



EB-RV1126-BC-191整板快速入门手册

版本	日期	原因
V2.0	2022/02/22	创建文档
V2.1	2022/05/23	增加了扩展子板的功能验证
V2.2	2022/11/16	升级适配V2. 2. 5版本SDK

目录

1 Vmare 下载及安装	3
1.1 Vmare 下载	3
1.2 Vmare 安装	3
2 Ubuntu 18.04 镜像下载及安装	4
2.1 Ubuntu 18.04 镜像下载	4
2.2 Ubuntu 18.04 镜像安装	5
3 SDK 下载及工程目录说明	7
3.1 SDK 的下载	7
3.2 工程目录说明	7
4 Ubuntu18.04里解压下载的sdk	7
4.1 拷贝下载的sdk	7
4.2 解压复制的sdk	9
5 SDK 编译说明	10
5.1 总体编译	13
5.2 总体编译生成的固件位置	13
5.3 分步编译	14
6 固件烧写	16
6.1 硬件准备	16
6.2 软件准备	17
6.3 固件烧写	18
6.4 adb的安装	22
6.4烧写固件注意事项	23
7 远程连接（adb,telnet 或串口、波特率等信息）	24
7.1 ssh 连接（示例为 windows 下操作）	24
7.2 scp 连接（示例为linux 下操作）	26
7.3.通过网络 ADB 调试	26
7.4调试串口	26
8 运行取频流的可执行文件	27
8.1 编译sdk取视频流的可执行文件	27
8.2 可执行程序的路径	27
8.3 Ubuntu NFS 服务器设置 启动NFS服务器	28
8.4 可执行文件传给板端	33
8.5运行可执行文件取流	35
9 适配Sensor	38
9.1 GC2053摄像头访问网络码流	38
9.2 IMX415 摄像头访问码流	39
10 扩展子板的功能验证	45
10.1 扩展子板连接	45
10.2 TF卡功能	46
10.3 WiFi模块	46
10.4 RS485	49

1 Vmare 下载及安装

1.1 Vmare 下载

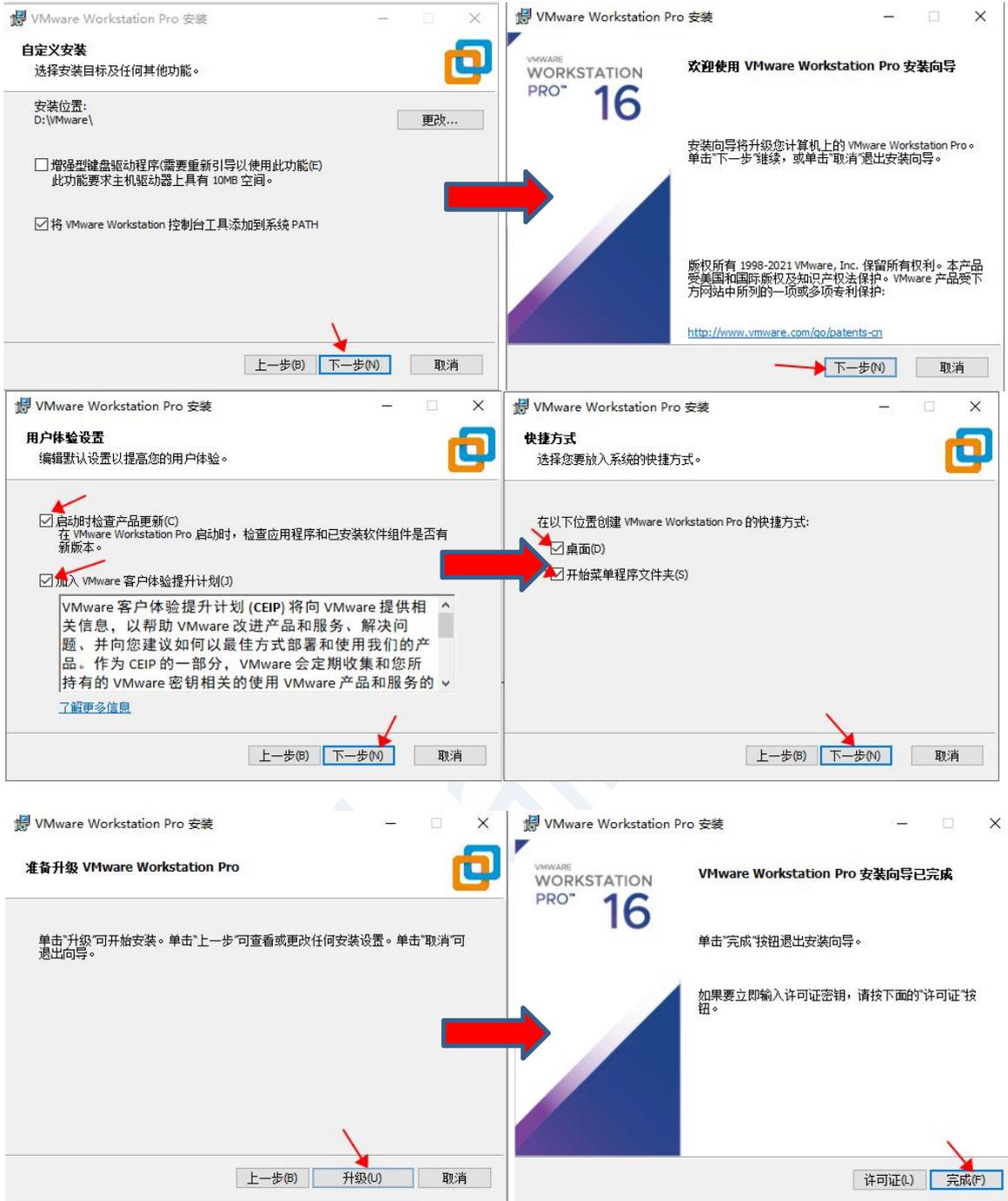
- 下载链接:<https://www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html>
- 下载选择“Workstation 16 Pro for Windows”，如下图所示：



1.2 Vmare 安装

- 安装步骤按下面图示方法操作：





2 Ubuntu 18.04 镜像下载及安装

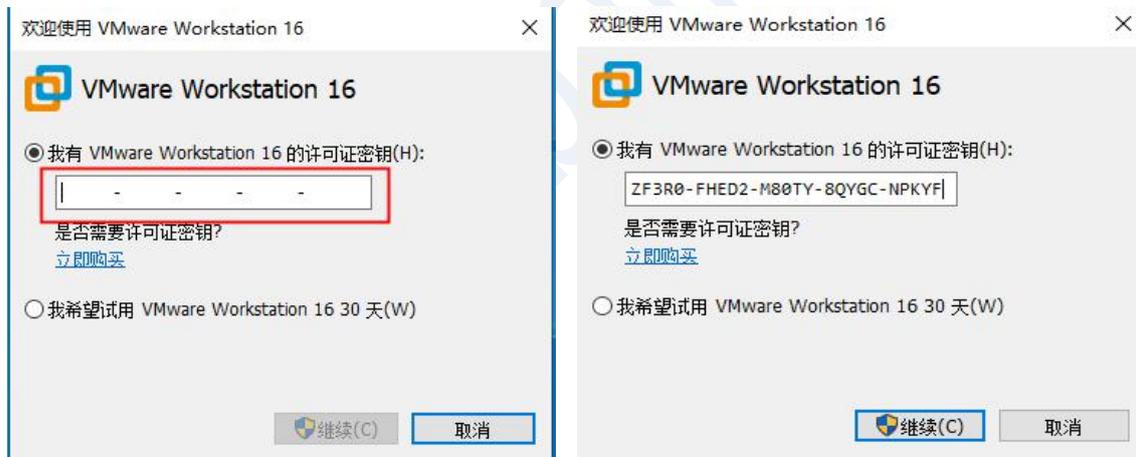
2.1 Ubuntu 18.04 镜像下载

- 下载链接: releases.ubuntu.com/18.04/
- 下载选择“ubuntu-18.04.6-desktop-amd64.iso”，如下图所示，选择红色框的镜像

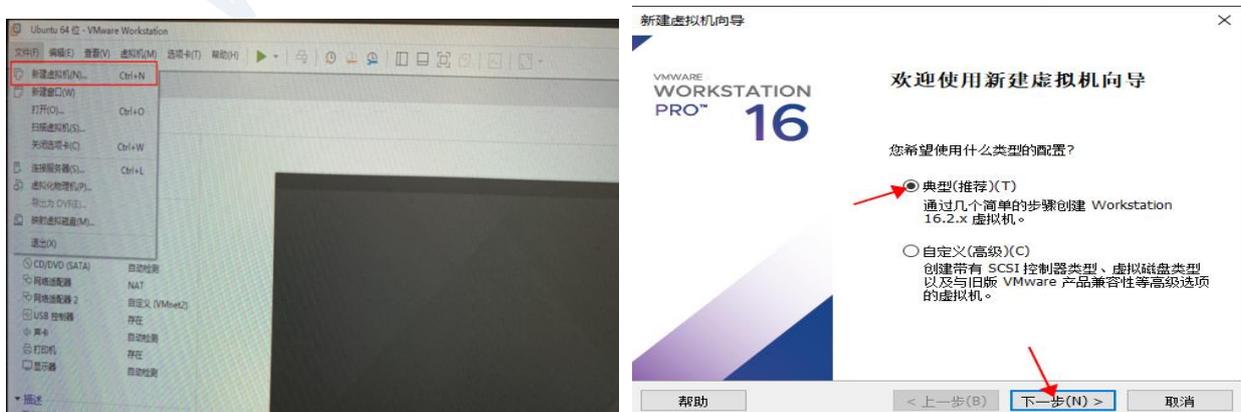
Parent Directory	-		
MD5SUMS-metalink	2020-02-12 13:42	296	
MD5SUMS-metalink.gpg	2020-02-12 13:42	916	
SHA256SUMS	2021-09-16 21:58	202	
SHA256SUMS.gpg	2021-09-16 21:58	833	
ubuntu-18.04.6-desktop-amd64.iso	2021-09-15 20:42	2.3G	Desktop image for 64-bit PC (AMD64) computers (standard download)
ubuntu-18.04.6-desktop-amd64.iso.torrent	2021-09-16 21:46	188K	Desktop image for 64-bit PC (AMD64) computers (BitTorrent download)
ubuntu-18.04.6-desktop-amd64.iso.zsync	2021-09-16 21:46	4.7M	Desktop image for 64-bit PC (AMD64) computers (zsync metafile)
ubuntu-18.04.6-desktop-amd64.list	2021-09-15 20:42	7.8K	Desktop image for 64-bit PC (AMD64) computers (file listing)
ubuntu-18.04.6-desktop-amd64.manifest	2021-09-15 20:36	59K	Desktop image for 64-bit PC (AMD64) computers (contents of live filesystem)
ubuntu-18.04.6-live-server-amd64.iso	2021-09-15 20:42	969M	Server install image for 64-bit PC (AMD64) computers (standard download)
ubuntu-18.04.6-live-server-amd64.iso.torrent	2021-09-16 21:45	76K	Server install image for 64-bit PC (AMD64) computers (BitTorrent download)
ubuntu-18.04.6-live-server-amd64.iso.zsync	2021-09-16 21:45	1.9M	Server install image for 64-bit PC (AMD64) computers (zsync metafile)
ubuntu-18.04.6-live-server-amd64.list	2021-09-15 20:42	10K	Server install image for 64-bit PC (AMD64) computers (file listing)
ubuntu-18.04.6-live-server-amd64.manifest	2021-09-15 20:36	14K	Server install image for 64-bit PC (AMD64) computers (contents of live filesystem)

2.2 Ubuntu 18.04 镜像安装

A. 打开 VMare Workstation 16 ， 并填写秘钥 ， 秘钥链接：
<https://www.jianshu.com/p/002ede3e0b17>



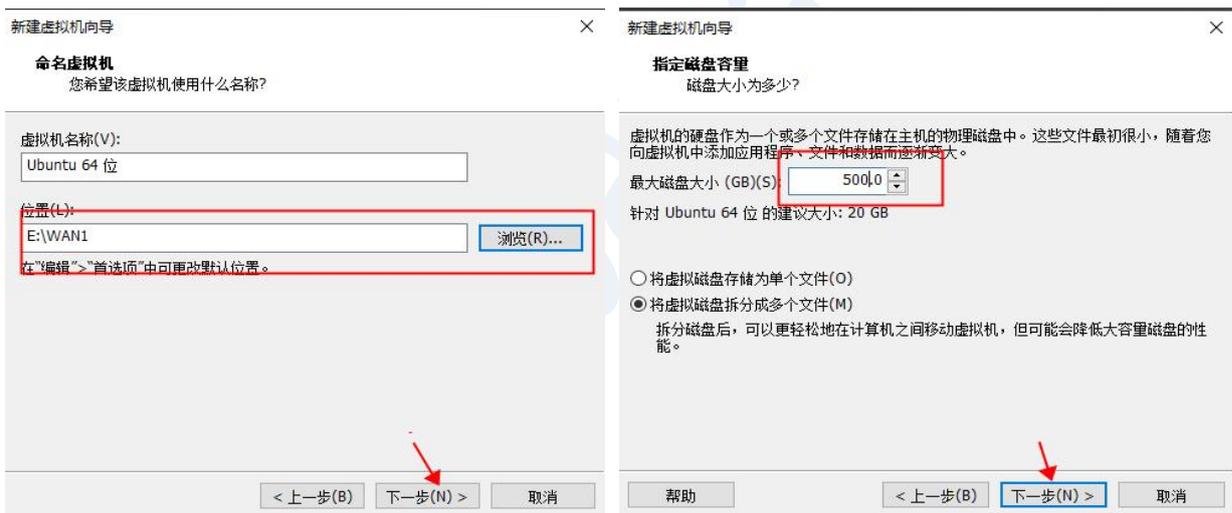
B. 继续后，在当前界面点击“文件”，选择“新建虚拟机”，然后下一步



C.选择下载好的镜像“ubuntu-18.04.6-desktop-amd64.iso”，点击进入下一步；然后填写新建的虚拟机的全名、用户名、密码以及确认密码，再点击进入下一步



D.选择安装的虚拟机位置 (建议在比较大的磁盘建立文件夹)，虚拟机的磁盘空间最好给500个G，双击点击下一步（安装的sdk较大）



E.虚拟机创建完成



3 SDK 下载及工程目录说明

3.1 SDK 的下载

下载链接：<https://pan.baidu.com/s/19MZ4rwwTBWnXugey3vmSaQ>

提取码：apnr

3.2 工程目录说明

进入工程目录下有 `buildroot`、`app`、`kernel`、`u-boot`、`device`、`docs`、`external` 等目录。每个目录或其子目录会对应一个 `git` 工程，提交需要在各自的目录下进行。

- `buildroot`: 定制根文件系统
- `app`: 存放上层应用程序
- `external`: 相关库，包括音频、视频等
- `kernel`: `kernel` 代码
- `device/rockchip`: 存放每个平台的一些编译和打包固件的脚本和预备文件
- `docs`: 存放开发指导文件、平台支持列表、工具使用文档、Linux 开发指南等
- `prebuilts`: 存放交叉编译工具链
- `rkbin`: 存放固件和工具
- `rockdev`: 存放编译输出固件
- `tools`: 存放一些常用工具
- `u-boot`: U-Boot 代码

4 Ubuntu18.04里解压下载的sdk

4.1 拷贝下载的sdk

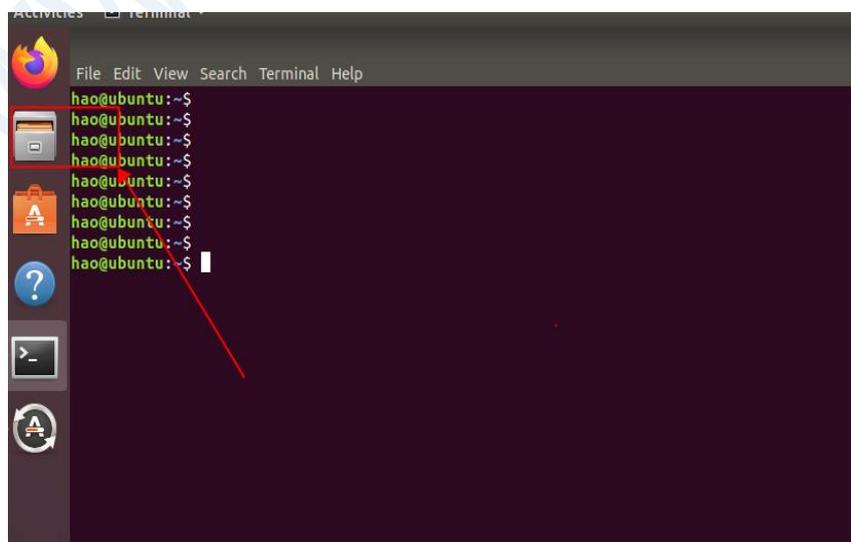
A. 选中已下载好的sdk文件，复制到 `ubuntu18.04` 里提前创建好的文件夹里



B. ctrl+ALT+T 打卡ubuntu18.04 命令行，然后mkdir +文件（文件名随意）

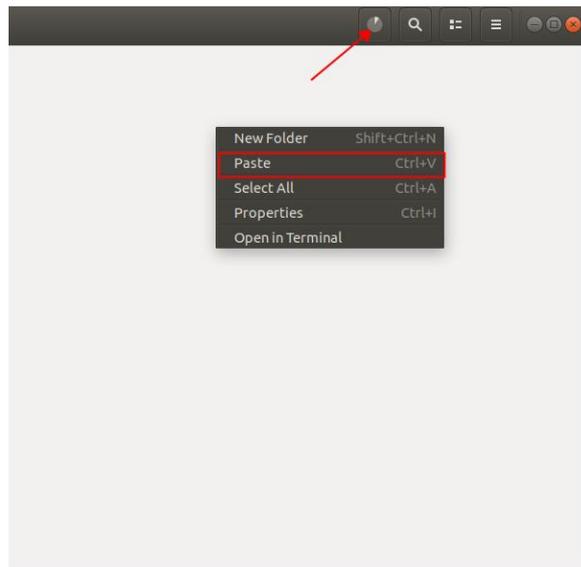
```
hao@ubuntu:~$ mkdir A191
hao@ubuntu:~$
hao@ubuntu:~$
hao@ubuntu:~$ ls
A191 Desktop Documents Downloads Music
hao@ubuntu:~$
```

C. 点击虚拟机文件夹，进入刚刚已经创建的文件夹，然后粘贴复制的sdk



D. 选择 paste 粘贴已复制的sdk，箭头标的“○”满了才算复制成功（不然会出现解压

失败)

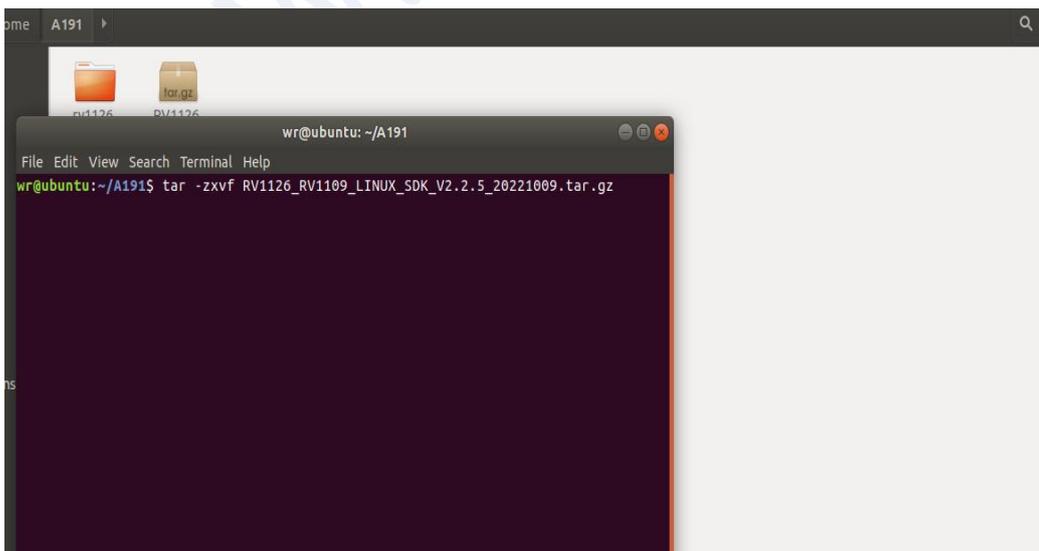


4. 2解压复制的sdk

A . 进入存放sdk的文件夹里 ls 显示存放的sdk

```
hao@ubuntu:~$  
hao@ubuntu:~$  
hao@ubuntu:~$  
hao@ubuntu:~$ cd A191  
hao@ubuntu:~/A191$  
hao@ubuntu:~/A191$  
hao@ubuntu:~/A191$ ls
```

B. 在命令行输入 `tar -zxvf RV1126_RV1109_LINUX_SDK_V2.2.5_20221009.tar.gz` 解压 sdk



C. 没有任何报错说明，解压完成

5 SDK 编译说明

在编译之前，按照下图所示连接对应接口，连接相应线材，可参考网盘资料：[EB-RV1126-BC-191型整板/组装测试教学视频](#)。其他请参考本板卡的其他硬件资料。



前提操作1: 首先执行“`sudo apt-get update`”，然后等待完成后再执行“`sudo apt-get upgrade`”，最后执行下面命令来安装一些所需文件（**注意：以下文本格式在直接复制时会格式出错**）

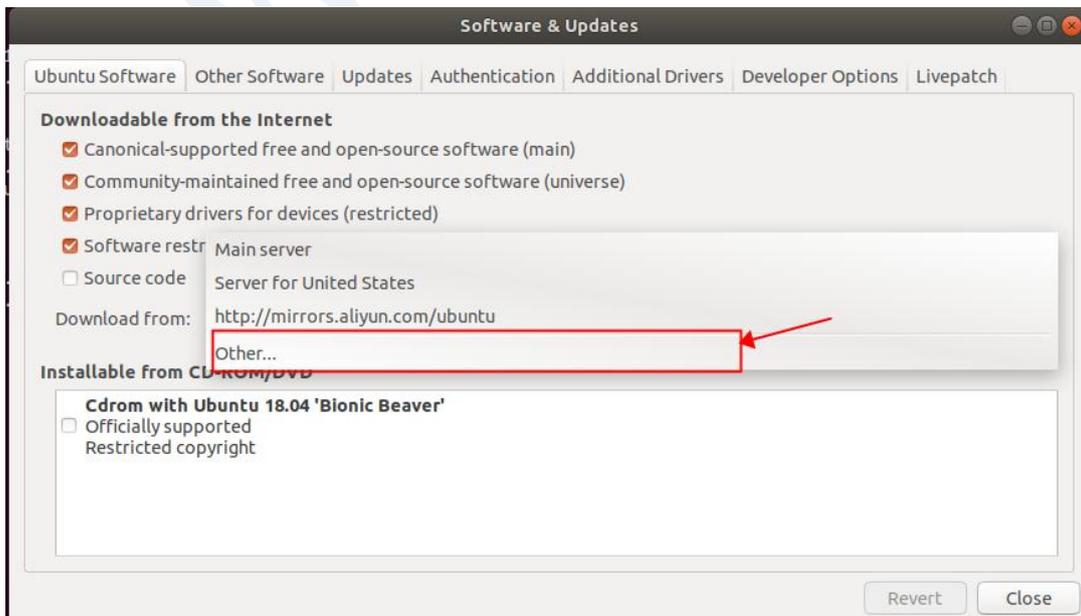
```
“sudo apt-get install repo gitk git-gui gcc-arm-linux-gnueabi u-boot-tools
device-tree-compiler gcc-aarch64-linux-gnu mtools parted libudev-dev libusb-
1.0-0-dev python-linaro-image-tools linaro-image-tools autoconf autotools-dev
libsigsegv2 m4 intltool libdrm-dev curl sed make binutils build-essential gcc
g++ bash patch gzip gawk bzip2 perl tar cpio python unzip rsync file bc wget
libncurses5 libqt4-dev libglib2.0-dev libgtk2.0-dev libglade2-dev cvs git
mercurial rsync openssh-client subversion asciidoc w3m dlatex graphviz python-
matplotlib libc6:i386 libssl-dev expect fakeroot cmake flex bison liblz4-tool
libtool keychain expect-dev ”
```

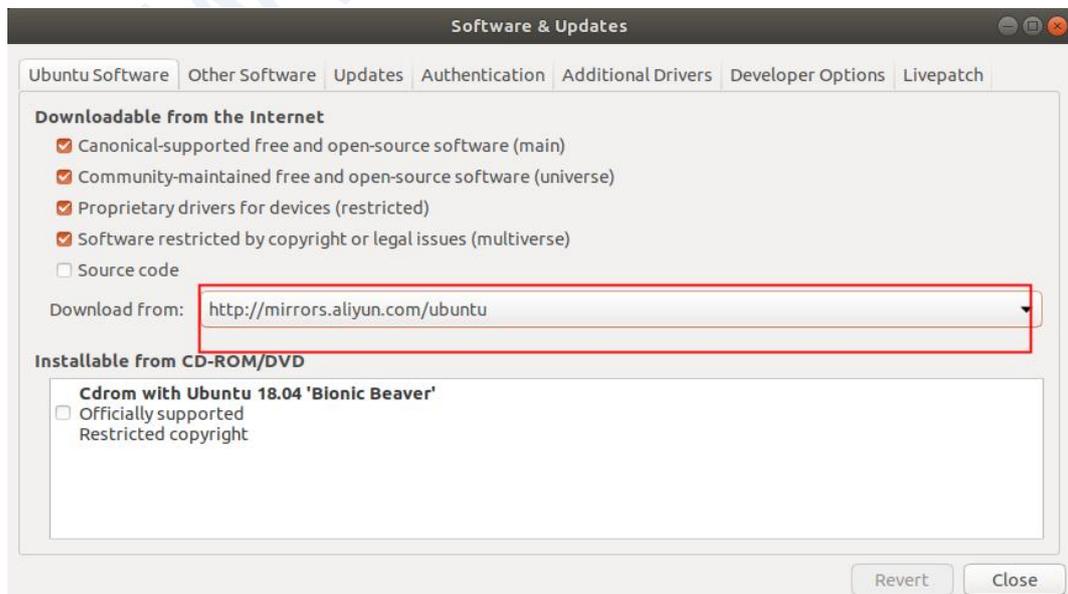
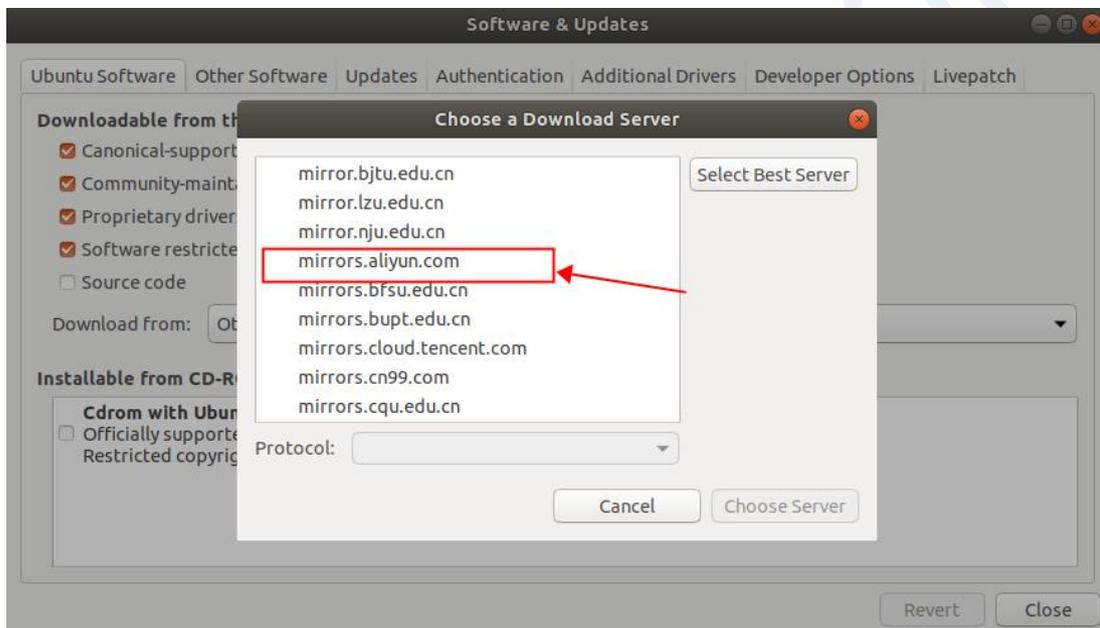
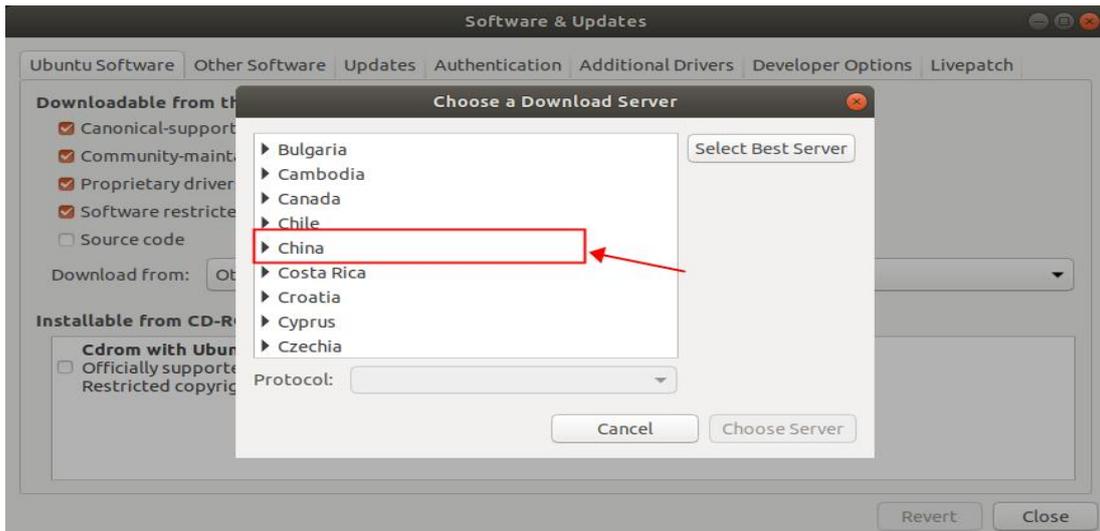
安装所需文件报错图片：

```

Get:336 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 llnano-image-tools amd64 2016.05-1.1 [14.8 kB]
Get:337 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 lmodern all 2.004.5-3 [9,631 kB]
Get:338 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 manpages-dev all 4.15-1 [2,217 kB]
Get:339 http://us.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 mercurial-common all 4.5.3-1ubuntu2.2 [2,199 kB]
Get:340 http://us.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 mercurial amd64 4.5.3-1ubuntu2.2 [193 kB]
Get:341 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 python-backports.functools-lru-cache all 1.4-2 [5,966 B]
Get:342 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 python-cycler all 0.10.0-1 [7,520 B]
Get:343 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 python-dateutil all 2.6.1-1 [60.6 kB]
Get:344 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 python-gi amd64 3.26.1-2ubuntu1 [197 kB]
Get:345 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 python-kerberos amd64 1.1.14-1 [22.5 kB]
Get:346 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 ttf-bitstream-vera all 1.10-8 [352 kB]
Get:347 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 python-matplotlib-data all 2.1.1-2ubuntu3 [3,774 kB]
Get:348 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 python-pyparsing all 2.2.0+dfsg1-2 [52.1 kB]
Get:349 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 python-tz all 2018.3-2 [31.6 kB]
Get:350 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 python-numpy amd64 1:1.13.3-2ubuntu1 [1,938 kB]
Get:351 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 python-subprocess32 amd64 3.2.7-3 [27.2 kB]
Get:352 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 python-matplotlib amd64 2.1.1-2ubuntu3 [3,901 kB]
Get:353 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 python-olefile all 0.45-1 [33.2 kB]
Get:354 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 python-pil amd64 5.1.0-1ubuntu0.7 [303 kB]
Get:355 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 python-tk amd64 2.7.17-1-18.04 [26.0 kB]
Get:356 http://us.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 qemu-user-static amd64 1:2.11+dfsg-1ubuntu7.38 [9,990 kB]
Get:357 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 qt-at-spi amd64 0.4.0-8 [58.0 kB]
Get:358 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 subversion amd64 1.9.7-4ubuntu1 [834 kB]
Get:359 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 tex-gyre all 20160520-1 [4,998 kB]
Get:360 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 vin-runtime all 2:8.0.1453-1ubuntu1.7 [5,435 kB]
Get:361 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 vin amd64 2:8.0.1453-1ubuntu1.7 [1,153 kB]
Get:362 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 vin-addon-manager all 0.5.7 [18.7 kB]
Get:363 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 vin-asciidoc all 0.6.10-2 [9,320 B]
Get:364 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 xmlto amd64 0.0.28-2 [26.6 kB]
Get:365 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 device-tree-compiler amd64 1.4.5-3 [239 kB]
Get:366 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 linux-ltbc-dev-arm64-cross all 4.15.0-35.38cross1.1 [859 kB]
Get:367 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 ltbc-dev-arm64-cross all 2.27-3ubuntu1cross1.1 [2,044 kB]
Get:368 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 linux-ltbc-dev-armhf-cross all 4.15.0-35.38cross1.1 [862 kB]
Get:369 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 ltbc-dev-armhf-cross all 2.27-3ubuntu1cross1.1 [1,899 kB]
Get:370 http://us.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 liblz4-tool amd64 0.0-r131-2ubuntu3.1 [75.0 kB]
Get:371 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 repo all 1.12.37-3ubuntu1 [21.5 kB]
Err:47 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 fonts-gfs-baskerville all 1.1-5
Connection failed [IP: 91.189.91.39 80]
Err:48 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 fonts-gfs-porson all 1.1-6
Connection failed [IP: 91.189.91.39 80]
Get:49 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-lang-greek all 2017.20180305-1 [76.3 MB]
Fetched 486 MB in 39min 50s (203 kB/s)
E: failed to fetch http://us.archive.ubuntu.com/ubuntu/pool/universe/f/fonts-gfs-baskerville/fonts-gfs-baskerville_1.1-5_all.deb Connection failed [IP: 91.189.91.39 80]
E: failed to fetch http://us.archive.ubuntu.com/ubuntu/pool/universe/f/fonts-gfs-porson/fonts-gfs-porson_1.1-6_all.deb Connection failed [IP: 91.189.91.39 80]
E: Unable to fetch some archives, maybe run apt-get update or try with --fix-missing?
    
```

解决办法：图形界面换源





安装成功：

```
Processing triggers for install-info (6.5.0.dfsg.1-2) ...
Processing triggers for libglib2.0-0:amd64 (2.56.4-0ubuntu0.18.04.9) ...
Setting up libatk1.0-dev:amd64 (2.28.1-1) ...
Setting up libgdk-pixbuf2.0-dev (2.36.11-2) ...
Setting up libharfbuzz-dev:amd64 (1.7.2-1ubuntu1) ...
Setting up libicu-le-hb-dev:amd64 (1.0.3+git161113-4) ...
Setting up libcairo2-dev:amd64 (1.15.10-2ubuntu0.1) ...
Setting up libpango1.0-dev (1.40.14-1ubuntu0.1) ...
Setting up libgtk2.0-dev (2.24.32-1ubuntu1) ...
Setting up libglade2-dev:amd64 (1:2.6.4-2) ...
Processing triggers for sgml-base (1.29) ...
Setting up docbook-xsl (1.79.1+dfsg-2) ...
Setting up sgml-data (2.0.10) ...
Processing triggers for sgml-base (1.29) ...
Setting up docbook-xml (4.5-8) ...
Processing triggers for sgml-base (1.29) ...
Setting up dblatex (0.3.10-2) ...
Setting up docbook-dsssl (1.79-9.1) ...
Setting up xmlto (0.0.28-2) ...
Processing triggers for sgml-base (1.29) ...
Setting up docbook-utils (0.6.14-3.3) ...
Setting up asettidoc-dblatex (8.6:10-2) ...
Processing triggers for tex-common (6.09) ...
Running updmap-sys. This may take some time... done.
Running mktexlsr /var/lib/texmf ... done.
Building format(s) --all.
This may take some time.. done.
hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224$
```

前提操作2: 安装完成后在工程的根目录下执行命令“`source envsetup.sh`”会出现很多选项，输入“85”来选择`rockchip_rv1126_rv1109_spi_nand`，接着在工程的根目录下执行“`./build.sh lunch`”，这时会出现很多选项，输入“4”来选择`BoardConfig-38x38-spi-nand.mk`。

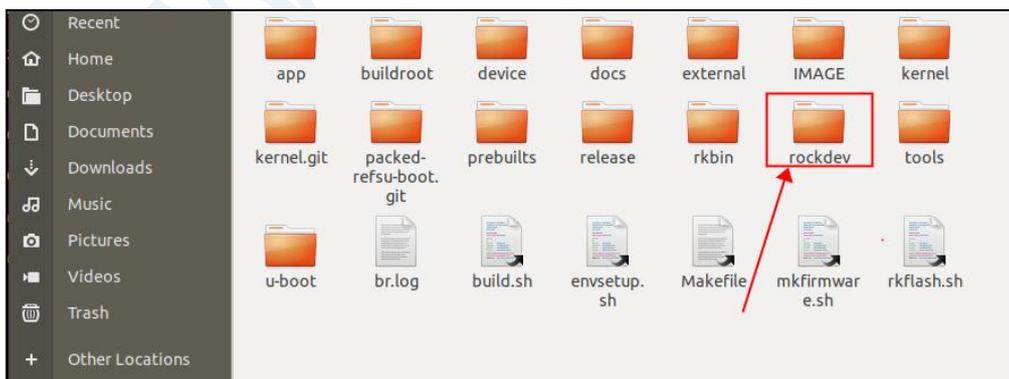
注意: 前提操作1安装一次就不需要再做了，前提操作2是每在一个新终端上编译都需要执行一次。

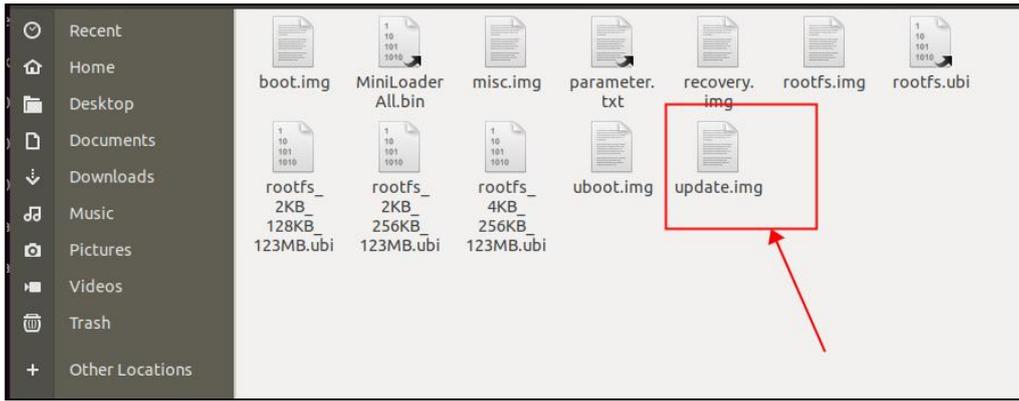
5. 1总体编译

在完前提操作下在工程根目录下执行命令“`./build.sh`”

5. 2总体编译生成的固件位置

生成的固件存放在 解压好的sdk 文件夹 `/rockdev` 里面





5. 3分步编译

一、U-Boot 编译

(1) u-boot 配置说明

使用 menuconfig 配置 U-Boot，选择需要的模块，最后保存退出。

rv1126_defconfig 文件在目录 u-boot/configs

命令格式：make "RK_UBOOT_DEFCONFIG"_defconfig

RK_UBOOT_DEFCONFIG 定义在./build.sh 选择的 BoardConfig*.mk

先进入压缩包目录然后按照以下命令

```
cd u-boot
```

```
make rv1126_defconfig
```

```
make menucon
```

保存配置到对应的文件 rv1126_defconfig

```
make savedefconfig
```

```
cp defconfig configs/rv1126_defconfig
```

(2) U-Boot 编译

进入到压缩包根目录执行./build.sh uboot

二、Kernel 编译

(1) Kernel 配置说明

执行以下命令前先保证处在工程根目录下

例如 device/rockchip/rv1126_rv1109/BoardConfig-38x38-spi-nand.mk

```
./build.sh device/rockchip/rv1126_rv1109/BoardConfig-38x38-spi-nand.mk
```

```
cd kernel
```

```
### 命令格式： make ARCH=arm "RK_KERNEL_DEFCONFIG"  
"RK_KERNEL_DEFCONFIG_FRAGMENT"
```

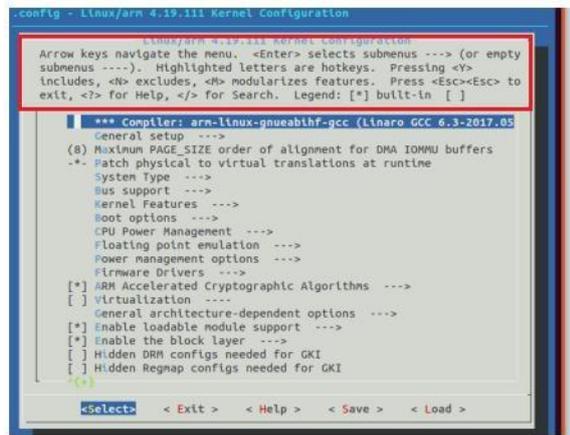
RK_KERNEL_DEFCONFIG 和 RK_KERNEL_DEFCONFIG_FRAGMENT 都定

义在./build.sh 选择的 BoardConfig*.mk

RK_KERNEL_DEFCONFIG_FRAGMENT 是可选项，具体看 BoardConfig*.mk 配置。

make ARCH=arm rv1126_defconfig

make ARCH=arm menuconfig //执行完后会出现下图



这里是操作说明，回车键时确认，y是选中
n是取消选中，m是模块化特征，按两下
ESC是退出，?是help，/是搜索，[*]表示选中，[]未选中

make ARCH=arm savedefconfig

cp defconfig arch/arm/configs/rv1126_defconfig

(2) Kernel 编译

执行下面命令之前要保证在工程根目录下

./build.sh kernel

```
DTC      arch/arm/boot/dts/rv1126-evb-ddr3-v13.dtb
CALL     scripts/checksyscalls.sh
CHK      include/generated/compile.h
Kernel:  arch/arm/boot/Image is ready
Kernel:  arch/arm/boot/zImage is ready
Image:   kernel.img (with zImage) is ready
CALL     scripts/checksyscalls.sh
Building modules, stage 2.
MODPOST 2 modules
Image:   resource.img (with rv1126-evb-ddr3-v13.dtb logo.bmp logo_kernel.bmp) is ready
Image:   boot.img (with Image resource.img) is ready
Image:   zboot.img (with zImage resource.img) is ready
Running build kernel succeeded.
```

以上 Running build_kernel succeeded. 表示编译内核成功。但这个过程中可能出现和总体编译中一样的错误，按照总体编译中的解决方案即可。

三、Rootfs 编译

(1) 目录 app 和 external 里的工程编译方法以及 Rootfs 配置说明

1. 先 SDK 根目录查看 Board Config 对应的 rootfs 是哪个配置

./build.sh -h rootfs

#####Current SDK Default [rootfs] Build Command####

source envsetup.sh rockchip_rv1126_rv1109_spi_nand

```
# make
```

```
### 2. source buildroot 对应的 defconfig
```

```
source envsetup.sh rockchip_rv1126_rv1109_spi_nand
```

```
### 3. 使用 menuconfig 配置文件系统，选择需要的模块，最后保存退出。
```

```
### 例如：ipc-daemon 的配置是 BR2_PACKAGE_IPC_DAEMON（查看  
buildroot/package/rockchip/ipc-daemon/Config.in）
```

```
make menuconfig # 进入 menuconfig 后，按 “/” 进入查找模式，输入
```

```
BR2_PACKAGE_IPC_DAEMON
```

```
### 4. 保存到 rootfs 配置文件
```

```
### ./buildroot/configs/rockchip_rv1126_rv1109_defconfig
```

```
make savedefconfig
```

(2) Rootfs 编译

```
### 查看 Rootfs 详细编译命令
```

```
./build.sh -h rootfs
```

```
### Rootfs 编译命令
```

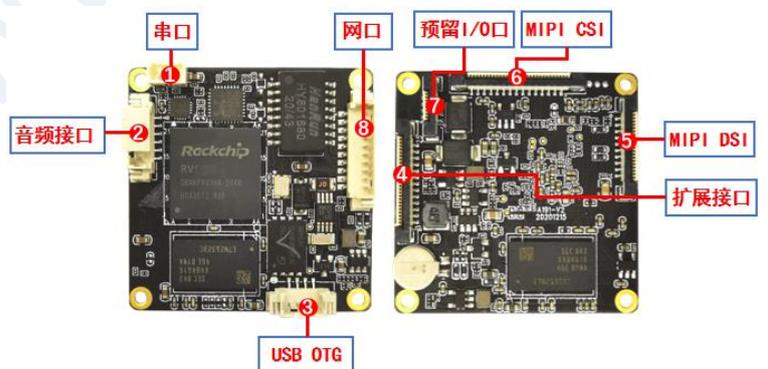
```
./build.sh rootfs
```

编译过程中可能出现如总体编译中的问题，按照解决方案解决即可。

6 固件烧写

6.1 硬件准备

在烧写固件之前，按照下图所示连接对应接口，连接出厂适配的尾线和USB烧写线。如有疑问可参考网盘资料：组装测试教学视频。



连接摄像头完成的示意图如下：

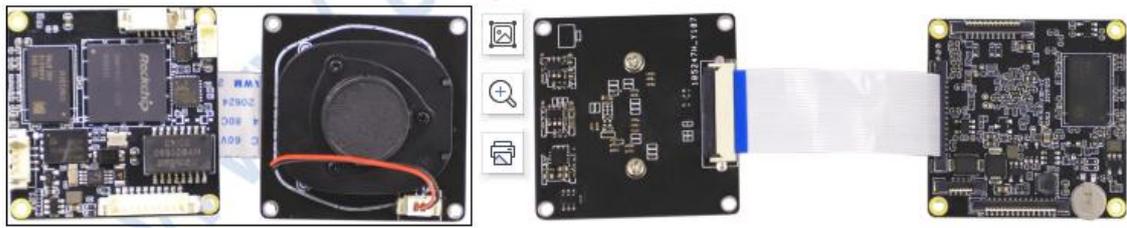


图 6.5 RV1126+IMX415 正面图（左）和 RV1126+IMX415 背面图（右）

6.2 软件准备

- 驱动安装，安装路径：我的网盘\艾伯瑞产品资料大全\EB-RV1126-BC-191型整板\开发工具\瑞芯微开发工具包/DriverAssitant_v4.91

文件名	修改时间	类型	大小
adb1.0.32.zip	2021-10-12 10:30	zip文件	607KB
AndroidTool_Release_v2.71.zip	2021-10-12 10:30	zip文件	2.44MB
DriverAssitant_v4.91.zip	2021-10-12 10:30	zip文件	9.37MB
VLC.zip	2021-10-12 10:30	zip文件	39.91MB
securecrt.zip	2021-10-12 10:30	zip文件	9.50MB

- 烧写工具安装，安装路径：我的网盘\艾伯瑞产品资料大全\EB-RV1126-BC-191型整板\开发工具\瑞芯微开发工具包/AndroidTool_Release_v2.71.zip，软件版本不得低于 **v2.71**

文件名	修改时间	类型	大小
adb1.0.32.zip	2021-10-12 10:30	zip文件	607KB
AndroidTool_Release_v2.71.zip	2021-10-12 10:30	zip文件	2.44MB
DriverAssitant_v4.91.zip	2021-10-12 10:30	zip文件	9.37MB
VLC.zip	2021-10-12 10:30	zip文件	39.91MB
securecrt.zip	2021-10-12 10:30	zip文件	9.50MB

- 其他：在编译完成后，工程目录下的 rockdev 下会生成相关文件，把 rockdev 文件夹拷贝到windows 环境下，打开瑞芯微开发工具，如下图 3.1 所示：



6.3 固件烧写

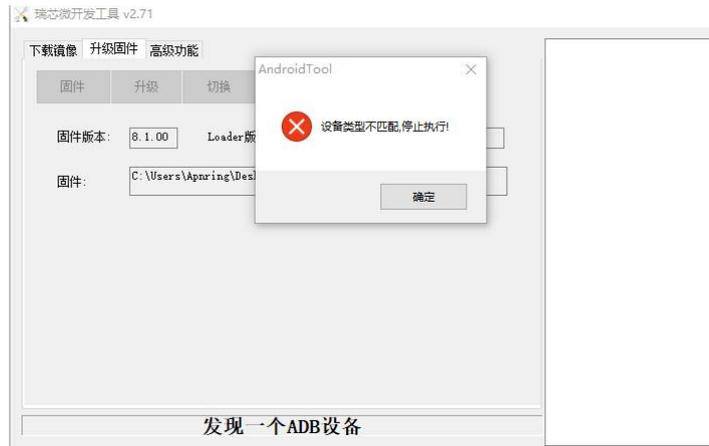
这里介绍两种方法来下载固件，在这之前必须要保证在 **MSAKROM** 模式下。

方法一：该方法比较简单，点击到“升级固件”这个界面下，先点击固件选择 windows 下 rockdev 下的 update.img 文件，然后点击“升级”，成功后如下图右侧的字样。注意事项在图中都已说明，主要是模式切换问题。



注意：从 loader 模式到 Maskrom 模式也可以点击“高级功能——进入 Maskrom”。

方法一容易出现的问题：设备类型不匹配，停止执行！



解决方法：

断开电源然后重新上电，点击“切换”，就会发现 LOADER 设备。点击“高级功能”，然后点击“进入 Maskrom”，等待一会可发现 Maskrom 设备。再点击“升级固件”、“擦除 Flash”（若擦除不成功，重复多次也擦不掉，则可断电然后重新上电，待发现 MASKROM 后，再点击擦除），等待擦除成功后点击“升级”，即可完成重烧固件。

方法一操作步骤：





方法二：点击到“下载镜像”的界面下，如下图所示。

首先第一步：起始地址的填写。查看 linux 或 windows 下 rockdev 的子文件 parameter.txt，里面有起始地址。

顺便解释一下 rockdev 下面的文件与 parameter.txt，如下图中的 vnvmm：

输入：cat rockdev/parameyer.txt

```

F:\WINDOWS: 8.1
MACHINE_MODEL: RV1126
MACHINE_ID: 007
MANUFACTURER: RV1126
MAGIC: 0x5041524B
ATAG: 0x00200800
MACHINE: 0xffffffff
CHECK_MASK: 0x80
PWR_HLD: 0,0,A,0,1
TYPE: GPT
CMDLINE: mtdparts=rk29xxnand:0x00000800@0x00001800(vnvmm),0x00002000@0x00002000(uboot),0x00004000@0x00004000(boot),0x000032000@0x00008000(rootfs),0x000032000@0x0003A000(oem),-@0x0006C000(userdata:grow)
uuid:rootfs=614e0000-0000-4b53-8000-1d28000054a9
  
```

@的左边表示大小，右边表示起始地址

在 rockdev下面并没有与此相关的文件，所以在“下载镜像”中并没有填写相关信息。其他的类推（除了 MiniLoaderAll.bin 和 parameter.txt，这两项虽然 parameter.txt 中没有，但在“下载镜像”中添加了）。

第二步：编写名字，根据下图中的名字就可以了，也可以自拟。主要是起始地址必须要保证正确。

第三步：路径，点击下图右侧红框里面的“...”，选择文件。

#	名字	路径
1	Loader	C:\Users\Apnring\Desktop\rockdev\MiniLoaderAll bin
2	Parameter	C:\Users\Apnring\Desktop\rockdev\parameter.txt
3	Uboot	C:\Users\Apnring\Desktop\rockdev\uboot.img
4	boot	C:\Users\Apnring\Desktop\rockdev\boot.img
5	rootfs	C:\Users\Apnring\Desktop\rockdev\rootfs.img
6	oem	C:\Users\Apnring\Desktop\rockdev\oem.img

第四步：在 MASKROM 模式下点击“执行”，成功如下图所示：

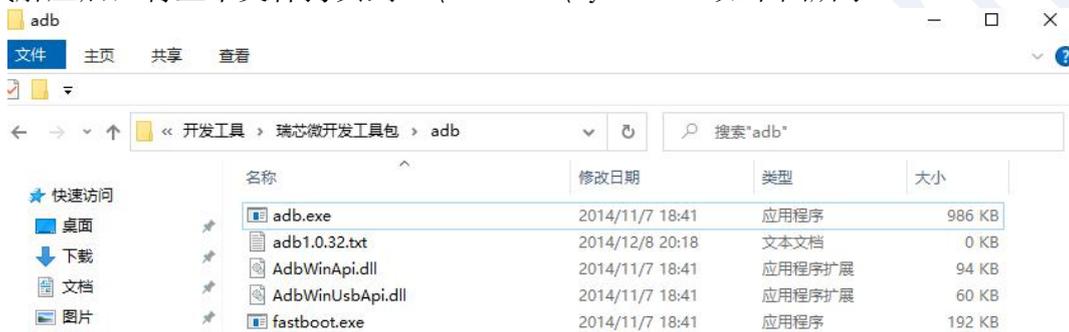


6.4 adb的安装

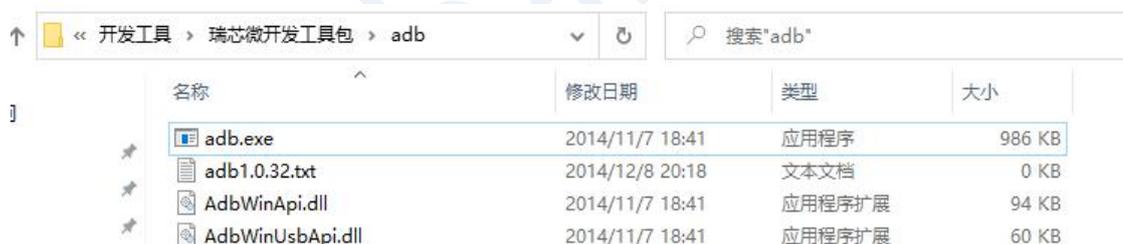
不管是方法一还是方法二在操作成功后设备模式变成了 **ADB 设备**。在这个基础上我们需要验证我们编好的固件下载进设备到底有没有用，这时需要用到 adb 工具。我们可以通过 adb 工具直接访问到设备内部并且通过相关命令可以操作设备。adb 工具在硬件资料的“软件或软件”文件夹下面，以下是 adb 工具的安装和用法。

(1) adb 的安装

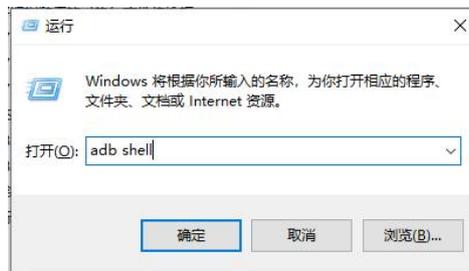
- 下载解压后，将五个文件拷贝到C:\Windows\System32:如下图所示:



- 将这4个文件拷贝至C:/Windows/System, 如下图所示:



- 将adb.exe和AdbWinApi.dll拷贝至C:/Windows/SysWOW64
- 右击电脑“开始”，点击“运行”或者直接快捷键“windows徽标+R打开运行界面”试着输入”adb shell”右键, 确定后结果如下图所示，到此adb工具安装成功



windows+R进入命令行

(2) adb 的使用

```
[root@RV1126_RV1109:/]#  
[root@RV1126_RV1109:/]# ls  
ls  
app          etc          linuxrc     oem          run          tmp          vendor  
bin          init        lost+found  opt          sbin         udisk  
busybox.config ko          media       proc         sdcard      userdata  
data        lib         misc       rockchip_test sys          usr  
dev         lib32      mnt        root         timestamp   var  
[root@RV1126_RV1109:/]#  
[root@RV1126_RV1109:/]#  
[root@RV1126_RV1109:/]#
```

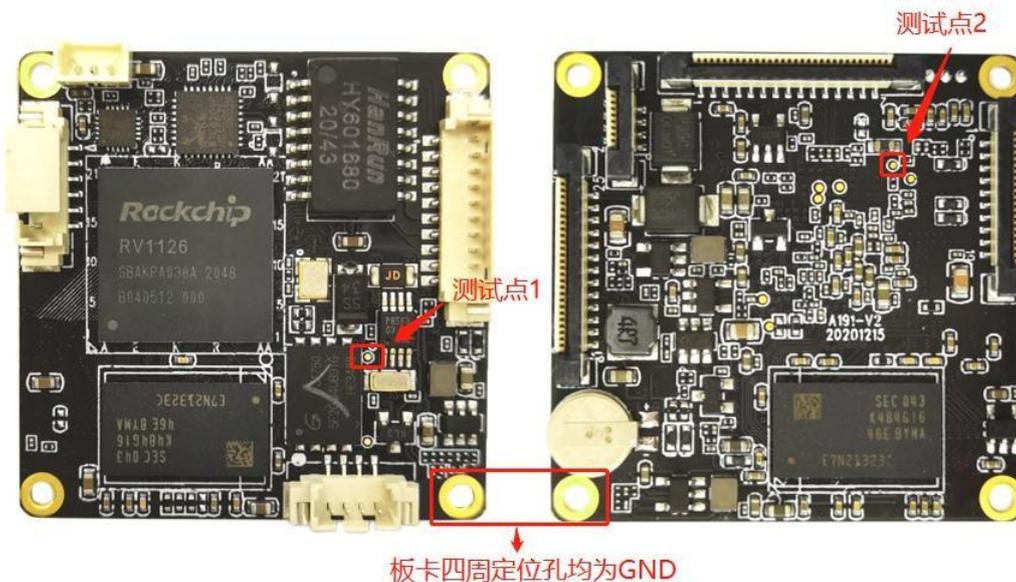
- 语法：“命令”，如上图，“ls”能成功显示，说明编写的固件 ok，并且烧写成功了。
- 配置网络：“ifconfig eth0 + IP”配置设备 ip（配置的ip就是ssh连接的ip）

Ps：默认设置100网段的IP地址，这个地址指的是连接的固件IP地址。

6. 4烧写固件注意事项

如若烧写固件时烧写工具无法获取到设备，操作步骤如下：

- (1) 打开瑞芯微开发工具。上电前，将测试点 1 或者测试点 2（如下图）和 GND 进行短接，短接的同时上电，就可以发现设备了。



- (2) 烧写完毕后要输入如下命令：`dmesg|grep 2053`

检查默认是否认到摄像头设备（ps：此步骤在务必该顺序下操作，后续拉流时再检查，可能会出现缓存的摄像头数据被系统数据覆盖，导致认不到摄像头设备）

```
C:\Windows\system32\adb.exe
adb server is out of date. killing...
adb daemon started successfully *
root@RV1126_RV1109:/# dmesg | grep 2053
dmesg | grep 2053
0.607441] gc2053 1-0037: driver version: 00.01.01
0.607468] gc2053 1-0037: GPIO lookup for consumer reset
0.607473] gc2053 1-0037: using device tree for GPIO lookup
0.607490] of_get_named_gpiod_flags: parsed 'reset-gpios' property of node '/i2c0
0.607515] gc2053 1-0037: GPIO lookup for consumer pwn
0.607519] gc2053 1-0037: using device tree for GPIO lookup
0.607529] of_get_named_gpiod_flags: can't parse 'pwn-gpios' property of node '/
0.607538] of_get_named_gpiod_flags: can't parse 'pwn-gpio' property of node '/i
0.607543] gc2053 1-0037: using lookup tables for GPIO lookup
0.607548] gc2053 1-0037: No GPIO consumer pwn found
0.607552] gc2053 1-0037: Failed to get pwn-gpios, maybe no used
0.607562] gc2053 1-0037: GPIO lookup for consumer power
0.607565] gc2053 1-0037: using device tree for GPIO lookup
0.607574] of_get_named_gpiod_flags: can't parse 'power-gpios' property of node '
0.607583] of_get_named_gpiod_flags: can't parse 'power-gpio' property of node '
0.607587] gc2053 1-0037: using lookup tables for GPIO lookup
0.607591] gc2053 1-0037: No GPIO consumer power found
0.607595] gc2053 1-0037: Failed to get power-gpios
0.607669] gc2053 1-0037: Linked as a consumer to regulator.2
0.607729] gc2053 1-0037: Linked as a consumer to regulator.4
0.607779] gc2053 1-0037: Linked as a consumer to regulator.3
0.607797] gc2053 1-0037: lane_num(2) pixel_rate(11880000)
0.607838] gc2053 1-0037: could not get sleep pinstate
0.612704] gc2053 1-0037: Detected GC2053 sensor
```

7 远程连接（adb,telnet 或串口、波特率等信息）

7.1 ssh 连接（示例为 windows 下操作）

首先在保证连接正常的情况下，按“windows + R”进入命令行，执行命令“ping +ip”（这个 ip 是连接摄像头设备的 ip，前面在 adb 用法处说过如何配置）看看是否 ping 的通，下图是 ping 通的情形，ping 不通就需要检查网线是否接好以及是否与设备在同一个网段。（保持在同一个网段 可以用电脑连wifi，板子通过网线连接电脑）

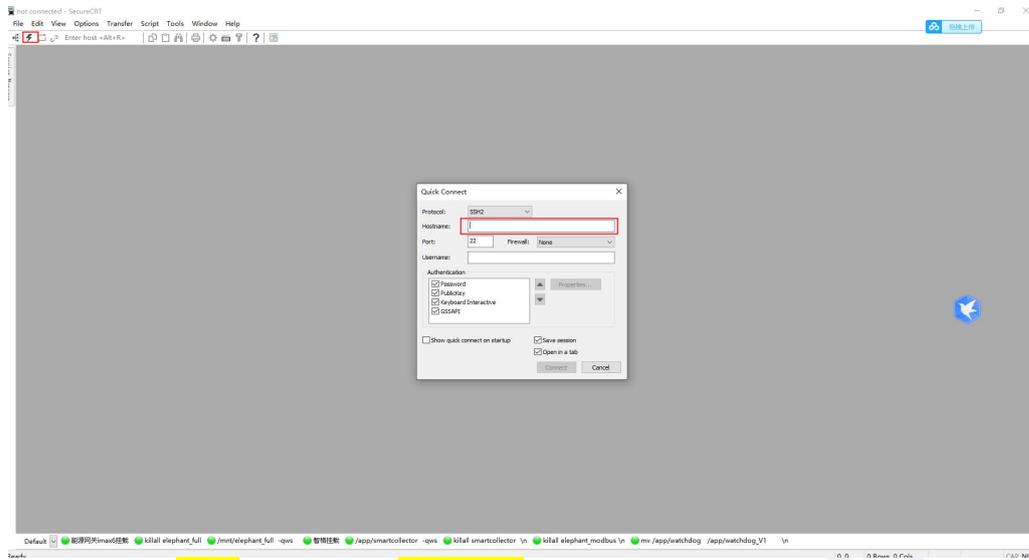
使用“Ctrl+C”停止ping命令，不然会一直在窗口ping下去

```
C:\Users\Apnring>ping 192.168.100.75
正在 Ping 192.168.100.75 具有 32 字节的数据:
来自 192.168.100.75 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.100.75 的回复: 字节=32 时间<1ms TTL=64
来自 192.168.100.75 的回复: 字节=32 时间<1ms TTL=64
```

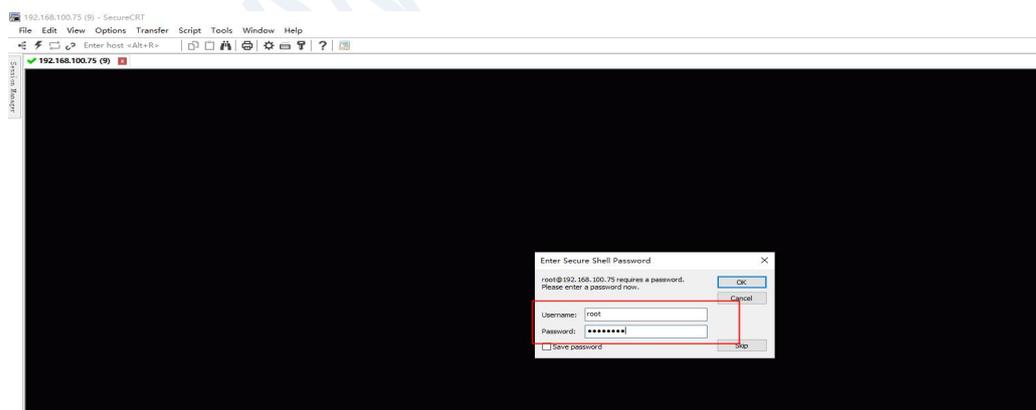
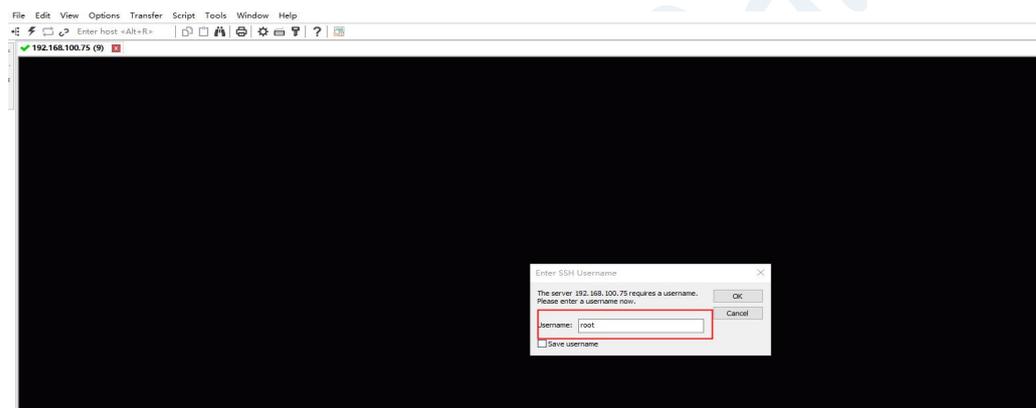
● 安装SecureCRT, 安装链接:

链接: <https://pan.baidu.com/s/1VWjTrt2GwsRrTFveeNYh1Q> , 提取码: apnr

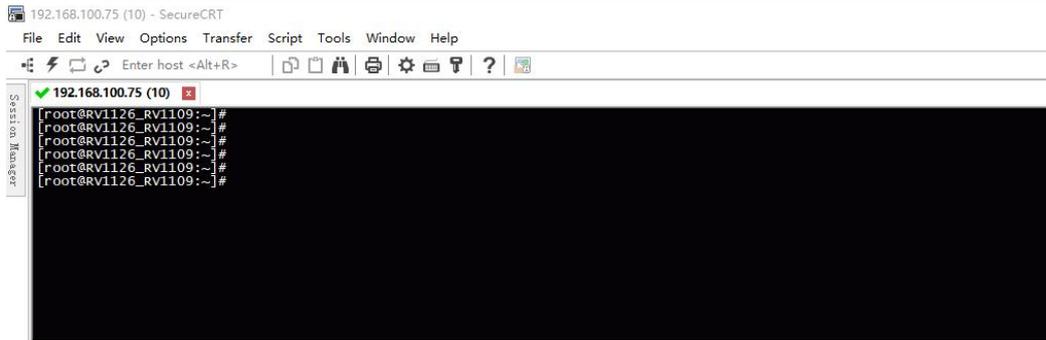
打开软件，点击快速连接（左上角的闪电符号），选择ssh模式下设置的板端地址，然后连接



- 输入用户名“root”和密码“rockchip”点击ok



- 连接成功如下图



7.2 scp 连接（示例为Linux 下操作）

从 PC 端上传文件 test-file 到 EVB 板的目录/userdata

```
scp /mnt/test.txt root@192.168.1.159:/userdata/
```

输入默认密码: rockchip

```
leo@ubuntu:/mnt$ scp /mnt/test.txt root@192.168.100.75:/userdata/  
root@192.168.100.75's password:  
test.txt                               100% 0 0.0KB/s 00:00
```

下载 EVB 板上的文件/userdata/test-file 下载到 PC 端

```
scp root@192.168.1.159:/userdata/test.txt /mnt/test.txt
```

7.3. 通过网络 ADB 调试

获取 EVB 板的 IP 地址 192.168.1.159

```
adb connect 192.168.1.159
```

```
adb devices
```

adb 登陆 EVB 板子调试

```
adb -s 192.168.1.159:5555 shell
```

从 PC 端上传文件 test-file 到 EVB 板的目录/userdata

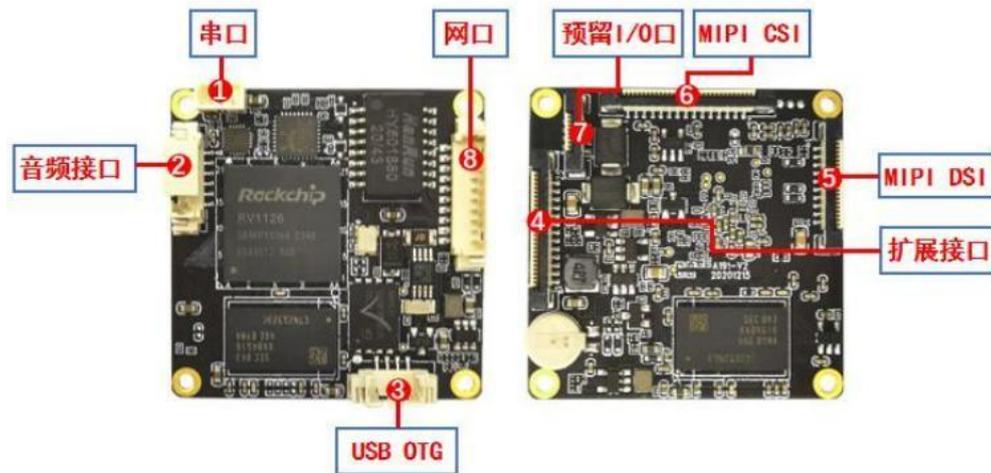
```
adb -s 192.168.1.159:5555 push test-file /userdata/
```

下载 EVB 板上的文件/userdata/test-file 下载到 PC 端

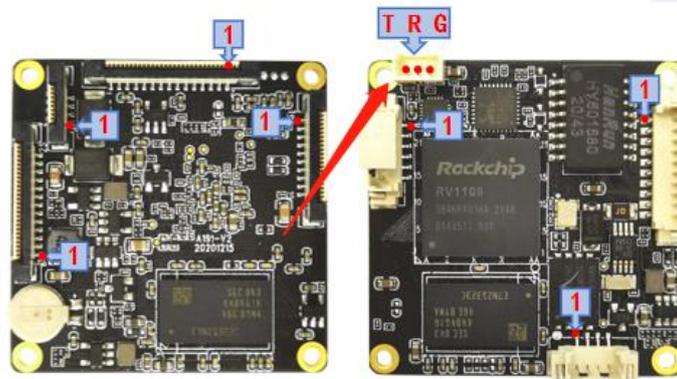
```
adb -s 192.168.1.159:5555 pull /userdata/test-file test-file
```

7.4 调试串口

- 调试串口的波特率为 1500000（若无此选项，可手动输入），无校验



- 串口的引脚定义如下图所示，图中“1”为接口的第一脚标注



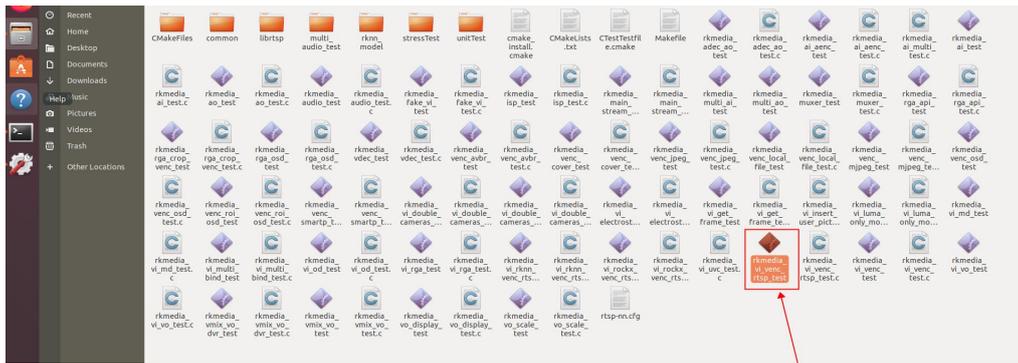
8 运行取频流的可执行文件

8.1 编译sdk取视频流的可执行文件

命令行进入解压好的sdk文件夹，在工程的根目录下执行命令“source envsetup.sh”会出现很多选项，输入“78”来选择“rockchip_rv1126_rv1109”接着在工程的根目录下执行“./build.sh lunch”，这时会出现很多选项，输入“4”来选择“BoardConfig-aybering.mk”，然后总体编译./build.sh。

8.2 可执行程序的路径

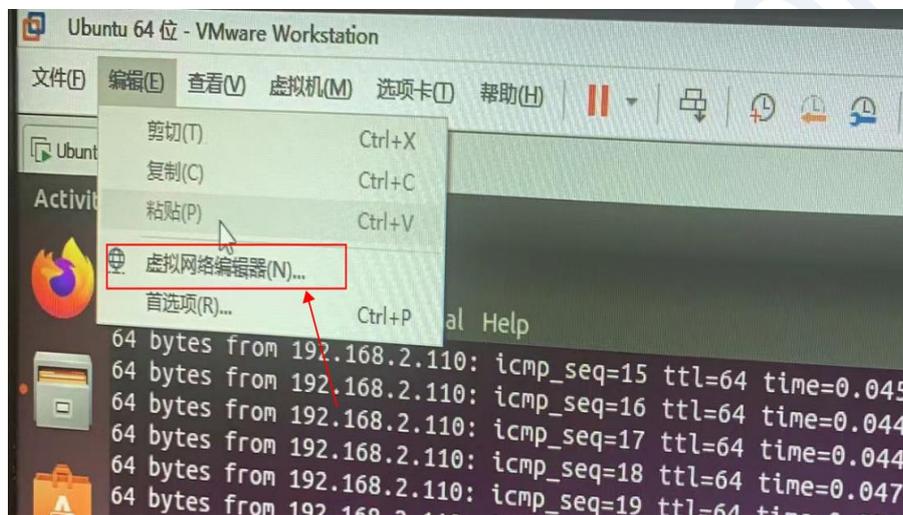
测试视频流可执行程序的路径：Home/自己创建的文件夹/rv1126_rv1109_linux_sdk_v2.2.5_2021224/buildroot/Output/rockchip_rv1126_rv1109/build/rkmedia/examples



8.3 Ubuntu NFS 服务器设置 启动NFS服务器

- 设置有线网卡桥接模式

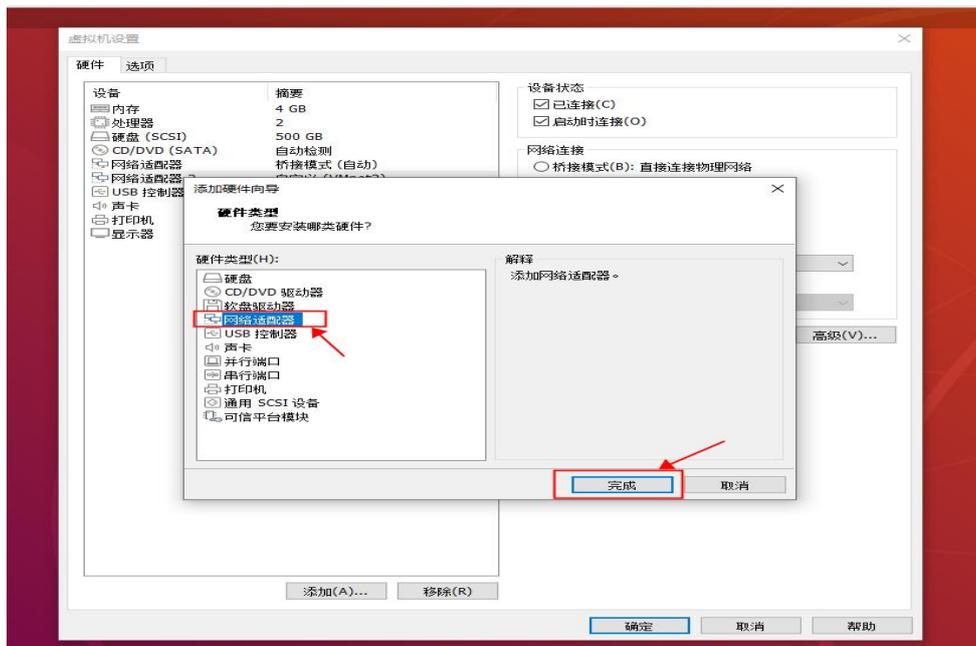
A. 点击虚拟网络编辑器



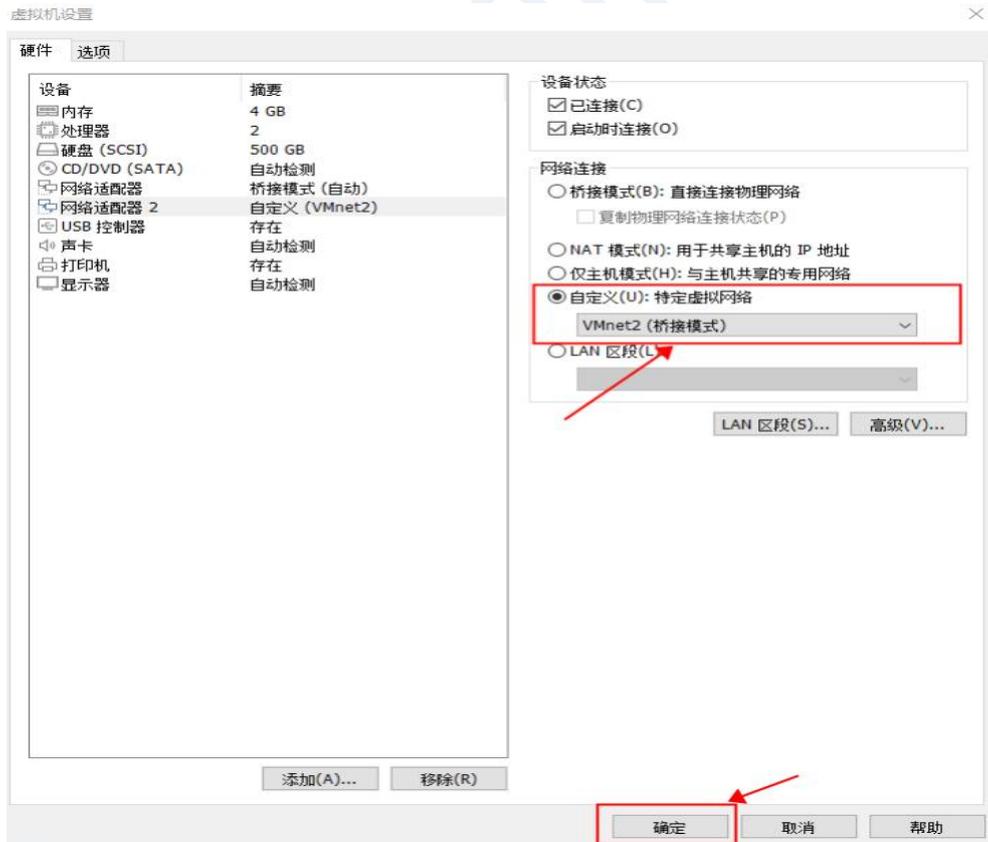
B. 将虚拟网路里面存在的VMnet2，改为有线网络桥接模式/不存在添加网络



C. 点击虚拟机选择设置，点击添加，选择网络适配器，点击完成



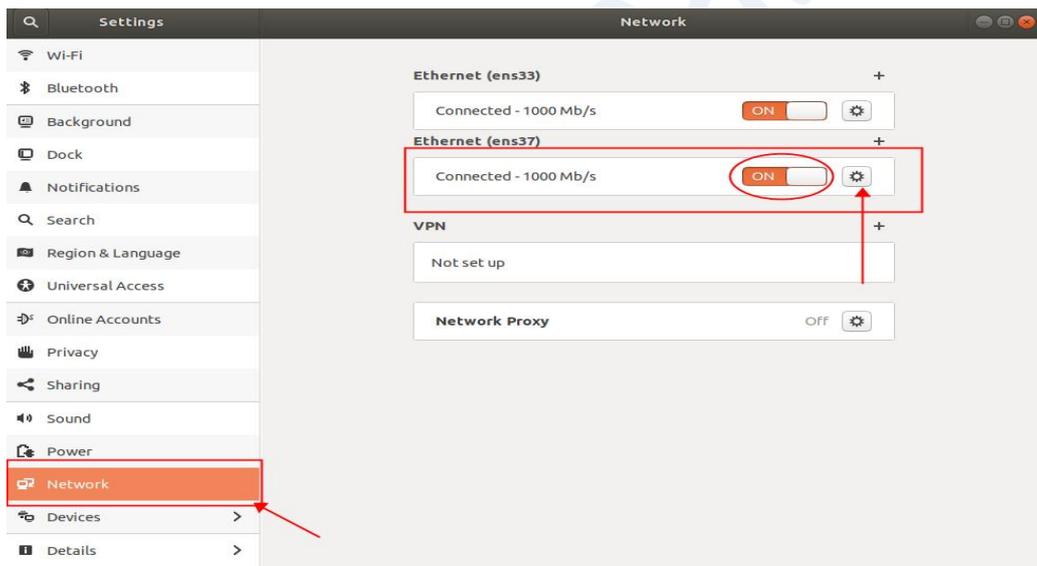
D. 在已经添加的网络适配器中选择自定义，选择VMnet2（桥接模式），或者添加自己设置的其他虚拟网络有线桥接模式，点击确定



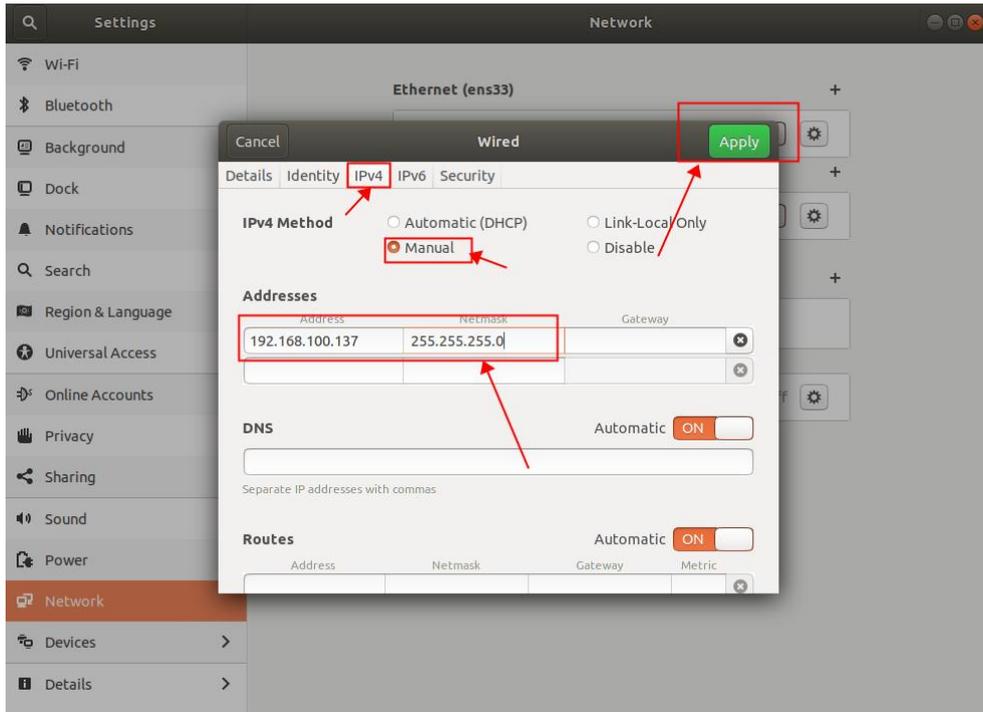
E. 点击ubuntu菜单栏，点击设置选项，选择Network，选择自己设的第二个虚拟网络，滑动打开网络，点击设置



Ps: 如下connected在设置好IP地址后需要on off切换刷新一次，新ip地址才会应用



F. 进入设置选择ipv4，点击Manual，填写设置的ip和Netmask（ip和板端网段192.168.100.*一致，Netmask:255.255.255.0）最后点击Apply



G. 验证设置的桥接虚拟网络 ping 192.168.100.137（ping的是自己设置的桥接的有线网卡的ip），如下图所示，说明验证的网络是通的

- ubuntu18.04安装NFS并启动NFS

A. 终端输入指令：`sudo apt install nfs-kernel-server`

```
Preparing to unpack .../5-nfs-kernel-server_1%3a1.3.4-2.1ubuntu5.5_amd64.deb ...
Unpacking nfs-kernel-server (1:1.3.4-2.1ubuntu5.5) ...
Setting up libnfsidmap2:amd64 (0.25-5.1) ...
Setting up keyutils (1.5.9-9.2ubuntu2) ...
Setting up libtirpc1:amd64 (0.2.5-1.2ubuntu0.1) ...
Setting up rpcbind (0.2.3-0.6ubuntu0.18.04.4) ...
Created symlink /etc/systemd/system/multi-user.target.wants/rpcbind.service → /lib/systemd/system/rpcbind.service.
Created symlink /etc/systemd/system/sockets.target.wants/rpcbind.socket → /lib/systemd/system/rpcbind.socket.
Setting up nfs-common (1:1.3.4-2.1ubuntu5.5) ...

Creating config file /etc/idmapd.conf with new version
Adding system user `statd' (UID 122) ...
Adding new user `statd' (UID 122) with group `nogroup' ...
Not creating home directory `/var/lib/nfs'.
Created symlink /etc/systemd/system/multi-user.target.wants/nfs-client.target → /lib/systemd/system/nfs-client.target.
Created symlink /etc/systemd/system/remote-fs.target.wants/nfs-client.target → /lib/systemd/system/nfs-client.target.
nfs-utils.service is a disabled or a static unit, not starting it.
Setting up nfs-kernel-server (1:1.3.4-2.1ubuntu5.5) ...
Created symlink /etc/systemd/system/multi-user.target.wants/nfs-server.service → /lib/systemd/system/nfs-server.service.
Job for nfs-server.service canceled.

Creating config file /etc/exports with new version

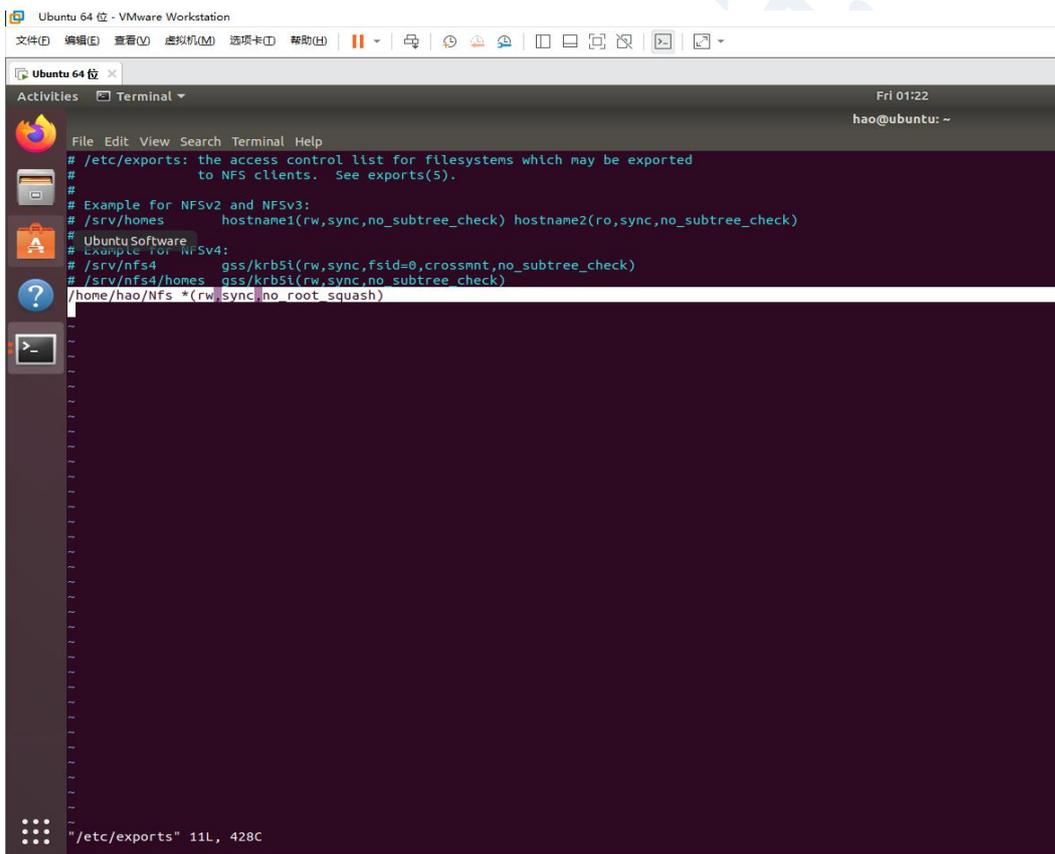
Creating config file /etc/default/nfs-kernel-server with new version
Processing triggers for systemd (237-3ubuntu10.53) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for ureadahead (0.100.0-21) ...
Processing triggers for libc-bin (2.27-3ubuntu1.4) ...
```

B. 在ubuntu 18.04 下创建一个和mont 共享的文件夹 `mkdir Nfs`(文件夹名随意)，然后重启NFS服务器,重启NFS服务器命令：`sudo /etc/init.d/nfs-kernel-server restart`

```
hao@ubuntu:~$
hao@ubuntu:~$
hao@ubuntu:~$
hao@ubuntu:~$ mkdir Nfs
hao@ubuntu:~$
hao@ubuntu:~$
hao@ubuntu:~$ ls
A191 Desktop Documents Downloads Music Nfs Pictures Public Templates Videos
hao@ubuntu:~$
hao@ubuntu:~$
hao@ubuntu:~$
hao@ubuntu:~$
```

```
hao@ubuntu:~$
hao@ubuntu:~$ sudo /etc/init.d/nfs-kernel-server restart
[sudo] password for hao:
[ok] Restarting nfs-kernel-server (via systemctl): nfs-kernel-server.service.
hao@ubuntu:~$
```

C. 修改配置文件 命令行输入 `sudo vim /etc/exports`, 然后在配置文件里面添加 `/home/hao/Nfs *(rw, sync, no_root_squash)`, (这里用到vim命令) 注意: 添加的/home/* (指Ubuntu用户名) /* (指创建的NFS服务器文件名) 根据自己创建的填写, 如下图所示:



```
Ubuntu 64 位 - VMware Workstation
文件(F) 编辑(E) 查看(V) 虚拟机(M) 选项(O) 帮助(H)
Ubuntu 64 位 x
Activities Terminal
Fri 01:22
hao@ubuntu: ~
File Edit View Search Terminal Help
# /etc/exports: the access control list for filesystems which may be exported
# to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes hostname1(rw, sync, no_subtree_check) hostname2(ro, sync, no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4 gss/krb5i(rw, sync, fsid=0, crossmnt, no_subtree_check)
# /srv/nfs4/homes gss/krb5i(rw, sync, no_subtree_check)
/home/hao/Nfs *(rw, sync, no_root_squash)
"/etc/exports" 11L, 428C
```

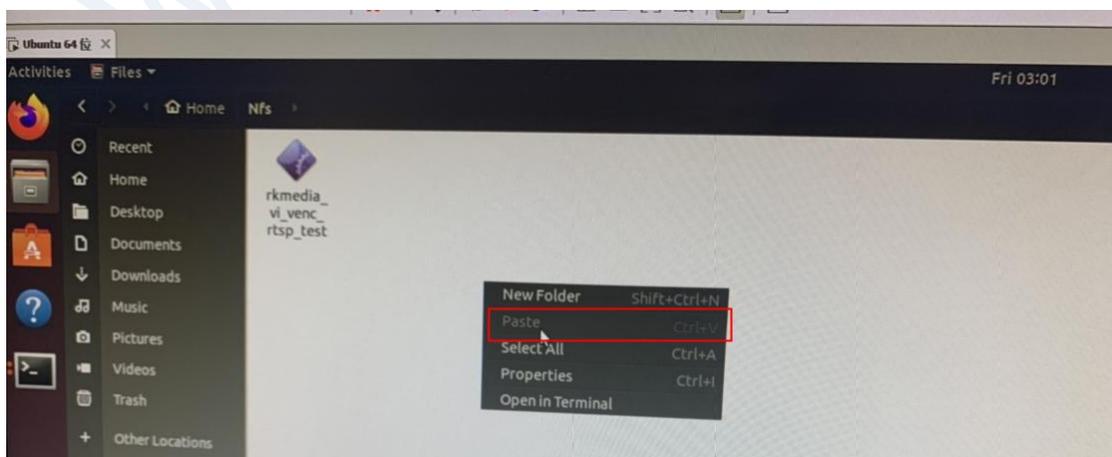
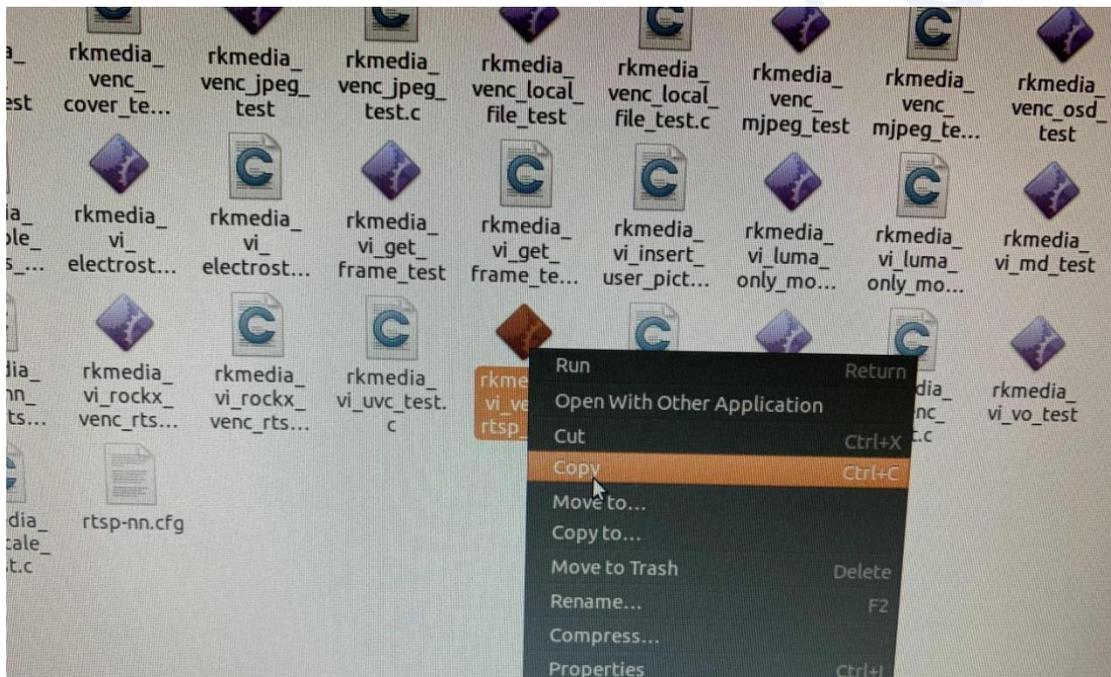
D. 再次重启 `sudo /etc/init.d/nfs-kernel-server restart`，启动NFS服务器 命令行输入 `sudo /etc/init.d/nfs-kernel-server status`

```
hao@ubuntu:~$ sudo /etc/init.d/nfs-kernel-server restart
[ ok ] Restarting nfs-kernel-server (via systemctl): nfs-kernel-server.service.
hao@ubuntu:~$ 
hao@ubuntu:~$ 
hao@ubuntu:~$ sudo /etc/init.d/nfs-kernel-server status
● nfs-server.service - NFS server and services
   Loaded: loaded (/lib/systemd/system/nfs-server.service; enabled; vendor preset: enabled)
   Active: active (exited) since Fri 2022-01-21 04:05:14 PST; 5s ago
     Process: 5643 ExecStopPost=/usr/sbin/exportfs -f (code=exited, status=0/SUCCESS)
     Process: 5642 ExecStopPost=/usr/sbin/exportfs -au (code=exited, status=0/SUCCESS)
     Process: 5641 ExecStop=/usr/sbin/rpc.nfsd 0 (code=exited, status=0/SUCCESS)
     Process: 5651 ExecStart=/usr/sbin/rpc.nfsd $RPCNFSDARGS (code=exited, status=0/SUCCESS)
     Process: 5650 ExecStartPre=/usr/sbin/exportfs -r (code=exited, status=0/SUCCESS)
    Main PID: 5651 (code=exited, status=0/SUCCESS)

Jan 21 04:05:13 ubuntu systemd[1]: Starting NFS server and services...
Jan 21 04:05:13 ubuntu exportfs[5650]: exportfs: /etc/exports [1]: Neither 'subtree_check' or 'no_subtree_check' specified for export "*/home/hao/Nfs".
Jan 21 04:05:13 ubuntu exportfs[5650]: Assuming default behaviour ('no subtree_check').
Jan 21 04:05:13 ubuntu exportfs[5650]: NOTE: this default has changed since nfs-utils version 1.0.x
Jan 21 04:05:14 ubuntu systemd[1]: Started NFS server and services.
hao@ubuntu:~$
```

8.4 可执行文件传给板端

- 将运行的可执行文件拷贝给/home/hao/Nfs（home/*/根据自己创建的）



- 将视频流的可执行文件通过NFS传给板端

A.cd/ , ls 进入ssh 连接显示终端的根目录，如下图所示：

```
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]# ls
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]# cd /
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]# ls
app      data      init      lib32     misc      opt        run        sys        udisk      userdata  var
bin      dev       ko        linuxrc  mnt       proc       sbin       timestamp  tmp        userdata  vendor
busybox.config  etc       lib       media    oem       root       sdcards   tmp        udisk     userdata  vendor
```

B. 在板端根目录创建nfs 文件（文件名随意，此文件夹用来接收共享的文件），如下图所示：

```
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]# mkdir nfs
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]# ls
app      data      init      lib32     misc      opt        run        sys        udisk      userdata  var
bin      dev       ko        linuxrc  mnt       proc       sbin       timestamp  tmp        userdata  vendor
busybox.config  etc       lib       media    nfs       oem       root       sdcards   tmp        udisk     userdata  vendor
```

C. 板端挂载命令：`mount -t nfs 192.168.100.137:/home/hao/Nfs /nfs -o nolock`，挂载成功图如下：

```
[root@RV1126_RV1109:~]# mount -t nfs 192.168.100.137:/home/hao/Nfs /nfs -o nolock
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]# ls
app      data      init      lib32     misc      oem        root       sdcards   tmp        udisk     usr
bin      dev       ko        linuxrc  mnt       nfs        run        sys       timestamp  userdata  var
busybox.config  etc       lib       media    oem       proc       sbin       sdcards   tmp        udisk     userdata  vendor
```

D. 将挂载的文件拷贝给 根目录的app 文件夹 命令行：`cp rkmedia_vi_venc_rtsp_test ../app`，然后进入app 文件夹 看是否拷贝成功

```
[root@RV1126_RV1109:/nfs]# ls
rkmedia_vi_venc_rtsp_test
[root@RV1126_RV1109:/nfs]#
[root@RV1126_RV1109:/nfs]#
[root@RV1126_RV1109:/nfs]#
[root@RV1126_RV1109:/nfs]#
[root@RV1126_RV1109:/nfs]# cp rkmedia_vi_venc_rtsp_test ../app
[root@RV1126_RV1109:/nfs]#
[root@RV1126_RV1109:/nfs]# ls
rkmedia_vi_venc_rtsp_test
[root@RV1126_RV1109:/nfs]#
[root@RV1126_RV1109:/nfs]# cd
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]# cd ../app
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]# ls
local_env  mpi_4]ane_800x480.sh  portttest  quectel-CM  rkmedia_vi_venc_rtsp_test  run.sh
```


E. 最后在进入根目录下的 app 文件，执行取视频流的可执行文件，取到流图如下：

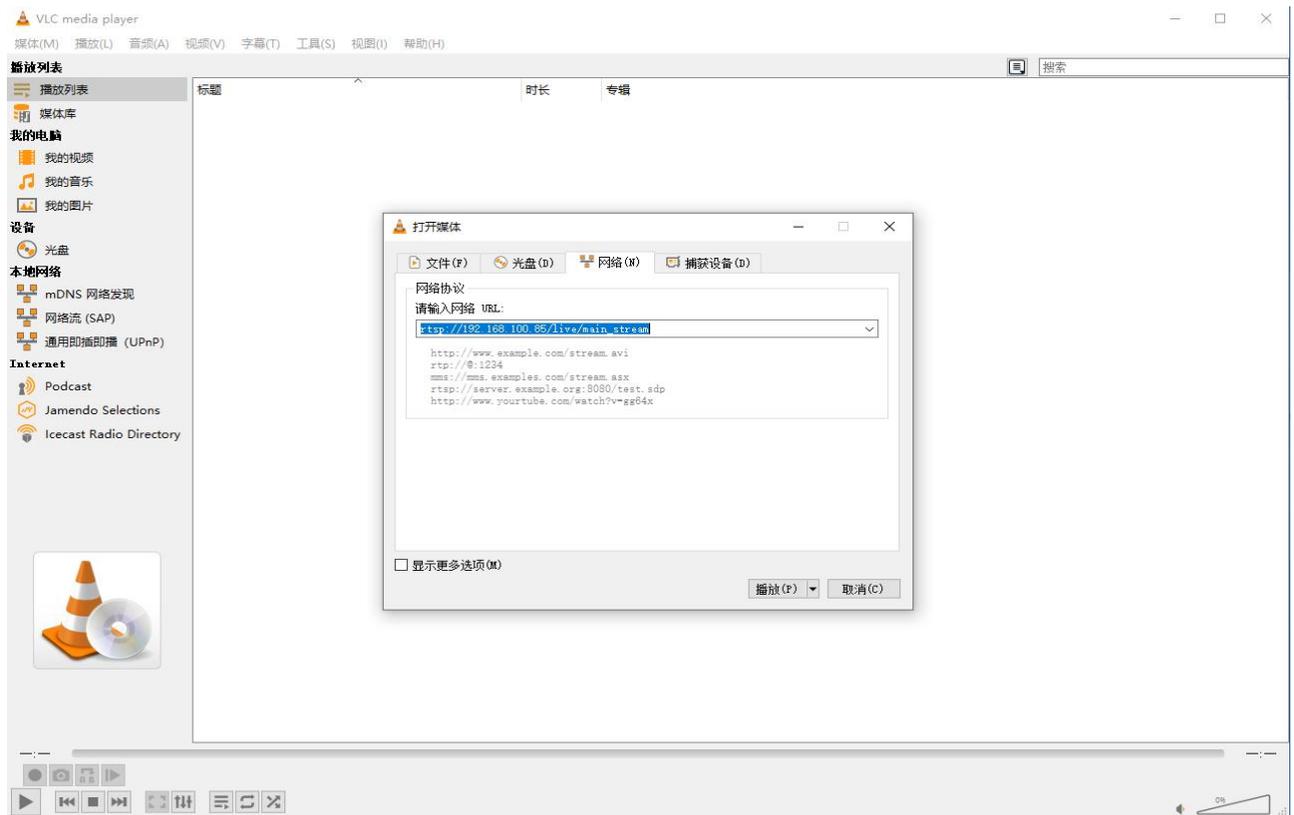
```
[root@RV1126_RV1109:/nfs/lib]#
[root@RV1126_RV1109:/nfs/lib]# cd ../../app
[root@RV1126_RV1109:/app]#
[root@RV1126_RV1109:/app]# ls
global_env  mipi_4lane_800x480.sh  portttest  quectel-CM  rkmedia_vi_venc_rtsp_test  run.sh
[root@RV1126_RV1109:/app]# ./rkmedia_vi_venc_rtsp_test
media get entity by name: rkcif-lvds-subdev is null
media get entity by name: rkcif-lite-lvds-subdev is null
media get entity by name: rkisp-mpfbc-subdev is null
media get entity by name: rkisp-dmaph is null
media get entity by name: rockchip-mipi-dphy-rx is null
rga built version:1.04 +2022-01-18 23:38:32
#Device: rkisp_scale0
#CodecName:H264
#Resolution: 1920x1080
#CameraIdx: 0
#bMultictx: 0
#AiQ xml dirpath: (null)
[INFO rtsp_demo.c:281:rtsp_new_demo] rtsp server demo starting on port 554
[DEBUG rtsp_demo.c:481:rtsp_new_session] add session path: /live/main_stream
##RKMEDIA Log level: 2
[RKMEDIA][SVS][Info]:text is all=2
[RKMEDIA][SVS][Info]:mode is all, log_level is 2
[RKMEDIA][SVS][Info]:RK_MPI_VI_EnableChn: Enable VI[0:0]:rkisp_scale0, 1920x1080 Start...
[RKMEDIA][SVS][Info]:RKAIQ: parsing /dev/media0
media get entity by name: rkcif-lvds-subdev is null
media get entity by name: rkcif-lite-lvds-subdev is null
[RKMEDIA][SVS][Info]:RKAIQ: parsing /dev/media1
media get entity by name: rkisp-mpfbc-subdev is null
media get entity by name: rkisp-dmaph is null
media get entity by name: rockchip-mipi-dphy-rx is null
[RKMEDIA][SVS][Info]:RKAIQ: model(rkisp0): isp-subdev entity name: /dev/v4l-subdev5
[RKMEDIA][SVS][Info]:RKAIQ: parsing /dev/media2
[RKMEDIA][SVS][Info]:RKAIQ: model(rkisp0): isp-subdev entity name: /dev/v4l-subdev0
[RKMEDIA][SVS][Info]:#V4l2Stream: camraId:0, device:rkisp_scale0
```

```
[RKMEDIA][VENC][Info]:MPP Encoder: qpMaxi use default value:48
[RKMEDIA][VENC][Info]:MPP Encoder: qpMini use default value:8
[RKMEDIA][VENC][Info]:MPP Encoder: qpMax use default value:48
[RKMEDIA][VENC][Info]:MPP Encoder: qpMin use default value:8
[RKMEDIA][VENC][Info]:MPP Encoder: qpInit use default value:-1
[RKMEDIA][VENC][Info]:MPP Encoder: qpStep use default value:2
[RKMEDIA][VENC][Info]:MPP Encoder: rotation = 0
[RKMEDIA][VENC][Info]:MPP Encoder: automatically calculate bsp with bps_target
[RKMEDIA][VENC][Info]:MPP Encoder: Set output block mode.
[RKMEDIA][VENC][Info]:MPP Encoder: Set input block mode.
[RKMEDIA][VENC][Info]:MPP Encoder: bps:[2304000,2073600,1843200] fps: [30/1]->[30/1], gop:30 qpInit:-1, qpMin:8, qpMax:48, qpMinI:8
[RKMEDIA][VENC][Info]:MPP Encoder: AVC: encode profile 77 level 0
mpp[2784]: mpp_enc: MPP_ENC_SET_RC_CFG bps 2073600 [1843200 : 2304000] fps [30:30] gop 30
mpp[2784]: h264e_ap1_v2: MPP_ENC_SET_PREP_CFG w:h [1920:1080] stride [1920:1080]
mpp[2784]: mpp_enc: send header for set cfg change input/format
[RKMEDIA][VENC][Info]:MPP Encoder: w x h(1920[1920] x 1080[1080])
mpp[2784]: mpp_enc: mode vbr bps [1843200:2073600:2304000] fps fix [30/1] -> fix [30/1] gop i [30] v [0]
[RKMEDIA][SVS][Info]:RK_MPI_VENC_CreateChn: Enable VENC[0], type:6 End.
[RKMEDIA][SVS][Info]:RK_MPI_SYS_Bind: Bind Mode[VI]:chn[0] to Mode[VENC]:chn[0]...
main initial finish
mpp[2784]: h264e_sps: set level to 4
#Get packet-0, size 81754
[DEBUG utils.c:160:rtsp_codec_data_parse_from_user_h264] sps 26
[DEBUG utils.c:168:rtsp_codec_data_parse_from_user_h264] pps 4
#Get packet-1, size 5947
#Get packet-2, size 6322
#Get packet-3, size 6423
#Get packet-4, size 8688
#Get packet-5, size 1196
#Get packet-6, size 12232
#Get packet-7, size 4782
#Get packet-8, size 9961
#Get packet-9, size 624
#Get packet-10, size 8836
#Get packet-11, size 6558
#Get packet-12, size 10178
#Get packet-13, size 1641
#Get packet-14, size 7767
#Get packet-15, size 2686
#Get packet-16, size 6070
#Get packet-17, size 2728
#Get packet-18, size 9156
#Get packet-19, size 1697
#Get packet-20, size 1511
#Get packet-21, size 2475
#Get packet-22, size 2286
#Get packet-23, size 1884
#Get packet-24, size 4092
#Get packet-25, size 3112
#Get packet-26, size 4919
#Get packet-27, size 3875
#Get packet-28, size 5254
#Get packet-29, size 5081
#Get packet-30, size 31216
#Get packet-31, size 970
#Get packet-32, size 1989
#Get packet-33, size 3247
#Get packet-34, size 4296
#Get packet-35, size 3997
#Get packet-36, size 4870
#Get packet-37, size 4946
#Get packet-38, size 5090
#Get packet-39, size 6688
#Get packet-40, size 6666
#Get packet-41, size 8181
#Get packet-42, size 8032
#Get packet-43, size 7938
```

9 适配Sensor

9.1 GC2053摄像头访问网络码流

板端与适配的摄像头连接好以后，打开电脑上 VLC 软件，点击“媒体”，选择“打开网络串流”，将下图中的流地址（rtsp://192.168.100.85/live/main_stream）填入，然后点击播放就可以看到图像了。



注意事项：

1. 只有在板端与适配的摄像头连接良好的情况下烧入固件，板端识别到摄像头后，IP会自动为192.168.100.85，在此基础上修改ip是可以通过VLC拉到流的。其他情况是拉不到流的。

2. 拉流时必须是在执行 `./rkmedia_vi_cenc_rtsp_test` 命令成功后一直在窗口打印 `packet` 行命令再通过VLC拉流，不能中断窗口打印命令。

- 将IMX415 取消注释，去掉 /* */ 如下图所示：

```
&i2c1 {
    status = "okay";
    clock-frequency = <400000>;

    imx415: imx415@1a {
        compatible = "sony,imx415";
        reg = <0x1a>;
        clocks = <&cru CLK_MIPICSI_OUT>;
        clock-names = "xvclk";
        power-domains = <&power RV1126_PD_VI>;
        pinctrl-names = "rockchip, camera_default";
        pinctrl-0 = <&mipiccsi_clk0>;
        avdd-supply = <&vcc3v3_sys>;
        dovdd-supply = <&vcc_1v8>;
        dvdd-supply = <&vcc_dvdd>;
        reset-gpios = <&gpio1 RK_PD5 GPIO_ACTIVE_LOW>;
        rockchip,camera-module-index = <1>;
        rockchip,camera-module-facing = "front";
        rockchip,camera-module-name = "YT10092";
        rockchip,camera-module-lens-name = "IR0147-28IRC-8M-F20";
        ir-cut = <&cam_ircut0>;
        flash-leds = <&flash_ir>;
        port {
            ucaml_out0: endpoint {
                remote-endpoint = <&mipi_in_ucam0>;
                data-lanes = <1 2 3 4>;
            };
        };
    };
};
```

- 将dts里面的 data-lanes = <1,2> 注释掉，最后wq 保存退出，如下图所示：

```
&csi_dphy0 {
    status = "okay";

    ports {
        #address-cells = <1>;
        #size-cells = <0>;
        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;

            mipi_in_ucam0: endpoint@1 {
                reg = <1>;
                remote-endpoint = <&ucaml_out0>;
                data-lanes = <1 2 3 4>;
                // data-lanes = <1 2>; // only for GC2053
            };
        };
        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;

            csidphy0_out: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&mipi_csi2_input>;
            };
        };
    };
};
```

```

};
&ndio {
  phy: phy0 {
    compatible = "ethernet-phy-ieee802.3-c22";
    reg = <0x0>;
  };
};
&mpi_csi2 {
  status = "okay";
  ports {
    #address-cells = <1>;
    #size-cells = <0>;
    port@0 {
      reg = <0>;
      #address-cells = <1>;
      #size-cells = <0>;
      mipi_csi2_input: endpoint@1 {
        reg = <1>;
        remote-endpoint = <=&csi_dphy0_out>;
        data-lanes = <1 2 3 4>;
        // data-lanes = <1 2>; // only for GC2053
      };
    };
    port@1 {
      reg = <1>;
      #address-cells = <1>;
      #size-cells = <0>;
      mipi_csi2_output: endpoint@0 {
        reg = <0>;
        remote-endpoint = <=&cif_mipi_in>;
        data-lanes = <1 2 3 4>;
        // data-lanes = <1 2>; // only for GC2053
      };
    };
  };
};
&mpu {
  /mipi_csi2_input

```

```

&csi_dphy0 {
  status = "okay";
  ports {
    #address-cells = <1>;
    #size-cells = <0>;
    port@0 {
      reg = <0>;
      #address-cells = <1>;
      #size-cells = <0>;
      mipi_in_ucam0: endpoint@1 {
        reg = <1>;
        remote-endpoint = <=&ucam_out0>;
        data-lanes = <1 2 3 4>;
        // data-lanes = <1 2>; // only for GC2053
      };
    };
    port@1 {
      reg = <1>;
      #address-cells = <1>;
      #size-cells = <0>;
      csidphy0_out: endpoint@0 {
        reg = <0>;
        remote-endpoint = <=&mipi_csi2_input>;
      };
    };
  };
};

```

```

};
&ndio {
  phy: phy0 {
    compatible = "ethernet-phy-ieee802.3-c22";
    reg = <0x0>;
  };
};
&mpi_csi2 {
  status = "okay";
  ports {
    #address-cells = <1>;
    #size-cells = <0>;
    port@0 {
      reg = <0>;
      #address-cells = <1>;
      #size-cells = <0>;
      mipi_csi2_input: endpoint@1 {
        reg = <1>;
        remote-endpoint = <=&csi_dphy0_out>;
        data-lanes = <1 2 3 4>;
        // data-lanes = <1 2>; // only for GC2053
      };
    };
    port@1 {
      reg = <1>;
      #address-cells = <1>;
      #size-cells = <0>;
      mipi_csi2_output: endpoint@0 {
        reg = <0>;
        remote-endpoint = <=&cif_mipi_in>;
        data-lanes = <1 2 3 4>;
        // data-lanes = <1 2>; // only for GC2053
      };
    };
  };
};
&show Applications
/mipi_csi2_input

```

```

port {
    /* MIPI CSI-2 endpoint */
    cif_mipi_in: endpoint {
        remote-endpoint = <&mipi_csi2_output>;
        data-lanes = <1 2 3 4>;
        // data-lanes = <1 2>; // only for GC2053
    };
};

&rkCIF_mipi_lvds_sditf {
    status = "okay";

    port {
        /* MIPI CSI-2 endpoint */
        mipi_lvds_sditf: endpoint {
            remote-endpoint = <&isp_in>;
            data-lanes = <1 2 3 4>;
            // data-lanes = <1 2>; // only for GC2053
        };
    };
};

&rkisp_vir0 {

```

- 命令行退到sdk 目录，再次编译总体编译固件（根据目录5SDK 编译说明），然后烧写固件(同目录6)如下图所示操作：

```

hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224/kernel/arch/arm/boot/dts$
hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224/kernel/arch/arm/boot/dts$ cd ..
hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224/kernel/arch/arm/boot$
hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224/kernel/arch/arm/boot$
hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224/kernel/arch/arm/boot$ cd ..
hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224/kernel/arch/arm$
hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224/kernel/arch/arm$
hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224/kernel/arch/arm$ cd ..
hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224/kernel/arch$
hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224/kernel/arch$
hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224/kernel/arch$ cd ..
hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224/kernel$
hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224/kernel$
hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224/kernel$ ls
abi_gki_aarch64_cuttlefish_whitelist  build.config.allmodconfig          build.config.gki                  COPYING                           fs
abi_gki_aarch64_qcom_whitelist        build.config.allmodconfig.aarch64  build.config.gki.aarch64         CREDITS                           includ
abi_gki_aarch64_whitelist             build.config.allmodconfig.arm       build.config.gki-debug.aarch64   crypto                             init
abi_gki_aarch64.xml                   build.config.allmodconfig.x86_64   build.config.gki-debug.x86_64   cuttlefish.fragment               ipc
arch                                    build.config.arm                    build.config.gki.x86_64          defconfig                           Kbuild
block                                   build.config.common                 build.config.x86_64              Documentation                       Kconfi
boot.img                                build.config.cuttlefish.aarch64    built-in.a                       drivers                             kernel
build.config.aarch64                   build.config.cuttlefish.x86_64     certs                             firmware                             kernel
hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224/kernel$
hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224/kernel$ cd ..
hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224$
hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224$
hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224$
hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224$
hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224$ ls
app          buildroot  device  envsetup.sh  IMAGE  kernel.git  mkfirmware.sh  prebuilts  rkbin      rockdev  u-boot
br.log  build.sh  docs    external    kernel  Makefile    packed-refsu-boot.git  release    rkflash.sh  tools
hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224$
hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224$

```



```
hao@ubuntu:~/A191/rv1126_rv1109_linux_sdk_v1.8.0_20210224$ ./build.sh
processing board option: allsave
processing option: allsave
=====
TARGET_ARCH=arm
TARGET_PLATFORM=rv1126_rv1109
TARGET_UBOOT_CONFIG=rv1126
TARGET_SPL_CONFIG=
TARGET_KERNEL_CONFIG=rv1126_defconfig
TARGET_KERNEL_DTS=rv1109-38-v10-spi-nand
TARGET_TOOLCHAIN_CONFIG=
TARGET_BUILDRoot_CONFIG=rockchip_rv1126_rv1109_spi_nand
TARGET_RECOVERY_CONFIG=rockchip_rv1126_rv1109_spi_nand_recovery
TARGET_PCBA_CONFIG=
TARGET_RAMBOOT_CONFIG=
=====
=====Start building uboot=====
TARGET_UBOOT_CONFIG=rv1126
=====
#
# configuration written to .config
#
Using .config as base
Merging ./arch/./configs/rk-sfc.config
Value of CONFIG_ROCKCHIP_SFC_IOMUX is redefined by fragment ./arch/./configs/rk-sfc.config:
Previous value: # CONFIG_ROCKCHIP_SFC_IOMUX is not set
New value: CONFIG_ROCKCHIP_SFC_IOMUX=y
#
# passed configuration written to .config (needs make)
```

- 然后 SSH 连接，连接成功后 `dmesg | grep 415` 查看是否有设备，如下图所示：

```
root@rv1126_rv1109:/app#
root@rv1126_rv1109:/app# dmesg | grep 415
0.414152] of_get_named_gpiod_flags: can't parse 'vsirq-gpios' property of node '/rkisp-vir0[0]'
0.607250] imx415 1-001a: driver version: 00.01.06
0.607273] imx415 1-001a: Get hdr mode failed! no hdr default
0.607292] imx415 1-001a: GPIO lookup for consumer reset
0.607296] imx415 1-001a: using device tree for GPIO lookup
0.607313] of_get_named_gpiod_flags: parsed 'reset-gpios' property of node '/i2c@ff510000/imx415@1a[0]' - status (0)
0.607337] imx415 1-001a: GPIO lookup for consumer power
0.607341] imx415 1-001a: using device tree for GPIO lookup
0.607350] of_get_named_gpiod_flags: can't parse 'power-gpios' property of node '/i2c@ff510000/imx415@1a[0]'
0.607360] of_get_named_gpiod_flags: can't parse 'power-gpio' property of node '/i2c@ff510000/imx415@1a[0]'
0.607366] imx415 1-001a: using lookup tables for GPIO lookup
0.607370] imx415 1-001a: No GPIO consumer power found
0.607374] imx415 1-001a: Failed to get power-gpios
0.607410] imx415 1-001a: could not get sleep pinstate
0.607484] imx415 1-001a: Linked as a consumer to regulator.3
0.607538] imx415 1-001a: Linked as a consumer to regulator.2
0.607587] imx415 1-001a: Linked as a consumer to regulator.4
0.676452] imx415 1-001a: Detected imx415 id 0000e0
0.676598] rockchip-mipi-dphy-rx ff4b0000.csi-dphy: match m01_f_imx415 1-001a:bus type 4
16.682710] imx415 1-001a: set fmt: cur_mode: 3864x2192, hdr: 0
16.763688] imx415 1-001a: set vblank 0x1fc vts 2700
16.764482] imx415 1-001a: set exposure(shr0) 2498 = cur_vts(2700) - val(202)
root@rv1126_rv1109:/app#
```

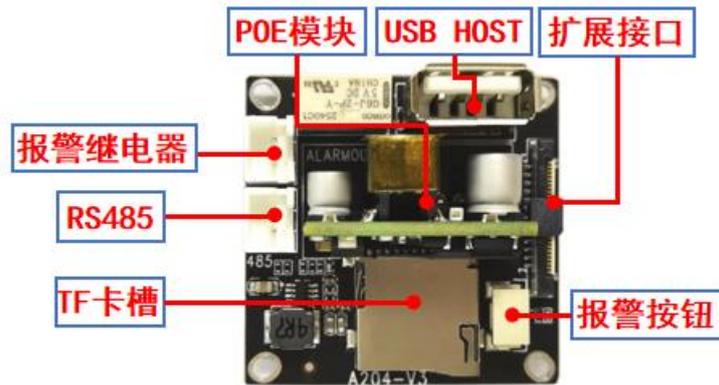
- 再将视频流的可执行文件通过 NFS 传给板子（操作跟目录9 一样），run 视频流的测试执行文件缺少库，可根据（目录9.6）来完成
- 最终取出视频流

10 扩展子板的功能验证

10.1 扩展子板连接

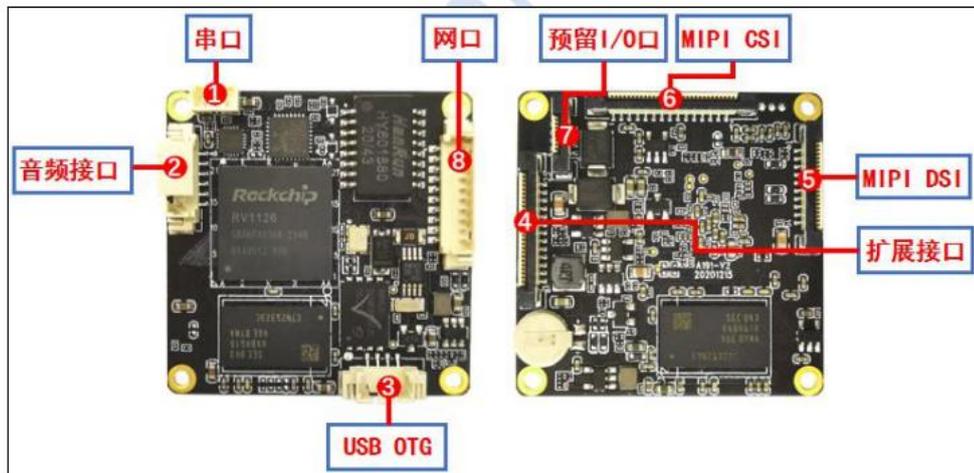
10.1.1 扩展子板功能介绍

此板卡是为适配 EB-RV1126-BC-191型整板而开发的的扩展板，扩展板具备 TF 卡、POE 模块、USB、报警等功能于一体，性能较稳定。



10.1.2 扩展子板连接

把 A204 型扩展板和 EB-RV1126-BC-191型整板的外围接口均做了如下面两图的说明，两者仅需将各自“扩展接口”接口相连接即可，连接使用 26pin 同向/FPC 软排线。



10.1.3 前提软硬件准备

拓展子板连接好后，固件烧写与远程连接同上述步骤目录6、7。

10.2 TF卡功能

10.2.1 检查TF卡识别是否成功

- 首先确认TF卡槽内已插好TF卡
- 进入板端输入 `fdisk -l` 命令打印出SD卡的系统信息成功，如下图所示：

```
192.168.100.88 x
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]# fdisk -l
Disk /dev/mmcblk2: 942 MB, 988282880 bytes, 1930240 sectors
2218 cylinders, 30 heads, 29 sectors/track
units: cylinders of 870 * 512 = 445440 bytes

Device            Boot StartCHS   EndCHS       StartLBA     EndLBA       Sectors  Size
  Id Type
/dev/mmcblk2p1    0,3,61     986,29,29     249         1930239     1929991  942M
 6 FAT16
[root@RV1126_RV1109:~]#
```

10.2.2 TF卡测速

- 测试写入速度：

```
time dd if=/dev/zero of=/mnt/sdcard/1.bin bs=1M count=100
```

```
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]# time dd if=/dev/zero of=/mnt/sdcard/1.bin bs=1M count=90
90+0 records in
90+0 records out
94371840 bytes (94 MB, 90 MiB) copied, 1.33277 s, 70.8 MB/s
real    0m 3.24s
user    0m 0.00s
sys     0m 3.16s
[root@RV1126_RV1109:~]# █
```

- 测试读取速度：

```
time dd if=/mnt/TF/1.bin of=/dev/null bs=10M
```

```
[root@RV1126_RV1109:~]# time dd if=/mnt/sdcard/1.bin of=/dev/null bs=10M
9+0 records in
9+0 records out
94371840 bytes (94 MB, 90 MiB) copied, 0.242414 s, 389 MB/s
real    0m 0.25s
user    0m 0.00s
sys     0m 0.25s
[root@RV1126_RV1109:~]#
```

10.3 WiFi模块

10.3.1 测试有无识别到WiFi网卡

- 在板端输入 `ifconfig -a`, 查看wifi节点
- 如果没有wlan0信息，输入：`ifconfig wlan0 up`, 再ifconfig查看信息

```
[root@RV1126_RV1109:/mnt]# ifconfig -a
eth0      Link encap:Ethernet  Hwaddr 5E:2A:FA:EE:8E:36
          inet addr:192.168.100.68  Bcast:192.168.100.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1459  errors:0  dropped:0  overruns:0  frame:0
          TX packets:637  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:1000
          RX bytes:1591249 (1.5 MiB)  TX bytes:108596 (106.0 KiB)
          Interrupt:60

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:4  errors:0  dropped:0  overruns:0  frame:0
          TX packets:4  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:1000
          RX bytes:292 (292.0 B)  TX bytes:292 (292.0 B)

wlan0     Link encap:Ethernet  Hwaddr 00:13:EF:F8:22:D6
          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)
```

10.3.2 wifi关闭功能验证

- 输入：`ifconfig wlan0 down`

```
[root@RV1126_RV1109:/sys/class]# cd ~
[root@RV1126_RV1109:/oem]# ifconfig wlan0 down
[root@RV1126_RV1109:/oem]# ifconfig
eth0      Link encap:Ethernet  Hwaddr 5E:2A:FA:EE:8E:36
          inet addr:192.168.100.68  Bcast:192.168.100.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2585  errors:0  dropped:0  overruns:0  frame:0
          TX packets:59168  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:1000
          RX bytes:1694465 (1.6 MiB)  TX bytes:33612802 (32.0 MiB)
          Interrupt:60

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:58  errors:0  dropped:0  overruns:0  frame:0
          TX packets:58  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:1000
          RX bytes:7628 (7.4 KiB)  TX bytes:7628 (7.4 KiB)
```

10.3.3 测试和连接wifi

- 第一步：配置文件内输入无线网账号密码，路径：`/etc/wpa_supplicant.conf`，在`/etc`目录下输入：`vi /etc/wpa_supplicant.conf`进入文件内添加无线网账号和密码

```
[root@RV1126_RV1109:~]# cat /etc/wpa_supplicant.conf
network={
    ssid="Ebaina"
    psk="86112366"
    key_mgmt=WPA-PSK
}
```

- 第二步：创建一个socket通信目录 `mkdir -p /var/run/wpa_supplicant`

- 第三步：测试和连接wifi

输入：`wpa_supplicant -dd -Dwext -iwlan0 -c /etc/wpa_supplicant.conf`

窗口持续打印，另拷贝起一个窗口界面进行后续操作，此窗口保留运行界面

```
[root@RV1126_RV1109:~/etc]#
[root@RV1126_RV1109:~/etc]#
[root@RV1126_RV1109:~/etc]#
[root@RV1126_RV1109:~/etc]#
[root@RV1126_RV1109:~/etc]# wpa_supplicant -dd -Dwext -iwlan0 -c /etc/wpa_supplicant.conf
wpa_supplicant v2.6
random: Trying to read entropy from /dev/random
Successfully initialized wpa_supplicant
Initializing interface 'wlan0' conf '/etc/wpa_supplicant.conf' driver 'wext' ctrl_interface 'N/A' bridge 'N/A'
Configuration file '/etc/wpa_supplicant.conf' -> /etc/wpa_supplicant.conf
Reading configuration file '/etc/wpa_supplicant.conf'
ctrl_interface='/var/run/wpa_supplicant'
ap_scan=1
update_config=1
Line: 5 - start of a new network block
ssid - hexdump_ascii(len=6):
 45 62 61 69 6e 61                               Ebaina
PSK (ASCII passphrase) - hexdump_ascii(len=8): [REMOVED]
key_mgmt: 0x2
PSK (from passphrase) - hexdump(len=32): [REMOVED]
Priority group 0
  id=0 ssid='Ebaina'
rfkill: Cannot get wiphy information
WEXT: RFKILL status not available
SIOCSIWRANGE: WE(compiled)=22 WE(source)=16 enc_capa=0xf
capabilities: key_mgmt 0xf enc 0x1f flags 0x0
ioctl[SIOCSIWAP]: Operation not permitted
WEXT: Failed to clear BSSID selection on disconnect
WEXT: Driver: rtl8188eu
```

Ps: `wpa_cli`的使用

运行`wpa_supplicant`，`wpa_supplicant`程序作为`wpa_cli`的服务端，必须先启动后，才能给`wpa_cli`访问使用。`wpa_supplicant`作为服务端已经运行，此时再启动`wpa_cli`客户端，即可通过`wpa_cli`客户端进行网络配置的操作。

- 第四步：扫描周边的AP并把扫描结果显示

输入如下命令：`wpa_cli -i wlan0 -p /var/run/wpa_supplicant scan`

`wpa_cli -i wlan0 -p /var/run/wpa_supplicant scan_results`

结果如下图所示：

```
192.168.100.68 x 192.168.100.68 (1) 192.168.100.68 (2)
[root@RV1126_RV1109:~]# wpa_cli -i wlan0 -p /var/run/wpa_supplicant scan
OK
[root@RV1126_RV1109:~]# wpa_cli -i wlan0 -p /var/run/wpa_supplicant scan_results
bssid / frequency / signal level / flags / ssid
24:2e:02:98:6b:cc 2462 -81 [WPA-PSK-CCMP] [WPA2-PSK-CCMP] [WPS] [ESS] Ebaina
88:44:77:cc:8e:25 2462 -33 [WPA2-PSK-CCMP] [ESS]
88:44:77:cc:8e:24 2462 -35 [WPA2-PSK-CCMP] [WPS] [ESS] HUAWEI-Z6E55Q
b4:f1:8c:69:73:fc 2462 -43 [WPA-PSK-CCMP] [WPA2-PSK-CCMP] [WPS] [ESS] ebaina666-5G
b4:f1:8c:69:74:01 2462 -45 [WPA2-PSK-CCMP] [WPS] [ESS]
dc:d8:7c:24:d6:b5 2437 -47 [WPA-PSK-CCMP] [WPA2-PSK-CCMP] [ESS] Apnringwifi_10000
06:be:d5:dc:40:67 2412 -58 [WPA-PSK-CCMP] [WPA2-PSK-CCMP] [ESS] A207
30:fb:b8:82:d1:78 2437 -61 [WPA-PSK-CCMP+TKIP] [WPA2-PSK-CCMP+TKIP] [WPS] [ESS] ChinaNet-w5Zh
34:6b:5b:8c:da:90 2437 -63 [WPA-PSK-CCMP+TKIP] [WPA2-PSK-CCMP+TKIP] [ESS] NUA109
38:54:9b:44:16:40 2442 -47 [WPA-PSK-CCMP+TKIP] [WPA2-PSK-CCMP+TKIP] [ESS] ChinaNet-Apnring
b0:eb:57:a2:bb:75 2437 -67 [WPA2-PSK-CCMP] [ESS]
0c:4b:54:15:ac:45 2462 -69 [WPA-PSK-CCMP] [WPA2-PSK-CCMP] [ESS] EmbedSoft
dc:fe:18:dc:5c:23 2462 -71 [WPA-PSK-CCMP] [WPA2-PSK-CCMP] [ESS] AIRLYNC-01
0c:4b:54:33:5c:3d 2412 -79 [WPA-PSK-CCMP] [WPA2-PSK-CCMP] [ESS] TP-LINK_office
24:cf:24:55:62:06 2417 -69 [WPA-PSK-CCMP+TKIP] [WPA2-PSK-CCMP+TKIP] [WPS] [ESS] Xiaomi_6205
3c:46:d8:65:c1:e8 2412 -75 [WPA-PSK-CCMP] [WPA2-PSK-CCMP] [ESS] HUAWEI_B311_AB41
32:42:40:be:e0:02 2457 -80 [WPA-PSK-CCMP] [WPA2-PSK-CCMP] [ESS]
34:6b:5b:8d:55:10 2462 -79 [WPA-PSK-CCMP+TKIP] [WPA2-PSK-CCMP+TKIP] [ESS] zhongju-1
38:54:9b:44:79:eb 2452 -75 [WPA-PSK-CCMP+TKIP] [WPA2-PSK-CCMP+TKIP] [ESS] ChinaNet-wLzS
24:11:45:b9:4e:48 2462 -83 [WPA2-PSK-CCMP] [ESS] Redmi
24:2e:02:98:6b:d1 2462 -85 [WPA2-PSK-CCMP] [ESS]
fc:e3:3c:4e:6a:f4 2417 -83 [WPA-PSK-CCMP+TKIP] [ESS] ChinaNet-1EK5
88:bc:c1:08:96:80 2437 -85 [WPA-PSK-CCMP+TKIP] [WPA2-PSK-CCMP+TKIP] [WPS] [ESS] ChinaNet-FDfh
a4:c7:4b:25:44:44 2462 -89 [WPA-PSK-CCMP] [WPA2-PSK-CCMP] [WPS] [ESS] zsw723
b0:eb:57:a2:bb:71 2437 -95 [WPA2-PSK-CCMP] [ESS]
b4:f1:8c:69:73:fd 2462 -43 [WPS] [ESS]
00:be:d5:dc:40:67 2412 -59 [ESS]
40:77:a9:64:e4:69 2412 -75 [ESS]
40:77:a9:64:e1:fd 2412 -83 [ESS]
```

- 第五步：wifi连接AP功能验证

检查wifi连接是否成功，输入：`wpa_cli -i wlan0 status`

```
AC
[root@RV1126_RV1109:~]# wpa_cli -i wlan0 status
bssid=24:2e:02:98:6b:cc
freq=0
ssid=Ebaina
id=0
mode=station
pairwise_cipher=CCMP
group_cipher=CCMP
key_mgmt=WPA2-PSK
wpa_state=COMPLETED
address=00:13:ef:f8:22:d6
uuid=d3b2b6ca-7e51-5bc8-966a-fff3785b9894
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
```

自动分配地址，输入：`udhcpd -i wlan0`

```
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]# udhcpd -i wlan0
udhcpd: started, v1.27.2
udhcpd: sending discover
udhcpd: sending discover
udhcpd: sending select for 192.168.100.198
udhcpd: sending select for 192.168.100.198
udhcpd: sending select for 192.168.100.198
udhcpd: lease of 192.168.100.198 obtained, lease time 86400
deleting routers
adding dns 192.168.100.1
```

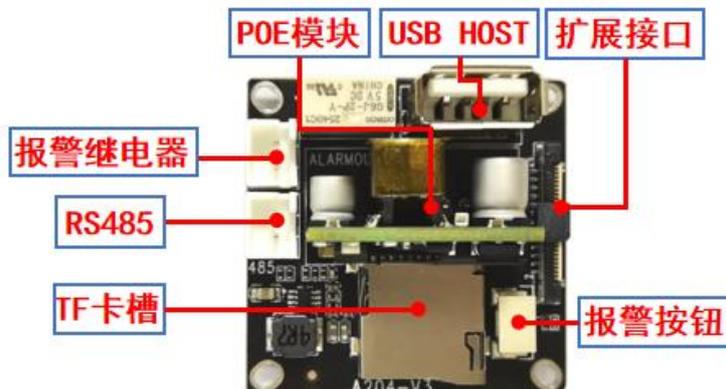
使用ping命令查看网络，可ping180.76.76.76

```
collisions:0 txqueuelen:1000
RX bytes:9017 (8.8 KiB) TX bytes:2138 (2.0 KiB)
[root@RV1126_RV1109:~]# ping -i wlan0 192.168.100.1
PING 192.168.100.1 (192.168.100.1): 56 data bytes
64 bytes from 192.168.100.1: seq=0 ttl=64 time=198.874 ms
64 bytes from 192.168.100.1: seq=1 ttl=64 time=30.553 ms
64 bytes from 192.168.100.1: seq=2 ttl=64 time=301.083 ms
64 bytes from 192.168.100.1: seq=3 ttl=64 time=67.523 ms
AC
--- 192.168.100.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
```

10.4 RS485

10.4.1扩展子板硬件介绍及原理图

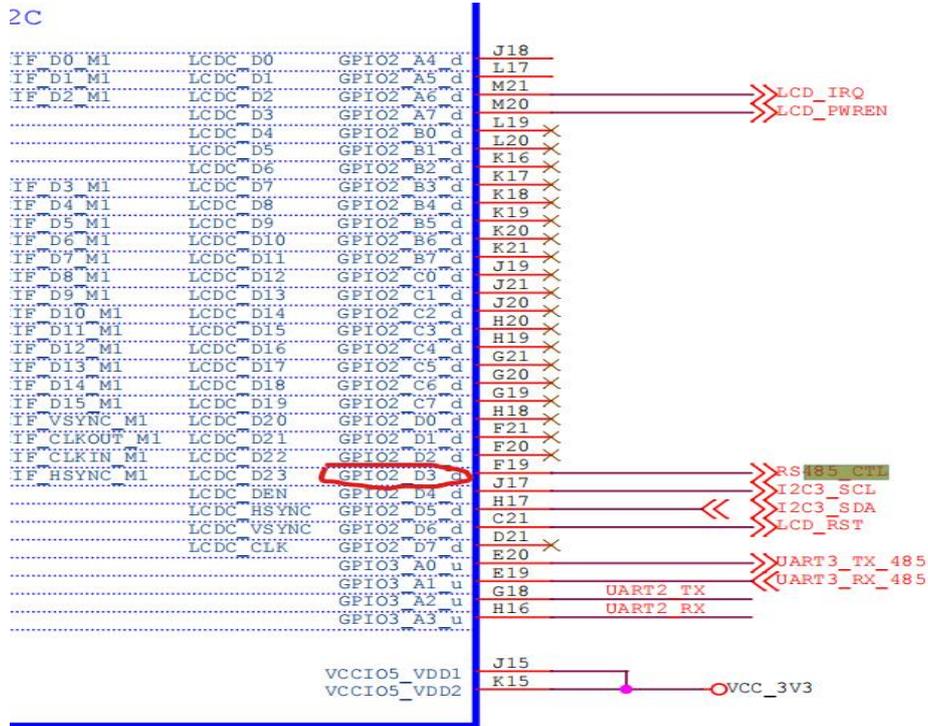
仔细确认RS485接口位置，后续硬件连接时参照如下图：



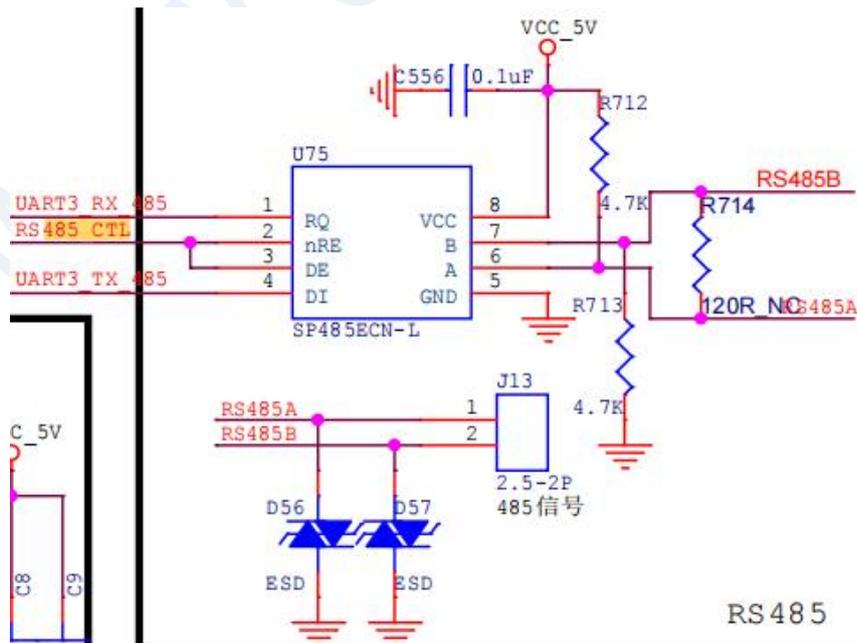
RS485_CTL控制脚对应的RV1126上的IO是GPIO2_D3，其硬件原理图如下（来源于EB-RV1126-BC-191型整板原理图），根据如下图可计算出GPIO2_D3的值：

公式为 $GPIOX_YZ = X*32 + (Y-A)*8 + Z$

如下： $GPIO2_D3_d = 2*32 + (4-1)*8 + 3 + 91$



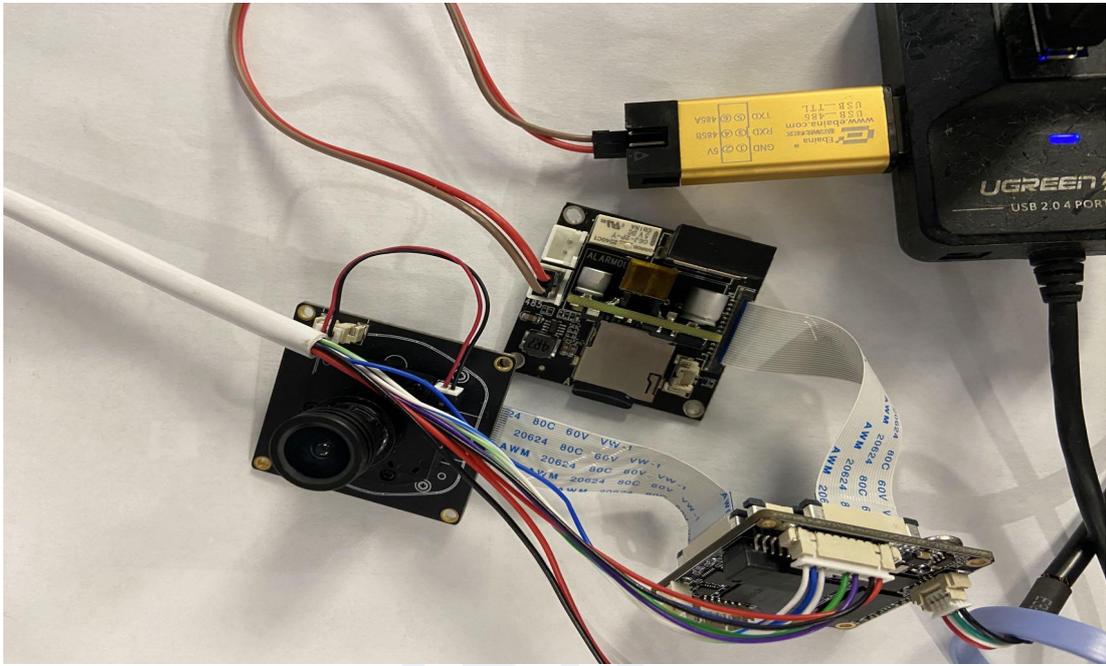
RS485接口的硬件原理图如下，具体请参考：A204-V3原理图.pdf



10.4.2 硬件连接

RS485接口只需要连接两根线485_A和485_B至串口工具上即可，如下图所示，无需接地。

Ps: 两根线接线具体顺序在传输信息后期可判断，若传输数据是乱码，重新交换485_A和485_B的顺序即可。



10.4.3 板端设置GPIO信息

- 第一步：在板端设置GPIO91参数，输入命令：`echo 91 > /sys/class/gpio/export`

Ps: 91是通过GPIO公式计算得到的值，执行完命令后，会在/sys/class/gpio/生成对应的gpio节点。

- 第二步：进入节点路径，输入如下命令：

```
cd /sys/class/gpio/gpio91      在91节点内操作
```

```
echo out > direction          设置为输出
```

```
echo 1 > value                 输出高
```

- 第三步：查看GPIO设置信息是否成功，输入命令：`cat /sys/kernel/debug/gpio`

结果如下图，可以看到GPIO91输出为高

```
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]# cat /sys/kernel/debug/gpio
gpiochip0: GPIOs 0-31, parent: platform/pinctrl, gpio0:
gpio-4 (          |vcc_sd          ) out lo
gpio-16 (         |rkwifi_wlan_poweren } out lo
gpio-18 (         |sysfs           ) out lo

gpiochip1: GPIOs 32-63, parent: platform/pinctrl, gpio1:
gpio-61 (         |reset          ) out hi

gpiochip2: GPIOs 64-95, parent: platform/pinctrl, gpio2:
gpio-69 (         |spk-ctl       ) out lo
gpio-91 (         |sysfs         ) out hi

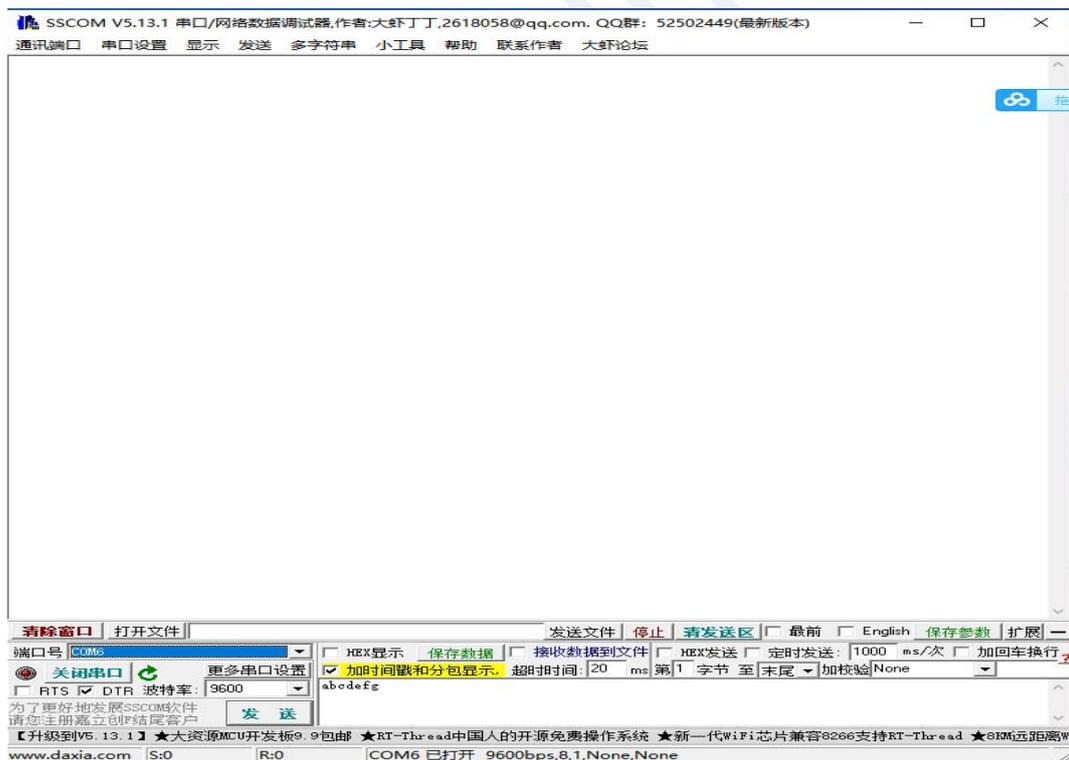
gpiochip3: GPIOs 96-127, parent: platform/pinctrl, gpio3:
gpio-104 (        |ircut-open    ) out lo
gpio-105 (        |ircut-close   ) out lo
gpio-117 (        |mdio-reset    ) out hi

gpiochip4: GPIOs 128-129, parent: platform/pinctrl, gpio4:
[root@RV1126_RV1109:~]#
[root@RV1126_RV1109:~]#
```

10.4.5 RS485传输数据验证

串口工具及介绍

- 这里使用的是sscom5.13.1.exe，软件界面安装打开如下，具体设置：“端口号”，下拉框选择对应哪个com口；“波特率”，设置为9600；“打开串口”，点击红色按钮切换



- 在板端输入想传输的数据，命令如下：echo test > /dev/ttyS3，在串口工具内即可收到数据，如下图所示：

The image shows two windows side-by-side. The left window is SecureCRT, displaying a terminal session with the command `echo test > /dev/ttyS3` and a list of sessions on the left. The right window is SSCOM V5.13.1, showing a serial terminal with received data: `[10:05:23.071]收←hello\r\n[10:05:25.071]收←hello\r\n[10:05:27.071]收←hello\r\n[10:05:29.072]收←hello\r\n[10:05:31.073]收←hello\r\n[10:05:33.073]收←hello\r\n[10:05:35.074]收←hello\r\n[10:05:37.075]收←hello\r\n[10:28:19.072]收←test`. The SSCOM window also has a configuration panel at the bottom with options for baud rate (9600), parity, and data bits.