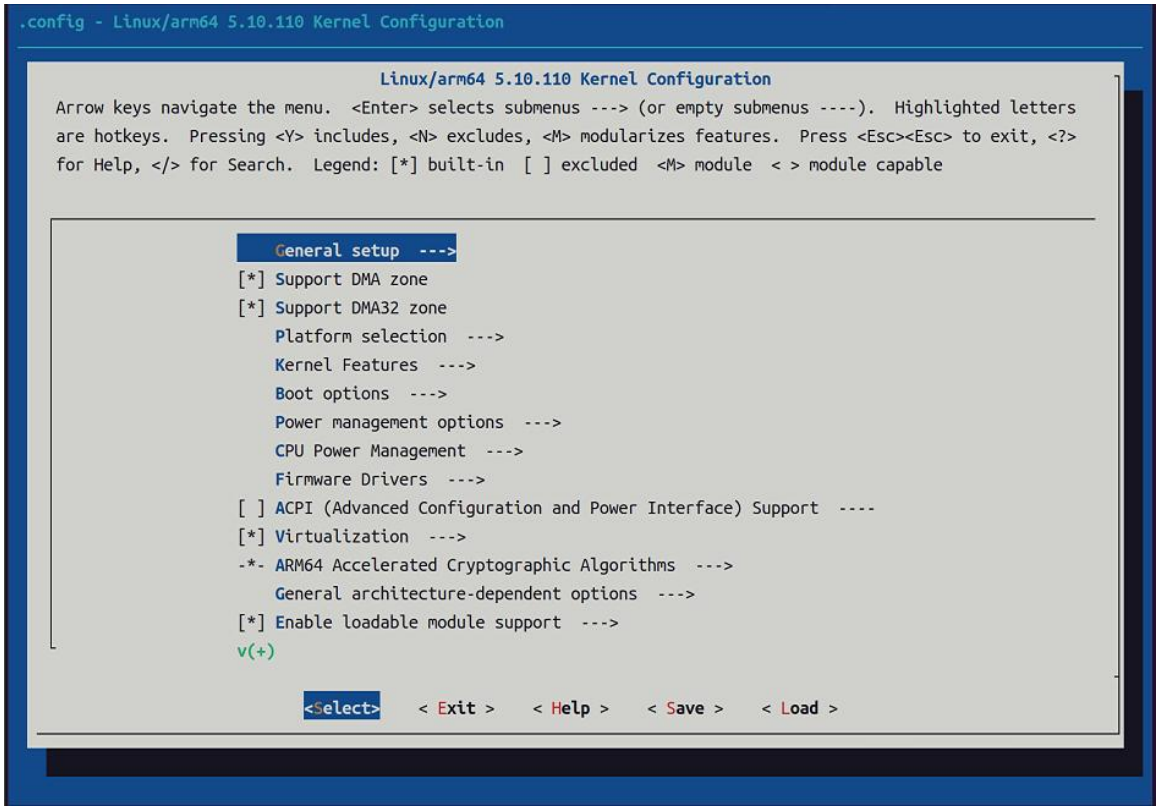
3) 接着选择开发板的型号。



4) 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。



5) 如果第 4) 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 [ 5.10.160 ]
```

b. 使用的交叉编译工具链的版本。

```
[ o.k. ] Compiler version [ aarch64-none-linux-gnu-gcc 11.2.1 ]
```

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-rockchip-rk3588-legacy.config ]
```

d. 编译生成的内核相关的 deb 包的路径。

```
[ o.k. ] Target directory [ orangepi-build/output/debs/ ]
```

e. 编译生成的内核镜像 deb 包的包名。

```
[ o.k. ] File name [ linux-image-legacy-rockchip-rk3588_1.0.0_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=orangepicm5-tablet
BRANCH=legacy BUILD_OPT=kernel KERNEL_CONFIGURE=no ]
```

7) 查看编译生成的内核相关的 deb 包。

a. **linux-dtb-legacy-rockchip-rk3588_1.0.0_arm64.deb** 包含内核使用的 dtb 文件。

b. **linux-headers-legacy-rockchip-rk3588_1.0.0_arm64.deb** 包含内核头文件。

c. **linux-image-legacy-rockchip-rk3588_1.0.0_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-legacy-rockchip-rk3588_1.0.0_arm64.deb
output/debs/linux-image-legacy-rockchip-rk3588_1.0.0_arm64.deb
output/debs/linux-headers-legacy-rockchip-rk3588_1.0.0_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-legacy-rockchip-rk3588_1.0.0_arm64.deb test/
test@test:~/orangepi_build/output/debs$ cd test
test@test:~/orangepi_build/output/debs/test$ dpkg -x \
```

```
linux-image-legacy-rockchip-rk3588_1.0.0_arm64.deb .
test@test:~/orangepi_build/output/debs/test$ ls
boot  etc  lib  linux-image-legacy-rockchip-rk3588_1.0.0_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-5.10.160-rockchip-rk3588
│   ├── System.map-5.10.160-rockchip-rk3588
│   └── vmlinuz-5.10.160-rockchip-rk3588
├── etc
│   └── kernel
├── lib
│   └── modules
├── linux-image-legacy-rockchip-rk3588_1.0.0_arm64.deb
└── usr
    ├── lib
    └── share
```

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-legacy-rockchip-rk3588_1.0.0_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-legacy-rockchip-rk3588
```

c. 再安装刚才上传的新的 linux 内核的 deb 包。

```
root@orangepi:~# dpkg -i linux-image-legacy-rockchip-rk3588_1.0.0_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
```

10) 其它有用的信息。

a. 内核配置文件存放位置如下所示，请不要到内核源码中去找开发板所使用的内核配置文件。

```
orangepi-build/external/config/kernel/linux-rockchip-rk3588-legacy-opicm5-tablet.config
```

b. 开发板使用的 dts 文件所在的位置为：

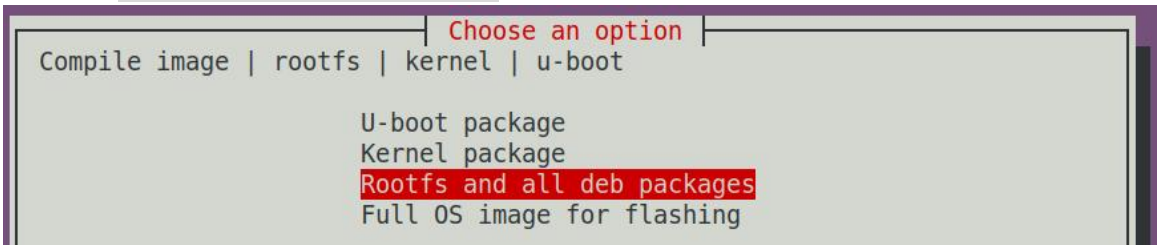
```
orangepi-build/kernel/orange-pi-5.10-rk35xx/arch/arm64/boot/dts/rockchip/rk3588s-orangepi-cm5-tablet.dts
```

6.5. 编译 rootfs

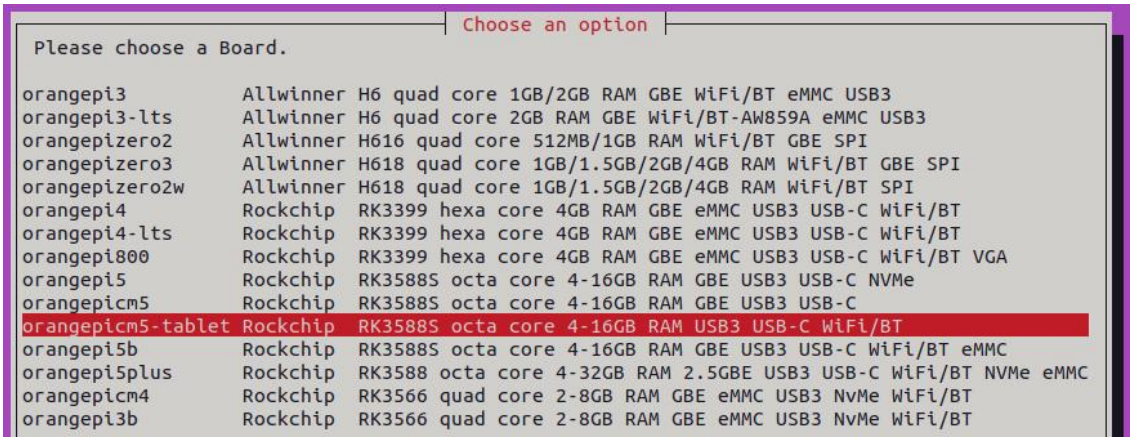
1) 运行 build.sh 脚本，记得加 sudo 权限。

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

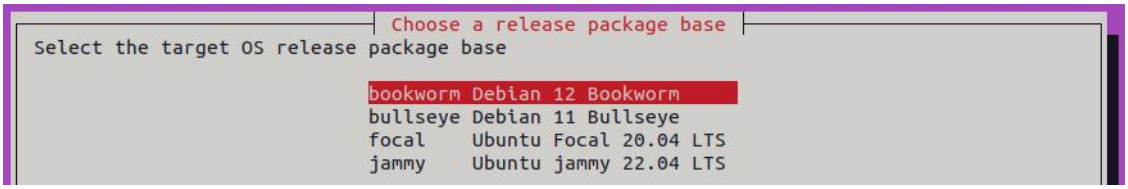
2) 选择 **Rootfs and all deb packages**，然后回车。



3) 接着选择开发板的型号。



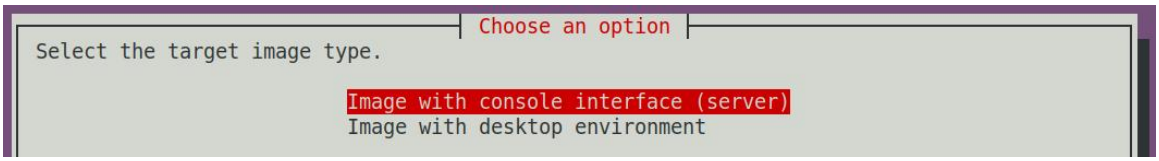
4) 然后选择 rootfs 的类型。



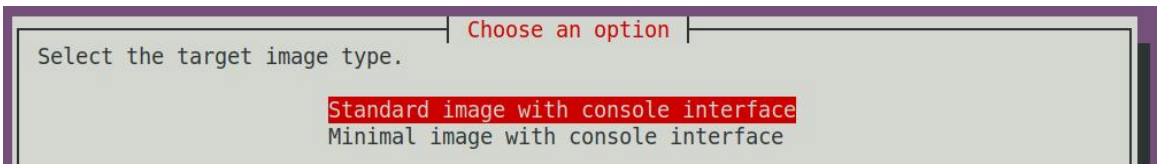
5) 然后选择镜像的类型。

a. **Image with console interface (server)**表示服务器版的镜像，体积比较小。

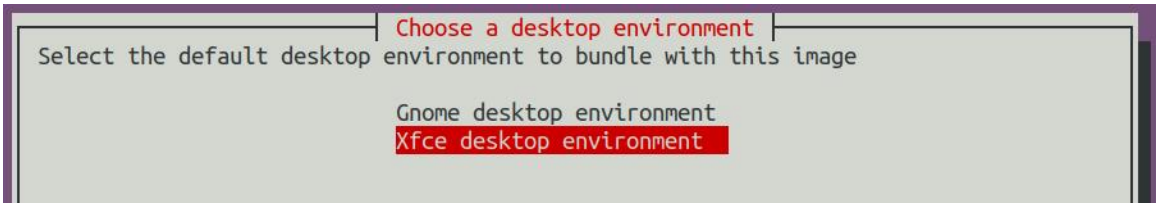
b. **Image with desktop environment** 表示带桌面的镜像，体积比较大。

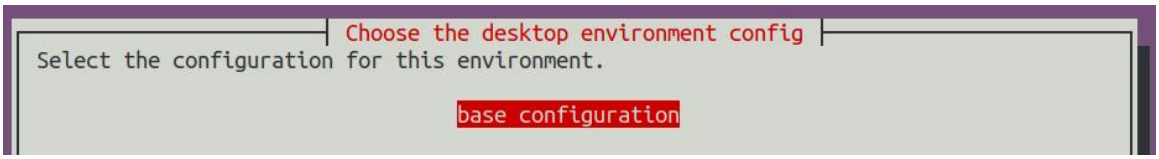


6) 如果是编译服务器版的镜像，还可以选择编译 Standard 版本或者 Minimal 版本，Minimal 版本预装的软件会比 Standard 版本少很多（**没特殊需求请不要选择 Minimal 版本，因为很多东西默认没有预装，部分功能可能用不了**）。

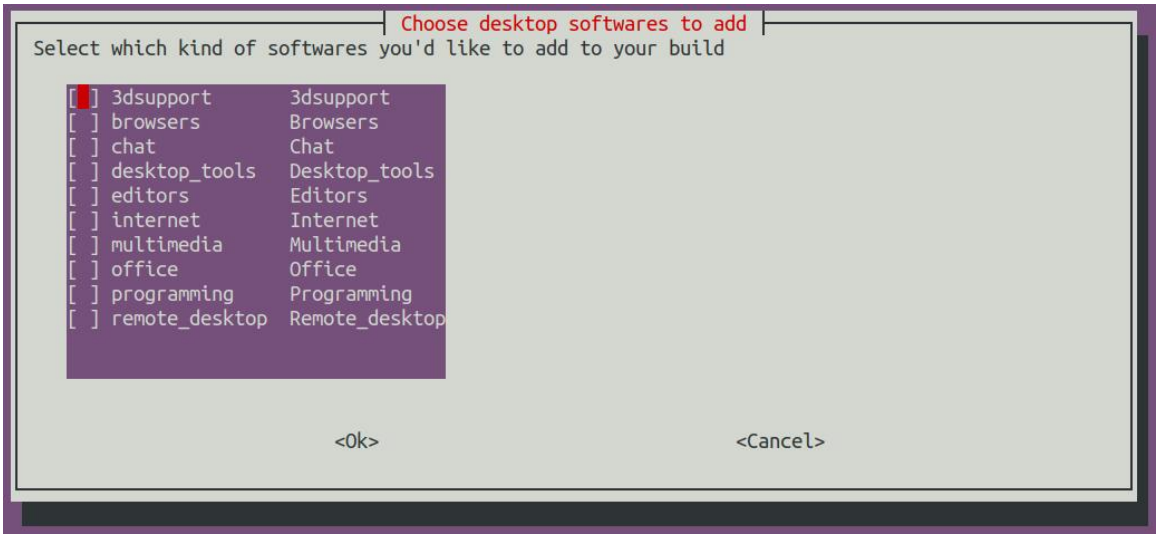


7) 如果是编译桌面版本的镜像还需要选择桌面环境的类型，目前 Ubuntu Jammy 主要维护 XFCE 和 Gnome 两种桌面，Ubuntu Focal 只维护 XFCE 桌面，Debian Bullseye 主要维护 XFCE 和 KDE 桌面，Debian Bookwork 主要维护 XFCE 桌面。





然后可以选择需要安装的额外的软件包。这里请按下回车键直接跳过。



8) 然后就会开始编译 rootfs，编译时提示的部分信息说明如下所示：

a. rootfs 的类型。

[o.k.] local not found [Creating new rootfs cache for **jammy**]

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 [**jammy-xfce-arm64.f930ff6ebbac1a72108a2e100762b18f.tar.lz4**]

d. 编译使用的时间。

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

9) 查看编译生成的 rootfs 压缩包。

a. **jammy-xfce-arm64.f930ff6ebbac1a72108a2e100762b18f.tar.lz4** 是 rootfs 的压缩包，名字各字段的含义为：
   a) **jammy** 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) **f930ff6ebbac1a72108a2e100762b18f** 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. **jammy-xfce-arm64.f930ff6ebbac1a72108a2e100762b18f.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/
jammy-xfce-arm64.f930ff6ebbac1a72108a2e100762b18f.tar.lz4
jammy-xfce-arm64.f930ff6ebbac1a72108a2e100762b18f.tar.lz4.current
jammy-xfce-arm64.f930ff6ebbac1a72108a2e100762b18f.tar.lz4.list
```

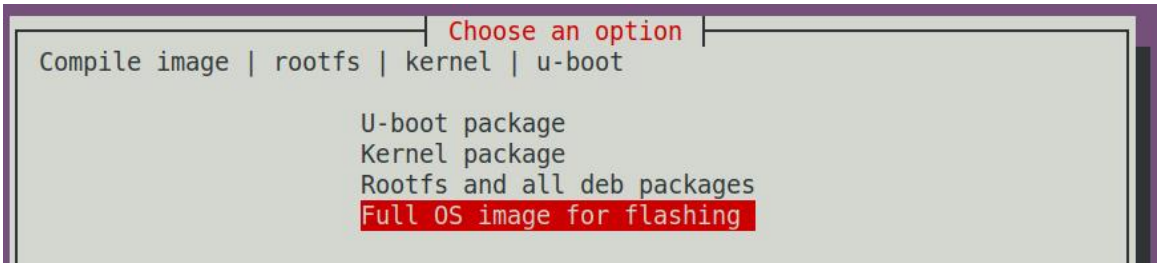
10) 如果需要的 rootfs 在 **external/cache/rootfs** 下已经存在，那么再次编译 rootfs 就会直接跳过编译过程，不会重新开始编译，编译镜像的时候也会去 **external/cache/rootfs** 下查找是否已经有缓存可用的 rootfs，如果有就直接使用，这样可以节省大量的下载编译时间。

6.6. 编译 linux 镜像

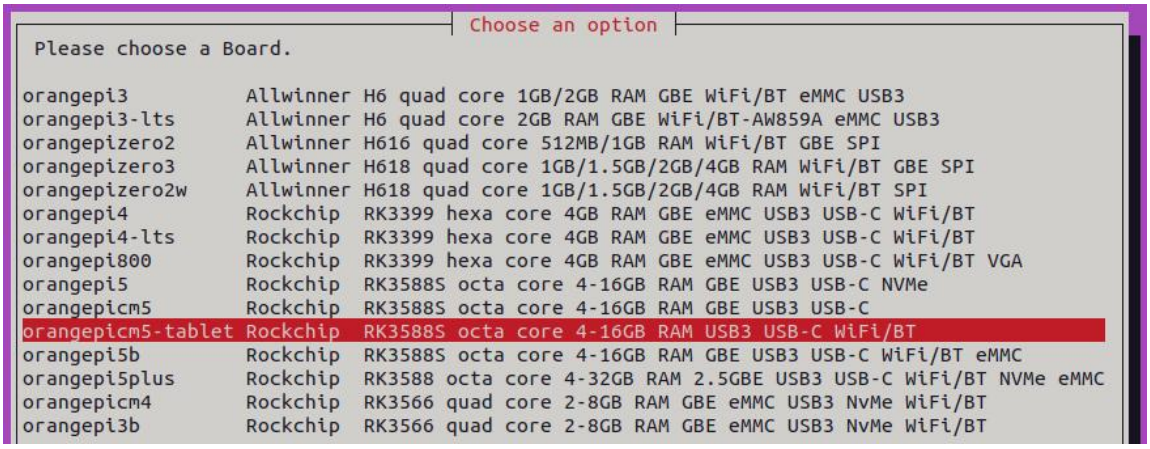
1) 运行 build.sh 脚本，记得加 sudo 权限。

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

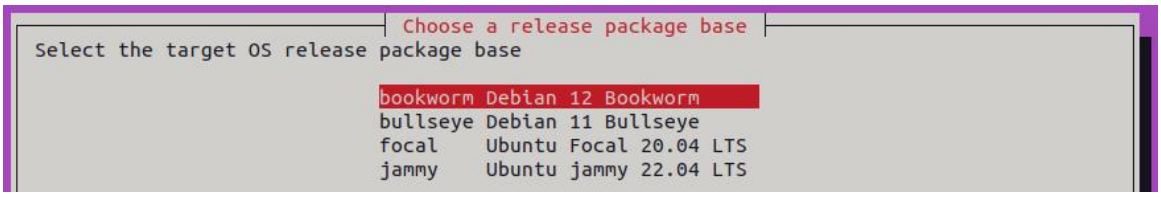
2) 选择 **Full OS image for flashing**，然后回车。



3) 然后选择开发板的型号。



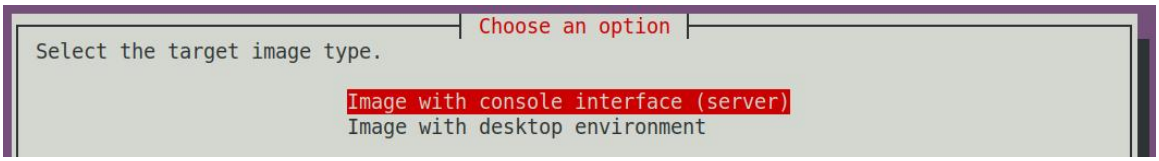
4) 然后选择 rootfs 的类型。



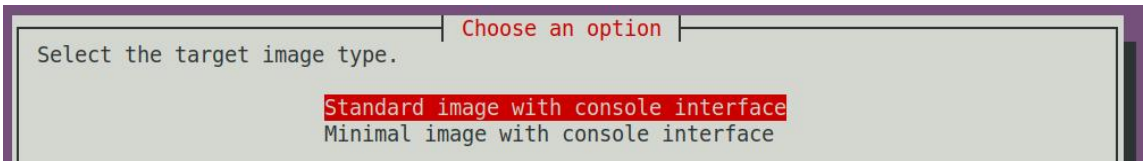
5) 然后选择镜像的类型。

a. **Image with console interface (server)**表示服务器版的镜像，体积比较小。

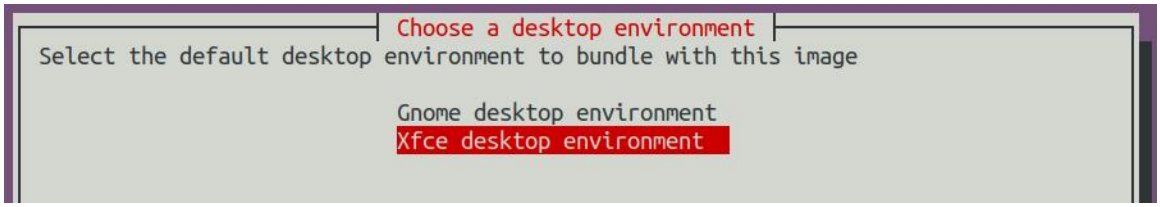
b. **Image with desktop environment** 表示带桌面的镜像，体积比较大。

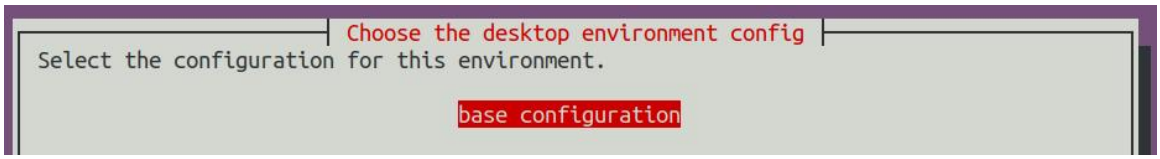


6) 如果是编译服务器版的镜像，还可以选择编译 Standard 版本或者 Minimal 版本，Minimal 版本预装的软件会比 Standard 版本少很多（**没特殊需求请不要选择 Minimal 版本，因为很多东西默认没有预装，部分功能可能用不了**）。

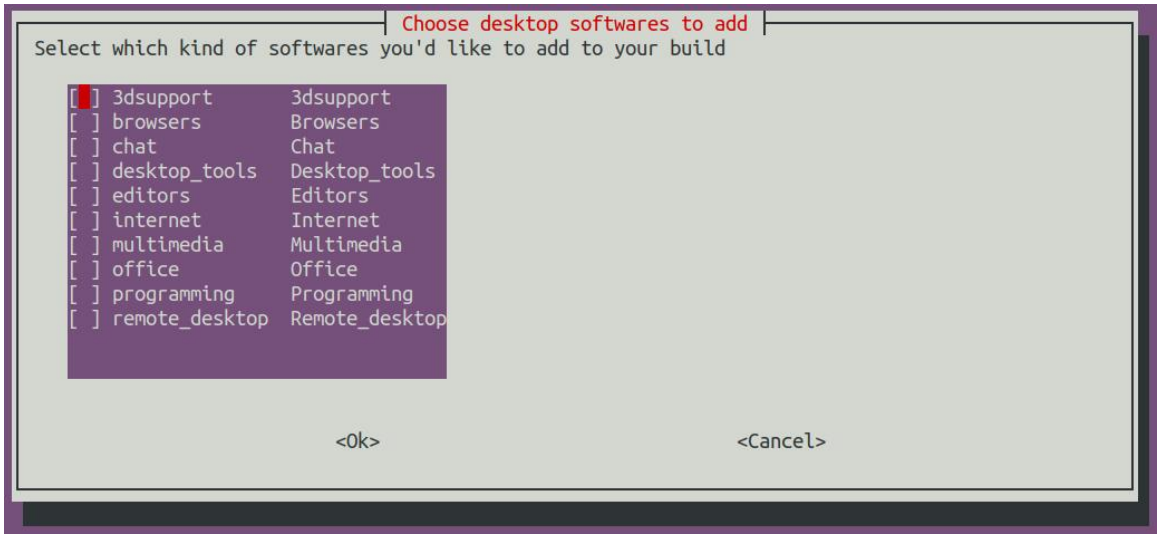


7) 如果是编译桌面版本的镜像还需要选择桌面环境的类型，目前 Ubuntu Jammy 主要维护 XFCE 和 Gnome 两种桌面，Ubuntu Focal 只维护 XFCE 桌面，Debian Bullseye 主要维护 XFCE 和 KDE 桌面，Debian Bookwork 主要维护 XFCE 桌面。





然后可以选择需要安装的额外的软件包。这里请按下回车键直接跳过。



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/orangepicm5-tablet_1.0.0_debian_bullseye_desktop_xfce_linux5.10.160/orangepicm5-tablet_1.0.0_debian_bullseye_desktop_xfce_linux5.10.160.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=orangepicm5-tablet BRANCH=legacy BUILD_OPT=image RELEASE=bullseye BUILD_MINIMAL=no BUILD_DESKTOP=no KERNEL_CONFIGURE=yes ]
```

7. Linux 开发手册

7.1. 在开发板的 linux 系统中单独编译内核源码的方法

1) 首先下载开发板的 Linux 内核源码。

```
orangepi@orangepi:~$ git clone --depth=1 -b orange-pi-5.10-rk35xx https://github.com/orangepi-xunlong/linux-orangepi
```

如果从 github 下载代码有问题，可以去开发板的官方工具中下载内核源码压缩包，然后上传到开发板的 linux 系统中，再解压即可。

解压内核源码压缩包的命令为：

orangepi@orangepi:~$ **tar zxf orange-pi-5.10-rk35xx.tar.gz**
orangepi@orangepi:~$ **mv orange-pi-5.10-rk35xx linux-orangepi**

解压后请执行下面的命令和 github 同步下源码，确保源码为最新的状态：

orangepi@orangepi:~$ **cd linux-orangepi**
orangepi@orangepi:~/linux-orangepi$ **git pull**

2) 然后配置下默认的内核配置。

```
orangepi@orangepi:~$ cd linux-orangepi
orangepi@orangepi:~/linux-orangepi$ make rockchip_linux_defconfig
```

rockchip_linux_defconfig 在内核源码中的路径为 arch/arm64/configs/

3) 然后编译内核源码。

```
orangepi@orangepi:~/linux-orangepi$ make -j10
```

4) 然后安装下内核模块。

```
orangepi@orangepi:~/linux-orangepi$ sudo make modules_install
```

内核模块的安装路径为：/lib/modules

执行完 sudo make modules_install 命令后可以看到/lib/modules/下会多了一个内核模块的文件夹：

orangepi@orangepicm5-tablet:~$ **ls /lib/modules**

5.10.160+ 5.10.160-rockchip-rk3588

5) 然后安装内核镜像和 uInitrd。

orangepi@orangepi:~/linux-orangepi$ **sudo make install**

内核镜像和 uInitrd 的安装路径为：/boot/

执行完 sudo make install 命令后可以看到/boot/下会多了一个内核文件：

orangepi@orangepicm5-tablet:~/orange-pi-5.10-rk3588$ **ls /boot/vmlinuz***

/boot/vmlinuz-5.10.160+ /boot/vmlinuz-5.10.160-rockchip-rk3588

系统启动时实际加载的是/boot/Image 这个文件，Image 是 vmlinuz 文件的拷贝。

6) 然后安装 dtb 文件到**/boot/dtb** 中。

orangepi@orangepi:~/linux-orangepi$ **sudo make dtbs_install INSTALL_DTBS_PATH=/boot/dtb/**

7) 然后重启 Linux 系统就会加载新编译的内核了。

orangepi@orangepi:~$ **uname -r**

5.10.160+

8. Android 13 系统的使用说明

8.1. 已支持的 Android 版本

Android 版本	内核版本
Android 13	Linux5.10

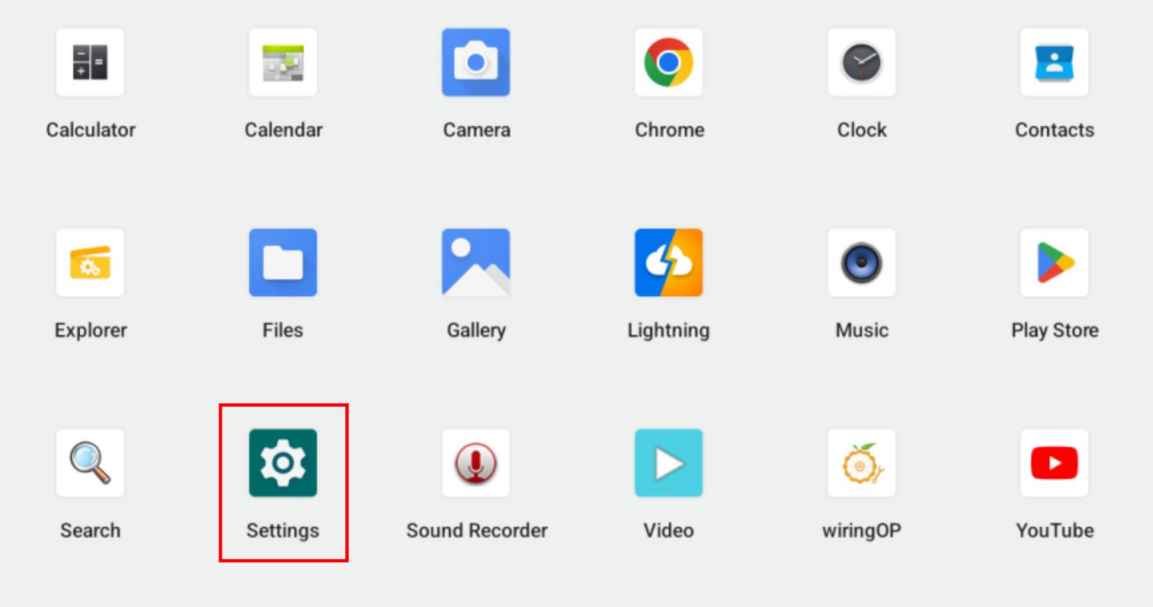
8.2. Android 功能适配情况

功能	Android 13
HDMI 显示	OK
HDMI 音频	OK
USB 2.0	OK
USB 3.0	OK
WIFI	OK
蓝牙	OK
调试串口	OK
FAN 风扇	OK
eMMC 启动	OK
GPIO（26pin）	OK
UART（26pin）	OK
SPI（26pin）	OK
I2C（26pin）	OK
PWM（26pin）	OK
Camera1	OK
Camera2	NO
Camera3	NO
LCD	OK
板载 MIC	OK
耳机播放	OK
耳机录音	OK
喇叭 x 2	OK
LED 灯	OK

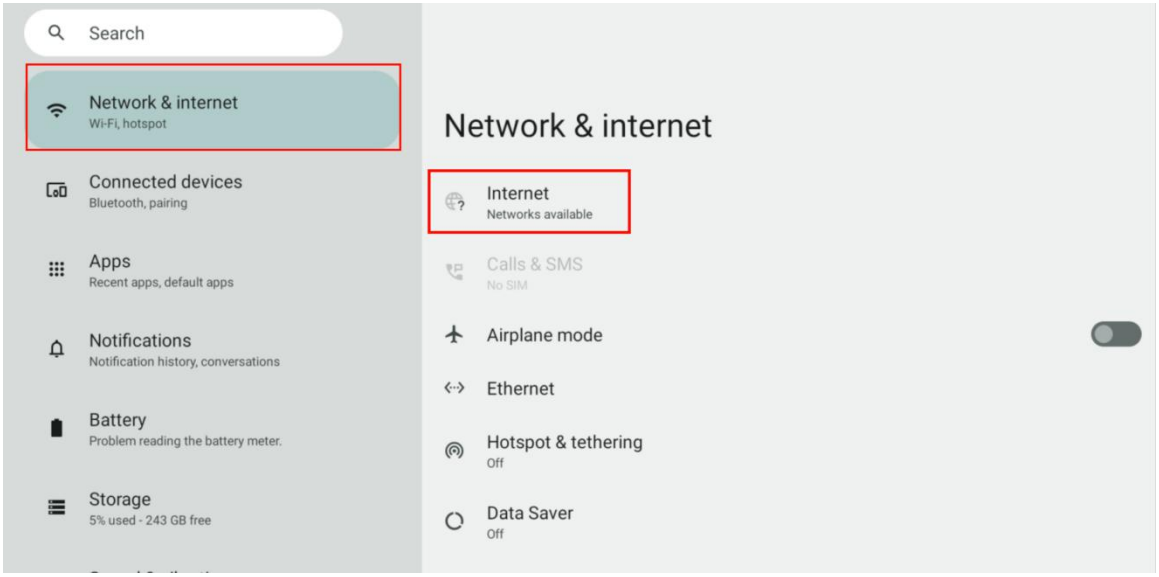
Type-C 转 USB 3.0	OK
Type-C 接口 DP 显示	OK
Type-C 接口 DP 音频	OK
TF 卡启动	OK
识别 NVMe SSD	OK
红外	OK
GPU	OK
NPU	OK
VPU	OK
开关机按键	OK
HDMI CEC 功能	NO

8.3. WIFI 的连接测试方法

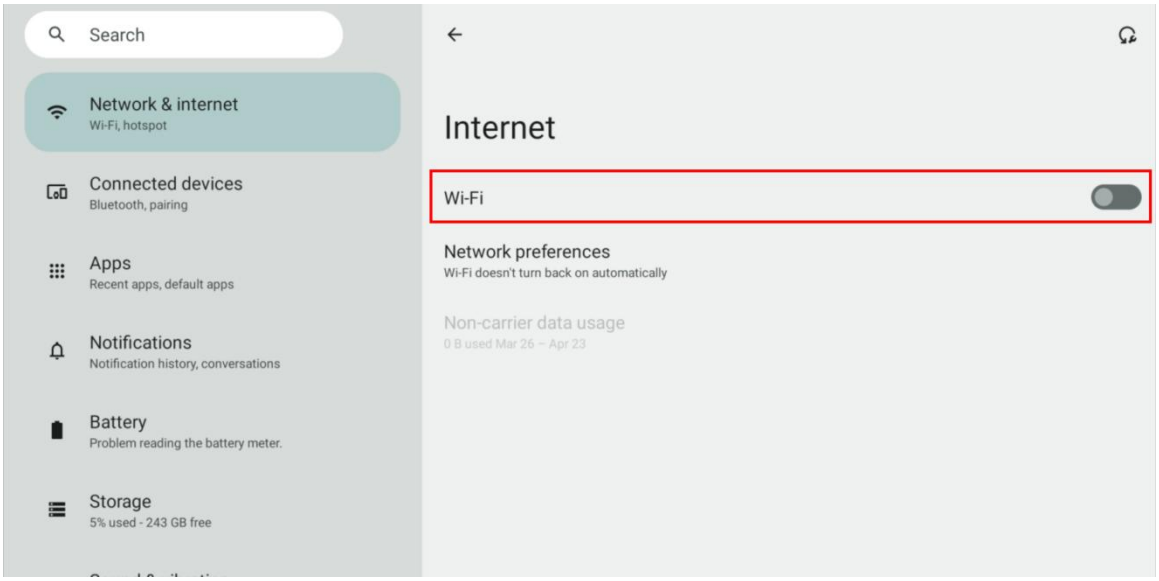
1) 首先点击进入 **Setting**



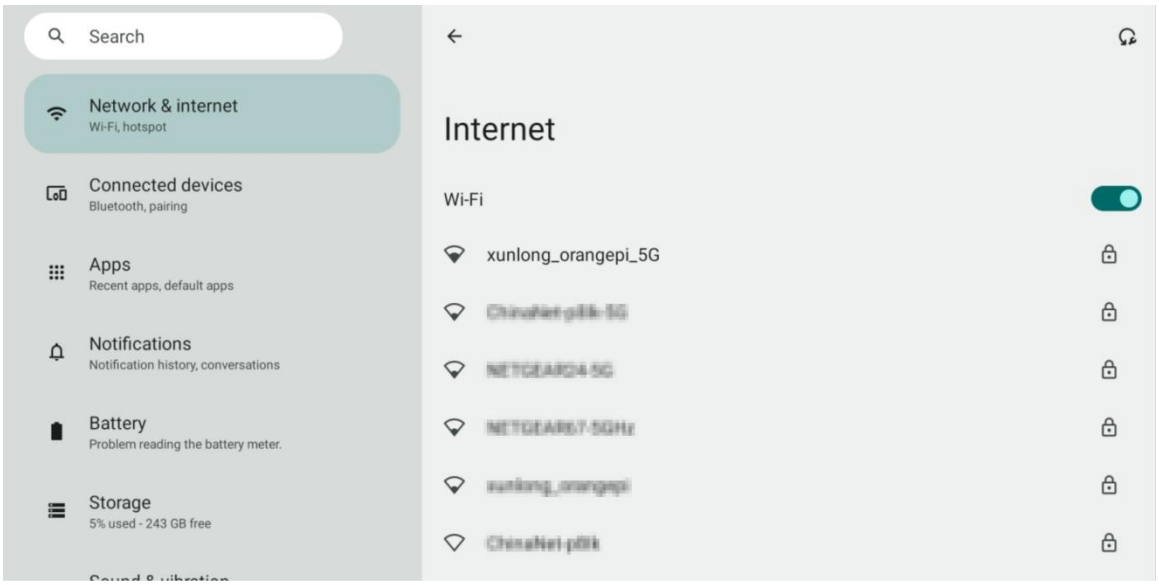
2) 然后选择 **Network & internet** 下面的 **Internet**



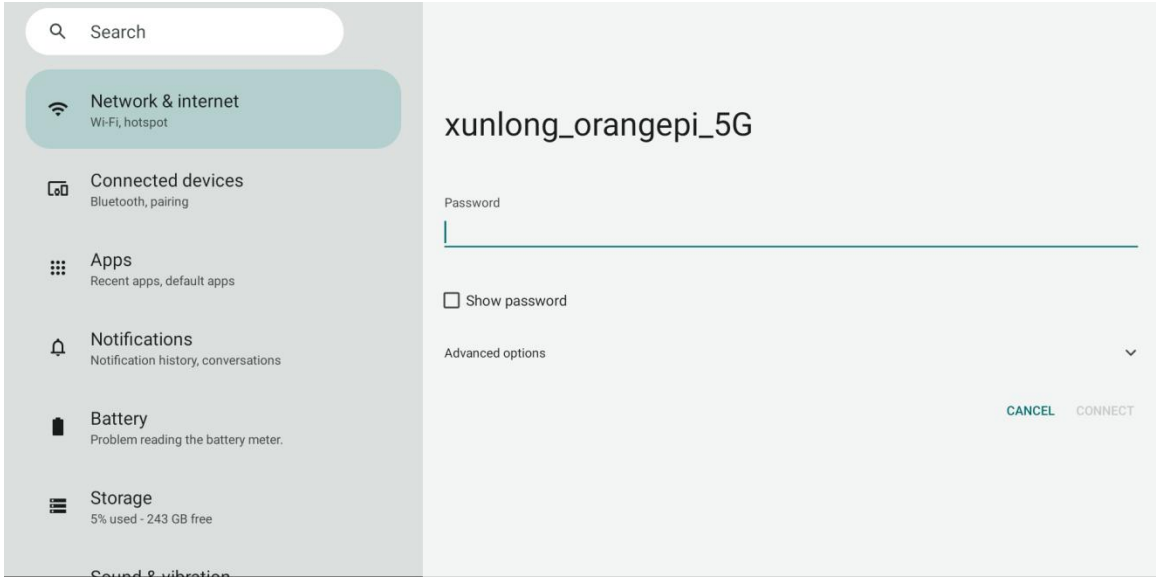
3) 然后打开 **Wi-Fi** 开关



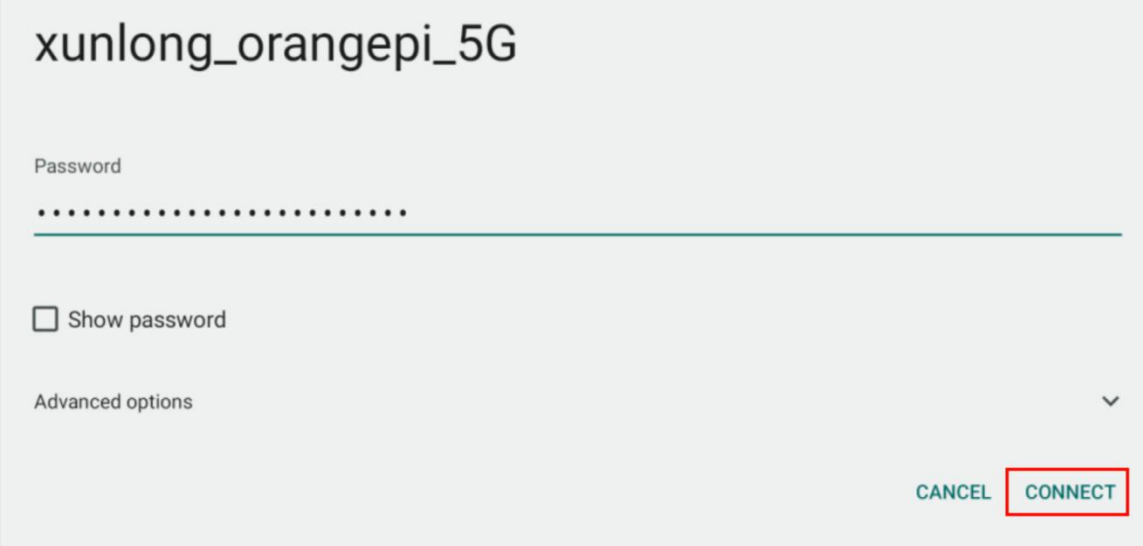
4) 打开 **Wi-Fi** 后如果一切正常，就可以扫描到附近的 Wi-Fi 热点了



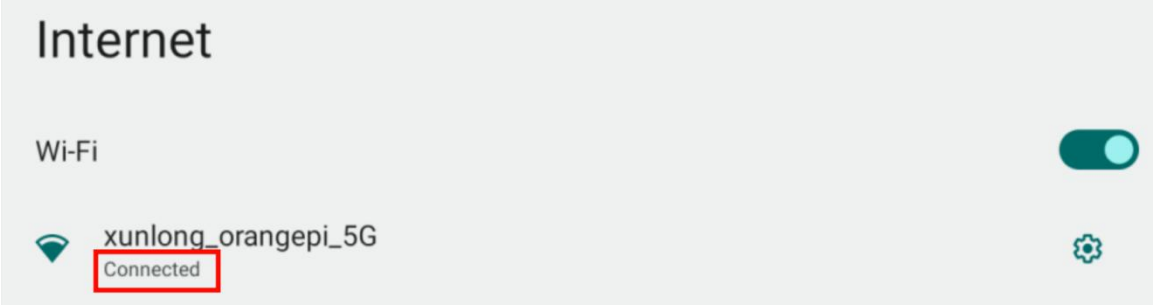
5) 然后选择想连接的 Wi-Fi 后会弹出下图所示的密码输入界面



6) 然后使用键盘输入 Wi-Fi 对应的密码，再使用鼠标点击 **CONNECT** 就会开始连接 Wi-Fi 了

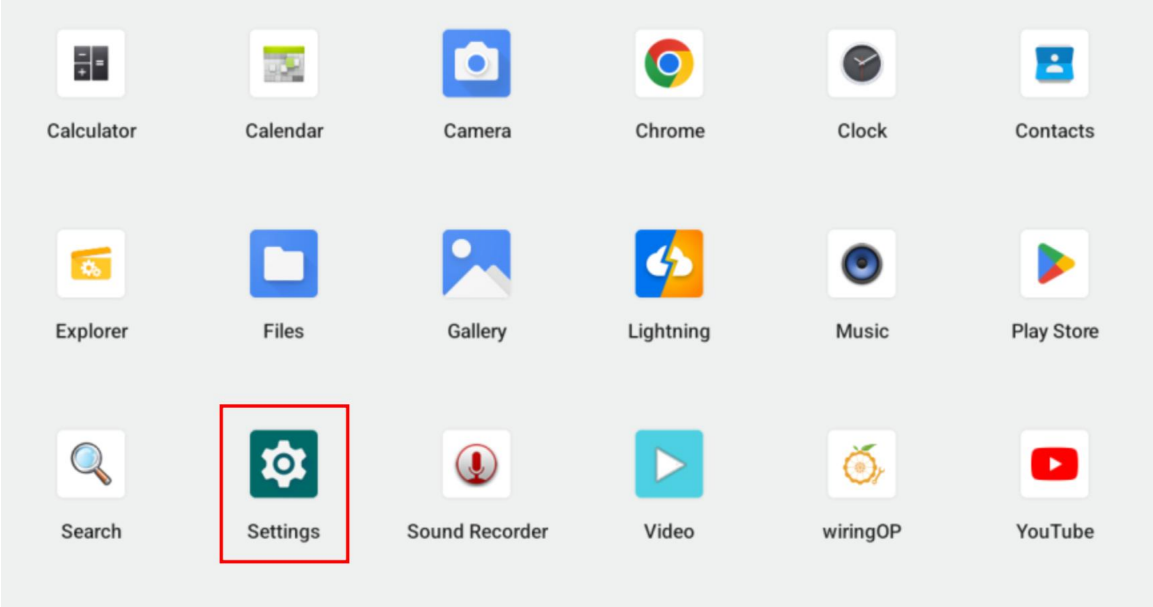


7) Wi-Fi 连接成功后的显示如下图所示：

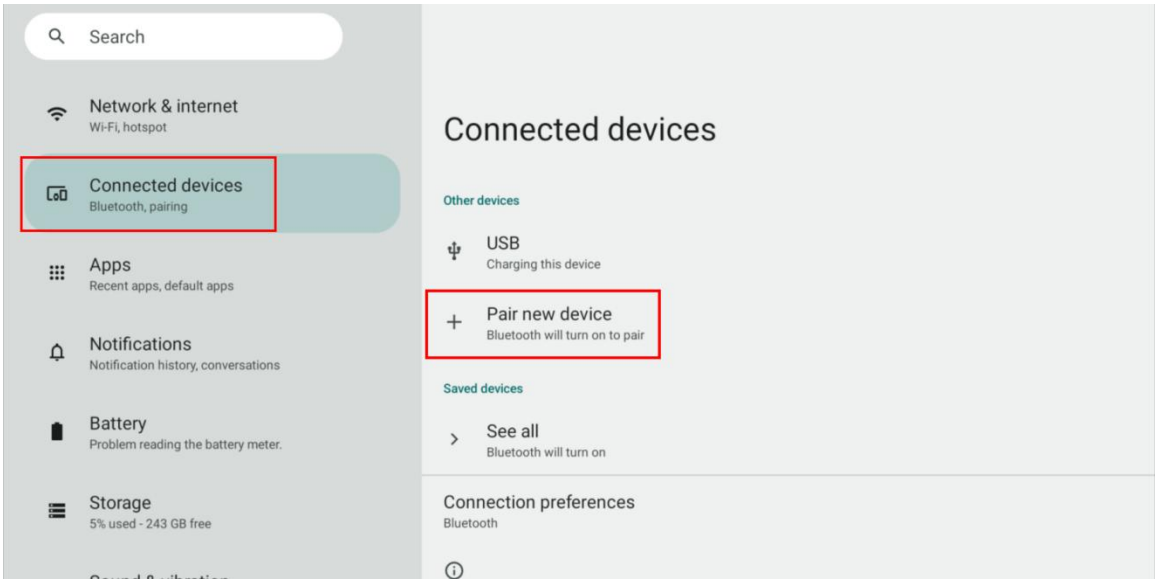


8.4. 蓝牙的测试方法

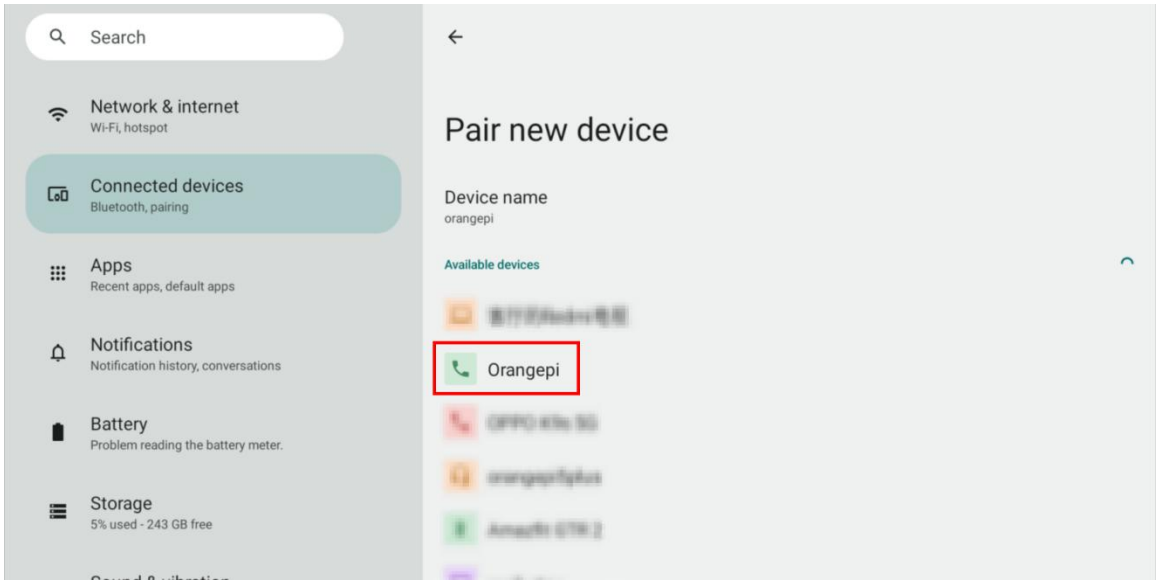
1) 首先点击进入 **Setting**



2) 然后选择 **Connected devices** 菜单下的 **Pair new device** 选项打开蓝牙并开始扫描周围的蓝牙设备



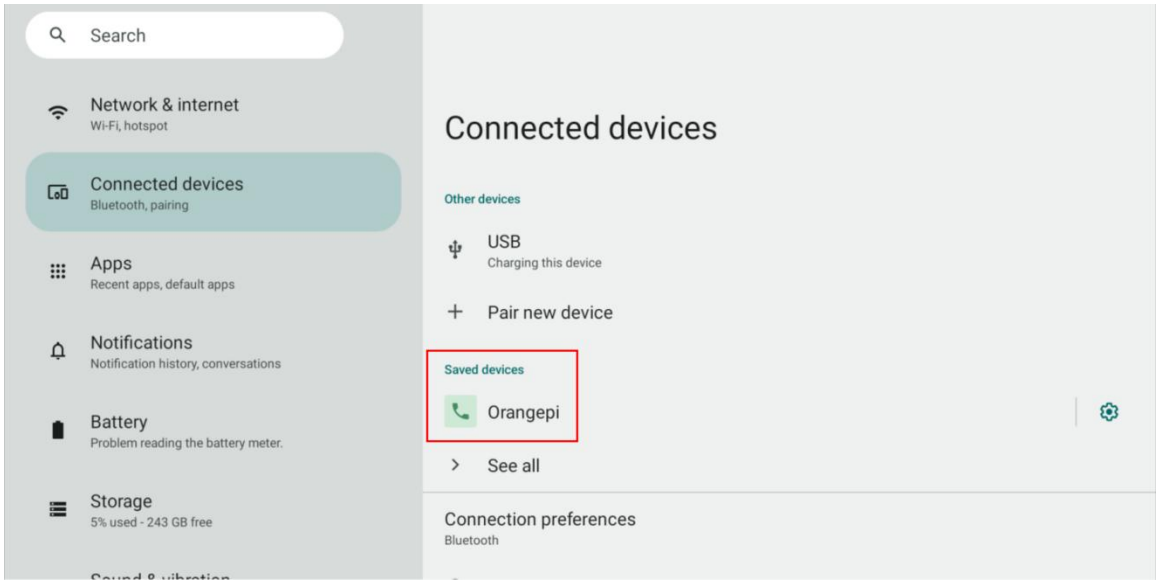
3) 搜索到的蓝牙设备会在 **Available devices** 下面显示出来



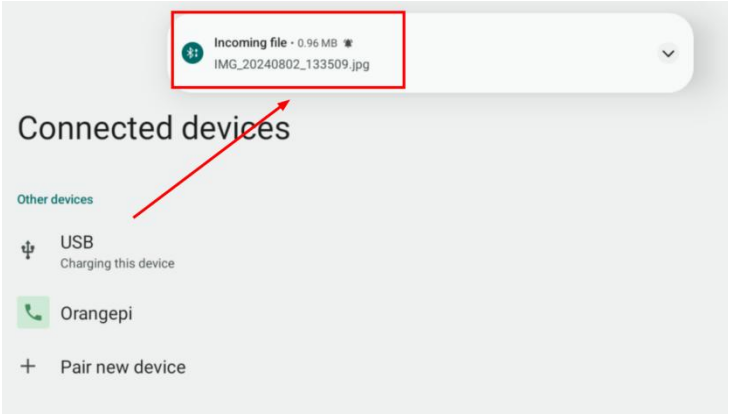
4）然后点击想要连接的蓝牙设备就可以开始配对了，当弹出下面的界面时，请使用鼠标选择 **Pair** 选项

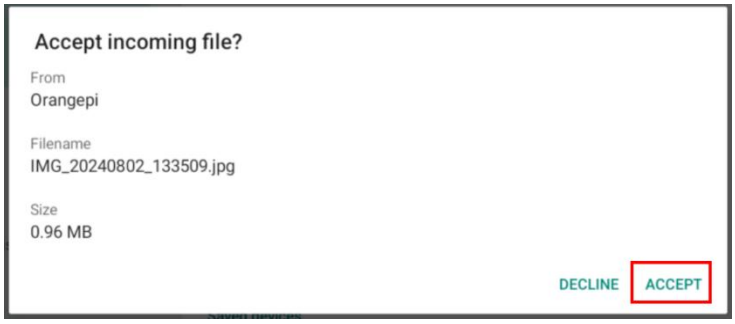
5) 这里测试的是开发板和安卓手机蓝牙的配置过程，此时在手机上会弹出下面的确认界面，在手机上也点击配对按钮后就会开始配对过程

6) 配对完成后，可以看到如下图所示的已配对的蓝牙设备

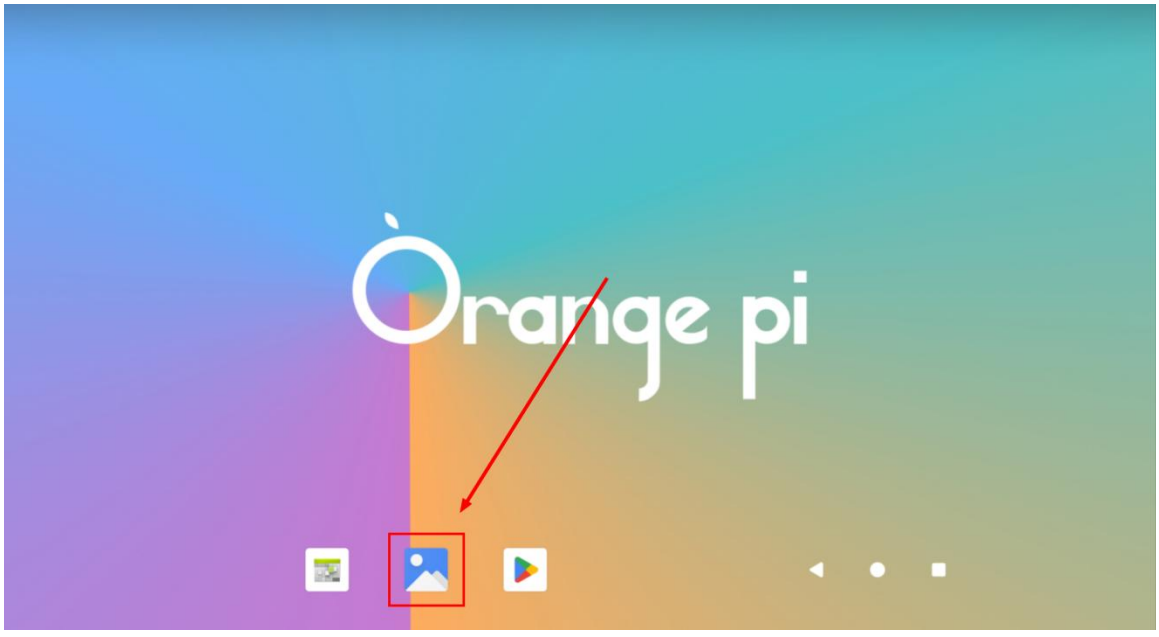


7) 此时可以使用手机蓝牙给开发板发送一张图片，发送后，在开发板的安卓系统中可以看到一条通知信息，点击这条消息，会进入下面的确认界面，然后点击 **Accept** 就可以开始接收手机发过来的图片了





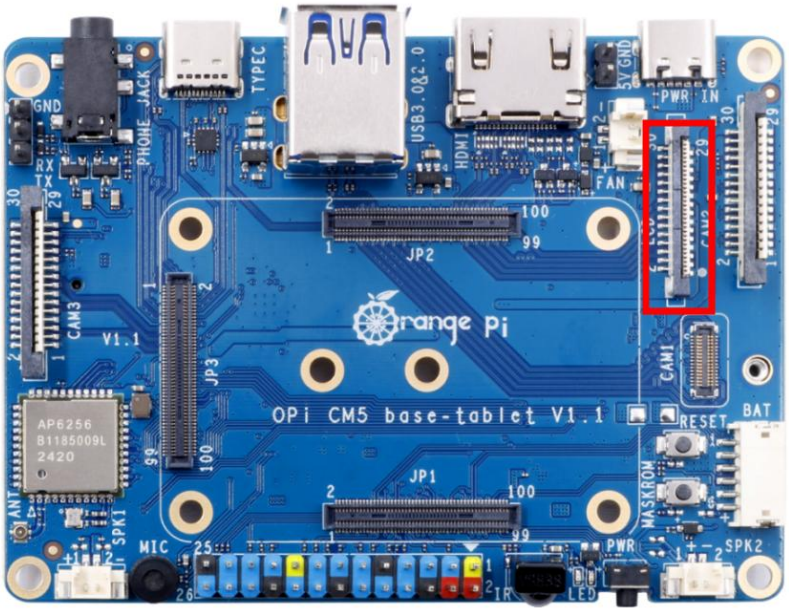
8) 开发板 Android 系统蓝牙接收到的图片可以在系统桌面的图库中打开 **Download** 文件夹查看



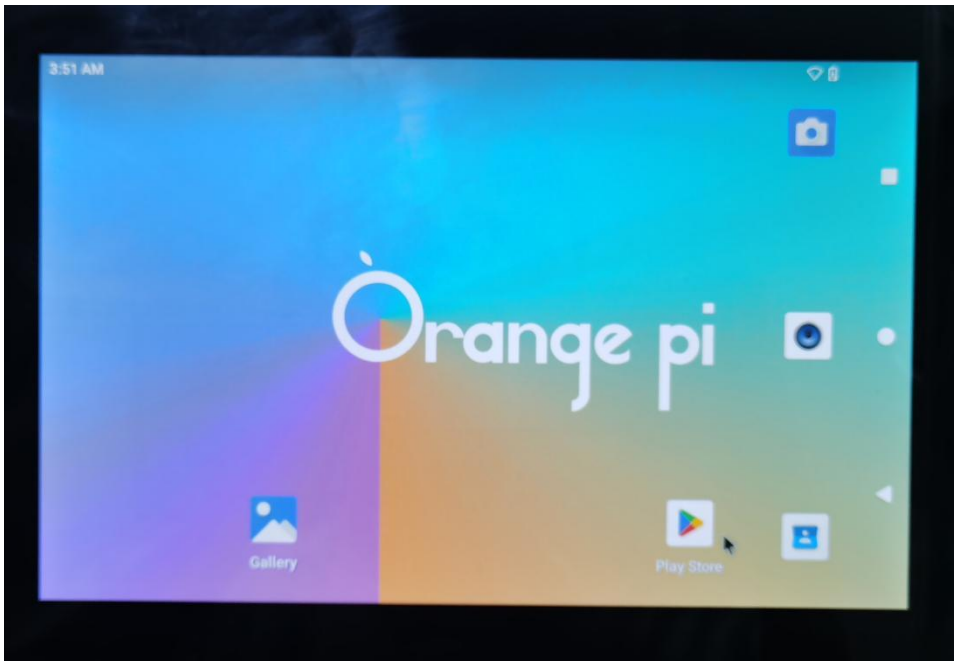
8.5. 10.1 寸 MIPI 屏幕的使用方法

1) 首先需要组装好屏幕，请参考 **10.1 寸 MIPI 屏幕的组装方法**。

2) 开发板上 mipi lcd 屏幕的接口的位置如下图所示：



1) 将组装好的屏幕接到开发板的 LCD 接口，给板子接通 Type-C 电源，并上电，系统启动后，就可以看到屏幕显示如下图所示

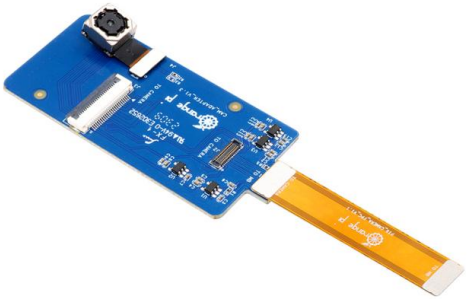


8.6. OV13850 和 OV13855 MIPI 摄像头的测试方法

目前开发板支持两款MIPI摄像头，OV13850 和OV13855，具体的图片如下所示：

a. 1300 万MIPI接口的OV13850 摄像头。

b. 1300 万MIPI接口的OV13855 摄像头。



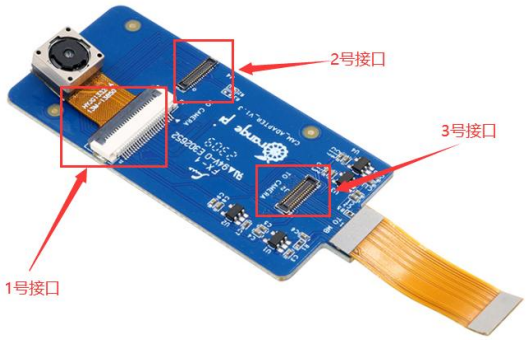
OV13850 和OV13855 摄像头使用的转接板和FPC排线是一样的，只是两款摄像头接在转接板上的位置不一样。FPC排线如下图所示，请注意FPC排线是有方向的，标注**TO MB**那端需要插到开发板的摄像头接口中，标注**TO CAMERA**那端需要插到摄像头转接板上。

摄像头转接板上总共有 3 个摄像头的接口，同一时间只能接一个使用，如下图所示，其中：

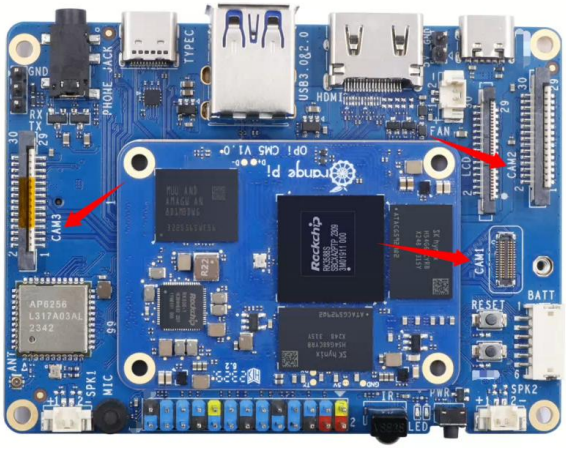
a. **1 号接口接 OV13850 摄像头**。

b. **2 号接口接 OV13855 摄像头**。

c. 3 号接口未使用，忽略即可。



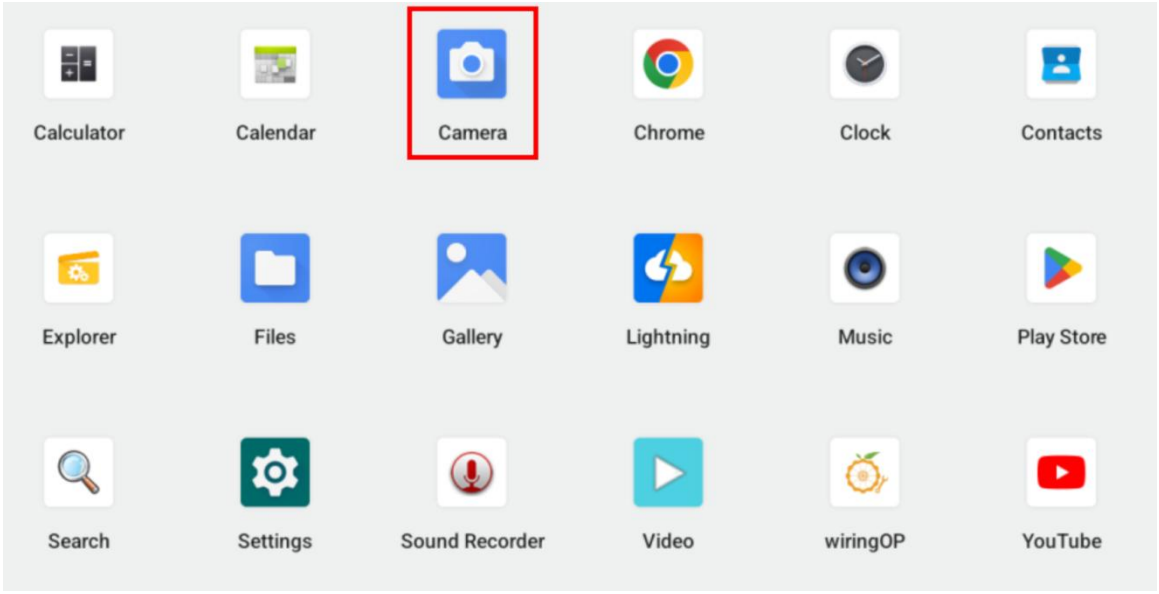
Orange Pi CM5 Base Tablet 开发板上总共有 3 个摄像头接口，只有 CAM1 可以用来接 OV13850 或 OV13855 摄像头。我们定义 Cam1、Cam2 和 Cam3 的位置如下图所示：



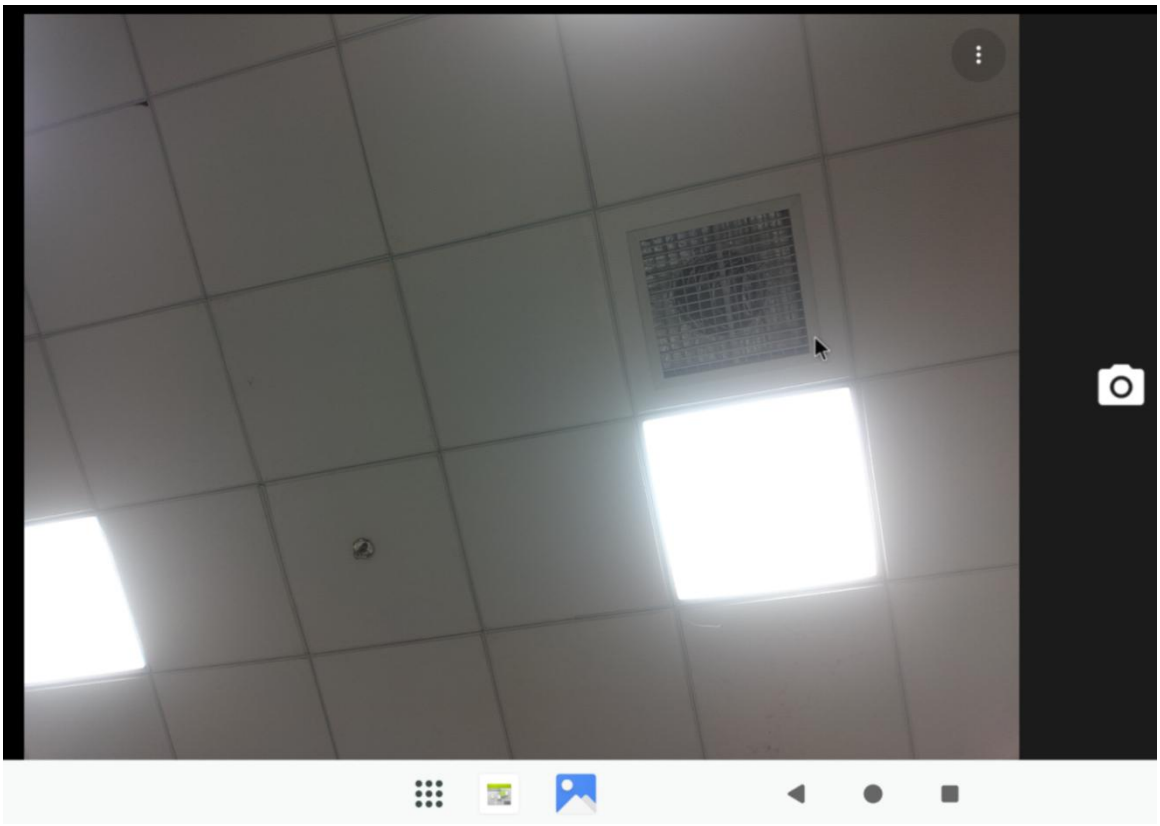
摄像头插在开发板的 Cam1 接口的方法如下所示：

连接好摄像头到开发板上后，我们可以使用下面的方法来测试下摄像头：

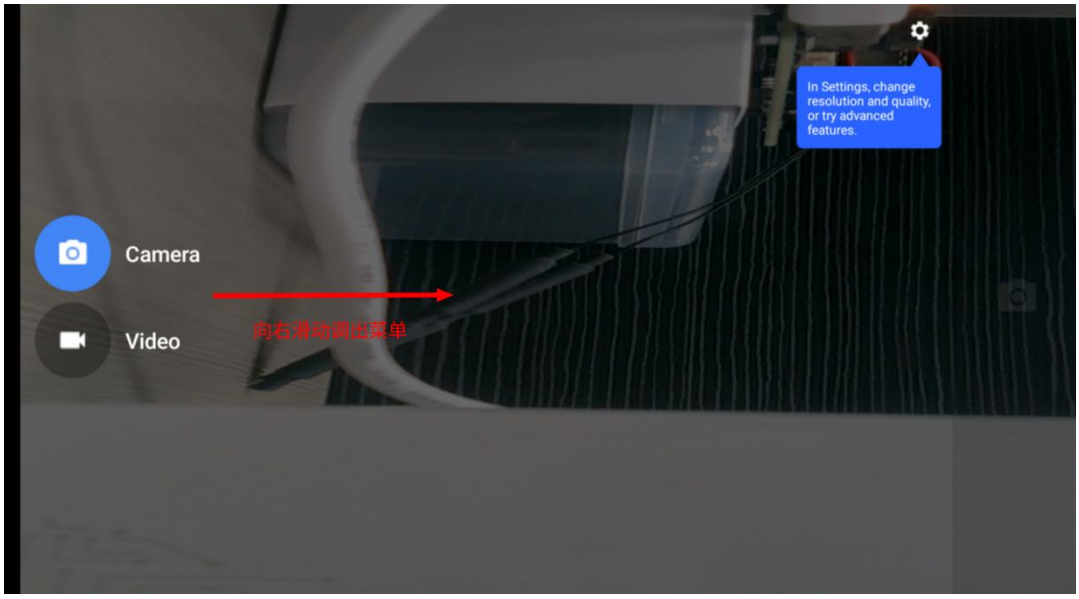
a. 首先打开相机APP



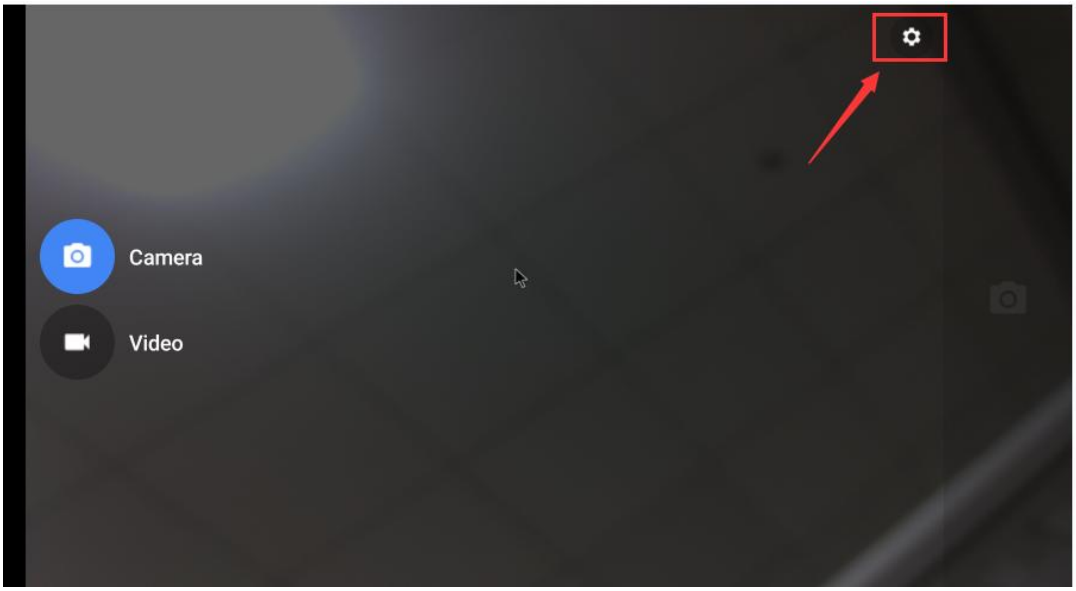
b. 然后就能看到摄像头的预览画面了



在摄像头 APP 下图红框所示的区域中按住鼠标然后向右拖动可以调出拍照和摄像的切换界面。点击 **Video** 即可切换到录像模式



点击下图所示的位置可以进入摄像头的设置界面

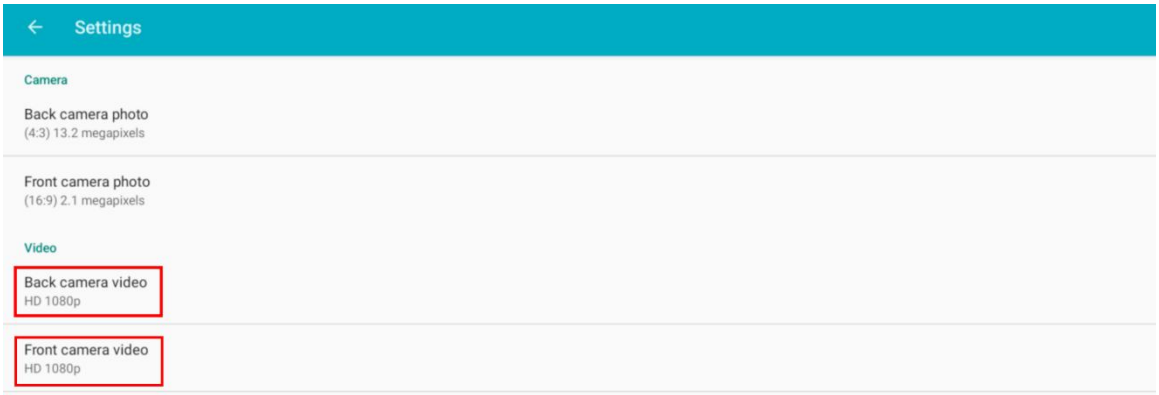


摄像头的设置界面如下所示：

目前测试 OV13850 不支持 4K 录制视频（OV13855 支持），最高只支持 1080p，录制视频时请在设置中将视频格式切换到 1080p，步骤如下所示：

a. 首先进入摄像头 APP 的设置界面，然后点击 **Resolution & quality**

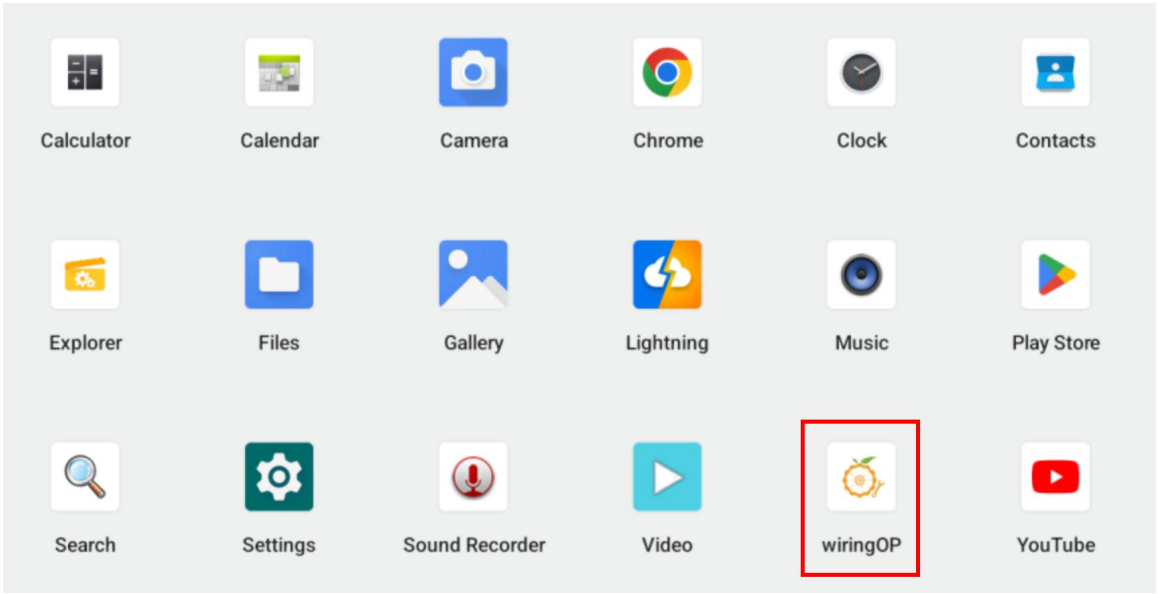
b. 然后在 **Video** 中将视频格式设置为 1080p



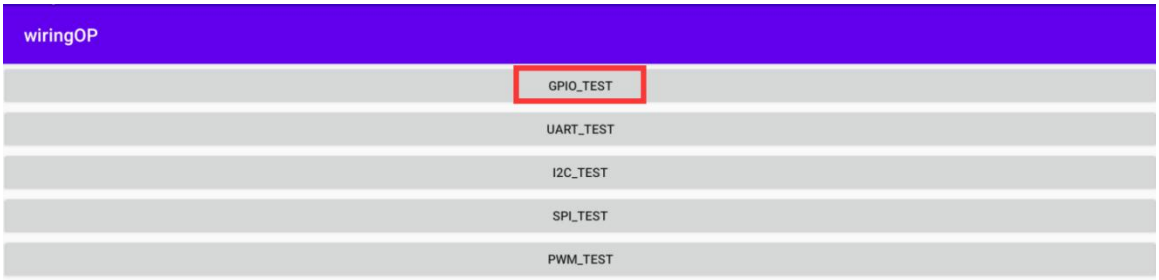
8.7. 26 Pin 接口 GPIO、UART、SPI 和 PWM 测试

8.7.1. 26 Pin GPIO 口测试

1) 首先点击 wiringOP 图标打开 wiringOP APP。

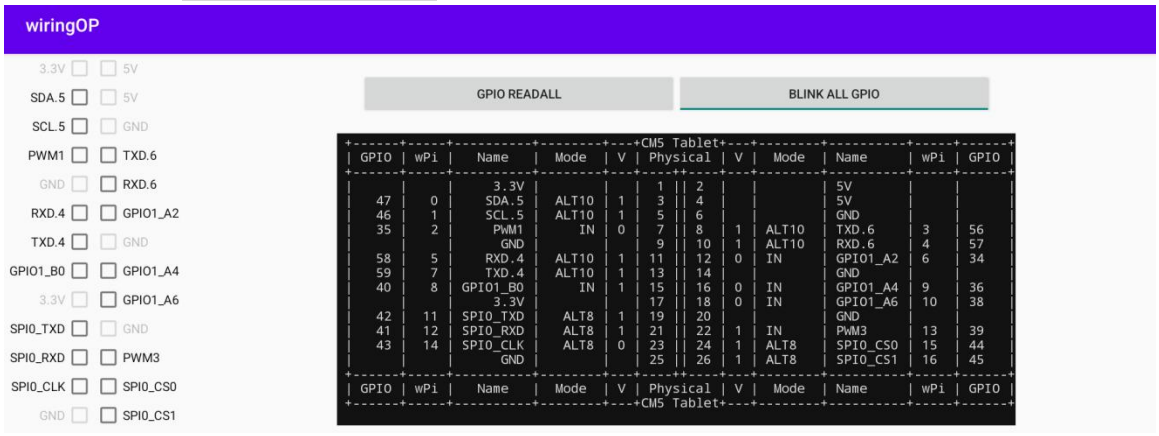


2) wiringOP APP 的主界面显示如下图所示，然后点击 **GPIO_TEST** 按钮打开 GPIO 测试界面。



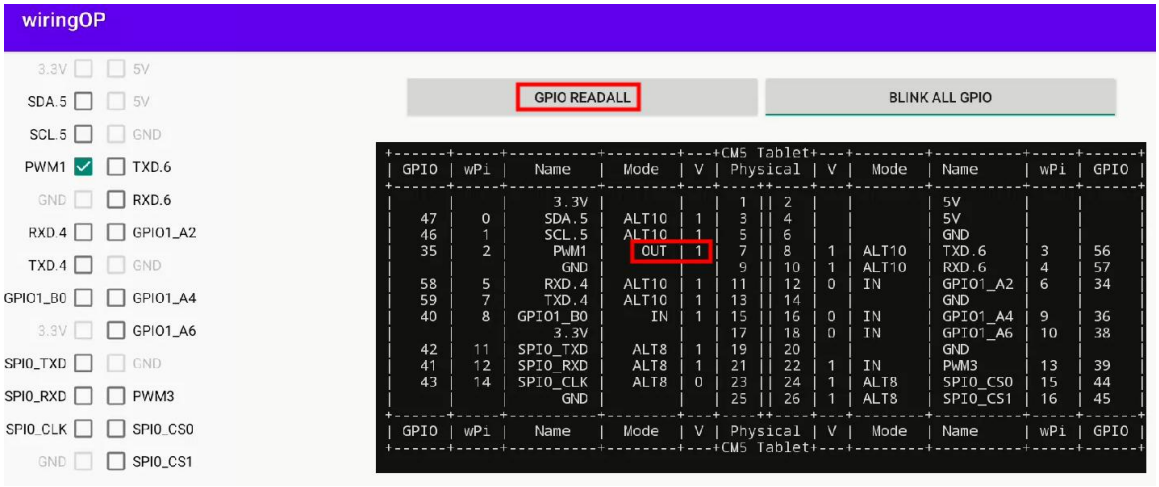
3) GPIO 测试界面如下图所示，左边的两排 **CheckBox** 按钮跟 26 Pin 引脚是一一对应的关系。当勾选 **CheckBox** 按钮时，对应的 GPIO 引脚会被设置为 **OUT** 模式，引脚电平设置为高电平；当取消勾选时，GPIO 引脚电平设置为低电平；当点击右边的 **GPIO READALL** 按钮时，可以获取到 wPi 号、GPIO 模式、引脚电平等信息；当点击 **BLINK ALL GPIO** 按钮时，程序会控制 17 个 GPIO 口不停的切换高低电平。

4) 然后点击 **GPIO READALL** 按钮，输出信息如下图所示：



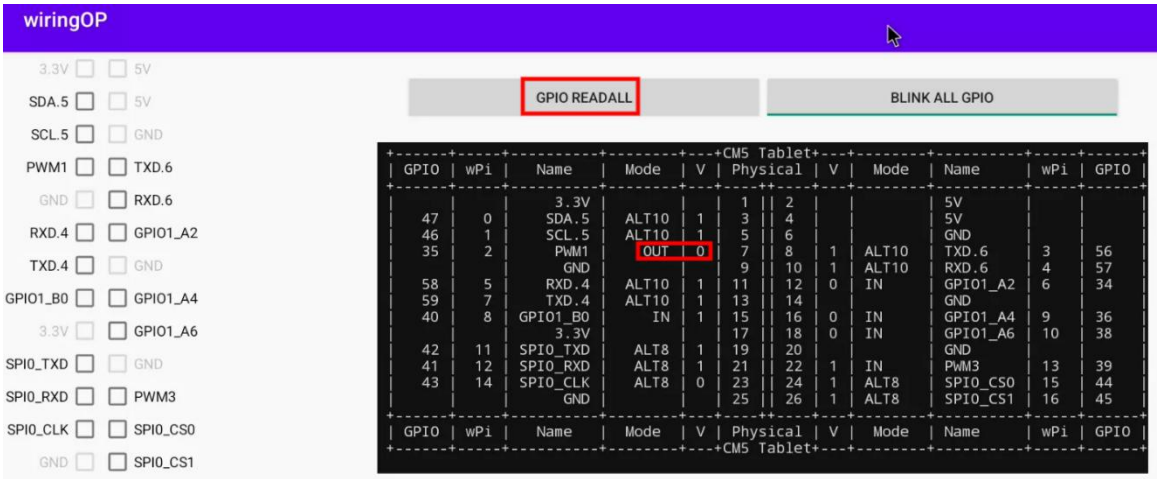
5) 开发板 26 Pin 中总共有 17 个 GPIO 口可以使用，下面以 7 号引脚——对应 GPIO 为 GPIO1_A3 ——对应 wPi 序号为 2——为例演示如何设置 GPIO 口的高低电平。首先点击 7 号引脚对应的 **CheckBox** 按钮，当按钮为选中状态时，7 号引脚会设置为高电平，设置完后可以使用万用表测量引脚的电压的数值，如果为 **3.3v**，说明设置高电平成功。

6) 然后点击 **GPIO READALL** 按钮，可以看到当前的 7 号引脚模式为 **OUT**，引脚电平为高电平。



7) 再次点击下图的 **CheckBox** 按钮取消勾选状态，7 号引脚会设置为低电平，设置完后可以使用万用表测量引脚的电压的数值，如果为 **0v**，说明设置低电平成功。

8) 然后点击 **GPIO READALL** 按钮，可以看到当前的 7 号引脚模式为 OUT，引脚电平为低电平。

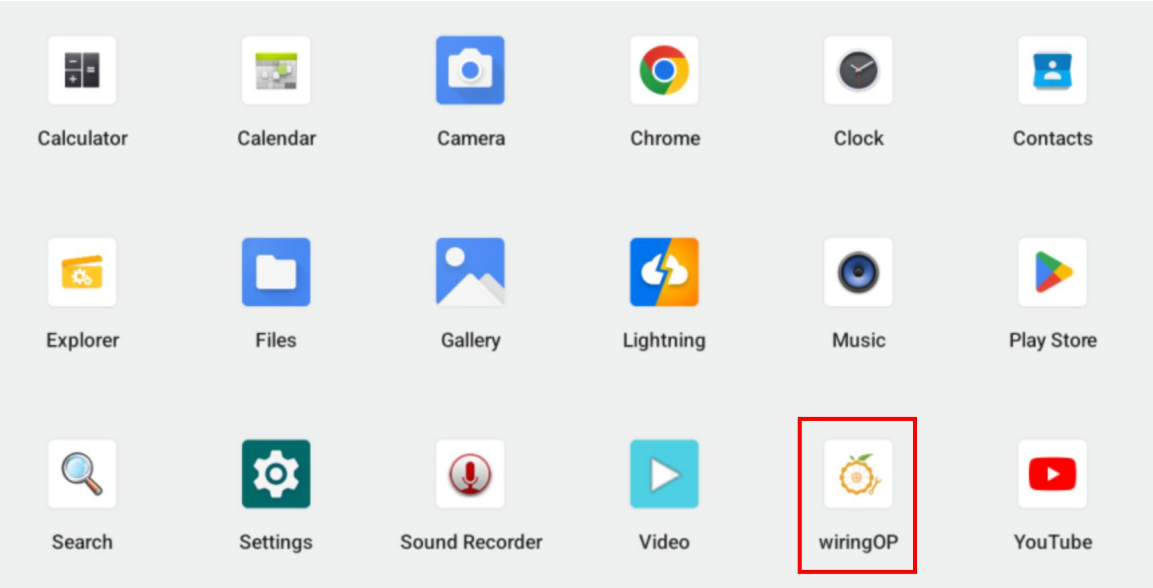


8.7.2. 26 Pin 的 UART 测试

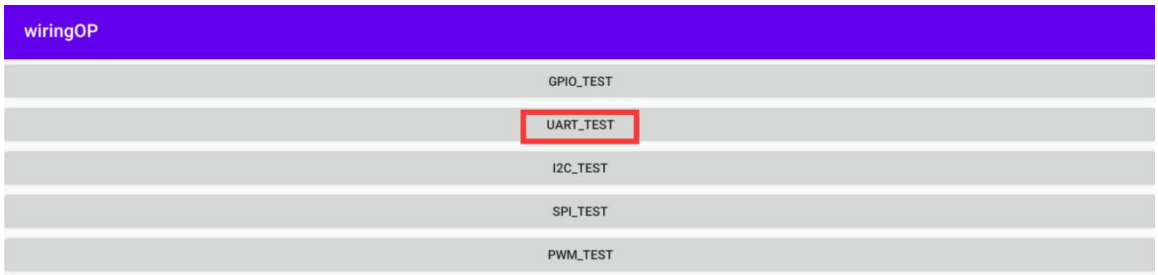
1) Android 中默认打开了 UART4 和 UART6 两个串口，在 26 Pin 的位置如下图所示，对应的设备节点是**/dev/ttyS4** 和**/dev/ttyS6**。

复用功能	GPIO	GPIO 序号	引脚序号	引脚序号	GPIO 序号	GPIO	复用功能
	3.3V		1	2		5V	
I2C5_SDA_M3	GPIO1_B7	47	3	4		5V	
I2C5_SCL_M3	GPIO1_B6	46	5	6		GND	
PWM1_M2	GPIO1_A3	35	7	8	56	GPIO1_D0	UART6_TX_M2
	GND		9	10	57	GPIO1_D1	UART6_RX_M2
UART4_TX_M0	GPIO1_D2	58	11	12	34	GPIO1_A2	
UART4_RX_M0	GPIO1_D3	59	13	14		GND	
	GPIO1_B0	40	15	16	36	GPIO1_A4	
	3.3V		17	18	38	GPIO1_A6	
SPI0_MOSI_M2	GPIO1_B2	42	19	20		GND	
SPI0_MISO_M2	GPIO1_B1	41	21	22	39	GPIO1_A7	PWM3_M3
SPI0_CLK_M2	GPIO1_B3	43	23	24	44	GPIO1_B4	SPI0_CS0_M2
	GND		25	26	45	GPIO1_B5	SPI0_CS1_M2

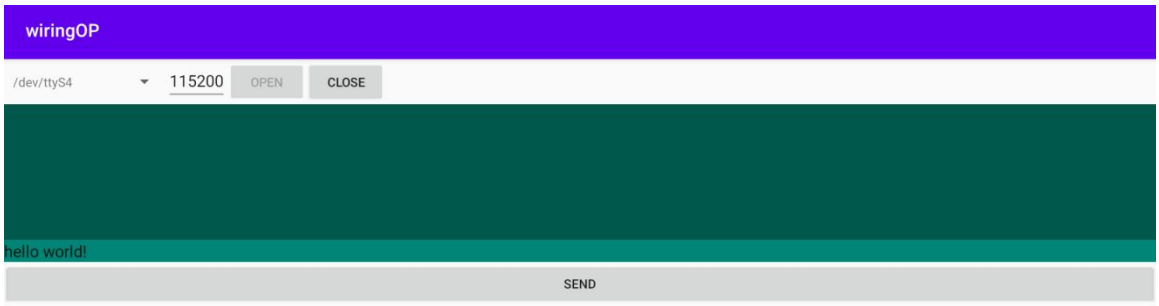
2) 首先点击 wiringOP 图标打开 wiringOP APP。



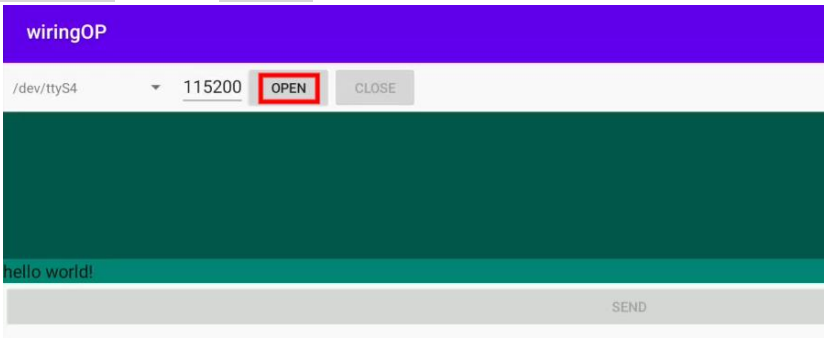
3) wiringOP APP 的主界面显示如下图所示，然后点击 **UART_TEST** 按钮打开 UART 测试界面。



4) APP 的串口测试界面如下图所示：

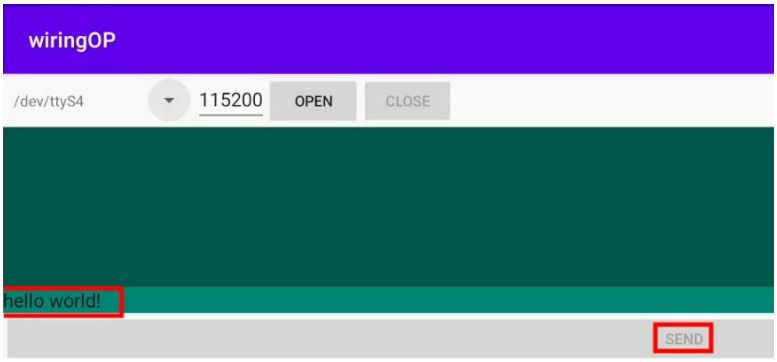


5) 接着选择想要测试的串口，比如**/dev/ttyS4**，然后在编辑框中输入想要设置的波特率，再点击 **OPEN** 按钮打开**/dev/ttyS4** 节点，打开成功后，**OPEN** 按钮变为不可选中状态，**CLOSE** 按钮和 **SEND** 按钮变为可选中状态。

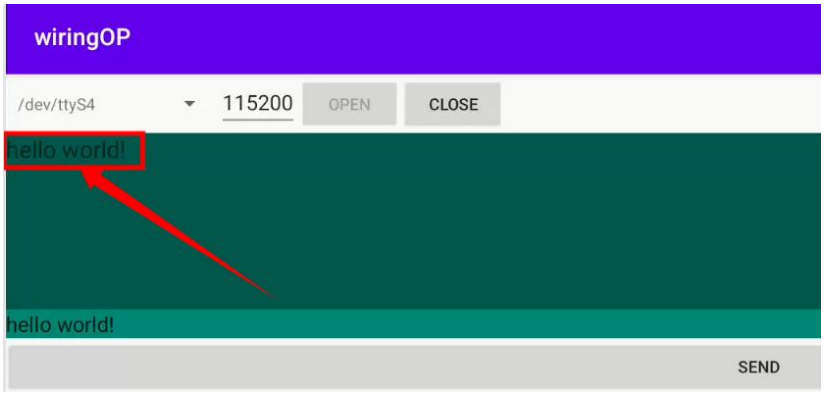


6) 然后使用杜邦线短接 uart4 的 RXD 和 TXD 引脚。

7) 然后可以在下面的发送编辑框中输入一段字符，点击 **SEND** 按钮开始发送。



8) 如果一切正常，接收框内会显示已接收到的字符串。



8.7.3. 26 Pin 的 SPI 测试

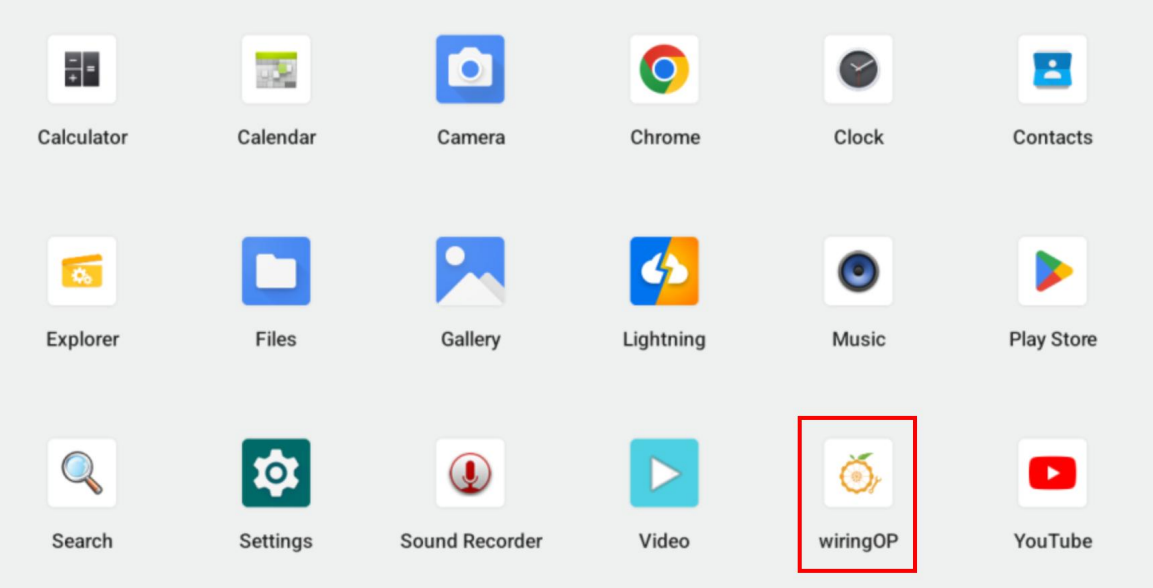
1) 由下表可知，开发板可用的 spi 为 spi0。

复用功能	GPIO	GPIO 序号	引脚序号	引脚序号	GPIO 序号	GPIO	复用功能
	3.3V		1	2		5V	
I2C5_SDA_M3	GPIO1_B7	47	3	4		5V	
I2C5_SCL_M3	GPIO1_B6	46	5	6		GND	
PWM1_M2	GPIO1_A3	35	7	8	56	GPIO1_D0	UART6_TX_M2
	GND		9	10	57	GPIO1_D1	UART6_RX_M2
UART4_TX_M0	GPIO1_D2	58	11	12	34	GPIO1_A2	
UART4_RX_M0	GPIO1_D3	59	13	14		GND	
	GPIO1_B0	40	15	16	36	GPIO1_A4	
	3.3V		17	18	38	GPIO1_A6	
SPI0_MOSI_M2	GPIO1_B2	42	19	20		GND	
SPI0_MISO_M2	GPIO1_B1	41	21	22	39	GPIO1_A7	PWM3_M3

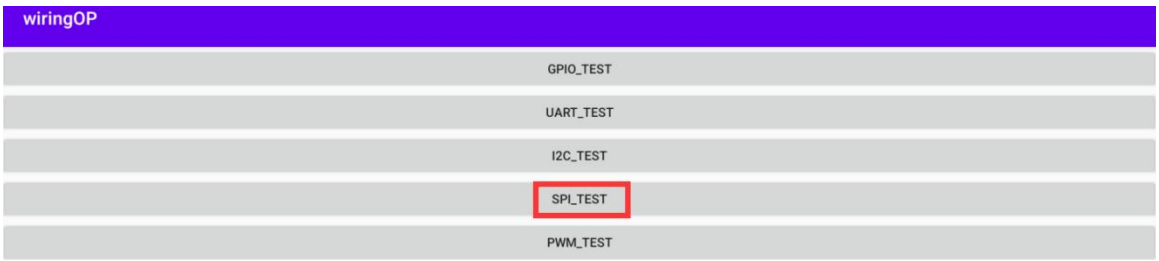
SPI0_CLK_M2	GPIO1_B3	43	23	24	44	GPIO1_B4	SPI0_CS0_M2
	GND		25	26	45	GPIO1_B5	SPI0_CS1_M2

2) 这里通过 w25q64 模块来测试 SPI 接口，首先在 SPI0 接口接入 w25q64 设备。

3) 然后点击 wiringOP 图标打开 wiringOP APP。

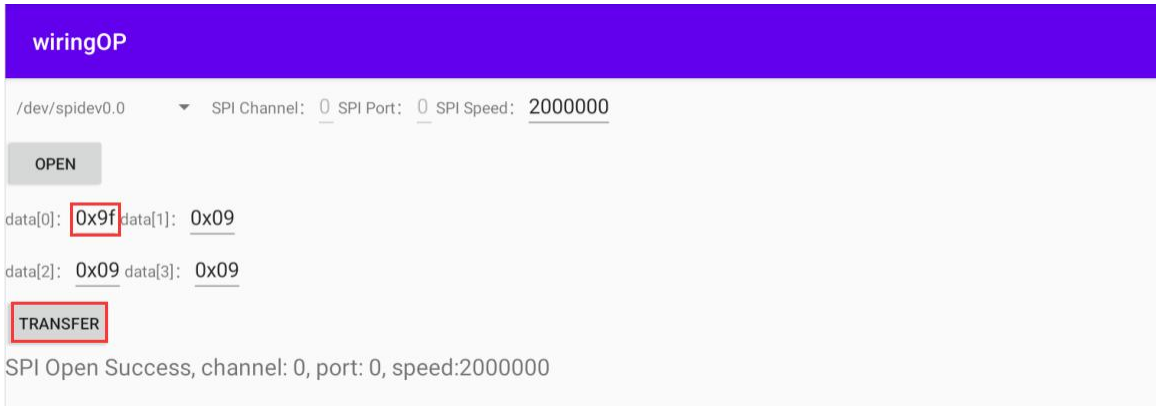


4) wiringOP APP 的主界面显示如下图所示，点击 SPI_TEST 按钮打开 SPI 的测试界面。

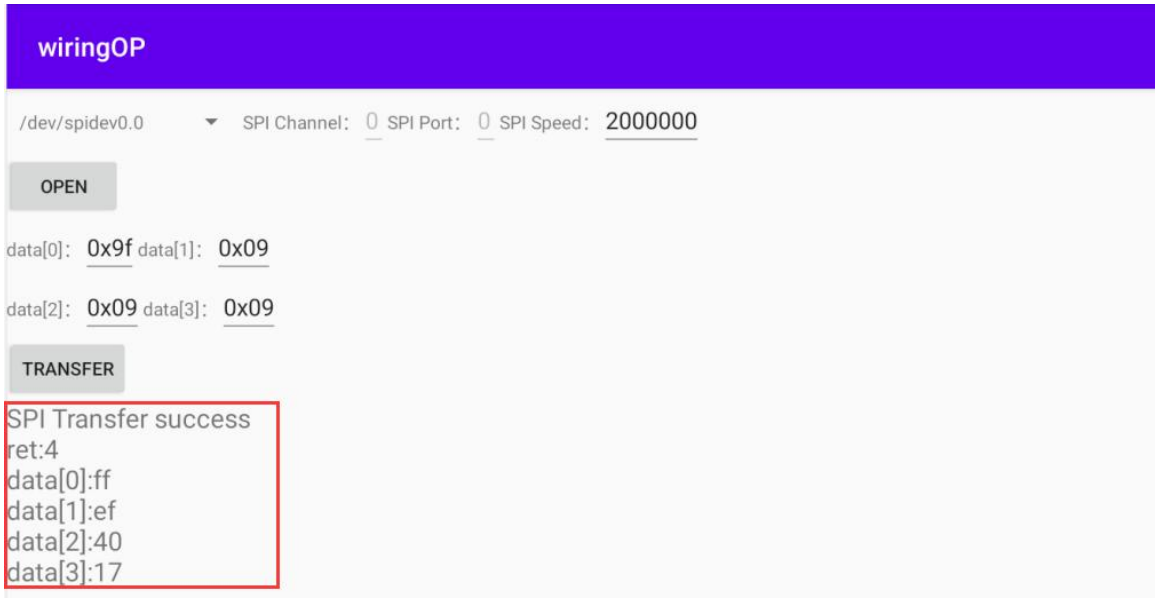


5) 然后点击 **OPEN** 按钮初始化 SPI。

6) 然后填充需要发送的字节，比如读取 w25q64 的 ID 信息，在 data[0]中填入地址 0x9f，然后点击 **TRANSFER** 按钮。



7) 最后 APP 会显示读取到的 ID 信息。



8) w25q64 模块的 MANUFACTURER ID 为 EFh，Device ID 为 4017h，跟上面读取

到的值是对应的（h 代表是 16 进制）

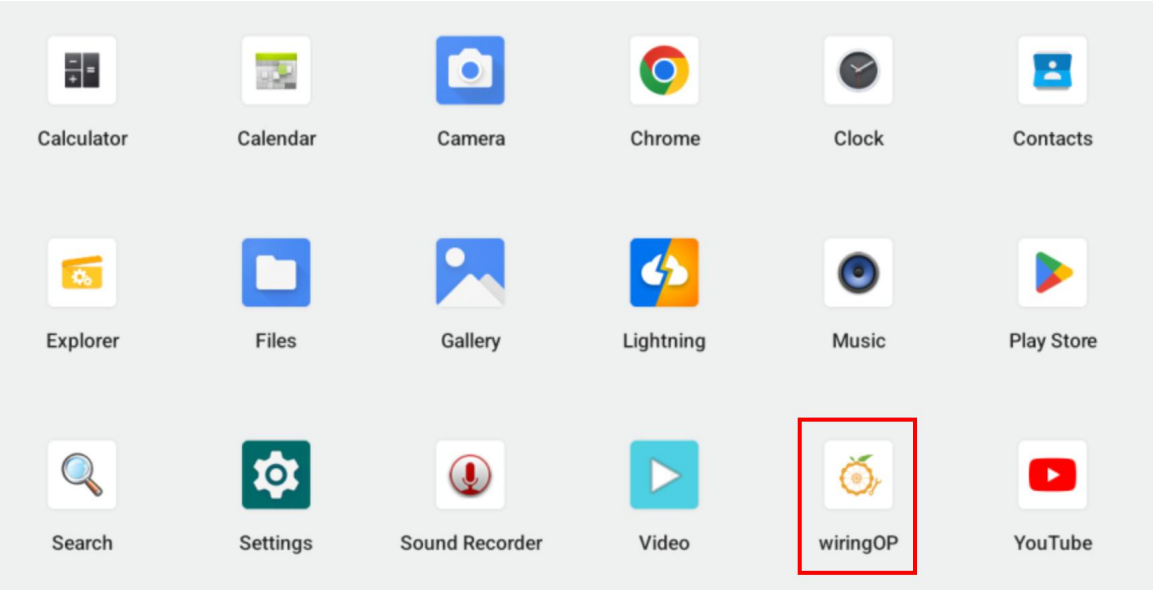
MANUFACTURER ID	(MF7 - MF0)	
Winbond Serial Flash	EFh	
Device ID	**(ID7 - ID0)**	**(ID15 - ID0)**
Instruction	**ABh, 90h, 92h, 94h**	**9Fh**
W25Q64FV (SPI)	16h	4017h
W25Q64FV (QPI)	16h	6017h

8. 7. 4. 26 Pin 的 PWM 测试

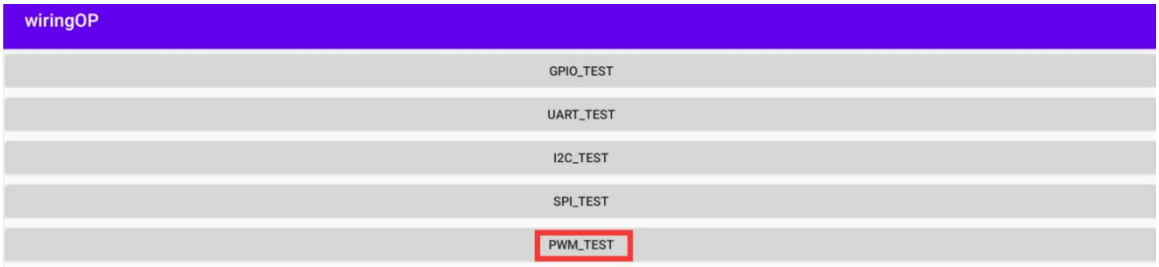
1) Android 默认开启了 **PWM1**、和 **PWM3**，对应的引脚在 26 Pin 的所在位置如下表所示：

复用功能	GPIO 序号	引脚序号	引脚序号	GPIO 序号	GPIO	复用功能
		1	2		5V	
I2C5_SDA_M3	47	3	4		5V	
I2C5_SCL_M3	46	5	6		GND	
PWM1_M2	35	7	8	56	GPIO1_D0	UART6_TX_M2
		9	10	57	GPIO1_D1	UART6_RX_M2
UART4_TX_M0	58	11	12	34	GPIO1_A2	
UART4_RX_M0	59	13	14		GND	
	40	15	16	36	GPIO1_A4	
		17	18	38	GPIO1_A6	
SPI0_MOSI_M2	42	19	20		GND	
SPI0_MISO_M2	41	21	22	39	GPIO1_A7	PWM3_M3
SPI0_CLK_M2	43	23	24	44	GPIO1_B4	SPI0_CS0_M2
		25	26	45	GPIO1_B5	SPI0_CS1_M2

2) 首先点击 wiringOP 图标打开 wiringOP APP。



3) 然后在 wiringOP 的主界面点击 **PWM_TEST** 按钮进入 PWM 的测试界面。

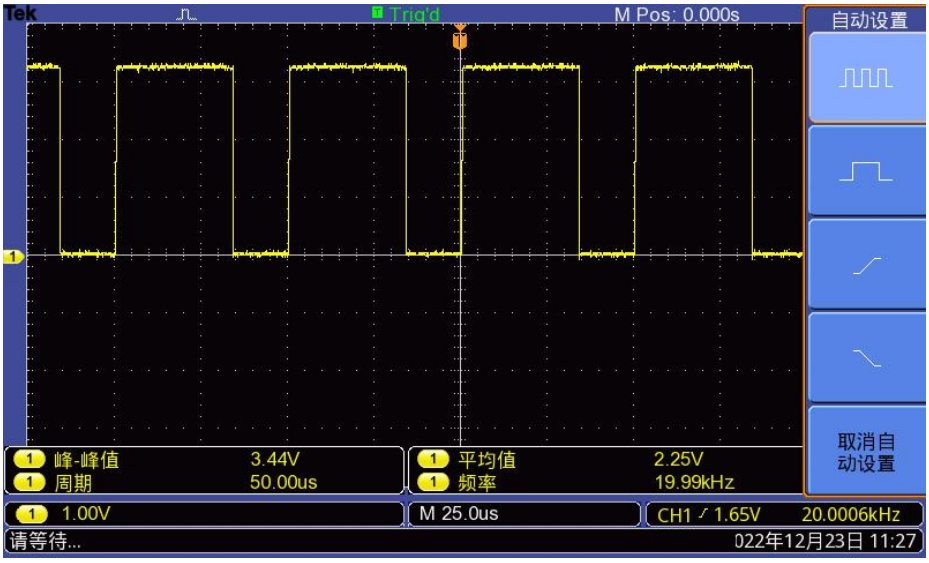


4) PWM1 的基地址 **fd8b0010**，PWM3 的基地址是 **fd8b0030**，这里 pwmchip0 右边显示的是 **fd8b0010.pwm**，说明默认选择的是 PWM1。如果想选择 PWM3，需要点击下拉选项选择其它的 pwmchip，当右边显示为 **fd8b0030.pwm** 时，说明选中的是 **PWM3**。

5) 然后确认 PWM 的通道，默认是 0 通道，并确认 PWM 的周期，默认的配置是 **50000ns**，转换为 PWM 频率是 **20KHz**，可自行修改，点击 **EXPORT** 按钮导出 **PWM1**。

6) 然后拖动下面的拖动条，就可以改变 PWM 的占空比，然后勾选 Enable 就可以输出 PWM 波形了。

7) 然后使用示波器测量开发板 26 Pin 中的第 7 号引脚就可以看到下面的波形了。

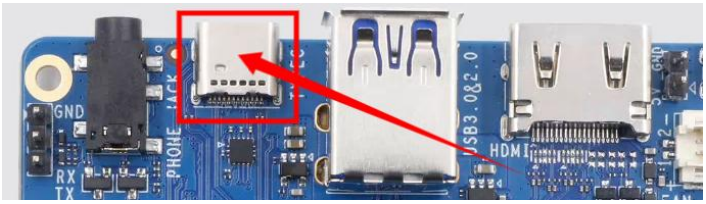


8.8. ADB 的使用方法

8.8.1. 使用数据线连接 adb 调试

1) 首先准备一根品质良好的 Type-C 数据线。

2) 然后通过 Type-C 数据线连接好开发板与 Ubuntu PC，开发板 Type-C 接口的位置如下图所示：



3) 然后在 Ubuntu PC 上安装 adb 工具。

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

4) 通过下面的命令可以查看识别到的 ADB 设备。

```
test@test:~$ adb devices
List of devices attached
S63QCF54CJ      device
test@test:~$ lsusb
Bus 003 Device 006: ID 2207:0006
```

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

```
test@test:~$ adb shell
console:/ $
```

6) 执行下面的命令可以重新挂载 Android 系统。

```
test@test:~$ adb root
test@test:~$ adb remount
```

7) 然后就可以传输文件到 Android 系统了。

```
test@test:~$ adb push example.txt /system/
```

8.8.2. 使用网络连接 adb 调试

使用网络 adb 无需 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无线网络已经连接好了，然后获取开发板的 IP 地址，后面要用到。

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

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

2) 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。

```
console:/ # setprop service.adb.tcp.port 5555
console:/ # stop adbd
console:/ # start adbd
```

3) 在 Ubuntu PC 上安装 adb 工具。

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

4) 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
```

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

```
test@test:~$ adb shell
console:/ #
```

9. 附录

9.1. 用户手册更新历史

版本	日期	更新说明
v1.0	2024-08-20	初始版本

9.2. 镜像更新历史

日期	更新说明
2024-08-20	Orangepicm5-tablet_1.0.0_ubuntu_jammy_server_linux6.1.43.7z Orangepicm5-tablet_1.0.0_ubuntu_focal_server_linux5.10.160.7z Orangepicm5-tablet_1.0.0_ubuntu_jammy_server_linux5.10.160.7z Orangepicm5-tablet_1.0.0_debian_bullseye_server_linux5.10.160.7z Orangepicm5-tablet_1.0.0_debian_bookworm_server_linux6.1.43.7z Orangepicm5-tablet_1.0.0_debian_bookworm_server_linux5.10.160.7z Orangepicm5-tablet_1.0.0_ubuntu_jammy_desktop_xfce_linux6.1.43.7z Orangepicm5-tablet_1.0.0_ubuntu_focal_desktop_xfce_linux5.10.160.7z Orangepicm5-tablet_1.0.0_ubuntu_jammy_desktop_xfce_linux5.10.160.7z Orangepicm5-tablet_1.0.0_debian_bullseye_desktop_xfce_linux5.10.160.7z Orangepicm5-tablet_1.0.0_debian_bookworm_desktop_xfce_linux6.1.43.7z Orangepicm5-tablet_1.0.0_debian_bookworm_desktop_xfce_linux5.10.160.7z Opios-droid-aarch64-opicm5-tablet-24.11-linux5.10.160-en.tar.gz Opios-droid-aarch64-opicm5-tablet-24.11-linux5.10.160.tar.gz OrangePiCM5-TABLET_RK3588S_Android13_v1.0.0.tar.gz Opios-arch-aarch64-gnome-opicm5_tablet-24.09-linux5.10.160.img.xz * 初始版本