Orange Pi 5B
用户手册

目录

1. Orange Pi 5B 的基本特性

1.1. 什么是 Orange Pi 5B

Orange Pi 5B 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（LPDDR4/4x）内存和 32GB/64GB/128GB/256GB 板载 eMMC，具有高达 8K 显示处理能力。

Orange Pi 5B 引出了相当丰富的接口，包括 HDMI 输出、Type-C、WIFI6、蓝牙、千兆网口、USB2.0、USB3.0 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 5B 支持 Orange Pi 官方研发的操作系统 Orange Pi OS，同时，支持 Android 12.1、Debian11、Ubuntu20.04 和 Ubuntu22.04 等操作系统。

1.2. Orange Pi 5B 的用途

我们可以用它实现：

- 一台 Linux 桌面计算机
- 一台 Linux 网络服务器
- Android 平板
- Android 游戏机等

当然还有其他更多的功能，因为 Orange Pi 5B 开发板可以安装 Debian 和 Ubuntu 这样的 Linux 系统，以及 Android 这样的系统，也就意味着我们可以在开发板硬件和软件支持的范围内，来实现各种各样的功能。

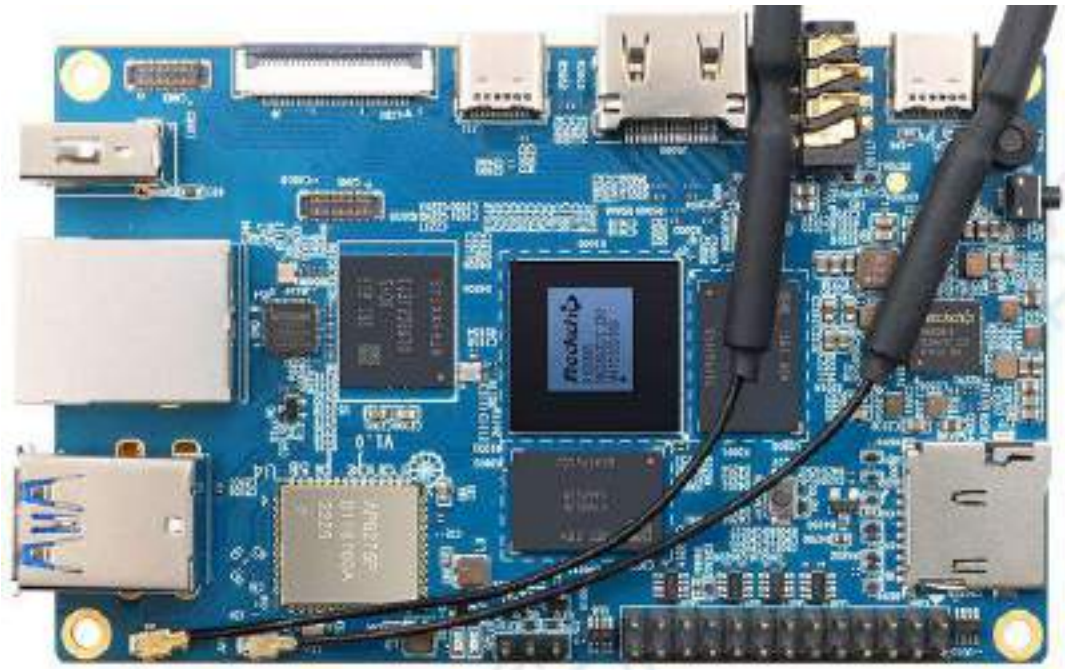
1.3. Orange Pi 5B 的硬件特性

硬件特性介绍	
CPU	• Rockchip RK3588S (8nm LP 制程） • 8 核 64 位处理器 • 4 核 Cortex-A76 和 4 核 Cortex-A55 大小核架构 • 大核主频最高 2.4GHz，小核主频最高 1.8GHz
GPU	• 集成 ARM Mali-G610 • OpenGL ES1.1/2.0/3.2、OpenCL 2.2 和 Vulkan 1.2
NPU	• 内置高达 6 Tops 算力的 AI 加速器 NPU • 支持 INT4/INT8/INT16 混合运算
视频输出	• HDMI 2.1，最大支持 8K @60Hz • DP1.4（DisplayPort） • 2 * MIPI D-PHY TX 4Lane
内存	4GB/8GB/16GB（LPDDR4/4x）
摄像头	• 1 * MIPI CSI 4Lane • 2 * MIPI D-PHY RX 4Lane
PMU	RK806-1
板载存储	• MicroSD (TF) Card 插槽 • 32/64/128/256 GB eMMC
以太网	10/100/1000Mbps 以太网 (YT8531C)
WIFI+BT	板载 WI-FI6+BT 5.0 模块（AP6275P），支持 BLE
音频	• 3.5mm 耳机孔音频输入/输出 • 板载 MIC 输入 • HDMI 输出
USB 接口	1 * USB3.0 接口 2 * USB2.0 接口（其中一个和 Type-C 接口共用） 1 * USB3.0 Type-C 接口
26pin 扩展排针	用于扩展 UART、PWM、I2C、SPI、CAN 和 GPIO 接口
调试串口	3pin 调试串口
LED 灯	电源指示灯和状态指示灯

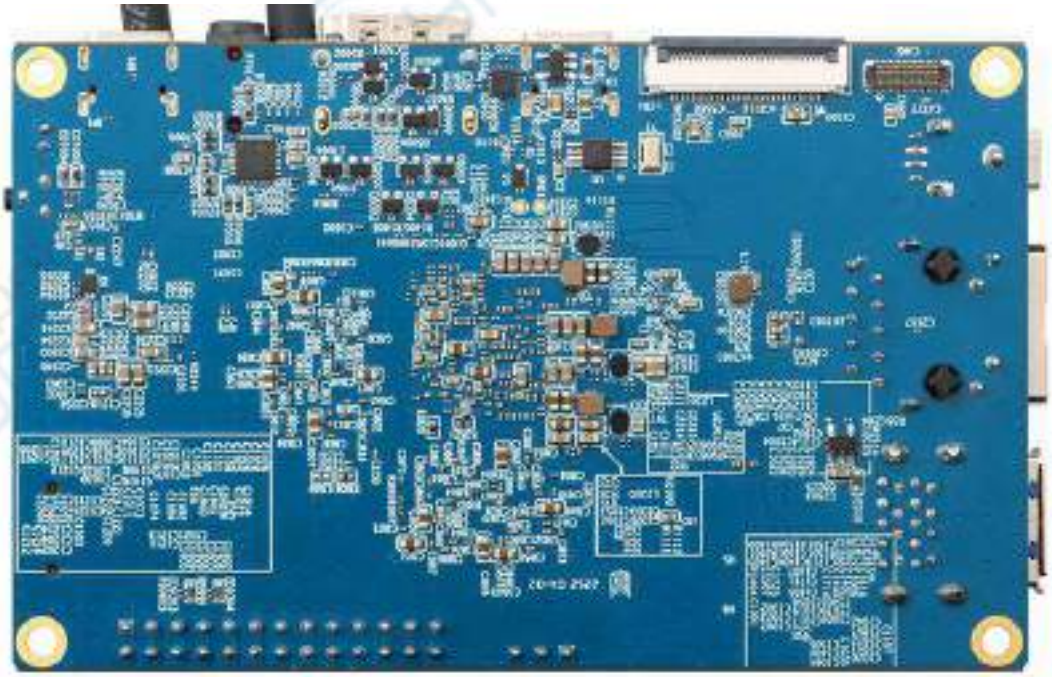
按键	1 * MaskROM 键，1 * RECOVERY，1 * 开关机键
供电	Type-C 接口供电 5V/4A；
支持的操作系统	Orange Pi OS（Droid）、Android12.1、Debian11、Ubuntu20.04 和 Ubuntu22.04 等操作系统
外观规格介绍	
产品尺寸	100mm*62mm
重量	46g
range Pi™ 是深圳市迅龙软件有限公司的注册商标	

1.4. Orange Pi 5B 的顶层视图和底层视图

顶层视图：



底层视图：



1.5. Orange Pi 5B 的接口详情图

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

1) TF 卡，最小 8GB 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。

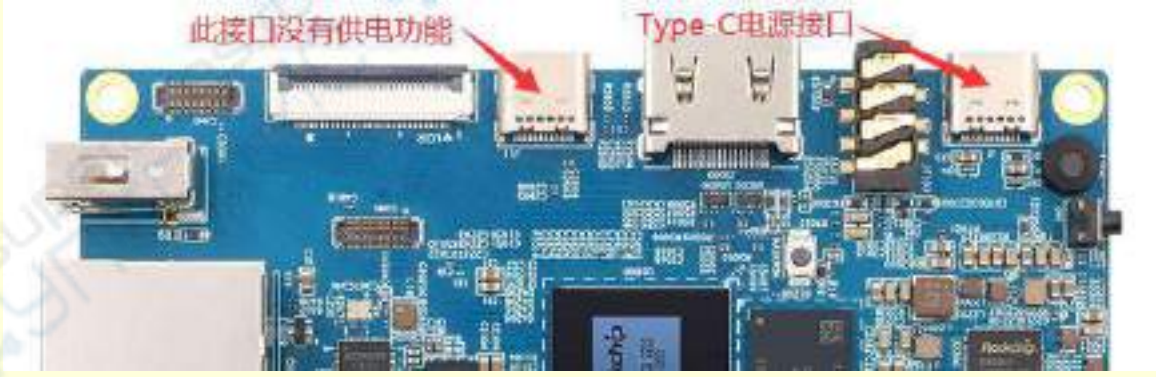
5) Type-C转HDMI线，通过Type-C接口将开发板连接到HDMI显示器或者电视进行显示

6) 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

7) 10.1 寸MIPI屏幕，用于显示开发板的系统界面

8) 电源适配器，Orange Pi 5B 建议使用 5V/4A 的 Type-C 电源供电

开发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，中间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。

9) 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

10) USB摄像头

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

注意，开发板插上Type-C接口的电源后 26pin排针上的 5V引脚就可以直接使用，无需其他设置，另外 26pin排针上的 5V引脚输出的电压是无法通过软件调节和关闭的（没有PWM的功能）。

12) 百兆或者千兆网线，用于将开发板连接到因特网

13) Type-C接口的数据线，用于烧录镜像、使用ADB等功能

14) 1300 万MIPI接口的OV13850 摄像头

15) 1300 万MIPI接口的OV13855 摄像头

16) 配套外壳（待添加图片和组装方法）

17) **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

18) 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:///html/hardWare/computerAndMicrocontrollers/service-and-support/Orange-Pi-5B.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-5B.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

2.3. 将 Linux 镜像烧录到 TF 卡的方法

2.3.1. 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或者OPi OS Arch这样的Linux发行版镜像。

2.3.1.1. 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/

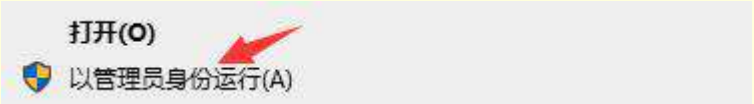
5) 进入 balenaEtcher 下载页面后，点击绿色的下载按钮就可以下载 balenaEtcher 的安装包，也可以通过下拉框选择 balenaEtcher 的 Portable 版本的软件，Portable 版本无需安装，双击打开就可以使用

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

所示

打开 balenaEtcher 时如果提示下面的错误：

请选择 balenaEtcher 后点击右键，然后选择以管理员身份运行。



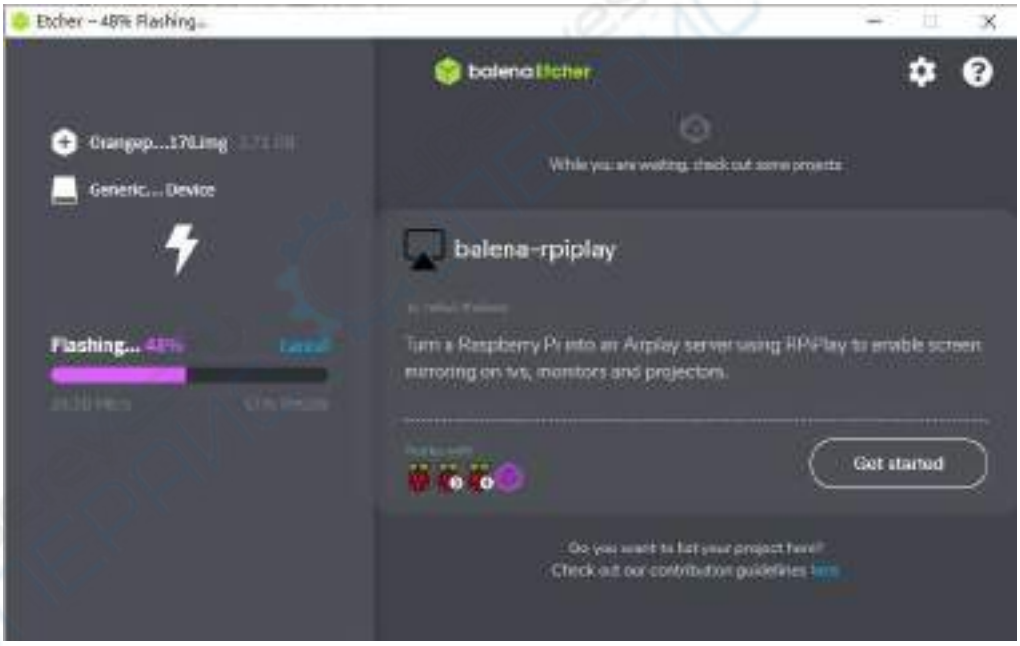
7) 使用 balenaEtcher 烧录 Linux 镜像的具体步骤如下所示

a. 首先选择要烧录的 Linux 镜像文件的路径

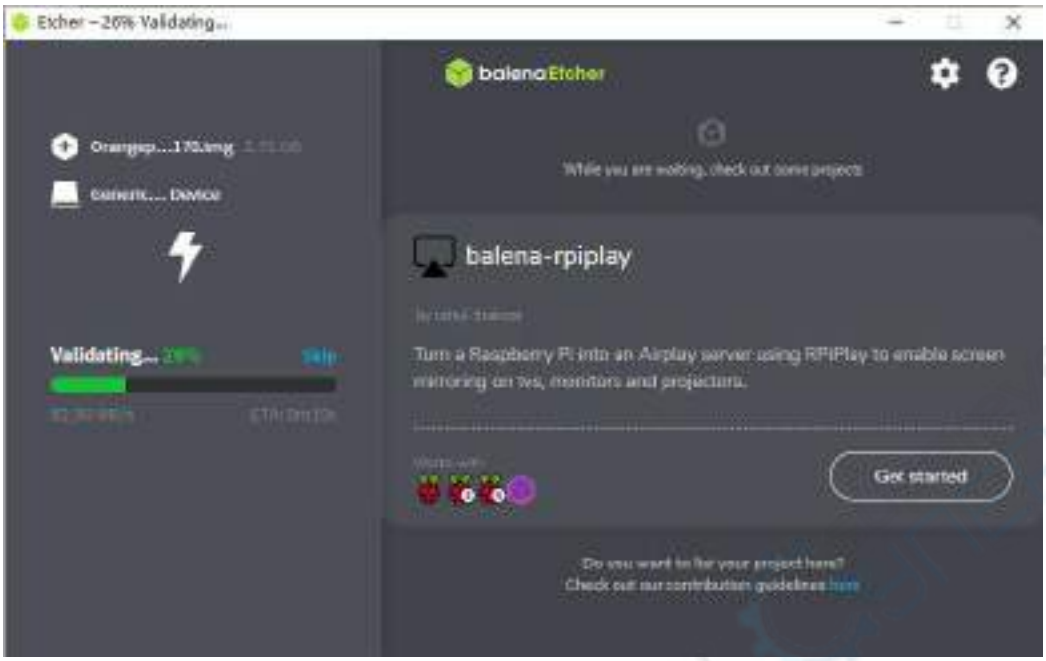
b. 然后选择 TF 卡的盘符

c. 最后点击 Flash 就会开始烧录 Linux 镜像到 TF 卡中

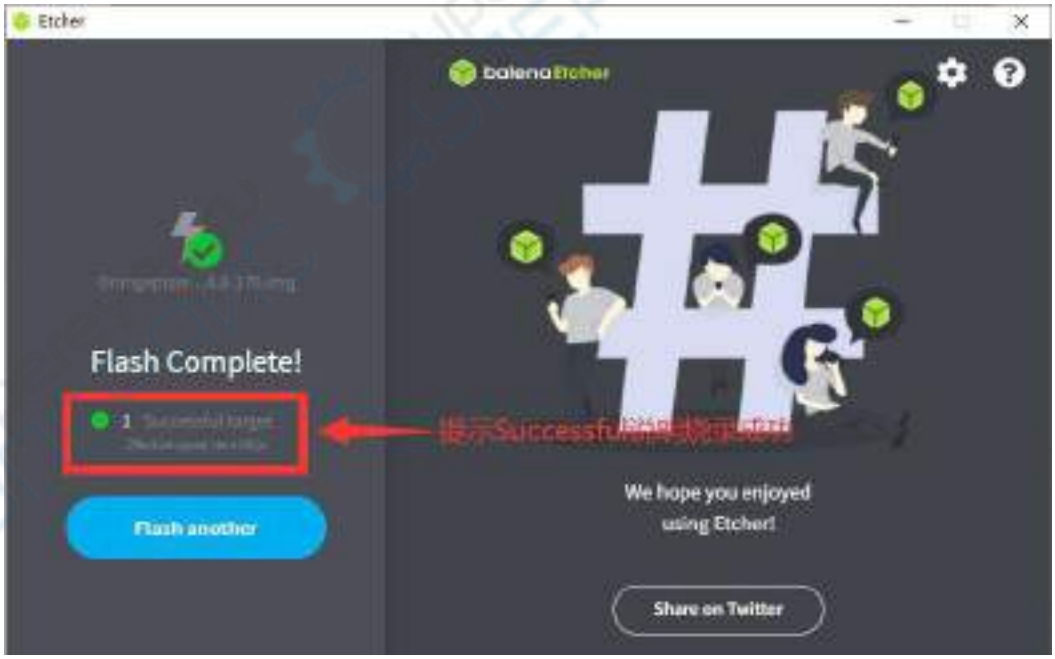
8) balenaEtcher 烧录 Linux 镜像的过程显示的界面如下图所示，另外进度条显示紫色表示正在烧录 Linux 镜像到 TF 卡中



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



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



2.3.1.2. 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 卡插入开发板的卡槽中

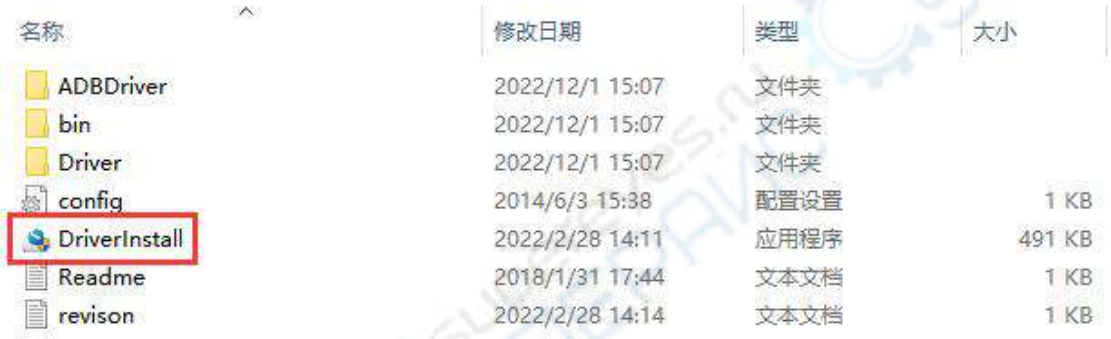
4) 然后从 Orange Pi 的资料下载页面下载瑞芯微驱动 **DriverAssitant_v5.12.zip** 和 **MiniLoader** 以及烧录工具 **RKDevTool_Release_v2.96.zip**，请确保下载的 **RKDevTool** 工具的的版本为 **v2.96**

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

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

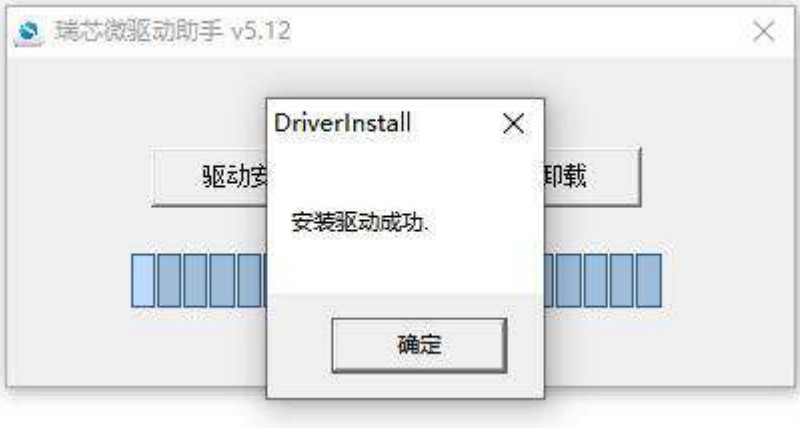
6) 然后用解压软件解压 **DriverAssitant_v5.12.zip**，再在解压后的文件夹中找到 **DriverInstall.exe** 可执行文件并打开即可



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

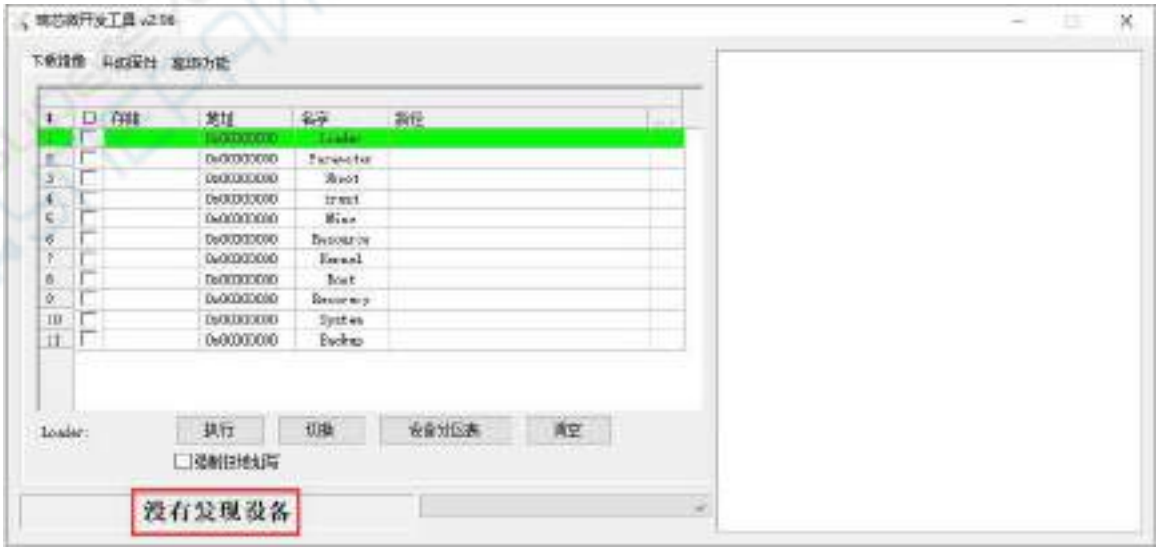
a. 点击“**驱动安装**”按钮

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



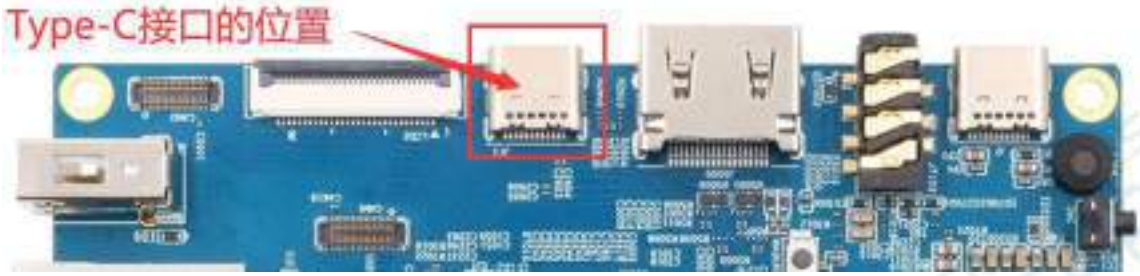
8) 然后解压 **RKDevTool_Release_v2.96.zip**，此软件无需安装，在解压后的文件夹中找到 **RKDevTool** 打开即可

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**”



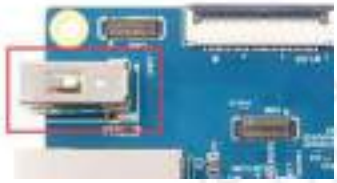
10) 然后开始烧录 Linux 镜像到 eMMC 中

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



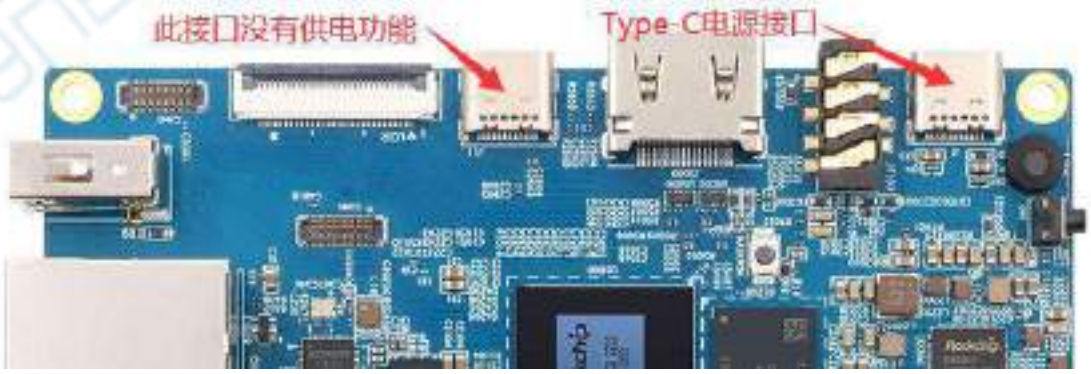
b. 确保开发板没有连接电源

c. 还需确保下图位置的白色 USB2.0 接口没有插入 USB 设备



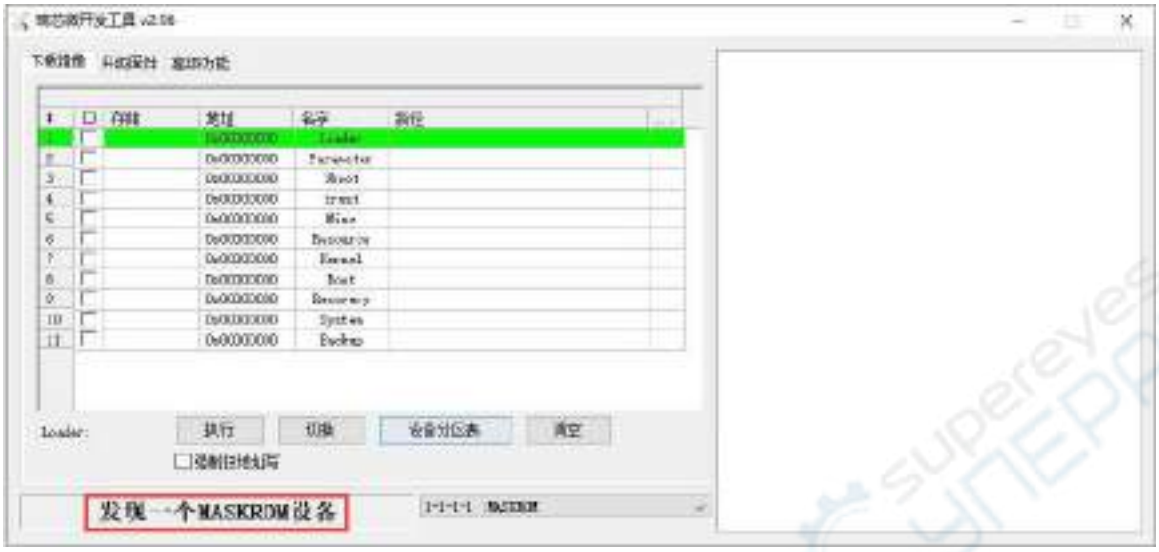
d. 然后按住开发板的 MaskROM 按键不放，MaskROM 按键在开发板的位置如下图所示：

e. 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

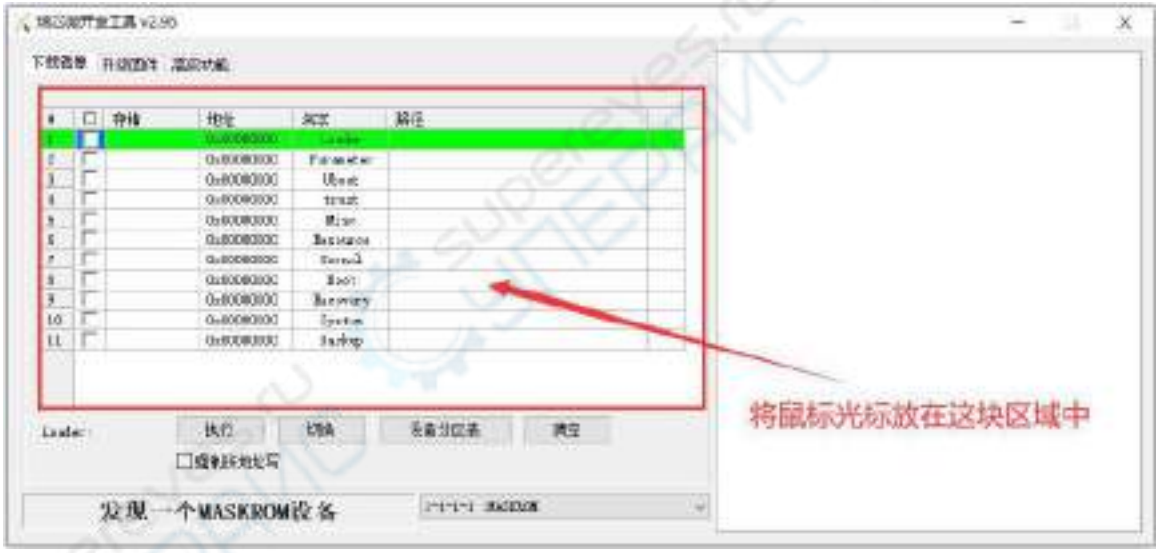


f. 如果前面的步骤顺利，此时开发板会进入 **MASKROM** 模式，在烧录工具的

界面上会提示“**发现一个 MASKROM 设备**”

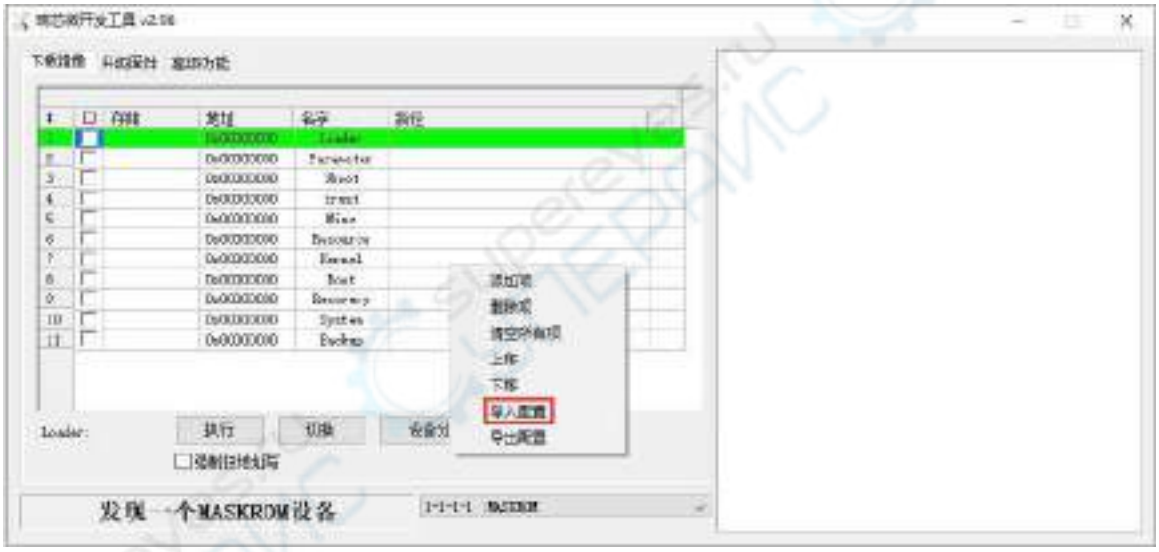


g. 然后将鼠标光标放在下面的这片区域中

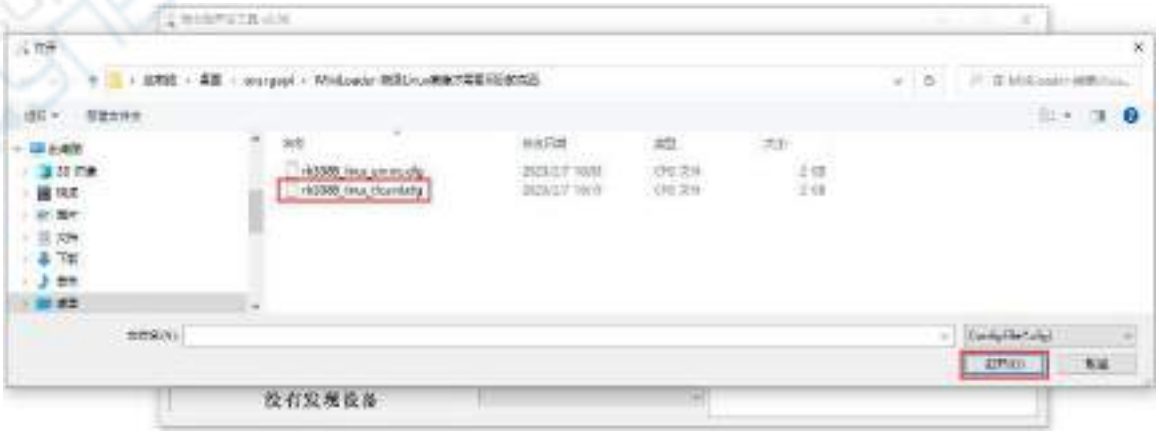


h. 然后点击鼠标右键会弹出下图所示的选择界面

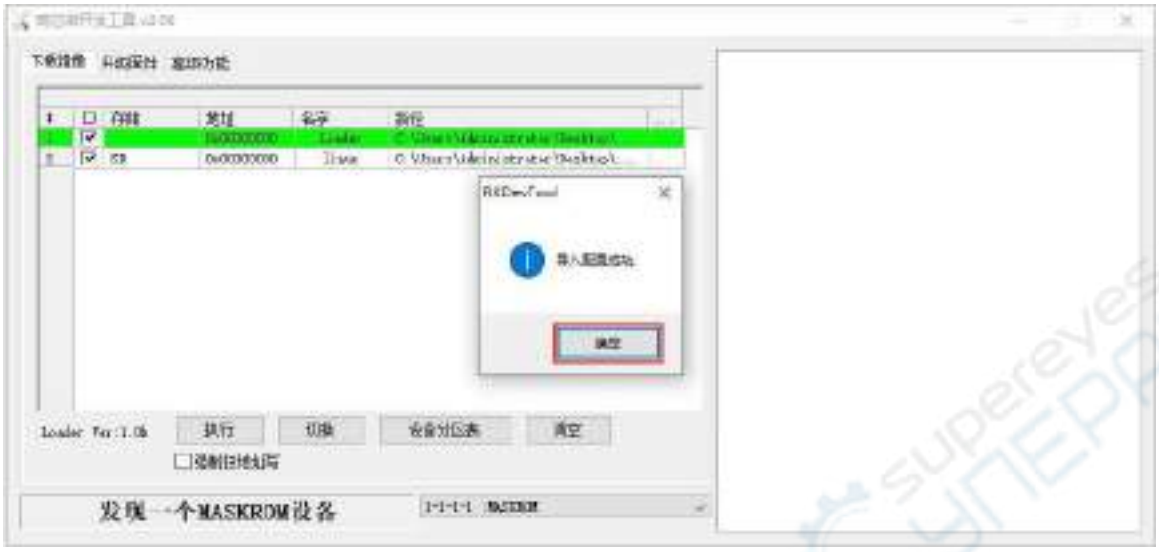
i. 然后选择**导入配置**选项



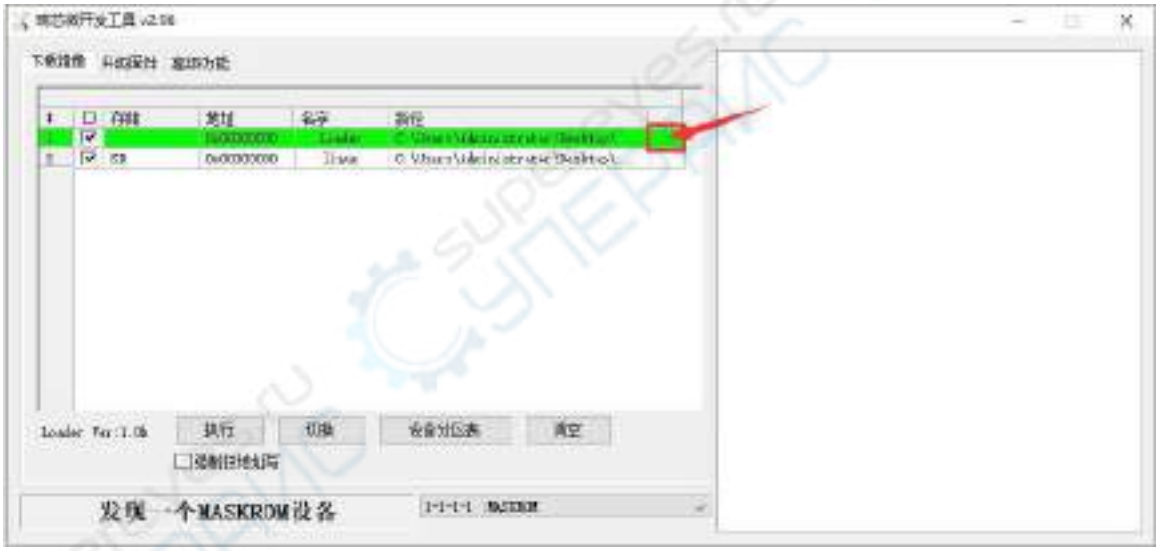
j. 然后选择前面下载的 **MiniLoader** 文件夹中的 **rk3588_linux_tfcard.cfg** 配置文件，再点击**打开**



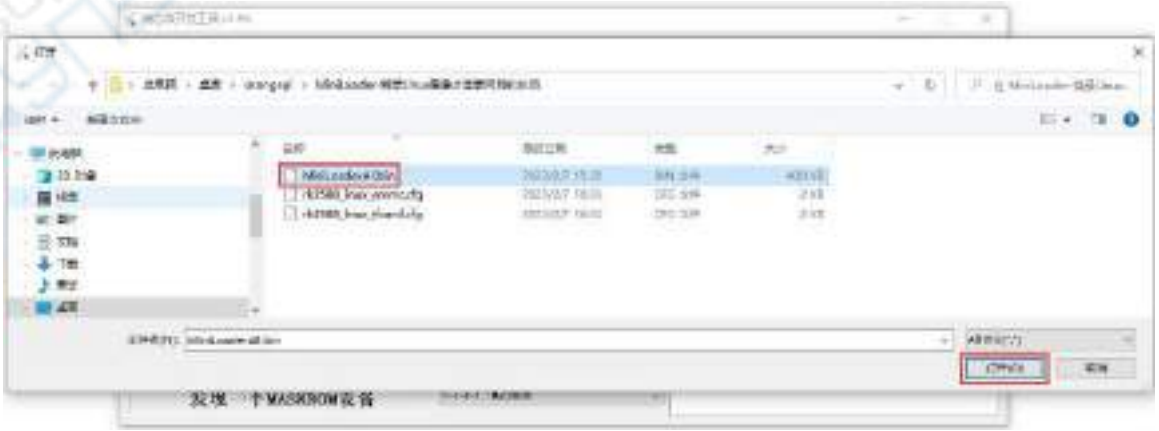
k. 然后点击**确定**



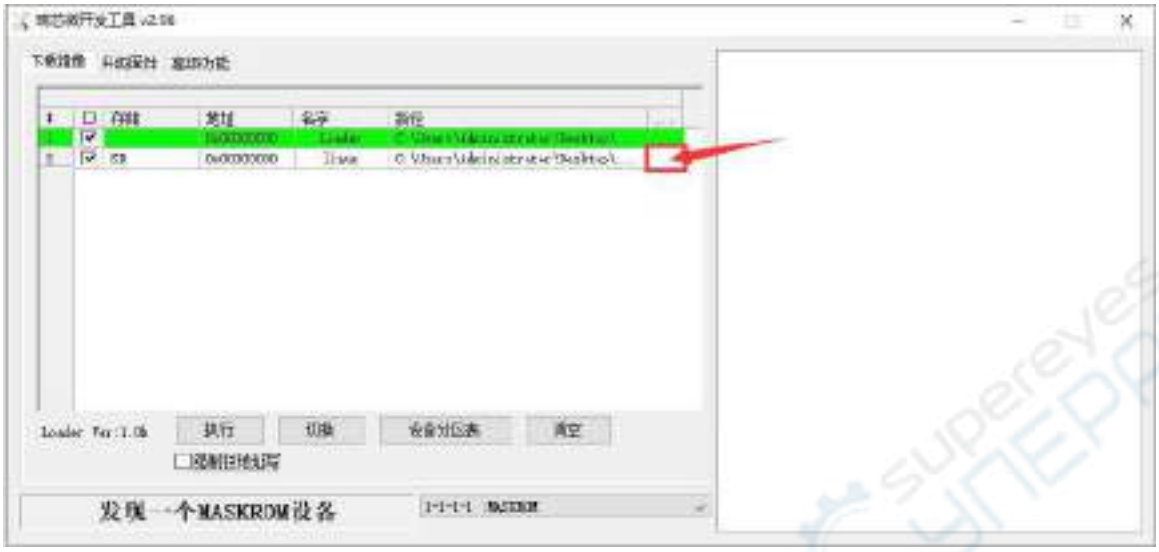
l. 然后点击下图所示的位置



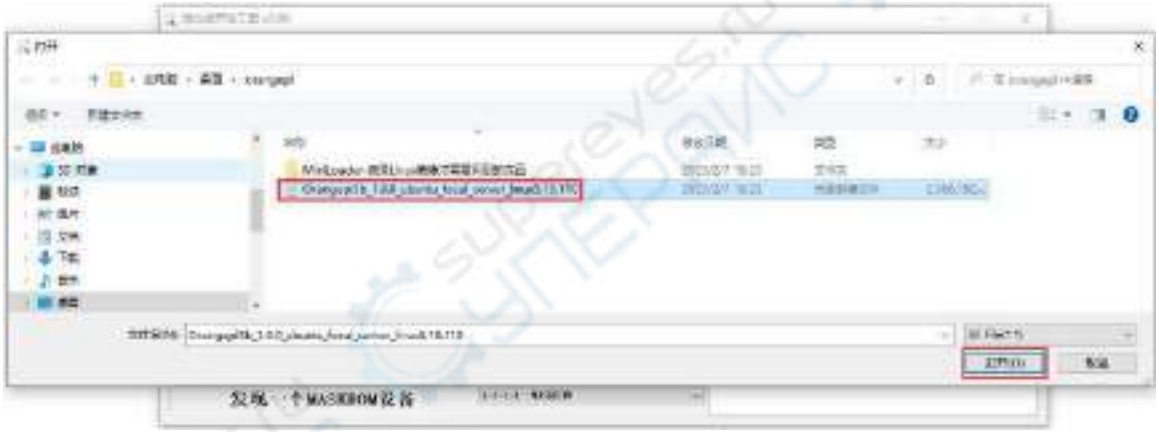
m. 再选择前面下载的 **MiniLoader** 文件夹中 **MiniLoaderAll.bin**，再点击**打开**



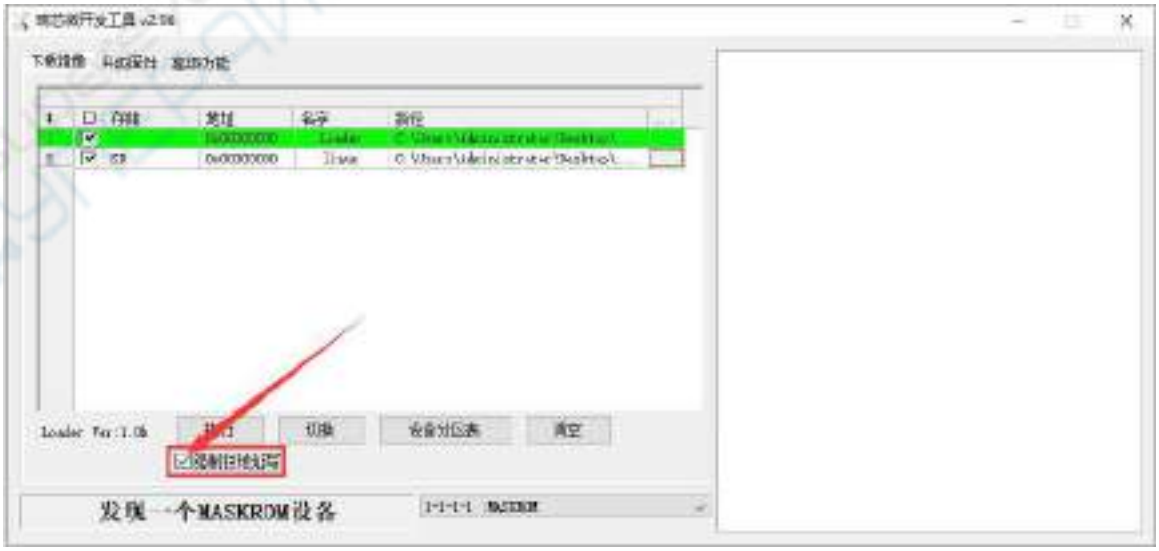
n. 然后点击下图所示的位置



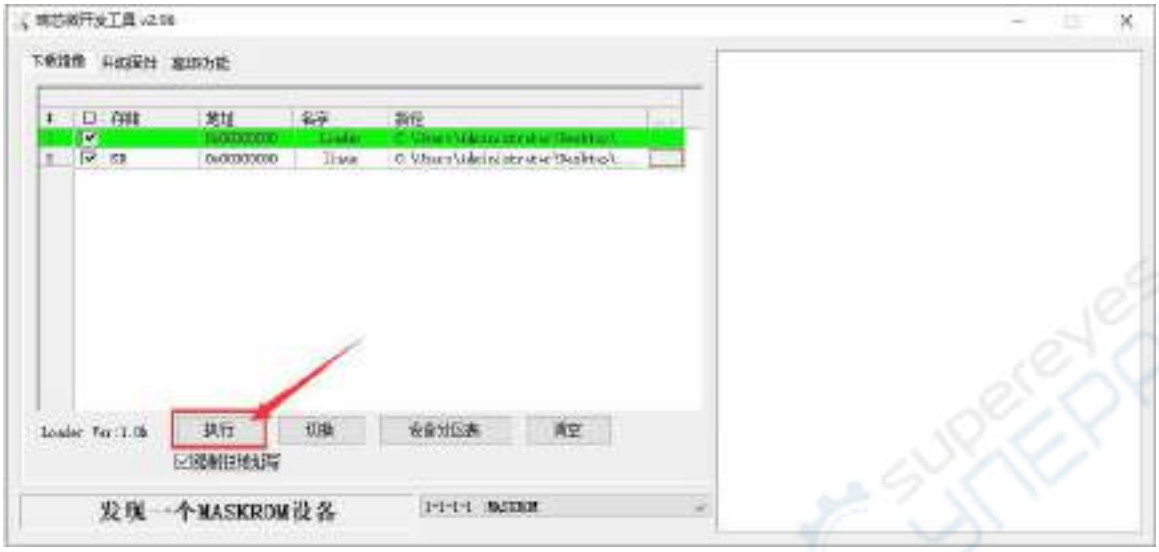
o. 然后选择想要烧录的 linux 镜像的路径，再点击**打开**



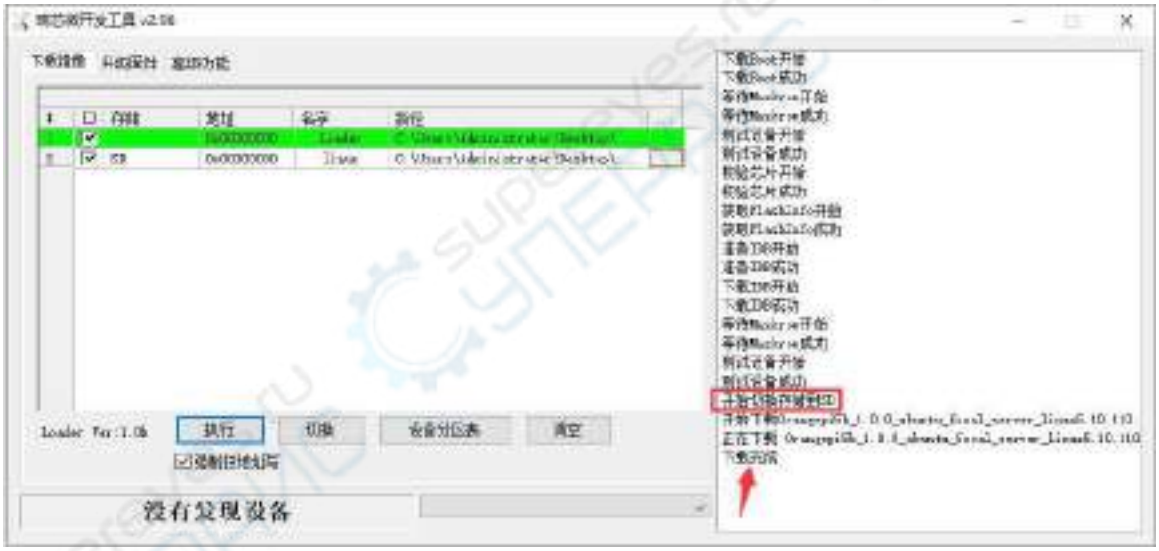
p. 然后请勾选上**强制按地址写**选项



q. 再点击执行按钮就会开始烧录 linux 镜像到开发板的 tf 卡中



r. linux 镜像烧录完后的显示 log 如下图所示



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

2. 3. 1. 3. 使用 Win32Diskimager 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

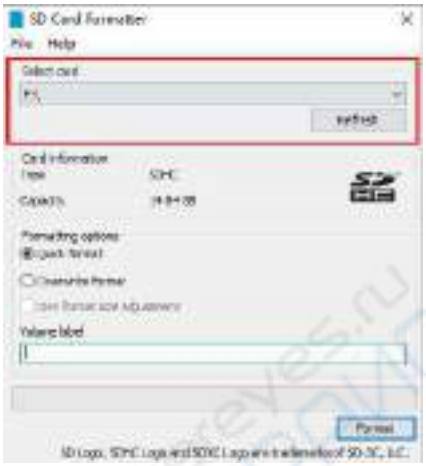
3) 接着格式化 TF 卡

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

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

b. 下载完后直接解压安装即可，然后打开软件

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



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

e. 格式化完 TF 卡后会弹出下图所示的信息，点击确定即可

4) 从Orange Pi的资料下载页面下载想要烧录的Linux操作系统镜像文件压缩包，然后使用解压软件解压，解压后的文件中，以“.img”结尾的文件就是操作系统的镜像文件，大小一般都在 2GB以上

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

a. Win32Diskimager 的下载页面为

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

b. 下载完后直接安装即可，Win32Diskimager 界面如下所示

a) 首先选择镜像文件的路径

b) 然后确认下 TF 卡的盘符和“**设备**”一栏中显示的一致

c) 最后点击“**写入**”即可开始烧录

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

2.3.2. 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或者OPi OS Arch这样的Linux发行版镜像。

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，请通过下拉框选择 Linux 版本的软件进行下载

5) 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

7z 结尾的压缩包的解压命令如下所示

```
test@test:~$ 7z x orangepi5b_1.0.0_debian_bullseye_desktop_xfce_linux5.10.110.7z
test@test:~$ ls orangepi5b_1.0.0_debian_bullseye_desktop_xfce_linux5.10.110.*
orangepi5b_1.0.0_debian_bullseye_desktop_xfce_linux5.10.110.7z
orangepi5b_1.0.0_debian_bullseye_desktop_xfce_linux5.10.110.sha     #校验和文件
orangepi5b_1.0.0_debian_bullseye_desktop_xfce_linux5.10.110.img     #镜像文件
```

6) 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

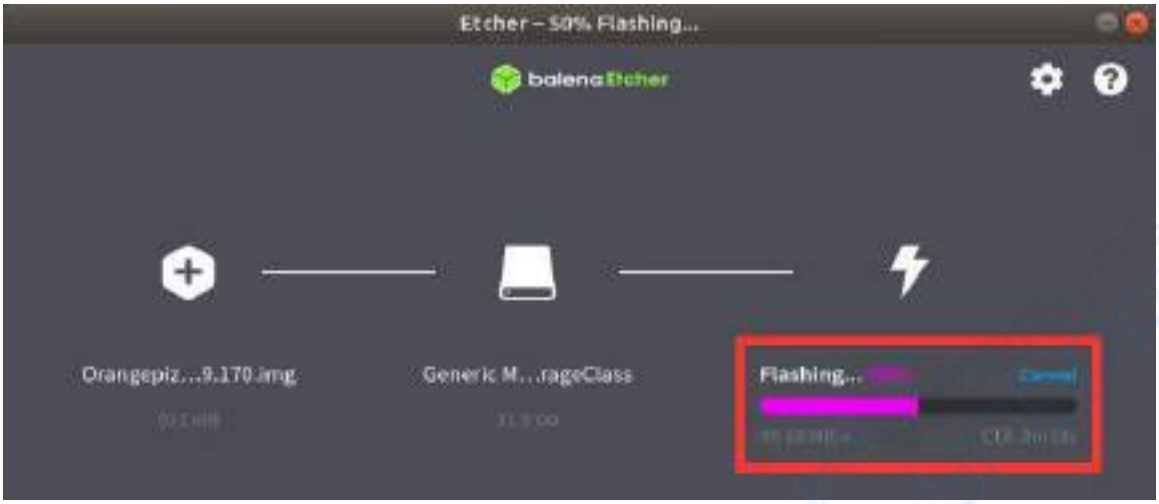
```
test@test:~$ sha256sum -c *.sha
orangepi5b_1.0.0_debian_bullseye_desktop_xfce_linux5.10.110.img: OK
```

7) 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

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

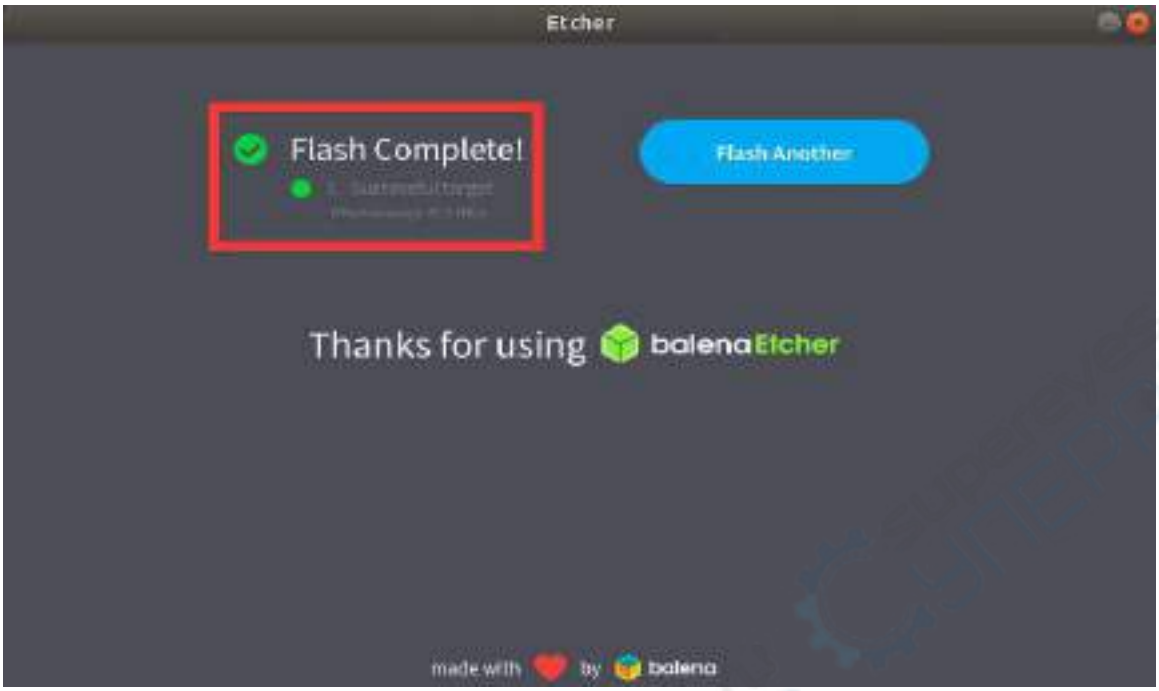
a. 首先选择要烧录的 Linux 镜像文件的路径
b. 然后选择 TF 卡的盘符
c. 最后点击 Flash 就会开始烧录 Linux 镜像到 TF 卡中

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



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



2.4. 烧录 Android 镜像到 TF 卡中的方法

注意，下面所有的操作都是在 Windows 电脑中进行的。

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

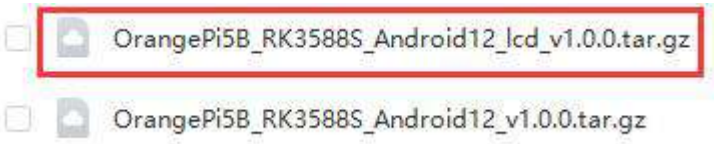
2) 然后使用读卡器把 TF 卡插入电脑

3) 然后从 Orange Pi 的资料下载页面下载 SDDiskTool 烧写工具，**请确保 SDDiskTool 工具的版本为最新的 v1.72**

4) 然后从 Orange Pi 的资料下载页面下载 Android 的镜像。打开安卓镜像的下载链接后可以看到下面两种类型的安卓镜像，它们的区别是：

a. 不带 lcd 的镜像是专门用于 HDMI 显示的，如果不使用 LCD 屏幕，请下载不带 lcd 的镜像

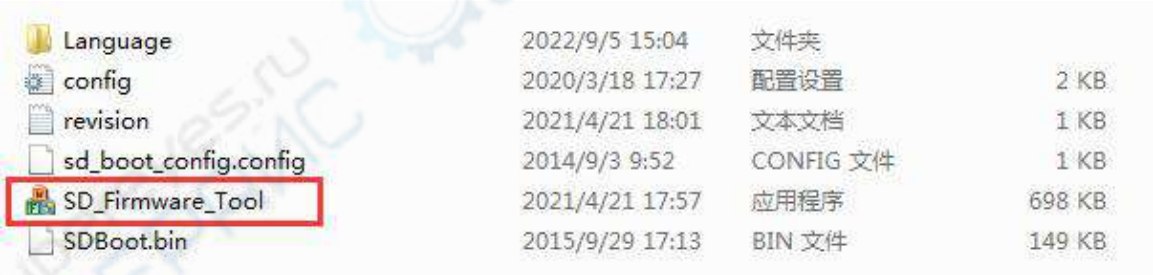
b. 如果要使用 LCD 屏幕，请选择带 lcd 的镜像



c. 带 box 的镜像为电视盒子类型的镜像

5) 然后使用解压软件解压下载的 Android 镜像的压缩包，解压后的文件中，以“.img”结尾的文件就是 Android 镜像文件，大小在 1GB 以上

6) 然后使用解压软件解压 **SDDiskTool_v1.72.zip**，此软件无需安装，在解压后的文件夹中找到 **SD_Firmware_Tool.exe** 打开即可



7) 打开 SDDiskTool 后，如果 TF 卡识别正常，会在“**选择可移动磁盘设备**”一栏中显示插入的磁盘设备，**请务必确认显示的磁盘设备和你想烧录的 TF 卡的盘符是一致的**，如果没有显示可以尝试拔插下 TF 卡

8) 确认完盘符后，可以先格式化下 TF 卡，点击 SDDiskTool 中的**恢复磁盘**按钮即可，也可使用前面提到的 **SD Card Formatter** 进行 TF 卡的格式化

9) 然后开始将 Android 镜像写入 TF 卡

a. 首先在“**选择功能模式**”中勾选“**SD 启动**”

b. 然后在“**选择升级固件**”一栏中选择 Android 镜像的路径

c. 最后点击“**开始创建**”按钮就会开始烧录 Android 镜像到 TF 卡中

10) 烧录完后即可退出 SDDiskTool 软件，然后就可以把 TF 卡从电脑中拔出来插到开发板中启动了

2.5. 烧录 Orange Pi OS (Droid)镜像到 TF 卡中的方法

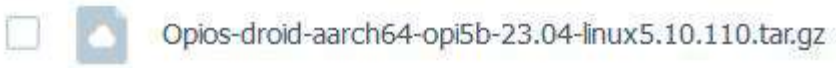
注意，下面所有的操作都是在 Windows 电脑中进行的。

1) 首先准备一张 8GB 或更大容量的 TF 卡，TF 卡的传输速度必须为 **class10** 或以上，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

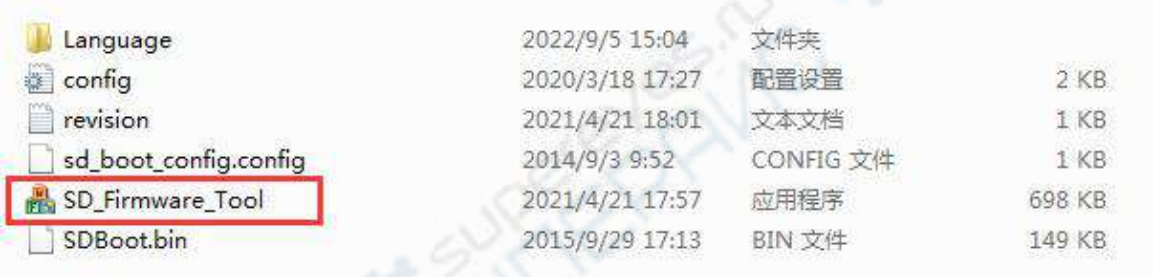
3) 然后从 Orange Pi 的资料下载页面下载 SDDiskTool 烧写工具，**请确保 SDDiskTool 工具的版本为最新的 v1.72**

4) 然后从 Orange Pi 的资料下载页面下载 Orange Pi OS (Droid)的镜像



5) 然后使用解压软件解压下载的 Orange Pi OS (Droid)镜像的压缩包，解压后的文件中，以“.img”结尾的文件就是 Orange Pi OS (Droid)镜像文件，大小在 1GB 以上

6) 然后使用解压软件解压 **SDDiskTool_v1.72.zip**，此软件无需安装，在解压后的文件夹中找到 **SD_Firmware_Tool.exe** 打开即可



7) 打开 SDDiskTool 后，如果 TF 卡识别正常，会在“**选择可移动磁盘设备**”一栏中显示插入的磁盘设备，**请务必确认显示的磁盘设备和你想烧录的 TF 卡的盘符是一致的**，如果没有显示可以尝试拔插下 TF 卡

8) 确认完盘符后，可以先格式化下 TF 卡，点击 SDDiskTool 中的**恢复磁盘**按钮即可，也可使用前面提到的 **SD Card Formatter** 进行 TF 卡的格式化

9）然后开始将 Orange Pi OS (Droid)镜像写入 TF 卡

a. 首先在“**选择功能模式**”中勾选“**SD 启动**”
b. 然后在“**选择升级固件**”一栏中选择 Orange Pi OS (Droid)镜像的路径
c. 最后点击“**开始创建**”按钮就会开始烧录 Orange Pi OS (Droid)镜像到 TF 卡中

10) 烧录完后即可退出 SDDiskTool 软件，然后就可以把 TF 卡从电脑中拔出来插到开发板中启动了

2.6. 将 Linux 镜像烧录到 eMMC 中的方法

2.6.1. 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或者OPi OS Arch这样的Linux发行版镜像。

1) 首先需要准备一根品质良好的 Type-C 接口的数据线

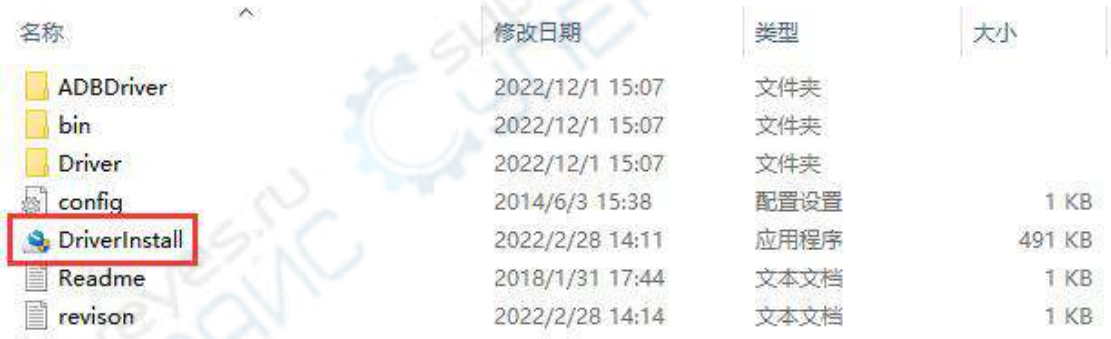
2) 然后从 Orange Pi 的资料下载页面下载瑞芯微驱动 **DriverAssitant_v5.12.zip** 和 **MiniLoader** 以及烧录工具 **RKDevTool_Release_v2.96.zip**，请确保下载的 **RKDevTool** 工具的的版本为 **v2.96**

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

3) 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

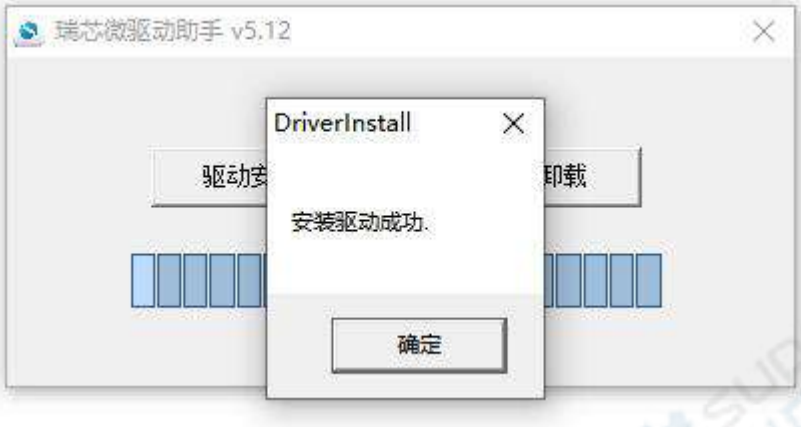
4) 然后用解压软件解压 **DriverAssitant_v5.12.zip**，再在解压后的文件夹中找到 **DriverInstall.exe** 可执行文件并打开即可



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

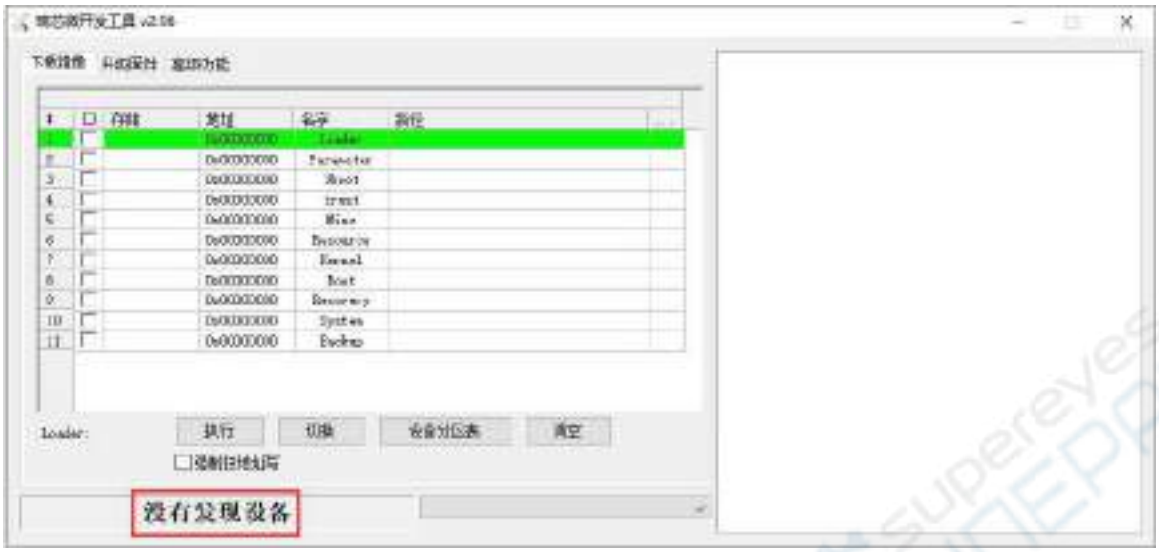
a. 点击“**驱动安装**”按钮

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



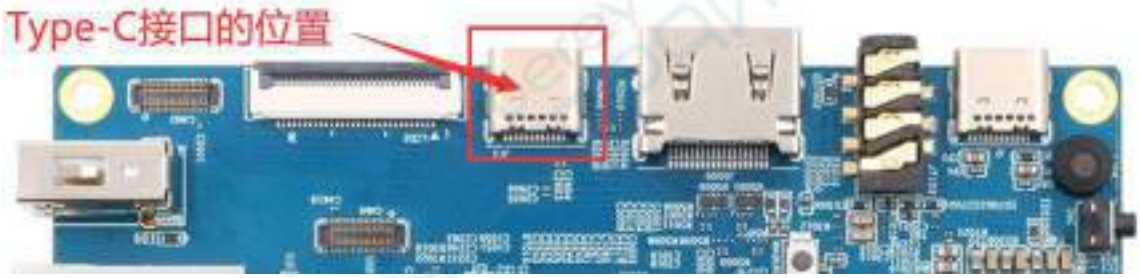
6) 然后解压 **RKDevTool_Release_v2.96.zip**，此软件无需安装，在解压后的文件夹中找到 **RKDevTool** 打开即可

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



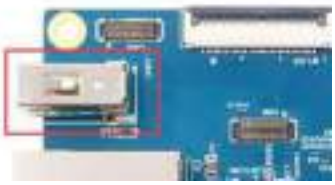
8) 然后开始烧录 Linux 镜像到 eMMC 中

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



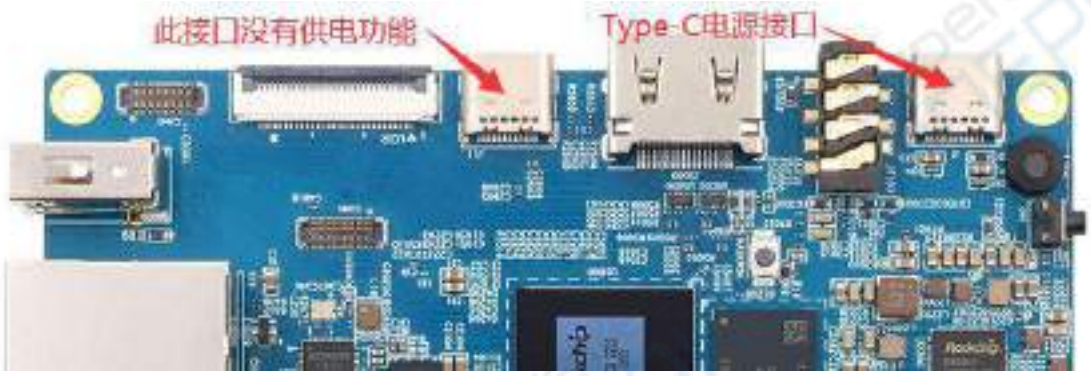
b. 确保开发板没有插入 TF 卡，没有连接电源

c. 还需确保下图位置的白色 USB2.0 接口没有插入 USB 设备

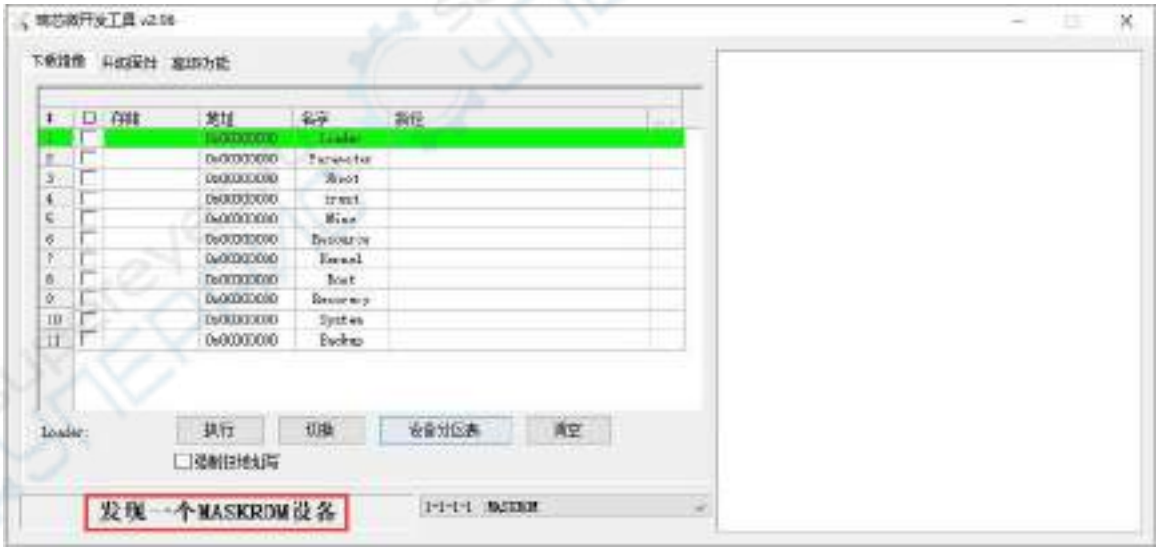


d. 然后按住开发板的 MaskROM 按键不放，MaskROM 按键在开发板的位置如下图所示：

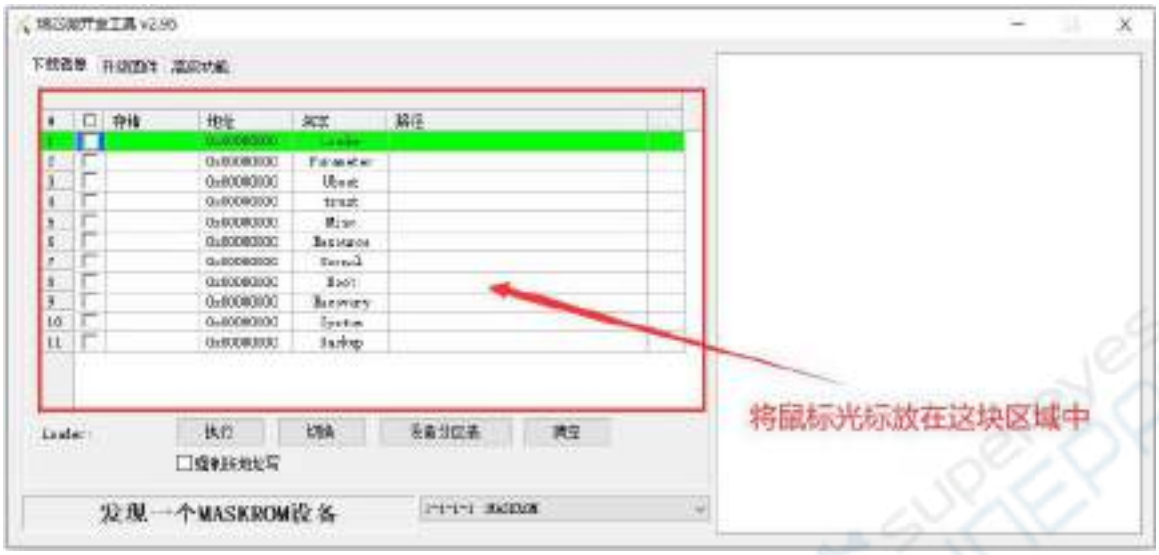
e. 然后给开发板接上 Type-C 接口的电源，并上电，然后就可以松开 MaskROM 按键了



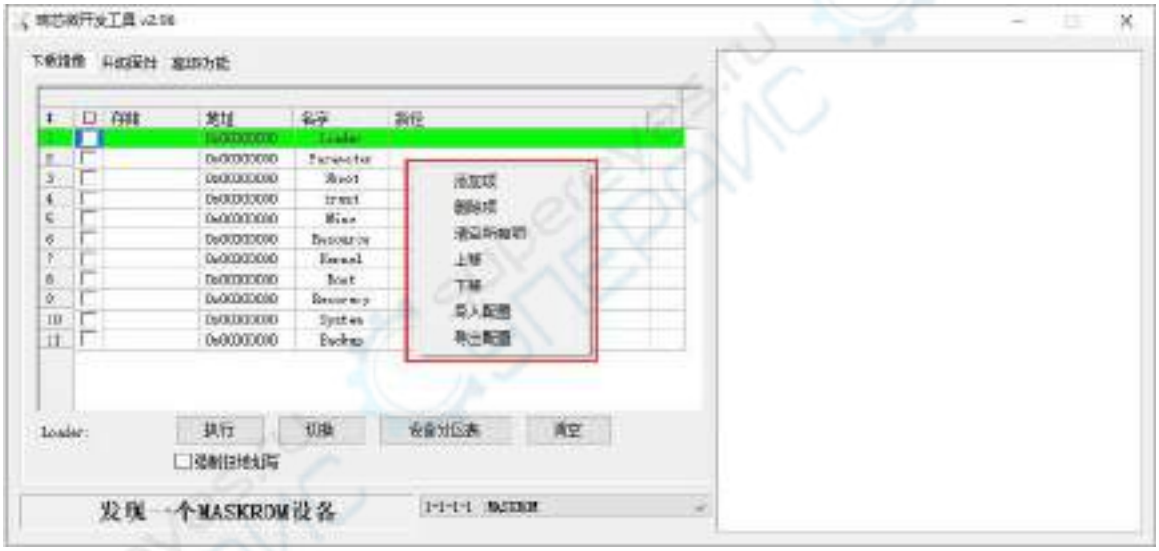
f. 如果前面的步骤顺利，此时开发板会进入 **MASKROM** 模式，在烧录工具的界面上会提示“**发现一个 MASKROM 设备**”



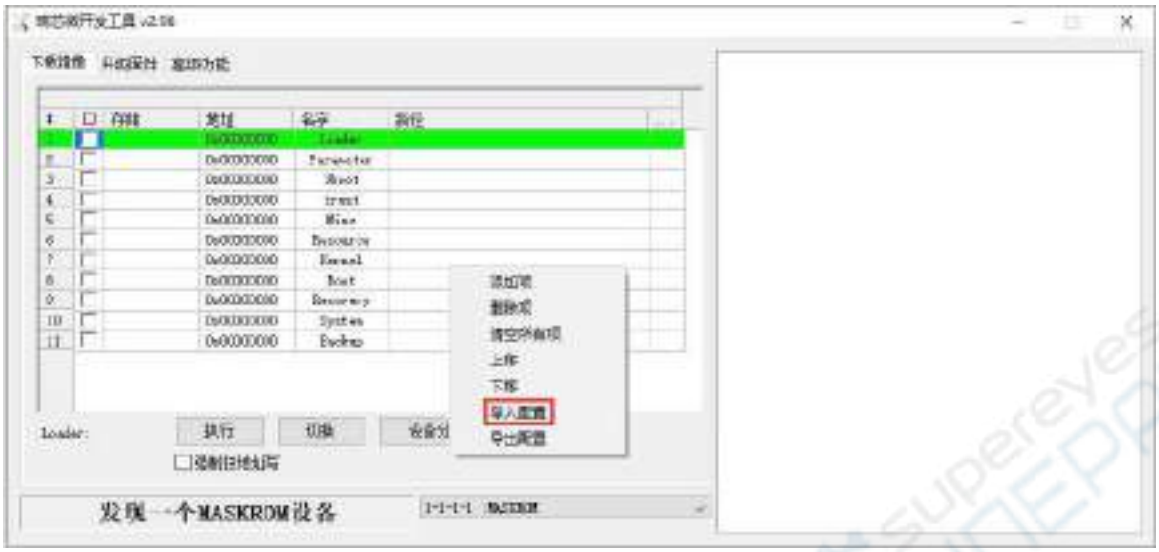
g. 然后将鼠标光标放在下面的这片区域中



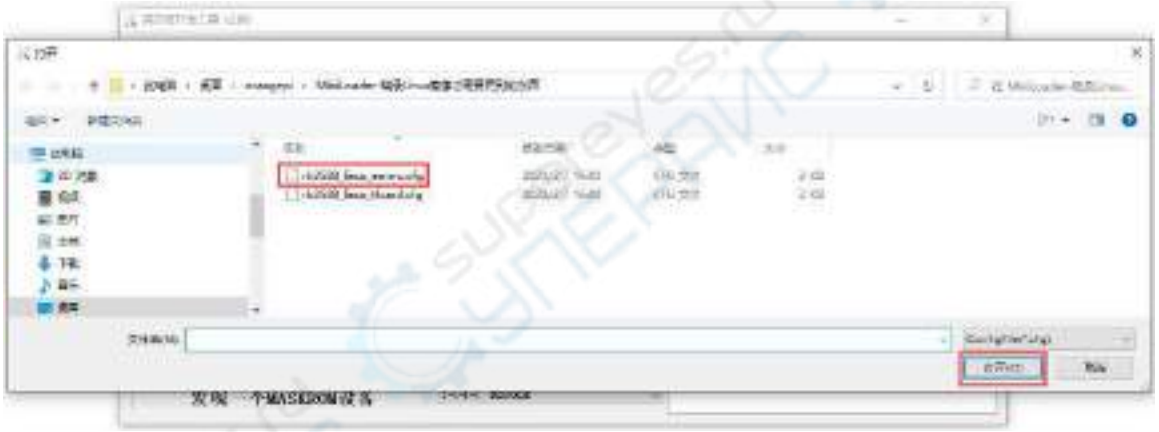
h. 然后点击鼠标右键会弹出下图所示的选择界面



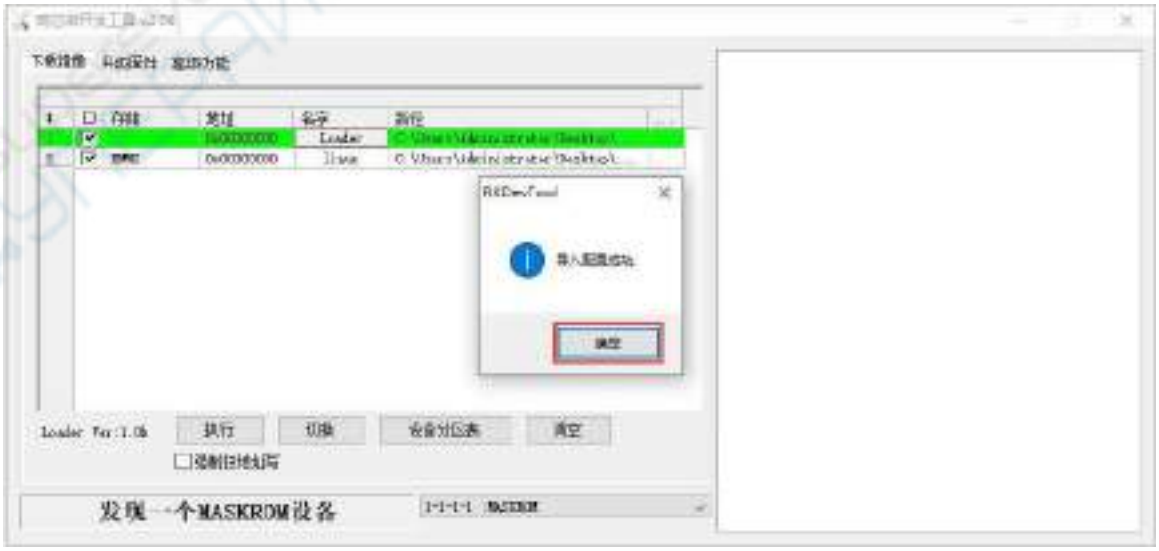
i. 然后选择**导入配置**选项



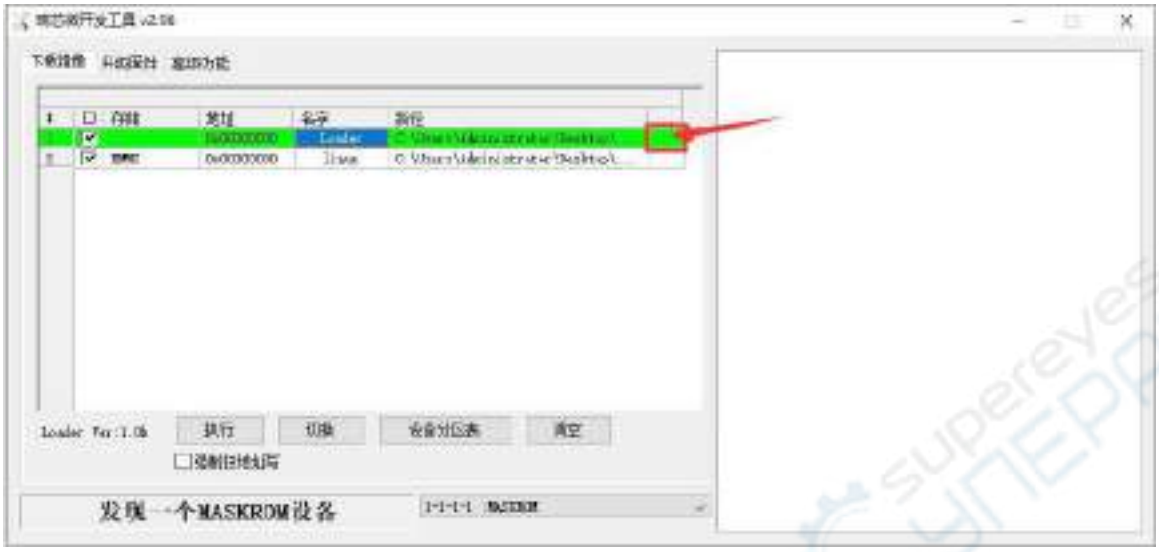
j. 然后选择前面下载的 **MiniLoader** 文件夹中的 **rk3588_linux_emmc.cfg** 配置文件，再点击**打开**



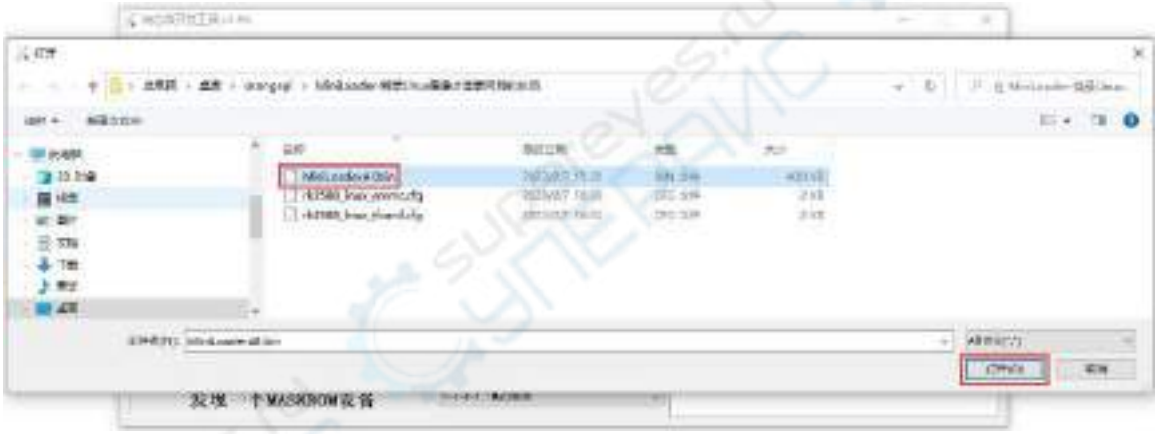
k. 然后点击**确定**



l. 然后点击下图所示的位置



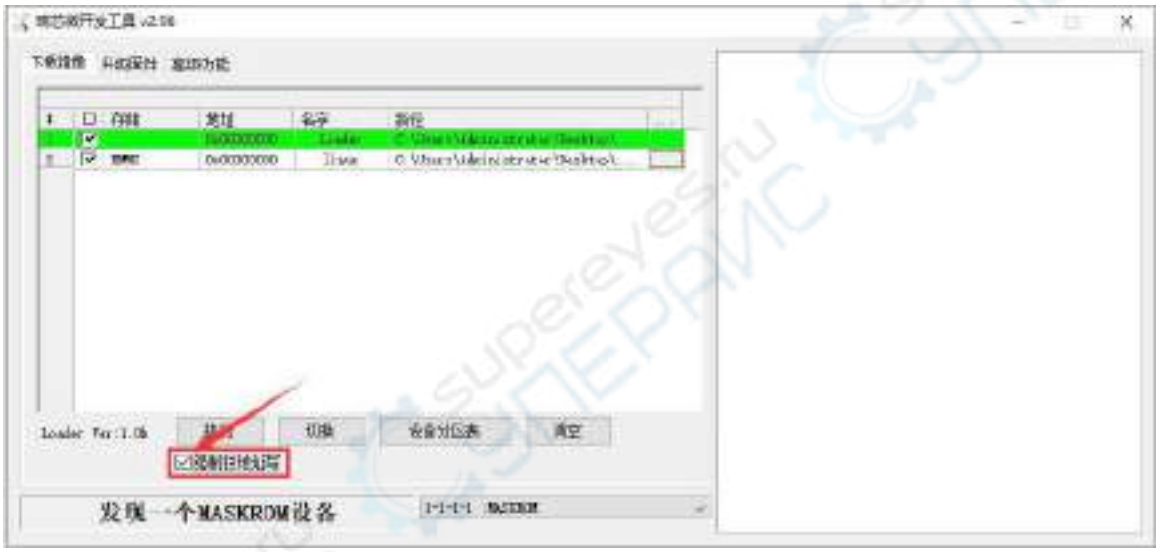
m. 再选择前面下载的 **MiniLoader** 文件夹中 **MiniLoaderAll.bin**，再点击**打开**



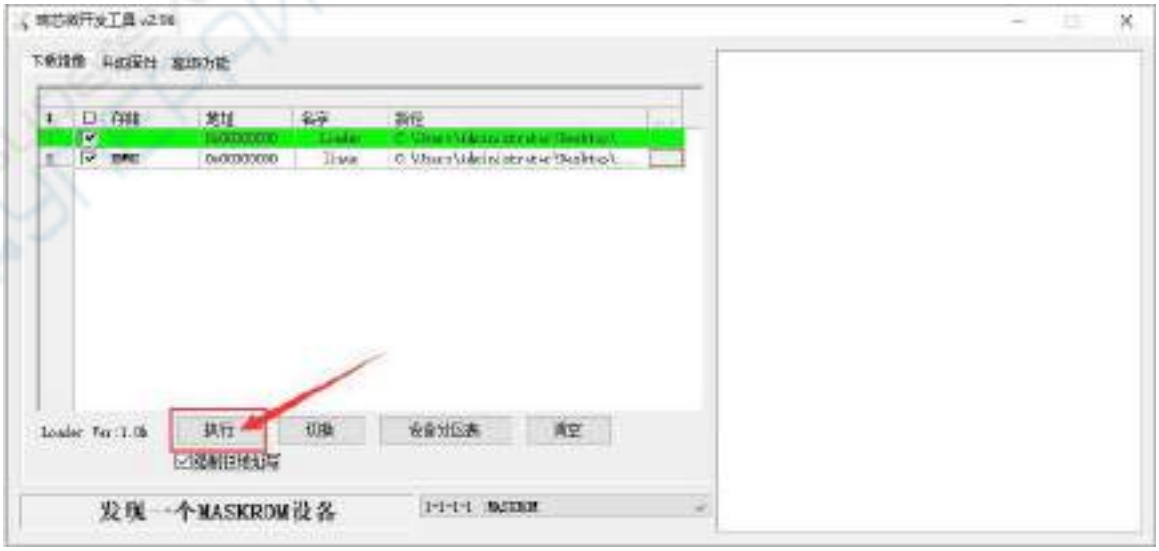
n. 然后点击下图所示的位置

o. 然后选择想要烧录的 linux 镜像的路径，再点击**打开**

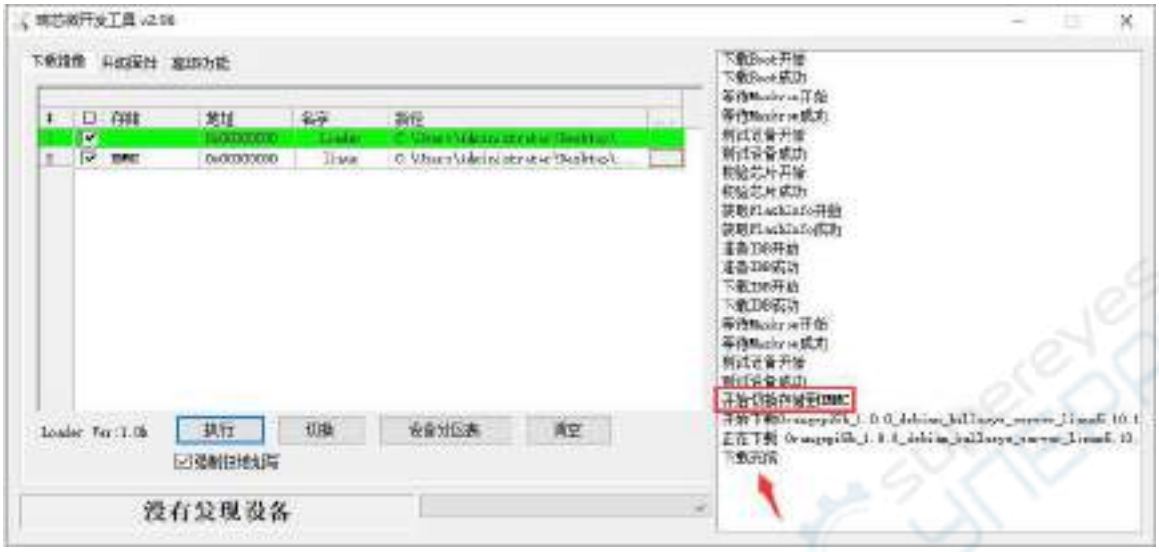
p. 然后请勾选上**强制按地址写**选项



q. 再点击执行按钮就会开始烧录 linux 镜像到开发板的 eMMC 中



r. linux 镜像烧录完后的显示 log 如下图所示



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

2.6.2. 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或者OPi OS Arch这样的Linux发行版镜像。

1) 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**将 Linux 镜像烧录到 TF 卡的方法**小节的说明。

2) 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 或者 Ubuntu 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。

3) 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
Orangepi5b_x.x.x_debian_bullseye_desktop_xfce_linux5.10.110.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。

4) 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/mmcblk0
```

5) 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/mmcblk0 count=1000 status=progress
orangepi@orangepi:~/Desktop$ sudo sync
```

6) 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=Orangepi5b_x.x.x_debian_bullseye_desktop_xfce_linux5.10.110.img of=/dev/mmcblk0 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。

7) 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。

8) 启动 eMMC 中的系统后，使用 **df -h** 命令可以看到实际的硬盘容量

```
orangepi@orangepi:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
tmpfs           769M  9.5M  760M   2% /run
/dev/mmcblk0p2   29G  5.4G   23G  20% /
tmpfs           3.8G     0  3.8G   0% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           3.8G   12K  3.8G   1% /tmp
/dev/mmcblk0p1  256M  116M  141M  46% /boot
/dev/zram1      188M  2.8M  171M   2% /var/log
tmpfs           769M   80K  769M   1% /run/user/1000
```

2.7. 将 Android 镜像烧录到 eMMC 中的方法

2.7.1. 通过 Type-C 线将 Android 镜像烧录到 eMMC 中的方法

注意，下面所有的操作都是在 Windows 电脑中进行的。

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_v2.96.zip**，请确保下载的 **RKDevTool** 工具的的版本为 **v2.96**

3) 然后从 Orange Pi 的资料下载页面下载 Android 的镜像。打开安卓镜像的下载

a. 不带 lcd 的镜像是专门用于 HDMI 显示的，如果不使用 LCD 屏幕，请下载不带 lcd 的镜像

b. 如果要使用 LCD 屏幕，请选择带 lcd 的镜像

c. 带 box 的镜像为电视盒子类型的镜像

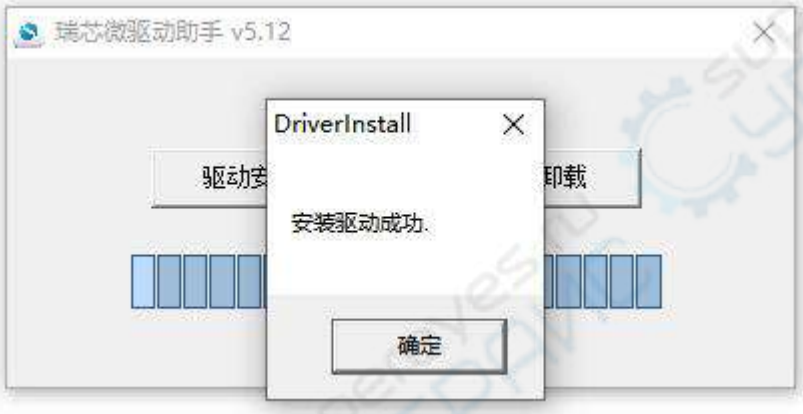
4) 然后使用解压软件解压下载的 Android 镜像的压缩包，解压后的文件中，以“.img”结尾的文件就是 Android 镜像文件，大小在 1GB 以上

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

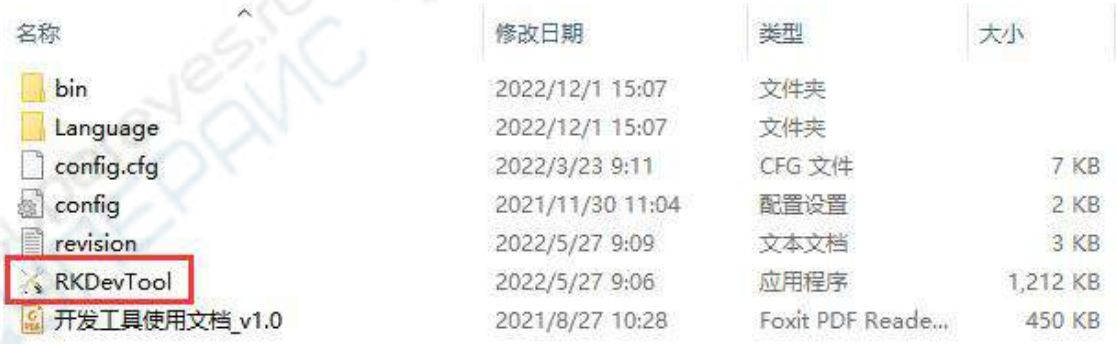
6) 打开 **DriverInstall.exe** 后安装瑞芯微驱动的步骤如下所示

a. 点击“**驱动安装**”按钮

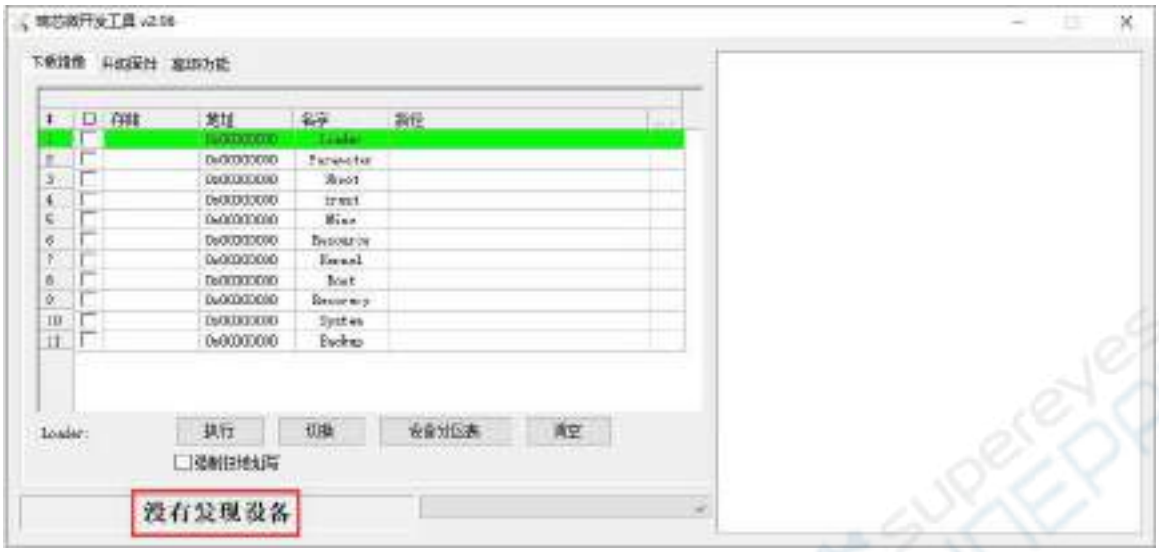
b. 等待一段时间后，会弹出窗口提示“**安装驱动成功**”，然后点击“**确定**”按钮即可



7) 然后解压 **RKDevTool_Release_v2.96.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 线连接上开发板，所以左下角会提示“**没有发现设备**”

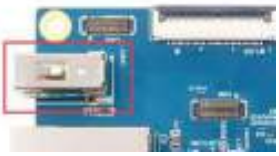


9) 然后开始烧录 Android 镜像到 eMMC 中

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

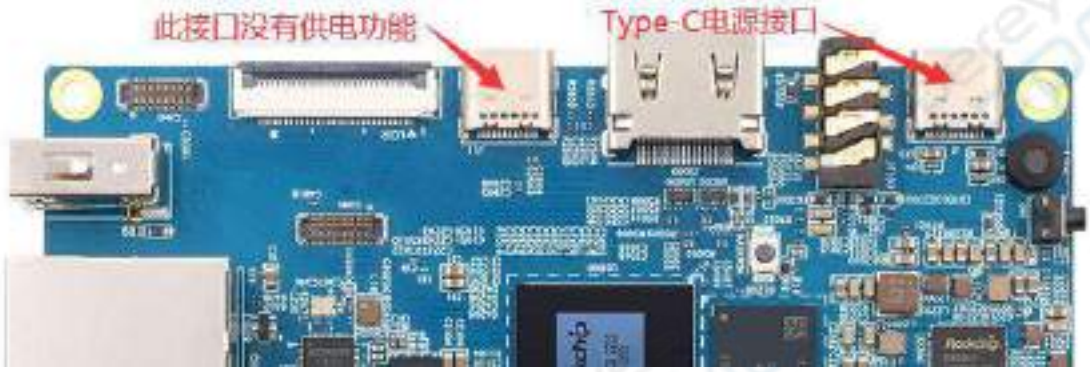
b. 确保开发板没有插入 TF 卡，没有连接电源

c. 还需确保下图位置的白色 USB2.0 接口没有插入 USB 设备

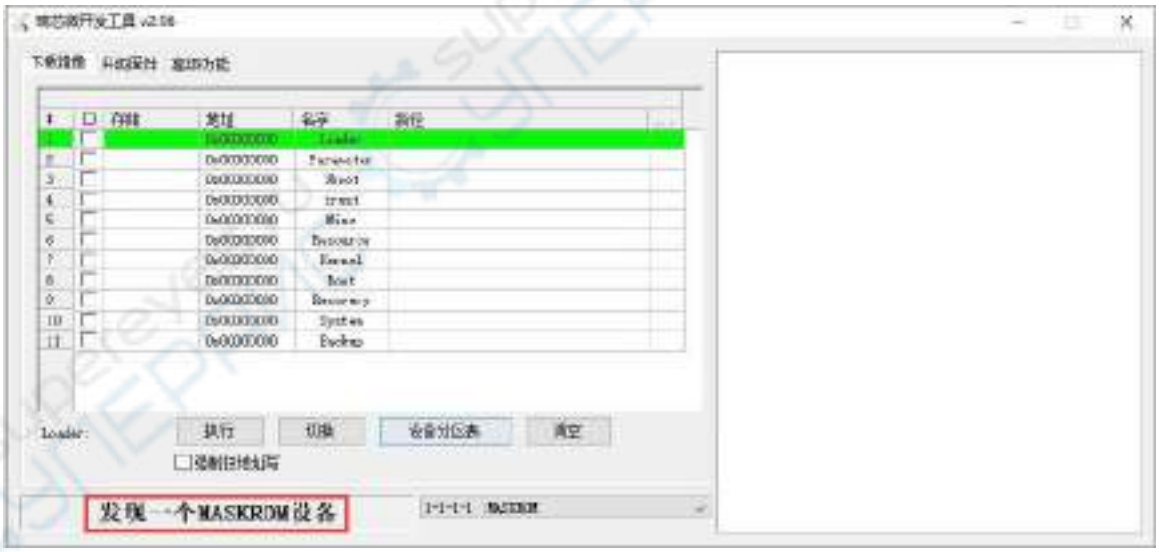


d. 然后按住开发板的 MaskROM 按键不放，MaskROM 按键在开发板的位置如下图所示：

e. 然后给开发板接上 Type-C 接口的电源，并上电



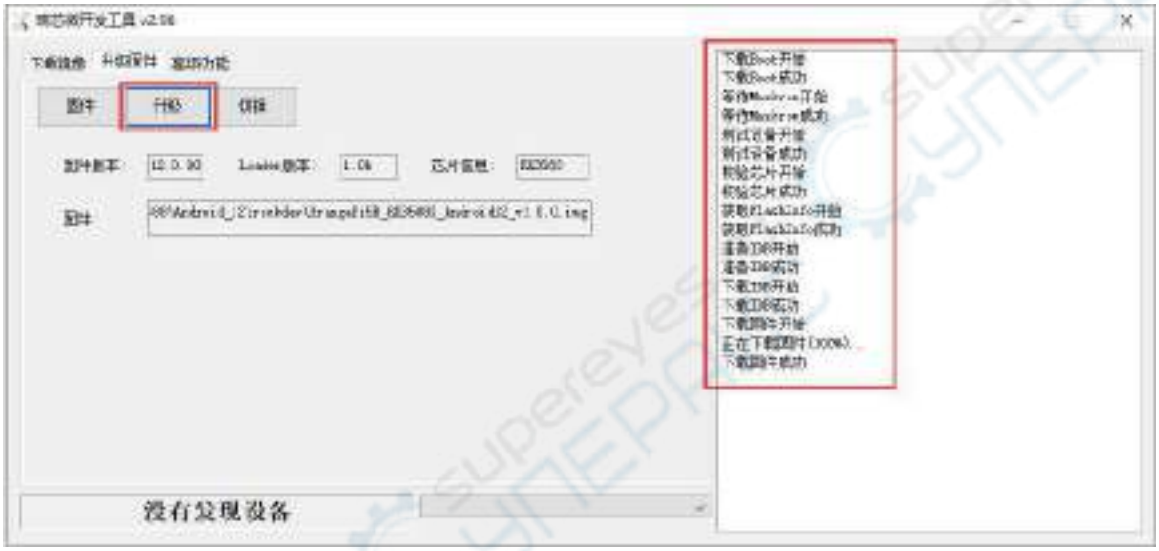
f. 如果前面的步骤顺利，此时开发板会进入 **MASKROM** 模式，在烧录工具的界面上会提示“**发现一个 MASKROM 设备**”



g. 然后点击烧录工具的“**升级固件**”一栏

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

i. 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。



2.7.2. 通过 TF 卡将 Android 12 镜像烧录到 eMMC 中的方法

注意，下面所有的操作都是在 Windows 电脑中进行的。

1) 首先准备一张 8GB 或更大容量的 TF 卡，TF 卡的传输速度必须为 **class10** 或以上，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) 然后从 Orange Pi 的资料下载页面下载 SDDiskTool 烧写工具，**请确保 SDDiskTool 工具的版本为最新的 v1.72**

4) 然后从 Orange Pi 的资料下载页面下载 Android 的镜像。打开安卓镜像的下载链接后可以看到下面两种类型的安卓镜像，它们的区别是：

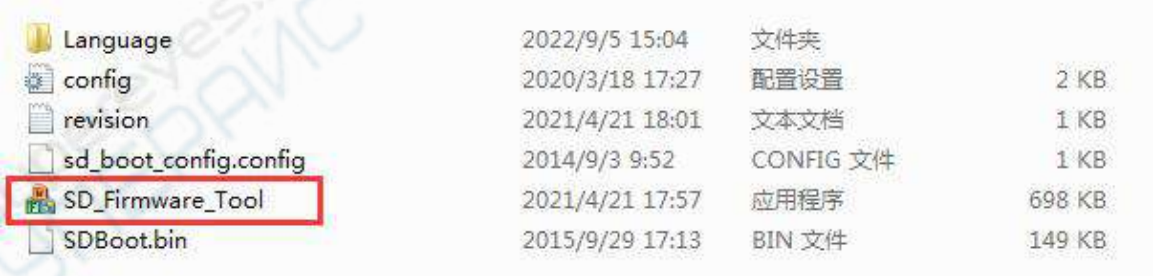
a. 不带 lcd 的镜像是专门用于 HDMI 显示的，如果不使用 LCD 屏幕，请下载不带 lcd 的镜像

b. 如果要使用 LCD 屏幕，请选择带 lcd 的镜像

c. 带 box 的镜像为电视盒子类型的镜像

5) 然后使用解压软件解压下载的 Android 镜像的压缩包，解压后的文件中，以“.img”结尾的文件就是 Android 镜像文件，大小在 1GB 以上

6) 然后使用解压软件解压 **SDDiskTool_v1.72.zip**，此软件无需安装，在解压后的文件夹中找到 **SD_Firmware_Tool.exe** 打开即可



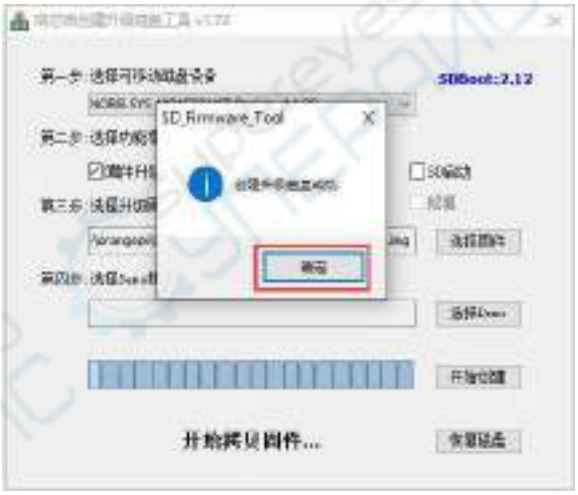
7) 打开 SDDiskTool 后，如果 TF 卡识别正常，会在“**选择可移动磁盘设备**”一栏中显示插入的磁盘设备，**请务必确认显示的磁盘设备和你想烧录的 TF 卡的盘符是一致的**，如果没有显示可以尝试拔插下 TF 卡

8) 确认完盘符后，可以先格式化下 TF 卡，点击 SDDiskTool 中的**恢复磁盘**按钮即可，也可使用前面提到的 **SD Card Formatter** 进行 TF 卡的格式化

9) 然后开始将 Android 镜像写入 TF 卡

a. 首先在**“选择可移动磁盘设备”**下面确认显示的盘符为 TF 卡对应的盘符
b. 然后在**“选择功能模式”**中选择**“固件升级”**
c. 然后在**“选择升级固件”**一栏中选择 Android 固件的路径
d. 最后点击**“开始创建”**按钮就会开始烧录

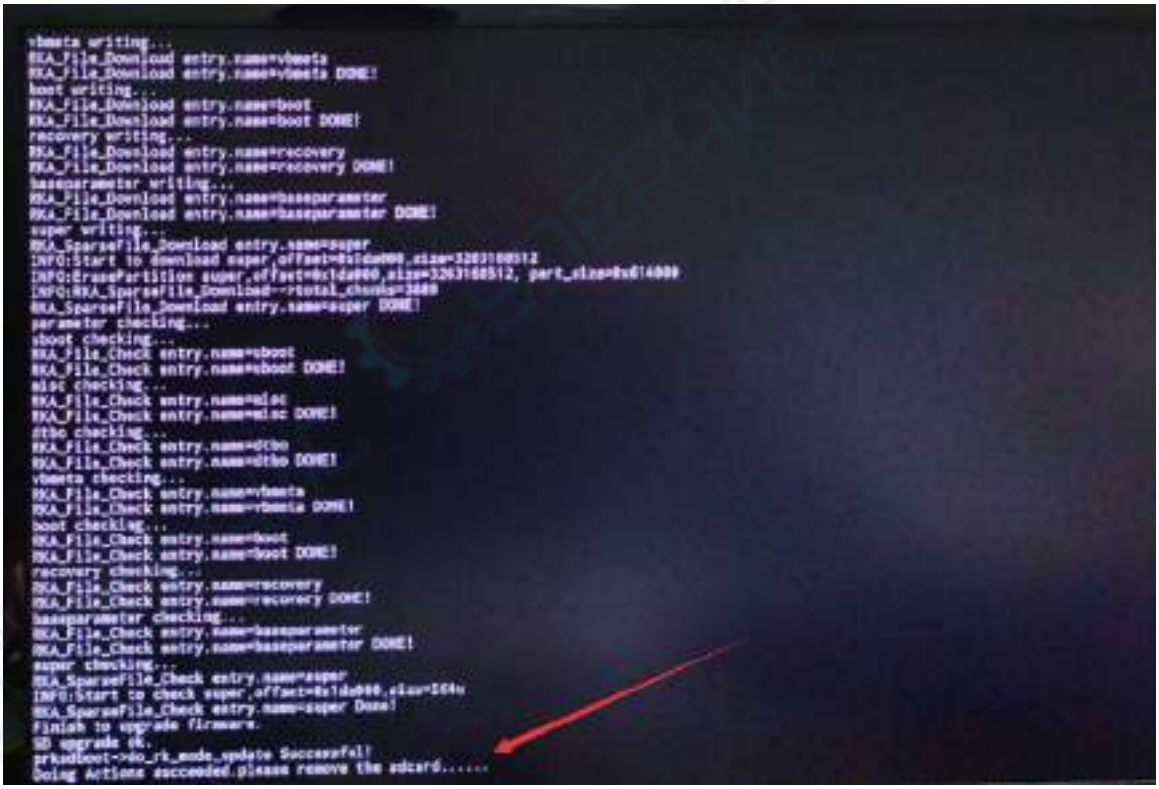
10) 烧录完成后的显示如下图所示，然后就可以退出 SDDiskTool



11) 然后把 TF 卡从电脑中拔出来插到开发板中，开发板上电启动后就会自动开始将 TF 卡中的 Android 镜像烧录到开发板的 eMMC 中

12) 如果开发板连接了 HDMI 显示器，还可以从 HDMI 显示器中看到烧录 Android 镜像到 eMMC 中的进度条

13) 当 HDMI 显示器显示如下信息时，说明烧录 Android 镜像到 eMMC 中已完成，此时就可以拔出 TF 卡，然后 eMMC 中的安卓系统就会开始启动。



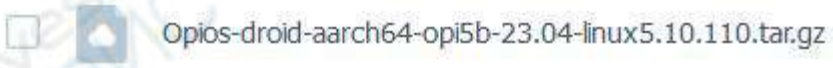
2.8. 将 Orange Pi OS (Droid)镜像烧录到 eMMC 中的方法

2.8.1. 通过 Type-C 线将 Orange Pi OS (Droid) 镜像烧录到 eMMC

注意，下面所有的操作都是在 Windows 电脑中进行的。

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_v2.96.zip**，请确保下载的 **RKDevTool** 工具的的版本为 **v2.96**

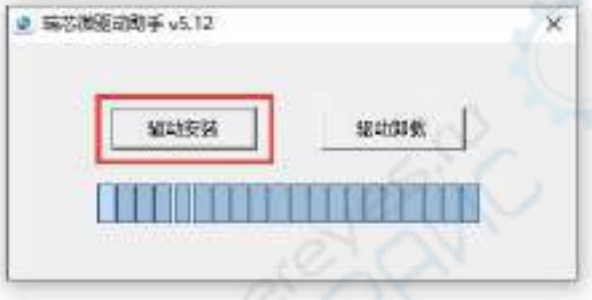


4) 然后使用解压软件解压下载的 Orange Pi OS (Droid)镜像的压缩包，解压后的文件中，以“.img”结尾的文件就是 Orange Pi OS (Droid)镜像文件，大小在 1GB 以上

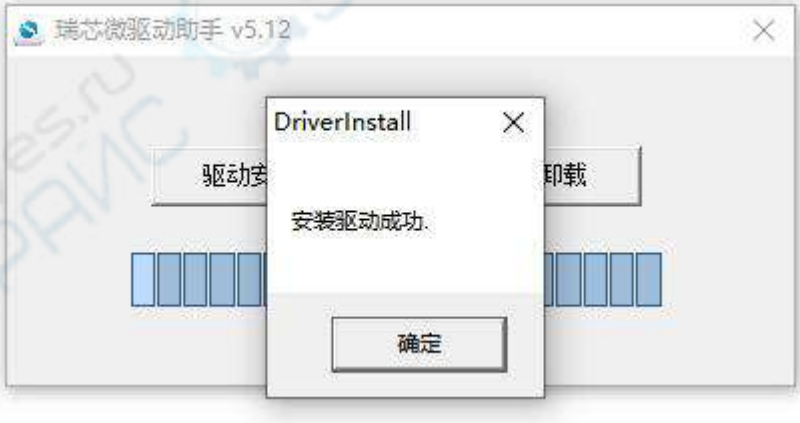
5) 然后用解压软件解压 **DriverAssitant_v5.12.zip**，再在解压后的文件夹中找到 **DriverInstall.exe** 可执行文件并打开即可

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



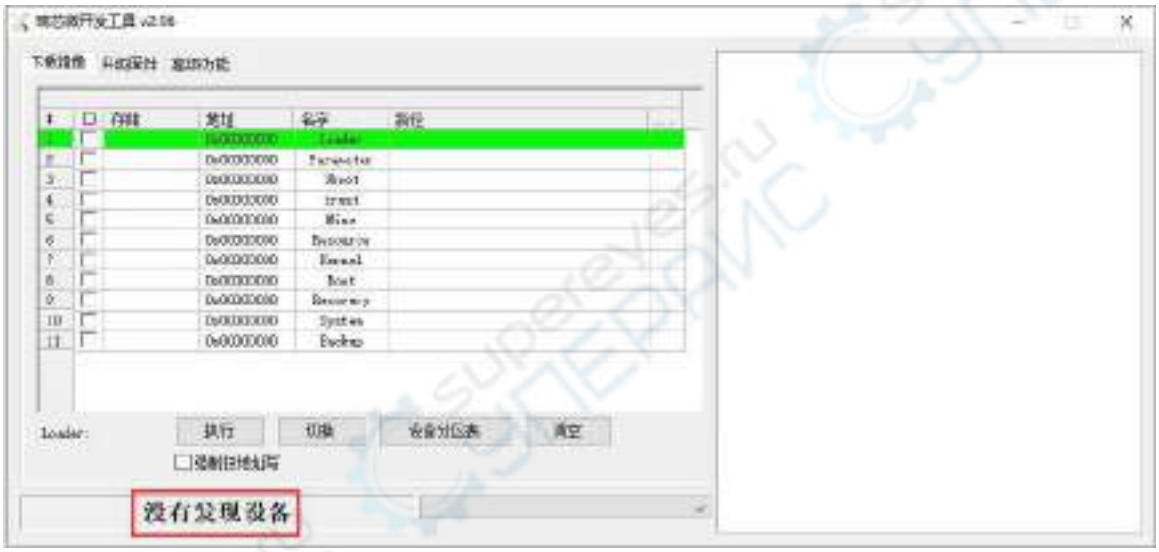
b. 等待一段时间后，会弹出窗口提示“**安装驱动成功**”，然后点击“**确定**”按钮即可



7) 然后解压 **RKDevTool_Release_v2.96.zip**，此软件无需安装，在解压后的文件夹中找到 **RKDevTool** 打开即可

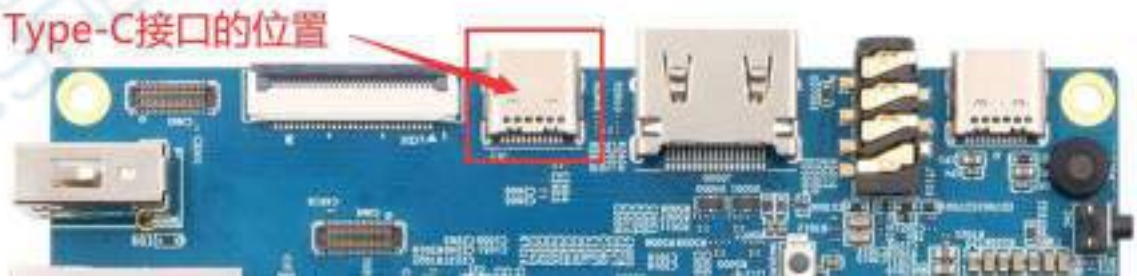
名称	修改日期	类型	大小
bin	2022/12/1 15:07	文件夹	
Language	2022/12/1 15:07	文件夹	
config.cfg	2022/3/23 9:11	CFG 文件	7 KB
config	2021/11/30 11:04	配置设置	2 KB
revision	2022/5/27 9:09	文本文档	3 KB
RKDevTool	2022/5/27 9:06	应用程序	1,212 KB
开发工具使用文档_v1.0	2021/8/27 10:28	Foxit PDF Reade...	450 KB

8) 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**”



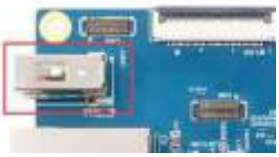
9) 然后开始烧录 Orange Pi OS (Droid)镜像到 eMMC 中

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



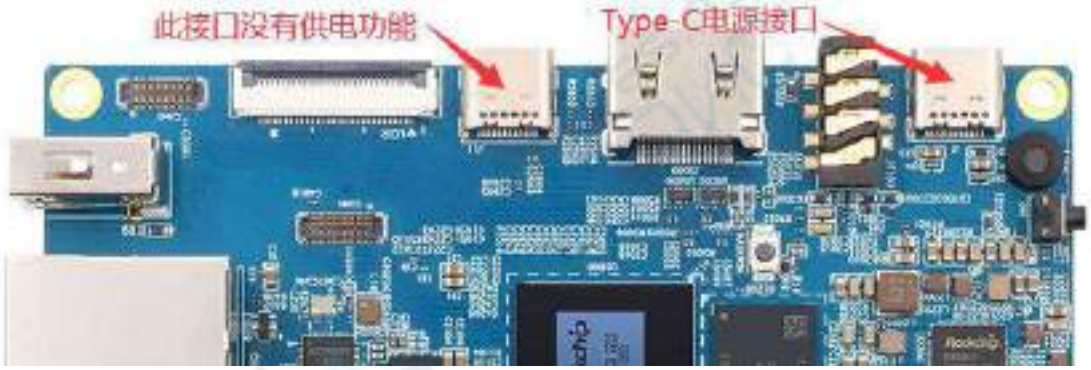
b. 确保开发板没有插入 TF 卡，没有连接电源

c. 还需确保下图位置的白色 USB2.0 接口没有插入 USB 设备

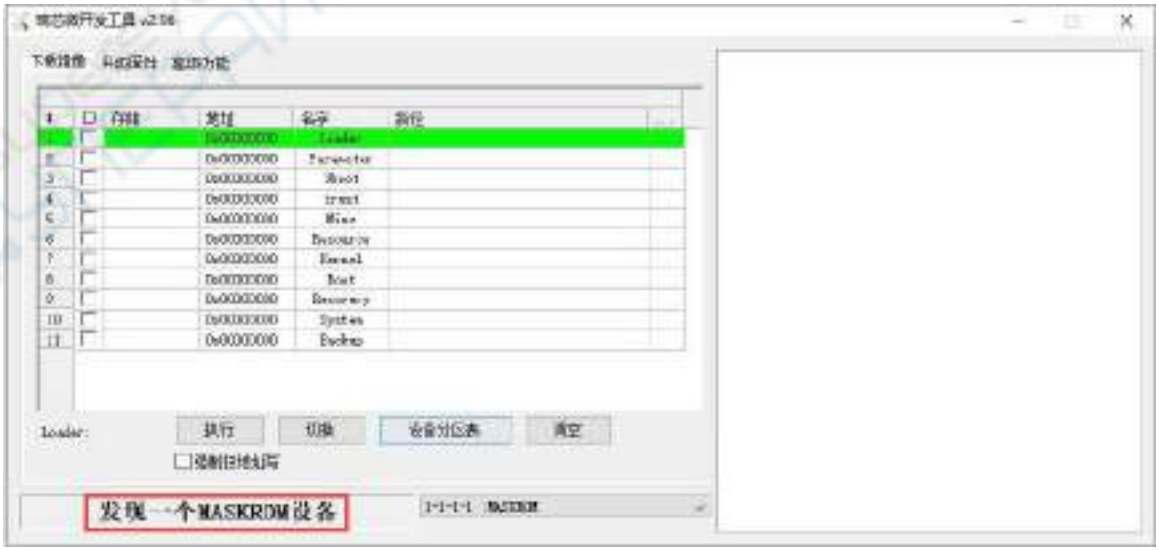


d. 然后按住开发板的 MaskROM 按键不放，MaskROM 按键在开发板的位置如下图所示：

e. 然后给开发板接上 Type-C 接口的电源，并上电



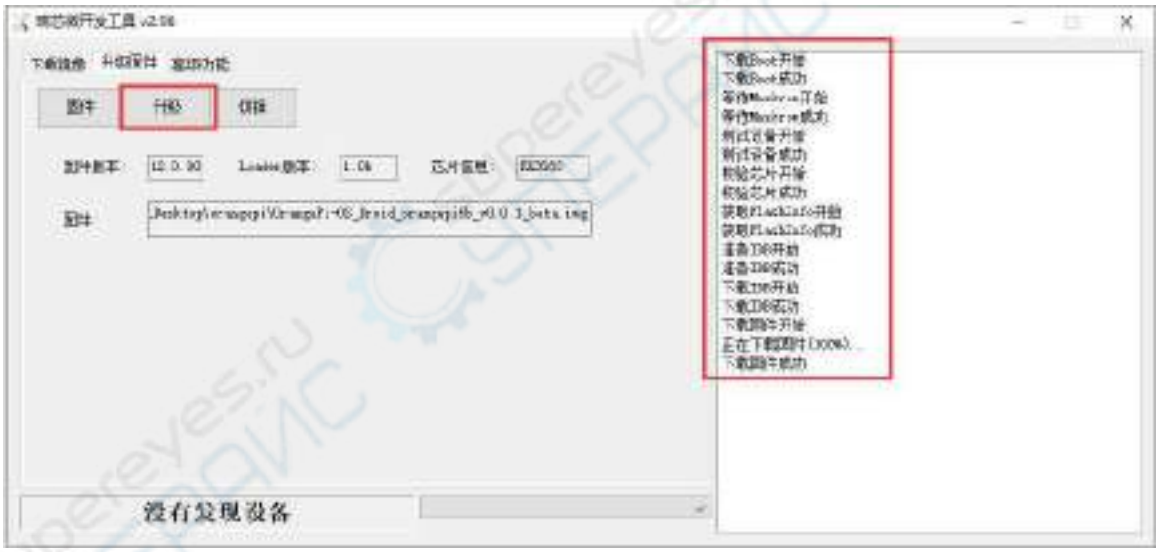
f. 如果前面的步骤顺利，此时开发板会进入 **MASKROM** 模式，在烧录工具的界面上会提示“**发现一个 MASKROM 设备**”



g. 然后点击烧录工具的“**升级固件**”一栏

h. 接着点击“**固件**”按钮选择需要烧录的 Orange Pi OS (Droid)镜像的路径

i. 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 Orange Pi OS (Droid)系统会自动启动。



2.8.2. 通过 TF 卡将 Orange Pi OS (Droid)镜像烧录到 eMMC

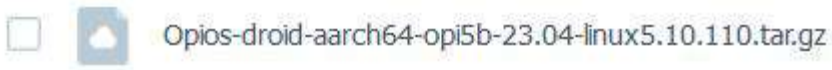
注意，下面所有的操作都是在 Windows 电脑中进行的。

1) 首先准备一张 8GB 或更大容量的 TF 卡，TF 卡的传输速度必须为 **class10** 或以上，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

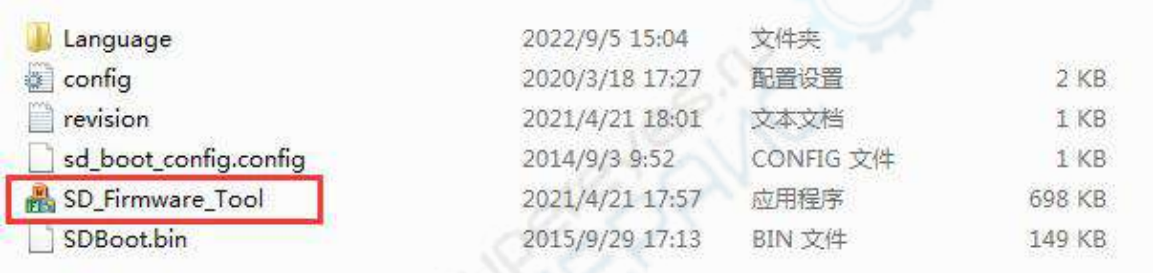
3) 然后从Orange Pi 的资料下载页面下载SDDiskTool烧写工具，**请确保SDDiskTool工具的版本为最新的 v1.72**

4) 然后从 Orange Pi 的资料下载页面下载 Orange Pi OS (Droid)的镜像



5) 然后使用解压软件解压下载的 Orange Pi OS (Droid)镜像的压缩包，解压后的文件中，以“.img”结尾的文件就是 Orange Pi OS (Droid)镜像文件，大小在 1GB 以上

6) 然后使用解压软件解压 **SDDiskTool_v1.72.zip**，此软件无需安装，在解压后的文件夹中找到 **SD_Firmware_Tool.exe** 打开即可



7) 打开 SDDiskTool 后，如果 TF 卡识别正常，会在“**选择可移动磁盘设备**”一栏中显示插入的磁盘设备，**请务必确认显示的磁盘设备和你想烧录的 TF 卡的盘符是一致的**，如果没有显示可以尝试拔插下 TF 卡

8) 确认完盘符后，可以先格式化下 TF 卡，点击 SDDiskTool 中的**恢复磁盘**按钮即可，也可使用前面提到的 **SD Card Formatter** 进行 TF 卡的格式化

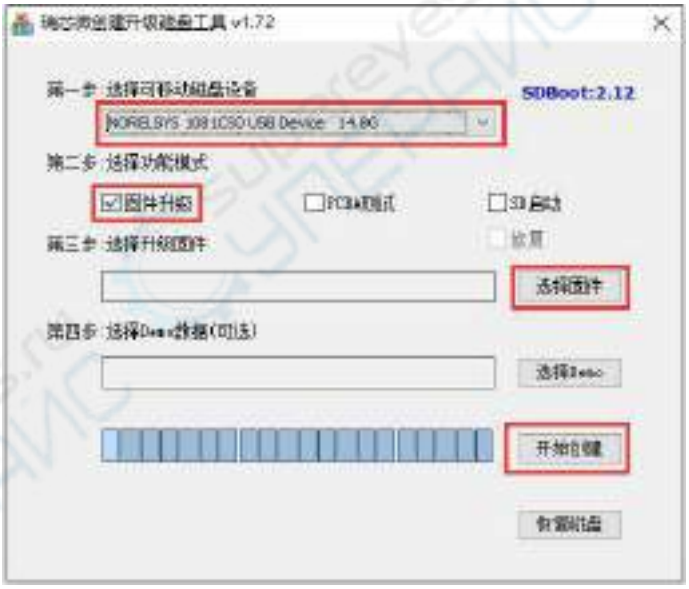
9）然后开始将 Orange Pi OS (Droid)镜像写入 TF 卡

a. 首先在**“选择可移动磁盘设备”**下面确认显示的盘符为 TF 卡对应的盘符

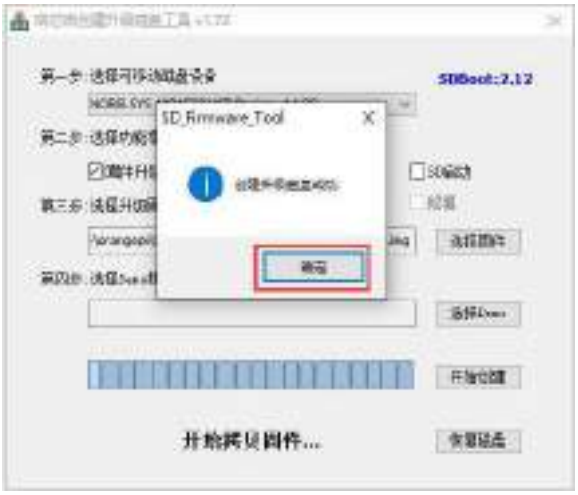
b. 然后在“**选择功能模式**”中选择**“固件升级”**

c. 然后在“**选择升级固件**”一栏中选择 Orange Pi OS (Droid)固件的路径

d. 最后点击“**开始创建**”按钮就会开始烧录



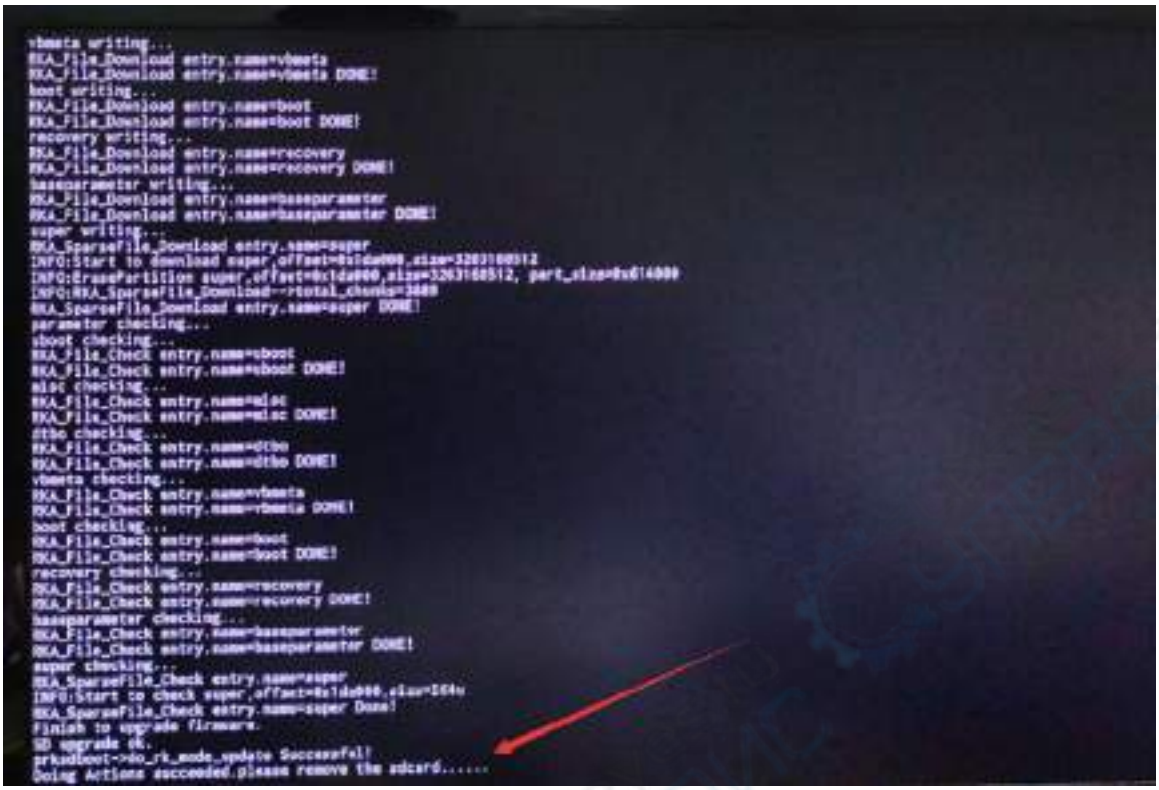
10) 烧录完成后的显示如下图所示，然后就可以退出 SDDiskTool



11) 然后把 TF 卡从电脑中拔出来插到开发板中，开发板上电启动后就会自动开始将 TF 卡中的 Orange Pi OS (Droid)镜像烧录到开发板的 eMMC 中

12) 如果开发板连接了 HDMI 显示器，还可以从 HDMI 显示器中看到烧录 Orange Pi OS (Droid)镜像到 eMMC 中的进度条

13) 当 HDMI 显示器显示如下信息时，说明烧录 Orange Pi OS (Droid)镜像到 eMMC 中已完成，此时就可以拔出 TF 卡，然后 eMMC 中的 Orange Pi OS (Droid)系统就会开始启动。



2.9. 启动香橙派开发板

1) 开发板有板载 eMMC，里面默认烧录有 Orange Pi OS (Droid)镜像，刚拿到开发板可以直接使用 eMMC 中的镜像进行启动和全功能测试。

2) 如果需要使用 linux 镜像，可以将烧录好 linux 镜像的 TF 卡插入香橙派开发板的 TF 卡插槽中。

3) 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。如果有 Type-C 转 HDMI 的线，也可以通过 Type-C 接口来显示开发板的系统界面。

4) 接上 USB 鼠标和键盘，用于控制香橙派开发板。

5) 开发板有以太网口，可以插入网线用来上网。

6) 连接一个 5V/4A 的 USB Type-C 接口的**高品质**的电源适配。

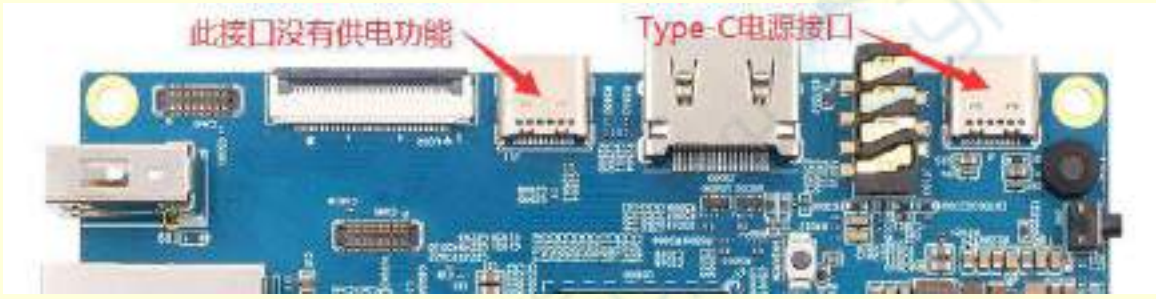
切记不要插入电压输出大于 5V 的电源适配器，会烧坏开发板。

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。

开发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，中间那个是没有供电功能的，请别接错了。



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

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

2.10. 调试串口的使用方法

2.10.1. 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 模块模块，请不要使用 CP2102 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) 开发板的调试串口 GND、RXD 和 TXD 引脚的对应关系如下图所示

3) USB 转 TTL 模块 GND、TXD 和 RXD 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

2.10.2. 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
```

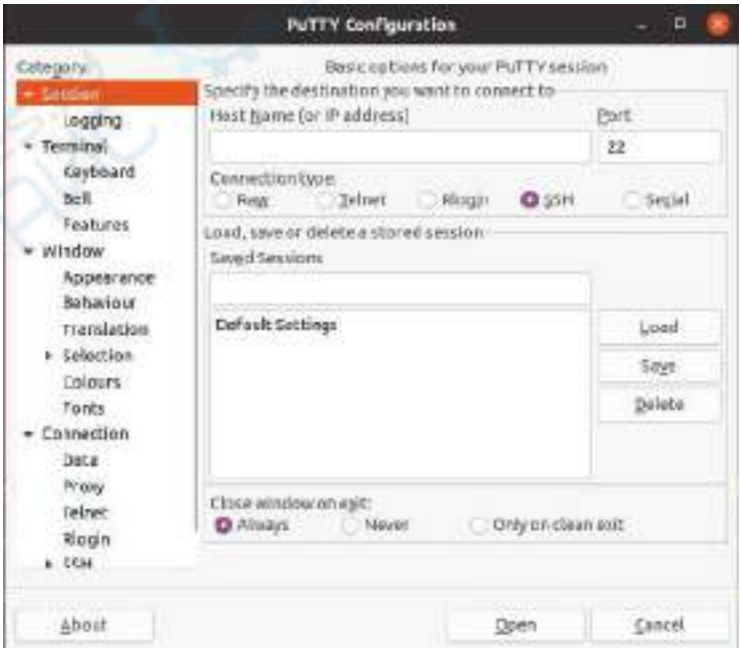
2) 然后使用下面的命令在 Ubuntu PC 上安装下 putty

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

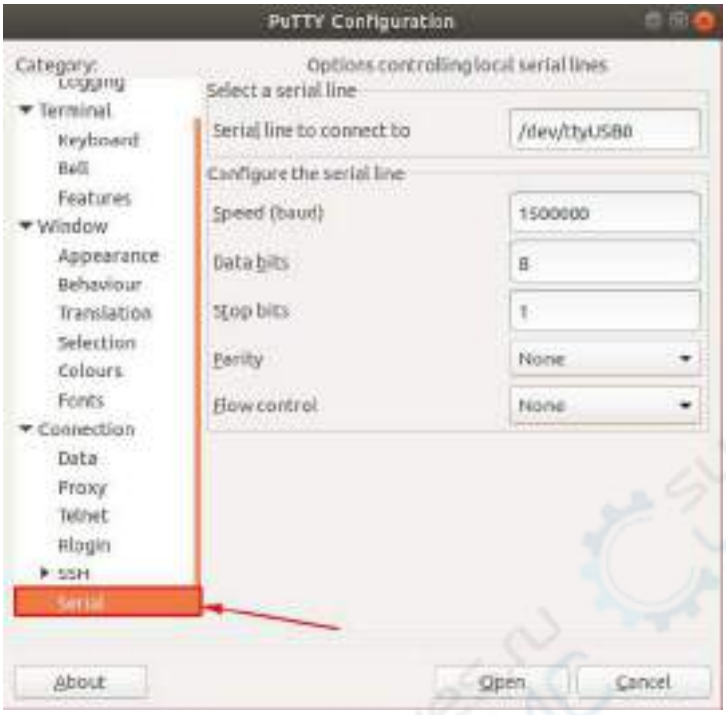
3) 然后运行 putty，**记得加 sudo 权限**

```
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）

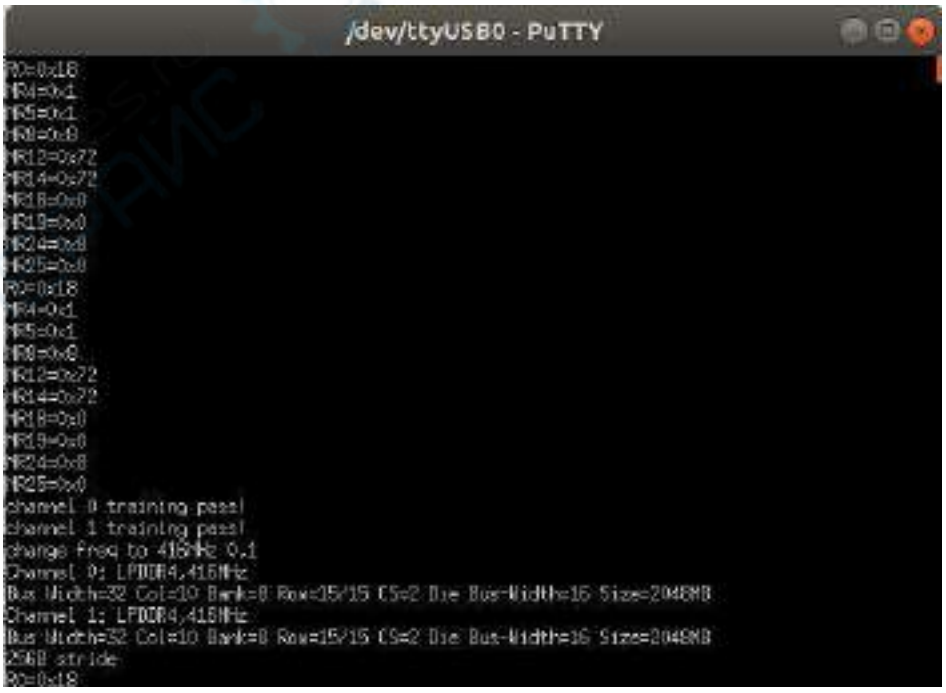
c. 设置 Flow control 为 None

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



2.10.3. Windows 平台调试串口的使用方法

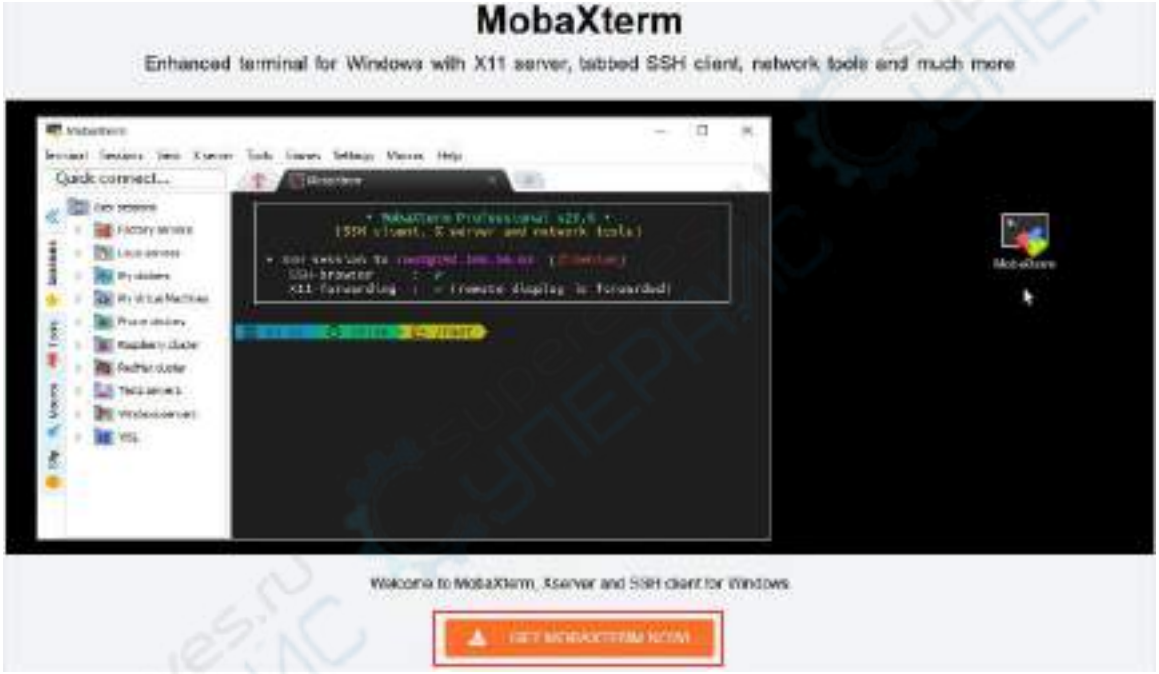
Windows 下可以使用的串口调试软件有很多，如 SecureCRT、MobaXterm 等，下面演示 MobaXterm 的使用方法，这款软件有免费版本，无需购买序列号即可使用。

1) 下载 MobaXterm

a. 下载 MobaXterm 网址如下

https://mobaxterm.mobatek.net

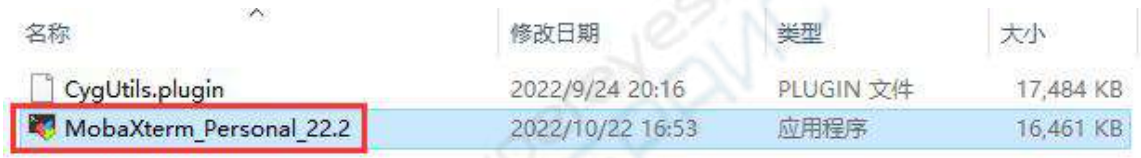
b. 进入 MobaXterm 下载网页后点击 **GET XOBATERM NOW!**



c. 然后选择下载 Home 版本

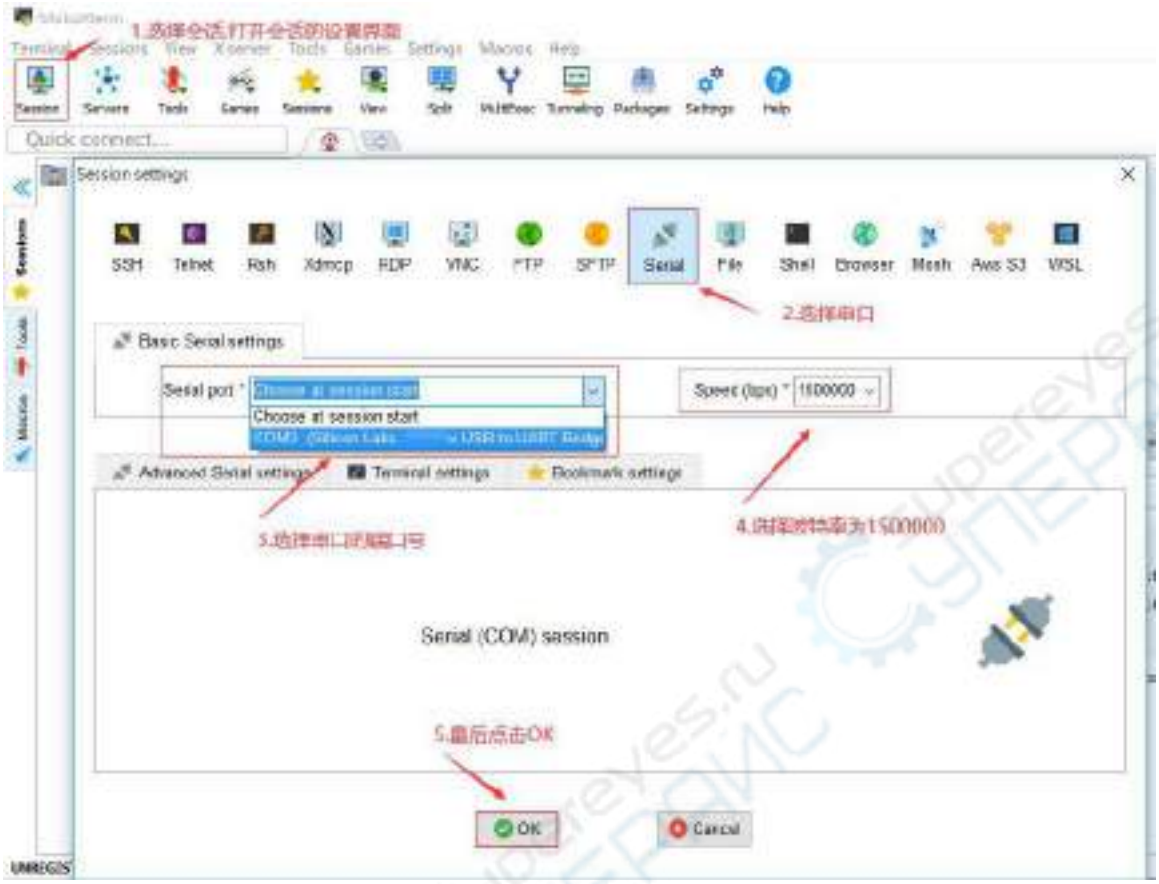
d. 然后选择 Portable 便携式版本，下载完后无需安装，直接打开就可以使用

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

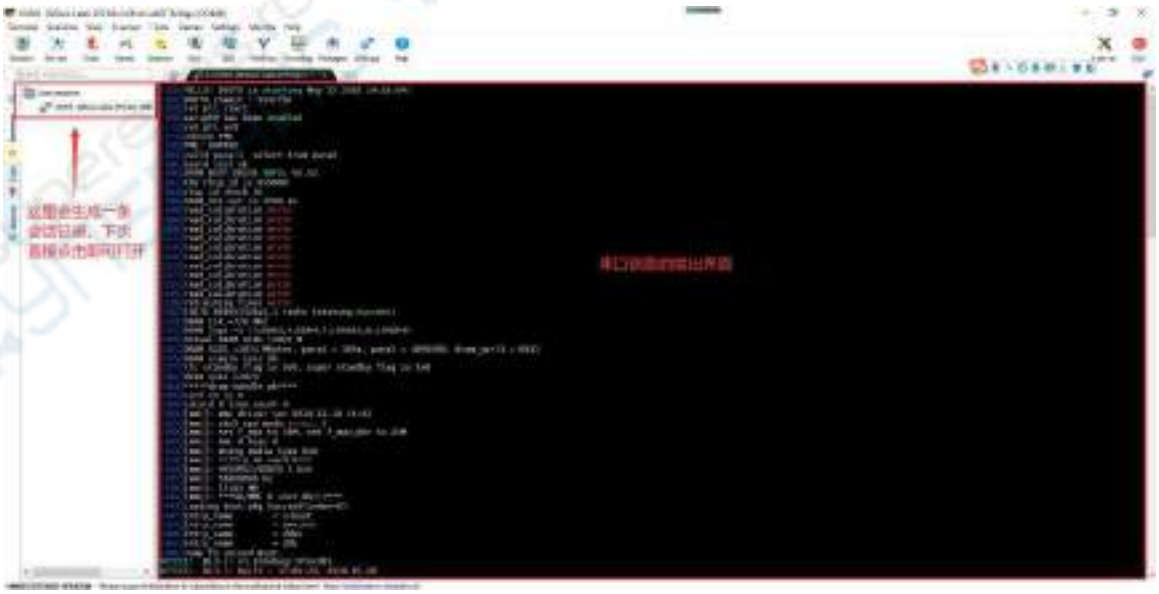


3) 打开软件后，设置串口连接的步骤如下

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



2.11. 使用开发板 26pin 接口中的 5v 引脚供电说明

我们推荐的开发板的供电方式是使用 5V/4A 的 Type C 接口的电源线插到开发板的 Type-C 电源接口来供电的。如果需要使用 26pin 接口中的 5V 引脚来给开发板供电，请确保使用的电源线和电源适配器能满足开发板的供电需求。如果有使用不稳定的情况，请换回 Type-C 电源供电。

1) 首先需要准备一根下图所示的电源线

上图所示的电源线在淘宝可以买到，请自行搜索购买。

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

a. 上图所示的电源线 USB A 口需要插到 5V/4A 的电源适配器接头上（**请不要插到电脑的 USB 接口来供电**）

b. 红色的杜邦线需要插到开发板 26pin 的 5V 引脚上

c. 黑色的杜邦线需要插到 26pin 接口的 GND 引脚上

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

3. Linux 系统使用说明

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

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

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

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

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

本章内容是基于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

Ubuntu 20.04 - Focal	Linux5.10	支持	支持
Ubuntu 22.04 - Jammy	Linux5.10	支持	支持

3.2. linux 系统适配情况

功能	Linux5.10 驱动	Debian11	Ubuntu20.04	Ubuntu22.04
USB2.0x2	OK	OK	OK	OK
USB3.0x1	OK	OK	OK	OK
USB Type-C 3.0	OK	OK	OK	OK
DP 显示	OK	OK	OK	OK
eMMC	OK	OK	OK	OK
AP6275P-WIFI	OK	OK	OK	OK
AP6275P-蓝牙	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
CAN（26pin）	OK	OK	OK	OK
PWM（26pin）	OK	OK	OK	OK
3pin 调试串口	OK	OK	OK	OK
TF 卡启动	OK	OK	OK	OK
HDMI 视频	OK	OK	OK	OK
HDMI 音频	OK	OK	OK	OK
OV13850 摄像头	OK	OK	OK	OK
OV13855 摄像头	OK	OK	OK	OK
LCD1	OK	OK	OK	OK
LCD2	OK	OK	OK	OK
千兆网口	OK	OK	OK	OK
网口状态灯	OK	OK	OK	OK
MIC	OK	OK	OK	OK
耳机播放	OK	OK	OK	OK
耳机录音	OK	OK	OK	OK
LED 灯	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. 本手册 linux 命令格式说明

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

```
```

如下所示，黄色方框里内容表示需要特别注意的内容，这里面的命令除外。

```
```

2) 命令前面的提示符类型说明

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

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

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

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

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

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

3) 哪些是需要输入的命令？

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

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

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

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

3.4. linux 系统登录说明

3.4.1. linux 系统默认登录账号和密码

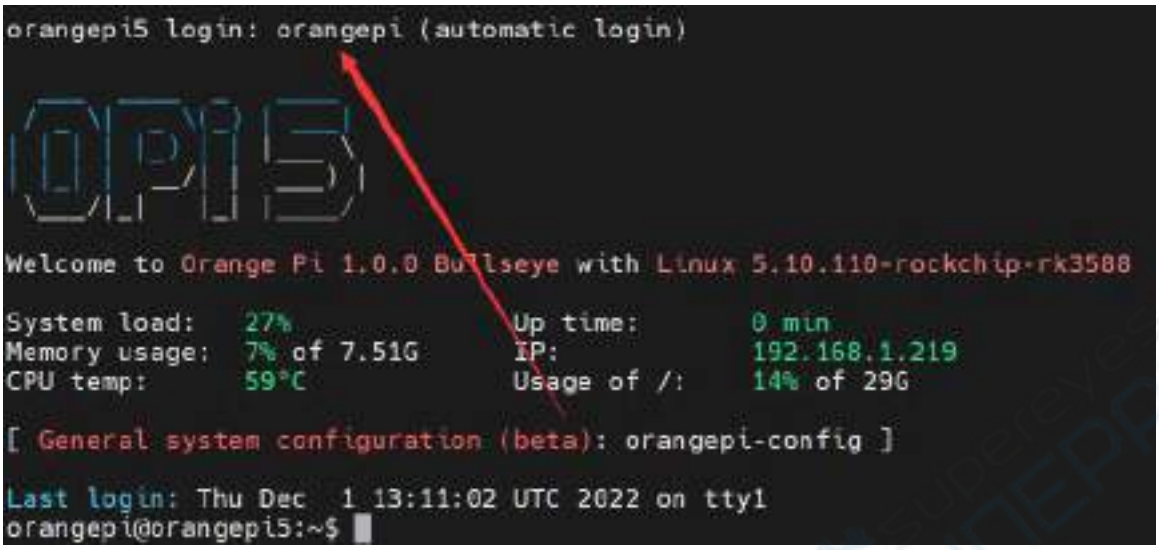
账号	密码
root	**orangepi**
orangepi	**orangepi**

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

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

3.4.2. 设置 linux 系统终端自动登录的方法

1) 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
```

3.4.3. linux 桌面版系统自动登录说明

1) 桌面版系统启动后会自动登录进入桌面，无需输入密码

2) 运行下面的命令可以禁止桌面版系统自动登录桌面

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

3) 然后重启系统就会出现登录对话框，此时需要输入密码才能进入系统

3.4.4. Linux 桌面版系统 root 用户自动登录的设置方法

1) 执行下面的命令可以设置桌面版系统使用 root 用户自动登录

```
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

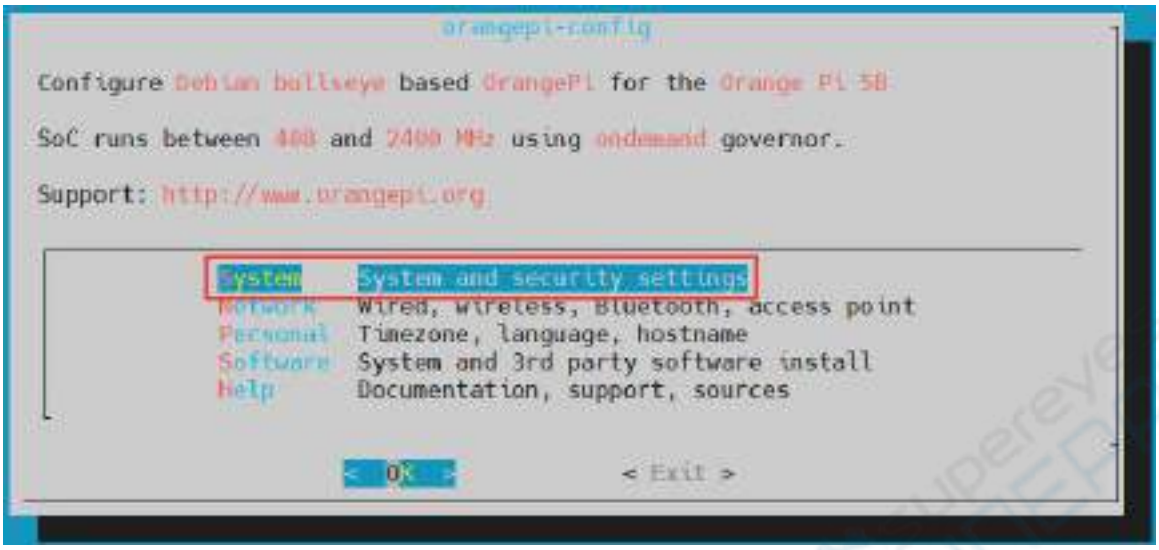
```
orangepi@orangepi:~$ sudo desktop_login.sh orangepi
```

3.4.5. Linux 桌面版系统禁用桌面的方法

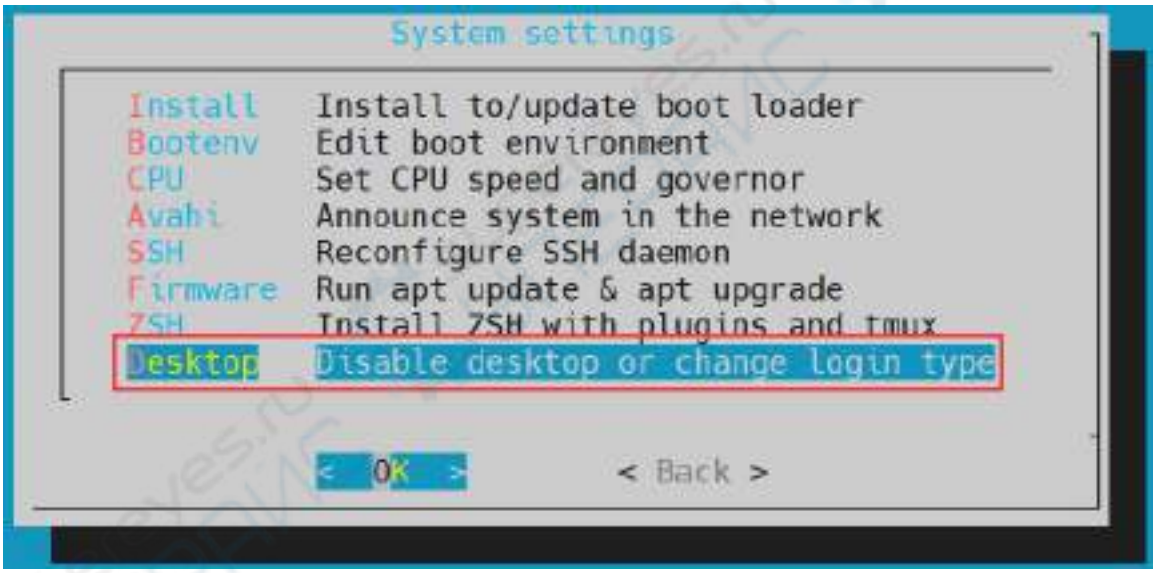
1) 首先在命令行中输入下面的命令，**请记得加 sudo 权限**

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

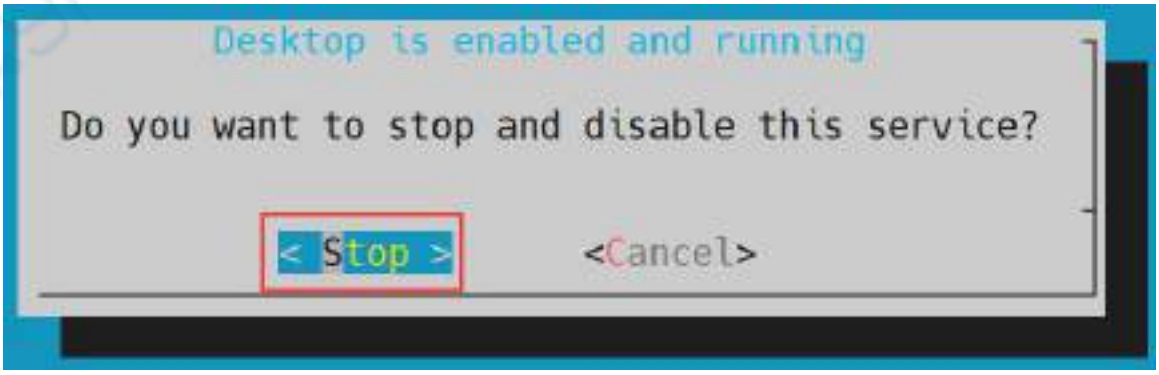
2) 然后选择 **System**



3) 然后选择 **Desktop**



4) 然后选择**<Stop>**



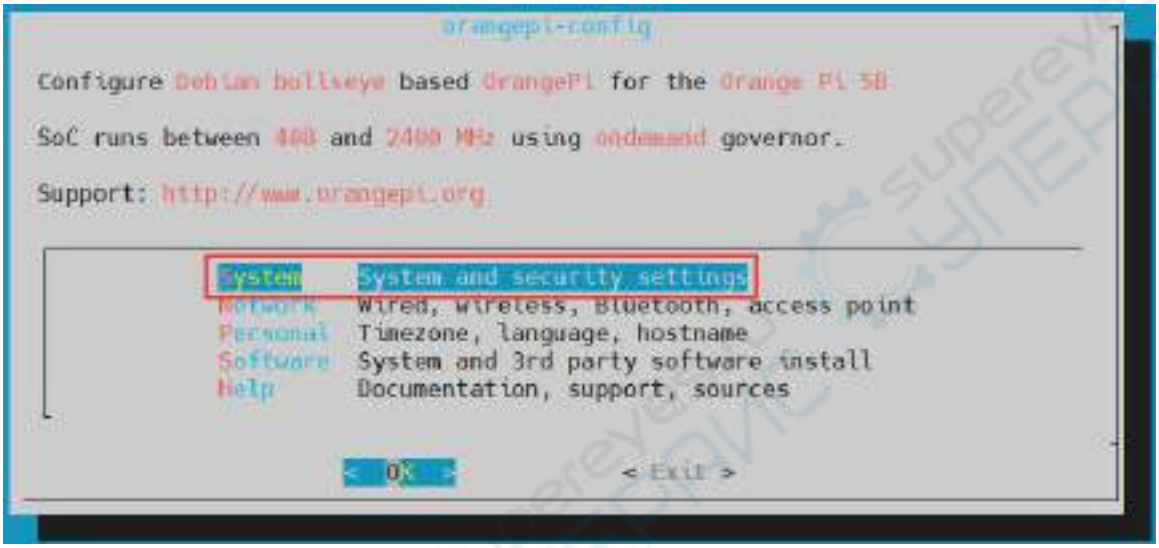
5) **然后重启 Linux 系统就会发现不会显示桌面了**

6) 重新打开桌面的步骤如下所示：

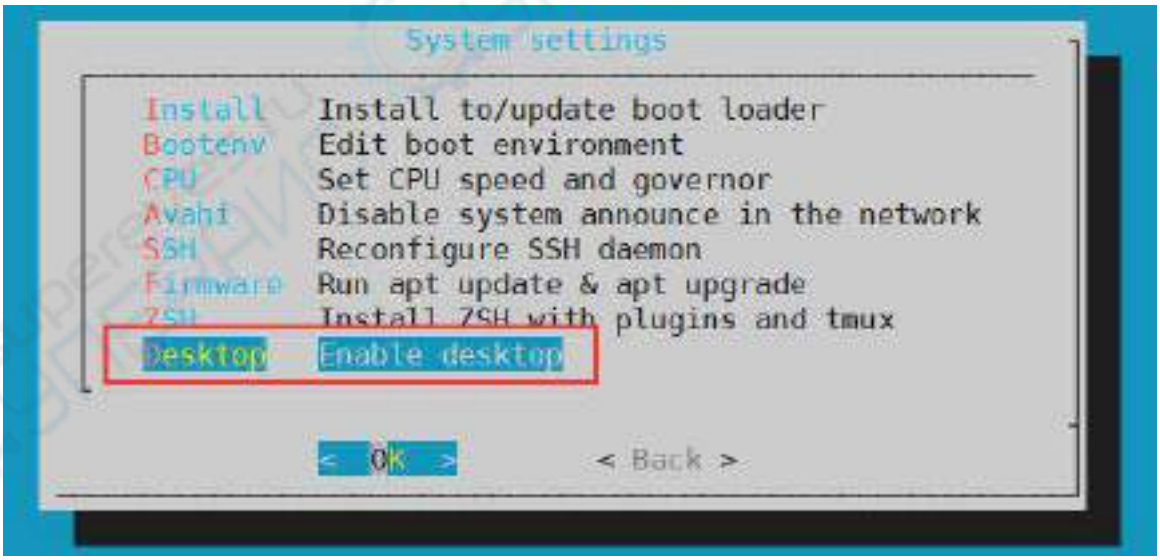
a. 首先在命令行中输入下面的命令，**请记得加 sudo 权限**

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

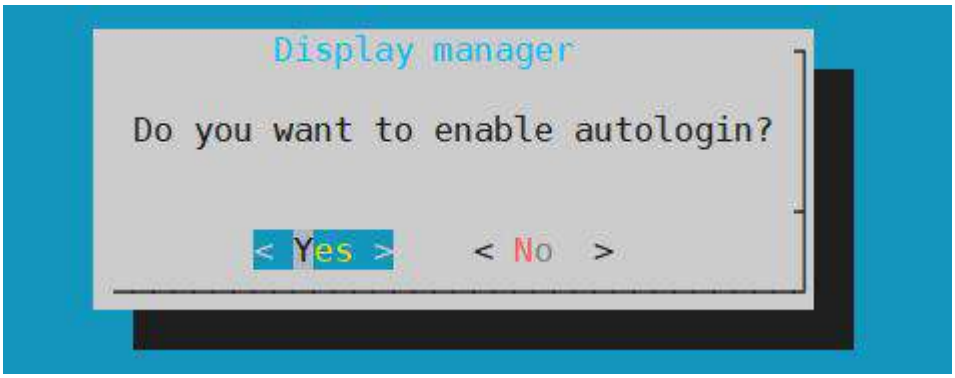
b. 然后选择 **System**



c. 然后选择 **Desktop Enable desktop**



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



e. 选择完后 HDMI 显示器就会显示桌面了

3.5. 板载 LED 灯测试说明

1) 开发板上有两个 LED 灯，一个绿灯，一个红灯，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。

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
```

3.6. 网络连接测试

3.6.1. 以太网口测试

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

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

3) 在开发板的 Linux 系统中查看 IP 地址的命令如下所示

```
orangepi@orangepi:~$ ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 4a:fe:2b:3d:17:1c brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.150/24 brd 192.168.1.255 scope global dynamic noprefixroute eth0
       valid_lft 43150sec preferred_lft 43150sec
    inet6 fe80::9a04:3703:faed:23be/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

当使用 ifconfig 查看 IP 地址时，如果提示下面的信息，是因为没有加 sudo 导致的，正确的命令是：sudo ifconfig

```
orangepi@orangepi:~$ ifconfig
Command 'ifconfig' is available in the following places
 * /sbin/ifconfig
 * /usr/sbin/ifconfig
The command could not be located because '/sbin:/usr/sbin' is not included in the PATH environment variable.
This is most likely caused by the lack of administrative privileges associated with your user account.
ifconfig: command not found
```

开发板启动后查看 IP 地址有三种方法：

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

A）首先检查 Linux 系统是否已经正常启动，如果开发板的绿灯在闪烁了，一般是正常启动了，如果只亮红灯，说明系统都没正常启动；

B）检查网线有没有插紧，或者换根网线试下；

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

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

另外需要注意的是开发板 DHCP 自动分配 IP 地址是不需要任何设置的。

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

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

3.6.2. WIFI 连接测试

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

3.6.2.1. 服务器版镜像通过命令连接 WIFI

当开发板没有连接以太网，没有连接 HDMI 显示器，只连接了串口时，推荐使

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

1) 先登录 linux 系统，有下面三种方式

a. 如果开发板连接了网线，可以通过 **ssh 远程登录 linux 系统**

a. 如果开发板连接好了调试串口，可以使用串口终端登录 linux 系统

b. 如果连接了开发板到HDMI显示器，可以通过HDMI显示的终端登录到linux系统

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

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

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

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

a. **wifi_name** 需要换成想连接的 WIFI 热点的名字

b. **wifi_passwd** 需要换成想连接的 WIFI 热点的密码

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

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

```
orangepi@orangepi:~$ ip addr show wlan0
11: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 23:8c:d6:ae:76:bb brd ff:ff:ff:ff:ff:ff
```

```
    inet 192.168.1.11/24 brd 192.168.1.255 scope global dynamic noprefixroute wlan0
       valid_lft 259192sec preferred_lft 259192sec
    inet6 240e:3b7:3240:c3a0:c401:a445:5002:ccdd/64 scope global dynamic
noprefixroute
       valid_lft 259192sec preferred_lft 172792sec
    inet6 fe80::42f1:6019:a80e:4c31/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

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

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

3.6.2.2. 服务器版镜像通过图形化方式连接 WIFI

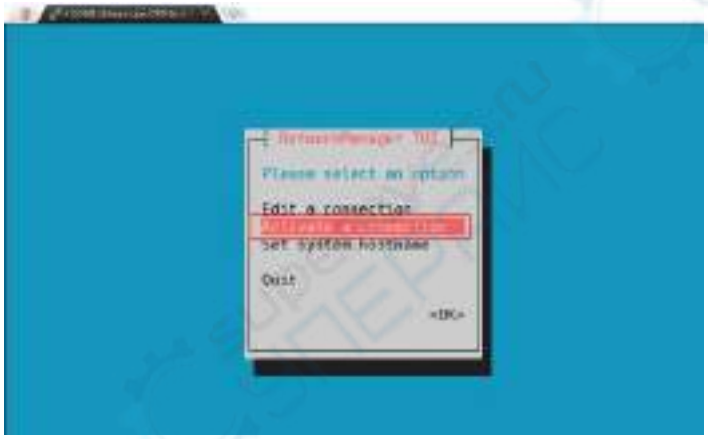
1) 先登录 linux 系统，有下面三种方式
   a. 如果开发板连接了网线，可以通过 **ssh 远程登录 linux 系统**
   b. 如果开发板连接好了调试串口，可以使用串口终端登录 linux 系统（串口软件请使用 MobaXterm，使用 minicom 无法显示图形界面）
   c. 如果连接了开发板到HDMI显示器，可以通过HDMI显示的终端登录到linux系统

2) 然后在命令行中输入 nmtui 命令打开 wifi 连接的界面

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

3) 输入 nmtui 命令打开的界面如下所示

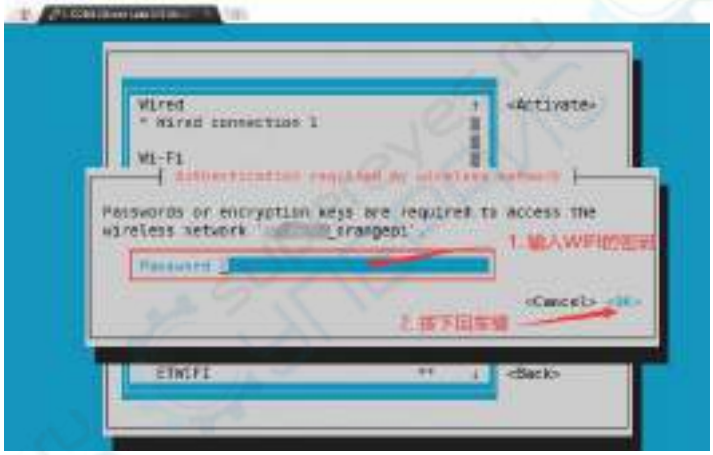
4) 选择 **Activate a connect** 后回车



5) 然后就能看到所有搜索到的 WIFI 热点

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

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



8) WIFI 连接成功后会在已连接的 WIFI 名称前显示一个“*”

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

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

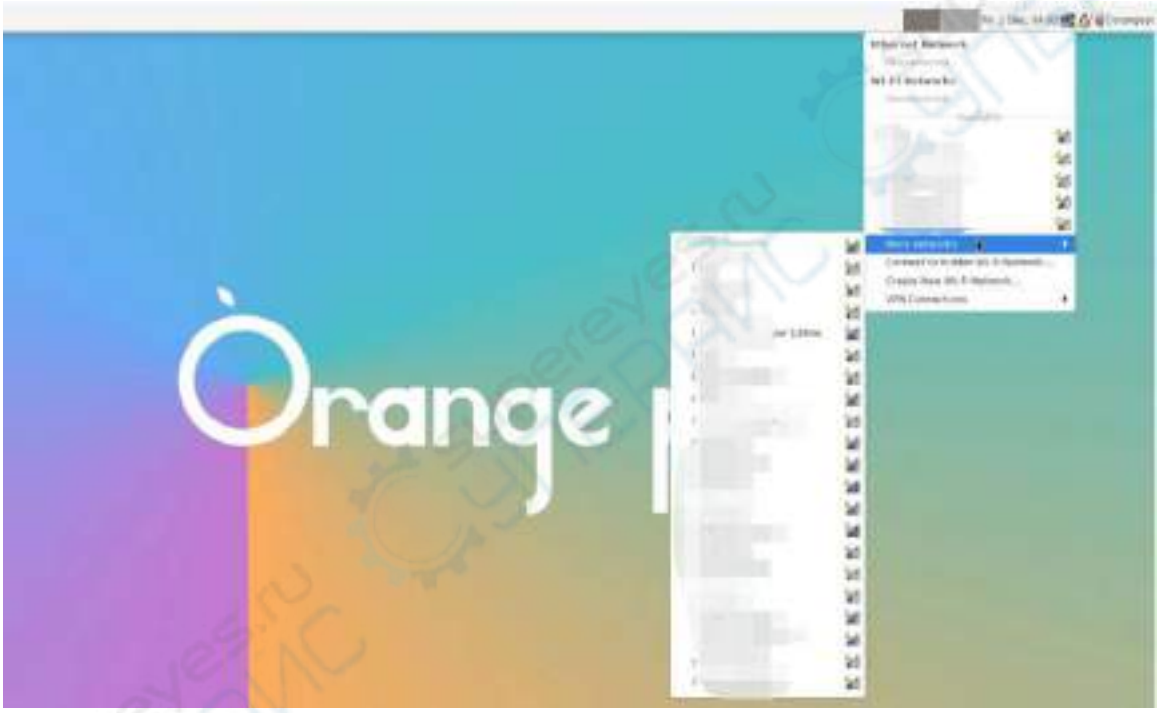
10) 使用 **ping** 命令可以测试 wifi 网络的连通性，**ping** 命令可以通过 **Ctrl+C** 快捷键来中断运行

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

3.6.2.3. 桌面版镜像的测试方法

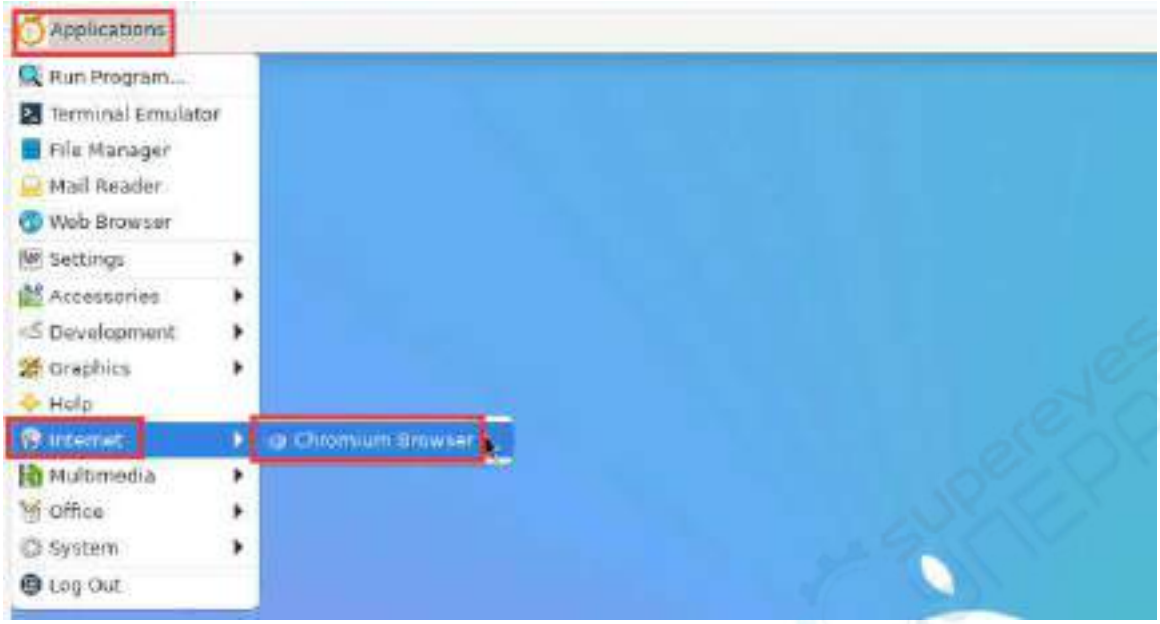
1) 点击桌面右上角的网络配置图标（测试 WIFI 时请不要连接网线）

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



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

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



5) 打开浏览器后如果能打开其他网页说明 WIFI 连接正常

3.6.3. 设置静态 IP 地址的方法

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

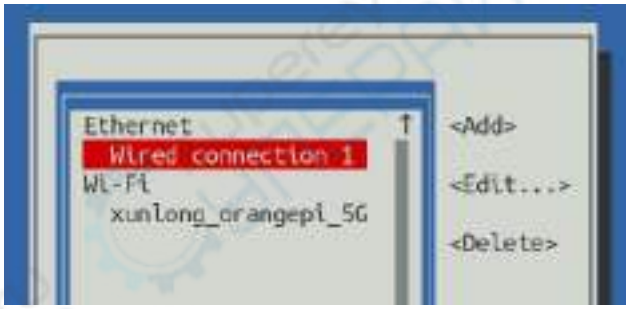
3.6.3.1. 使用 nmtui 命令来设置静态 IP 地址

1) 首先运行 **nmtui** 命令

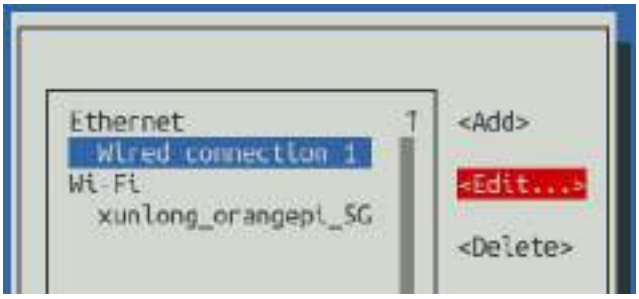
```
orangepi@orangepi:~$ nmtui
```

2) 然后选择 **Edit a connection** 并按下回车键

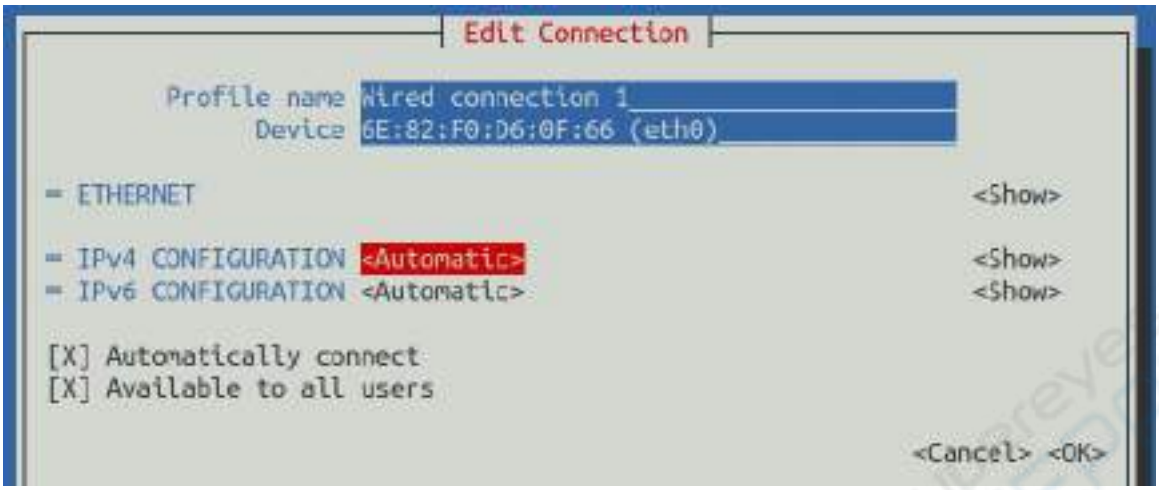
3) 然后选择需要设置静态 IP 地址的网络接口，比如设置 **Ethernet** 接口的静态 IP 地址选择 **Wired connection 1** 就可以了



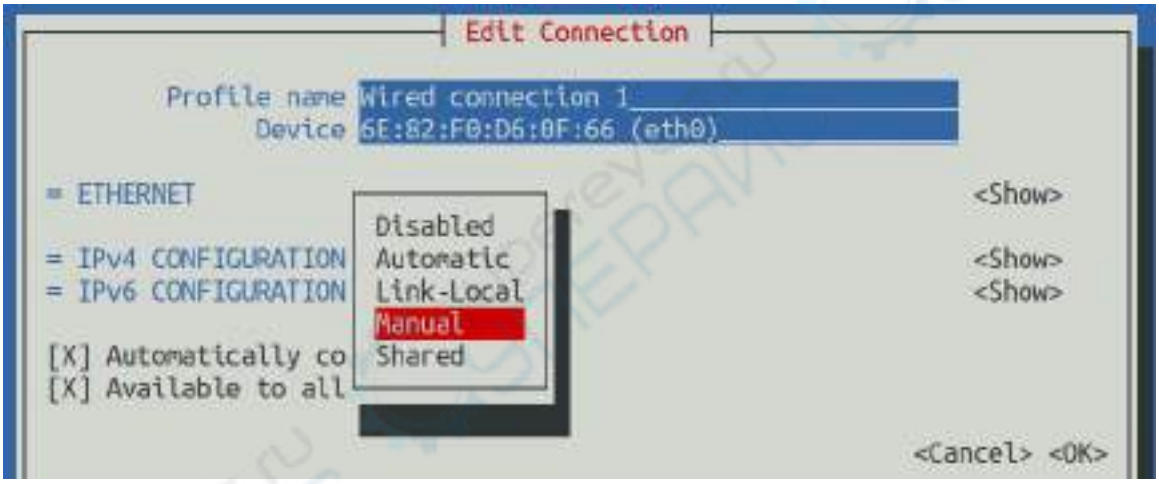
4) 然后通过 **Tab** 键选择 **Edit** 并按下回车键



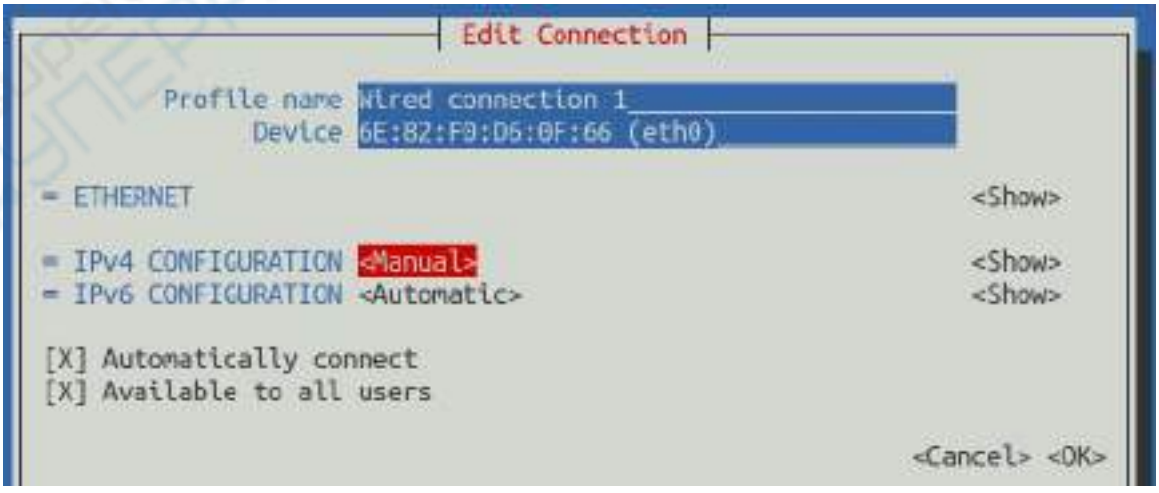
5) 然后通过 Tab 键将光标移动到下图所示的**<Automatic>**位置进行 IPv4 的配置



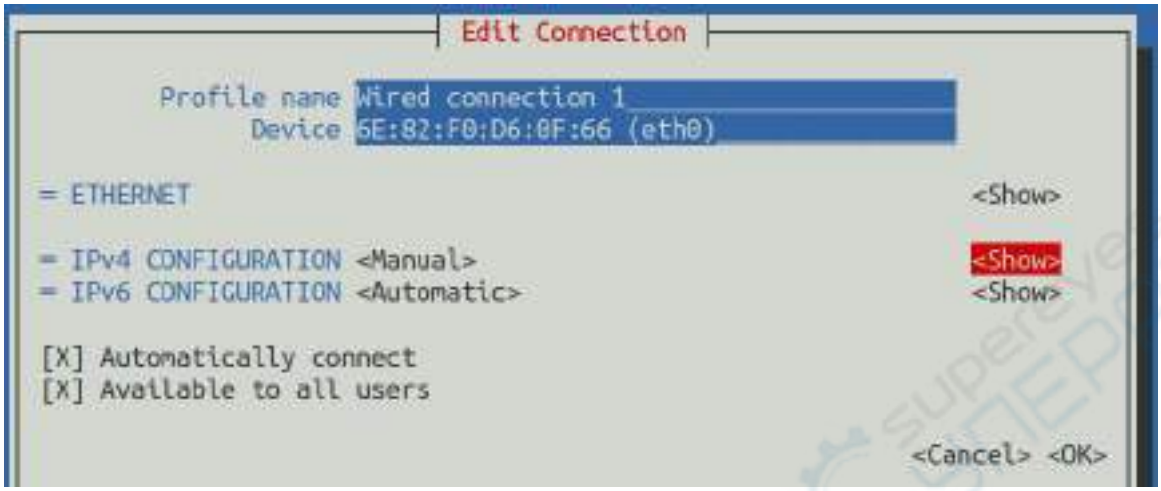
6) 然后回车，通过上下方向键选择 **Manual**，然后回车确定



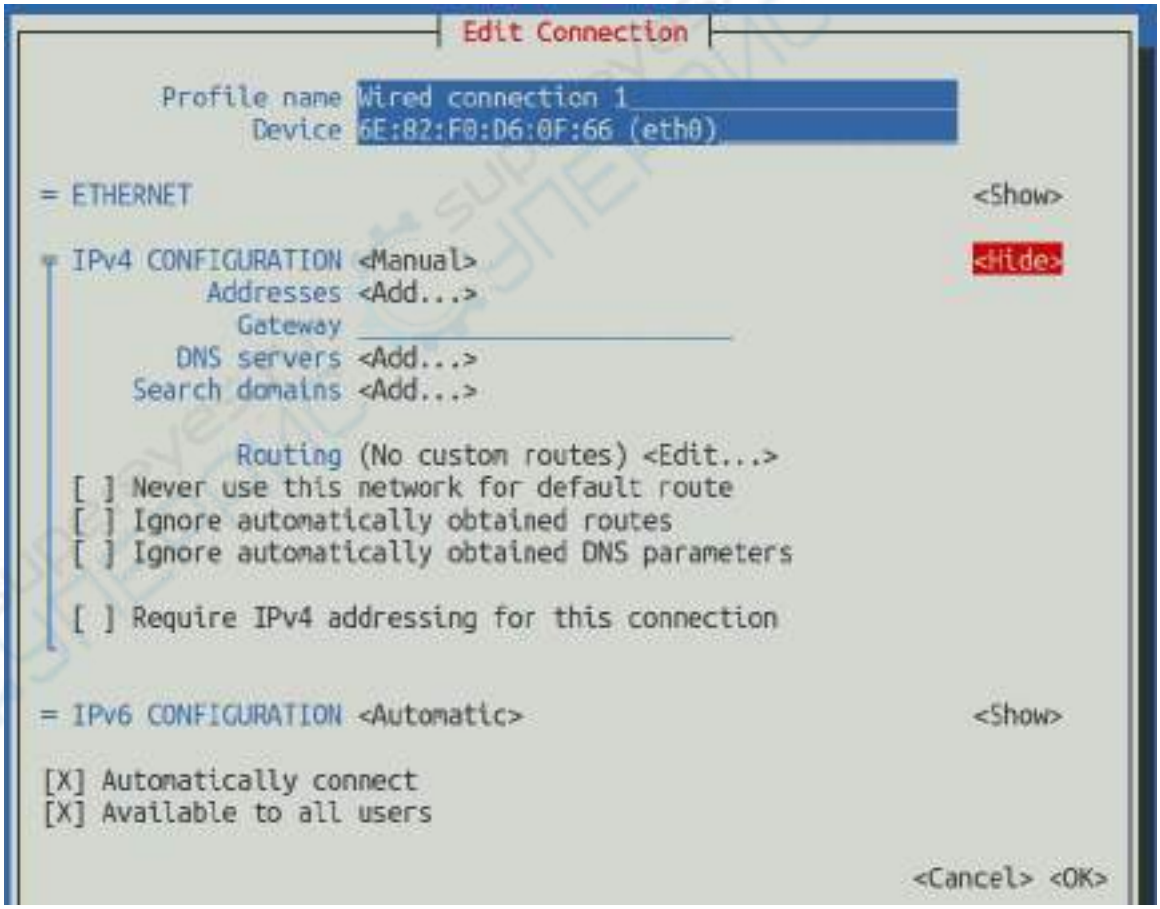
7) 选择完后的显示如下图所示



8) 然后通过 Tab 键将光标移动到**<Show>**

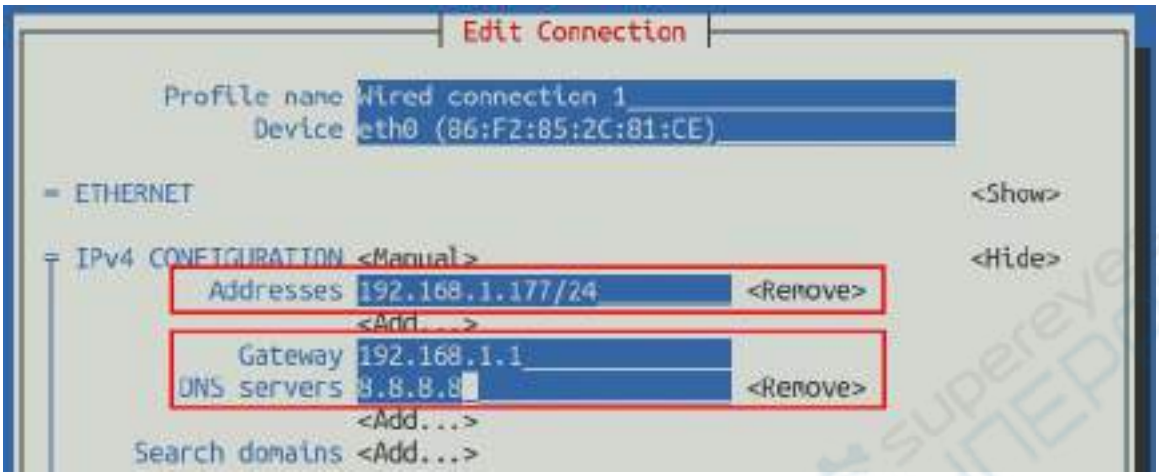


9) 然后回车，回车后会弹出下面的设置界面

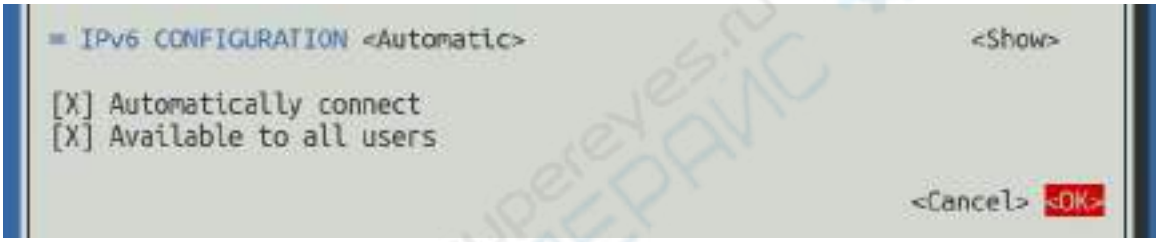


10) 然后就可以在下图所示的位置设置 IP 地址(Addresses)、网关(Gateway)和 DNS

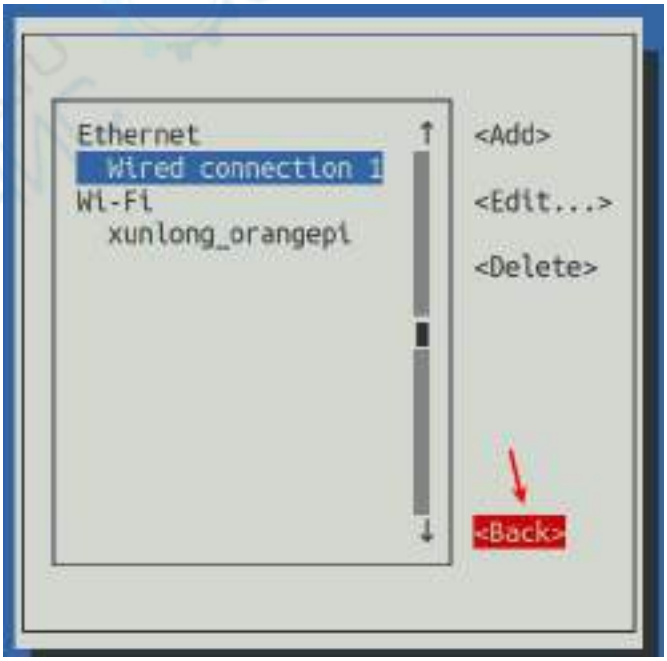
服务器的地址（里面还有很多其他设置选项，请自行探索），**请根据自己的具体需求来设置，下图中设置的值只是一个示例**



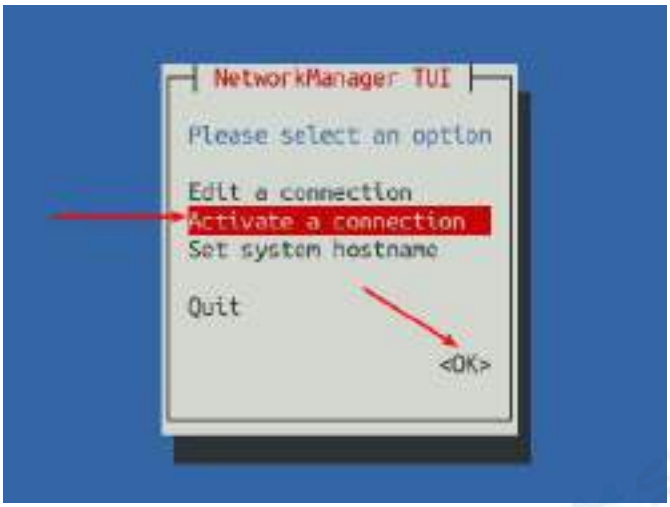
11) 设置完后将光标移动到右下角的**<OK>**，然后回车确认



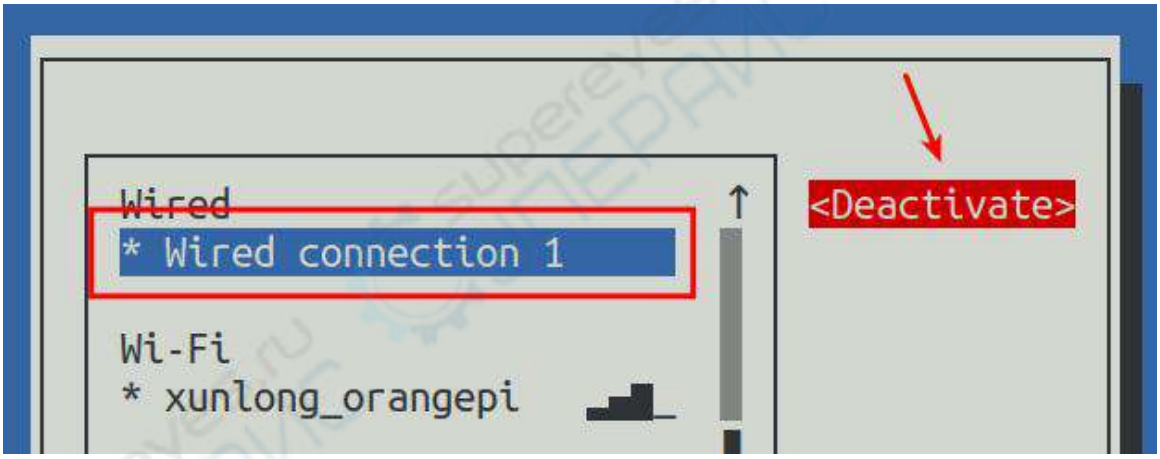
12) 然后点击**<Back>**回退到上一级选择界面



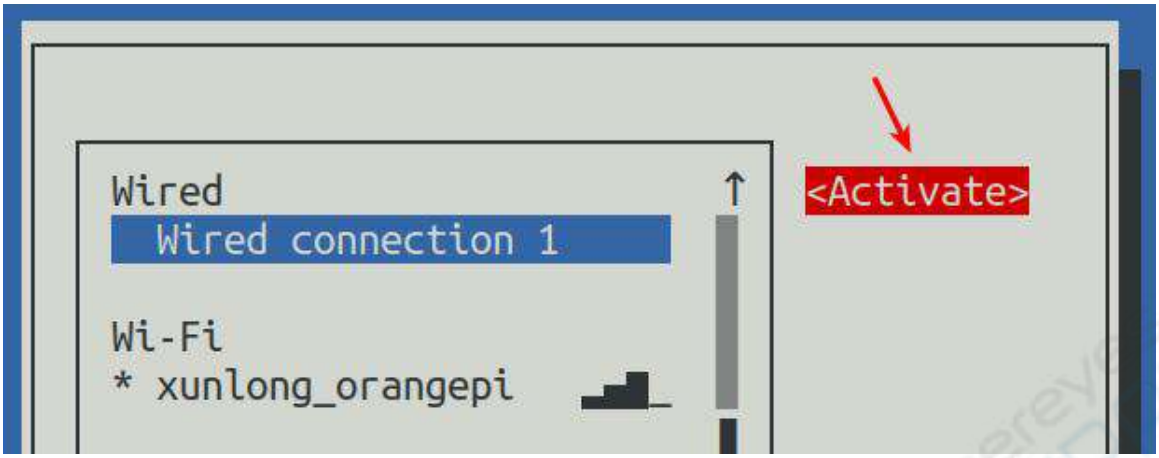
13) 然后选择 **Activate a connection**，再将光标移动到**<OK>**，最后点击回车



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



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



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

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

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

```
        valid_lft forever preferred_lft forever
```

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

```
orangepi@orangepi:~$ ping 192.168.1.47 -I eth0
PING 192.168.1.47 (192.168.1.47) from 192.168.1.188 eth0: 56(84) bytes of data.
64 bytes from 192.168.1.47: icmp_seq=1 ttl=64 time=0.233 ms
64 bytes from 192.168.1.47: icmp_seq=2 ttl=64 time=0.263 ms
64 bytes from 192.168.1.47: icmp_seq=3 ttl=64 time=0.273 ms
64 bytes from 192.168.1.47: icmp_seq=4 ttl=64 time=0.269 ms
64 bytes from 192.168.1.47: icmp_seq=5 ttl=64 time=0.275 ms
^C
--- 192.168.1.47 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4042ms
rtt min/avg/max/mdev = 0.233/0.262/0.275/0.015 ms
```

3.6.3.2. 使用 nmcli 命令来设置静态 IP 地址

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

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

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

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

3) 然后输入下面的命令，其中

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

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

4) 然后重启 linux 系统

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

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

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

3. 6. 4. AP6275P PCIe 网卡通过 create_ap 创建 WIFI 热点的方法

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

https://github.com/oblique/create_ap

如果使用的是最新的镜像，那么就已经预装了create_ap脚本，可以通过create_ap命令来创建WIFI热点，create_ap的基本命令格式如下所示：

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

*** options：可以通过该参数指定加密方式、WIFI热点的频段、频宽模式、网络共享方式等，具体可以通过create_ap -h获取到有哪些option**
*** wifi-interface：无线网卡的名称**
*** interface-with-internet：可以联网的网卡名称，一般是eth0**
*** access-point-name：热点名称**
*** passphrase：热点的密码**

3.6.4.1. create_ap 以 NAT 模式创建 WIFI 热点的方法

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

```
orangepi@orangepi5b:~$ sudo create_ap -m nat wlan0 eth0 orangepi orangepi
```

2）如果有下面的信息输出，说明 WIFI 热点创建成功

```
orangepi@orangepi5b:~$ sudo create_ap -m nat wlan0 eth0 orangepi orangepi
Config dir: /tmp/create_ap.wlan0.conf.fPItFUJ2
PID: 3831
Network Manager found, set ap0 as unmanaged device... DONE
Creating a virtual WiFi interface... ap0 created.
Sharing Internet using method: nat
hostapd command-line interface: hostapd_cli -p
/tmp/create_ap.wlan0.conf.fPItFUJ2/hostapd_ctrl
ap0: interface state UNINITIALIZED->ENABLED
ap0: AP-ENABLED
```

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

4）连接成功后的显示如下图所示

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

```
orangepi@orangepi5b:~$ ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.150  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::938f:8776:5783:afa2  prefixlen 64  scopeid 0x20<link>
        ether 4a:a0:c8:25:42:82  txqueuelen 1000  (Ethernet)
        RX packets 25370  bytes 2709590 (2.7 MB)
        RX errors 0  dropped 50  overruns 0  frame 0
        TX packets 3798  bytes 1519493 (1.5 MB)
```

```
        TX errors 0    dropped 0 overruns 0    carrier 0    collisions 0
        device interrupt 83
```

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

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

```
orangepi@orangepi5b:~$ sudo create_ap -m nat wlan0 eth0 orangepi orangepi -g 192.168.2.1
```

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

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

```
orangepi@orangepi:~$ sudo create_ap -m nat wlan0 eth0 orangepi orangepi --freq-band 5
```

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

```
orangepi@orangepi:~$ sudo create_ap -m nat wlan0 eth0 orangepi orangepi --hidden
```

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

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

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

```
orangepi@orangepi:~$ sudo create_ap -m bridge wlan0 eth0 orangepi orangepi
```

2）如果有下面的信息输出，说明 WIFI 热点创建成功

```
orangepi@orangepi:~$ sudo create_ap -m bridge wlan0 eth0 orangepi orangepi
[sudo] password for orangepi:
Config dir: /tmp/create_ap.wlan0.conf.fg9U5Xgt
PID: 3141
Network Manager found, set ap0 as unmanaged device... DONE
Creating a virtual WiFi interface... ap0 created.
Sharing Internet using method: bridge
Create a bridge interface... br0 created.
hostapd command-line interface: hostapd_cli -p
/tmp/create_ap.wlan0.conf.fg9U5Xgt/hostapd_ctrl
ap0: interface state UNINITIALIZED->ENABLED
ap0: AP-ENABLED
```

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

4）连接成功后的显示如下图所示

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

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

```
        TX errors 0    dropped 0 overruns 0    carrier 0    collisions 0
        device interrupt 83
```

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

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

```
orangepi@orangepi:~$ sudo create_ap -m bridge wlan0 eth0 orangepi orangepi --freq-band 5
```

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

```
orangepi@orangepi:~$ sudo create_ap -m bridge wlan0 eth0 orangepi orangepi --hidden
```

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

3.7. SSH 远程登录开发板

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

3.7.1. Ubuntu 下 SSH 远程登录开发板

1) 获取开发板的 IP 地址

2) 然后就可以通过 ssh 命令远程登录 linux 系统

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

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

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

3) 成功登录系统后的显示如下图所示

如果 ssh 无法正常登陆 linux 系统，首先请检查下开发板的 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：

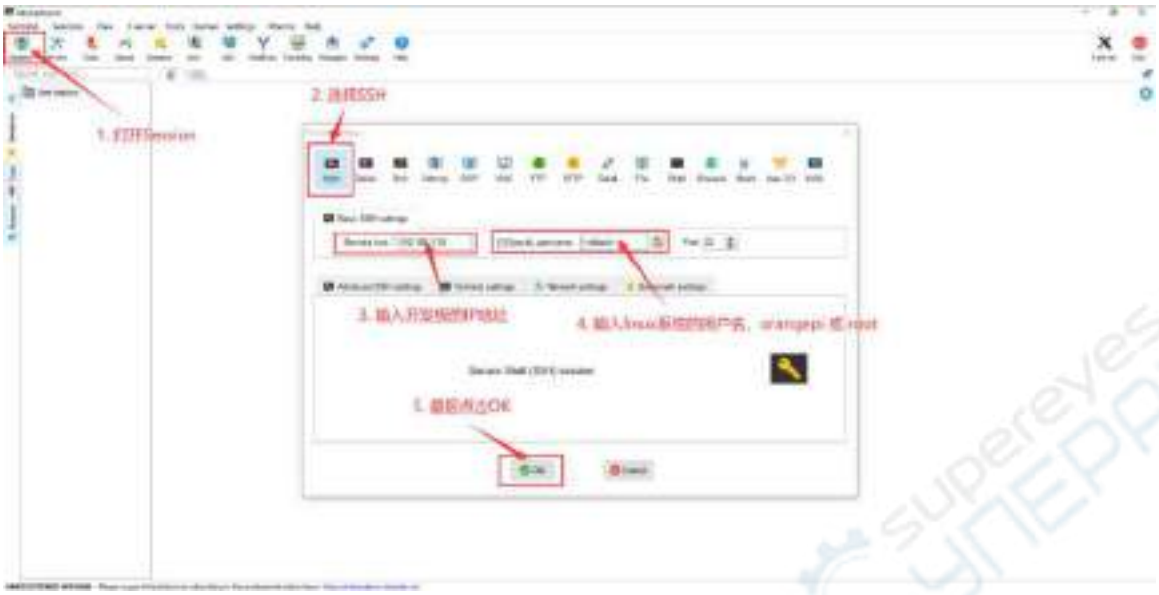
root@orangepi:~# **reset_ssh.sh**

如果还不行，请重烧系统试下。

3.7.2. Windows 下 SSH 远程登录开发板

1) 首先获取开发板的 IP 地址

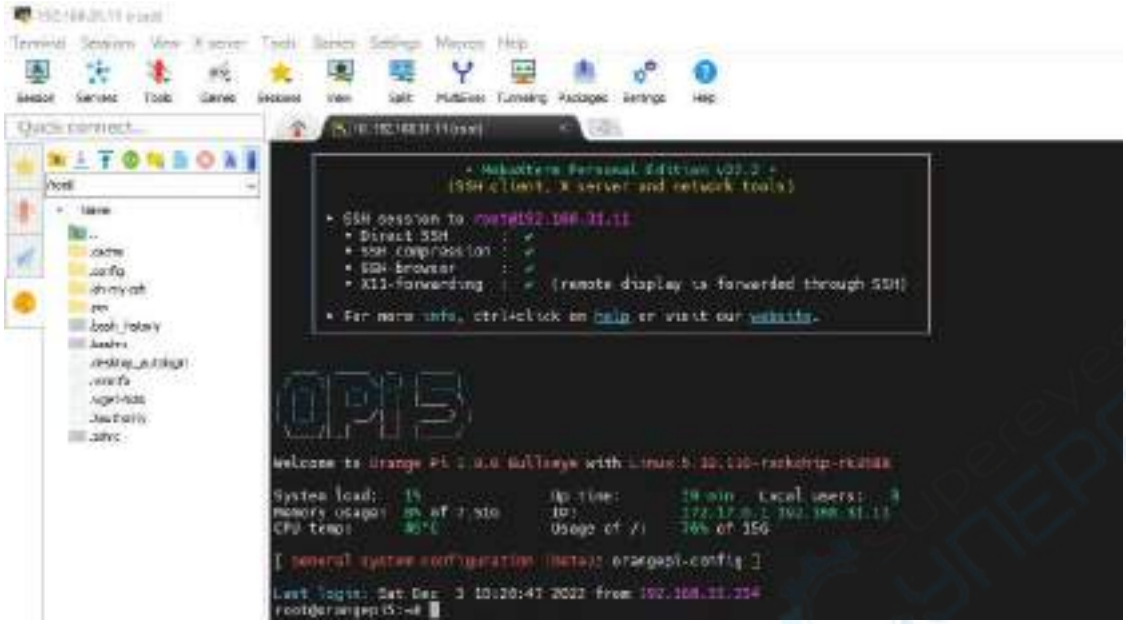
2) 在 windows 下可以使用 MobaXterm 远程登录开发板，首先新建一个 ssh 会话
   a. 打开 **Session**
   b. 然后在 **Session Setting** 中选择 **SSH**
   c. 然后在 **Remote host** 中输入开发板的 IP 地址
   d. 然后在 **Specify username** 中输入 linux 系统的用户名 **root** 或 **orangepi**
   e. 最后点击 **OK** 即可



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



3.8. ADB 的使用方法

3.8.1. 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@orangepi5b:/#     <--- 看到这个提示符后说明已成功登录开发板
```

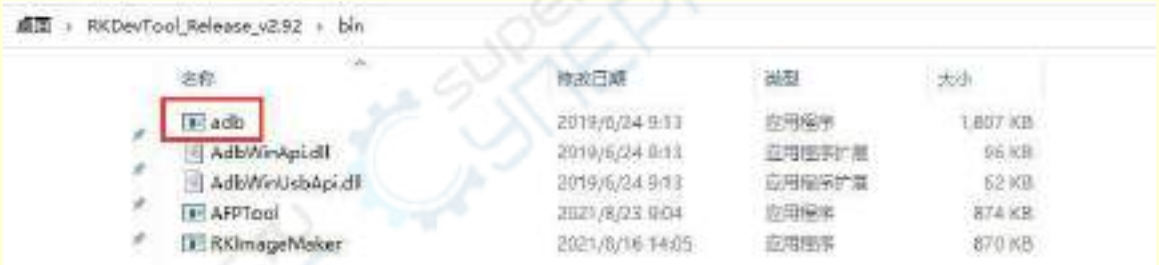
6) 使用 adb 上传文件到开发板的命令如下所示

```
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（烧录 Android 镜像到 SPIFlash+NVMe SSD 中的方法一小节有用到这个软件）中的 adb 程序。



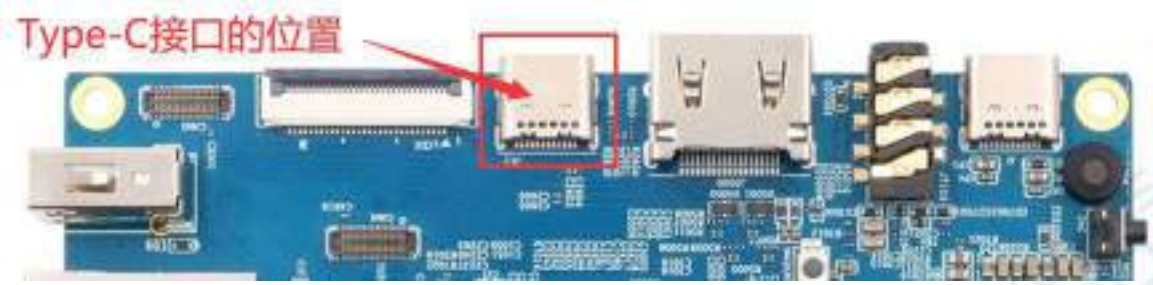
在 Windows 中使用 adb 的示例如下所示：

3.8.2. 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) 然后请确保下面的 USB 接口没有插 USB 设备

3) 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：



4) 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
```

5) 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
```

6) 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
```

7) 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@orangepi5b:/#     <--- 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（烧录 Android 镜像到 SPIFlash+NVMe SSD 中的方法一小节有用到这个软件）中的 adb 程序。

在 Windows 中使用 adb 的示例如下所示：

3.9. 上传文件到开发板 Linux 系统中的方法

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

3.9.1.1. 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
```

3.9.1.2. 使用 filezilla 上传文件的方法

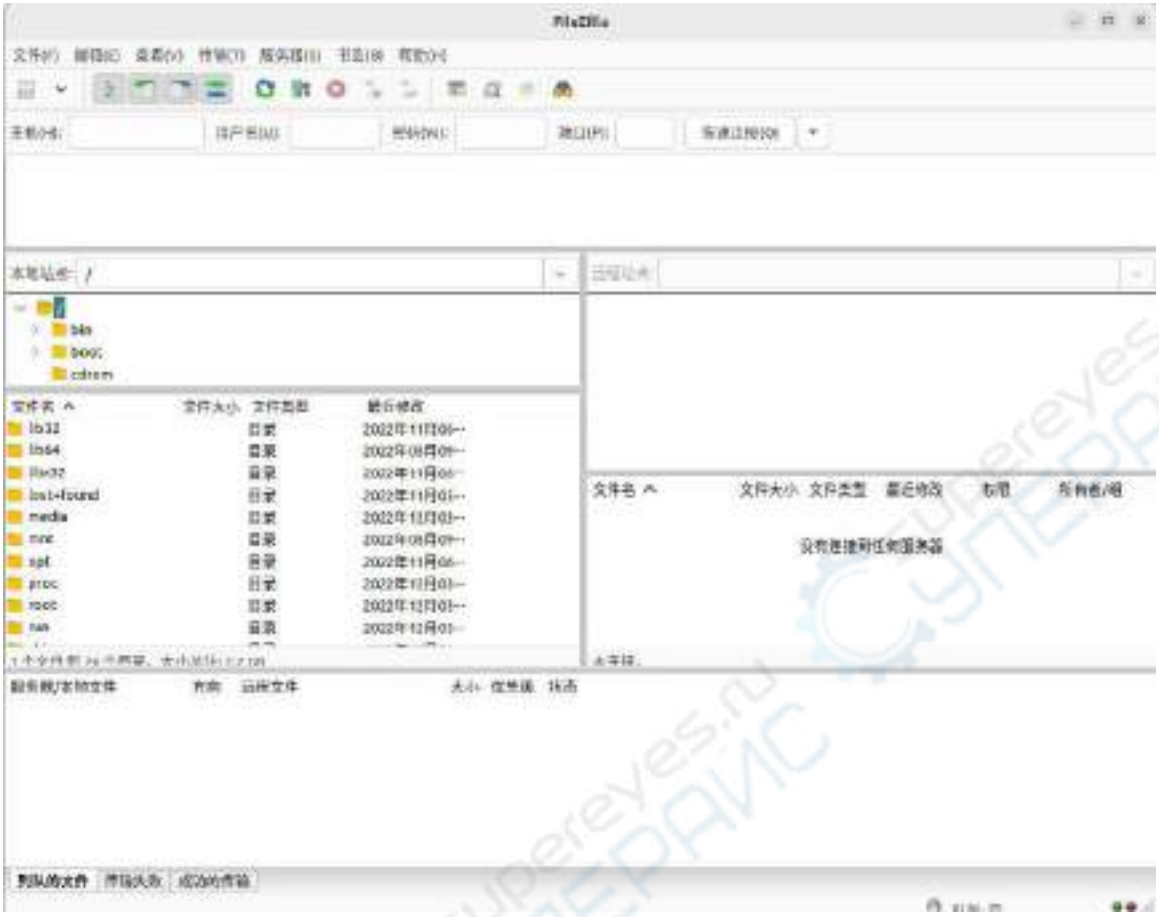
1) 首先在 Ubuntu PC 中安装 filezilla

```
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

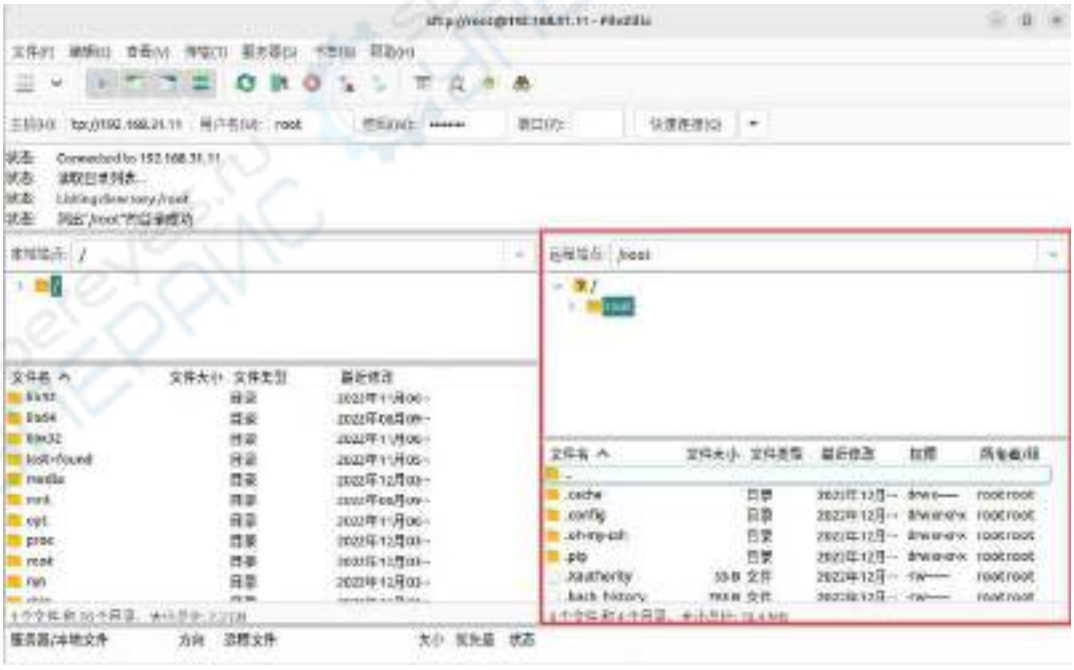


4) 连接开发板的方法如下图所示

5) 然后选择**保存密码**，再点击**确定**

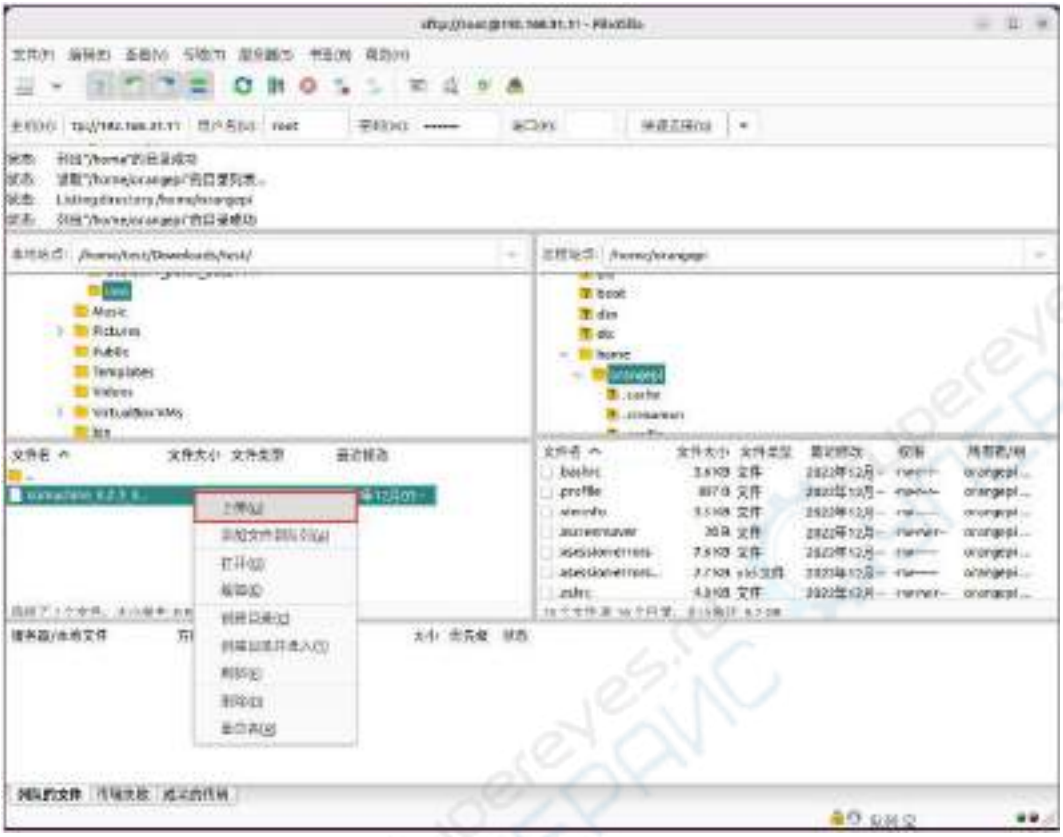
6) 然后选择**总是信任该主机**，再点击**确定**

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

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

3.9.2.1. 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

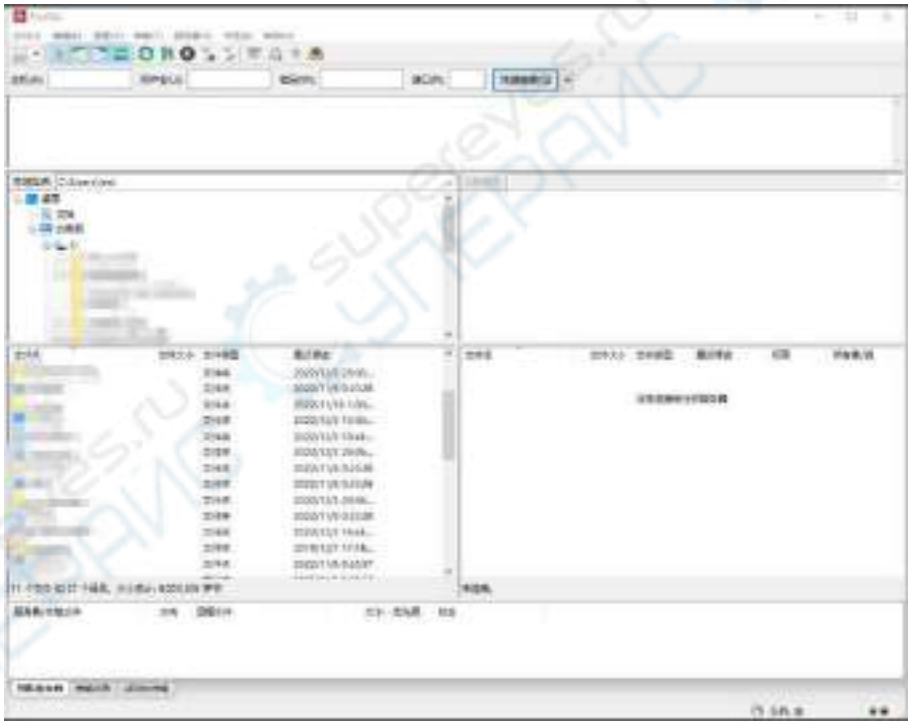
https://filezilla-project.org/download.php#close

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

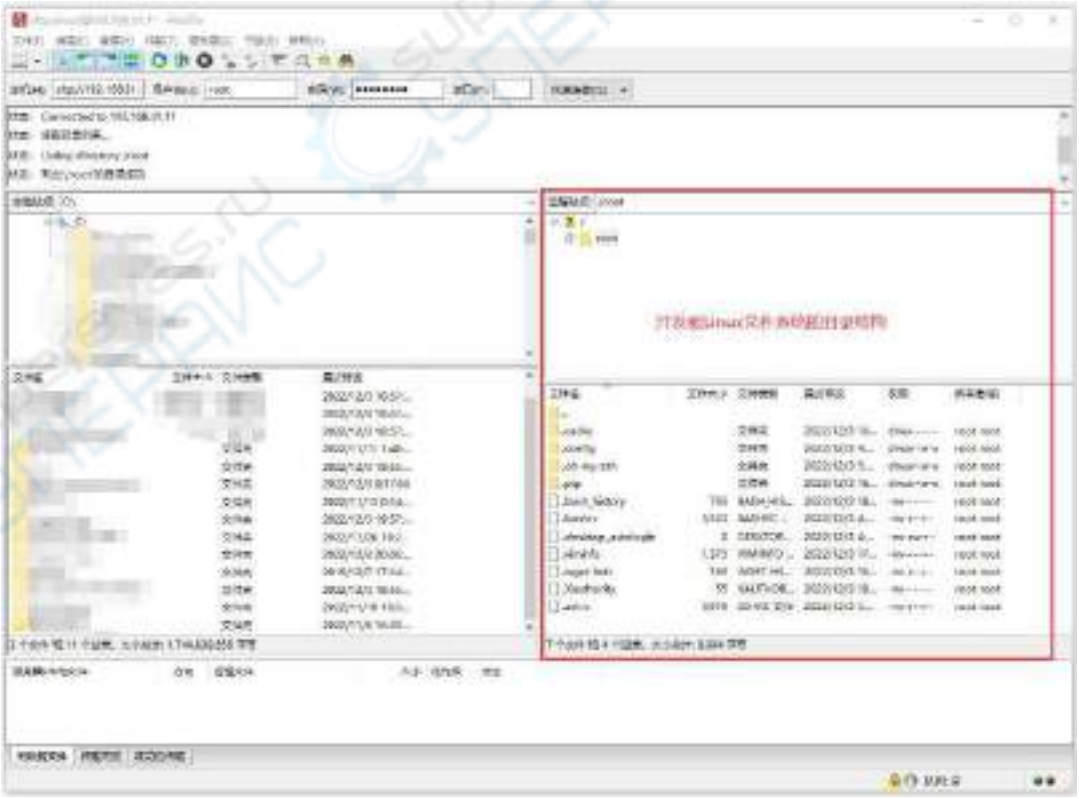


4) 连接开发板的方法如下图所示：

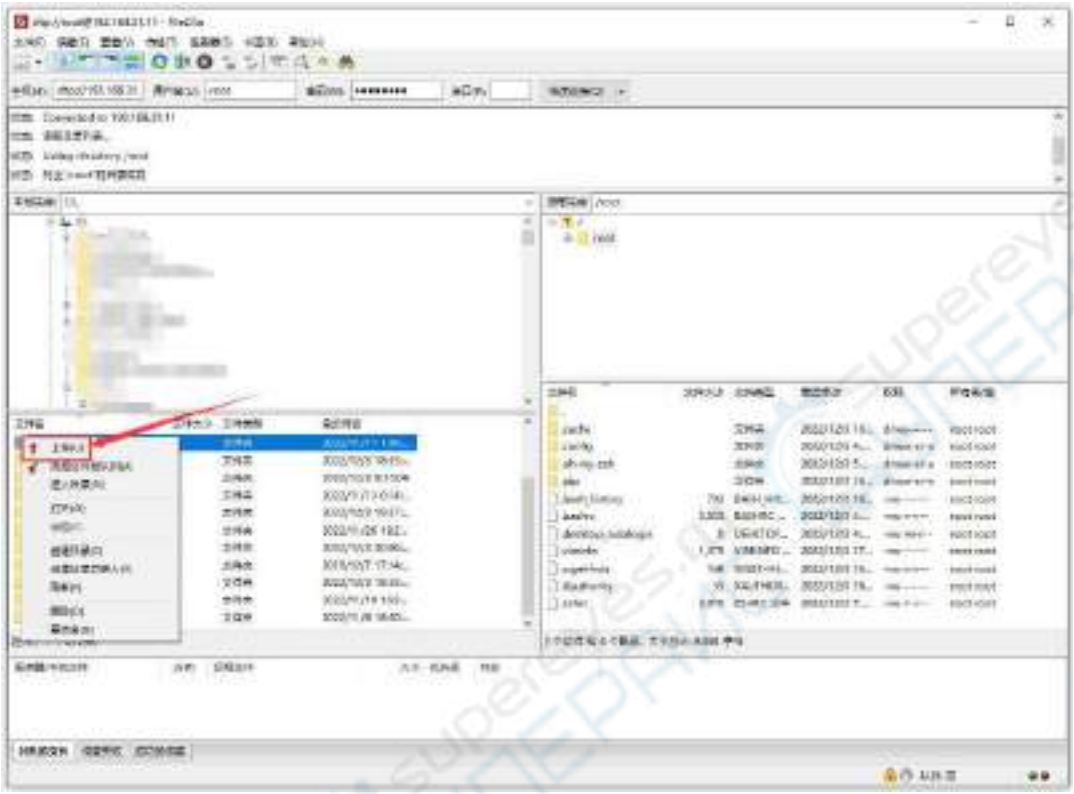
5) 然后选择**保存密码**，再点击**确定**

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

3.10. HDMI 测试

3.10.1. HDMI 显示测试

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

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

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

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

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

3.10.2. HDMI 转 VGA 显示测试

1) 首先需要准备下面的配件

a. HDMI 转 VGA 转换器

b. 一根 VGA 线

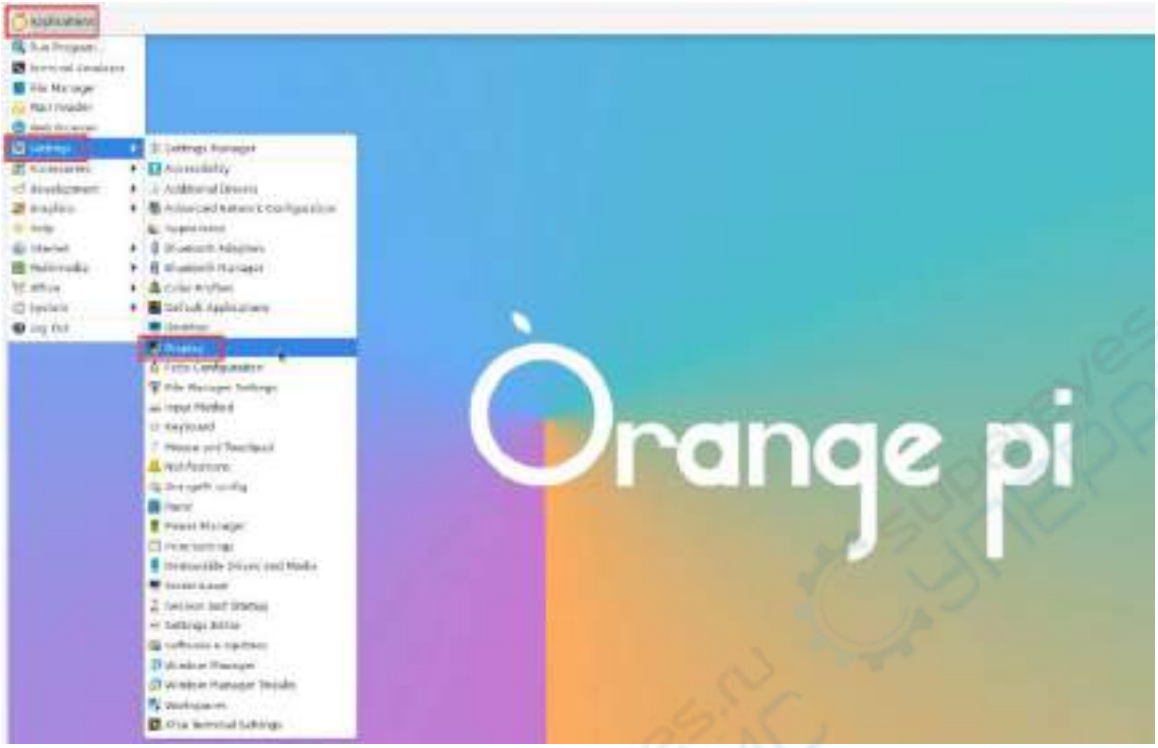
c. 一个支持 VGA 接口的显示器或者电视

2) HDMI 转 VGA 显示测试如下所示

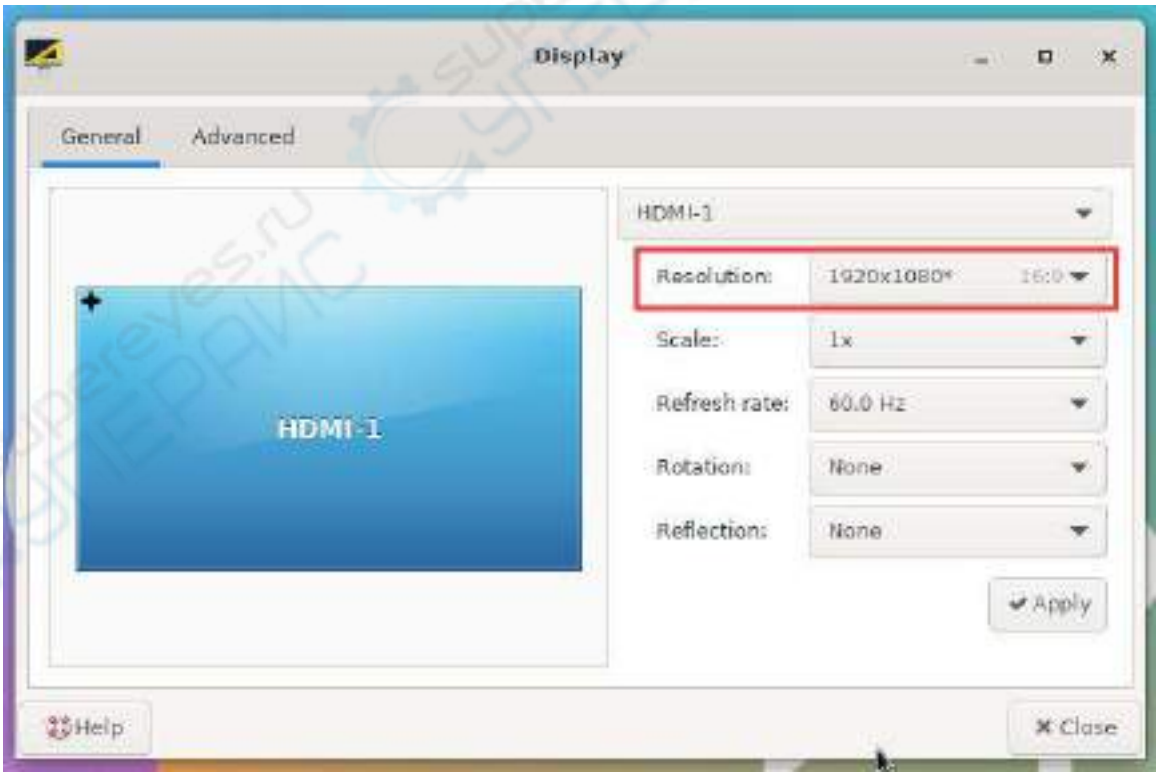
使用 HDMI 转 VGA 显示时，开发板以及开发板的 Linux 系统是不需要做任何设置的，只需要开发板 HDMI 接口能正常显示就可以了。所以如果测试有问题，请检查 HDMI 转 VGA 转换器、VGA 线以及显示器是否有问题。

3.10.3. HDMI 分辨率设置方法

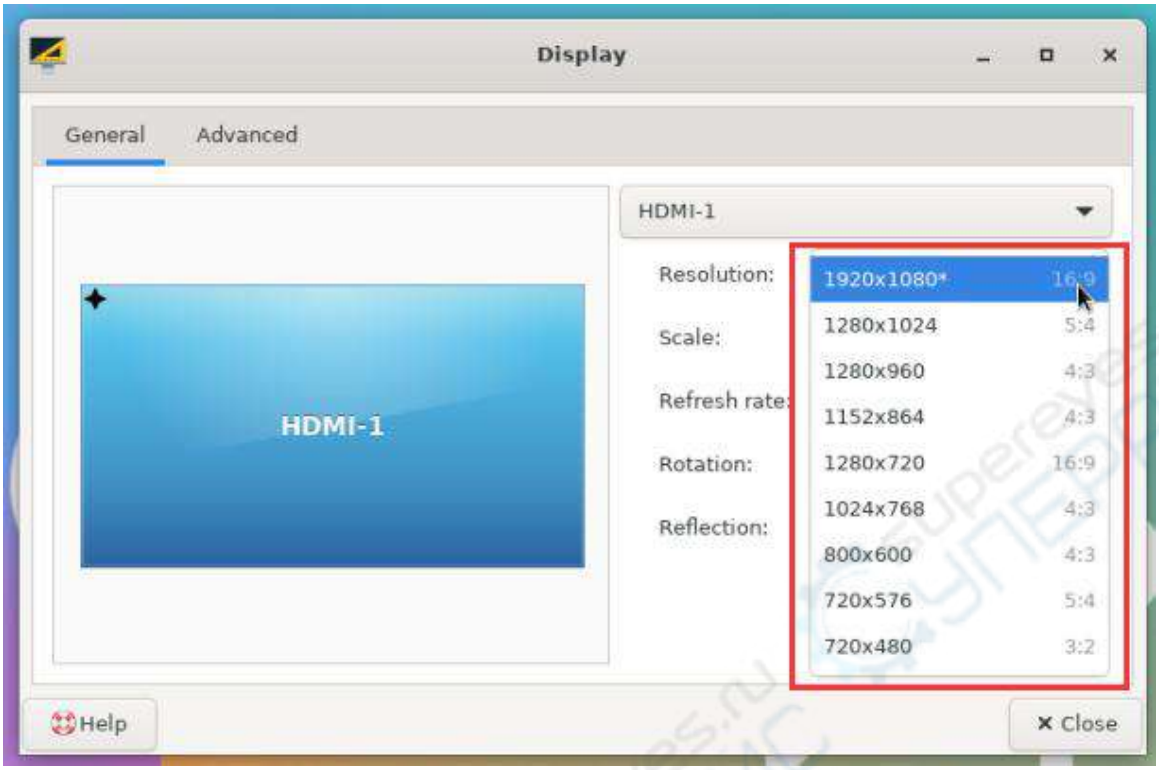
1) 首先在 **Settings** 中打开 **Display**



2) 然后就能看到系统当前的分辨率

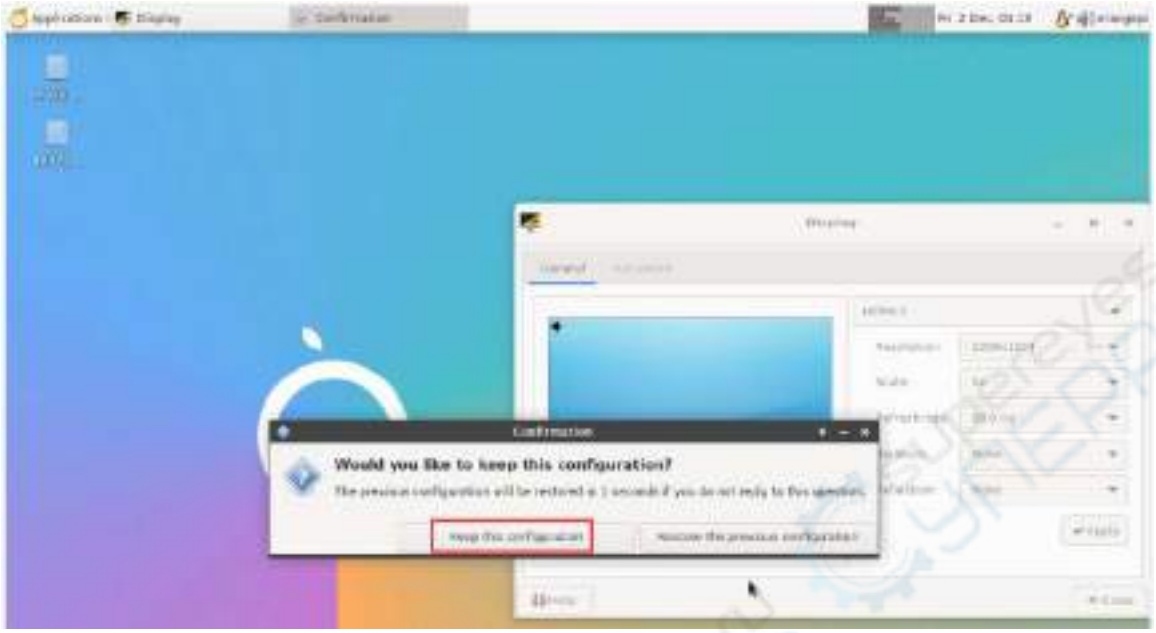


3) 点击 Resolution（分辨率）的下拉框，就可以看到显示器当前支持的所有分辨率



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

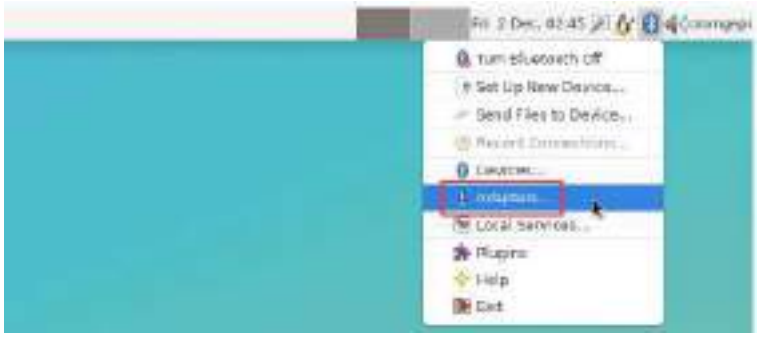


3.11. 蓝牙使用方法

3.11.1. 桌面版镜像的测试方法

1) 点击桌面右上角的蓝牙图标

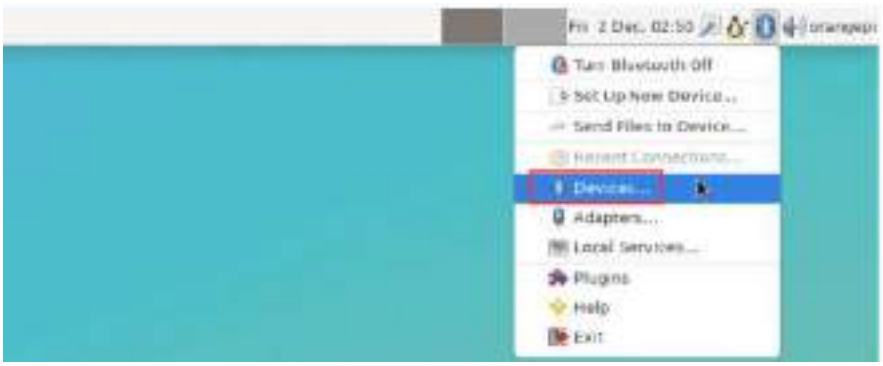
2) 然后选择适配器



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

3.12. USB 接口测试

USB 接口是可以接 USB hub 来扩展 USB 接口的数量的。

3.12.1. 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）

3.12.2. 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
```

3.12.3. 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

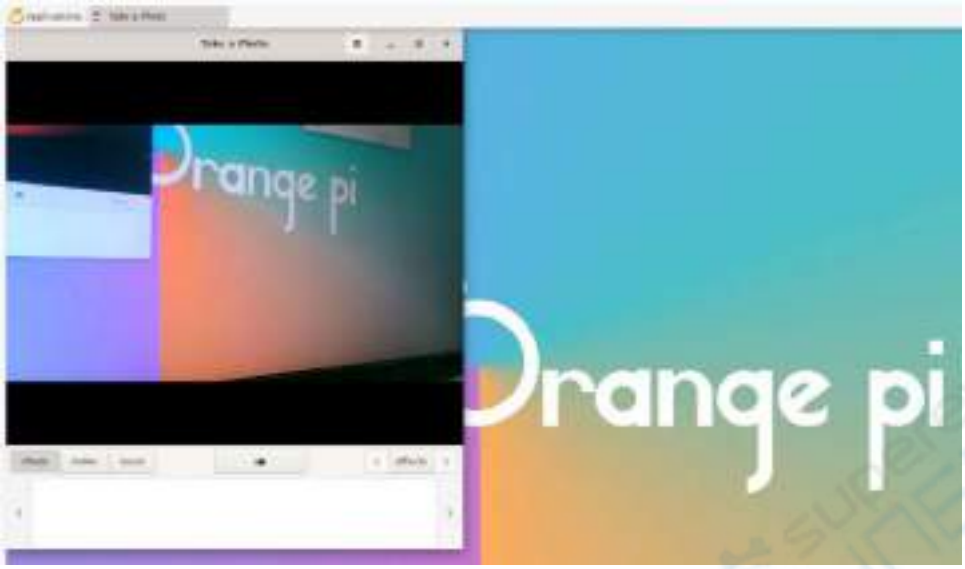
```
orangepi@orangepi:~$ v4l2-ctl --list-devices
Q8 HD Webcam: Q8 HD Webcam (usb-fc880000.usb-1):
        /dev/video0
        /dev/video1
        /dev/media0
```

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：

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

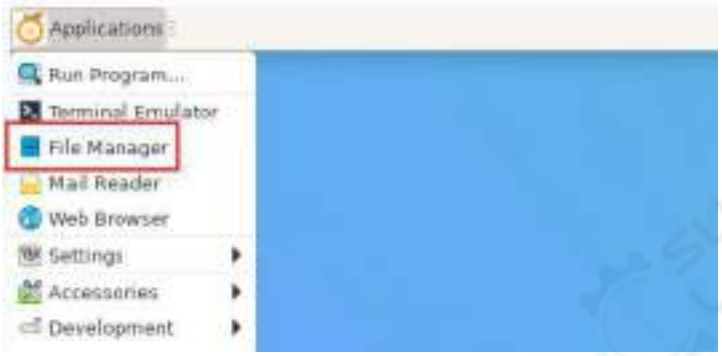
```
orangepi@orangepi:~$ scp image.jpg test@192.168.1.55:/home/test（根据实际情况修改 IP 地址和路径）
```

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

3.13. 音频测试

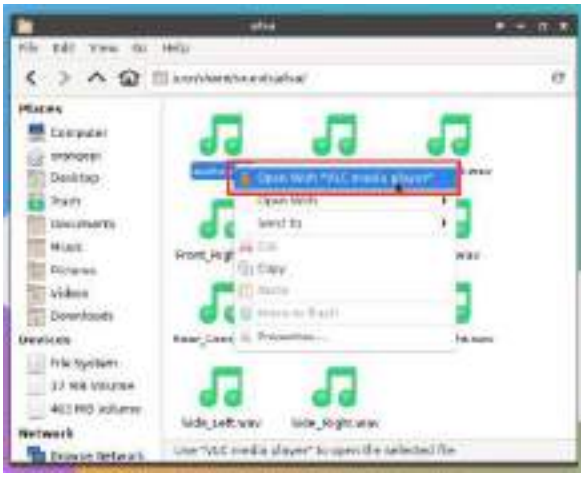
3.13.1. 在桌面系统中测试音频方法

1) 首先打开文件管理器



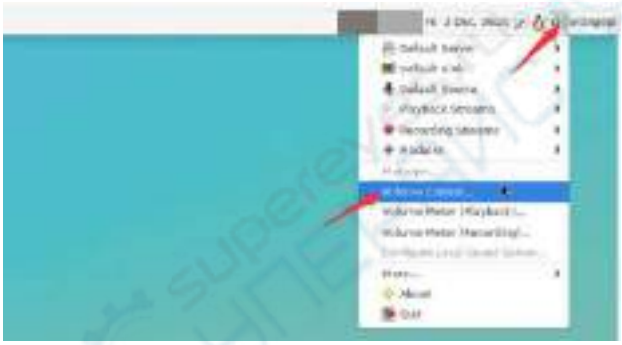
2) 然后找到下面这个文件（如果系统中没有这个音频文件，可以自己上传一个音频文件到系统中）

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

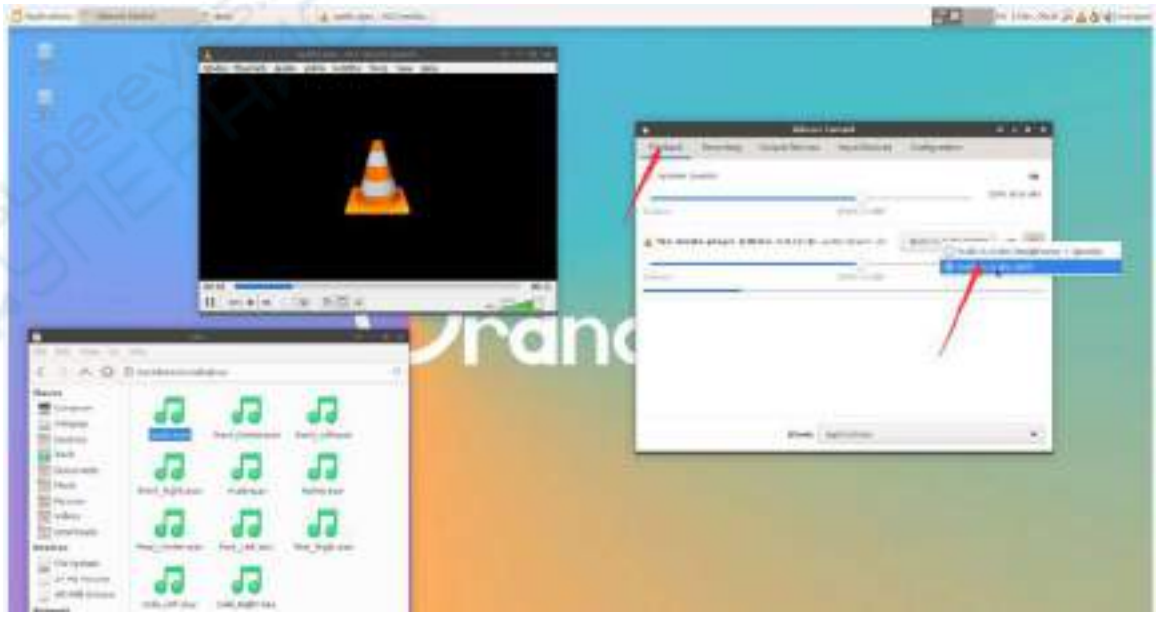


4) 切换 HDMI 播放和耳机播放等不同音频设备的方法

a. 首先打开音量控制界面



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



3.13.2. 使用命令播放音频的方法

3.13.2.1. 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
**** List of PLAYBACK Hardware Devices ****
card 0: rockchipdp0 [rockchip-dp0], device 0: rockchip-dp0 spdif-hifi-0 [rockchip-dp0 spdif-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: rockchiphdmi0 [rockchip-hdmi0], device 0: rockchip-hdmi0 i2s-hifi-0 [rockchip-hdmi0 i2s-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 2: rockchipes8388 [rockchip-es8388], device 0: dailink-multicodecs ES8323.6-0010-0 [dailink-multicodecs ES8323.6-0010-0]
  Subdevices: 1/1
  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
```

3.13.2.2. 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
**** List of PLAYBACK Hardware Devices ****
card 0: rockchipdp0 [rockchip-dp0], device 0: rockchip-dp0 spdif-hifi-0 [rockchip-dp0 spdif-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: rockchiphdmi0 [rockchip-hdmi0], device 0: rockchip-hdmi0 i2s-hifi-0 [rockchip-hdmi0 i2s-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 2: rockchipes8388 [rockchip-es8388], device 0: dailink-multicodecs ES8323.6-0010-0 [dailink-multicodecs ES8323.6-0010-0]
  Subdevices: 1/1
  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
```

3.13.3. 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
```

3.14. 温度传感器

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)
```

3.15. 26 Pin 接口引脚说明

1) Orange Pi 5B 开发板 26 pin 接口引脚的顺序请参考下图

2) Orange Pi 5B 开发板 26 pin 接口引脚的功能如下表所示

a. 下面是 26pin 完整的引脚图

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

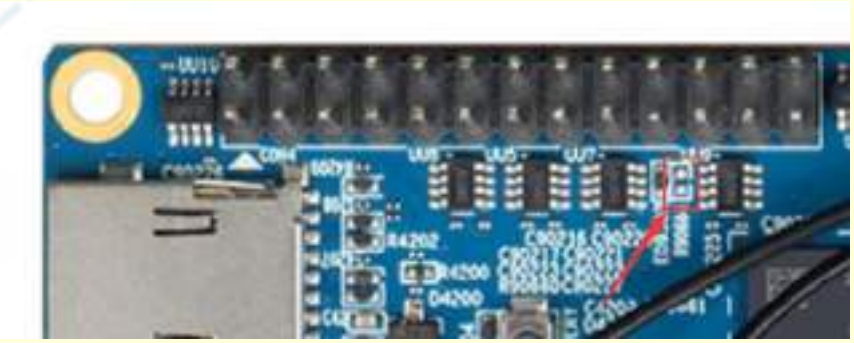
复用功能	复用功能	复用功能	GPIO	GPIO序号	引脚序号
			3.3V		1
PWM13_M2 (febf0010)	UART1_RX_M1 (feb40000)	I2C5_SDA_M3	GPIO1_B7	47	3
	UART1_TX_M1	I2C5_SCL_M3	GPIO1_B6	46	5
		PWM15_IR_M2 (febf0030)	GPIO1_C6	54	7
			GND		9
	PWM14_M1 (febf0020)	CAN1_RX_M1	GPIO4_B2	138	11
		CAN1_TX_M1	GPIO4_B3	139	13
PWM3_IR_M0 (fd8b0030)	I2C1_SCL_M2	CAN2_RX_M1	GPIO0_D4	28	15
			3.3V		17
I2C3_SCL_M0	UART3_TX_M0 (feb60000)	SPI4_MOSI_M0	GPIO1_C1	49	19
I2C3_SDA_M0	UART3_RX_M0	SPI4_MISO_M0	GPIO1_C0	48	21
	PWM3_IR_M2 (fd8b0030)	SPI4_CLK_M0	GPIO1_C2	50	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	复用功能
2		5V			
4		5V			
6		GND			
8	131	GPIO4_A3	UART0_TX_M2 (fd890000)		
10	132	GPIO4_A4	UART0_RX_M2		
12	29	GPIO0_D5	CAN2_TX_M1	I2C1_SDA_M2	
14		GND			
16	59	GPIO1_D3	UART4_RX_M0 (feb70000)	I2C1_SDA_M4	PWM1_M1 (fd8b0010)
18	58	GPIO1_D2	UART4_TX_M0	I2C1_SCL_M4	PWM0_M1 (fd8b0000)
20		GND			
22		PowerKey			
24	52	GPIO1_C4	SPI4_CS1_M0		
26	35	GPIO1_A3	PWM1_M2 (fd8b0010)		

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。

26pin 中的第 22 号引脚是悬空的，默认无任何用途。但是如果将开发板中下面的电阻（大小为 100R）焊接上的话，可以当开关机引脚使用（当将此引脚接到 GND 会触发关机动作）。



3) 26pin 接口中总共有 **16** 个 GPIO 口，所有 GPIO 口的电压都是 **3.3v**

3. 16. 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。

```
root@orangepi5b:~# gpio readall
 +------+-----+----------+--------+---+   PI5B   +---+--------+----------+-----+------+
 | 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      |     |      |
 |   54 |   2 |    PWM15 |     IN | 1 |  7 || 8  | 0 |     IN | RXD.0    |   3 |  131 |
 |      |     |      GND |        |   |  9 || 10 | 0 |     IN | TXD.0    |   4 |  132 |
 |  138 |   5 |  CAN1_RX |     IN | 1 | 11 || 12 | 1 |     IN | CAN2_TX  |   6 |   29 |
 |  139 |   7 |  CAN1_TX |     IN | 1 | 13 || 14 |   |        | GND      |     |      |
 |   28 |   8 |  CAN2_RX |     IN | 1 | 15 || 16 | 1 |     IN | SDA.1    |   9 |   59 |
 |      |     |     3.3V |        |   | 17 || 18 | 1 |     IN | SCL.1    |  10 |   58 |
 |   49 |  11 | SPI4_TXD |     IN | 1 | 19 || 20 |   |        | GND      |     |      |
 |   48 |  12 | SPI4_RXD |     IN | 1 | 21 || 22 |   |        | PowerKey |     |      |
 |   50 |  13 | SPI4_CLK |     IN | 1 | 23 || 24 | 1 |     IN | SPI4_CS1 |  14 |   52 |
 |      |     |      GND |        |   | 25 || 26 | 1 |     IN | PWM1     |  15 |   35 |
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 | GPIO | wPi |   Name   |  Mode  | V | Physical | V |  Mode  |   Name   | wPi | GPIO |
 +------+-----+----------+--------+---+   PI5B   +---+--------+----------+-----+------+
root@orangepi5b:~#
```

wiringOP 目前主要适配了设置 GPIO 口输入输出，设置 GPIO 口输出高低电平以及设置上下拉电阻的功能。像硬件 PWM 这样的功能是用不了的。

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，Orange Pi 5B 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 5B 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:~$ 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@orangepi5b:~# gpio readall
 +------+-----+----------+--------+---+   PI5B   +---+--------+----------+-----+------+
 | 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      |     |      |
 |   54 |   2 |   PWM15  |     IN | 1 |  7 || 8  | 0 |   IN   | RXD.0    |   3 |  131 |
 |      |     |      GND |        |   |  9 || 10 | 0 |   IN   | TXD.0    |   4 |  132 |
 |  138 |   5 |  CAN1_RX |     IN | 1 | 11 || 12 | 1 |   IN   | CAN2_TX  |   6 |   29 |
 |  139 |   7 |  CAN1_TX |     IN | 1 | 13 || 14 |   |        | GND      |     |      |
 |   28 |   8 |  CAN2_RX |     IN | 1 | 15 || 16 | 1 |   IN   | SDA.1    |   9 |   59 |
 |      |     |     3.3V |        |   | 17 || 18 | 1 |   IN   | SCL.1    |  10 |   58 |
 |   49 |  11 |  SPI4_TXD |     IN | 1 | 19 || 20 |   |        | GND      |     |      |
 |   48 |  12 |  SPI4_RXD |     IN | 1 | 21 || 22 |   |        | PowerKey |     |      |
 |   50 |  13 |  SPI4_CLK |     IN | 1 | 23 || 24 | 1 |   IN   | SPI4_CS1 |  14 |   52 |
 |      |     |      GND |        |   | 25 || 26 | 1 |   IN   | PWM1     |  15 |   35 |
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 | GPIO | wPi |   Name   |  Mode  | V | Physical | V |  Mode  |   Name   | wPi | GPIO |
 +------+-----+----------+--------+---+   PI5B   +---+--------+----------+-----+------+
root@orangepi5b:~#
```

3.17. 26pin 接口 GPIO、I2C、UART、SPI 和 PWM 测试

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

```
orangepi@orangepi:~$ sudo vim /boot/orangepiEnv.txt
overlays=i2c1-m2 lcd1 ov13850-c1 pwm13-m2 spi4-m0-cs1-spidev uart0-m2
```

3.17.1. 26pin GPIO 口测试

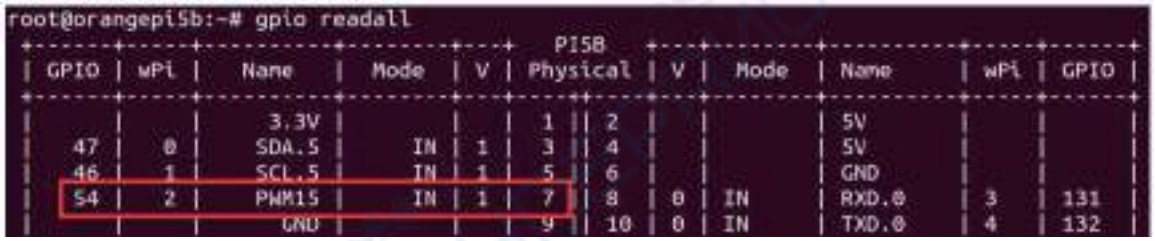
Orange Pi 发布的 linux 系统中有预装一个 blink_all_gpio 程序，这个程序会设置 26pin 中的所有 16 个 GPIO 口不停的切换高低电平。

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@orangepi5b:~$ 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 **16** 个 GPIO 口可以使用，下面以 7 号引脚——对应 GPIO 为 GPIO1_C6 ——对应 wPi 序号为 2——为例演示如何设置 GPIO 口的高低电平



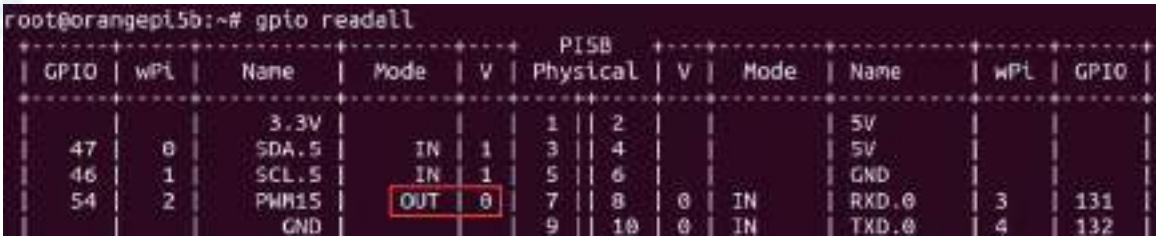
2) 首先设置 GPIO 口为输出模式，其中第三个参数需要输入引脚对应的 wPi 的序号

```
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



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

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

使用 gpio readall 可以看到 7 号引脚的值(V)变为了 1

```
root@orangepi5b:~# gpio readall
 +------+-----+----------+--------+---+  PI5B  +---+--------+----------+-----+------+
 | 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 |     |      |
 |   54 |   2 |   PWM15  |    OUT | 1 |  7 || 8  | 0 |     IN |   RXD.0  |   3 |  131 |
 |      |     |      GND |        |   |  9 || 10 | 0 |     IN |   TXD.0  |   4 |  132 |
```

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

3.17.2. 26pin GPIO 口上下拉电阻的设置方法

注意，Orange Pi 5B 只有下面 4 个 GPIO 引脚可以正常设置上下拉电阻功能，其它的 GPIO 引脚因为外部有 3.3V 上拉，所以设置下拉是无效的。

```
root@orangepi5b:~# gpio readall
 +------+-----+----------+--------+---+  PI5B  +---+--------+----------+-----+------+
 | 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 |     |      |
 |   54 |   2 |   PWM15  |     IN | 1 |  7 || 8  | 0 |     IN |   RXD.0  |   3 |  131 |
 |      |     |      GND |        |   |  9 || 10 | 0 |     IN |   TXD.0  |   4 |  132 |
 |  138 |   5 |  CAN1_RX |     IN | 1 | 11 || 12 | 1 |     IN |  CAN2_TX |   6 |   25 |
 |  139 |   7 |  CAN1_TX |     IN | 1 | 13 || 14 |   |        |      GND |     |      |
 |   28 |   8 |  CAN2_RX |     IN | 1 | 15 || 16 | 1 |     IN |   SDA.1  |   9 |   59 |
 |      |     |     3.3V |        |   | 17 || 18 | 1 |     IN |   SCL.1  |  10 |   58 |
 |   49 |  11 |  SPI4_TXD|     IN | 1 | 19 || 20 |   |        |      GND |     |      |
 |   48 |  12 |  SPI4_RXD|     IN | 1 | 21 || 22 |   |        |  PowerKey|     |      |
 |   50 |  13 |  SPI4_CLK|     IN | 1 | 23 || 24 | 1 |     IN |  SPI4_CS1|  14 |   52 |
 |      |     |      GND |        |   | 25 || 26 | 1 |     IN |    PWM1  |  15 |   35 |
```

1) 下面以 11 号引脚——对应 GPIO 为 GPIO4_B2 ——对应 wPi 序号为 5——为例演示如何设置 GPIO 口的上下拉电阻

```
root@orangepi5b:~# gpio readall
 +------+-----+----------+--------+---+---+PI5B+---+---+--------+----------+-----+------+
 | 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 |     |      |
 |   54 |   2 |   PWM15  |     IN | 1 |  7 || 8  | 0 |   IN   |   RXD.0  |   3 |  131 |
 |      |     |      GND |        |   |  9 || 10 | 0 |   IN   |   TXD.0  |   4 |  132 |
 |  138 |   5 | CAN1_RX  |     IN | 1 | 11 || 12 | 1 |   IN   | CAN2_TX  |   6 |   29 |
 |  139 |   7 | CAN1_TX  |     IN | 1 | 13 || 14 |   |        |      GND |     |      |
 |   28 |   8 | CAN2_RX  |     IN | 1 | 15 || 16 | 1 |   IN   |   SDA.1  |   9 |   59 |
 |      |     |     3.3V |        |   | 17 || 18 | 1 |   IN   |   SCL.1  |  10 |   58 |
 |   49 |  11 | SPI4_TXD |     IN | 1 | 19 || 20 |   |        |      GND |     |      |
 |   48 |  12 | SPI4_RXD |     IN | 1 | 21 || 22 |   |        | PowerKey |     |      |
```

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 5 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 5 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 5
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 5 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 5
0
```

3.17.3. 26pin SPI 测试

1) 由 26pin 接口的原理图可知，Orange Pi 5B 可用的 spi 为 spi4

在 linux 系统中，26pin 中的 spi4 默认是关闭的，需要手动打开才能使用。

在/boot/orangepiEnv.txt 中加入下面红色字体部分的配置，然后重启 Linux 系统就可以打开 spi4。

orangepi@orangepi:~$ **sudo vim /boot/orangepiEnv.txt**
overlays=spi4-m0-cs1-spidev

2) 先查看下 linux 系统中是否存在 **spidev4.1** 的设备节点，如果存在，说明 SPI4 已经设置好了，可以直接使用

```
orangepi@orangepi:~$ ls /dev/spidev4.1
/dev/spidev4.1
```

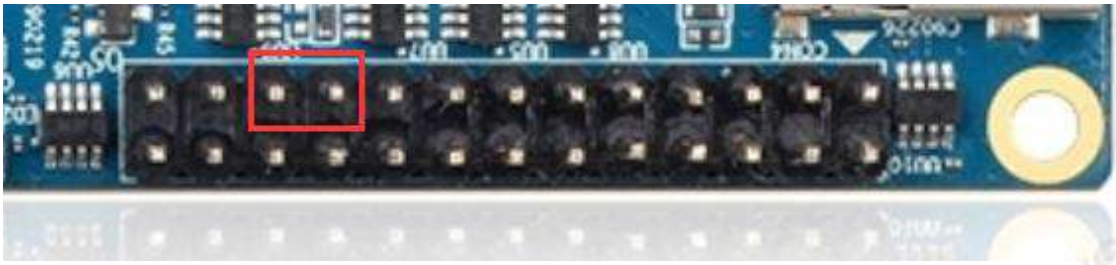
注意，/dev/spidev4.0 是无法使用的，请使用/dev/spidev4.1，不要搞错了。

3) 先不短接 SPI4 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/spidev4.1
spi mode: 0x0
bits per word: 8
max speed: 500000 Hz (500 KHz)
TX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF F0 0D    | ......................@....▒▒.
RX | FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF    | ................................
```

4) 然后短接 SPI4 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/spidev4.1
spi mode: 0x0
bits per word: 8
max speed: 500000 Hz (500 KHz)
TX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF F0 0D    | ......................@....▒▒.
RX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF F0 0D    | ......................@....▒▒.
```

3. 17. 4. 26pin I2C 测试

1) 由下表可知，Orange Pi 5B 可用的 i2c 为 i2c1、i2c3 和 i2c5 共三组 i2c 总线

从上表中可以看到，i2c1 既可以从 26pin 中的 12 和 15 号引脚导出来（i2c1_m2），也可以从 26pin 中的 16 和 18 号引脚导出来（i2c1_m4），请按照自己的需要选择一组即可。请不要以为这是两组不同的 i2c 总线。

在 linux 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/orangepiEnv.txt 中加入下面红色字体部分的配置，然后重启 Linux 系统就可以同时打开 i2c1、i2c3 和 i2c5，如果只需要打开一个，那么就填写一个即可。

选择 i2c1_m2 的设置如下所示：

```
orangepi@orangepi:~$ sudo vim /boot/orangepiEnv.txt
overlays=i2c1-m2  i2c3-m0  i2c5-m3

选择 i2c1_m4 的设置如下所示：
orangepi@orangepi:~$ sudo vim /boot/orangepiEnv.txt
overlays=i2c1-m4  i2c3-m0  i2c5-m3
```

2) 启动 linux 系统后，先确认下/dev 下存在 i2c 的设备节点

```
orangepi@orangepi:~$ ls /dev/i2c-*
/dev/i2c-0   /dev/i2c-10   /dev/i2c-3   /dev/i2c-6   /dev/i2c-9
/dev/i2c-1   /dev/i2c-2    /dev/i2c-5   /dev/i2c-7
```

3) 然后在 26pin 接头的 i2c 引脚上接一个 i2c 设备

	i2c1-m2	i2c1-m4	i2c3-m0	i2c5-m3
sda 引脚	对应 12 号引脚	对应 16 号引脚	对应 21 号引脚	对应 3 号引脚
sck 引脚	对应 15 号引脚	对应 18 号引脚	对应 19 号引脚	对应 5 号引脚
3.3v 引脚	对应 1 号引脚	对应 1 号引脚	对应 1 号引脚	对应 1 号引脚
5v 引脚	对应 2 号引脚	对应 2 号引脚	对应 2 号引脚	对应 2 号引脚
gnd 引脚	对应 6 号引脚	对应 6 号引脚	对应 6 号引脚	对应 6 号引脚

3.3v 引脚和 5v 引脚一般只用接一个即可，请根据具体接的 i2c 设备来选择接 3.3v 引脚还是 5v 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 i2cdetect -y 1      #i2c1 的命令
orangepi@orangepi:~$ sudo i2cdetect -y 3      #i2c3 的命令
orangepi@orangepi:~$ sudo i2cdetect -y 5      #i2c5 的命令
```

```
orangepi@orangepi5: $ sudo i2cdetect -y 5
[sudo] password for orangepi:
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: 50 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
orangepi@orangepi5: $
```

3.17.5. 26pin 的 UART 测试

1) 由下表可知，Orange Pi 5B 可用的 uart 为 uart0、uart1、uart3 和 uart4 共四组 uart 总线

在 linux 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/orangepiEnv.txt 中加入下面红色字体部分的配置，然后重启 Linux 系统就可以同时打开 uart0、uart1、uart3 和 uart4，如果只需要打开一个，那么就填写一个即可。

orangepi@orangepi:~$ **sudo vim /boot/orangepiEnv.txt**
overlays=uart0-m2 uart1-m1 uart3-m0 uart4-m0

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/ttyS0 /dev/ttyS1 /dev/ttyS3 /dev/ttyS4 /dev/ttyS9

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

	uart0	uart1	uart3	uart4

tx 引脚	对应 8 号引脚	对应 5 号引脚	对应 19 号引脚	对应 18 号引脚
rx 引脚	对应 10 号引脚	对应 3 号引脚	对应 21 号引脚	对应 16 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 UART0

```
orangepi@orangepi:~$ sudo gpio serial /dev/ttyS0
[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 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
```

c. 测试 UART3

```
orangepi@orangepi:~$ sudo gpio serial /dev/ttyS3
[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 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
```

3.17.6. PWM 的测试方法

1) 由下表可知，Orange Pi 5B 可用的 pwm 有 pwm0、pwm1、pwm3、pwm13、pwm14 和 pwm15 共六路 pwm

从上表中可以看到:

pwm1 既可以从 26pin 中的 16 号引脚导出来（pwm1_m1），也可以从 26pin 中的 26 号脚导出来（pwm1_m2）

pwm3 既可以从 26pin 中的 15 号引脚导出来（pwm3_m0），也可以从 26pin 中的 23 号脚导出来（pwm3_m2）

请按照自己的需要选择对应的引脚即可。请不要以为这是两路不同的 pwm 总线。

在 linux 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/orangepiEnv.txt 中加入下面红色字体部分的配置，然后重启 Linux 系统就可以同时打开 pwm0、pwm13、pwm14 和 pwm15，如果只需要打开一个，那么就填写一个即可。

orangepi@orangepi:~$ **sudo vim /boot/orangepiEnv.txt**
overlays=pwm0-m1 pwm13-m2 pwm14-m1 pwm15-m2

选择 pwm1_m1 的设置如下所示，pwm1-m1 和 pwm1-m2 请不要同时打开：
orangepi@orangepi:~$ **sudo vim /boot/orangepiEnv.txt**
overlays=pwm1-m1

选择 pwm1_m2 的设置如下所示：
orangepi@orangepi:~$ **sudo vim /boot/orangepiEnv.txt**
overlays=pwm1-m2

选择 pwm3_m0 的设置如下所示，pwm3-m0 和 pwm3-m2 请不要同时打开：
orangepi@orangepi:~$ **sudo vim /boot/orangepiEnv.txt**
overlays=pwm3-m0

选择 pwm3_m2 的设置如下所示：
orangepi@orangepi:~$ **sudo vim /boot/orangepiEnv.txt**
overlays=pwm3-m2

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 pwm15 后，查看**/sys/class/pwm/**下的 pwmchipX 会由两个变成了三个

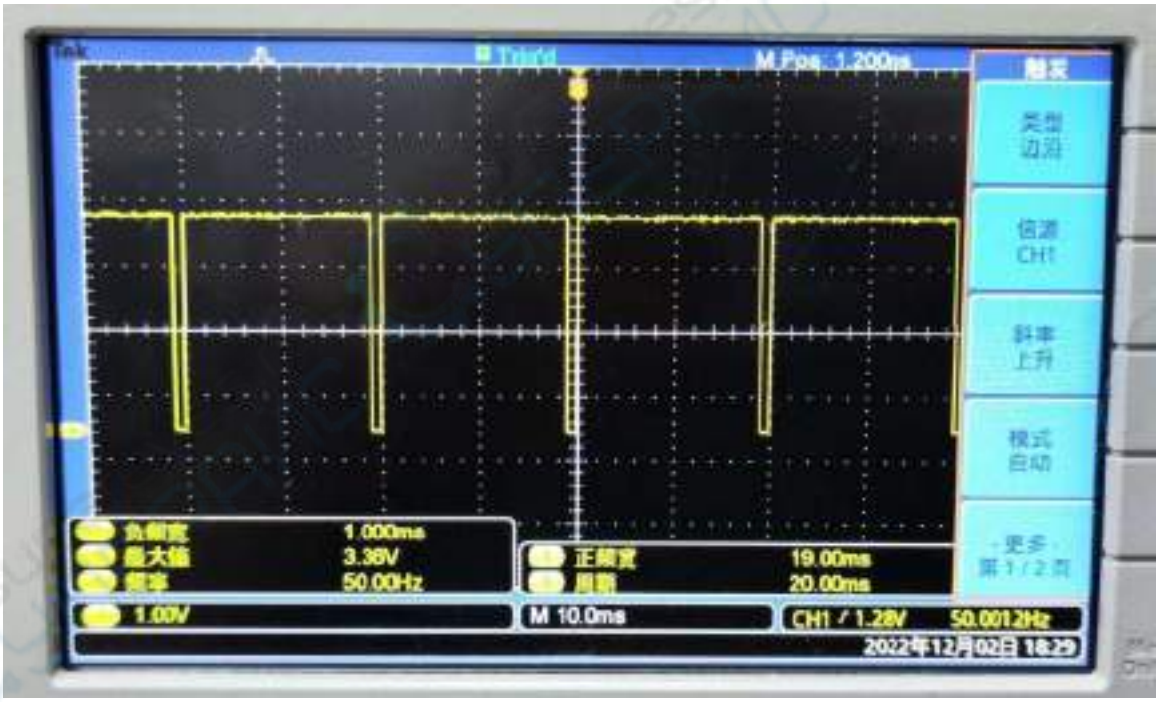
```
orangepi@orangepi:~$ ls /sys/class/pwm/
pwmchip0    pwmchip1    pwmchip2
```

3) 上面哪个 pwmchip 对应 pwm15 呢，我们先查看下 **ls /sys/class/pwm/ -l** 命令的输出，如下所示：

4) 然后由下表可知，pwm15 寄存器的基地址为 febf0030，再看 **ls /sys/class/pwm/ -l** 命令的输出，可以看到 pwmchip2 中链接到了 febf0030.pwm，所以 pwm15 对应 pwmchip 为 pwmchip2

5) 然后使用下面的命令可以让 pwm15 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/pwmchip2/export
root@orangepi:~# echo 20000000 > /sys/class/pwm/pwmchip2/pwm0/period
root@orangepi:~# echo 1000000 > /sys/class/pwm/pwmchip2/pwm0/duty_cycle
root@orangepi:~# echo 1 > /sys/class/pwm/pwmchip2/pwm0/enable
```



6) 上面演示的 pwm15 的测试方法，其他 pwm 测试方法都是类似的。

3.17.7. CAN 的测试方法

3.17.7.1. 打开 CAN 的方法

1) 由下表可知，Orange Pi 5B 可用的 CAN 总线为 CAN1 和 CAN2 共两组 CAN 总线

在 linux 系统中，26pin 中的 CAN 默认都是关闭的，需要手动打开才能使用。

在/boot/orangepiEnv.txt 中加入下面红色字体部分的配置，然后重启 Linux 系统就可以同时打开 CAN1 和 CAN2，如果只需要打开一个，那么就填写一个即可。

orangepi@orangepi:~$ **sudo vim /boot/orangepiEnv.txt**
overlays=can1-m1 can2-m1

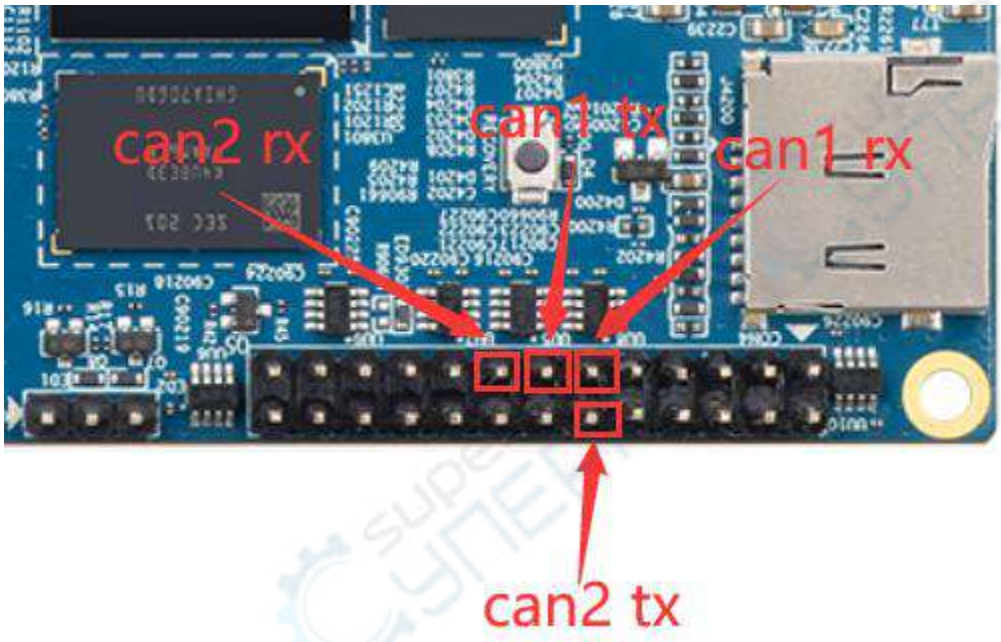
2) 进入 linux 系统后，使用 **sudo ifconfig -a** 命令如果能看到 CAN 的设备，就说明 CAN 已正确打开了

```
orangepi@orangepi:~$ sudo ifconfig -a
can0: flags=128<NOARP>  mtu 16
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 10  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 91

can1: flags=128<NOARP>  mtu 16
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 10  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
```

```
        TX packets 0    bytes 0 (0.0 B)
        TX errors 0    dropped 0 overruns 0    carrier 0    collisions 0
        device interrupt 92
```

3) CAN1 和 CAN2 对应的引脚为

	CAN1	CAN2
TX 引脚	对应 13 号引脚	对应 12 号引脚
RX 引脚	对应 11 号引脚	对应 15 号引脚



3.17.7.2. 使用 CANalyst-II 分析仪测试收发消息

1) 测试使用的 CANalyst-II 分析仪如下图所示

2) CANalyst-II 分析仪资料下载链接

https://www.zhcxgd.com/3.html

3) 首先要安装 USBCANToolSetup 这个软件

4) USBCANToolSetup 安装后的快捷方式为

5) 另外还需要安装一下 USB 驱动程序

6) CANalyst-II 分析仪的 USB 接口那端需要接到电脑的 USB 接口中

7) 测试 CAN 功能还需要准备一个下图所示的 CAN 收发器，CAN 收发器主要功能是将 CAN 控制器的 TTL 信号转换成 CAN 总线的差分信号

a. CAN 收发器的 3.3V 引脚需要接开发板 26pin 中的 3.3V 引脚
b. CAN 收发器的 GND 引脚需要接开发板 26pin 中的 GND 引脚
c. CAN 收发器的 CAN TX 引脚需要接开发板 26pin 中 CAN 总线的 TX 引脚

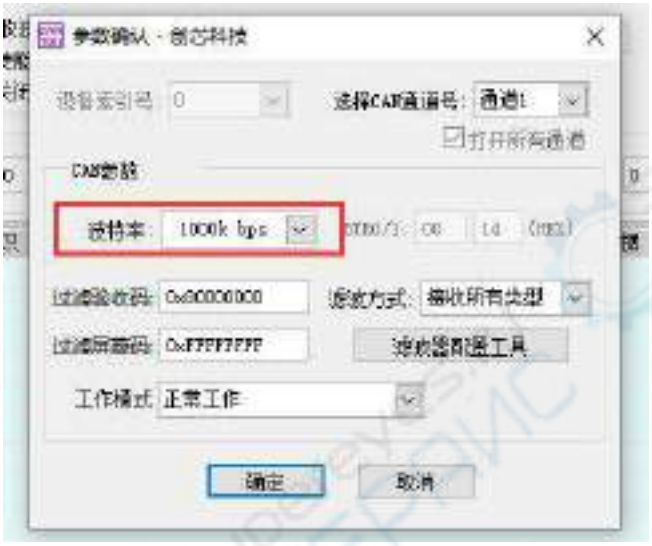
d. CAN 收发器的 CAN RX 引脚需要接开发板 26pin 中 CAN 总线的 RX 引脚
e. CAN 收发器的 CANL 引脚需要接分析仪的 H 接口
f. CAN 收发器的 CANL 引脚需要接分析仪的 L 接口

8) 然后可以打开 USB-CAN 软件

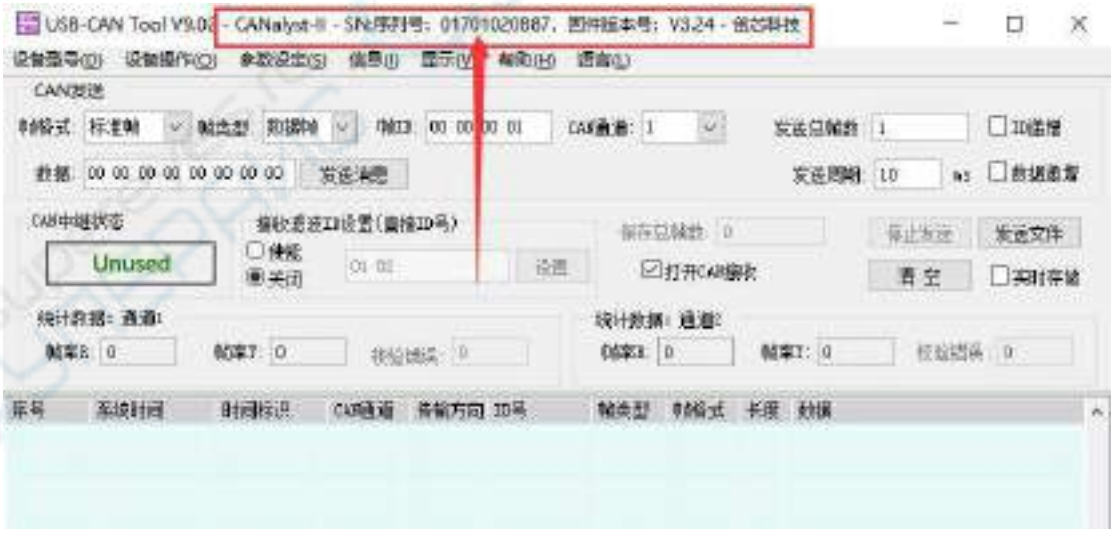
9) 然后点击启动设备

10) 然后点击确定

11) 再设置波特率为 1000k bps



12) 成功打开后 USB-CAN 软件会显示序列号等信息



13) 开发板接收 CAN 消息测试

a. 首先在开发板的 Linux 系统中设置下 CAN 总线的波特率为 **1000kbps**

```
orangepi@orangepi:~$ sudo ip link set can0 down
```

```
orangepi@orangepi:~$ sudo ip link set can0 type can bitrate 1000000
orangepi@orangepi:~$ sudo ip link set can0 up
```

b. 然后运行 **candump can0** 命令准备接收消息

```
orangepi@orangepi:~$ sudo candump can0
```

c. 然后在 USB-CAN 软件中发送一个消息给开发板

d. 如果开发板中可以接收到分析仪发送的消息说明 CAN 总线能正常使用

```
orangepi@orangepi:~$ sudo candump can0
    can0  001  [8]  01 02 03 04 05 06 07 08
```

14) 开发板发送 CAN 消息测试

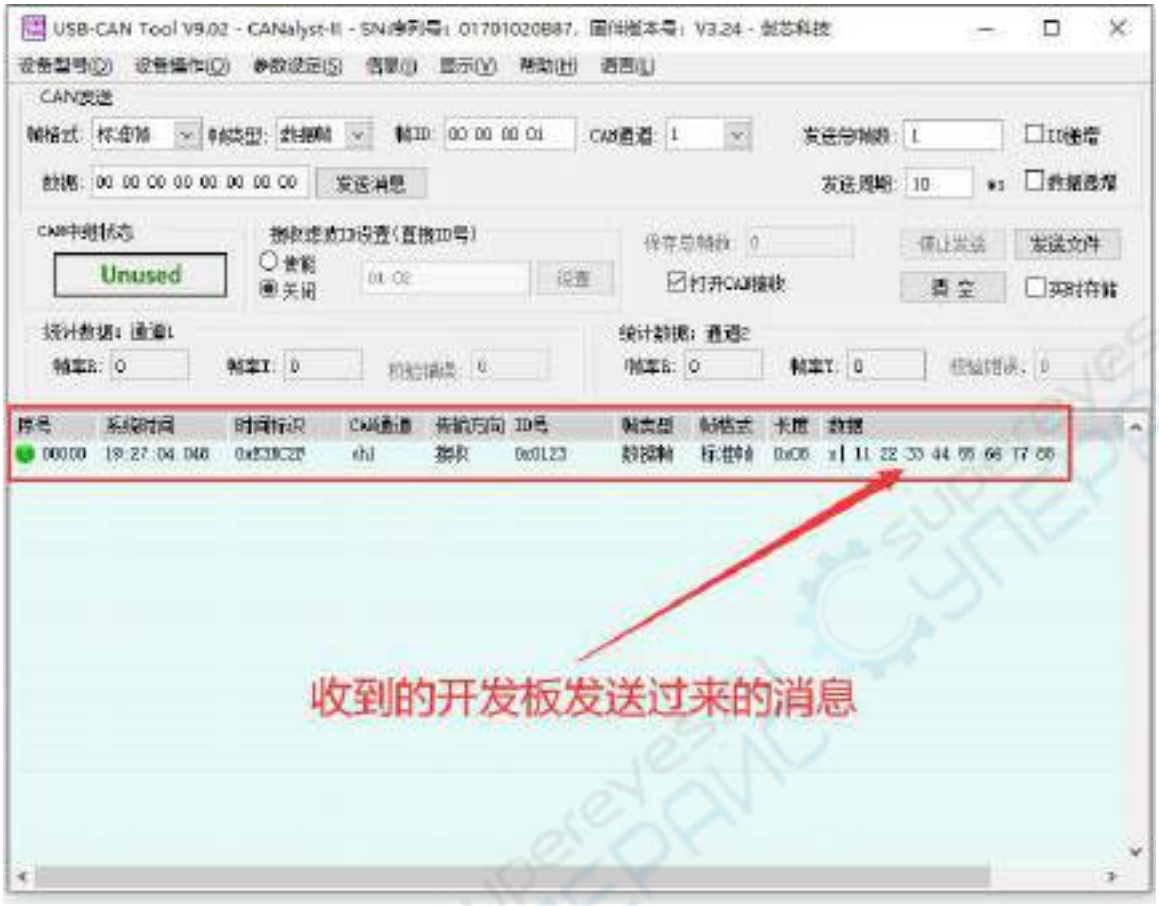
a. 首先在 Linux 系统中设置下 CAN 的波特率为 **1000kbps**

```
orangepi@orangepi:~$ sudo ip link set can0 down
orangepi@orangepi:~$ sudo ip link set can0 type can bitrate 1000000
orangepi@orangepi:~$ sudo ip link set can0 up
```

b. 再开发板中执行 **cansend** 命令，发送一个消息

```
orangepi@orangepi:~$ sudo cansend can0 123#1122334455667788
```

c. 如果 USB-CAN 软件可以接收到开发板发过来的消息说明通信成功



3.18. 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。

3.18.1. 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。

代码如果下不下来，请去官方工具中下载源码压缩包。


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

Cloning into 'wiringOP-Python'...
remote: Enumerating objects: 602, done.
remote: Counting objects: 100% (40/40), done.
remote: Compressing objects: 100% (28/28), done.
remote: Total 602 (delta 20), reused 26 (delta 12), pack-reused 562
Receiving objects: 100% (602/602), 309.30 KiB | 1.23 MiB/s, done.
Resolving deltas: 100% (349/349), done.
Submodule 'wiringOP' (https://github.com/orangepi-xunlong/wiringOP.git) registered for path 'wiringOP'
Cloning into '/root/test/wiringOP-Python/wiringOP'...
remote: Enumerating objects: 654, done.
remote: Counting objects: 100% (273/273), done.
remote: Compressing objects: 100% (33/33), done.
remote: Total 654 (delta 244), reused 245 (delta 238), pack-reused 381
Receiving objects: 100% (654/654), 360.54 KiB | 1.73 MiB/s, done.
Resolving deltas: 100% (424/424), done.
Submodule path 'wiringOP': checked out '85f1331cd8fda668115461ec1c06cb342057eb03'
```

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)
       |

>>>
```

3.18.2. 26pin GPIO 口测试

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

```
root@orangepi5b:~# gpio readall
 +------+-----+----------+--------+---+   PI5B   +---+--------+----------+-----+------+
 | 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 |     |      |
 |   54 |   2 |    PWM15 |     IN | 1 |  7 || 8  | 0 |     IN |   RXD.0  |   3 |  131 |
 |      |     |      GND |        |   |  9 || 10 | 0 |     IN |   TXD.0  |   4 |  132 |
 |  138 |   5 |  CAN1_RX |     IN | 1 | 11 || 12 | 1 |     IN |  CAN2_TX |   6 |   29 |
 |  139 |   7 |  CAN1_TX |     IN | 1 | 13 || 14 |   |        |      GND |     |      |
 |   28 |   8 |  CAN2_RX |     IN | 1 | 15 || 16 | 1 |     IN |   SDA.1  |   9 |   59 |
 |      |     |     3.3V |        |   | 17 || 18 | 1 |     IN |   SCL.1  |  10 |   58 |
 |   49 |  11 | SPI4_TXD |     IN | 1 | 19 || 20 |   |        |      GND |     |      |
 |   48 |  12 | SPI4_RXD |     IN | 1 | 21 || 22 |   |        |  PowerKey |     |      |
 |   50 |  13 | SPI4_CLK |     IN | 1 | 23 || 24 | 1 |     IN |  SPI4_CS1 |  14 |   52 |
 |      |     |      GND |        |   | 25 || 26 | 1 |     IN |    PWM1  |  15 |   35 |
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 | GPIO | wPi |   Name   |  Mode  | V | Physical | V |  Mode  |   Name   | wPi | GPIO |
 +------+-----+----------+--------+---+   PI5B   +---+--------+----------+-----+------+
root@orangepi5b:~#
```

1) 开发板 26pin 中总共有 **16** 个 GPIO 口可以使用，下面以 7 号引脚——对应 GPIO 为 GPIO1_C6 ——对应 wPi 序号为 2——为例演示如何设置 GPIO 口的高低电平

```
root@orangepi5b:~# gpio readall
 +------+-----+----------+--------+---+   PI5B   +---+--------+----------+-----+------+
 | 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 |     |      |
 |   54 |   2 |    PWM15 |     IN | 1 |  7 || 8  | 0 |     IN |   RXD.0  |   3 |  131 |
 |      |     |      GND |        |   |  9 || 10 | 0 |     IN |   TXD.0  |   4 |  132 |
```

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. 18. 3. 26pin SPI 测试

1) 由 26pin 接口的原理图可知，Orange Pi 5B 可用的 spi 为 spi4

在 linux 系统中，26pin 中的 spi4 默认是关闭的，需要手动打开才能使用。

在/boot/orangepiEnv.txt 中加入下面红色字体部分的配置，然后重启 Linux 系统就可以打开 spi4。

orangepi@orangepi:~$ **sudo vim /boot/orangepiEnv.txt**
overlays=spi4-m0-cs1-spidev

2) 先查看下 linux 系统中是否存在 **spidev4.1** 的设备节点，如果存在，说明 SPI4 已经设置好了，可以直接使用

```
orangepi@orangepi:~$ ls /dev/spidev4.1
/dev/spidev4.1
```

注意，/dev/spidev4.0 是无法使用的，请使用/dev/spidev4.1，不要搞错了。

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

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

4) 先不短接 SPI4 的 mosi 和 miso 两个引脚，运行 spidev_test.py 的输出结果如下所示，可以看到 TX 和 RX 的数据不一致

```
root@orangepi:~/wiringOP-Python# cd examples
root@orangepi:~/wiringOP-Python/examples# python3 spidev_test.py \
--channel 4 --port 1
spi mode: 0x0
max speed: 500000 Hz (500 KHz)
Opening device /dev/spidev4.1
TX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF F0 0D    |..............@.|
RX | FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF    |................|
```

5) 然后使用杜邦线短接 SPI4 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 SPI4 回环测试正常

```
root@orangepi:~/wiringOP-Python# cd examples
root@orangepi:~/wiringOP-Python/examples# python3 spidev_test.py \
--channel 4 --port 1
spi mode: 0x0
max speed: 500000 Hz (500 KHz)
Opening device /dev/spidev4.1
TX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF F0 0D    |..............@.|
RX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF F0 0D    |..............@.|
```

3. 18. 4. 26pin I2C 测试

1) 由下表可知，Orange Pi 5B 可用的 i2c 为 i2c1、i2c3 和 i2c5 共三组 i2c 总线

从上表中可以看到，i2c1 既可以从 26pin 中的 12 和 15 号引脚导出来（i2c1_m2），也可以从 26pin 中的 16 和 18 号引脚导出来（i2c1_m4），请按照自己的需要选择一组即可。请不要以为这是两组不同的 i2c 总线。

在 linux 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/orangepiEnv.txt 中加入下面红色字体部分的配置，然后重启 Linux 系统就可以同时打开 i2c1、i2c3 和 i2c5，如果只需要打开一个，那么就填写一个即可。

选择 i2c1_m2 的设置如下所示：

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

overlays=i2c1-m2 i2c3-m0 i2c5-m3

选择 i2c1_m4 的设置如下所示：

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

overlays=i2c1-m4 i2c3-m0 i2c5-m3

2) 启动 linux 系统后，先确认下/dev 下存在 i2c 的设备节点

```
orangepi@orangepi:~$ ls /dev/i2c-*
/dev/i2c-0   /dev/i2c-10   /dev/i2c-3   /dev/i2c-6   /dev/i2c-9
/dev/i2c-1   /dev/i2c-2    /dev/i2c-5   /dev/i2c-7
```

3) 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

	i2c1-m2	i2c1-m4	i2c3-m0	i2c5-m3
sda 引脚	对应 12 号引脚	对应 16 号引脚	对应 21 号引脚	对应 3 号引脚
sck 引脚	对应 15 号引脚	对应 18 号引脚	对应 19 号引脚	对应 5 号引脚
3.3v 引脚	对应 1 号引脚	对应 1 号引脚	对应 1 号引脚	对应 1 号引脚
5v 引脚	对应 2 号引脚	对应 2 号引脚	对应 2 号引脚	对应 2 号引脚
gnd 引脚	对应 6 号引脚	对应 6 号引脚	对应 6 号引脚	对应 6 号引脚

3.3v 引脚和 5v 引脚一般只用接一个即可，请根据具体接的 i2c 设备来选择接 3.3v 引脚还是 5v 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 i2cdetect -y 1      #i2c1 的命令
orangepi@orangepi:~$ sudo i2cdetect -y 3      #i2c3 的命令
orangepi@orangepi:~$ sudo i2cdetect -y 5      #i2c5 的命令
```

```
orangepi@orangepi5: $ sudo i2cdetect -y 5
[sudo] password for orangepi:
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: 50 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
orangepi@orangepi5: $
```

5) 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-5"
Thu 2023-01-05  14:57:55
Thu 2023-01-05  14:57:56
Thu 2023-01-05  14:57:57
^C
```

```
exit
```

3.18.5. 26pin 的 UART 测试

1) 由下表可知，Orange Pi 5B 可用的 uart 为 uart0、uart1、uart3 和 uart4 共四组 uart 总线

在 linux 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/orangepiEnv.txt 中加入下面红色字体部分的配置，然后重启 Linux 系统就可以同时打开 uart0、uart1、uart3 和 uart4，如果只需要打开一个，那么就填写一个即可。

orangepi@orangepi:~$ **sudo vim /boot/orangepiEnv.txt**
overlays=uart0-m2 uart1-m1 uart3-m0 uart4-m0

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/ttyS0  /dev/ttyS1  /dev/ttyS3  /dev/ttyS4  /dev/ttyS9
```

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

	uart0	uart1	uart3	uart4
tx 引脚	对应 8 号引脚	对应 5 号引脚	对应 19 号引脚	对应 18 号引脚
rx 引脚	对应 10 号引脚	对应 3 号引脚	对应 21 号引脚	对应 16 号引脚

4) 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

a. 测试 UART0

```
root@orangepi:~/wiringOP-Python/examples# python3 serialTest.py --device \
"/dev/ttyS0"

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

b. 测试 UART1

```
root@orangepi:~/wiringOP-Python/examples# python3 serialTest.py --device \
"/dev/ttyS1"

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

c. 测试 UART3

```
root@orangepi:~/wiringOP-Python/examples# python3 serialTest.py --device \
"/dev/ttyS3"

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

d. 测试 UART4

```
root@orangepi:~/wiringOP-Python/examples# python3 serialTest.py --device \
"/dev/ttyS4"

Out:    0: ->    0
```

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

3.19. 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
```

3.20. 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
```

3.21. 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:~$ 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.
.....
```

3.22. 下载安装 arm64 版本 balenaEtcher 的方法

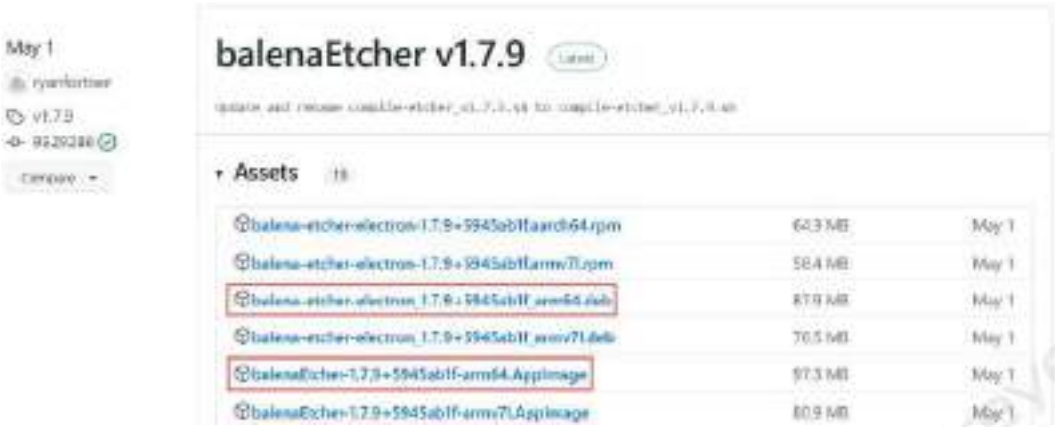
1) balenaEtcher arm64 版本的下载地址为：

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

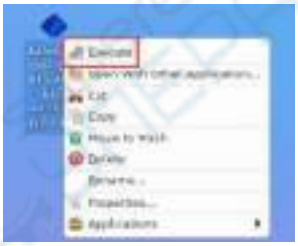
c. balenaEtcher 打开后的界面如下所示：

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



3.23. 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
```

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
```

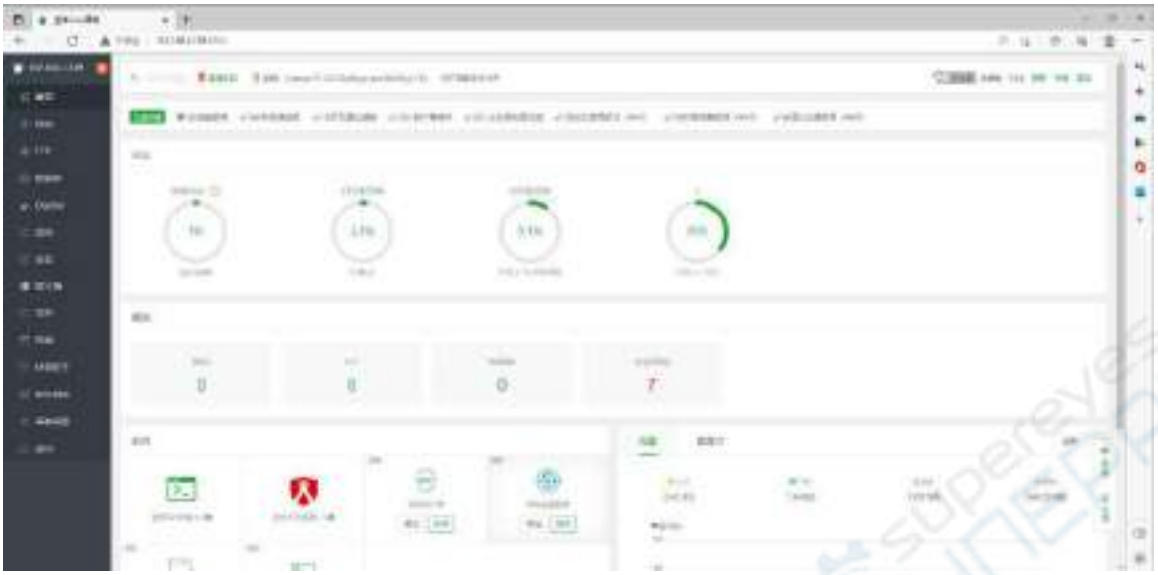
4) 然后要做的就是耐心等待，当看到终端输出下面的打印信息时，说明宝塔已经安装完成，整个安装过程大约耗时 12 分钟，根据网络速度的不同可能会有一些差别

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

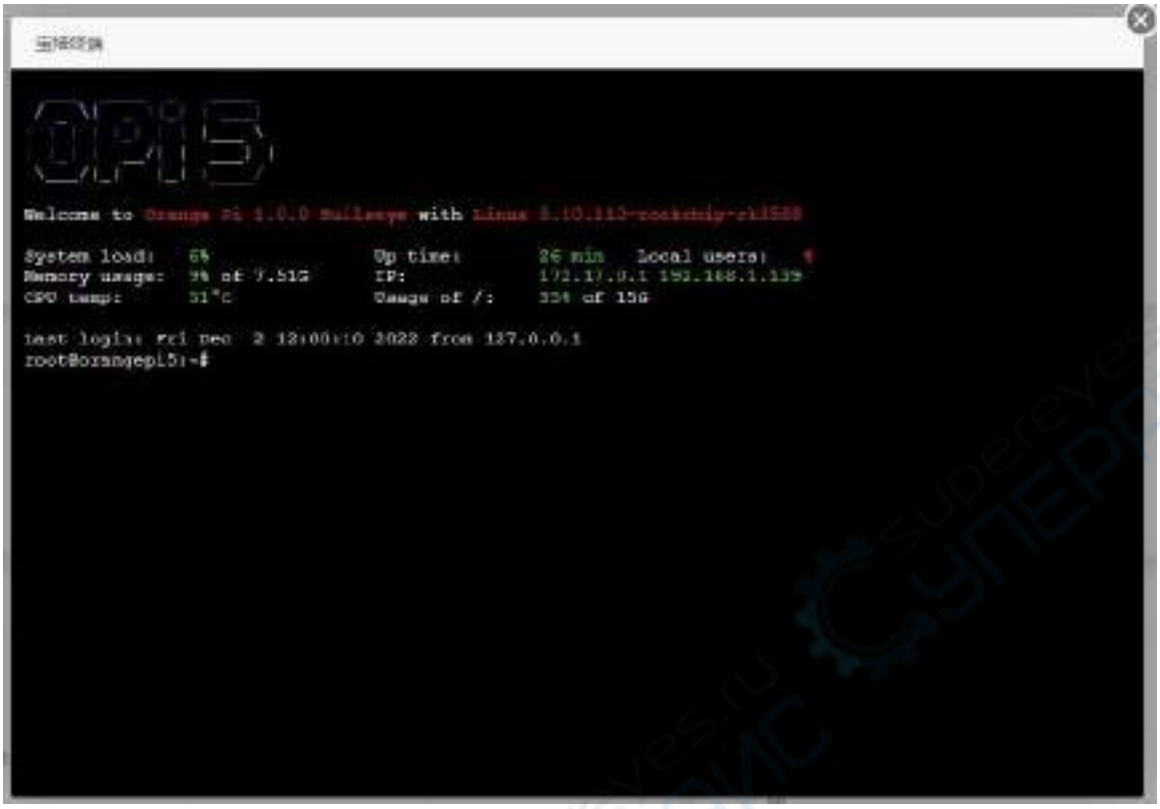
8) 最终显示的界面如下图所示，可以很直观的看到开发板 Linux 系统的一些状态信息，比如负载状态、CPU 的使用率、内存使用率和存储空间的使用情况等



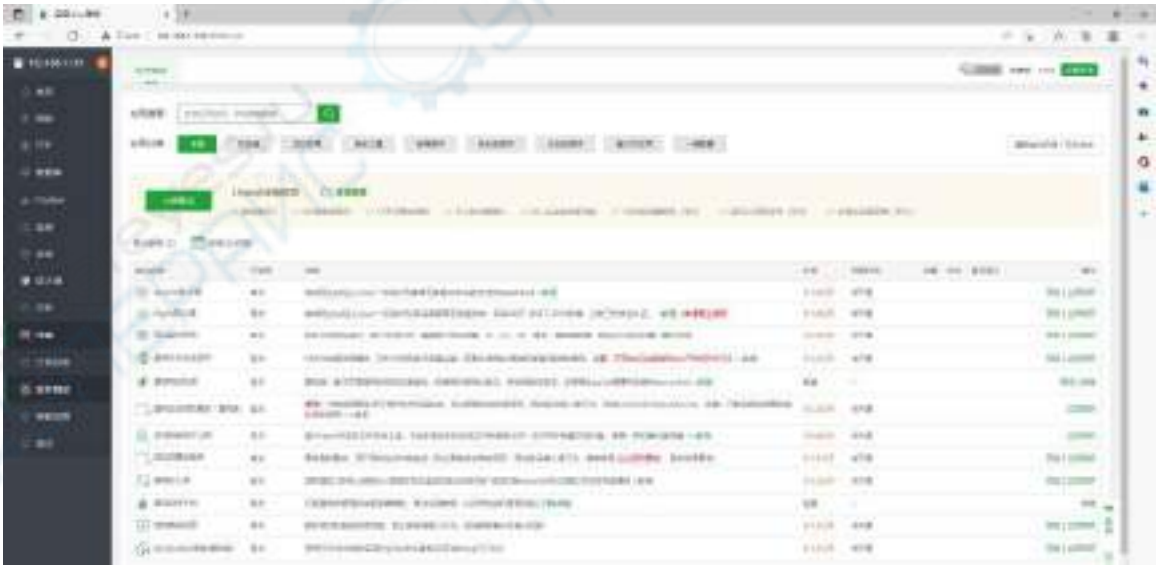
9) 测试宝塔的 SSH 终端登录

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

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**

3.24. 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。

3.24.1. Debian 11 系统的安装方法

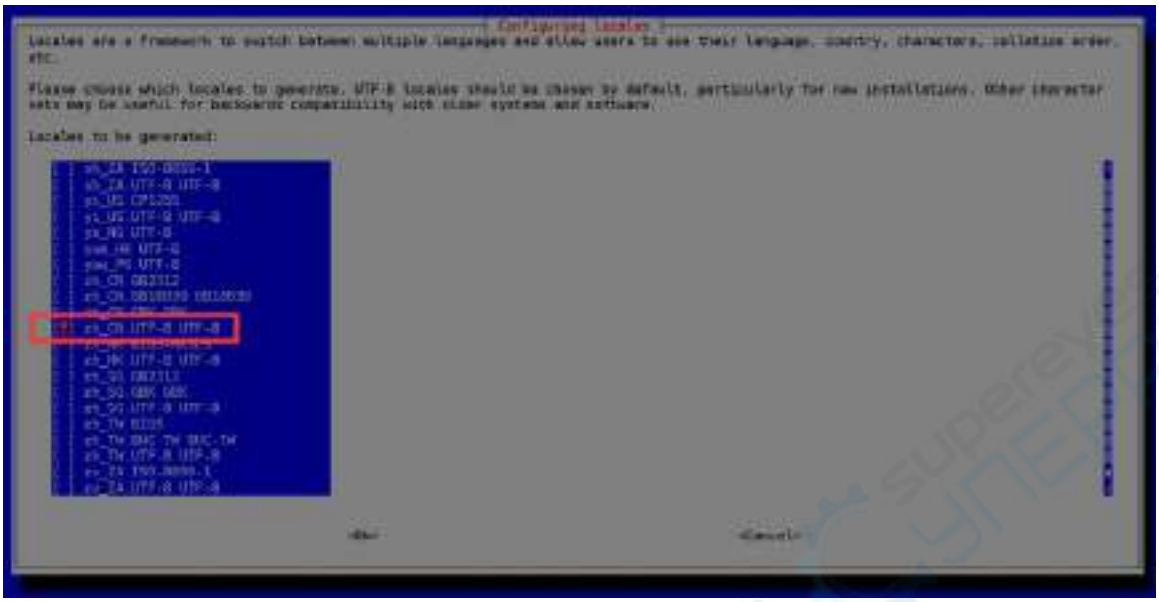
1) 首先设置默认 **locale** 为中文

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）

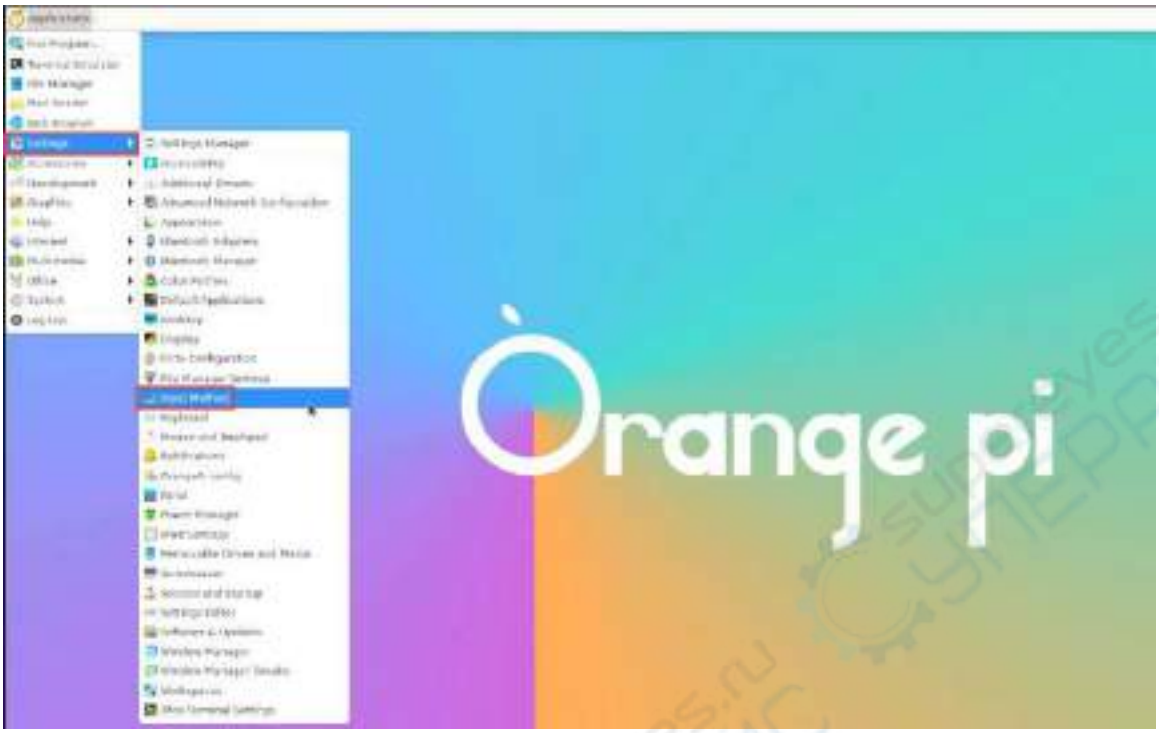


c. 然后设置默认 **locale** 为 **zh_CN.UTF-8**

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.
```

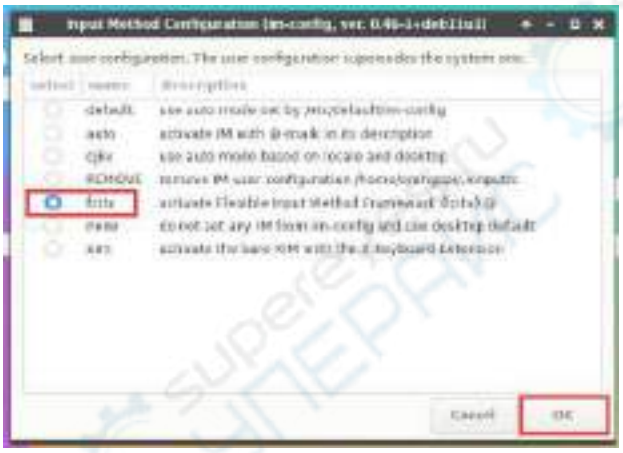
2) 然后打开 **Input Method**



3) 然后选择 **OK**

4) 然后选择 **Yes**

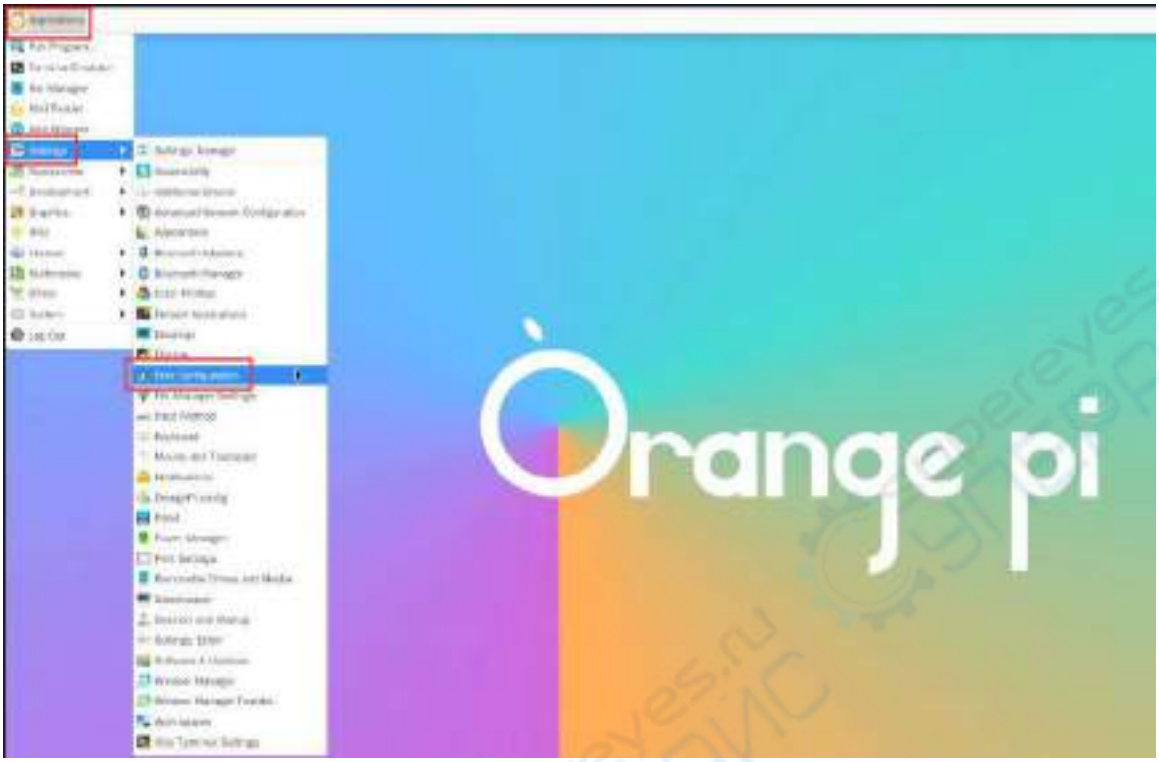
5) 然后选择 **fcitx**



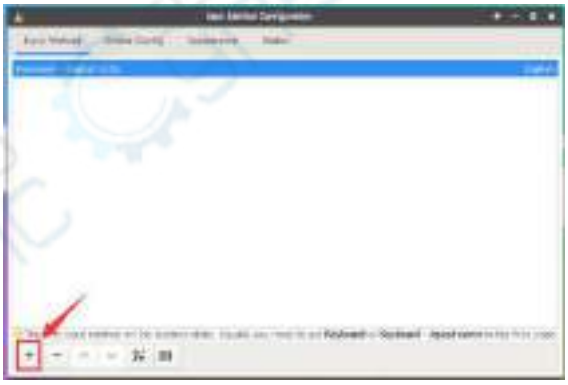
6) 然后选择 **OK**

7) **然后重启 Linux 系统才能使配置生效**

8) 然后打开 **Fcitx configuration**

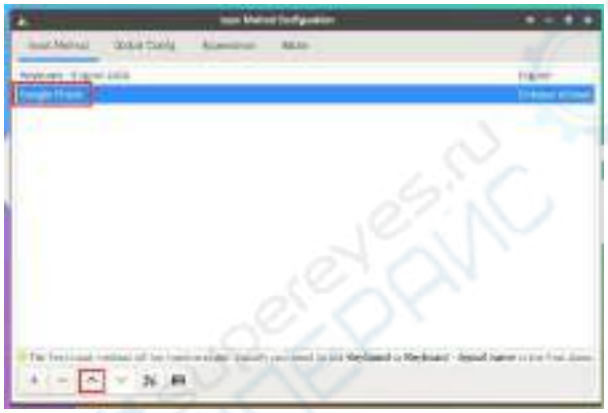


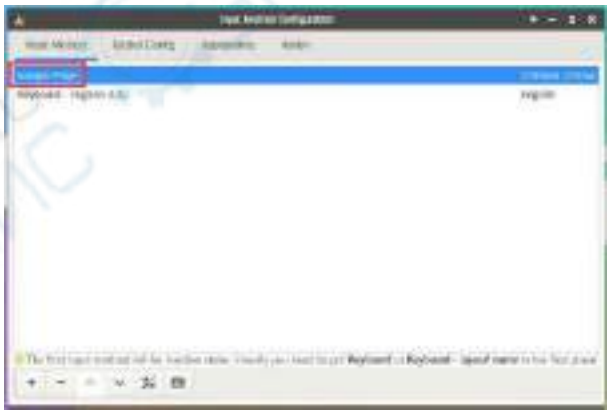
9) 然后点击下图所示位置的+号



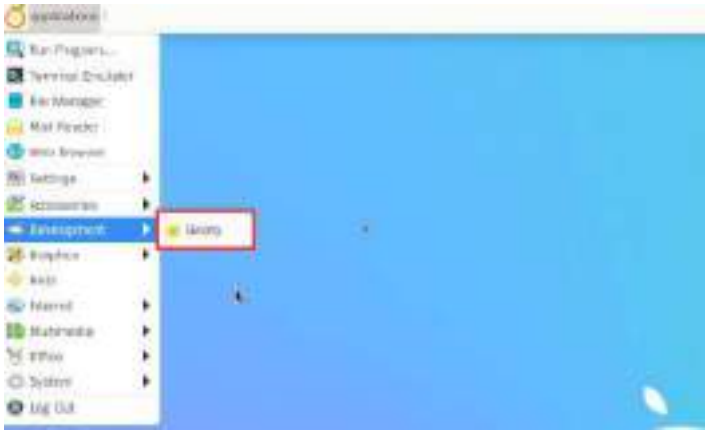
10) 然后搜索 **Google Pinyin** 再点击 **OK**

11) 然后将 **Google Pinyin** 放到最前面

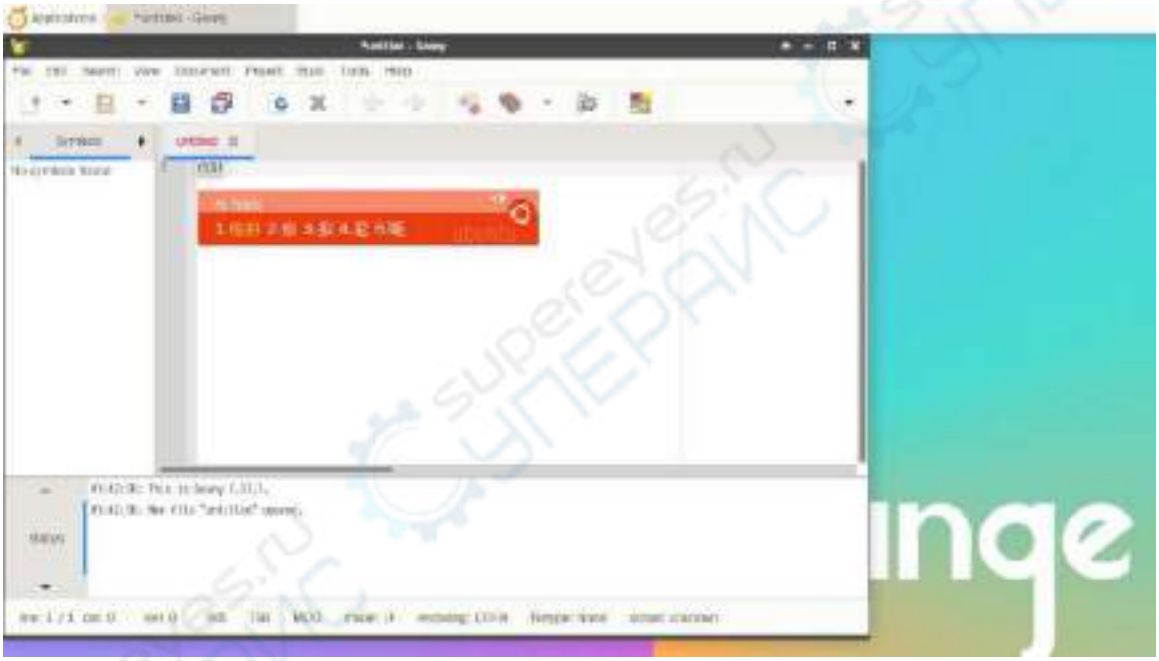




12) 然后打开 **Geany** 这个编辑器测试下中文输入法



13) 中文输入法测试如下所示

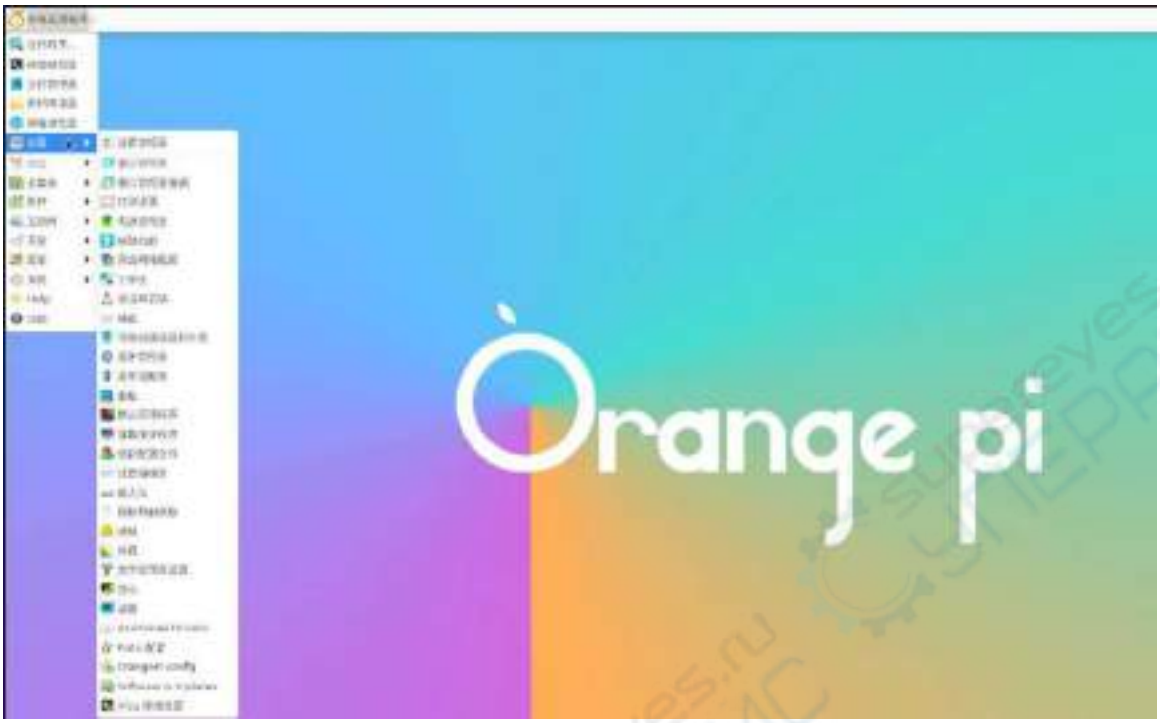


14) 通过 **Ctrl+Space** 快捷键可以切换中英文输入法

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 **zh_CN.UTF-8**

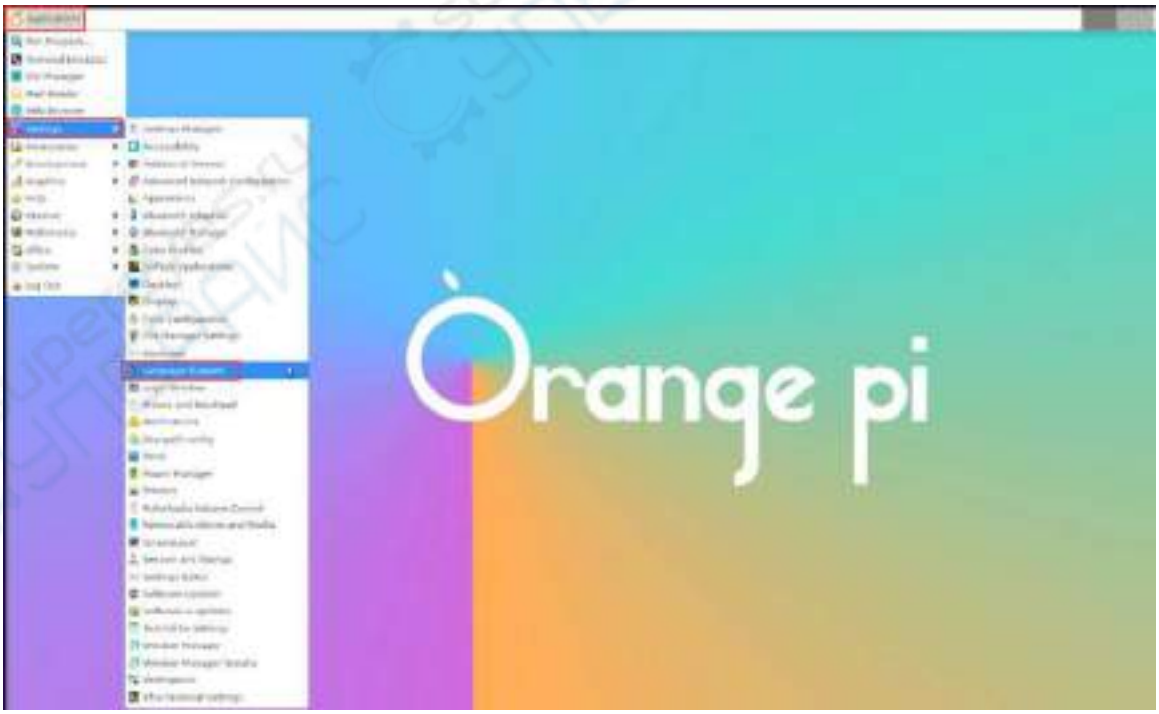
```
orangepi@orangepi:~$ sudo vim /etc/default/locale
#    File generated by update-locale
LC_MESSAGES=zh_CN.UTF-8
LANG=zh_CN.UTF-8
LANGUAGE=zh_CN.UTF-8
```

16) 然后**重启系统**就能看到系统显示为中文了

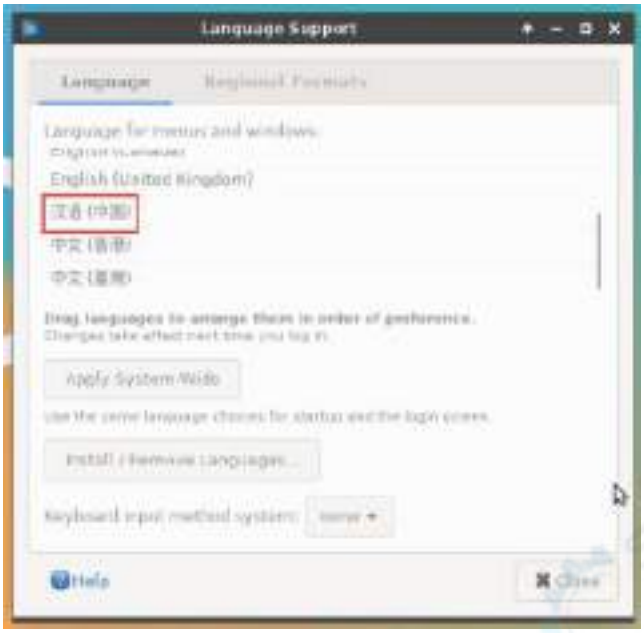


3.24.2. Ubuntu 20.04 系统的安装方法

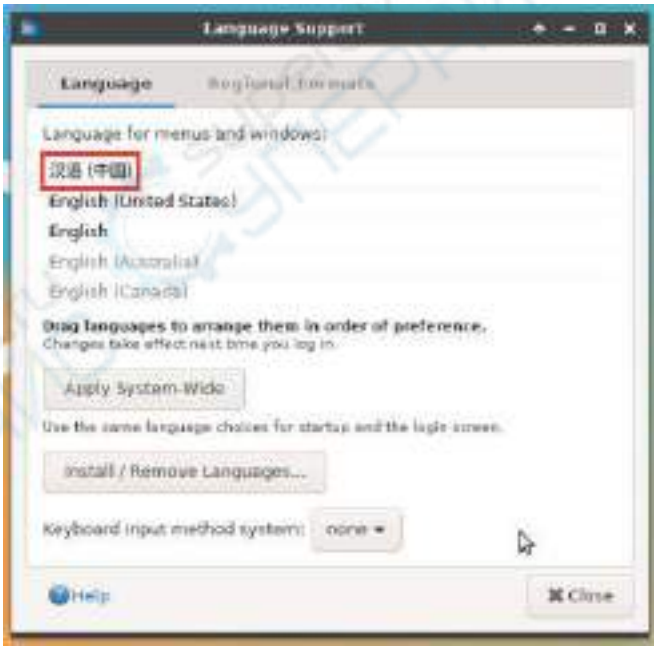
1) 首先打开 **Language Support**



2) 然后找到**汉语（中国）**选项

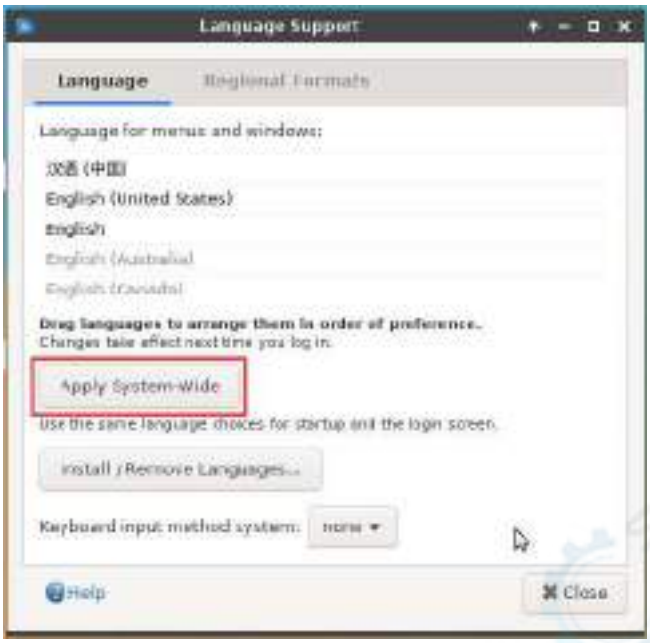


3) 然后请使用鼠标左键选中**汉语（中国）**并按住不动，然后往上将其拖到最开始的位置，拖完后的显示如下图所示：



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



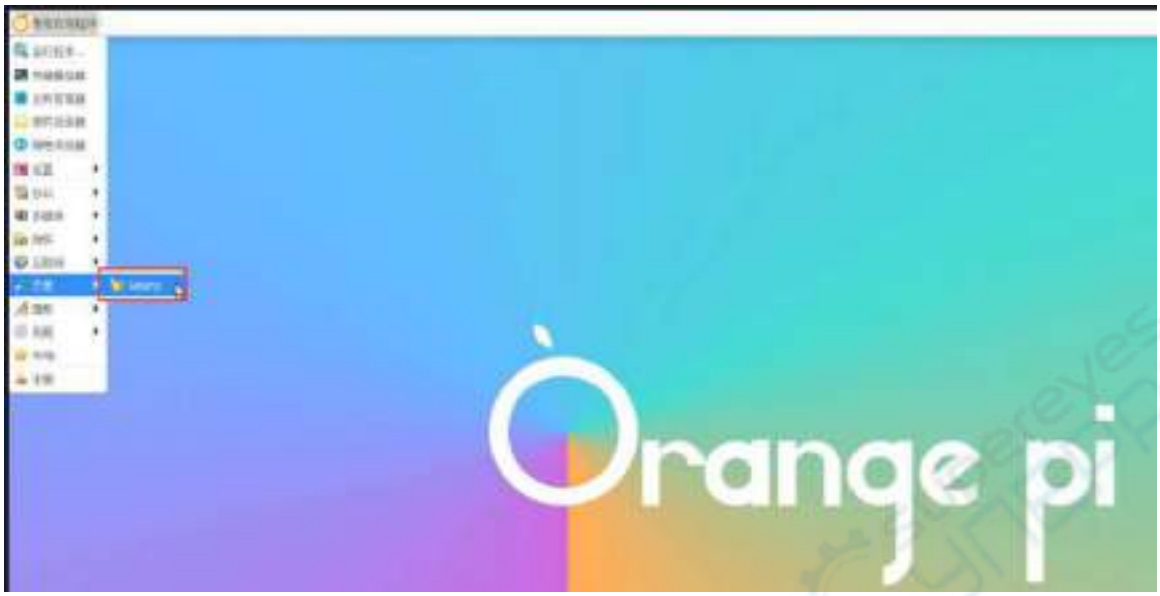
5) 然后设置 **Keyboard input method system** 为 **fcitx**

6) **然后重启 Linux 系统使配置生效**

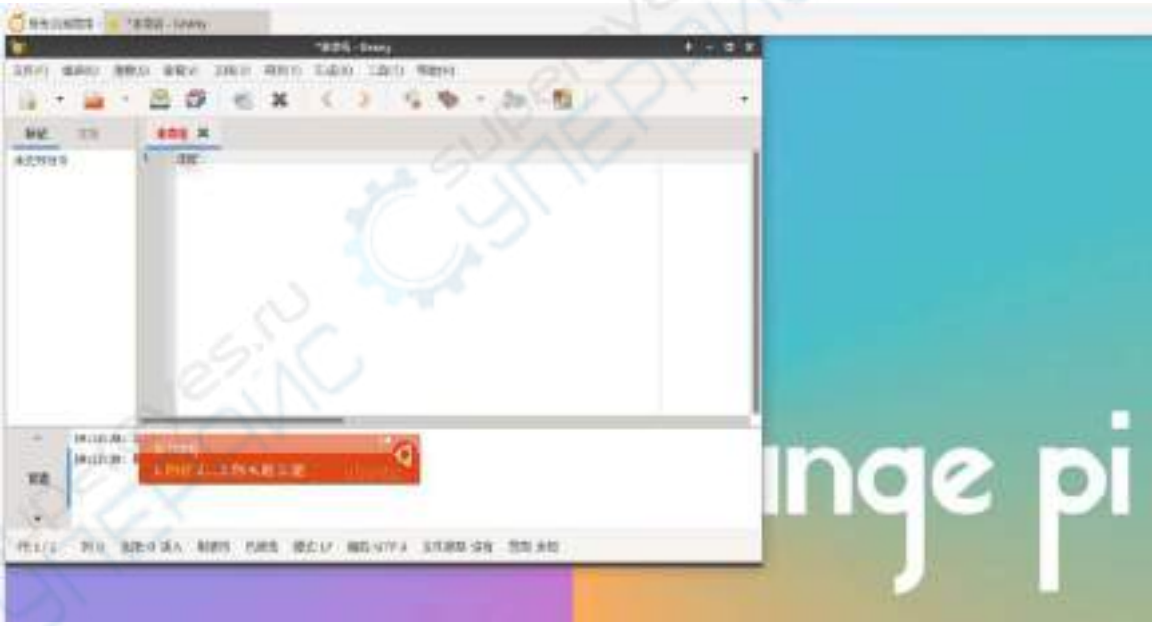
7) 重新进入系统后，在下面的界面请选择**不要再次询问我**，然后请根据自己的喜好决定标准文件夹是否也要更新为中文

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

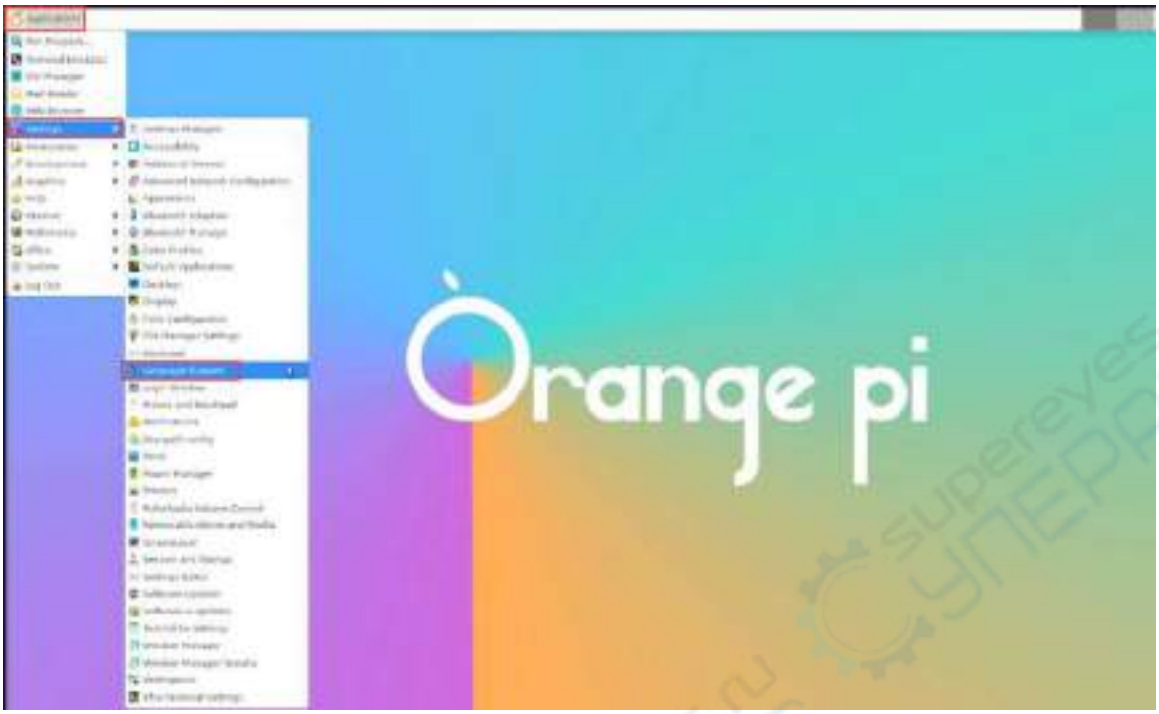


10) 打开 **Geany** 后，默认还是英文输入法，我们可以通过 **Ctrl+Space** 快捷键来切换成中文输入法，然后就能输入中文了



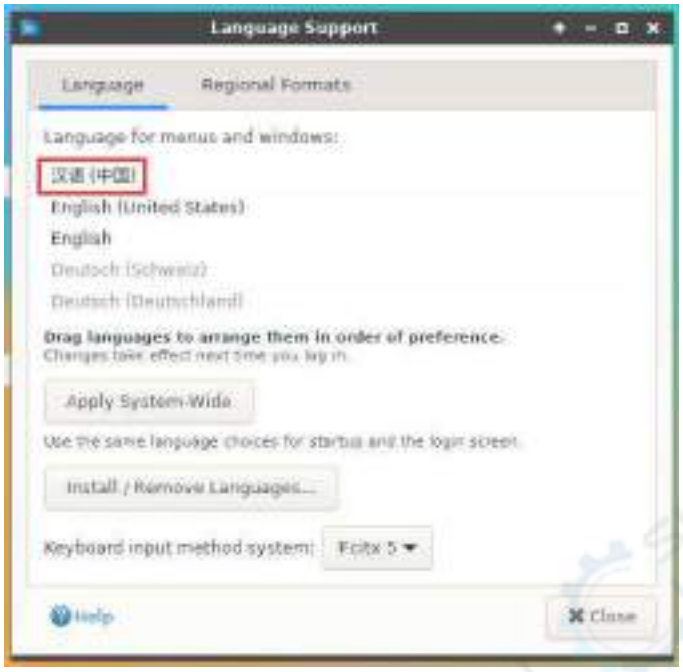
3.24.3. Ubuntu 22.04 系统的安装方法

1) 首先打开 **Language Support**



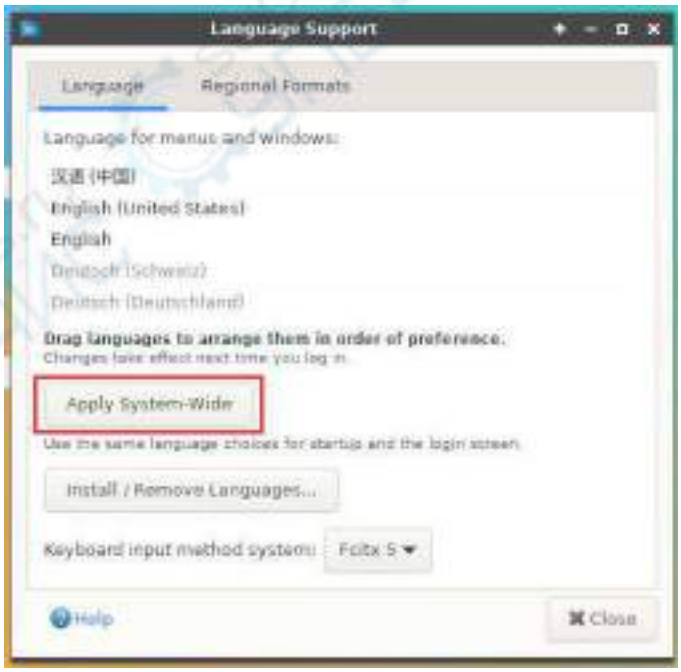
2) 然后找到**汉语（中国）**选项

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

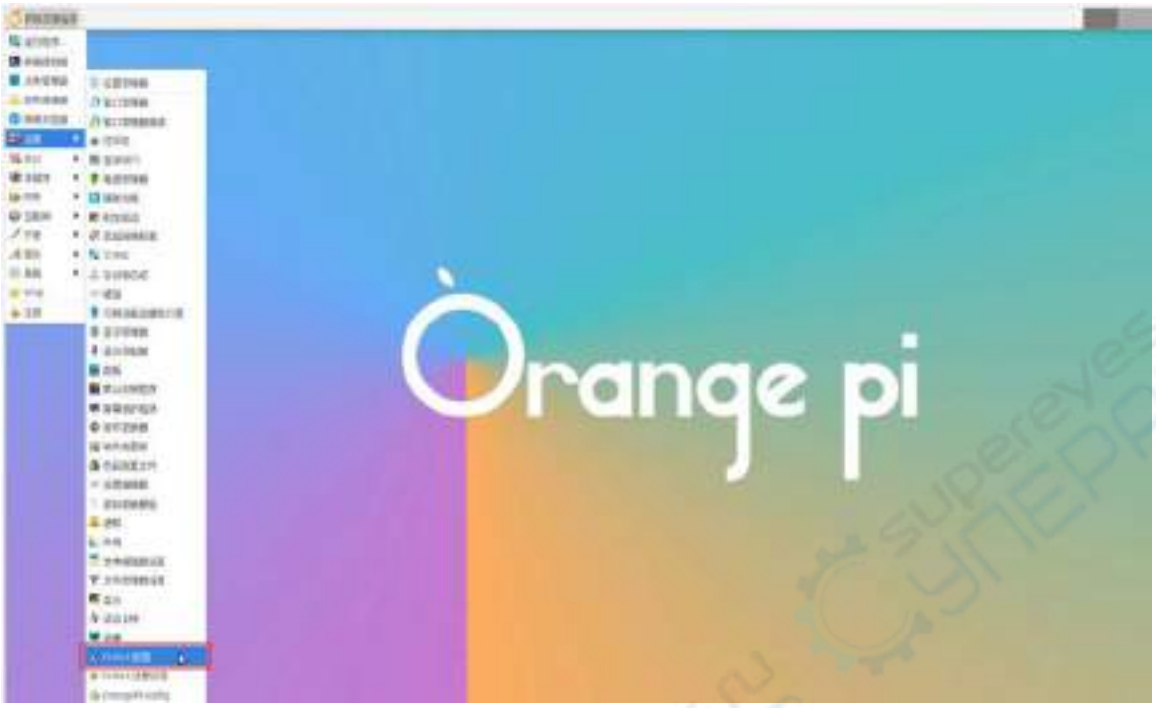


5) **然后重启 Linux 系统使配置生效**

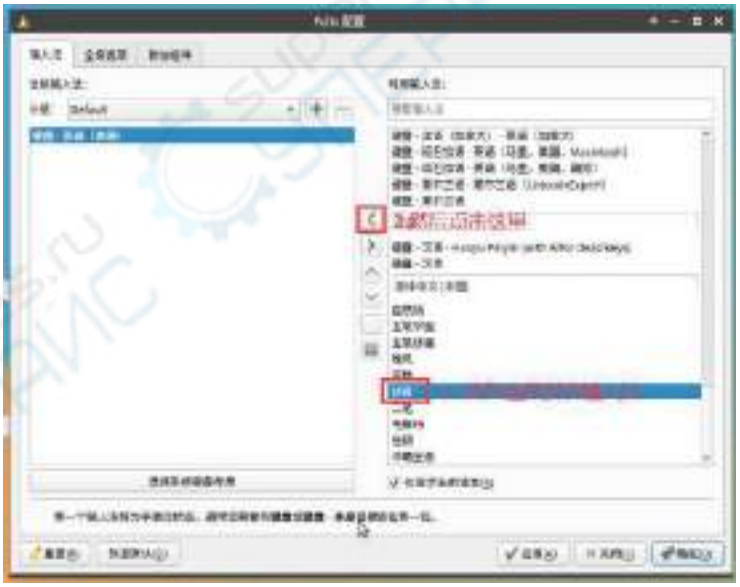
6) 重新进入系统后，在下面的界面请选择**不要再次询问我**，然后请根据自己的喜好决定标准文件夹是否也要更新为中文

7) 然后可以看到桌面都显示为中文了

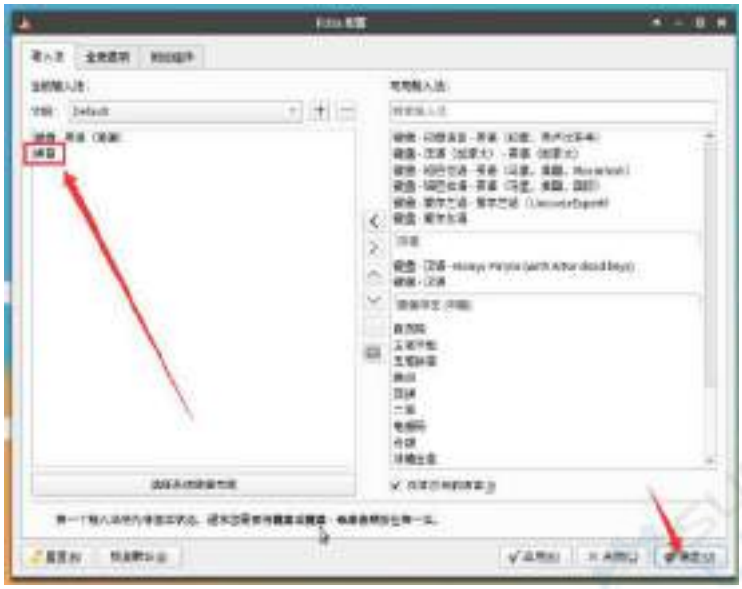
8) 然后打开 Fcitx5 配置程序



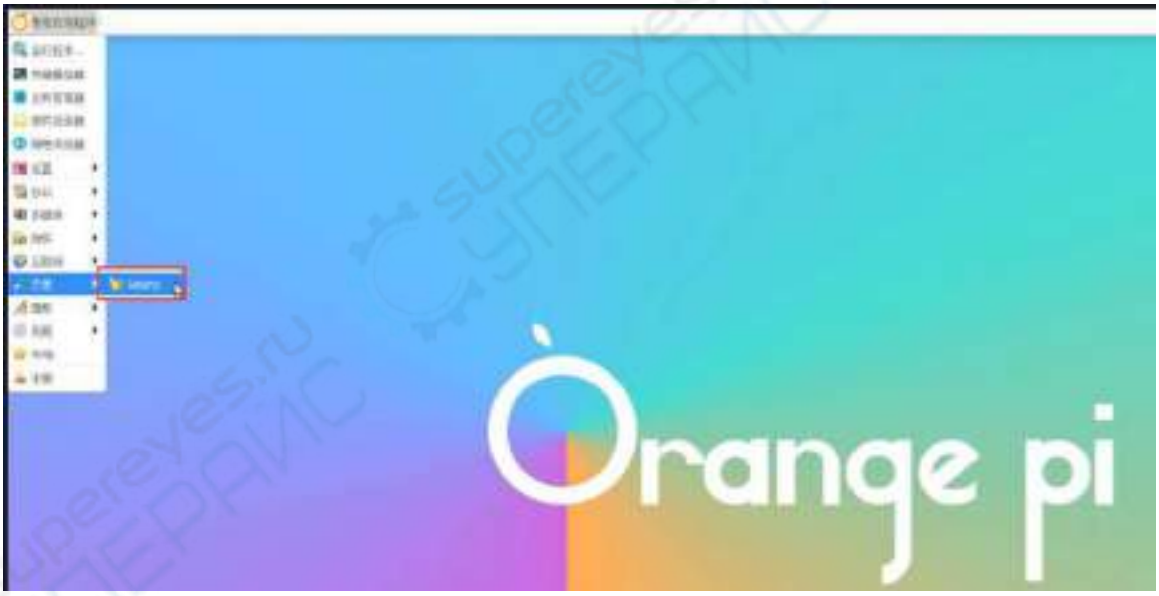
9) 然后选择使用拼音输入法



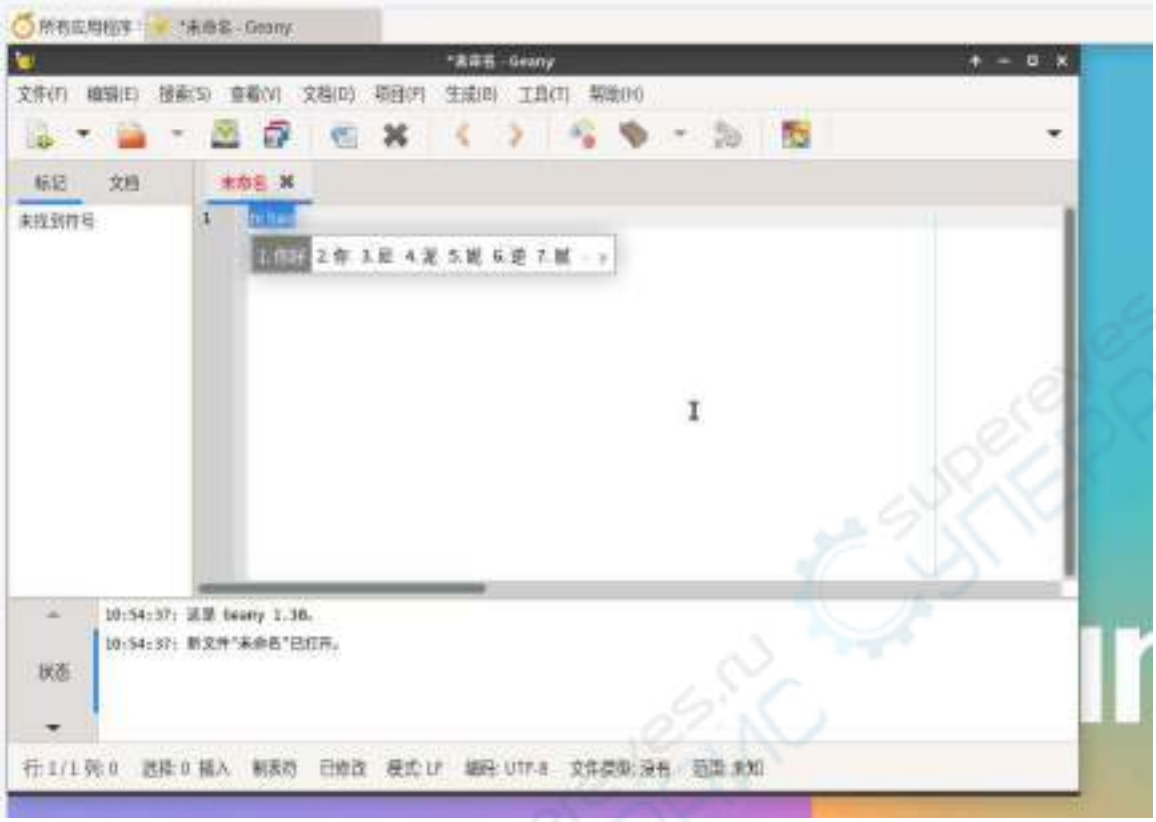
10) 选择后的界面如下所示，再点击确定即可



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



3.25. 远程登录 Linux 系统桌面的方法

3.25.1. 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：

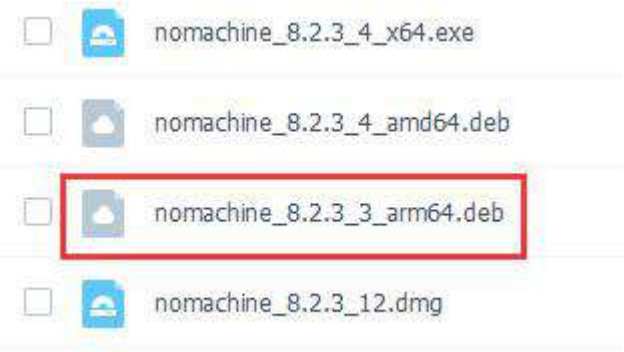
注意，这个下载链接可能会变，请认准 Armv8/Arm64 版本的 deb 包。

https://downloads.nomachine.com/download/?id=116&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_8.2.3_3_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**

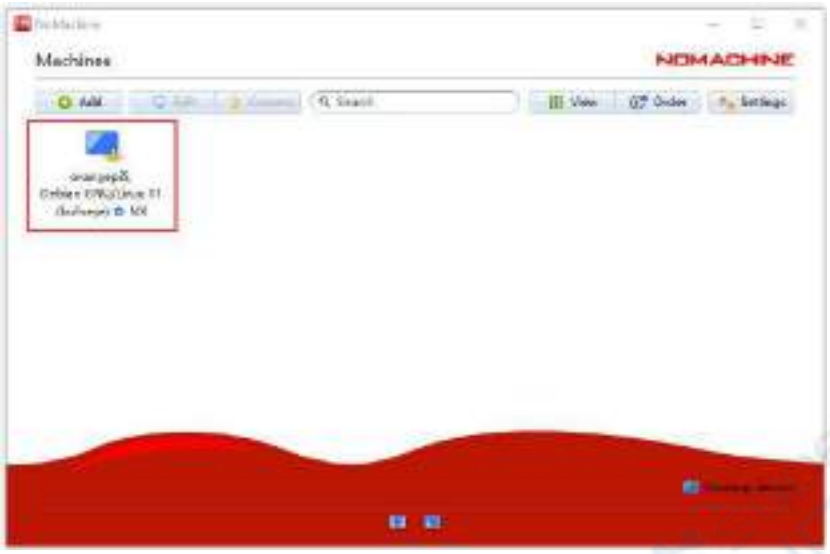
```
orangepi@orangepi:~$ sudo dpkg -i nomachine_8.2.3_3_arm64_arm64.deb
```

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

https://downloads.nomachine.com/download/?id=8

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

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



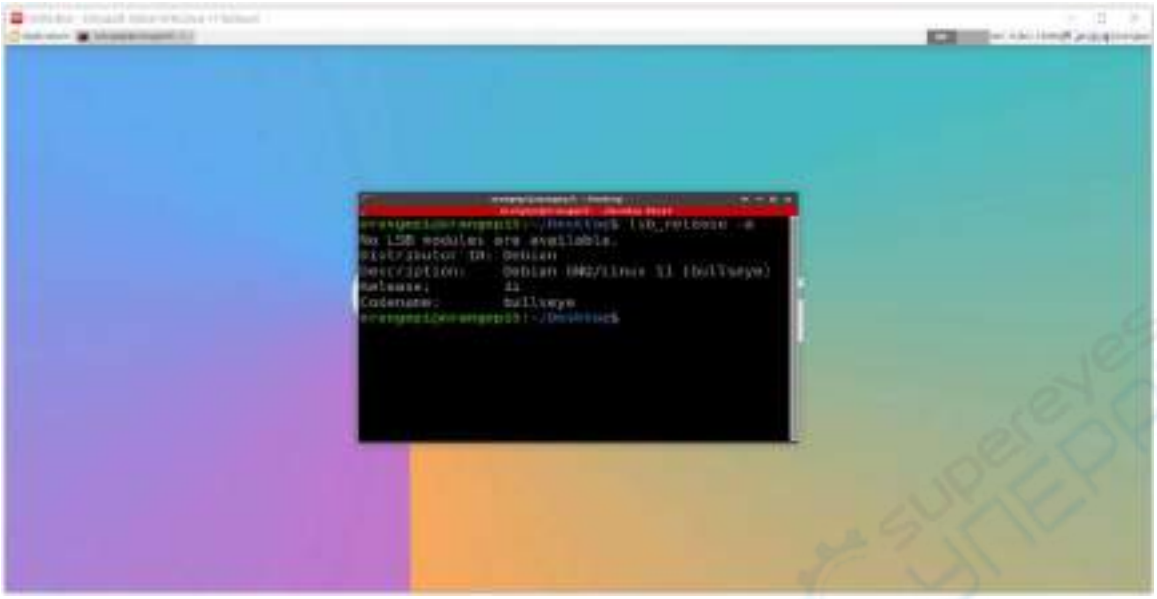
6) 然后点击 **OK**

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 **Login** 开始登陆

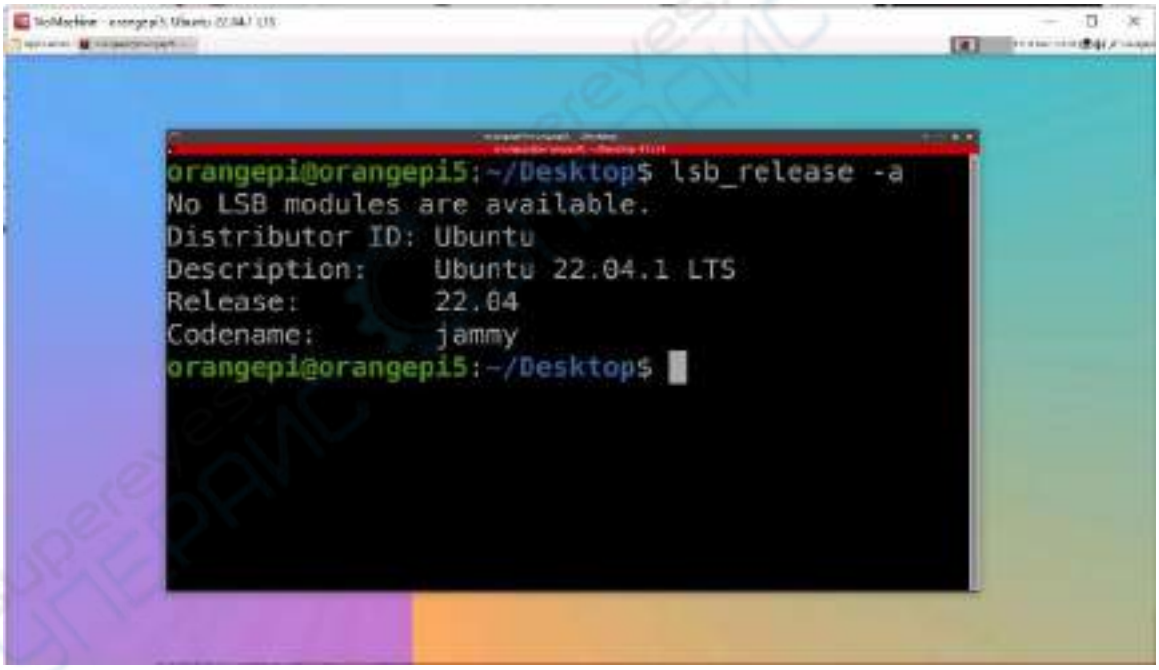
8) 然后在接下来的界面中都点击 OK

9) 最后就能看到开发板 Linux 系统的桌面了

a. Debian11



b. Ubuntu22.04



3.25.2. 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。

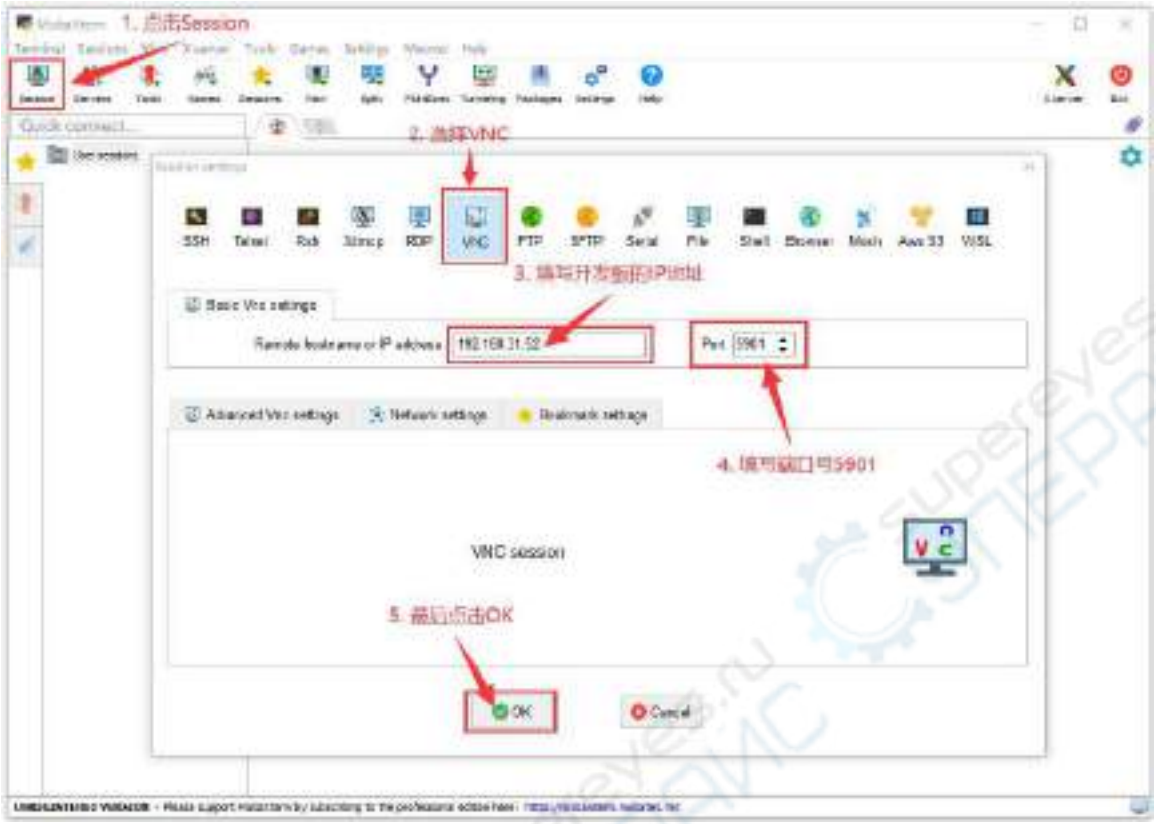
Ubuntu20.04 测试 VNC 很多问题，请不要使用这种方法。

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 orangepi5b:1

Creating default startup script /root/.vnc/xstartup
Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/orangepi5b:1.log

Killing Xtightvnc process ID 3047

New 'X' desktop is orangepi5b:1

Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/orangepi5b: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：

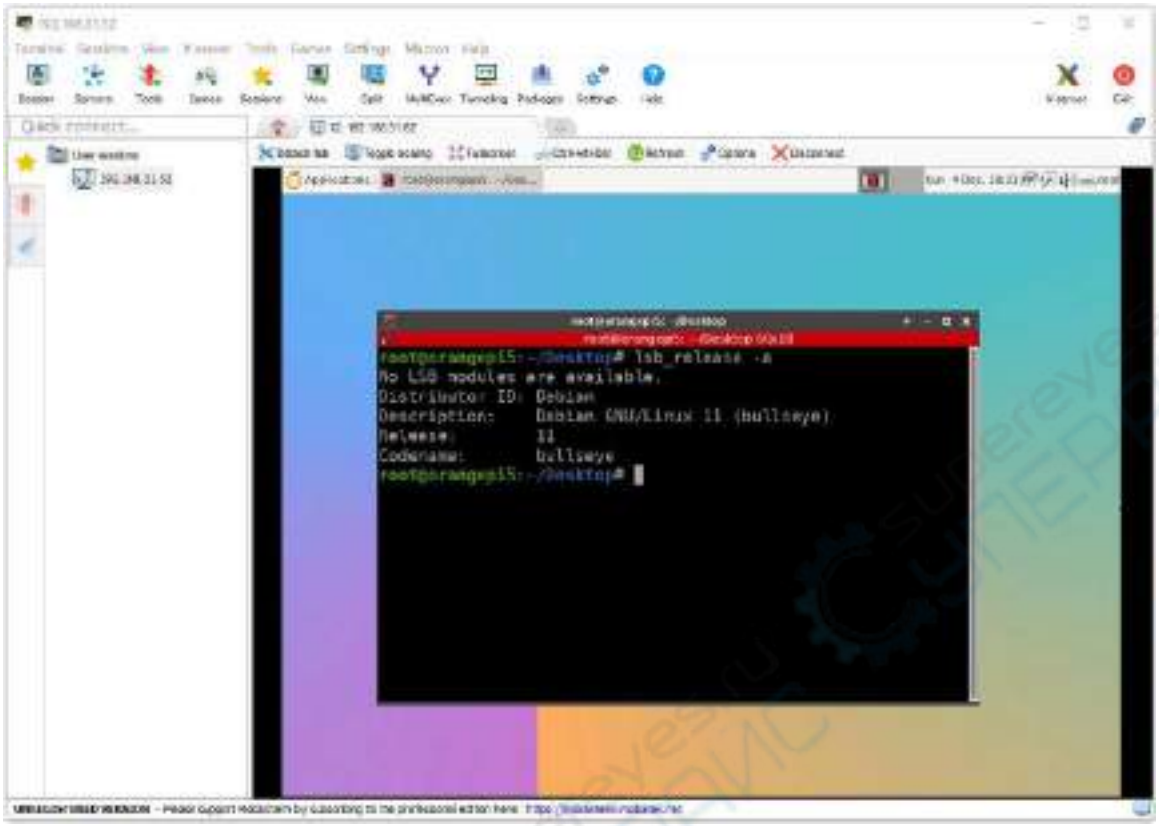
a. 首先点击 Session，然后选择 VNC，再填写开发板的 IP 地址和端口，最后点击 OK 确认



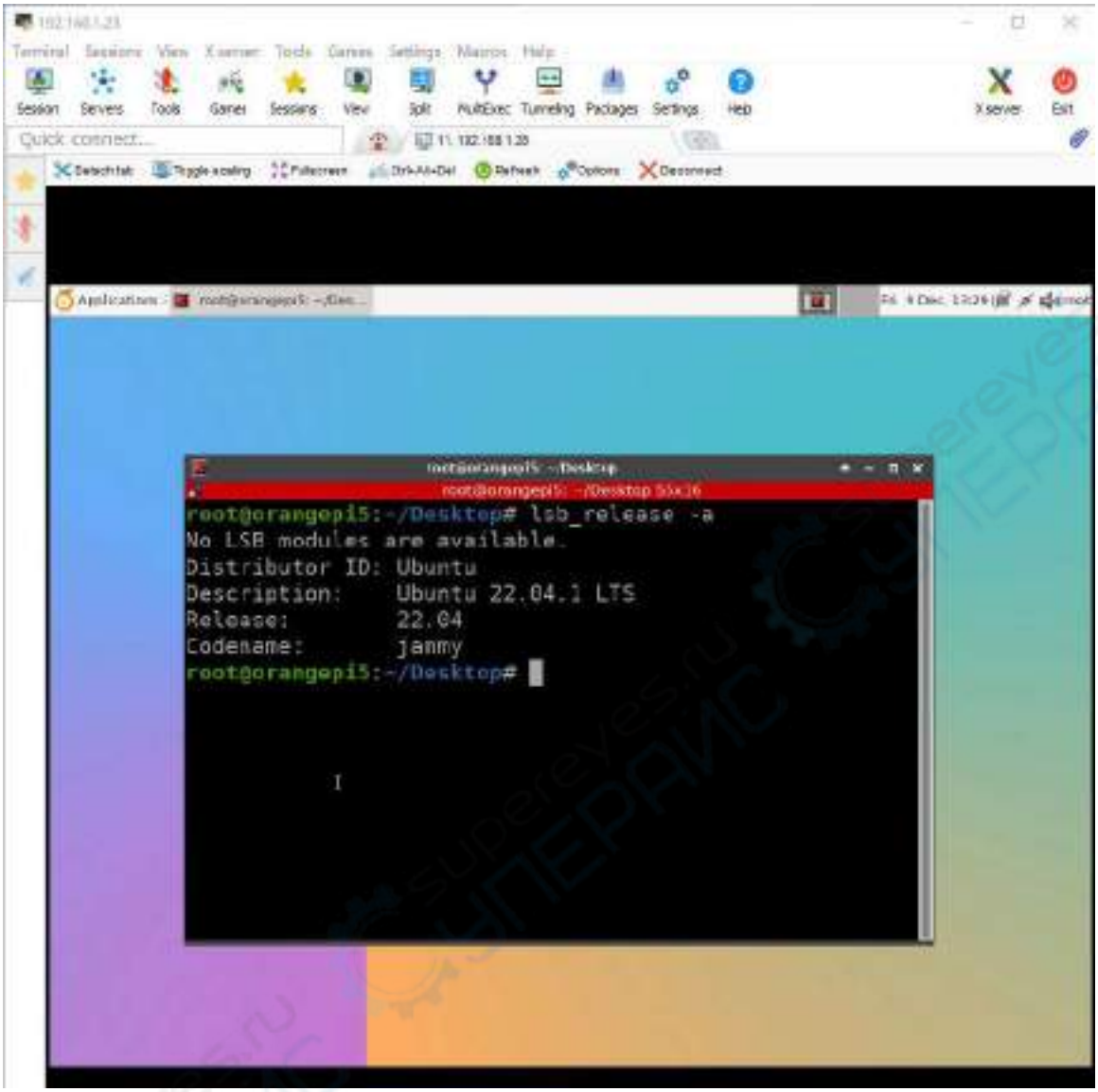
b. 然后输入前面设置的 VNC 的密码

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

a) Debian11 登录显示如下所示



b) Ubuntu22.04 登录显示如下所示



3) 使用 Windows 自带的**远程桌面连接**应用登录开发板 Linux 系统桌面的步骤为

a. 首先打开 Windows 自带的**远程桌面连接**

b. 然后输入开发板的 IP 地址

c. 然后根据下图说明设置连接信息

a) **Session：**需要选择 vnc-any

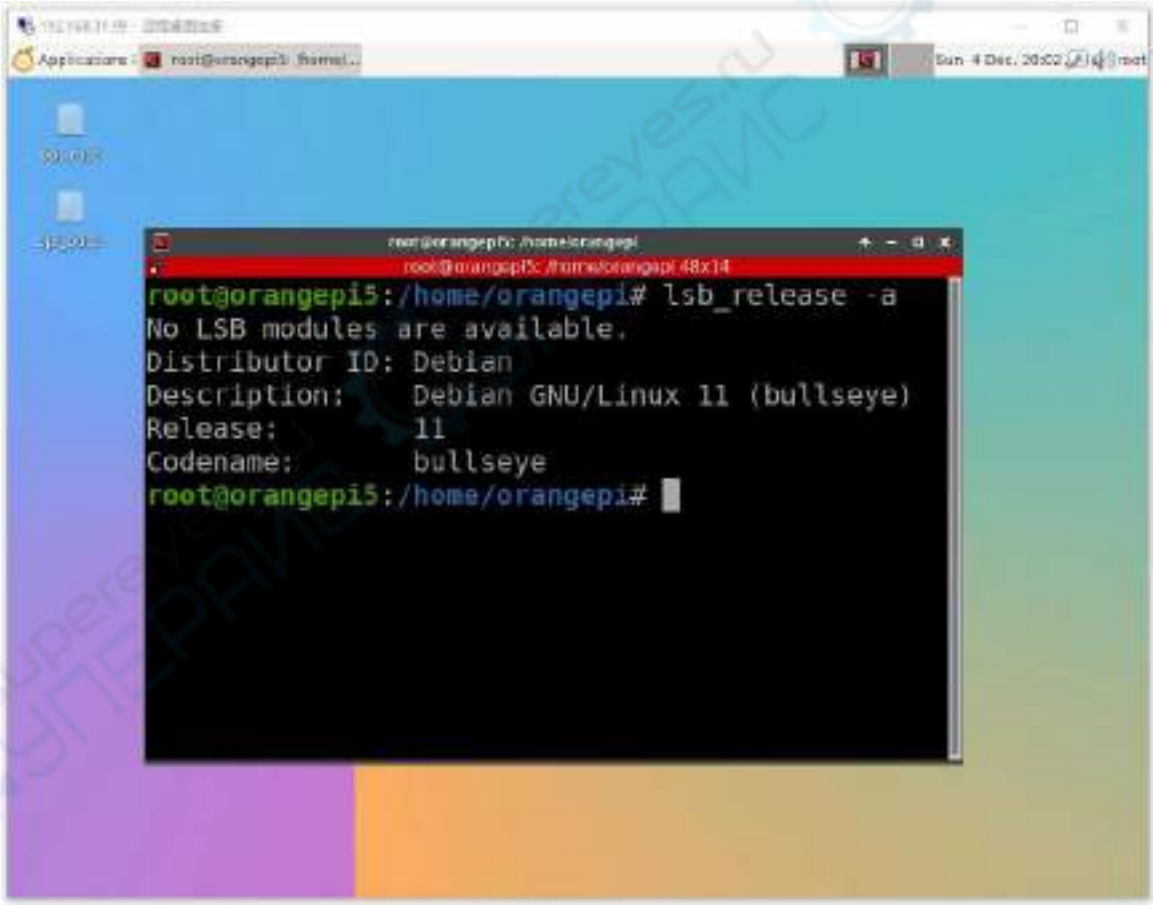
b) **ip：**可以输入 127.0.0.1 或者开发板的 IP 地址

c) **port：**一般为 5901

d) **password：**需要输入 vnc 的密码

d. 成功登录开发板 Linux 系统桌面的显示如下图所示

a) Debian11 登录显示如下所示



b) Ubuntu22.04 目前无法使用，请不要使用这种方法

3.26. Linux 系统支持的部分编程语言测试

3.26.1. 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!
```

3.26.2. 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!
```

3.26.3. Ubuntu Jammy 系统

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

a. gcc 的版本如下所示

```
orangepi@orangepi:~$ gcc --version
gcc (Ubuntu 11.2.0-19ubuntu1) 11.2.0
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

b. 编写 C 语言的 **hello_world.c** 程序

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

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

```
        return 0;
}
```

c. 然后编译运行 **hello_world.c**

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

5) Ubuntu Jammy 默认安装有 Python3

a. Python3 具体版本如下所示

```
orangepi@orangepi:~$ python3
Python 3.10.4 (main, Apr  2 2022, 09:04:19) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

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

```
orangepi@orangepi:~$ vim hello_world.py
print('Hello World!')
```

c. 运行 **hello_world.py** 的结果如下所示

```
orangepi@orangepi:~$ python3 hello_world.py
Hello World!
```

6) Ubuntu Jammy 默认没有安装 Java 的编译工具和运行环境

a. 可以使用下面的命令安装 openjdk-18

```
orangepi@orangepi:~$ sudo apt install -y openjdk-18-jdk
```

b. 安装完后可以查看下 Java 的版本

```
orangepi@orangepi:~$ java --version
openjdk 18-ea 2022-03-22
OpenJDK Runtime Environment (build 18-ea+36-Ubuntu-1)
OpenJDK 64-Bit Server VM (build 18-ea+36-Ubuntu-1, mixed mode, sharing)
```

c. 编写 Java 版本的 **hello_world.java**

```
orangepi@orangepi:~$ vim hello_world.java
public class hello_world
{
        public static void main(String[] args)
        {
```

```
                System.out.println("Hello World!");
        }
}
```

d. 然后编译运行 **hello_world.java**

```
orangepi@orangepi:~$ javac hello_world.java
orangepi@orangepi:~$ java hello_world
Hello World!
```

3.27. 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
```

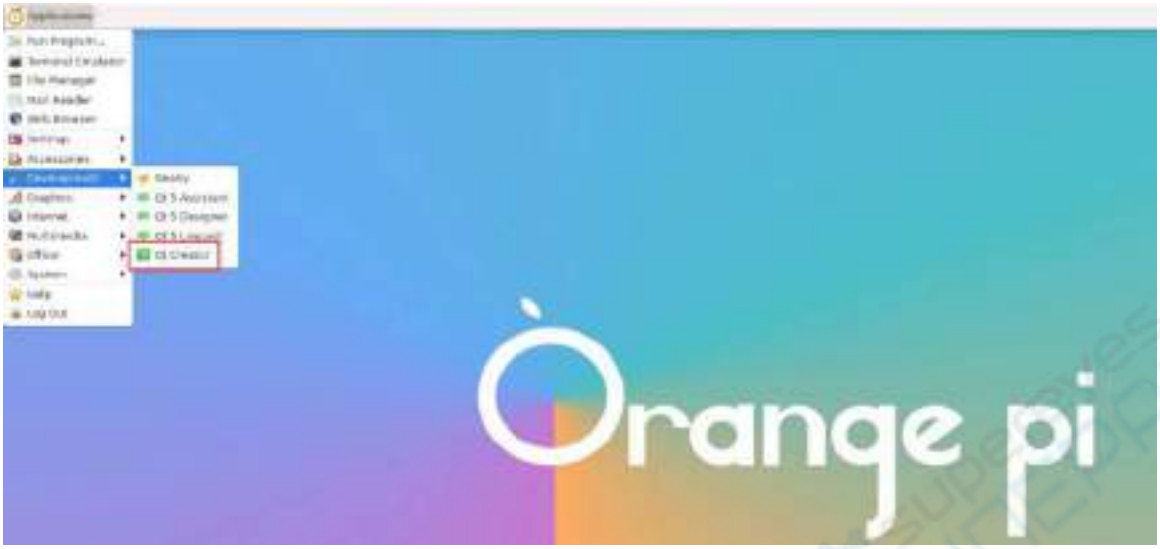
b. Ubuntu22.04 自带的 QT 版本为 **5.15.3**

```
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
```

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

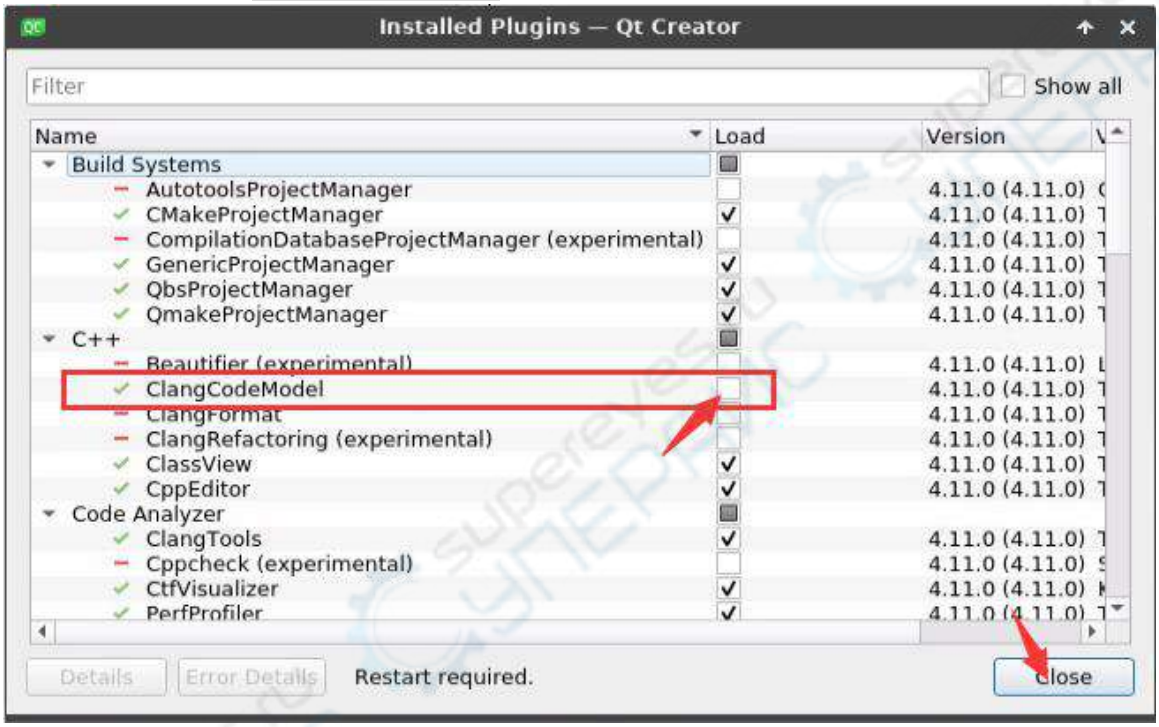
b. QT Creator 在 **Ubuntu22.04** 中的默认版本如下所示

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

6) 然后设置下 QT

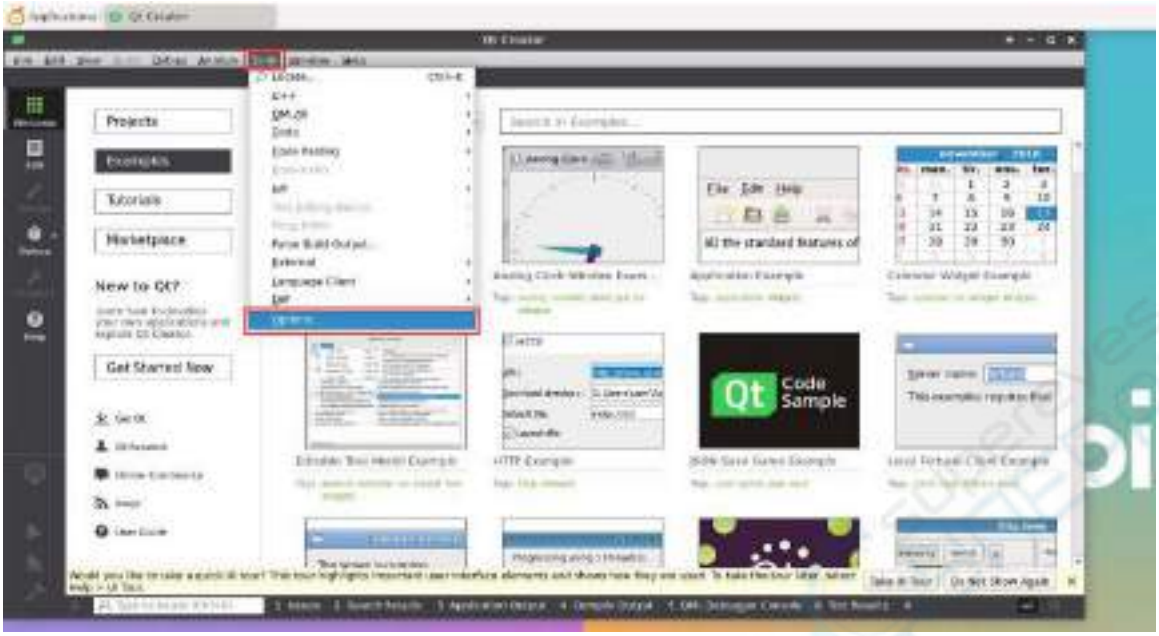
a. 首先打开 **Help->About Plugins...**

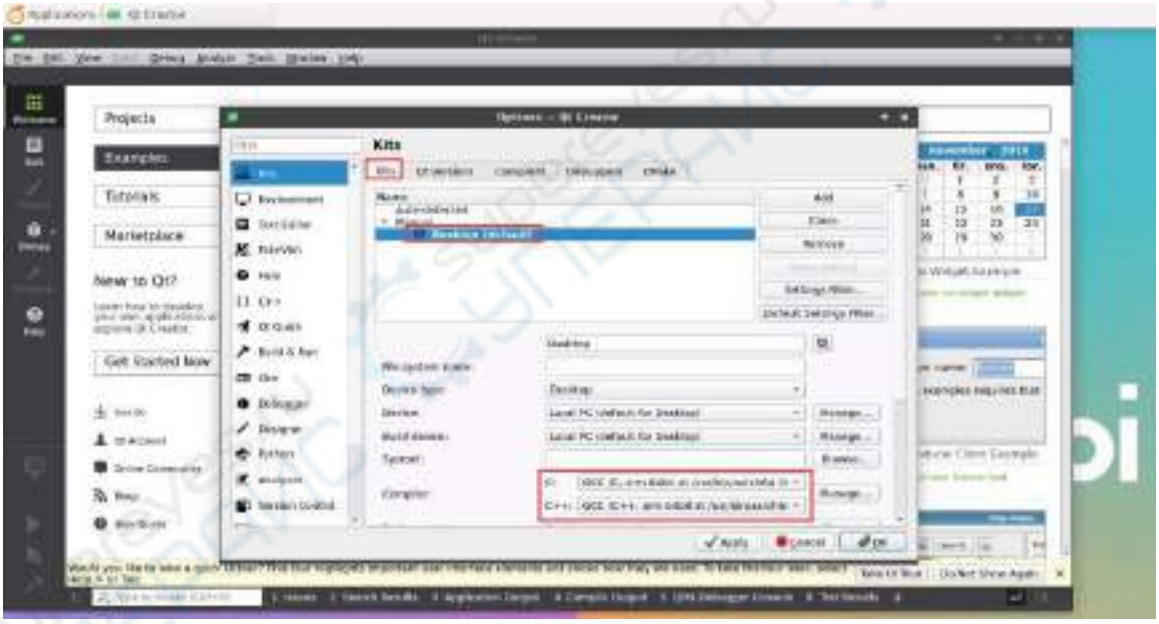
b. 然后去掉 **ClangCodeModel** 的那个勾



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

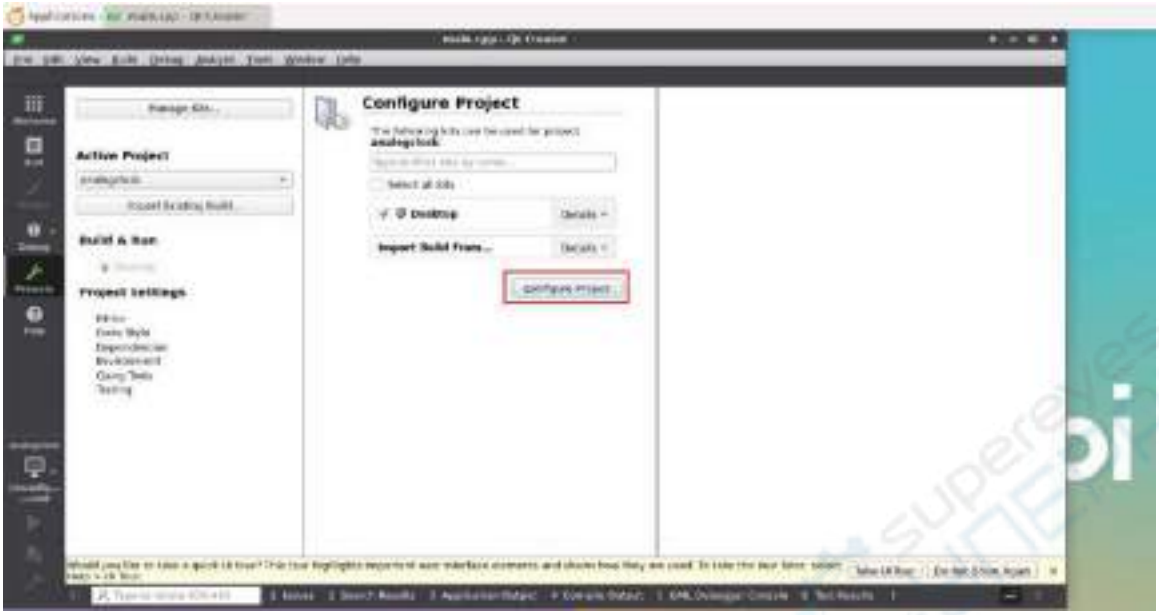




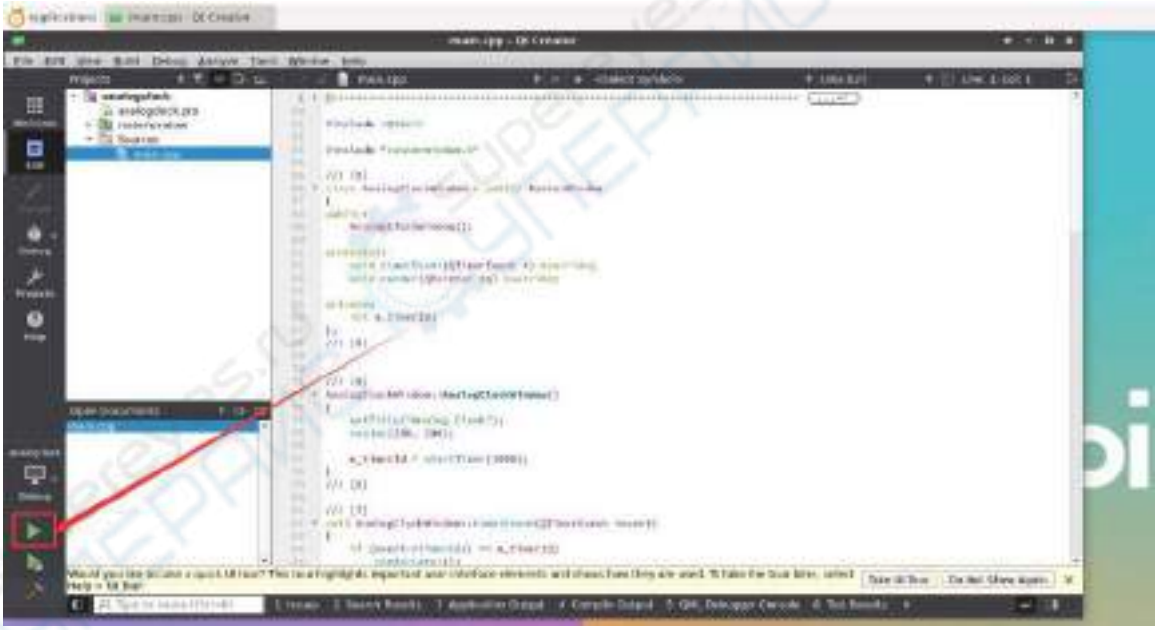
7) 然后就可以打开一个示例代码

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

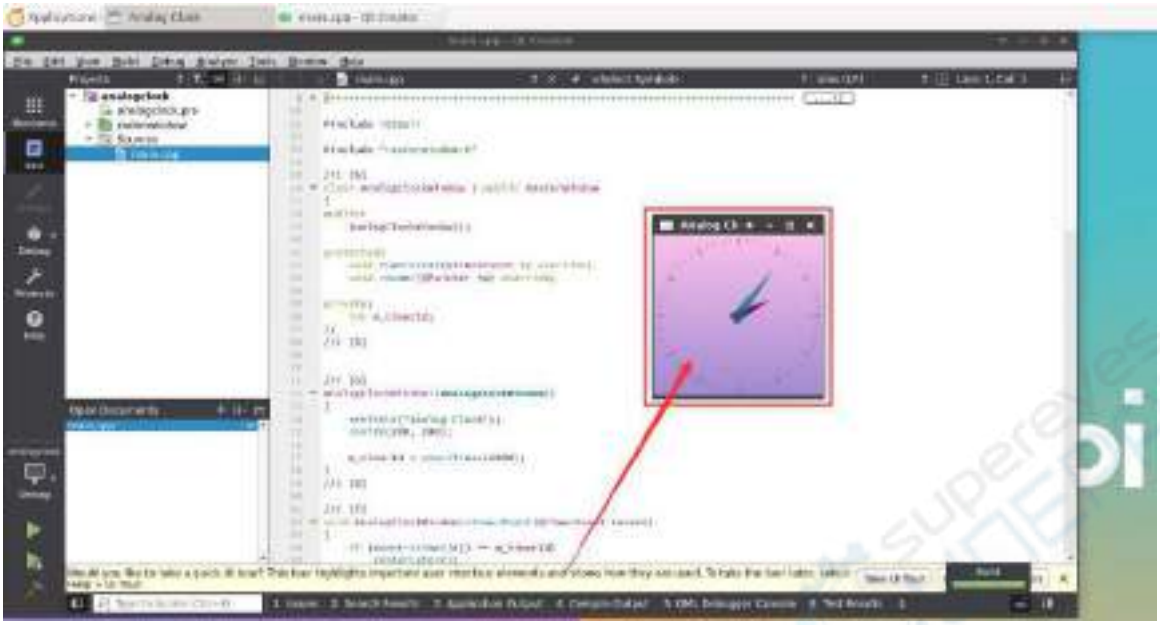
9) 然后点击下 **Configure Project**



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

3.28. ROS 安装方法

3.28.1. 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

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@orangepi5b:~$ **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

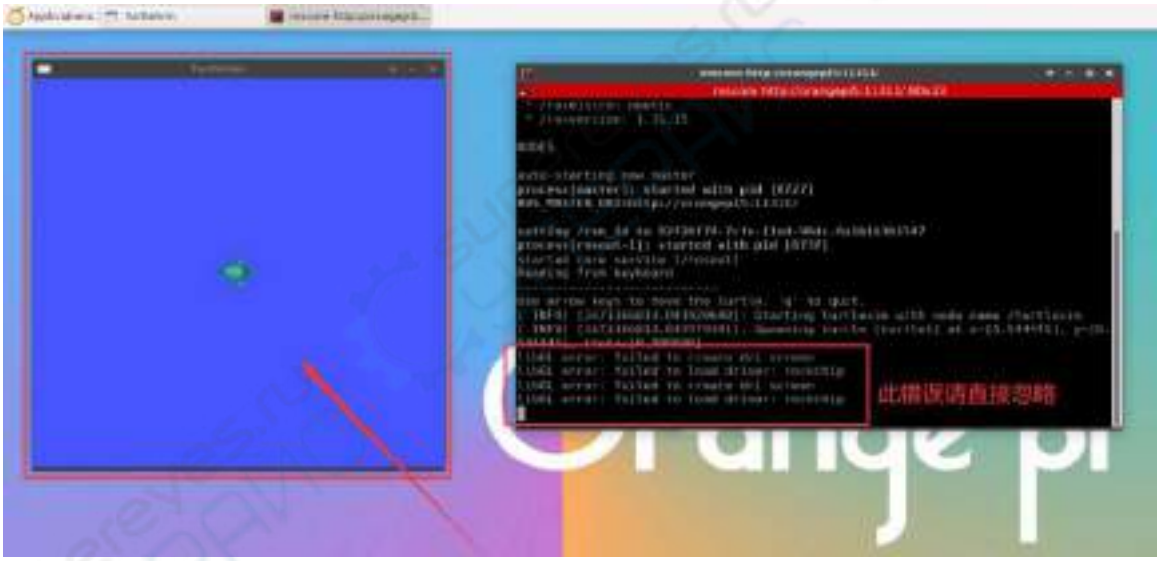
```
orangepi@orangepi:~$ source /opt/ros/noetic/setup.bash
orangepi@orangepi:~$ sudo rosdep init
Wrote /etc/ros/rosdep/sources.list.d/20-default.list
Recommended: please run

        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
```

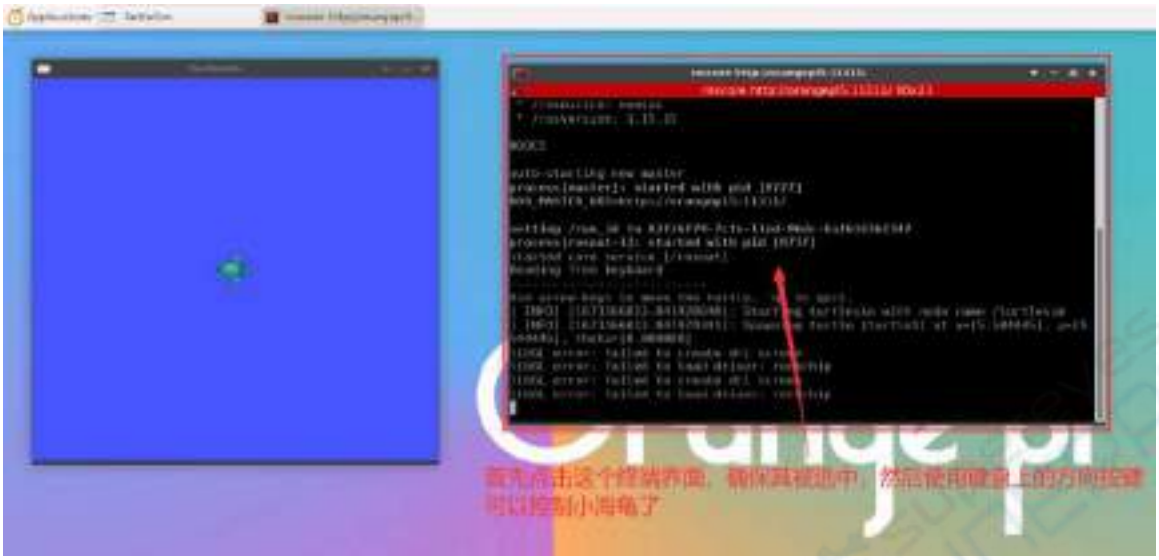
6) 然后在**桌面**中打开一个命令行终端窗口，再使用 **test_ros.sh** 脚本可以启动一个小海龟的例程来测试下 ROS 是否能正常使用

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

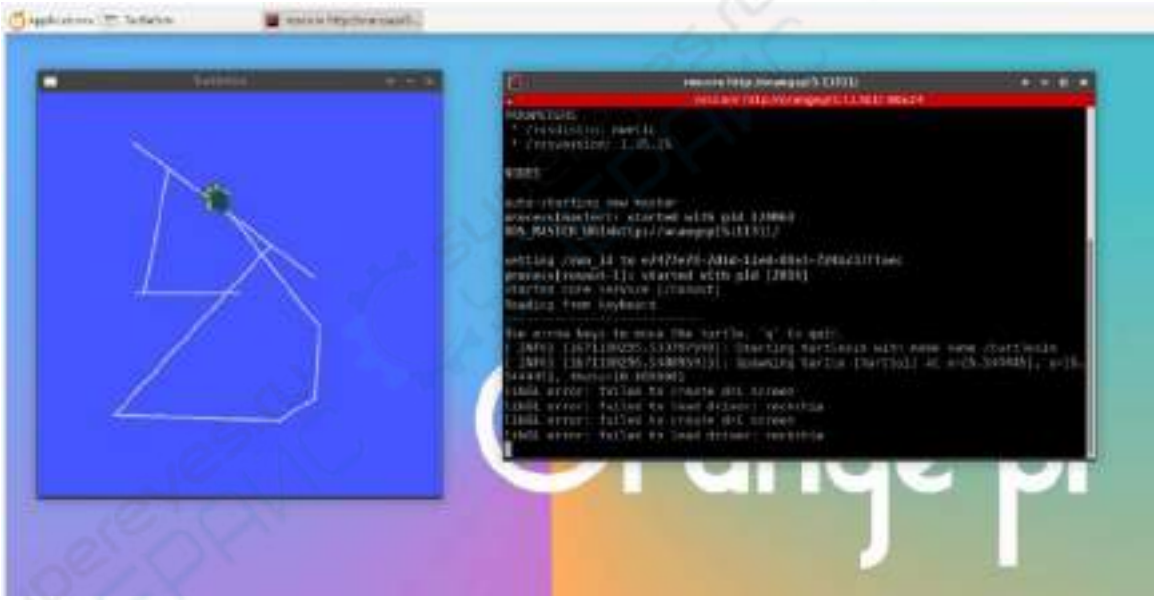
7) 运行完 **test_ros.sh** 脚本后，会弹出下图所示的一个小海龟



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



3.28.2. 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

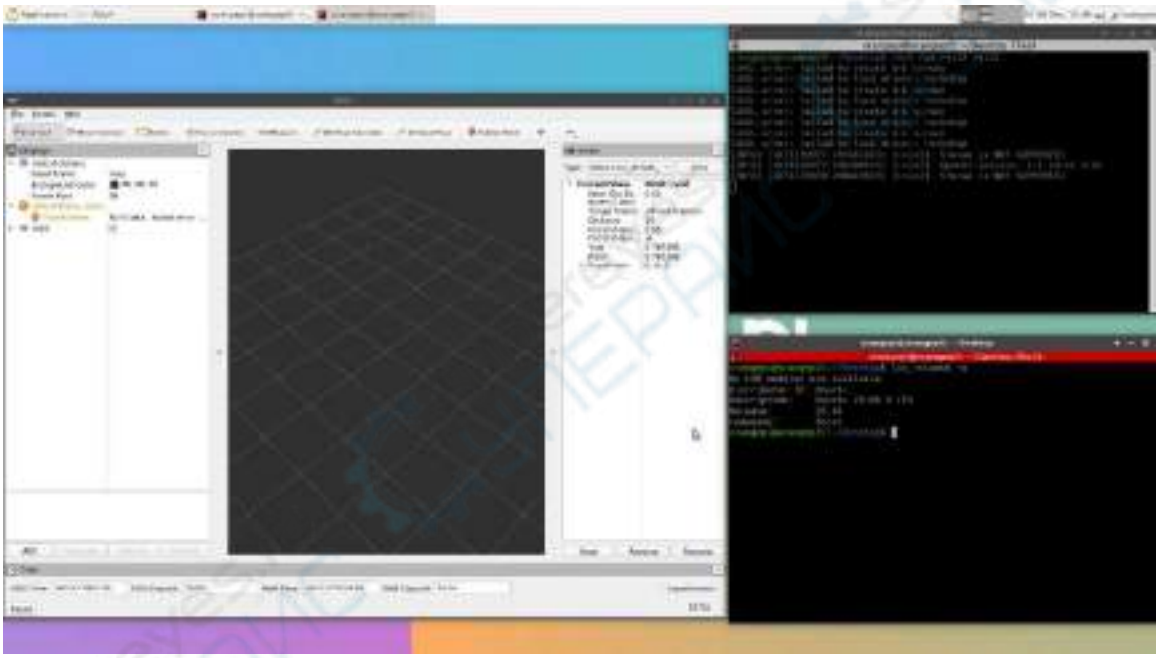
```
usage: ros2 [-h] Call `ros2 <command> -h` for more detailed usage. ...

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@orangepi5b:~$ 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

3.28.3. 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.
```

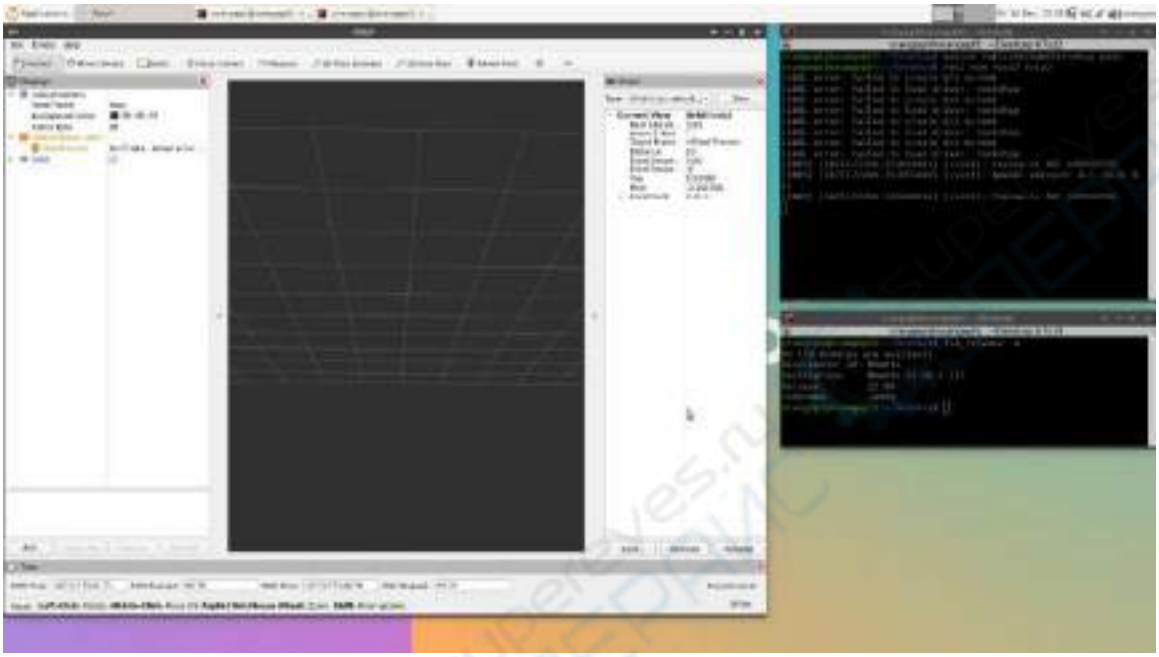
3) 然后可以使用 **test_ros.sh** 脚本测试下 ROS 2 是否安装成功，如果能看到下面的打印，说明 ROS 2 能正常运行

```
orangepi@orangepi5b:~$ 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
```

3.29. 安装内核头文件的方法

1) 内核头文件的获取方法有两种：

a. 方法一：从开发板资料下载页面的官方工具中下载。

b. 方法二：使用 orangepi-build 编译内核源码会自动生成内核头文件的 deb 包，具体方法请参考 **4.4.编译 linux 内核**一小节的说明。

2) 然后将内核头文件 deb 包上传到开发板的 Linux 系统中，上传方法可以参考**上传文件到开发板 Linux 系统中的方法**一小节的说明

3) 然后使用下面的命令安装内核头文件 deb 包

内核头文件 deb 包的名字需要替换为实际的名字，请不要照抄。

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

4) 安装完后在**/usr/src** 下就能看到内核头文件所在的文件夹

```
orangepi@orangepi:~$ ls /usr/src
linux-headers-5.10.110-rockchip-rk3588
```

5) 然后可以编写一个 hello 内核模块测试下内核头文件

a. 首先编写 hello 内核模块的代码，如下所示：

```
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测试。


```
orangepi@orangepi:~$ make
make -C /lib/modules/5.10.110-rockchip-rk3588/build M=/home/orangepi modules
make[1]: Entering directory '/usr/src/linux-headers-5.10.110-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-5.10.110-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
```

3.30. 10.1 寸 MIPI LCD 屏幕的使用方法

3.30.1. 10.1 寸 MIPI 屏幕的组装方法

1) 首先准备需要的配件

b. 屏幕转接板+31pin 转 40pin 排线

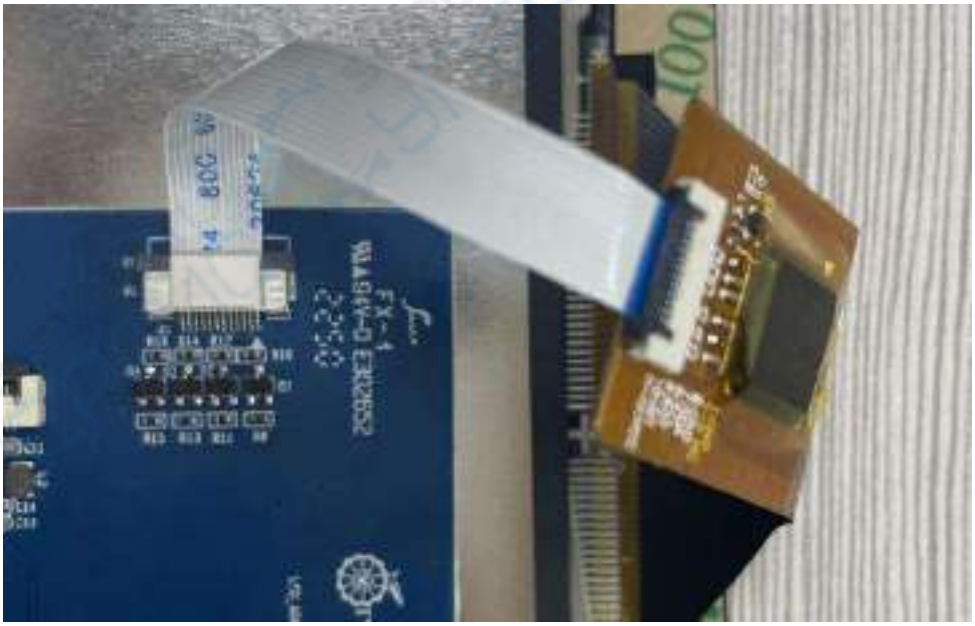
c. 30pin MIPI 排线

d. 12pin 触摸屏排线

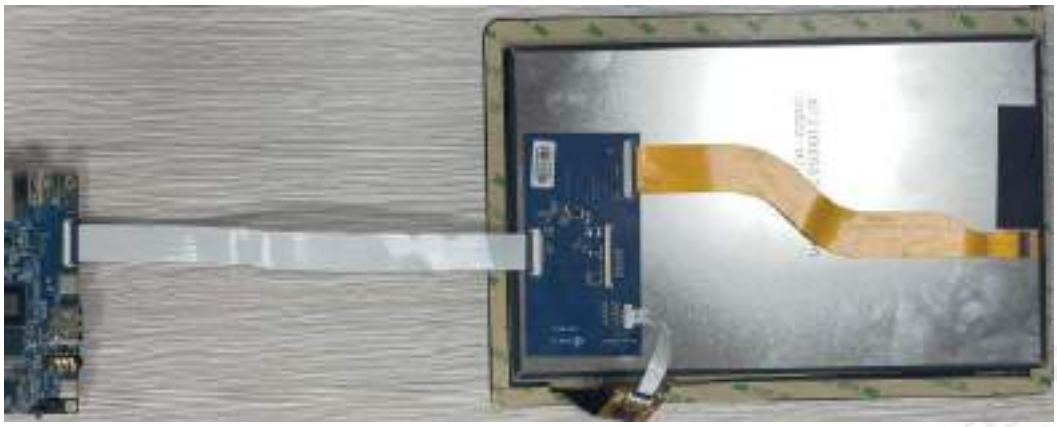
2) 按照下图将 12pin 触摸屏排线、31pin 转 40pin 排线、30pin MIPI 排线接到屏幕转接板上，注意**触摸屏排线蓝色的绝缘面朝下**，其它两根排线绝缘面朝上，如果接错会导致无显示或者不能触摸的问题

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 40pin 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



3.30.2. 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) 开发板上有两个 mipi lcd 屏幕的接口，我们定义：

a. lcd1 接口的位置为：

b. lcd2 接口的位置为：

v.1.0.0 和 v.1.0.2 版本的 linux 镜像 lcd dtbo 的配置和上面的定义是反的，使用时请注意下。

v.1.0.4 和 v.1.0.4 以后版本的 linux 镜像将 lcd dtbo 的配置改过来了，和开发板

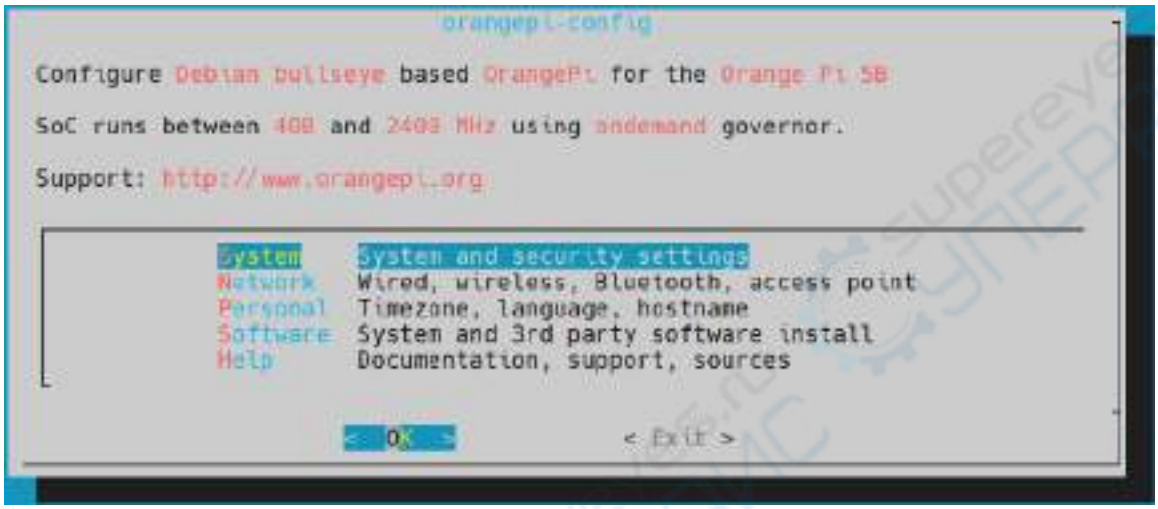
上丝印显示的 lcd 序号保持一致。

3) 打开 mipi lcd 配置的步骤如下所示：

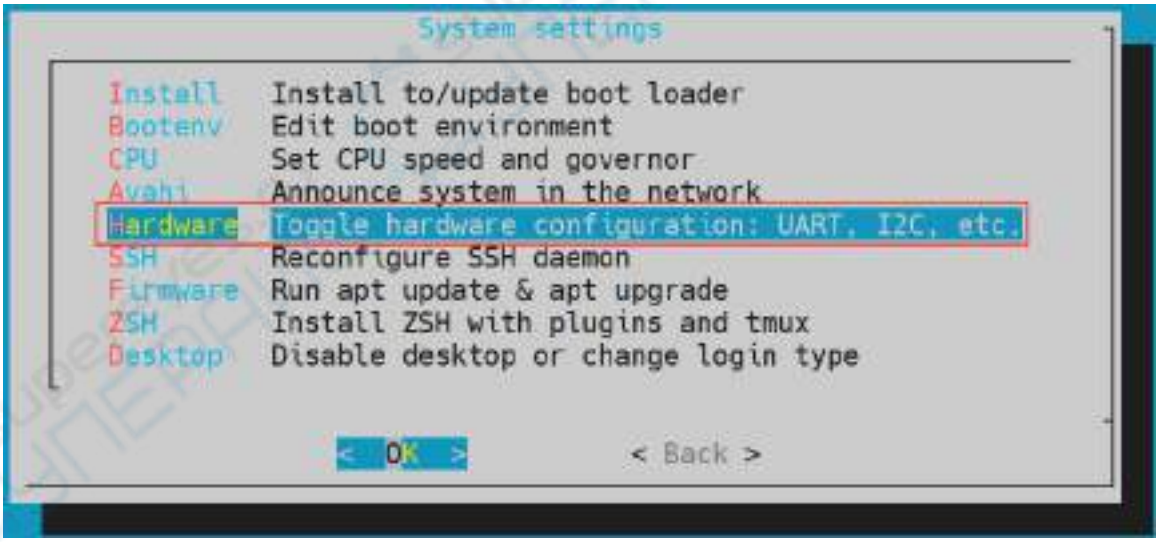
a. 首先运行下 **orangepi-config**，普通用户记得加 **sudo** 权限

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

b. 然后选择 **System**



c. 然后选择 **Hardware**

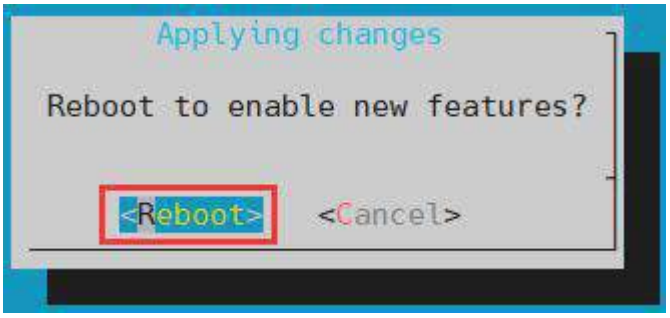


d. 然后使用键盘的方向键定位到 lcd1 或者 lcd2（想用哪个就打开哪个，有两个屏也可以同时打开），再使用**空格**选中

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=lcd1** 或者 **overlays=lcd2** 或者 **overlays=lcd1 lcd2** 这句配置。设置完后可以先检查下。如果不存在这句配置，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=lcd1** 或者 **overlays=lcd2** 或者 **overlays=lcd1 lcd2** 这句配置也是可以。

orangepi@orangepi:~$ **cat /boot/orangepiEnv.txt | grep "lcd"**
overlays=lcd1 **#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）：

3.30.3. 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** 这句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

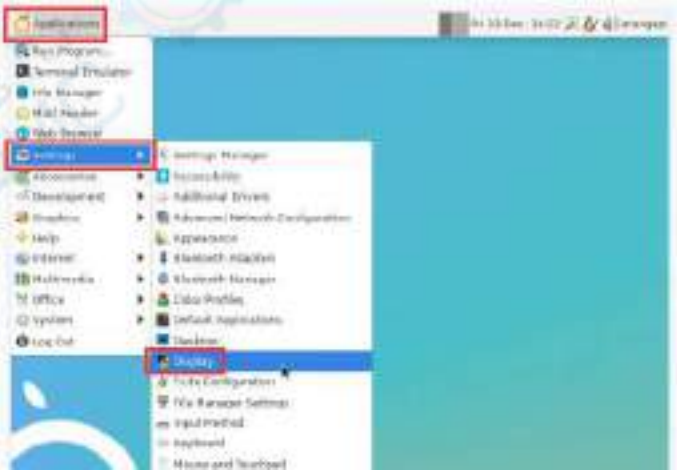
```
orangepi@orangepi:~$ sudo vim /boot/orangepiEnv.txt
overlays=lcd1
extraargs=cma=64M fbcon=rotate:3
```

注意，/boot/orangepiEnv.txt 中如果默认有 extraargs=cma=64M 这行配置，fbcon=rotate:3 这个配置添加到 extraargs=cma=64M 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

3.30.4. 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

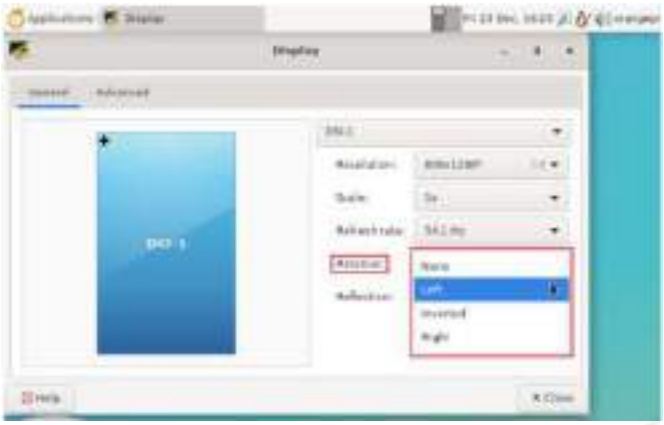


2) 然后在 **Rotation** 中选择想要旋转的方向

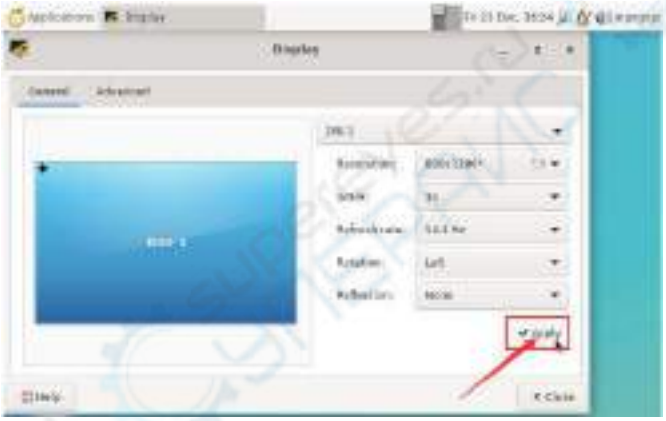
a. **None**: 不旋转
b. **Left**: 向左旋转 90 度

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
```

3.31. 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:~$ 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:~$ 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
```

3.32. 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。

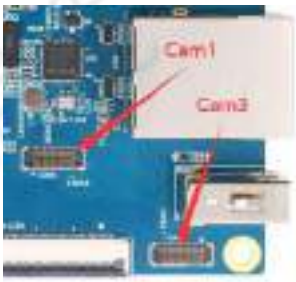
摄像头转接板上总共有 3 个摄像头的接口，同一时间只能接一个使用，如下图所示，其中：

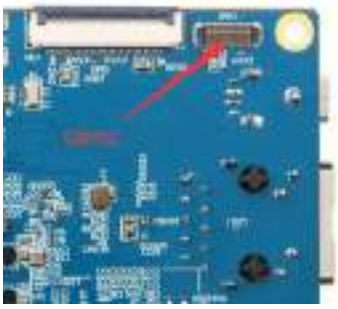
a. **1 号接口接 OV13850 摄像头**

b. **2 号接口接 OV13855 摄像头**

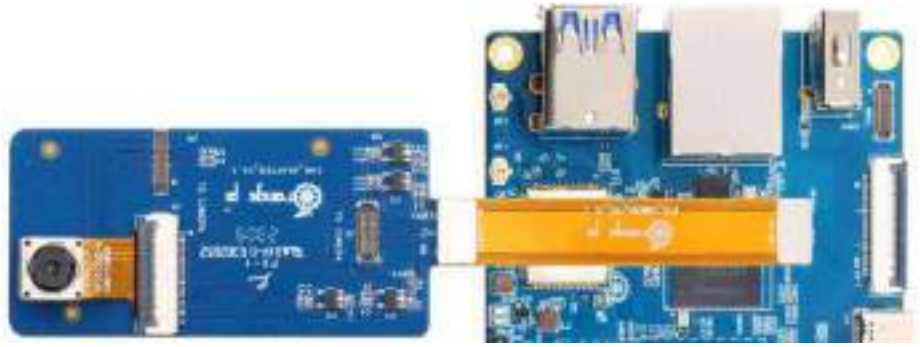
c. 3 号接口未使用，忽略即可

Orange Pi 5B 开发板上总共有 3 个摄像头接口，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：



摄像头插在开发板的 Cam2 接口的方法如下所示：

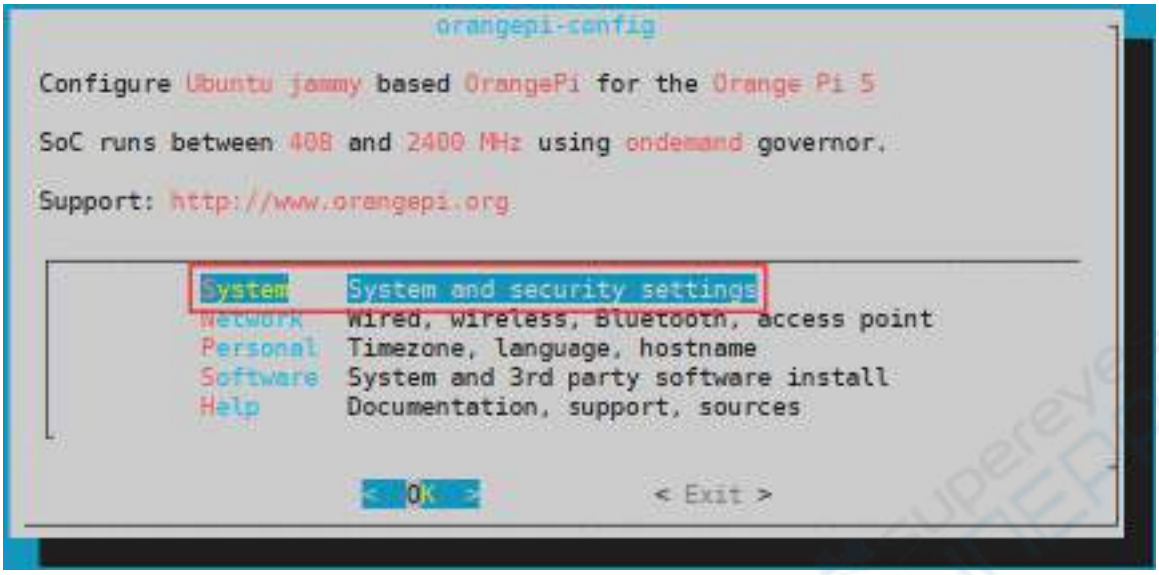
摄像头插在开发板的 Cam3 接口的方法如下所示：

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：

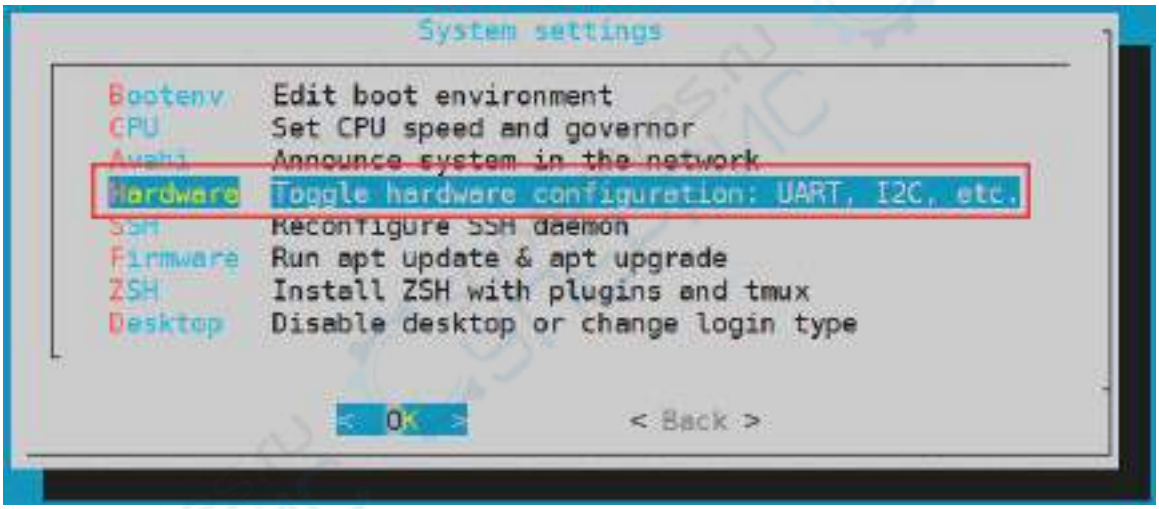
a. 首先运行下 **orangepi-config**，普通用户记得加 **sudo** 权限

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

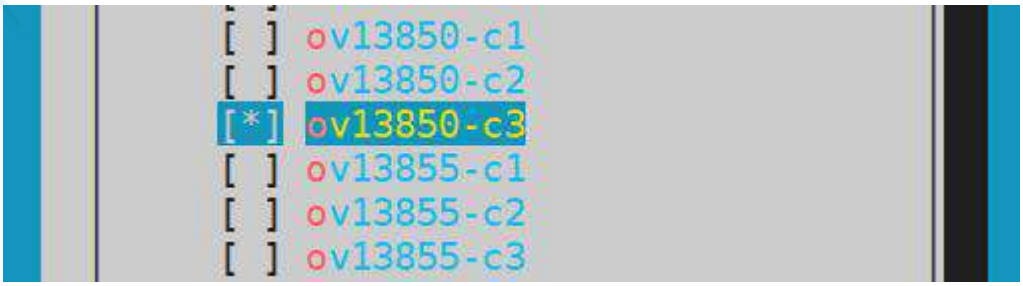
b. 然后选择 **System**



c. 然后选择 **Hardware**



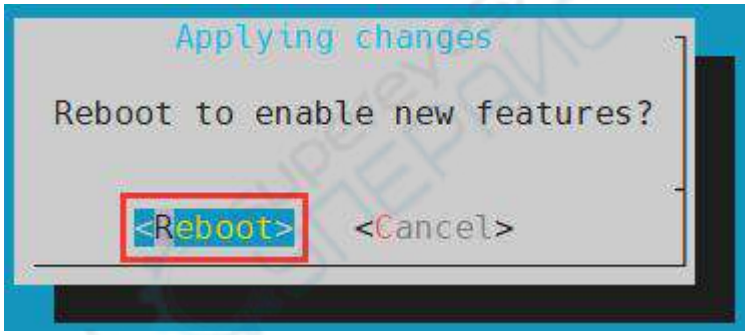
d. 然后使用使用键盘的方向键定位到下图所示的位置，再使用**空格**选中想要打开的摄像头，其中 **ov13850-c1** 表示在开发板的 Cam1 接口中使用 ov13850 摄像头，**ov13855-c2** 表示在开发板的 Cam2 接口中使用 ov13855 摄像头，其它的配置以此类推即可。



e. 然后选择**<Save>**保存

f. 然后选择**<Back>**

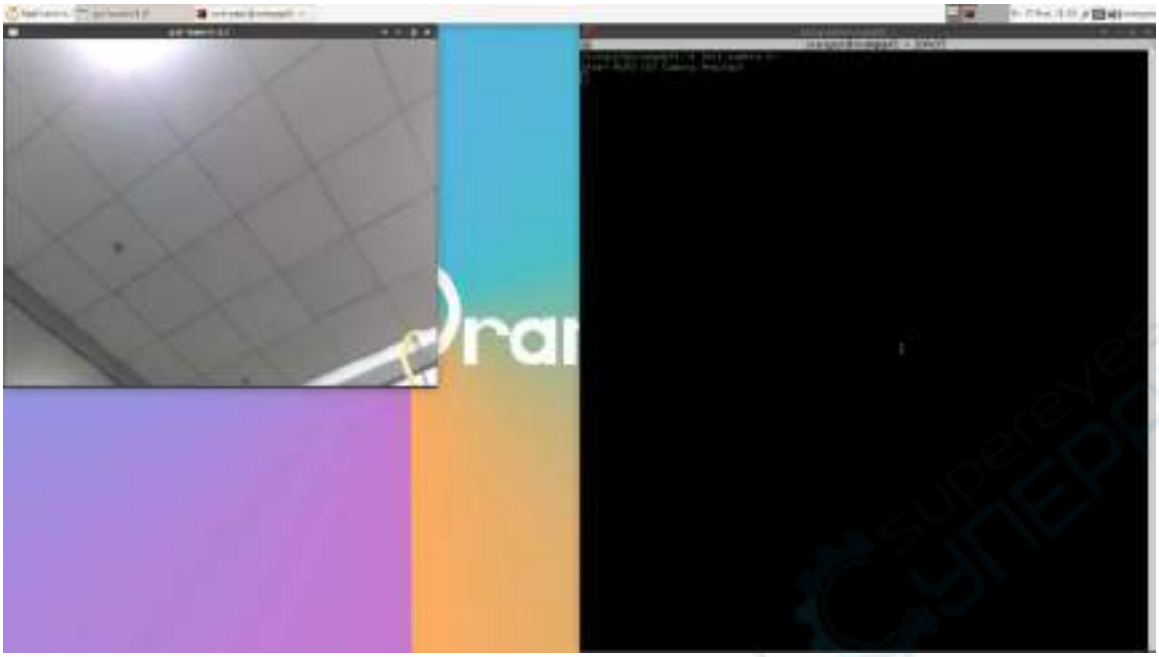
g. 然后选择**<Reboot>**重启系统使配置生效



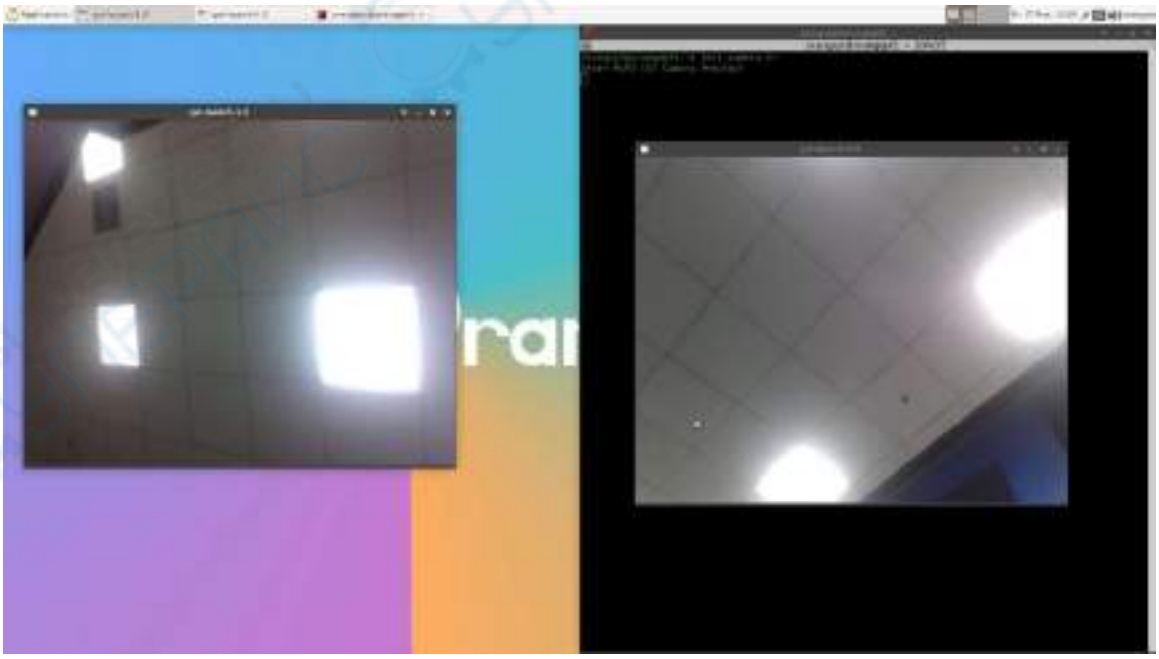
h. 然后在桌面系统中打开一个终端，再运行下面的脚本

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

i. 然后就能看到摄像头的预览画面了



除了单摄外， 我们还可以同时使用两个摄像头。需要注意的是，目前测试双摄像头请使用 **Cam1+Cam3** 的组合（支持 ov13850 和 ov13855 混搭）。接好双摄后，然后和前面步骤一样，通过 **orangepi-config** 打开 Cam1+Cam3 的配置，再重启系统，然后在桌面中打开终端运行 **test_camera.sh** 脚本即可看到两个摄像头的预览画面，如下图所示：



摄像头 dts 配置请参考下面的链接，有需求的可以自行修改；

https://github.com/orangepi-xunlong/linux-orangepi/blob/orange-pi-5.10-rk3588/arch/arm64/boot/dts/rockchip/rk3588s-orangepi-5-camera1.dtsi

https://github.com/orangepi-xunlong/linux-orangepi/blob/orange-pi-5.10-rk3588/arch/arm64/boot/dts/rockchip/rk3588s-orangepi-5-camera2.dtsi

https://github.com/orangepi-xunlong/linux-orangepi/blob/orange-pi-5.10-rk3588/arch/arm64/boot/dts/rockchip/rk3588s-orangepi-5-camera3.dtsi

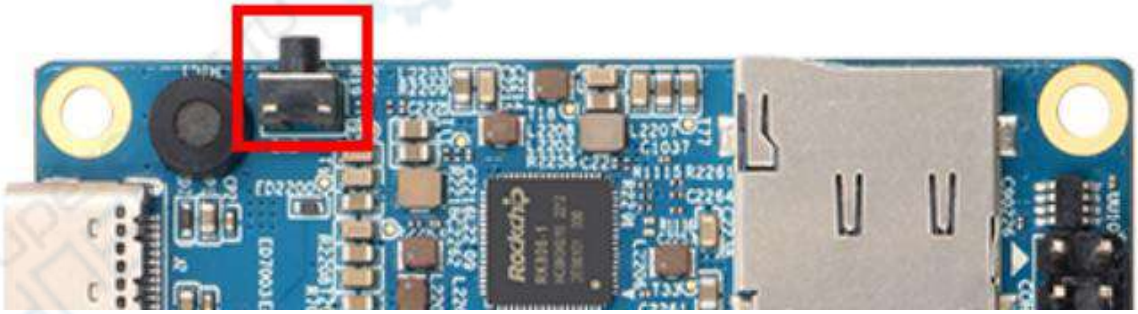
dt overlay 的配置在下面的目录中：
https://github.com/orangepi-xunlong/linux-orangepi/tree/orange-pi-5.10-rk3588/arch/arm64/boot/dts/rockchip/overlay

3.33. 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。

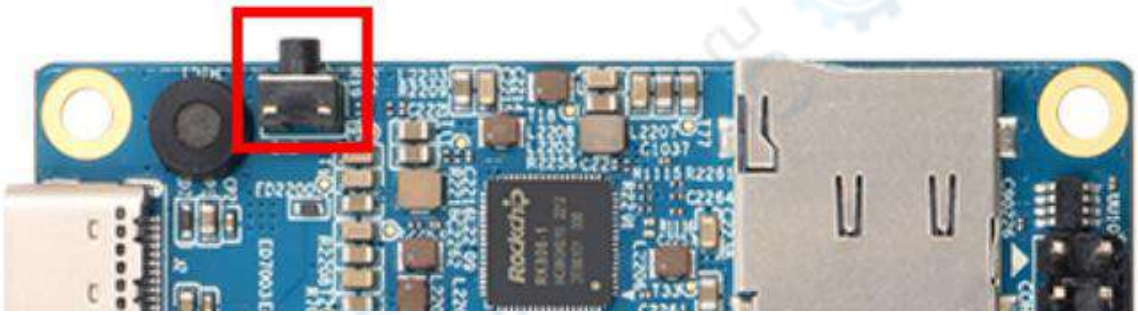
```
orangepi@orangepi:~$ sudo poweroff
```

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

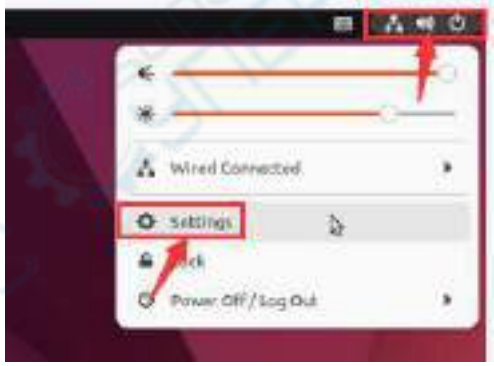
功能	Ubuntu22.04 Gnome Wayland
USB2.0x2	OK
USB3.0x1	OK
USB Type-C 3.0	OK
DP 显示	OK
eMMC	OK
AP6275P-WIFI	OK
AP6275P-蓝牙	OK
GPIO（26pin）	OK
UART（26pin）	OK
SPI（26pin）	OK
I2C（26pin）	OK
CAN（26pin）	OK
PWM（26pin）	OK
3pin 调试串口	OK
TF 卡启动	OK
HDMI 视频	OK
HDMI 音频	OK
OV13850 摄像头	OK
OV13855 摄像头	OK
LCD1	OK
LCD2	OK
千兆网口	OK
网口状态灯	OK

MIC	**OK**
耳机播放	**OK**
耳机录音	**OK**
LED 灯	**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**
Kodi 硬解视频	**OK**
MPV 硬解视频	**OK**

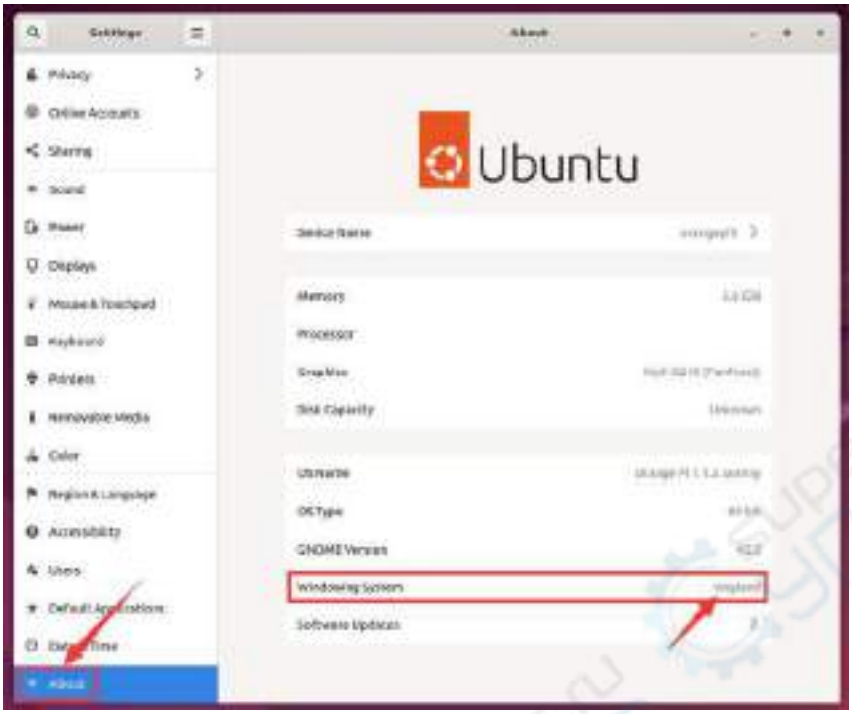
4.2. 确认系统当前使用的窗口系统为 wayland 的方法

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



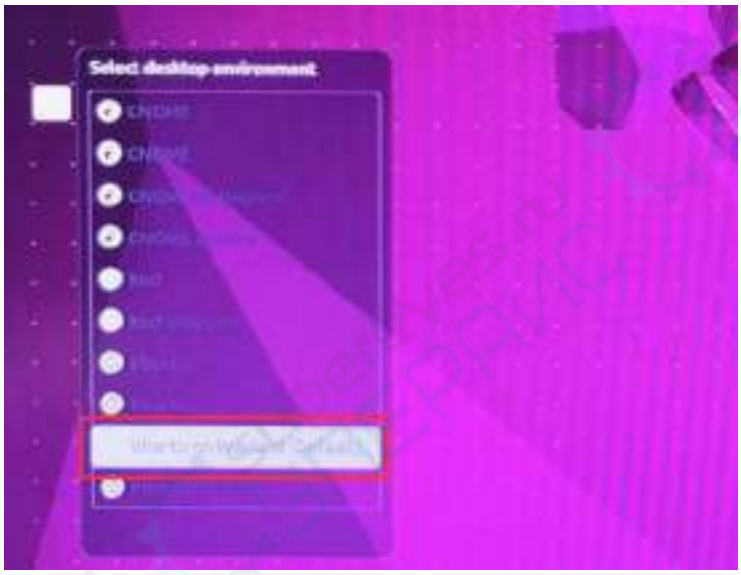
b. 然后选择 **About**，如果 **Windowing System** 一栏显示的 **wayland** 说明设置正确



2) 当 **Log Out** 出系统后会进入下面的登录界面

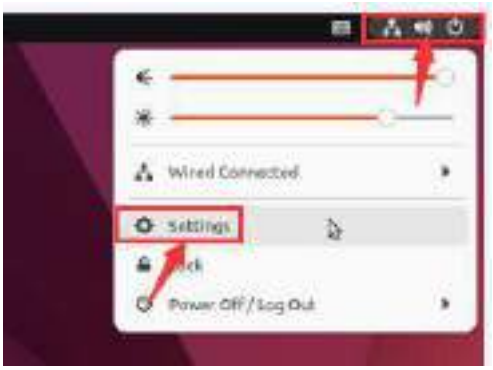
3) 再次登录系统前请先点击下图所示的位置

4) 然后选择 **Ubuntu on Wayland**，再输入密码登录系统

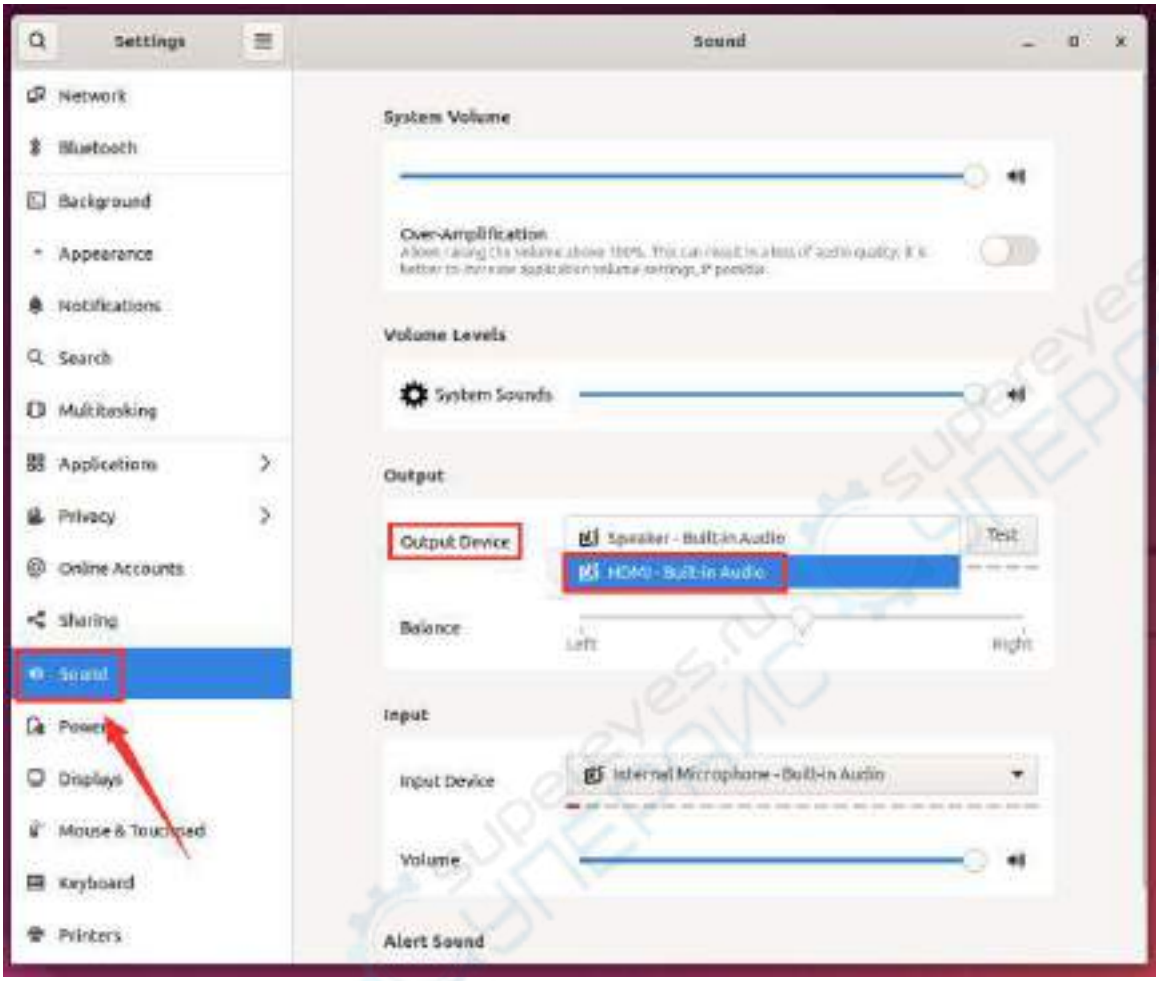


4.3. 切换默认音频设备的方法

1) 首先打开设置



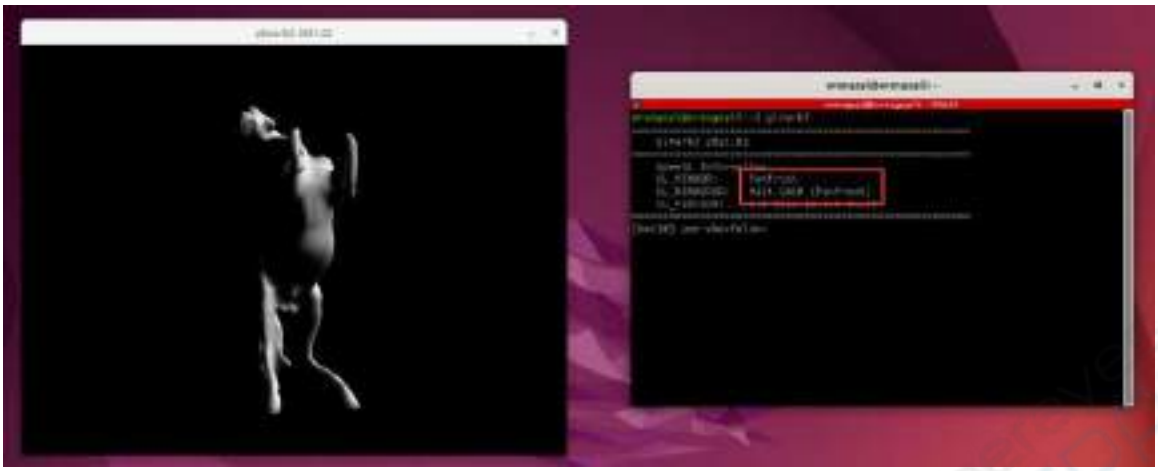
2) 然后选择 **Sound**，然后在 **Output Device** 中选择想要使用的音频设备即可



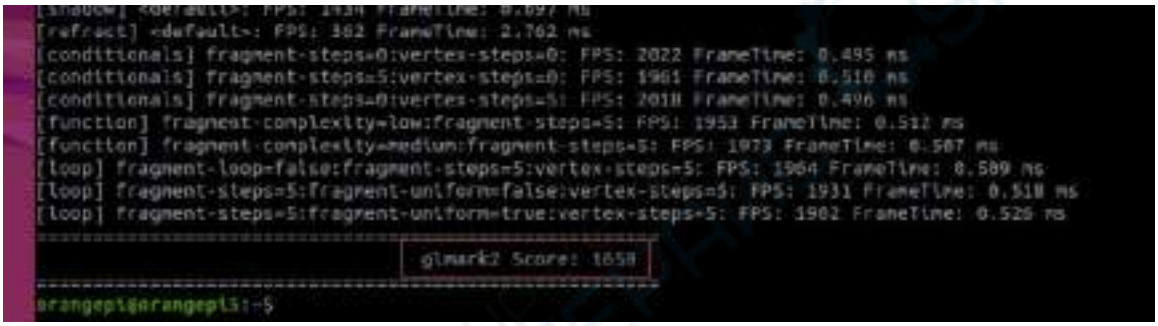
4. 4. GPU 的测试方法

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
```

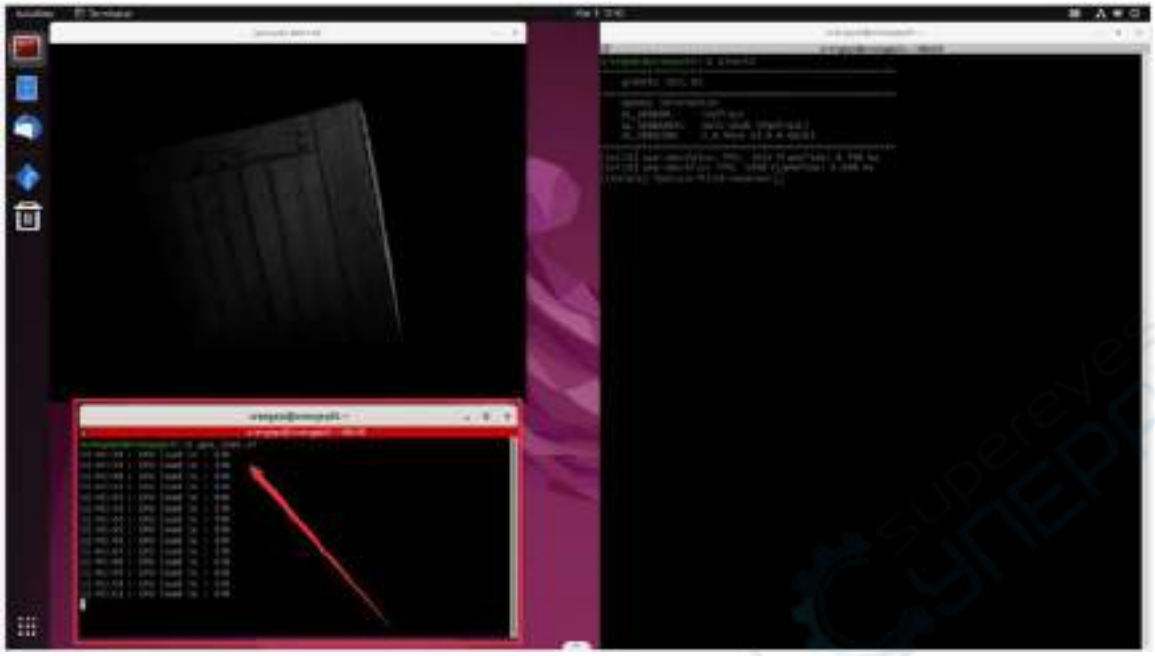


2) glmark2 跑分测试一般为 1000 多分



3) 运行 **gpu_load.sh** 脚本可以查看 GPU 当前的负载情况

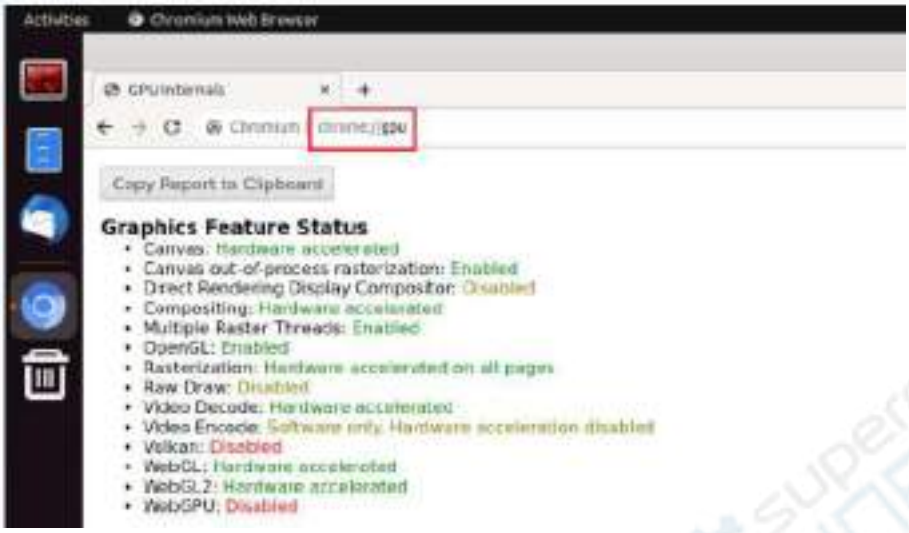
```
orangepi@orangepi:~$ gpu_load.sh
```



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

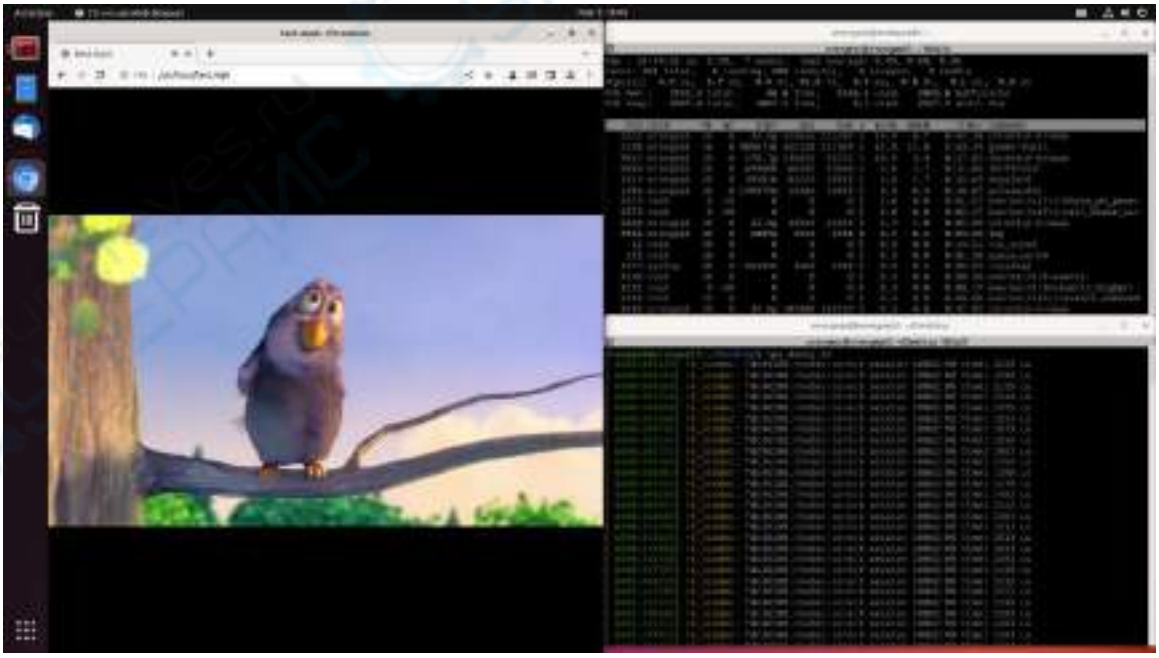


3) 然后可以打开视频网站播放一个视频文件，或者在浏览器中输入下面的路径名播放系统自带的一个测试视频文件

```
/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

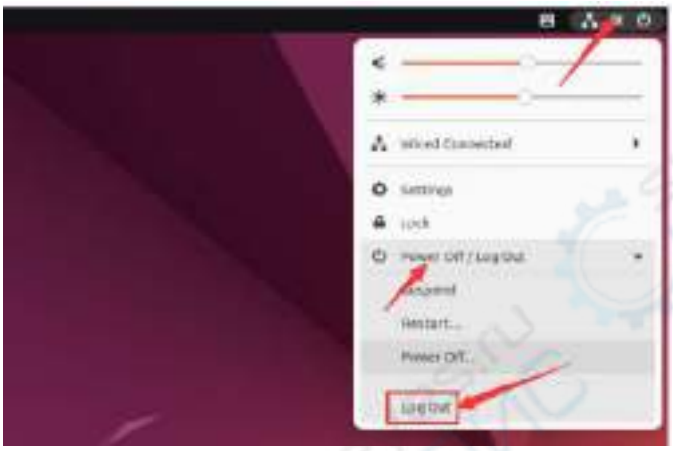
```
orangepi@orangepi:~$ vpu_debug.sh
```

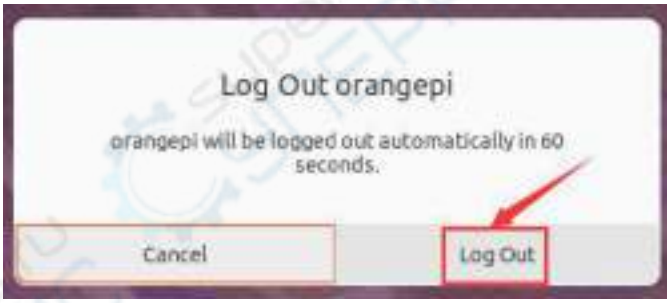


4.6. Kodi 硬解播放视频的测试方法

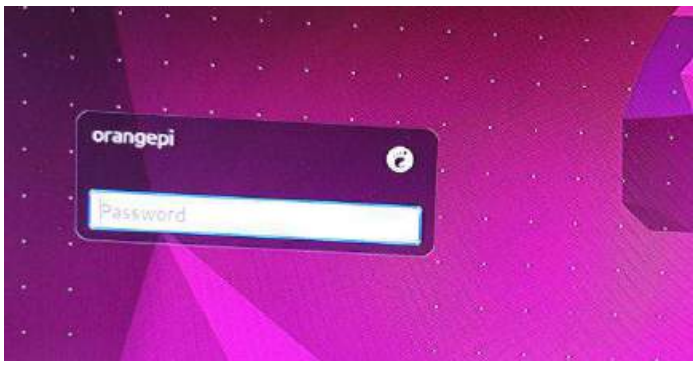
注意，在 Wayland 桌面中直接打开 Kodi 显示会有有问题，请严格按照下面的方法打开 Kodi。

1) 首先登出系统



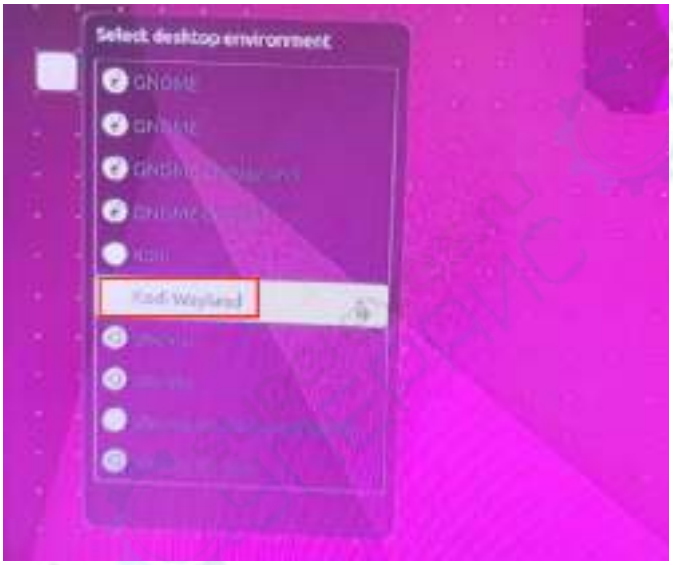


2) 当登出系统后会进入下面的登录界面

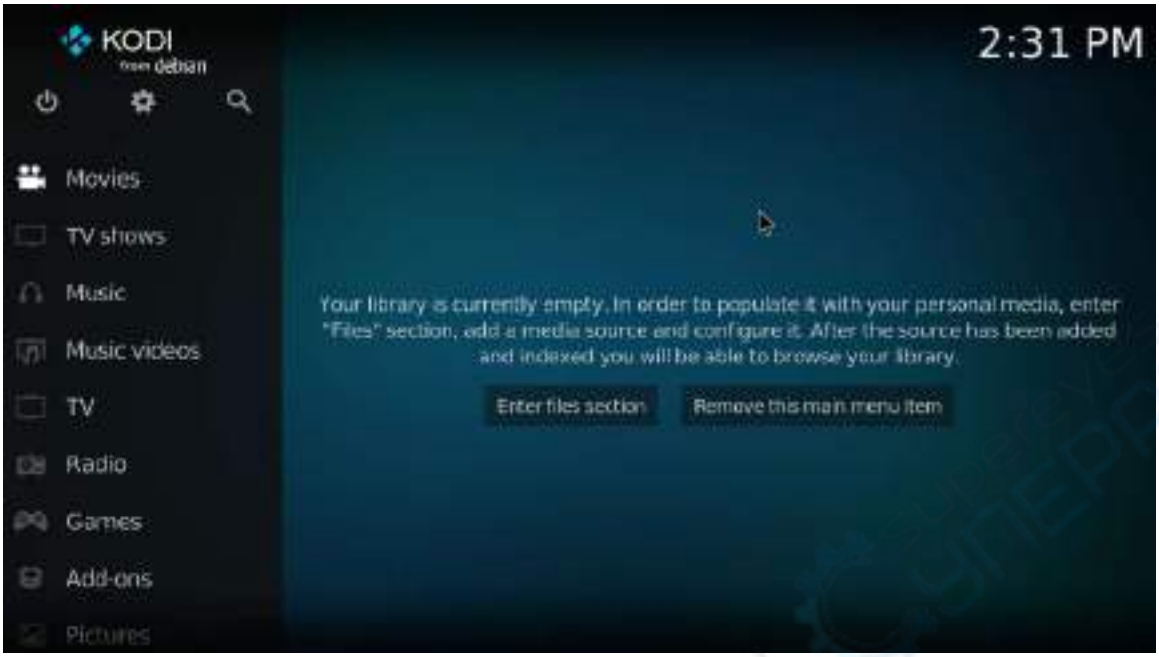


3) 然后点击下图所示的位置

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

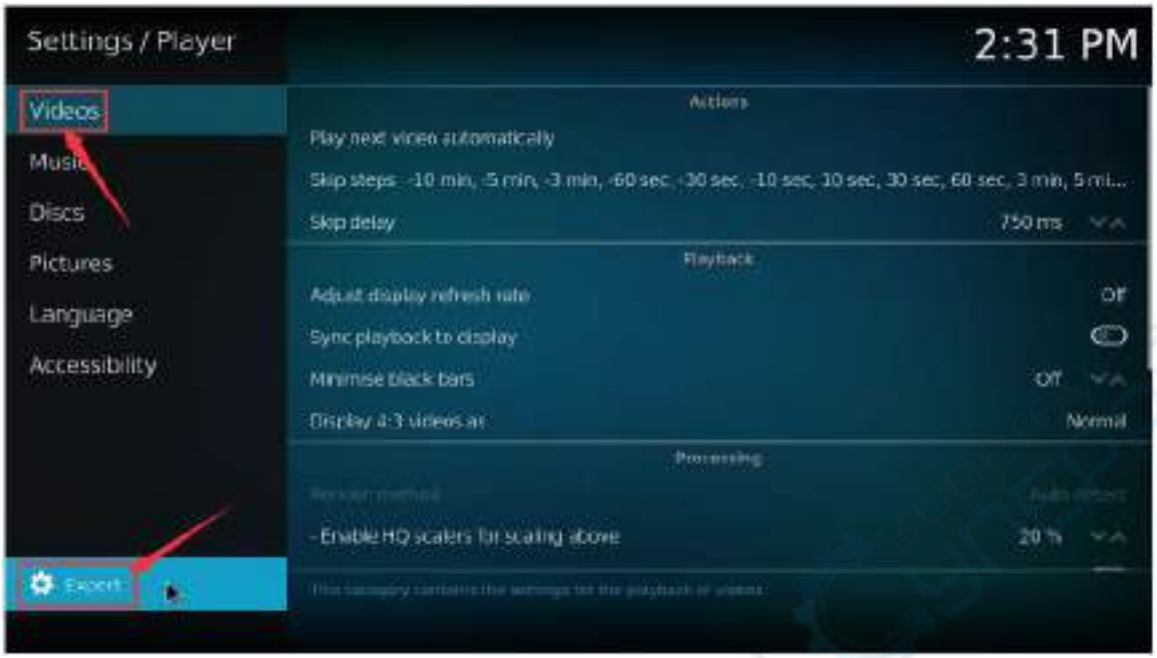


6) 然后点击设置

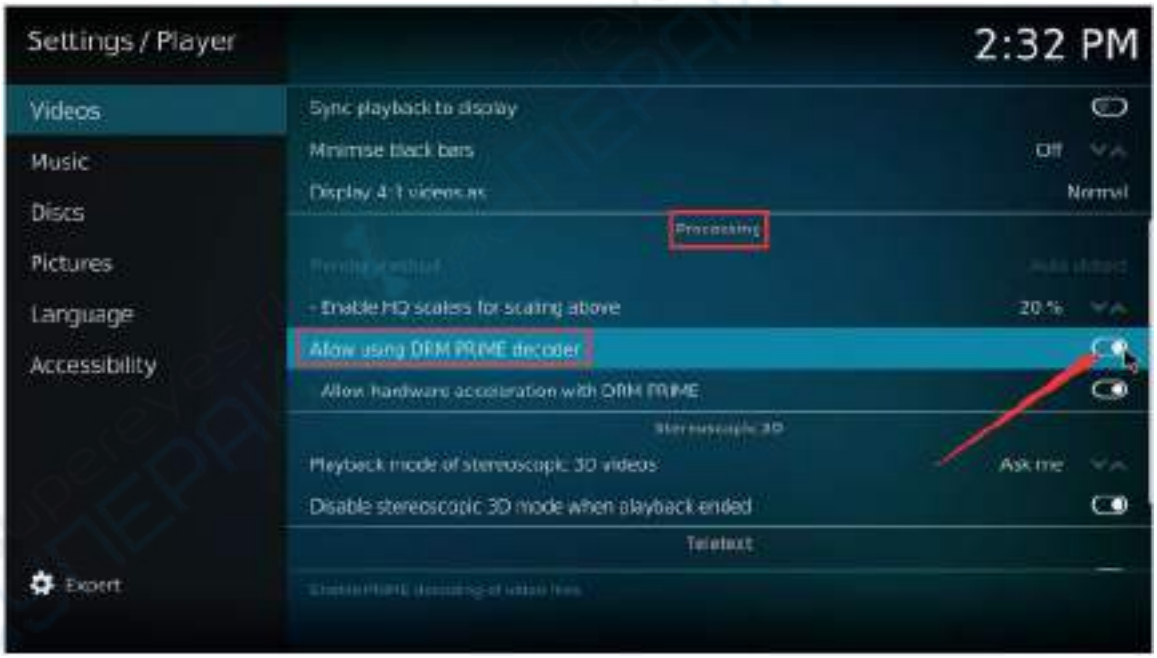
7) 然后选择 **Player**

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

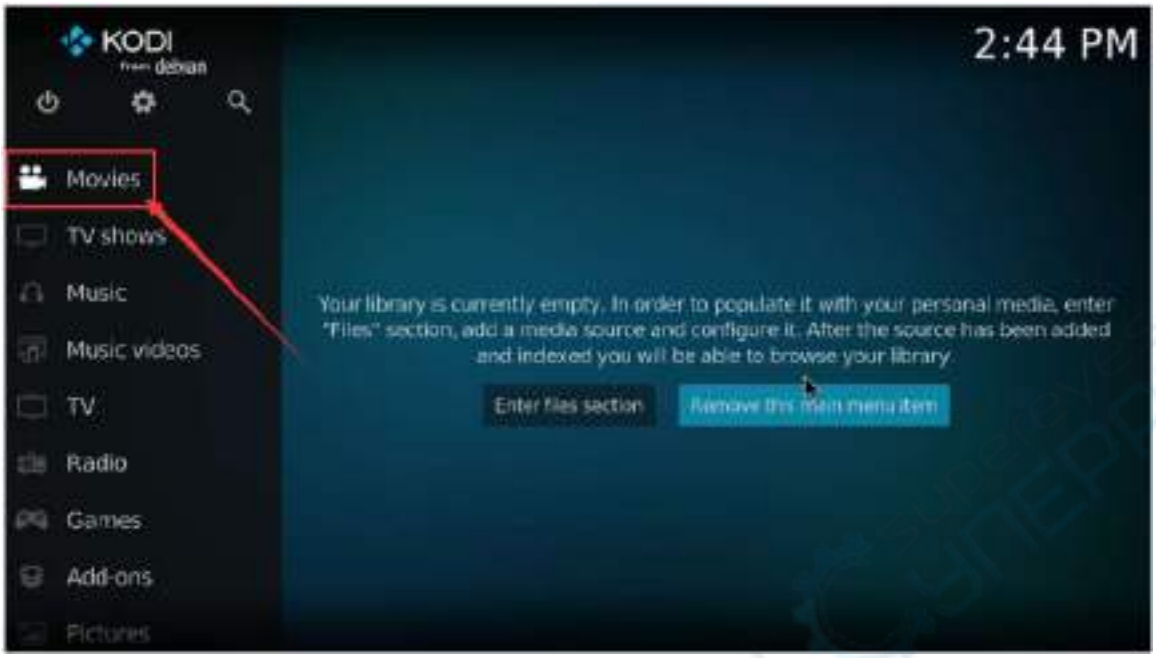


10) 然后在 **Processing** 设置中打开 **Allow using DRM PRIME decoder**

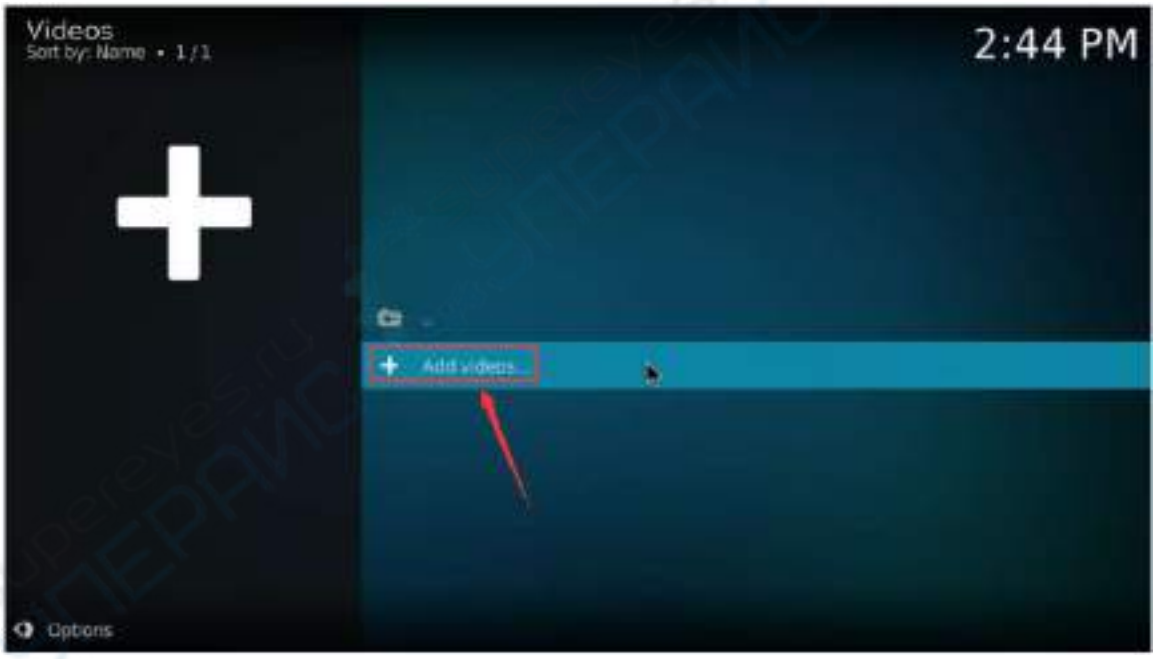


11) 然后我们来导入一个系统自带的测试视频测试下，你也可以上传想要播放的视频到系统中，然后导入播放

a. 首先进入主界面，然后选择 **Movies**

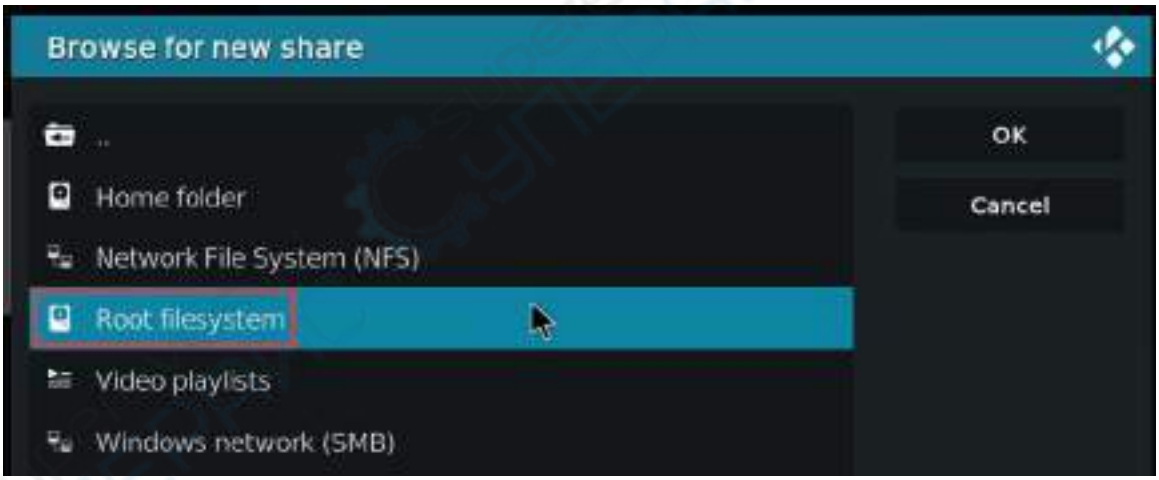


b. 然后选择 **Add videos...**



c. 然后选择 **Browse**

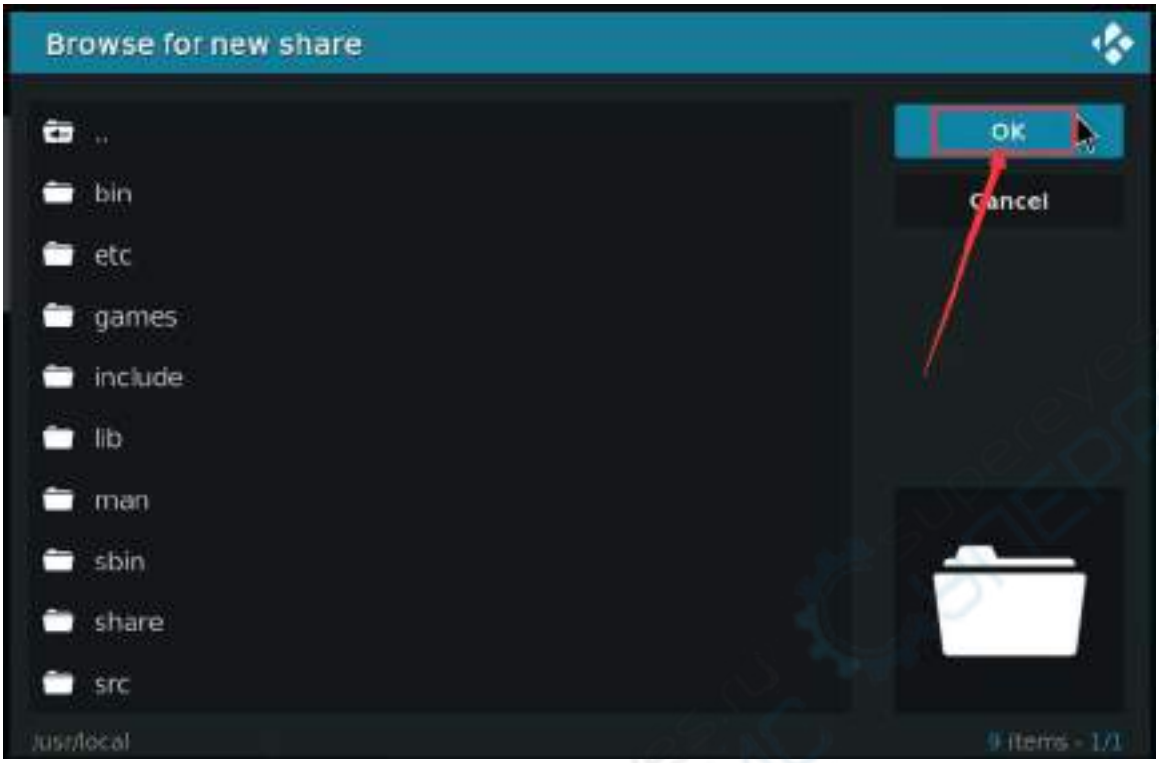
d. 然后选择 **Root filesystem**



e. 然后选择 **usr**

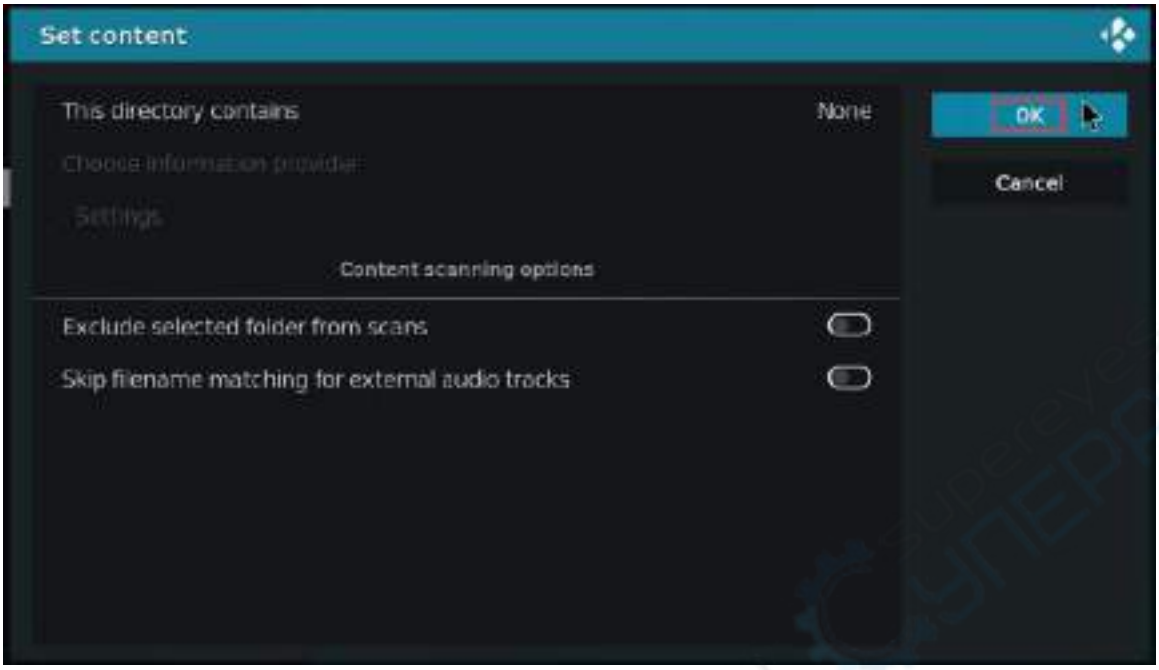
f. 然后选择 **local**

g. 然后选择 **OK**

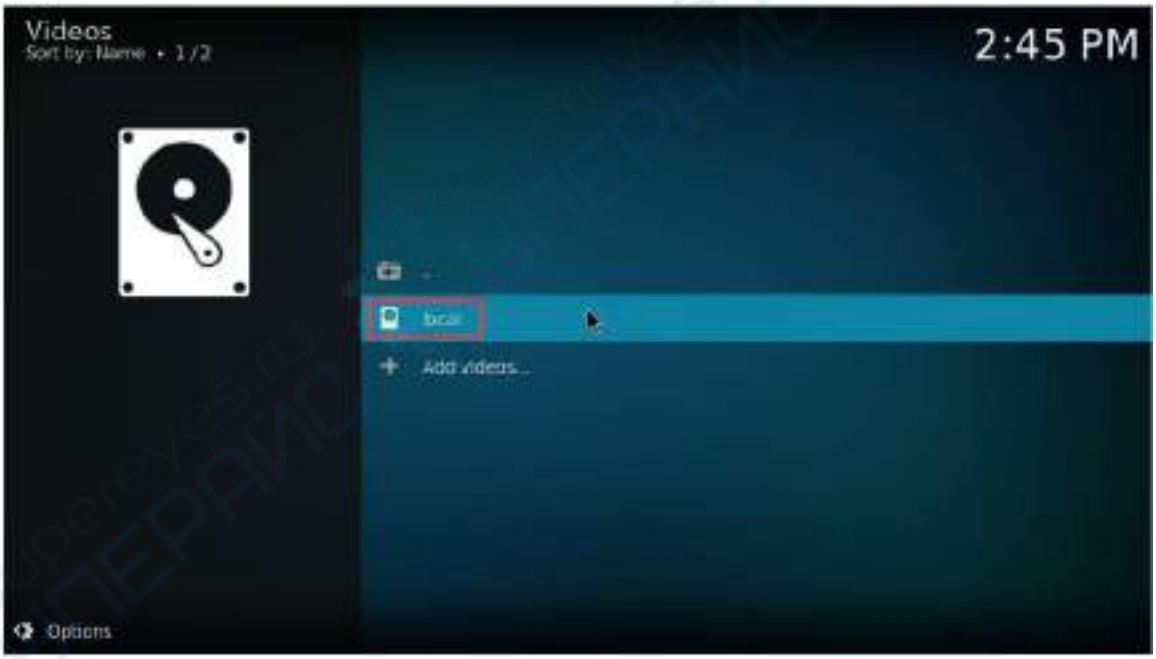


h. 然后选择 **OK**

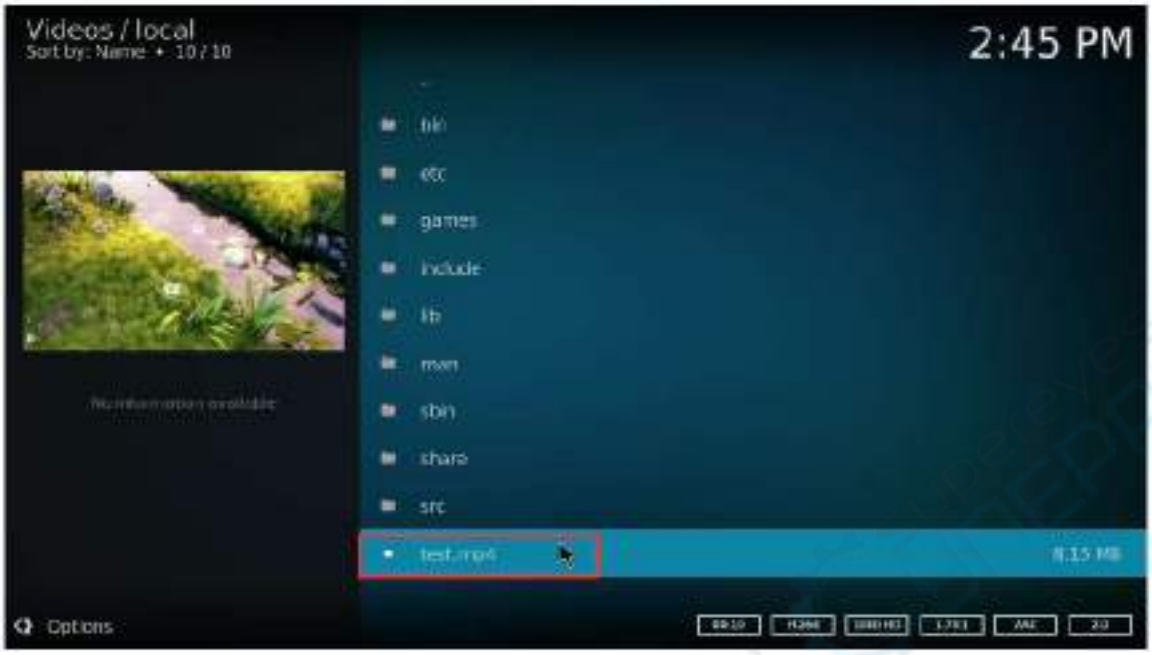
i. 然后选择 OK



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

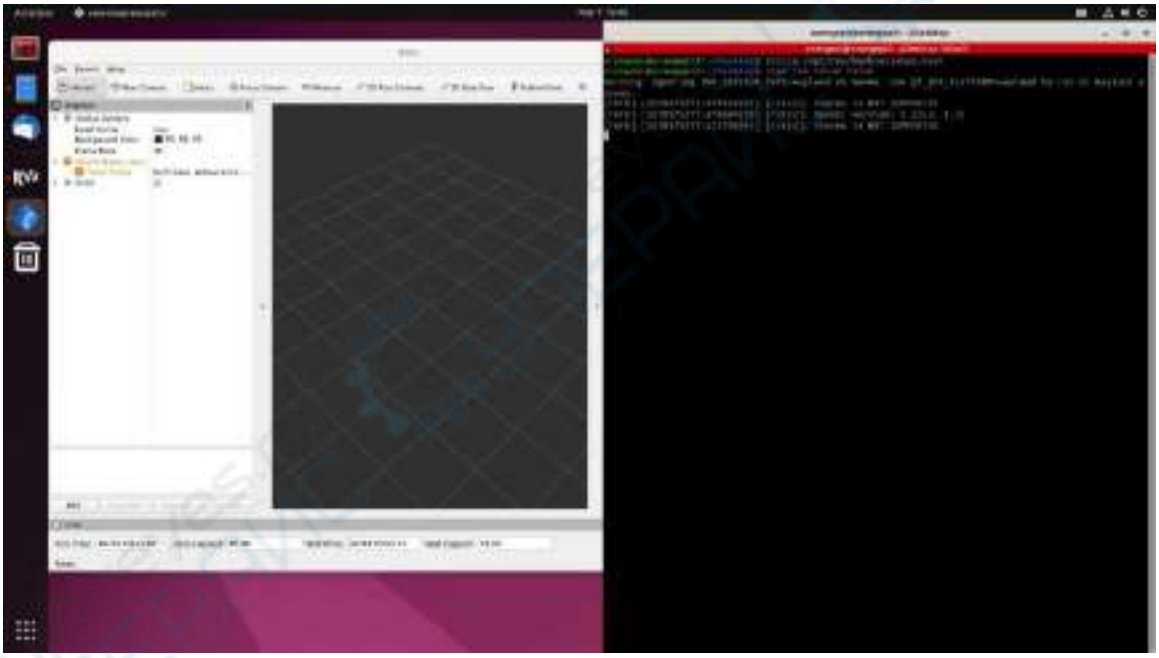
```
usage: ros2 [-h] Call `ros2 <command> -h` for more detailed usage. ...

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@orangepi5b:~$ 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
```

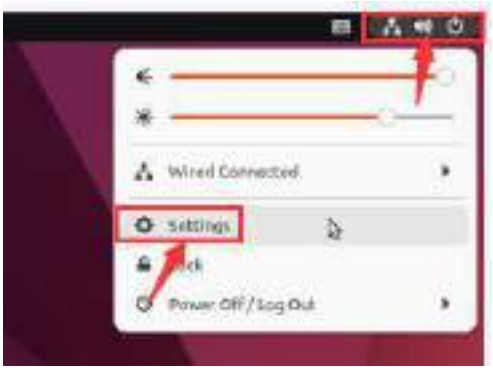


5) 参考文档

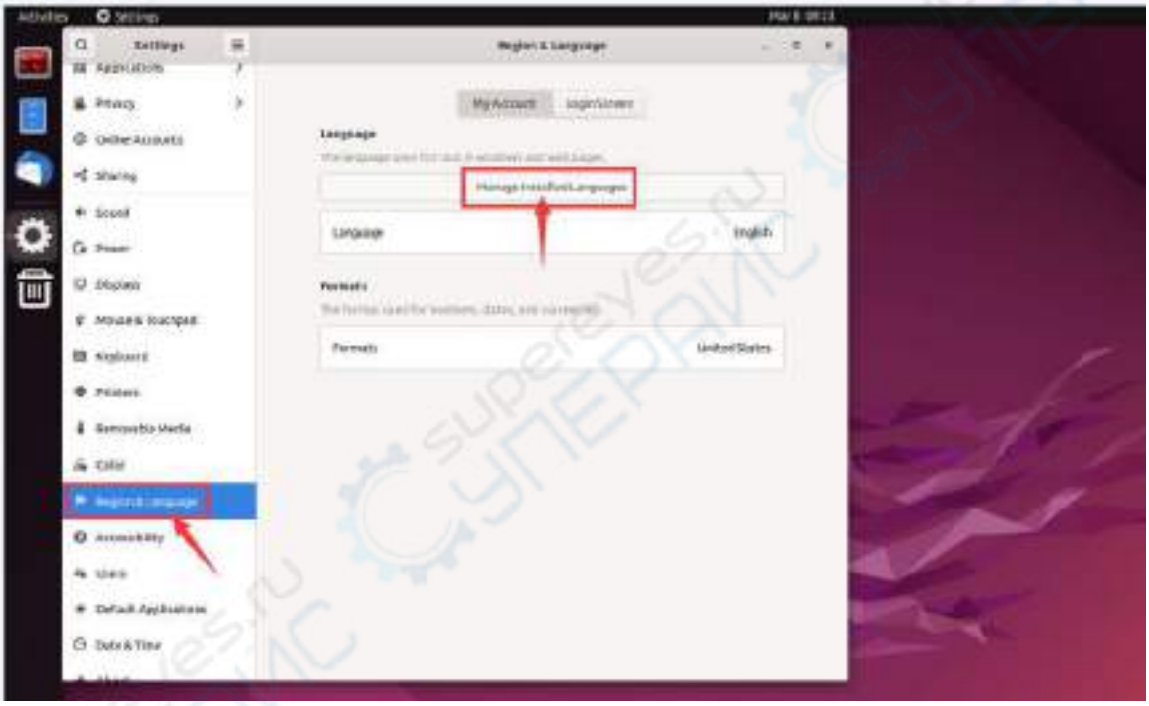
http://docs.ros.org/en/humble/index.html
http://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debians.html

4.8. 设置中文环境以及安装中文输入法的方法

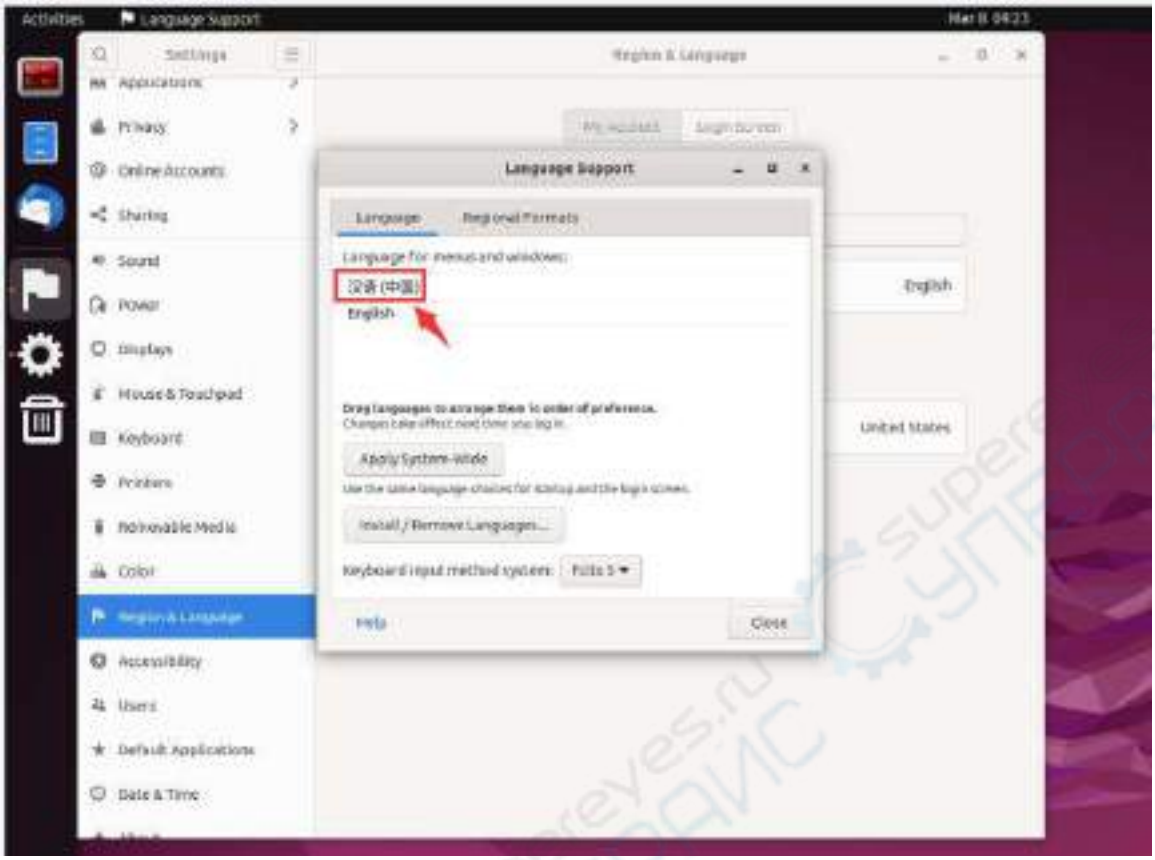
1) 首先打开设置



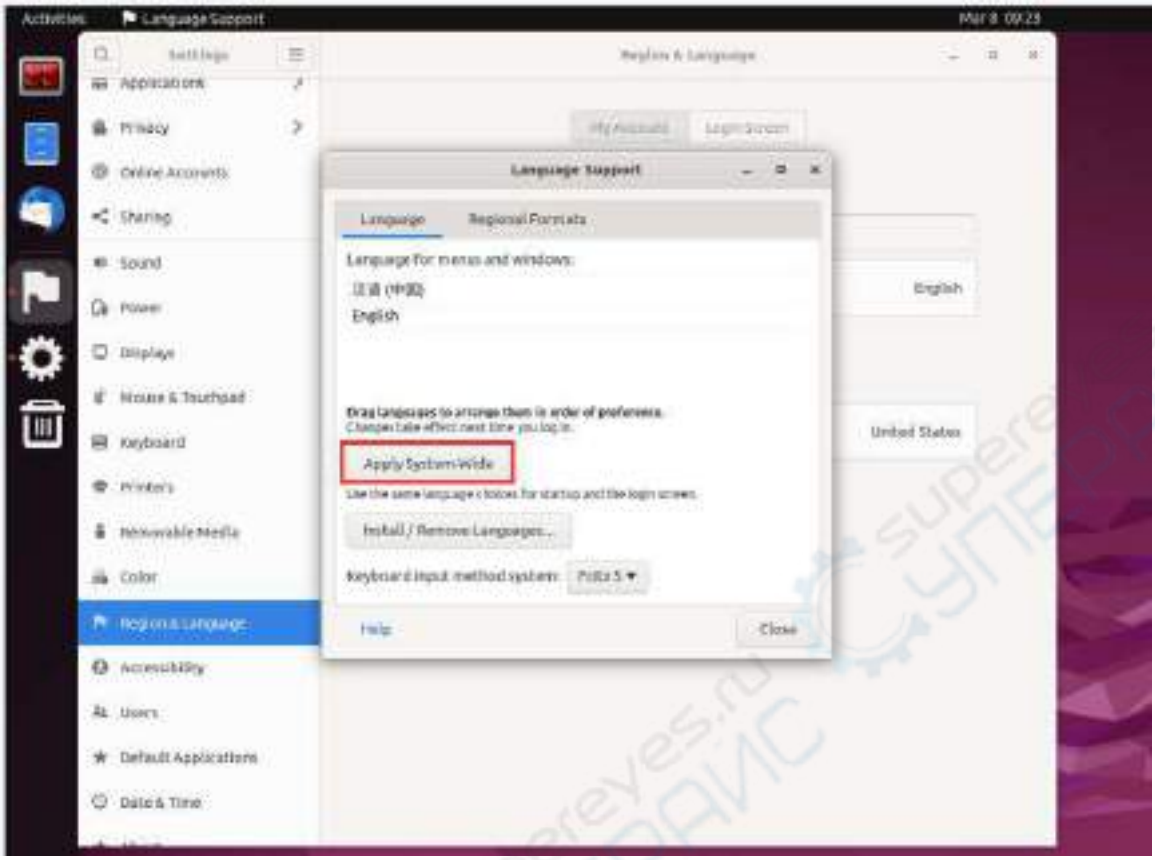
2) 然后找到 **Region & Language** 选项，然后点击 **Manage Installed Languages** 选项



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



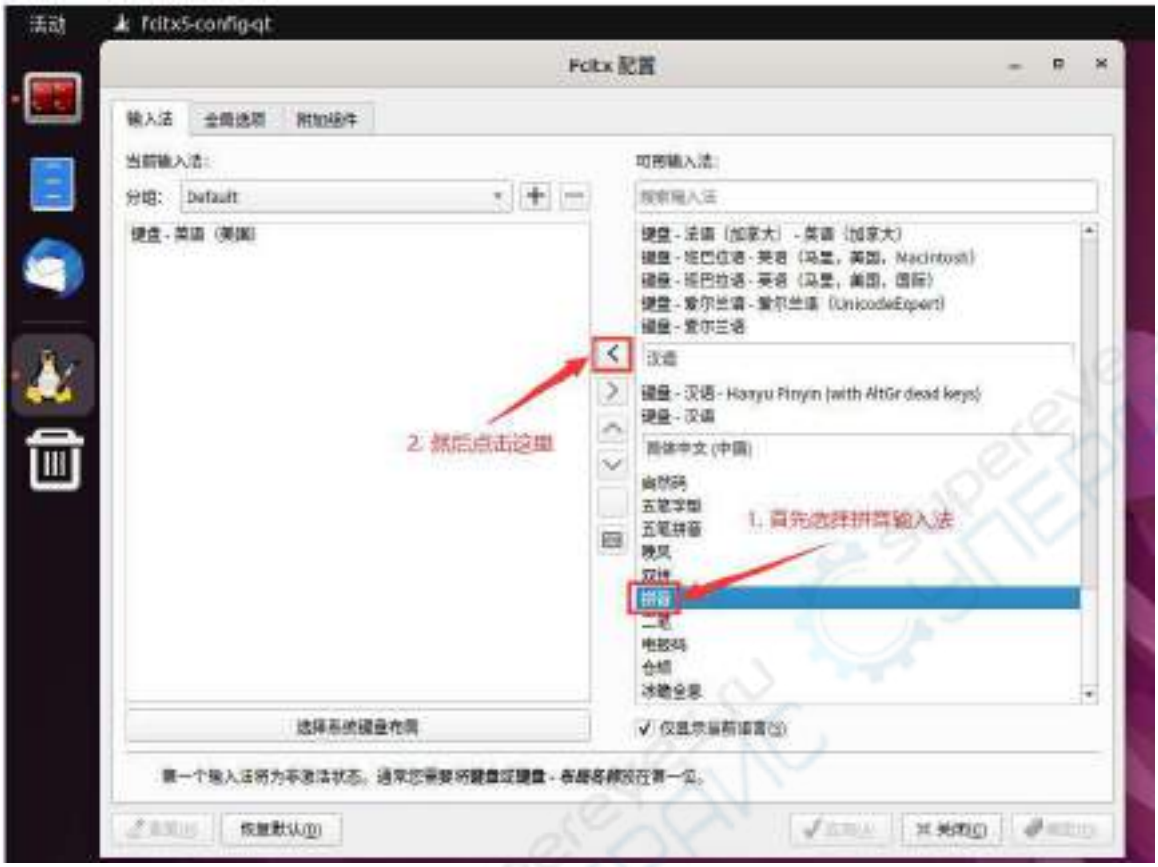
5) **然后重启 Linux 系统使配置生效**

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

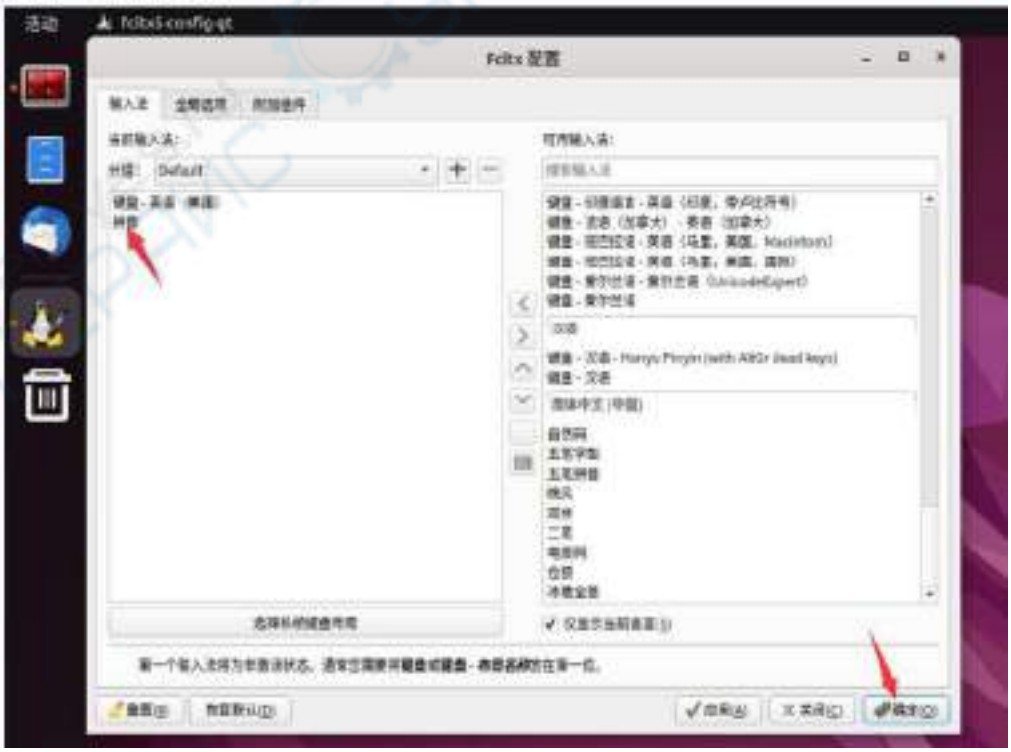
7) 然后可以看到桌面都显示为中文了

8) 然后打开 Fcitx5 配置程序

9) 然后选择使用拼音输入法

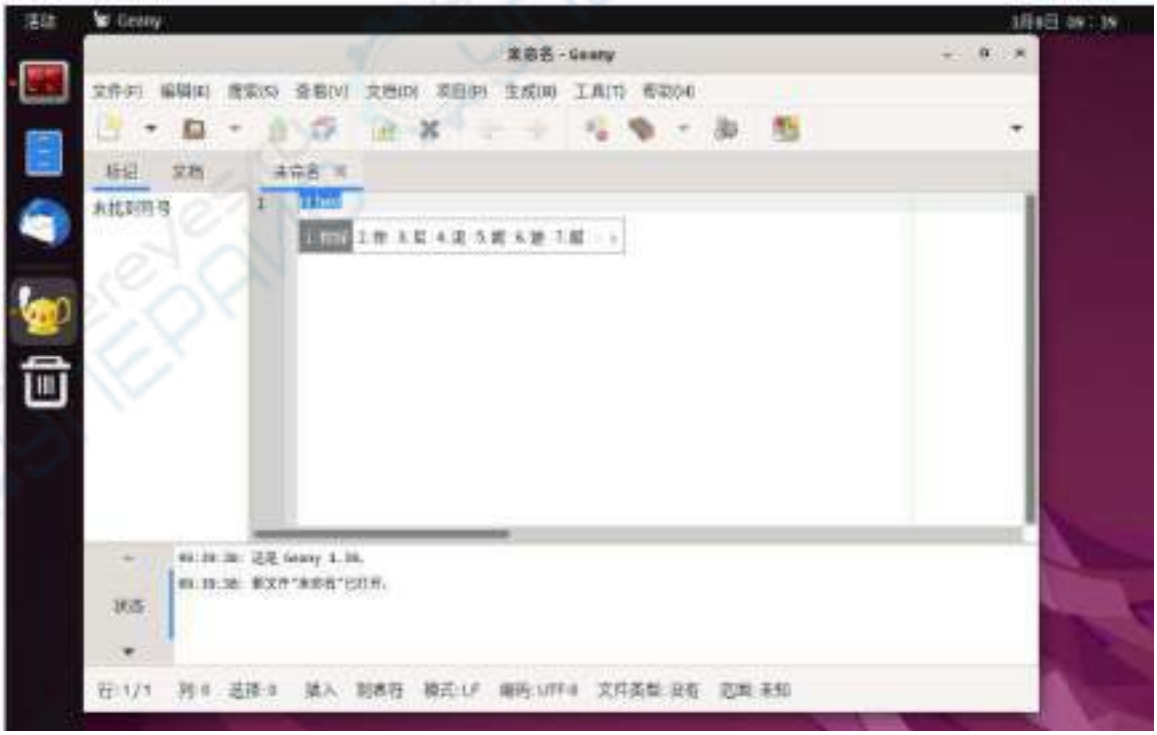


10) 选择后的界面如下所示，再点击确定即可



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



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
USB2.0x2	OK
USB3.0x1	OK
USB Type-C 3.0	OK
eMMC 启动系统	OK
DP 显示	OK
AP6275P-WIFI	OK
AP6275P-蓝牙	OK
GPIO（26pin）	OK
UART（26pin）	OK
SPI（26pin）	OK
I2C（26pin）	OK
CAN（26pin）	OK
PWM（26pin）	OK
3pin 调试串口	OK
TF 卡启动	OK
HDMI 视频	OK
HDMI 音频	OK
OV13850 摄像头	OK
OV13855 摄像头	OK
LCD1	OK
LCD2	OK
千兆网口	OK
网口状态灯	OK
MIC	OK
耳机播放	OK
耳机录音	OK
LED 灯	OK
GPU	OK

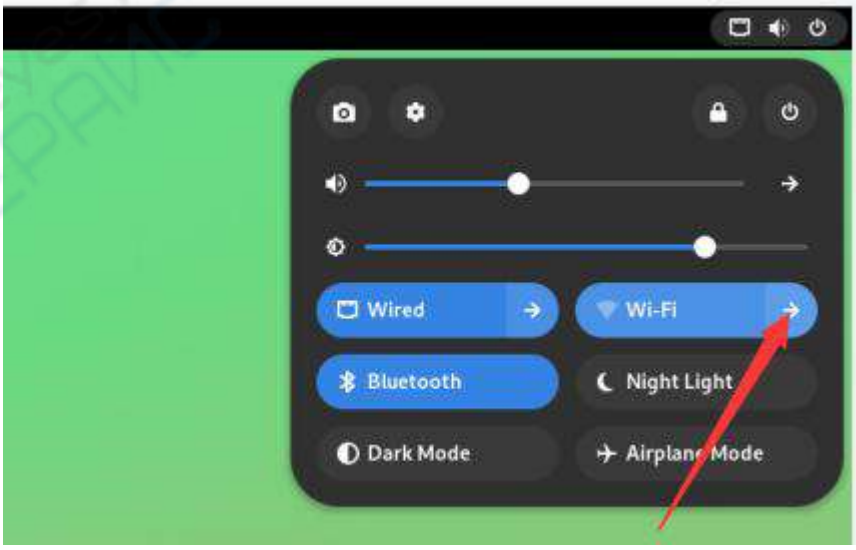
NPU	NO
VPU	OK
开关机按键	OK
看门狗测试	OK
Chromium 硬解视频	NO
MPV 硬解播放视频	OK

5. 2. AP6275P PCIe WIFI6+蓝牙模块的使用方法

1) 连接 WIFI 的步骤如下所示：

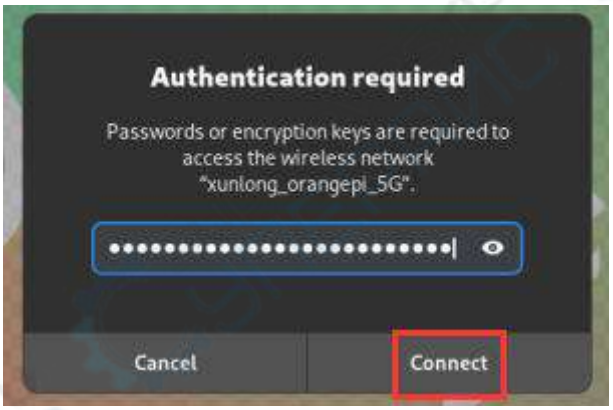
a. 首先点击桌面右上角的这块区域

b. 然后选择 Wi-Fi

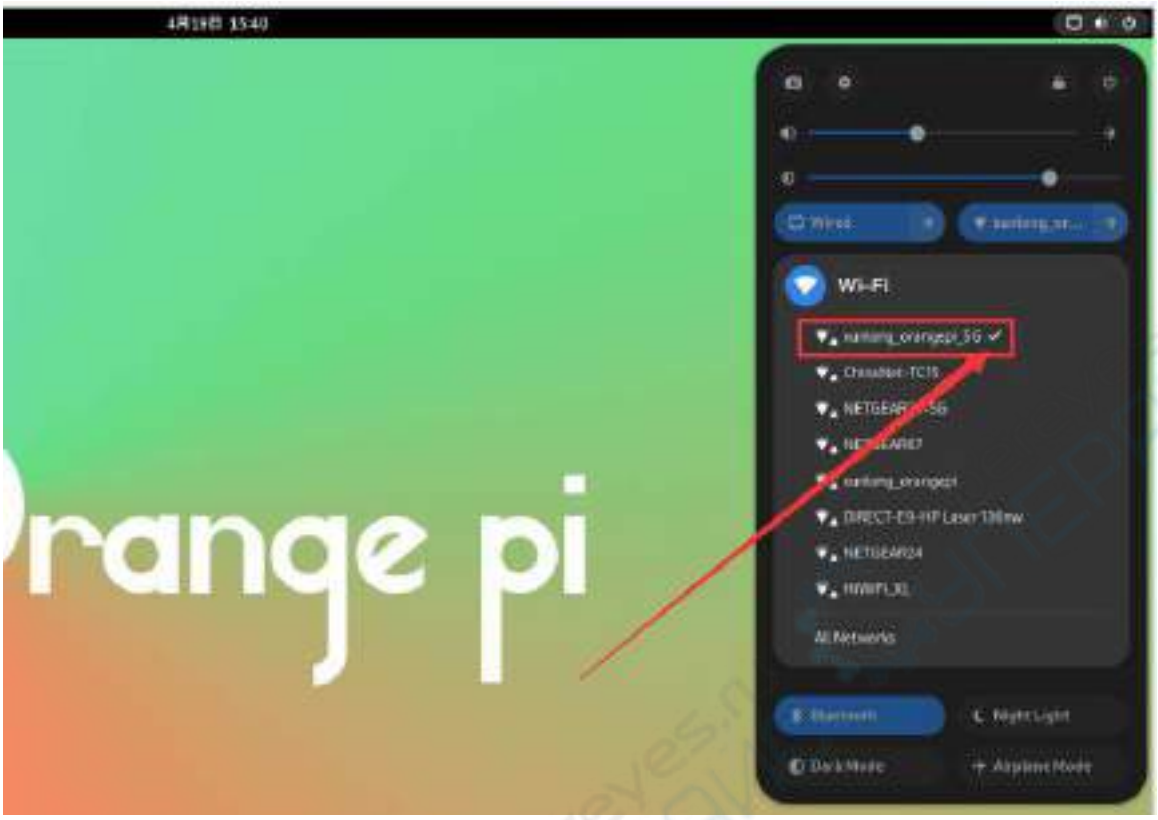


c. 然后选择想要连接的 WIFI

d. 然后输入 WIFI 的密码，再点击 **Connect**



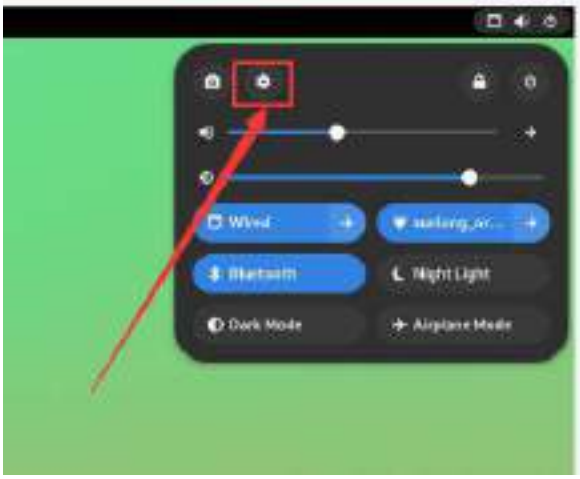
e. 然后再次进入下面的界面就能看到 WIFI 已连接



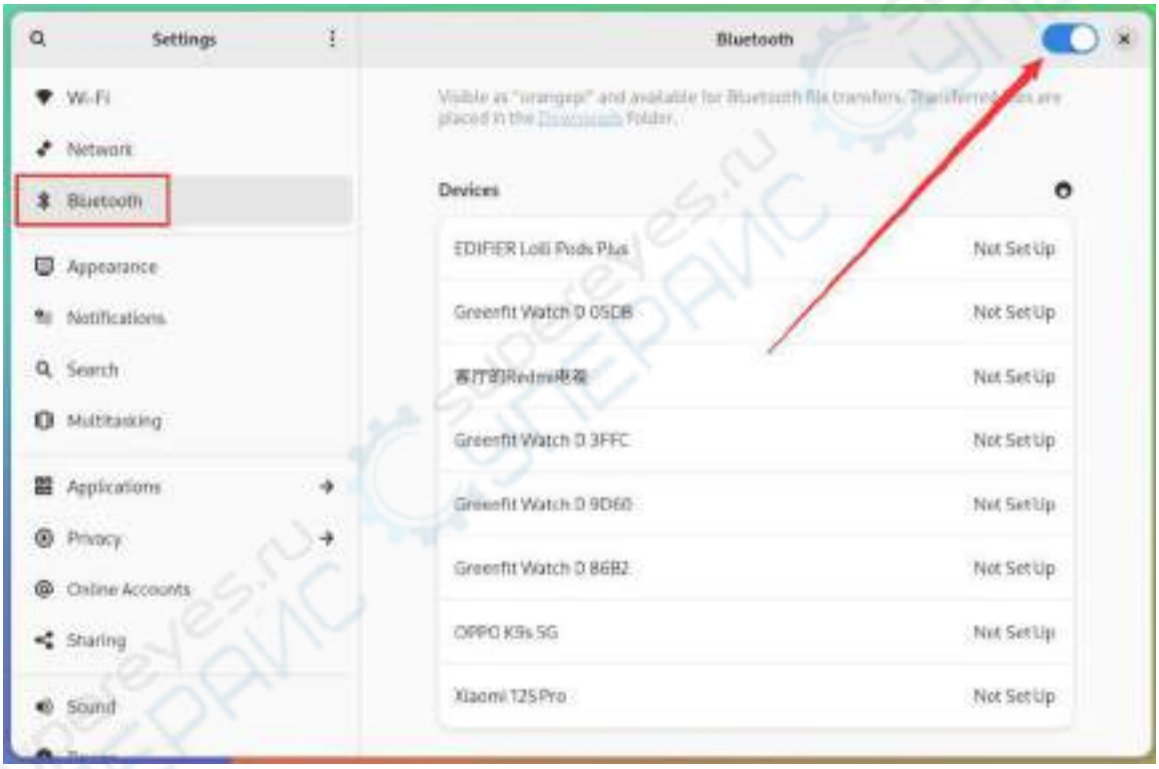
2) 蓝牙的使用示例：

a. 首先点击桌面右上角的这块区域

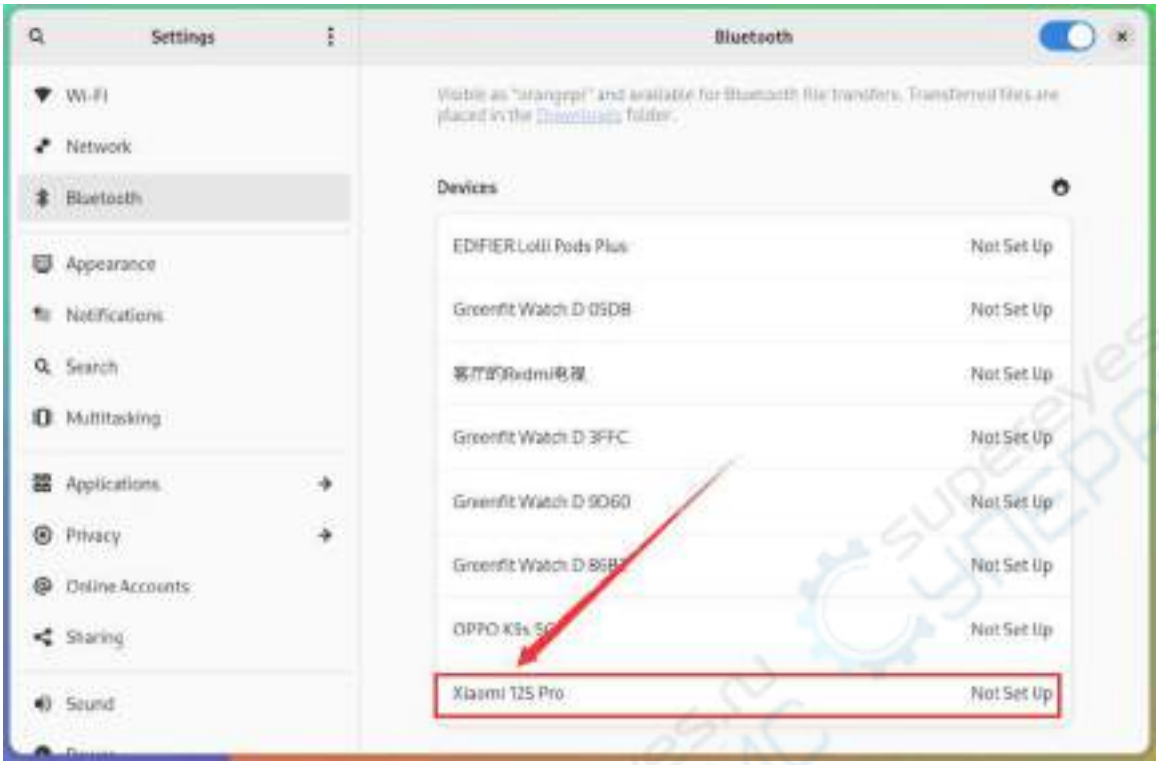
b. 然后打开设置



c. 然后在设置中选择蓝牙，并确保蓝牙右上角的开关按钮已打开

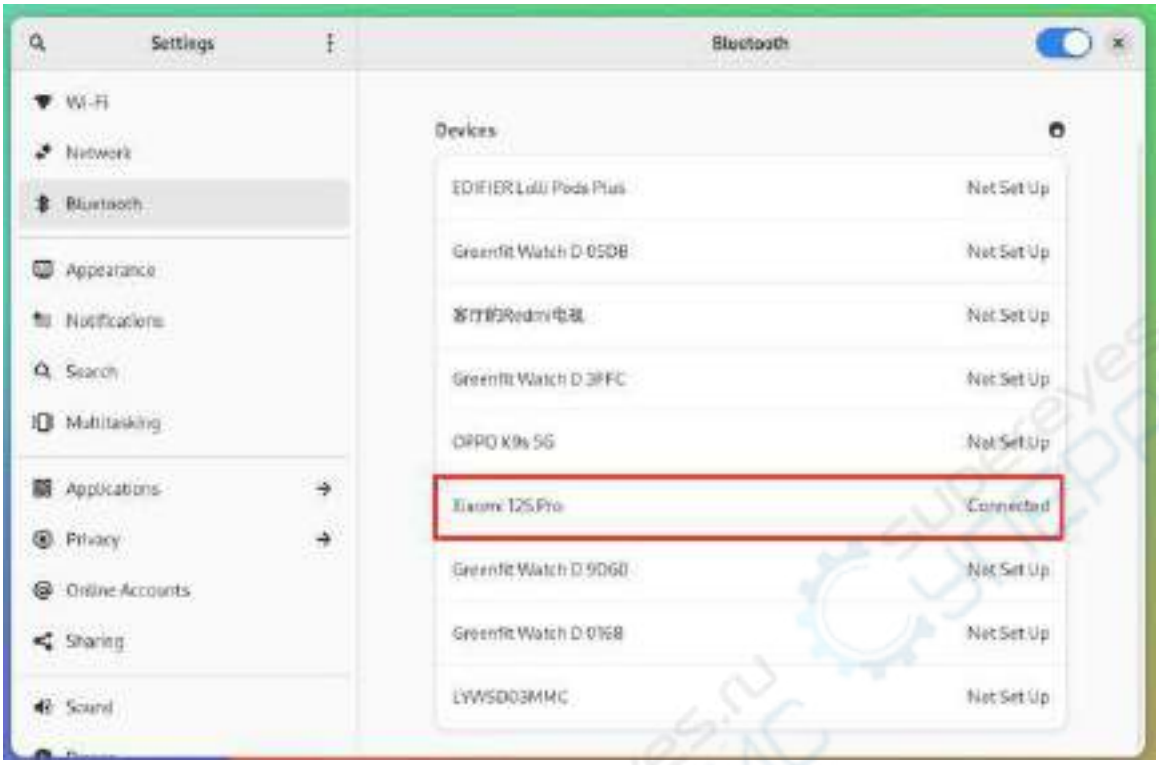


d. 然后选择想要配置对的蓝牙设备，比如和安卓手机配对

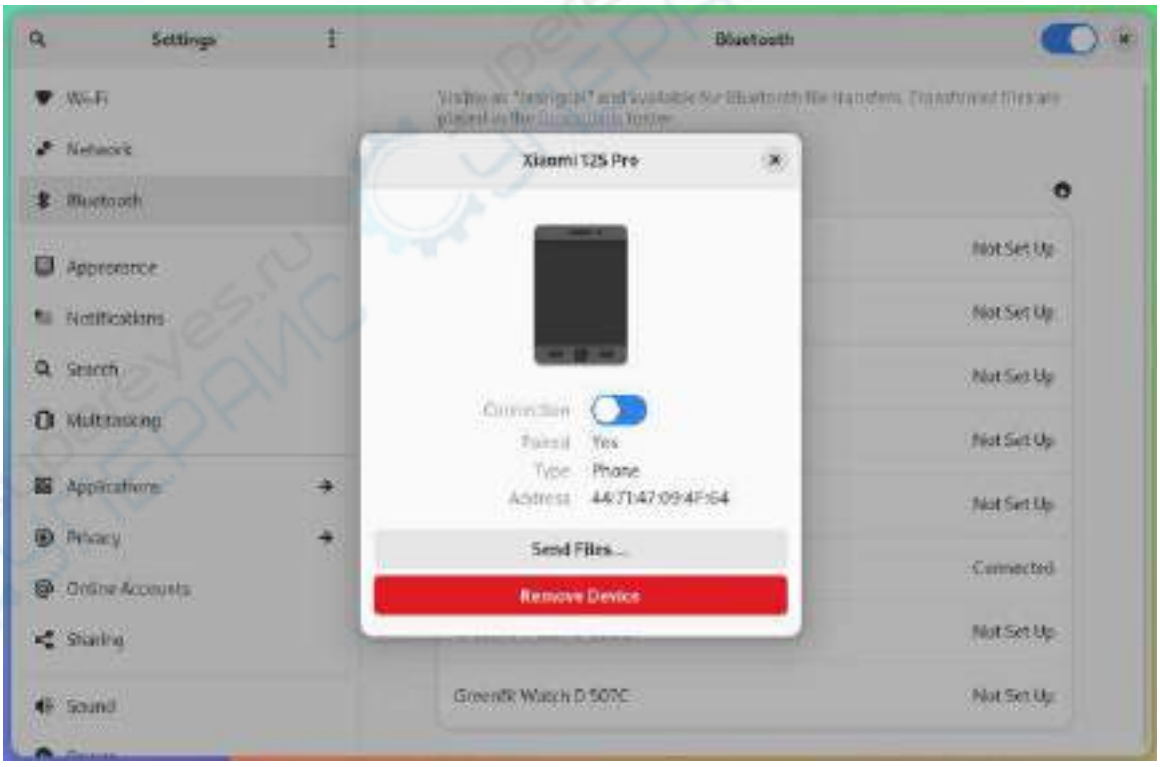


e. 然后点击 **Confirm**，手机端也需要确认配对

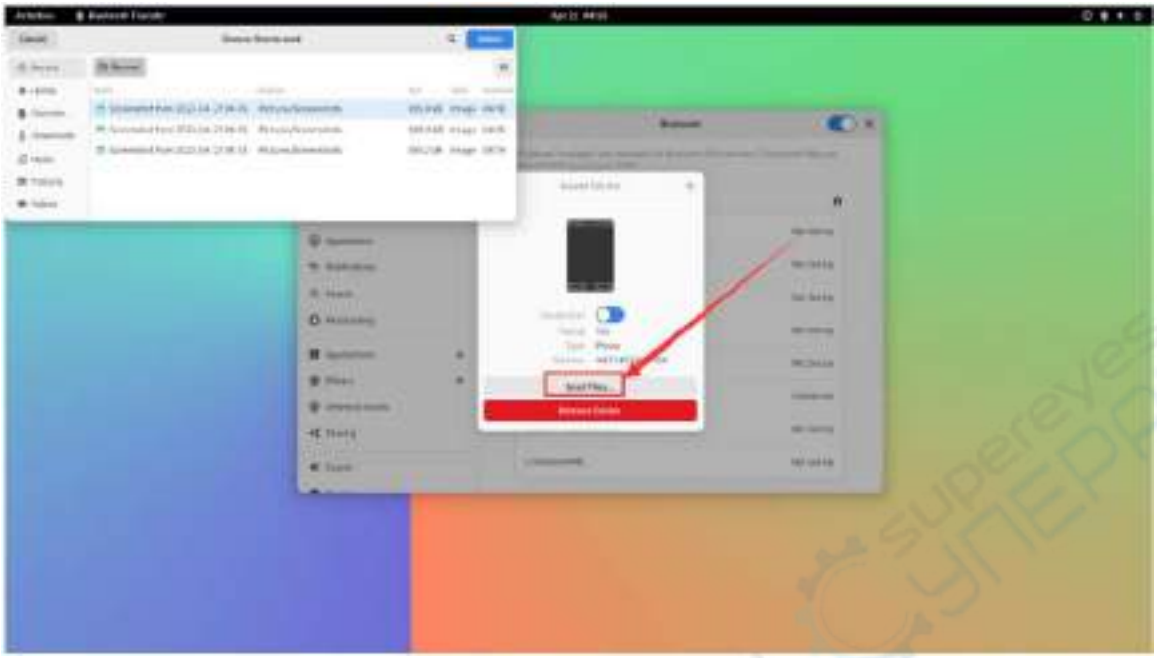
f. 蓝牙和安卓手机连接后的显示如下所示：



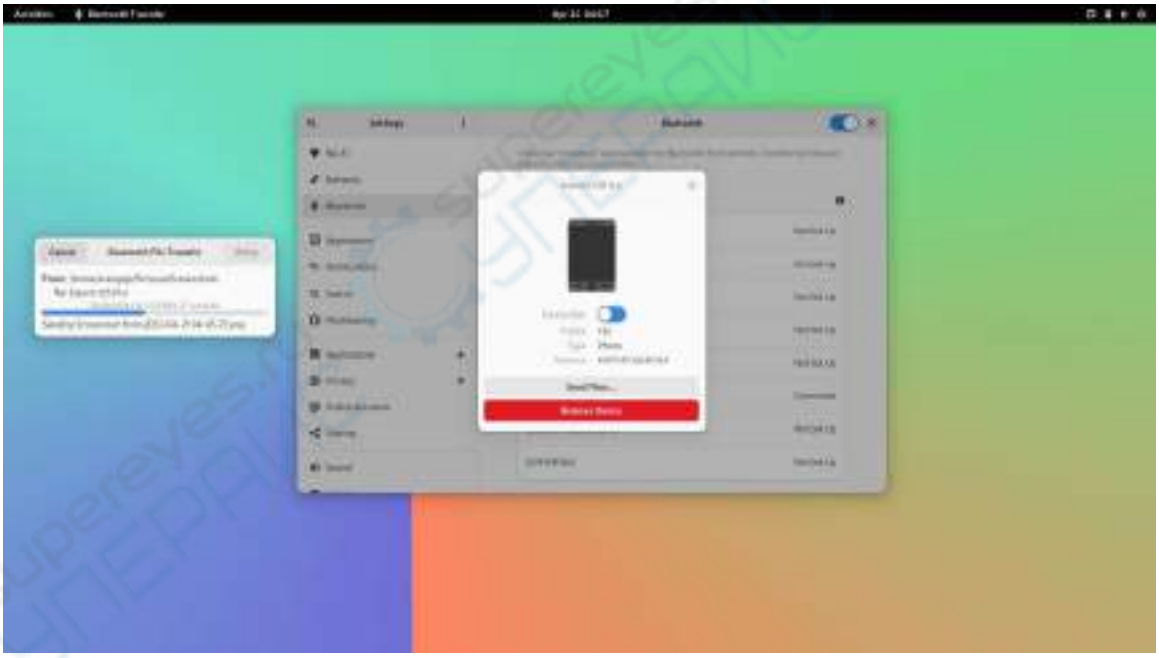
g. 然后点击已配对的蓝牙设备会弹出下图所示的操作界面



h. 此时点击 **Send Files...**就可给手机发送一个文件



i. 蓝牙发送图片给手机的示意图如下所示：



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

b. 屏幕转接板+31pin 转 40pin 排线

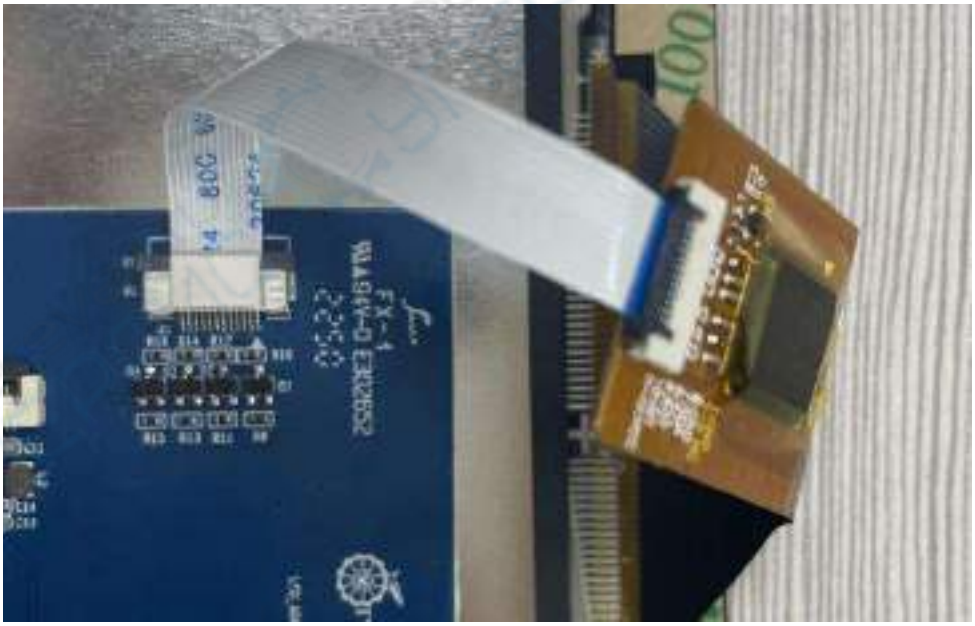
c. 30pin MIPI 排线

d. 12pin 触摸屏排线

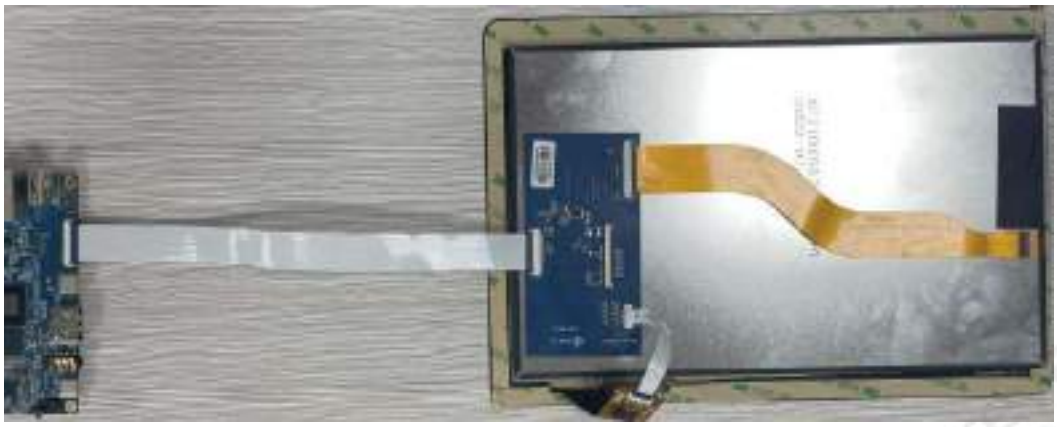
2) 按照下图将 12pin 触摸屏排线、31pin 转 40pin 排线、30pin MIPI 排线接到屏幕转接板上，注意**触摸屏排线蓝色的绝缘面朝下**，其它两根排线绝缘面朝上，如果接错会导致无显示或者不能触摸的问题

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 40pin 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



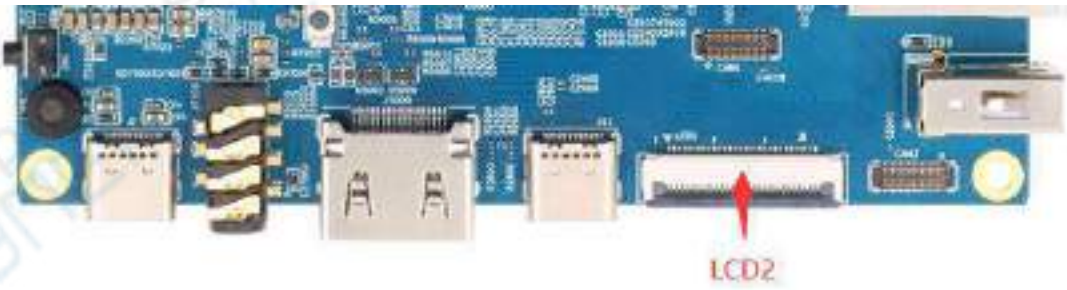
5.3.2. 打开 10.1 寸 MIPI LCD 屏幕配置的方法

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) 开发板上有两个 mipi lcd 屏幕的接口，我们定义：

a. lcd1 接口的位置为：

b. lcd2 接口的位置为：



3) 打开 mipi lcd 配置的方法如下所示：

a. 如果要打开 LCD1，请在**/boot/extlinux/extlinux.conf** 中加入下面的配置

```
[orangepi@orangepi ~]$ sudo vim /boot/extlinux/extlinux.conf
LABEL Orange Pi
LINUX /Image
```

```
FDT /dtbs/rockchip/rk3588s-orangepi-5b.dtb
FDTOVERLAYS /dtbs/rockchip/overlay/rk3588-lcd1.dtbo     #需要添加的配置
```

b. 如果要打开 LCD2，请在**/boot/extlinux/extlinux.conf** 中加入下面的配置

```
[orangepi@orangepi ~]$ sudo vim /boot/extlinux/extlinux.conf
LABEL Orange Pi
LINUX /Image
FDT /dtbs/rockchip/rk3588s-orangepi-5b.dtb
FDTOVERLAYS /dtbs/rockchip/overlay/rk3588-lcd2.dtbo     #需要添加的配置
```

c. 如果要同时打开 LCD1 和 LCD2，请在**/boot/extlinux/extlinux.conf** 中加入下面的配置（**两个 LCD 的配置需要写在一行，请不要写到两行**）

```
[orangepi@orangepi ~]$ sudo vim /boot/extlinux/extlinux.conf
LABEL Orange Pi
LINUX /Image
FDT /dtbs/rockchip/rk3588s-orangepi-5b.dtb
FDTOVERLAYS    /dtbs/rockchip/overlay/rk3588-lcd1.dtbo    /dtbs/rockchip/overlay/rk3588-lcd2.dtbo
```

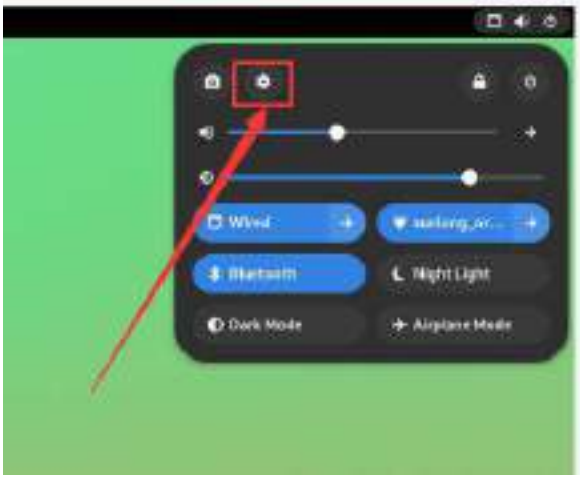
4) **然后重启 OPi OS Arch 系统**

5) 重启后可以看到 lcd 屏幕的显示如下所示（默认为竖屏）：

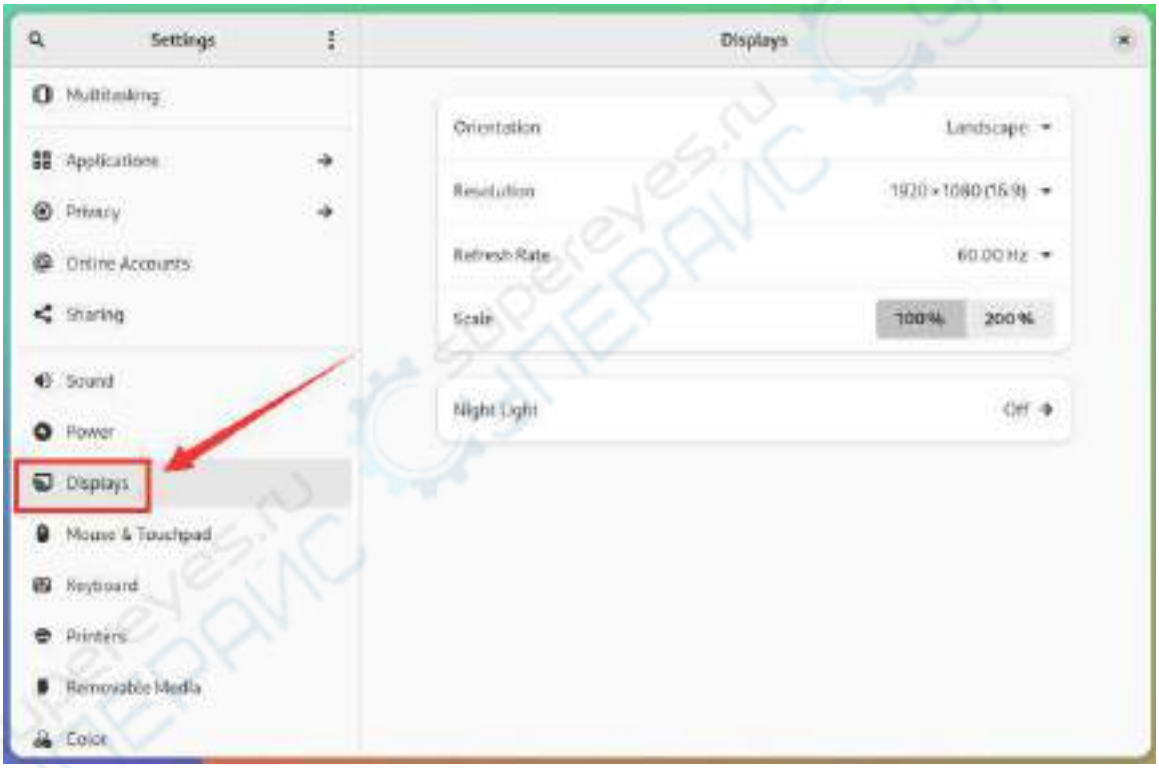
5.3.3. 旋转显示和触摸方向的方法

1) 首先点击桌面右上角的这块区域

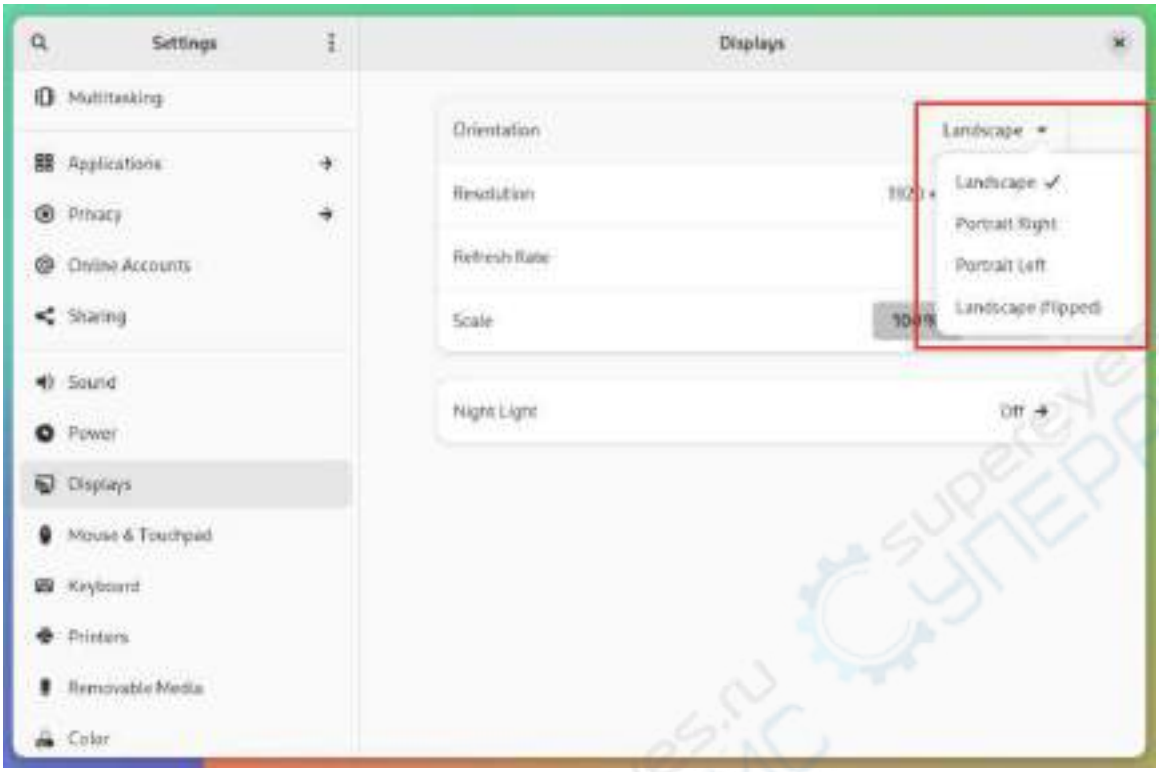
2) 然后打开设置



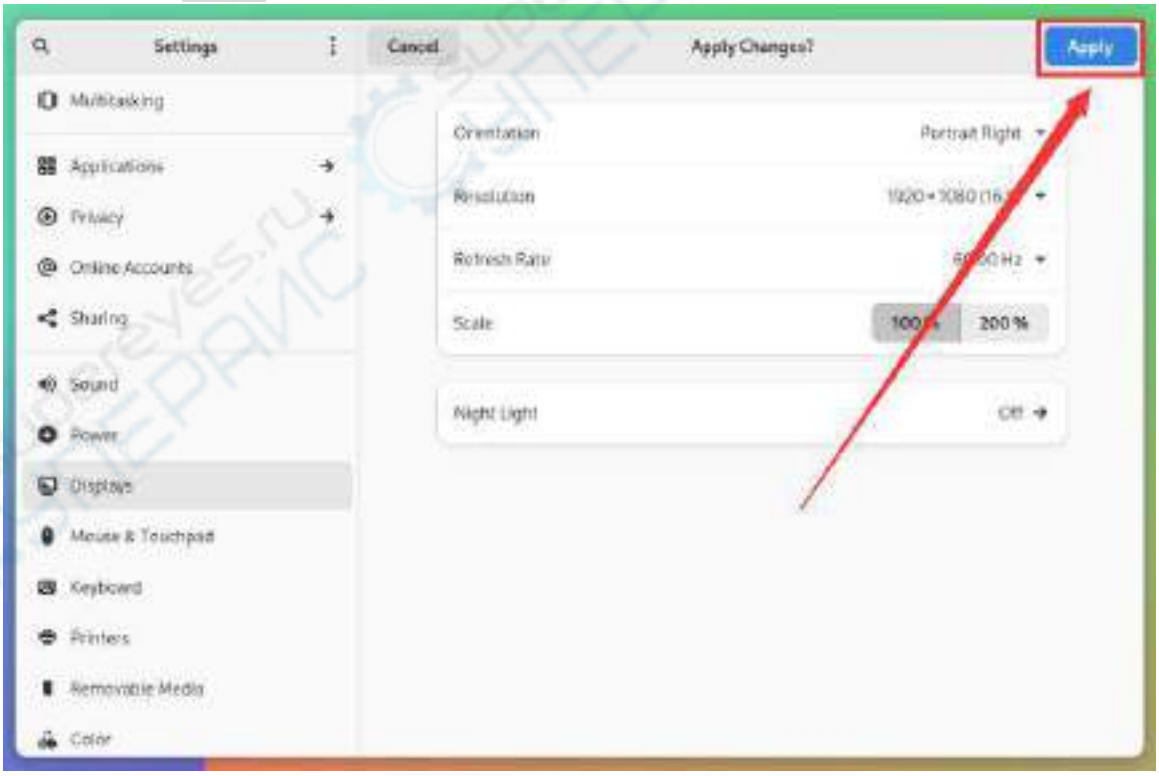
3) 然后选择 **Displays**



4) 然后在 **Displays** 的 **Orientation** 中选择想要旋转的方向

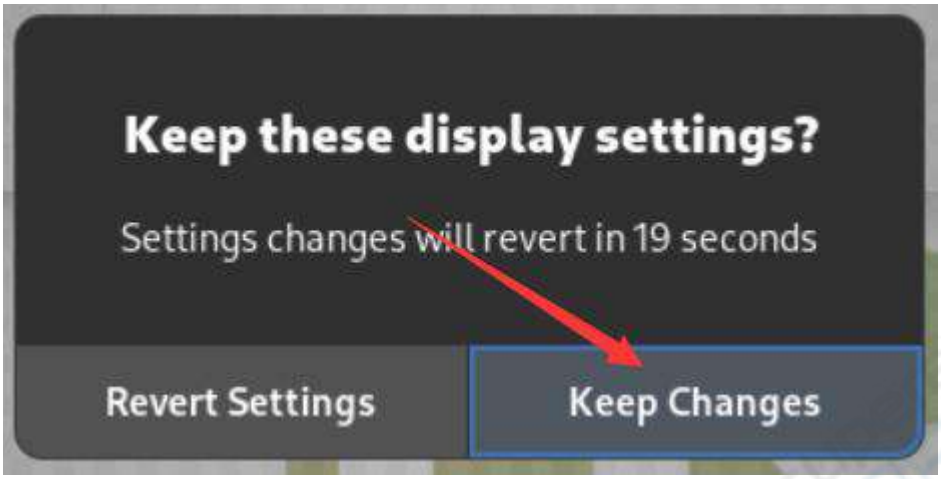


5) 然后选择 **Apply**



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

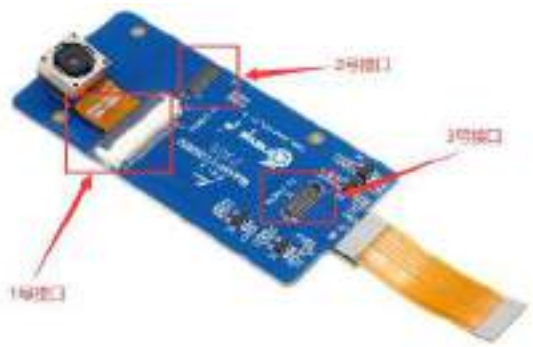
OV13850 和OV13855 摄像头使用的转接板和FPC排线是一样的，只是两款摄像头接在转接板上的位置不一样。FPC排线如下图所示，请注意FPC排线是有方向的，标注**TO MB**那端需要插到开发板的摄像头接口中，标注**TO CAMERA**那端需要插到摄像头转接板上。

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：

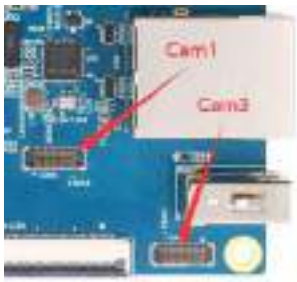
d. **1 号接口接 OV13850 摄像头**

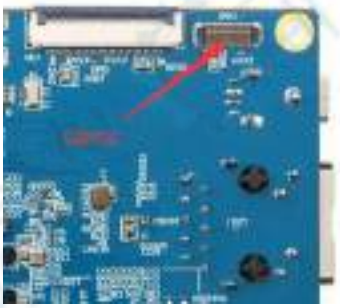
e. **2 号接口接 OV13855 摄像头**

f. 3 号接口未使用，忽略即可

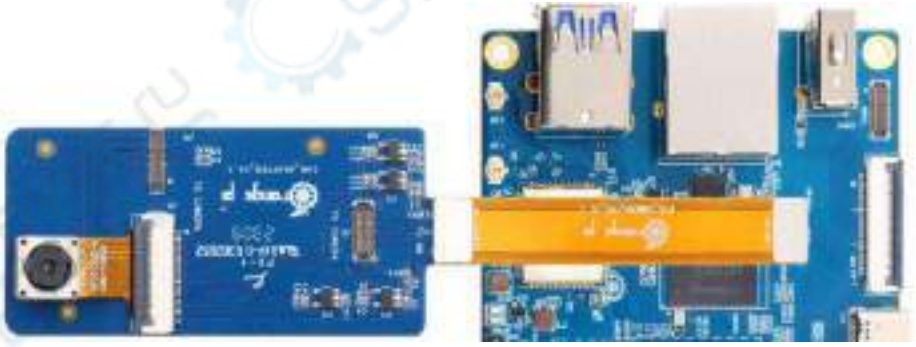


Orange Pi 5B 开发板上总共有 3 个摄像头接口，我们定义 Cam1、Cam2 和 Cam3 的位置如下图所示：





摄像头插在开发板的 Cam1 接口的方法如下所示：



摄像头插在开发板的 Cam2 接口的方法如下所示：

摄像头插在开发板的 Cam3 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-5b.dtb
FDTOVERLAYS /dtbs/rockchip/overlay/rk3588-ov13850-c1.dtbo #需要添加的配置
```

上面红色字体演示的是打开 **Cam1 接口接 ov13850** 的配置，其他接口的配置如下表所示，将对应的 dtbo 配置添加到 **FDTOVERLAYS** 后面即可。如果要同时添加多个配置，请用空格隔开。

摄像头	dtbo 的配置
Cam1 接 ov13850	**/dtbs/rockchip/overlay/rk3588-ov13850-c1.dtbo**
Cam2 接 ov13850	**/dtbs/rockchip/overlay/rk3588-ov13850-c2.dtbo**
Cam3 接 ov13850	**/dtbs/rockchip/overlay/rk3588-ov13850-c3.dtbo**

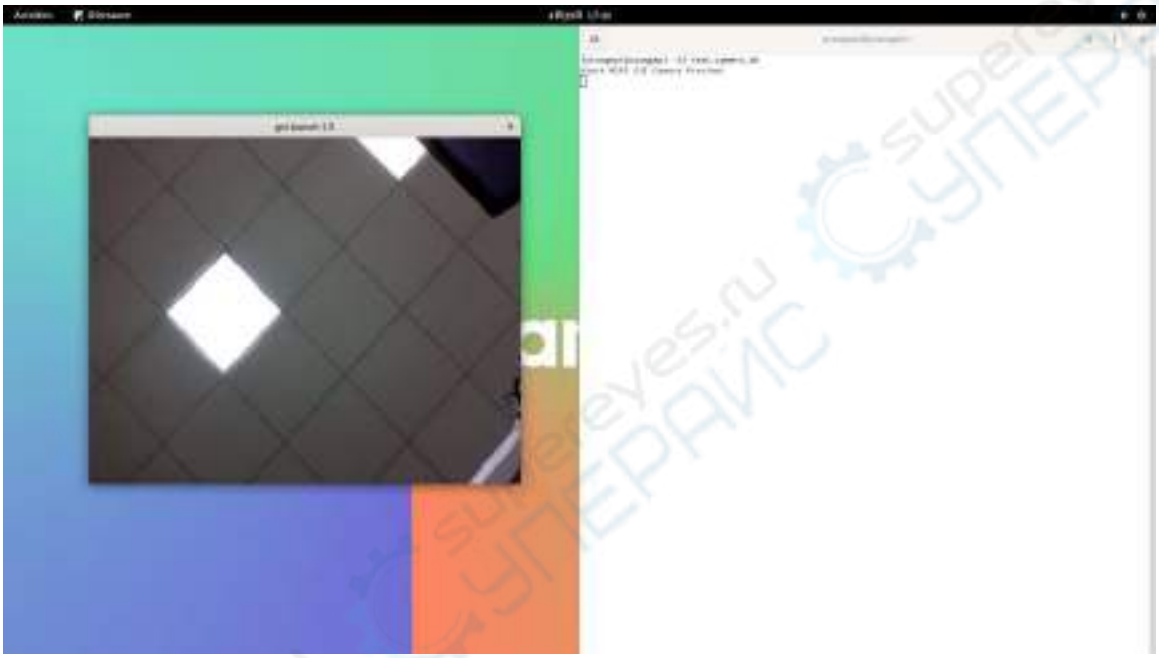
Cam1 接 ov13855	**/dtbs/rockchip/overlay/rk3588-ov13855-c1.dtbo**
Cam2 接 ov13855	**/dtbs/rockchip/overlay/rk3588-ov13855-c2.dtbo**
Cam3 接 ov13855	**/dtbs/rockchip/overlay/rk3588-ov13855-c3.dtbo**

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

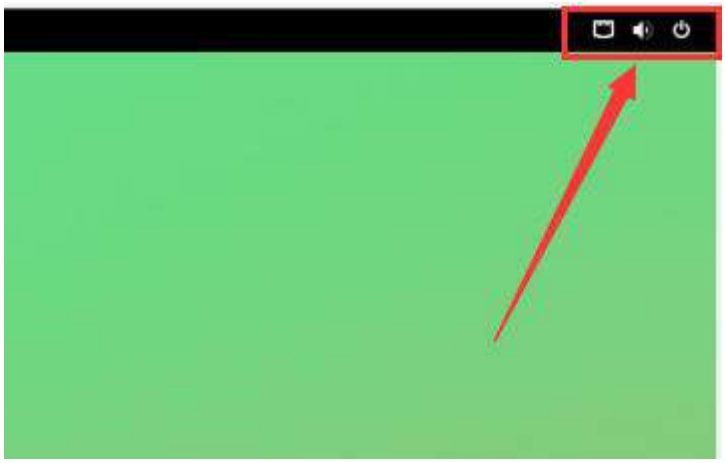
```
orangepi@orangepi:~$ test_camera.sh
```

d. 然后就能看到摄像头的预览画面了

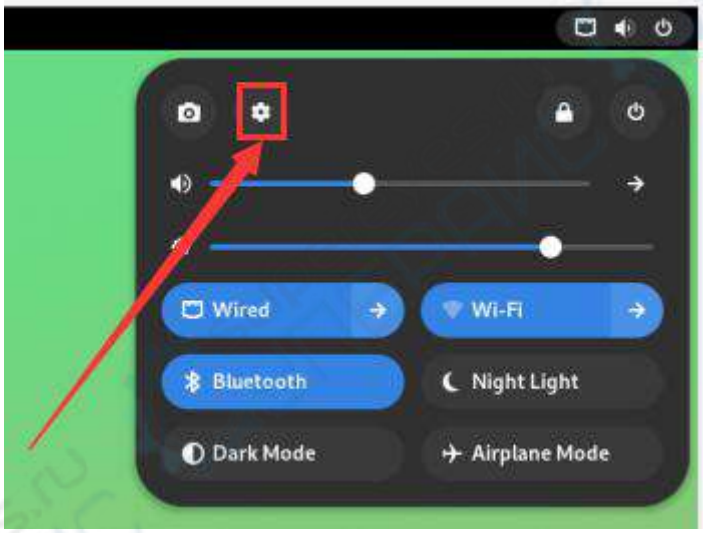


5.5. 设置中文环境以及安装中文输入法的方法

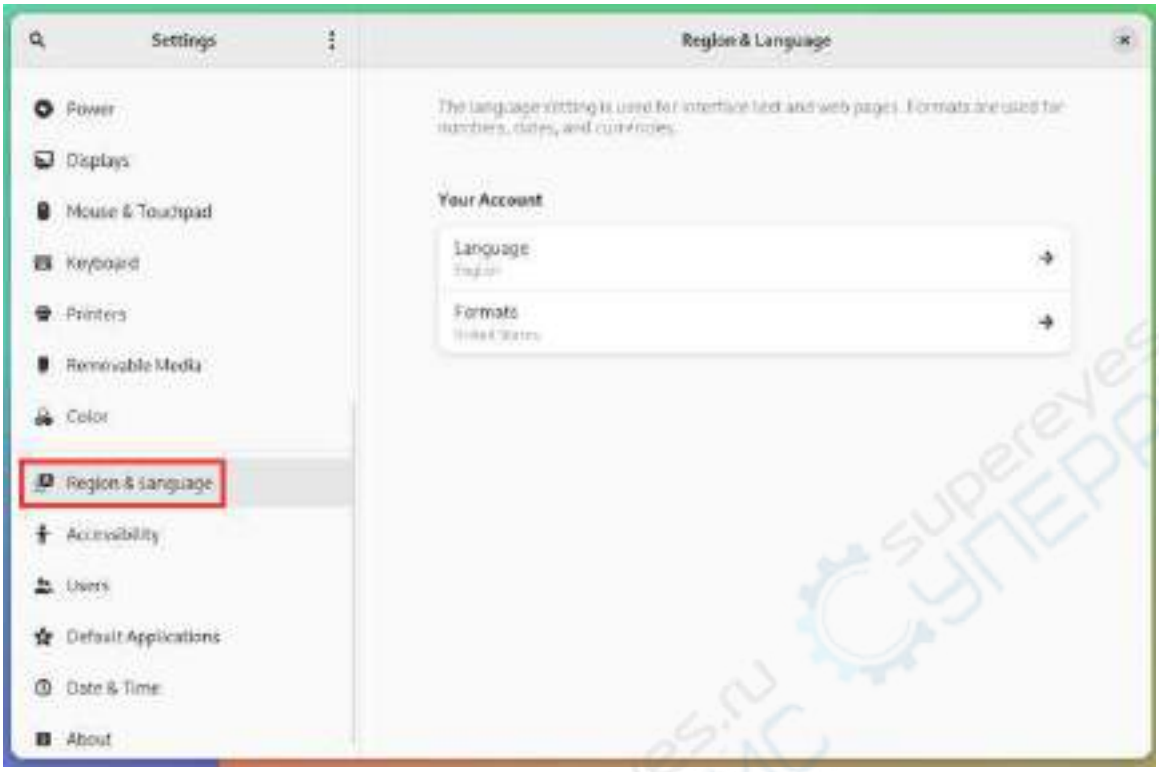
1) 首先点击桌面右上角的这块区域



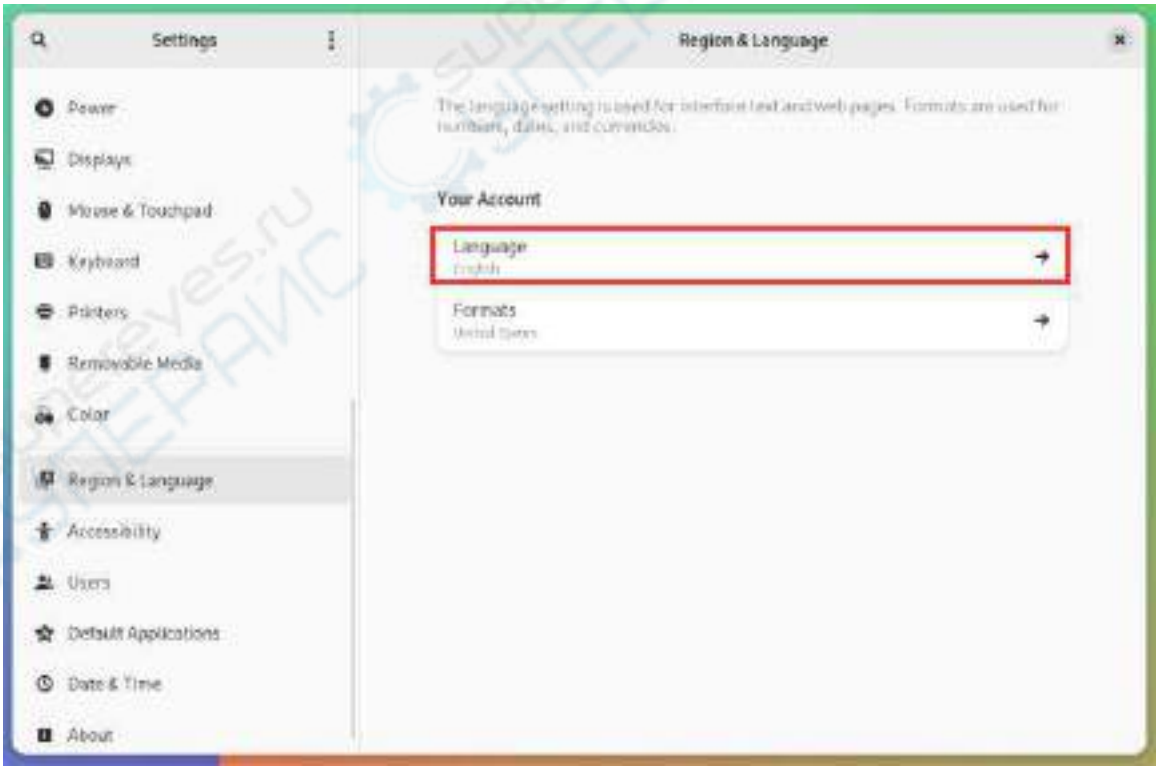
2) 然后打开设置



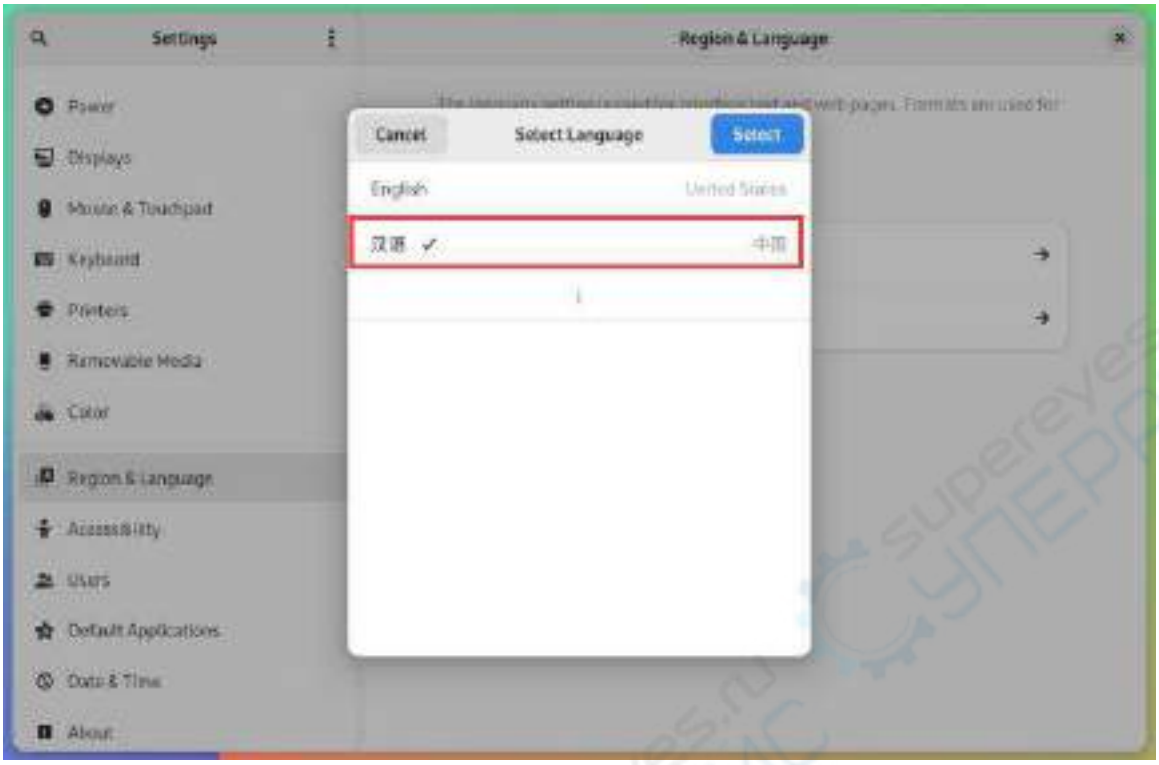
3) 然后找到 **Region & Language** 选项



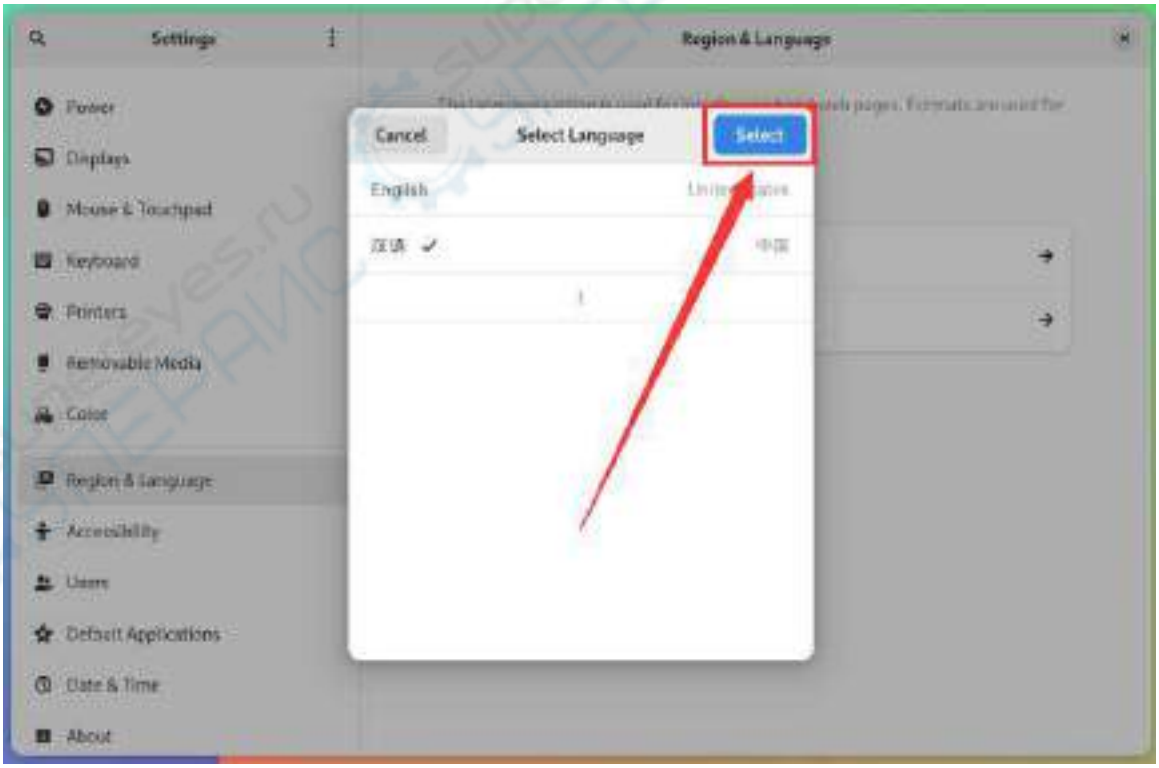
4) 然后选择 **Language**



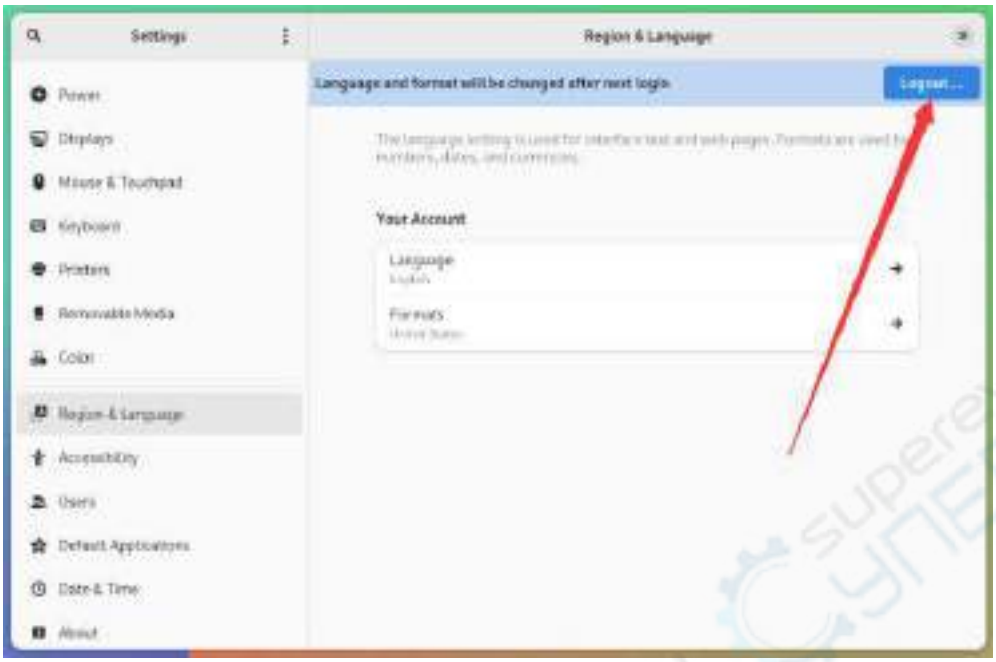
5) 然后选择汉语



6) 然后点击 **Select**



7) 然后点击 **Logout...**登出系统，再重新登入系统



8) 然后可以看到桌面都显示为中文了

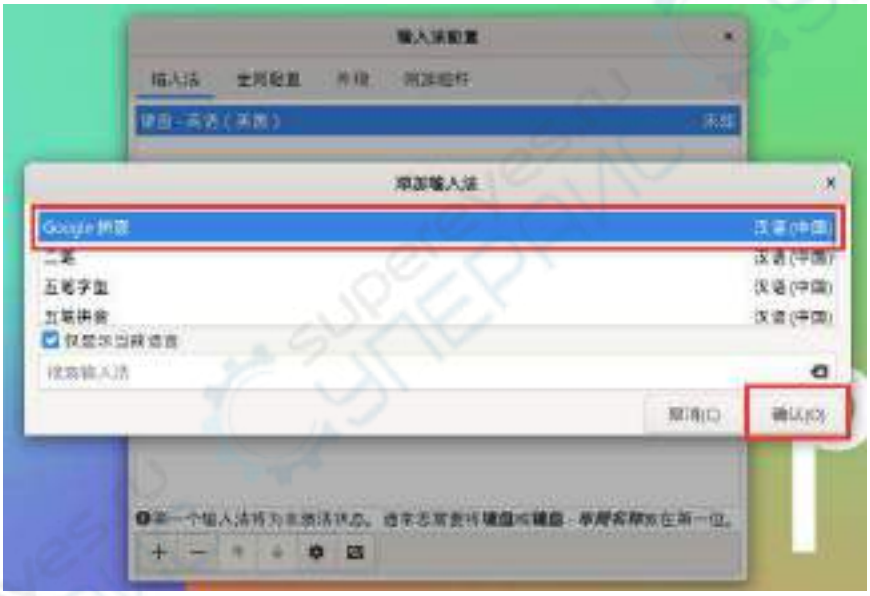
9) 然后安装下 fcitx-im 和 fcitx-configtool

```
[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**

10) 然后打开 Fcitx 配置程序

11) 然后添加 **Google 拼音**输入法



12) 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。

```
[orangepi@orangepi ~]$ gpio readall
 +------+-----+----------+--------+---+   PI5B   +---+--------+----------+-----+------+
 | 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 |     |      |
 |   54 |   2 |   PWM15  |     IN | 1 |  7 || 8  | 0 |   IN   |   RXD.0  |   3 |  131 |
 |      |     |      GND |        |   |  9 || 10 | 0 |   IN   |   TXD.0  |   4 |  132 |
 |  138 |   5 |  CAN1_RX |     IN | 1 | 11 || 12 | 1 |   IN   |  CAN2_TX |   6 |   29 |
 |  139 |   7 |  CAN1_TX |     IN | 1 | 13 || 14 |   |        |      GND |     |      |
 |   28 |   8 |  CAN2_RX |     IN | 1 | 15 || 16 | 1 |   IN   |    SDA.1 |   9 |   59 |
 |      |     |     3.3V |        |   | 17 || 18 | 1 |   IN   |    SCL.1 |  10 |   58 |
 |   49 |  11 |  SPI4_TXD |     IN | 1 | 19 || 20 |   |        |      GND |     |      |
 |   48 |  12 |  SPI4_RXD |     IN | 1 | 21 || 22 |   |        |  PowerKey |     |      |
 |   50 |  13 |  SPI4_CLK |     IN | 1 | 23 || 24 | 1 |   IN   |  SPI4_CS1 |  14 |   52 |
 |      |     |      GND |        |   | 25 || 26 | 1 |   IN   |     PWM1 |  15 |   35 |
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 | GPIO | wPi |   Name   |  Mode  | V | Physical | V |  Mode  |   Name   | wPi | GPIO |
 +------+-----+----------+--------+---+   PI5B   +---+--------+----------+-----+------+
[orangepi@orangepi ~]$
```

wiringOP 目前主要适配了设置 GPIO 口输入输出，设置 GPIO 口输出高低电平以及设置上下拉电阻的功能，像硬件 PWM 这样的功能是用不了的。

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，Orange Pi 5B 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 5B 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

```
[orangepi@orangepi ~]$ gpio readall
 +------+-----+----------+--------+---+   PI5B   +---+--------+----------+-----+------+
 | 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      |     |      |
 |   54 |   2 |   PWM15  |     IN | 1 |  7 || 8  | 0 | IN     | RXD.0    |   3 |  131 |
 |      |     |      GND |        |   |  9 || 10 | 0 | IN     | TXD.0    |   4 |  132 |
 |  138 |   5 |  CAN1_RX |     IN | 1 | 11 || 12 | 1 | IN     | CAN2_TX  |   6 |   29 |
 |  139 |   7 |  CAN1_TX |     IN | 1 | 13 || 14 |   |        | GND      |     |      |
 |   28 |   8 |  CAN2_RX |     IN | 1 | 15 || 16 | 1 | IN     | SDA.1    |   9 |   59 |
 |      |     |     3.3V |        |   | 17 || 18 | 1 | IN     | SCL.1    |  10 |   58 |
 |   49 |  11 | SPI4_TXD |     IN | 1 | 19 || 20 |   |        | GND      |     |      |
 |   48 |  12 | SPI4_RXD |     IN | 1 | 21 || 22 |   |        | PowerKey |     |      |
 |   50 |  13 | SPI4_CLK |     IN | 1 | 23 || 24 | 1 | IN     | SPI4_CS1 |  14 |   52 |
 |      |     |      GND |        |   | 25 || 26 | 1 | IN     | PWM1     |  15 |   35 |
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 | GPIO | wPi |     Name | Mode   | V | Physical | V | Mode   | Name     | wPi | GPIO |
 +------+-----+----------+--------+---+   PI5B   +---+--------+----------+-----+------+
[orangepi@orangepi ~]$
```

5.7. 26pin 接口 GPIO、I2C、UART、SPI、CAN 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-5b.dtb

FDTOVERLAYS /dtbs/rockchip/overlay/rk3588-i2c1-m2.dtbo /dtbs/rockchip/overlay/rk3588-uart0-m2.dtbo

5.7.1. 26pin GPIO 口测试

1) 开发板 26pin 中总共有 **16** 个 GPIO 口可以使用，下面以 7 号引脚——对应 GPIO 为 GPIO1_C6 ——对应 wPi 序号为 2——为例演示如何设置 GPIO 口的高低电平

```
orangepi@orangepi ~]$ gpio readall
+------+-----+----------+--------+---+   PI5B   +---+--------+----------+-----+------+
| 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      |     |      |
|   54 |   2 |   PWM15  |     IN | 1 |  7 || 8  | 0 | IN     | RXD.0    |   3 |  131 |
|      |     |      GND |        |   |  9 || 10 | 0 | IN     | TXD.0    |   4 |  132 |
|  138 |   5 | CAN1_RX  |     IN | 1 | 11 || 12 | 1 | IN     | CAN2_TX  |   6 |   29 |
```

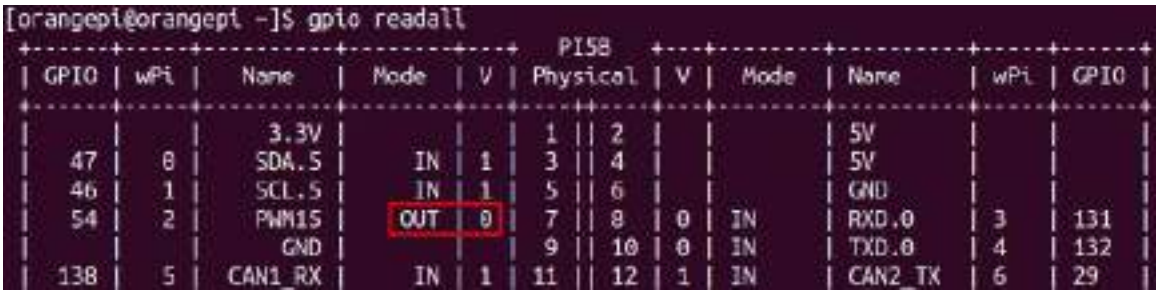
2) 首先设置 GPIO 口为输出模式，其中第三个参数需要输入引脚对应的 wPi 的序号

```
[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
```

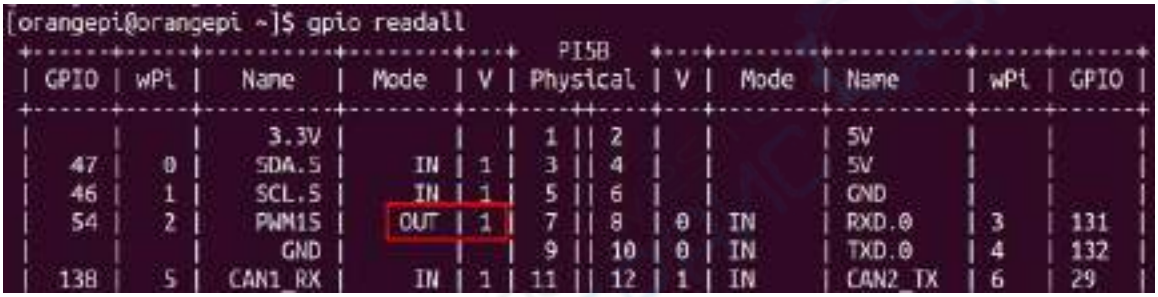
使用 gpio readall 可以看到 7 号引脚的值(V)变为了 0



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



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

注意，Orange Pi 5 只有下面 4 个 GPIO 引脚可以正常设置上下拉电阻功能，其它的 GPIO 引脚因为外部有 3.3V 上拉，所以设置下拉是无效的。

```
[orangepi@orangepi ~]$ gpio readall
                                        PI5B
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 | 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    |     |      |
 |   54 |   2 |   PWM15  |     IN | 1 |  7 ||  8 | 0 |     IN | RXD.0    |   3 |  131 |
 |      |     |      GND |        |   |  9 || 10 | 0 |     IN | TXD.0    |   4 |  132 |
 |  138 |   5 | CAN1_RX  |     IN | 1 | 11 || 12 | 1 |     IN | CAN2_TX  |   6 |   29 |
 |  139 |   7 | CAN1_TX  |     IN | 1 | 13 || 14 |   |        |   GND    |     |      |
 |   28 |   8 | CAN2_RX  |     IN | 1 | 15 || 16 | 1 |     IN | SDA.1    |   9 |   59 |
 |      |     |    3.3V  |        |   | 17 || 18 | 1 |     IN | SCL.1    |  10 |   58 |
 |   49 |  11 | SPI4_TXD |     IN | 1 | 19 || 20 |   |        |   GND    |     |      |
 |   48 |  12 | SPI4_RXD |     IN | 1 | 21 || 22 |   |        | PowerKey |     |      |
 |   50 |  13 | SPI4_CLK |     IN | 1 | 23 || 24 | 1 |     IN | SPI4_CS1 |  14 |   52 |
 |      |     |      GND |        |   | 25 || 26 | 1 |     IN | PWM1     |  15 |   35 |
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 | GPIO | wPi |   Name   |  Mode  | V | Physical | V |  Mode  |   Name   | wPi | GPIO |
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
                                        PI5B
[orangepi@orangepi ~]$
```

1) 下面以 11 号引脚——对应 GPIO 为 GPIO4_B2 ——对应 wPi 序号为 5——为例演示如何设置 GPIO 口的上下拉电阻

```
[orangepi@orangepi ~]$ gpio readall
                                        PI5B
 +------+-----+----------+--------+---+----++----+---+--------+----------+-----+------+
 | 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    |     |      |
 |   54 |   2 |   PWM15  |     IN | 1 |  7 ||  8 | 0 |     IN | RXD.0    |   3 |  131 |
 |      |     |      GND |        |   |  9 || 10 | 0 |     IN | TXD.0    |   4 |  132 |
 |  138 |   5 | CAN1_RX  |     IN | 1 | 11 || 12 | 1 |     IN | CAN2_TX  |   6 |   29 |
 |  139 |   7 | CAN1_TX  |     IN | 1 | 13 || 14 |   |        |   GND    |     |      |
 |   28 |   8 | CAN2_RX  |     IN | 1 | 15 || 16 | 1 |     IN | SDA.1    |   9 |   59 |
 |      |     |    3.3V  |        |   | 17 || 18 | 1 |     IN | SCL.1    |  10 |   58 |
```

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 5 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 5 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 5
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 5 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 5
0
```

5. 7. 3. 26pin SPI 测试

1) 由 26pin 接口的原理图可知，Orange Pi 5B 可用的 spi 为 spi4

在 OPi OS Arch 系统中，26pin 中的 spi4 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 spi4。

[orangepi@orangepi ~]$ sudo vim /boot/extlinux/extlinux.conf
LABEL Orange Pi
LINUX /Image
FDT /dtbs/rockchip/rk3588s-orangepi-5b.dtb
FDTOVERLAYS /dtbs/rockchip/overlay/rk3588-spi4-m0-cs1-spidev.dtbo

2) 先查看下 OPi OS Arch 系统中是否存在 **spidev4.1** 的设备节点，如果存在，说明 SPI4 已经设置好了，可以直接使用

```
[orangepi@orangepi ~]$ ls /dev/spidev4.1
/dev/spidev4.1
```

3) 先不短接 SPI4 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/spidev4.1
spi mode: 0x0
bits per word: 8
max speed: 500000 Hz (500 KHz)
TX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF F0 0D    | ......................@....▒▒.
RX | FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF    | ................................
```

4) 然后短接 SPI4 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/spidev4.1
spi mode: 0x0
bits per word: 8
max speed: 500000 Hz (500 KHz)
TX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF F0 0D    | ......................@....▒▒.
RX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF F0 0D    | ......................@....▒▒.
```

5. 7. 4. 26pin I2C 测试

1) 由下表可知，Orange Pi 5B 可用的 i2c 为 i2c1、i2c3 和 i2c5 共三组 i2c 总线

从上表中可以看到，i2c1 既可以从 26pin 中的 12 和 15 号引脚导出来（i2c1_m2），也可以从 26pin 中的 16 和 18 号引脚导出来（i2c1_m4），请按照自己的需要选择一组即可。请不要以为这是两组不同的 i2c 总线。

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 i2c1、i2c3 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-5b.dtb
FDTOVERLAYS   /dtbs/rockchip/overlay/rk3588-i2c1-m2.dtbo
/dtbs/rockchip/overlay/rk3588-i2c3-m0.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-*
/dev/i2c-0   /dev/i2c-10   /dev/i2c-3   /dev/i2c-6   /dev/i2c-9
/dev/i2c-1   /dev/i2c-2    /dev/i2c-5   /dev/i2c-7
```

3) 然后在 26pin 接头的 i2c 引脚上接一个 i2c 设备

	i2c1-m2	i2c1-m4	i2c3-m0	i2c5-m3
sda 引脚	对应 12 号引脚	对应 16 号引脚	对应 21 号引脚	对应 3 号引脚
sck 引脚	对应 15 号引脚	对应 18 号引脚	对应 19 号引脚	对应 5 号引脚
3.3v 引脚	对应 1 号引脚	对应 1 号引脚	对应 1 号引脚	对应 1 号引脚
5v 引脚	对应 2 号引脚	对应 2 号引脚	对应 2 号引脚	对应 2 号引脚
gnd 引脚	对应 6 号引脚	对应 6 号引脚	对应 6 号引脚	对应 6 号引脚

3.3v 引脚和 5v 引脚一般只用接一个即可，请根据具体接的 i2c 设备来选择接 3.3v 引脚还是 5v 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 1          #i2c1 的命令
[orangepi@orangepi ~]$ sudo i2cdetect -y 3          #i2c3 的命令
[orangepi@orangepi ~]$ sudo i2cdetect -y 5          #i2c5 的命令
```

5.7.5. 26pin 的 UART 测试

1) 由下表可知，Orange Pi 5B 可用的 uart 为 uart0、uart1、uart3 和 uart4 共四组 uart 总线

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 uart0、uart1、uart3 和 uart4，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-5b.dtb
FDTOVERLAYS   /dtbs/rockchip/overlay/rk3588-uart0-m2.dtbo
/dtbs/rockchip/overlay/rk3588-uart1-m1.dtbo
/dtbs/rockchip/overlay/rk3588-uart3-m0.dtbo
/dtbs/rockchip/overlay/rk3588-uart4-m0.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/ttyS0   /dev/ttyS1   /dev/ttyS3   /dev/ttyS4   /dev/ttyS9
```

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

	uart0	uart1	uart3	uart4
tx 引脚	对应 8 号引脚	对应 5 号引脚	对应 19 号引脚	对应 18 号引脚
rx 引脚	对应 10 号引脚	对应 3 号引脚	对应 21 号引脚	对应 16 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 UART0

```
[orangepi@orangepi ~]$ sudo gpio serial /dev/ttyS0
[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 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
```

c. 测试 UART3

```
[orangepi@orangepi ~]$ sudo gpio serial /dev/ttyS3
[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 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
```

5.7.6. PWM 的测试方法

1) 由下表可知，Orange Pi 5 可用的 pwm 有 pwm0、pwm1、pwm3、pwm13、pwm14 和 pwm15 共六路 pwm

从上表中可以看到:

pwm1 既可以从 26pin 中的 16 号引脚导出来（pwm1_m1），也可以从 26pin 中的 26 号脚导出来（pwm1_m2）

pwm3 既可以从 26pin 中的 15 号引脚导出来（pwm3_m0），也可以从 26pin 中

的 23 号脚导出来（pwm3_m2）

请按照自己的需要选择对应的引脚即可。请不要以为这是两路不同的 pwm 总线。

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、pwm13、pwm14 和 pwm15，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-5b.dtb
FDTOVERLAYS   /dtbs/rockchip/overlay/rk3588-pwm0-m1.dtbo
/dtbs/rockchip/overlay/rk3588-pwm13-m2.dtbo
/dtbs/rockchip/overlay/rk3588-pwm14-m1.dtbo
/dtbs/rockchip/overlay/rk3588-pwm15-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 pwm15 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
```

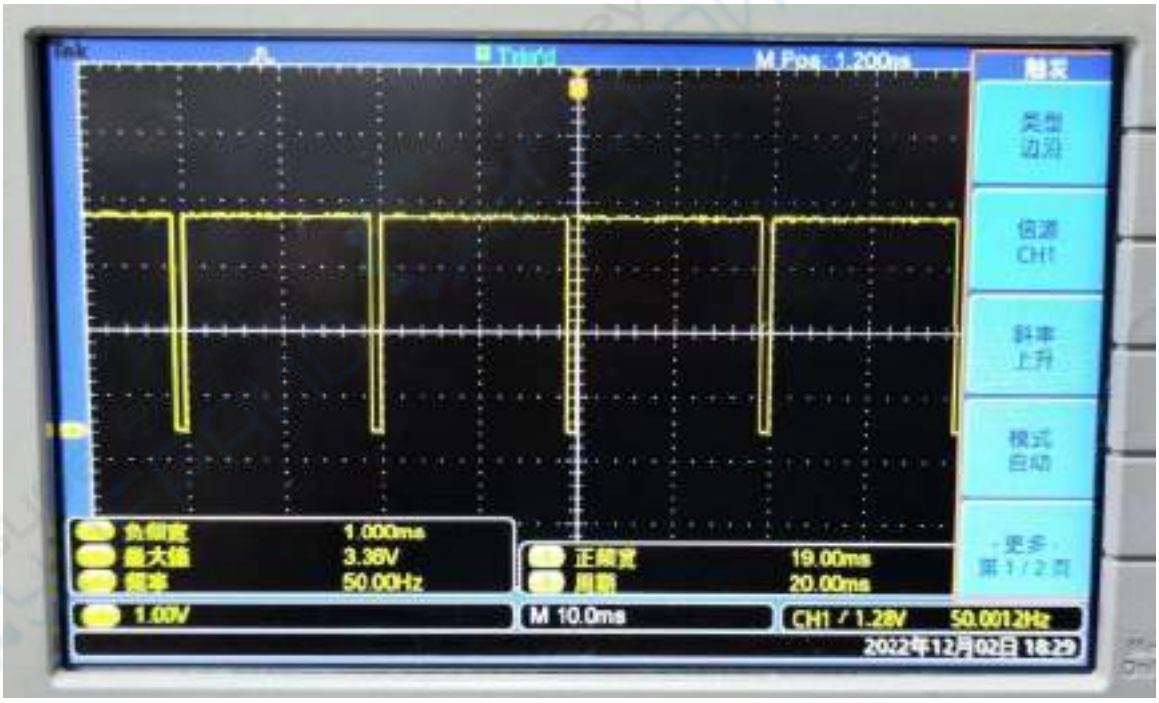
3) 上面哪个 pwmchip 对应 pwm15 呢，我们先查看下 **ls /sys/class/pwm/ -l** 命令的输出，如下所示：

```
[orangepi@orangepi ~]$ ls /sys/class/pwm/ -l
total 0
lrwxrwxrwx 1 root root 0 Apr 20 07:33 pwmchip0 -> ../../devices/platform/fd8b0020.pwm/pwm/pwmchip0
lrwxrwxrwx 1 root root 0 Apr 20 07:33 pwmchip1 -> ../../devices/platform/febd0020.pwm/pwm/pwmchip1
lrwxrwxrwx 1 root root 0 Apr 20 07:33 pwmchip2 -> ../../devices/platform/febf0030.pwm/pwm/pwmchip2
[orangepi@orangepi ~]$
```

4) 然后由下表可知，pwm15 寄存器的基地址为 febf0030，再看 **ls /sys/class/pwm/ -l** 命令的输出，可以看到 pwmchip2 中链接到了 febf0030.pwm，所以 pwm15 对应 pwmchip 为 pwmchip2

5) 然后使用下面的命令可以让 pwm15 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/pwmchip2/export
[root@orangepi orangepi]# echo 20000000 > /sys/class/pwm/pwmchip2/pwm0/period
[root@orangepi orangepi]# echo 1000000 > /sys/class/pwm/pwmchip2/pwm0/duty_cycle
[root@orangepi orangepi]# echo 1 > /sys/class/pwm/pwmchip2/pwm0/enable
```



6) 上面演示的 pwm15 的测试方法，其他 pwm 测试方法都是类似的。

5.7.7. CAN 的测试方法

1) 由下表可知，Orange Pi 5B 可用的 CAN 总线为 CAN1 和 CAN2 共两组 CAN 总线

在 OPi OS Arch 系统中，26pin 中的 CAN 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 CAN1 和 CAN2，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-5b.dtb
FDTOVERLAYS   /dtbs/rockchip/overlay/rk3588-can1-m1.dtbo
/dtbs/rockchip/overlay/rk3588-can2-m1.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。

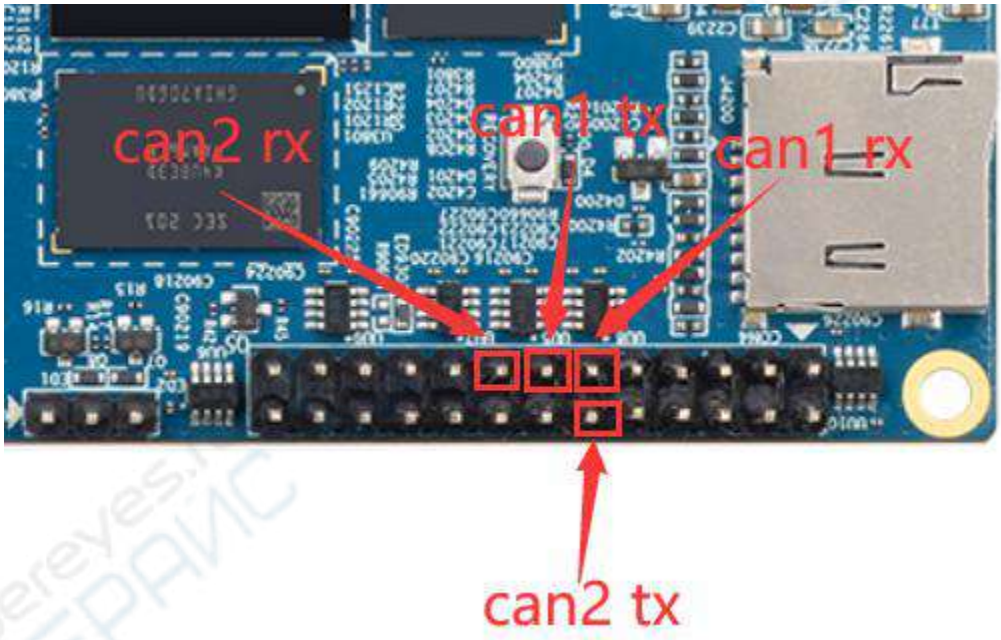
2) 进入 OPi OS Arch 系统后，使用 **sudo ifconfig -a** 命令如果能看到 CAN 的设备节点，就说明 CAN 已正确打开了

```
[orangepi@orangepi ~]$ sudo pacman -Syy net-tools
[orangepi@orangepi ~]$ sudo ifconfig -a
can0: flags=128<NOARP>  mtu 16
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 10  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 91
```

```
can1: flags=128<NOARP>  mtu 16
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 10  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 92
```

3) CAN1 和 CAN2 对应的引脚为

	CAN1	CAN2
TX 引脚	对应 13 号引脚	对应 12 号引脚
RX 引脚	对应 11 号引脚	对应 15 号引脚



4) 使用 CANalyst-II 分析仪测试 CAN 收发消息的方法请参考下**使用 CANalyst-II 分析仪测试收发消息**一小节的内容。

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（尤其是 SSD 启动时）。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-desktop-amd64.iso
或者
https://repo.huaweicloud.com/ubuntu-releases/22.04/ubuntu-22.04.1-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。

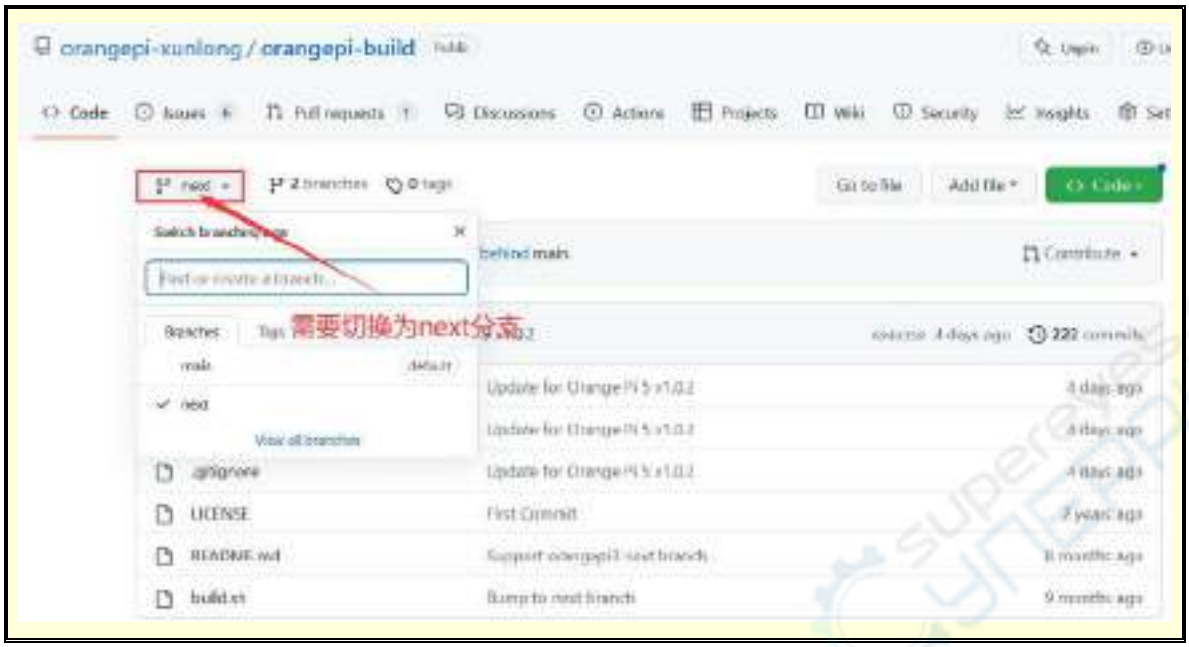
6.2. 获取 linux sdk 的源码

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 5B 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**

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。如果以后支持主线内核了，就会添加一个 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
```

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. linux 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-rk3588
```

b. 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-rk3588** 的文件夹存放的就是 RK3588/RK3588S 系列开发板 legacy 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

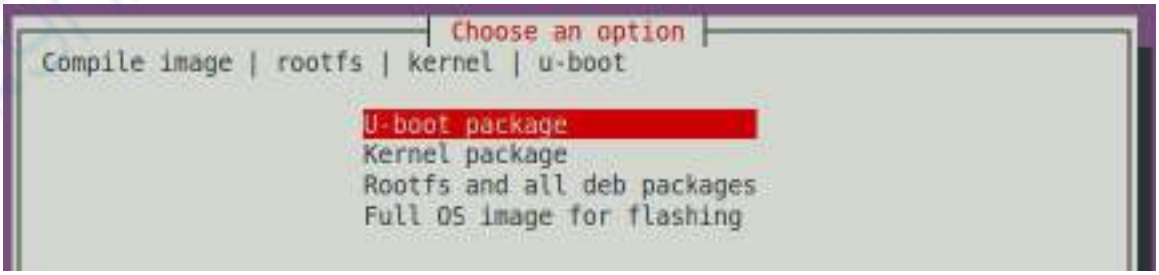
```
test@test:~/orangepi-build$ ls
build.sh  external  kernel  LICENSE  output  README.md  scripts  toolchains  u-boot  userpatches
```

6.3. 编译 u-boot

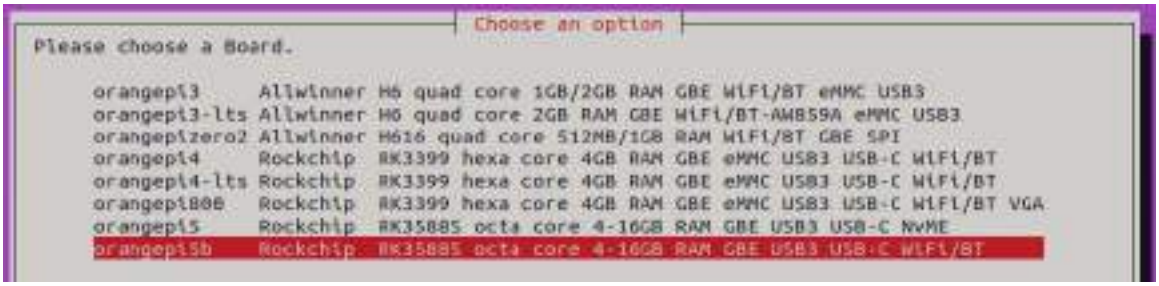
1) 运行 build.sh 脚本，记得加 sudo 权限

```
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-orangepi5b_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=orangepi5b 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-orangepi5b_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-orangepi5b_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-orangepi5b_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-orangepi5b_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-orangepi5b_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-orangepi5b-legacy
```

c. 再安装刚才上传的新的 u-boot 的 deb 包

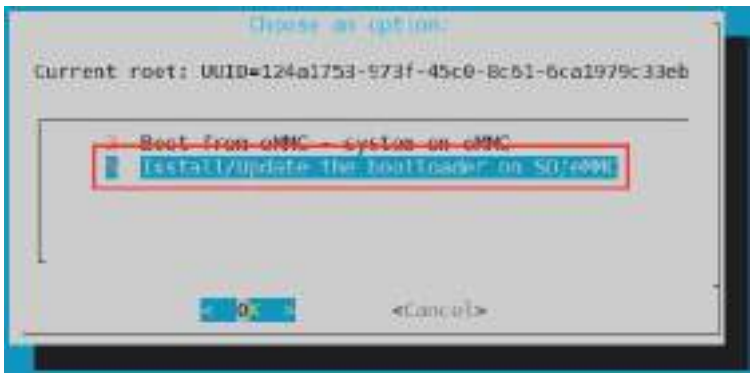
```
root@orangepi:~# dpkg -i linux-u-boot-legacy-orangepi5b_1.0.0_arm64.deb
```

d. 然后运行 nand-sata-install 脚本

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

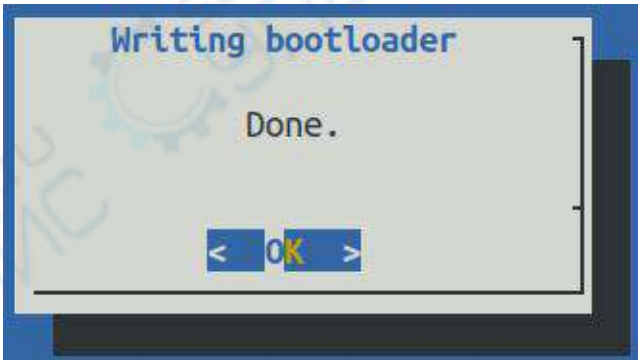
e. 然后选择 **5 Install/Update the bootloader on SD/eMM** 来更新 TF 卡中的

u-boot（如果是 eMMC 启动的 linux 系统，更新的就是 eMMC 中的 u-boot）



f. 按下回车键后首先会弹出一个 Warring

g. 再按下回车键就会开始更新 u-boot，更新完后会显示下面的信息



h. 然后就可以重启开发板来测试 u-boot 的修改是否生效了

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_5b_defconfig

b. u-boot 2017.09 源码中，开发板使用 dts 文件为

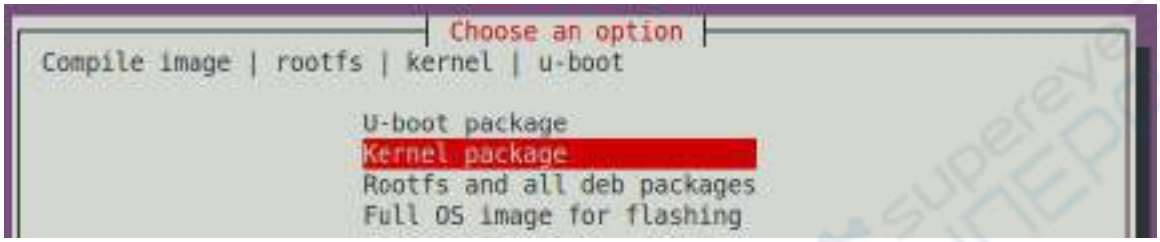
orangepi-build/u-boot/v2017.09-rk3588/arch/arm/dts/rk3588s-orangepi-5b.dts

6.4. 编译 linux 内核

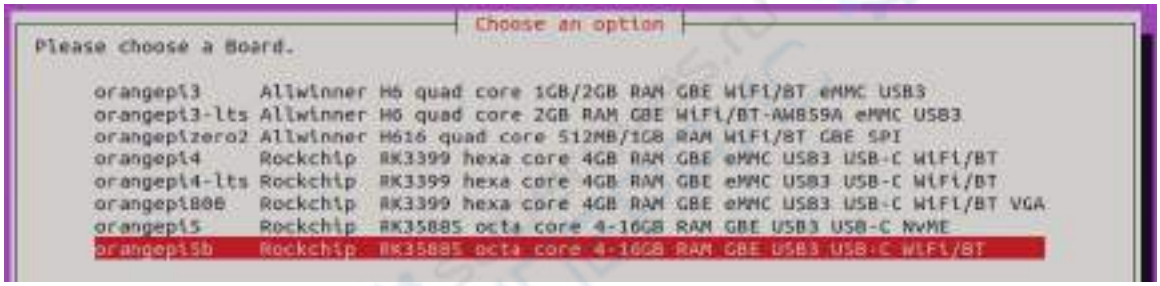
1) 运行 build.sh 脚本，记得加 sudo 权限

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

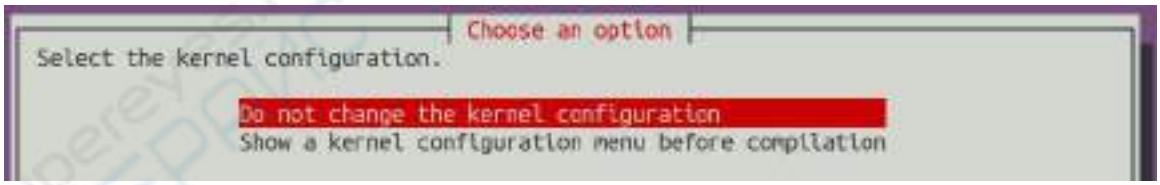
2) 选择 **Kernel package**，然后回车



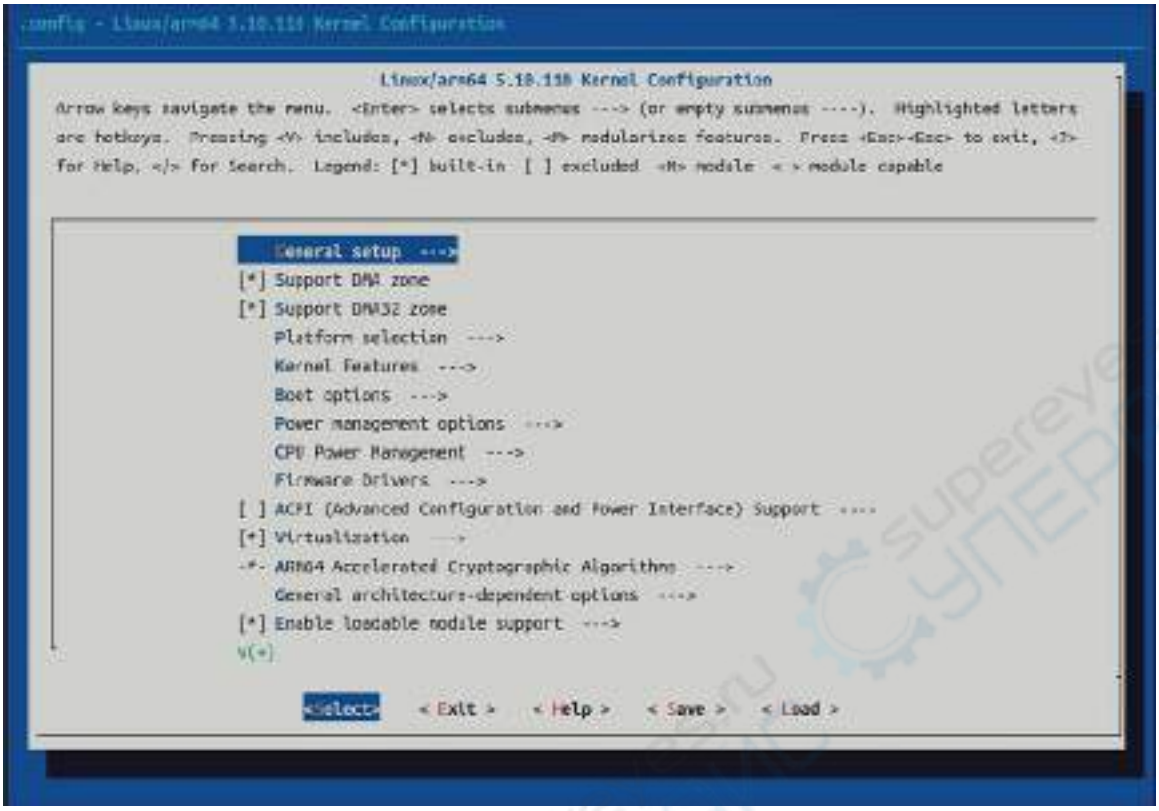
3) 接着选择开发板的型号



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/nconf.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/nconf
scripts/kconfig/nconf  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 [ 5.10.110 ]
```

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=orangepi5b 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

```
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-5.10.110-rockchip-rk3588
│   ├── System.map-5.10.110-rockchip-rk3588
│   └── vmlinuz-5.10.110-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.config
```

b. 开发板使用的 dts 文件所在的位置为

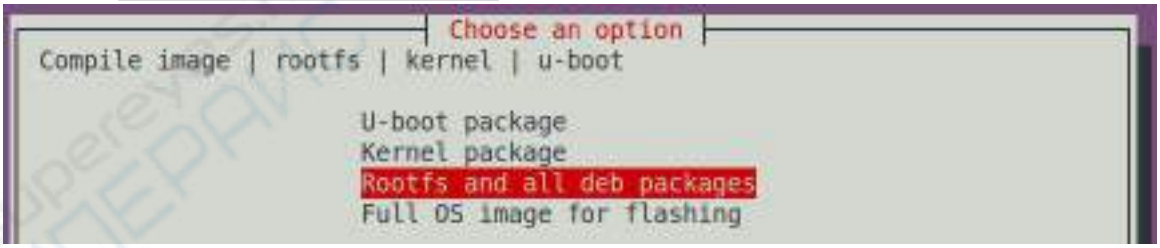
```
orangepi-build/kernel/orange-pi-5.10-rk3588/arch/arm64/boot/dts/rockchip/rk3588s-orangepi-5b.dts
```

6.5. 编译 rootfs

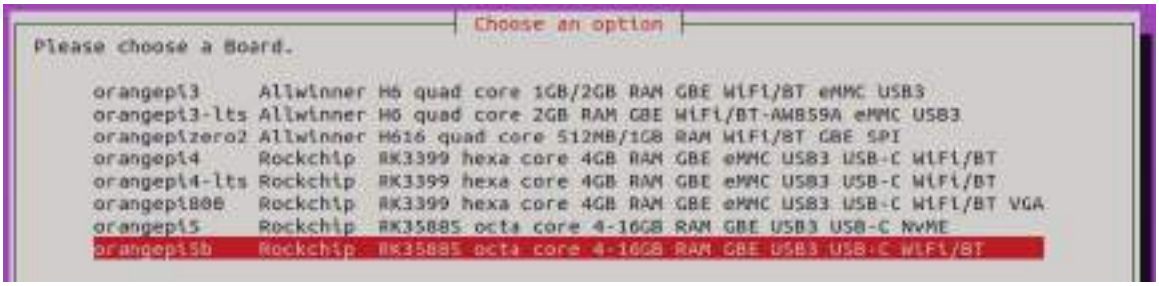
1) 运行 build.sh 脚本，记得加 sudo 权限

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

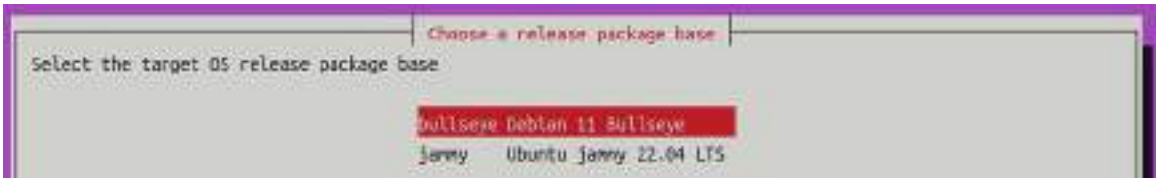
2) 选择 **Rootfs and all deb packages**，然后回车



3) 接着选择开发板的型号

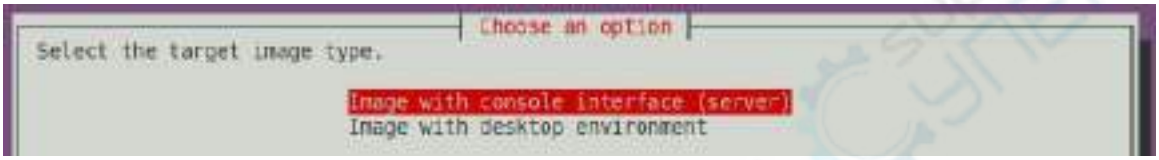


4) 然后选择 rootfs 的类型

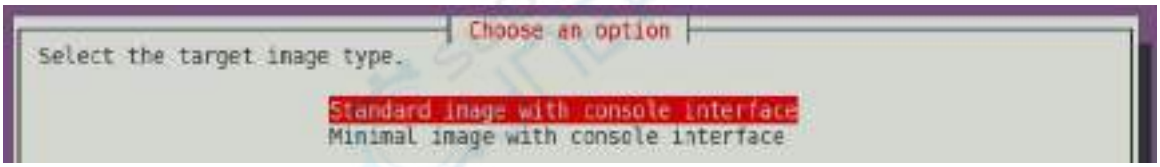


5) 然后选择镜像的类型

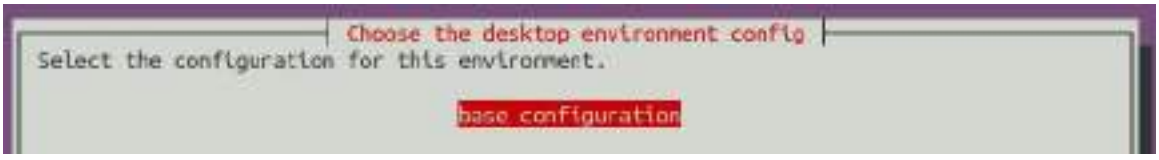
a. **Image with console interface (server)**表示服务器版的镜像，体积比较小
b. **Image with desktop environment** 表示带桌面的镜像，体积比较大



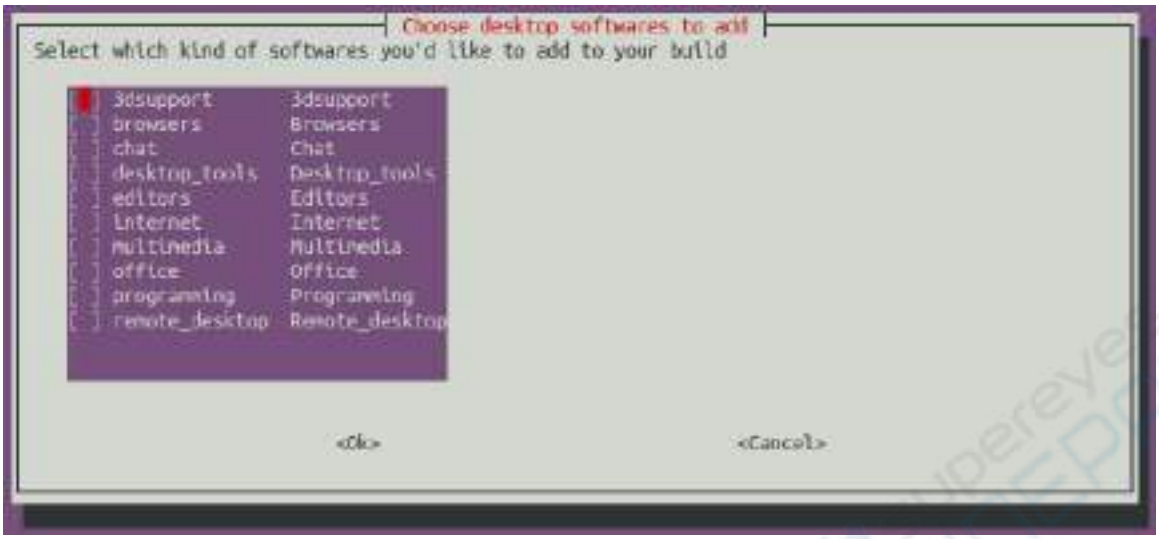
6) 如果是编译服务器版的镜像，还可以选择编译 Standard 版本或者 Minimal 版本，Minimal 版本预装的软件会比 Standard 版本少很多（**没特殊需求请不要选择 Minimal 版本，因为很多东西默认没有预装，部分功能可能用不了**）



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 支持 XFCE 和 Gnome 两种桌面，Ubuntu Focal 和 Debian 只支持 XFCE



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/
bullseye-xfce-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4
bullseye-xfce-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4.current
bullseye-xfce-arm64.5250ec7002de9e81a41de169f1f89721.tar.lz4.list
```

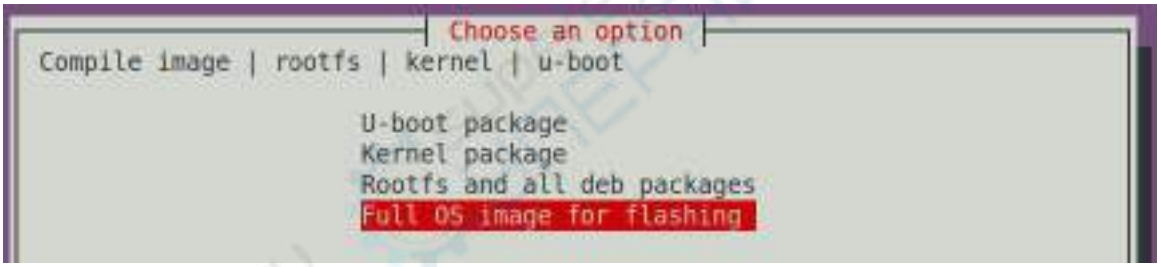
10) 如果需要的 rootfs 在 **external/cache/rootfs** 下已经存在，那么再次编译 rootfs 就会直接跳过编译过程，不会重新开始编译，编译镜像的时候也会去 **external/cache/rootfs** 下查找是否已经有缓存可用的 rootfs，如果有就直接使用，这样可以节省大量的下载编译时间。

6.6. 编译 linux 镜像

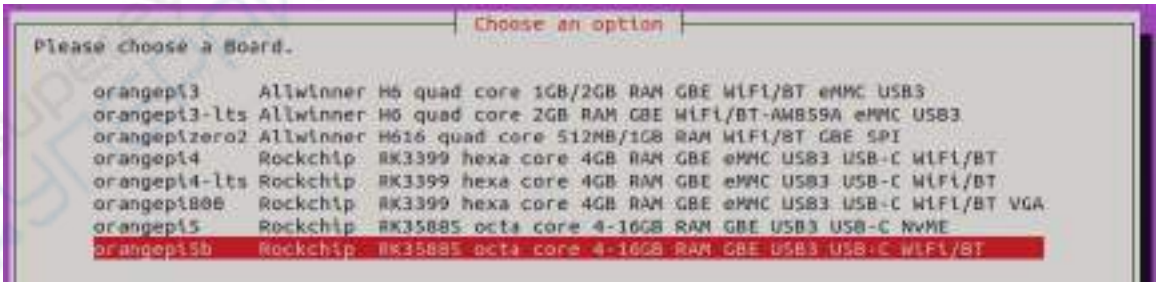
1) 运行 build.sh 脚本，记得加 sudo 权限

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

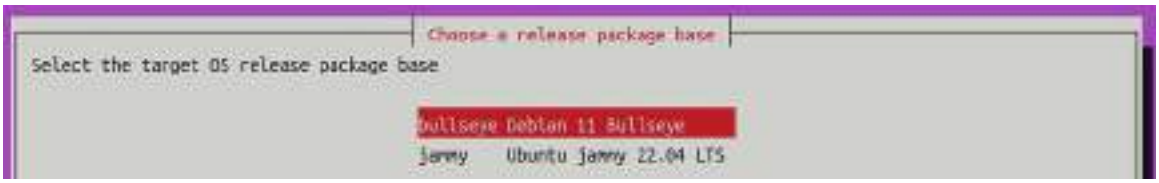
2) 选择 **Full OS image for flashing**，然后回车



3) 然后选择开发板的型号



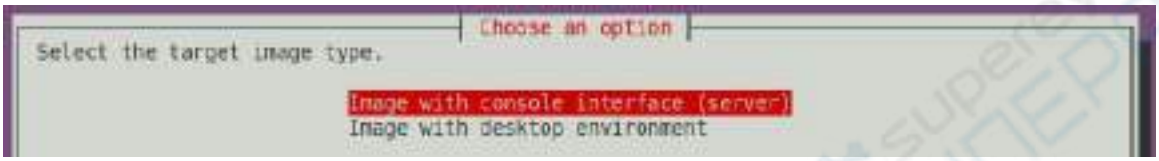
4) 然后选择 rootfs 的类型



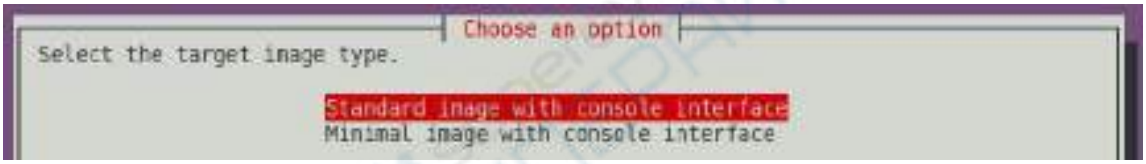
5) 然后选择镜像的类型

a. **Image with console interface (server)**表示服务器版的镜像，体积比较小

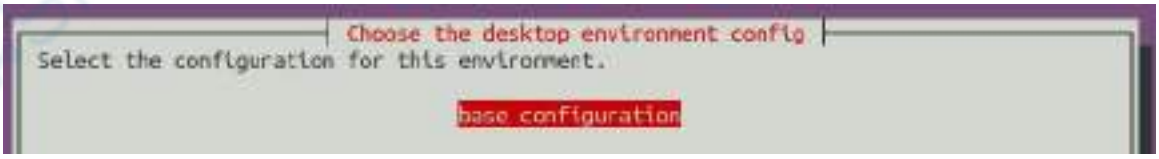
b. **Image with desktop environment** 表示带桌面的镜像，体积比较大



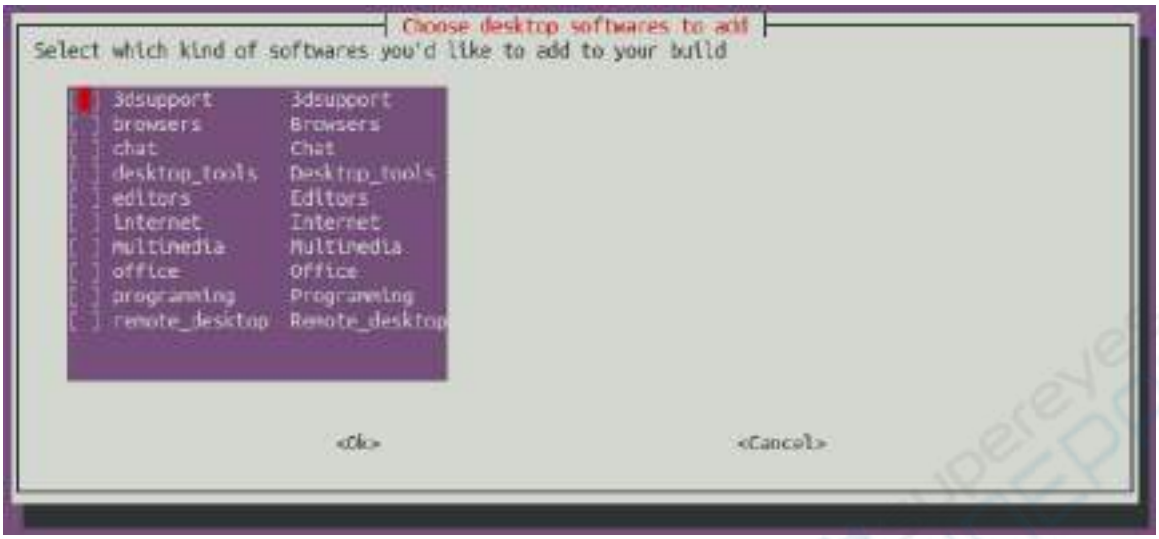
6) 如果是编译服务器版的镜像，还可以选择编译 Standard 版本或者 Minimal 版本，Minimal 版本预装的软件会比 Standard 版本少很多（**没特殊需求请不要选择 Minimal 版本，因为很多东西默认没有预装，部分功能可能用不了**）



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 支持 XFCE 和 Gnome 两种桌面，Ubuntu Focal 和 Debian 只支持 XFCE



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/orangepi5b_1.0.0_debian_bullseye_linux5.10.110_xfce_desktop/ora
```

ngepi5b_1.0.0_debian_bullseye_linux5.10.110_xfce_desktop.img]

b. 编译使用的时间

[o.k.] Runtime [19 min]

c. 重复编译镜像的命令，使用下面的命令无需通过图形界面选择，可以直接开始编译镜像

[o.k.] Repeat Build Options [**sudo ./build.sh BOARD=orangepi5b 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-rk3588 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资料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-rk3588.tar.gz**

orangepi@orangepi:~$ **mv orange-pi-5.10-rk3588 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@orangepi5b:~$ ls /lib/modules
5.10.110+  5.10.110-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@orangepi5b:~/orange-pi-5.10-rk3588$ ls /boot/vmlinuz*
/boot/vmlinuz-5.10.110+  /boot/vmlinuz-5.10.110-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
5.10.110+
```

8. Android 12 系统的使用说明

8.1. 已支持的 Android 版本

Android 版本	内核版本
Android 12	Linux5.10
Android 12 Box	Linux5.10

8.2. Android 功能适配情况

功能	Android 12	Android12 Box
USB2.0x2	OK	OK
USB3.0x1	OK	OK
USB Type-C 3.0	OK	OK
DP 显示	OK	OK
eMMC 启动	OK	OK
AP6275P-WIFI	OK	OK
AP6275P-蓝牙	OK	OK
GPIO（26pin）	OK	OK
UART（26pin）	OK	OK
SPI（26pin）	OK	OK
I2C（26pin）	OK	OK
PWM（26pin）	OK	OK
3pin 调试串口	OK	OK
TF 卡启动	OK	OK
HDMI 视频	OK	OK
HDMI 音频	OK	OK
OV13850 摄像头	OK	OK
OV13855 摄像头	OK	OK
LCD1	OK	NO
LCD2	OK	NO
千兆网口	OK	OK
网口状态灯	OK	OK

MIC	**OK**	**OK**
耳机播放	**OK**	**OK**
耳机录音	**OK**	**OK**
LED 灯	**OK**	**OK**
GPU	**OK**	**OK**
NPU	**OK**	**OK**
VPU	**OK**	**OK**
开关机按键	**OK**	**OK**
HDMI CEC 功能	**NO**	**OK**

8.3. WIFI 的连接测试方法

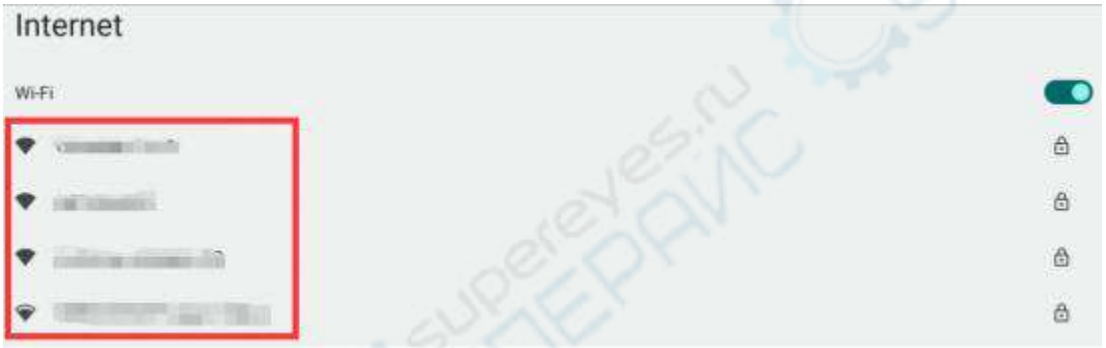
1) 首先点击进入 **Setting**

2) 然后选择 **Network & internet**

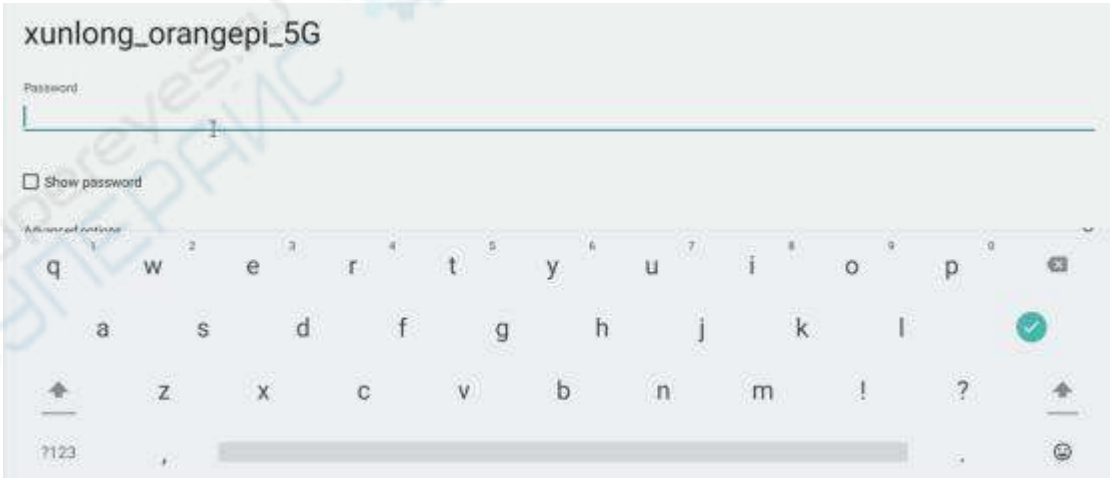
3) 然后选择 **Internet**

4) 然后打开 **Wi-Fi** 开关

5) 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

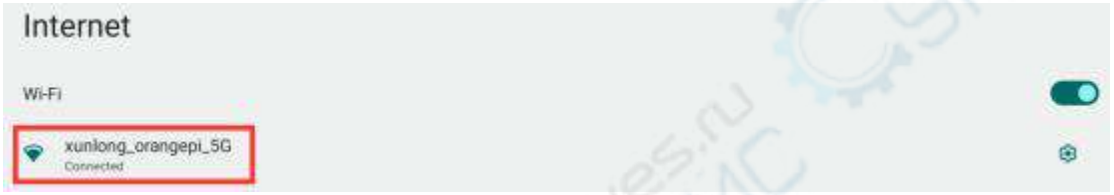


6) 然后选择想连接的 Wi-Fi 后会弹出下图所示的密码输入界面



7) 然后使用键盘输入 Wi-Fi 对应的密码，再使用鼠标点击虚拟键盘中的回车按钮就会开始连接 Wi-Fi 了

8) Wi-Fi 连接成功后的显示如下图所示：



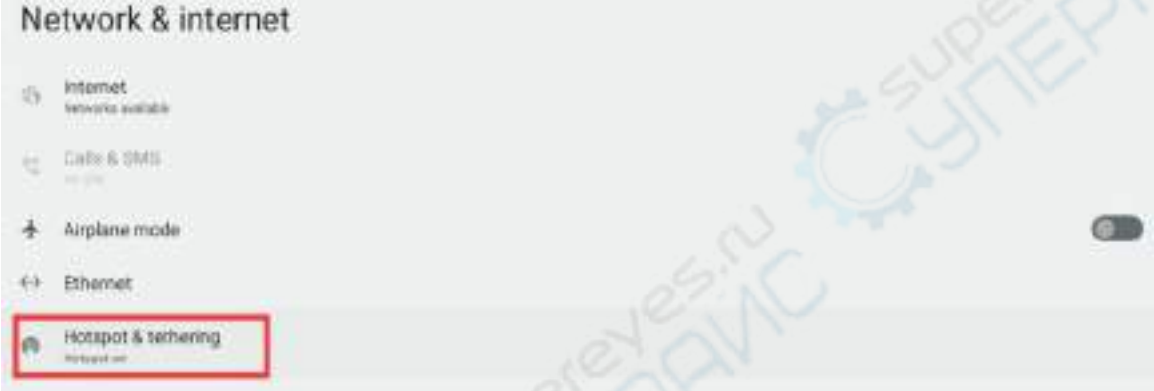
8. 4. Wi-Fi hotspot 的使用方法

1) 首先请确保以太网口已连接网线，并且能正常上网

2) 然后选择 **Settings**

3) 然后选择 **Network & internet**

4) 然后选择 **Hotspot & tethering**



5) 然后选择 **Wi-Fi hotspot**

6) 然后打开 **Wi-Fi hotspot**，下图中还可以看到生成的热点的名字和密码，记住它们，在连接热点的时候要用到（如果需要修改热点的名字和密码，需要先关闭 Wi-Fi hotspot，然后才能修改）

7）此时可以拿出你的手机，如果一切正常，在手机搜索到的 WI-FI 列表中就能找到上图 **Hotspot name** 下面显示的同名（**这里为 AndroidAP_6953**）的 WIFI 热点了。然后可以点击 **AndroidAP_6953** 连接热点，密码在上图的 **Hotspot password** 下面可以看到

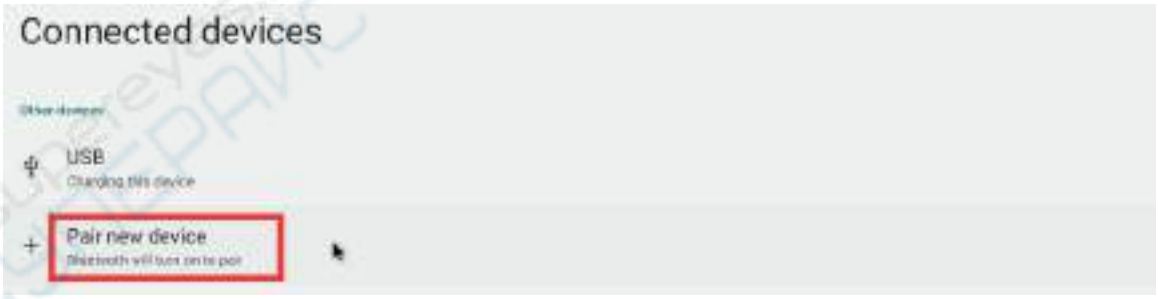
8) 连接成功后显示如下图所示（不同手机界面会有区别，具体界面以你手机显示的为准）。此时就可以在手机上打开一个网页看下能否上网了，如果能正常打开网页，说明开发板的 **WI-FI Hotspot** 能正常使用

8.5. 蓝牙的测试方法

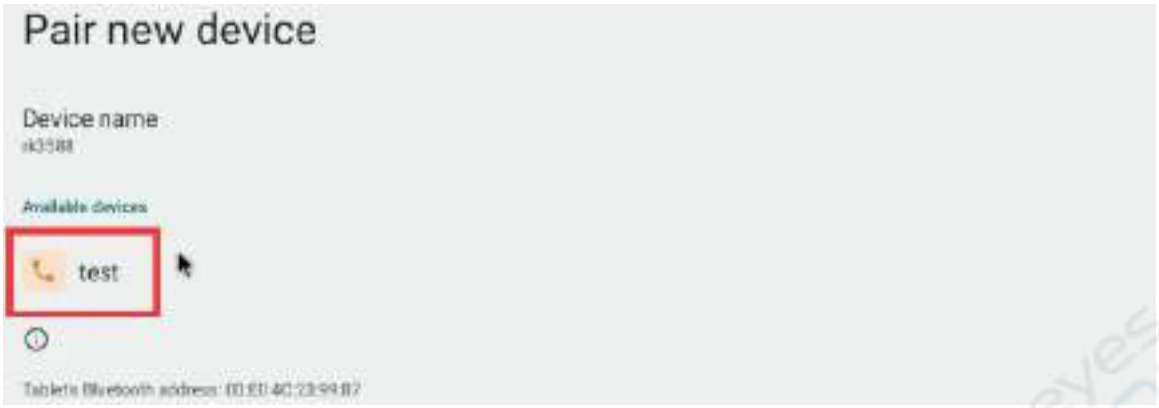
1) 首先点击进入 **Setting**

2) 然后选择 **Connected devices**

3) 然后点击 **Pair new device** 打开蓝牙并开始扫描周围的蓝牙设备



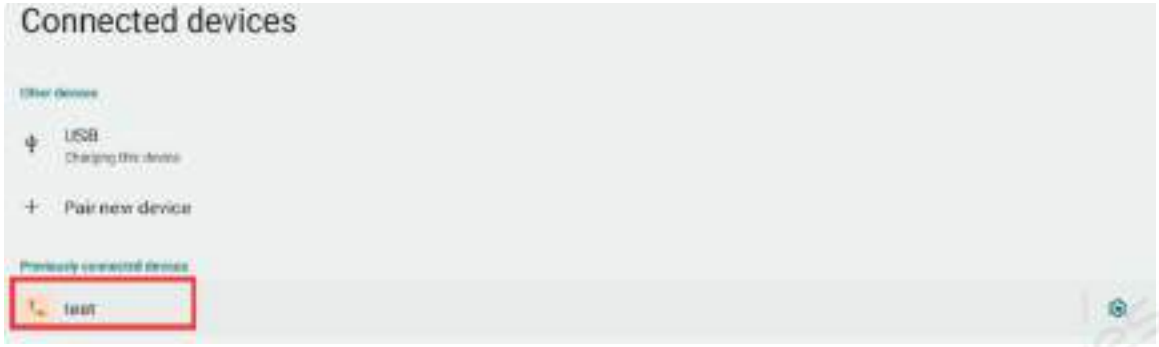
4) 搜索到的蓝牙设备会在 **Available devices** 下面显示出来



5）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

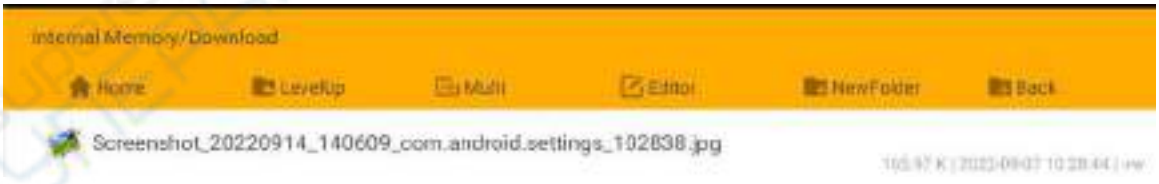
6) 这里测试的是开发板和安卓手机蓝牙的配置过程，此时在手机上会弹出下面的确认界面，在手机上也点击配对按钮后就会开始配对过程

7) 配对完成后，可以看到如下图所示的已配对的蓝牙设备



8) 此时可以使用手机蓝牙给开发板发送一张图片，发送后，在开发板的安卓系统中可以看到下面的确认界面，然后点击 **Accept** 就可以开始接收手机发过来的图片了

9) 开发板 Android 系统蓝牙接收到的图片可以在文件管理器中打开 **Download** 目录查看



8.6. 10.1 寸 MIPI 屏幕的使用方法

请确保使用的 Android 镜像为下面的版本的镜像： orangepi5b_RK3588S_Android12_lcd_v1.x.x.img

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) 开发板上有两个 mipi lcd 屏幕的接口，我们定义：

c. lcd1 接口的位置为：

d. lcd2 接口的位置为：

3) 将组装好的屏幕接到 lcd1 或者 lcd2 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

8.7. 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

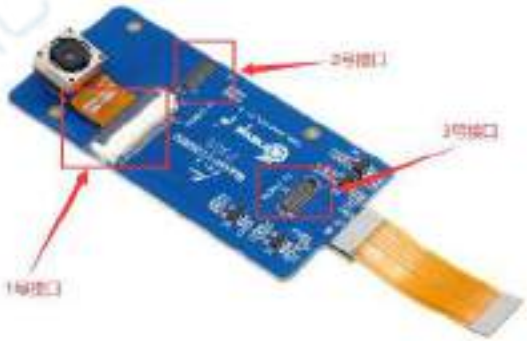
OV13850 和OV13855 摄像头使用的转接板和FPC排线是一样的，只是两款摄像头接在转接板上的位置不一样。FPC排线如下图所示，请注意FPC排线是有方向的，标注**TO MB**那端需要插到开发板的摄像头接口中，标注**TO CAMERA**那端需要插到摄像头转接板上。

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**

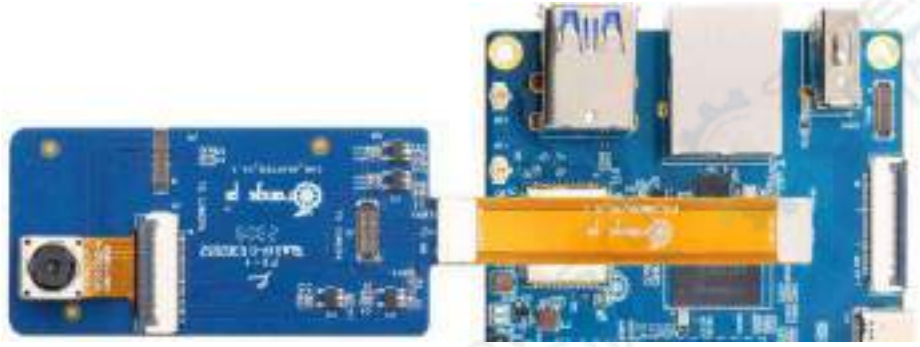
b. **2 号接口接 OV13855 摄像头**

c. 3 号接口未使用，忽略即可



Orange Pi 5B 开发板上总共有 3 个摄像头接口，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：



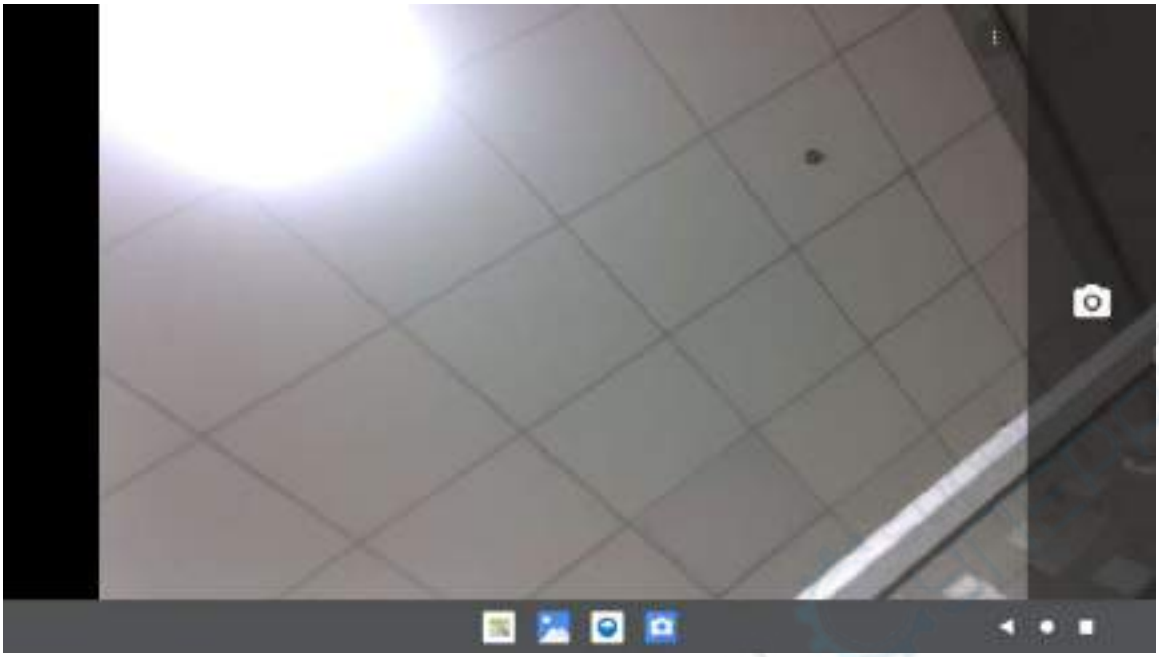
摄像头插在开发板的 Cam2 接口的方法如下所示：

摄像头插在开发板的 Cam3 接口的方法如下所示：

安卓系统默认打开的是 **Cam1** 和 **Cam3** 的配置，所以如果要使用摄像头，请选择 **Cam1** 和 **Cam3** 接口中的一个。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. 在桌面中打开相机 APP

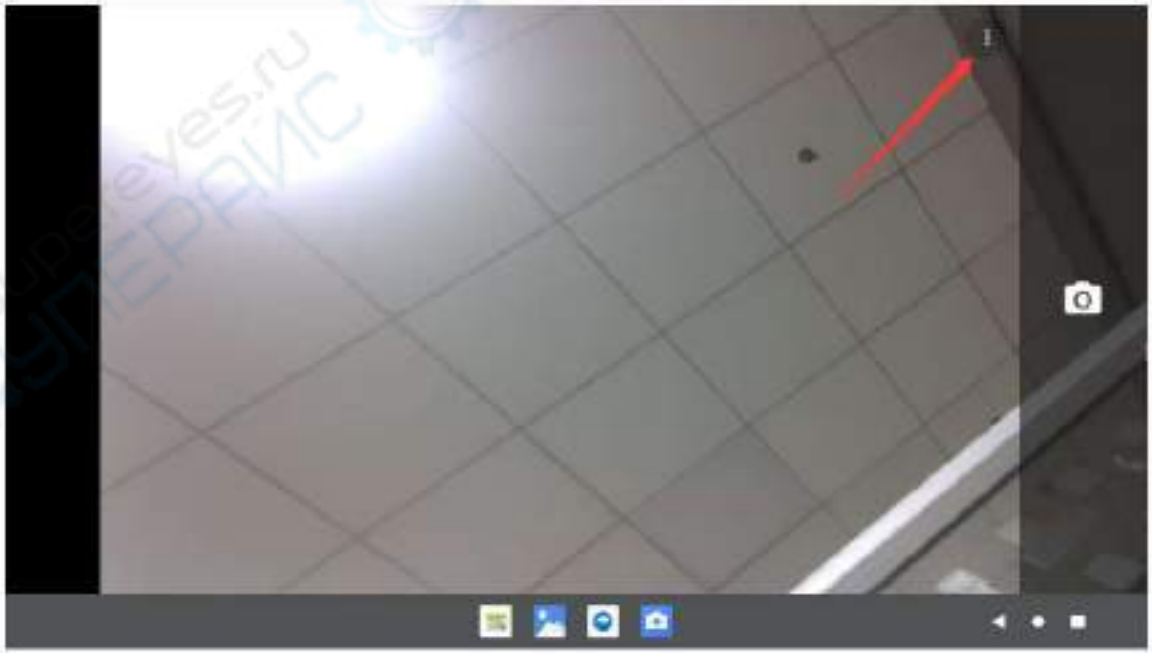
b. 然后就能看到摄像头的预览画面了



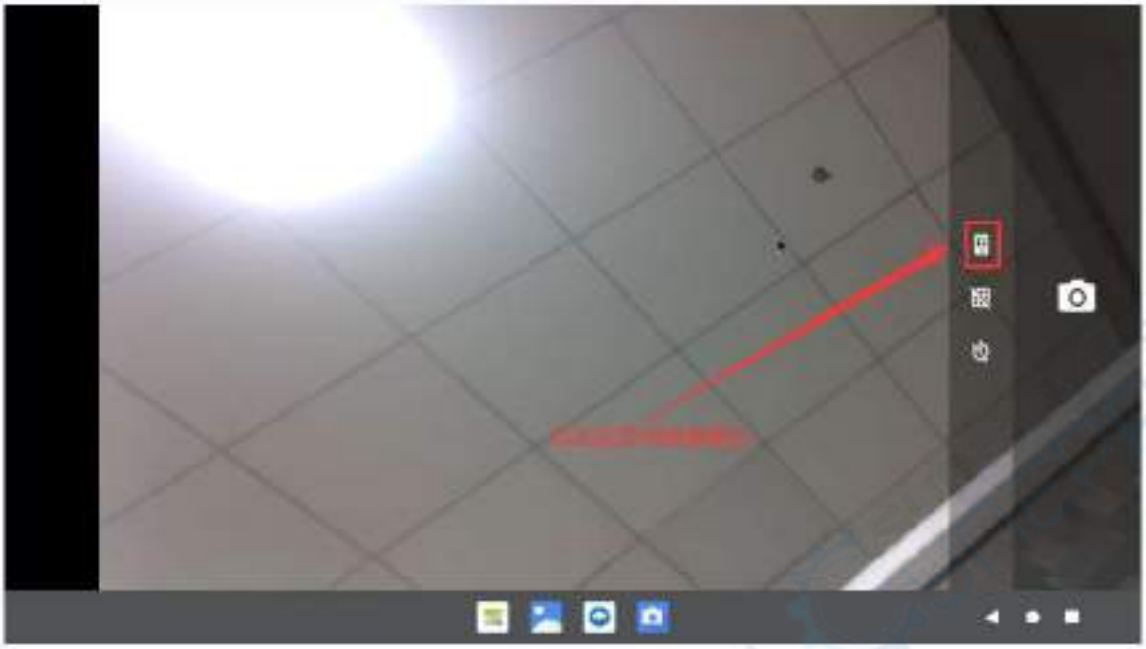
除了单摄外， 我们还可以同时接两个摄像头。需要注意的是，目前测试双摄像头请使用 **Cam1+Cam3** 的组合（支持 ov13850 和 ov13855 混搭）。接好双摄后，然后和前面步骤一样，打开摄像头 APP 后即可看到其中一个摄像头的画面。

切换另一个摄像头的方法为：

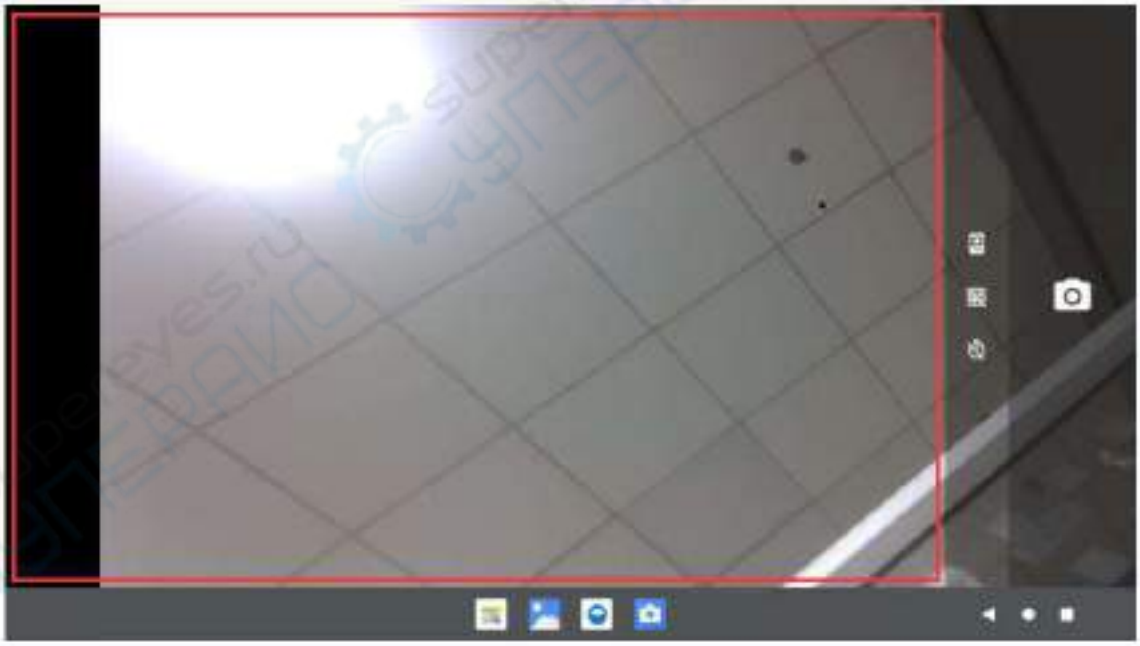
a. 首先点击右上角的这三个点



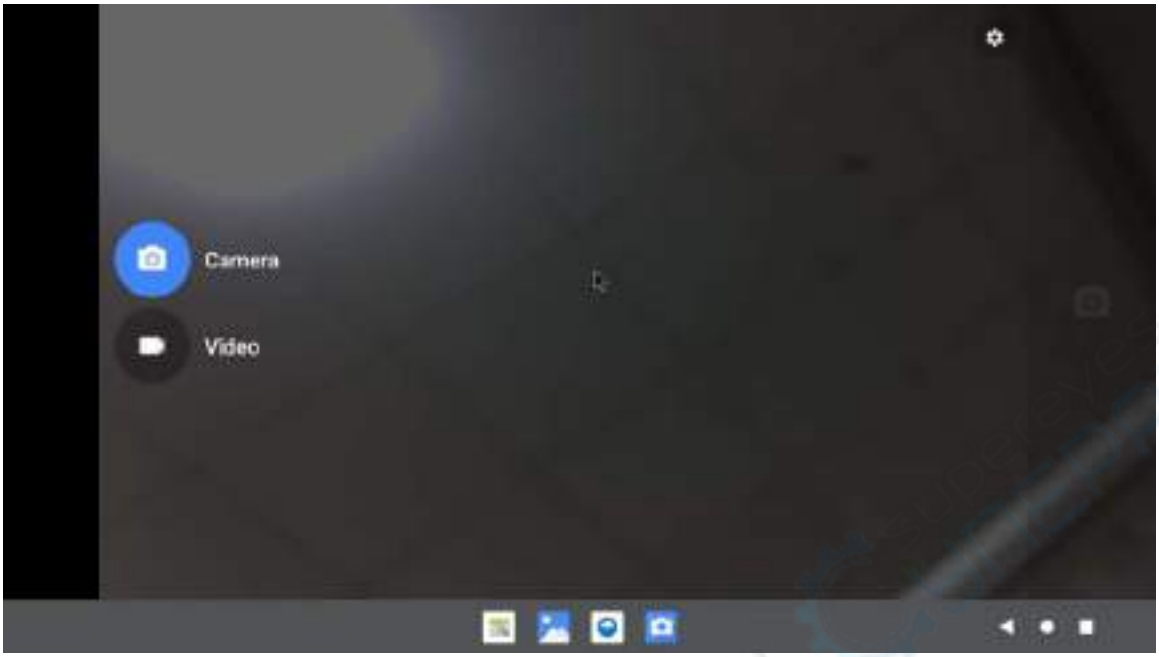
b. 然后点击下图所示的位置即可切换摄像头



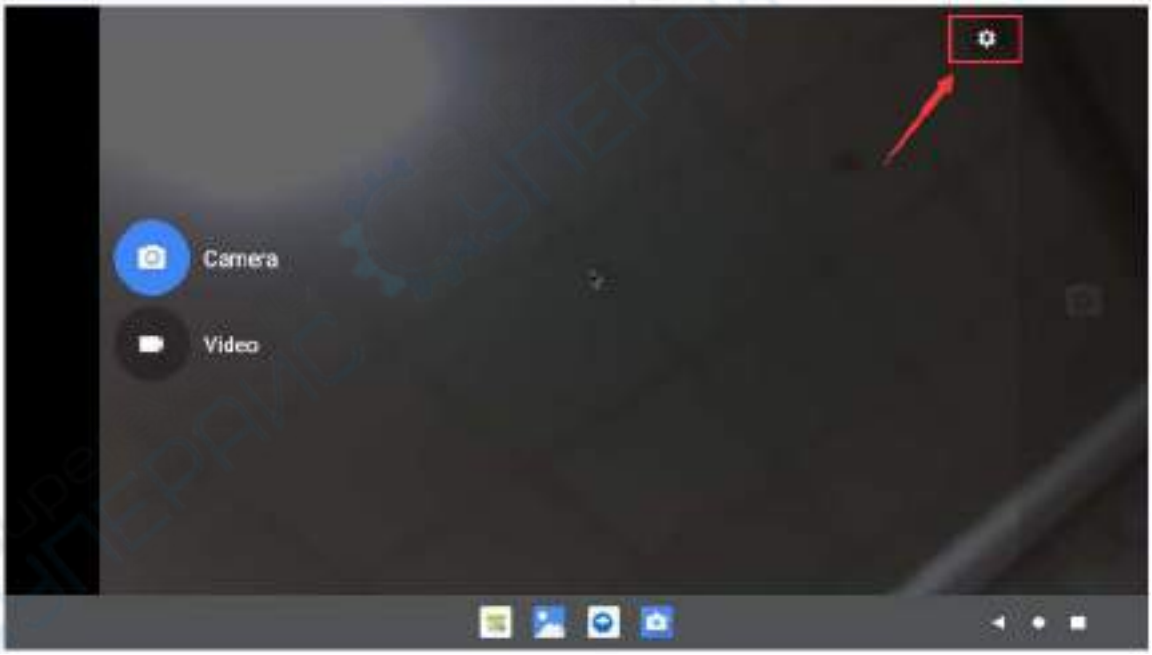
在摄像头 APP 下图红框所示的区域中按住鼠标然后向右拖动可以调出拍照和摄像的切换界面



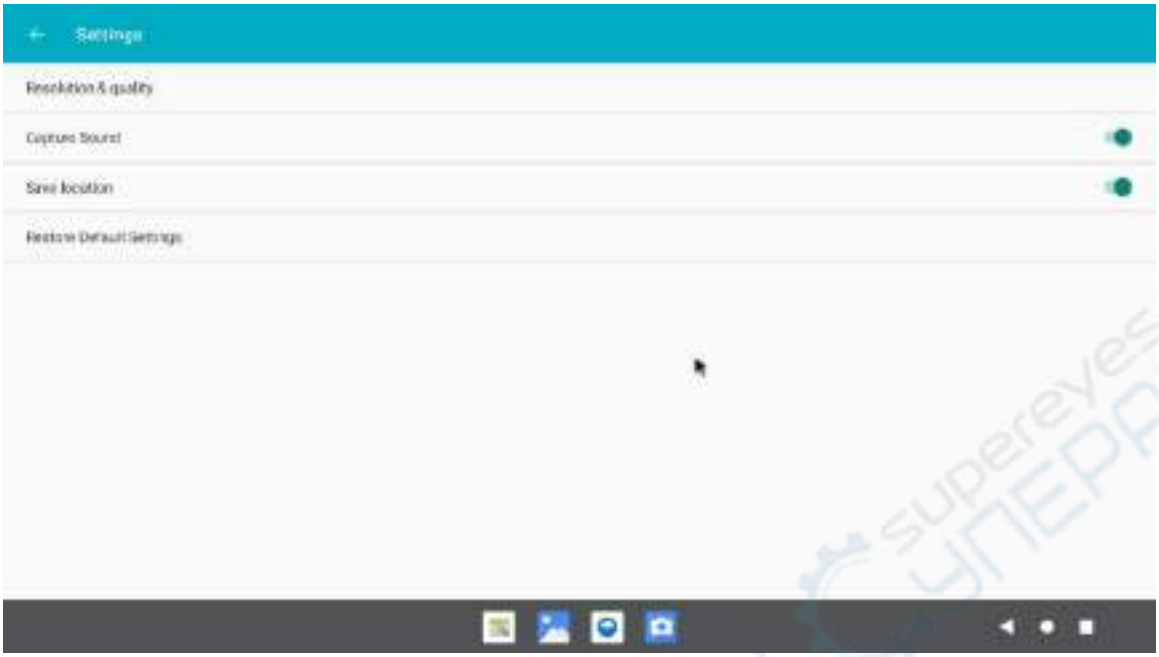
拍照和摄像的切换界面如下所示，点击 **Video** 即可切换到录像模式



点击下图所示的位置可以进入摄像头的设置界面



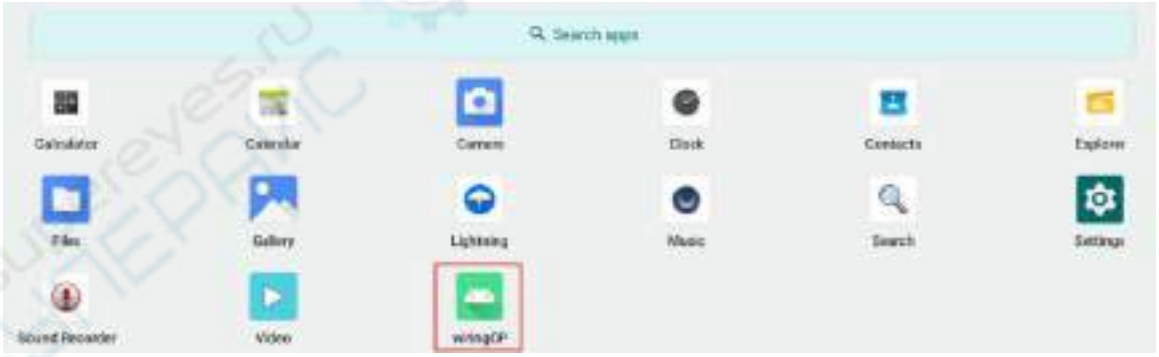
摄像头的设置界面如下所示：



8.8. 26pin 接口 GPIO、UART、SPI 和 PWM 测试

8.8.1. 26pin GPIO 口测试

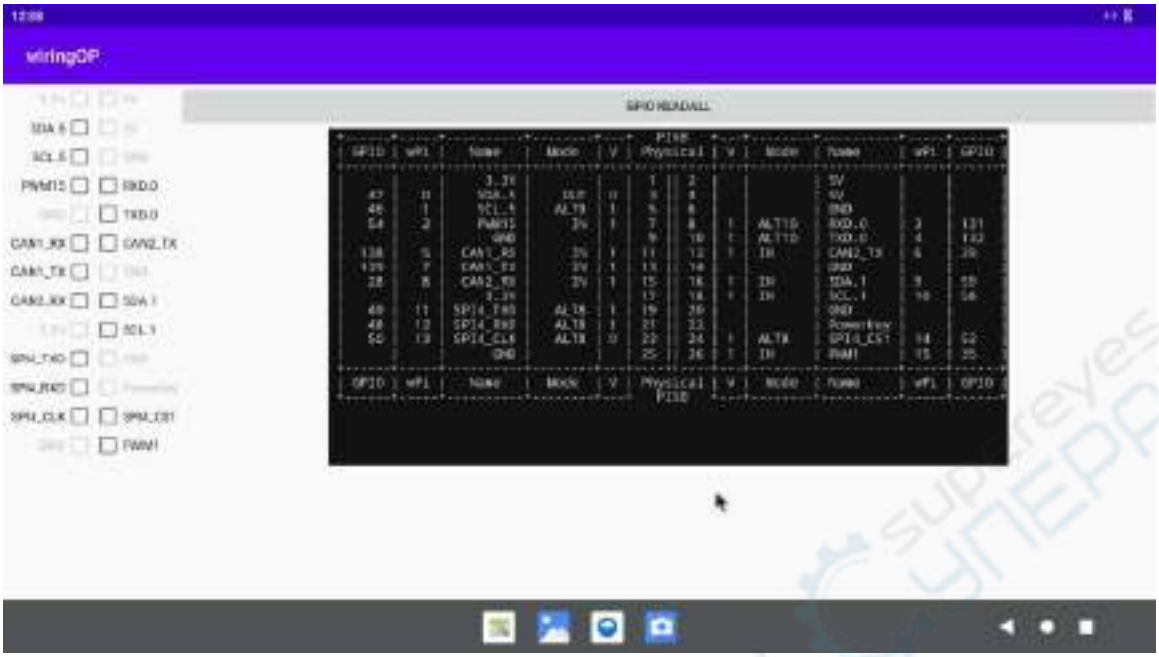
1) 首先点击 wiringOP 图标打开 wiringOP APP



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

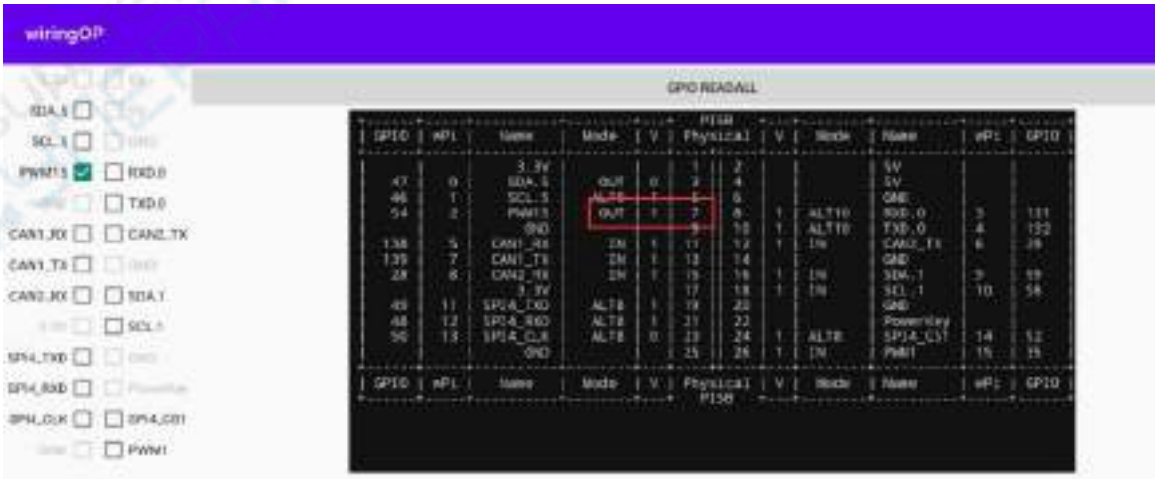
3) GPIO 测试界面如下图所示，左边的两排 **CheckBox** 按钮跟 26pin 引脚是一一对应的关系。当勾选 **CheckBox** 按钮时，对应的 GPIO 引脚会被设置为 **OUT** 模式，引脚电平设置为高电平；当取消勾选时，GPIO 引脚电平设置为低电平；当点击右边的 **GPIO READALL** 按钮时，可以获取到 wPi 号、GPIO 模式、引脚电平等信息。

4) 然后点击 **GPIO READALL** 按钮，输出信息如下图所示：

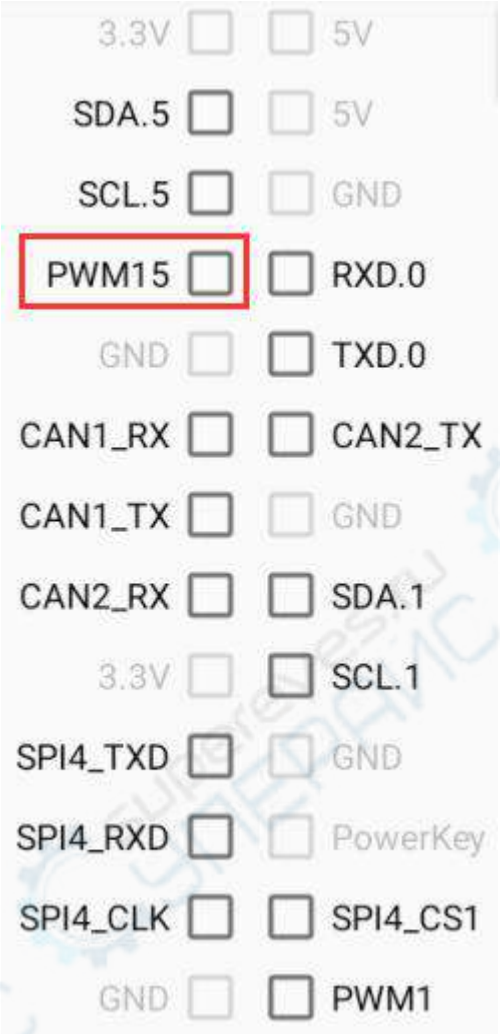


5) 开发板 26pin 中总共有 16 个 GPIO 口可以使用，下面以 7 号引脚——对应 GPIO 为 GPIO1_C6 ——对应 wPi 序号为 2——为例演示如何设置 GPIO 口的高低电平。首先点击 7 号引脚对应的 **CheckBox** 按钮，当按钮为选中状态时，7 号引脚会设置为高电平，设置完后可以使用万用表测量引脚的电压的数值，如果为 **3.3v**，说明设置高电平成功

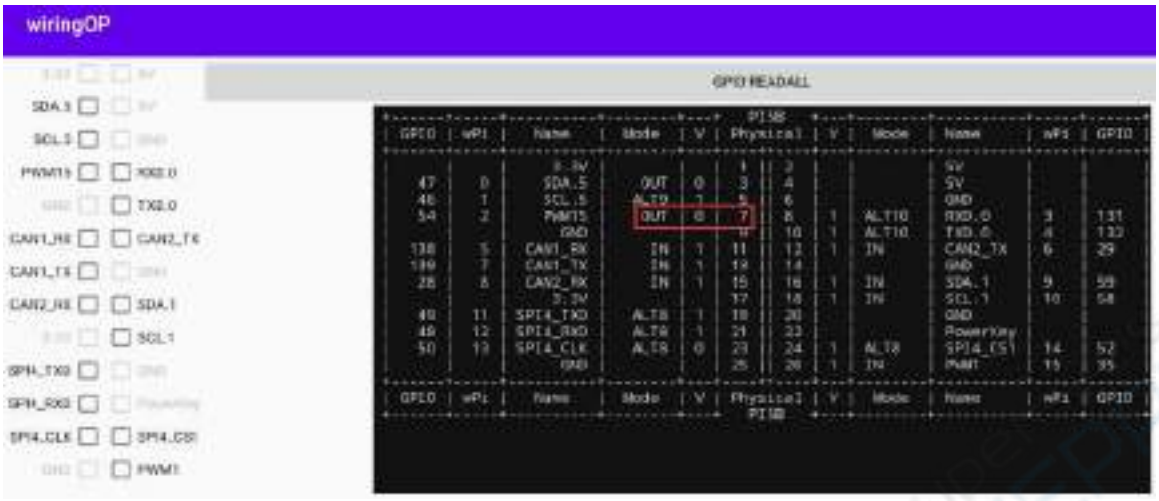
6) 然后点击 **GPIO READALL** 按钮，可以看到当前的 7 号引脚模式为 **OUT**，引脚电平为高电平



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



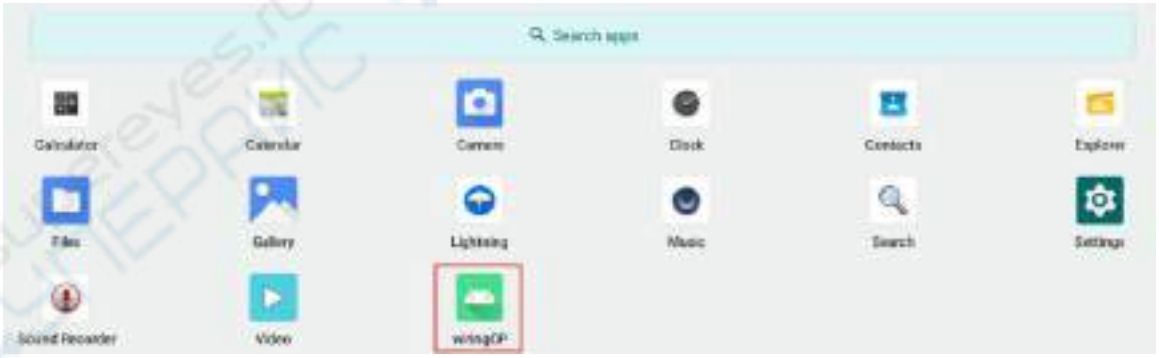
8) 然后点击 **GPIO READALL** 按钮，可以看到当前的 7 号引脚模式为 OUT，引脚电平为低电平



8.8.2. 26pin 的 UART 测试

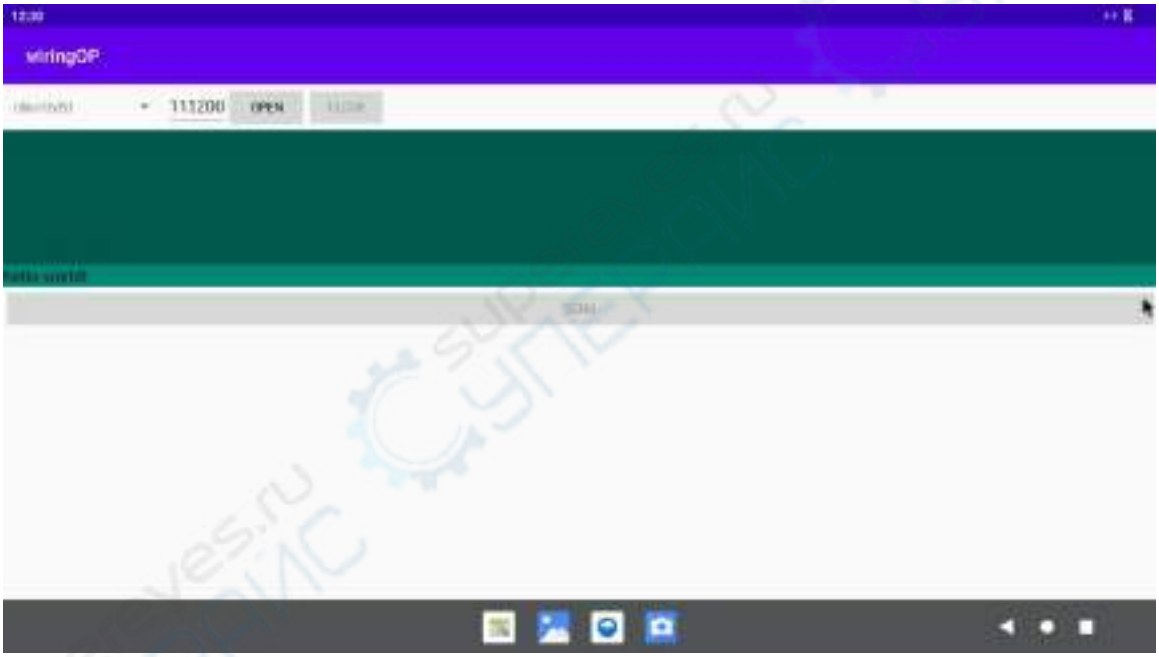
1) Android 中默认只打开了 UART0 一个串口，UART0 在 26pin 的位置如下图所示，对应的设备节点是**/dev/ttyS0**

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

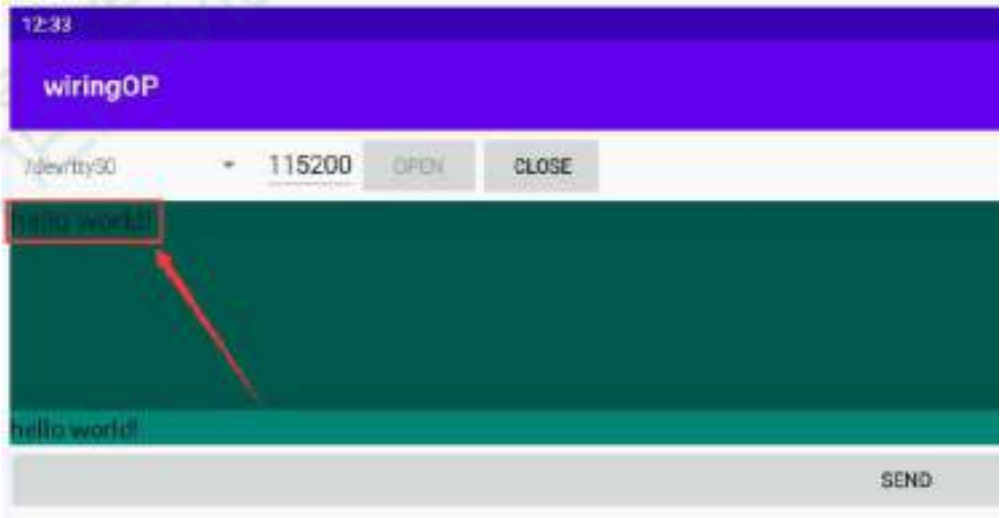


5) 接着在编辑框中输入想要设置的波特率，然后点击 **OPEN** 按钮打开**/dev/ttyS0** 节点，打开成功后，**OPEN** 按钮变为不可选中状态，**CLOSE** 按钮和 **SEND** 按钮变为可选中状态

6) 然后使用杜邦线短接 uart0 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

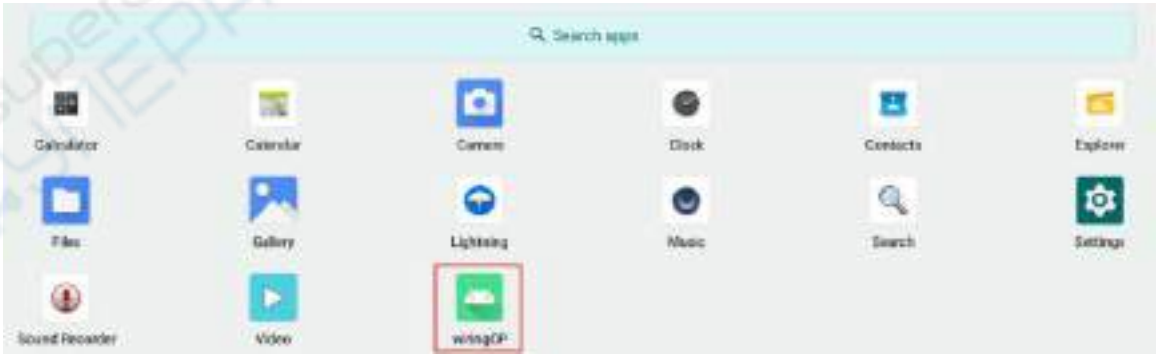


8.8.3. 26pin 的 SPI 测试

1) 由 26pin 接口的原理图可知，Orange Pi 5B 可用的 spi 为 spi4

2) 这里通过 w25q64 模块来测试 SPI 接口，首先在 SPI4 接口接入 w25q64 设备

3) 然后点击 wiringOP 图标打开 wiringOP APP

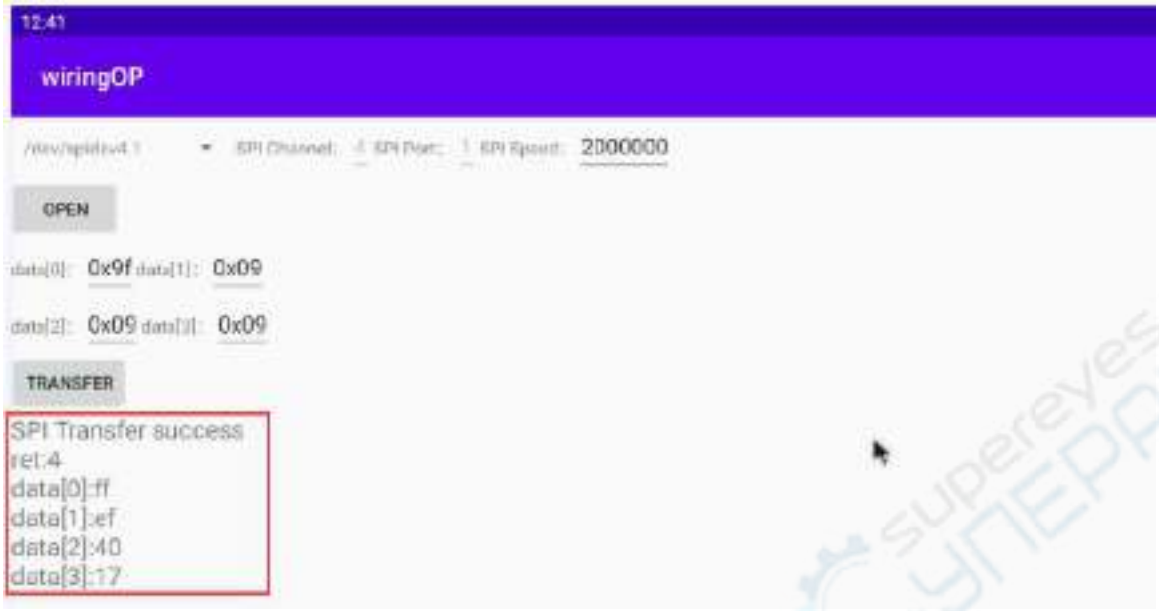


4) wiringOP APP 的主界面显示如下图所示，点击 SPI_TEST 按钮打开 SPI 的测试界面

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



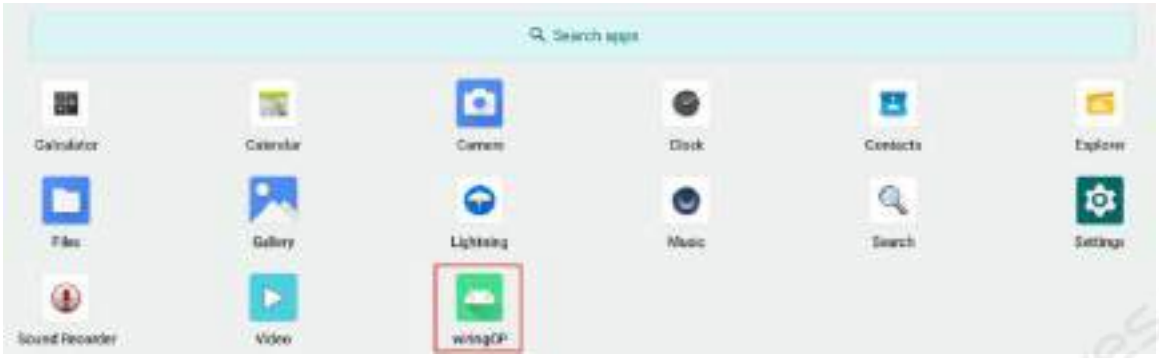
8) w25q64 模块的 MANUFACTURER ID 为 EFh，Device ID 为 4017h，跟上面读取到的值是对应的（h 代表是 16 进制）

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. 8. 4. 26pin 的 PWM 测试

1) Android 默认只开启了 **PWM15**，对应的引脚在 26pin 的所在位置如下图所示

2) 首先点击 wiringOP 图标打开 wiringOP APP



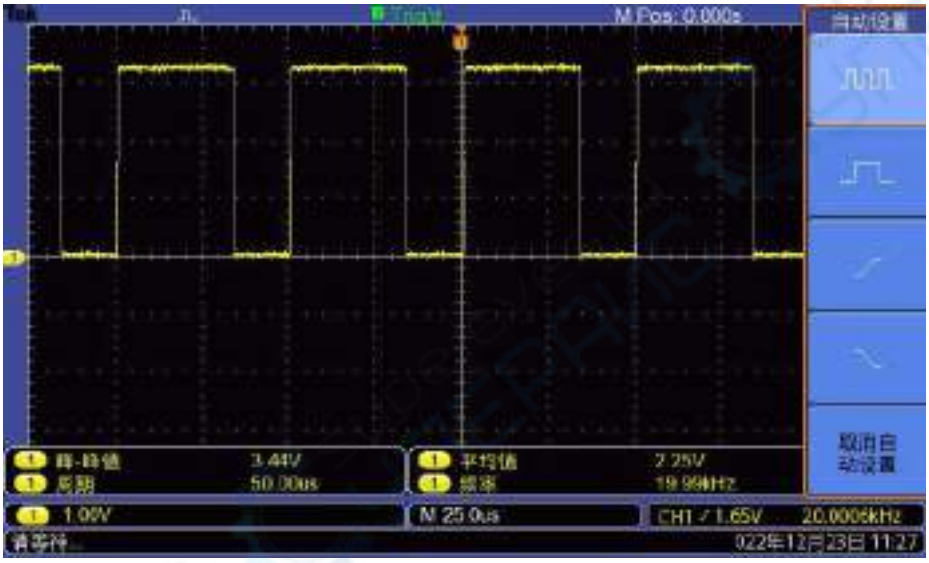
3) 然后在 wiringOP 的主界面点击 **PWM_TEST** 按钮进入 PWM 的测试界面

4) PWM15 对应的基地址是 **febf0030**，这里 pwmchip0 右边显示的刚好就是 **febf0030.pwm**， 如果显示的基地址不对，请点击下拉选项选择其它的 pwmchip，直到右边显示 **febf0030** 为止

5) 然后确认 PWM 的周期，默认的配置是 **50000ns**，转换为 PWM 频率是 **20KHz**，可自行修改，点击开启按钮导出 **PWM15**

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 26pin 中的第 7 号引脚就可以看到下面的波形了



8.9. ADB 的使用方法

8.9.1. 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) 然后使用 Type-C 数据线将开发板连接到电脑的 USB 接口中（请同时使用 TypeC 电源给开发板供电)

3) 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) 执行命令重新挂载 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/
```

8.9.2. 使用网络连接 adb 调试

使用网络 adb 无需 USB Typc C 接口的数据线来连接电脑和开发板，而是通过网络来通信，所以首先请确保开发板的有线或者无线网络已经连接好了，然后获取开发板的 IP 地址，后面要用到。

1) 确保 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:/ #
```

8.10. Android Box 测试过的 2.4G USB 遥控器

1) 目前测试过的一款 2.4G USB 遥控器如下图所示

a. 包含一个遥控器

b. 一个 USB 无线接收器

2) Android Box 系统无需任何配置，插上就可以用了

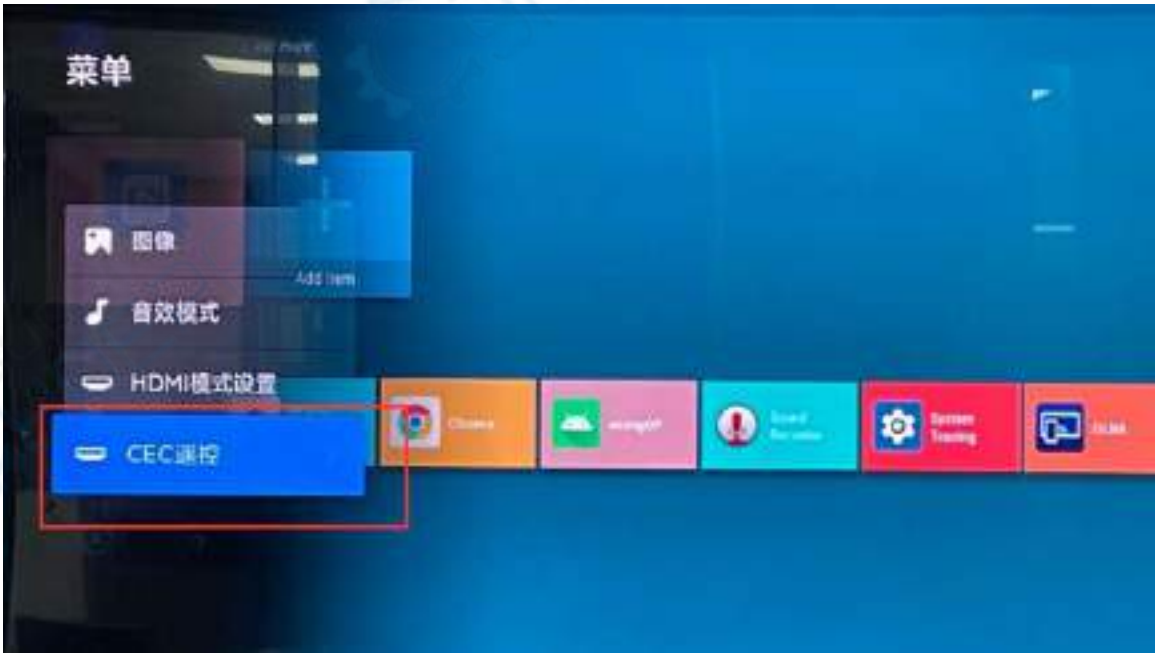
8.11. Android Box 系统 HDMI CEC 功能的使用方法

HDMI CEC 允许用户只用一个遥控器，就能通过 HDMI 控制所有连接的设备，基于这个功能，我们用电视机的遥控器就可以控制开发板。

测试此功能前，请确保您的电视机是支持 HDMI CEC 的。

1) 首先将开发板通过 HDMI 线连接到电视，然后上电启动

2) 然后在电视的设置中开启 HDMI CEC 功能，不同电视开启方式可能有差异，这里以小米电视为例，按下遥控器的菜单键，然后选中 CEC 遥控并按下确认键



3) 然后选择“开”就可以打开 HDMI CEC 遥控

4) 此时就可以通过电视的遥控器控制开发板的 Android Box 系统了

9. Android 12 源码的编译方法

9.1. 下载 Android 12 的源码

1) 首先从百度云盘或者谷歌网盘下载 Android 12 源码的分卷压缩包

a. 百度云盘

b. 谷歌网盘

2) Android 12 源码的分卷压缩包下载完后，请先检查下 MD5 校验和是否正确，如果不正确，请重新下载源码

```
test@test:~$ md5sum -c Android_12.tar.gz.md5sum
Android_12.tar.gz00: 确定
Android_12.tar.gz01: 确定
Android_12.tar.gz02: 确定
Android_12.tar.gz03: 确定
Android_12.tar.gz04: 确定
Android_12.tar.gz05: 确定
Android_12.tar.gz06: 确定
Android_12.tar.gz07: 确定
```

3) 然后需要将多个压缩文件合并成一个，再进行解压

```
test@test:~$ cat Android_12.tar.gz0* > Android_12.tar.gz
test@test:~$ tar -xvf Android_12.tar.gz
```

9.2. 编译 Android 12 的源码

1) 首先安装编译 Android12 源码需要的软件包

```
test@test:~$ sudo apt-get update
test@test:~$ sudo apt-get install -y git gnupg flex bison gperf build-essential \
zip curl zlib1g-dev gcc-multilib g++-multilib libc6-dev-i386 \
lib32ncurses5-dev x11proto-core-dev libx11-dev lib32z1-dev ccache \
libgl1-mesa-dev libxml2-utils xsltproc unzip
test@test:~$ sudo apt-get install -y u-boot-tools
```

2) 源码中有 build.sh 编译脚本，编译参数如下

c. **-U**：编译 uboot
d. **-K**：编译 kernel
e. **-A**：编译 android
f. **-u**：打包生成 update.img 与 update_spi_nvme.img
g. **-o**：编译 OTA 包
h. **-d**：指定 kernel dts

3) 编译 uboot、kernel、android 并打包成 update.img

a. 编译支持 HDMI 8K 显示镜像（默认关闭 LCD）的命令如下所示：

```
test@test:~$ cd Android_12
test@test:~/ Android_12$ export BOARD=orangepi5b
test@test:~/ Android_12$ source build/envsetup.sh
test@test:~/ Android_12$ lunch rk3588s_s-userdebug
test@test:~/ Android_12$ ./build.sh -AUKu
```

b. 编译支持 LCD 显示镜像的命令如下所示：

```
test@test:~$ cd Android_12
test@test:~/ Android_12$ export BOARD=orangepi5
test@test:~/ Android_12$ export DUAL_LCD=true
```

```
test@test:~/ Android_12$ source build/envsetup.sh
test@test:~/ Android_12$ lunch rk3588s_s-userdebug
test@test:~/ Android_12$ ./build.sh -AUKu
```

4) 编译完成后会打印下面的信息

```
********rkImageMaker ver 2.1********
Generating new image, please wait...
Writing head info...
Writing boot file...
Writing firmware...
Generating MD5 data...
MD5 data generated successfully!
New image generated successfully!
Making update.img OK.
Make update image ok!
```

5) 最终生成的镜像文件会放在 **rockdev/Image-rk3588s_s** 目录下。其中 **update.img** 就是用来烧录的安卓镜像文件

```
test@test:~/Android_12$ cd rockdev/Image-rk3588s_s
test@test:~/Android_12/rockdev/Image-rk3588s_s $ ls update.img
update.img
```

10. 附录

10.1. 用户手册更新历史

版本	日期	更新说明
v1.0	2023-04-12	初始版本
v1.1	2023-04-21	1. Orange Pi OS Arch 系统适配情况 2. OPi OS Arch: 10.1 寸 MIPI LCD 屏幕的使用方法 3. OPi OS Arch: AP6275P PCIe WIFI6+蓝牙模块的使用方法 4. OPi OS Arch: OV13850 和 OV13855 MIPI 摄像头的测试方法 5. OPi OS Arch: 安装 wiringOP 的方法 6. OPi OS Arch: 26pin GPIO、I2C、UART、SPI、CAN 和 PWM 测试 7. 交换 LCD1 和 LCD2 的位置说明，和开发板上的丝印保持一致 8. Linux: 26pin GPIO 口上下拉电阻的设置方法 9. Linux: 26pin 中 CAN 总线的使用方法 10. OPi OS Arch: 设置中文环境以及安装中文输入法的方法

10.2. 镜像更新历史

日期	更新说明
2023-04-12	Orangepi5b_1.0.0_debian_bullseye_server_linux5.10.110.7z Orangepi5b_1.0.0_debian_bullseye_desktop_xfce_linux5.10.110.7z Orangepi5b_1.0.0_debian_bullseye_desktop_kde-plasma_linux5.10.110.7z Orangepi5b_1.0.0_ubuntu_focal_server_linux5.10.110.7z Orangepi5b_1.0.0_ubuntu_focal_desktop_xfce_linux5.10.110.7z Orangepi5b_1.0.0_ubuntu_jammy_server_linux5.10.110.7z Orangepi5b_1.0.0_ubuntu_jammy_desktop_xfce_linux5.10.110.7z Orangepi5b_1.0.0_ubuntu_jammy_desktop_gnome_linux5.10.110.7z OrangePi5B_RK3588S_Android12_v1.0.0.tar.gz OrangePi5B_RK3588S_Android12_lcd_v1.0.0.tar.gz OrangePi5B_RK3588S_Android12-box_v1.0.0.tar.gz

	Opios-droid-aarch64-opi5b-23.04-linux5.10.110.tar.gz Opios-droid-aarch64-opi5b-23.04-linux5.10.110-en.tar.gz * 初始版本
2023-04-14	Orangepi5b_1.0.2_debian_bullseye_server_linux5.10.110.7z Orangepi5b_1.0.2_debian_bullseye_desktop_xfce_linux5.10.110.7z Orangepi5b_1.0.2_debian_bullseye_desktop_kde-plasma_linux5.10.110.7z Orangepi5b_1.0.2_ubuntu_focal_server_linux5.10.110.7z Orangepi5b_1.0.2_ubuntu_focal_desktop_xfce_linux5.10.110.7z Orangepi5b_1.0.2_ubuntu_jammy_server_linux5.10.110.7z Orangepi5b_1.0.2_ubuntu_jammy_desktop_xfce_linux5.10.110.7z Orangepi5b_1.0.2_ubuntu_jammy_desktop_gnome_linux5.10.110.7z * 修复当 eMMC 中烧录安卓或者 Orange Pi OS (Droid)系统时，TF 卡中的 Linux 系统无法启动的问题
2023-04-21	Opios-arch-aarch64-gnome-opi5b-23.04-linux5.10.110.img.xz * 初始版本 Orangepi5b_1.0.4_debian_bullseye_server_linux5.10.110.7z Orangepi5b_1.0.4_debian_bullseye_desktop_xfce_linux5.10.110.7z Orangepi5b_1.0.4_debian_bullseye_desktop_kde-plasma_linux5.10.110.7z Orangepi5b_1.0.4_ubuntu_focal_server_linux5.10.110.7z Orangepi5b_1.0.4_ubuntu_focal_desktop_xfce_linux5.10.110.7z Orangepi5b_1.0.4_ubuntu_jammy_server_linux5.10.110.7z Orangepi5b_1.0.4_ubuntu_jammy_desktop_xfce_linux5.10.110.7z Orangepi5b_1.0.4_ubuntu_jammy_desktop_gnome_linux5.10.110.7z * 交换 LCD1 和 LCD2 的 dtbo 配置，和开发板上的丝印保持一致