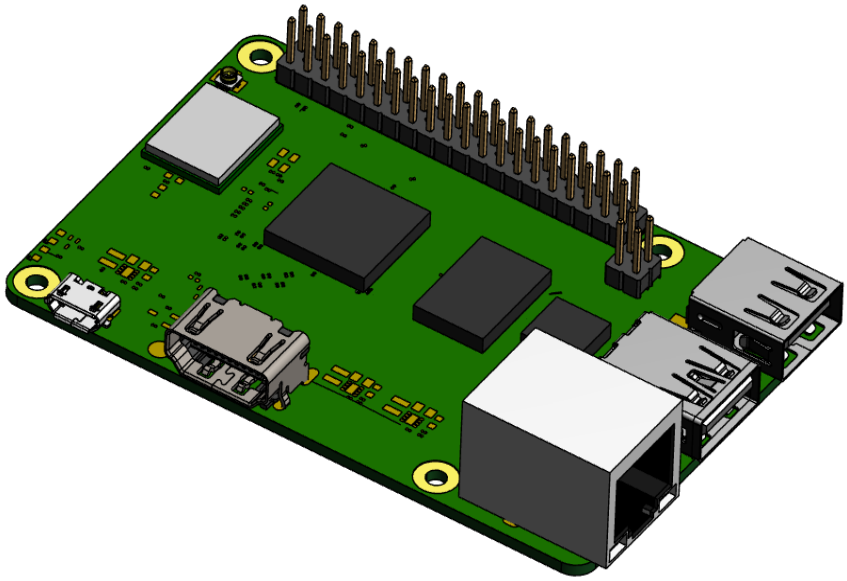


睿莓 1 (REIMEI1)

一款性能优异且极具性价比的单板计算机

上海晶珩电子科技有限公司

2023-03-18

版权声明

睿莓 1 (REIEMI1) 及其相关知识产权为上海晶珩电子科技有限公司所有。
上海晶珩电子科技有限公司拥有本文件的版权并保留所有权利。未经上海晶珩电子科技有限公司的书面许可，不得以任何方式和形式修改、分发或复制本文件的任何部分。

免责声明

上海晶珩电子科技有限公司不保证本手册中的信息是最新的、正确的、完整的或高质量的。上海晶珩电子科技有限公司也不对这些信息的进一步使用作出保证。如果由于使用或不使用本手册中的信息，或由于使用错误或不完整的信息而造成的物质或非物质相关损失，只要没有证明是上海晶珩电子科技有限公司的故意或过失，就可以免除对上海晶珩电子科技有限公司的责任索赔。上海晶珩电子科技有限公司明确保留对本手册的内容或部分内容进行修改或补充的权利，无需特别通知。

目　录

1 产品概述

睿莓 1 是一款具有优异性能、小巧紧凑的，并且极具性价比的单板计算机。

1.1 目标应用

- 多媒体创作
- 人工智能
- 创客开发
- 智能制造

1.2 规格参数

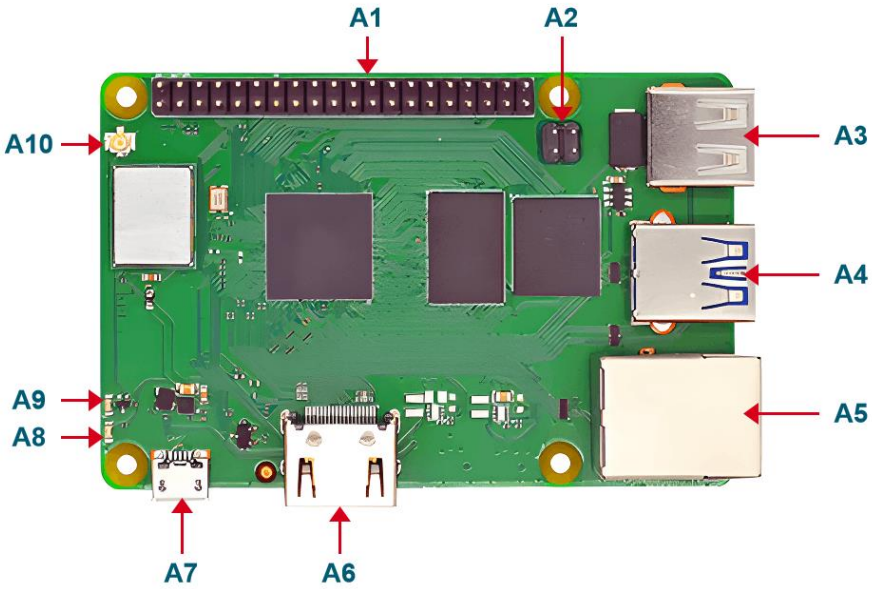
1.2.1 硬件

功能	参数
CPU	Amlogic S905X4 4 核，Cortex-A55 (ARMv8-A)，64 位，2.0GHz
GPU	ARM Mail-G31MP2
内存	1GB / 2GB / 4GB 可选
eMMC 闪存	8GB / 16GB / 32GB 可选
SD 卡	micro SD 卡
以太网	1x 10/100M 以太网，支持 POE HAT
WiFi / 蓝牙	2.4G / 5.8G 双频 WiFi，蓝牙 5.0
HDMI	1x 标准 HDMI
USB Host	1x USB 2.0 Type A, 1x USB 3.0 Type A
GPIO	28 路 GPIO 供用户使用，部分 GPIO 可以复用为 UART、I2C、SPI
LED 指示灯	红色(电源指示)，绿色(系统状态指示)
电源输入	5V@2.5A
尺寸	85(长) x 56(宽) mm
天线配件	支持可选 WiFi / BT 外置天线
工作环境温度	可选-25 ~ 70°C 环境温度

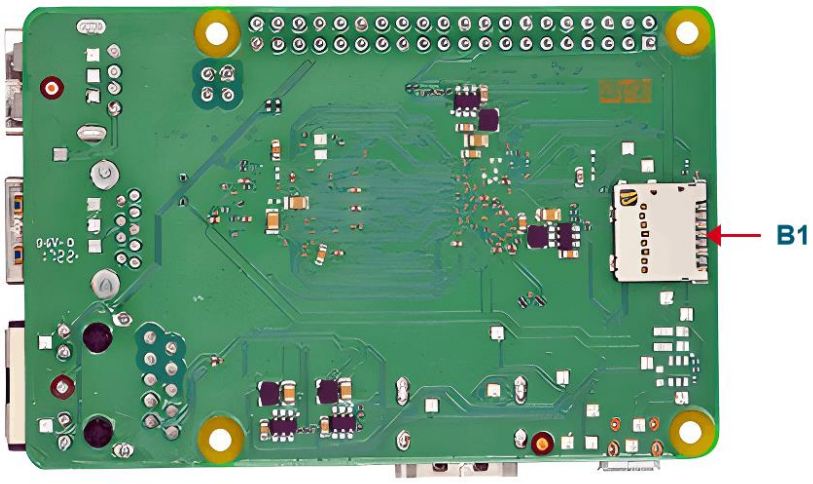
1.2.2 软件

功能	参数
操作系统	Debian 11, 64-bit
内核	Linux 5.4.180 64-bit
视频输出	HDMI 2.1 至 4Kp75，支持 CEC、HDR 和 HDCP 2.2，CVBS
视频解码	AV1 MP-10 L5.1 up to 4Kx2K @ 60fps
	VP9 Profile-2 up to 4Kx2K @ 60fps
	H.265 HEVC MP-10 @ L5.1 up to 4Kx2K @ 60fps
	AVS2-P2 Profile up to 4Kx2K @ 60fps
	H.264 AVC HP @ L5.1 up to 4Kx2K @ 30fps
	H.264 MVC up to 1080p60
	MPEG-4 ASP @ L5 up to 1080p60 (ISO-14496)
	WMV/VC-1 SP/MP/AP up to 1080p60
	AVS-P16(AVS+) /AVS-P2 JiZhun Profile up to 1080p60
	MPEG-2 MP/HL up to 1080p60 (ISO-13818)
	MPEG-1 MP/HL up to 1080p60 (ISO-11172)
	RealVideo 8/9/10 up to 1080p60
	HDR – HDR10/10+, HLG, Dolby Vision, TCH PRIME
SDK /库/工具	Mesa Graphics Library with OpenGL ES 3.2, Vulkan 1.0/1.1, and OpenCL 2.0 support
	V4L2 M2M Video Decoder Interface
	QT5 with hardware accelerated Wayland backend
	Gstreamer with hardware decode support

1.3 功能布局



编号	功能描述	编号	功能描述
A1	2×20Pin Header	A6	HDMI Type A 接口
A2	POE 接头	A7	Micro USB 接口
A3	USB 2.0	A8	LED 红色
A4	USB 3.0	A9	LED 绿色
A5	以太网 RJ45 接口	A10	天线 IPEX 接口

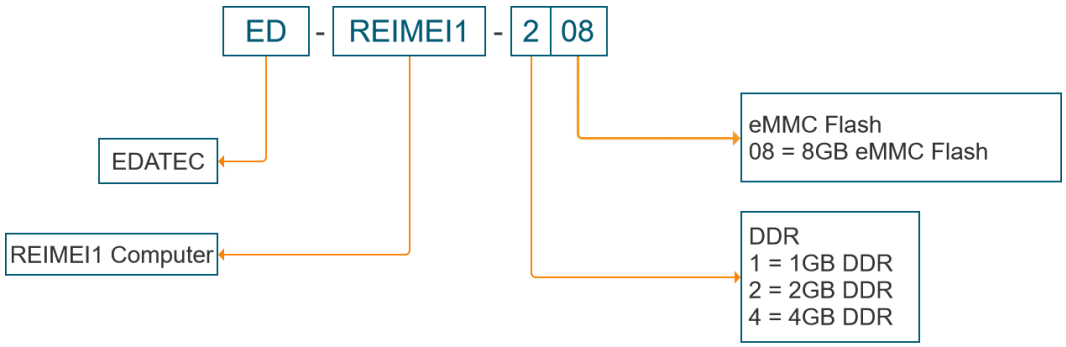


编号	功能描述
B1	Micro SD 卡槽

1.4 包装清单

- 1x 睿莓 1 主机
- [选配]1x 2.4GHz/5GHz WiFi/BT 天线

1.5 订购编码



Example

Part# : ED-REIMEI1-208
Configration : REIMEI1 Computer
2GB DDR and 8GB eMMC Flash

2 快速启动

快速启动主要是指引您如何连接设备，安装系统，首次启动配置以及网络配置等。

2.1 设备清单

- 1x 睿莓 1 主机
- 1x 2.4GHz/5GHz WiFi/BT 天线

2.2 硬件连接

您需要自备如下配件：

- 1x 5V@3A USB micro-B 电源
- 1x 网线
- 1x HDMI 标准连接线
- HDMI 显示屏
- 鼠标
- 键盘

硬件连接：

1. 将 WiFi/BT 天线安装在睿莓 1 主机上
2. 使用标准 HDMI 连接显示屏及睿莓 1
3. 插入网线
4. 连接键盘鼠标
5. 上电开机，推荐使用 5V@3A USB micro-B 电源

2.3 首次启动

设备上电之后就会自动开机，开机时红色指示灯常量，绿色指示灯闪烁。如果发现设备无法启动，绿色指示灯不闪烁，屏幕无显示，则说明此时系统无法启动，可能因为 eMMC 中无系统，请参考镜像下载及系统烧录对 eMMC 进行烧录。

2.3.1 启动桌面

睿莓 1 支持 weston 桌面环境和 X11 桌面环境。
镜像默认进入到命令行，如果用户希望启动桌面，则需要执行以下命令。

```
sudo systemctl start weston
```

如果用户希望启动 X11 桌面，则需要执行以下命令。

```
sudo systemctl start lightdm
```

2.3.2 查看系统版本

```
uname -a
```

2.3.3 设置时区

修改时区，将时间设置为中国时间：

```
sudo timedatectl set-timezone Asia/Shanghai
```

2.3.4 查看系统分区情况

使用 lsblk 命令可以获取当前分区情况，还有分区大小以及挂载路径。

```
lsblk
```

2.3.5 查看存储空间

查看当前存储空间信息：

```
df -h
```

2.3.6 设置 Hostname

修改/etc/hostname 文件，将当前 hostname 改为想要设置的 hostname，修改完成后保存：

```
sudo nano /etc/hostname
```

修改/etc/hosts 文件，将 hosts 中当前 hostname 改为想要设置的 hostname，修改完成后保存：

```
sudo nano /etc/hosts
```

修改完成后重启设备即可：

```
sudo reboot
```

2.3.7 关机

```
sudo poweroff
```

2.3.8 重启

```
sudo reboot
```

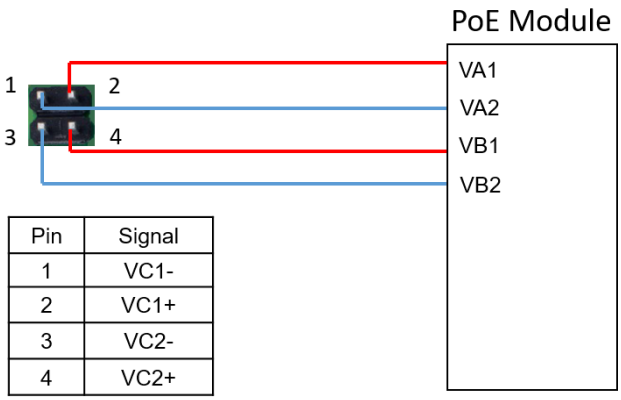
3 接线指南

3.1 IPEX-1

睿莓 1 板载的 IPEX-1 连接器用于外接 2.4GHz/5GHz 天线。使用母头 IPEX 连接天线，正对主板 IPEX 公头，向下按压即可固定天线。

3.2 POE

睿莓 1 支持网口 POE 供电，但是需要另配 POE HAT 模块。与 PoE 模块的接线示意图如下：



Pin	Signal
1	VC1-
2	VC1+
3	VC2-
4	VC2+

4 软件操作指引

睿莓 1 操作系统基于 Debian OS 构建，此章节包含了关于 Debian OS 的一些功能基本使用方法。

Debian OS 基于开源系统 Raspberry Pi OS 开发，内核版本为 5.4.180，支持 Weston 桌面（硬解），支持 QT5（Wayland backend，支持硬解），兼容大部分 Raspberry Pi OS 软件生态。

TIP: 目前系统还处于开发阶段，内部包含的各种树莓派系统文件还未去除，包括/boot 目录下的引导镜像及 dts 等。

TIP: 系统中还保留了 raspi-config，用户可以在系统中直接使用 raspi-config 来完成部分配置，如连接 WiFi，使能 ssh 等。

4.1 使用 SSH 连接到设备

4.1.1 使能 SSH

开机自动使能 SSH：
在设备启动时，开机前向 boot 分区放入一个名为 ssh 的空文件，开机后会自动使能 SSH。
在 SD 卡上使能 SSH 请参考烧录系统最后一步：烧录 SD 卡
在 eMMC 上使能 SSH 请参考烧录系统最后一步： 烧录 eMMC

命令使能 SSH：

```
sudo raspi-config
```

输入上方命令后，会出现一个命令行界面，第三个接口配置，找到 SSH，选择 yes 使能 SSH 功能。
3 Interface Options -> 2 SSH -> Yes

4.1.2 SSH 工具

Windows 端推荐使用 putty 来实现 SSH 远程连接。

- Putty 下载：Download PuTTY - a free SSH and telnet client for Windows

4.1.3 获取设备 IP 地址

有以下方式可以获取到 IP 地址：

命令查看 IP

如果设备连接到显示器且接有键盘，可以使用 ifconfig 命令查看当前 IP。

查看路由器信息

查看路由器上列出的设备 IP 地址，找到设备。设备默认 hostname 为 phantom。

使用 nmap 扫描

安装 nmap 工具: Nmap: the Network Mapper - Free Security Scanner

执行以下命令对网段 192.168.1.0~255 进行扫描:

```
nmap -sn 192.168.1.0/24
```

执行完成后屏幕会显示出所有扫描到的设备。

4.1.4 连接到系统

```
ssh phantom@<IP>
```

用户名：phantom
密码 ：phantom
端口号：22

4.2 通过调试串口连接到系统

睿莓 1 上带有一个调试串口，使用 USB 转串口（TTL 电平），串口连接到睿莓 40pin 的 6 脚(GND)，8 脚(TXD)，10 脚(RXD)。另一端连接到电脑 USB，使用串口工具找到转接器，设置波特率为 921600、数据位设置为 8 位、校验位为无、停止位为 1 位。

用户名：phantom
密码 ：phantom

4.3 使用 APT 工具管理软件

管理安装、升级和删除软件的最简单方法是使用 Debian 的 APT（高级打包工具）。要更新软件，您可

以从终端窗口使用 apt 工具。

更新 deb 包列表：

```
sudo apt update
```

安装 deb 包：

```
sudo apt install tree
```

卸载 deb 包：

```
sudo apt remove tree
```

卸载 deb 包（同时删除对应配置文件）

```
sudo apt purge tree
```

4.4 X11 桌面使用简介

X 窗口系统（X Window System），通常称为 X11 或简称为 X，它是一种位图显示的窗口系统，广泛应用于 Unix、类 Unix、Linux 系统中。X11 系统本质上只是工具包及架构规范，它有很多不同的实现，当前依据 X11 规范架构所开发的实现体中以 X.org 最为普遍和最受欢迎。

X11 系统是一个分层架构，它分为 Server 和 Client。X Server 负责图形界面的显示和用户的输入，而 Client 程序需要连接到 X Server，然后请求 X Server 绘制图形界面，同时从 X Server 接受用户的输入。在桌面系统上，X Server 和 Client 程序往往安装在同一台机器上，使用中基本感觉不到它是分层的。

X11 桌面环境是将各种组件捆绑在一起，以提供常见的图形用户界面元素，同时包含一组集成的应用程序和实用程序，最重要的是桌面环境提供一个窗口管理器，在图形用户界面的窗口系统中，窗口管理器控制窗口的行为、位置以及外观。睿莓 1 使用 Raspberry Pi 操作系统相同的桌面环境 PIXEL，它是 LXDE 桌面环境的修改版。

X Server 的启动方式有两种，一种是通过显示管理器启动，另一种是手动启动。显示管理器（display manager）除了启动 X Server 还包括启动相应的 Client 程序，构成一个完整的桌面环境，显示管理器通常是一个图形用户界面，在系统桌面启动流程结束时进行显示，以代替默认的 shell。

睿莓 1 的 X11 桌面环境使用 lightdm 作为显示管理器。

4.4.1 启动 X11 桌面

启动 X11 桌面，则需要执行以下命令。

```
sudo systemctl start lightdm
```

4.4.2 配置系统启动到 X11 桌面

配置系统开机自动启动到 X11 桌面有两种方式，1）通过命令行执行 raspi-config 命令进行配置，2）通过图像界面 Raspberry Pi Configuration 工具进行配置。

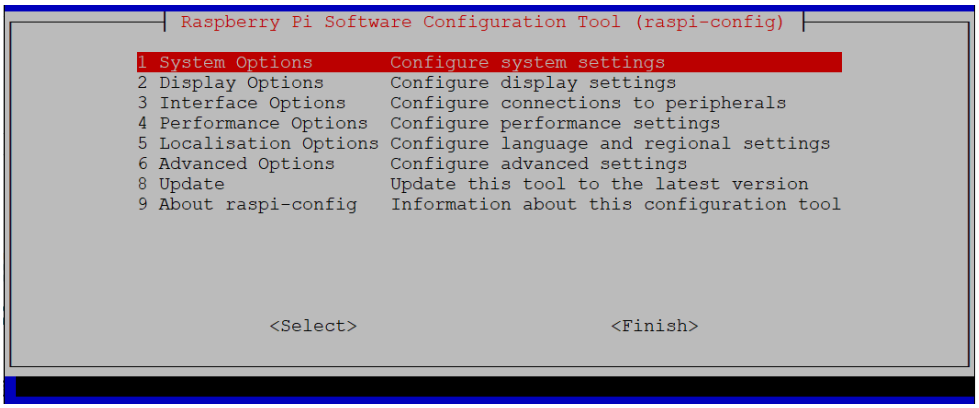
开机自动启动到 X11 桌面需要保证 lightdm 服务已经为运行状态，请先执行如下指令使能 lightdm 服务。

```
sudo systemctl enable lightdm
```

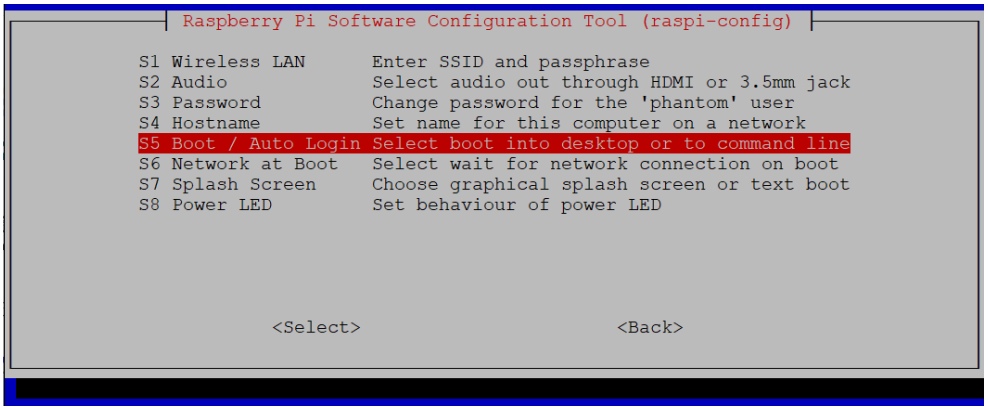
1. 通过命令行执行 raspi-config 命令进行配置

```
sudo raspi-config
```

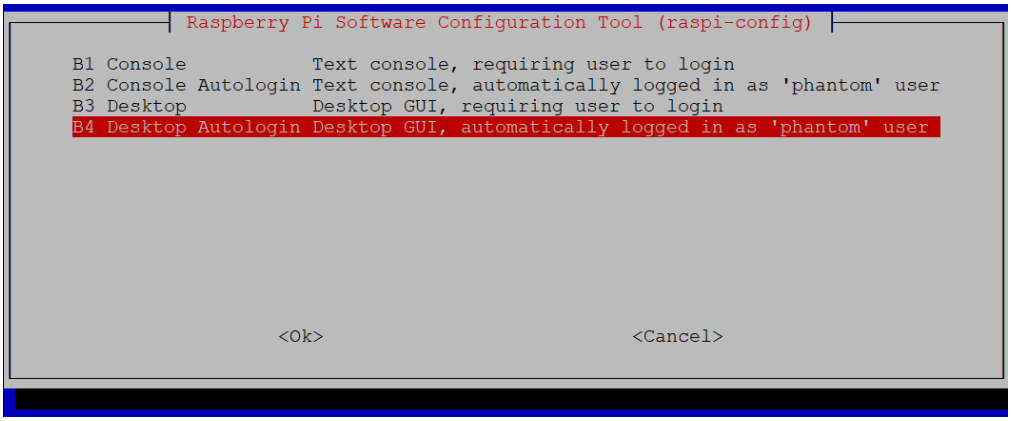
选择 System Options



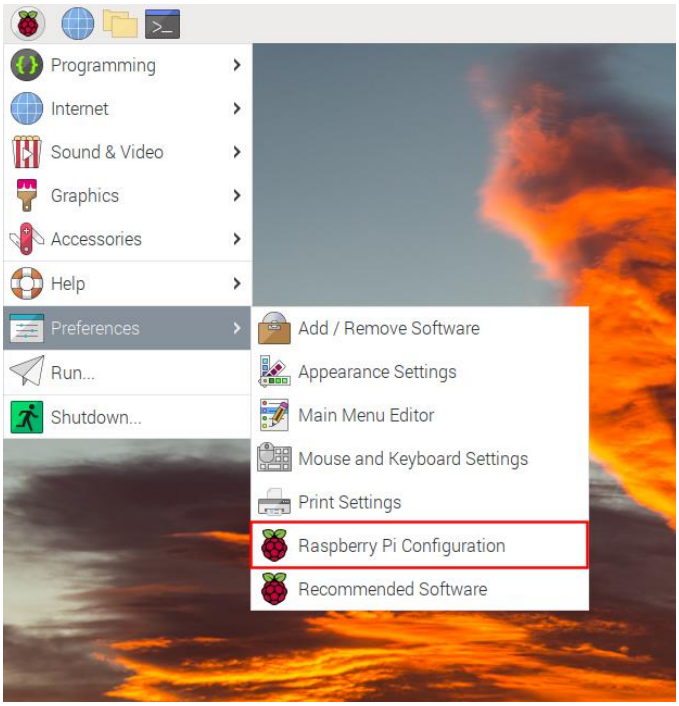
选择 Boot / Auto Login



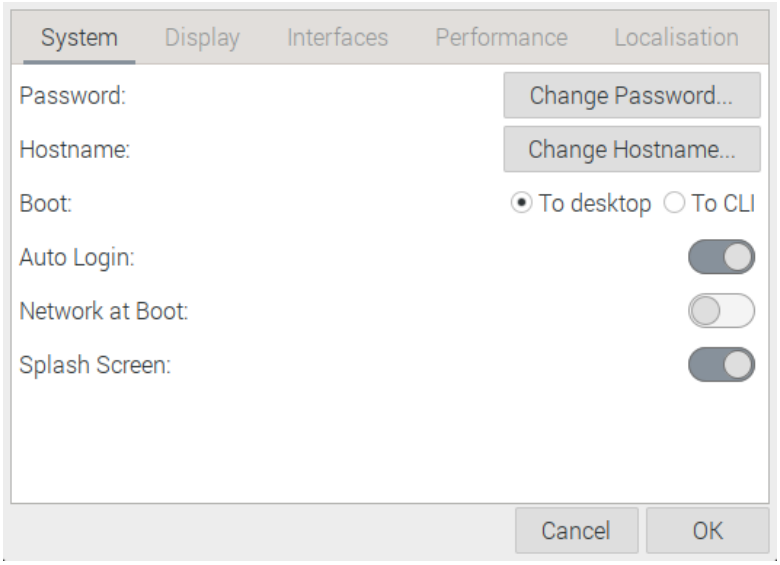
选择 Desktop Autologin，启动桌面并自动登录



2. 通过图像界面 Raspberry Pi Configuration 工具进行配置



在 System 页面，配置 Boot 为 To desktop，并使能 Auto Login.

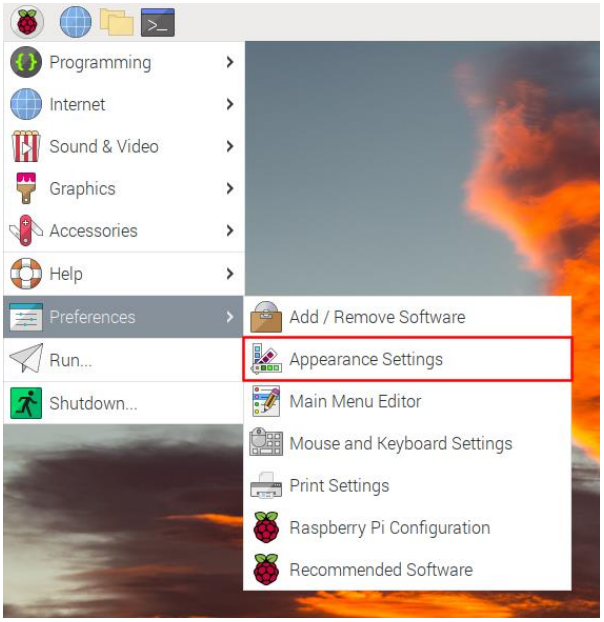


4.4.3 桌面任务栏

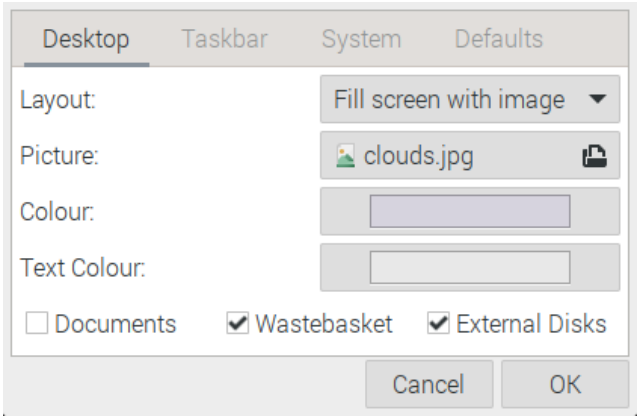
编号	功能描述
1	开始菜单
2	Web 浏览器
3	主用户文件夹
4	Terminal 快捷方式
5	蓝牙连接图标
6	WiFi 连接图标
7	HDMI 声音输出音量配置
8	系统时间显示

4.4.4 桌面个性化设置

选择 Perferences->Appearance Settings.



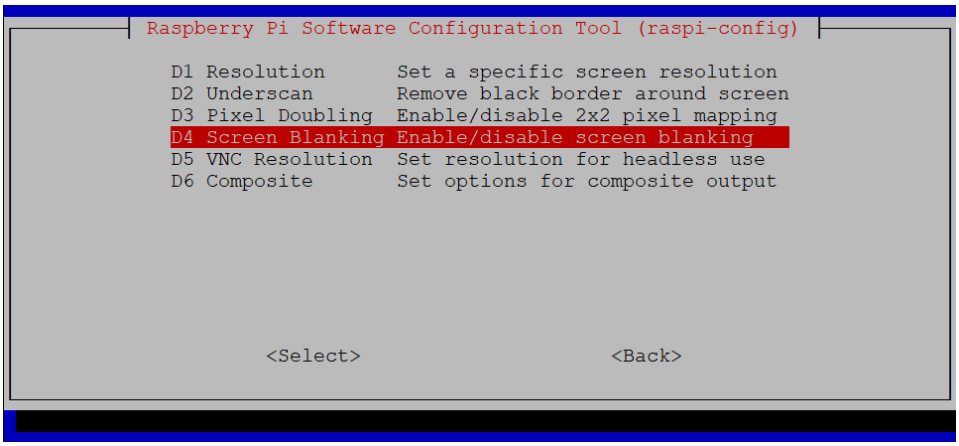
在 Appearance Settings 中可以配置桌面图片显示、任务栏和系统字体等信息。



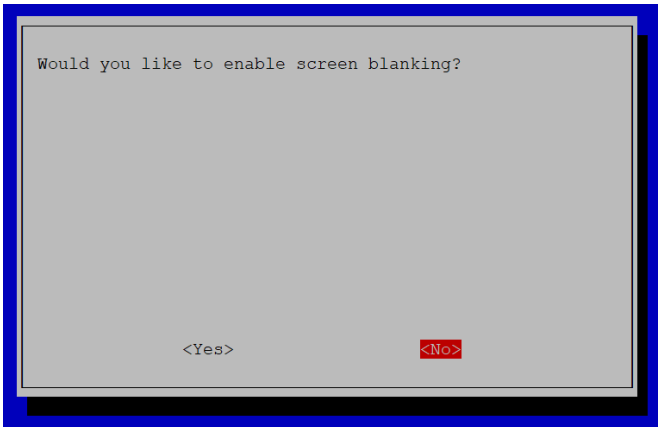
4.4.5 关闭自动息屏功能

1. 通过命令行执行 raspi-config 命令进行配置

选择 Display Options，然后选中 Screen Blanking，按回车键进行确认。

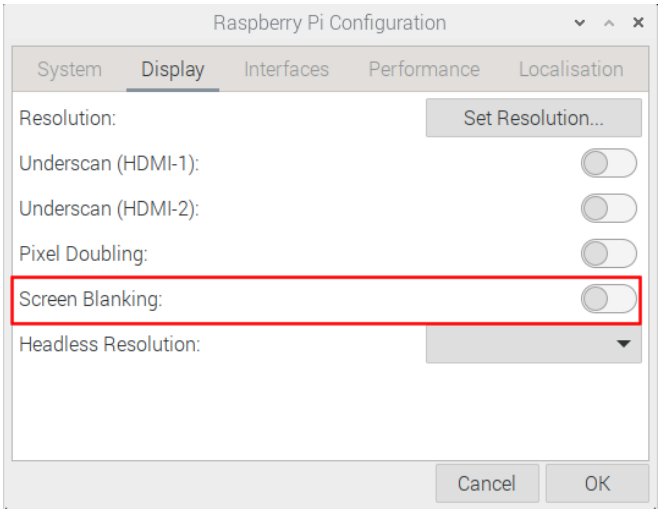


选择 No 不启用自动息屏功能。



2. 通过图像界面 Raspberry Pi Configuration 工具进行配置

在 Display 页面选择禁用 Screen Blanking



4.4.6 查看 X11 桌面系统日志

通过查看 X11 系统日志可以帮助调试问题

查看 lightdm 日志

```
#需要 root 权限才能查看 lightdm 日志
sudo cat /var/log/lightdm/lightdm.log
```

查看 Xorg 日志

```
cat /var/log/Xorg.0.log
```

4.4.7 X11 桌面截屏

系统默认已预安装 scrot 工具，如果没有安装请执行如下指令：

```
sudo apt-get install scrot
```

按下键盘上的 PrtScn 屏幕打印按键即可截屏，文件保持在主用户文件夹/home/phantom。

对当前终端窗口进行截屏

```
scrot -u
```

用鼠标选择一个窗口或者区域进行截屏

```
scrot -s
```

添加延迟截屏

```
scrot -d x
```

表示延迟 x 秒进行截屏

在远程 SSH 命令行执行截屏
先执行如下指令

```
export DISPLAY=:0.0
```

然后执行 scrot 指令，截图将会保存在当前命令行终端所在的目录。

4.4.8 播放音频

查看声卡

```
aplay -l
```

用户可以使用以下命令播放声音：

```
aplay test.mp3
```

当然也支持用户指定声卡设备录音：

```
aplay -Dhw:<sound_card_id> test.mp3
```

4.5 Weston 桌面使用简介

Wayland 是一套 display server(Wayland compositor)与 client 间的通信协议，它定位于在 Linux 上替换 X 图形系统，应用程序可以使用该协议与显示服务器进行对话，以使自己可以并同时获得用户的输入，Wayland 的显示服务器被称为合成器，应用程序就是 Wayland 的客户端，而 weston 就是一个 Wayland 合成器的参考实现，它提供了一个基本的桌面应用环境。睿莓 1 的 weston 桌面实现了图形硬解码。

4.5.1 Weston 桌面启动和关闭

系统默认没有启用 weston 应用。您可以按照如下方式启动、关闭、配置开机自动启动 weston 桌面。

- **启动 weston 桌面**

```
sudo systemctl start weston
```

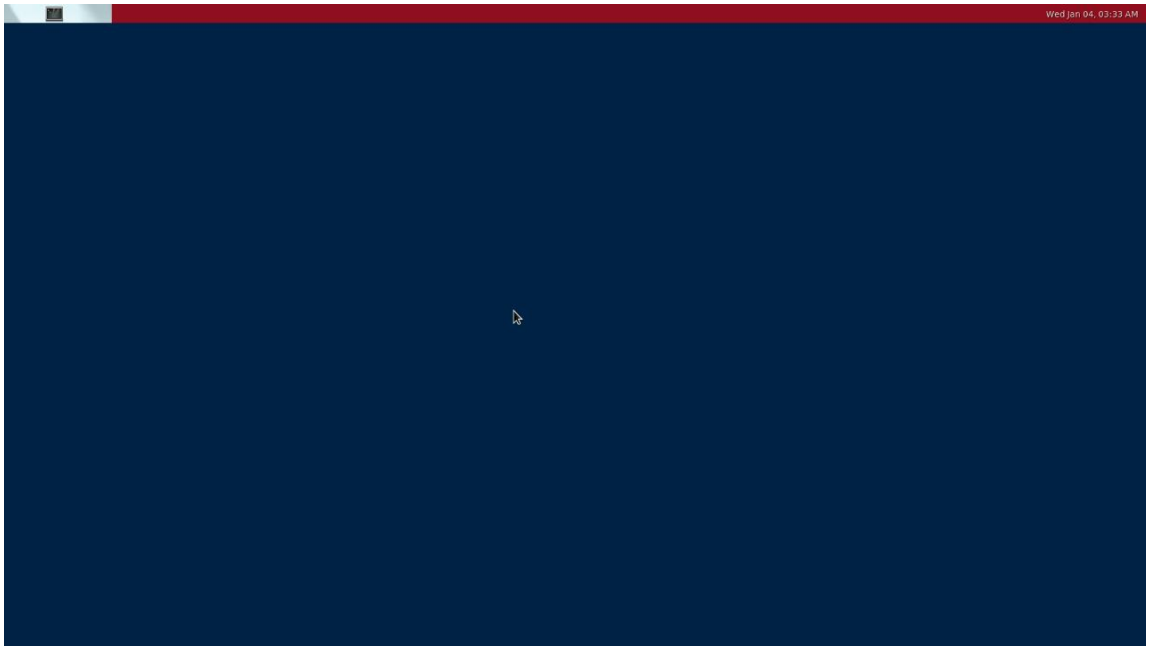
- **关闭 weston 桌面**

```
sudo systemctl stop weston
```

- **重启 weston 桌面**

```
sudo systemctl restart weston
```

Weston 默认桌面显示如下图所示，您可以根据您的需要更换桌面背景颜色、状态栏颜色、桌面图片、添加应用启动快捷方式以及是否显示状态栏等个性化定制，详细请参考 Weston 高级配置。



4.5.2 配置系统启动到 weston 桌面

系统默认以命令行方式启动。参考以下指令配置系统为桌面模式。

修改 weston.service 服务文件

```
sudo nano /etc/systemd/system/weston.service
```

删除 WantedBy 前面的#，使 WantedBy 配置生效。

```
[Unit]
Description=Weston Wayland Compositor
RequiresMountsFor=/run

[Service]
User=root
PAMName=login
EnvironmentFile=-/etc/default/weston
ExecStart=/usr/bin/weston-start -v -e -- $OPTARGS

[Install]
WantedBy=multi-user.target
```

使能 weston 开机自动启动

```
sudo systemctl enable weston
```

4.5.3 Weston 桌面系统日志

通过查看 weston 系统日志可以帮助调试问题

```
cat /var/log/weston.log
```

4.5.4 Weston 桌面的截屏

修改/etc/default/weston

```
sudo nano /etc/default/weston
```

在最后一行加入 OPTARGS="--debug"，然后重启 weston

```
sudo systemctl restart weston
```

执行截屏命令

```
weston-screenshooter
```

截屏命令快捷键为 Win+s，执行截屏命令后将会在对应目录生成 wayland-screenshot-xxx-xxx.png 的图片文件，默认保存目录为系统根目录/。

4.5.5 Weston 桌面的录屏

快键键 Win+r 执行开始/停止屏幕录制，生成.wcap 格式的文件，这是一种低损耗的 weston 专有格式，可以通过 wcap 工具进行解码：

```
wcap-decode --yuv4mpeg2 capture.wcap > capture.y4m
```

y4m 是一种原始格式，可以使用 vlc 打开，也可以使用 ffmpeg 进行编码：

```
ffmpeg -y -i capture.y4m -c:v libx264 -pix_fmt yuv420p capture.mp4
```

4.5.6 播放视频

睿莓 1 系统支持视频硬解码，视频硬件依赖于 Wayland，请保持 weston 桌面是开启状态。视频播放的详细操作请参考 Gstreamer。

4.6 系统配置

4.6.1 网络配置

4.6.1.1 扫描可用 WiFi 网络

```
sudo iwlist wlan0 scan
```

4.6.1.2 连接到 WiFi

方法 1：

```
sudo raspi-config
```

选择 1 System Options，找到 S1 Wireless LAN，第一次使用需要选择国家，中国是 CN，然后会要求输入 WiFi 名称，然后再输入 WiFi 密码，保存退出即可。如果设置了国家代码则需要重启一次。

方法 2：

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

在文件中添加以下内容

```
country=CN
network={
        ssid="WiFi_SSID"
        psk="Password"
}
```

Ctrl+X 退出，回车保存。

4.6.1.3 设置静态 IP

配置静态 IP，设置 eth0 网卡的静态 IP 为 192.168.168.108，设置默认路由为 192.168.168.1，设置 DNS 为 192.168.168.1(DNS 可以省略)：

```
sudo nano /etc/dhcpcd.conf

interface eth0
static ip_address=192.168.168.108/24
static route=192.168.168.1
static domain_name_servers=192.168.168.1
```

4.6.1.4 设置网络优先级

当在多个网络同时可用时，如希望指定网络优先级需要按照以下方法操作。

将接口 wlan0 的网络优先级设置为 200，值越小优先级越高：

```
sudo nano /etc/dhcpcd.conf

#在文件中添加以下内容
interface wlan0
metric 200
```

Ctrl+X 退出，回车保存。

NOTE:请连接好 2.4GHz/5GHz WiFi/BT 天线，否则可能因为信号较弱，造成 WiFi 连接失败。

4.6.2 蓝牙配置

睿莓 1 默认使能蓝牙功能，如果需要对蓝牙进行设置，则可以使用 bluetoothctl 命令设置蓝牙。

4.6.2.1 bluetoothctl 使用

扫描

```
bluetoothctl scan on/off
```

发现设备

```
bluetoothctl discoverable on/off
```

信任设备

```
bluetoothctl trust [MAC]
```

配对

```
bluetoothctl pair [MAC]
```

连接

```
bluetoothctl connect [MAC]
```

断开连接

```
bluetoothctl disconnect [MAC]
```

更多关于蓝牙命令配置

```
bluetoothctl
help
```

NOTE:请连接好 2.4GHz/5GHz WiFi/BT 天线，否则可能因为信号较弱，蓝牙不能扫描都周围设备。

4.6.3 添加外置存储

挂载 SD 卡需要注意 SD 中不能烧录镜像，否则系统将会从 SD 卡启动。

挂载 SD 卡到/mnt 目录

```
sudo mount /dev/mmcblk0 /mnt
```

卸载 SD 卡

```
sudo umount /mnt
```

4.6.3.1 设置自动挂载

1. 获取存储设备的 UUID：

   ```
   lsblk
   ```

2. 找到设备对应的 UUID，假设设备 UUID 为 5C24-1453。
3. 打开 fstab 文件：

   ```
   sudo nano fstab
   ```

4. 将 UUID 写入到 fstab 文件中：

   ```
   UUID=5C24-1453 /mnt/mydisk fstype defaults,auto,users,rw,nofail 0 0
   ```

4.6.4 用户管理

4.6.4.1 sudo 机制

phantom 默认被加入 sudoer 用户组，被允许使用 root 权限，执行命令时可以通过在命令前加 sudo 来使用 root 权限执行命令。

4.6.4.2 切换到 root 用户

切换到 root 用户

```
sudo -s
```

4.6.4.3 创建新用户

```
sudo adduser <username>
```

执行命令后会要求输入密码等信息，创建完成后会在 home 目录为新用户生成一个新文件夹。

4.6.4.4 禁用默认用户

```
sudo passwd -l phantom
```

4.6.4.5 启用默认用户

```
sudo passwd -u phantom
```

4.7 X11 桌面高级配置

4.7.1 lightdm 配置文件

/usr/share/lightdm/lightdm.conf.d/01_debian.conf 为系统配置，普通用户不可编辑
/etc/lightdm/lightdm.conf 全局通用配置修改文件
/etc/lightdm/pi-greeter.conf greeter 配置文件
/etc/lightdm/lightdm.conf 配置可以覆盖系统配置参数。

lightdm.conf 默认配置：

```
[Seat:*]

greeter-session=pi-greeter
greeter-hide-users=false

display-setup-script=/usr/share/dispsetup.sh

autologin-user=phantom
```

greeter-hide-users 表示是否隐藏用户列表。
greeter-setup-script 指定在欢迎界面启动前运行的命令。
session-setup-script 用户会话启动之前运行，如果失败，用户会话将不启动。
session-cleanup-script 在欢迎界面或用户会话退出之后运行。
display-setup-script 用于指定在 X server 启动后执行的命令。
display-stopped-script 用于指定在 X server 退出后执行的命令。

lightdm.conf 配置中 greeter-session 字段配置为 pi-greeter，即对应/etc/lightdm/pi-greeter.conf 配置文件。

pi-greeter.conf 默认配置

```
[greeter]
default-user-image=/usr/share/raspberrypi-artwork/raspberry-pi-logo.png
desktop_bg=#d6d6d3d3dede
wallpaper=/usr/share/rpd-wallpaper/clouds.jpg
wallpaper_mode=crop
gtk-theme-name=PiXflat
gtk-icon-theme-name=PiXflat
gtk-font-name=PibotoLt 12
```

主要完成默认桌面的配置，包括桌面背景图片、默认字体、主题。

4.7.2 配置禁止自动息屏

编辑/etc/lightdm/lightdm.conf 配置文件

```
sudo nano /etc/lightdm/lightdm.conf
```

修改 xserver-command 参数值

```
xserver-command=X -s 0 -dpms
```

重启设备后生效。

4.8 Weston 高级配置

根据实际的应用场景，您可能需要对现有 weston 桌面的配置进行调整，比如更改桌面背景颜色、更换桌面图片、去掉状态栏、添加桌面快捷方式等需求。

4.8.1 Weston 配置文件简介

Weston 从其启动命令行参数和配置文件中获取配置信息，睿莓 1 开发板 Weston 桌面的配置文件为 /etc/aml-weston.ini。

aml-weston.ini 文件由多个 section 组成，这些 section 可以按任何顺序出现，也可以省略以使用默认配置值。每个 section 的格式如下：

```
[SectionHeader]
Key1=Value1
Key2=Value2
```

使用#注释掉某一行，则这一行被忽略。

```
#Key2=Value2
```

SectionHeader 部分可以为如下字段：

core	The core modules and options
libinput	Input device configuration
shell	Desktop customization
launcher	Add launcher to the panel
output	Output configuration
input-method	Onscreen keyboard input
keyboard	Keyboard layouts
terminal	Terminal application options
xwayland	XWayland options
screen-share	Screen sharing options
autolaunch	Autolaunch options

Value 可能的值类型包括字符串、有符号和无符号 32 位整数以及布尔值。字符串不能被引用，不支持任何转义序列，并一直运行到行的末尾。整数可以以十进制（例如 123）、八进制（例如 0173）和十六进制（例如 0x7b）形式给出。布尔值只能是 true 或 false。

每个 section 中 Key、Value 的详细说明请参考配置文件官方文档。

4.8.2 Weston 桌面定制化

- **更改桌面背景颜色**

设置桌面背景颜色为蓝色，在[shell]块修改 background-color 字段的值。

```
[shell]
background-color=0xff0000ff
```

32bits 十六进制数字以 8bits 为一组按顺序依次表示透明、红色、绿色和蓝色：

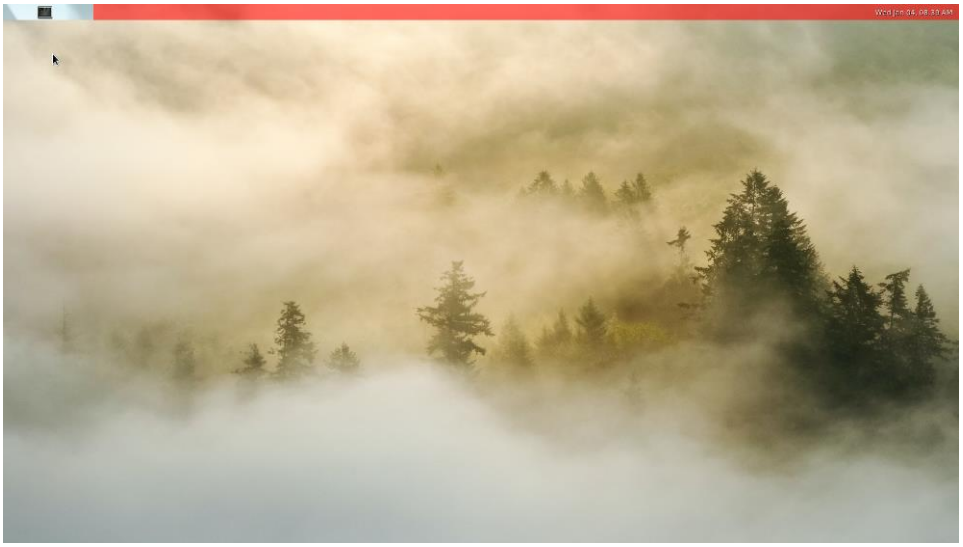
```
0xffff0000      红色
0xff00ff00      绿色
0xff0000ff      蓝色
0x00ffffff      全透明
```

- **添加桌面背景图片**

桌面图片位置为/home/phantom/Pictures/desktop.jpg，在[shell]块添加 background-image 字段，并填入正确桌面图片路径。

```
[shell]
background-image=/home/phantom/Pictures/desktop.jpg
```

重启设备后，即可显示新的桌面图片，示例如下：



- **配置状态栏显示**

设置状态栏为不显示

在[shell]块添加 panel-position 字段，值为 none，同时指定背景颜色为全透明

```
[shell]
panel-position=none
background-color=0x00FFFFFF
```

此外还可以配置 panel-position 字段的值为 top, bottom, left, right，依次表示在上方、底部、左边、右边显示状态栏。

4.8.3 添加桌面快捷方式

在[launcher]块添加 icon 字段和 path 字段，icon 字段用于指定可执行程序快捷方式的显示图片，path 字段指定可执行文件的路径。

目前系统已经在状态栏定义了 3 个快捷方式，根据您的需要进行修改或添加：

```
[launcher]
icon=/usr/share/weston/icon_flower.png
path=/usr/bin/weston-flower

[launcher]
icon=/usr/share/weston/icon_ivi_smoke.png
path=/usr/bin/weston-smoke

[launcher]
icon=/usr/share/icons/gnome/32x32/apps/utilities-terminal.png
path=/usr/bin/weston-terminal --shell=/usr/bin/bash
```

4.9 编译工具链

系统默认已安装 gcc 编译器，版本为 10.2.1

```
phantom@phantom:/ $ 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.
```

如果您需要使用交叉编译，可以安装 linaro 交叉编译工具，Linora 是免费授权版 ARM 交叉编译工具的提供商。你可以在 Arm GNU Downloads 页面获得官方已编译好的交叉工具链。
选择 gcc-arm-10.2-2020.11-x86_64-aarch64-none-linux-gnu.tar.xz 下载，将压缩包解压并将工具链目录添加到用户配置文件。
以 ubuntu20 系统为例，假设 gcc-arm-10.2-2020.11-x86_64-aarch64-none-linux-gnu.tar.xz 已放在~/tools 目录下。

```
cd ~/tools
xz -d gcc-arm-10.2-2020.11-x86_64-aarch64-none-linux-gnu.tar.xz
tar xvf gcc-arm-10.2-2020.11-x86_64-aarch64-none-linux-gnu.tar

echo PATH=$PATH:~/tools/gcc-arm-10.2-2020.11-x86_64-aarch64-none-linux-gnu/bin >> ~/.bashrc
```

4.10 接口设备文件

功能接口	外设	Linux 设备文件
eMMC	MMC1	/dev/mmcblk1
Micro SD	MMC0	/dev/mmcblk0
Uart	UART0	/dev/ttyS0
i2c0	I2C	/dev/i2c-0
i2c1	I2C	/dev/i2c-1
spi dev0	SPI dev	/dev/spidev0.0
spi dev1	SPI dev	/dev/spidev0.1

4.11 40-Pin GPIO 编程指南

睿莓 1 具有一个 2.54mm 间距 2X20P 的排针，将主控芯片的 28 个 GPIO 引出，用户可通过软件去控制这些 GPIO，目前已支持 2 路 I2C、1 路 UART、1 路 SPI 以及多路输入输出可配置 GPIO。

TIP: 睿莓 1 40-Pin 管脚兼容树莓派 40-Pin 管脚（I2C、UART、SPI）。

4.11.1 使用 libgpiod 控制 GPIO

安装 ligpiod

```
sudo apt-get update
#安装 libgpiod 的静态库及头文件
sudo apt-get install libgpiod-dev
#安装基于 libgpiod 的命令行工具
sudo apt-get install gpiod
```

libgpiod 支持 6 个命令行测试命令，分别是：

- gpiodetect - 列出系统上存在的所有 gpiochips，它们的名称，标和 GPIO 线数
- gpioinfo - 列出指定 gpiochips 的所有行、它们的名称、消费者、方向、活动状态和附加标志
- gpioget - 读取指定 GPIO 线的值
- gpioset - 设置指定 GPIO 线的值，可能保留这些线导出并等待超时、用户输入或信号
- gpiofind - 找到 gpiochip 名称和给定行名称的行偏移
- gpiomon - 等待 GPIO 线上的事件，指定要观看的事件，退出前要处理多少事件或事件应该报告给控制台

查看 gpiochip 信息

```
phantom@phantom:~ $ gpioinfo
gpiochip0 - 87 lines:
        line   0:      "PIN27"         kernel   input   active-high [used]
        line   1:      "PIN28"         kernel   input   active-high [used]
        line   2:  "EMMC_DAT0"         kernel   input   active-high [used]
        line   3:  "EMMC_DAT1"         kernel   input   active-high [used]
```

```
line   4:  "EMMC_DAT2"          kernel    input   active-high [used]
line   5:  "EMMC_DAT3"          kernel    input   active-high [used]
line   6:  "EMMC_DAT4"          kernel    input   active-high [used]
line   7:  "EMMC_DAT5"          kernel    input   active-high [used]
line   8:  "EMMC_DAT6"          kernel    input   active-high [used]
line   9:  "EMMC_DAT7"          kernel    input   active-high [used]
line  10:   "EMMC_CLK"          kernel    input   active-high [used]
line  11:   "NAND_ALE"          unused    input   active-high
line  12:   "EMMC_CMD"          kernel    input   active-high [used]
line  13:             "-"       unused    input   active-high
line  14:   "EMMC_RST"          unused    input   active-high
line  15: "EMMC_NAND_DQS" kernel input active-high [used]
line  16:             "-"       unused    input   active-high
line  17:             "-"       unused    input   active-high
line  18:    "SD_DAT0"          kernel    input   active-high [used]
line  19:    "SD_DAT1"          kernel    input   active-high [used]
line  20:    "SD_DAT2"          kernel    input   active-high [used]
line  21:    "SD_DAT3"          kernel    input   active-high [used]
line  22:     "SD_CLK"          kernel    input   active-high [used]
line  23:     "SD_CMD"          kernel    input   active-high [used]
line  24:      "SD_CD"           "cd"     input   active-high [used]
line  25:    "USB_PSU" "fe03a080.usb3phy" output active-low [used]
line  26:  "VDDEE_PWM"          unused    input   active-high
line  27: "VDDCPU_PWM"          kernel    input   active-high [used]
line  28:         "LED"          "act"   output   active-high [used]
line  29:   "DEBUG_TX"          kernel    input   active-high [used]
line  30:   "DEBUG_RX"          kernel    input   active-high [used]
line  31:      "PIN40"          unused    input   active-high
line  32:      "PIN31"          unused    input   active-high
line  33:      "PIN12"          unused    input   active-high
line  34:             "-"       unused    input   active-high
line  35:      "PIN32"          unused    input   active-high
line  36:      "PIN29"          unused    input   active-high
line  37:       "PIN8"          kernel    input   active-high [used]
line  38:      "PIN10"          kernel    input   active-high [used]
line  39:             "-"       unused    input   active-high
line  40:      "PIN35"          unused    input   active-high
line  41:   "HDMI_SDA"          kernel    input   active-high [used]
line  42:   "HDMI_SCL"          kernel    input   active-high [used]
line  43:   "HDMI_HPD"          kernel    input   active-high [used]
line  44:   "HDMI_CEC"          kernel    input   active-high [used]
line  45:      "PIN19"          kernel    input   active-high [used]
line  46:      "PIN21"          kernel    input   active-high [used]
line  47:      "PIN24"        "spi0.0"   output   active-high [used]
```

```
        line  48:         "PIN23"          kernel     input   active-high [used]
        line  49: "PCIE_RESET"           unused     input   active-high
        line  50: "WIFI_SD_D0"           kernel     input   active-high [used]
        line  51: "WIFI_SD_D1"           kernel     input   active-high [used]
        line  52: "WIFI_SD_D2"           kernel     input   active-high [used]
        line  53: "WIFI_SD_D3"           kernel     input   active-high [used]
        line  54: "WIFI_SD_CLK" kernel input active-high [used]
        line  55: "WIFI_SD_CMD" kernel input active-high [used]
        line  56:               "-"          unused     input   active-high
        line  57:               "-"          unused     input   active-high
        line  58:               "-"          unused     input   active-high
        line  59:               "-"          unused     input   active-high
        line  60:         "BT_ON"     "shutdown"    output   active-high [used]
        line  61:         "WL_ON"          unused     input   active-high
        line  62: "BTUART_A_TX" kernel input active-high [used]
        line  63: "BTUART_A_RX" kernel input active-high [used]
        line  64: "BTUART_A_CTS_N" kernel input active-high [used]
        line  65: "BTUART_A_RTS_N" kernel input active-high [used]
        line  66:               "-"          unused     input   active-high
        line  67:               "-"          unused     input   active-high
        line  68:               "-"          unused     input   active-high
        line  69:               "-"          unused     input   active-high
        line  70:          "PIN3"           kernel     input   active-high [used]
        line  71:          "PIN5"           kernel     input   active-high [used]
        line  72:         "PIN18"           unused     input   active-high
        line  73:         "PIN22"           unused     input   active-high
        line  74:         "PIN37"           unused     input   active-high
        line  75:         "PIN13"           unused     input   active-high
        line  76:         "PIN15"           unused     input   active-high
        line  77:         "PIN16"           unused     input   active-high
        line  78:         "PIN26"        "spi0.1"    output   active-high [used]
        line  79:         "PIN11"           unused     input   active-high
        line  80:         "PIN36"           unused     input   active-high
        line  81:         "PIN38"           unused     input   active-high
        line  82:         "PIN33"           unused     input   active-high
        line  83:          "PIN7"           unused     input   active-high
        line  84:     "LAN_LEDG"            kernel     input   active-high [used]
        line  85:     "LAN_LEDY"            kernel     input   active-high [used]
        line  86:               "-"          unused     input   active-high
```

可以看到系统只有一个 gpiochip0，共有 87 个 GPIO 管脚，已被驱动或系统占用的 GPIO 会在最后一列显示为[used].

根据数据手册 **ED-REIMEI1_Datasheet_CN.pdf** 文档中的 GPIO 管脚名称，结合 gpioinfo 的返回结果查找到

对应的 line 号。以引脚 7 为例说明，查数据手册引脚 7 的管脚名称为 PIN7，PIN7 对应的 line 号为 83。

```
phantom@phantom:~ $ gpioinfo | grep PIN7
        line   83:        "PIN7"        unused     input    active-high
```

配置 GPIO 输出

```
#设置 chip0 的 line83 管脚为低电平
gpioset 0 83=0

#设置 chip0 的 line83 管脚为高电平
gpioset 0 83=1
```

配置 GPIO 输入

```
#读取 chip0 的 line83 管脚状态
gpioget 0 83
```

返回值为 1 时，表示 line83 即 PIN7 管脚为高电平，返回值为 0 时，表示 line83 即 PIN7 管脚为低电平。

4.11.2 i2c_dev

设备已支持两路 i2c 总线
/dev/i2c-0
/dev/i2c-1

设备	引脚号	管脚名称	功能
i2c-0	27	PIN27	SDA
	28	PIN28	SCL
i2c-1	3	PIN3	SDA
	5	PIN5	SCL

系统已安装 i2c-tools
可以使用 i2cdetect 查看 i2c 总线上的设备

```
#查看 i2c-0 总线上的设备
i2cdetect -y 0

#查看 i2c-1 总线上的设备
i2cdetect -y 1
```

i2c-dev 设备应用程序开发参考示例。

4.11.3 spi_dev

设备已支持两路 spidev

/dev/spidev0.0
/dev/spidev0.1

设备	引脚号	管脚名称	功能
spidev	24	PIN24	SPI_CE0
	26	PIN26	SPI_CE1
	19	PIN19	SPI_MOSI
	21	PIN21	SPI_MISO
	23	PIN23	SPI_CLK

spidev 设备应用程序开发的参考示例。

4.12 QT 编程指南

系统默认已安装 Qt，版本为 5.15.2，已安装 OpenGL ES 库，版本为 3.2，本 Qt 开发环境图形显示后端基于 Wayland 实现，支持硬件解码，weston 作为 Wayland 的合成器，需要保持开启状态，开启及使能 weston 桌面请参考 Weston 桌面启动和关闭。

4.12.1 Qt 环境快速应用

本系统 Qt 环境已提供大量测试示例程序，并且已经编译好可供直接使用，测试示例目录为 /usr/lib/aarch64-linux-gnu/qt5/examples/。

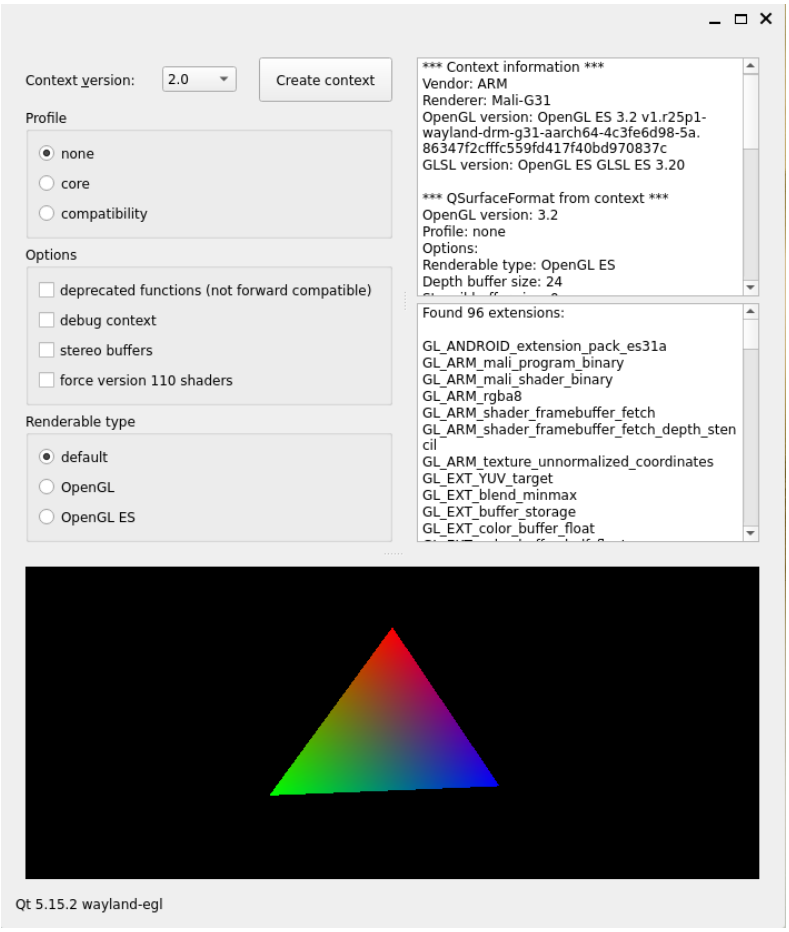
如您希望重新编译请参考命令行方式编译 Qt Widgets 应用程序和命令行方式编译 Qt Quick(QML)应用程序。

执行 Qt openGL ES 示例

```
/usr/lib/aarch64-linux-gnu/qt5/examples/opengl/contextinfo/contextinfo
```

TIP：如果希望通过 SSH 远程终端方式执行 Qt 图形程序，需要在可执行程序前添加 sudo WAYLAND_DISPLAY=wayland-0 XDG_RUNTIME_DIR=/run/user/0 QT_QPA_PLATFORM=wayland-egl

Renderable type 请选择 default 或 OpenGL ES，点击 Create context，执行结果如下:



4.12.2 安装其它的依赖库

根据您的应用软件需要，可以需要安装某些特定库：

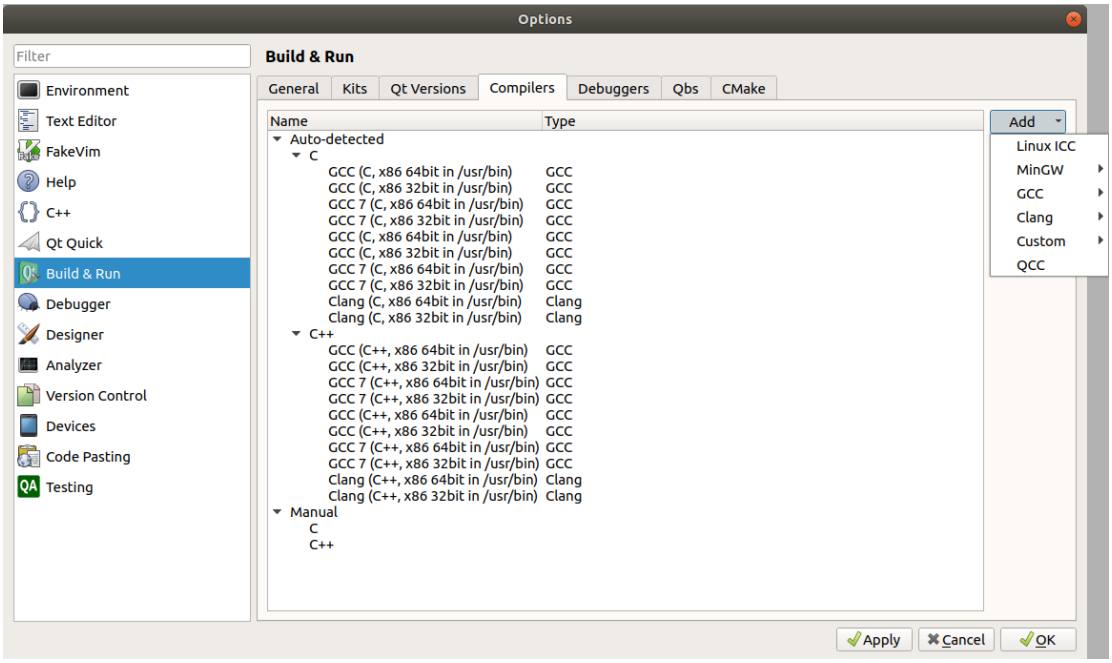
```
#安装 qml 开发环境
sudo apt-get install qtdeclarative5-dev

#安装 QtMultimedia
sudo apt-get install qtmultimedia5-dev

#安装 Qtserialport
sudo apt-get install libqt5serialport5-dev

#安装 opengl 的开发环境
sudo apt-get install libgles2-mesa-dev

#安装 QtMySQL
sudo apt-get install libqt5sql5-mysql
```

TIP：如果在运行程序时报缺少某个库的错误，可用如上方式替换成对应库名进行安装。

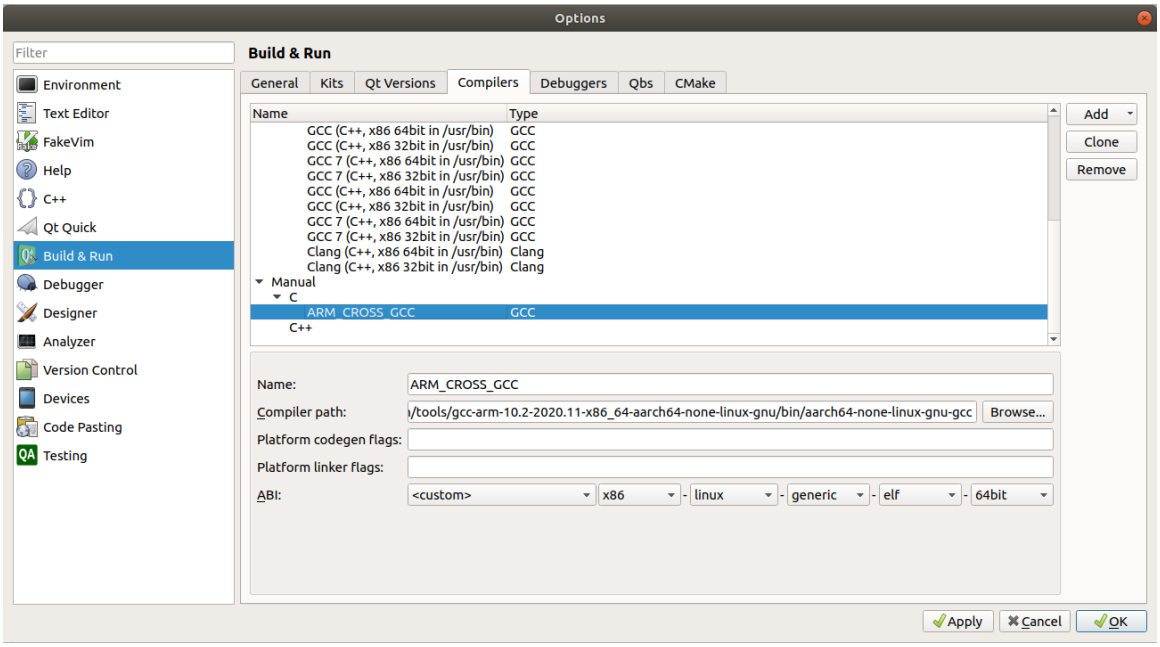
4.12.3 配置 Qt Creator 交叉编译环境

目前系统不支持在设备端通过 apt install 的方式安装 Qt Creator。可以通过在 Host PC 端安装 Qt Creator，完成应用软件界面部分的开发，然后将工程整体复制到睿莓 1 开发板，用命令行方式完成整体的编译，除了基于 opengl 的应用必须在开发板上进行编译，对于其他简单的应用同样也可以在 Host PC 端通过交叉编译的方式完成编译，再将可执行文件复制到睿莓 1 开发板运行。

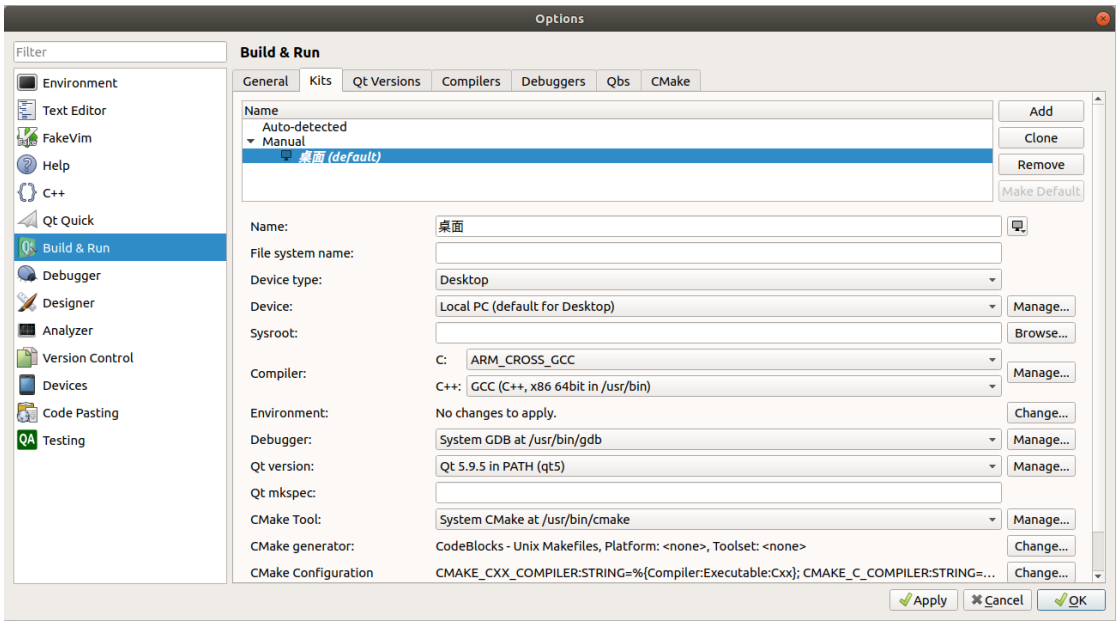
打开 Qt Creator 进入集成开发环境，选择菜单栏 Tools 下 Options 选项，打开左边的 Build & Run 菜单。选中 Manual 下的 C，然后点击 Add，选中 GCC。



在下方出现的配置框中，输入一个便于区分的名字，然后配置工具链的路径，选中下方的 Browse…，选中 aarch64-none-linux-gnu-gcc 可执行文件的位置。然后点击 OK 完成配置。



最后修改构建套件(Kits)，选中桌面(default)，以 C 编译器为例，在下拉列表中选中刚才添加的交叉编译工具的名称，按类似的方法完成 C++交叉编译工具的添加。



4.12.4 命令行方式编译 Qt Widgets 应用程序

以 hellogles3 为例

```
cd /usr/lib/aarch64-linux-gnu/qt5/examples/opengl/hellogles3
sudo qmake hellogles3.pro
sudo make
```

执行./hellogles3 即可运行。

4.12.5 命令行方式编译 Qt Quick(QML)应用程序

- 在 HOST PC 端安装 Qt Creator.
- 新建工程，Projects 选择 Application(Qt Quick)
- 将生成的工程整体复制到开发板
- 在设备端安装 qml 应用依赖库 qtdeclarative5-dev

生成的 main.qml 即为 qml 源文件，xxxx.pro 即为工程文件

```
#安装 qml 依赖库
sudo apt-get install qtdeclarative5-dev

#生成工程的 makefile，xxxx 为对应的工程名
qmake xxxx.pro
make
```

4.13 Gstreamer

GStreamer 是一个非常强大和通用的用于开发流媒体应用程序的框架，应用程序可以通过管道（Pipeline）的方式，将多媒体处理的各个步骤串联起来，达到预期的效果。每个步骤通过元素（Element）基于 GObject 对象系统通过插件（plugins）的方式实现，方便了各项功能的扩展。

gst-launch-1.0 用于创建及执行一个 pipline，因此通常使用 gst-launch 先验证相关功能，然后再编写相应应用。

安装 gstreamer 工具

```
sudo apt-get install gstreamer1.0-tools
```

4.13.1 H264 mkv 格式视频解码

测试视频下载地址。

如果希望通过 SSH 远程终端执行 gst-launch-1.0，需要在命令前添加 sudo WAYLAND_DISPLAY=wayland-0 XDG_RUNTIME_DIR=/run/user/0 QT_QPA_PLATFORM=wayland-egl，如果是在 weston 桌面直接通过终端窗口执行，则可以不用添加本段前置指令。

```
sudo WAYLAND_DISPLAY=wayland-0 XDG_RUNTIME_DIR=/run/user/0
QT_QPA_PLATFORM=wayland-egl gst-launch-1.0 -vvv filesrc
location=/home/phantom/Videos/jellyfish-5-mbps-hd-h264.mkv ! matroskademux ! h264parse !
v4l2h264dec ! video/x-raw,format=NV12 ! waylandsink
```

4.13.2 mp4 格式解码

目前系统 gst-launch-1.0 不支持 MP4 格式解码，需要通过 ffmpeg 转码为 H264 格式。

```
ffmpeg -i input.mp4 -c:v copy -bsf h264_mp4toannexb output.h264
```

然后再通过 gst-launch-1.0 解码

```
gst-launch-1.0 -vvv filesrc location=output.h264 ! h264parse ! v4l2h264dec ! video/x-
raw,format=NV12 ! waylandsink
```

4.14 Docker

睿莓 1 支持 Docker 服务。

4.14.1 Docker 的安装

1. 更新 apt 包索引并安装包，以允许 apt 通过 HTTPS 使用存储库

```
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg lsb-release
```

2. 添加 Docker 的官方 GPG 密钥

```
sudo mkdir -m 0755 -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
```

3.设置 repository

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

4. 再次更新 apt 包索引

```
sudo apt-get update
```

5. 安装 Docker Engine、containerd 和 Docker Compose

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

4.14.2 Docker 的卸载

```
sudo apt-get remove docker docker-engine docker.io containerd runc
```

更多详细说明可以参考官方文档 Install Docker Engine on Debian。

4.14.3 查看 Docker

查看 Docker 版本

```
docker --version
```

查看 Docker 系统信息

```
sudo docker info
```

```
Client:
 Context:    default
 Debug Mode: false
 Plugins:
  buildx: Docker Buildx (Docker Inc.)
    Version:  v0.10.2
    Path:     /usr/libexec/docker/cli-plugins/docker-buildx
  compose: Docker Compose (Docker Inc.)
    Version:  v2.16.0
    Path:     /usr/libexec/docker/cli-plugins/docker-compose
```

```
Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 0
 Server Version: 23.0.1
 Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Using metacopy: false
  Native Overlay Diff: true
  userxattr: false
 Logging Driver: json-file
 Cgroup Driver: systemd
 Cgroup Version: 2
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
 Swarm: inactive
 Runtimes: io.containerd.runc.v2 runc
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: 2456e983eb9e37e47538f59ea18f2043c9a73640
 runc version: v1.1.4-0-g5fd4c4d
 init version: de40ad0
 Security Options:
  apparmor
  seccomp
   Profile: builtin
  cgroupns
 Kernel Version: 5.4.180-phantom
 Operating System: Debian GNU/Linux 11 (bullseye)
 OSType: linux
 Architecture: aarch64
 CPUs: 4
 Total Memory: 1.937GiB
 Name: phantom
 ID: cfebd162-a5bc-49cd-a20c-63f114b333ec
 Docker Root Dir: /var/lib/docker
 Debug Mode: false
 Registry: https://index.docker.io/v1/
 Experimental: false
 Insecure Registries:
  127.0.0.0/8
 Live Restore Enabled: false
```

只有 Client 和 Server 都运行正常才表示 Docker 安装成功，查看 Docker 服务为运行状态。

查看 docker 服务是否运行

```
sudo systemctl status docker
```

```
phantom@phantom:~ $ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2023-03-09 11:47:34 GMT; 13h ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 717 (dockerd)
      Tasks: 10
     Memory: 85.2M
        CPU: 1.477s
     CGroup: /system.slice/docker.service
             └─717 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

如果未运行请执行

```
sudo systemctl start docker
```

4.14.4 使用 Docker

拉取镜像

```
sudo pull <image_name>
```

列出所有本地镜像

```
sudo docker image ls
```

启动容器

```
sudo run <image_name>
```

进入容器并使用 bash 作为 shell

```
sudo docker exec -it <container_id> /bin/bash
```

通过镜像创建容器并进入容器

```
sudo docker run -it <image_name> /bin/bash
```

它等价于执行如下指令：

```
sudo docker run <image_name>
sudo docker exec -it <container_id> /bin/bash
```

后台运行容器

```
sudo docker run -d -it <image_name> /bin/bash
```

为容器指定一个新名字

```
sudo docker run -it --name <container_id> <image_name> /bin/bash
```

指定主机和容器端口映射

```
sudo docker run -it --name <container_id> -p [host port]:[container port] <image_name> /bin/bash
```

查看所有容器

```
sudo docker ps
```

停止运行的容器并删除容器

```
sudo docker stop <container_id> && docker rm <container_id>
```

删除镜像

```
sudo docker image rm <image_name>
```

4.15 Bootloader 配置指南

系统使用 u-boot 作为 BootLoader，在 boot 阶段通过读取 boot 分区下的 boot.conf 文件获得启动配置参数，可以指定系统内核、启动参数、设备树文件的加载路径。

4.15.1 指定配置文件路径

在 boot 分区的根目录有一个 boot.conf 文件，load_prefix、dtb、bootargs 分别指定了系统配置的加载路径、设备树文件、启动参数文件。

/boot/boot.conf

```
# boot.conf - boot configuration file for phantom/pm3

# Set to 1 to print the proprerties to the uart.
config_print=0

# Search for Image, dtb.img and bootargs.txt under this sub-directory
load_prefix=linux/

[board_type=1]
dtb=phantom.dtb

[board_type=1 boot_mode=0]
bootargs=bootargs-phantom-sd.txt

[board_type=1 boot_mode=1]
bootargs=bootargs-phantom-emmc.txt
```

load_prefix 指定了启动配置的文件的搜索路径。
dtb 指定设备树文件的名称。
bootargs 指定了启动参数配置文件的名称。
boot_mode 指定启动参数配置文件加载的加载顺序。
config_print 为 1 会将当前启动配置输出到调试串口。

4.15.2 修改 bootargs 参数

bootargs-phantom-emmc.txt 和 bootargs-phantom-sd.txt 分别对应 emmc 和 sd 卡两种方式的启动参数。

```
phantom@phantom:/ $ cat /boot/linux/bootargs-phantom-emmc.txt
earlycon=aml_uart,0xfe078000,921600 console=ttyS0,921600 console=tty1 loglevel=8 jtag=apao
root=/dev/mmcblk1p2 rootfstype=ext4 rootwait
```

可以修改调试串口波特率

```
earlycon=aml_uart,0xfe078000,115200 console=ttyS0,115200 console=tty1 loglevel=8 jtag=apao
root=/dev/mmcblk1p2 rootfstype=ext4 rootwait
```

修改日志输出等级

```
earlycon=aml_uart,0xfe078000,921600 console=ttyS0,921600 console=tty1 loglevel=1 jtag=apao
root=/dev/mmcblk1p2 rootfstype=ext4 rootwait
```

4.16 使用 dtoverlay

睿莓 1 支持 Device Tree overlay 功能，通过配置 boot.conf 文件实现。

4.16.1 dtoverlay 配置说明

overlay 是通过应用于 base dtb 以扩展或修改它的补丁。它们以.dtbo 文件的形式存储在 overlay 子目录中，系统将从<load_prefix>overlays/目录加载.dtbo 文件。
系统默认的 base dtb 为 phantom.dtb，默认 overlay 路径为/boot/linux/overlays/。

通过配置 boot.conf 文件可以添加要支持的 dtoverlay

```
sudo nano /boot/boot.conf
```

示例

```
dtoverlay=<overlay_name>,<param=value>
```

它等同于

```
dtoverlay=<overlay_name>
dtparam=<param=value>
```

打开 dtdebug 调试信息输出

```
dtdebug=1
```

使能 dtdebug 调试信息输出后，您将可以通过调试串口看到启动时 dtoverlay 的加载信息。

NOTE:修改完 boot.conf 文件后请先执行 sync 指令以将修改同步 FLASH 存储介质，然后再重启设备。

4.16.2 目前支持的 dtoverlay

- 使能板载的 uartA 串口

```
[board_type=1]
dtb=phantom.dtb
dtoverlay=uartA
```

使能 uartA 后，它对应/dev/ttyS1 设备。

NOTE: uartA 默认作为调试串口，使能 uartA 后，它将作为普通串口而不再作为调试串口。

- 使能板载的 uartC 串口

```
[board_type=1]
dtb=phantom.dtb
dtoverlay=uartC
```

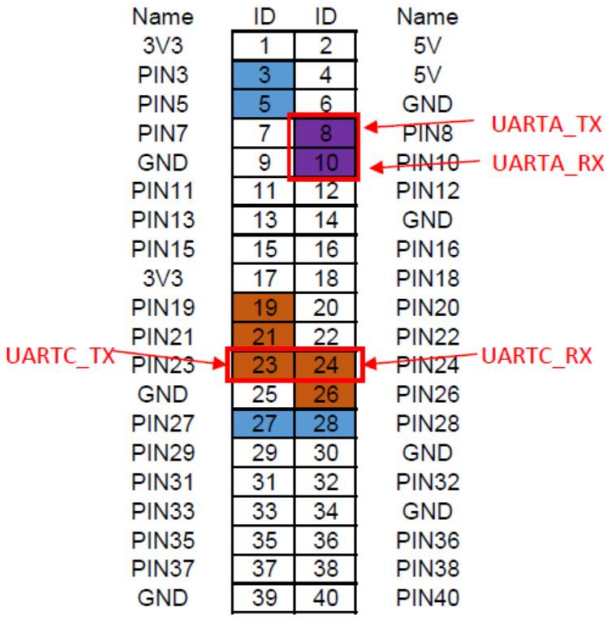
使能 uartC 后，它对应/dev/ttyS2 设备。

NOTE: 睿莓 1 的 uartC 与 spi 的管脚冲突，使能 uartC 串口后将不能使用 spi。

- 同时使能 uartA 和 uartC

```
[board_type=1]
dtb=phantom.dtb
dtoverlay=uartA
dtoverlay=uartC
```

uartA 和 uartC 的管脚定义：



- 使能基于 SC16IS752 i2c 方式的串口扩展

```
[board_type=1]
dtb=phantom.dtb
dtoverlay=sc16is752-i2c,int_pin=72,addr=0x48
```

int_pin对应实际连接的中断GPIO管脚，addr对应扩展板实际的i2c设备地址。

Serial Expansion HAT为基于SC16IS752的串口扩展板，它通过I2C接口扩展2路串口和8路GPIO。
Serial Expansion HAT扩展板使用i2c-1，扩展板可以直接与睿莓1的40PIN连接使用。扩展的两路串口设备分别为/dev/ttySC0和/dev/ttySC1，以及gpiochip1

- 使能基于 SC16IS752 spi 方式的串口扩展

```
[board_type=1]
dtb=phantom.dtb
dtoverlay=sc16is752-spi0,int_pin=77
```

int_pin 对应实际连接的中断 GPIO 管脚

2-CH RS232 HAT为采用SC16IS752+SP3232方案的双通道隔离型RS232扩展板。

NOTE: 2-CH RS232 HAT 扩展板的管脚与睿莓 1 的 40PIN 不兼容，直接插入扩展板将不能使用，请使用杜邦线连接睿莓 1 的 spi0 接口使用。

NOTE: 使用基于 SC16IS752 spi 方式的串口扩展 overlay 时，请不要使能 uartC，因为 uartC 会禁用 spi0 接口。

基于 SC16IS752 方案的扩展板支持堆叠，当您同时配置基于 sc16is752 的多个 dtoverlay 时，注意定义的中断管脚 int_pin 不能相同。扩展的多路串口设备分别为/dev/ttySC0、/dev/ttySC1、/dev/ttySC2、/dev/ttySC3，依次类推。

您可以通过如下指令查看每个扩展板对应的串口和 gpio：

```
sudo cat /sys/kernel/debug/gpio
```

```
gpiochip2: GPIOs 409-416, parent: spi/spi0.0, spi0.0, can sleep:

gpiochip1: GPIOs 417-424, parent: i2c/1-0048, 1-0048, can sleep:

gpiochip0: GPIOs 425-511, parent: platform/fe000000.apb4:pinctrl@4000, periphs-banks:
```

gpiochip0 为板载的原生 gpio，gpiochip1 对应设备地址为 0x48 的 I2C 扩展的 GPIO，gpiochip2 对应 SPI 扩展的 GPIO。根据 gpiochip 的编号，可以确定/dev/ttySC0、/dev/ttySC1 为设备地址为 0x48 的 I2C 扩展的串口，/dev/ttySC2、/dev/ttySC3 为 SPI 扩展的串口。

5 操作系统安装

5.1 镜像下载

在出厂时我们已经在 eMMC 中烧录了系统，用户可以直接使用。
我们提供了出厂镜像，如果系统恢复出厂设置，请点击以下链接下载出厂镜像。

镜像下载地址：https://www.123pan.com/s/vjW7Vv-w9Q7A.html

5.2 系统烧录

睿莓 1 支持 SD 卡和 eMMC 系统双启动，SD 卡启动的优先级更高。

如果希望烧写系统到 eMMC，需要系统已通过 SD 卡启动，然后通过 dd 命令间接烧录 eMMC。

5.2.1 烧录 SD 卡

安装烧录工具，推荐 balenaEtcher：

- balenaEtcher：https://www.balena.io/etcher/
- SD 卡: 至少使用 8GB 以上容量的 SD 卡(如果打算使用 SD 卡烧录 eMMC，SD 卡容量最低要求 16GB 以上)

烧录步骤：

1. 打开 balenaEtcher，选择要烧录的文件
2. 选择要烧录的 SD 卡
3. 等待烧录完成

启用 SSH：

镜像默认禁能 SSH 功能，如果希望开机后使用 SSH 远程连接到设备，则需要在开机前在 boot 分区下创建一个名为 ssh 的空文件，以保证设备开机后自动使能 SSH 功能。

5.2.2 烧录 eMMC

eMMC 目前仅支持从 SD 卡烧录，出厂时默认已经在 eMMC 中烧录好镜像，用户可以直接使用，如果发现设备无法启动，绿色指示灯不闪烁，则说明此时系统无法启动，需要使用 SD 卡将镜像烧入 eMMC 中。

```
lsblk

NAME             MAJ:MIN RM    SIZE RO TYPE MOUNTPOINT
mmcblk0          179:0      0 14.8G    0 disk
 ├──mmcblk0p1    179:1      0    256M    0 part /boot
 └──mmcblk0p2    179:2      0 14.6G    0 part /
mmcblk1          179:32     0    7.3G    0 disk
 └──mmcblk1p1    179:33     0    7.3G    0 part
mmcblk1boot0 179:64     0      4M    0 disk
mmcblk1boot1 179:96     0      4M    0 disk
```

SD 卡的分区名为 mmcblk0，可以看见 SD 卡有两个分区，一个是 mmcblk0p1，另一个是 mmcblk0p2。第二个为 eMMC 的分区，因为默认没有烧录系统所以只有一个分区 mmcblk1p1。
如果第二个分区已经烧录系统，那么使用 lsblk 命令后会显示以下内容

```
lsblk

NAME             MAJ:MIN RM    SIZE RO TYPE MOUNTPOINT
mmcblk0          179:0      0 14.8G    0 disk
 ├──mmcblk0p1    179:1      0    256M    0 part /boot
 └──mmcblk0p2    179:2      0 14.6G    0 part /
mmcblk1           179:32     0    7.3G    0 disk
 ├──mmcblk1p1    179:33     0    256M    0 part
```

```
└─mmcblk1p2  179:34   0  5.9G  0 part
mmcblk1boot0 179:64   0    4M  0 disk
mmcblk1boot1 179:96   0    4M  0 disk
```

烧录准备

eMMC 烧录只能通过 SD 卡写入，所以首先需要一张已经烧录好睿莓系统的 SD 卡，启动系统，将即将烧录的系统放入 SD 卡中，示例中将镜像直接放在默认用户 phantom 的文件夹下，文件夹绝对路径为 /home/phantom。

烧录系统到 eMMC：

```
sudo dd if=<镜像路径> of=/dev/mmcblk1 bs=4MiB
#示例
sudo dd if=/home/phantom/phantom_2022-12-03.img of=/dev/mmcblk1 bs=4MiB
sync
```

耐心等待命令执行完毕。
执行完毕后会显示以下内容：

```
1483+1 records in
1483+1 records out
```

使用 lsblk 可以看到 mmcblk1 拥有 p1，p2 两个分区：

```
lsblk

NAME         MAJ:MIN RM  SIZE  RO TYPE MOUNTPOINT
mmcblk0      179:0    0  14.8G  0 disk
├─mmcblk0p1  179:1    0  256M   0 part  /boot
└─mmcblk0p2  179:2    0  14.6G  0 part  /
mmcblk1      179:32   0  7.3G   0 disk
├─mmcblk1p1  179:33   0  256M   0 part
└─mmcblk1p2  179:34   0  5.9G   0 part
mmcblk1boot0 179:64   0  4M     0 disk
mmcblk1boot1 179:96   0  4M     0 disk
```

启用 SSH：

默认镜像没有使能 SSH 服务，如果希望开机即可使用 SSH 远程连接到设备，则按照以下步骤操作：

```
sudo mount /dev/mmcblk1p1 /mnt
sudo touch /mnt/ssh
sudo umount /mnt
```

6 故障排除

6.1 HDMI 无显示

上电后 HDMI 可能会出现无显示的情况，此时首先检查屏幕连接是否正确，然后可以直接使用 SSH 登录界面(如何知道设备 IP 地址)，然后使能桌面服务，查看是否有画面显示。

```
sudo systemctl start weston.service
```

6.2 开机绿灯常亮设备无法启动

这种情况基本因为 eMMC 中没有镜像的同时 SD 卡中也没有可用系统，此时应参考操作系统安装烧录 SD 卡，或者为 eMMC 烧录一个系统。

6.3 SSH 不可用

由于系统默认禁能 SSH 功能，如果希望使用 SSH 则参考使用 SSH 连接到设备。

7 FAQ

7.1 默认用户名密码

用户名：phantom
密码： phantom

7.2 是否支持 docker 服务

最新的系统镜像支持 docker 服务。

7.3 是否预安装 Linux Header 包

最新系统已预安装 Linux Header 包，请不要使用 apt install 的方式安装 Linux Header 包。

8 Known Issues

目前睿莓 1 系统还在不断优化中，目前我们已知有如下问题：

#	分类	描述	问题说明
1	HDMI	连接部分 HDMI 显示器无图像输出	目前不支持部分非标 HDMI 显示器
2		屏幕自动息屏后不能通过鼠标或键盘唤醒	目前需要手工禁用自动息屏功能。

9 关于我们

9.1 关于 EDATEC

EDATEC 位于上海，是 Raspberry Pi 的全球设计合作伙伴之一。我们的愿景是提供基于 Raspberry Pi 技术平台的物联网、工业控制、自动化、绿色能源和人工智能的硬件解决方案。

我们提供标准的硬件解决方案，定制设计和制造服务，以加快电子产品的开发和上市时间。

9.2 联系方式

邮箱 – sales@edatec.cn / support@edatec.cn
手机 – +86-18621560183
网站 – https://www.edatec.cn
地址 – 上海市嘉定区嘉罗公路 1661 号 24 栋 301 室