

Tinker Board 2S Manual

入门

简介

- Tinker Board 2 搭载 64 位 Armv8 架构的 Rockchip RK3399 处理器芯片，相较于其他热门 SBC 主板，性能大幅提升。
- 另外还采用 Arm big.LITTLE 技术。这是一种异构处理技术，可以将双核 2.0 GHz Cortex-A72 和四核 1.5 GHz Cortex-A53 整合在同一个 SoC 中，配合 big.LITTLEzido 自动任务分配软件，可以确保每个程序都能使用正确的 CPU 核心，从而达到更具竞争力的处理速度。

开发板版本

- 开发板有三个版本可选：
 - Tinker Board 2 基础版，不带 eMMC，2G 内存。
 - Tinker Board 2S 升级版，带 16GB eMMC，2G 内存。
 - Tinker Board 2S 升级版，带 16GB eMMC，4G 内存。
- 如果您购买的是 Tinker Board 2 基础版，不带 eMMC，直接查看 TF 卡烧录镜像部分和登录即可。

安装系统

下载官方镜像

- Tinker-board 官方为 Tinker Board 2S 提供了两个版本的系统镜像，分别是 Debian 和 Android。
- 你可以直接到 [tinker-board 官网](#) 下载最新的官方镜像。

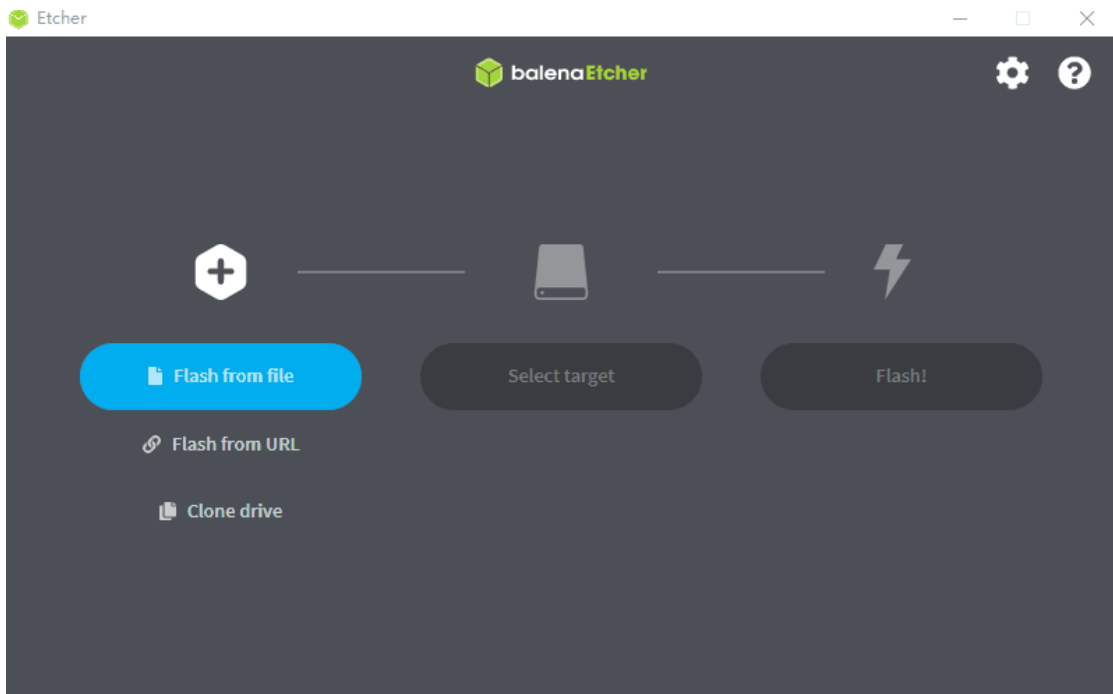
在 TF 卡上安装镜像

器材准备

1. TF 卡读卡器。
2. 至少 8GB 的 TF 卡。

镜像烧录

- 避免在烧录镜像过程中出错，烧录镜像前需要使用 [Panasonic SDFormatter-SD 卡格式化软件](#) 软件格式化 TF 卡。
 - 格式化的时候特别注意，如果你的电脑有插入了别的移动硬盘，不要格式化错驱动器了。
1. 下载并安装 [balenaEtcher-烧录镜像软件](#)。
 2. 打开 balenaEtcher 烧录软件，将 TF 卡插入读卡器，将读卡器插入电脑。
 3. 选择下载好的镜像文件，单击“写入”并等待写入完成。



在 EMMC 上安装镜像 (适用 Tinker Board 2S)

器材准备

1. Tinker Board 2S 主板。

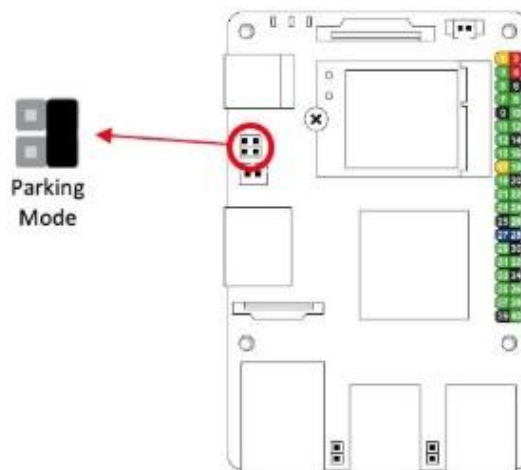
2. Type A 转 Type C 双公口 USB3.0 线。
3. 12~19V DC 接口 (5.5/2.5 mm DC 接口)电源适配器。

准备工作

1. 下载 RK 驱动助手 [DriverAssitant-RK 驱动助手](#)和 [balenaEtcher-烧录镜像软件](#)。
2. 打开 RK 驱动助手 DriverAssitant 安装 USB 驱动程序，此过程无需连接 Tinker Board 2S，安装完成后重启电脑。



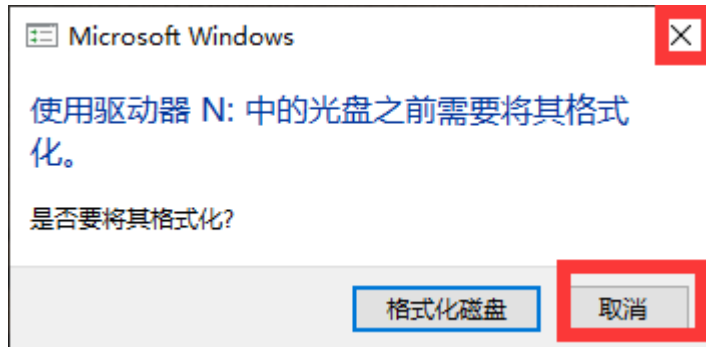
3. 确保跳线帽已经安装在在如图两根排针上（Tinker Board 2S 上默认安装黑色跳线帽）。



4. 开发板上如果有 TF 卡，要先移除 TF 卡。默认启动 TF 卡镜像。
5. 将 Type C 线一端连接电脑 USB 端口，另一端连接 Tinker Board 2S 的 Type C 接口。

6. 给 Tinker Board 2S 主板接通电源，电脑会提示你是否需要格式化磁盘，**点击取消或者关闭**

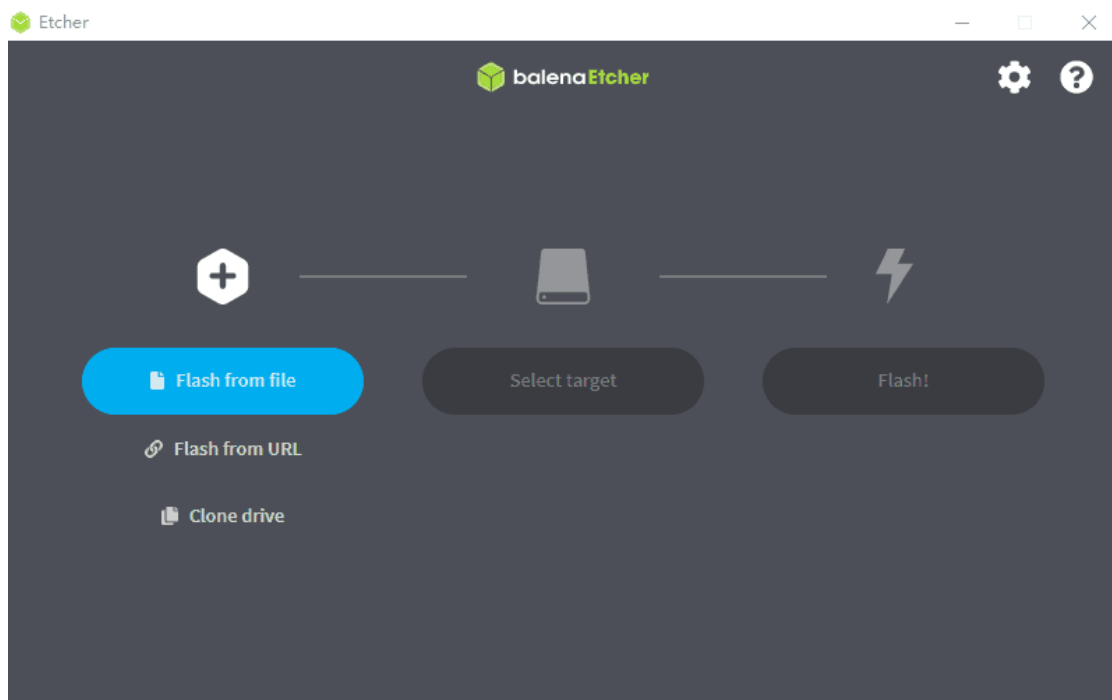
窗口，千万不要格式化磁盘！会把板子的 uboot 也一并格式化掉！



镜像烧录

1. 用管理员身份运行 balenaEtcher，点击 Flash from file 选择对应的 Debian 镜像。

2. 点击 Select target 选镜像择对应的路径如下图，然后点击 flash 开始烧录。



登录

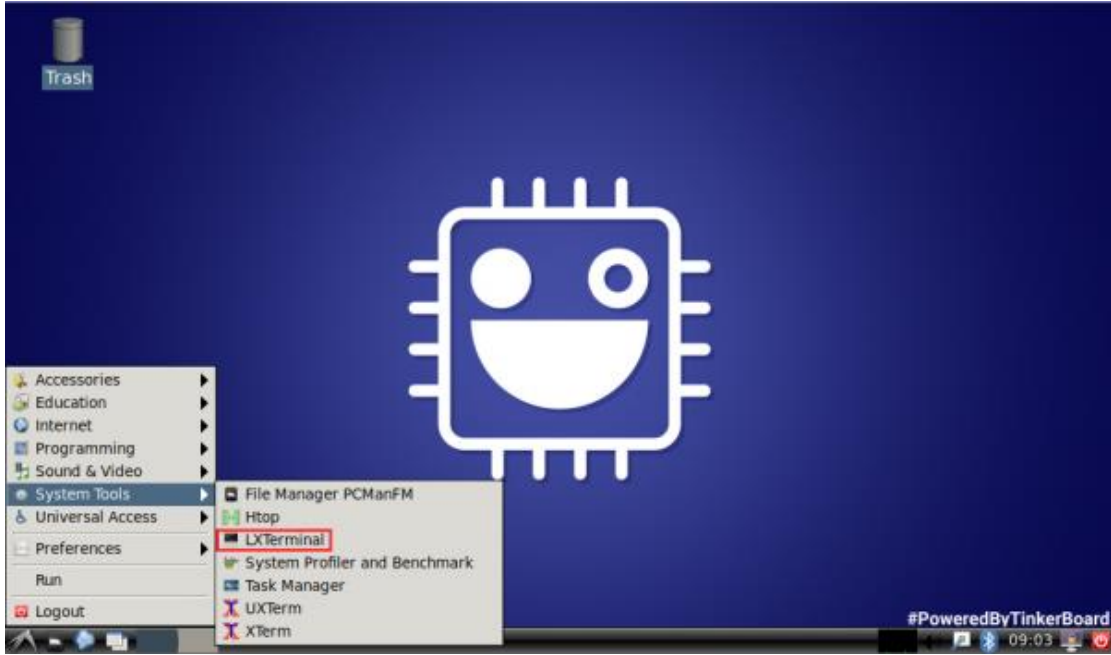
- Debian 默认用户帐户（非 root 用户）

- 登录名: linaro

登录密码: linaro

本地登录

- 如果你使用的是屏幕和键盘来控制 Tinker Board 2，需要知道如何启动终端。



远程登录

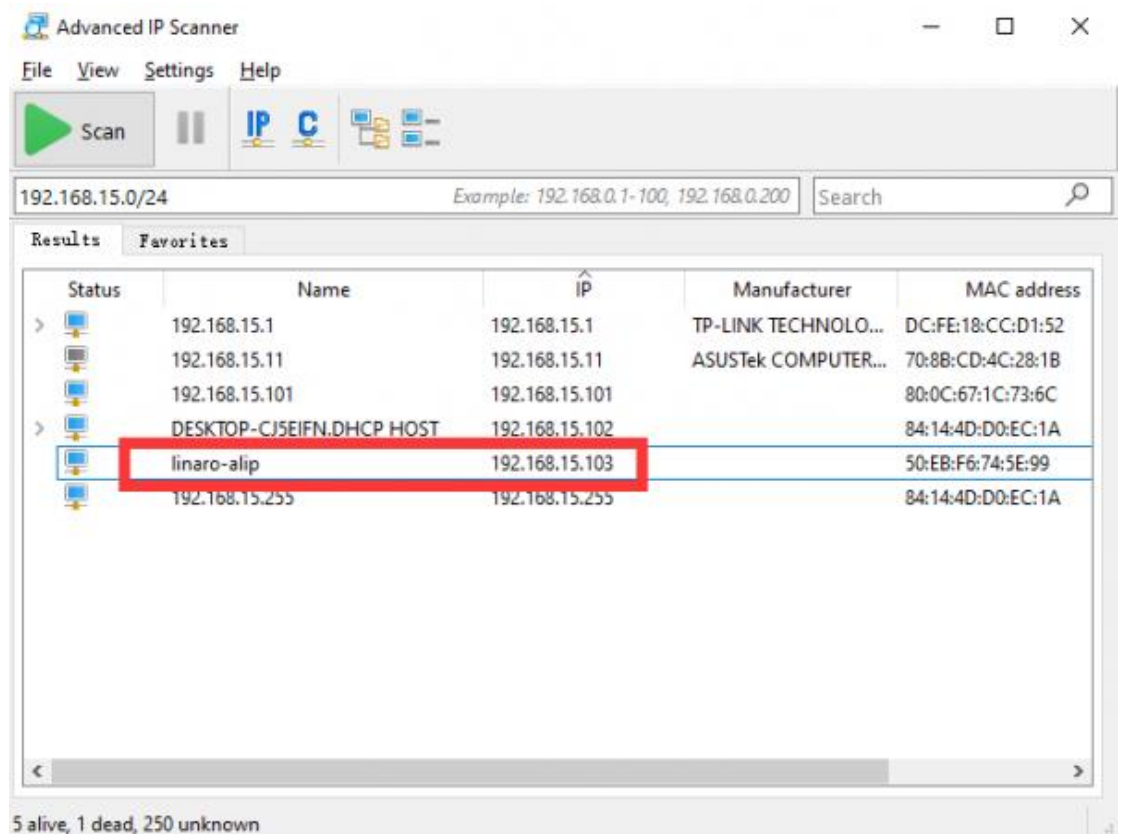
准备工作

- 用一根网线一端连接 Tinker Board 2，另一端连接路由器的 LAN 端口。
- 确保 Tinker Board 2 与你的电脑出于一个路由器下或同一网段。

获取 Tinker Board 2 的 IP 地址

- 方法一：登录路由器查找 Tinker Board 2 的 IP 地址。
- 方法二：你可以通过一些局域网 IP 扫描工具，这里以 [Advanced IP Scanner](#) 为例程
 1. 运行 Advanced IP Scanner
 2. 点击 Scan 按钮，扫描当前局域网内的 IP 地址

3. 找到**所有** Manufacturer 中有 linaro-alip 字样的 IP 地址并记录



4. 将设备上电，并确保设备连接上网络后

5. 重新点击 Scan 按钮，扫描当前局域网内的 IP 地址

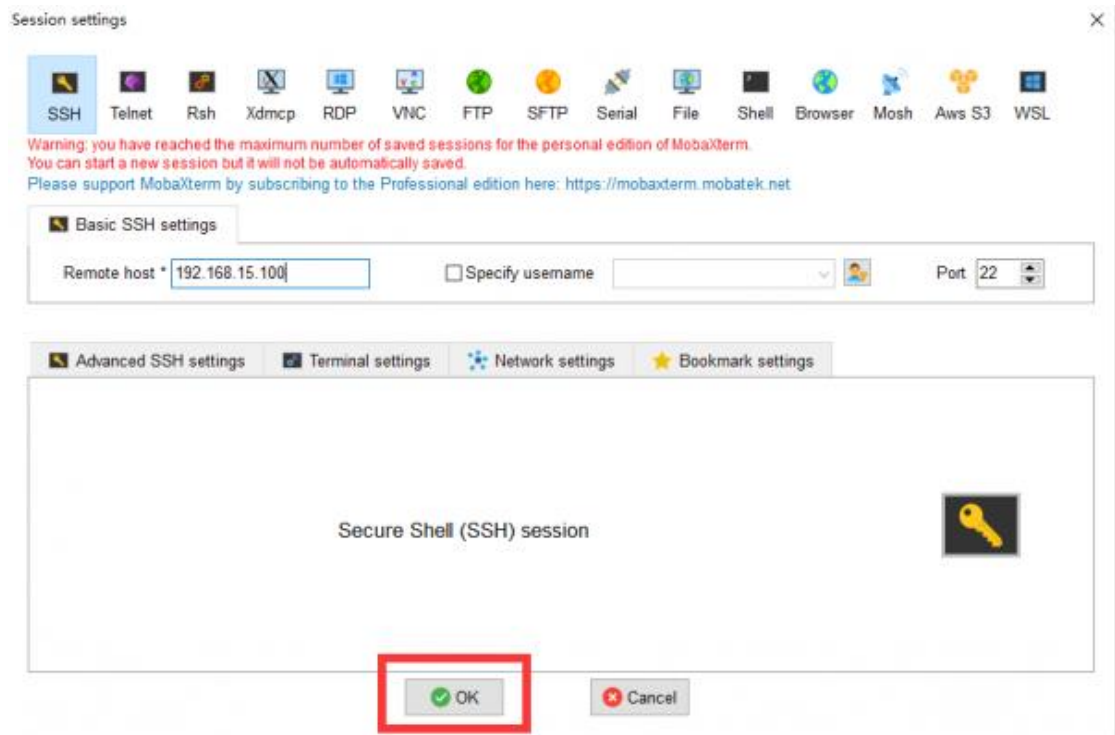
6. 排除掉**所有先前记录的** Manufacturer 中有 linaro-alip 字样的 IP 地址，剩下的就是你的 linaro-alip IP 地址了

使用 MobaXterm 登录

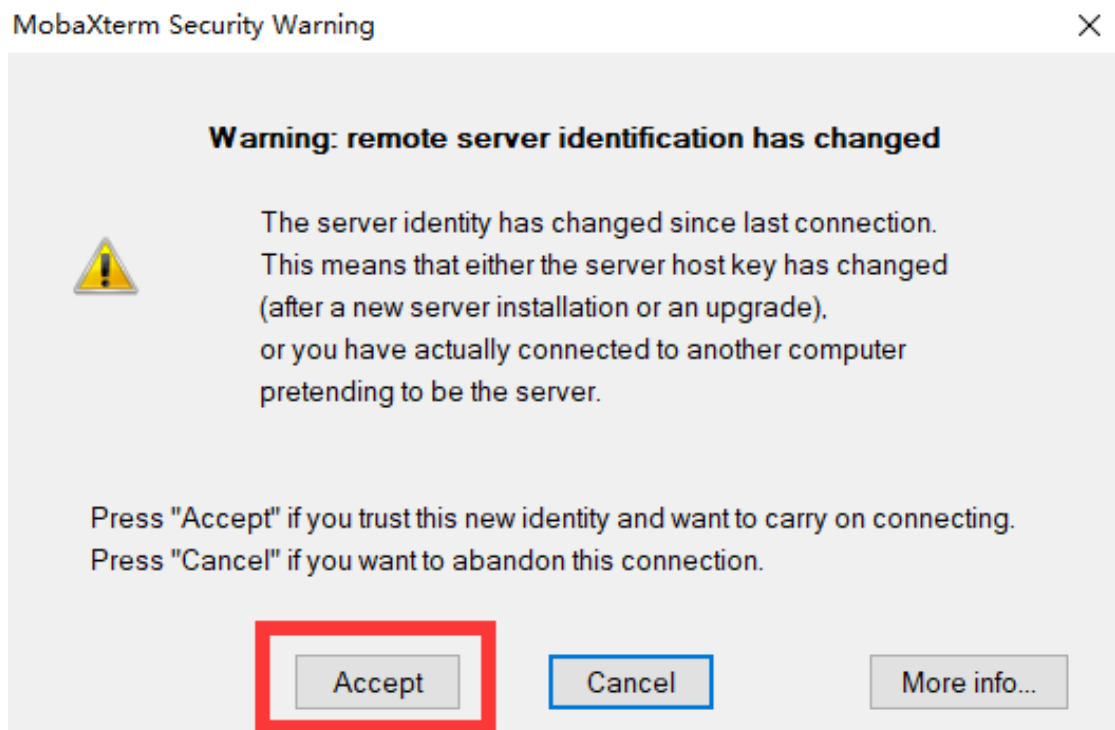
终端窗口

1. 下载 [MobaXterm](#) 远程登录软件,解压即可使用。
2. 打开 MobaXterm 远程登录软件，选择 Session,选择 ssh。

3. 在 Remote host 输入我们前面查询到的 IP 地址 192.168.15.100（根据自己的实际 IP 来填写），填写完成后，点击 ok。



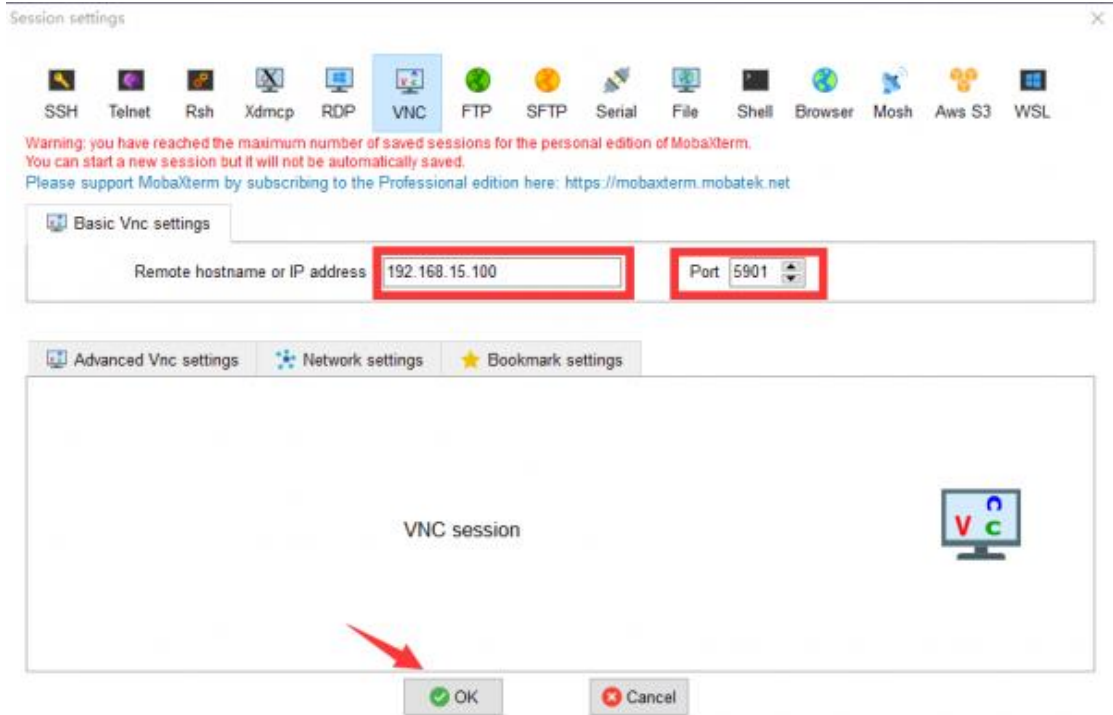
4. 点击 accept，输入登录名：linaro，回车输入登录密码：linaro（在输入密码时，屏幕没有变化属于正常现象，点击回车确认即可）



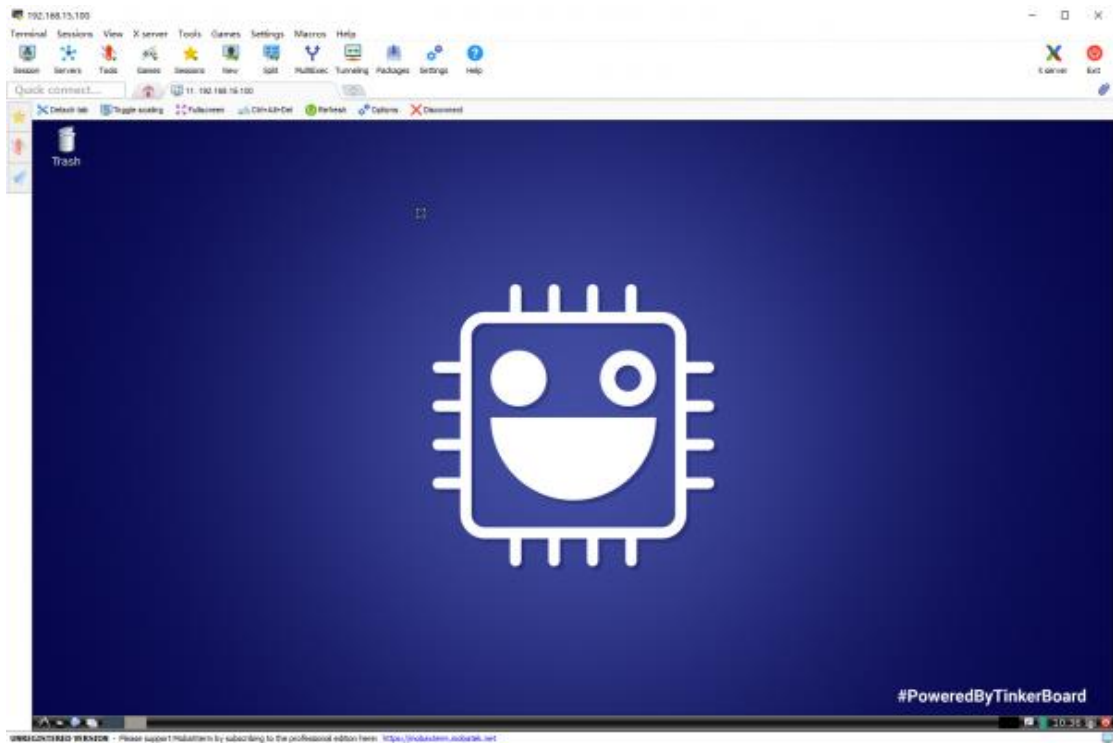
远程桌面

1. 打开 MobaXterm 远程登录软件，选择 Session,选择 vnc。

2. 在 Remote host 输入我们前面查询到的 IP 地址 192.168.15.100（根据自己的实际 IP 来填写），填写完成后，点击 ok。



3. 点击 ok，回车输入登录密码：linaro



使用 VNC 登录

Tinker Board 2 上开启 VNC Server

1. 在 Tinker Board 2 下载 VNC Server，可以参考配置部分 [Interfacing Options](#)。

2. 下载安装完成后, 开启 VNC Server ,需要设置一个大于 6 位的登录密码。(输入密码时屏幕上没有变化属于正常现象, 密码输入完成后按回车键确认即可)

```
[vnc] Please Input vnc password more than six words.
Using password file /root/.vnc/passwd
VNC directory /root/.vnc does not exist, creating.
Password:
Verify:
Would you like to enter a view-only password (y/n)? n
[vnc] vnc passwd success.
```

3. 执行完毕后就自动创建了服务器 5901 端口, 回车确认。

```
The VNC Server is enabled 5901 Port.

<Ok>
```


电脑上安装 VNC Viewer

1. 在电脑上下载安装 [VNC-Viewer](#)。

使用 VNC Viewer 登录 Tinker Board 2

1. 打开 VNC Viewer,输入 Tinker Board 2 的 IP 地址和端口号回车确认。例如:

```
192.168.15.100:5901
```

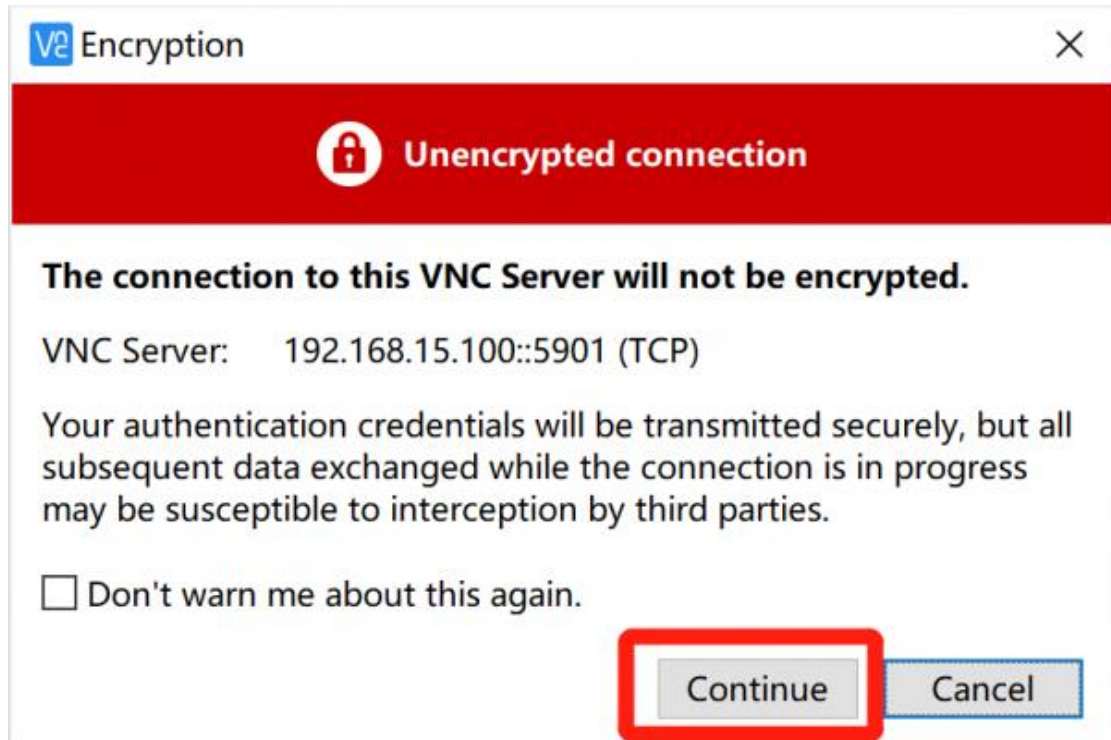
 VNC Viewer

File View Help

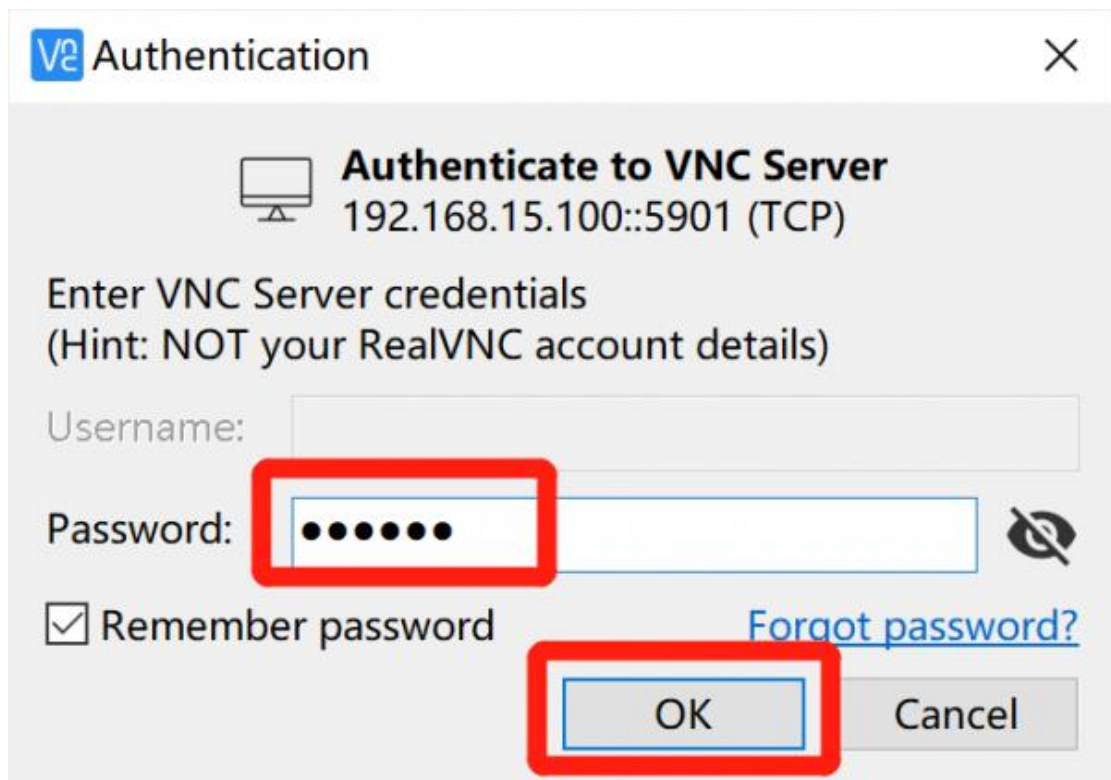

by RealVNC

192.168.15.100:5901

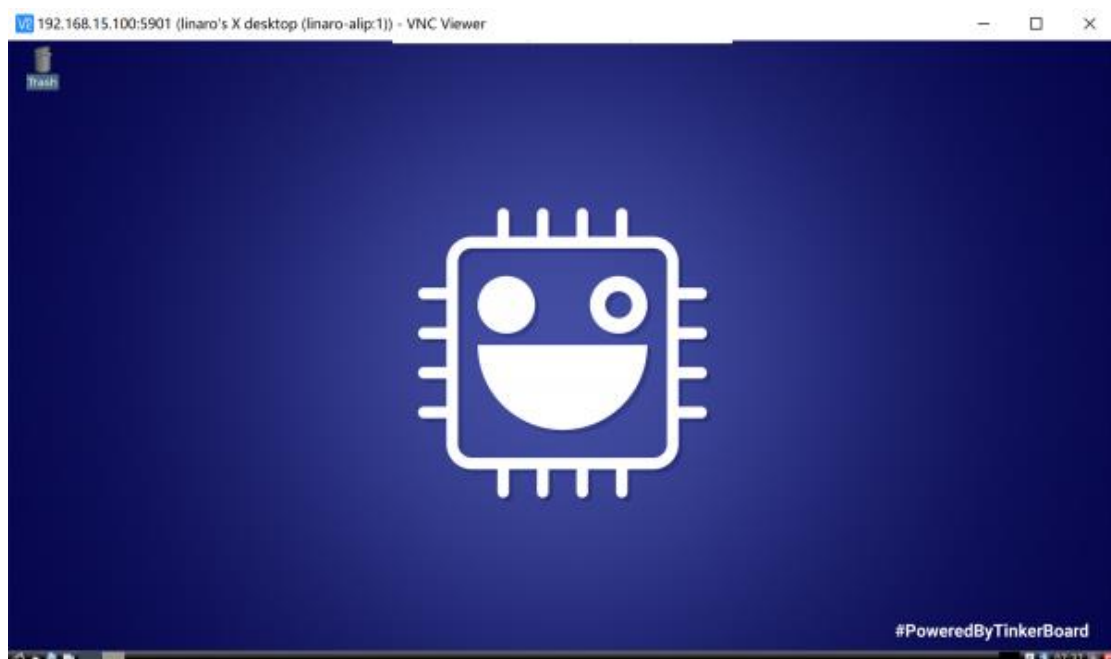
2. 点击 continue:



3. 输入前面设置的 VNC 登录密码，点击 ok:



4. 此时就成功登录 Tinker Board 2 了。



Linux 操作基础

Linux

- Linux 是一种通用的操作系统,它的网络功能非常强大,对内存等硬件的消耗也小,多用于网络服务器中。

- 极其稳定、极其安全、可移植性、可扩展性强。
- 具有开放源码、没有版权、技术社区用户多等特点，开放源码使得用户可以自由裁剪，灵活性强，功能强大，成本低。
- 缺点入门比 Windows 难，由于缺少桌面周边生态，使用 Linux 桌面系统的用户也非常少。

为什么学习 Linux

- Linux 系统被广泛应用在人们的日常生活用品中，如手机、智能家居、汽车电子、可穿戴设备等，只不过很多人不知道自己使用的电子产品里面运行的是 Linux 系统，例如我们用的安卓手机系统就是基于 Linux 内核开发的。
 - 全球 100 万个顶尖领域中超过 90% 在使用 Linux 系统；全球大部分的股票交易市场是基于 Linux 系统部署的，包括纽交所、纳斯达克等；
 - 全球知名的淘宝、亚马逊、易趣、沃尔玛等电子商务平台都在使用 Linux 系统。
 - 参与 Linux 内核开发的开发人员和公司也是最多的、最活跃的。因此了解和学习 Linux 是十分重要的。

终端

- 计算机上的终端允许用户对其系统进行大量控制。Windows 的用户可能已经接触过 Command Prompt Powershell，而 mac OS 的用户可能已经熟悉了 Terminal。所有这些工具都允许用户通过使用命令直接操作他们的系统。

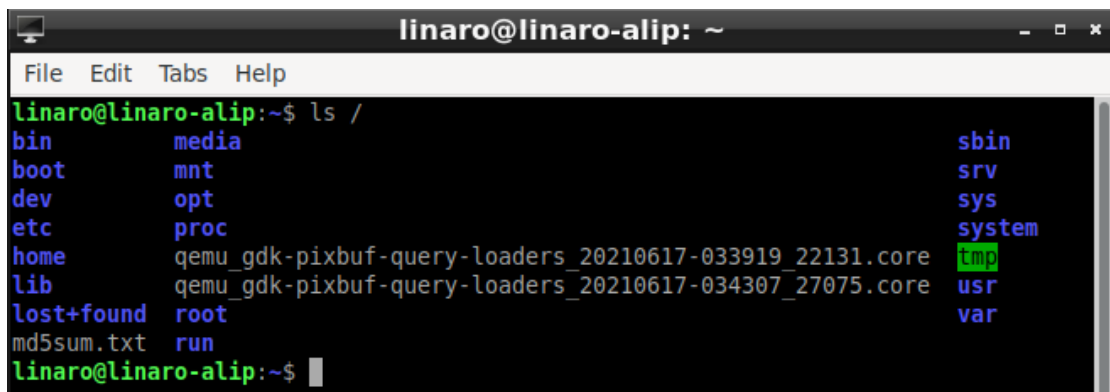


- linaro 当前用户名/登录名
- linaro-alip 默认主机名
- ~ 当前用户所在的目录是/home/linaro
- \$ 字符表示当前登录的是普通用户

- # 字符表示登录的是 root 用户
- 在终端窗口中键入命令，然后按键盘上的回车键运行命令。

Linux 系统目录结构

- 在 Windows 中每一个分区都是一个树形结构，有多少个分区就有多少个树形结构，而 Linux 中只有一个树形结构，所有的文件、分区都是存在于一个树形结构中。在这个结构中,最上层的是根目录，其他所有的目录、文件、分区都是在根目录下建立的。读者可以通过 `ls /` 命令来查看整个根目录的文档,如图所



```
linaro@linaro-alip: ~  
File Edit Tabs Help  
linaro@linaro-alip:~$ ls /  
bin          media          sbin  
boot         mnt            srv  
dev          opt            sys  
etc          proc           system  
home         qemu_gdk-pixbuf-query-loaders_20210617-033919_22131.core tmp  
lib          qemu_gdk-pixbuf-query-loaders_20210617-034307_27075.core usr  
lost+found  root           var  
md5sum.txt  run  
linaro@linaro-alip:~$
```

示。

- 以下是对这些目录的解释：

常用目录

- /home: 用户目录。
 - 除了 root 用户外，其他所有的使用者的数据都存放在这个目录下，在 Tinker Board 2 的系统中，/home 目录中有一个 linaro 的子目录,这个就是 linaro 用户的默认目录。
- /bin: 主要放置系统的必备执行文件目录。
 - 放置与 Linux 系统有关(包括运行图形界面所需的)的二进制可执行文件,如 `ls`、`mkdir`、`rm` 等。
- /boot: 引导目录。

- 用于存放系统引导程序，如 linux 内核以及启动配置文件，里面的 config.txt 也是用户配置使用频率最高的文件。
- /etc: 系统配置文件存放目录。
 - 在这个目录中存放了几乎所有的 Linux 系统软件所需的配置文件，如果需要对这个目录中的文本进行修改,那么最好是先将需要修改的文件进行备份，以保证在修改之后还可以回到原来的状态。

其它目录

- /dev : 设备目录。在 Linux 系统中，所有设备都视为文件，而在这个目录中存放了所有设备，例如第一个 SATA 硬盘或 U 盘会被识别为 sda 文件，而 SATA 硬盘或 U 盘第一个分区会被识别为 sda1 文件。
- /lib: 基本系统的动态链接库存放位置，在这个目录中存放了可以维持一个基本系统启动所需要的库文件。如果没有这个目录，系统程序根本就无法工作。
- /lost + found/分区系统的目录。系统非正常关机而产生的文件通常都存放在这里，另外由 fsck 等程序进行硬盘修复后产生的文件也存放在这里。
- /media: linux 系统会自动识别一些设备，例如 U 盘、光驱等等，当识别后，Linux 会把识别的设备挂载到这个目录下。
- /mnt: 传统的外部设备挂载点。在早期的时候，除了系统分区外的其他分区，如 U 盘等设备，都会被挂载到这个目录下以供用户读写。不过现在已经被 /media/取代。
- /proc: proc 是一个虚拟文件系统。这个目录是存放在内存中的，因此不会占用硬盘空间，系统或用户通过读取这些设备来了解它们的信息。例如，可以使用 cat /proc/cpuinfo 命令来查看 CPU 信息。
- /root: 该目录为系统管理员，也称作超级权限者的用户主目录。
- /srv: 该目录存放一些服务启动之后需要提取的数据。

- /sys: 与/proc 目录一样, 也是一个虚拟目录, 是由内核中的 sysfs 系统来实现的, 其作用与 proc 有些类似, 但除了与 proc 具有相同的查看和设定内核参数功能之外, 还有为 Linux 统一设备模型作为管理之用。
- /tmp: 临时目录。由程序所产生的临时文件都会存放在该目录下, 不用担心这个目录会占用太多的空间, 因为每次系统启动都会清除这个目录的内容; 同时这也是系统为数不多的拥有所有用户可读写属性的目录。
- /usr: Linux 系统所安装的程序都是存放在该目录中的, 类似于 windows 下的 program files 目录。
- /run: 用于存放系统启动时描述系统信息的文件。这个目录最初是在/var/目录下的, 但是现在被提升到根目录下。

重要目录

- 在 Linux 系统中, 有几个目录是比较重要的, 平时需要注意不要误删除或者随意更改内部文件。
- /etc: 这个是系统中的配置文件, 如果你更改了该目录下的某个文件可能会导致系统不能启动。
- /bin, /sbin, /usr/bin, /usr/sbin: 这是系统预设的执行文件的放置目录, 比如 ls 就是在/bin/ls 目录下的。
- /bin, /usr/bin 是给系统用户使用的指令 (除 root 外的普通用户), 而/sbin,/usr/sbin 则是给 root 使用的指令。
- /var: 这是一个非常重要的目录, 系统上跑了很多程序, 那么每个程序都会有相应的日志产生, 而这些日志就被记录到这个目录下, 具体在/var/log 目录下。

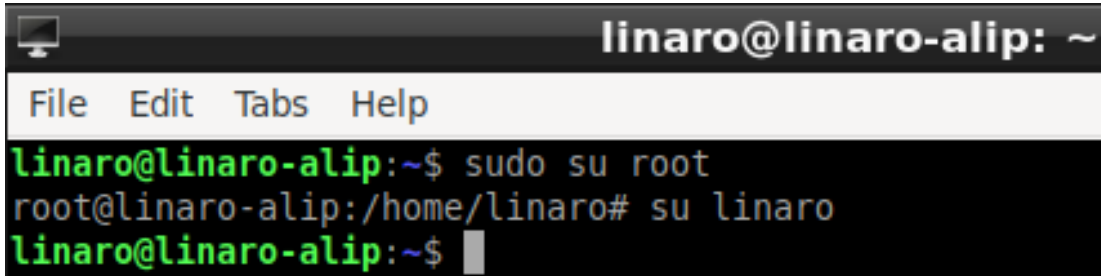
常见指令介绍

文件系统

sudo

- sudo 命令以系统管理者的身份执行指令。
- 要想使用 root 用户，可使用 linaro 用户登录，执行下面命令

```
sudo su #切换为超级用户  
su linaro #切换普通用户
```

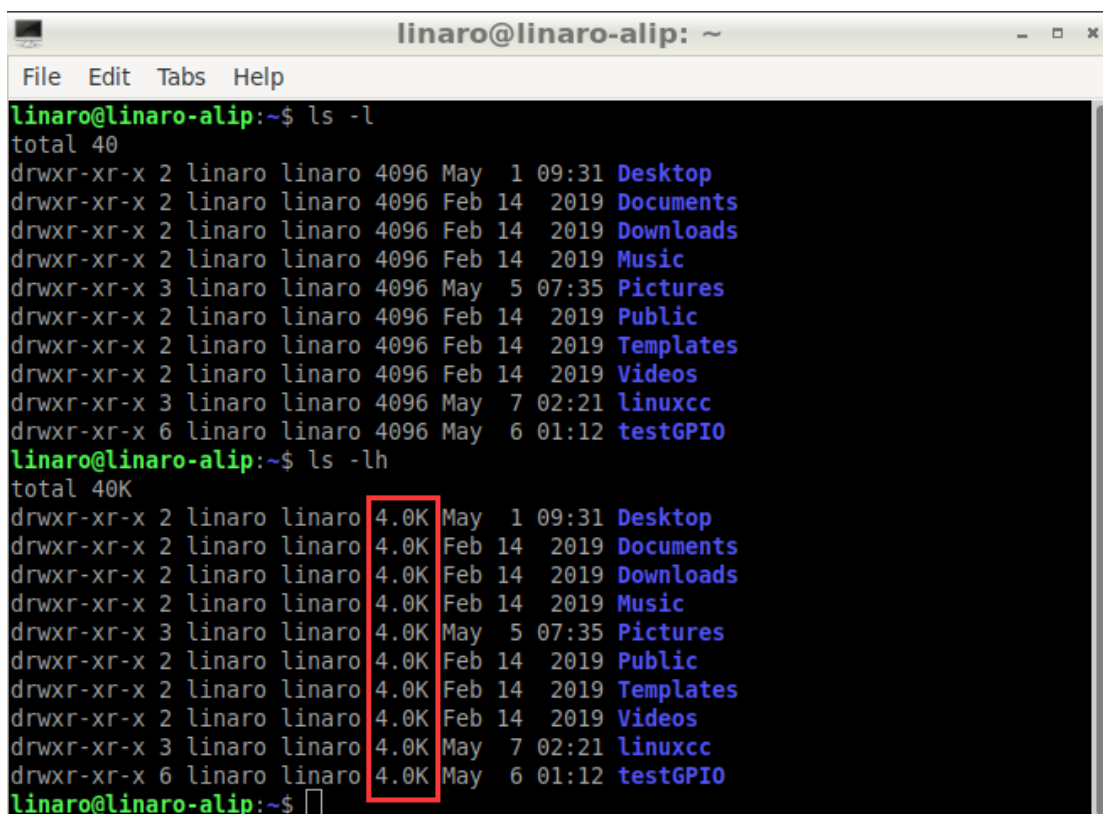


```
linaro@linaro-alip: ~  
File Edit Tabs Help  
linaro@linaro-alip:~$ sudo su root  
root@linaro-alip:/home/linaro# su linaro  
linaro@linaro-alip:~$
```

ls

- ls 命令用于显示指定工作目录下之内容（列出目前工作目录所含之文件及子目录）。
- 常用的指令：

- ls
- ls -a #显示所有文件及目录（. 开头的隐藏文件也会列出）
- ls -l #除文件名称外，亦将文件型态、权限、拥有者、文件大小等资讯详细列出
- ls -lh #文件大小以容易理解的格式列出，例如 4K



```
linaro@linaro-alip: ~  
File Edit Tabs Help  
linaro@linaro-alip:~$ ls -l  
total 40  
drwxr-xr-x 2 linaro linaro 4096 May 1 09:31 Desktop  
drwxr-xr-x 2 linaro linaro 4096 Feb 14 2019 Documents  
drwxr-xr-x 2 linaro linaro 4096 Feb 14 2019 Downloads  
drwxr-xr-x 2 linaro linaro 4096 Feb 14 2019 Music  
drwxr-xr-x 3 linaro linaro 4096 May 5 07:35 Pictures  
drwxr-xr-x 2 linaro linaro 4096 Feb 14 2019 Public  
drwxr-xr-x 2 linaro linaro 4096 Feb 14 2019 Templates  
drwxr-xr-x 2 linaro linaro 4096 Feb 14 2019 Videos  
drwxr-xr-x 3 linaro linaro 4096 May 7 02:21 linuxcc  
drwxr-xr-x 6 linaro linaro 4096 May 6 01:12 testGPIO  
linaro@linaro-alip:~$ ls -lh  
total 40K  
drwxr-xr-x 2 linaro linaro 4.0K May 1 09:31 Desktop  
drwxr-xr-x 2 linaro linaro 4.0K Feb 14 2019 Documents  
drwxr-xr-x 2 linaro linaro 4.0K Feb 14 2019 Downloads  
drwxr-xr-x 2 linaro linaro 4.0K Feb 14 2019 Music  
drwxr-xr-x 3 linaro linaro 4.0K May 5 07:35 Pictures  
drwxr-xr-x 2 linaro linaro 4.0K Feb 14 2019 Public  
drwxr-xr-x 2 linaro linaro 4.0K Feb 14 2019 Templates  
drwxr-xr-x 2 linaro linaro 4.0K Feb 14 2019 Videos  
drwxr-xr-x 3 linaro linaro 4.0K May 7 02:21 linuxcc  
drwxr-xr-x 6 linaro linaro 4.0K May 6 01:12 testGPIO  
linaro@linaro-alip:~$
```


- 想要学习了解指令更多参数，我们可以使用 help 指令来查看：

```
ls --help
```

chmod

- chmod 命令是控制用户对文件的权限的命令。
- Linux/Unix 的文件调用权限分为三级：文件所有者（Owner）、用户组（Group）、其它用户（Other Users）。

- 在下图中，显示了 Linux 根目录下的详细文件信息。在这些文件信息中，最重要的就是第一列，它详细描述了文件和目录的权限，而第三与第四列则显示了这个文件和目录属于哪一个用户或组。

```

linaro@linaro-alip: ~
File Edit Tabs Help
linaro@linaro-alip:~$ ls -lh /
total 24M
drwxr-xr-x  2 root root 4.0K Apr 30 07:04 bin
drwxr-xr-x  5 root root 1.0K Apr 29 01:16 boot
drwxr-xr-x 17 root root 3.8K May  7 07:24 dev
drwxr-xr-x 114 root root 4.0K May  6 10:21 etc
drwxr-xr-x  3 root root 4.0K Jun 17  2021 home
drwxr-xr-x 17 root root 4.0K Apr 30 07:04 lib
drwx----- 2 root root 16K Dec 22 07:27 lost+found
-rw-r--r--  1 root root 4.4M Jun 17  2021 md5sum.txt
drwxr-xr-x  2 root root 4.0K Jun 17  2021 media
drwxr-xr-x  2 root root 4.0K Jun 17  2021 mnt
drwxr-xr-x  2 root root 4.0K Jun 17  2021 opt
dr-xr-xr-x 249 root root  0 Jan  1  1970 proc
-rw-r--r--  1 root root 9.4M Jun 17  2021 qemu_gdk-pixbuf-query-loaders_20210617-033919_22131.core
-rw-r--r--  1 root root 9.4M Jun 17  2021 qemu_gdk-pixbuf-query-loaders_20210617-034307_27075.core
drwx----- 6 root root 4.0K May  6 10:57 root
drwxr-xr-x 20 root root 760 May  7 03:23 run
drwxr-xr-x  2 root root 4.0K Apr 30 07:04 sbin
drwxr-xr-x  2 root root 4.0K Jun 17  2021 srv
dr-xr-xr-x 14 root root  0 May  7 07:17 sys
drwxr-xr-x  4 root root 4.0K Dec 22 07:00 system
drwxrwxrwt 17 root root 4.0K May  7 06:36 tmp
drwxr-xr-x 10 root root 4.0K Feb 14  2019 usr
drwxr-xr-x 11 root root 4.0K Jun 17  2021 var
linaro@linaro-alip:~$

```

- Linux 的文件属性可以分为三种：只读 (r)、写 (w) 和可执行 (x)。但是上面的文件属性却分为 10 小格，这是因为除了第一格显示目录外，另外三组每组三格分别表示文件所有者权限、同一组内的权限以及其他用户权限。
 - 第一栏中如果显示 d,则表示这是一个目录；如果是链接文件，则在这里显示 l;如果是设备文件，则显示 c。
 - 第一个 rwx 栏位:-rwx----- 表示文件拥有者所拥有的权限。
 - 第二个 rwx 栏位:---rwx--- 表示同一工作组内用户权限。
 - 第三个 rwx 栏位: -----rwx 表示其他用户权限。
 - 例如：
 - rwx rwx rwx 表示无论哪个用户都可以对这个文件读写与执行。
 - rw- --- --- 表示只有文件拥有者有读写权限，但是没有执行权限。
 - rw -rw -rw 表示所有用户都有读写权。

- 符号模式**

- who(用户类型)

who	用户类型	说明
u	user	文件所有者
g	group	文件所有者所在组
o	others	所有其他用户
a	all	所用用户, 相当于 ugo

- operator(符号模式表)

Operator	说明
+	为指定的用户类型增加权限
-	去除指定用户类型的权限
=	设置指定用户权限的设置, 即将用户类型的所有权限重新设置

- permission 的符号模式表

模式	名字	说明
r	读	设置为可读权限
w	写	设置为可写权限
x	执行权限	设置为可执行权限
X	特殊执行权限	只有当文件为目录文件, 或者其他类型的用户有可执行权限时, 才将文件权限设置可执行
s	setuid/gid	当文件被执行时, 根据 who 参数指定的用户类型设置文件的 setuid 或者 setgid 权限
t	粘贴位	设置粘贴位, 只有超级用户可以设置该位, 只有文件所有者 u 可以使用该位

- 符号模式实例

1. 给 file 的所有用户增加读权限

```
chmod a+r file
```

2. 删除 file 的所有用户的执行权限

```
chmod a-x file
```

3. 给 file 的所有用户增加读写权限

```
chmod a+rw file
```

4. 给 file 的所有用户增加读写执行权限

```
chmod +rwx file
```

- 对 file 的所有者设置读写权限，清空该用户组和其他用户对 file 的所有权限（空格代表无权限）

```
chmod u=rw,go= file
```

- 对目录 waveshare 和其子目录层次结构中的所有文件给用户增加读权限，而对用户组和其他用户删除读权限

```
chmod -R u+r,go-r waveshare
```

• 八进制语法

- chmod 命令可以使用八进制数来指定权限。文件或目录的权限位是由 9 个权限位来控制，每三位为一组，它们分别是文件所有者（User）的读、写、执行，用户组（Group）的读、写、执行以及其它用户（Other）的读、写、执行。

#	权限	rwX	二进制
7	读 + 写 + 执行	rwX	111
6	读 + 写	rw-	110
5	读 + 执行	rx-	101
4	只读	r--	100
3	写 + 执行	-wx	011
2	只写	-w-	010
1	只执行	--x	001
0	无	---	000

- 例如：765 的解释如下：
 - 所有者的权限用数字表达：属主的那三个权限位的数字加起来的总和。如 rwX，也就是 $4+2+1$ ，应该是 7。
 - 用户组的权限用数字表达：属组的那个权限位数字的相加的总和。如 rw-，也就是 $4+2+0$ ，应该是 6。
 - 其它用户的权限数字表达：其它用户权限位的数字相加的总和。如 rx-，也就是 $4+0+1$ ，应该是 5。

- 常用的数字权限

- 400 -r----- 拥有者能够读，其他任何人不能进行任何操作；
- 644 -rw-r--r-- 拥有者都能够读，但只有拥有者可以编辑；
- 660 -rw-rw---- 拥有者和组用户都可读和写，其他人不能进行任何操作；
- 664 -rw-rw-r-- 所有人都可读，但只有拥有者和组用户可编辑；
- 700 -rwx----- 拥有者能够读、写和执行，其他用户不能任何操作；
- 744 -rwxr--r-- 所有人都能读，但只有拥有者才能编辑和执行；
- 755 -rwxr-xr-x 所有人都能读和执行，但只有拥有者才能编辑；
- 777 -rwxrwxrwx 所有人都能读、写和执行（该设置不建议使用）。

- 实例

- 给 file 的所有用户增加读权限,拥有者和组用户可编辑权限

```
sudo chmod 664 file
```

touch

- touch 命令用于修改文件或者目录的时间属性，包括存取时间和更改时间。若文件不存在，系统会建立一个新的文件。
- 例如，在当前目录下，使用该指令创建一个空白文件"file.txt"，输入如下命令：

```
touch file.txt
```

mkdir

- mkdir 命令用于创建目录。
- 在工作目录下，建立一个名为 waveshare 的子目录：

```
sudo mkdir waveshare
```

- 在工作目录下建立一个名为 waveshare/test 的目录。

```
sudo mkdir -p waveshare/test
```

- 若 waveshare 目录原本不存在，则建立一个。（注：本例若不加 -p 参数，且原本 waveshare 目录不存在，则产生错误。）

cd

- 切换当前工作目录。

- `cd ..` #返回上一层目录
- `cd /home/linaro` #进入/home/linaro 目录
- `cd` #返回用户目录

cp

- cp 命令主要用于复制文件或目录。

- 参数:

- -a: 此选项通常在复制目录时使用，它保留链接、文件属性，并复制目录下的所有内容。其作用等于 dpR 参数组合。
- -d: 复制时保留链接。这里所说的链接相当于 Windows 系统中的快捷方式。
- -f: 覆盖已经存在的目标文件而不给出提示。
- -i: 与 -f 选项相反，在覆盖目标文件之前给出提示，要求用户确认是否覆盖，回答 y 时目标文件将被覆盖。
- -p: 除复制文件的内容外，还把修改时间和访问权限也复制到新文件中。
- -r: 若给出的源文件是一个目录文件，此时将复制该目录下所有的子目录和文件。
- -l: 不复制文件，只是生成链接文件。

- 使用指令 `cp` 将当前目录 `test/` 下的所有文件复制到新目录 `newtest` 下，输入如下命令：

```
sudo cp -r test/ newtest
```

mv

- `mv` 命令用来为文件或目录改名、或将文件或目录移入其它位置。
- 参数:
 - `-b`: 当目标文件或目录存在时，在执行覆盖前，会为其创建一个备份。
 - `-i`: 如果指定移动的源目录或文件与目标的目录或文件同名，则会先询问是否覆盖旧文件，输入 `y` 表示直接覆盖，输入 `n` 表示取消该操作。
 - `-f`: 如果指定移动的源目录或文件与目标的目录或文件同名，不会询问，直接覆盖旧文件。
 - `-n`: 不要覆盖任何已存在的文件或目录。
 - `-u`: 当源文件比目标文件新或者目标文件不存在时，才执行移动操作。
- 使用指令 `mv` 将当前目录 `test/` 下的 `file1` 文件复制到新目录 `/home/linaro` 下，输入如下命令：

```
sudo mv file1 /home/linaro
```

rm

- `rm` 命令用于删除一个文件或者目录。
- 参数:
 - `-i` 删除前逐一询问确认。
 - `-f` 即使原档案属性设为唯读，亦直接删除，无需逐一确认。
 - `-r` 将目录及以下之档案亦逐一删除。
-

- 删除文件可以直接使用 `rm` 命令，若删除目录则必须配合选项“-r”，例如：

```
sudo rm test.txt
```

- `rm`：是否删除 一般文件 "test.txt"? y

```
sudo rm homework
```

- `rm`：无法删除目录"homework"：是一个目录

```
sudo rm -r homework
```

- `rm`：是否删除 目录 "homework"? y

reboot

- `reboot` 命令用于用来重新启动计算机,更改 Tinker Board 2 的配置经常需要重启。
- 参数:
 - `-n`：在重开机前不做将记忆体资料写回硬盘的动作
 - `-w`：并不会真的重开机，只是把记录写到 `/var/log/wtmp` 档案里
 - `-d`：不把记录写到 `/var/log/wtmp` 档案里 (`-n` 这个参数包含了 `-d`)
 - `-f`：强迫重开机，不呼叫 `shutdown` 这个指令
 - `-i`：在重开机之前先把所有网络相关的装置先停止
- 重新启动

```
sudo reboot
```

shutdown

- Tinker Board 2 的关机是不能直接拔掉电源线的,因为 Tinker Board 2 会将内存作为暂存区,如果直接拔掉电源线会使一些在内存中的数据没有来得及写入 SD 卡中, 从而造成数据的丢失或是损坏 SD 卡上的数据,造成系统无法启动。

- 参数

- -t seconds : 设定在几秒钟之后进行关机程序。
- -k : 并不会真的关机, 只是将警告讯息传送给所有使用者。
- -r : 关机后重新开机。
- -h : 关机后停机。
- -n : 不采用正常程序来关机, 用强迫的方式杀掉所有执行中的程序后自行关机。
- -c : 取消目前已经进行中的关机动作。
- -f : 关机时, 不做 fsck 动作(检查 Linux 档系统)。
- -F : 关机时, 强迫进行 fsck 动作。
- time : 设定关机的时间。
- message : 传送给所有使用者的警告讯息。

- 实例

- 立即关机

```
sudo shutdown -h now
```

- 指定 10 分钟后关机

```
sudo shutdown -h 10
```

- 重新启动计算机

```
sudo shutdown -r now
```

- 无论使用哪一个命令来关闭系统都需要 root 用户权限，如果用户使用 linaro 这样的普通用户，可以使用 sudo 命令暂时获得 root 权限。

pwd

- 该 pwd 命令显示当前工作目录的名称：在 Tinker Board 2 上，输入 pwd 将输出类似 /home/linaro。

head

- 该 head 命令显示文件的开头。可用于-n 指定要显示的行数（默认为 10 行），或与-c 指定字节数。

```
head test.py -n 5
```

tail

- 该 tail 显示文件的结尾。-c 字节或-n 行数指定文件中的起始点

df

- 用于 df 显示已安装文件系统中可用和使用的磁盘空间。用于 df -h 以可读的格式查看输出，使用 M 表示 MB，而不是显示字节数。

```
df -h
```

zip

- zip 命令用于压缩文件,zip 是个使用广泛的压缩程序，压缩后的文件后缀名为 .zip。
- 如果在我们在 /home/linaro/waveshare 目录下，将这个目录下所有文件和文件夹打包为当前目录下的 waveshare.zip,可以执行以下命令：

```
zip -q -r waveshare.zip *
```

uzip

- unzip 命令用于解压缩 zip 文件，unzip 为.zip 压缩文件的解压缩程序。

```
unzip waveshare.zip -d user/
```

- 其中-d 为指定文件解压缩后所要存储的目录。

tar

- tar 命令是用来建立，还原备份文件的工具程序，它可以加入，解开备份文件内的文件。
- 压缩文件：

```
tar -cvzf waveshare.tar.gz *
```

- 解压文件：

```
tar -xvzf waveshare.tar.gz
```

apt

- apt (Advanced Packaging Tool) 是一个在 Debian 和 Ubuntu 中的 Shell 前端软件包管理器。
- apt 命令提供了查找、安装、升级、删除某一个、一组甚至全部软件包的命令，而且命令简洁而又好记。
- apt 命令执行需要超级管理员权限(root)。
- apt 常用命令
 - 列出所有可更新的软件清单命令：sudo apt update
 - 升级软件包：sudo apt upgrade
 - 列出可更新的软件包及版本信息：apt list --upgradeable
 - 升级软件包，升级前先删除需要更新软件包：sudo apt full-upgrade
 - 安装指定的软件命令：sudo apt install <package_name>

- 安装多个软件包: `sudo apt install <package_1> <package_2> <package_3>`
 - 更新指定的软件命令: `sudo apt update <package_name>`
 - 显示软件包具体信息,例如: 版本号, 安装大小, 依赖关系等等: `sudo apt show <package_name>`
 - 删除软件包命令: `sudo apt remove <package_name>`
 - 清理不再使用的依赖和库文件: `sudo apt autoremove`
 - 移除软件包及配置文件: `sudo apt purge <package_name>`
 - 查找软件包命令: `sudo apt search <keyword>`
 - 列出所有已安装的包: `apt list --installed`
 - 列出所有已安装的包的版本信息: `apt list --all-versions`
- 例如我们安装 nano 编辑器

```
sudo apt install nano
```

网络

ifconfig

- 用于在不带任何参数 (即) `ifconfig` 运行时显示当前系统上接口的网络配置详细信息。

- 用 SSH 连接时可以通过 ifconfig 查找 IP 地址，例如这台设备的有线连接 IP 地址和无线连接 IP 地址分别是：192.168.15.103 和 192.168.15.102

```
linaro@linaro-alip: ~  
File Edit Tabs Help  
linaro@linaro-alip:~$ sudo ifconfig  
dummy0: flags=195<UP,BROADCAST,RUNNING,NOARP> mtu 1500  
inet6 fe80::c96c:a019:af99:3946 prefixlen 64 scopeid 0x20<link>  
ether ba:42:aa:77:34:06 txqueuelen 1000 (Ethernet)  
RX packets 0 bytes 0 (0.0 B)  
RX errors 0 dropped 0 overruns 0 frame 0  
TX packets 133 bytes 44785 (43.7 KiB)  
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
inet 192.168.15.103 netmask 255.255.255.0 broadcast 192.168.15.255  
inet6 fe80::70ff:2258:21a8:128a prefixlen 64 scopeid 0x20<link>  
ether 50:eb:f6:74:5e:99 txqueuelen 1000 (Ethernet)  
RX packets 12150 bytes 888350 (867.5 KiB)  
RX errors 0 dropped 0 overruns 0 frame 0  
TX packets 21598 bytes 21651206 (20.6 MiB)  
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
device interrupt 24  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
inet 127.0.0.1 netmask 255.0.0.0  
inet6 ::1 prefixlen 128 scopeid 0x10<host>  
loop txqueuelen 1 (Local Loopback)  
RX packets 114 bytes 8894 (8.6 KiB)  
RX errors 0 dropped 0 overruns 0 frame 0  
TX packets 114 bytes 8894 (8.6 KiB)  
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
wlp1s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
inet 192.168.15.102 netmask 255.255.255.0 broadcast 192.168.15.255  
inet6 fe80::33c2:8e14:d6dd:e2a7 prefixlen 64 scopeid 0x20<link>  
ether 90:e8:68:8b:41:33 txqueuelen 1000 (Ethernet)  
RX packets 3782 bytes 35960573 (34.2 MiB)  
RX errors 0 dropped 102 overruns 0 frame 0  
TX packets 492 bytes 282194 (275.5 KiB)  
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
linaro@linaro-alip:~$
```

hostname

- 该 hostname 命令显示系统的当前主机名。我们使用 Tinker Board 2 的时候经常需要使用远程工具，而默认的网络配置 IP 地址采用动态分配，会造成 IP 地址不确定的问题，当我们的 Tinker Board 2 IP 地址发生变化时，可以使用主机名登录。

1. 登录 Tinker Board 2，修改 hosts 文件，命令如下：

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

- 将 lianro-alip 替换成要修改的名字，例如 waveshare,按下键盘 Ctrl+s 保存，按下键盘的 Ctrl+x 退出:

```
GNU nano 3.2 /etc/hosts
127.0.0.1 localhost
::1 localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
127.0.1.1 tinker
```

2. 修改 hostname 文件,将这里 linaro 也替换成要修改的名字,例如 asus, 按下键盘 Ctrl+x, 按下键盘的 y,回车确认:

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

3. 修改完成重启 Tinker Board 2 即可:

```
sudo reboot
```

4. 修改主机名也可以用 sudo tinker-config 中: [Hostname](#)
5. 我们也可以使用如下命令查看 IP 地址:

```
hostname -I
```

配置

连接 WIFI

1. 切换到超级用户模式

```
sudo su root
```

2. 打开 WIFI

```
nmcli r wifi on
```

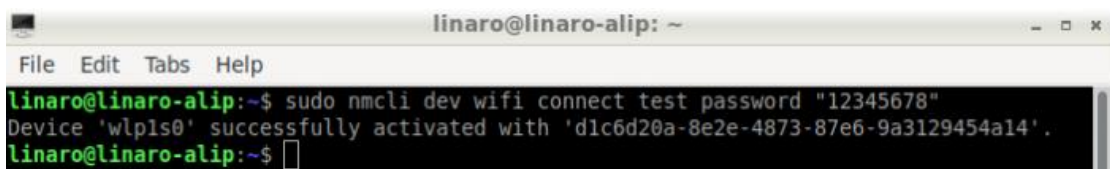
3. 扫描 WIFI

```
nmcli dev wifi
```

4. 连接到 WIFI 网络（“wifi_name” 和 “wifi_password” 需要替换为您的实际 WiFi 的 SSID 和密码。）

```
nmcli dev wifi connect wifi_name password "wifi_password"
```

5. 显示 “successfully” 就成功连接无线网络了，主板下次开机自动连接到您指定的 WiFi。



```
linaro@linaro-alip: ~  
File Edit Tabs Help  
linaro@linaro-alip:~$ sudo nmcli dev wifi connect test password "12345678"  
Device 'wlp1s0' successfully activated with 'd1c6d20a-8e2e-4873-87e6-9a3129454a14'.  
linaro@linaro-alip:~$
```

tinker-config 工具

- tinker-config 是 Tinker Board 2 官方镜像自带的一个系统配置工具。
- 要打开配置工具，请在命令行中键入以下内容，回车确认：

```
sudo tinker-config
```

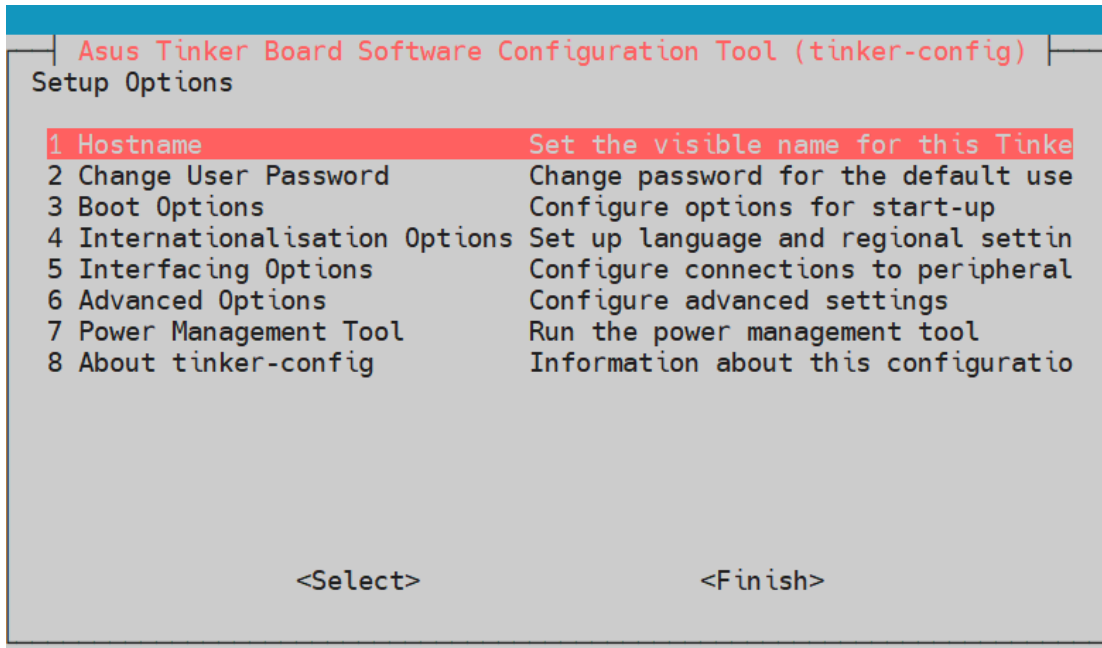


```
Asus Tinker Board Software Configuration Tool (tinker-config)  
Setup Options  
1 Hostname Set the visible name for this Tinker Board on a network  
2 Change User Password Change password for the default user (linaro)  
3 Boot Options Configure options for start-up  
4 Internationalisation Options Set up language and regional settings to match your location  
5 Interfacing Options Configure connections to peripherals  
6 Advanced Options Configure advanced settings  
7 Power Management Tool Run the power management tool  
8 About tinker-config Information about this configuration tool  
  
<Select> <Finish>
```

- 基本操作：键盘上的上、下键进行菜单项目的选择，回车键进入，左右键进行 OK 和 cancel 等按钮的选择，Esc 键取消返回，空格键为选择定选项。最后改完后选择 Finish 退出即可，可能会重启。

Hostname

- Tinker Board 2 系统默认的主机名为 linaro-alip，你可以修改自己专属的主机名。



Change User Password

- Tinker Board 2 系统默认的登录密码：linaro,你可以修改登录密码来保护账户安全。

Boot Options

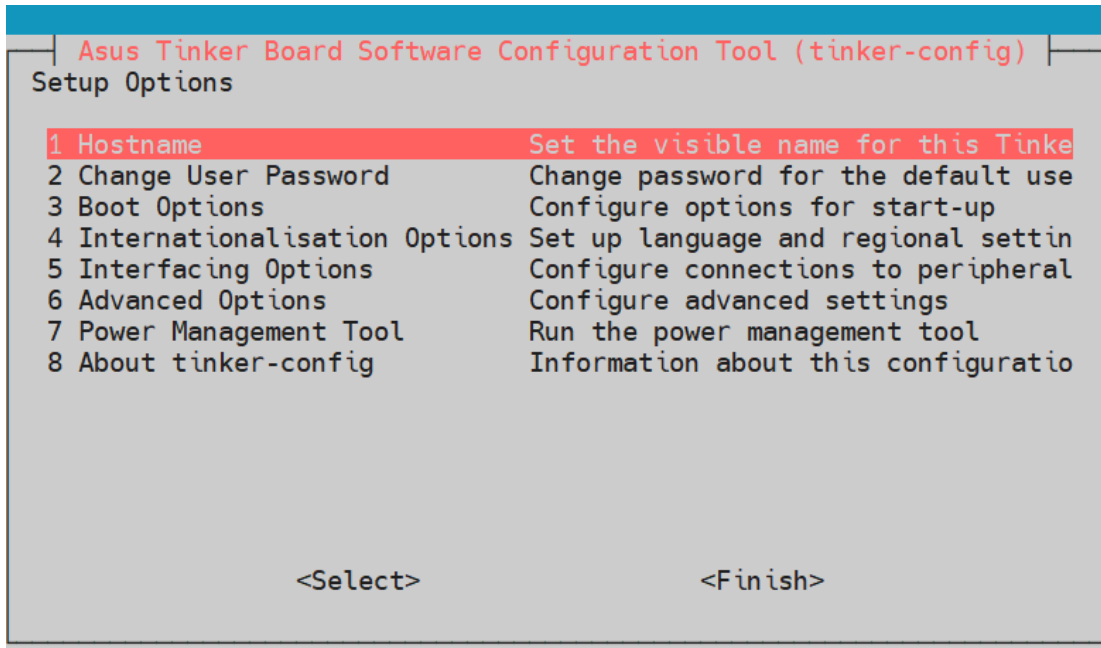
- Desktop / CLI: 选择是启动到桌面环境/终端命令模式。

Internationalisation Options

- Change Locale:设置语言和区域。
- Change Timezone:设置时区。
- Change Keyboard Layout:设置键盘布局。

Interfacing Options

- 在此子菜单中，有以下选项可启用/禁用：VNC, Camera。例如我们启用 vnc:



Advanced Options

- A1 Expand Filesystem : 允许您扩展文件系统。扩展文件系统将最大化分区的大小以匹配物理 TF 卡上可用的大小。
- A2 Resolution: 设置屏幕分辨率。

Power Management Tool

- 查看 Tinker Board 2 的设备信息、系统配置。

- 监测当前的 CPU 的温度、频率、内存占用情况等。

Device Info

```

Name = Tinker Board 2S - 16GB
Version = 1.02
SN/PPID = e5ddf8e2f04e80d6

SoC/CPU = Tinker
Memory = 2GB
Storage = eMMC(mmcblk1) 14.6G
           No SD card(mmcblk0)

OS = Debian GNU/Linux
Version = 10 (buster)
Platform = aarch64

Build = 2.0.4-20211222-release
Kernel:
Release = Linux version 4.4.194
Version = #1 SMP Wed Dec 22 06:57:56 UTC 2021

```

System Config

CPU:
Governor = auto

	Min. freq.	Max. freq.
4 x Arm Cortex-A53	408	1512
2 x Arm Cortex-A72	408	2016

GPU:
Governor = auto

	Min. freq.	Max. freq.
Arm Mali-T86x	200	800

Monitor

CPU:

	Curr. freq.	Temperature
4 x Arm Cortex-A53	600	50.00°C
2 x Arm Cortex-A72	1200	50.00°C

GPU:

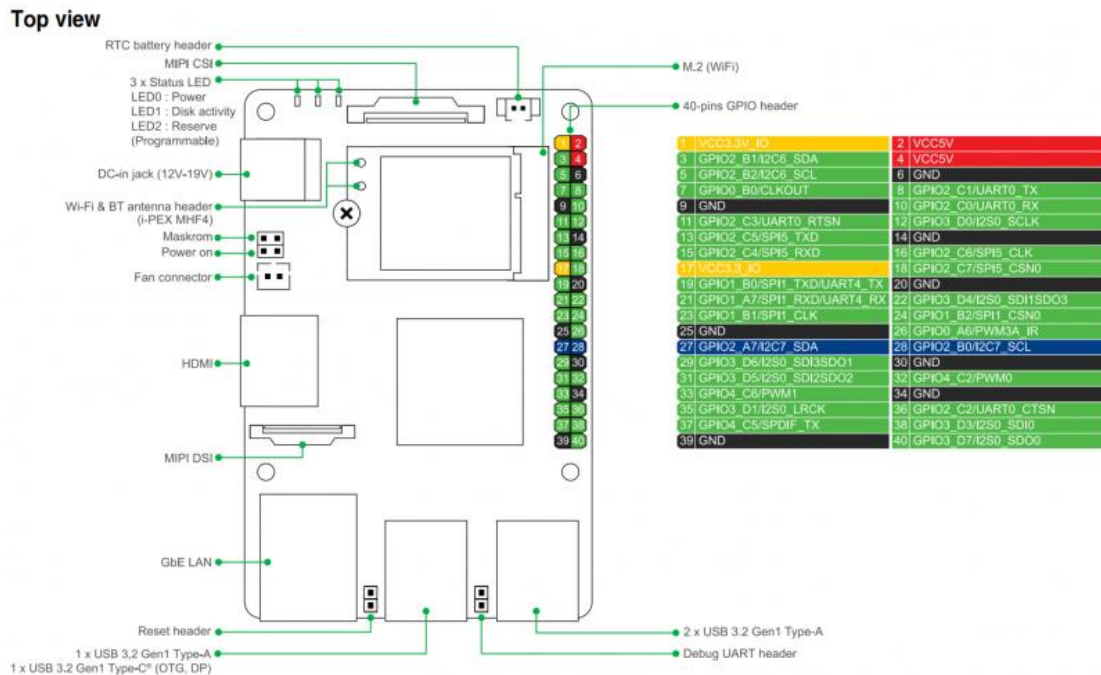
	Curr. freq.	Temperature
Arm Mali-T86x	200	48.75°C

CPU usage: 7%
GPU usage: 0%
Memory usage: 528 / 2006 MB (26%)

Press (C)PU or (G)PU to change governor, Ctrl + C to exit.

LED

- 如图所示，Tinker board 2s 有三个状态 led 灯，分别为 LED0 (Power)、LED1 (disk activity) 和 LED2 (Reserve)。



LED 控制

- 在嵌入式 Linux 中，LED 接口非常普遍，但由于它的特殊性，LED 接口的使用和标准的 Linux 接口存在较大的差异。

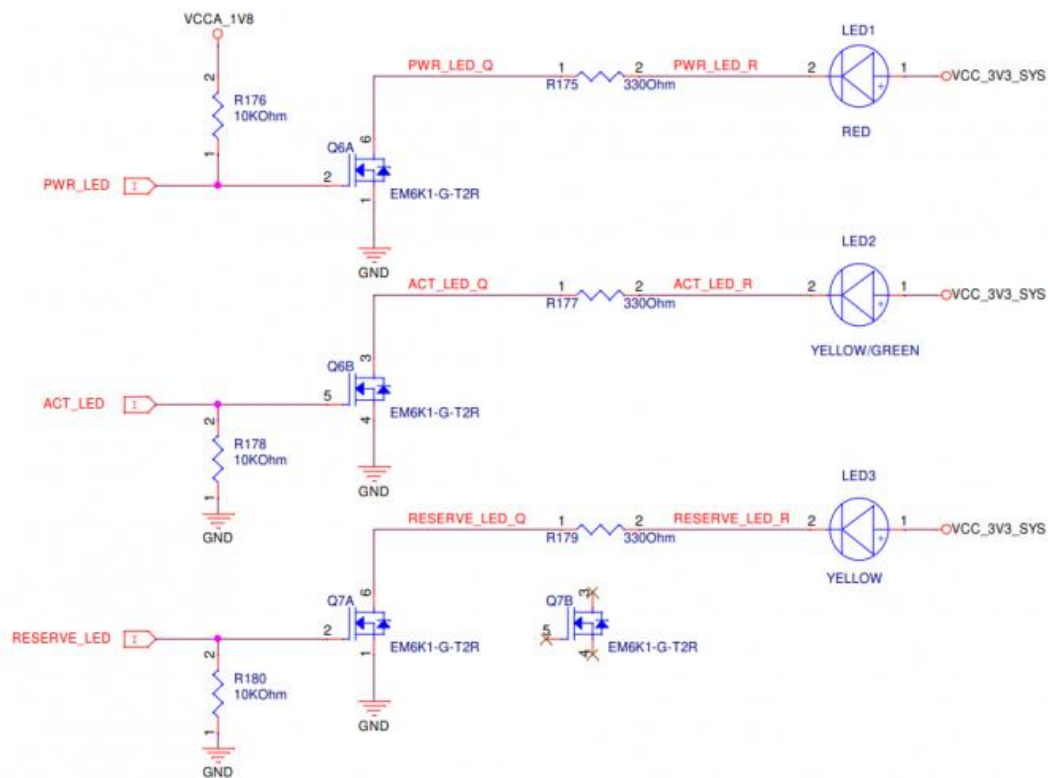
- 对于 Tinker Board 2s, LED 操作接口位于 /sys/class/leds 目录下。首先切换为 root 用户, 并且进入 LED 目录下:

- `sudo su root`

```
cd /sys/class/leds/
```

- 查看下官方的原理图, 根据 NMOS 管特性可以分析出三颗 LED 都是高电平驱动:

LED



控制 LED0

- 进入 pwr-led 目录:

- `cd pwr-led`
- `echo "1" > brightness #点亮`

```
echo "0" > brightness #熄灭
```

控制 LED1

- 进入 act-led 目录:

- `cd act-led`
- `echo "1" > brightness #点亮`

```
echo "0" > brightness #熄灭
```

控制 LED2

- 进入 rsv-led 目录:

- `cd rsv-led`
- `echo "1" > brightness #点亮`

```
echo "0" > brightness #熄灭
```

GPIO

简介

- GPIO 全称: 通用型输入输出端口 (General-purpose input/output)
- Tinker Board 2 引脚分类
 - 电源引脚:5v, 3.3v, GND(Ground)。
 - 常规 GPIO 控制引脚:可以通过编写程序控制这些引脚的高低电平。
 - 特殊 GPIO 通讯引脚: SPI 通讯, 12C 通讯, TxD/RxD 串口通讯。

- Tinker Board 2 有一个 40 针的扩展接头。

GPIO	Pin definition	Pin#		Pin definition	GPIO
	VCC3.3_IO	1	2	VCC5V	
73	GPIO2_B1/I2C6_SDA	3	4	VCC5V	
74	GPIO2_B2/I2C6_SCL	5	6	GND	
8	GPIO0_B0/CLKOUT	7	8	GPIO2_C1/UART0_TX	81
	GND	9	10	GPIO2_C0/UART0_RX	80
83	GPIO2_C3/UART0_RTSEN	11	12	GPIO3_D0/I2S0_SCLK	120
85	GPIO2_C5/SPI5_TXD	13	14	GND	
84	GPIO2_C4/SPI5_RXD	15	16	GPIO2_C6/SPI5_CLK	86
	VCC3.3_IO	17	18	GPIO2_C7/SPI5_CSN0	87
40	GPIO1_B0/SPI1_TXD/UART4_TX	19	20	GND	
39	GPIO1_A7/SPI1_RXD/UART4_RX	21	22	GPIO3_D4/I2S0_SDI1SDO3	124
41	GPIO1_B1/SPI1_CLK	23	24	GPIO1_B2/SPI1_CSN0	42
	GND	25	26	GPIO0_A6/PWM3A_IR	6
71	GPIO2_A7/I2C7_SDA	27	28	GPIO2_B0/I2C7_SCL	72
126	GPIO3_D6/I2S0_SDI1SDO1	29	30	GND	
125	GPIO3_D5/I2S0_SDI1SDO2	31	32	GPIO4_C2/PWM0	146
150	GPIO4_C6/PWM1	33	34	GND	
121	GPIO3_D1/I2S0_LRCK	35	36	GPIO2_C2/UART0_CTSN	82
149	GPIO4_C5/SPDIF_TX	37	38	GPIO3_D3/I2S0_SDI0	123
	GND	39	40	GPIO3_D7/I2S0_SDO0	127

GPIO 端口操作

GPIO 编号计算

- GPIO 有 5 个 bank, GPIO0 到 GPIO4, 每个 bank 有 32 个 pin, 命名如下:

- GPIO0_A0 ~ A7
- GPIO0_B0 ~ B7
- GPIO0_C0 ~ C7
- GPIO0_D0 ~ D7
-
- GPIO1_A0 ~ A7
-
- GPIO1_D0 ~ D7
-
- GPIO4_D0 ~ D7

- 对于 Linux 4.4 内核, GPIO 数量可以计算如下, 以 GPIO3_D7 (40PIN GPIO 上的 PIN40) 为例:

- $\text{GPIO4_D5} = 3 \times 32 + 3 \times 8 + 7 = 127$
- (A=0, B=1, C=2, D=3)

- 设置 GPIO3_D7 输出:

- `cd /sys/class/gpio`
- `echo 127 > export`
- `cd gpio127`
- `echo out > direction`
- `echo 1 > value # 输出高`
- `echo 0 > value # 输出低`

Python 控制

1. 使用如下命令查看 Tinker Board 2s 预装 wiringPi 和 RPi 库

```
ls /usr/local/share
```



```
linaro@linaro-alip: ~  
File Edit Tabs Help  
linaro@linaro-alip:~$ ls /usr/local/share/  
ca-certificates  gpio_lib_c_rk3399      man  tinker-power-management  
fonts            gpio_lib_python_rk3399 mraa  
linaro@linaro-alip:~$
```

2. 使用 gpio readall 命令查看引脚对应情况

```
linaro@linaro-alip: ~  
File Edit Tabs Help  
linaro@linaro-alip:~$ gpio readall  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| CPU | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | CPU |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| 73 | 8 | 3.3v | | | 1 | 2 | | | 5v | | |  
| 74 | 9 | SDA.6 | I2C | 1 | 3 | 4 | | | 5v | | |  
| 8 | 7 | SCL.6 | I2C | 1 | 5 | 6 | | | 0v | | |  
| | | CLKOUT2 | CLK | 1 | 7 | 8 | 1 | IN | GPIO2C1 | 15 | 81 |  
| | | 0v | | | 9 | 10 | 1 | IN | GPIO2C0 | 16 | 80 |  
| 83 | 0 | GPIO2C3 | IN | 1 | 11 | 12 | 0 | IN | GPIO3D0 | 1 | 120 |  
| 85 | 2 | MOSI.5 | SPI | 0 | 13 | 14 | | | 0v | | |  
| 84 | 3 | MISO.5 | SPI | 1 | 15 | 16 | 0 | SPI | SCLK.5 | 4 | 86 |  
| | | 3.3v | | | 17 | 18 | 1 | SPI | CE5.0 | 5 | 87 |  
| 40 | 12 | GPIO1B0 | IN | 1 | 19 | 20 | | | 0v | | |  
| 39 | 13 | GPIO1A7 | IN | 1 | 21 | 22 | 0 | IN | GPIO3D4 | 6 | 124 |  
| 41 | 14 | GPIO1B1 | IN | 1 | 23 | 24 | 1 | IN | GPIO1B2 | 10 | 42 |  
| | | 0v | | | 25 | 26 | 0 | IN | GPIO0A6 | 11 | 6 |  
| 71 | 30 | GPIO2A7 | IN | 1 | 27 | 28 | 1 | IN | GPIO2B0 | 31 | 72 |  
| 126 | 21 | GPIO3D6 | IN | 0 | 29 | 30 | | | 0v | | |  
| 125 | 22 | GPIO3D5 | IN | 0 | 31 | 32 | 1 | OUT | GPIO4C2 | 26 | 146 |  
| 150 | 23 | GPIO4C6 | OUT | 1 | 33 | 34 | | | 0v | | |  
| 121 | 24 | GPIO3D1 | IN | 0 | 35 | 36 | 1 | IN | GPIO2C2 | 27 | 82 |  
| 149 | 25 | GPIO4C5 | IN | 0 | 37 | 38 | 0 | IN | GPIO3D3 | 28 | 123 |  
| | | 0v | | | 39 | 40 | 0 | IN | GPIO3D7 | 29 | 127 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| CPU | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | CPU |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
linaro@linaro-alip:~$
```

3. 安装 Python 库

4. `sudo apt-get update`
5. `sudo apt-get install python3-pip`

```
sudo pip3 install python-periphery
```

6. 编写测试程序吗，使用 CPU 编号

7. `from periphery import GPIO`
8. `import time`
- 9.
10. `LED_Pin = 146 #Physical Pin-32 is GPIO 146`
11. `# Open GPIO /sys/class/gpio/gpio73 with output direction`
12. `LED_GPIO = GPIO(LED_Pin, "out")`
- 13.
14. `while True:`
15. `try:`

```
16.     LED_GPIO.write(True)
17.     time.sleep(0.5)
18.     LED_GPIO.write(False)
19.     time.sleep(0.5)
20. except KeyboardInterrupt:
21.     LED_GPIO.write(False)
22.     break
23. except IOError:
24.     print ("Error")
25.
26.LED_GPIO.close()
```

27.运行程序:

```
sudo python3 gpio.py
```

wiringPi 控制

1. 编写测试程序 gpio.c

```
2. #include <wiringPi.h>
3.
4. int main(int argc, char * argv[])
5. {
6.     char i;
7.     wiringPiSetup();
8.     pinMode(26, OUTPUT);
9.     for(i = 0; i < 10; ++i)
10.    {
11.        digitalWrite(26, HIGH);
12.        delay(500);
13.        digitalWrite(26, LOW);
14.        delay(500);
15.    }
```



```
16.    digitalWrite(26, HIGH);  
17.    return 0;  
18. }
```

19.编译程序

```
gcc gpio.c -o gpio -lwiringPi
```

20.运行程序

```
sudo ./gpio
```

C 语言控制

1. 下载 [C 语言程序包](#)

2. 编译程序

```
sudo make
```

3. 运行程序

```
sudo ./main 146
```

I2C

简介

- I2C, 中文全称为集成电路总线;
- 它是一种串行通信总线, 使用多主从架构, 由飞利浦公司在 20 世纪 80 年代为了让主板和嵌入式系统用以连接低速周边设备而开发。I2C 的正确读法为 “I 平方 C” (“I-squared-C”) 或者 ‘I 方 C’

I2C 硬件连接方式

- I2C 总线在物理连接上非常简单，分别由 SDA(串行数据线)和 SCL(串行时钟线)及上拉电阻组成。
- 通信原理是通过控制 SCL 和 SDA 高低电平来产生 I2C 总线协议所需要的信号进行数据传输。
- 在总线空闲状态时，SCL 和 SDA 会上拉电阻拉高，保持着高电平。

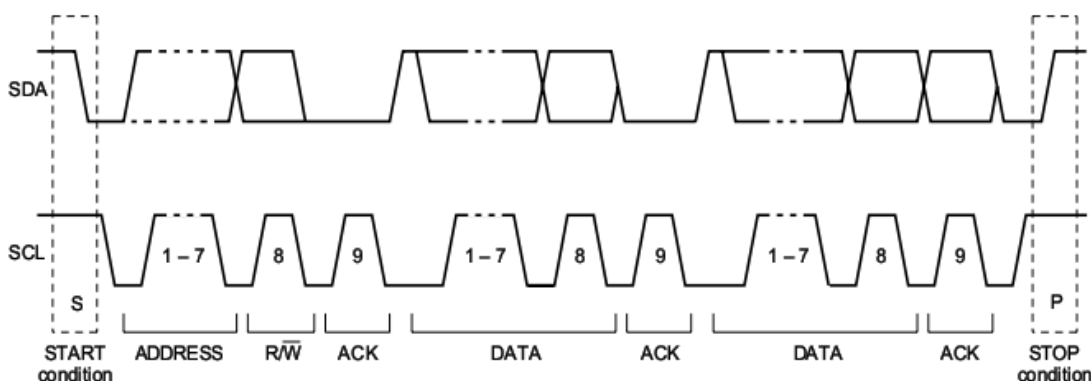
通过其物理连接方式，我们就可以知道 I2C 协议是串行、同步通讯协议。

特征

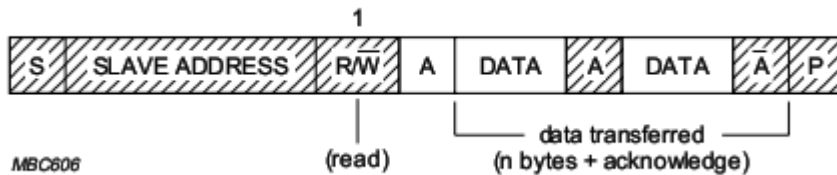
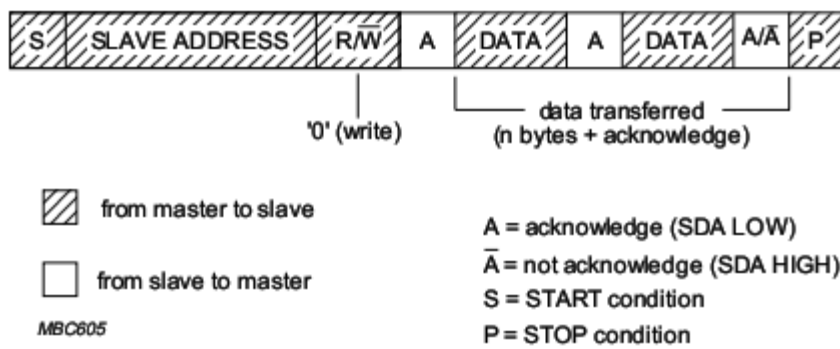
- I2C 总线上的每一个设备都会对应这唯一的 I2C 地址，部分从设备可以通过外围电路改变 I2C 地址。
- 主从设备之间就通过这个地址来确定与哪个器件进行通信。
- I2C 总线上的主设备与从设备之间以字节(8 位)为单位进行双向的数据传输。

协议详解

- I2C 协议规定，总线上数据的传输必须以一个起始信号作为开始条件，以一个结束信号作为传输的停止条件。起始和结束信号总是由主设备产生(意味着从设备不可以主动发起通信，所有的通信都是主设备发起的，主可以发出询问的指令，然后等待从设备的通信)。



- 总线在空闲状态时，SCL 和 SDA 都保持着高电平，当 SCL 为高电平而 SDA 由高到低的跳变，表示产生一个起始条件；当传输结束时，SCL 为高而 SDA 由低到高的跳变，表示产生一个停止条件。
- 在起始条件产生后，总线处于忙状态，由本次数据传输的主从设备独占总线，其他 I2C 器件无法访问总线；在停止条件产生后，本次数据传输的主从设备将释放总线，总线再次处于空闲状态。



- 数据传输以字节为单位。主设备在 SCL 线上产生每个时钟脉冲的过程中将在 SDA 线上传输一个数据位，当一个字节按数据位从高位到低位的顺序传输完后，紧接着从设备将拉低 SDA 线，回传给主设备一个应答位，此时才认为一个字节真正的被传输完成。当然，并不是所有的字节传输都必须有一个应答位，比如：当从设备不能再接收主设备发送的数据时，从设备将不回传应答位。

Rockchip I²C 主要参数

- 兼容 I2C 与 SMBus 总线
- 支持主模式下的 I²C 总线
- 软件可编程时钟频率和传输速率高达 1000Kbit/秒
- 支持 7 位和 10 位寻址模式
- 中断或轮询驱动多个字节数据传输
- 时钟拉伸和等待状态

Tinker Board 2 I2C 接口

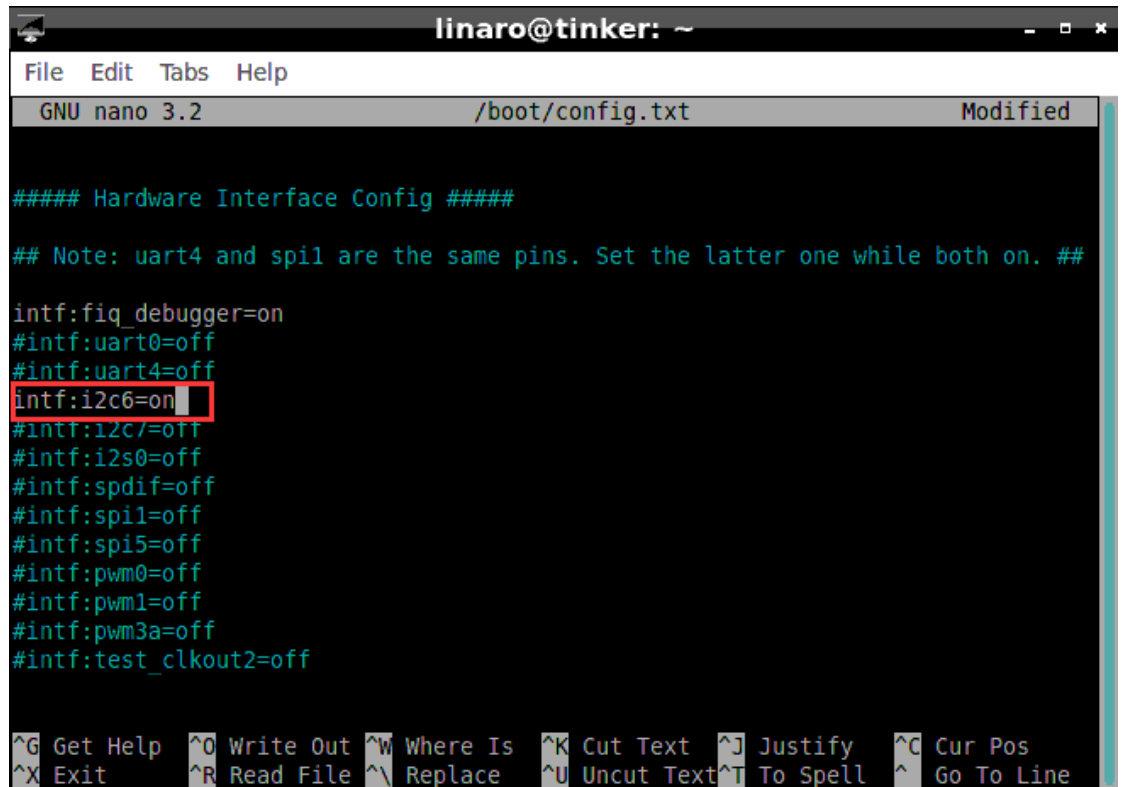
开启 I2C 接口

- Tinker Board 2 一共有两个 i2c 通道，分别是 I2C-6（引脚 3 和引脚 5）和 I2C-7（引脚 27 和引脚 28），以开启 I2C-6 为例，开启方法如下：

1. 打开配置文件 config.txt:

```
2. sudo nano /boot/config.txt
```

将#intf:i2c6=off 改为 intf:i2c6=on



```
linaro@tinker: ~
File Edit Tabs Help
GNU nano 3.2 /boot/config.txt Modified

##### Hardware Interface Config #####

## Note: uart4 and spi1 are the same pins. Set the latter one while both on. ##

intf:fiq_debugger=on
#intf:uart0=off
#intf:uart4=off
intf:i2c6=on
#intf:i2c7=off
#intf:i2s0=off
#intf:spdif=off
#intf:spi1=off
#intf:spi5=off
#intf:pwm0=off
#intf:pwm1=off
#intf:pwm3a=off
#intf:test_clkout2=off

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

3. 修改完成后，按住键盘的 Ctrl+s 保存，Ctrl+x 退出，重启设备：

```
sudo reboot
```

4. 重启完成后，我们在终端输入如下命令查看 I2C-6 开启成功：

```
ls /dev/i2c*
```



```
linaro@tinker: ~
File Edit Tabs Help

linaro@tinker:~$ ls /dev/i2c*
/dev/i2c-0 /dev/i2c-10 /dev/i2c-4 /dev/i2c-8
/dev/i2c-1 /dev/i2c-2 /dev/i2c-6 /dev/i2c-9
linaro@tinker:~$
```

i2c 测试

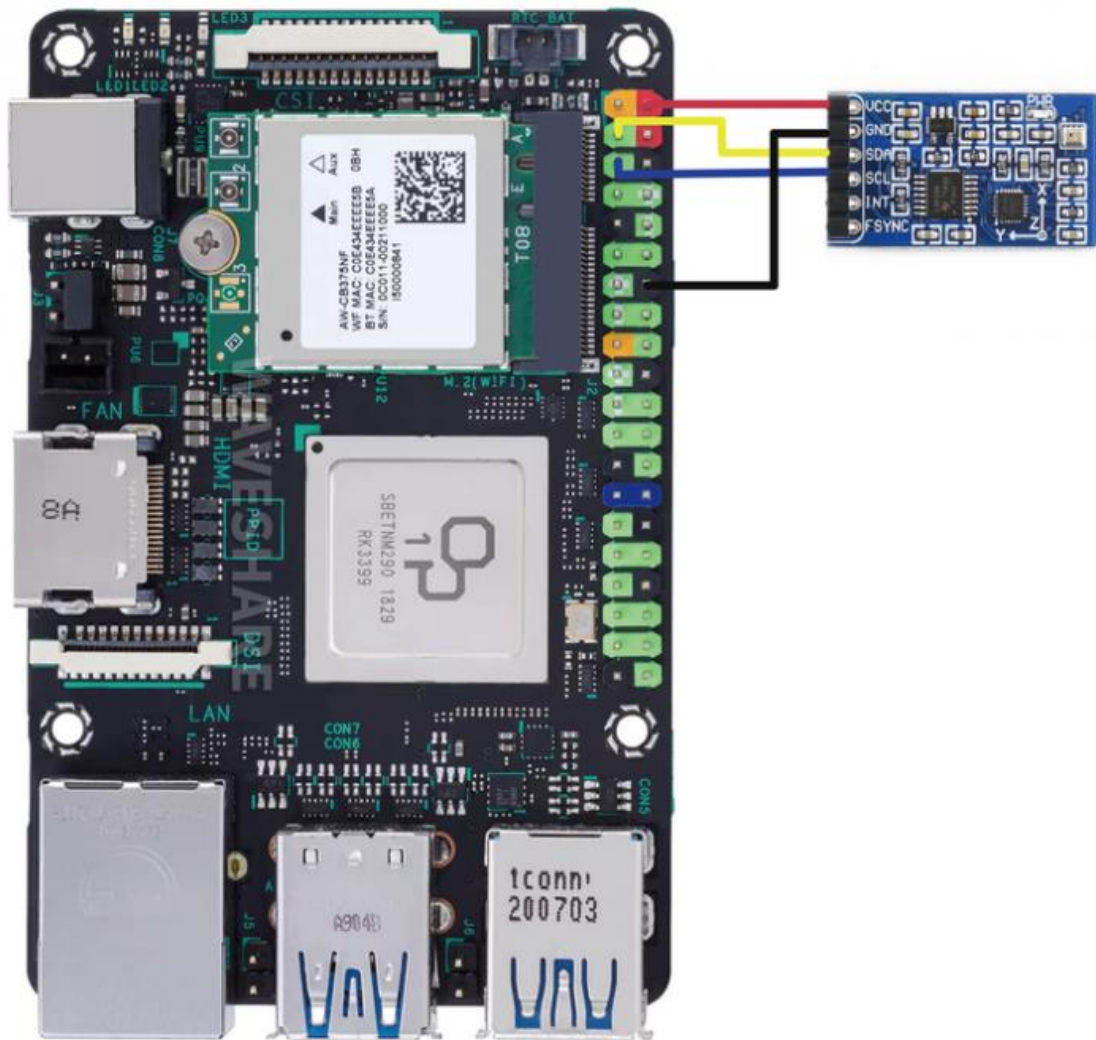
- 本实例采用的是 [10 轴惯性导航模块](#)。

硬件连接

-

Tinker Board 2 连接引脚对应关系

10 DOF	Tinker Board 2 Board 物理引脚序号	功能
VCC	5V	电源输入
GND	GND	电源地
SDA	3	I2C 数据输入
SCL	5	I2C 时钟信号



-

i2cdetect

- 查询 i2c 设备:

```
sudo i2cdetect -y -r -a 6
```

```
linaro@tinker:~$ sudo i2cdetect -y -r -a 6
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  68  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  77  --  --  --  --  --  --  --  --
```

- 参数: -y 是无视交互问题直接执行, -r 是 SMBus read byte 命令, -a 是所有地址, 6 是指 i2c-6。

- 扫描寄存器数据:

```
sudo i2cdump -y 6 0x68
```

```
linaro@tinker:~$ sudo i2cdump -y 6 0x68
No size specified (using byte-data access)
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  0123456789abcdef
00: ea 00 00 00 00 40 01 00 00 00 00 00 00 00 00 00  ?....@?.....
10: 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00  .....?.....
20: 00 00 00 00 00 00 00 e4 00 00 00 00 00 00 cd a9 09  .....?.....???
30: 0d d9 f5 00 14 00 1b 00 06 0b 90 1f 00 49 00 5f  ????.??.?????.I._
40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
60: 00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00  .....?.....
70: 00 00 00 00 03 40 00 03 10 00 00 00 1e 22 00 00  ....?@.??..."..
80: ea 00 00 00 00 40 01 00 00 00 00 00 00 00 00 00  ?....@?.....
90: 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00  .....?.....
a0: 00 00 00 00 00 00 00 e4 00 00 00 00 00 00 cd 9c 09  .....?.....???
b0: 31 d9 de ff fa 00 0e ff f1 0b 40 1f 00 49 00 5f  1???.??.??@?.I._
c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
e0: 00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00  .....?.....
f0: 00 00 00 00 03 40 00 03 10 00 00 00 21 f5 00 00  ....?@.??...!?...

```

- 寄存器数据写入:

```
sudo i2cset -y 6 0x68 0x90 0x55
```

```

linaro@tinker:~$ sudo i2cset -y 6 0x68 0x90 0x55
linaro@tinker:~$ sudo i2cdump -y 6 0x68
No size specified (using byte-data access)
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f      0123456789abcdef
00: ea 00 00 00 00 40 01 00 00 00 00 00 00 00 00 00  ?....@?.....
10: 55 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00  U.....?.....
20: 00 00 00 00 00 00 00 e4 00 00 00 00 00 00 cd 82 09  .....?.....???
30: 22 d9 c6 00 0c 00 12 ff f9 0b c0 1f 00 49 00 5f  "???.??.?????.I.
40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
60: 00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00  .....?.....
70: 00 00 00 00 03 40 00 03 10 00 00 00 24 98 00 00  ....?@.??...$?..
80: ea 00 00 00 00 40 01 00 00 00 00 00 00 00 00 00  ?....@?.....
90: 55 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00  U.....?.....
a0: 00 00 00 00 00 00 00 e4 00 00 00 00 00 cd ab 09  .....?.....???
b0: 27 d9 be 00 14 00 11 ff fa 0b c0 1f 00 49 00 5f  '???.??.?????.I.
c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
e0: 00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00  .....?.....
f0: 00 00 00 00 03 40 00 03 10 00 00 00 27 5f 00 00  ....?@.??...'..

```

- 参数:

参数	含义
6	代表 I2C 设备号
0x68	代表 I2C 设备地址
0x90	代表寄存器地址
0x55	代表向寄存器写入的数据

- 寄存器数据读出:

```
sudo i2cget -y 6 0x68 0x90
```

```

linaro@tinker:~$ sudo i2cget -y 6 0x68 0x90
0x55

```

- 参数:

参数	含义
1	代表 I2C 设备号
0x68	代表 I2C 设备地址
0x90	代表寄存器地址

I2C 编程

- 本实例采用的是 [10 轴惯性导航模块](#)。

下载测试程序

- 打开 Tinker Board 2 终端, 执行:

- `sudo apt-get install p7zip-full -y` #如果已安装, 可直接进入下一步
- `sudo wget https://www.waveshare.net/w/upload/2/27/Tinker-10_DOF_IMU_Sensor_D_Code.7z`
- `7z x Tinker-10_DOF_IMU_Sensor_D_Code.7z`

```
cd Tinker-10_DOF_IMU_Sensor_D_Code/wiringPi
```

运行测试程序

- 以下命令请在 Tinker Board 2 下执行, 否则不在索引不到目录;

- `sudo make`

```
sudo ./10Dof-D
```

现象

- 在运行程序后, 就会输出以下数据:

```
linaro@tinker:~/TinkerBoard/wiringPi$ sudo ./10Dof-D
Motion sensor is ICM-20948
Pressure sensor is BMP280

/-----/

Roll: 7.87      Pitch: 4.55      Yaw: -174.05

Acceleration: X: -1614      Y: 283      Z: -1235

Gyroscope: X: 3      Y: 5      Z: 0

Magnetic: X: 3      Y: -10      Z: -11

Pressure: 671174.75      Altitude: -13533.85

Temperature: 3.3
```

SPI

简介

- SPI 全称为串行外设接口 (Serial Peripheral Interface) , 其是一种高速的, 全双工, 同步通信总线.
- 它以主从方式工作, 这种模式通常一个主设备对应一个或多个从设备, 双向数据传输时需要 4 根线, 单向数据传输时可以裁切为 3 根线。

硬件连接

1. MOSI – 主设备输出/从设备输入引脚。该引脚对应主设备数据发送引脚, 从设备数据接受引脚
2. MISO – 主设备输入/从设备输出引脚。该引脚对应主设备数据接受引脚, 从设备数据发送引脚
3. SCLK – 同步时钟, 通常由主设备输出.
4. CS – 从设备选择。用来选择从设备。它的功能是让主设备可以与特定从设备通信, 避免数据线上的冲突。

工作方式

- Linux 内核用 CPOL 和 CPHA 的组合来表示当前 SPI 的四种工作模式:

• CPOL=0, CPHA=0	SPI_MODE_0
• CPOL=0, CPHA=1	SPI_MODE_1
• CPOL=1, CPHA=0	SPI_MODE_2
• CPOL=1, CPHA=1	SPI_MODE_3

- CPOL: 表示时钟信号的初始电平的状态, 0 为低电平, 1 为高电平。
- CPHA: 表示在哪个时钟沿采样, 0 为第一个时钟沿采样, 1 为第二个时钟沿采样。

- SPI 的四种工作模式波形图如下:

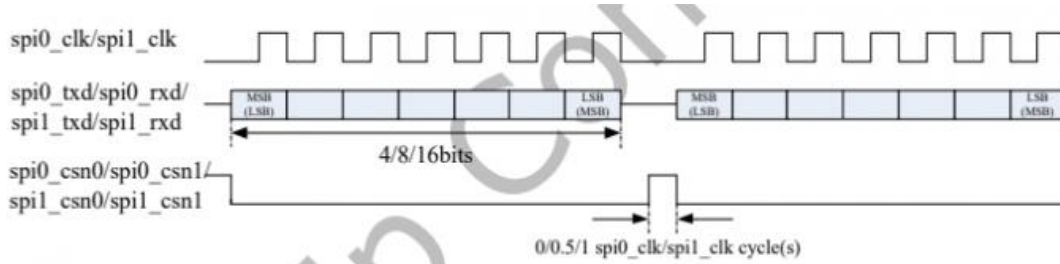


Fig. 13-3 SPI Format (SCPH=0 SCPOL=0)

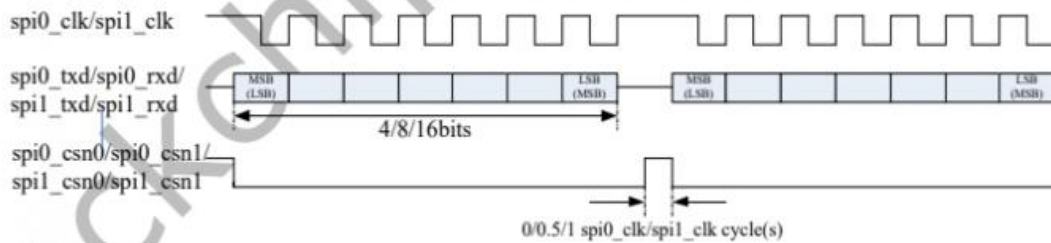


Fig. 13-4 SPI Format (SCPH=0 SCPOL=1)

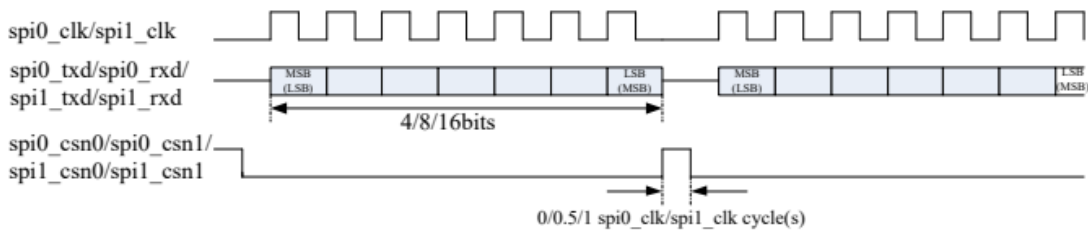


Fig. 13-5 SPI Format (SCPH=1 SCPOL=0)

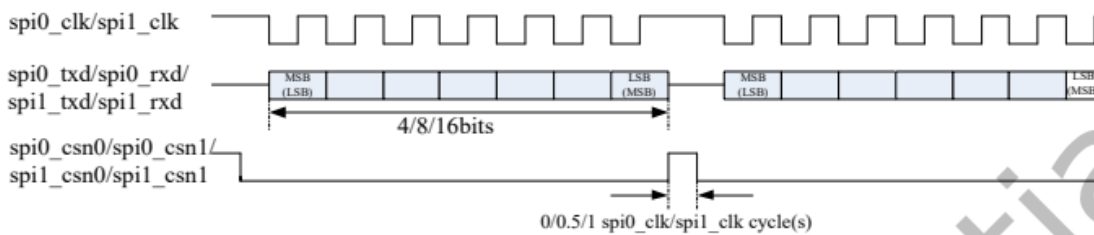


Fig. 13-6 SPI Format (SCPH=1 SCPOL=1)

开启 SPI 接口

- Tinker Board 2 一共有两个 spi 通道，分别是 spi-1 (引脚 13、引脚 15、引脚 16 和引脚 18) 和 spi-5 (引脚 19、引脚 21、引脚 23 和引脚 24)，以开启 spi-5 为例，开启方法如下:

1. 打开配置文件 config.txt:

```
2. sudo nano /boot/config.txt
```

将#intf:spi5=off 改为 intf:spi5=on

3. 修改完成后，按住键盘的 Ctrl+s 保存，Ctrl+x 退出，重启设备:

```
sudo reboot
```

4. 重启完成后，我们在终端输入如下命令查看 spi-5 开启成功：

```
ls /dev/spi*
```

CSI

- Tinker Board 2 支持树莓派摄像头 Camera Module v1(感光芯片为豪威科技 OV5647)和 Camera Module v2 (感光芯片为索尼 IMX219) ；
- Tinker Board 2 的系统默认使用的是 Camera Module v2。
- 如果您使用的是 Camera Module v1(感光芯片豪威科技 OV5647)摄像头需要手动切换，参考 [Interfacing Options](#) 开启。

Choose a camera setting (Need Reboot)

```
0 IMX219 (Raspberry Camera V2, 8MP)
1 OV5647 (Raspberry Camera V1, 5MP)
```

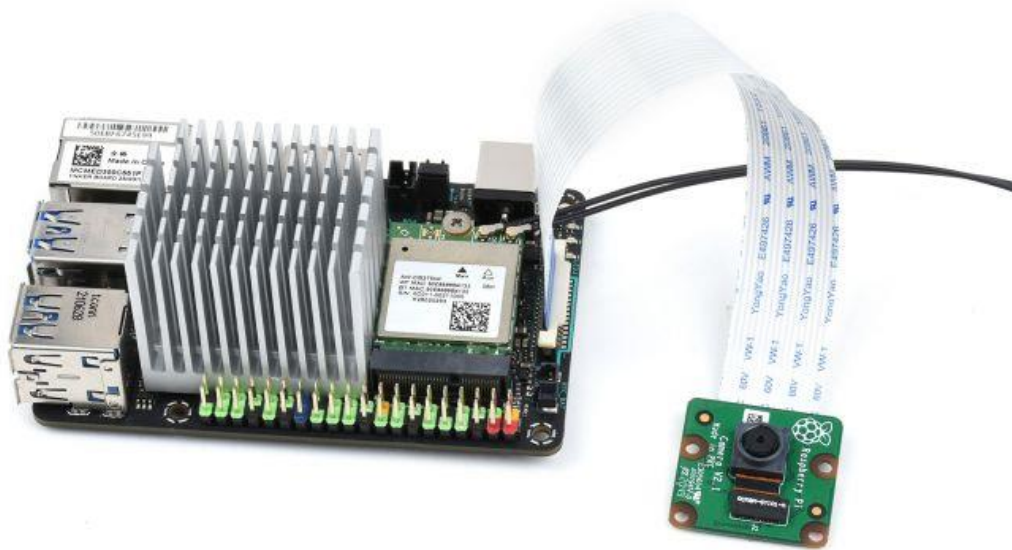
<Ok>

<Cancel>

硬件连接

- **要测试摄像头，建议给 Tinker Board 2 接入 HDMI 屏幕或者 DSI 屏幕**

- Tinker Board 2 主板上的 CSI（摄像头）和 DSI（显示器）两个接口的封装是相似的，接线的时候注意不要接错了。CSI 摄像头接口在 Power 接口边上。



开启摄像头

- 由于 tinkerboard 默认安装了 Gstreamer，所以我们可以直接使用。

1. 首先我们查看摄像头是否连接成功

```
v4l2-ctl -v
```

```
linaro@tinker:~$ v4l2-ctl -V
Format Video Capture Multiplanar:
  Width/Height      : 1920/1080
  Pixel Format      : 'RGB3' (24-bit RGB 8-8-8)
  Field            : None
  Number of planes  : 1
  Flags            :
  Colorspace       : sRGB
  Transfer Function: sRGB
  YCbCr/HSV Encoding: Default
  Quantization     : Full Range
  Plane 0         :
    Bytes per Line : 5760
    Size Image     : 6220800
linaro@tinker:~$
```

2. 直接打开终端输入

```
gst-launch-1.0 v4l2src ! video/x-raw,format=NV12,width=640,height=480 ! videoconvert ! autovideosink
```



USB

fswebcam

安装

- Debian 的软件源配置文件是 `/etc/apt/sources.list`。将系统自带的该文件做个备份，将该文件替换为下面内容，即可使用 TUNA 的软件源镜像

- `# 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/debian/ buster main contrib non-free`
- `# deb-src https://mirrors.tuna.tsinghua.edu.cn/debian/ buster main contrib non-free`
- `deb https://mirrors.tuna.tsinghua.edu.cn/debian/ buster-updates main contrib non-free`
- `# deb-src https://mirrors.tuna.tsinghua.edu.cn/debian/ buster-updates main contrib non-free`
-
- `deb https://mirrors.tuna.tsinghua.edu.cn/debian/ buster-backports main contrib non-free`
- `# deb-src https://mirrors.tuna.tsinghua.edu.cn/debian/ buster-backports main contrib non-free`
-
- `deb https://mirrors.tuna.tsinghua.edu.cn/debian-security buster/updates main contrib non-free`

```
# deb-src https://mirrors.tuna.tsinghua.edu.cn/debian-security buster/updates  
main contrib non-free
```

- 打开 Tinker Board 2 终端，输入指令安装 `fswebcam` 软件

- `sudo apt update`
- `sudo apt upgrade`

```
sudo apt install fswebcam
```

用法

- 输入命令 `fswebcam` 后跟文件名，将使用 USB 摄像头拍摄照片，并保存到指定的文件名：

```
fswebcam -d /dev/video5 -r 640x480 ~/image01.jpg
```

此命令将显示以下信息:

```
--- Opening /dev/video5...  
  
Trying source module v4l2...  
  
/dev/video5 opened.  
  
No input was specified, using the first.  
  
--- Capturing frame...  
  
Captured frame in 0.00 seconds.  
  
--- Processing captured image...  
  
Writing JPEG image to '/home/linaro/image01.jpg'.
```

- 全分辨率拍摄照片, 不带有横幅:

```
fswebcam -d /dev/video5 -r 1280x720 --no-banner image3.jpg
```

MJPEG-streamer

- 将 USB 摄像头插上, 查看是否找到设备, 输入指令:

```
v4l2-ctl --list-devices
```

```
linaro@tinker:~$ v4l2-ctl --list-devices  
rkisp1-statistics (platform: rkisp1):  
    /dev/video3  
    /dev/video4  
  
rkisp1_mainpath (platform:ff910000.rkisp1):  
    /dev/video0  
    /dev/video1  
    /dev/video2  
  
USB 2.0 Camera (usb-xhci-hcd.11.auto-1.1):  
    /dev/video5  
    /dev/video6  
  
linaro@tinker:~$
```

- 其中 USB2.0 Camera 就是摄像头, 说明找到 usb 设备了。

- 查看设备驱动是否正常: 输入指令:

```
ls /dev/video*
```

- 其中 video5 就是摄像头驱动。

- 安装必要的库,输入以下指令:

```
sudo apt install subversion -y
```

- 下载 mjpeg-stream 到树莓派:

- `sudo git clone https://github.com/jacksonliam/mjpg-streamer.git`
- `cd mjpg-streamer/mjpg-streamer-experimental`
- `sudo make all`

```
sudo make install
```

- 打开 start.sh

- `cd mjpg-streamer/mjpg-streamer-experimental/`
- `sudo nano start.sh`

```
将./mjpg_streamer -i "./input_uvc.so " -o "./output_http.so -w ./www"
```

修改为

```
./mjpg_streamer -i "./input_uvc.so -d /dev/video5" -o "./output_http.so -w ./www"
```

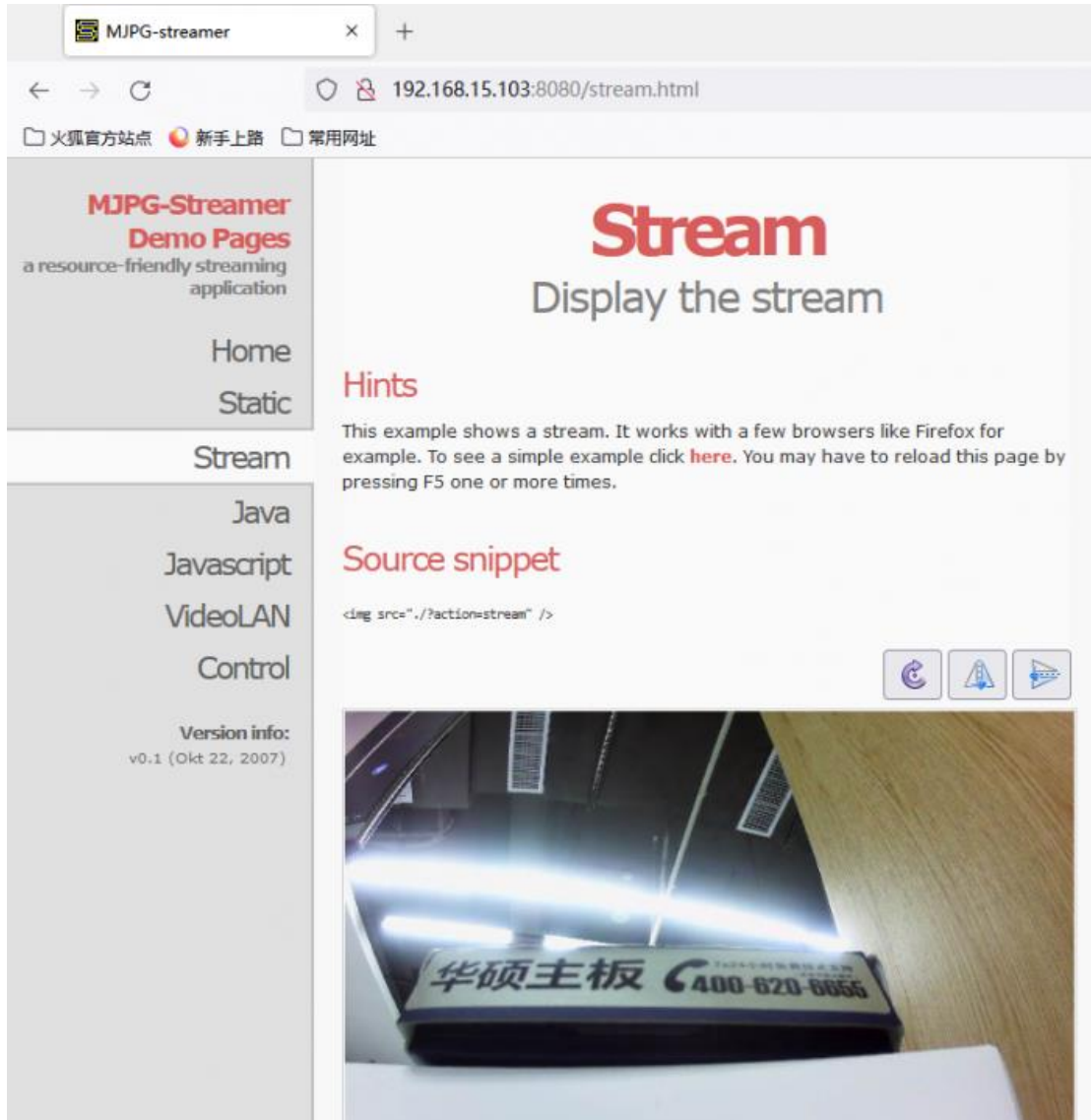
- 运行 start.sh:

- `cd mjpg-streamer/mjpg-streamer-experimental/`


```
sudo ./start.sh
```

- 用 Firefox 浏览器输入 `http://<Tinker IP 地址>:8080`,例如我的 IP 地址是 192.168.15.103

```
http://192.168.15.103:8080
```



- 此时应该可以看到摄像头的监控画面了。

DSI

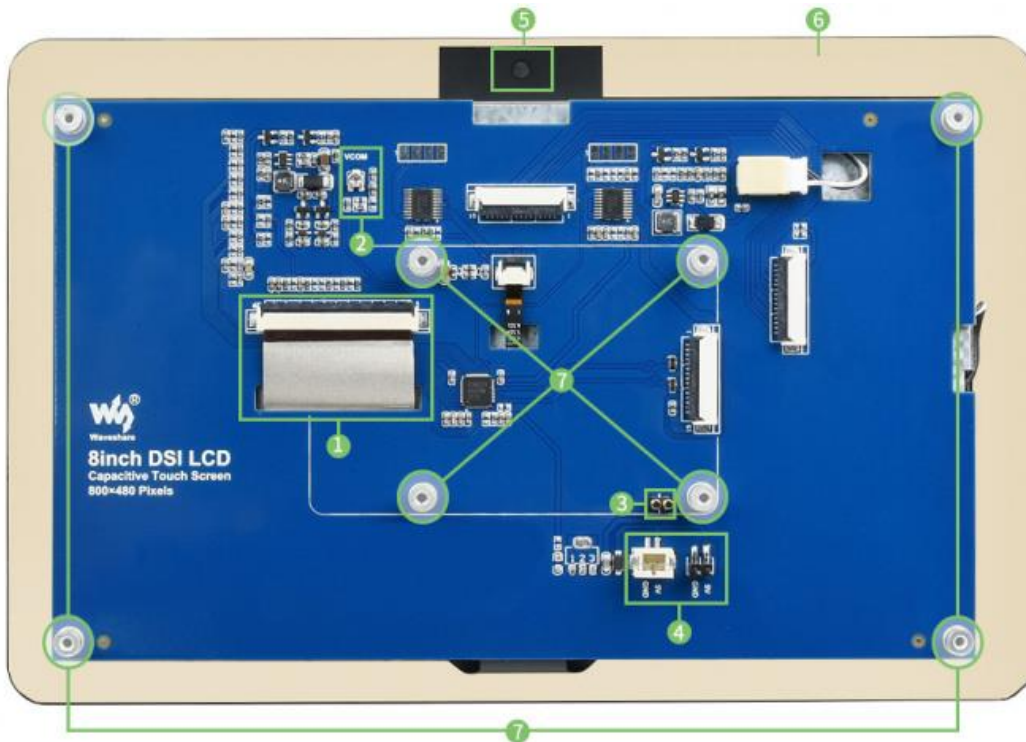
- Tinker Board 2 支持微雪 DSI 显示触摸屏，本实例采用的是 [8 寸电容触控屏](#)。

产品特点

- 8 寸电容触摸屏，硬件分辨率为 800×480
- 电容式 5 点触摸控制

- 钢化玻璃电容触摸面板，硬度达 6H
- 支持 Tinker Board 2 须另购转接线 [DSI-Cable-15cm](#)
- 直接通过 Tinker Board 2 的 DSI 接口驱动 LCD，刷新率可达 60Hz
- 配合 Tinker Board 2 使用时，支持 Debian 和安卓，免驱
- 支持通过软件控制背光亮度

特色设计



1. LCD FFC 线抗干扰设计，工业使用更稳定。
2. VCOM 电压调节电阻，可调节 VCOM 电压，以达到最佳的显示效果。
3. 可使用顶针进行供电，减少繁琐的接线。
4. 2 种形式的 5V 电源输出接口，方便用户接散热风扇或给其他小功率设备供电。
5. 触摸板预留摄像头孔位，方便用户集成外部摄像头。
6. 大盖板设计，方便设计外壳，易于嵌入各种设备中。
7. 贴片螺母支撑与固定主板，结构更紧凑。

搭配 Tinker Board 2 使用

硬件连接

1. 使用 22Pin 转 15Pin 的 FPC 排线，将 8inch DSI LCD 的 DSI 接口连接到 Tinker Board 2 的 DSI 接口。
 - **注意：软排线和屏幕连接金属面朝上，软排线和开发板连接金属面朝向 HDMI 接口**
2. 为了方便使用，可以把 Tinker Board 2 通过螺丝固定的 8inch DSI LCD 的背面，并组装上铜柱。(Tinker Board 2GPIO 接口将通过顶针给 LCD 供电)

如下图所示：



背光控制

- 在终端输入以下命令可以控制背光亮度：

```
• sudo su root
```

```
echo X > /sys/class/backlight/panel_backlight/brightness
```

其中 X 表示 0~255 中的任意数字。0 表示背光最暗，255 表示背光最亮。例如：

```
echo 100 > /sys/class/backlight/panel_backlight/brightness
```

```
echo 0 > /sys/class/backlight/panel_backlight/brightness
```

```
echo 255 > /sys/class/backlight/panel_backlight/brightness
```