[野火] 快速使用手册-基于 LubanCat-RV110x 系列板卡

EmbedFire

野火电子

2025 年 02 月 13 日

Contents

关于野火

开源共享，共同进步

野火在发布第一块 STM32 开发板之初，就喊出 **开源共享，共同进步**的口号，把代码和文档教程都免费提供给用户下载，而我们也一直把这个理念贯穿至今。

目前我们的产品已经包括 瑞萨 *RA* 系列、*STM32*、*i.MX RT* 系列、上海先辑、*GD32*、普冉、*Linux*、瑞芯微、全志、鲁班猫卡片电脑、华芯微特、捷联微芯、雅特力、*FPGA*、*Altera*、*Xilinx*、紫光同创、嵌入式操作系统教程、下载器等分支，覆盖电子工程应用领域的各种常用技术，其中教学类产品的代码和文档一直保持着开源的姿态发布到网络上，为电子工程师排忧解难，让嵌入式没有难用的技术是我们最大的愿望。

联系方式

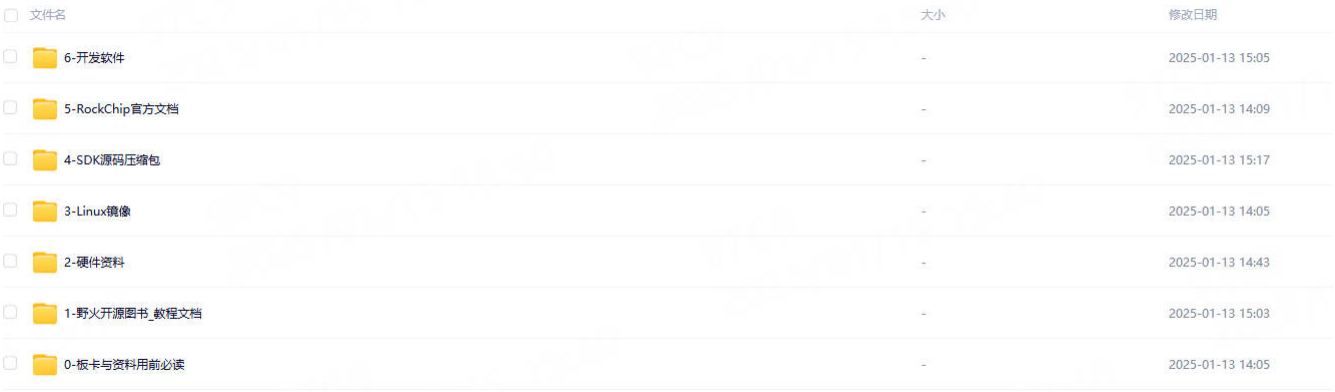
- 地址：东莞市大岭山镇石大路 2 号艺华综合办公大楼 301 1~4 楼
- 官网：https://www.embedfire.com
- 资料：https://doc.embedfire.com
- 论坛：https://www.firebbs.cn
- 天猫：https://yehuosm.tmall.com
- 京东：https://yehuo.jd.com
- 邮箱：embedfire@embedfire.com
- 电话：0769-33894118

图 1: 关注野火公众号，获取更多资讯，免费获取野火所有产品资料。

第 1 章　资料获取

1.1 百度网盘

链接: https://pan.baidu.com/s/1i0I6K_MLiA9zW4_w_VNLdg?pwd=teyf 提取码: teyf



- 0-板卡与资料用前必读: 百度网盘存放文件说明
- 1-野火开源图书 _ 教程文档: 野火开源图书 _ 教程文档，存放着快速使用手册的 pdf 文件
- 2-硬件资料: 硬件资料，包括板卡原理图，尺寸图，位号图等
- 3-Linux 镜像: Linux 镜像，spi-nand 和 buildroot 镜像
- 4-SDK 源码压缩包: RV06 的 SDK 源码压缩包，版本会更新的慢，建议使用 github 下载
- 5-RockChip 官方文档: RockChip 官方文档
- 6-开发软件: 开发软件，烧录软件，驱动等

1.2 github

SDK github 地址: https://github.com/LubanCat/RV06_03_Linux_SDK

```
# SDK 下载
git clone https://github.com/LubanCat/RV06_03_Linux_SDK.git --depth=1
```

第 2 章　用户名和密码

用户名	密码
root	root

第 3 章　镜像升级变动

3.1 2025-02-12

1. 修复 app 配置错误问题
2. 移除 wifi 初始化代码
3. otg 移除默认 adb 功能
4. kernel 修复 gadget 功能
5. 移除所有 dosc
6. SDK readme 修复文本描述错误，修改格式
7. readme.md 更新内核和 buildroot 配置文件修改命令
8. kernel 添加 u 盘和读卡器的内核配置
9. config RV06 内核配置添加 pwm oneshot 功能
10. dts rv06 添加 30pin 引脚的初始配置，修改 adc 的设备名

3.2 2025-01-15

1. 优化了 otg gadget 脚本
2. 优化了网络工具配置，去掉了 udhcpc，ifup,ifdown 等脚本，替换成 dhcpd
3. usb rndis 使用静态地址，避免每次重启 usb rndis 地址变化

3.3 2025-01-13

初次提交

第 4 章　镜像烧录

LubanCat RV1106 系列板卡支持多种启动方式，包括 SPI NAND 启动、TF 卡启动等方式。

本文档主要介绍怎么把镜像分别烧录到 SPI NAND 和 TF 卡上。

4.1 TF 卡烧录

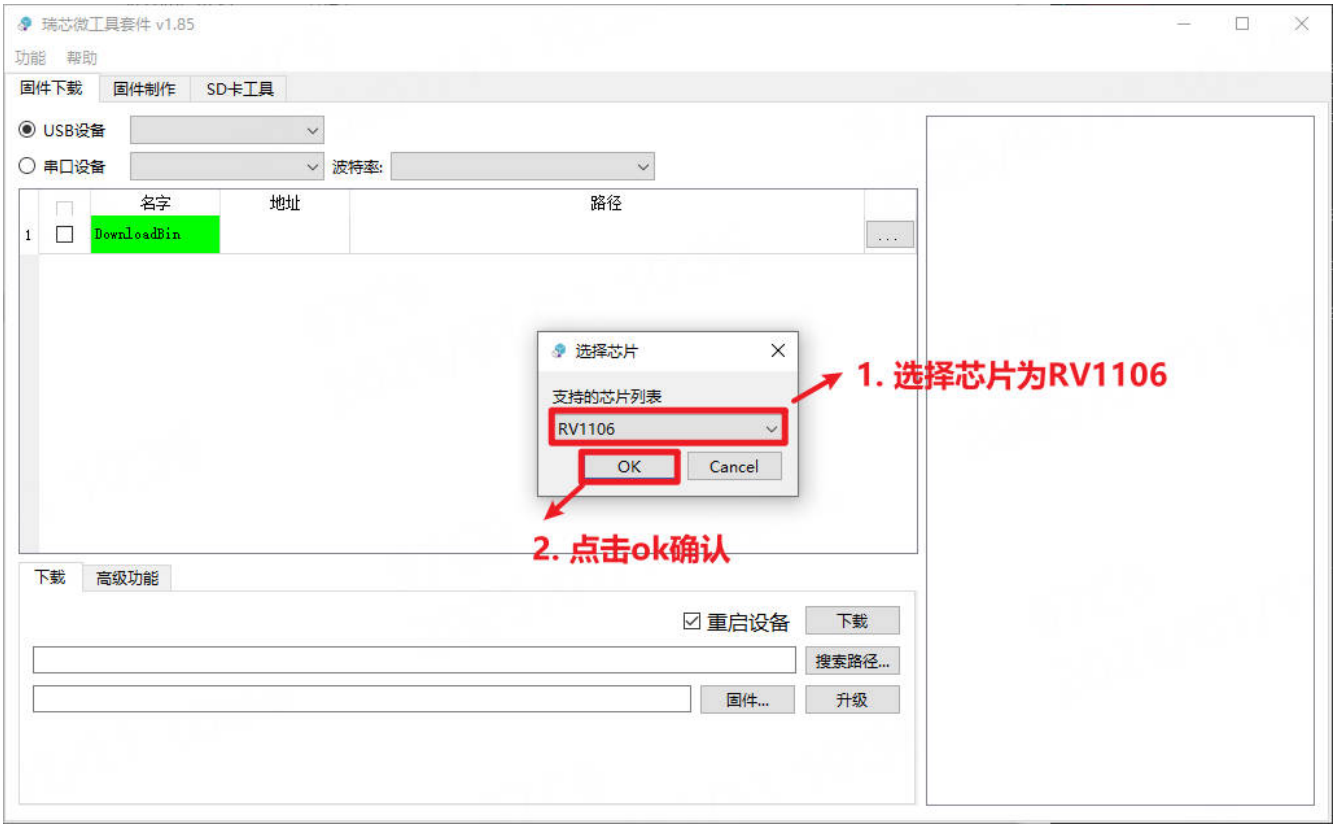
烧录时需要准备以下材料

1. 读卡器
2. 一台 windows 电脑
3. 8G 或 8G 以上的 TF 卡

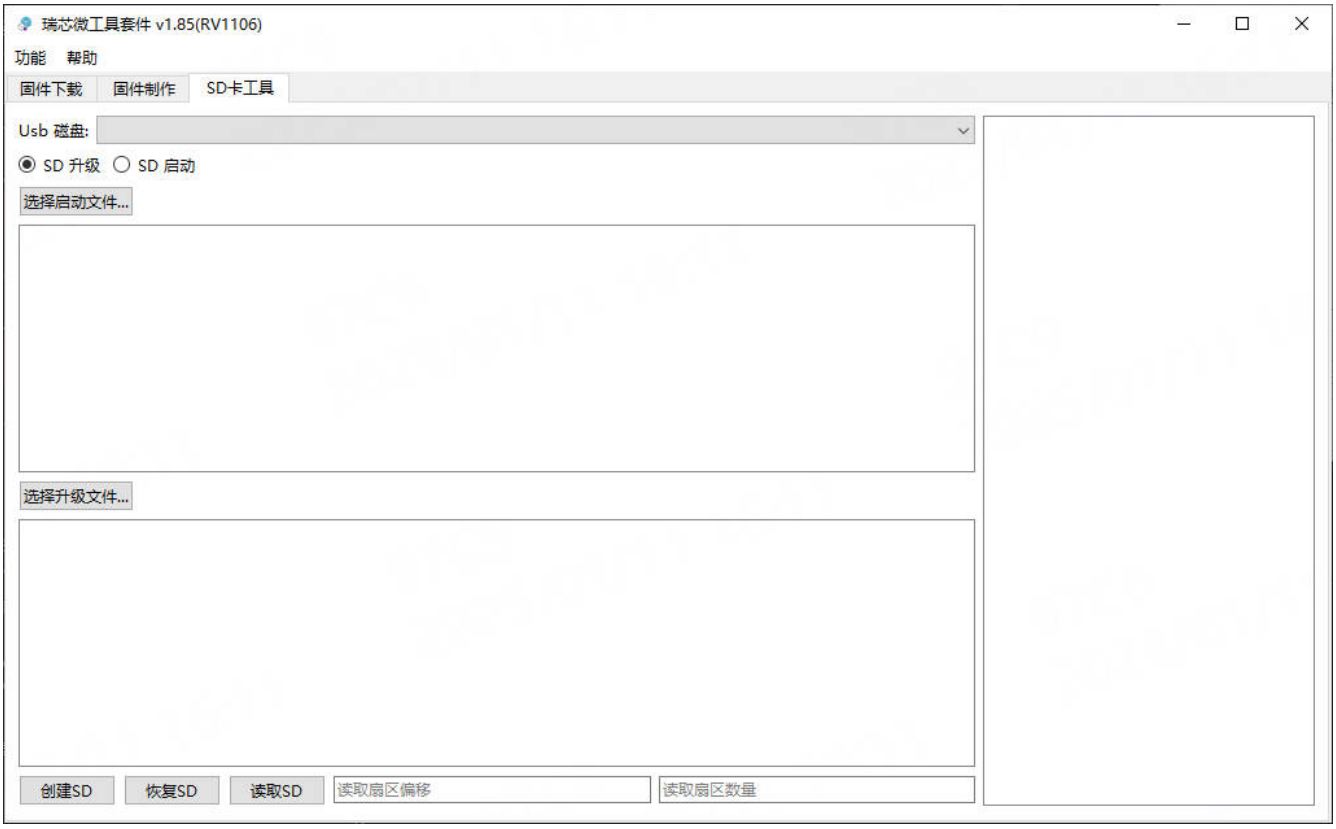
4.1.1 安装烧录软件

- 找到烧录软件 `SocToolKit.zip`，可以在百度网盘上下载
- 解压成文件夹，然后双击运行 `SocToolKit.exe`

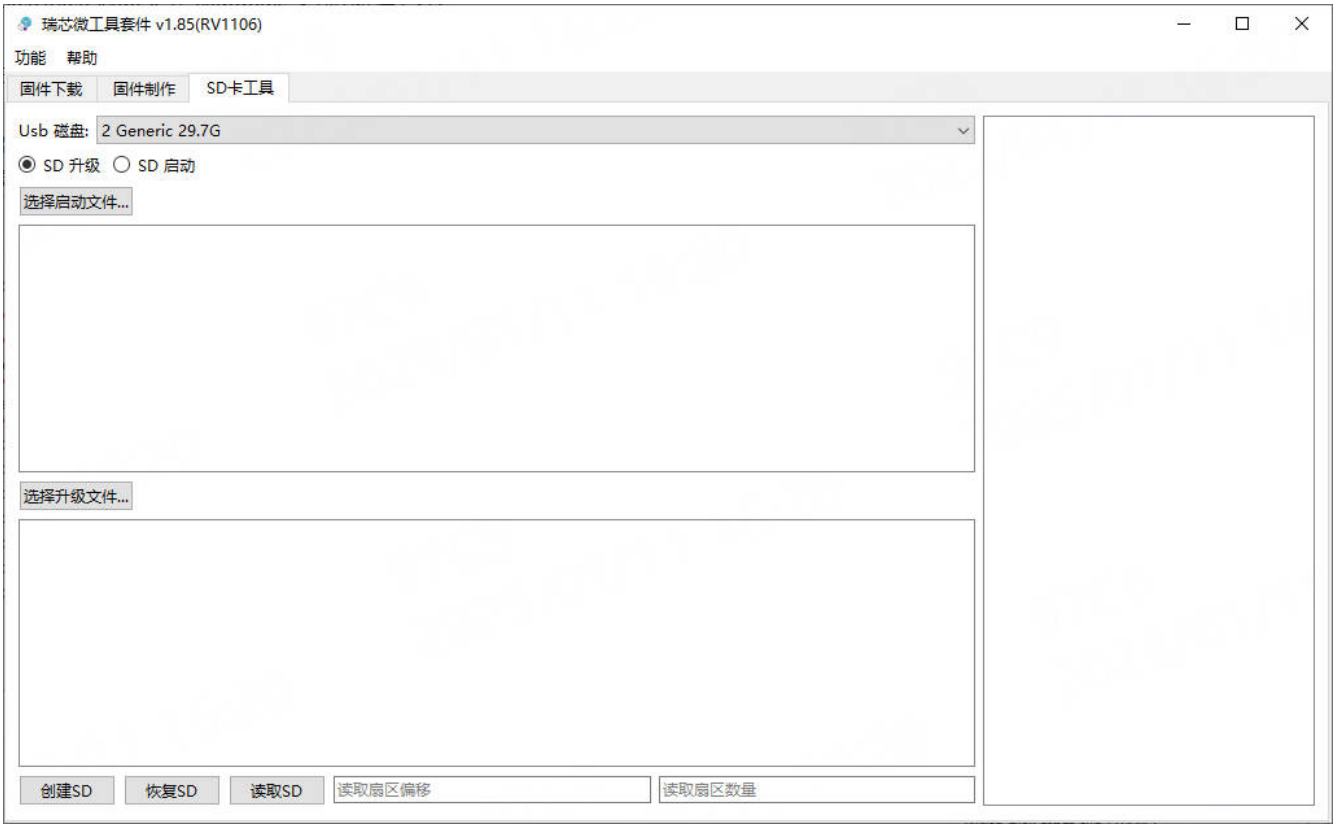
- 进入烧录工具后会弹出小窗，在小窗这里选择芯片 RK1106，然后点击 OK



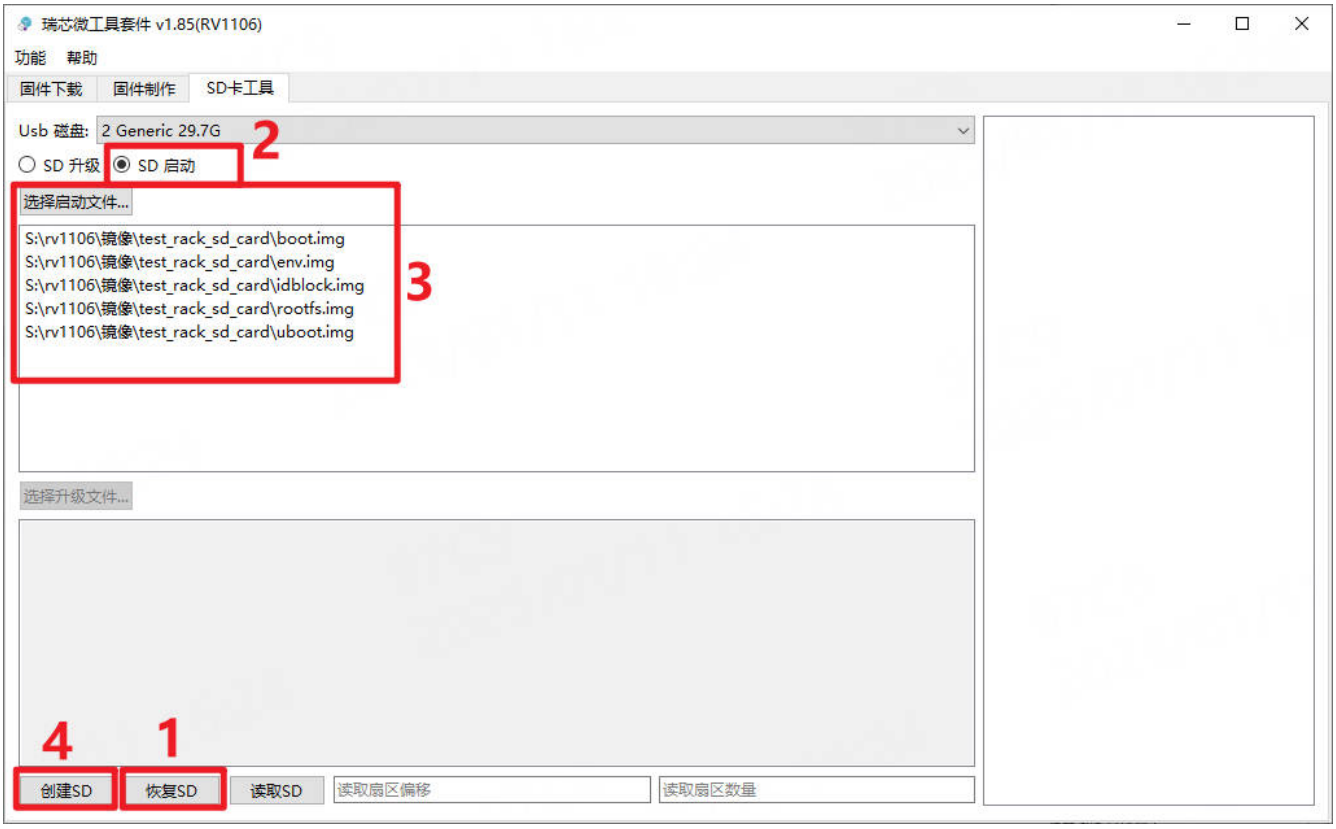
- 点击 SD 卡工具进入到 TF 卡烧录界面



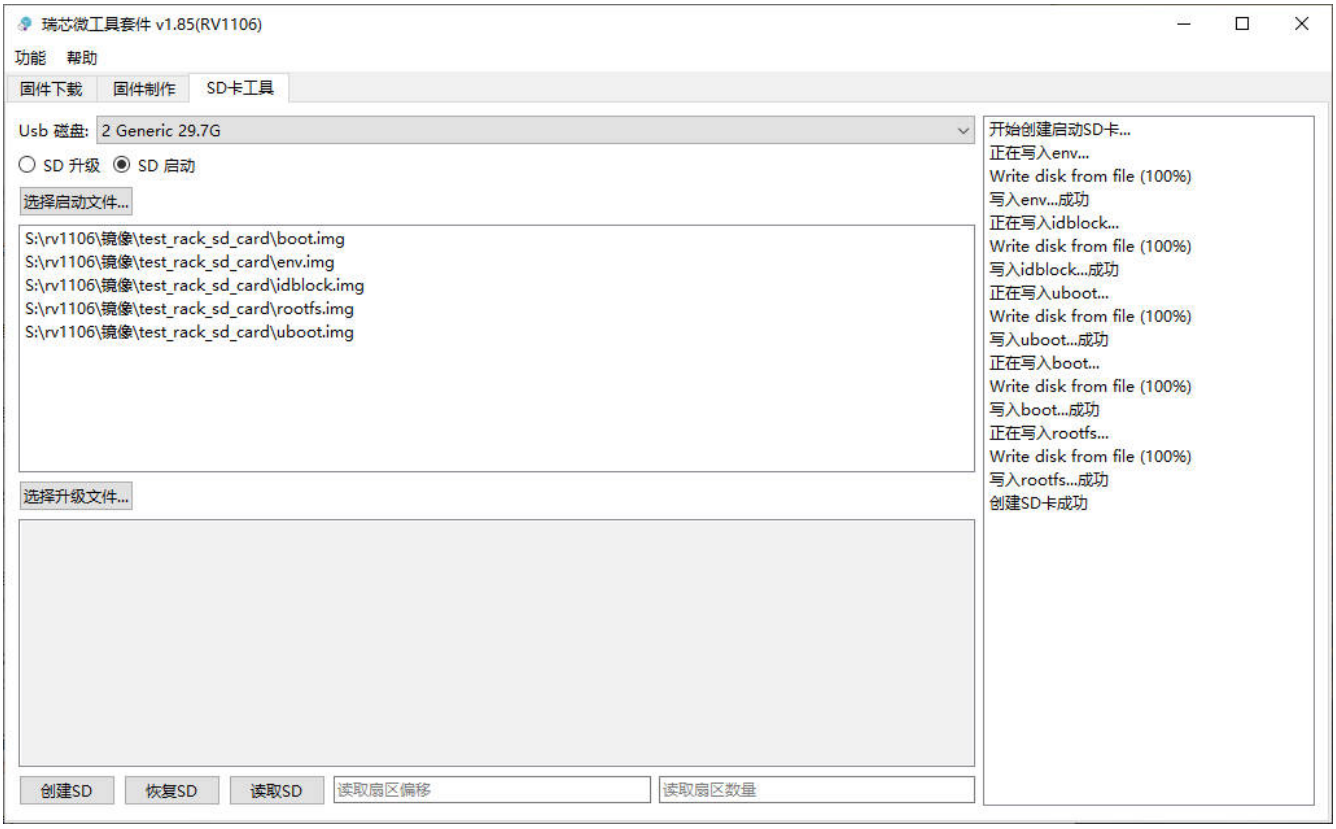
- 将 TF 卡插入到读卡器中，然后插入电脑，会看到识别出了 SD 卡设备



- 首先选择 恢复 SD，选择 SD 启动，选择启动文件，sd 卡烧录的镜像，最后点击创建 SD，则将镜像烧录到 SD 卡中


瑞芯微工具套件 v1.85(RV1106)
功能 帮助
固件下载
固件制作
SD卡工具
Usb 磁盘:
2 Generic 29.7G
2
SD 升级
SD 启动
选择启动文件...
S:\rv1106\镜像\test_rack_sd_card\boot.img
S:\rv1106\镜像\test_rack_sd_card\env.img
S:\rv1106\镜像\test_rack_sd_card\idblock.img
S:\rv1106\镜像\test_rack_sd_card\rootfs.img
S:\rv1106\镜像\test_rack_sd_card\uboot.img
3
选择升级文件...
4
1
创建SD
恢复SD
读取SD
读取扇区偏移
读取扇区数量

- 烧录成功如下图所示



4.2 SPI NAND 烧录

SPI NAND 烧录需要使用板卡上的 type-c 接口对其进行烧录，烧录时需要准备以下材料

1. 带 usb 通讯功能的 type-c 数据线
2. 一台 windows 电脑

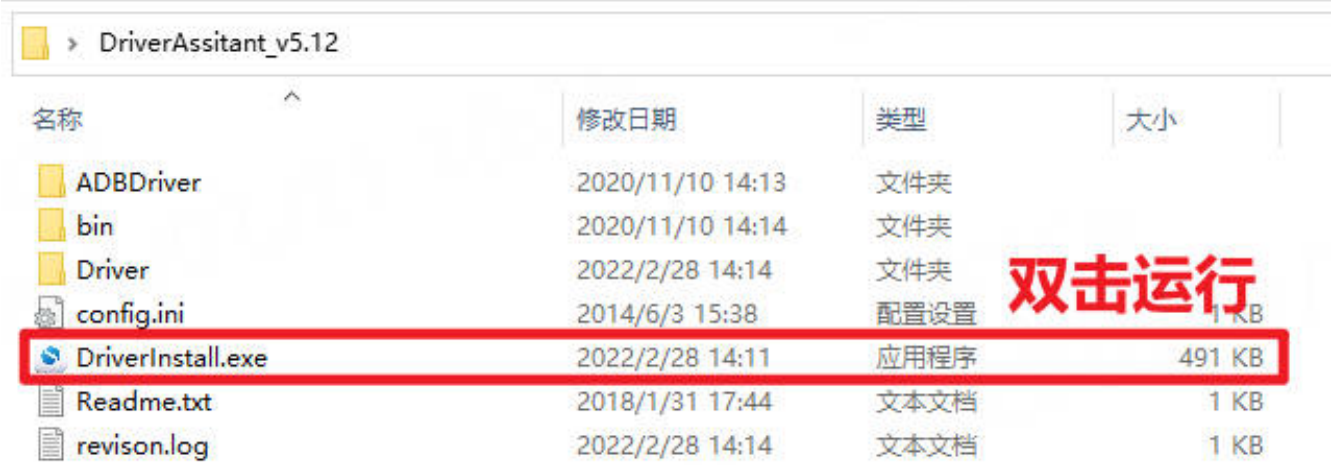
下面介绍烧录步骤

4.2.1 安装 windows 驱动

如果你是电脑上是第一次使用瑞芯微的芯片进行开发，那么你需要安装驱动。

安装步骤如下：

- 找到驱动文件 `DriverAssitant_v5.12.zip`，可以在百度网盘上下载
- 解压驱动文件，然后双击运行 `DriverInstall.exe` 安装驱动



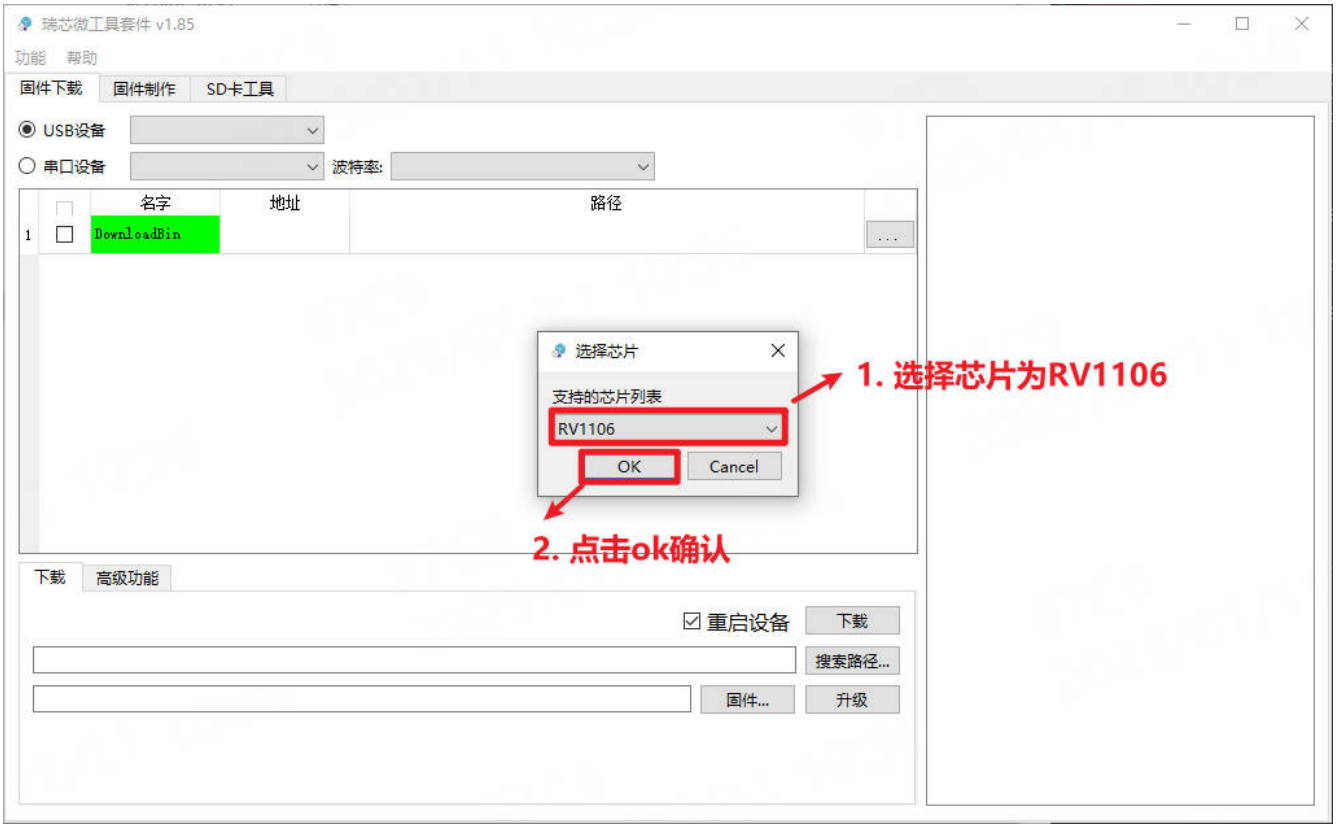
- 安装时需要先点击卸载驱动，然后再安装驱动。

- 安装成功后会显示以下图片，即安装成功

4.2.2 安装烧录软件

- 找到烧录软件 `SocToolKit.zip`，可以在百度网盘上下载
- 解压成文件夹，然后双击运行 `SocToolKit.exe`

- 进入烧录工具后会弹出小窗，在小窗这里选择芯片 RK1106，然后点击 OK

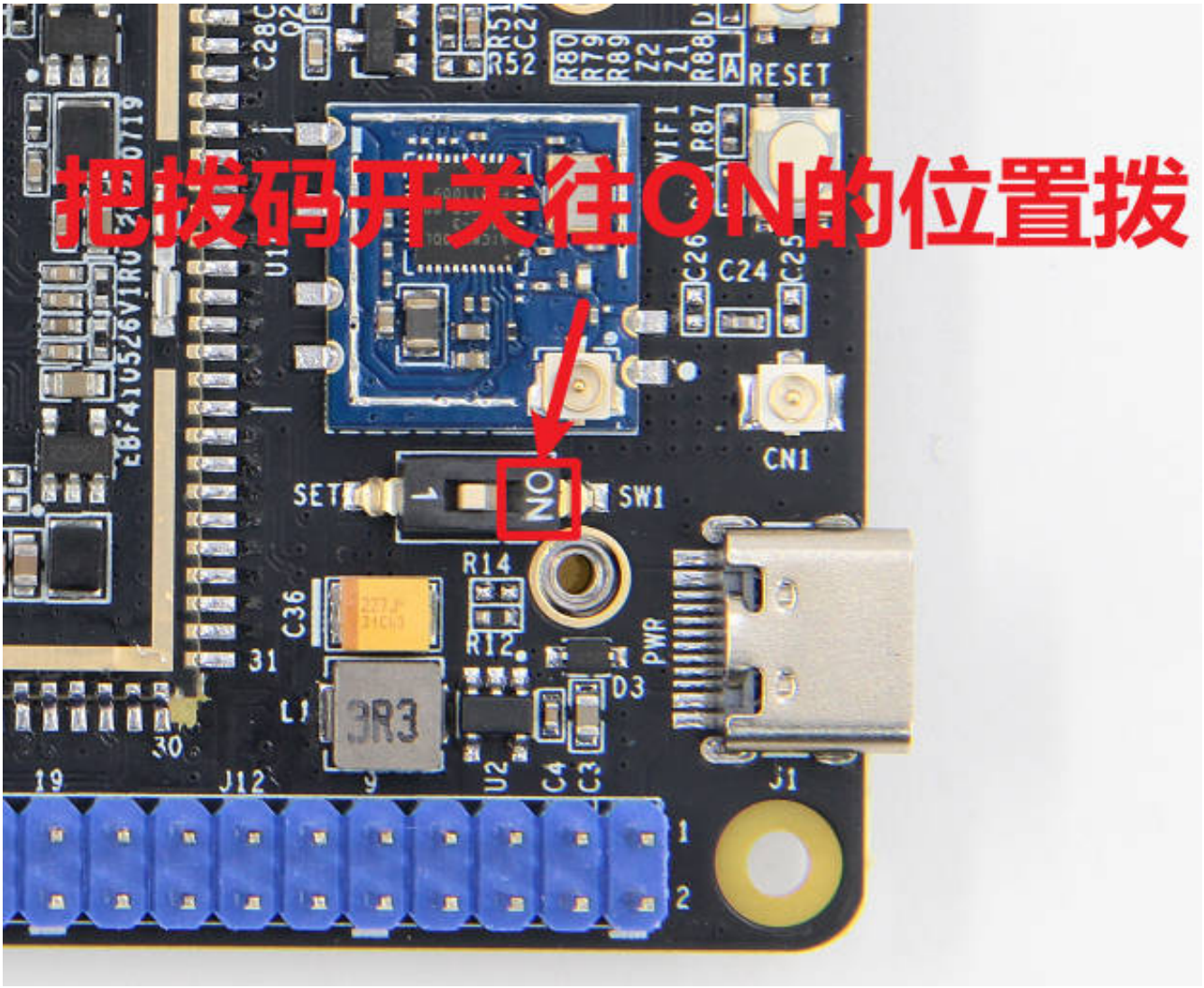


4.2.3 连接板卡

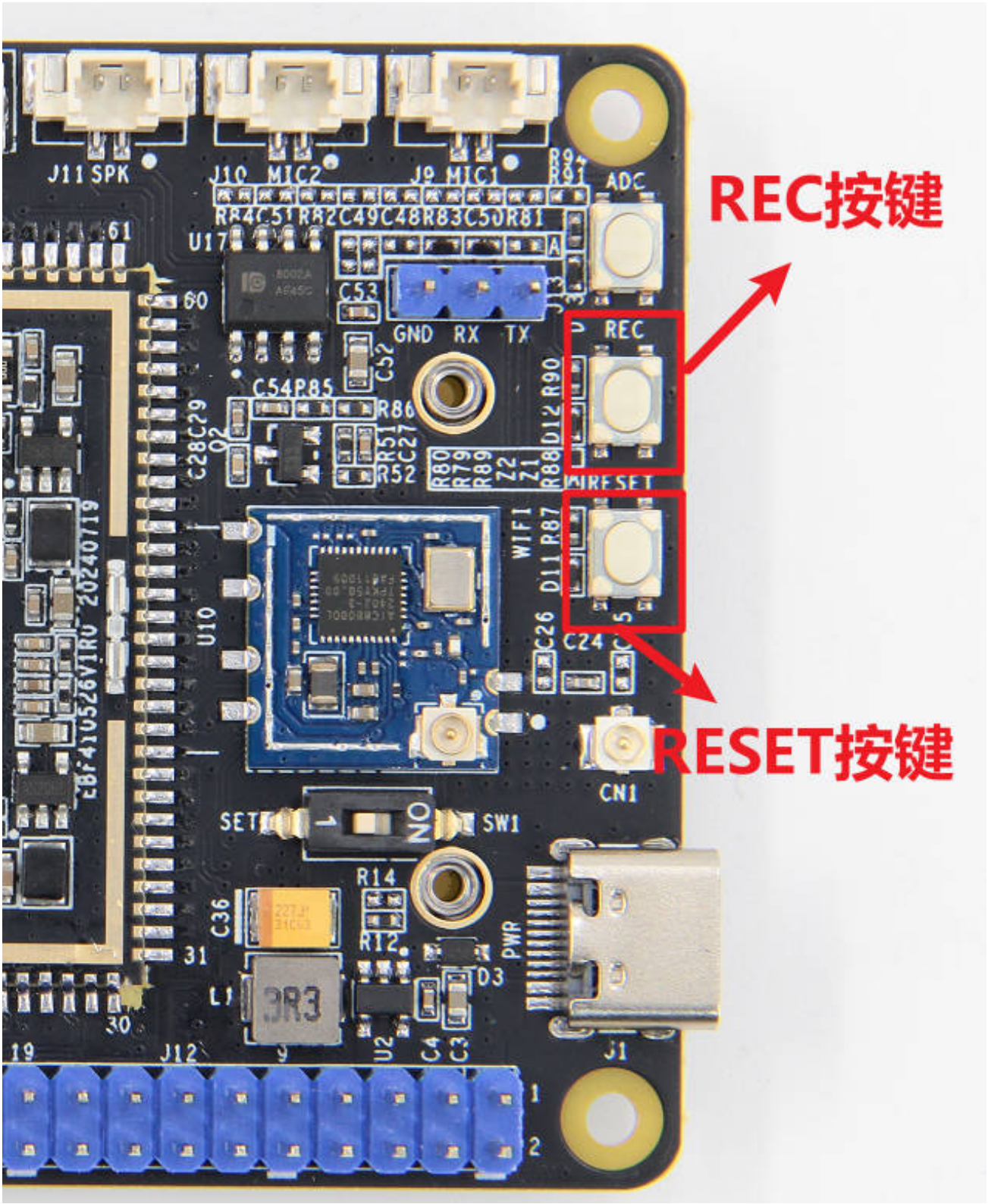
连接板卡有两种方法：

4.2.3.1 方法一

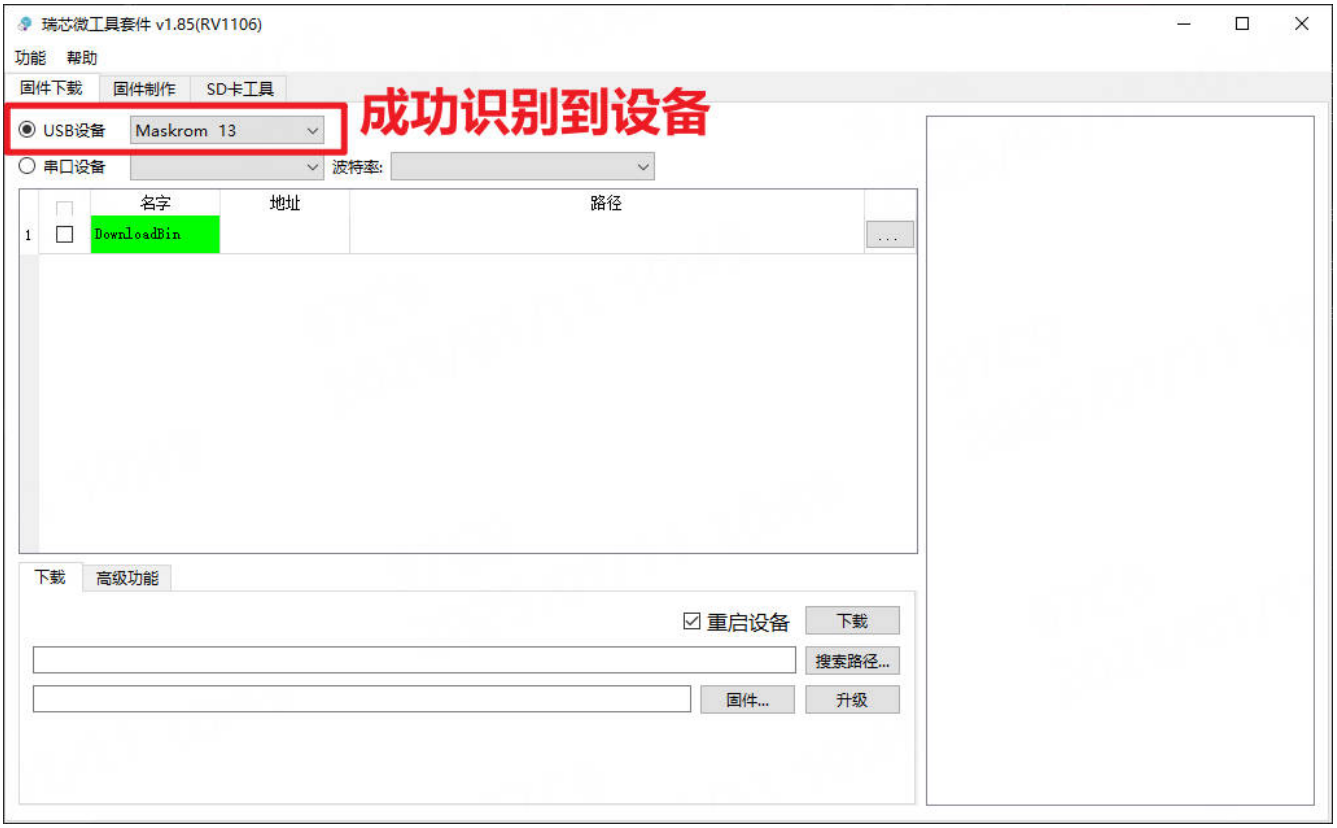
- 先断开板卡的所有供电
- 将板卡上的拨码开关拨到 ON 位置


把拨码开关往ON的位置拨
RESET
CN1
SET
SW1
ON
R14
C36
PWR
D3
L1
3R3
J12
J1
U2
C24

- 按下板子上的 REC 键


REC按键
RESET按键
J11 SPK
J10 MIC2
J9 MIC1
ADC
REC
RESET
GND RX TX
WIFI
SET
NO
SW1
CN1
J1
J12
PWR

- 用带 usb 通讯功能的 type-c 数据线连接板卡和电脑
- 连接后，烧录软件会识别到设备了



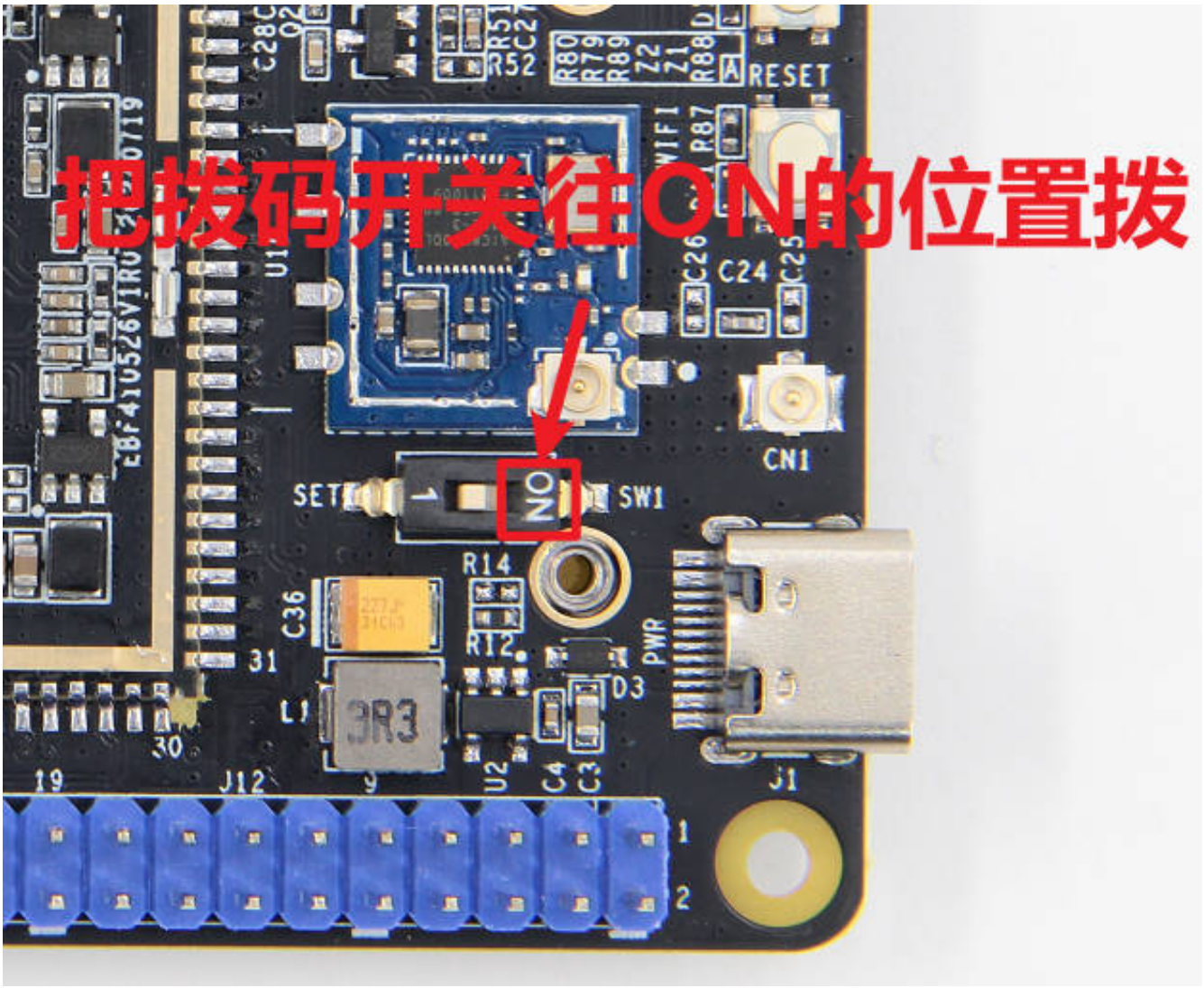
- 识别到设备后就可以松开按键了

警告： 如果按照上面的方法连接后，烧录软件没有识别到设备，那么可以尝试下面的方法

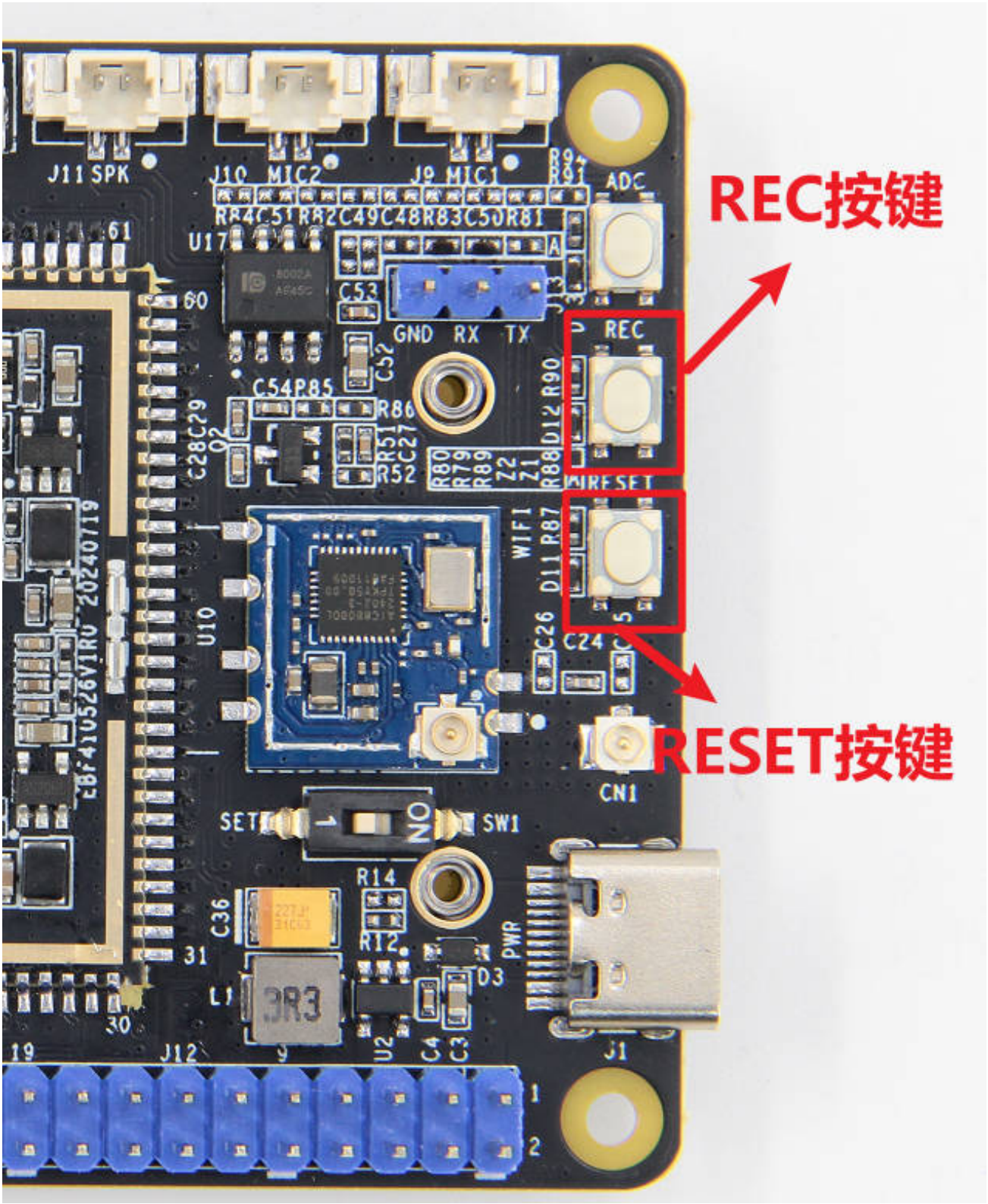
- 重新执行上面操作
- 换根数据线
- 重新安装驱动

4.2.3.2 方法二

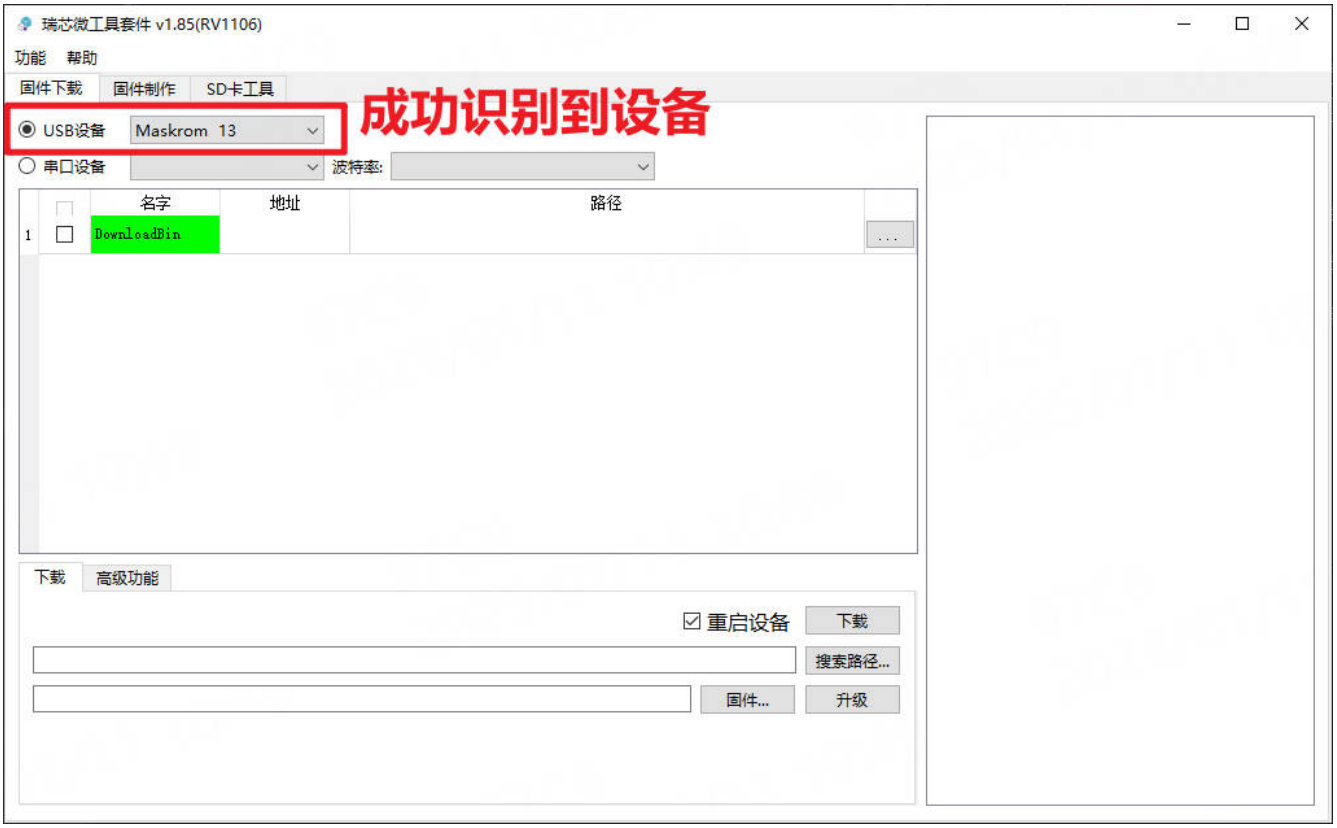
- 先连接上电源的供电，用带 usb 通讯功能的 type-c 数据线连接板卡和电脑
- 将板卡上的拨码开关拨到 ON 位置



- 先按下板子上的 REC 键，然后按下板子上的 RESET 键，再松开 RESET 键


REC按键
RESET按键
J11 SPK
J10 MIC2
J9 MIC1
ADC
REC
RESET
GND RX TX
SET
SW1
CN1
J1
J12
PWR
WIFI

- 烧录软件会识别到设备了



- 识别到设备后就可以松开 REC 按键

警告： 如果按照上面的方法连接后，烧录软件没有识别到设备，那么可以尝试下面的方法

- 重新执行上面操作
- 换根数据线
- 重新安装驱动

4.2.4 获取镜像

镜像会在百度网盘上有提供，下载地址如下：

需要选择 spi nand 的镜像版本进行下载，不然无法烧录到里面

一键烧录需要的镜像如下：

表 1: 一键烧录镜像

镜像名称	描述
update.img	一键烧录镜像 (必须)

分区烧录需要的镜像如下：

表 2: 分区烧录镜像

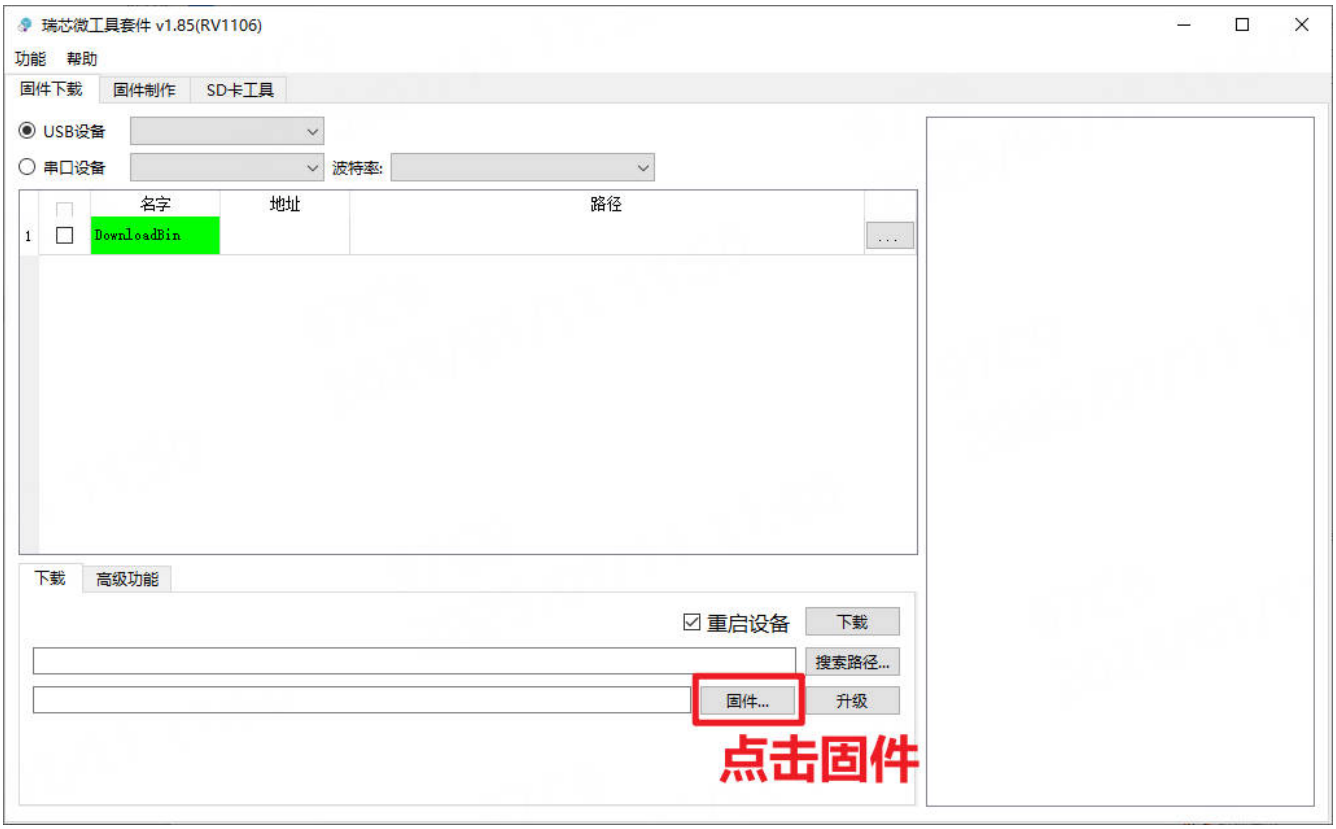
镜像名称	描述
download.bin	下载镜像需要使用（必须）
env.img	环境分区（必须）
idblock.img	可选
uboot.img	Uboot 分区（可选）
boot.img	boot 分区（可选）
rootfs.img	文件系统分区（可选）

4.2.5 烧录镜像

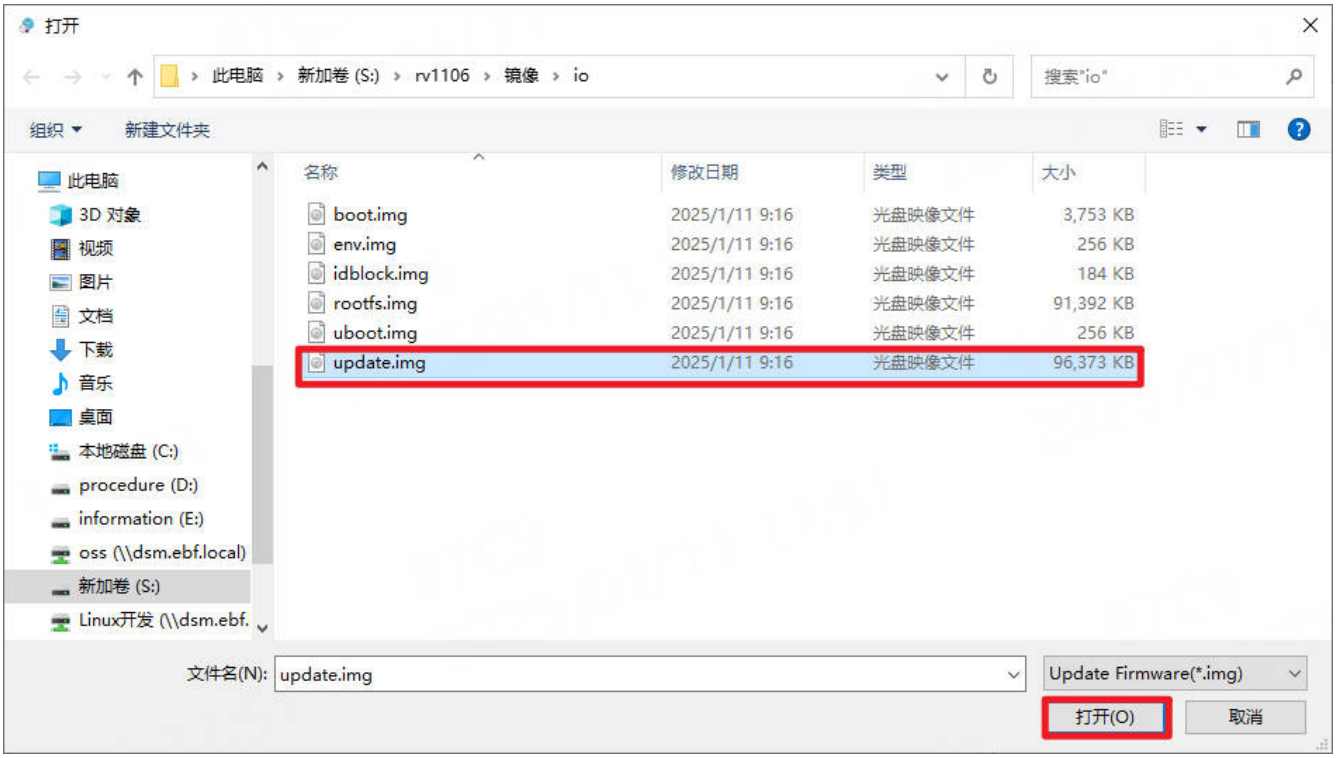
烧录镜像有两种方法：分区烧录和一键烧录

4.2.5.1 一键烧录

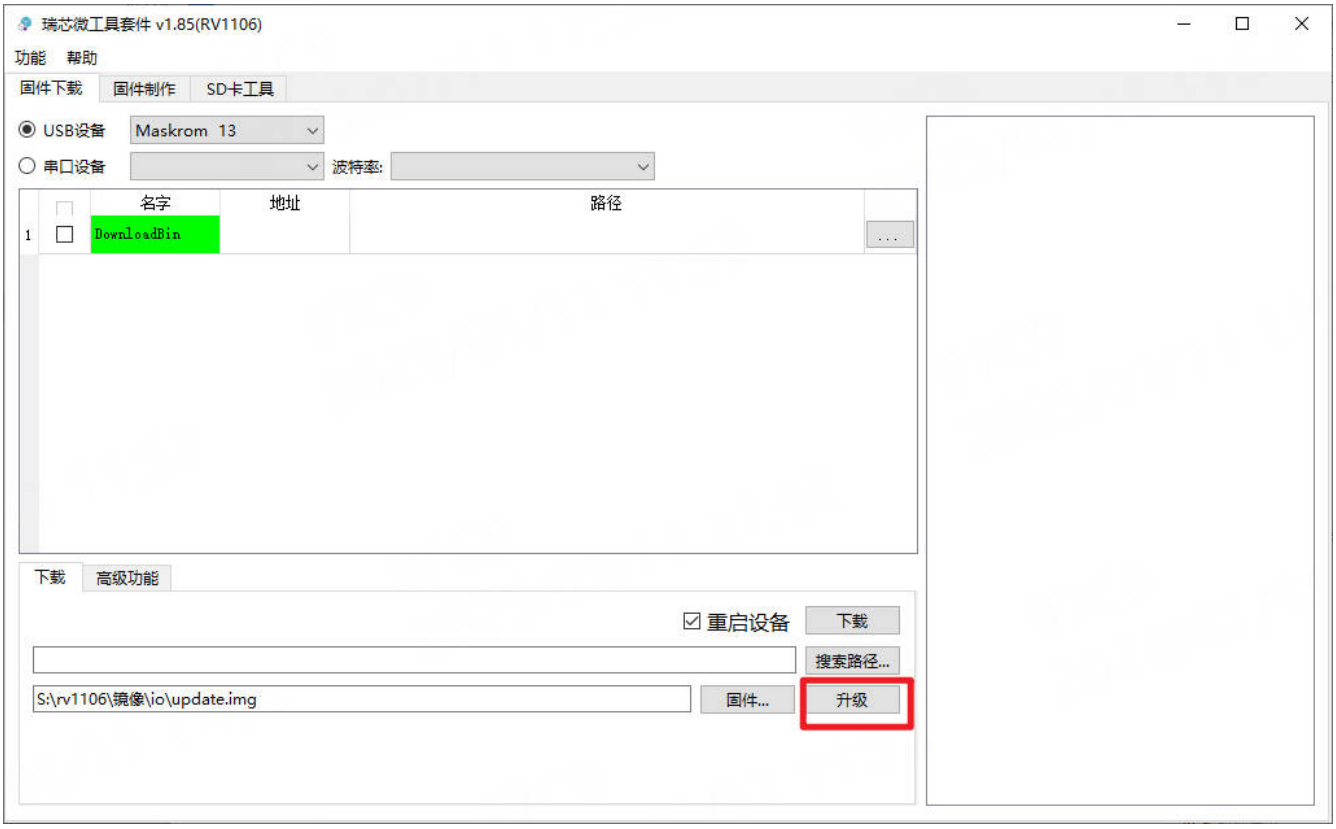
- 找到你要下载的镜像路径
- 点击烧录工具的下方的 固件按钮



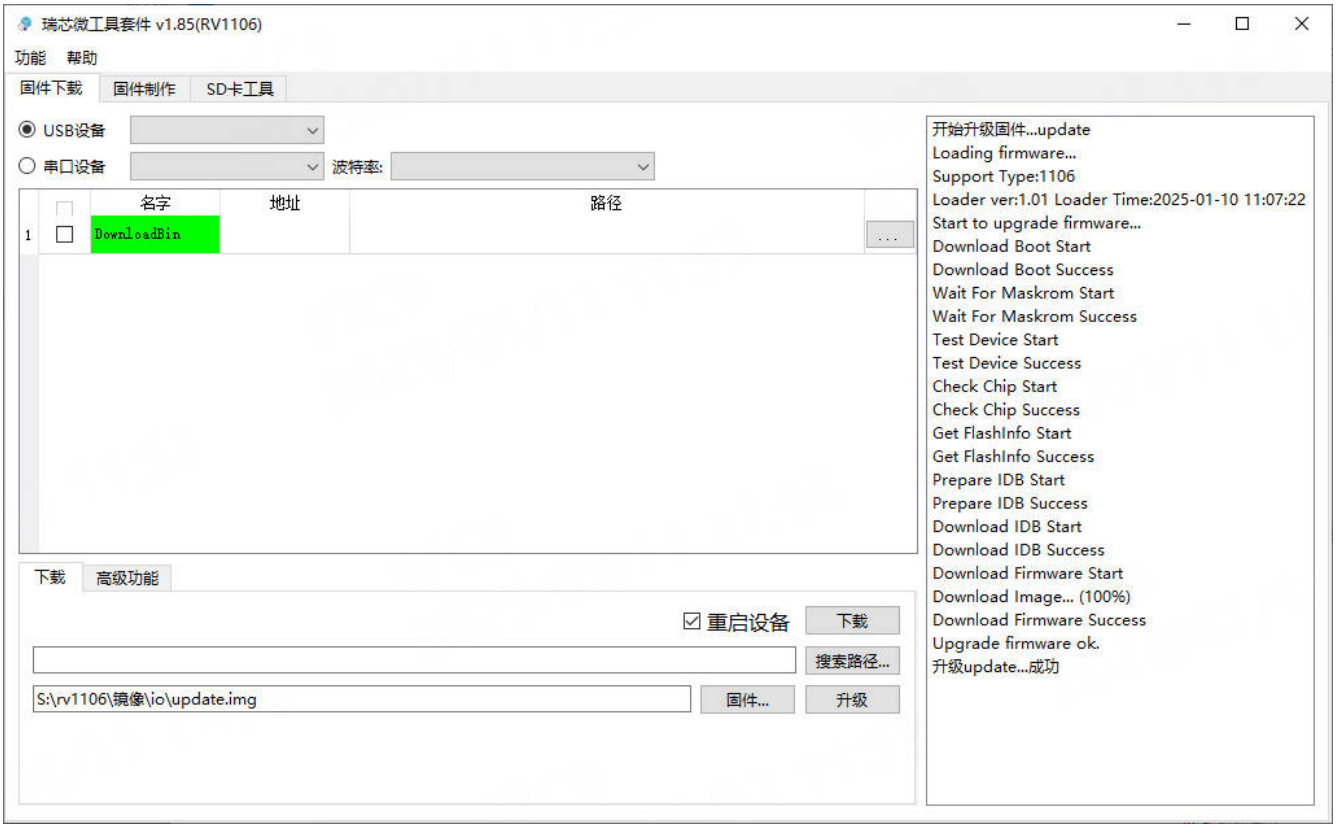
- 然后选择固件 `update.img` ， 点击 打开


打开
此电脑 › 新加卷 (S:) › rv1106 › 镜像 › io
搜索"io"
组织
新建文件夹
此电脑
3D 对象
视频
图片
文档
下载
音乐
桌面
本地磁盘 (C:)
procedure (D:)
information (E:)
oss (\\dsm.ebf.local)
新加卷 (S:)
Linux开发 (\\dsm.ebf.
名称
修改日期
类型
大小
boot.img 2025/1/11 9:16 光盘映像文件 3,753 KB
env.img 2025/1/11 9:16 光盘映像文件 256 KB
idblock.img 2025/1/11 9:16 光盘映像文件 184 KB
rootfs.img 2025/1/11 9:16 光盘映像文件 91,392 KB
uboot.img 2025/1/11 9:16 光盘映像文件 256 KB
update.img 2025/1/11 9:16 光盘映像文件 96,373 KB
文件名(N): update.img
Update Firmware(*.img)
打开(O)
取消

- 点击 升级进行烧录


瑞芯微工具套件 v1.85(RV1106)
功能 帮助
固件下载
固件制作
SD卡工具
USB设备
Maskrom 13
串口设备
波特率:
名字
地址
路径
1
DownloadBin
...
下载
高级功能
重启设备
下载
搜索路径...
S:\rv1106\镜像\io\update.img
固件...
升级

- 烧录成功如下图所示



4.2.5.2 分区烧录

警告： 分区烧录只适用于板卡上的分区分布和分区烧录的分区分布一致的情况，否则烧录后可能会导致启动不起来

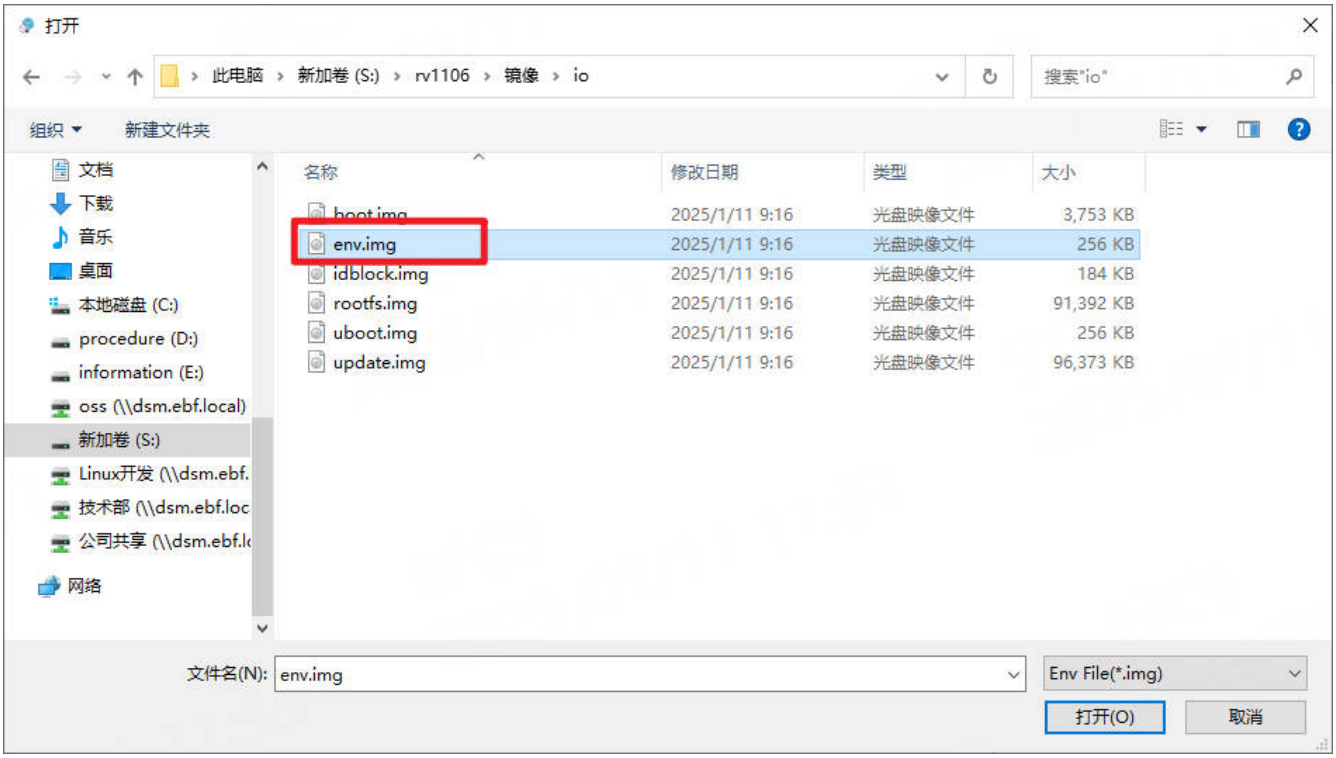
- 分区烧录是指只烧录镜像中的某一个分区，比如只烧录 `boot` 分区，或者只烧录 `rootfs` 分区。
- 分区烧录的前提是板卡上已经有一个完整的镜像，不然分区烧录后可能会导致启动不起来的情况。

烧录步骤如下：

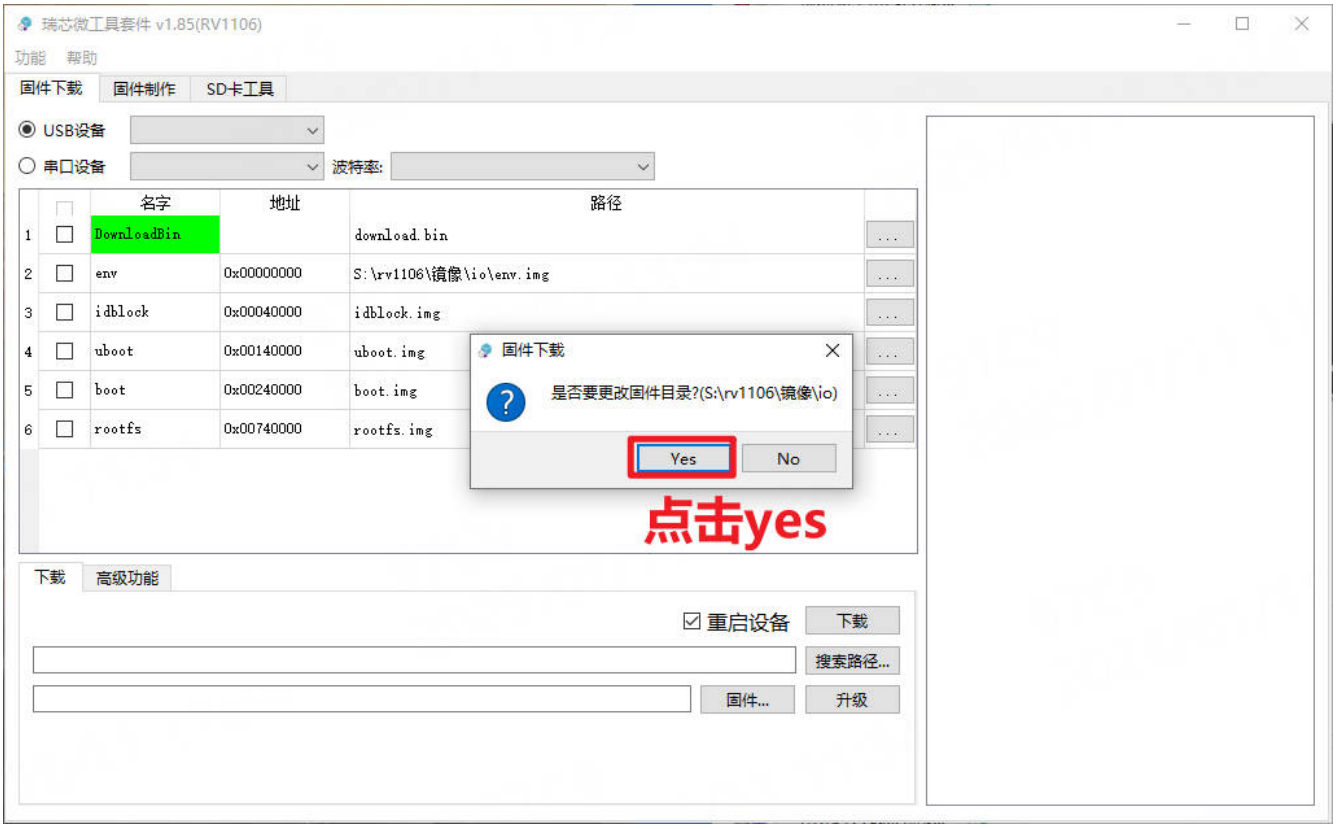
- 先确认下载分区镜像的路径，在烧录软件上蓝框区域右击显示二级菜单，然后点击 `env` 或

parameter 创建下载项

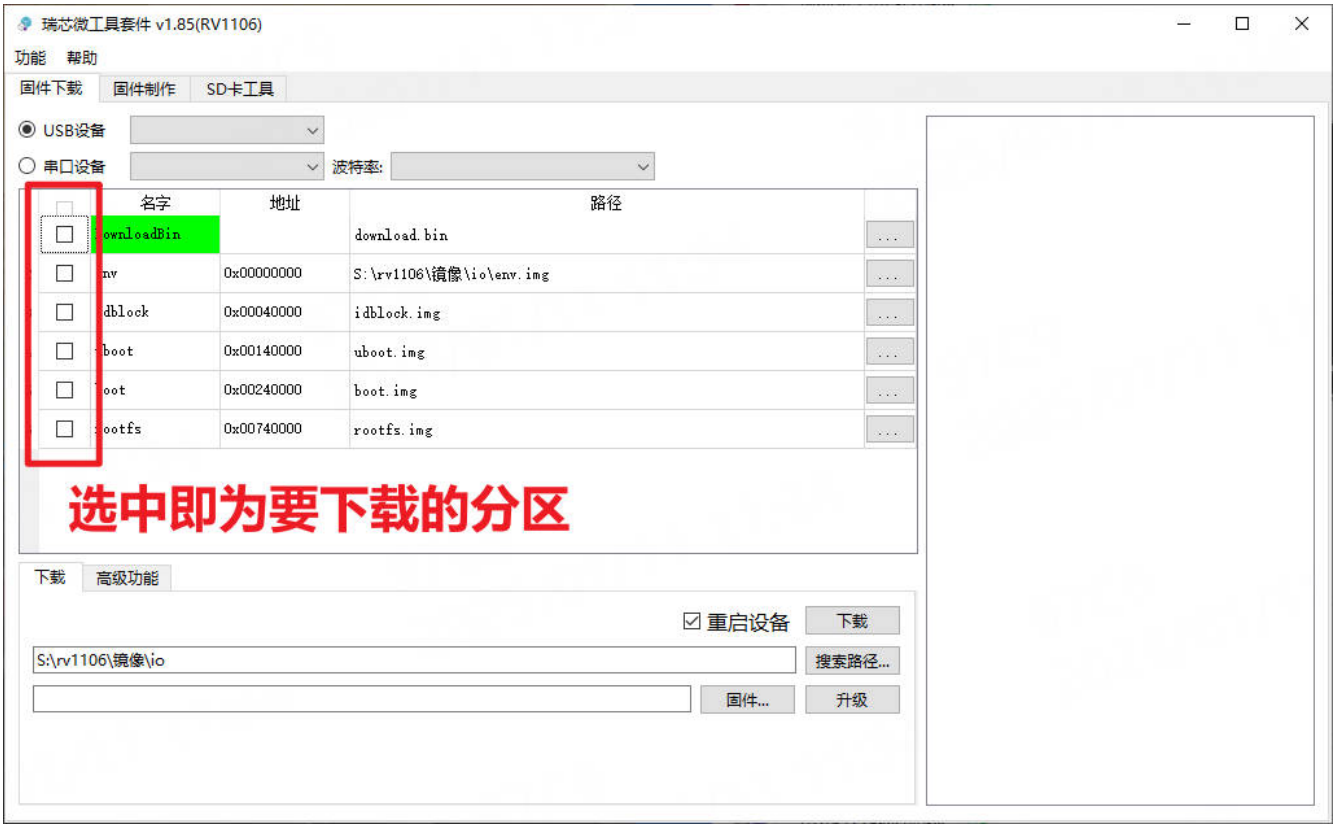
- 然后进入到文件管理器，选择下载镜像的路径，点击 env.img 文件，然后点击 打开



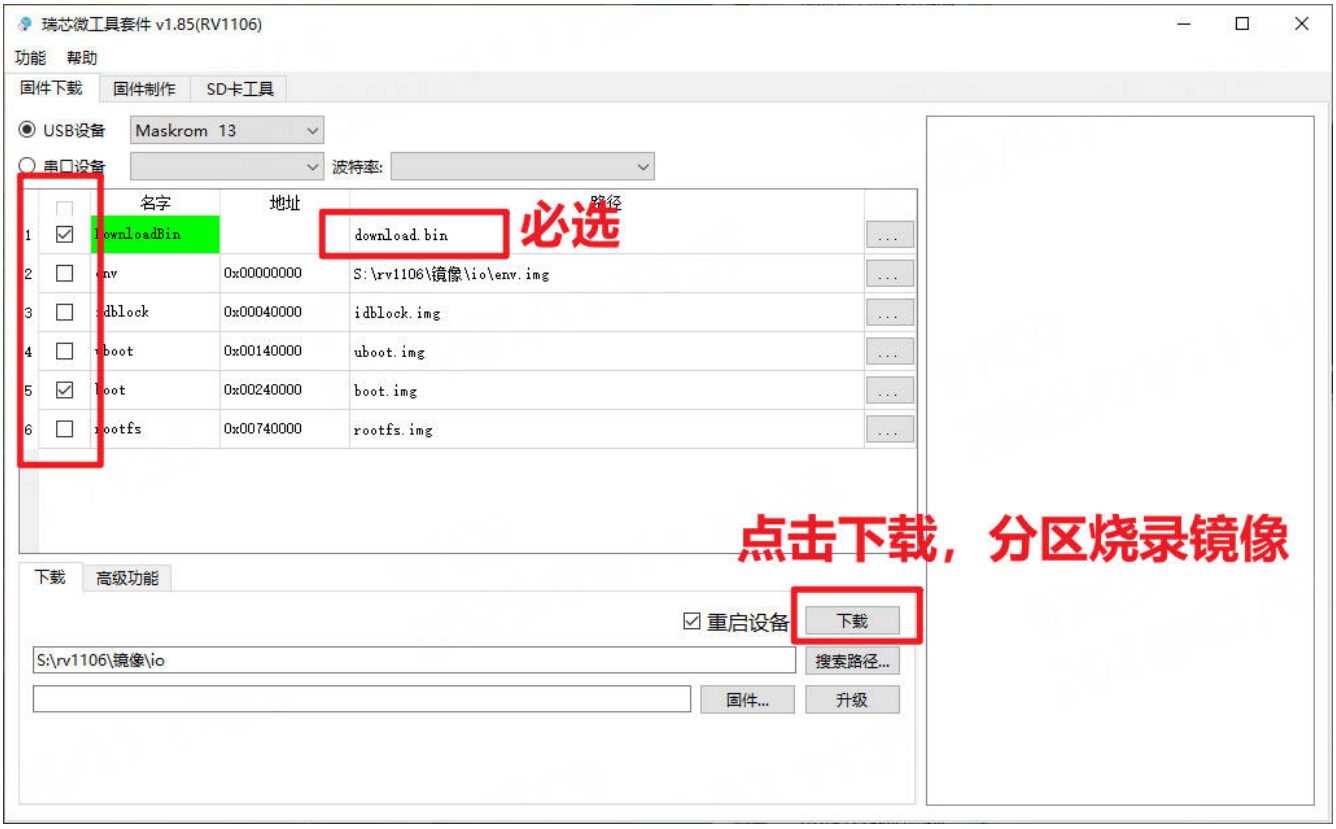
- 然后会弹出一个窗口，点击 yes



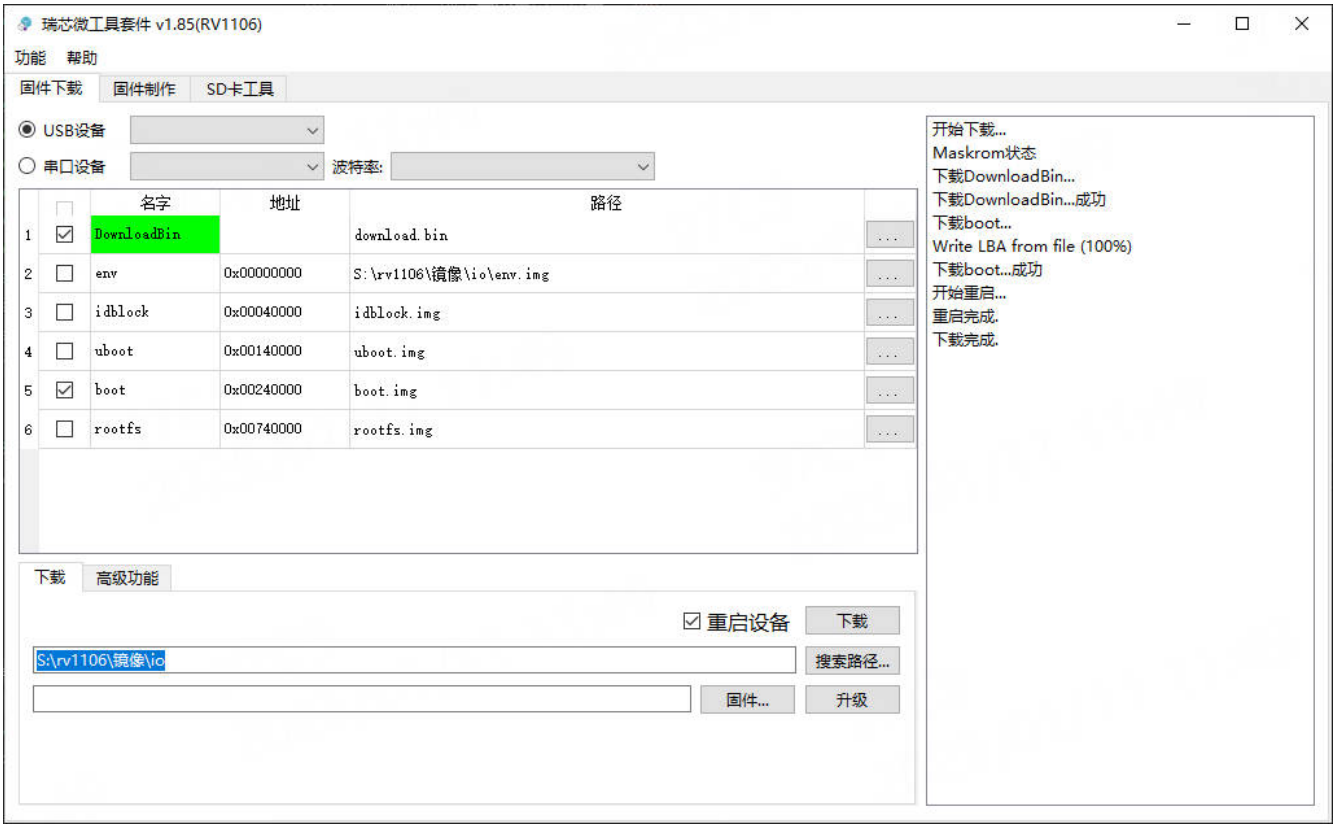
- 分区烧录的软件位置就会出现分区列表，左边的框框选中就为选择下载分区



- download.bin 文件是必选的，然后选择其他要下载的分区，点击下载即可


瑞芯微工具套件 v1.85(RV1106)
功能 帮助
固件下载 固件制作 SD卡工具
USB设备 Maskrom 13
串口设备 波特率:
名字 地址 路径
1 DownloadBin download.bin
2 env 0x00000000 S:\rv1106\镜像\io\env.img
3 idblock 0x00040000 idblock.img
4 uboot 0x00140000 uboot.img
5 boot 0x00240000 boot.img
6 rootfs 0x00740000 rootfs.img
必选
点击下载，分区烧录镜像
下载 高级功能
重启设备
下载
S:\rv1106\镜像\io
搜索路径...
固件...
升级

- 下图为下载成功


瑞芯微工具套件 v1.85(RV1106)
功能 帮助
固件下载 固件制作 SD卡工具
USB设备
串口设备
波特率:
名字 地址 路径
1 DownloadBin download.bin
2 env 0x00000000 S:\rv1106\镜像\io\env.img
3 idblock 0x00040000 idblock.img
4 uboot 0x00140000 uboot.img
5 boot 0x00240000 boot.img
6 rootfs 0x00740000 rootfs.img
下载 高级功能
重启设备
下载
S:\rv1106\镜像\io
搜索路径...
固件...
升级
开始下载...
Maskrom状态
下载DownloadBin...
下载DownloadBin...成功
下载boot...
Write LBA from file (100%)
下载boot...成功
开始重启...
重启完成.
下载完成.

4.2.6 串口烧录

第 5 章　启动

板卡启动的时候，瞬间电流会比较大，建议使用电源电流大于 1A 的电源供电。

如果接入其他更多外设则需要使用能提供更大电流的电源适配器或者能输出更大电流的 usb 接口

如果是使用电脑供电：

1. 建议使用电脑主板上的 usb 接口供电，一般是台式电脑的后背板上的 usb 接口，前面板的 usb 接口供电较弱，可能输出电流不太够。
2. 建议使用电脑主板上的 usb3.0 接口供电，usb3.0 接口输出电流较大。
3. 建议使用自带供电的拓展坞供电

5.1 启动顺序

板卡支持两种启动方式，spi-nand 启动和 TF 卡启动，TF 卡启动的优先级高于 spi-nand 启动

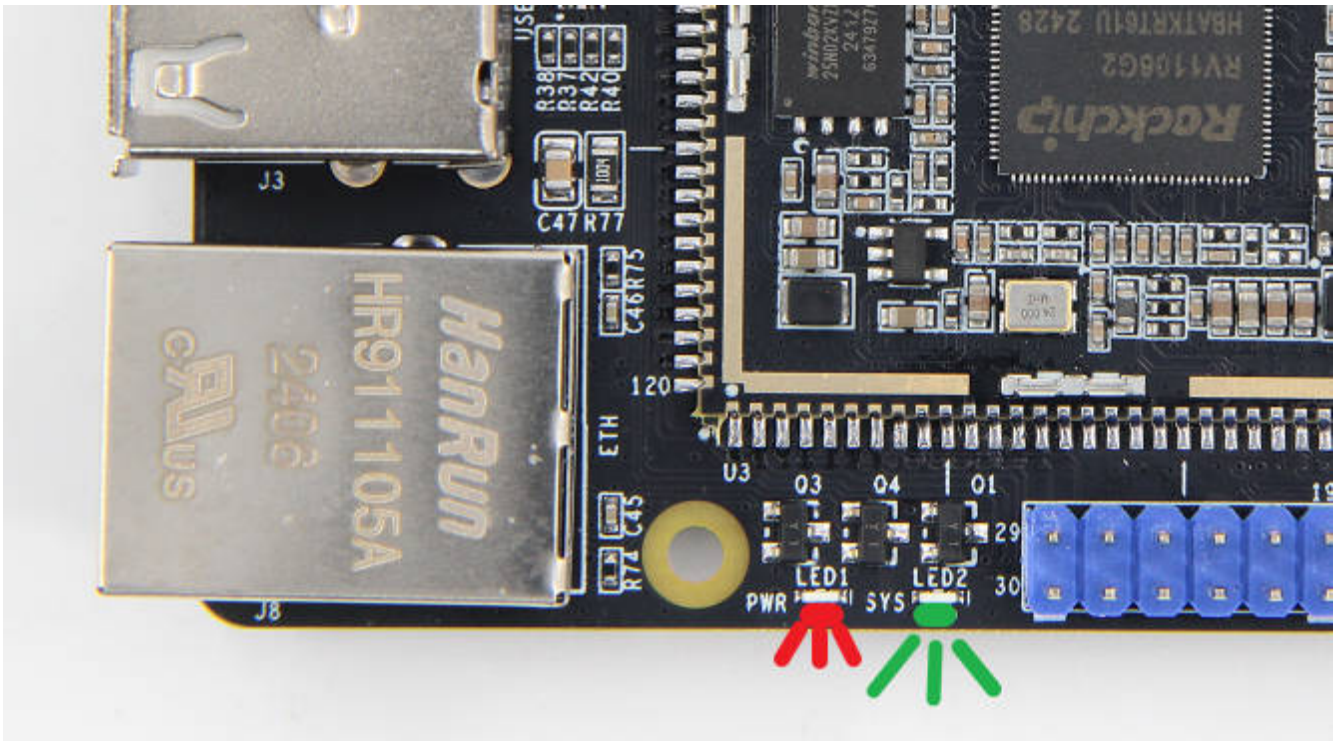
表 1: 启动顺序

spi-nand	TF 卡	启动方式
有镜像	无镜像	spi-nand 启动
无镜像	有镜像	TF 卡启动
有镜像	有镜像	TF 卡启动

5.2 板卡启动

供电和镜像烧录都准备好后，就可以启动板卡

直接插上电源，板卡会自动启动，启动成功，板上的绿灯会持续闪动



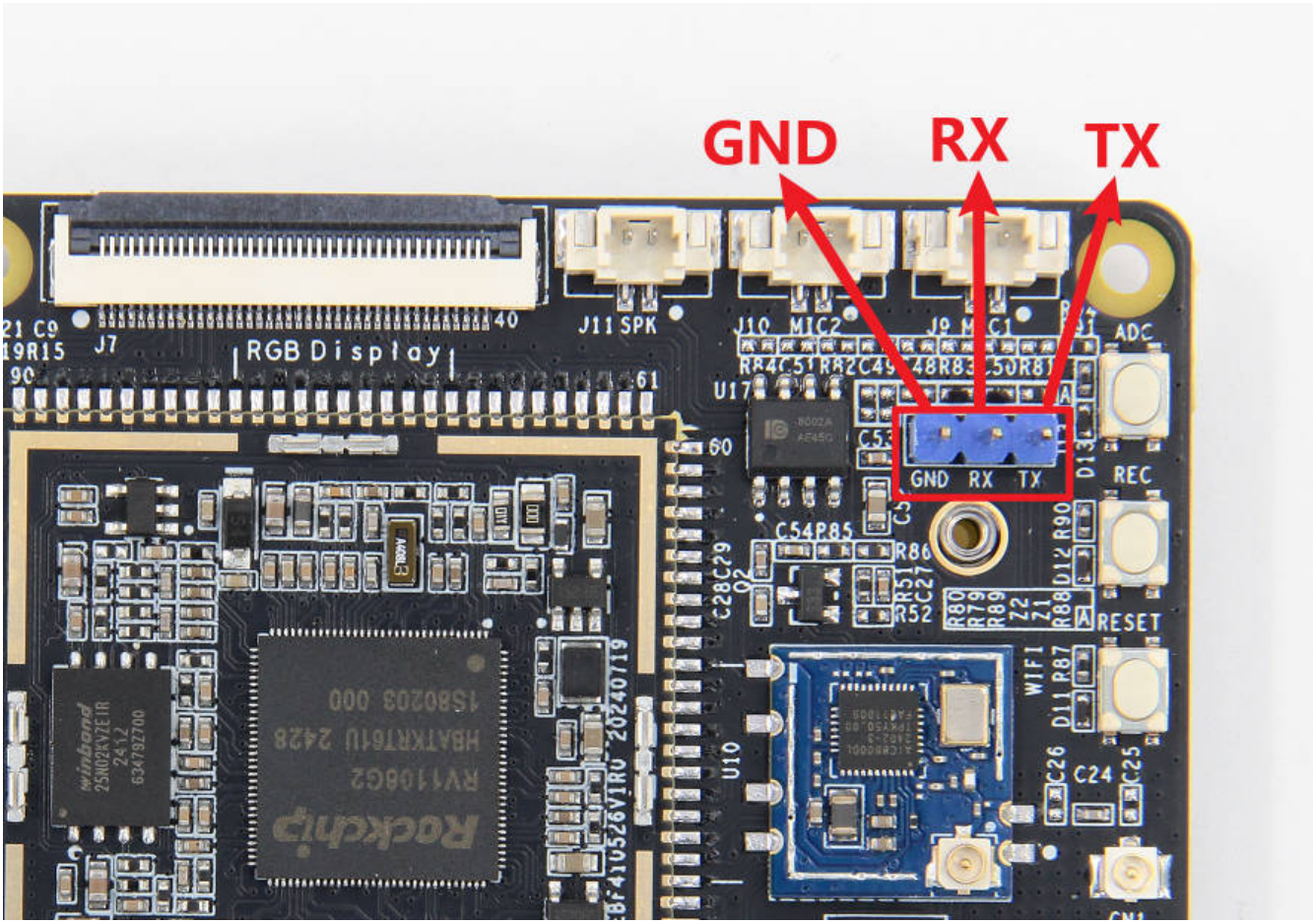
警告： 如果板卡启动后，板上的绿灯没有闪动或者绿灯闪了一下就不闪了，大概率是电源输出电流不够，需要更换更大输出电流的电源

第 6 章　串口调试

板子的所有调试信息都可以通过串口输出，串口调试是调试板子的重要手段之一。

6.1 串口位置

板卡的串口位于按键的旁边，是一个 3P 的排针，如图下所示



6.2 串口工具安装

使用串口工具之前，需要先安装串口工具

Windows 系统下常用的串口工具有：

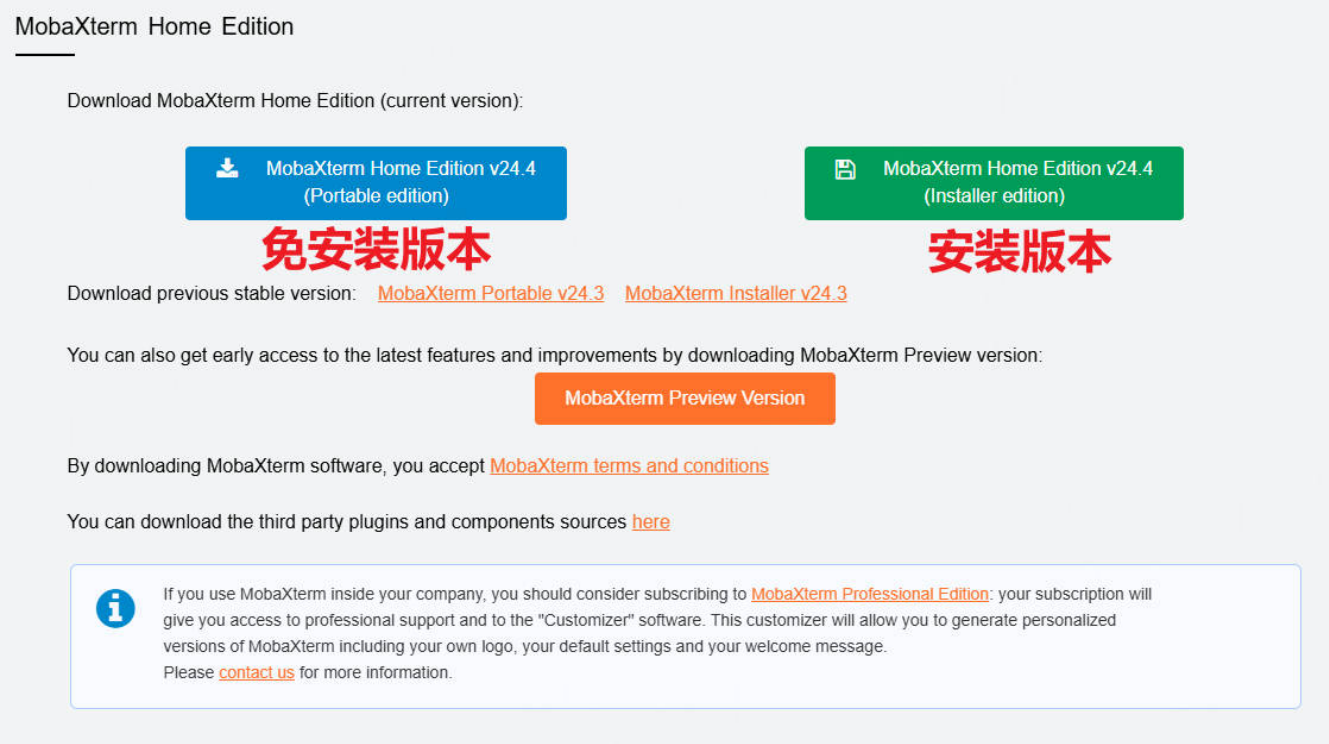
- PuTTY：支持串口、SSH、Telnet 等协议
- Tera Term：开源终端模拟器，支持串口、SSH、Telnet 等
- SecureCRT：商业工具，支持串口、SSH、Telnet 等，功能强大
- RealTerm：专注于串口通信，支持数据发送、接收和调试
- CoolTerm：简单易用的串口调试工具
- Termite：轻量级串口工具
- HTerm：开源串口工具，支持多种协议
- Serial Port Utility：适用于 Windows 的串口调试工具，支持数据发送和接收。
- ZOC Terminal：商业工具，支持串口、SSH、Telnet 等
- MobaXterm：多功能工具，支持串口、SSH、远程桌面等
- Termius：跨平台工具，支持串口、SSH 等

串口工具有很多，不同的工具有不同的特点，可以根据自己的需求选择合适的工具。

`MobaXterm` 是一个综合性比较强的工具，可以用该工具调试其他的更多的接口，这里以 `MobaXterm` 为例。

- 百度云资料中的 软件工具也带有该免安装版本的工具
- `MobaXterm` 的下载链接为

```
https://mobaxterm.mobatek.net/download-home-edition.html
```



根据自己的需要选择合适的版本下载即可。

- 安装过程就不演示了

6.3 电脑连接串口调试工具

串口调试工具有很多种，支持波特率 115200 即可，这里推荐几种串口调试工具

- CH340X USB 转 TTL 串口调试工具
- CH343 USB 转 TTL 串口调试工具
- cp210x USB 转 TTL 串口调试工具

这里以常用的 CH340X 为例

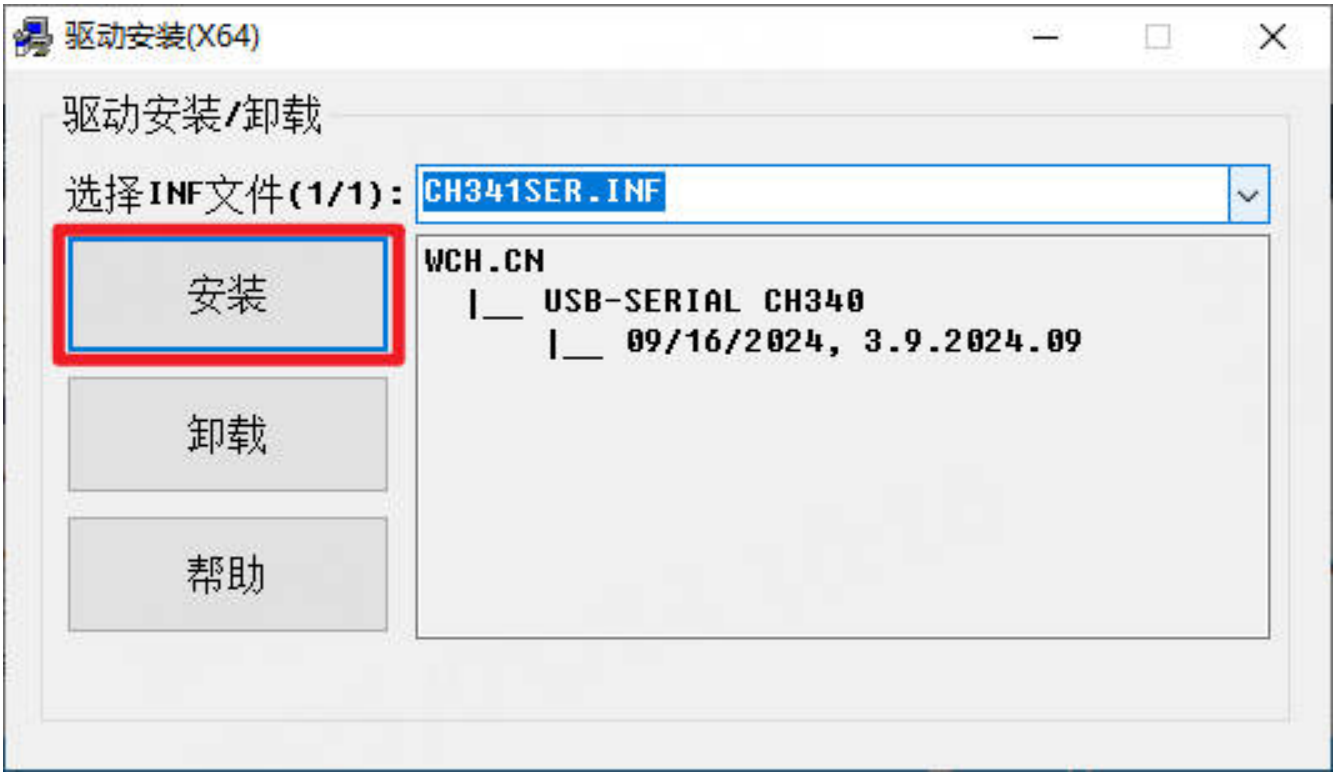
在 windows 系统下，使用 CH340X 工具，需要安装串口驱动（以前安装过则不需要再安装了）

驱动在百度云资料中的 软件工具文件夹中会有

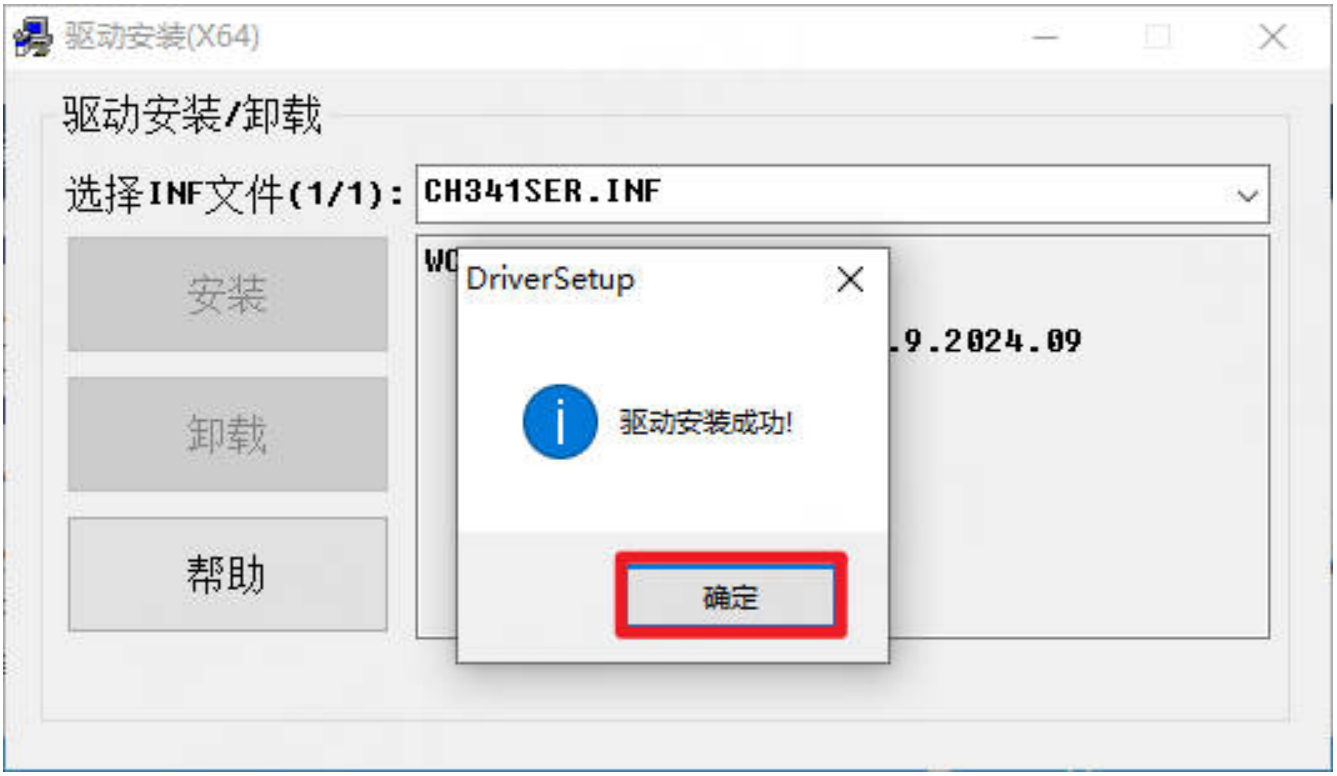
或者进入官网进行下载

```
https://www.wch.cn/downloads/CH341SER_EXE.html
```

- 双击 CH341SER.EXE 进行安装，会弹出如下界面，点击 安装即可



- 安装完成后，会弹出如下界面，点击 完成即可



在电脑上连接 CH340X 工具，进入设备管理器，可以看到如下界面

如上图所示，COM11 是我的串口调试工具的端口号，不同的电脑端口号可能不一样，需要以自己的实际出发

6.4 板卡连接串口调试工具

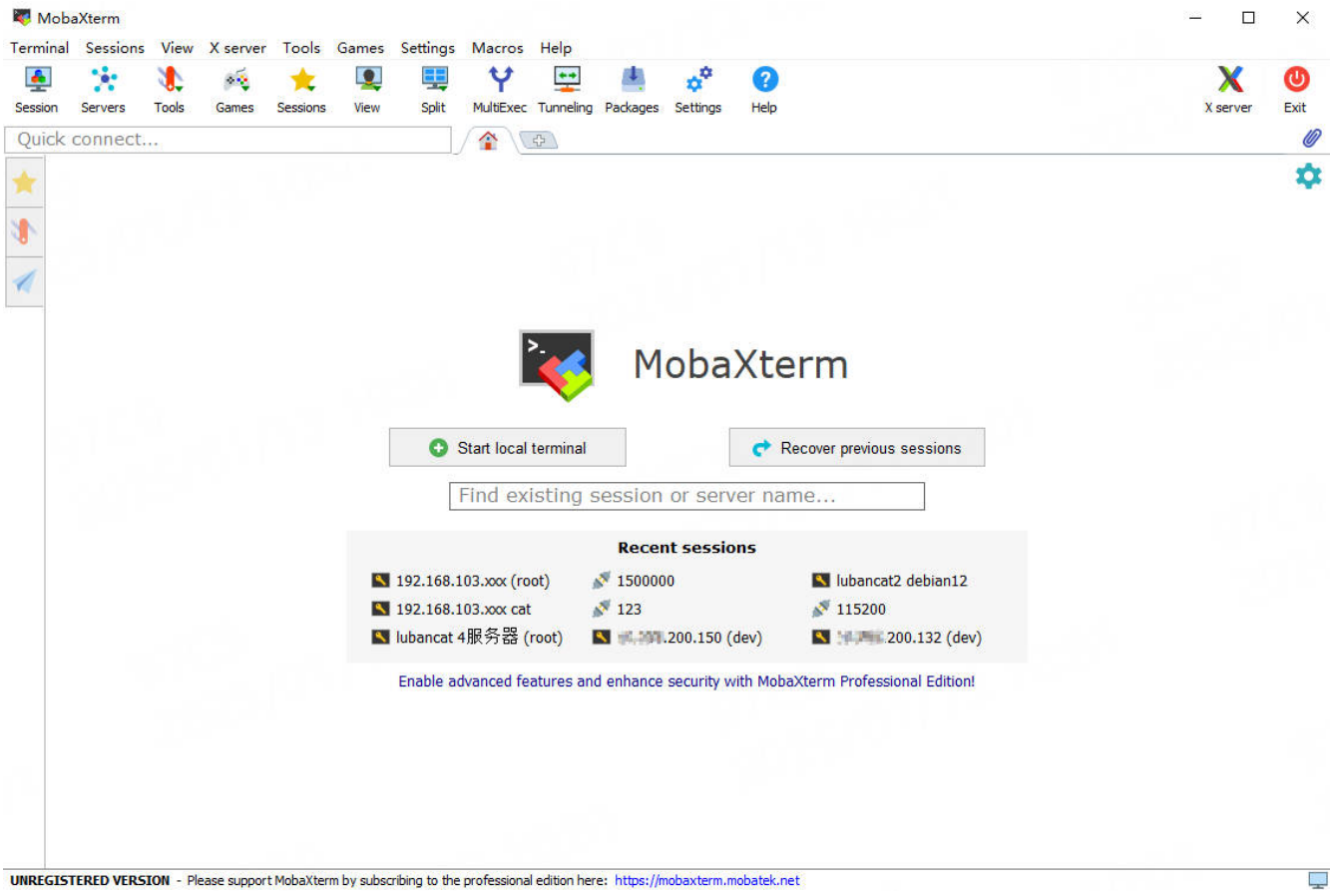
板卡连接串口调试工具需要用到杜邦线，将板卡的串口和串口调试工具连接起来，以下面表格的连接方式

表 1: 杜邦线连接方式

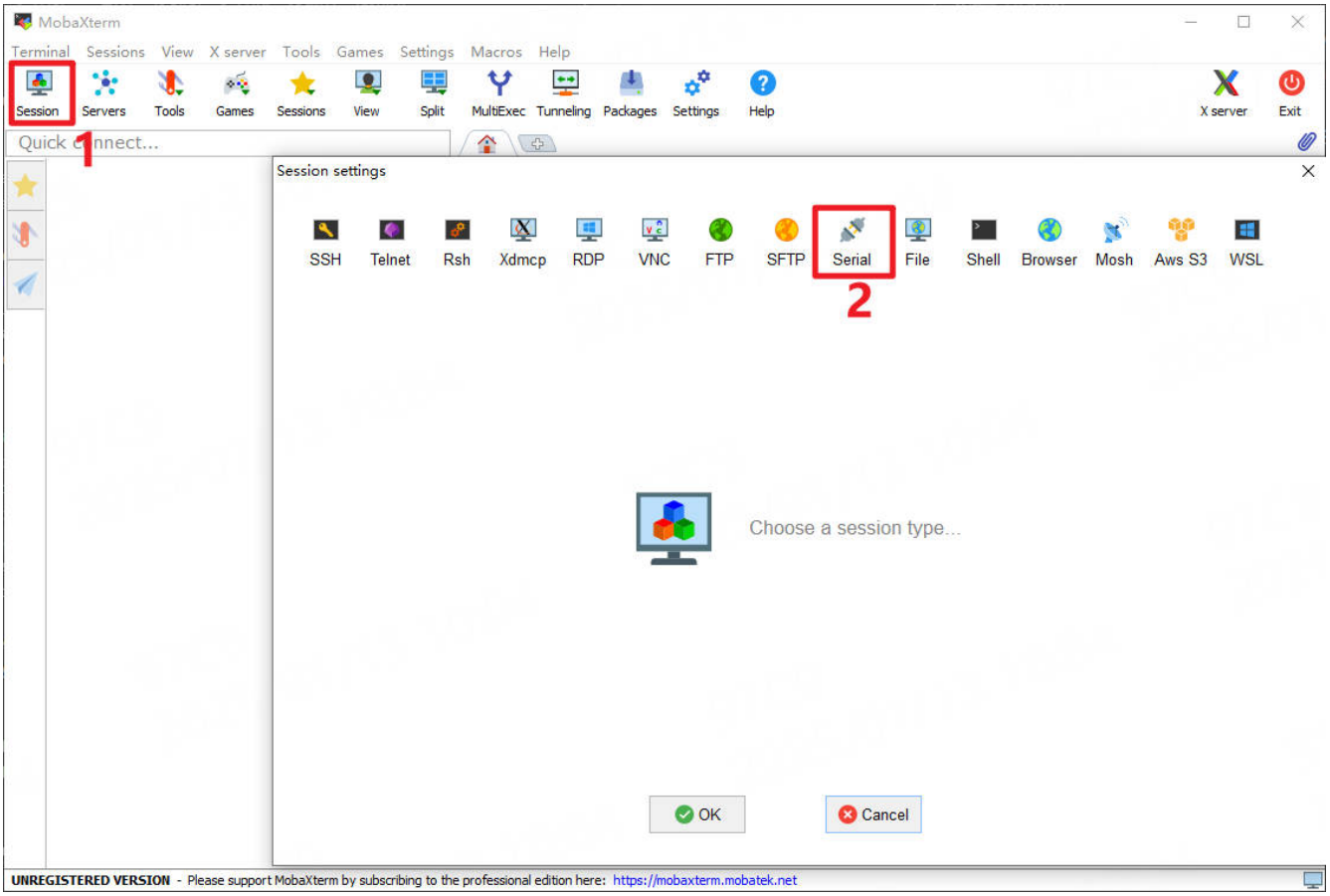
板卡	串口调试工具
GND	GND
TX	RX
RX	TX

6.5 配置串口工具

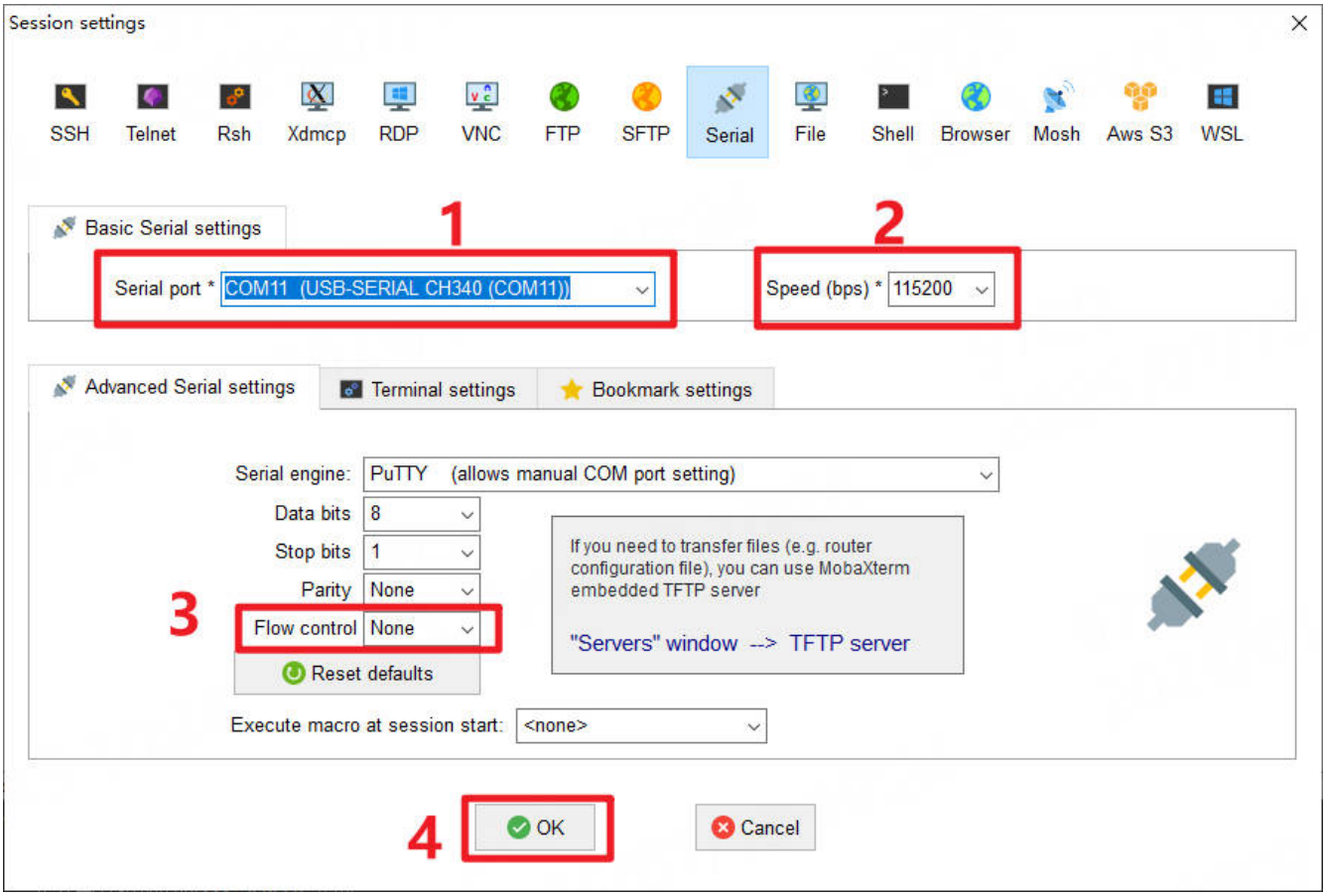
安装完成后打开工具可以看到如下界面



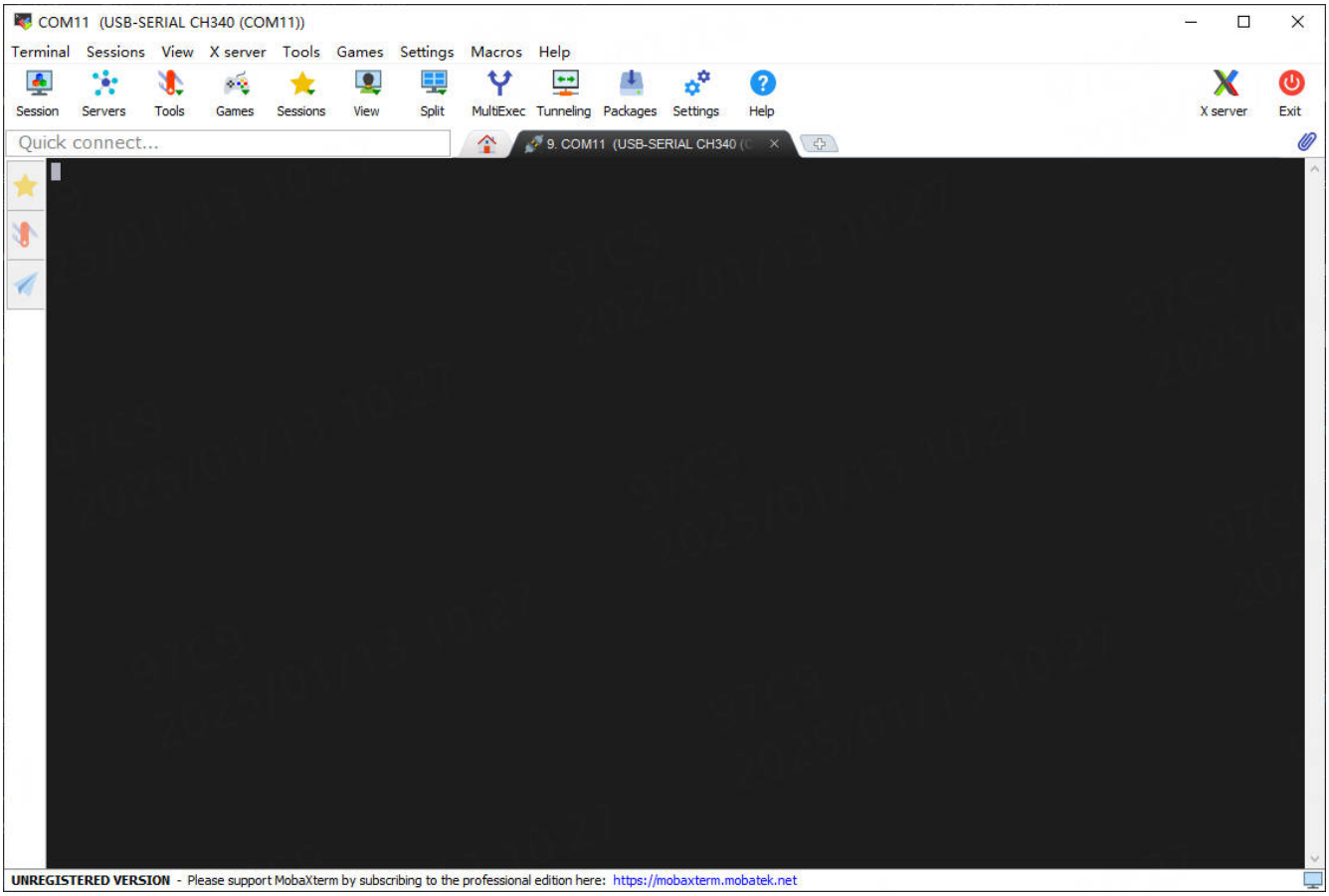
- 点击左上角的 `Session`，进入配置界面，然后点击 `Serial`，进入串口配置界面



- 修改端口号，由上文可知我的端口号是 COM11 ，所以这里填写 COM11 (需要根据自己实际情况填写)，波特率填写 115200 ，Flow control 设置为 None ，点击 OK 即可

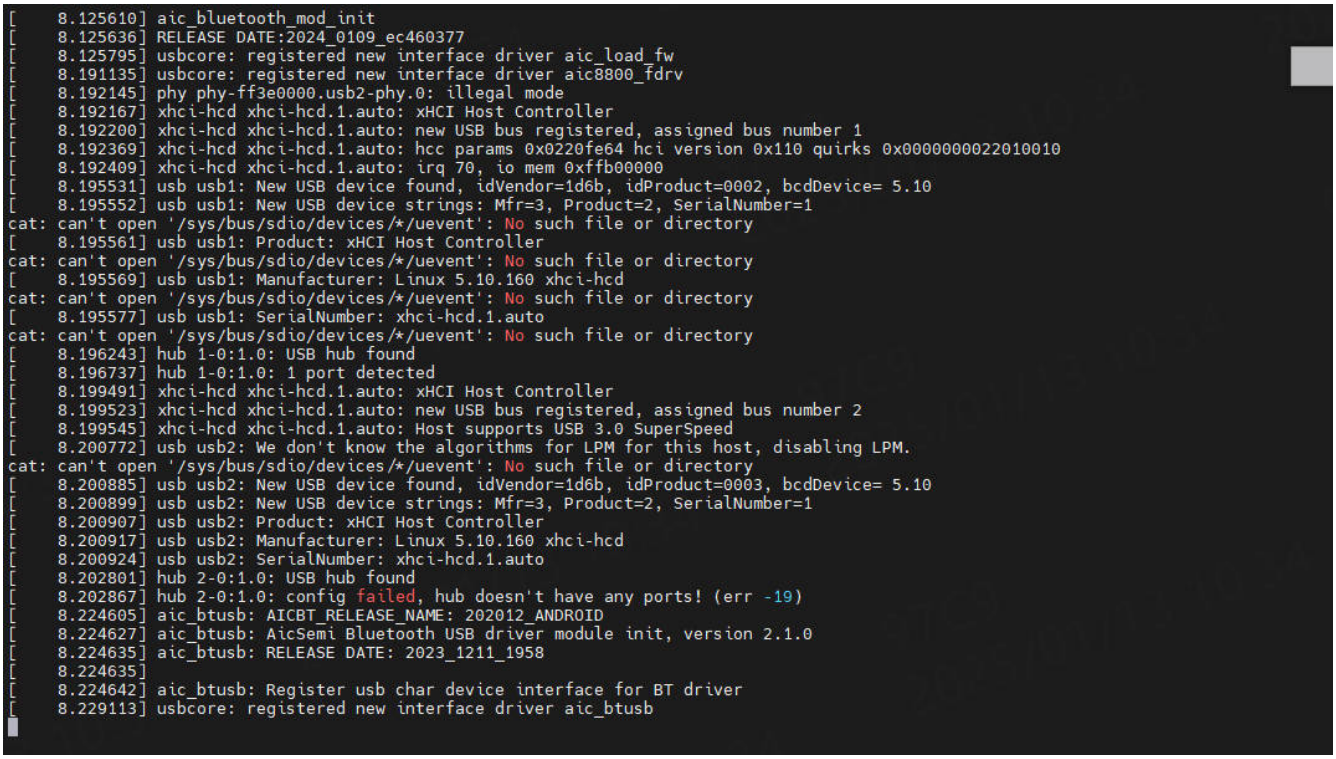


- 完成后可以看到如下界面

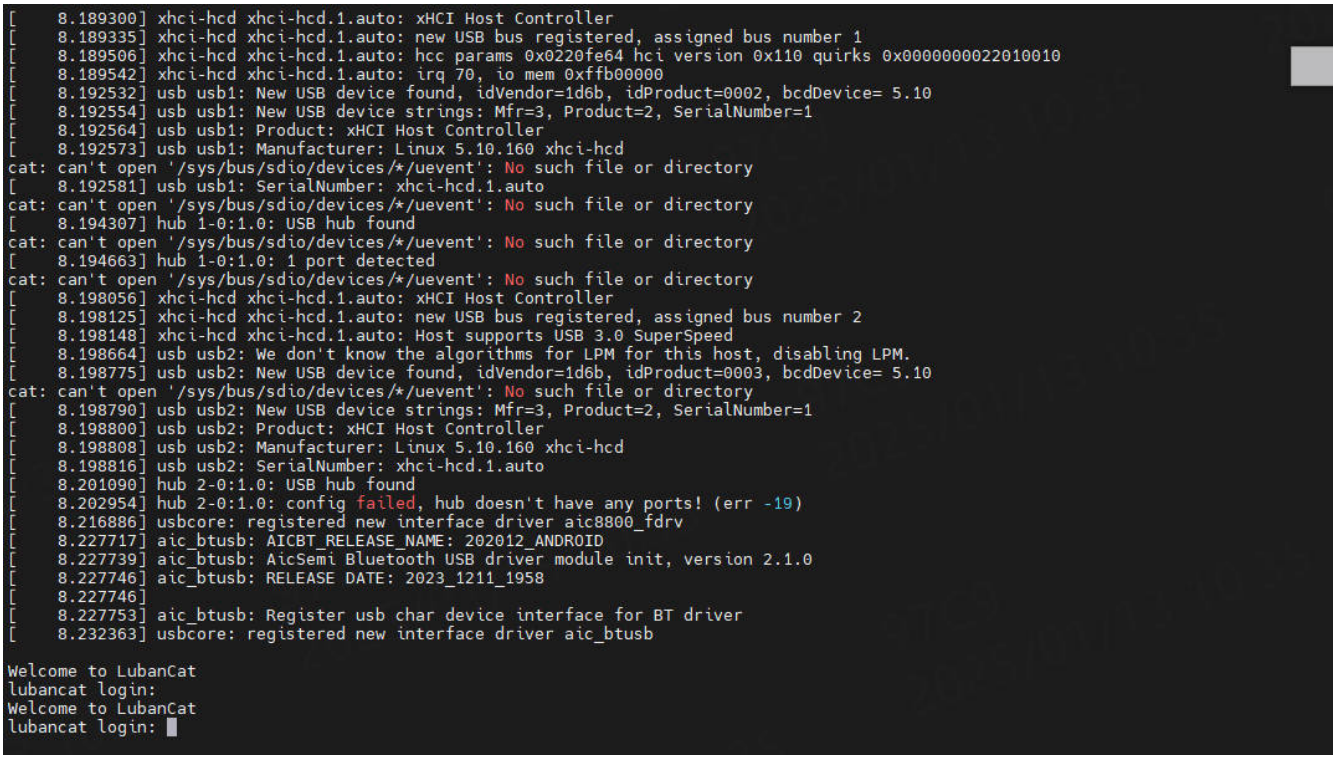


6.6 板卡上电

- 板卡上电后就能看到板卡输出打印消息



- 按下两次 回车键就可以看到登录信息了



输入用户名 root 和密码 root (输入密码时不会显示字符，输入后按 回车键即可) 即可登录

```
[    8.192532] usb usb1: New USB device found, idVendor=1d6b, idProduct=0002, bcdDevice= 5.10
[    8.192554] usb usb1: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[    8.192564] usb usb1: Product: xHCI Host Controller
[    8.192573] usb usb1: Manufacturer: Linux 5.10.160 xhci-hcd
cat: can't open '/sys/bus/sdio/devices/*/uevent': No such file or directory
[    8.192581] usb usb1: SerialNumber: xhci-hcd.1.auto
cat: can't open '/sys/bus/sdio/devices/*/uevent': No such file or directory
[    8.194307] hub 1-0:1.0: USB hub found
cat: can't open '/sys/bus/sdio/devices/*/uevent': No such file or directory
[    8.194663] hub 1-0:1.0: 1 port detected
cat: can't open '/sys/bus/sdio/devices/*/uevent': No such file or directory
[    8.198056] xhci-hcd xhci-hcd.1.auto: xHCI Host Controller
[    8.198125] xhci-hcd xhci-hcd.1.auto: new USB bus registered, assigned bus number 2
[    8.198148] xhci-hcd xhci-hcd.1.auto: Host supports USB 3.0 SuperSpeed
[    8.198664] usb usb2: We don't know the algorithms for LPM for this host, disabling LPM.
[    8.198775] usb usb2: New USB device found, idVendor=1d6b, idProduct=0003, bcdDevice= 5.10
cat: can't open '/sys/bus/sdio/devices/*/uevent': No such file or directory
[    8.198790] usb usb2: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[    8.198800] usb usb2: Product: xHCI Host Controller
[    8.198808] usb usb2: Manufacturer: Linux 5.10.160 xhci-hcd
[    8.198816] usb usb2: SerialNumber: xhci-hcd.1.auto
[    8.201090] hub 2-0:1.0: USB hub found
[    8.202954] hub 2-0:1.0: config failed, hub doesn't have any ports! (err -19)
[    8.216886] usbcore: registered new interface driver aic8800_fdrv
[    8.227717] aic_btusb: AICBT_RELEASE_NAME: 202012_ANDROID
[    8.227739] aic_btusb: AicSemi Bluetooth USB driver module init, version 2.1.0
[    8.227746] aic_btusb: RELEASE DATE: 2023_1211_1958
[    8.227746]
[    8.227753] aic_btusb: Register usb char device interface for BT driver
[    8.232363] usbcore: registered new interface driver aic_btusb

Welcome to LubanCat
lubancat login:
Welcome to LubanCat
lubancat login: root
Password:
#
#
#
```

第 7 章　网络连接

RV06 板卡有多种网络连接方式，包括有线网口连接、wifi 连接、USB RNDIS 连接。下面分别介绍这三种连接方式。

7.1 网口连接

- RV06 板卡板载 100M 有线网口，支持 10/100M 自适应，支持全双工/半双工自适应，
- 当网口未插入时, 网口的黄灯和绿灯都是熄灭的状态
- 当系统识别到网口插入时，黄灯会随网络状态闪烁，绿灯会常亮
- 插入网口后系统会自动获取 ip 地址，可以通过以下命令查看 ip 地址

```
ifconfig
```

```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 2A:62:33:CC:9A:A8
          inet addr:192.168.103.157  Bcast:192.168.103.255  Mask:255.255.255.0
          inet6 addr: fe80::b152:83fe:59ae:c697/64 Scope:Link
          inet6 addr: fd0f:2820:2b7b:0:1527:e688:1a45:1e91/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:562 errors:0 dropped:0 overruns:0 frame:0
          TX packets:120 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:51590 (50.3 KiB)  TX bytes:11306 (11.0 KiB)
          Interrupt:55

lo        Link encap:Local Loopback
```

(下页继续)

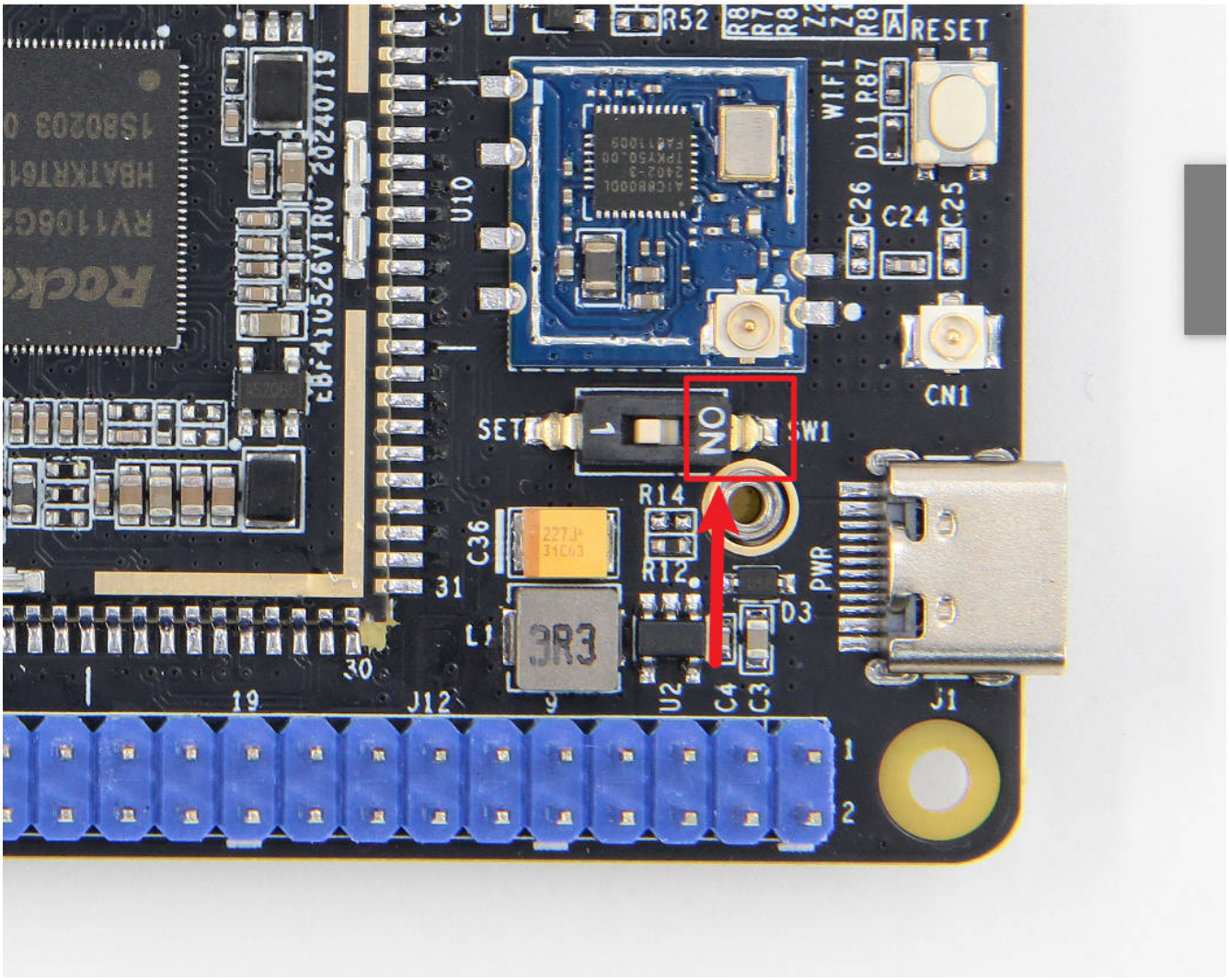
(续上页)

```
14          inet addr:127.0.0.1  Mask:255.0.0.0
15          inet6 addr: ::1/128 Scope:Host
16          UP LOOPBACK RUNNING  MTU:65536  Metric:1
17          RX packets:118 errors:0 dropped:0 overruns:0 frame:0
18          TX packets:118 errors:0 dropped:0 overruns:0 carrier:0
19          collisions:0 txqueuelen:1000
20          RX bytes:8910 (8.7 KiB)  TX bytes:8910 (8.7 KiB)
```

7.2 USB RNDIS 连接

RNDIS 使用的是固定 IP 地址：`192.168.137.100`

- RV06 板卡的 type-c 口是 otg 接口，可以将该接口设置成 `device` 模式
- 由于 RV1106 芯片上只有一个 USB2.0 的接口，切换使用 type-c 接口时，需要将 type-c 接口的 usb 开启，即需要将板载的拨码开关拨到 `ON`
- 运行的时候需要使用 usb 线连接电脑和板卡上。



- 配置文件 /etc/usb_config，将 OTG_MODE 设置为 peripheral

```
# cat /etc/usb_config
# OTG_MODE:host or peripheral
# OTG_MODE=peripheral
OTG_MODE=host

# GADGET_CONFIG:usb_mtp_en usb_ums_en usb_ntb_en
# usb_acm_en usb_uac1_en usb_uac2_en usb_uvc_en usb_rndis_en usb_hid_en
GADGET_CONFIG="usb_rndis_en"
```

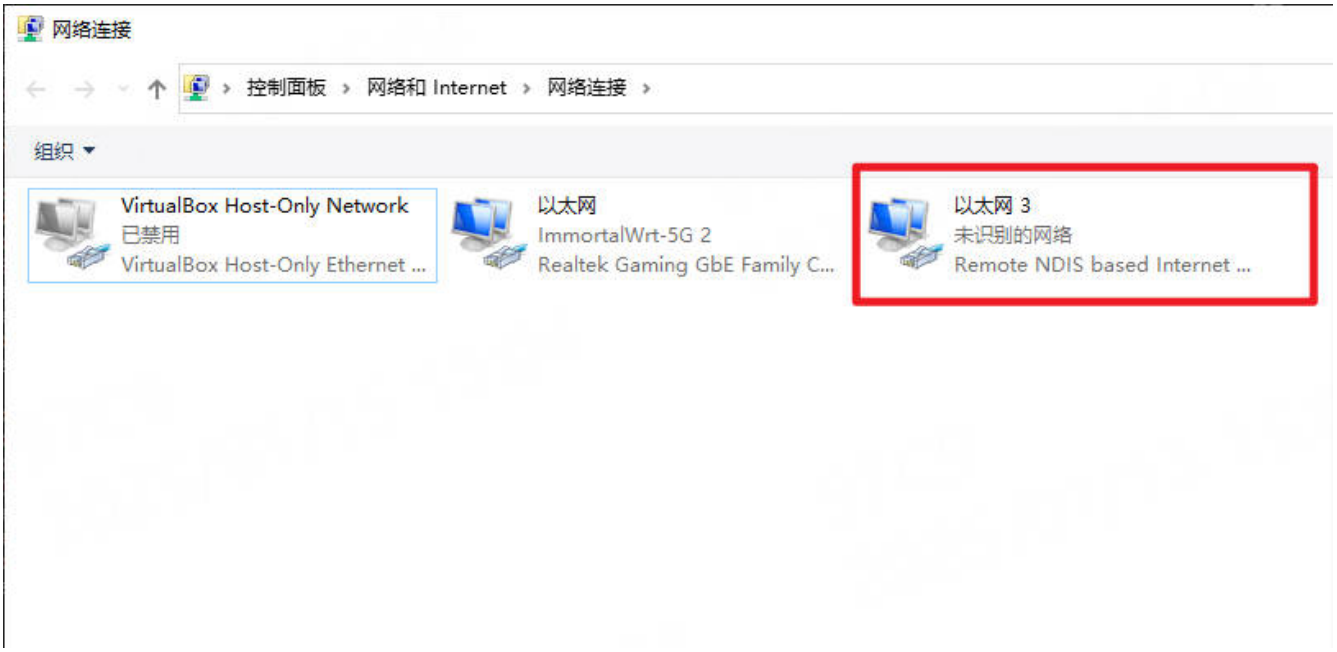
- 将上述文件配置改成下面配置

```
# cat /etc/usb_config
# OTG_MODE:host or peripheral
OTG_MODE=peripheral
# OTG_MODE=host

# GADGET_CONFIG:usb_mtp_en usb_ums_en usb_ntb_en
# usb_acm_en usb_uac1_en usb_uac2_en usb_uvc_en usb_rndis_en usb_hid_en
GADGET_CONFIG="usb_rndis_en"
```

- 重启或者执行下面命令生效

```
reboot
# 或
/etc/init.d/S50usbdevice start
```

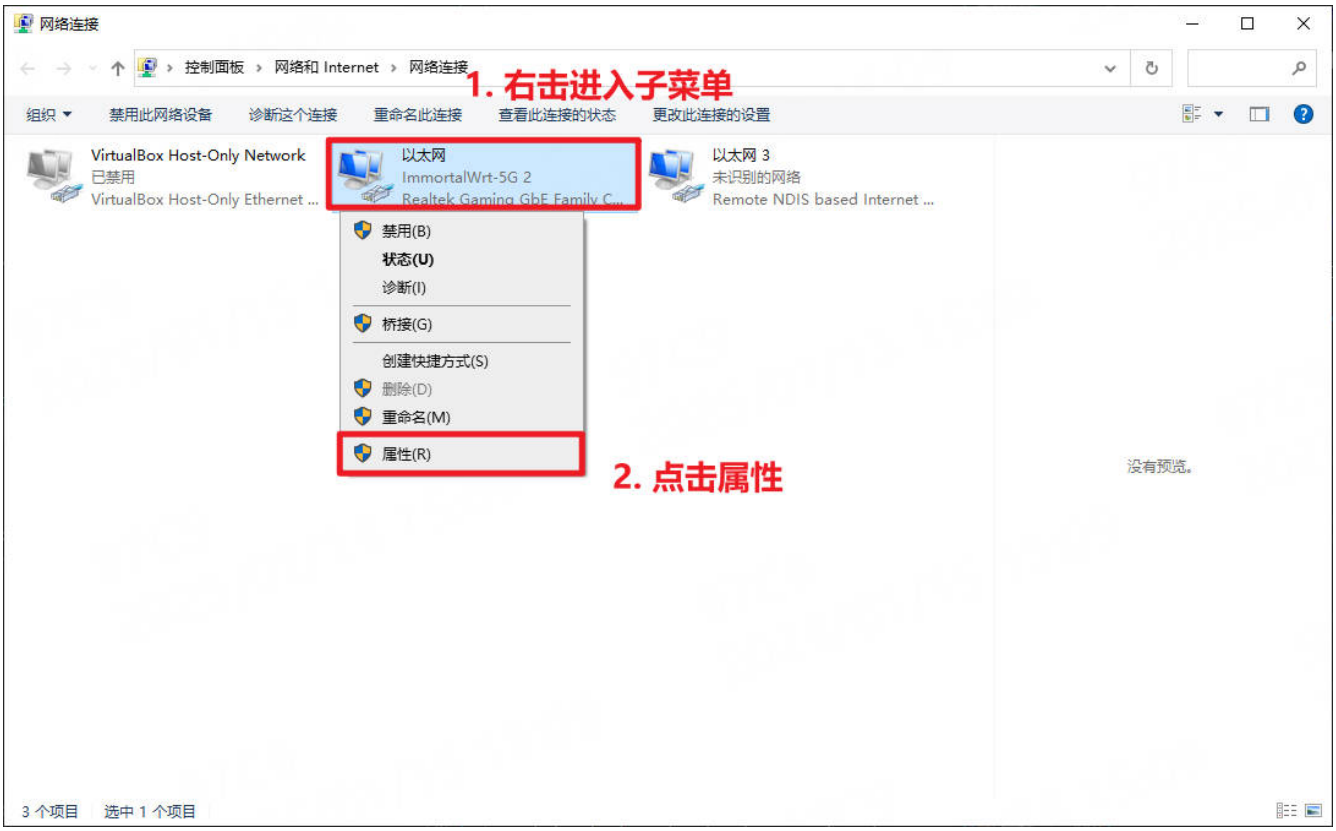
RNDIS 在 Windows 系统上是免驱的，所以可以看到 更改适配器选项中的 网络连接里会多了下面的设置



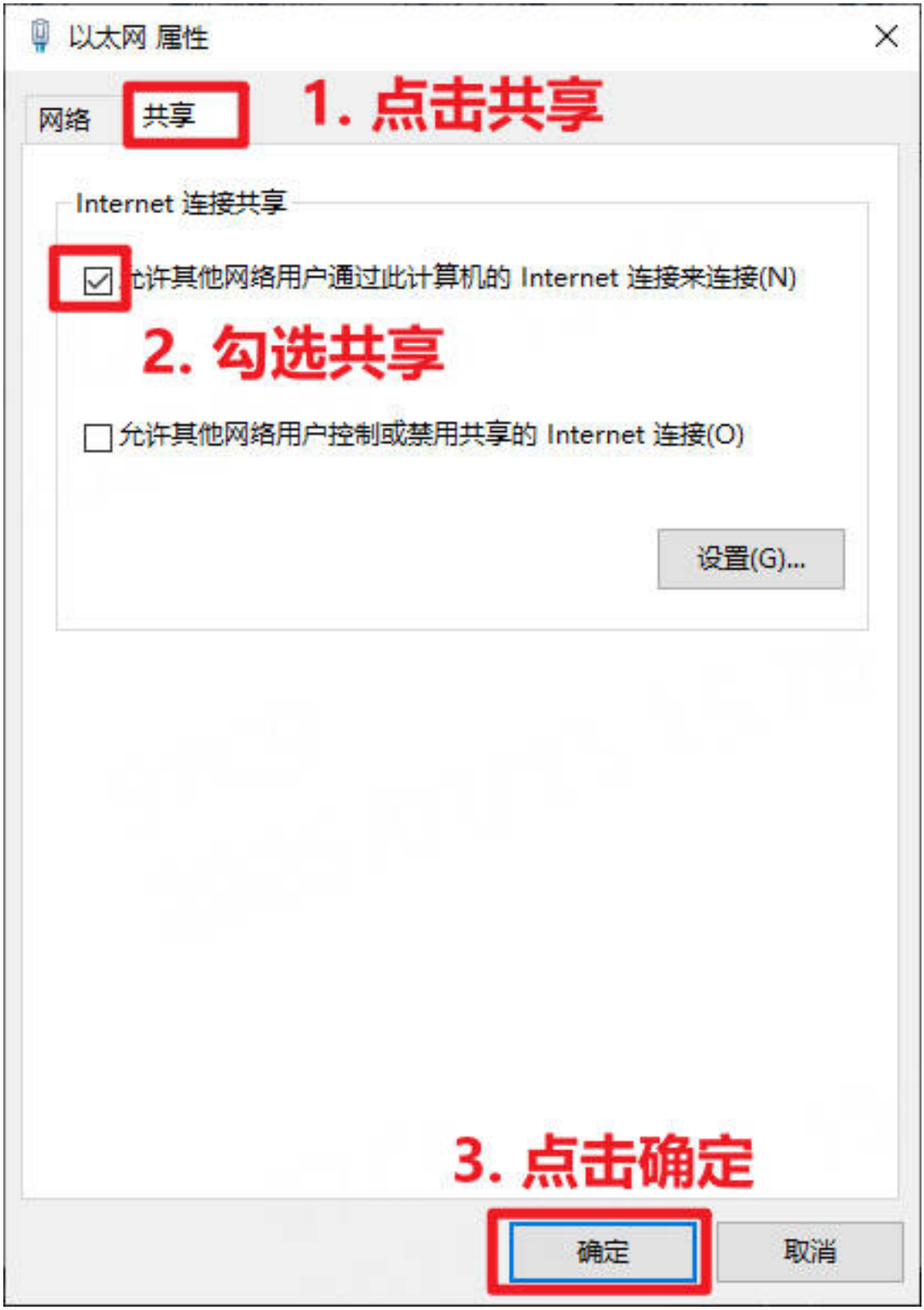
7.2.1 共享网络

如果想让板卡连接上网络，可以启动网络共享

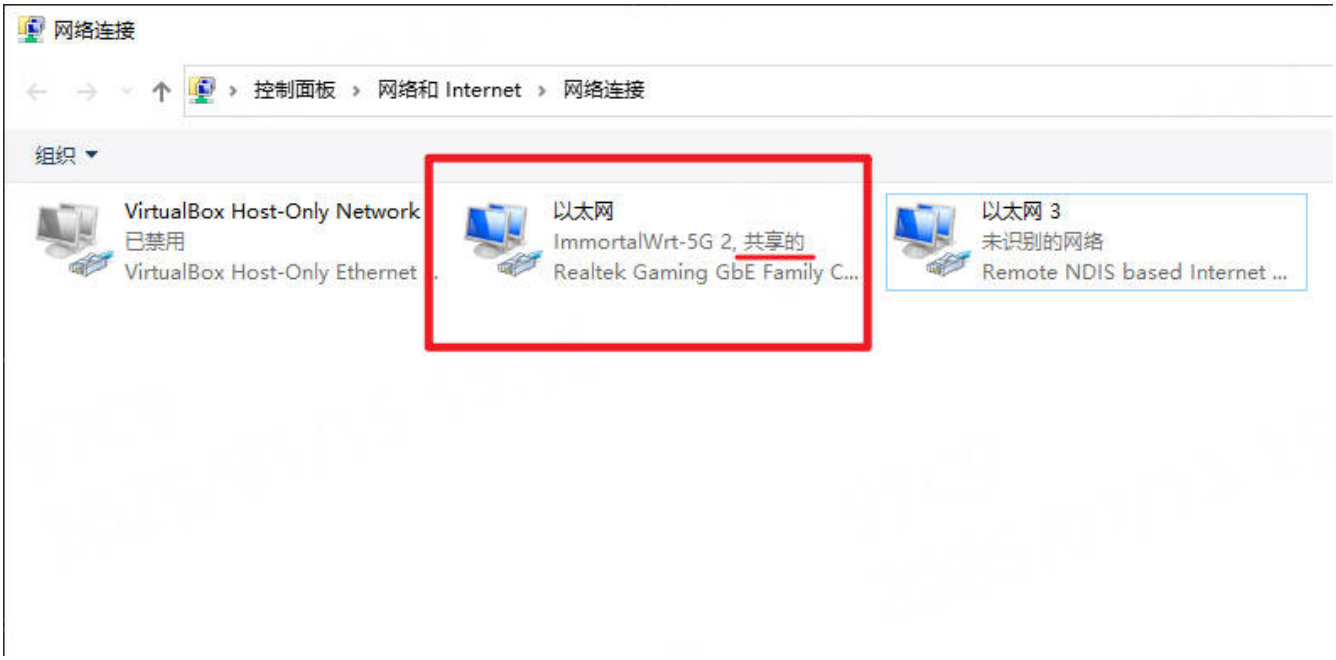
- 进入 更改适配器选项中的 网络连接
- 右击有网络的连接，点击 属性



- 然后点击 共享，勾选 允许其他网络用户通过此计算机的 Internet 连接来连接，点击确定

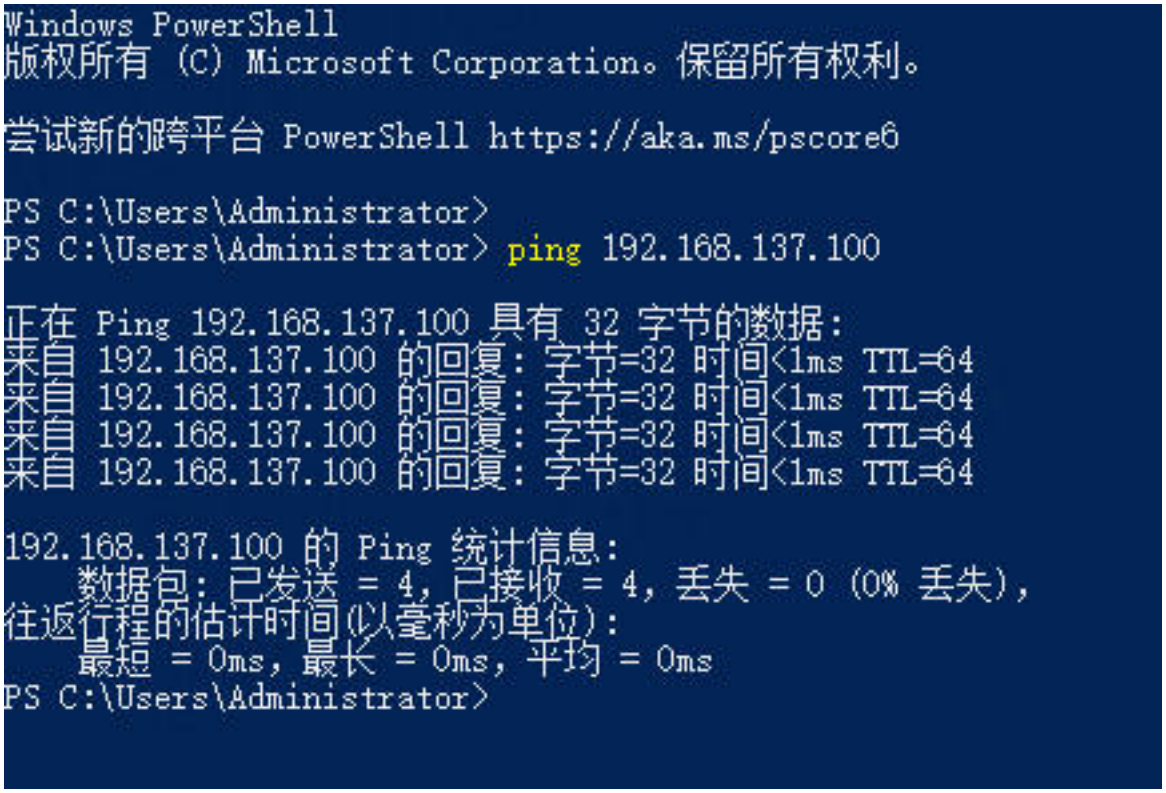

以太网 属性
网络
共享
1. 点击共享
Internet 连接共享
允许其他网络用户通过此计算机的 Internet 连接来连接(N)
2. 勾选共享
允许其他网络用户控制或禁用共享的 Internet 连接(O)
设置(G)...
3. 点击确定
确定
取消

- 然后可以看到网络上写着 共享的



这时候板卡就可以连接到电脑上，使用电脑的网络进行上网了。

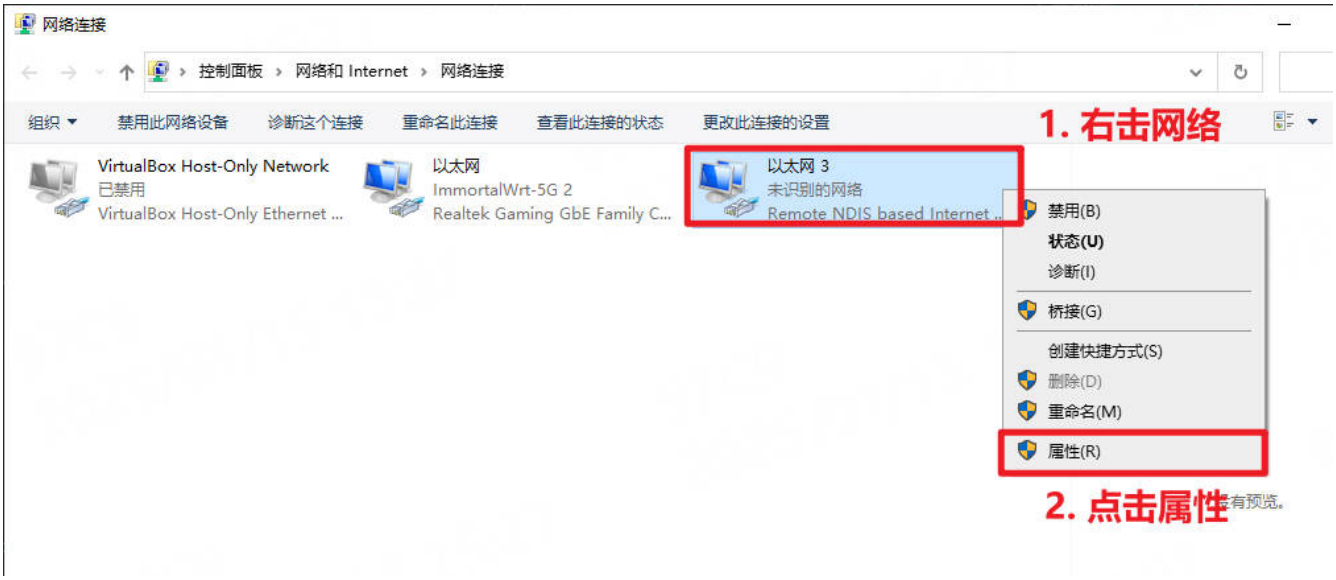
可以使用在 power shell 上使用 ping 查看网络连接情况



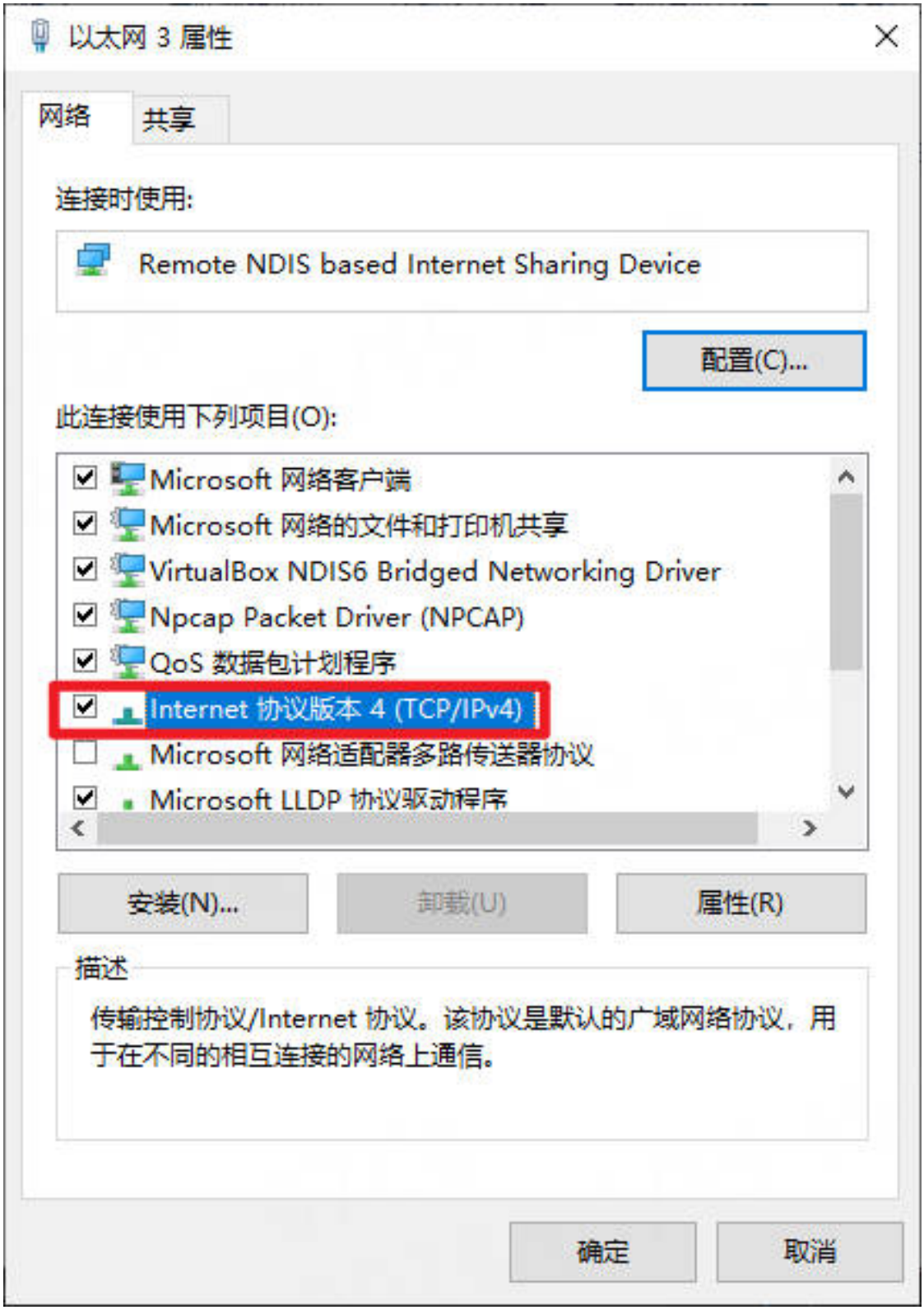
7.2.2 手动设置

如果不想让板子运行在有网络的环境

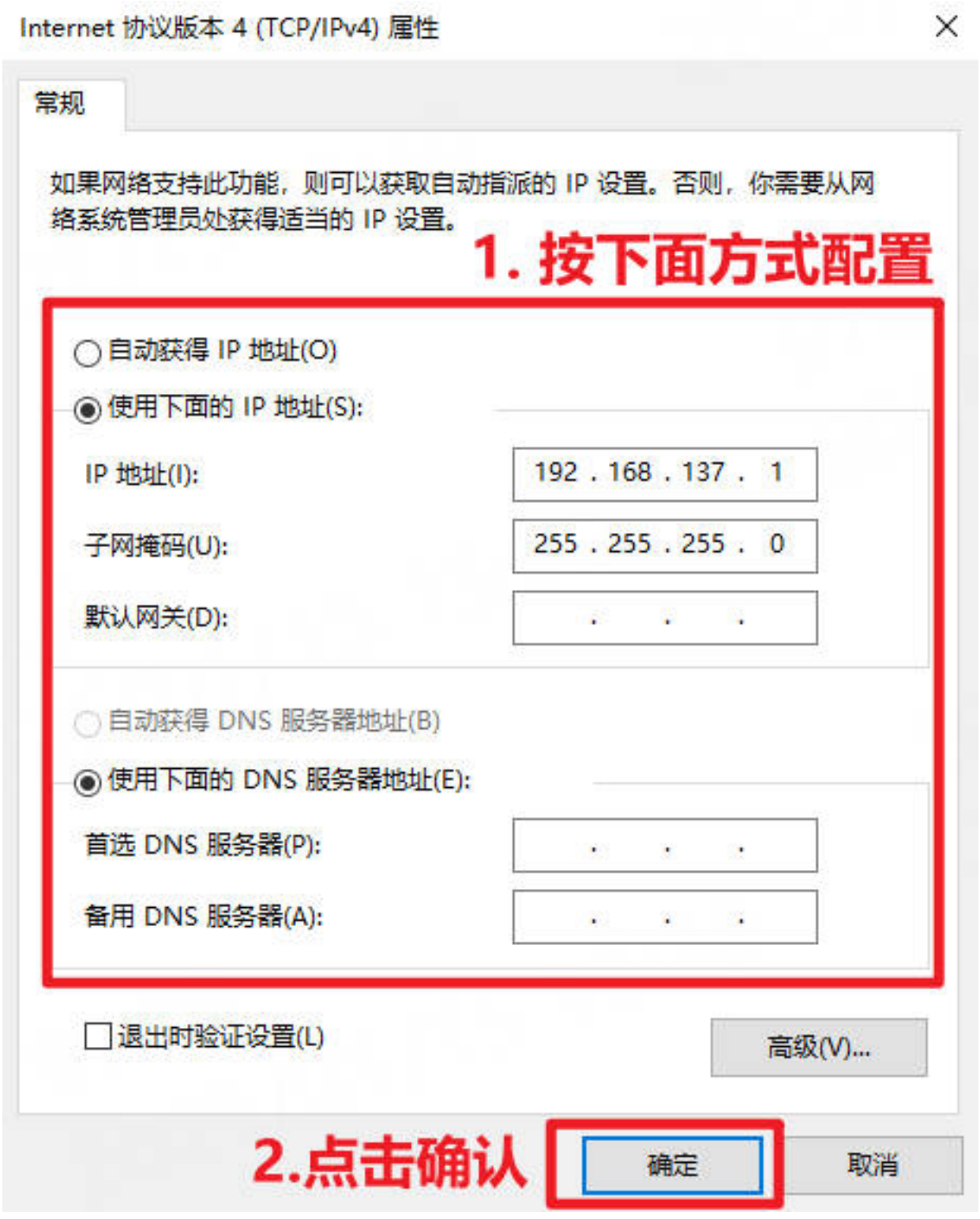
- 进入 更改适配器选项中的 网络连接
- 右击 rndis 的连接，点击 属性



- 双击 Internet 协议版本 4 (TCP/IPv4)


以太网 3 属性
网络
共享
连接时使用:
Remote NDIS based Internet Sharing Device
配置(C)...
此连接使用下列项目(O):
Microsoft 网络客户端
Microsoft 网络的文件和打印机共享
VirtualBox NDIS6 Bridged Networking Driver
Npcap Packet Driver (NPCAP)
QoS 数据包计划程序
Internet 协议版本 4 (TCP/IPv4)
Microsoft 网络适配器多路传送器协议
Microsoft LLDP 协议驱动程序
安装(N)...
卸载(U)
属性(R)
描述
传输控制协议/Internet 协议。该协议是默认的广域网络协议，用于在不同的相互连接的网络上通信。
确定
取消

- 进入配置，按下图配置


Internet 协议版本 4 (TCP/IPv4) 属性
常规
如果网络支持此功能，则可以获取自动指派的 IP 设置。否则，你需要从网络系统管理员处获得适当的 IP 设置。
1. 按下面方式配置
自动获得 IP 地址(O)
使用下面的 IP 地址(S):
IP 地址(I): 192 . 168 . 137 . 1
子网掩码(U): 255 . 255 . 255 . 0
默认网关(D):
自动获得 DNS 服务器地址(B)
使用下面的 DNS 服务器地址(E):
首选 DNS 服务器(P):
备用 DNS 服务器(A):
退出时验证设置(L)
高级(V)...
2.点击确认
确定
取消

- 最后点击确定，这时候板卡就可以连接到电脑上。

7.2.3 取消 rndis 连接

- 配置文件 /etc/usb_config ，将 OTG_MODE 设置为 host

```
# cat /etc/usb_config
# OTG_MODE:host or peripheral
OTG_MODE=peripheral
# OTG_MODE=host

# GADGET_CONFIG:usb_mtp_en usb_ums_en usb_ntb_en
# usb_acm_en usb_uac1_en usb_uac2_en usb_uvc_en usb_rndis_en usb_hid_en
GADGET_CONFIG="usb_rndis_en"
```

- 将上述文件配置改成下面配置

```
# cat /etc/usb_config
# OTG_MODE:host or peripheral
# OTG_MODE=peripheral
OTG_MODE=host
```

(下页继续)

(续上页)

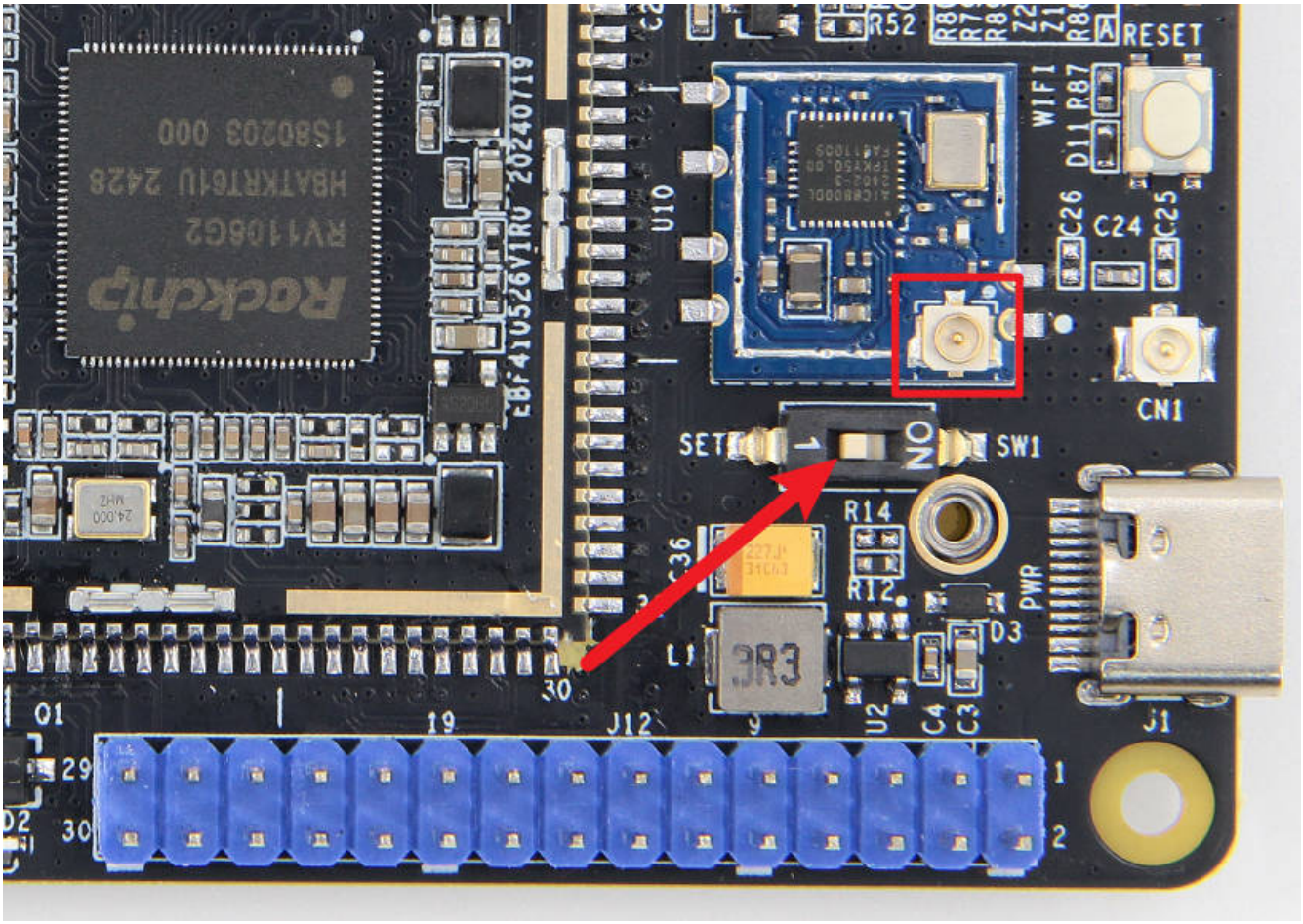
```

# GADGET_CONFIG:usb_mtp_en usb_ums_en usb_ntb_en
# usb_acm_en usb_uac1_en usb_uac2_en usb_uvc_en usb_rndis_en usb_hid_en
GADGET_CONFIG="usb_rndis_en"
```

- 重启或者执行下面命令生效

```
reboot
# 或
/etc/init.d/S50usbdevice stop
```

7.3 wifi 连接

- RV06 板卡板载 USB 接口的 2.4GHz 单频 WIFI6 模块 `AIC8800DL`
- 由于 RV1106 芯片上只有一个 USB2.0 的接口，切换使用 WIFI 时，需要将 type-c 接口的 usb 关闭切换到 wifi 上，需要将板载的拨码开关拨到 `1`
- RV06 板卡连接 WIFI 需要连接 WiFi 天线，天线接口是 `ipx-1` 类型，需要注意的是，天线需要接到板载模块上面，连接板上的天线接口不会有信号，如下图所示



7.3.1 配置文件配置 wifi

RV06 使用 wpa_supplicant 工具进行 wifi 连接，连接 wifi 前需要先配置 wifi 的 ssid 和密码，配置文件路径为 /etc/wpa_supplicant.conf

```
# cat /etc/wpa_supplicant.conf
ctrl_interface=/var/run/wpa_supplicant
update_config=1

network={
        ssid="aEBF_Guest"
```

(下页继续)

(续上页)

```
        psk="BQTH4kH9"
        key_mgmt=WPA-PSK
}
```

- 配置文件中 ssid 和 psk 为需要连接的 wifi 的名称和密码，key_mgmt 为加密方式，WPA-PSK 为 WIFI 密码加密方式
- 配置文件中可以配置多个 wifi，系统会自动选择信号最好的 wifi 进行连接

7.3.2 wpa_cli 连接 wifi

- 使用 `wpa_cli` 工具连接 wifi，命令行输入下面命令，进入配置

```
wpa_cli
```

```
# wpa_cli
wpa_cli v2.10
Copyright (c) 2004-2022, Jouni Malinen <j@w1.fi> and contributors

This software may be distributed under the terms of the BSD license.
See README for more details.


Selected interface 'wlan0'

Interactive mode

>
```

- 输入命令后可以看到进入了新的命令行，输入 `help` 可以查看 wpa_cli 的命令帮助

```
> help
commands:
  status [verbose] = get current WPA/EAPOL/EAP status
  ifname = get current interface name
  ping = pings wpa_supplicant
  relog = re-open log-file (allow rolling logs)
  note <text> = add a note to wpa_supplicant debug log
  mib = get MIB variables (dot1x, dot11)
  help [command] = show usage help
  interface [ifname] = show interfaces/select interface
  level <debug level> = change debug level
  license = show full wpa_cli license
  quit = exit wpa_cli
  set = set variables (shows list of variables when run without arguments)
  dump = dump config variables
  get <name> = get information
  driver_flags = list driver flags
  logon = IEEE 802.1X EAPOL state machine logon
  logoff = IEEE 802.1X EAPOL state machine logoff
  pmksa = show PMKSA cache
  pmksa_flush = flush PMKSA cache entries
  reassociate = force reassociation
  reattach = force reassociation back to the same BSS
  preauthenticate <BSSID> = force preauthentication
  identity <network id> <identity> = configure identity for an SSID
  password <network id> <password> = configure password for an SSID
  new_password <network id> <password> = change password for an SSID
  pin <network id> <pin> = configure pin for an SSID
  otp <network id> <password> = configure one-time-password for an SSID
  psk_passphrase <network id> <PSK/passphrase> = configure PSK/passphrase␣
↪for an SSID
  passphrase <network id> <passphrase> = configure private key passphrase
```

(下页继续)

(续上页)

```
    for an SSID
  sim <network id> <pin> = report SIM operation result
  bssid <network id> <BSSID> = set preferred BSSID for an SSID
  bssid_ignore <BSSID> = add a BSSID to the list of temporarily ignored BSSs
  bssid_ignore clear = clear the list of temporarily ignored BSSIDs
  bssid_ignore = display the list of temporarily ignored BSSIDs
  blacklist = deprecated alias for bssid_ignore
  log_level <level> [<timestamp>] = update the log level/timestamp
  log_level = display the current log level and log options
  list_networks = list configured networks
  select_network <network id> = select a network (disable others)
  enable_network <network id> = enable a network
  disable_network <network id> = disable a network
  add_network = add a network
  remove_network <network id> = remove a network
  set_network <network id> <variable> <value> = set network variables (shows
    list of variables when run without arguments)
  get_network <network id> <variable> = get network variables
  dup_network <src network id> <dst network id> <variable> = duplicate␣
↪network variables
  list_creds = list configured credentials
  add_cred = add a credential
  remove_cred <cred id> = remove a credential
  set_cred <cred id> <variable> <value> = set credential variables
  get_cred <cred id> <variable> = get credential variables
  save_config = save the current configuration
  disconnect = disconnect and wait for reassociate/reconnect command before
    connecting
  reconnect = like reassociate, but only takes effect if already␣
↪disconnected
  scan = request new BSS scan
  scan_results = get latest scan results
```

(下页继续)

(续上页)

```
  abort_scan = request ongoing scan to be aborted
  bss <<idx> | <bssid>> = get detailed scan result info
  get_capability <eap/pairwise/group/key_mgmt/proto/auth_alg/channels/freq/
↪modes> = get capabilities
  reconfigure = force wpa_supplicant to re-read its configuration file
  terminate = terminate wpa_supplicant
  interface_add <ifname> <confname> <driver> <ctrl_interface> <driver_param>
    <bridge_name> <create> <type> = adds new interface, all parameters but
    <ifname> are optional. Supported types are station ('sta') and AP ('ap')
  interface_remove <ifname> = removes the interface
  interface_list = list available interfaces
  ap_scan <value> = set ap_scan parameter
  scan_interval <value> = set scan_interval parameter (in seconds)
  bss_expire_age <value> = set BSS expiration age parameter
  bss_expire_count <value> = set BSS expiration scan count parameter
  bss_flush <value> = set BSS flush age (0 by default)
  ft_ds <addr> = request over-the-DS FT with <addr>
  wps_pbc [BSSID] = start Wi-Fi Protected Setup: Push Button Configuration
  wps_pin <BSSID> [PIN] = start WPS PIN method (returns PIN, if not␣
↪hardcoded)
  wps_check_pin <PIN> = verify PIN checksum
  wps_cancel Cancels the pending WPS operation
  wps_reg <BSSID> <AP PIN> = start WPS Registrar to configure an AP
  wps_ap_pin [params..] = enable/disable AP PIN
  wps_er_start [IP address] = start Wi-Fi Protected Setup External Registrar
  wps_er_stop = stop Wi-Fi Protected Setup External Registrar
  wps_er_pin <UUID> <PIN> = add an Enrollee PIN to External Registrar
  wps_er_pbc <UUID> = accept an Enrollee PBC using External Registrar
  wps_er_learn <UUID> <PIN> = learn AP configuration
  wps_er_set_config <UUID> <network id> = set AP configuration for enrolling
  wps_er_config <UUID> <PIN> <SSID> <auth> <encr> <key> = configure AP
  ibss_rsn <addr> = request RSN authentication with <addr> in IBSS
```

(下页继续)

(续上页)

```
  suspend = notification of suspend/hibernate
  resume = notification of resume/thaw
  roam <addr> = roam to the specified BSS
  vendor_elem_add <frame id> <hexdump of elem(s)> = add vendor specific IEs␣
↪to frame(s)
    0: Probe Req (P2P), 1: Probe Resp (P2P) , 2: Probe Resp (GO), 3: Beacon␣
↪(GO), 4: PD Req, 5: PD Resp, 6: GO Neg Req, 7: GO Neg Resp, 8: GO Neg␣
↪Conf, 9: Inv Req, 10: Inv Resp, 11: Assoc Req (P2P), 12: Assoc Resp (P2P)
  vendor_elem_get <frame id> = get vendor specific IE(s) to frame(s)
    0: Probe Req (P2P), 1: Probe Resp (P2P) , 2: Probe Resp (GO), 3: Beacon␣
↪(GO), 4: PD Req, 5: PD Resp, 6: GO Neg Req, 7: GO Neg Resp, 8: GO Neg␣
↪Conf, 9: Inv Req, 10: Inv Resp, 11: Assoc Req (P2P), 12: Assoc Resp (P2P)
  vendor_elem_remove <frame id> <hexdump of elem(s)> = remove vendor␣
↪specific IE(s) in frame(s)
    0: Probe Req (P2P), 1: Probe Resp (P2P) , 2: Probe Resp (GO), 3: Beacon␣
↪(GO), 4: PD Req, 5: PD Resp, 6: GO Neg Req, 7: GO Neg Resp, 8: GO Neg␣
↪Conf, 9: Inv Req, 10: Inv Resp, 11: Assoc Req (P2P), 12: Assoc Resp (P2P)
  sta_autoconnect <0/1> = disable/enable automatic reconnection
  tdls_discover <addr> = request TDLS discovery with <addr>
  tdls_setup <addr> = request TDLS setup with <addr>
  tdls_teardown <addr> = tear down TDLS with <addr>
  tdls_link_status <addr> = TDLS link status with <addr>
  wmm_ac_addts <uplink/downlink/bidi> <tsid=0..7> <up=0..7> [nominal_msdu_
↪size=#] [mean_data_rate=#] [min_phy_rate=#] [sba=#] [fixed_nominal_msdu]␣
↪= add WMM-AC traffic stream
  wmm_ac_delts <tsid> = delete WMM-AC traffic stream
  wmm_ac_status = show status for Wireless Multi-Media Admission-Control
  tdls_chan_switch <addr> <oper class> <freq> [sec_channel_offset=] [center_
↪freq1=] [center_freq2=] [bandwidth=] [ht|vht] = enable channel switching␣
↪with TDLS peer
  tdls_cancel_chan_switch <addr> = disable channel switching with TDLS peer
↪<addr>
```

(下页继续)

(续上页)

```
  signal_poll = get signal parameters
  signal_monitor = set signal monitor parameters
  pktcnt_poll = get TX/RX packet counters
  reauthenticate = trigger IEEE 802.1X/EAPOL reauthentication
  raw <params..> = Sent unprocessed command
  flush = flush wpa_supplicant state
  radio_work = radio_work <show/add/done>
  vendor <vendor id> <command id> [<hex formatted command argument>] = Send␣
↪vendor command
  neighbor_rep_request [ssid=<SSID>] [lci] [civic] = Trigger request to AP␣
↪for neighboring AP report (with optional given SSID in hex or enclosed in␣
↪double quotes, default: current SSID; with optional LCI and location␣
↪civic request)
  twt_setup [dialog=<token>] [exponent=<exponent>] [mantissa=<mantissa>]␣
↪[min_twt=<Min TWT>] [setup_cmd=<setup-cmd>] [twt=<u64>] [requestor=0|1]␣
↪[trigger=0|1] [implicit=0|1] [flow_type=0|1] [flow_id=<3-bit-id>]␣
↪[protection=0|1] [twt_channel=<twt chanel id>] [control=<control-u8>] =␣
↪Send TWT Setup frame
  twt_teardown [flags=<value>] = Send TWT Teardown frame
  erp_flush = flush ERP keys
  mac_rand_scan <scan|sched|pno|all> enable=<0/1> [addr=mac-address␣
↪mask=mac-address-mask] = scan MAC randomization
  get_pref_freq_list <interface type> = retrieve preferred freq list for␣
↪the specified interface type
  p2p_lo_start <freq> <period> <interval> <count> = start P2P listen offload
  p2p_lo_stop = stop P2P listen offload
  dpp_qr_code report a scanned DPP URI from a QR Code
  dpp_bootstrap_gen type=<qrcode> [chan=..] [mac=..] [info=..] [curve=..]␣
↪[key=..] = generate DPP bootstrap information
  dpp_bootstrap_remove *|<id> = remove DPP bootstrap information
  dpp_bootstrap_get_uri <id> = get DPP bootstrap URI
  dpp_bootstrap_info <id> = show DPP bootstrap information
```

(下页继续)

(续上页)

```
132   dpp_bootstrap_set <id> [conf=..] [ssid=<SSID>] [ssid_charset=#] [psk=<PSK>
      ↪] [pass=<passphrase>] [configurator=<id>] [conn_status=#] [akm_use_
      ↪selector=<0|1>] [group_id=..] [expiry=#] [csrattrs=..] = set DPP␣
      ↪configurator parameters
133   dpp_auth_init peer=<id> [own=<id>] = initiate DPP bootstrapping
134   dpp_listen <freq in MHz> = start DPP listen
135   dpp_stop_listen = stop DPP listen
136   dpp_configurator_add [curve=..] [key=..] = add DPP configurator
137   dpp_configurator_remove *|<id> = remove DPP configurator
138   dpp_configurator_get_key <id> = Get DPP configurator's private key
139   dpp_configurator_sign conf=<role> configurator=<id> = generate self DPP␣
      ↪configuration
140   dpp_pkex_add add PKEX code
141   dpp_pkex_remove *|<id> = remove DPP pkex information
142   dpp_controller_start [tcp_port=<port>] [role=..] = start DPP controller
143   dpp_controller_stop = stop DPP controller
144   dpp_chirp own=<BI ID> iter=<count> = start DPP chirp
145   dpp_stop_chirp = stop DPP chirp
146   all_bss = list all BSS entries (scan results)
147   mscs <add|remove|change> [up_bitmap=<hex byte>] [up_limit=<integer>]␣
      ↪[stream_timeout=<in TUs>] [frame_classifier=<hex bytes>] = Configure MSCS␣
      ↪request
148   scs [scs_id=<decimal number>] <add|remove|change> [scs_up=<0-7>]␣
      ↪[classifier_type=<4|10>] [classifier params based on classifier type]␣
      ↪[tclas_processing=<0|1>] [scs_id=<decimal number>] ... = Send SCS request
149   dscp_resp <[reset]>/<[solicited] [policy_id=1 status=0...]> [more] = Send␣
      ↪DSCP response
150   dscp_query wildcard/domain_name=<string> = Send DSCP Query
151 >
```

- 可以看到命令会比较多，这里我们只介绍一些常用的命令

表 1: wpa_cli 常用命令

命令	命令说明	举例
status	获取当前 wifi 连接状态	status
scan	搜索 wifi	scan
scan_results	搜到到的全部 wifi 名称等信息	scan_results
set_network	设置 wifi 的 SSID 和 psk	set_network 1 ssid “PPP”
list_network	列出所有的配置文件中的信息	list_network
add_network	添加一个网络	add_network
disale_network	禁止 WiFi	disale_network 1
select_network	选择一个网络	select_network 1
remove_network	根据网络 ID 删除删除一个网络 ID，根据网络 ID 删除	remove_network 1
quit	退出 wpa_cli	quit
disconnect	断开连接	disconnect
reconnect	重新连接	reconnect

下面将通过一个例子来演示如何使用 wpa_cli 命令来连接 wifi

```
# wpa_cli
wpa_cli v2.10
Copyright (c) 2004-2022, Jouni Malinen <j@w1.fi> and contributors

This software may be distributed under the terms of the BSD license.
See README for more details.

```

(下页继续)

(续上页)

```
8
9  Selected interface 'wlan0'
10
11 Interactive mode
12
13 > status
14 bssid=34:f7:16:dc:b7:5c
15 freq=2462
16 ssid=aEBF_Guest
17 id=1
18 mode=station
19 wifi_generation=4
20 pairwise_cipher=CCMP
21 group_cipher=CCMP
22 key_mgmt=WPA2-PSK
23 wpa_state=COMPLETED
24 ip_address=10.169.169.7
25 address=38:7a:cc:65:78:20
26 uuid=45fde3ec-a3a3-53d8-9046-2f6af7bd32e0
27 ieee80211ac=1
28 >
29 > list_network
30 network id / ssid / bssid / flags
31 0       aEBF_Guest      any     [CURRENT]
32 >
33 > disconnect
34 OK
35 <3>CTRL-EVENT-DISCONNECTED bssid=34:f7:16:dc:b7:5c reason=3 locally_
    ↪generated=1
36 <3>CTRL-EVENT-DSCP-POLICY clear_all
37 <3>CTRL-EVENT-REGDOM-CHANGE init=CORE type=WORLD
38 >
```

(下页继续)

(续上页)

```
> remove_network 0
OK
<3>CTRL-EVENT-NETWORK-REMOVED 0
>
> list_networks
network id / ssid / bssid / flags
>
> scan
OK
<3>CTRL-EVENT-SCAN-STARTED
<3>CTRL-EVENT-SCAN-RESULTS
>
> scan_results
bssid / frequency / signal level / flags / ssid
fc:a0:5a:06:74:64       2447    -29     [WPA-PSK-CCMP+TKIP][WPA2-PSK-
↪CCMP+TKIP][ESS]    oraybox-2.4G
48:88:99:c9:16:82       2432    -33     [WPA2-PSK+SAE-CCMP-256+GCMP-
↪256+CCMP+GCMP][ESS] TOTO_M0
34:fc:a1:73:4c:2b       2422    -62     [WPA-PSK-CCMP+TKIP][WPA2-PSK-
↪CCMP+TKIP][WPS][ESS]       228
82:ea:07:e3:f3:81       2462    -61     [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]
00:4b:f3:97:0c:6f       2472    -64     [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS] ␣
↪    MERCURY_0C6F
34:f7:16:dc:b7:5c       2462    -20     [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS] ␣
↪    aEBF_Guest
34:f7:16:dc:bd:a7       2452    -49     [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS] ␣
↪    aEBF_Guest
42:c8:09:5d:2e:79       2412    -58     [WPA-PSK-CCMP+TKIP][WPA2-PSK-
↪CCMP+TKIP][WPS][ESS]       228-5G
36:f7:16:1c:b7:5c       2462    -21     [ESS]   aEBF_Office
36:f7:16:1c:bd:a7       2452    -50     [ESS]   aEBF_Office
36:f7:16:1c:bd:83       2462    -59     [ESS]   aEBF_Office
```

(下页继续)

(续上页)

```
>
> add_network
0
<3>CTRL-EVENT-NETWORK-ADDED 0
>
> set_network 0 ssid "TOTO_M0"
OK
>
> set_network 0 key_mgmt WPA-PSK
OK
>
> set_network 0 psk "pass123456789"
OK
>
> enable_network 0
OK
>
> save_config
OK
>
> list_networks
network id / ssid / bssid / flags
0       TOTO_M0 any
>
> select_network 0
OK
<3>CTRL-EVENT-SCAN-STARTED
<3>CTRL-EVENT-SCAN-RESULTS
<3>WPS-AP-AVAILABLE
<3>Trying to associate with 48:88:99:c9:16:82 (SSID='TOTO_M0' freq=2432 MHz)
<3>Associated with 48:88:99:c9:16:82
<3>CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
```

(下页继续)

(续上页)

```
<3>WPA: Key negotiation completed with 48:88:99:c9:16:82 [PTK=CCMP GTK=CCMP]
<3>CTRL-EVENT-CONNECTED - Connection to 48:88:99:c9:16:82 completed [id=1 id_str=]
>
> status
bssid=48:88:99:c9:16:82
freq=2432
ssid=TOTO_M0
id=0
mode=station
wifi_generation=6
pairwise_cipher=CCMP
group_cipher=CCMP
key_mgmt=WPA2-PSK
wpa_state=COMPLETED
ip_address=192.168.103.139
address=38:7a:cc:65:78:20
uuid=45fde3ec-a3a3-53d8-9046-2f6af7bd32e0
ieee80211ac=1
>
> quit
# ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 38:7A:CC:65:78:20
          inet addr:192.168.103.139  Bcast:192.168.103.255  Mask:255.255.255.0
          inet6 addr: fdf5:49a3:57bb::ff5/128 Scope:Global
          inet6 addr: fdf5:49a3:57bb:0:94c0:a9ae:3183:6cd9/64 Scope:Global
          inet6 addr: fe80::3a7a:ccff:fe65:7820/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:39595 errors:0 dropped:5 overruns:0 frame:0
          TX packets:192662 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2619579 (2.4 MiB)  TX bytes:288643449 (275.2 MiB)
```

(下页继续)

(续上页)

```
127
128 # ping baidu.com
129 PING baidu.com (39.156.66.10): 56 data bytes
130 64 bytes from 39.156.66.10: seq=0 ttl=47 time=43.113 ms
131 64 bytes from 39.156.66.10: seq=1 ttl=47 time=42.159 ms
132 64 bytes from 39.156.66.10: seq=2 ttl=47 time=57.048 ms
133 64 bytes from 39.156.66.10: seq=3 ttl=47 time=42.163 ms
134
135 # cat /etc/wpa_supplicant.conf
136 ctrl_interface=/var/run/wpa_supplicant
137 update_config=1
138
139 network={
140         ssid="TOTO_M0"
141         psk="pass123456789"
142         key_mgmt=WPA-PSK
143 }
144
145
146 cat
```

1. `status` ： 可以看到刚开始的时候板卡连接了 `aEBF_Guest` 这个 WIFI，IP 地址是 10.169.169.7
2. `list_network`：可以看到当前连接的 WiFi 是 `aEBF_Guest` ，编号为 0
3. `disconnect`：断开 `aEBF_Guest` WIFI 的连接
4. `remove_network 0`：删除编号为 0 的 WIFI，也就是 `aEBF_Guest` 这个 WIFI
5. `list_networks`：可以看到 WIFI 列表为空
6. `scan`：扫描附近的 WIFI
7. `scan_results`：可以看到扫描到了很多的 WIFI，后面以 `TOTO_M0` 这个 WIFI 连接为例

8. `add_network`：添加一个 WIFI 网络，获得编号为 `0`
9. `set_network 0 ssid "TOTO_M0"`：设置编号为 `0` 的 WIFI 的 ssid 为 `TOTO_M0`
10. `set_network 0 key_mgmt WPA-PSK`：设置编号为 `0` 的 WIFI 的加密方式为 `WPA-PSK`
11. `set_network 0 psk "pass123456789"`：设置编号为 `0` 的 WIFI 的密码为 `pass123456789`
12. `enable_network 0`：启用编号为 `0` 的 WIFI
13. `save_config`：保存配置
14. `list_networks`：可以看到 WIFI 列表中多了一个编号为 `0` 的 WIFI，SSID 为 `TOTO_M0`
15. `select_network 0`：连接编号为 `0` 的 WIFI, 可以看到连接成功了
16. `status`：可以看到当前连接的 WiFi 是 `TOTO_M0`，并且 IP 地址是 192.168.103.139
17. `quit`：退出程序
18. `ifconfig wlan0`：可以看到 wlan0 网卡的信息，包括 IP 地址，MAC 地址等
19. `ping baidu.com`：可以看到 ping 百度服务器的结果
20. `cat /etc/wpa_supplicant.conf`：可以看到 wpa_supplicant 的配置文件，包括连接的 WiFi 的 SSID 和密码等

整个过程还是比较复杂的，需要耐心操作。

第 8 章　远程调试

RV06 板卡支持多种方式的远程调试，分别是：

- SSH
- telnet

8.1 SSH

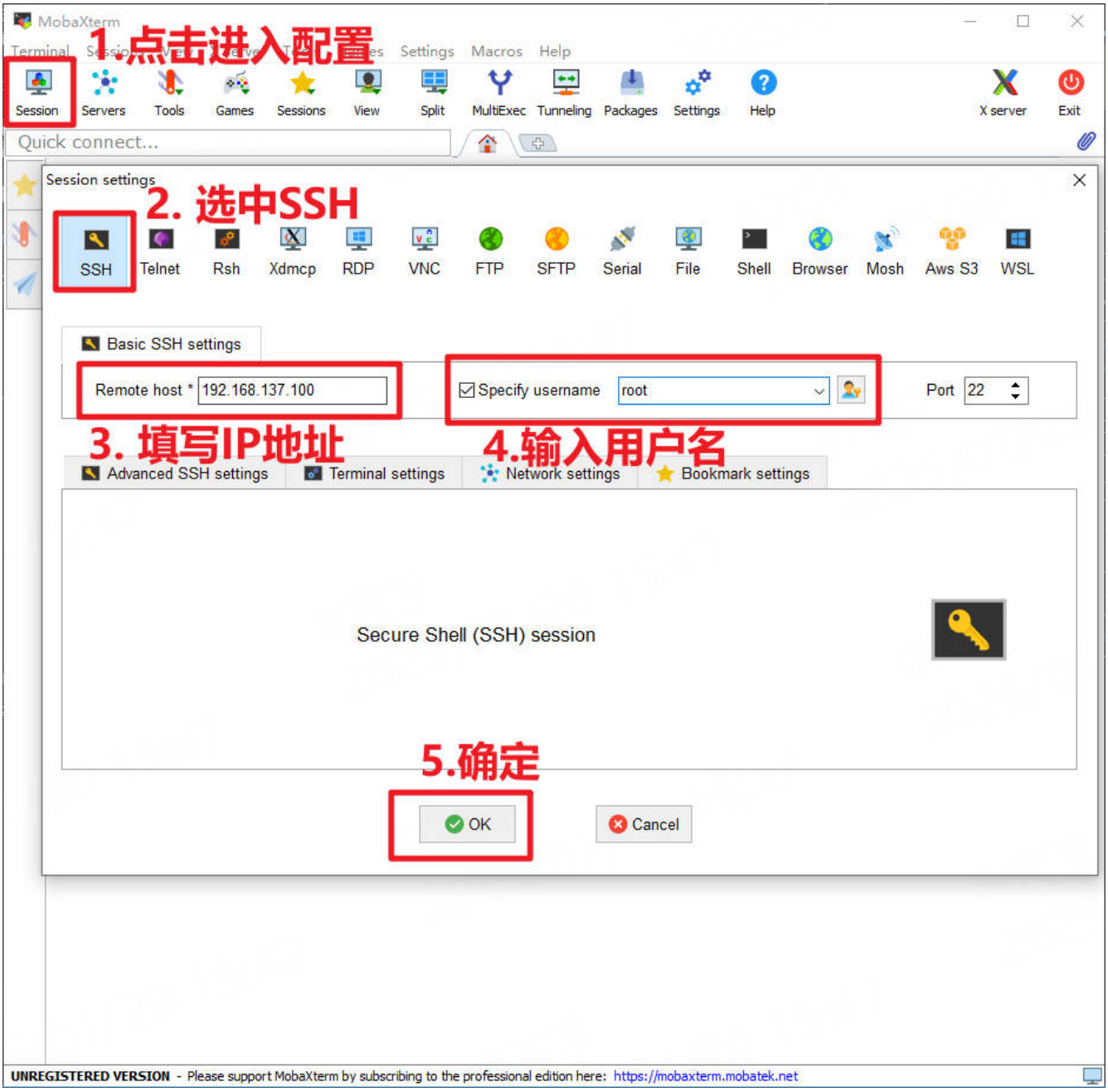
SSH 叫安全外壳协议（Secure Shell），是一种加密的网络传输协议，可在不安全的网络中网络服务提供安全的传输环境。它通过在网络中创建安全隧道来实现 SSH 客户端和服务器之间的连接。

- SSH 是通过网络作为载体的，因此在想要使用 SSH 连接板卡，首先需要确保板卡和电脑之间能够正常通信。
- 如果是使用网线或者 wifi 连接的话，需要将电脑和板卡连接在同一个网段，可以使用串口连接，然后使用 `ifconfig` 命令查看板卡的 IP 地址。
- 如果是使用 rndis, 则使用的 IP 为：`192.168.137.100`

RV06 板卡默认开启 SSH 服务，默认用户名和密码为：root/root

这里以 RNDIS 为例，使用 SSH 连接板卡：

首先打开 `SSH` 工具，这里以前面的使用的 `MobaXterm` 为例.



- 然后会让你输入密码，输入密码即可

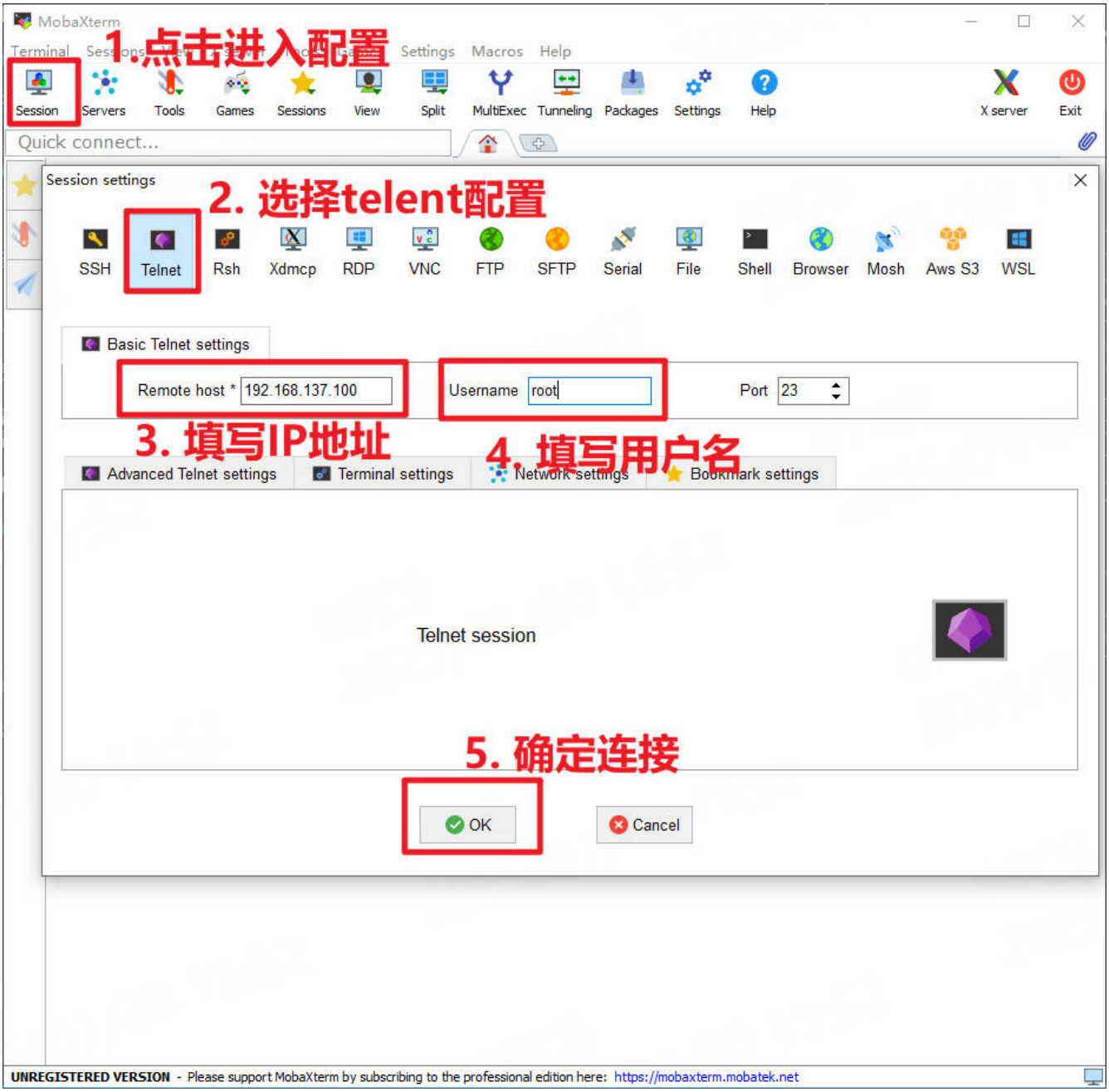
8.2 telnet

Telnet 是一种应用层协议，使用于互联网及局域网中，使用虚拟终端的形式，提供双向、以文字字符串为主的命令行接口交互功能。属于 TCP/IP 协议族的其中之一，是互联网远程登录服务的标准协议和主要方式，常用于服务器的远程控制，可供用户在本地主机执行远程主机上的工作。

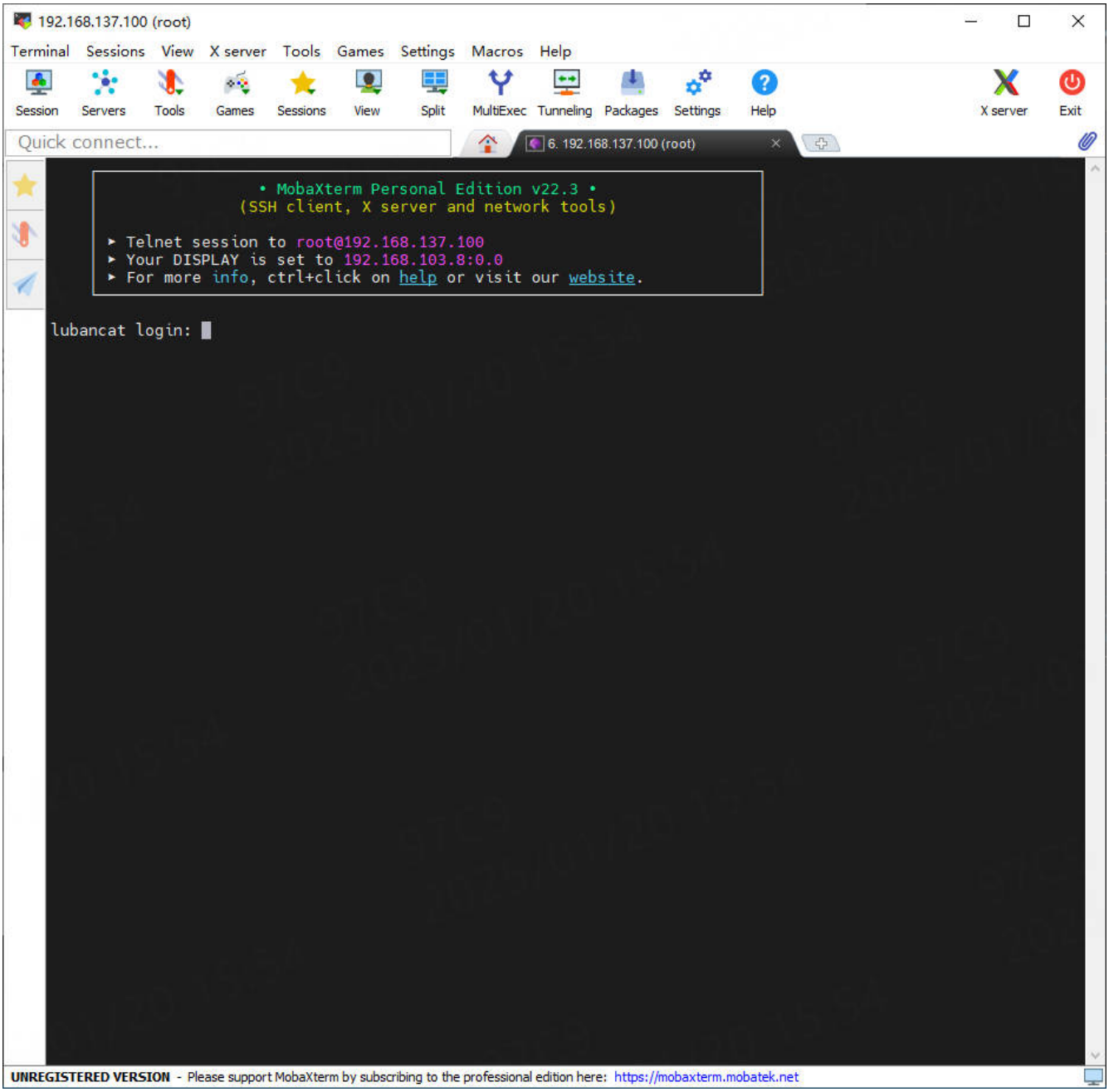
- 如果是使用网线或者 wifi 连接的话，需要将电脑和板卡连接在同一个网段，可以使用串口连接，然后使用 `ifconfig` 命令查看板卡的 IP 地址。
- 如果是使用 rndis, 则使用的 IP 为：`192.168.137.100`

RV06 板卡默认开启 telnet 服务，默认用户名和密码为：root/root

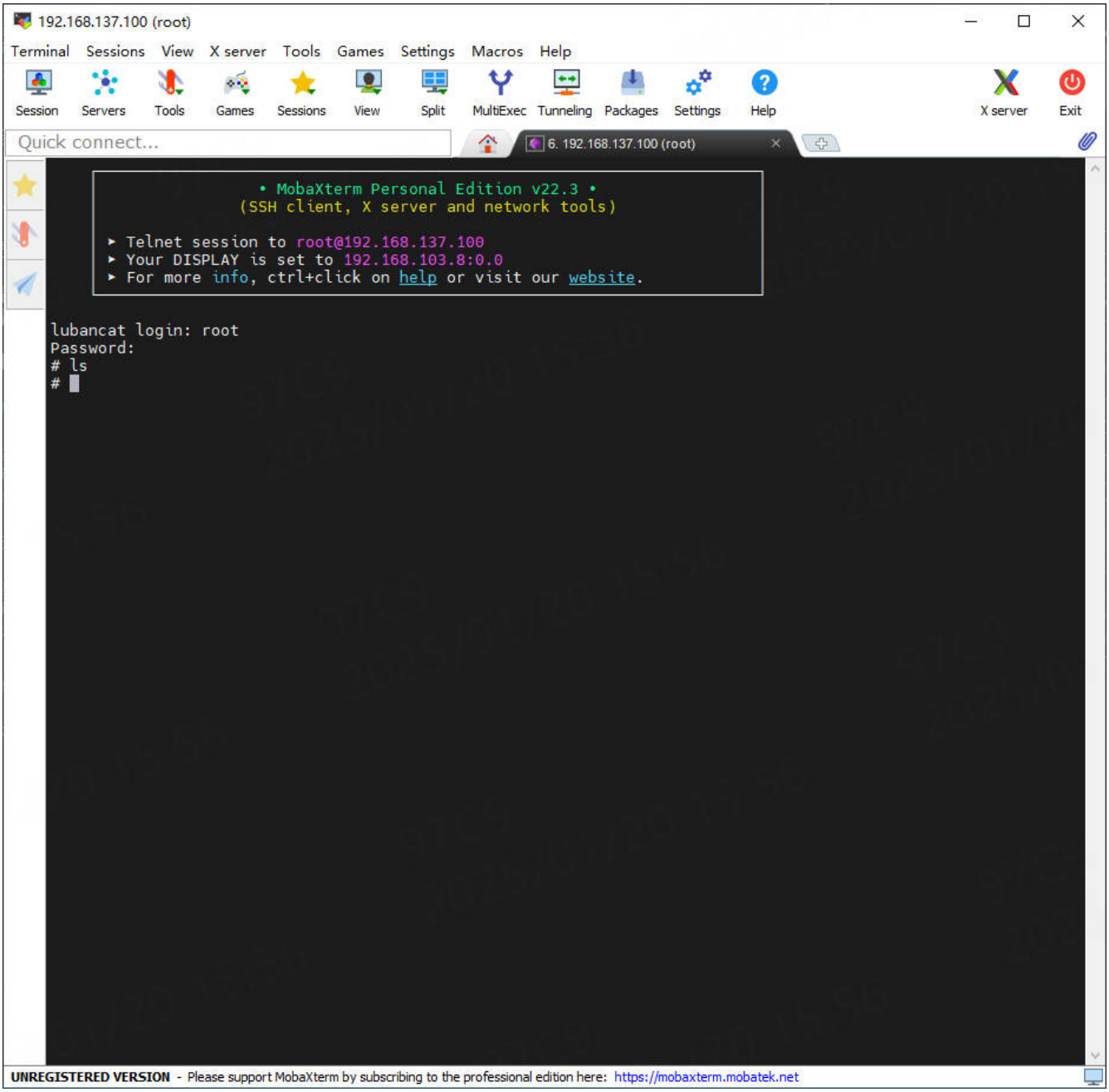
这里以 RNDIS 为例，使用 telnet 连接板卡：



然后会进入登录页面，输入用户名和密码即可



- 输入后按下确认键可以看到下图的情况，表示连接成功


192.168.137.100 (root)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
6. 192.168.137.100 (root)
• MobaXterm Personal Edition v22.3 •
(SSH client, X server and network tools)
▸ Telnet session to root@192.168.137.100
▸ Your DISPLAY is set to 192.168.103.8:0.0
▸ For more info, ctrl+click on help or visit our website.
lubancat login: root
Password:
ls
#
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

第 9 章　文件传输

9.1 SSH 文件传输

SSH 文件传输支持 SCP 和 SFTP 两种协议：

- SCP 是 Secure Copy 的缩写，是一种在 Linux 下用来进行远程拷贝文件的命令。
- SFTP 是 SSH File Transfer Protocol 的缩写，是一种安全的文件传输协议。

注解： SSH 文件传输需要先获取板卡的 IP 地址。IP 地址获取可以参考网络获取章节

9.1.1 SCP

SCP (Secure Copy) 是一个在 Linux 下用来进行远程拷贝文件的命令。推荐 Linux 系统和板卡之间的文件传输使用

SCP 命令的基本格式如下：

```
scp [参数] [源文件] [目标文件]
```

其中，参数可以是以下选项之一：

- -r：递归复制整个目录。
- -p：保留源文件的修改时间和访问时间。
- -P：指定远程主机的端口号。默认端口号是 22。

假设我的板卡的 ip 地址 `192.168.137.100`

例如，将 Linux PC 虚拟机中的文件 *file.txt* 复制到板卡的 */root/* 目录下，可以使用以下命令：

```
# 将文件 file.txt 复制到板卡的 /root/ 目录下
scp file.txt root@192.168.137.100:/root/
# 将文件夹 dir 复制到板卡的 /root/ 目录下
scp -r dir root@192.168.137.100:/root/
```

- 除了能将文件从本地复制到远程主机，SCP 还可以将文件从远程主机复制到本地主机。

```
# 将板卡的 /root/file.txt 复制到 Linux PC 虚拟机的 /home/ 目录下
scp root@192.168.137.100:/root/file.txt /home/
# 将板卡的 /root/dir 目录复制到 Linux PC 虚拟机的 /home/ 目录下
scp -r root@192.168.137.100:/root/dir /home/
```

传输的时候需要输入密码，输入板卡的密码即可。

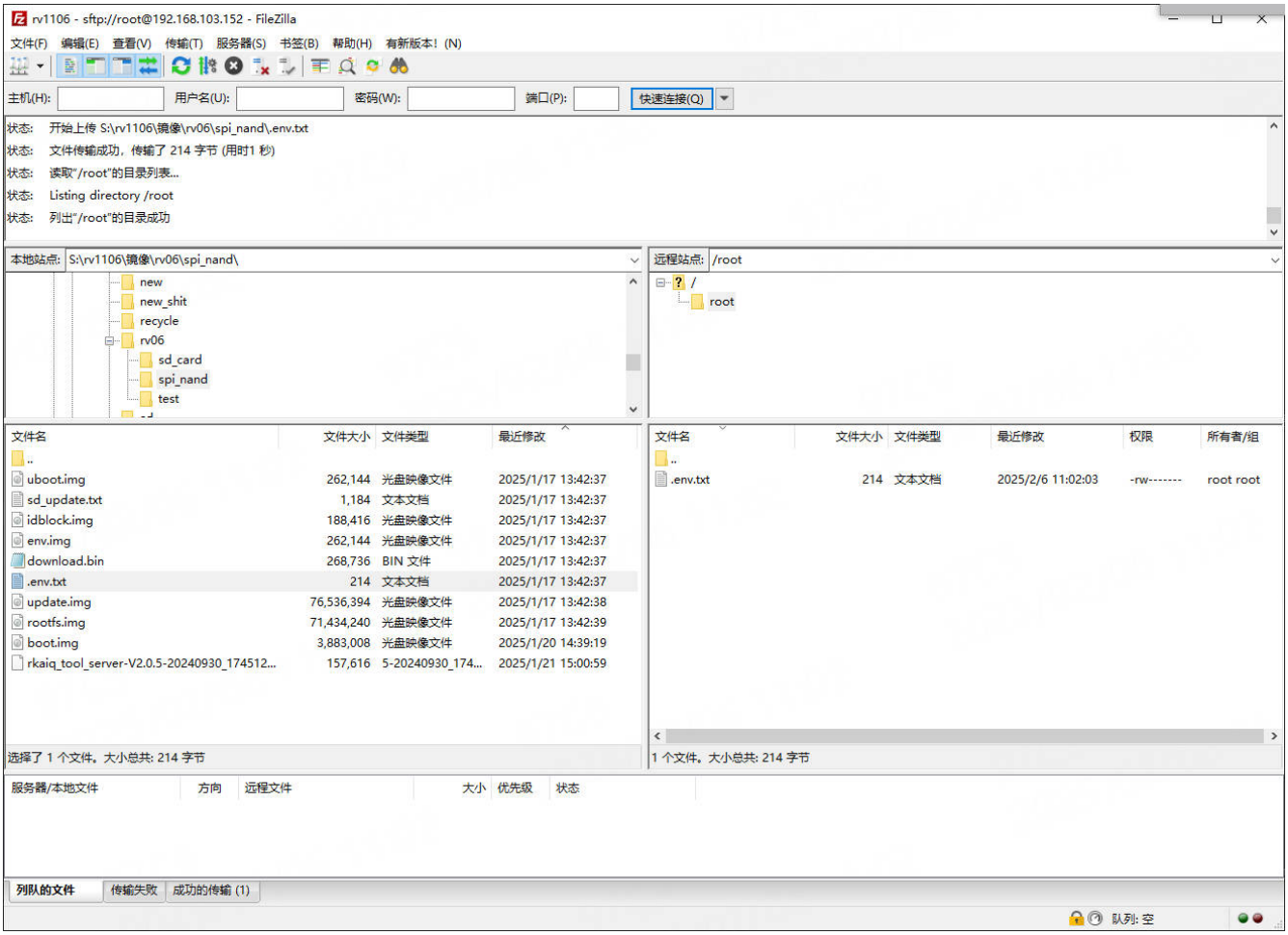
9.1.2 SFTP

SFTP (SSH File Transfer Protocol) 是一个安全的文件传输协议，可以在本地和远程系统之间传输文件。推荐 Windows 系统和板卡之间的文件传输使用

SFTP 传输软件有很多，例如 FileZilla 、WinSCP 等。这里以 FileZilla 为例。

1. 下载 FileZilla 客户端软件，下载地址：https://filezilla-project.org/download.php?type=client
2. 安装 FileZilla 客户端软件
3. 打开 FileZilla 客户端软件，点击 文件 -> 站点管理器 -> 新站点，选择 SFTP 协议，填写板卡的 IP 地址、用户名、密码、端口号等信息，点击 连接即可连接到板卡。

4. 连接成功即可看到下面内容



5. 双击文件即可对文件进行上传和下载

9.2 TF 卡文件传输

使用 TF 卡插入板卡后，板卡会自动挂载 TF 卡，挂载到 `/mnt/sdcard` 目录下。

然后就可以通过 `cp` 命令或者 `mv` 命令进行文件的复制和移动。

9.3 u 盘文件传输

- 使用 u 盘时，需要把拨码开关拨到 1 的位置
- 插入 u 盘后可以看到系统会识别到 u 盘，u 盘不会自动挂载，需要自己手动挂载

```
[   82.696591] usb 1-1.1: new high-speed USB device number 9 using xhci-hcd
[   82.808823] usb 1-1.1: New USB device found, idVendor=090c,␣
↪idProduct=1000, bcdDevice=11.00
[   82.808896] usb 1-1.1: New USB device strings: Mfr=1, Product=2,␣
↪SerialNumber=3
[   82.808923] usb 1-1.1: Product: LanKxin USB 3.1
[   82.808946] usb 1-1.1: Manufacturer: LanKxin
[   82.808970] usb 1-1.1: SerialNumber: LKX-2114E2LQK4
[   82.815882] usb-storage 1-1.1:1.0: USB Mass Storage device detected
[   82.819982] scsi host0: usb-storage 1-1.1:1.0
[   84.028046] scsi 0:0:0:0: Direct-Access     LanKxin  LanKxin USB 3.1 ␣
↪1100 PQ: 0 ANSI: 6
[   84.035715] sd 0:0:0:0: Attached scsi generic sg0 type 0
[   84.038041] sd 0:0:0:0: [sda] 60948480 512-byte logical blocks: (31.2 GB/
↪29.1 GiB)
[   84.039482] sd 0:0:0:0: [sda] Write Protect is off
[   84.041005] sd 0:0:0:0: [sda] Write cache: enabled, read cache: enabled,␣
↪doesn't support DPO or FUA
[   84.054603]  sda: sda1
[   84.059363] sd 0:0:0:0: [sda] Attached SCSI removable disk
```

- 可以看到 u 盘的设备名为 sda ，分区名为 sda1

手动挂载 u 盘命令：

```
mkdir -p /mnt/usb
mount /dev/sda1 /mnt/usb
```

```
# ls /mnt/usb/
01.h264
1.flac
1.mp3
1.wav
2.mp3
```

第 10 章　引脚分布

RV06 的 30pin 引脚分布如下

LuBanCat RV06系列引脚图			
功能	物理引脚		功能
3.3V	1	2	5V
I2C3_SDA	3	4	5V
I2C3_SCL	5	6	GND
GPIO0_A0	7	8	UART3_TX
GND	9	10	UART3_RX
GPIO0_A1	11	12	PWM7
GPIO0_A2	13	14	GND
GPIO0_A4	15	16	GPIO0_A5
3.3V	17	18	GPIO5_B0
MOSI	19	20	GND
MISO	21	22	GPIO5_B1
SCLK	23	24	CS0
GND	25	26	CS1
I2C4_SDA	27	28	I2C4_SCL
GPIO1_B0	29	30	GND

第 11 章　GPIO

GPIO 是 General Purpose I/O 的缩写，即通用输入输出端口，简单来说就是 MCU/CPU 可控制的引脚，这些引脚通常有多种功能，最基本的是高低电平输入检测和输出，部分引脚还会与主控器的片上外设绑定，如作为串口、I2C、网络、电压检测的通讯引脚。

Linux 提供了 GPIO 子系统驱动框架，使用该驱动框架即可灵活地控制板子上的 GPIO。

11.1 引脚分布图

LuBanCat RV06系列引脚图			
功能	物理引脚		功能
3.3V	1	2	5V
I2C3_SDA	3	4	5V
I2C3_SCL	5	6	GND
GPIO0_A0	7	8	UART3_TX
GND	9	10	UART3_RX
GPIO0_A1	11	12	PWM7
GPIO0_A2	13	14	GND
GPIO0_A4	15	16	GPIO0_A5
3.3V	17	18	GPIO5_B0
MOSI	19	20	GND
MISO	21	22	GPIO5_B1
SCLK	23	24	CS0
GND	25	26	CS1
I2C4_SDA	27	28	I2C4_SCL
GPIO1_B0	29	30	GND

11.2 GPIO 命名

Rockchip Pin 的 ID 按照控制器 (bank)+ 端口 (port)+ 索引序号 (pin) 组成。

- 控制器和 GPIO 控制器数量一致 (GPIO0, GPIO1, GPIO2, GPIO3, GPIO4)
- 端口固定 A、B、C 和 D，每个端口仅有 8 个索引号,(a=0,b=1,c=2,d=3)
- 索引序号固定 0、1、2、3、4、5、6、7

rv1106 具有 5 个 GPIO 控制器，可以控制 32 个 IO，作为 GPIO 功能时，端口行为由 GPIO 控制器寄存器配置。

GPIO1_C4 表达的意思为第 1 组控制器，端口号为 C，索引号为 4。该引脚号的计算公式为 `32 x 1 + 2 x 8 + 4 = 52`

- 例如 GPIO1_C4，计算公式为 `32 x 1 + 2 x 8 + 4 = 52`
- 例如 GPIO3_B2，计算公式为 `32 x 3 + 8 x 1 + 2 = 106`
- 例如 GPIO0_D6，计算公式为 `32 x 0 + 8 x 3 + 6 = 30`
- 例如 GPIO4_A7，计算公式为 `32 x 4 + 8 x 0 + 7 = 135`

板卡上除了主控芯片原生 IO，也使用 i2c 扩展 gpio(GPIO5), 使用拓展 IO 的计算方式不一样，基地址是 272，因此基于 GPIO5 的引脚公式需要加上基地址,

- 例如 GPIO5_PA4，计算公式为 `272 + 0 x 8 + 4 = 276`
- 例如 GPIO5_PB4，计算公式为 `272 + 1 x 8 + 4 = 284`

11.3 使用 GPIO sysfs 接口控制 IO

在 Linux 中，最常见的读写 GPIO 方式就是用 GPIO sysfs interface，是通过操作 */sys/class/gpio* 目录下的 *export* 、*unexport* 、*gpio{N}/direction*, *gpio{N} /value* （用实际引脚号替代 {N}）等文件实现的，经常出现 shell 脚本里面。在 kernel 4.8 开始，加入了 libgpiod 的支持；而原有基于 sysfs 的访问方式，将被逐渐放弃。

表 1: GPIO 举例计算

引脚	控制器	端口号	索引号	计算结果
GPIO1_C4	1	C	4	52 (32 x 1 + 8 x 2 + 4)
GPIO3_B2	3	B	2	106 (32 x 3 + 8 x 1 + 2)
GPIO0_D6	0	D	6	30 (32 x 0 + 8 x 3 + 6)
GPIO5_PA4	4	0	4	276 (272 + 8 x 0 + 4)
GPIO5_PB4	4	1	6	284 (272 + 8 x 1 + 4)

```
# 以下所有操作均需要打开管理者权限使用
# 使能引脚 GPIO1_C4
echo 52 > /sys/class/gpio/export

# 设置引脚为输入模式
echo in > /sys/class/gpio/gpio52/direction
# 读取引脚的值
cat /sys/class/gpio/gpio52/value

# 设置引脚为输出模式
echo out > /sys/class/gpio/gpio52/direction
# 设置引脚为低电平
echo 0 > /sys/class/gpio/gpio52/value
# 设置引脚为高电平
echo 1 > /sys/class/gpio/gpio52/value

# 复位引脚
echo 52 > /sys/class/gpio/unexport
```

11.4 使用 libgpiod 控制 IO(推荐)

libgpiod 是一种字符设备接口，GPIO 访问控制是通过操作字符设备文件（比如 */dev/gpiodchip0* ）实现的，并通过 libgpiod 提供一些命令工具、c 库以及 python 封装。

```
# 安装 gpiod 命令行工具
sudo apt install gpiod
```

- gpiod 工具的使用方法与 sysfs 接口的不同，gpiod 是以控制器为单位，然后再详细到端口号和索引号，即 gpiod 使用两个数据确定引脚

表 2: GPIO 举例计算

引脚	控制器	端口号	索引号	gpiod 的使用结果
GPIO1_C4	1	C	4	1 20(8 x 2 + 4)
GPIO3_B2	3	B	2	3 10(8 x 1 + 2)
GPIO0_D6	0	D	6	0 30(8 x 3 + 6)
GPIO5_PB4	5	1	4	5 12(8 x 1 + 4)
GPIO5_PA4	5	0	4	5 4(8 x 0 + 4)

常用的命令行如下，可使用 *-h* 查看命令相对应的使用说明（以 GPIO1_C4 为例）

表 3: **libgpiod 命令**

命令	作用	使用举例	说明
gpiodetect	列出所有的 GPIO 控制器	gpiodetect(无参数)	列出所有的 GPIO 控制器
gpioinfo	列出 gpio 控制器的引脚情况	gpioinfo 1	列出第一组控制器引脚组情况
gpioset	设置 gpio	gpioset 1 20=0	设置第一组控制器编号 20 引脚为低电平
gpioget	获取 gpio 引脚状态	gpioget 1 20	获取第一组控制器编号 20 的引脚状态
gpiomon	监控 gpio 的状态	gpiomon 1 20	监控第一组控制器编号 20 的引脚状态

重要： Rockchip Pin 的 ID 按照控制器 (bank)+ 端口 (port)+ 索引序号 (pin) 组成。其中端口号和索引号会合并成一个数值传入到 gpiod 里去并不是所有的引脚都能够使用 libgpiod 控制，例如 led 之类的一些已经被使用的引脚。当使用这些被定义的引脚就会出现，设备繁忙，进而无法使用

11.5 FAQs

Q1：当使用 GPIO 时出现 `gpioset: error setting the GPIO line values: Device or res`

A1：说明 GPIO 被占用了，占用的原因可能是设备树里把该引脚作为 gpio 或者其他复用功能被使用了。

第 12 章　I2C 通讯

本章简单介绍使用 I2C 总线与外部设备的通讯，讲解 I2C 设备的简单使用

12.1 i2c

I2C(Inter - Integrated Circuit) 是一种通用的总线协议。它是由 Philips(飞利浦) 公司，现 NXP(恩智浦) 半导体开发的一种简单的双向两线制总线协议标准。

LubanCat-RK 系列板卡的 I2C 控制器支持下列功能

- 兼容 I2C 与 SMBus 总线
- 仅支持主模式下的 I2C 总线
- 软件可编程时钟频率支持到 400kbps, 最高可达 1000kbps
- 支持 7 位和 10 位寻址模式
- 一次中断或轮询至多 32 个字节的数据传输

LubanCat-RV06 板卡的 30pin 接口上有两个 I2C 接口，分别是 I2C-3 和 I2C-4

LuBanCat RV06系列引脚图			
功能	物理引脚		功能
3.3V	1	2	5V
I2C3_SDA	3	4	5V
I2C3_SCL	5	6	GND
GPIO0_A0	7	8	UART3_TX
GND	9	10	UART3_RX
GPIO0_A1	11	12	PWM7
GPIO0_A2	13	14	GND
GPIO0_A4	15	16	GPIO0_A5
3.3V	17	18	GPIO5_B0
MOSI	19	20	GND
MISO	21	22	GPIO5_B1
SCLK	23	24	CS0
GND	25	26	CS1
I2C4_SDA	27	28	I2C4_SCL
GPIO1_B0	29	30	GND

表 1: 通用 i2c 引脚

I2C	引脚	功能
I2C-SCL	5,28	i2c 的时钟信号线
I2C-SDA	3,27	i2c 的数据线

12.2 检查 I2C 设备

可以通过一下命令查看 i2c 总线有没有开启

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

如下所示:

```
# ls /dev/i2c-*
/dev/i2c-3  /dev/i2c-4
#
```

可以看到出现了两个 I2C 设备，分别是 i2c-3 和 i2c-4

12.3 连接设备

将 mpu6050 接入到 i2c-3 的总线上，如下图所示

```
# 板卡与 mpu6050 连接

板子  ------  mpu6050
3.3V(1)  ------  VCC
GND(6)  ------  GND
SCL(5)  ------  SCL
SDA(3)  ------  SDA
```

12.4 i2c-tools 测试

LubanCat-RV06 板卡自带 i2c-tools

使用 i2c-tools 工具包提供了一些非常方便的工具来对系统的 I2C 总线进行调试，安装后可使用的命令有 i2cdetect、i2cdump、i2cset 以及 i2cget, 用于扫描 I2C 总线上的设备、读写指定设备的寄存器等。

然后查看挂载在 i2c-3 上的器件情况，输出内容如下所示：

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

其中“68”是为 MPU6050 的设备地址，常用的命令还有以下几个。

```
# 检测当前系统有几组 i2c 总线
i2cdetect -l

# 查看 i2c-0 接口上的设备
i2cdetect -a -y 3

# 读取指定设备的全部寄存器的值。
i2cdump  -f -y 3 0x68

# 读取指定 IIC 设备的某个寄存器的值，如下读取地址为 0x68 器件中的 0x01 寄存器值。
```

(下页继续)

(续上页)

```
i2cget -f -y 3 0x68 0x01

# 写入指定 IIC 设备的某个寄存器的值，如下设置地址为 0x68 器件中的 0x01 寄存器值为 0x6f;

i2cset -f -y 3 0x68 0x01 0x6f
```

12.5 读取陀螺仪传感器数据实验

12.5.1 实验说明

本教程将通过 IIC 接口读取陀螺仪 (MPU6050) 的原始数据。本次实验会以 i2c-3 做为示例，i2c-4 的操作和 i2c-3 的一样，当然，如果您没有 mpu6050 模块，可以通过学习操作 mpu6050 的方式操作您想要操作的 i2c 设备在测试程序中大约每一秒读取并显示一次 MPU6050 的原始数据

查看 IIC 设备文件，确保 IIC 3 接口已经使能

```
root@lubancat:~# ls /dev/i2c-*
/dev/i2c-3  /dev/i2c-4
root@lubancat:~#
```

其中“i2c-3”就是 MPU6050 使用到的 IIC 3 接口总线。

12.5.2 ioctl 函数

在编写应用程序时需要使用 ioctl 函数设置 i2c 相关配置，其函数原型如下

```
#include <sys/ioctl.h>

int ioctl(int fd, unsigned long request, ...);
```

其中对于终端 request 的值常用的有以下几种

I2C_RETRIES	设置收不到 ACK 时的重试次数，默认为 1
I2C_TIMEOUT	设置超时时限的 jiffies
I2C_SLAVE	设置从机地址
I2C_SLAVE_FORCE	强制设置从机地址
I2C_TENBIT	选择地址长度 0 为 7 位地址，非 0 为 10 位

12.5.3 编写应用程序

根据 ioctl 相关参数即可编写与 i2c 相关的接口函数，读取 mpu6050 原始数据程序如下

代码较长复制粘贴容易乱序，可以下载我们提供的源码 `i2c_mpu6050.c`

列表 1: quick_start/i2c/i2c_mpu6050.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <stdint.h>
4  #include <unistd.h>
5  #include <sys/types.h>
6  #include <sys/stat.h>
7  #include <fcntl.h>
8  #include <linux/i2c.h>
9  #include <linux/i2c-dev.h>
10 #include <sys/ioctl.h>
11
12 /* 寄存器地址 */
13 #define SMPLRT_DIV      0x19
14 #define PWR_MGMT_1      0x6B
15 #define CONFIG          0x1A
16 #define ACCEL_CONFIG    0x1C
17
18 #define ACCEL_XOUT_H    0x3B
```

(下页继续)

(续上页)

```
19 #define ACCEL_XOUT_L     0x3C
20 #define ACCEL_YOUT_H     0x3D
21 #define ACCEL_YOUT_L     0x3E
22 #define ACCEL_ZOUT_H     0x3F
23 #define ACCEL_ZOUT_L     0x40
24 #define GYRO_XOUT_H      0x43
25 #define GYRO_XOUT_L      0x44
26 #define GYRO_YOUT_H      0x45
27 #define GYRO_YOUT_L      0x46
28 #define GYRO_ZOUT_H      0x47
29 #define GYRO_ZOUT_L      0x48
30
31 //从机地址 MPU6050 地址
32 #define Address          0x68
33
34 //MPU6050 操作相关函数
35 static int mpu6050_init(int fd,uint8_t addr);
36 static int i2c_write(int fd, uint8_t addr,uint8_t reg,uint8_t val);
37 static int i2c_read(int fd, uint8_t addr,uint8_t reg,uint8_t * val);
38 static short GetData(int fd,uint8_t addr,unsigned char REG_Address);
39
40 int main(int argc,char *argv[] )
41 {
42     int  fd;
43     fd = I2C_SLAVE;
44
45     if(argc < 2){
46         printf("Wrong use !!!!\n");
47             printf("Usage: %s [dev]\n",argv[0]);
48             return -1;
49         }
50
```

(下页继续)

(续上页)

```
51      fd = open(argv[1], O_RDWR); // open file and enable read and  write
52      if (fd < 0){
53          perror("Can't open\n"); // open i2c dev file fail
54          exit(1);
55      }
56
57      //初始化 MPU6050
58      mpu6050_init(fd,Address);
59      while(1){
60          usleep(1000 * 10);
61          printf("ACCE_X:%6d\n ", GetData(fd,Address,ACCEL_XOUT_H));
62          usleep(1000 * 10);
63          printf("ACCE_Y:%6d\n ", GetData(fd,Address,ACCEL_YOUT_H));
64          usleep(1000 * 10);
65          printf("ACCE_Z:%6d\n ", GetData(fd,Address,ACCEL_ZOUT_H));
66          usleep(1000 * 10);
67          printf("GYRO_X:%6d\n ", GetData(fd,Address,GYRO_XOUT_H));
68          usleep(1000 * 10);
69          printf("GYRO_Y:%6d\n ", GetData(fd,Address,GYRO_YOUT_H));
70          usleep(1000 * 10);
71          printf("GYRO_Z:%6d\n\n ", GetData(fd,Address,GYRO_ZOUT_H));
72          sleep(1);
73      }
74
75      close(fd);
76
77      return 0;
78  }
79
80  static int mpu6050_init(int fd,uint8_t addr)
81  {
82      i2c_write(fd, addr,PWR_MGMT_1,0x00);   //配置电源管理，0x00，正常启动
```

(下页继续)

(续上页)

```
83      i2c_write(fd, addr,SMPLRT_DIV,0x07);   //设置 MPU6050 的输出分频既设置采样
84      i2c_write(fd, addr,CONFIG,0x06);   //配置数字低通滤波器和帧同步引脚
85      i2c_write(fd, addr,ACCEL_CONFIG,0x01);   //设置量程和 X、Y、Z 轴加速度自检
86
87      return 0;
88  }
89
90  static int i2c_write(int fd, uint8_t addr,uint8_t reg,uint8_t val)
91  {
92      int retries;
93      uint8_t data[2];
94
95      data[0] = reg;
96      data[1] = val;
97
98      //设置地址长度: 0 为 7 位地址
99      ioctl(fd,I2C_TENBIT,0);
100
101     //设置从机地址
102     if (ioctl(fd,I2C_SLAVE,addr) < 0){
103         printf("fail to set i2c device slave address!\n");
104         close(fd);
105         return -1;
106     }
107
108     //设置收不到 ACK 时的重试次数
109     ioctl(fd,I2C_RETRIES,5);
110
111     if (write(fd, data, 2) == 2){
112         return 0;
113     }
114     else{
```

(下页继续)

(续上页)

```
115         return -1;
116     }
117 
118 }
119 
120 static int i2c_read(int fd, uint8_t addr,uint8_t reg,uint8_t * val)
121 {
122     int retries;
123 
124     //设置地址长度: 0 为 7 位地址
125     ioctl(fd,I2C_TENBIT,0);
126 
127     //设置从机地址
128     if (ioctl(fd,I2C_SLAVE,addr) < 0){
129         printf("fail to set i2c device slave address!\n");
130         close(fd);
131         return -1;
132     }
133 
134     //设置收不到 ACK 时的重试次数
135     ioctl(fd,I2C_RETRIES,5);
136 
137     if (write(fd, &reg, 1) == 1){
138         if (read(fd, val, 1) == 1){
139                 return 0;
140         }
141     }
142     else{
143         return -1;
144     }
145 }
146 
```

(下页继续)

(续上页)

```
147 static short GetData(int fd,uint8_t addr,unsigned char REG_Address)
148 {
149     char H, L;
150 
151     i2c_read(fd, addr,REG_Address, &H);
152     usleep(1000);
153     i2c_read(fd, addr,REG_Address + 1, &L);
154     return (H << 8) +L;
155 }
```

将代码放到虚拟机上，然后使用交叉编译工具进行交叉编译，然后将生成的可执行文件发送到板卡上，执行即可。

```
1 # 交叉编译程序
2 arm-rockchip830-linux-uclibcgnueabihf-gcc i2c_mpu6050.c -o mpu6050
3 
4 # 将生成的可执行文件发送到板卡上
5 scp mpu6050 root@192.168.103.100:/root
6 
7 ./mpu6050 /dev/i2c-3
```

三次数据的采集如下所示

```
1 # ./mpu6050 /dev/i2c-3
2 ACCE_X:  5384
3 ACCE_Y: -8954
4 ACCE_Z:  9190
5 GYRO_X: -2002
6 GYRO_Y:  4846
7 GYRO_Z: -1821
8 
9 ACCE_X: -7798
```

(下页继续)

(续上页)

```
ACCE_Y:-14158
ACCE_Z: -2296
GYRO_X:-32768
GYRO_Y: 21344
GYRO_Z: 32767

ACCE_X:-21530
ACCE_Y: 12048
ACCE_Z:-22990
GYRO_X: 32767
GYRO_Y:-19531
GYRO_Z:-32768
```

第 13 章　SPI 通信

SPI 是一种全双工的通信方式，可以实现高速的数据传输。SPI 总线上有一个主设备和一个或多个从设备。主设备通过片选信号来选择从设备，然后通过 SPI 总线与从设备进行通信。SPI 总线上的数据传输是通过一个数据输入线和一个数据输出线来实现的，数据输入线和数据输出线是分开的，因此 SPI 总线是全双工的。

LubanCat-RV06 板卡的 30pin 接口上只有 1 个 SPI 接口，有两个 CS 引脚

LuBanCat RV06系列引脚图			
功能	物理引脚		功能
3.3V	1	2	5V
I2C3_SDA	3	4	5V
I2C3_SCL	5	6	GND
GPIO0_A0	7	8	UART3_TX
GND	9	10	UART3_RX
GPIO0_A1	11	12	PWM7
GPIO0_A2	13	14	GND
GPIO0_A4	15	16	GPIO0_A5
3.3V	17	18	GPIO5_B0
MOSI	19	20	GND
MISO	21	22	GPIO5_B1
SCLK	23	24	CS0
GND	25	26	CS1
I2C4_SDA	27	28	I2C4_SCL
GPIO1_B0	29	30	GND

SPI	引脚	功能
MOSI	19	主设备输出/从设备输入
MISO	21	主设备输入/从设备输出
CLOCK	23	时钟信号线
CS0	24	片选信号线 0
CS1	26	片选信号线 1

13.1 检查 SPI 设备

```
# ls /dev/spidev*
/dev/spidev1.0  /dev/spidev1.1
#
```

- 可以看到出现了两个 SPI 设备，分别是 spidev1.0 和 spidev1.1，spidev1.0 和 spidev1.1 的区别在于片选信号的不同，spidev1.0 使用 CS0 , spidev1.1 使用 CS1

13.2 回环测试程序

根据 ioctl 相关参数即可编写 spi 相关测试程序，此处为了简单仅做回环测试实验，只需要将 SPI3 的 MIOS 与 MOSI 引脚 (板卡上的 19 和 21) 使用跳线帽短接即可。

代码较长复制粘贴容易乱序，可以下载我们提供的源码 `spi_selftest.c`

列表 1: quick_start/spi/spi_selftest.c

```
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
```

(下页继续)

(续上页)

```c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <linux/spi/spidev.h>

/*SPI 接收 、发送 缓冲区 */
unsigned char tx_buffer[100] = "hello the world !";
unsigned char rx_buffer[100];


int fd;                                                    // SPI 控制引脚的设备文件描述符
static unsigned  mode = SPI_MODE_2;          //用于保存 SPI 工作模式
static uint8_t bits = 8;                                   // 接收、发送数据位数
static uint32_t speed = 10000000;                          // 发送速度
static uint16_t delay;                                     //保存延时时间

void transfer(int fd, uint8_t const *tx, uint8_t const *rx, size_t len)
{
          int ret;

          struct spi_ioc_transfer tr = {
               .tx_buf = (unsigned long)tx,
               .rx_buf = (unsigned long)rx,
               .len = len,
               .delay_usecs = delay,
               .speed_hz = speed,
               .bits_per_word = bits,
               .tx_nbits = 1,
               .rx_nbits = 1
          };
```

(下页继续)

(续上页)

```
36
37          ret = ioctl(fd, SPI_IOC_MESSAGE(1), &tr);
38          if (ret < 1)
39                  printf("can't send spi message\n");
40  }
41
42  void spi_init(int fd)
43  {
44          int ret = 0;
45
46        // spi mode 设置 SPI 工作模式
47          ret = ioctl(fd, SPI_IOC_WR_MODE32, &mode);
48          if (ret == -1)
49                  printf("can't set spi mode\n");
50
51        //  bits per word  设置一个字节的位数
52          ret = ioctl(fd, SPI_IOC_WR_BITS_PER_WORD, &bits);
53          if (ret == -1)
54            printf("can't set bits per word\n");
55
56        //  max speed hz  设置 SPI 最高工作频率
57          ret = ioctl(fd, SPI_IOC_WR_MAX_SPEED_HZ, &speed);
58          if (ret == -1)
59            printf("can't set max speed hz\n");
60
61        // 打印
62           printf("spi mode: 0x%x\n", mode);
63      printf("bits per word: %d\n", bits);
64      printf("max speed: %d Hz (%d KHz)\n", speed, speed / 1000);
65  }
66
67  int main(int argc, char *argv[])
```

(下页继续)

(续上页)

```
68 {
69         int fd;
70 
71         if(argc < 2){
72     printf("Wrong use !!!!\n");
73         printf("Usage: %s [dev]\n",argv[0]);
74         return -1;
75     }
76         /* 打开 SPI 设备 */
77     fd = open(argv[1], O_RDWR); // open file and enable read and  write
78     if (fd < 0){
79         printf("Can't open %s \n",argv[1]); // open i2c dev file fail
80         exit(1);
81     }
82 
83         /* 初始化 SPI */
84         spi_init(fd);
85 
86         /* 执行发送 */
87         transfer(fd, tx_buffer, rx_buffer, sizeof(tx_buffer));
88 
89         /* 打印 tx_buffer 和 rx_buffer*/
90         printf("tx_buffer: \n %s\n ", tx_buffer);
91         printf("rx_buffer: \n %s\n ", rx_buffer);
92 
93         close(fd);
94 
95         return 0;
96 }
```

13.2.1 编译运行

将代码放到虚拟机上，然后使用交叉编译工具进行交叉编译，然后将生成的可执行文件发送到板卡上，执行即可。

```
# 交叉编译程序
arm-rockchip830-linux-uclibcgnueabihf-gcc spi_selftest.c -o spi_selftest

# 将生成的可执行文件发送到板卡上
scp spi_selftest root@192.168.103.100:/root

# 执行
./spi_selftest /dev/spidev1.0
```

运行结果如下：

```
# ./spi_selftest /dev/spidev1.0
spi mode: 0x2
bits per word: 8
max speed: 10000000 Hz (10000 KHz)
tx_buffer:
hello the world !
rx_buffer:
hello the world !
#
```

第 14 章　PWM

PWM 是脉冲宽度调制，是一种模拟控制信号，通过改变脉冲的宽度来控制模拟信号的输出。

LubanCat-RV06 板卡的 30pin 接口上默认只有 1 个 PWM 接口

LuBanCat RV06系列引脚图			
功能	物理引脚		功能
3.3V	1	2	5V
I2C3_SDA	3	4	5V
I2C3_SCL	5	6	GND
GPIO0_A0	7	8	UART3_TX
GND	9	10	UART3_RX
GPIO0_A1	11	12	PWM7
GPIO0_A2	13	14	GND
GPIO0_A4	15	16	GPIO0_A5
3.3V	17	18	GPIO5_B0
MOSI	19	20	GND
MISO	21	22	GPIO5_B1
SCLK	23	24	CS0
GND	25	26	CS1
I2C4_SDA	27	28	I2C4_SCL
GPIO1_B0	29	30	GND

PWM	引脚	功能
PWM7	12	PWM 输出

14.1 检查 PWM 设备

```
ls /sys/class/pwm/
```

```
# ls /sys/class/pwm/
pwmchip0  pwmchip7
#
```

- 可以看到出现了两个 PWM 设备，分别是 pwmchip0 和 pwmchip7，pwmchip0 是屏幕背光，pwmchip7 是 30pin 排针上的 PWM 接口。

14.2 PWM 模式

PWM 模式有两种模式 - `Continous`：连续模式，周期性的输出 PWM 信号 - `OneShot`：单次模式，输出特定次数的 PWM 信号

14.3 命令行操作

14.3.1 Continous

```
# 将 pwm7 导出到用户空间
echo 0 > /sys/class/pwm/pwmchip7/export

# 设置 pwm 周期 单位为 ns
```

(下页继续)

(续上页)

```
echo 1000000 > /sys/class/pwm/pwmchip7/pwm0/period

# 设置占空比
echo 500000 > /sys/class/pwm/pwmchip7/pwm0/duty_cycle

# 设置 pwm 极性
echo "normal" > /sys/class/pwm/pwmchip7/pwm0/polarity

# 使能 pwm
echo 1 > /sys/class/pwm/pwmchip7/pwm0/enable
```

- 取消使用

```
# 取消将 pwm3 导出到用户空间
echo 0 > /sys/class/pwm/pwmchip7/unexport
```

14.3.2 OneShot

```
# 将 pwm7 导出到用户空间
echo 0 > /sys/class/pwm/pwmchip7/export

# 设置 pwm 周期 单位为 ns
echo 1000000 > /sys/class/pwm/pwmchip7/pwm0/period

# 设置占空比
echo 500000 > /sys/class/pwm/pwmchip7/pwm0/duty_cycle

# 设置 pwm 极性
echo "normal" > /sys/class/pwm/pwmchip7/pwm0/polarity

# 设置输出脉冲的次数，最大值为 256
```

(下页继续)

(续上页)

```
14 echo 256 > /sys/class/pwm/pwmchip7/pwm0/oneshot_count
15
16 # 使能 pwm
17 echo 1 > /sys/class/pwm/pwmchip7/pwm0/enable
```

如果想再次出发 OneShot 模式，需要把 pwm 重新使能

```
1 # 不使能 pwm
2 echo 0 > /sys/class/pwm/pwmchip7/pwm0/enable
3
4 # 使能 pwm
5 echo 1 > /sys/class/pwm/pwmchip7/pwm0/enable
```

- 取消使用

```
1 # 取消将 pwm3 导出到用户空间
2 echo 0 > /sys/class/pwm/pwmchip7/unexport
```

第 15 章　串口

通用异步收发器（Universal Asynchronous Receiver/Transmitter），通常称作 UART，是一种串行、异步、全双工的通信协议，在嵌入式领域应用的非常广泛。

LubanCat-RV06 板卡的 30pin 接口上默认只有 1 个 uart 接口

LuBanCat RV06系列引脚图			
功能	物理引脚		功能
3.3V	1	2	5V
I2C3_SDA	3	4	5V
I2C3_SCL	5	6	GND
GPIO0_A0	7	8	UART3_TX
GND	9	10	UART3_RX
GPIO0_A1	11	12	PWM7
GPIO0_A2	13	14	GND
GPIO0_A4	15	16	GPIO0_A5
3.3V	17	18	GPIO5_B0
MOSI	19	20	GND
MISO	21	22	GPIO5_B1
SCLK	23	24	CS0
GND	25	26	CS1
I2C4_SDA	27	28	I2C4_SCL
GPIO1_B0	29	30	GND

表 1: 通用 uart 引脚

uart	引脚	功能
UART_TX	8	发送信号线
UART_RX	10	接收信号线

15.1 检查串口设备

```
# 执行命令查看终端设备
ls /dev/ttyS*
```

运行结果：

```
# ls /dev/ttyS*
/dev/ttyS3
#
```

- /dev/ttyS3 设备就是 30pin 排针上的 uart 接口。

15.2 stty 工具

stty 是 Linux 下用于设置串口参数的工具，可以设置波特率、数据位、停止位、奇偶校验等参数。

15.2.1 查看串口配置

```
# 在板卡的终端执行如下命令
stty -F /dev/ttyS3
```

运行结果：

```
# stty -F /dev/ttyS3
speed 9600 baud; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; swtch = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^
↪R;
werase = ^W; lnext = ^V; flush = ^O; min = 1; time = 0;
-brkint -imaxbel
#
```

- speed 9600 baud; line = 0; 说明串口波特率为 9600，line = 0; 表示无奇偶校验位。
- intr = ^C; quit = ^; erase = ^?; kill = ^U; eof = ^D; eol = <undef>; eol2 = <undef>; swtch = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W; lnext = ^V; flush = ^O; min = 1; time = 0; -brkint -imaxbel 为串口配置参数。

15.2.2 设置串口参数

```
# 设置通讯速率，其中 ispeed 为输入速率，ospeed 为输出速率
stty -F /dev/ttyS3 ispeed 115200 ospeed 115200
```

运行结果：

```
# stty -F /dev/ttyS3 ispeed 115200 ospeed 115200
# stty -F /dev/ttyS3
speed 115200 baud; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; swtch = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^
↪R;
werase = ^W; lnext = ^V; flush = ^O; min = 1; time = 0;
-brkint -imaxbel
#
```

- 可以看到串口波特率已经修改为 115200。

15.2.3 关闭回显

```
# 关闭回显
stty -F /dev/ttyS3 -echo
```

15.3 回环测试

将 TXD 和 RXD 引脚连接起来

```
# 关闭回显
stty -F /dev/ttyS3 -echo

# 读取串口数据
cat /dev/ttyS3 &

# 发送数据
echo "Hello!" > /dev/ttyS3
```

运行结果

```
# stty -F /dev/ttyS3 -echo
# cat /dev/ttyS3 &
[1] 539
# echo "Hello!" > /dev/ttyS3
# Hello!
```

- 可以看到板卡的终端输出了”Hello!”，说明串口正常工作。

15.4 串口通讯实验（Shell）

15.4.1 连接串口

实验前需要使用串口线或 USB 转串口线把它与板卡与电脑连接起来。

- 板子 - 电脑
- TXD —RXD
- RXD —TXD
- GND —GND

```
# 在板卡的终端执行如下命令
stty -F /dev/ttyS3

# 设置通讯速率，其中 ispeed 为输入速率，ospeed 为输出速率
stty -F /dev/ttyS3 ispeed 115200 ospeed 115200
```

15.4.2 与 Windows 主机通讯

配置好串口调试助手后，尝试使用如下命令测试发送数据：

```
# 在板卡上的终端执行如下指令
# 使用 echo 命令向终端设备文件写入字符串"Hello!"、"I'm lubancat"
echo Hello! > /dev/ttyS3
echo "I'm lubancat" > /dev/ttyS3
#Windows 上的串口调试助手会接收到内容
```

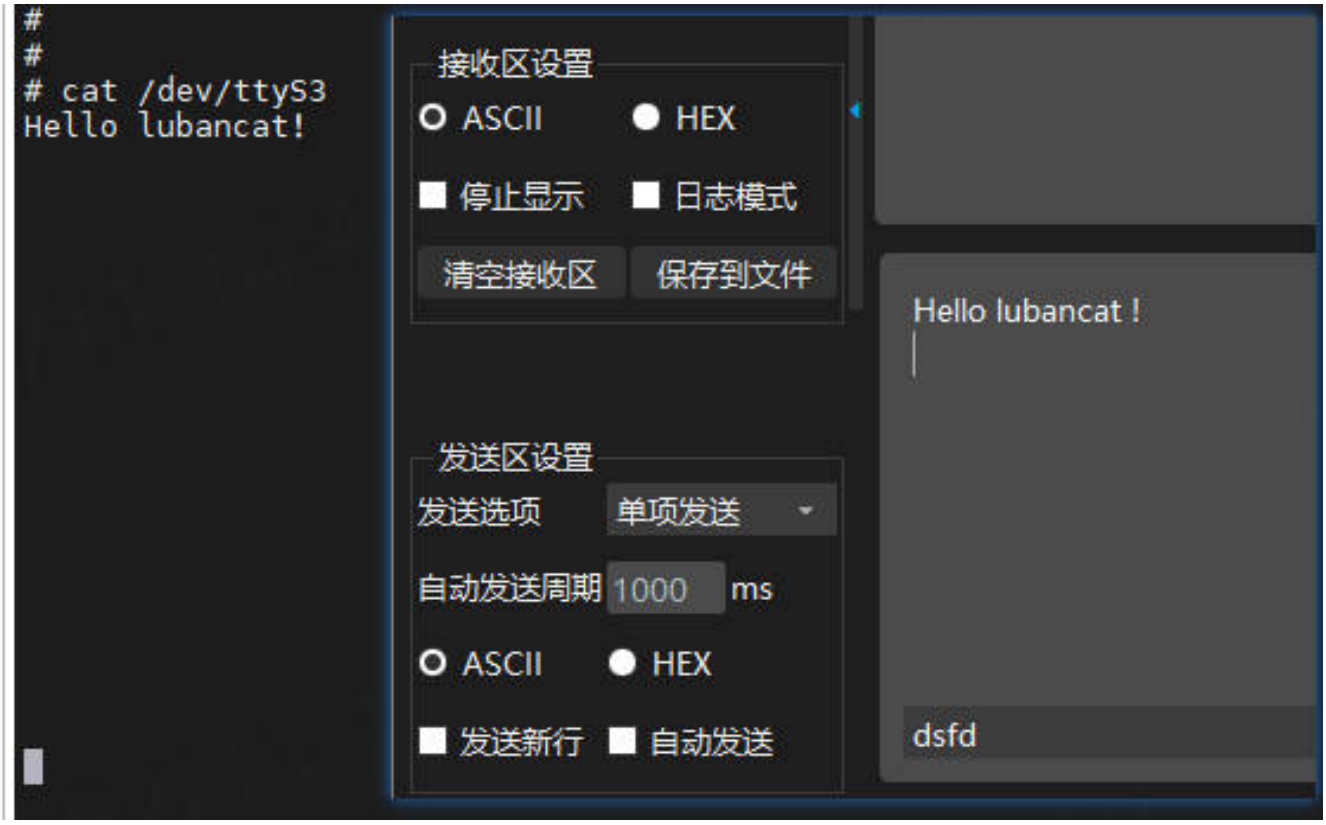
如下图:

可以看到，往/dev/ttyS3 设备文件写入的内容会直接通过串口线发送至 Winodws 的主机。

而读取设备文件则可接收 Winodws 主机发往板卡的内容，可以使用 cat 命令来读取：

```
# 在板卡上的终端执行如下指令
# 使用 cat 命令读取终端设备文件
cat /dev/ttyS3
#cat 命令会等待
# 使用串口调试助手发送字符串
# 字符串最后必须加回车！
# 板卡的终端会输出接收到的内容
```

如下图:


#
#
cat /dev/ttyS3
Hello lubancat!
接收区设置
ASCII
HEX
停止显示
日志模式
清空接收区
保存到文件
Hello lubancat !
发送区设置
发送选项
单项发送
自动发送周期
1000
ms
ASCII
HEX
发送新行
自动发送
dsfd

第 16 章　按键

LubanCat-RV06 板卡上有三个按键，分别是 reset，recovery，ADC 按键。

- reset：仅有复位板卡的功能
- recovery：在板卡上电的时候按住 recovery 按键，可以进入恢复模式，可以烧录固件。进入 linux 系统后可以作为其他按键使用，默认配置为 volume+ 按键。
- ADC：作为 ADC 按键使用，进入 linux 系统后默认配置为 volume- 按键。

在本小节内容中，介绍怎么使用 recovery 和 ADC 按键。

16.1 evtest 工具

linux 系统中有一个工具 evtest 可以用来测试输入设备，可以用来测试按键。

```
evtest
```

运行结果如下：

```
# evtest
No device specified, trying to scan all of /dev/input/event*
Available devices:
/dev/input/event0:      recovery-key
/dev/input/event1:      adc-key
Select the device event number [0-1]:
```

可以看到有 recovery-key 和 adc-key 两个设备，分别对应 recovery 和 ADC 按键。

我们在命令行中输入 0，读取 recovery-key 按键信息。

```
# evtest
No device specified, trying to scan all of /dev/input/event*
Available devices:
/dev/input/event0:      recovery-key
/dev/input/event1:      adc-key
Select the device event number [0-1]: 0
Input driver version is 1.0.1
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "recovery-key"
Supported events:
Event type 0 (EV_SYN)
Event type 1 (EV_KEY)
    Event code 115 (KEY_VOLUMEUP)
Properties:
Testing ... (interrupt to exit)
```

- 可以看到 recovery-key 对应的是 KEY_VOLUMEUP ，即 volume+ 按键。

按下并松开 recovery 按键，可以看到输出信息

```
# evtest
No device specified, trying to scan all of /dev/input/event*
Available devices:
/dev/input/event0:      recovery-key
/dev/input/event1:      adc-key
Select the device event number [0-1]: 0
Input driver version is 1.0.1
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "recovery-key"
Supported events:
Event type 0 (EV_SYN)
Event type 1 (EV_KEY)
    Event code 115 (KEY_VOLUMEUP)
```

(下页继续)

(续上页)

```
Properties:
Testing ... (interrupt to exit)
Event: time 1739258561.998534, type 1 (EV_KEY), code 115 (KEY_VOLUMEUP), value 1
Event: time 1739258561.998534, -------------- SYN_REPORT ------------
Event: time 1739258562.205217, type 1 (EV_KEY), code 115 (KEY_VOLUMEUP), value 0
Event: time 1739258562.205217, -------------- SYN_REPORT ------------
```

- 可以看到 value 1 表示按下按键，value 0 表示松开按键。

16.2 dev 接口

16.2.1 查看按键设备

按键在设备树配置成按键后，会在 /dev/input/ 目录下生成对应的设备节点。

```
ls /dev/input/
```

结果如下：

```
# ls /dev/input/
by-path  event0   event1
#
```

可以看到有 event0 和 event1 两个设备节点。单凭设备节点无法区分是哪个按键。

使用下面命令可以查看 event0 和 event1 对应的按键。

```
ls -l /dev/input/by-path/
```

运行结果：

```
# ls -l /dev/input/by-path/
total 0
lrwxrwxrwx    1 root     root             9 Feb 10 06:09 platform-recovery-
↪key-event -> ../event0
lrwxrwxrwx    1 root     root             9 Feb 10 06:09 platform-adc-key-
↪event -> ../event1
```

可以看到 platform-recovery-key-event 对应 event0 , platform-adc-key-event 对应 event1 。

而 platform-recovery-key-event 对应的是 recovery 按键，platform-adc-key-event 对应的是 ADC 按键。

16.2.2 读取按键信息

这里以读取 recovery 按键为例，读取按键按下的信息。

需要使用下面命令获取按键信息

```
# 读取按键信息
hexdump /dev/input/event0

# 然后按下 recovery 按键，然后松开
```

结果如下：

```
# hexdump /dev/input/event0
0000000 f755 67aa 9842 0009 0001 0073 0001 0000
0000010 f755 67aa 9842 0009 0000 0000 0000 0000
0000020 f755 67aa bf53 000c 0001 0073 0000 0000
0000030 f755 67aa bf53 000c 0000 0000 0000 0000
```

可以看到按下按键后，会有一些信息输出。

分析

第一行：0000000 f755 67aa 9842 0009 0001 0073 0001 0000

- 第一列是 event 事件序列号，0000000
- 第 2-5 列是时间戳，f755 67aa 9842 0009
- 第 6-7 列是事件类型，0001 0073，其中 0001 表示按键事件，0073 表示按键值
- 第 8 列是按键值，0001 表示按下按键

第二行是 0000010 f755 67aa 9842 0009 0000 0000 0000 0000

- 是同步信息，可以忽略

第三行是 0000020 f755 67aa bf53 000c 0001 0073 0000 0000

- 第一列是 event 事件序列号，0000020
- 第 2-5 列是时间戳，f755 67aa bf53 000c
- 第 6-7 列是事件类型，0001 0073，其中 0001 表示按键事件，0073 表示按键值
- 第 8 列是按键值，0000 表示松开按键

第四行是 0000030 f755 67aa bf53 000c 0000 0000 0000 0000

- 是同步信息，可以忽略

16.3 更换按键值

可以修改设备树的 ADC 节点从而修改该按键的按键值。

这是默认的设备树配置：

```
1     recovery-key {
2             compatible = "adc-keys";
3             io-channels = <&saradc 0>;
4             io-channel-names = "buttons";
```

(下页继续)

(续上页)

```
5               poll-interval = <100>;
6               keyup-threshold-microvolt = <1800000>;
7
8               key {
9                       label = "key_volumeup";
10                      linux,code = <KEY_VOLUMEUP>;
11                      press-threshold-microvolt = <0>;
12              };
13      };
14
15      adc-key {
16              compatible = "adc-keys";
17              io-channels = <&saradc 1>;
18              io-channel-names = "buttons";
19              poll-interval = <100>;
20              keyup-threshold-microvolt = <1800000>;
21
22              key {
23                      label = "key_volumedown";
24                      linux,code = <KEY_VOLUMEDOWN>;
25                      press-threshold-microvolt = <0>;
26              };
27      };
```

主要修改两个地方：

- `label = "key_volumeup";`：将 key_volumeup 修改成其他的按键名称
- `linux,code = <KEY_VOLUMEUP>;`：将 KEY_VOLUMEUP 修改成其他的按键值

按键值可以在 SDK 的该文件 `sysdrv/source/kernel/include/uapi/linux/input-event-codes.h` 翻阅。

第 17 章　4G 通信

17.1 4g 模块支持列表

理论上支持所有使用 usb 接口的带 rndis 协议的 4g 模块，这里列出用在 LubanCat-RV06 板卡上测试过的模块

17.1.1 ec20

4G 模块支持野火提供的 4G 无线网卡

可以支持两种接插模式

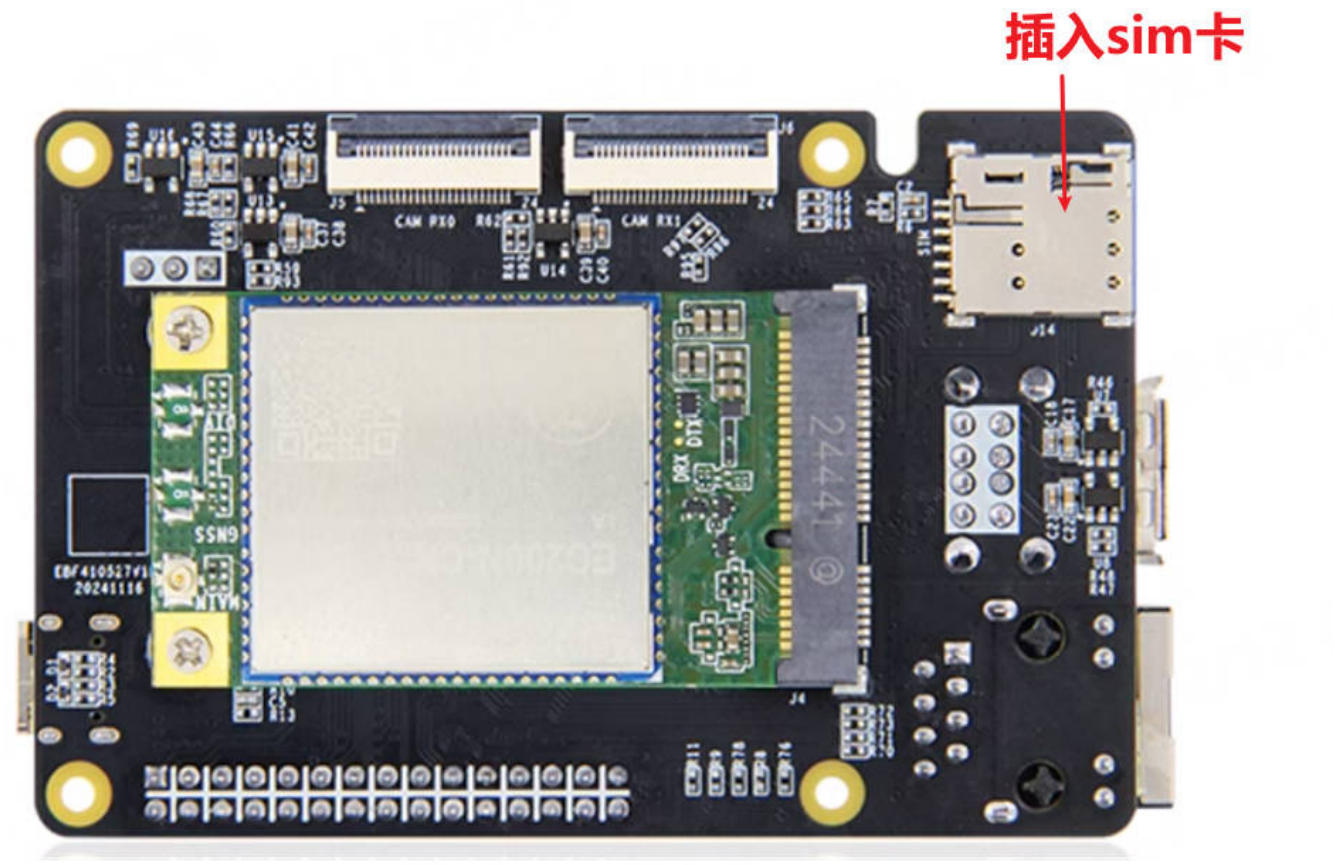
- MINI PCI-E 接口
- 搭配转接板使用 USB 接口

购买链接：《野火官方旗舰店》

17.2 模块安装

17.2.1 MINI PCI-E 接口

首先按图示的样式安装好 4G 模块



连接完成后，在转接板的背面会有 sim 卡卡槽，sim 卡使用的是最小号的 sim 卡

17.2.2 搭配转接板使用 USB 接口

首先按图示的样式安装好 4G 模块

连接完成后，在转接板的背面会有 sim 卡卡槽，sim 卡使用的是最大号的 sim 卡

然后就可以插到板卡的 usb-hub 上了

17.3 连接 4G 网络

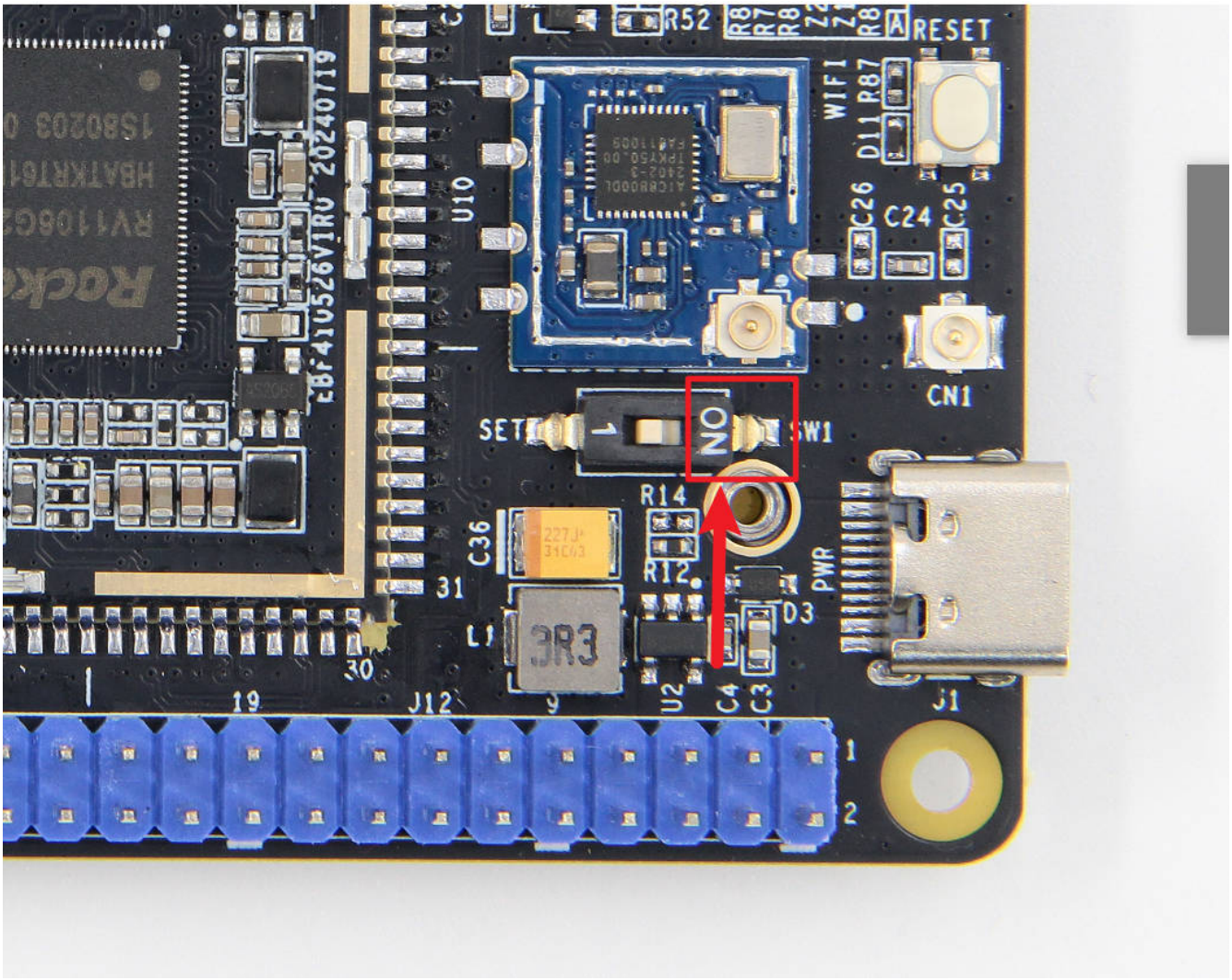
简单流程：

1. 使能接口
2. 连接模块，插入 sim 卡
3. 等待模块成功加载
4. 检查 sim 卡是否正常工作

5. 配置模块网卡模式 6.

17.3.1 使能 usb 接口

连接 4G 模块前首先需要把拨码开关拨到 on 的位置，使能 pcie 和 usb-hub



17.3.2 启动板卡

启动后会看到终端会打印下面的内容

```
[   14.427761] usb 1-1.1: new high-speed USB device number 4 using xhci-hcd
[   14.545579] usb 1-1.1: New USB device found, idVendor=2c7c,␣
↪idProduct=0125, bcdDevice= 3.18
[   14.545624] usb 1-1.1: New USB device strings: Mfr=1, Product=2,␣
↪SerialNumber=0
[   14.545643] usb 1-1.1: Product: Android
[   14.545660] usb 1-1.1: Manufacturer: Android
[   14.576175] rndis_host 1-1.1:1.0 usb0: register 'rndis_host' at usb-xhci-
↪hcd.1.auto-1.1, RNDIS device, 4e:19:2e:e8:f1:14
[   14.583542] option 1-1.1:1.2: GSM modem (1-port) converter detected
[   14.584197] usb 1-1.1: GSM modem (1-port) converter now attached to␣
↪ttyUSB0
[   14.586463] option 1-1.1:1.3: GSM modem (1-port) converter detected
[   14.588231] usb 1-1.1: GSM modem (1-port) converter now attached to␣
↪ttyUSB1
[   14.589712] option 1-1.1:1.4: GSM modem (1-port) converter detected
[   14.591508] usb 1-1.1: GSM modem (1-port) converter now attached to␣
↪ttyUSB2
[   14.594738] option 1-1.1:1.5: GSM modem (1-port) converter detected
[   14.595880] usb 1-1.1: GSM modem (1-port) converter now attached to␣
↪ttyUSB3
```

使用下面命令可以看 4G 模块有没有连接

```
ls /dev/ttyUSB*
```

如果有输出说明 4G 模块已经连接成功，如下所示

```
# ls /dev/ttyUSB*
```

(下页继续)

(续上页)

```
/dev/ttyUSB0  /dev/ttyUSB1  /dev/ttyUSB2  /dev/ttyUSB3
#
```

17.3.3 检查 sim 卡是否正常工作

```
# 在后台显示/dev/ttyUSB2 返回的信息 (如果想退出后台显示, 使用后面的命令: pkill cat)
cat /dev/ttyUSB2 &

# 检查 sim 卡的状态
echo -e "AT+CPIN?\r\n" >  /dev/ttyUSB2

# 如果返回值为如下即为获取 sim 卡成功
+CPIN: READY

OK

# 如果返回值为如下即为获取 sim 卡失败

+CME ERROR: 10

+CME ERROR: 13
```

注解： 如果遇到无法读卡的问题，可能是卡的触电没有紧密与模块的卡座亲密接触，也可能是模块不支持该品类的卡，最佳的解决方法是联系模块的提供商，让他们给你的问题解决

17.3.4 配置模块网卡模式

注解： 如果之前成功配置过网卡的模式，网卡是会把成功配置的网卡模式保存，就不需要额外再配置网卡模式了（使用网络可以跳过此步）

移远模块共有 4 种模式，但不是每种模块都支持 4 种模式，需根据实际模块确定。

- 0:rmnet 模式：通过 QMI 工具发的 QMI 命令，获取公网 IP。这种模式可以配合 usb_ecm 驱动或高通 GobiNet 驱动使用。
- 1:ecm 模式：通过标准的 CDC-ECM 发起 data call，是发送标准的 ECM 命令，获取局域网 ip。这种模式配合 cdc_ether 驱动使用
- 2:mbim 模式：Mobile Broadband Interface Model，正宗的移动宽带接口模型，专门用于 3G/4G/5G 模块的，只在 win8 以上的 windows 上使用。一般只在 WINDOWS 下使用。
- 3:rndis 模式：基于 USB 实现 RNDIS 实际上就是 TCP/IP over USB，就是在 USB 设备上跑 TCP/IP，让 USB 设备看上去像一块网卡获取局域网 ip。这种方式最简单，模块插上手机卡之后，模块会自动拨号上网。

表 1: AT 命令列表

命令	功能
AT+QCFG=” usbnet”	查询网卡模式
AT+QCFG=”usbnet”,1	设置网卡为 ECM 模式
AT+QCFG=”usbnet”,3	设置网卡为 RNDIS 模式

17.3.5 获取当前网卡模式

由于模块是使用 AT 指令操作的。所以可以使用 minicom 来进行操作，这里为了简化操作，没使用 minicom，而是直接在命令行上操作

端口位置：/dev/ttyUSB2

```
# 查询当前模式

# 在后台显示/dev/ttyUSB2 返回的信息 (如果想退出后台显示, 使用后面的命令: pkill cat)
cat /dev/ttyUSB2 &

# 查询当前网卡的模式
echo -e "AT+QCFG=\"usbnet\"\r\n" >  /dev/ttyUSB2

# 例子:
# cat /dev/ttyUSB2 &
[1] 777
# echo -e "AT+QCFG=\"usbnet\"\r\n" >  /dev/ttyUSB2
# AT+QCFG="usbnet"

+QCFG: "usbnet",0

OK
```

可以看到返回了 0，这个是网卡在 rmnet 模式

- +QCFG: "usbnet",3：RNDIS 模式
- +QCFG: "usbnet",2：MBIM 模式
- +QCFG: "usbnet",1：ECM 模式
- +QCFG: "usbnet",0：rmnet 模式

17.3.5.1 配置模块网卡模式为 ECM

```
# 在后台显示/dev/ttyUSB2 返回的信息 (如果想退出后台显示, 使用后面的命令: pkill cat)
cat /dev/ttyUSB2 &

```

(下页继续)

(续上页)

```
# 配置为 ECM 模式 (返回: OK 代表配置成功)
echo -e "AT+QCFG=\"usbnet\",1\r\n" >  /dev/ttyUSB2
```

- 返回 OK 代表配置成功
- 如果返回 ERROR，可能是模块不支持 ECM 模式，可以尝试 RNDIS 模式
- 如果 4G 模块没有重新启动，可以执行下面命令重启模块

```
# 重启模块 (重启模块才能生效)
echo -e "AT+CFUN=1,1\r\n" >  /dev/ttyUSB2
```

17.3.5.2 配置模块网卡模式为 RNDIS

```
# 在后台显示/dev/ttyUSB2 返回的信息 (如果想退出后台显示, 使用后面的命令: pkill cat)
cat /dev/ttyUSB2 &

# 配置为 RNDIS 模式 (返回: OK 代表配置成功)
echo -e "AT+QCFG=\"usbnet\",3\r\n" >  /dev/ttyUSB2
```

- 返回 OK 代表配置成功
- 如果返回 ERROR，可以重新上电或者拔掉 usb 重新插入
- 如果 4G 模块没有重新启动，可以执行下面命令重启模块

```
# 重启模块 (重启模块才能生效)
echo -e "AT+CFUN=1,1\r\n" >  /dev/ttyUSB2
```

17.3.6 连接网络

ec20 会自动拨号

完成上面步骤后，LubanCat-RV06 板卡会生成一个 4G 网卡的节点，可以使用 ifconfig 查看

```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 2E:CA:C3:12:43:6A
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:56

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:48 errors:0 dropped:0 overruns:0 frame:0
          TX packets:48 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3552 (3.4 KiB)  TX bytes:3552 (3.4 KiB)

usb0      Link encap:Ethernet  HWaddr B6:87:81:B0:0A:C0
          inet addr:192.168.137.100  Bcast:192.168.137.255  Mask:255.255.255.0
          inet6 addr: 2409:8954:d913:46d7:1d5a:a647:ffb0:d8c7/64 Scope:Global
          inet6 addr: fe80::c553:dcb7:15bc:b437/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:31 errors:0 dropped:0 overruns:0 frame:0
          TX packets:44 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2745 (2.6 KiB)  TX bytes:6180 (6.0 KiB)
```

(下页继续)

(续上页)

```
28
29 wlan0     Link encap:Ethernet  HWaddr 38:7A:CC:65:78:20
30          UP BROADCAST MULTICAST  MTU:1500  Metric:1
31          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
32          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
33          collisions:0 txqueuelen:1000
34          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
35
36 #
```

- 可以看到 usb0 是 4G 模块的网卡节点，他的 ip 是 192.168.137.100，

  这个是系统设置 usb-gadget 的 rndis 功能的静态 ip，此时是不能正常上网的，需要按照下面操作设置成动态 ip

17.3.6.1 设置 usb0 为动态 ip

```
1 # 修改文件
2 vi /etc/dhcpcd.conf
```

注释文件末尾对 usb0 的配置

```
1 interface usb0
2 static ip_address=192.168.137.100/24
3 static routers=192.168.137.1
4 static domain_name_servers=8.8.8.8 8.8.4.4
```

改成

```
1 # interface usb0
2 # static ip_address=192.168.137.100/24
3 # static routers=192.168.137.1
```

(下页继续)

(续上页)

```
# static domain_name_servers=8.8.8.8 8.8.4.4
```

- 然后保存文件

使用下面命令应用网络配置，对 usb0 进行动态 ip 配置

```
dhcpcd -n usb0
```

执行成功后，查看 usb0 的 ip 地址

```
# ifconfig usb0
usb0      Link encap:Ethernet  HWaddr B6:87:81:B0:0A:C0
          inet addr:192.168.225.59  Bcast:192.168.225.255  Mask:255.255.255.0
          inet6 addr: 2409:8954:d913:46d7:1d5a:a647:ffb0:d8c7/64 Scope:Global
          inet6 addr: fe80::c553:dcb7:15bc:b437/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:152 errors:0 dropped:0 overruns:0 frame:0
          TX packets:179 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:11702 (11.4 KiB)  TX bytes:23290 (22.7 KiB)
```

- 然后可以 ping mi.com 试试，有些物联卡会限制网站的访问，所以可能 ping 不通，可以 ping 其他网站试试

```
ping mi.com
```

成功如下所示

```
# ping mi.com
PING mi.com (111.13.141.215): 56 data bytes
64 bytes from 111.13.141.215: seq=0 ttl=48 time=79.015 ms
64 bytes from 111.13.141.215: seq=1 ttl=48 time=66.884 ms
64 bytes from 111.13.141.215: seq=2 ttl=48 time=78.165 ms
```

(下页继续)

(续上页)

```
64 bytes from 111.13.141.215: seq=3 ttl=48 time=77.889 ms
64 bytes from 111.13.141.215: seq=4 ttl=48 time=63.532 ms
```

如果你无法 ping 通网络，出现以下情况

```
root@lubancat:~# ping mi.com
ping: unknown host
```

可能你是接了网线的（假设使用了 eth0 接口），而默认路由表却不是 4G 模块，如果想使用网卡上网，则需要更新路由表，解决办法如下：

```
route del -net 0.0.0.0 eth0
route add -net 0.0.0.0 usb0
```

第 18 章　音频

LubanCat-RV06 板卡板载了两个麦克风输入接口，一个 3W 的扬声器输出

这三个接口均使用 YTC-A1251-02AB (1.25mm 1x2P 直插)

- MIC1: 左声道差分输入
- MIC2: 右声道差分输入
- SPK: 左声道声音输出

由于麦克风输入采用的是差分输入，所以在使用时需要注意：- 麦克风输入的信号线必须成对使用，即左声道信号线必须和左声道地线成对使用，右声道信号线必须和右声道地线成对使用 - 麦克风输入的信号线必须和板卡上的接口一一对应，即左声道信号线必须插在 MIC1 接口，右声道信号线必须插在 MIC2 接口

由于扬声器输出采用的是单声道输出，所以在使用时需要注意：- 扬声器输出信号线必须插在 SPK 接口

18.1 麦克风使能

系统启动后，默认只有 MIC1 麦克风输入接口是打开的，如果需要录制右声道的声音，需要手动打开 MIC2 麦克风输入接口

```
# 打开 MIC2 麦克风输入接口
amixer cset numid=19 5
# 或者
amixer sset 'ADC Mode' DiffadcLR
```

18.2 声音录制

18.2.1 获取录音设备

```
# 获取录音设备
arecord -l
```

运行结果：

```
# arecord -l
**** List of CAPTURE Hardware Devices ****
card 0: rv1106acodec [rv1106-acodec], device 0: ffae0000.i2s-rv1106-hifi␣
↪ff480000.acodec-0 [ffae0000.i2s-rv1106-hifi ff480000.acodec-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
#
```

- card0 就是录音设备

18.2.2 录制声音

使用命令 `arecord` 可以录制声音，录制声音的命令格式如下：

```
arecord -f cd -Dhw:0 -d 10 test.wav
```

- `-f cd`: 采样格式为 CD 音质
- `-Dhw:0`: 使用 card0 设备录音
- `-d 10`: 录制 10 秒
- `test.wav`: 录音文件名

录制 10 秒后就会自动停止，录制的声音保存在 test.wav 文件中，可以使用 ctrl + c 终止录音，终止后会自动保存录音文件中。

18.3 声音播放

18.3.1 获取播放设备

```
# 获取播放设备
aplay -l
```

运行结果：

```
# aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: rv1106acodec [rv1106-acodec], device 0: ffae0000.i2s-rv1106-hifi␣
↪ff480000.acodec-0 [ffae0000.i2s-rv1106-hifi ff480000.acodec-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
#
```

18.3.2 播放声音

使用命令 aplay 可以播放声音，播放声音的命令格式如下：

```
aplay -Dhw:0 test.wav
```

- -Dhw:0: 使用 card0 设备播放
- test.wav: 播放文件名

18.4 声卡控制

18.4.1 命令行配置

命令行配置主要使用 amixer 工具，该工具可以用来配置声卡的各种参数

18.4.1.1 查看所有控制器

```
# 查看所有控制器
amixer controls
```

结果如下：

```
# amixer controls
numid=4,iface=MIXER,name='ADC ALC Left Volume'
numid=5,iface=MIXER,name='ADC ALC Right Volume'
numid=6,iface=MIXER,name='ADC Digital Left Volume'
numid=7,iface=MIXER,name='ADC Digital Right Volume'
numid=8,iface=MIXER,name='ADC HPF Cut-off'
numid=2,iface=MIXER,name='ADC MIC Left Gain'
numid=22,iface=MIXER,name='ADC MIC Left Switch'
numid=3,iface=MIXER,name='ADC MIC Right Gain'
numid=23,iface=MIXER,name='ADC MIC Right Switch'
numid=20,iface=MIXER,name='ADC MICBIAS Voltage'
numid=21,iface=MIXER,name='ADC Main MICBIAS'
numid=19,iface=MIXER,name='ADC Mode'
numid=1,iface=MIXER,name='I2STDM Digital Loopback Mode'
numid=17,iface=MIXER,name='AGC Left Approximate Sample Rate'
numid=18,iface=MIXER,name='AGC Right Approximate Sample Rate'
numid=11,iface=MIXER,name='ALC AGC Left Max Volume'
numid=13,iface=MIXER,name='ALC AGC Left Min Volume'
```

(下页继续)

(续上页)

```
numid=15,iface=MIXER,name='ALC AGC Left Switch'
numid=9,iface=MIXER,name='ALC AGC Left Volume'
numid=12,iface=MIXER,name='ALC AGC Right Max Volume'
numid=14,iface=MIXER,name='ALC AGC Right Min Volume'
numid=16,iface=MIXER,name='ALC AGC Right Switch'
numid=10,iface=MIXER,name='ALC AGC Right Volume'
numid=26,iface=MIXER,name='DAC Control Manually'
numid=25,iface=MIXER,name='DAC HPMIX Volume'
numid=24,iface=MIXER,name='DAC LINEOUT Volume'
numid=27,iface=MIXER,name='DAC VCM Manually'
#
```

- `numid`: 参数的编号
- `iface`: 参数的类型
- `name`: 参数的名称

比较重要的参数有：

- `ADC MIC Left Gain`: MIC1 麦克风输入增益
- `ADC MIC Right Gain`: MIC2 麦克风输入增益
- `ADC MIC Left Switch`: MIC1 麦克风输入开关
- `ADC MIC Right Switch`: MIC2 麦克风输入开关
- `ADC Mode`: 麦克风输入模式
- `DAC LINEOUT Volume`: LINEOUT 输出音量
- `DAC Control Manually`: DAC 输出控制模式

18.4.1.2 读取控制器的值

```
# 读取控制器的值
amixer cget numid=19
```

结果如下：

```
# amixer cget numid=19
numid=19,iface=MIXER,name='ADC Mode'
  ; type=ENUMERATED,access=rw------,values=1,items=6
  ; Item #0 'DiffadcL'
  ; Item #1 'SingadcL'
  ; Item #2 'DiffadcR'
  ; Item #3 'SingadcR'
  ; Item #4 'SingadcLR'
  ; Item #5 'DiffadcLR'
  : values=5
#
```

可以看到 ADC Mode 的值为 DiffadcLR，即差分输入模式。

18.4.1.3 设置控制器的值

```
# 设置控制器的值
amixer cset numid=19 5
```

设置 ADC Mode 的值为 DiffadcLR。

运行结果：

```
# amixer cget numid=19
numid=19,iface=MIXER,name='ADC Mode'
  ; type=ENUMERATED,access=rw------,values=1,items=6
```

(下页继续)

(续上页)

```
  ; Item #0 'DiffadcL'
  ; Item #1 'SingadcL'
  ; Item #2 'DiffadcR'
  ; Item #3 'SingadcR'
  ; Item #4 'SingadcLR'
  ; Item #5 'DiffadcLR'
  : values=5
#
```

18.4.1.4 查看所有控制器及其值

```
# 查看所有控制器及其值
amixer contents
```

结果很长就不展开了

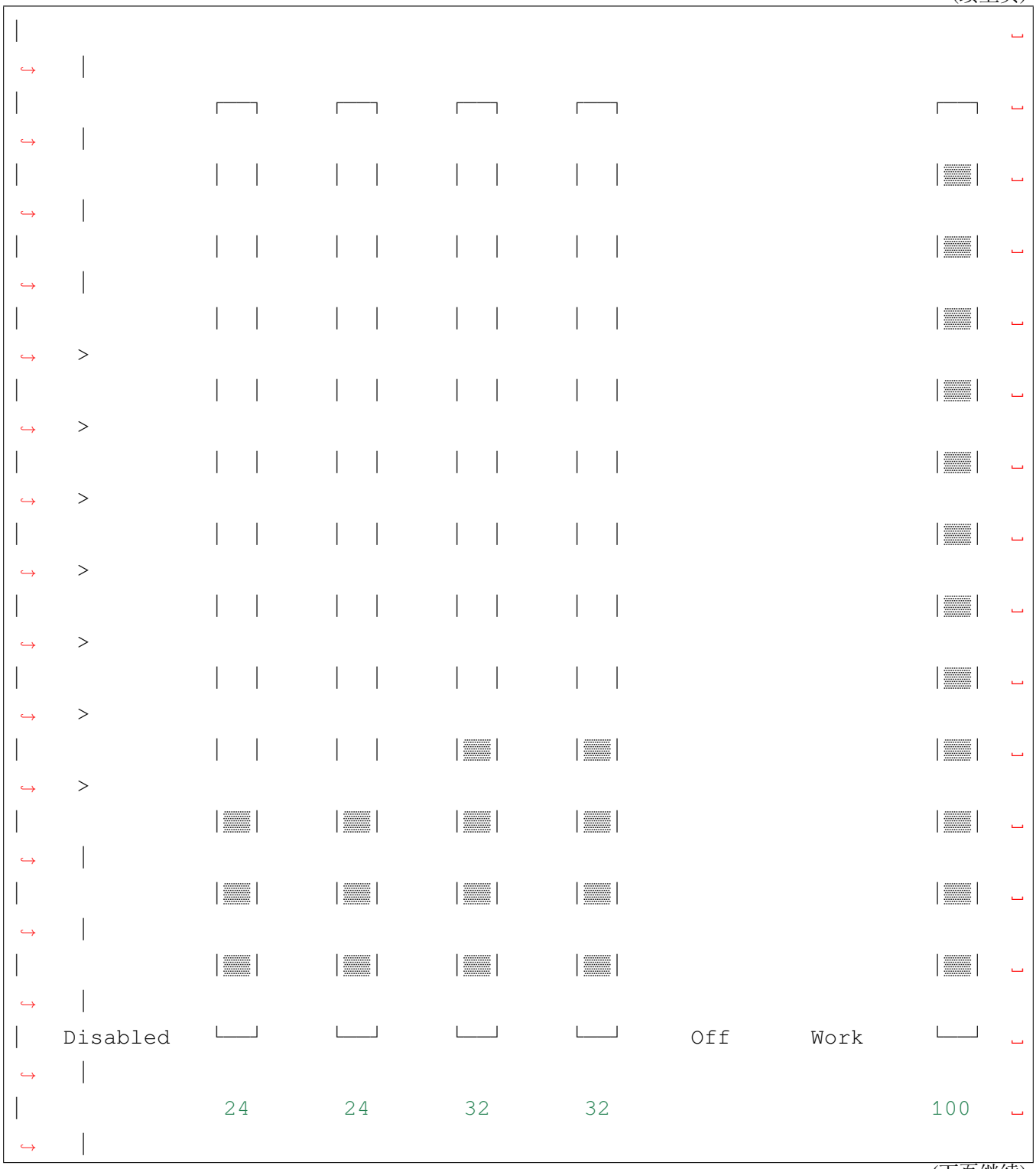
18.4.2 图形化配置

图形化配置使用命令 alsa_mixer，该命令会打开一个图形化界面，可以方便的配置声卡的各种参数

```
┌──────────────────────────────── AlsaMixer v1.2.11 ──────────────────────────────────┐
│ Card: rv1106-acodec                                  F1:  Help                      │
│ Chip:                                                F2:  System information        │
│ View: F3:[Playback] F4: Capture  F5: All             F6:  Select sound card         │
│ Item: I2STDM Digital Loopback Mode [Disabled]        Esc: Exit                      │
└─────────────────────────────────────────────────────────────────────────────────────┘
```

(下页继续)

(续上页)



(下页继续)

(续上页)

```
22 |  <I2STDM D>ADC ALC  ADC ALC  ADC Digi ADC Digi ADC HPF  ADC MIC  ADC MIC ␣
   ↪    |
23 |                                                                        ␣
   ↪    |
24 └──────────────────────────────────────────────────────────────────────────┘
```

- 可以自行探索，这里不过多赘述

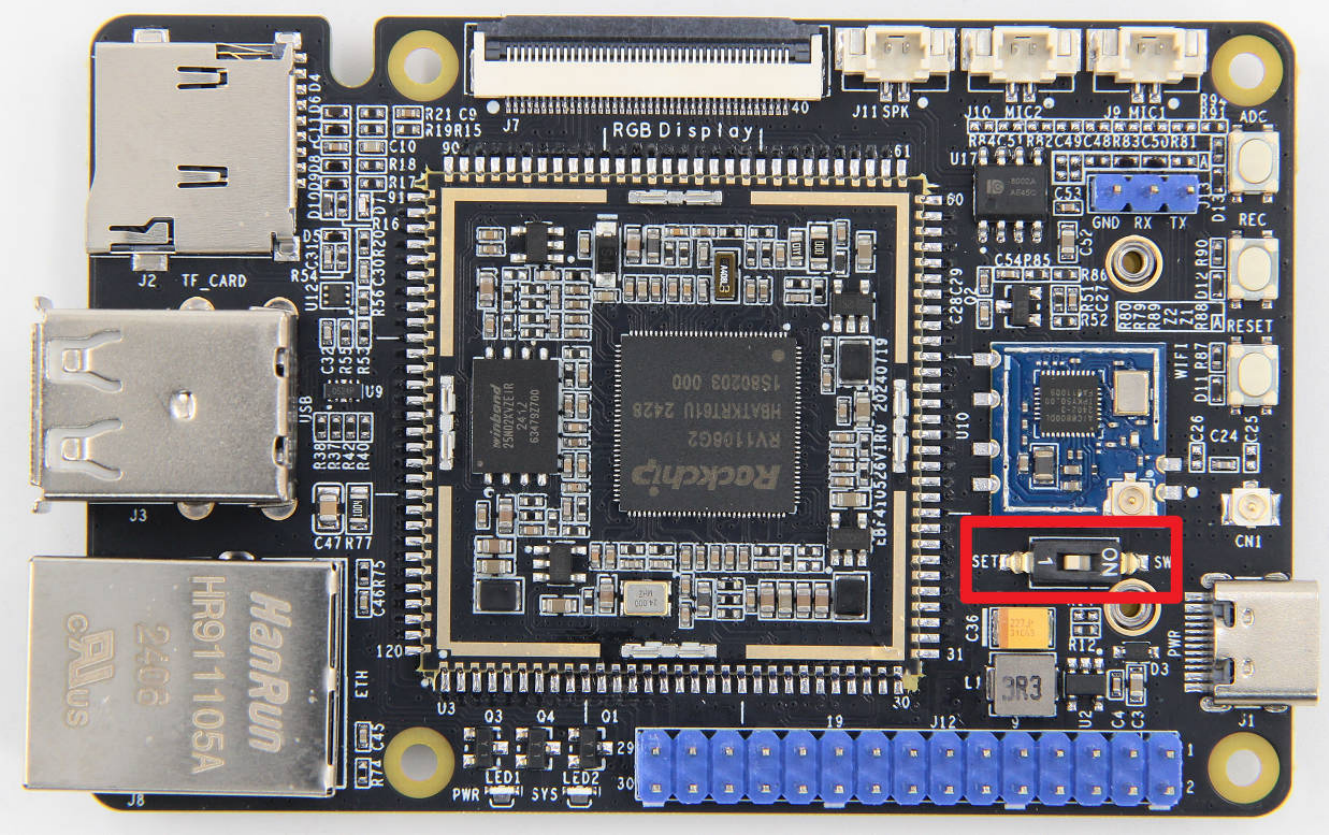
第 19 章　USB

RV1106 这颗 SOC 只集成了一个 USB2.0 接口，支持 USB Host 和 USB Device 两种模式。

LubanCat-RV06 板卡上有两种类型的 usb 信号链路

- USB Type-C 接口。
- 两个 USB Type-A 接口，USB WIFI，mini-pcie 接口。

这两个不能同时用，通过一个拨码开关进行选择，如下图所示



- 当拨码开关拨到 on 时，Type-C 接口可以使用，此时 Type-C 接口可以由软件定义为 USB Host 或 USB Device。

- 当拨码开关拨到 1 时，Type-A 接口，WIFI,mini-pcie 接口均可使用，只能使用为 USB Host。

19.1 USB Host

19.1.1 配置方式

配置文件路径：/etc/usb_config

配置方法，修改配置文件中的 OTG_MODE 设置成 host 即可。

如下所示：

```
# OTG_MODE:host or peripheral
# OTG_MODE=peripheral
OTG_MODE=host

# GADGET_CONFIG:usb_mtp_en usb_adb_en usb_ums_en usb_ntb_en
# usb_acm_en usb_uac1_en usb_uac2_en usb_uvc_en usb_rndis_en usb_hid_en
GADGET_CONFIG="usb_rndis_en"
```

- 重启或者执行下面命令生效

```
reboot
# 或
/etc/init.d/S50usbdevice start
```

19.2 USB Device

配置文件路径：/etc/usb_config

配置方法，修改配置文件中的 OTG_MODE 设置成 peripheral 即可。

如下所示：

```
# OTG_MODE:host or peripheral
OTG_MODE=peripheral
# OTG_MODE=host

# GADGET_CONFIG:usb_mtp_en usb_adb_en usb_ums_en usb_ntb_en
# usb_acm_en usb_uac1_en usb_uac2_en usb_uvc_en usb_rndis_en usb_hid_en
GADGET_CONFIG="usb_rndis_en"
```

- 重启或者执行下面命令生效

```
reboot
# 或
/etc/init.d/S50usbdevice start
```

19.2.1 gadget 功能

LubanCat-RV06 板卡支持以下 gadget 功能：

- USB MTP
- USB ADB
- USB UMS
- USB NTB
- USB ACM
- USB UAC1
- USB UAC2
- USB UVC
- USB RNDIS
- USB HID

设置方法需要修改配置文件 /etc/usb_config 中的 GADGET_CONFIG 变量。

有些功能可能使用不正常，需要自行去修改/etc/init.d/S50usbdevice 文件进行配置。

- 重启或者执行下面命令生效

```
reboot
# 或
/etc/init.d/S50usbdevice start
```

第 20 章　环境搭建

RV1106 的镜像构建依赖于 x64 架构的 Linux 环境，理论上可以使用以下环境编译 RV1106 的镜像：

- 运行 Linux 系统的服务器或台式机
- 运行 Linux 系统的虚拟机
- WSL2/WSL
- Docker
- 其他可以运行 Linux 系统的环境

警告： 以上的环境都需要是 x64 架构的 Linux 系统，不支持 ARM 架构的 Linux 系统。

注解： 编译时产生的中间文件较多，建议使用具有较大磁盘空间的环境，最好预留 20G 硬盘空间。

20.1 虚拟机搭建 (从零开始)

本小节会使用 Ubuntu20.04 系统的虚拟机为例，介绍如何从零开始搭建 RV1106 镜像编译环境。

20.1.1 基础虚拟机安装

使用虚拟机搭建编译环境，可以参考以下步骤：

1. 安装虚拟机软件，例如 VMware、VirtualBox 等。
2. 创建一个 Ubuntu20.04 的虚拟机，并安装 Ubuntu20.04 操作系统。

20.1.2 依赖软件安装

```
sudo apt-get install repo git ssh make gcc \
gcc-multilib g++-multilib module-assistant \
expect g++ gawk texinfo libssl-dev \
bison flex fakeroot cmake unzip gperf autoconf \
device-tree-compiler libncurses5-dev
```

注解： 以上命令会安装编译 RV1106 镜像所需的依赖软件。

安装完成即可进行下一步操作。

第 21 章 SDK

21.1 SDK 获取

SDK 可以通过 github 或者百度网盘获取

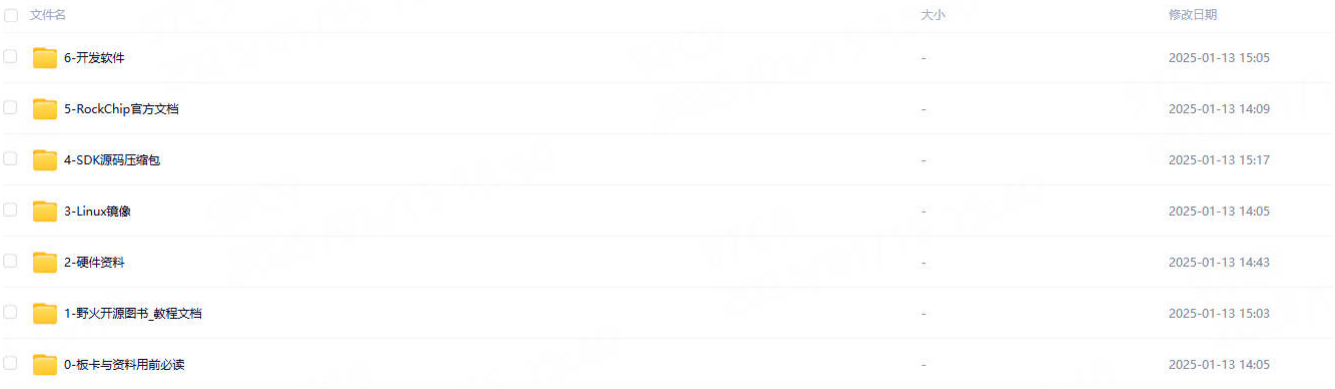
21.1.1 github

SDK github 地址: `https://github.com/LubanCat/RV06_03_Linux_SDK`

```
# SDK 下载
git clone https://github.com/LubanCat/RV06_03_Linux_SDK.git --depth=1
```

21.1.2 百度网盘

链接: https://pan.baidu.com/s/1i0I6K_MLiA9zW4_w_VNLdg?pwd=teyf 提取码: teyf



- `4-SDK 源码压缩包`: RV06 的 SDK 源码压缩包，版本会更新的慢，建议使用 github 下载

解压步骤:

1. 先将 SDK 源码压缩包放到 linux 系统上
2. 创建一个空的文件夹
3. 将源码包解压到这个文件夹
4. 执行命令恢复所有文件

```
# 创建新文件夹
mkdir LubanCat_RV1106_RV1103_Linux_SDK

# 将源码包放到该文件夹的同一个目录

# 解压源码包
tar -xf LubanCat_RV1106_RV1103_Linux_SDK_20250212.tgz -C LubanCat_RV1106_
↪RV1103_Linux_SDK

# 解压完后切换文件夹
cd LubanCat_RV1106_RV1103_Linux_SDK

# 恢复所有文件
git config --global --add safe.directory ./
git reset --hard

# 成功
```

21.2 SDK 解析

```
.
├── build.sh -> project/build.sh
├── docs
├── media
├── project
```

(下页继续)

(续上页)

```
├── readme_cn.txt -> project/readme_cn.txt
├── readme_en.txt -> project/readme_en.txt
├── sysdrv
└── tools
```

表 1: sdk

文件	文件描述
build.sh	多功能编译脚本，用于编译 SDK
docs	SDK 文档目录。包含 SDK 的使用说明、API 文档等。里面有很多详细的 SDK 用法，可以参考使用里面的文档对系统进行深度定制
media	存放 RV1106 的多媒体库，视频编码，isp，rga 等
project	存放用户的程序以及板子的配置文件
sysdrv	存放 uboot，kernel，buildroot,kernel-driver 以及一些编译工具等
tools	存放交叉编译工具以及在 Windows 和 Linux 的一些工具

21.2.1 docs

```
docs/
├── Copyright_Statement.md
├── en
│   ├── audio
│   ├── bsp
```

(下页继续)

(续上页)

```
6  |    ├── ipc
7  |    ├── isp
8  |    ├── media
9  |    ├── Rockchip_Quick_Start_Linux_IPC_SDK_EN.pdf -> ipc/Rockchip_Quick_
   ↪Start_Linux_IPC_SDK_EN.pdf
10 |    └── security
11 └── zh
12     ├── audio
13     ├── bsp
14     ├── ipc
15     ├── isp
16     ├── iva
17     ├── media
18     ├── Rockchip_Quick_Start_Linux_IPC_SDK_CN.pdf -> ipc/Rockchip_Quick_
   ↪Start_Linux_IPC_SDK_CN.pdf
19     ├── Rockchip_User_Guide_Bug_System_CN.pdf
20     ├── Rockchip_User_Guide_SDK_Application_And_Synchronization_CN.pdf
21     └── security
```

表 2: dosc 解析

文档	文档描述
audio	音频相关的文档
bsp	板级支持包相关的文档，包含，GPIO，I2C，SPI，GMAC，PWM，存储等外设的使用文档
ipc	SDK 的一些快速使用手册
isp	图像处理相关的文档
iva	智能视频分析相关的文档
media	多媒体相关的文档
Security	安全相关的文档

注解： zh 是中文文档，en 是英文文档，中文文档更新

21.2.2 media

```
media
├── alsa-lib
├── avs
├── cfg
├── common_algorithm
├── isp
├── iva
├── ive
├── libdrm
├── libv4l
├── Makefile
├── Makefile.param
├── mpp
├── out
├── readme_cn.txt
├── readme_en.txt
├── rga
├── rkpostisp
├── rockit
├── samples
├── security
└── sysutils
```

表 3: media 解析

媒体库名	功能
cfg	配置模块是否编译
alsa-lib	Advanced Linux Sound Architecture (ALSA) library
avs	全景拼接（只支持 RK3588）
common_algorithm	音频 3A 算法、移动检测、遮挡检测
isp	isp 图像处理算法
iva	智能视频分析算法（只支持 RV1106/RV1103/RK3588）
ive	智能视频分析硬件加速引擎（只支持 RV1106/RV1103）
libdrm	Direct Rendering Manager
libv4l	video4linux2 设备用🄵层接口
mali	GPU firmware 以及库文件（注：只支持 RK3588，mali_csffw.bin 必须放在/lib/firmware 目录）
mpp	编解码接口，给 rkmedia 和 rockit 调用，不建议直接调用 mpp
rga	RGA 是一个独立的 2D 硬件加速器
rkmedia	多媒体接口（适用 RV1126/RV1109 平台）
rockit	多媒体接口（推荐）
sysutils	外设参考接口（ADC/GPIO/TIME/WATCHDOG）
samples	测试例程

21.2.3 project

```
project/
├── app
├── build.sh
├── cfg
├── cfg-all-items-introduction.txt
├── make_meta
├── readme_cn.txt
├── readme_en.txt
└── scripts
```

表 4: project 解析

文件	文件描述
app	用户的程序存放目录
build.sh	根目录下的编译脚本
cfg	配置文件存放目录，里面存放了板子的配置文件，用户可以根据自己的需求进行修改
make_meta	摄像头相关的文件，一般用不上
scripts	编译相关的一些脚本文件

21.2.4 sysdrv

```
sysdrv/
├── cfg
├── drv_ko
├── Makefile
├── Makefile.param
├── readme_cn.txt
```

(下页继续)

(续上页)

```
├── readme_en.txt
├── source
└── tools
```

表 5: sysdrv 解析

文件	文件描述
cfg	配置文件存放目录，里面存放了 uboot，kernel，buildroot，以及软件包的配置
drv_ko	部分 kernel 驱动源码存放目录
source	uboot，kernel，buildroot 源码存放目录
tools	存放板卡的一些工具还有系统构建相关的工具

21.2.5 tools

```
tools/
├── linux
│   ├── Linux_Pack_Firmware
│   ├── Linux_Upgrade_Tool
│   ├── SocToolKit
│   ├── toolchain
│   └── ToolsRelease.txt
└── windows
    ├── DriverAssitant_v5.12.zip
    ├── FactoryTool_v1.73.1.7z
    ├── Rockchip_AVS_tool
    ├── SocToolKit
```

(下页继续)

(续上页)

```
13      └── ToolsRelease.txt
```

表 6: tools 解析

文件	文件描述
toolchain	交叉编译工具链
Linux_Pack_Firmware	Linux 下的固件打包工具
Linux_Upgrade_Tool	Linux 下的升级工具
SocToolKit	固件烧录工具
DriverAssitant_v5.12.zip	Linux 下的升级工具
Linux_Upgrade_Tool	windows 驱动
FactoryTool_v1.73.1.7z	量产升级工具

第 22 章　SDK 编译

22.1 安装交叉编译工具

```
# 切换文件夹
cd tools/linux/toolchain/arm-rockchip830-linux-uclibcgnueabihf/
# 安装交叉编译工具
source env_install_toolchain.sh
```

安装完成后，执行下面命令查看是否安装成功

```
# 检测是否安装成功
arm-rockchip830-linux-uclibcgnueabihf-gcc --version

# 如果出现下面内容即工具链安装成功
root@dev135:~/rv1106/rv1106_ipc_linux# arm-rockchip830-linux-
↪uclibcgnueabihf-gcc --version
arm-rockchip830-linux-uclibcgnueabihf-gcc (crosstool-NG 1.24.0) 8.3.0
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

22.2 板卡配置

- project/cfg/ 目录下存放了板子的配置文件，每个板子一个配置文件，文件前面带了 . 的则是隐藏文件，不适应于野火的配置文件
- project/cfg-all-items-introduction.txt 文件是板子配置文件的说明文档，里面会有详细的配置介绍

用户可以使用下面命令选择要编译的板卡

```
./build.sh lunch
```

运行结果如下所示

```
root@dev135:~/rv1106/rv1106_ipc_linux# ./build.sh lunch
ls: cannot access 'BoardConfig*.mk': No such file or directory

You're building on Linux
Lunch menu...pick a combo:

BoardConfig-*.mk naming rules:
BoardConfig-" 启动介质"-" 电源方案"-" 硬件版本"-" 应用场景".mk
BoardConfig-"boot medium"-"power solution"-"hardware version"-"application".
↪mk

----------------------------------------------------------------
0. BoardConfig_IPC/BoardConfig-SD_CARD-NONE-RV1106_LubanCat-RV06.mk
                            boot medium(启动介质): SD_CARD
                        power solution(电源方案): NONE
                        hardware version(硬件版本): RV1106_LubanCat
                            application(应用场景): RV06
----------------------------------------------------------------

----------------------------------------------------------------
1. BoardConfig_IPC/BoardConfig-SPI_NAND-NONE-RV1106_LubanCat-RV06.mk
                            boot medium(启动介质): SPI_NAND
                        power solution(电源方案): NONE
                        hardware version(硬件版本): RV1106_LubanCat
                            application(应用场景): RV06
----------------------------------------------------------------

Which would you like? [0]:
```

- 输入你想要构建的板卡号，然后回车即可

22.3 编译

22.3.1 一键自动编译

一键自动编译包含了后面的内容

```
./build.sh lunch     # 选择参考板级
./build.sh           # 一键自动编译
```

22.3.2 编译 U-Boot

```
./build.sh clean uboot
./build.sh uboot
```

- 生成镜像文件：output/image/download.bin、output/image/idblock.img 和 output/image/uboot.img

22.3.3 编译 kernel

```
./build.sh clean kernel
./build.sh kernel
```

- 生成镜像文件：output/image/boot.img

22.3.4 编译 rootfs

```
./build.sh clean rootfs
./build.sh rootfs
```

编译后使用./build.sh firmware 命令打包成 rootfs.img 生成镜像文件：output/image/rootfs.img

22.3.5 编译 media

```
./build.sh clean media
./build.sh media
```

生成文件的存放目录：output/out/media_out

22.3.6 编译 app

```
./build.sh clean app
./build.sh app
```

生成文件的存放目录：output/out/app_out

注解： app 依赖 media

22.3.7 编译内核驱动

```
./build.sh clean driver
./build.sh driver
```

生成文件的存放目录：output/out/sysdrv_out/kernel_drv_ko/

22.3.8 打包 env.img

```
./build.sh env
```

生成文件的存放目录：output/image/env.img

22.3.9 固件打包

```
./build.sh firmware
```

生成文件的存放目录：output/image

22.4 编译产物

编译出来的镜像主要有以下的内容

表 1: 编译产物

固件镜像名称	说明
download.bin	烧录工具升级通讯的设备端程序，只会下载到板子内存
env.img	包含分区表和启动参数（SDK 默认的 env 分区放在 0 地址）
idblock.img	loader 镜像（包含 DDR 初始化），负责加载 U-Boot
uboot.img	uboot 镜像
boot.img	Linux 内核镜像
rootfs.img	根文件系统镜像
oem.img	oem 镜像（可选）
userdata.img	userdata 镜像（可选）

版权说明

野火电子保留本项目的所有版权。