

EVM-RK3562 用户手册

产品名称： EVM-RK3562 评估板

核 心 板： Core-RK3562 V1.0

修订人员： 杨伟锋

修订历史：

版本	日期	原因	修订者
V1.00	2024/08/13	初始版本	杨伟锋

目录

1. 测试产品	5
1.1. RK3562 核心板简介	5
1.2. RK3562 开发板简介	5
1.3. 软件资源介绍	7
2. 使用前的准备	9
2.1. USB 驱动安装	9
2.2. 串口登录	9
2.2.1. 使用 Typec 串口调试	9
2.2.2. 使用 MobaXterm 登录	11
2.2.3. SSH 网络登录	12
2.3. SFTP 文件传输	13
3. 编译环境的搭建	15
3.1. 安装虚拟机软件 VMware	15
3.2. Ubuntu 系统虚拟机的安装	15
3.2.1. VMware 安装 Ubuntu 镜像	15
3.2.2. Ubuntu22 网络配置	22
3.2.3. apt 命令指定国内源	23
3.2.4. 开启 Ubuntu 下的 FTP 服务	24
4. EVM-RK3562 平台 SDK 的使用	25
4.1. 编译环境要求	25
4.1.1. 编译环境配置	25
4.2. 源码获取	25
4.3. Buildroot (Linux) 系统编译	26
4.3.1. 全局编译	27
4.3.2. 单独编译 uboot	28
4.3.3. 单独编译 kernel	28
4.3.4. 单独编译 buildroot	28
4.3.5. 单独编译 recovery	28
4.3.6. 单独编译 buildroot 应用	29
5. 镜像升级	30
5.1. 在 windows 下烧写固件	31
5.1.1. 升级流程	31
5.1.2. 擦除 flash 操作	32
5.1.3. 单独升级指定分区镜像	32
5.2. 在虚拟机 ubuntu 中烧写固件	33

6. 系统信息查询	36
6.1. 查看系统信息	36
6.2 查看系统资源	37
6.2. 磁盘和分区	37
6.3. 查看网络信息	37
6.4. 查看进程信息	37
6.5. 查看用户信息	38
6.6. 查看 CPU 温度	38
7. 功能测试	39
7.1. DDR 测试	39
7.2. RTC 功能测试	39
7.3. 看门狗测试	40
7.4. TF 测试	41
7.5. 调频测试	42
7.6. SARADC 功能测试	42
7.7. emmc 测试	43
7.7.1. emmc 写入测试	43
7.7.2. emmc 读取测试	43
7.8. USB 测试	44
7.8.1. 鼠标测试	44
7.8.2. USB2.0 测试	44
7.9. 网络测试	45
7.10. LCD 背光调节	47
7.11. 放/录音测试	48
7.11.1. SPKOUT 播放声音	48
7.11.2. MIC 录音测试	49
7.12. SQLite3 测试	49
7.13. 4G/5G 测试	50
7.14. WIFI 测试	52
7.15. 蓝牙测试	53
7.16. CAN 测试	57
7.17. UART 测试	59
7.17.1. RS232 (UART2 CON5 端子) 测试	59
7.17.2. RS232 (UART5 CON6 端子) 测试	60
7.17.3. RS485 测试	61

7.18. 蜂鸣器测试	63
7.19. 烧录口 OTG 测试	64
7.19.1. 插入 Type-c 转 USB 线测试	64
7.19.2. 移除 Type-c 转 USB 线测试	64
7.19.3. OTG U 盘识别	64
7.20. 摄像头测试	64
7.20.1. 单路摄像头测试	64
7.20.2. 双路摄像头测试	66
8. 免责声明	69

1. 测试产品

1.1. RK3562 核心板简介

RK3562 核心板基于瑞芯微 RK3562J/RK3562 处理器，具备四核 ARM Cortex-A53 与单核 ARM Cortex-M0 的高效组合，主频高达 2.0GHz。这款全国产工业核心板专为满足工业自动化与消费电子领域的需求而设计，提供强大的计算能力和灵活的控制功能。

与此同时，RK3562 核心板同时具备强大的图形处理能力，与 OpenGL ES 1.1/2.0/3.2、OpenCL 2.0 以及 Vulkan 1.1 完全兼容。内置独立的 NPU，算力达 1TOPS，支持 INT4/INT8/INT16/FP16 数据类型的混合操作，支持轻量级的人工智能应用场景。具备丰富的外围接口，具备 PCIE 2.1、USB3.0、CAN、UART、SPI、MIPI CSI、GMAC 等多种接口。支持 H.264 编解码、H.265 等解码方式，支持 MIPI DSI、RGB、LVDS 等三种显示接口。

总之，RK3562 芯片是一款性能强大、功能丰富的处理器芯片，适用于高性能计算、人工智能、图形处理、多媒体应用等领域。它的出现将为各种智能设备和应用带来更好的性能和体验。



图 1 Core-RK3562 核心板正反面图

1.2. RK3562 开发板简介

眺望 EVM-RK3562 V1.0 评估板与 Core-RK3562J 核心板采用的是接插件的连接方式，主要的接口如图 2，图 3 所示：

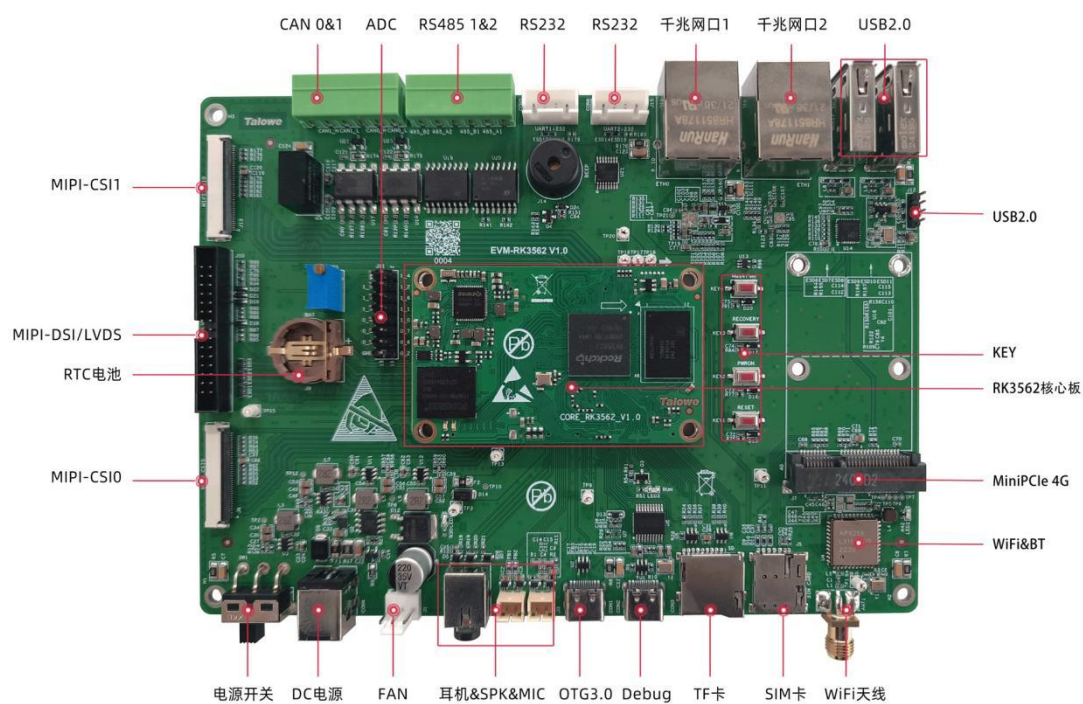


图 2 EVM-RK3562 评估板外观（正面）

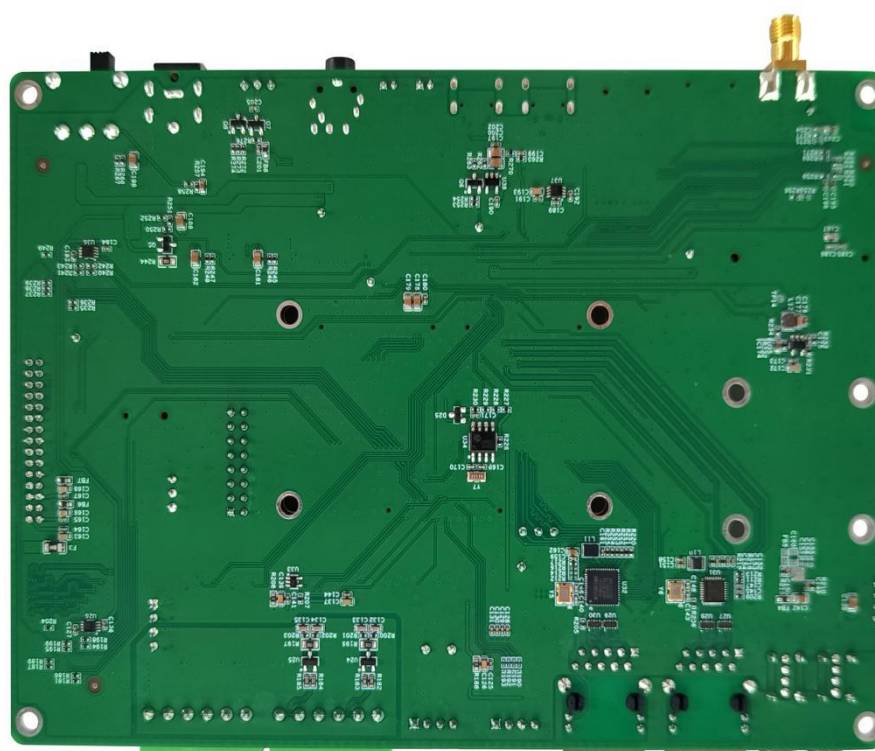


图 3 EVM-RK3562 评估板外观（反面）

注:图片仅供参考,以实际销售产品为准

1.3. 软件资源介绍

EVM-RK3562 评估板提供完善的 Linux SDK，Linux 内核版本为 5.10.198，具体软件资源如下表所示：

表 1 TW-RK3562-EVM 评估板软件资源表

软件资源	说明驱动程序源代码在内核中的位置	设备名
LCD 背光驱动	drivers/video/backlight/pwm_bl.c	/sys/class/backlight/l
USB 接口 U 盘	drivers/usb/storage/	-
USB 鼠标	drivers/hid/usbhid/	-
以太网	drivers/net/ethernet/stmicro/stmmac drivers/net/ethernet/motorcomm/*	-
SD/TF	drivers/mmc/host/dw_mmc-rockchip.c	/dev/mmcblk1pX
EMMC	drivers/mmc/host/dw_mmc-rockchip.c	/dev/mmcblk2pX
摄像头 OV13855	drivers/media/i2c/ov13855.c	/dev/videoX
摄像头 GC8034	drivers/media/i2c/GC8034.c	/dev/videoX
LCD	drivers/gpu/drm/rockchip/rockchip_vop2_reg.c	-
MIPI CSI	drivers/media/platform/rockchip/cif/dev.c	-
MIPI DSI	drivers/phy/rockchip/phy-rockchip-inno-mipi-dphy.c	-
LCD 触摸驱动	drivers/input/touchscreen/gt9xx/* drivers/input/touchscreen/gt9xx/gt9xx.c	/dev/input/eventX
RTC 实时时钟驱动	drivers/rtc/rtc-pcf8563.c	/dev/rtc0
串口	drivers/tty/serial/8250/8250_dw.c	/dev/ttySX
按键驱动	drivers/input/keyboard/adc-keys.c	/dev/input/eventX
LED	drivers/leds/leds-gpio.c	-
PMIC	drivers/mfd/rk808.c	-
PCIE	drivers/pci/host/pcie-rockchip.c	-

WATCHDOG	drivers/watchdog/dw_wdt.c	-
SPI	drivers/spi/spi-rockchip.c	-
PWM	drivers/pwm/pwm-rockchip.c	-

2. 使用前的准备

2.1. USB 驱动安装

用户可以从“Talowe-RK3562 产品资料 V1.0\2、软件资料\开发工具/USB 驱动”中下载 USB 驱动软件。



下载完成之后，按照指示安装完成即可。此时将开发板的 OTG 下载口与电脑端连接后，打开设备管理器即可看到该端口在电脑上的识别标志：

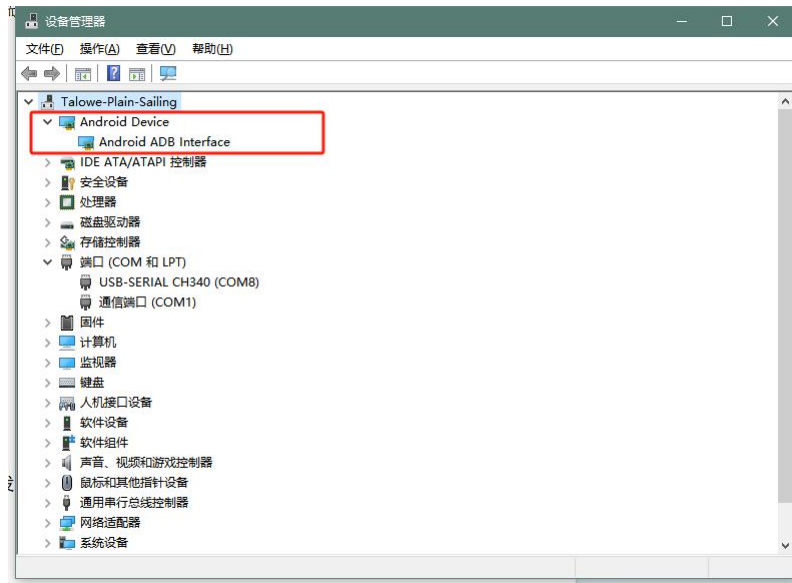


图 4 电脑识别到开发板的下载接口

2.2. 串口登录

2.2.1. 使用 Typec 串口调试

EVM-RK3562 评估板有单独引出来的 DEBUG 串口，为底板上的 CON2 端口。首先准备好 Typec 接口的数据线，实物如图 5 所示：



图 5 Typec 数据线

将 USB 数据线与 EVM-RK3562 评估板的 Typec 端口进行连接，连接如图 6 所示：

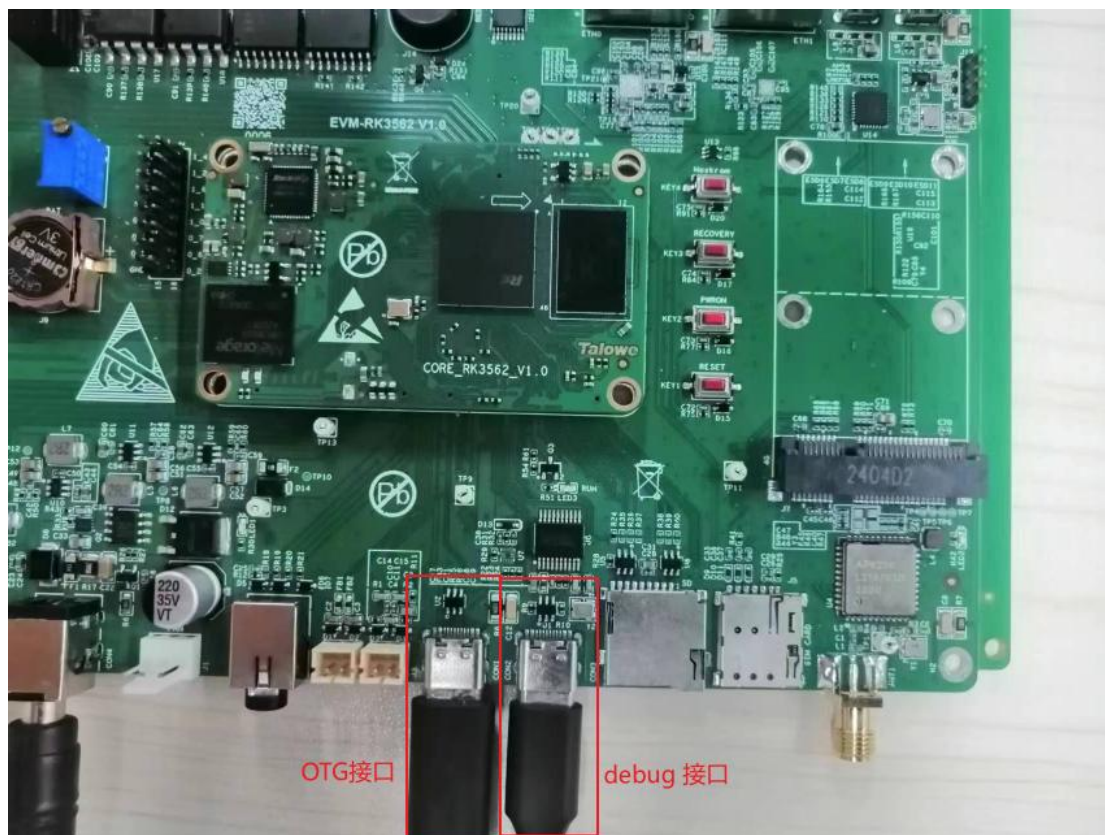


图 6 Typec 数据线与底板连接图

电脑和 EVM-RK3562 评估板的串口连接好之后，设备管理器会出现 CH340 的串口信息。

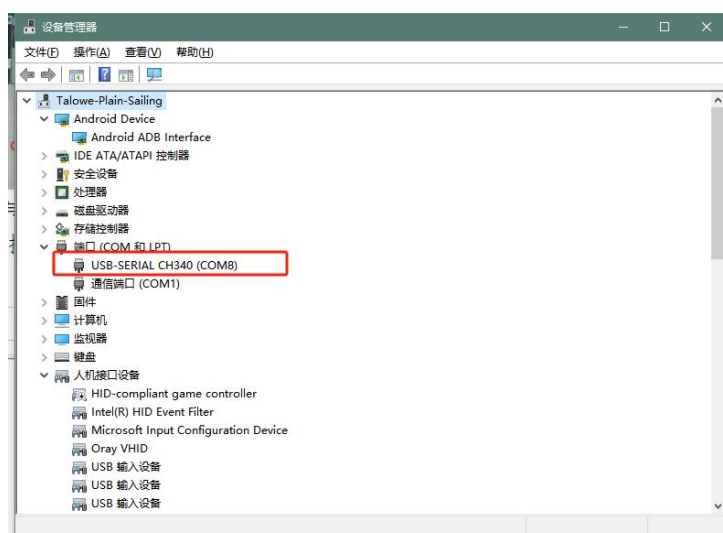


图 7 计算机识别到的串口设备

注：此处的端口号根据实际情况来定，例如这里识别到的 CH340 的端口号是 COM8。

2.2.2. 使用 MobaXterm 登录

● 安装 MobaXterm

用户可以从“Talowe-RK3562 产品资料 V1.02、软件资料\开发工具”下载 MobaXterm 安装程序。双击该文件开始安装，由于其安装过程比较简单，此处不再详细叙述。

● 运行 MobaXterm 软件

首先双击安装好的 MobaXterm 图标，在如下界面之中点击“会话”打开“会话设置”对话框：

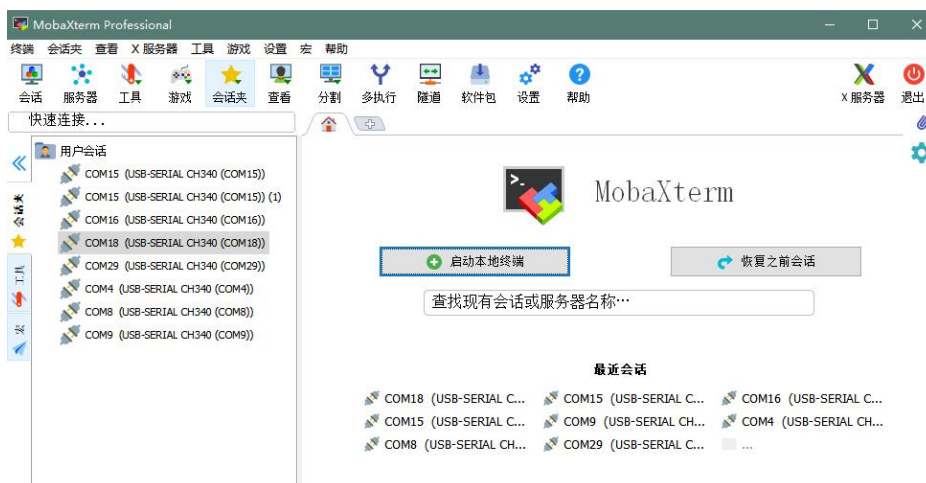


图 11 MobaXterm 软件界面

在“会话设置”对话框之中点击“串口”按钮进行基本串口设置选择实际的串行端口（例如本例中选择的是 COM8 端口，2.2.1 小节有详细步骤），速度选择 1500000bps 之后点击“好的”打开串口。

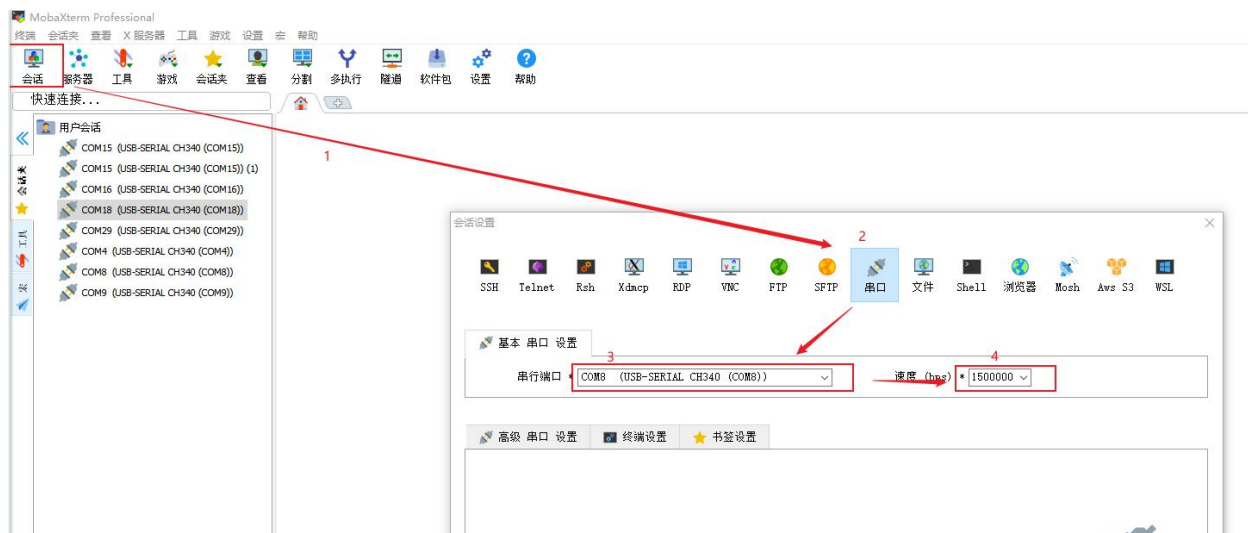


图 12 基本串口配置

串口登录成功画面如下：

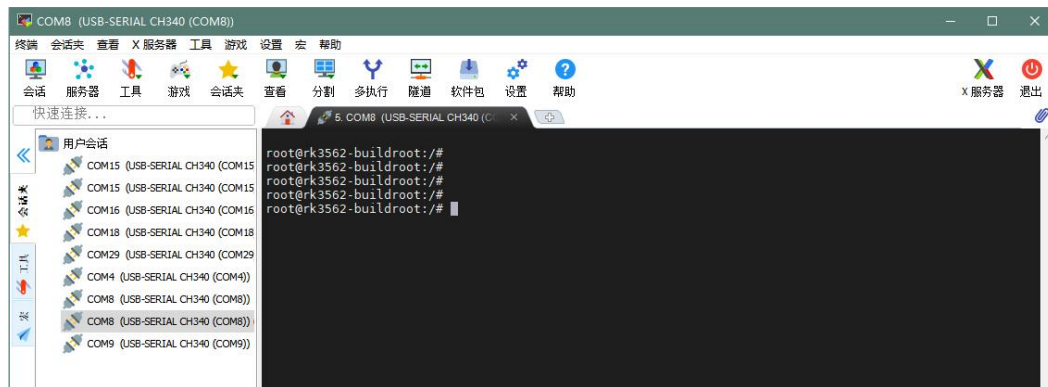


图 13 MobaXterm 登陆成功界面

2.2.3. SSH 网络登录

首先将板子网口 ETH0 接入网线，查询其对于的 ip 地址：

```
root@rk3562-buildroot:/# ifconfig
enp1s0  Link encap:Ethernet  HWaddr 00:55:7B:B5:7D:F7
        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:95 Base address:0x8000

eth0    Link encap:Ethernet  HWaddr 2A:52:60:0B:27:53
        inet addr:192.168.0.53  Bcast:192.168.0.255  Mask:255.255.255.0
        inet6 addr: fd00::2852:60ff:fedb:2753/64 Scope:Global
        inet6 addr: fe80::2852:60ff:fedb:2753/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:415  errors:0  dropped:0  overruns:0  frame:0
        TX packets:50  errors:0  dropped:0  overruns:0  carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:38044 (37.1 KiB)  TX bytes:5180 (5.0 KiB)
        Interrupt:78

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: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)
```

打开 MobaXterm 软件，点击会话按钮，执行如下操作：

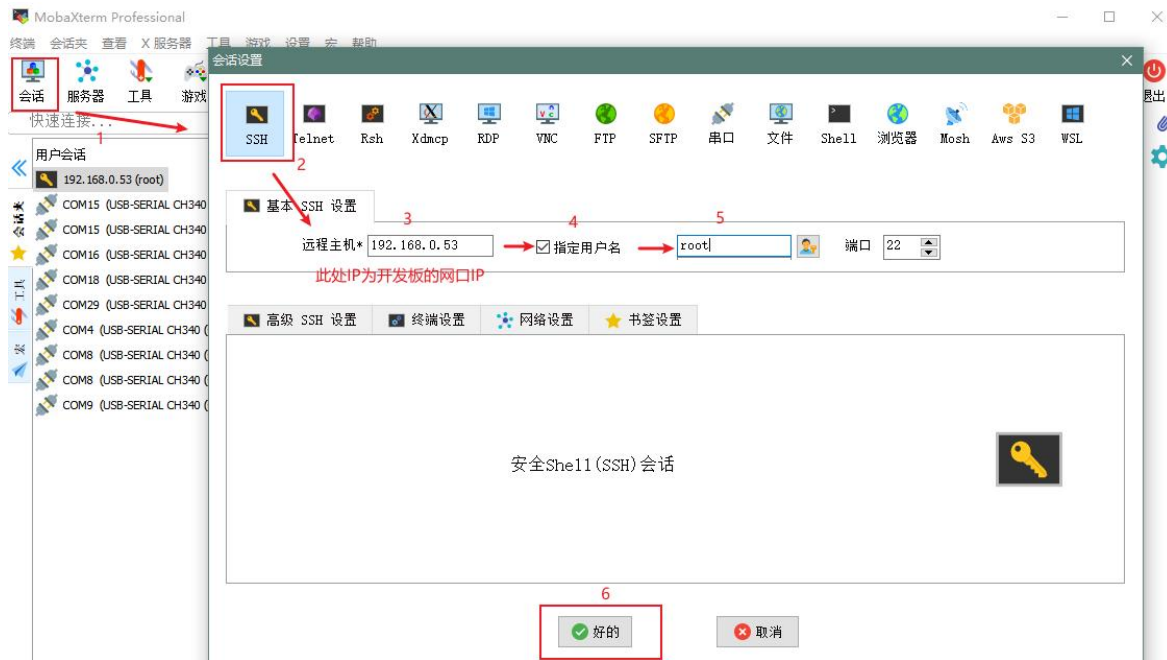


图 14 ssh 登录配置

登录过程中需要输入密码，初始密码为：**rockchip**

ssh 登录成功之后，即可对开发板进行操作：

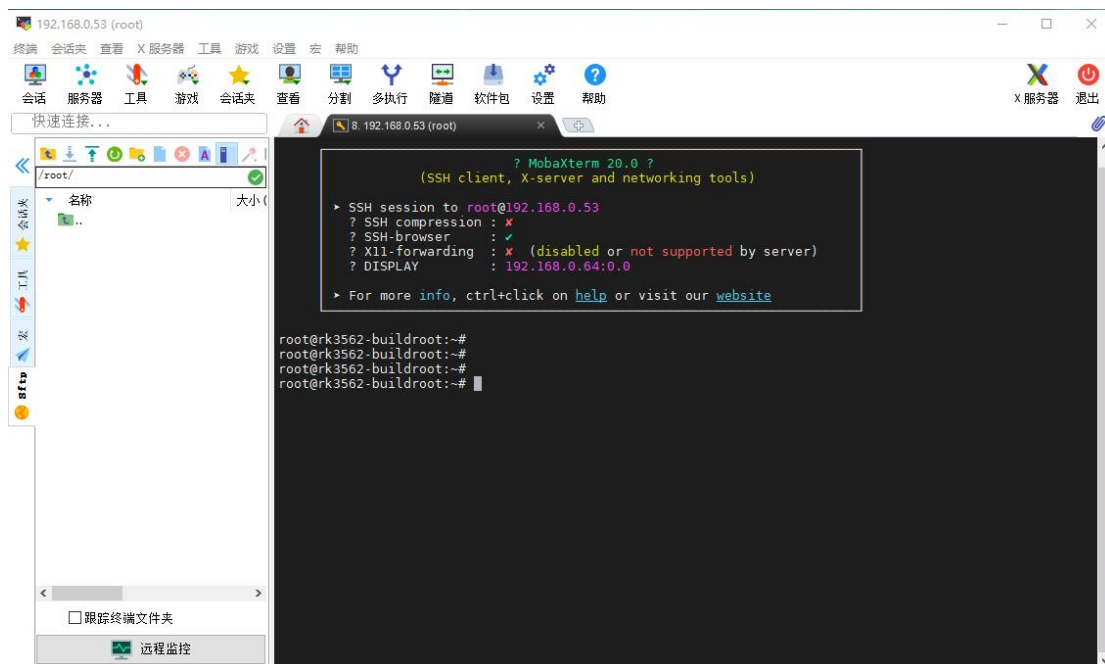


图 15 ssh 登录成功

2.3. SFTP 文件传输

RK3562 开发板支持 SFTP 服务并启动时已自动开启，下面介绍如何利用 SFTP 工具进行文件传输。

路径：Talowe-RK3562 产品资料 V1.0\2、软件资料\开发工具\FileZilla，在 windows 上安装好 filezilla 工具，并按照下图所示步骤进行设置。

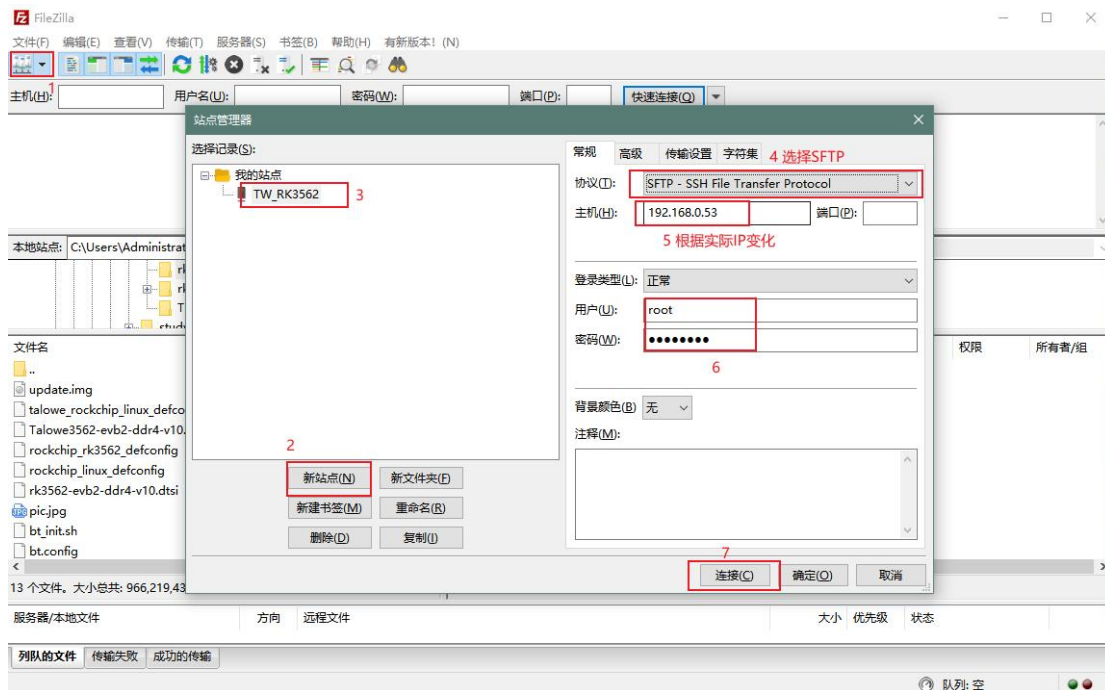


图 16 ftp 添加开发板的站点

用户名: root

密码: rockchip

登录成功后便可以进行上传下载操作。

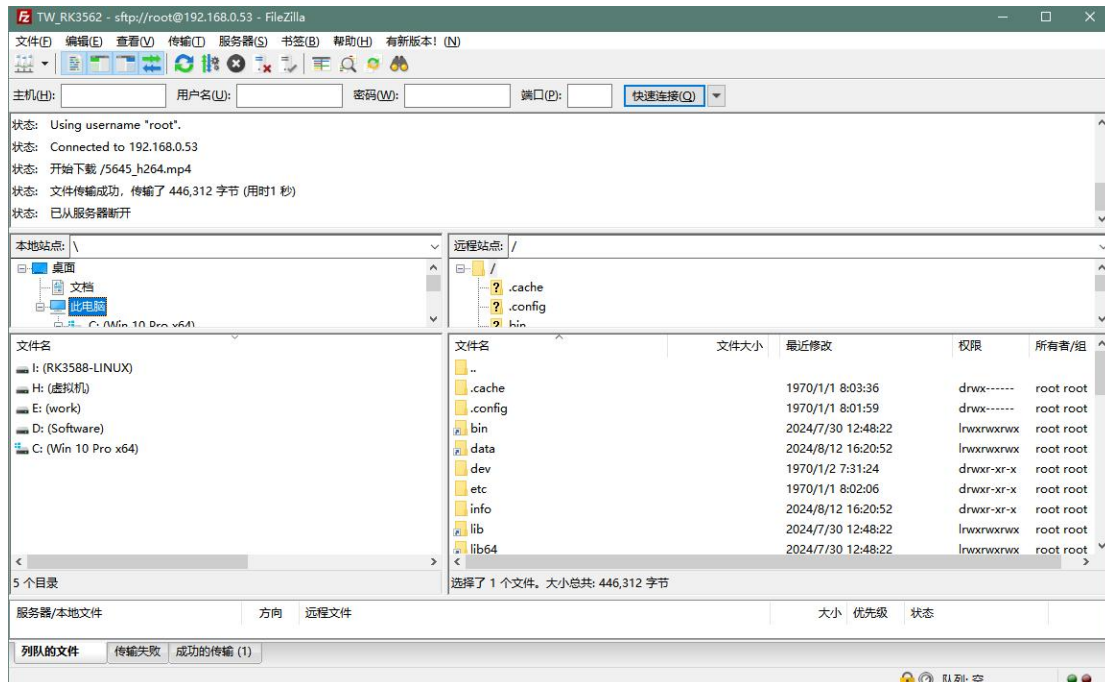


图 17 sftp 文件传输

3. 编译环境的搭建

3.1. 安装虚拟机软件 VMware

在安装 Ubuntu 的时候，我们需要使用 VMware 来启动它，当我们需要使用到 Ubuntu 系统的时候，只需要通过打开安装有 Ubuntu 系统的虚拟机即可。

Vmware Workstation 软件可以在 Vmware 官网下载，下载地址：<https://www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html>。下载最新版的 Vmware Workstation Pro 17。

3.2. Ubuntu 系统虚拟机的安装

本文中的介绍及开发均是在 Ubuntu22.04 上进行的。

Ubuntu22.04 的下载地址：[Ubuntu 22.04.4 LTS \(Jammy Jellyfish\)](https://ubuntu.com/releases/22.04.4-lts)

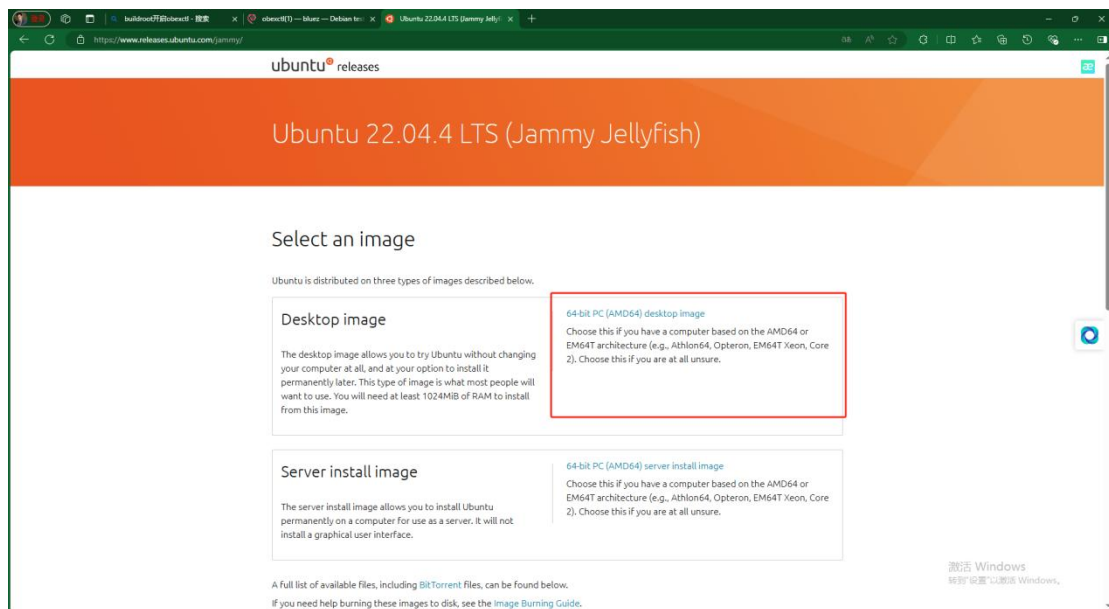


图 18 Ubuntu 下载

3.2.1. VMware 安装 Ubuntu 镜像

打开 VMware 软件，点击创建新的虚拟机。进入以下界面，勾选自定义（高级）点击下一步



图 19 Ubuntu 安装向导-1

选择对应 VMware 版本的兼容性，版本可在帮助->关于 VMware Workstation 中查看，确认无误点击下一步。



图 20 Ubuntu 安装向导-2

选择安装程序光盘映像文件，点击下一步。



图 21 Ubuntu 安装路径

输入全名、用户名和密码，点击下一步：



图 22 Ubuntu 安装向导-3

输入虚拟机名称及配置安装位置，可根据需要选择虚拟机的安装位置，点击下一步：

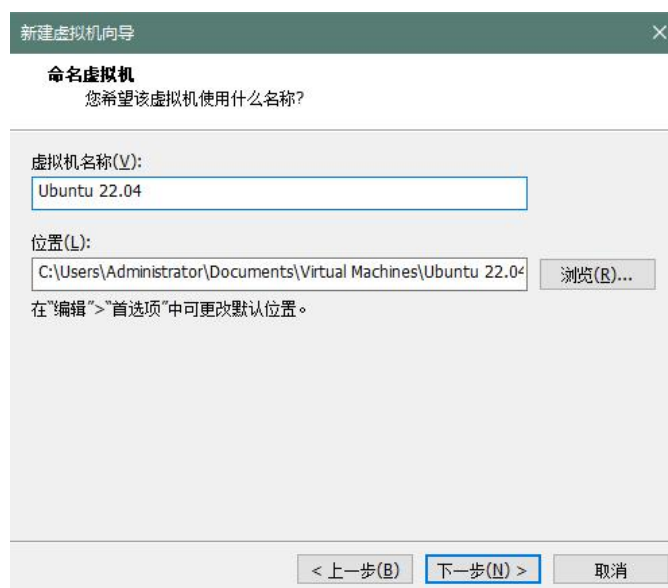


图 23 Ubuntu 名称

配置核心数，点击下一步：

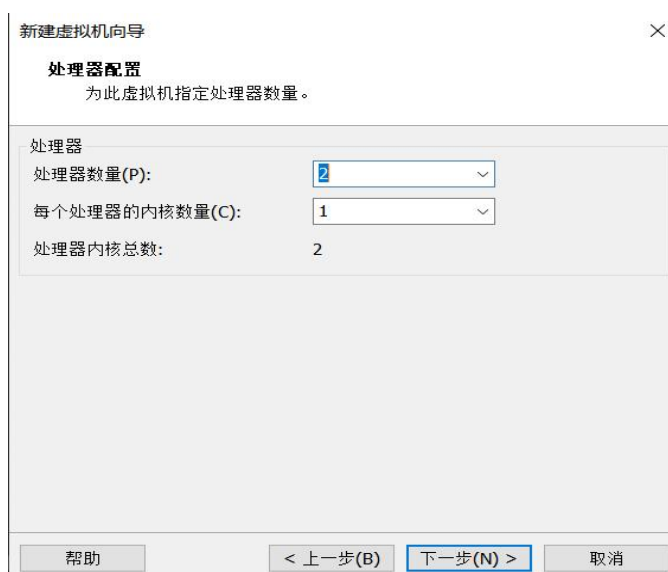


图 24 配置处理器数量

配置合适的内存空间，选择下一步：



图 25 设置虚拟机内存

使用 NAT 形式组网或桥接网络，点击下一步：

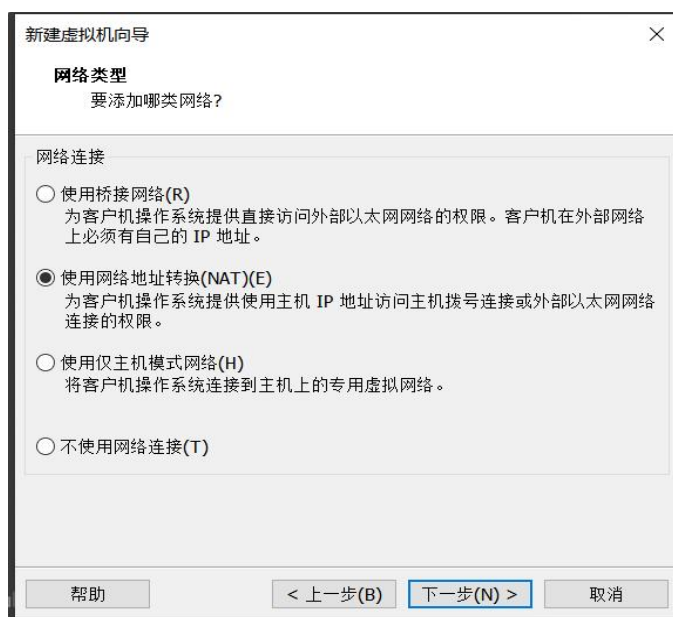


图 26 添加网络类型

使用推荐的 I/O 控制器，点击下一步：



图 27 选择 I/O 控制器类型

使用推荐的磁盘类型，点击下一步：



图 28 选择磁盘类型

使用默认选项，创建新的虚拟磁盘，并点击下一步：

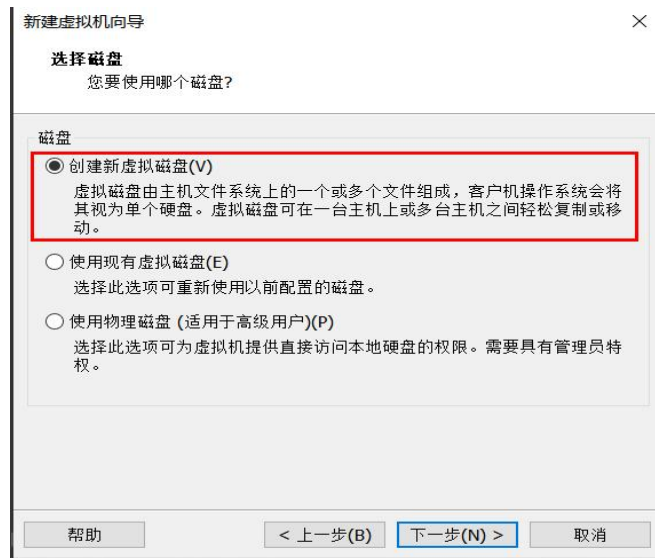


图 29 创建新虚拟磁盘

分配磁盘大小为 120G（自行配置），点击下一步：

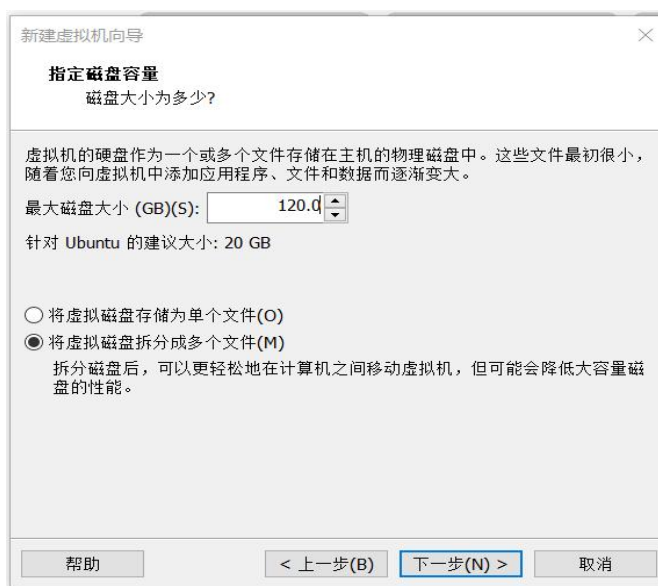


图 30 配置最大磁盘大小

根据需要将虚拟磁盘存储为单个文件或多个文件，使用默认点击下一步：



图 31 选择指定的磁盘文件

点击完成：

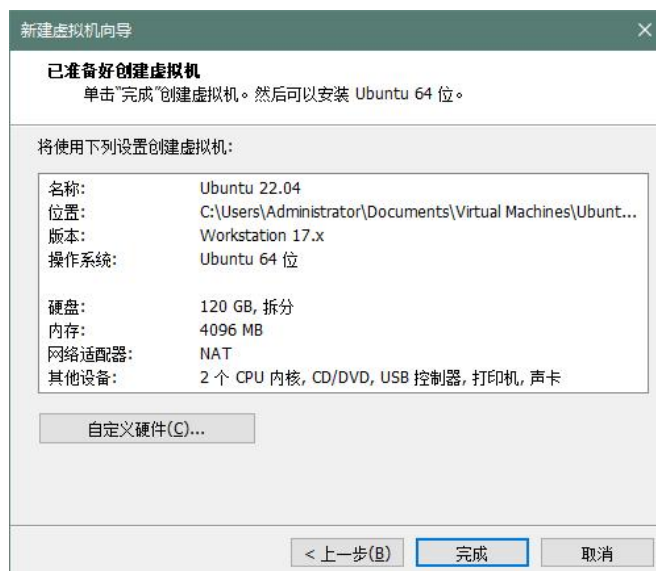


图 32 完成虚拟机创建

之后虚拟机则开始安装镜像，耐心等待。

3.2.2. Ubuntu22 网络配置

打开配置文件：

```
sudo gedit /etc/netplan/01-network-manager-all.yaml
```

```
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
```

```
ethernets:
  ens33:
    dhcp4: false
    addresses: [192.168.0.136/24]
    gateway4: 192.168.0.1
    nameservers:
      addresses: [192.168.94.1,8.8.8.8]
```

重启 netplan

```
sudo netplan apply
```

如果启动虚拟机镜像时，ifconfig 只有 lo 时，解决方法如下

```
sudo dhclient ens33
```

```
sudo netplan apply
```

①首先停止网络服务

```
service network-manager stop
```

②删除设备网卡状态管理文件；

```
sudo rm -rf /var/lib/NetworkManager/NetworkManager.state
```

③重新启动网络服务，这时网络状态将会重新加载刷新写入到文件。

```
service network-manager start
```

3.2.3. apt 命令指定国内源

由于 Ubuntu 默认的源是国外的，可能有时候无法访问，建议切换国内镜像源（清华源、阿里源等）。

Ubuntu22.04 系统中源的位置存放在/etc/apt/sources.list 文件中，替换相应链接即可，该操作需要使用 root 用户。下面以阿里源为例：

安全起见需要备份需要修改的文件，如下操作：

```
cp /etc/apt/sources.list /etc/apt/sources.list.bak
```

修改/etc/apt/sources.list 文件为以下内容：

```
deb http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe multiverse
```

```
deb http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe
multiverse
deb http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe
multiverse
```

保存后退出。在终端执行如下命令：

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

```
sudo apt-get install net-tools
sudo apt-get install libncurses5-dev
```

3.2.4. 开启 Ubuntu 下的 FTP 服务

```
sudo apt-get install vsftpd
```

安装完成以后使用 VI 命令打开/etc/vsftpd.conf，命令如下：

```
sudo vi /etc/vsftpd.conf
```

打开 vsftpd.conf 文件以后找到如下两行：

```
local_enable=YES
write_enable=YES
```

确保上面两行前面没有“#”，有的话就取消掉

修改完 vsftpd.conf 以后保存退出，使用如下命令重启 FTP 服务：

```
sudo /etc/init.d/vsftpd restart
```

4. EVM-RK3562 平台 SDK 的使用

4.1. 编译环境要求

Linux 编译对环境并无特别要求，但是考虑到 EVM-RK3562 同时支持 android、linux，建议使用 android 编译环境，安装 linux 编译依赖软件包。

项目	内存	CPU	软件
编译主机(linux)	>= 8G	64 位 x86	64 位 Ubuntu22.04 及以上
Windows 主机	无要求	无要求	64 位 windows7 及以上

如无特别说明，本文档所有操作均基于以上硬件、软件环境下完成。

本文档操作不需要使用 root 用户，配置编译环境安装软件使用 sudo 即可。在 ubuntu22.04 下，将 android 源码拷贝、下载到当前用户目录下，拷贝、下载、解压、编译源码均在普通用户下执行。请勿使用 root 用户或者 sudo 命令拷贝、下载、解压、编译源码，可能会导致编译出错。特别说明：不能放在虚拟机的共享文件夹中编译！！！！

4.1.1. 编译环境配置

SDK 开发环境是在 Ubuntu 系统上开发与测试。我们推荐使用 Ubuntu 22.04 的系统进行编译。其他的 Linux 版本可能需要对软件包做相应调整。除了系统要求外，还有其他软硬件方面的要求。硬件要求:64 位系统，硬盘空间大于 50G。如果您进行多个构建，将需要更大的硬盘空间。软件要求:Ubuntu 22.04 系统:编译 SDK 环境搭建所依赖的软件包安装命令如下:

```
sudo apt-get update && sudo apt-get install git ssh make gcc libssl-dev \
liblz4-tool expect expect-dev g++ patchelf chrpath gawk texinfo chrpath \
diffstat binfmt-support qemu-user-static live-build bison flex fakeroot \
cmake gcc-multilib g++-multilib unzip device-tree-compiler ncurses-dev \
libgucharmap-2-90-dev bzip2 expat gpgv2 cpp-aarch64-linux-gnu libgmp-dev \
libmpc-dev bc python-is-python3 python2
```

建议使用 Ubuntu22.04 系统或更高版本开发，若编译遇到报错，可以视报错信息，安装对应的软件包。

4.2. 源码获取

请通过我司销售途径获取源码。将源码拷贝到 ubuntu 22.04 当前登录用户目录或新建 sdk 目录下。解压：

```
$ tar -xvzf rk3562-linux-sdk.tar.gz -C 解压目录
```

解压完成后进入 sdk 后可以看到

```

y@y-virtual-machine:~/rk3562_sdk$ ll
总用量 10745756
drwxr-xr-x 18 y y      4096 8月 27 08:59 ./
drwxr-xr-x 29 y y      4096 8月 27 08:51 ../
drwxr-xr-x  4 y y      4096 4月 25 15:48 app/
drwxr-xr-x 19 y y      4096 8月 27 08:59 buildroot/
lrwxrwxrwx  1 y y        39 1月  9 2024 build.sh -> device/rockchip/common/scripts/build.sh*
lrwxrwxrwx  1 y y        22 7月 30 12:30 common -> device/rockchip/common/
lrwxrwxrwx  1 y y        21 4月 25 15:48 Copyright_Statement.md -> docs/licenses/LICENSE*
drwxr-xr-x 10 y y      4096 4月 25 15:48 debian/
drwxr-xr-x  3 y y      4096 8月  6 15:48 device/
drwxr-xr-x  7 y y      4096 4月 25 15:48 docs/
drwxr-xr-x 27 y y      4096 4月 25 15:49 external/
drwxrwxr-x  8 y y      4096 8月 27 08:59 .git/
drwxr-xr-x 11 y y      4096 4月 25 15:48 hal/
lrwxrwxrwx  1 y y        11 1月  9 2024 kernel -> kernel-5.10/
drwxr-xr-x 29 y y      4096 8月 27 08:59 kernel-5.10/
lrwxrwxrwx  1 y y        31 1月  9 2024 Makefile -> device/rockchip/common/Makefile
drwxrwxr-x  9 y y      4096 8月 27 08:59 output/
drwxr-xr-x  3 y y      4096 1月  9 2024 prebuilts/
lrwxrwxrwx  1 y y        32 7月 30 12:30 README.md -> device/rockchip/common/README.md
-rw-r--r--  1 y y 11003573013 1月  9 2024 rk3562_linux_sdk.tar.gz
drwxr-xr-x 10 y y      4096 8月 27 08:57 rkbin/
lrwxrwxrwx  1 y y        41 1月  9 2024 rkflash.sh -> device/rockchip/common/scripts/rkflash.sh*
lrwxrwxrwx  1 y y        15 8月 27 08:59 rockdev -> output/firmware/
drwxr-xr-x 17 y y      4096 4月 25 15:48 rtos/
drwxr-xr-x  6 y y      4096 1月  9 2024 tools/
drwxr-xr-x 28 y y      4096 8月 27 08:57 u-boot/
drwxr-xr-x  9 y y      4096 4月 25 15:48 yocto/

```

图 34 sdk 目录下的文件

sdk 目录说明

SDK

--app:	存放上层应用 APP，主要是一些应用Demo。
--buildroot:	基于 Buildroot（2021）开发的根文件系统。
--debian:	基于 Debian bullseye(11) 开发的根文件系统。
--device/rockchip:	存放芯片板级配置以及 SDK 编译和打包固件的脚本和文件等。
--docs:	存放通用开发指导文档、芯片平台相关文档、Linux 系统开发相关文档、其他参考文档等。
--external:	存放第三方相关仓库，包括显示、音视频、摄像头、网络、安全等。
--kernel:	存放 Kernel 开发的代码。
--hal:	存放基于 RK HAL 硬件抽象层的裸机开发库，用于 AMP方案。
--output:	存放每次生成的固件信息、编译信息、XML、主机环境等。
--prebuilts:	存放交叉编译工具链。
--rkbin:	存放 Rockchip 相关二进制和工具。
--rockdev:	存放编译输出固件,实际软链接到 output/firmware 。
--rtos:	存放基于 RT-Thread 4.1 开发的根文件系统。
--tools:	存放 Linux 和 Window 操作系统下常用工具。
--u-boot:	存放基于 v2017.09 版本进行开发的 U-Boot 代码。
--yocto:	存放基于 Yocto 4.0 开发的根文件系统。

4.3. Buildroot（Linux）系统编译

无论最后要使用什么系统，都请先全局编译一次 buildroot，之后再进行其他系统的编译，避免出现需要某个依赖的工具找不到的情况。以下操作，若无特殊说明，均在 SDK 根目录下进行。

4.3.1. 全局编译

环境资源配置，在 SDK 目录下输入 `source buildroot/envsetup.sh` 选择 44:

```
43. rockchip rk3528 recovery
44. rockchip rk3562
45. rockchip rk3562_32
46. rockchip rk3562_dictpen
47. rockchip rk3562_ramboot
48. rockchip rk3562_recovery
49. rockchip rk3562_robot
50. rockchip rk3566
51. rockchip rk3566_32
52. rockchip rk3566_recovery
53. rockchip rk3566_rk3568_base
54. rockchip rk3566_rk3568_ramboot
55. rockchip rk3568
56. rockchip rk3568_32
57. rockchip rk3568_recovery
58. rockchip rk3588
59. rockchip rk3588_base
60. rockchip rk3588_ramboot
61. rockchip rk3588_recovery
Which would you like? [1]: 44
```

图 35 环境资源配置

板级配置，在 SDK 目录下输入 `./build.sh lunch` 选择 6

#选择板型、系统（首次编译\更换板型\更换系统才需要执行）

```
y@y-virtual-machine:~/rk3562_sdk$ ./build.sh lunch
Log colors: message notice warning error fatal

Log saved at /home/y/rk3562_sdk/output/sessions/2024-08-14_14-28-11
Pick a defconfig:

1. rockchip_defconfig
2. rockchip_rk3562_dictpen_test3_v20_defconfig
3. rockchip_rk3562_evb1_lp4x_v10_amp_defconfig
4. rockchip_rk3562_evb1_lp4x_v10_defconfig
5. rockchip_rk3562_evb2_ddr4_v10_amp_defconfig
6. rockchip_rk3562_evb2_ddr4_v10_defconfig
7. rockchip_rk3562_robot_evb1_lp4x_v10_defconfig
8. rockchip_rk3562_robot_evb2_ddr4_v10_defconfig
Which would you like? [1]: 6
```

图 36 板级配置

选择完成后，进行首次编译，执行全局编译，命令如下：

```
./build.sh # 全局编译，编译 uboot、kernel、recovery
```

编译成功后将会在 `sdk/linux/rockdev/` 目录下生成如下镜像文件。

```
ls rockdev/
```

```
boot.img
```

#内核、设备树、logo 分区

```
linux-headers.tar
```

#linux 头文件压缩包

MiniLoaderAll.bin	#引导 uboot、检测启动方式
misc.img	#记录 recovery 升级信息
oem.img	#存放 OEM 资源分区
parameter.txt	#系统分区表、内核传参
recovery.img	#恢复还原分区
rootfs.img	#根文件系统分区
uboot.img	#引导系统分区
update.img	#整包固件 用于整包烧写
userdata.img	

编译成功后将会在 rockdev/ 目录下生成镜像文件，整包烧录请使用镜像：update.img。

如需清空整个 SDK，删除所有编译生成的文件，可采用如下指令

```
./build.sh cleanall # 清除编译 uboot、kernel、recovery
```

4.3.2. 单独编译 uboot

执行命令：

```
./build.sh u-boot
```

生成镜像路径： u-boot/

生成镜像： uboot.img、trust.img 等

4.3.3. 单独编译 kernel

执行命令：

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

生成镜像路径： rockdev/

生成镜像： boot.img

4.3.4. 单独编译 buildroot

编译完 buildroot 完成后会自动编译 rootfs

```
./build.sh buildroot
```

生成镜像路径： rockdev/

生成镜像： rootfs.img

4.3.5. 单独编译 recovery

```
./build.sh recovery
```

4.3.6. 单独编译 buildroot 应用

sdk 可以单独编译 buildroot 的单个应用设置单个模块，比如单独编译 QFacialGate 应用

```
make qcamera -dirclean && make qcamera -rebuild -j16
```

单独编译 rkweifbt

```
make rkweifbt-dirclean && make rkweifbt-rebuild -j16
```

5. 镜像升级

升级工具：RKDevTool_Release.zip（v2.81 或更高版本）。

升级工具路径：Talowe-RK3562 产品资料 V1.0\2、软件资料\开发工具\瑞芯微官方烧写工具

使用方式：解压到本地目录，打开 RKDevTool.exe 即可。

工具界面：

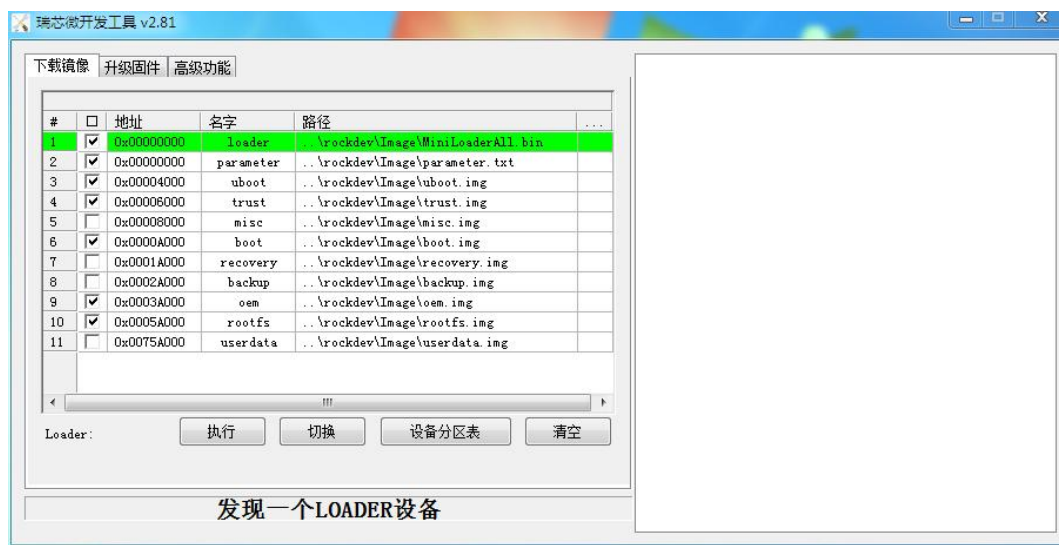


图 4.1 工具界面

若第一次使用瑞芯微平台，使用瑞芯微官方烧写工具前需要确定 PC 已成功安装瑞芯微烧写驱动。

驱动安装路径为 Talowe-RK3562 产品资料 V1.0\2、软件资料\开发工具\瑞芯微烧录驱动\DriverAssitant_v5.1.1，双击路径下的 DriverInstall.exe 文件进行驱动的安装。



图 4.2 驱动安装界面

5.1. 在 windows 下烧写固件

5.1.1. 升级流程

如果是不同系统的切换升级，如 android 系统切换 linux 系统，则最好先擦除以后再升级。

1. 主板上电（12V）。
2. 主板 USB（OTG 接口）与电脑连接。
3. 按住主板 Recovery 按键后，按一下主板 reset 按键或重新上电（保持该状态 5 秒）。
4. 升级工具识别到 LOADER 或者 MASKROOM 设备 后方可松开 Recovery 升级按键。

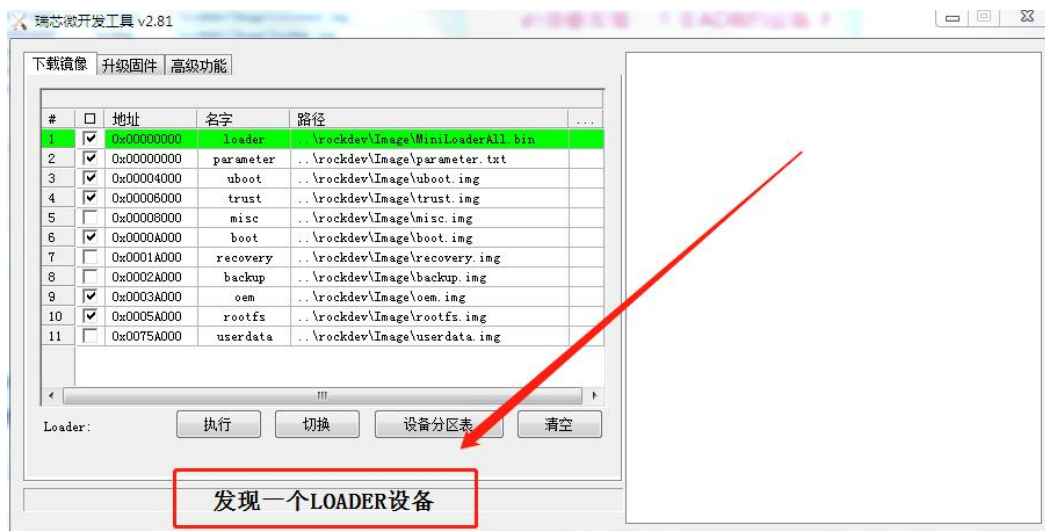


图 36 识别到开发板



图 37 镜像升级

固件获取途径：

1、光盘所提供的固件：Talowe-RK3562 产品资料 V1.0\2、软件资料\系统固件更新

2、个人编译生成的固件：<SDK 根目录>/rockdev/update.img

5.1.2. 擦除 flash 操作

板子切换到 loader 或 maskroom 模式，选择对应的升级固件，进入升级工具进入擦除 Flash。



图 38 擦除 flash 操作

5.1.3. 单独升级指定分区镜像

单独升级指定分区镜像常用于开发、调试过程中。最常用是单独升级 uboot、boot、rootfs 分区。勾选分区表示此次升级分区，可以同时勾选多个分区，同时烧录多个分区。

此处以升级内核分区 boot.img 为例，仅修改 kernel 驱动、设备树或者开机 logo，编译 kernel，只要升级 boot 分区即可。其他分区烧录与之类似，加载对应的文件即可：

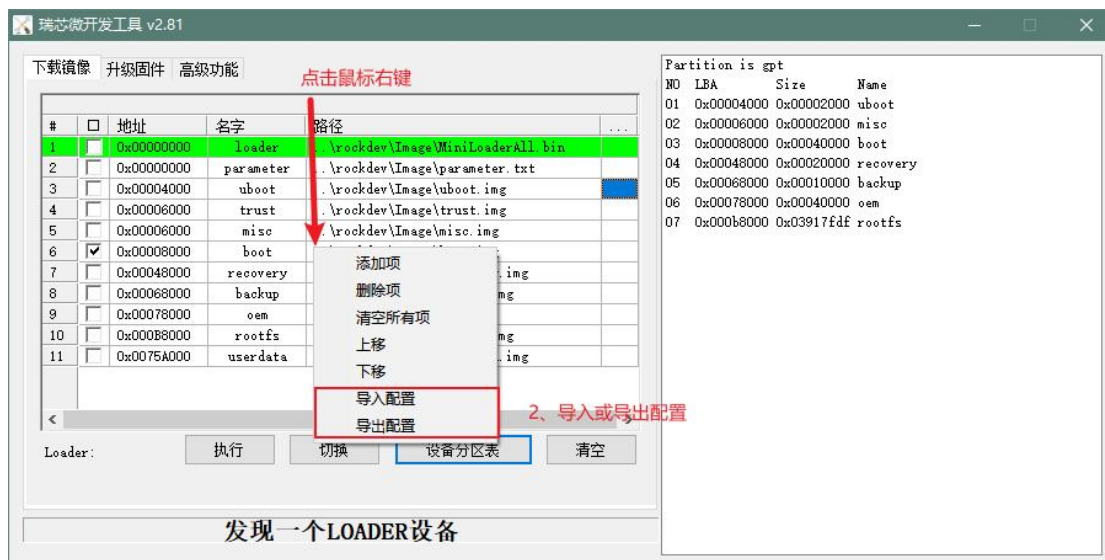


图 39 单独升级 rootfs 镜像

选中 <SDK 根目录>/rockdev/ 目录下编译生成的 boot.img，点击打开，点击执行，升级镜像，等待提示升级成功，之后会自动重启。

注意：编译和烧录分区镜像前请先整包编译，否则没有 rockdev/ 目录

用户可通过导入配置或者导出配置从而进行分区的配置形成文件，以便下一次的使用。



5.2. 在虚拟机 ubuntu 中烧写固件

在 ubuntu 中烧写固件，与在 window 中烧写类似，使用[SDK 根目录]/tools/linux/Linux_Upgrade_Tool/Linux_Upgrade_Tool/目录中的 upgrade_tool 工具进行烧写。首先按钮一下步骤进入升级模式：

1. 主板上电（12V）。

2. 主板 USB（OTG 接口）与电脑连接。
3. 按住主板 Recovery 按键，按一下主板 reset 按键或者重新上电（保持该状态 5 秒）。
4. 进入 LOADER 或者 MASKROOM 模式（Linux 下没有显示，保证上一个步骤正确即可进入该模式）。
5. 将 USB Device 连接到虚拟机中。

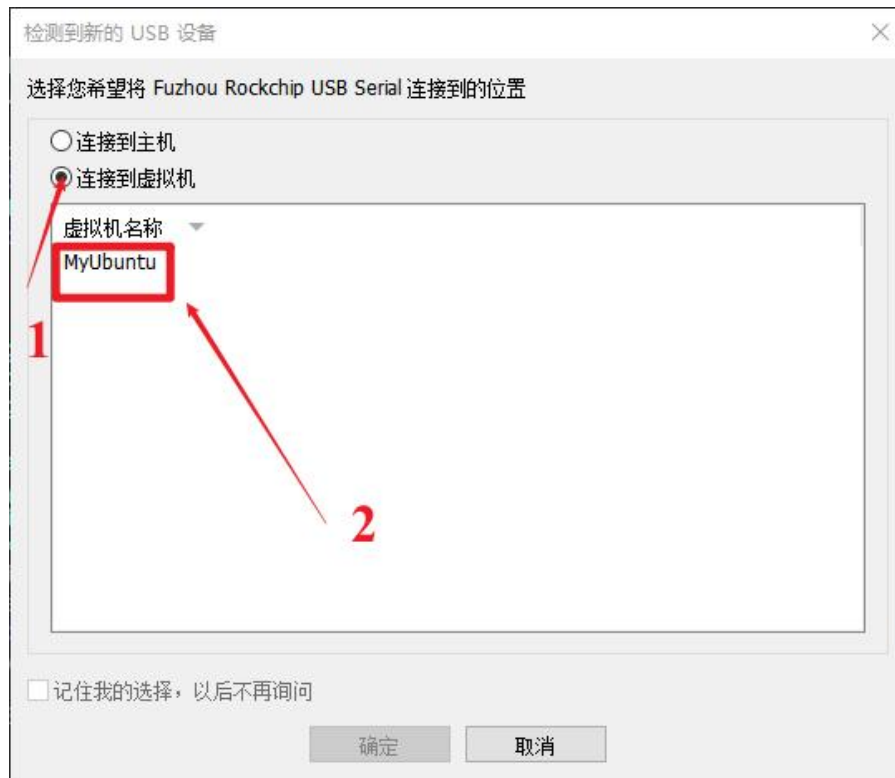


图 40 将开发板链接到虚拟机

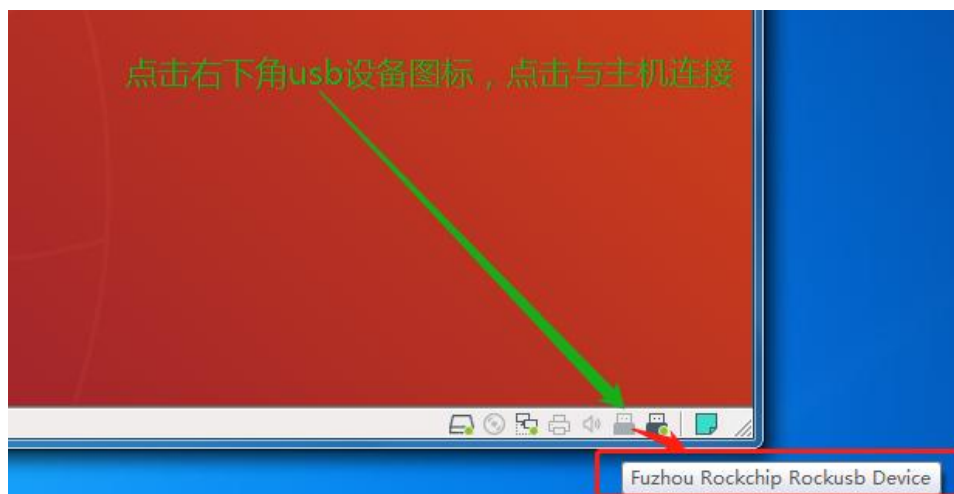


图 41 识别到开发板

点击右下角 usb 设备图标，点击连接（与主机断开连接）。

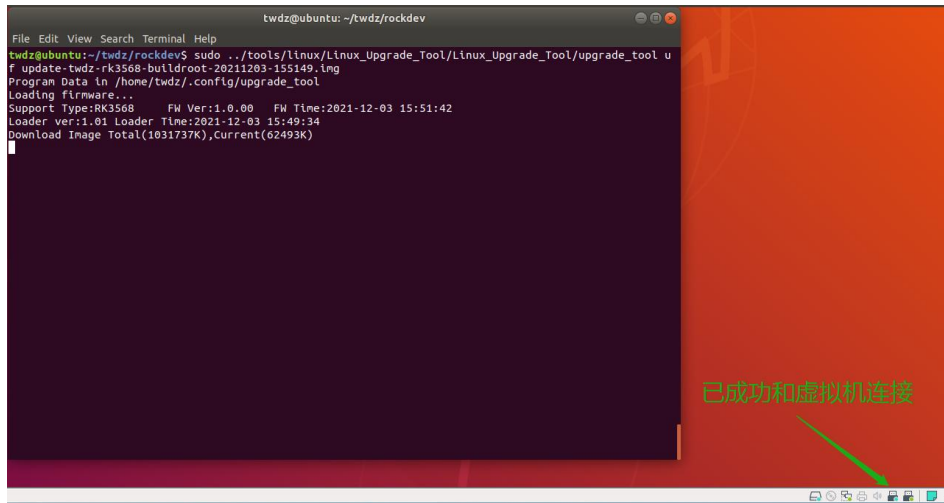


图 42 已链接到开发板

当 usb 成功与虚拟机连接后，进入生成固件的目录/rockdev/，在目录中执行以下命令，执行烧写：

```
sudo ../tools/linux/Linux_Upgrade_Tool/Linux_Upgrade_Tool/upgrade_tool uf update.img
```

烧写完成后，会提示 Upgrade firmware ok!

6. 系统信息查询

6.1. 查看系统信息

查看系统内核版本，使用 `uname` 命令。

```
uname -a
```

```
root@rk3562-buildroot:/# uname -a
Linux rk3562-buildroot 5.10.198 #40 SMP Tue Aug 13 17:52:34 CST 2024 aarch64 GNU/Linux
```

查看操作系统信息：

```
cat /etc/issue
```

```
root@rk3562-buildroot:/# cat /etc/issue
Welcome to RK3562 Buildroot
```

查看环境变量信息：

```
env
```

```
root@rk3562-buildroot:/# env
SHELL=/bin/sh
GST_V4L2_PREFERRED_FOURCC=Nv12:YU12:Nv16:YUY2
GST_VIDEO_CONVERT_PREFERRED_FORMAT=Nv12:Nv16:I420:YUY2
PIXMAN_USE_RGA=1
UMS_R0=0
CHROMIUM_FLAGS=--enable-wayland-ime
GST_V4L2_USE_LIBV4L2=1
WESTON_DRM_MIN_BUFFERS=2
WL_OUTPUT_VERSION=3
UMS_MOUNTPOINT=/mnt/ums
GST_INSPECT_NO_COLORS=1
UMS_MOUNT=0
PULSE_HOME=/userdata/.pulse
EDITOR=/bin/vi
WESTON_DRM_KEEP_RATIO=1
GST_DEBUG_NO_COLOR=1
PWD=/
TSLIB_CONSOLEDEVICE=none
UMS_SIZE=256M
HOME=/
LANG=en_US.UTF-8
WESTON_FREEZE_DISPLAY=/tmp/.freeze_weston
WAYLANDSINK_FORCE_DMABUF=1
GST_V4L2SRC_DEFAULT_DEVICE=/dev/video-camera0
TSLIB_CALIBFILE=/etc/pointercal
TERM=vt102
USER=root
UMS_FILE=/userdata/ums_shared.img
AUTOAUDIO_SINK_PREFERRED=pulsesink
ADB_SHELL=/bin/bash
GST_V4L2SRC_RK_DEVICES=_mainpath:_selfpath:_bypass:_scale
WESTON_DRM_MIRROR=1
UMS_FSTYPE=vfat
SHLVL=1
TSLIB_PLUGIN_DIR=/usr/lib/ts
GST_VIDEO_FLIP_USE_RGA=1
USB_FUNCS=adb
WESTON_DISABLE_ATOMIC=1
XDG_RUNTIME_DIR=/var/run
PLAYBIN2_PREFERRED_AUDIO_SINK=pulsesink
TSLIB_TSDEVICE=/dev/input/event1
TSLIB_CONFFILE=/etc/ts.conf
TSLIB_FBDEVICE=/dev/fb0
GST_VIDEO_CONVERT_USE_RGA=1
PATH=/usr/bin:/usr/sbin
storage_media=emmc
GST_V4L2SRC_MAX_RESOLUTION=3840x2160
GST_VIDEO_DECODER_QOS=0
=/usr/bin/env
```

图 44 环境变量信息

其它相关信息：

```
cat /proc/cpuinfo      # 查看 CPU 信息
hostname               # 查看计算机名
lsusb -tv              # 列出所有 USB 设备
lsmod                  # 列出加载的内核模块
```

6.2 查看系统资源

查看系统内存使用情况

```
free -h
```

```
root@rk3562-buildroot:/# free -h
              total        used         free       shared    buff/cache   available
Mem:          1.9Gi         100Mi        1.7Gi          11Mi         113Mi        1.8Gi
Swap:          0B           0B           0B
```

查看系统磁盘使用情况

```
df -h
```

```
root@rk3562-buildroot:/# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        6.0G  660M   5.0G  12% /
devtmpfs         972M   8.0K  972M   1% /dev
tmpfs            983M   12K   983M   1% /tmp
tmpfs            983M  420K   983M   1% /run
tmpfs            983M  188K   983M   1% /var/log
tmpfs            983M    0     983M   0% /dev/shm
/dev/mmcblk0p7  121M   12M   103M  11% /oem
/dev/mmcblk0p8  22G   474K   22G   1% /userdata
```

系统资源其它信息：

```
free -m              # 查看内存使用量和交换区使用量
uptime               # 查看系统运行时间、用户数、负载
du -sh <目录名>      # 查看指定目录的大小
grep MemTotal /proc/meminfo # 查看内存总量
grep MemFree /proc/meminfo # 查看空闲内存量
cat /proc/loadavg     # 查看系统负载
```

6.2. 磁盘和分区

```
fdisk -l             # 查看所有分区
cat /proc/partitions # 查看文件可看到各分区信息。
```

6.3. 查看网络信息

```
ifconfig             # 查看所有网络接口的属性
route -n              # 查看路由表
netstat               # 查看所有监听端口和建立的连接
```

6.4. 查看进程信息

```
ps -ef               # 查看所有进程
top                  # 实时显示进程状态（按 q 退出）
```

6.5. 查看用户信息

w	# 查看活动用户
id <用户名>	# 查看指定用户信息
cut -d: -f1 /etc/passwd	# 查看系统所有用户
cut -d: -f1 /etc/group	# 查看系统所有组

6.6. 查看 CPU 温度

输入如下命令查看温度值：

```
cat /sys/class/thermal/thermal_zone0/temp
```

```
root@rk3562-buildroot:/# cat /sys/class/thermal/thermal_zone0/temp
41240
```

7. 功能测试

7.1. DDR 测试

输入下面指令查看双倍速率同步动态随机存储器的版本：

```
memtester 1024
```

```
root@rk3562-buildroot:/# memtester 1024
memtester version 4.5.1_20231020 (32-bit)
Copyright (C) 2001-2020 Charles Cazabon.
Licensed under the GNU General Public License version 2 (only).

pagesize is 4096
pagesizemask is 0xfffffffffff000
want 1024MB (1073741824 bytes)
got 1024MB (1073741824 bytes), trying mlock ...locked.
testing from phyaddress:0x7ebfe000
get chip name: rk3562
no available chip info, using default mapping
get ddr bw: bw_x32
io bw x32
Loop 1:
  Stuck Address      : ok
  Random Value       : ok
  Compare XOR        : ok
  Compare SUB        : ok
  Compare MUL        : ok
  Compare DIV        : ok
  Compare OR         : ok
  Compare AND        : ok
  Sequential Increment: ok
  Solid Bits         : ok
  Block Sequential   : ok
  Checkerboard       : ok
  Bit Spread        : ok
  Bit Flip          : ok
  Walking Ones       : ok
  Walking Zeroes     : ok
  8-bit Writes       : ok
  16-bit Writes      : ok
```

图 45 DDR 测试信息图

7.2. RTC 功能测试

(1) 设置软件时间

```
date -s "2023-6-16 17:23:00"
```

```
root@rk3562-buildroot:/# date
Fri Jun 16 17:25:45 UTC 2023
```

(2) 将软件的时间同步到硬件

```
hwclock -wu
```

(3) 设置硬件时钟

```
hwclock -r
```

```
root@rk3562-buildroot:/# hwclock -r
Fri Jun 16 17:23:12 2023  0.000000 seconds
```

(4) 系统启动后，会自动将硬件时钟同步到系统时钟之中，重启开发板之后，进入系统后读取系统时间，可以看到时间已经同步。

```
date
```

```
root@rk3562-buildroot:/# date
Fri Jun 16 17:31:27 UTC 2023
```

(5) 将硬件时钟同步到系统时钟

```
hwclock -s
```

第一次使用 RTC 的时候，会报错 RTC 电压不足，此时只需要先将系统时钟写入到 RTC 时钟后再使用。

```
hwclock -w
```

```
root@rk3562-buildroot:/# hwclock
[ 58.196374] rtc-pcf8563 4-0051: low voltage detected, date/time is not reliable.
hwclock: RTC_RD_TIME: Invalid argument
root@rk3562-buildroot:/# hwclock -w
root@rk3562-buildroot:/# hwclock
Thu Jan 1 00:01:20 1970 0.000000 seconds
```

7.3. 看门狗测试

watchdog 是指看门狗 (监控芯片)。看门狗，又叫 watchdog，从本质上来说就是一个定时器电路，一般有一个输入和一个输出，其中输入叫做喂狗，输出一般连接到另外一个部分的复位端，一般是连接到单片机。看门狗的功能是定期的查看芯片内部的情况，一旦发生错误就向芯片发出重启信号。看门狗命令在程序的中断中拥有最高的优先级。

(1) 输入如下命令打开看门狗并执行喂狗操作，系统不会重启

```
talowe_watchdog
```

```
root@rk3562-buildroot:/# talowe_watchdog
Watchdog Ticking Away!
```

(2) 按下 ctrl+c 结束测试程序时，停止喂狗，此时看门狗处于打开状态，10s 后系统复位；

```
root@rk3562-buildroot:/# talowe_watchdog
Watchdog Ticking Away!
^C[ 222.420813] watchdog: watchdog0: watchdog did not stop!
```

(3) 若不执行复位操作，请在预备关机的 10s 之内输入如下命令关闭看门狗即可：

```
talowe_watchdog -d
```

```
root@rk3562-buildroot:/# talowe_watchdog
Watchdog Ticking Away!
^C[ 51.856973] watchdog: watchdog0: watchdog did not stop!

root@rk3562-buildroot:/#
root@rk3562-buildroot:/# talowe_watchdog -d
Watchdog card disabled.
```

(4) 输入以下命令，将启动看门狗，并设置系统重启时间为 10s 且不执行喂狗操作，10s 后将系统将自动复位。

talowe_watchdogrestart

7.4. TF 测试

TF 卡自动挂载目录为/mnt/sdcard/,并且支持热插拔。插入 TF 卡终端显示如下信息证明 TF 卡读取成功:

```
root@rk3562-buildroot:/# [ 162.410004] mmc_host mmc1: Bus speed (slot 0) = 500000000Hz (slot req 500000000Hz, actual 500000000Hz div = 0)
[ 162.410181] mmc1: new high speed SDHC card at address aaaa
[ 162.414414] mmcblk1: mmc1:aaaa SC16G 14.8 GiB
[ 162.420653] mmcblk1: p1
[ 162.672883] FAT-fs (mmcblk1p1): utf8 is not a recommended IO charset for FAT filesystems, filesystem will be case sensitive!
[ 162.678370] FAT-fs (mmcblk1p1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
```

图 46 TF 信息打印

查看挂载目录, 通过此可查看 TF 卡的挂载目录:

df -h

```
root@rk3562-buildroot:/# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        6.0G  640M   5.1G  12% /
devtmpfs         972M   8.0K   972M   1% /dev
tmpfs            983M  108K   983M   1% /tmp
tmpfs            983M  424K   983M   1% /run
tmpfs            983M  188K   983M   1% /var/log
tmpfs            983M    0   983M   0% /dev/shm
/dev/mmcblk0p7   121M   12M   103M  11% /oem
/dev/mmcblk0p8   22G   295K   22G   1% /userdata
/dev/mmcblk1p1   15G   3.0G   12G  21% /mnt/sdcard
```

图 47 查看挂载设备信息

(1) TF 卡写入测试

将测试文件写入 TF 卡中, 其中/dev/zero 是测试文件的路径, /mnt/sdcard/test 是写入到 TF 卡的路径, 路径根据具体情况进行更改。

```
sync && echo 3 > /proc/sys/vm/drop_caches //清除缓存
dd if=/dev/zero of=/mnt/sdcard/test bs=1M count=500 conv=fsync
```

```
root@rk3562-buildroot:/# dd if=/dev/zero of=/mnt/sdcard/test bs=1M count=500 conv=fsync
500+0 records in
500+0 records out
524288000 bytes (524 MB, 500 MiB) copied, 32.0727 s, 16.3 MB/s
```

(2) TF 卡读取测试

建议重启开发板或执行清除缓存指令后再测试读取 TF 卡的速度, 输入指令如下:

```
sync && echo 3 > /proc/sys/vm/drop_caches //清除缓存
dd if=/mnt/sdcard/test of=/dev/null bs=1M
```

```
root@rk3562-buildroot:/# dd if=/mnt/sdcard/test of=/dev/null bs=1M
500+0 records in
500+0 records out
524288000 bytes (524 MB, 500 MiB) copied, 22.4733 s, 23.3 MB/s
```

(3) 先执行 `cd` 执行退出 TF 卡的挂载目录后使用 `umount` 命令卸载 TF 卡后再拔掉 TF 卡

```
cd /
umount /mnt/sdcard
```

(4) 用 `df` 命令查看，TF 卡已经被卸载。

```
root@rk3562-buildroot:/# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        6.0G  640M   5.1G  12% /
devtmpfs         972M   8.0K   972M   1% /dev
tmpfs            983M  108K   983M   1% /tmp
tmpfs            983M  424K   983M   1% /run
tmpfs            983M  188K   983M   1% /var/log
tmpfs            983M    0   983M   0% /dev/shm
/dev/mmcblk0p7   121M   12M   103M  11% /oem
/dev/mmcblk0p8   22G   295K   22G   1% /userdata
```

7.5. 调频测试

- 查看内核中支持的所有的 `cpufreq governor` 的类型。其中 `userspace` 表示用户模式，在此模式下允许其他用户程序调节 CPU 频率。

```
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_governors
```

```
conservative ondemand userspace powersave performance schedutil
```

- 查看 CPU 所支持频率的范围：

```
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies
```

```
root@rk3562-buildroot:/# cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies
408000 600000 816000 1008000 1200000
```

- 将模式设置为用户模式，频率修改为 1200000：

```
echo userspace > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
echo 1200000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
```

- 查看修改后当前频率：

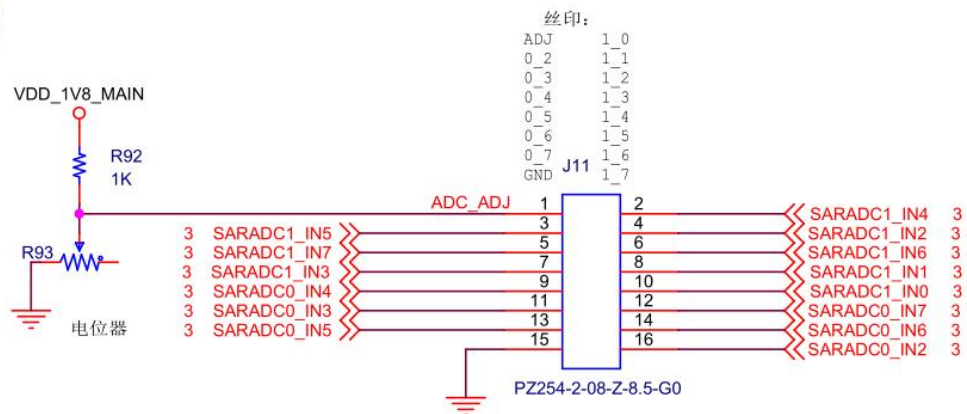
```
cat /sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_cur_freq
```

```
root@rk3562-buildroot:/# cat /sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_cur_freq
1200000
```

7.6. SARADC 功能测试

RK3562 平台引出了 ADC 接口，位于 RK3562 评估板中的 J11 座子。

ADC



SARADC0_IN5 输入电压测试（此处为 ADC0 通道 5）

```
cd /sys/bus/iio/devices/iio:device0 //ADC0
cat in_voltage5_raw //通道 5
```

```
root@rk3562-buildroot:/sys/bus/iio/devices/iio:device0# cat in_voltage5_raw
819
```

将 SARADC0_IN5 接地测试:

```
root@rk3562-buildroot:/sys/bus/iio/devices/iio:device0# cat in_voltage5_raw
0
```

可见接地后电压变化，同理，其他 ADC 也可通过该方法进行测试。

```
cat /sys/bus/iio/devices/iio:device1/in_voltage4_raw //SARADC1_IN4
cat /sys/bus/iio/devices/iio:device1/in_voltage5_raw //SARADC1_IN5
cat /sys/bus/iio/devices/iio:device0/in_voltage3_raw //SARADC0_IN3
```

7.7. emmc 测试

7.7.1. emmc 写入测试

测试 eMMC 的读写速度，以读写 ext4 文件系统为例。

```
sync && echo 3 > /proc/sys/vm/drop_caches
dd if=/dev/zero of=/test bs=1M count=500 conv=fsync
```

```
root@rk3562-buildroot:/# dd if=/dev/zero of=/test bs=1M count=500 conv=fsync
500+0 records in
500+0 records out
524288000 bytes (524 MB, 500 MiB) copied, 4.57447 s, 115 MB/s
```

7.7.2. emmc 读取测试

为确保数据准确，请重启开发板或者清除缓存后测试读取速度，测试指令如下所示：

```
sync && echo 3 > /proc/sys/vm/drop_caches
dd if=/test of=/dev/null bs=1M
```

```

root@rk3562-buildroot:/# sync && echo 3 > /proc/sys/vm/drop_caches
[ 998.092221] sh (859): drop_caches: 3
root@rk3562-buildroot:/# dd if=/test of=/dev/null bs=1M
500+0 records in
500+0 records out
524288000 bytes (524 MB, 500 MiB) copied, 1.64033 s, 320 MB/s

```

7.8. USB 测试

7.8.1. 鼠标测试

将鼠标插入 EVM-RK3562 的底板的 USB 接口上,串口终端会打印如下信息:

```

root@rk3562-buildroot:/# [ 566.148927] usb 1-1.4: new low-speed USB device number 3 using ehci-platform
[ 566.258442] usb 1-1.4: New USB device found, idVendor=10c4, idProduct=8105, bcdDevice= 0.00
[ 566.258518] usb 1-1.4: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[ 566.258546] usb 1-1.4: Product: USB OPTICAL MOUSE
[ 566.258568] usb 1-1.4: Manufacturer: YSPRINGTECH
[ 566.257536] input: YSPRINGTECH USB OPTICAL MOUSE as /devices/platform/fed00000.usb/usb1/1-1/1-1.4/1-1.4.1.0/0003:10c4:8105.0001/input/input3
[ 566.312777] hid-generic 0003:10c4:8105.0001: input,hidraw0: USB HID v1.11 Mouse [YSPRINGTECH USB OPTICAL MOUSE] on usb-fed00000.usb-1.4/input0
[08:42:14.532] event3 - YSPRINGTECH USB OPTICAL MOUSE: is tagged by udev as: Mouse
[08:42:14.533] event3 - YSPRINGTECH USB OPTICAL MOUSE: device is a pointer
[08:42:14.534] libinput: configuring device "YSPRINGTECH USB OPTICAL MOUSE".
[08:42:14.534] associating input device event3 with output LVDS-1 (none by udev)

```

7.8.2. USB2.0 测试

EVM-RK3562 底板支持 USB2.0 接口 (JU1、JU2), 用户可以在任何一个板载 USB HOST 接口上连接 USB 鼠标、USB 键盘、U 盘等设备, 并支持以上设备的热插拔。这里用挂载 U 盘为例进行演示, 目前 U 盘测试支持到 32G, 32G 以上并未测试。

插入 U 盘后终端会打印出 U 盘的信息, 不同品牌型号的 U 盘, 串口终端打印出来的信息会有所区别。打印的 U 盘信息如下:

```

root@rk3562-buildroot:/# [ 673.635606] usb 1-1.4: new high-speed USB device number 5 using ehci-platform
[ 673.738810] usb 1-1.4: New USB device found, idVendor=0951, idProduct=1666, bcdDevice= 1.10
[ 673.738884] usb 1-1.4: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 673.738910] usb 1-1.4: Product: DataTraveler 3.0
[ 673.738932] usb 1-1.4: Manufacturer: Kingston
[ 673.738953] usb 1-1.4: SerialNumber: E0D5E55CCF6E731E9251046
[ 673.743701] usb-storage 1-1.4:1.0: USB Mass Storage device detected
[ 673.746327] scsi host0: usb-storage 1-1.4:1.0
[ 674.774808] scsi 0:0:0:0: Direct-Access Kingston DataTraveler 3.0 PMAP PQ: 0 ANSI: 6
[ 674.778562] sd 0:0:0:0: [sda] 60604416 512-byte logical blocks: (31.0 GB/28.9 GiB)
[ 674.780086] sd 0:0:0:0: [sda] Write Protect is off
[ 674.781464] sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DP0 or FUA
[ 674.896697] sda: sda1
[ 674.904385] sd 0:0:0:0: [sda] Attached SCSI removable disk
[ 675.169911] FAT-fs (sda1): utf8 is not a recommended IO charset for FAT filesystems, filesystem will be case sensitive!
[ 675.174373] FAT-fs (sda1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.

```

图 48 U 盘参数打印信息

● 查看挂载目录

```
df -h
```

```

root@rk3562-buildroot:/# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        6.0G  1.2G  4.6G  20% /
devtmpfs        972M   8.0K  972M   1% /dev
tmpfs           983M  108K  983M   1% /tmp
tmpfs           983M  444K  983M   1% /run
tmpfs           983M  192K  983M   1% /var/log
tmpfs           983M    0  983M   0% /dev/shm
/dev/mmcblk0p7  121M   12M  103M  11% /oem
/dev/mmcblk0p8   22G  305K   22G   1% /userdata
/dev/mmcblk1p1   15G  3.5G   12G  24% /mnt/sdcard
/dev/sda1        29G  3.0G   26G  11% /mnt/udisk

```

● 查看 U 盘的内容。

```
ls -l /mnt/udisk
```

```
root@rk3562-buildroot:/# ls -l /mnt/udisk/
total 3052880
-rwxr-xr-x 1 root root 524288000 Aug 13 2024 500M.file
drwxr-xr-x 2 root root 16384 Jul 12 2024 'System Volume Information'
-rwxr-xr-x 1 root root 3686454 Jan 1 1980 filename.bmp
drwxr-xr-x 3 root root 16384 Aug 5 2024 ppp
```

- U 盘写入测试，不同品牌的 U 盘的写入速度有所区别。

```
sync && echo 3 > /proc/sys/vm/drop_caches
dd if=/dev/zero of=/mnt/udisk/testa bs=1M count=500 conv=fsync
```

```
root@rk3562-buildroot:/# dd if=/dev/zero of=/mnt/udisk/testa bs=1M count=500 conv=fsync
500+0 records in
500+0 records out
524288000 bytes (524 MB, 500 MiB) copied, 42.1417 s, 12.4 MB/s
```

- U 盘读取测试，由于上一个步骤写入的数据还在 U 盘的缓冲区里面，因此为了保证数据准确，重启开发板或者清除缓存之后再输入命令进行测试。

```
sync && echo 3 > /proc/sys/vm/drop_caches
dd if=/mnt/udisk/testa of=/dev/null bs=1M
```

```
root@rk3562-buildroot:/# dd if=/mnt/udisk/testa of=/dev/null bs=1M
500+0 records in
500+0 records out
524288000 bytes (524 MB, 500 MiB) copied, 23.4804 s, 22.3 MB/s
```

- U 盘使用完成后，在拔出 U 盘前，需要使用 umount 卸载

```
umount /mnt/udisk
```

7.9. 网络测试

- EVM-RK3562 开发板配有两个单口千兆网口插座（J15、J16），其中 J16 采用是将 PCIE 转为网口来使用。将网线与开发板上的网络插座进行连接，输入执行 `ifconfig -a`，查看是否存在 `eth0`、`enp1s0` 网卡、IP 地址等（网口 `Eth0`、`Eth1` 见开发板上丝印）：

```
ifconfig -a
```

```
root@rk3562-buildroot:/# ifconfig -a
can0      Link encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          NOARP MTU:16 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:10
          RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
          Interrupt:88

can1      Link encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          NOARP MTU:16 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:10
          RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
          Interrupt:89

enp1s0    Link encap:Ethernet HWaddr 00:55:7B:B5:7D:F7
          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:96

eth0      Link encap:Ethernet HWaddr 2A:52:60:DB:27:53
          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:79
```

图 49 查看网络信息

- 插入网线后会识别到网线 (eth0) 已插入

```
root@rk3562-buildroot:/# [ 826.373066] rk_gmac-dwmac ffa80000.ethernet eth0: Link is Up - 1Gbps/Full - flow control rx/tx
[ 826.373228] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[ 826.373585] [dhd] CFG80211-ERROR) wl_cfg80211_netdev_notifier_call : wdev null. Do nothing
```

- 开启网口 eth0

```
ifconfig eth0 up
```

- 自动获取 ip, 根据具体接的网口来定

```
udhcpc -i eth0
```

```
root@rk3562-buildroot:/# udhcpc -i eth0
udhcpc: started, v1.36.0
udhcpc: broadcasting discover
udhcpc: broadcasting select for 192.168.0.53, server 192.168.0.1
udhcpc: lease of 192.168.0.53 obtained from 192.168.0.1, lease time 7107
deleting routers
adding dns 192.168.0.1
```

注意：如果执行命令后仍未自动分配到网址，可以检查一下网线是否插紧、测试指令是否为所测试的网口。如果以上操作无误仍无法识别 ip，则重启开发板重复以上步骤。

- 测试以太网连接。

```
ping baidu.com -I eth0
```

```
root@rk3562-buildroot:/# ping baidu.com -I eth0
PING baidu.com (110.242.68.66) from 192.168.0.53 eth0: 56(84) bytes of data.
64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=1 ttl=49 time=42.0 ms
64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=2 ttl=49 time=41.6 ms
64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=3 ttl=49 time=41.8 ms
64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=4 ttl=49 time=41.7 ms
```

同理，测试 PCIE 网口只需要将网线插入 eth1 的网线插座中，并将命令中的

eth0 改为 enp1s0 则可以。

```
root@rk3562-buildroot:/# udhcpc -i enp1s0
udhcpc: started, v1.36.0
udhcpc: broadcasting discover
udhcpc: broadcasting select for 192.168.0.74, server 192.168.0.1
udhcpc: lease of 192.168.0.74 obtained from 192.168.0.1, lease time 6450
deleting routers
adding dns 192.168.0.1
root@rk3562-buildroot:/# ping baidu.com -I enp1s0
PING baidu.com (110.242.68.66) from 192.168.0.74 enp1s0: 56(84) bytes of data.
64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=1 ttl=49 time=41.8 ms
64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=2 ttl=49 time=41.9 ms
64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=3 ttl=49 time=41.8 ms
64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=4 ttl=49 time=41.5 ms
```

若出现域名解析错误，可以尝试重新设置 DNS。

```
root@rk3562-buildroot:/# ping baidu.com -I eth0
ping: baidu.com: Temporary failure in name resolution
```

● 设置 DNS

若需要使用到域名访问互联网，则需要配置 DNS 服务器。编辑“/etc/resolv.conf”文件（如果不存在则创建一个新的文件），并在其中添加一个或者多个 DNS 服务器的 IP 地址。在/etc/resolv.conf 添加如下内容：

```
nameserver 114.114.114.114
nameserver 8.8.8.8
```

如上添加内容即为将 114.114.114.114 作为首选的 DNS 服务器，将 8.8.8.8 作为备选的 DNS 服务器。

```
echo "nameserver 8.8.8.8" >> /etc/resolv.conf
echo "nameserver 114.114.114.114" >> /etc/resolv.conf
```

7.10. LCD 背光调节

背光的亮度设置范围为（0--255），255 表示亮度最高，0 表示关闭背光亮亮度。在 LVDS 接口上（J10）接上 LVDS 屏幕后，上电启动。进入系统后在终端输入如下命令进行背光测试。

- 查看支持背光型号并且在终端打印出来：

```
ls /sys/class/backlight
```

```
root@rk3562-buildroot:/# ls /sys/class/backlight
backlight1
```

- 查看当前屏幕背光值：

```
cat /sys/class/backlight/backlight1/brightness
```

```
root@rk3562-buildroot:/# cat /sys/class/backlight/backlight1/brightness
200
```

- 背光熄灭：

```
echo 0 > /sys/class/backlight/backlight1/brightness
```

- 背光亮起:

```
echo 100 > /sys/class/backlight/backlight1/brightness
```

7.11. 放/录音测试

7.11.1. SPKOUT 播放声音

将耳机插到 EVM-RK3562 的底板之上用于播放声音,也可将外放设备接入 (J2 端子) 进行音频的播放。

获取音频捕获设备列表, 通过如下命令可以获取用于录音的设备:

```
arecord -l
```

```
root@rk3562-buildroot:/# arecord -l
**** List of CAPTURE Hardware Devices ****
card 0: rockchiprk809 [rockchip-rk809], device 0: daiLink-multicodecs rk817-hifi-0 [daiLink-multicodecs rk817-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

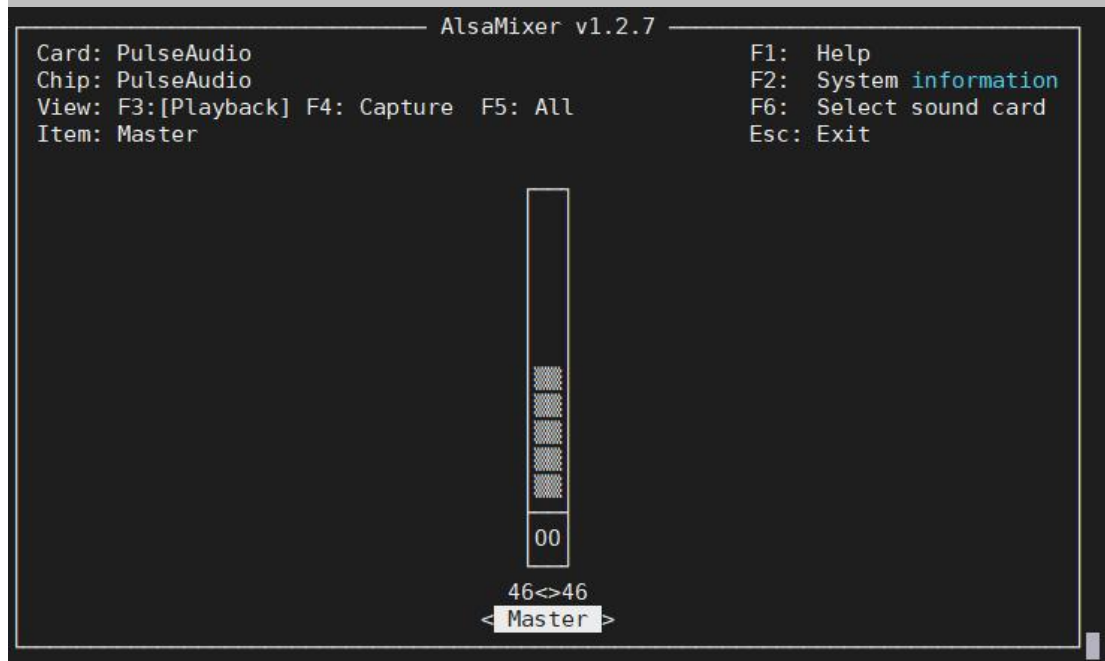
播放音频:

```
gst-play-1.0 /talowe_test/test.mp3
```

```
root@rk3562-buildroot:/# gst-play-1.0 /talowe_test/test.mp3
Press 'k' to see a list of keyboard shortcuts.
Now playing /talowe_test/test.mp3
Redistribute latency...
Redistribute latency...
0:00:06.5 / 0:04:30.3
```

音频播放音量的调节:

```
alsamixer
```



7.11.2. MIC 录音测试

- 采集声音 10 秒，并且保存为 wav 格式（按 CTRL+C 可提前结束录制）。

```
arecord -d 10 -f cd -t wav test1.wav
```

```
root@rk3562-buildroot:/# arecord -d 10 -f cd -t wav test1.wav
Recording WAVE 'test1.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
^CAborted by signal Interrupt...
```

- 使用耳机播放采集的声音。

```
aplay test1.wav
```

```
root@rk3562-buildroot:/# aplay test1.wav
Playing WAVE 'test1.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
root@rk3562-buildroot:/#
```

7.12. SQLite3 测试

- SQLite3 是一款轻型的数据库，是遵守 ACID 的关系型数据库管理系统，占用资源低。EVM-RK3562 开发板移植的是 3.39.2 版本的 sqlite3。

```
sqlite3
```

```
root@rk3562-buildroot:/# sqlite3
SQLite version 3.39.2 2022-07-21 15:24:47
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
```

- 创建表 tbl1

```
sqlite> create table tbl1 (one varchar(10), two smallint);
```

- tbl1 表内插入数据

```
sqlite> insert into tbl1 values('twdz!',10);
sqlite> insert into tbl1 values('best wish to you!',20);
```

- 查询表 tbl1 中的内容

```
sqlite> select * from tbl1;
```

```
sqlite> select * from tbl1;
twdz!|10
best wish to you!|20
```

- 删除表 tal1 中的 twdz! 数据

```
sqlite> delete from tbl1 where one = 'twdz!';
```

- 再查询表 tbl1 中的内容

```
sqlite> select * from tbl1;
```

```
sqlite> select * from tbl1;  
best wish to you!|20
```

- 退出数据库（或者使用.exit 命令）

```
sqlite> .quit  
root@rk3562-buildroot:/#
```

7.13. 4G/5G 测试

在开发板启动之前接入 4G/5G 模块，注意需要在 4G/5G 模块上插上天线，并插入 SIM 卡，启动开发板。

- (1) 在/dev 目录下查看设备节点的状态

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

```
root@rk3562-buildroot:/# ls /dev/ttyUSB*  
ttyUSB0 ttyUSB1 ttyUSB2 ttyUSB3
```

- (2) 设备识别成功之后，输入下面指令(指令后加 “&” 表示后台运行)进行拨号上网测试。

```
quectel-CM &
```

```
root@rk3562-buildroot:/# quectel-CM &  
[2] 1190  
root@rk3562-buildroot:/# [08-16_06:44:37:638] Quectel_QConnectManager_Linux_V1.6.0.24  
[08-16_06:44:37:639] Find /sys/bus/usb/devices/1-1.1 idVendor=0x2c7c idProduct=0x125, bus=0x001, dev=0x004  
[08-16_06:44:37:640] Auto find qmichannel = /dev/cdc-wdm0  
[08-16_06:44:37:640] Auto find usbnet_adapter = wwan0  
[08-16_06:44:37:640] netcard driver = qmi_wwan, driver version = 5.10.198  
[08-16_06:44:37:640] Modem works in QMI mode  
[08-16_06:44:37:648] /proc/1155/fd/7 -> /dev/cdc-wdm0  
[08-16_06:44:37:648] /proc/1155/exe -> /usr/bin/quectel-CM  
[08-16_06:44:37:648] requestDeactivateDefaultPDP WdsConnectionIPv4Handle  
[08-16_06:44:37:677] ifconfig wwan0 0.0.0.0  
[08-16_06:44:37:690] ifconfig wwan0 down  
[ 125.427891] [dhd] CFG80211-ERROR) wl_cfg80211_netdev_notifier_call : wdev null. Do nothing  
[ 125.432704] [dhd] CFG80211-ERROR) wl_cfg80211_netdev_notifier_call : wdev null. Do nothing  
[08-16_06:44:37:871] QmiWwanThread exit  
[08-16_06:44:37:873] qmi_main exit  
[08-16_06:44:39:650] cdc_wdm fd = 7  
[08-16_06:44:39:724] Get clientWDS = 7  
[08-16_06:44:39:757] Get clientDMS = 1  
[08-16_06:44:39:789] Get clientNAS = 2  
[08-16_06:44:39:820] Get clientUIM = 1  
[08-16_06:44:39:852] Get clientWDA = 1  
[08-16_06:44:39:884] requestBaseBandVersion EC20CEHDLGR06A10M1G  
[08-16_06:44:40:012] requestGetSIMStatus SIMStatus: SIM_READY  
[08-16_06:44:40:044] requestGetProfile[1] ctnet///0  
[08-16_06:44:40:077] requestRegistrationState2 MCC: 460, MNC: 11, PS: Attached, DataCap: LTE  
[08-16_06:44:40:109] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED  
[08-16_06:44:40:109] ifconfig wwan0 0.0.0.0  
[ 127.847839] [dhd] CFG80211-ERROR) wl_cfg80211_netdev_notifier_call : wdev null. Do nothing  
[ 127.849581] [dhd] CFG80211-ERROR) wl_cfg80211_netdev_notifier_call : wdev null. Do nothing  
[08-16_06:44:40:125] ifconfig wwan0 down  
[ 127.862408] [dhd] CFG80211-ERROR) wl_cfg80211_netdev_notifier_call : wdev null. Do nothing  
[ 127.867092] [dhd] CFG80211-ERROR) wl_cfg80211_netdev_notifier_call : wdev null. Do nothing  
[08-16_06:44:40:205] requestSetupDataCall WdsConnectionIPv4Handle: 0x86deade0  
[08-16_06:44:40:334] ifconfig wwan0 up  
[ 128.072476] [dhd] CFG80211-ERROR) wl_cfg80211_netdev_notifier_call : wdev null. Do nothing  
[ 128.073508] [dhd] CFG80211-ERROR) wl_cfg80211_netdev_notifier_call : wdev null. Do nothing  
[08-16_06:44:40:349] busybox udhcpd -f -n -q -t 5 -i wwan0  
udhcpd: started, v1.36.0  
udhcpd: broadcasting discover  
udhcpd: broadcasting select for 10.20.3.59, server 10.20.3.60  
udhcpd: lease of 10.20.3.59 obtained from 10.20.3.60, lease time 7200  
[08-16_06:44:40:620] deleting routers  
[08-16_06:44:40:669] adding dns 202.96.134.33  
[08-16_06:44:40:669] adding dns 202.96.128.166
```

查看是否生成了 wwan0 网卡

```
ifconfig -a
```

```
wwan0    Link encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
inet addr:10.20.3.59 P-t-P:10.20.3.59 Mask:255.255.255.248
inet6 addr: fe80::aea8:alc:f710:33cc/64 Scope:Link
UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1500 Metric:1
RX packets:44 errors:0 dropped:0 overruns:0 frame:0
TX packets:73 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:6443 (6.2 KiB) TX bytes:7596 (7.4 KiB)
```

(3) 4G 信号检测

如果执行脚本后无法正常拨号上网，可用以下方法进行调试：

检查 4G 模块信号是否正常，通过串口调试指令检测

```
microcom /dev/ttyUSB2
AT+CSQ
```

#执行指令后，还需输入 AT 指令

#该指令用于检查网络强度和 SIM 卡情况

```
root@rk3562-buildroot:/# microcom /dev/ttyUSB2
AT+CSQ
+CSQ: 20,99
OK
```

0-31之间 若该值为99 则为无信号

退出串口调试界面无法通过 Ctrl+c 退出，应通过 Ctrl+x 退出。

若信号异常，可用该指令多试几次操作：

```
echo -e "AT+CFUN=1,1\r\n" >/dev/ttyUSB2 #重新使能 4g 模块
```

若信号仍处于异常状态，请检查天线或 SIM 卡状态是否完好。

(4) Ping 百度进行测试

```
ping www.baidu.com -I wwan0
```

```
root@rk3562-buildroot:/# ping baidu.com -I wwan0
PING baidu.com (110.242.68.66) from 10.20.3.59 wwan0: 56(84) bytes of data.
64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=1 ttl=49 time=99.1 ms
64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=2 ttl=49 time=137 ms
64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=3 ttl=49 time=132 ms
64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=4 ttl=49 time=105 ms
64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=5 ttl=49 time=84.0 ms
```

若出现以下情况，则需追加域名解析服务器在进行 Ping 百度测试：

```
root@rk3562-buildroot:/# ping baidu.com -I eth0
ping: baidu.com: Temporary failure in name resolution
```

```
echo "nameserver 8.8.8.8" >> /etc/resolv.conf
```

```
echo "nameserver 114.114.114.114" >> /etc/resolv.conf
```

(5) 4G 测试是完成之后为方便后续测试手动杀掉该进程。也可直接重启。

```
ps -ef
```

```
root 1190 675 0 06:44 ttyFIQ0 00:00:00 quectel-CM
root 1219 2 0 06:47 ? 00:00:00 [kworker/0:0-events_freezable]
root 1221 2 0 06:49 ? 00:00:00 [kworker/u8:0-devfreq_wq]
root 1222 2 0 06:52 ? 00:00:00 [kworker/0:1-events_freezable]
root 1227 675 0 06:56 ttyFIQ0 00:00:00 ps -ef
```

kill 1075

至此，4G 测试完毕！

7.14. WIFI 测试

以下操作连接的 wifi 名称是：Talowe-2.4G-0, 密码为：Talowe888（wifi 名称和密码根据自己的来）。

(1) 输入如下命令连接 wifi: wifi start.sh+ssid+密码

wifi_start.sh Talowe-2.4G-0 Talowe888

```
root@rk3562-buildroot:/# wifi_start.sh Talowe-2.4G-0 Talowe888
connect to WiFi ssid: Talowe-2.4G-0, Passwd: Talowe888
[ 657.509741] [dhd] WLC_E_IF: NO IF set, event Ignored
[ 657.509939] [dhd] CFGP2P-ERROR) wl_cfgp2p_disable_discovery : do nothing, not initialized
[ 657.564179] [dhd] CFGP2P-ERROR) wl_cfgp2p_del_p2p_disc_if : P2P interface unregistered
[ 657.645596] [dhd] [wlan0] wl_cfg80211_disconnect : Reason 3, act 1, bssid f8:6f:b0:b0:5c:7e
[ 657.654935] [dhd] [wlan0] wl_iw_event : [0 times] disconnected with f8:6f:b0:b0:5c:7e, event 11, reason 8
[ 657.655008] [dhd] [wlan0] wl_handle_link_down : Link down: WLC_E_LINK(16), reason 2 from f8:6f:b0:b0:5c:7e
[ 657.655051] [dhd] [wlan0] wl_ext_iapsta_event : [S] Link down with f8:6f:b0:b0:5c:7e, WLC_E_DISASSOC(11), reason 8
[ 657.655093] [dhd] [wlan0] wl_handle_link_down : Disconnect event sent to upper layer: event 16 e->reason=33554432 reason=2 ie_len=0 from f8:6f:b0:b0:5c:7e
[ 657.655163] [dhd] [wlan0] wl_iw_event : Link Down with f8:6f:b0:b0:5c:7e, reason=2
[ 657.655183] [dhd] [wlan0] wl_ext_iapsta_event : [S] Link down with f8:6f:b0:b0:5c:7e, WLC_E_LINK(16), reason 2
Successfully initialized wpa supplicant
[ 658.582969] [dhd] P2P interface registered
nl80211: kernel reports: Registration to specific type not supported
[ 658.605489] [dhd] WLC_E_IF: NO IF set, event Ignored
root@rk3562-buildroot:/# [ 658.658559] [dhd] [wlan0] wl_run_escan : LEGACY_SCAN sync ID: 4, bssid: 0
[ 661.018457] [dhd] CFG80211-ERROR) wl_set_set_cipher : set wsec_info error (0)
[ 661.024101] [dhd] [wlan0] wl_conn_debug_info : Connecting with f8:6f:b0:b0:5c:7e ssid "Talowe-2.4G-0", len (13), channel=2g-1(chan_cr
[ 661.123303] [dhd] [wlan0] wl_iw_event : Link UP with f8:6f:b0:b0:5c:7e
[ 661.123398] [dhd] [wlan0] wl_ext_iapsta_event : [S] Link UP with f8:6f:b0:b0:5c:7e
[ 661.127567] [dhd] [wlan0] wl_bss_connect_done : Report connect result - connection succeeded
[ 661.144932] [dhd] [wlan0] wl_add_keyext : key index (0)
```

连接成功

图 51 WiFi 连接成功

(2) 自动获取 IP:

udhcpc -i wlan0

```
root@rk3562-buildroot:/# udhcpc -i wlan0
udhcpc: started, v1.36.0
udhcpc: broadcasting discover
udhcpc: broadcasting select for 192.168.0.150, server 192.168.0.1
udhcpc: lease of 192.168.0.150 obtained from 192.168.0.1, lease time 6800
deleting routers
route: SIOCADDRT: Network is unreachable
adding dns 192.168.0.1
root@rk3562-buildroot:/#
```

(3) 输入如下命令 ping 外网:

ping baidu.com -I wlan0

```
root@rk3562-buildroot:/# ping baidu.com -I wlan0
PING baidu.com (110.242.68.66) from 192.168.0.48 wlan0: 56(84) bytes of data.
64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=1 ttl=49 time=101 ms
64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=3 ttl=49 time=71.8 ms
64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=4 ttl=49 time=66.9 ms
64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=5 ttl=49 time=74.6 ms
64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=6 ttl=49 time=80.5 ms
64 bytes from 110.242.68.66 (110.242.68.66): icmp_seq=7 ttl=49 time=71.8 ms
```

图 52 网络连接测试

7.15. 蓝牙测试

EVM-RK3562 中使用的为 AP6256 模块，其集成了蓝牙模块和 wifi 模块。

(1) 打开蓝牙设备节点并查看

<code>hciconfig hci0 up</code>	#打开蓝牙设备节点
<code>hciconfig -a</code>	#查看蓝牙设备信息

```
root@rk3562-buildroot:/# hciconfig hci0 up
root@rk3562-buildroot:/# hciconfig -a
hci0:    Type: Primary  Bus: UART
        BD Address: B8:13:32:9A:8D:97  ACL MTU: 1021:8  SCO MTU: 64:1
        UP RUNNING
        RX bytes:901 acl:0 sco:0 events:65 errors:0
        TX bytes:3783 acl:0 sco:0 commands:65 errors:0
        Features: 0xbf 0xfe 0xcf 0xfe 0xdb 0xff 0x7b 0x87
        Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
        Link policy: RSWITCH SNIFF
        Link mode: PERIPHERAL ACCEPT
        Name: 'BlueZ 5.68'
        Class: 0x6c0000
        Service Classes: Rendering, Capturing, Audio, Telephony
        Device Class: Miscellaneous,
        HCI Version: 5.2 (0xb)  Revision: 0x466
        LMP Version: 5.2 (0xb)  Subversion: 0x6606
        Manufacturer: Broadcom Corporation (15)
```

图 53 蓝牙节点信息

(2) Obexd 和 bluetoothd 蓝牙守护进程的开启

```
/usr/libexec/bluetooth/bluetoothd -C -E -d -n & #开机已默认开启该配置
#开启 obexd 进程
export $(dbus-launch)
/usr/libexec/bluetooth/obexd -r / -a -d -n &
```

(3) 蓝牙配对命令

<code>bluetoothctl</code>	#打开 bluez 蓝牙工具
<code>power on</code>	#启动蓝牙设备
<code>discoverable on</code>	#设置为可发现模式
<code>agent on</code>	#启动代理
<code>default-agent</code>	#设置当前代理为默认

(4) 开发板从机配对

当蓝牙设置为可被发现模式后，可以通过手机从而进行与开发板蓝牙的配对。
打开手机蓝牙设置，此处会发现一个“BlueZ 5.68”的设备，点击进行配对。

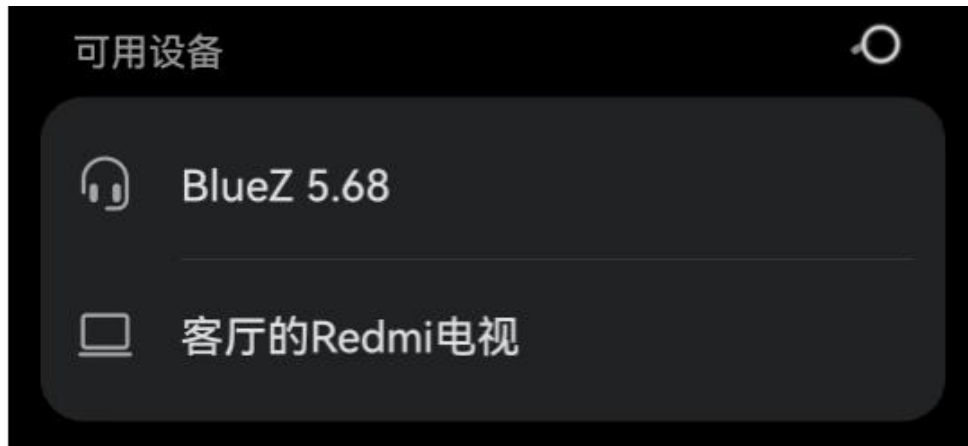


图 54 连接蓝牙

在开发板上确认配对，输入 yes

```
AdvertisementMonitor path registered
[bluetooth]# power on
Changing power on succeeded
[bluetooth]# discoverable on
Changing discoverable on succeeded
[CHG] Controller B8:13:32:9A:8D:97 Discoverable: yes
[bluetooth]# agent on
Agent is already registered
[bluetooth]# default-agent
Default agent request successful
[CHG] Device 30:AA:E4:CA:28:98 Connected: yes
Request confirmation
[agent] Confirm passkey 466727 (yes/no):
```

手机点击配对后会弹出该信息 输入yes 确定配对

```
Request confirmation
[agent] Confirm passkey 778327 (yes/no): yes
Authorize service
[agent] Authorize service 00001108-0000-1000-8000-00805f9b34fb (yes/no): Request canceled
Authorize service
[agent] Authorize service 0000110d-0000-1000-8000-00805f9b34fb (yes/no): yes
```

(5) 开发板主机模式进行配对

由于上一个步骤已经配对过了开发板的蓝牙，为能顺利的让开发板配对上手机端，我们要先移除手机蓝牙设备。执行如下指令：

```
devices #查看已检索到的设备
[bluetooth]# devices
Device 30:AA:E4:CA:28:98 YWF
remove 30:AA:E4:CA:28:98 #此处根据实际的 MAC 地址来填写
[bluetooth]# remove 30:AA:E4:CA:28:98
[DEL] Device 30:AA:E4:CA:28:98 YWF
Device has been removed
```

打开蓝牙扫描功能

```
scan on #开启蓝牙的扫描功能
[bluetooth]# scan on
Discovery started
[CHG] Controller B8:13:32:9A:8D:97 Discovering: yes
[NEW] Device 7A:D2:B2:9B:5D:A8 7A-D2-B2-9B-5D-A8
[NEW] Device 08:F0:B6:2D:F1:72 EDIFIER AtomBuds
[NEW] Device 30:AA:E4:CA:28:98 YWF
[DEL] Device 7A:D2:B2:9B:5D:A8 7A-D2-B2-9B-5D-A8
[NEW] Device 7A:D2:B2:9B:5D:A8 7A-D2-B2-9B-5D-A8
[NEW] Device 7D:F2:38:02:D4:7B 7D-F2-38-02-D4-7B
```

当扫描到自己所需要的设备后，关闭扫描并且查看设备

```
scan off
devices
```

```
[bluetooth]# scan off
Discovery stopped
[CHG] Device 7D:F2:38:02:D4:7B TxPower is nil
[CHG] Device 7D:F2:38:02:D4:7B RSSI is nil
[CHG] Device 7A:D2:B2:9B:5D:A8 RSSI is nil
[CHG] Device 30:AA:E4:CA:28:98 RSSI is nil
[CHG] Device 08:F0:B6:2D:F1:72 RSSI is nil
[CHG] Controller B8:13:32:9A:8D:97 Discovering: no
[bluetooth]# devices
Device 08:F0:B6:2D:F1:72 EDIFIER AtomBuds
Device 30:AA:E4:CA:28:98 YWF
Device 7A:D2:B2:9B:5D:A8 7A-D2-B2-9B-5D-A8
Device 7D:F2:38:02:D4:7B 7D-F2-38-02-D4-7B
```

连接设备（根据自己的实际 MAC 地址来进行连接）

```
pair 30:AA:E4:CA:28:98
```

```
[bluetooth]# pair 30:AA:E4:CA:28:98
Attempting to pair with 30:AA:E4:CA:28:98
[CHG] Device 30:AA:E4:CA:28:98 Connected: yes
Request confirmation
[agent] Confirm passkey 219715 (yes/no): yes
[CHG] Device 30:AA:E4:CA:28:98 Bonded: yes
[CHG] Device 30:AA:E4:CA:28:98 Modalias: bluetooth:v010Fp107Ed1436
[CHG] Device 30:AA:E4:CA:28:98 UUIDs: 0000046a-0000-1000-8000-00805f9b34fb
[CHG] Device 30:AA:E4:CA:28:98 UUIDs: 00001105-0000-1000-8000-00805f9b34fb
[CHG] Device 30:AA:E4:CA:28:98 UUIDs: 0000110a-0000-1000-8000-00805f9b34fb
[CHG] Device 30:AA:E4:CA:28:98 UUIDs: 0000110c-0000-1000-8000-00805f9b34fb
[CHG] Device 30:AA:E4:CA:28:98 UUIDs: 00001112-0000-1000-8000-00805f9b34fb
[CHG] Device 30:AA:E4:CA:28:98 UUIDs: 00001115-0000-1000-8000-00805f9b34fb
[CHG] Device 30:AA:E4:CA:28:98 UUIDs: 00001116-0000-1000-8000-00805f9b34fb
[CHG] Device 30:AA:E4:CA:28:98 UUIDs: 0000111f-0000-1000-8000-00805f9b34fb
[CHG] Device 30:AA:E4:CA:28:98 UUIDs: 0000112f-0000-1000-8000-00805f9b34fb
[CHG] Device 30:AA:E4:CA:28:98 UUIDs: 00001132-0000-1000-8000-00805f9b34fb
[CHG] Device 30:AA:E4:CA:28:98 UUIDs: 00001200-0000-1000-8000-00805f9b34fb
[CHG] Device 30:AA:E4:CA:28:98 UUIDs: 00001800-0000-1000-8000-00805f9b34fb
[CHG] Device 30:AA:E4:CA:28:98 UUIDs: 00001801-0000-1000-8000-00805f9b34fb
[CHG] Device 30:AA:E4:CA:28:98 UUIDs: 0000fe35-0000-1000-8000-00805f9b34fb
[CHG] Device 30:AA:E4:CA:28:98 UUIDs: 11c8b310-80e4-4276-afc0-f81590b2177f
[CHG] Device 30:AA:E4:CA:28:98 UUIDs: 8ce255c0-200a-11e0-ac64-0800200c9a66
[CHG] Device 30:AA:E4:CA:28:98 UUIDs: 9664aa26-d76c-43ad-9775-d310f253a408
[CHG] Device 30:AA:E4:CA:28:98 ServicesResolved: yes
[CHG] Device 30:AA:E4:CA:28:98 Paired: yes
Pairing successful
```

图 55 蓝牙连接成功

- 连接成功之后手机通过蓝牙给开发板发送图片（此处使用华为设备，其他设备同理）



图 56 手机发送图片给开发板

接收到的图片放在开发板的 / 目录下：

```
root@rk3562-buildroot:/# ls
bin                info               media              rockchip-test     sys                udisk
busybox.fragment  lib               mnt               root               system            userdata
data              lib64            oem               run               talowe_test       usr
dev              linuxrc          opt               sbin              test.jpg          var
etc              lost+found       proc             sdcard            tmp              vendor
```

将该图片拷贝到 U 盘里面，在电脑上打开图片：

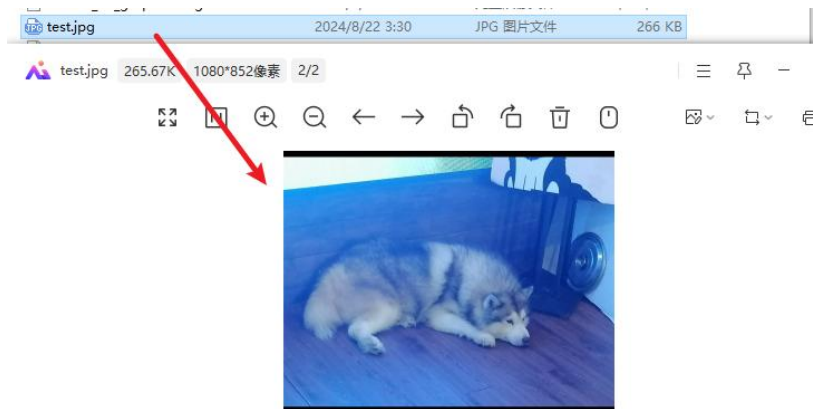


图 57 开发板接收到的图片

至此，手机发送给开发板图片成功！！！！

- 用开发板给手机发送文件
输入如下指令开启 obexctl

```
obexctl
```

```
root@rk3562-buildroot:/# obexctl
[NEW] Client /org/bluez/obex
[obex]#
```

连接设备：

```
connect 30:AA:E4:CA:28:98
```

```
[obex]# connect F8:71:0C:28:72:13
Attempting to connect to F8:71:0C:28:72:13
[NEW] Session /org/bluez/obex/client/session4 [default]
[NEW] ObjectPush /org/bluez/obex/client/session4
Connection successful
```

将/talowe/test.mp3 文件发送到手机，输入指令后在手机中进行确认则开始传输：

```
send /talowe_test/test.mp3
```

```
[30:AA:E4:CA:28:98]# obexd[928]: obexd/client/transfer.c:obc_transfer_register() 0x5563b835b0 registered /org/bluez/obex/client/session1/transfer0
obexd[928]: obexd/client/session.c:obc_session_ref() 0x5563bac370: ref=3
obexd[928]: obexd/client/session.c:obc_session_ref() 0x5563bac370: ref=4
obexd[928]: obexd/client/session.c:session_process_transfer() Transfer(0x5563b835b0) started
obexd[928]: obexd/client/session.c:obc_session_unref() 0x5563bac370: ref=3
[NEW] Transfer /org/bluez/obex/client/session1/transfer0
Transfer /org/bluez/obex/client/session1/transfer0
Status: queued
Name: test.mp3
Size: 432595
Filename: talowe_test/test.mp3
Session: /org/bluez/obex/client/session1
[CHG] Transfer /org/bluez/obex/client/session1/transfer0 Status: active
[CHG] Transfer /org/bluez/obex/client/session1/transfer0 Transferred: 8046 (@8KB/s 00:56)
[CHG] Transfer /org/bluez/obex/client/session1/transfer0 Transferred: 24204 (@16KB/s 04:26)
[CHG] Transfer /org/bluez/obex/client/session1/transfer0 Transferred: 161547 (@137KB/s 00:30)
[CHG] Transfer /org/bluez/obex/client/session1/transfer0 Transferred: 258495 (@96KB/s 00:41)
[CHG] Transfer /org/bluez/obex/client/session1/transfer0 Transferred: 355443 (@96KB/s 00:40)
[CHG] Transfer /org/bluez/obex/client/session1/transfer0 Transferred: 460470 (@105KB/s 00:36)
[CHG] Transfer /org/bluez/obex/client/session1/transfer0 Transferred: 540330 (@88KB/s 00:42)
```

传输进度

图 58 开发板发送文件进度

```
obexd[928]: obexd/client/transfer.c:obc_transfer_unregister() 0x5563b835b0 unregistered /org/bluez/obex/client/session1/transfer0
obexd[928]: obexd/client/transfer.c:obc_transfer_free() 0x5563b835b0
obexd[928]: obexd/client/session.c:obc_session_unref() 0x5563bac370: ref=3
obexd[928]: obexd/client/session.c:obc_session_unref() 0x5563bac370: ref=2
[CHG] Transfer /org/bluez/obex/client/session1/transfer0 Status: complete
[DEL] Transfer /org/bluez/obex/client/session1/transfer0
```

传输成功

图 60 文件传输完成

最后输入 exit 退出 obexctl

至此，蓝牙测试全部完成！！！！

7.16. CAN 测试

EVM-RK3562 底板有两路 CAN 总线接口，位于底板的 J11 座子在开发板终端执行如下命令查看 CAN 网络设备：

```
ifconfig -a
```

```
root@rk3562-buildroot:/# ifconfig -a
can0    Link encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
        NOARP MTU:16 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:10
        RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
        Interrupt:88

can1    Link encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
        NOARP MTU:16 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:10
        RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
        Interrupt:89

enp1s0  Link encap:Ethernet HWaddr 00:55:7B:B5:7D:F7
        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:96 Base address:0x0000
```

图 61 can 网络设备

便捷测试：将 CAN 0 的 H 端子与 CAN 1 设备 H 端连接；CAN0 的 L 端

子与其 CAN 1 设备 L 端子连接。进行 can 回传测试。

(1) 设备 CAN 设备波特率

```
ip link set can0 type can bitrate 500000
ip link set can1 type can bitrate 500000
```

(2) 打开 CAN 设备

```
ifconfig can0 up
ifconfig can1 up
```

```
root@rk3562-buildroot:/# ip link set can0 type can bitrate 500000
root@rk3562-buildroot:/# ip link set can1 type can bitrate 500000
root@rk3562-buildroot:/# ifconfig can0 up
[ 110.258712] IPv6: ADDRCONF(NETDEV_CHANGE): can0: link becomes ready
root@rk3562-buildroot:/# ifconfig can1 up
[ 114.506676] IPv6: ADDRCONF(NETDEV_CHANGE): can1: link becomes ready
```

(3) 客户端发送数据服务端接收数据，can0 设备当服务端（服务端先执行以下命令）

```
candump can0 &
```

(4) CAN1 设备当客户端（客户端发送数据）

```
cansend can1 1F334455#1122334455667788
```

(5) CAN0 接收到 CAN1 发送的数据

```
root@rk3562-buildroot:/# candump can0 &
[1] 1381
root@rk3562-buildroot:/# cansend can1 1F334455#1122334455667788
can0 1F334455 [8] 11 22 33 44 55 66 77 88
root@rk3562-buildroot:/# cansend can1 1F334455#1122334455667788
can0 1F334455 [8] 11 22 33 44 55 66 77 88
root@rk3562-buildroot:/# cansend can1 1F334455#1122334455667788
can0 1F334455 [8] 11 22 33 44 55 66 77 88
```

```
killall candump //关闭 can0 后台程序
```

也可 CAN1 当服务端接收数据，CAN0 当发送端发送数据

```
candump can1 &
cansend can0 1F334455#1122334455667788
```

```
root@rk3562-buildroot:/# candump can1 &
[1] 1486
root@rk3562-buildroot:/# cansend can0 1F334455#1122334455667788
can1 1F334455 [8] 11 22 33 44 55 66 77 88
root@rk3562-buildroot:/# cansend can0 1F334455#1122334455667788
can1 1F334455 [8] 11 22 33 44 55 66 77 88
root@rk3562-buildroot:/# cansend can0 1F334455#1122334455667788
can1 1F334455 [8] 11 22 33 44 55 66 77 88
```

CAN 与 PC 通讯需要 USB-CAN 和特定的测试软件，通过 USB-CAN 来进行 CAN 测试，先设置对应 CAN 的波特率并打开 CAN

```
root@rk3562-buildroot:/# ip link set can0 type can bitrate 500000
root@rk3562-buildroot:/# ifconfig can0 up
[ 43.234838] IPv6: ADDRCONF(NETDEV_CHANGE): can0: link becomes ready
```

USB-CAN 发送数据到 can0

```
root@rk3562-buildroot:/# candump can0 &
[1] 1331
root@rk3562-buildroot:/# can0 000 [8] 00 01 02 03 04 05 06 07
can0 000 [8] 00 01 02 03 04 05 06 07
can0 000 [8] 00 01 02 03 04 05 06 07
```

GCAN Tools

文件 操作 视图 窗口 帮助

CanOpen OBD II 曲线Curver 智能解码 DBC Rec DBC Send 伺服Servo 工作模式 数据转发 滚动显示帧数: 100000

CAN1 Receive/Transmit CAN2 Receive/Transmit

保存数据 实时保存 暂停显示 显示模式 清除 滤波设置 高级屏蔽 显示错误帧 双通道合并 错误帧率: 0

序号	帧间隔时间us	名称	帧ID	帧类型	帧格式	DLC	数据	帧数量
00000001	012.423.015	发送成功	000	DATA	STANDARD	8	00 01 02 03 04 05 06 07	1
00000002	000.551.841	发送成功	000	DATA	STANDARD	8	00 01 02 03 04 05 06 07	1
00000003	000.432.010	发送成功	000	DATA	STANDARD	8	00 01 02 03 04 05 06 07	1

CAN1 发送数据到 USB-CAN:

```
root@rk3562-buildroot:/# cansend can0 1F334455#1122334455667788
can0 1F334455 [8] 11 22 33 44 55 66 77 88
root@rk3562-buildroot:/# cansend can0 1F334455#1122334455667788
can0 1F334455 [8] 11 22 33 44 55 66 77 88
root@rk3562-buildroot:/# cansend can0 1F334455#1122334455667788
can0 1F334455 [8] 11 22 33 44 55 66 77 88
root@rk3562-buildroot:/# cansend can0 1F334455#1122334455667788
can0 1F334455 [8] 11 22 33 44 55 66 77 88
```

GCAN Tools

文件 操作 视图 窗口 帮助

CanOpen OBD II 曲线Curver 智能解码 DBC Rec DBC Send 伺服Servo 工作模式 数据转发 滚动显示帧数: 100

CAN1 Receive/Transmit CAN2 Receive/Transmit

保存数据 实时保存 暂停显示 显示模式 清除 滤波设置 高级屏蔽 显示错误帧 双通道合并 错误帧率: 100

序号	帧间隔时间us	名称	帧ID	帧类型	帧格式	DLC	数据	帧数量
00000001	020.970.113	接收	1F334455	DATA	EXTENDED	8	11 22 33 44 55 66 77 88	1
00000002	000.400.107	接收	1F334455	DATA	EXTENDED	8	11 22 33 44 55 66 77 88	1
00000003	000.351.892	接收	1F334455	DATA	EXTENDED	8	11 22 33 44 55 66 77 88	1
00000004	002.200.638	接收	1F334455	DATA	EXTENDED	8	11 22 33 44 55 66 77 88	1

至此，CAN 测试全部完成。

7.17. UART 测试

7.17.1. RS232 (UART2 CON5 端子) 测试

EVM-RK3562 底板上有两路引出的 RS232 串口，通过把底板的串口和电脑端的串口调试工具相连接来进行串口通讯测试。

串口工具选择波特率为 115200。

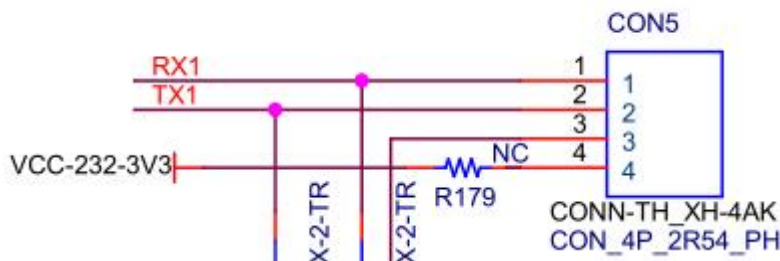


图 62 RS232 的管脚图

测试指令：

```
stty -F /dev/ttyS2 -echo raw
```

#设串口为 raw 模式并关闭回显

```
stty -F /dev/ttyS2 ispeed 115200 ospeed 115200 cs8 -parity
```

```
cat /dev/ttyS2 &
```

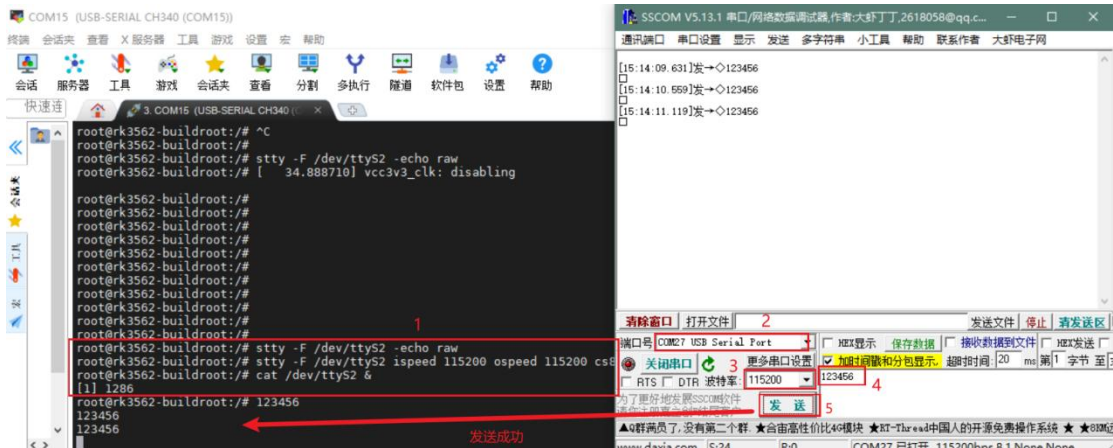
#串口 2 后台读取数据

```
echo talowe_uart_test > /dev/ttyS2
```

#向串口 2 写入数据

设置好串口工具后发送数据，板子能够接收到串口工具所发送的数据：

注意：串口工具 RTS、DTR（流控）不能打开，不然有可能接受不到数据



输入指令板子发送数据，发送后，能够在串口工具中看到板子所发送数据：

```
root@rk3562-buildroot:/# ^C
root@rk3562-buildroot:/# echo talowe_uart_test > /dev/ttyS2
root@rk3562-buildroot:/# echo talowe_uart_test > /dev/ttyS2
root@rk3562-buildroot:/# echo talowe_uart_test > /dev/ttyS2
root@rk3562-buildroot:/# echo talowe_uart_test > /dev/ttyS2
root@rk3562-buildroot:/#
```

123456
123456
123456

15:14:36.034]收←talowe_uart_test
15:14:36.578]收←talowe_uart_test
15:14:37.026]收←talowe_uart_test
15:14:37.442]收←talowe_uart_test
15:14:37.842]收←talowe_uart_test

也可以不使用串口工具，通过短接 RS232 座子的 TX、RX 进行回传测试。

```
root@rk3562-buildroot:/# stty -F /dev/ttyS2 -echo raw
root@rk3562-buildroot:/# stty -F /dev/ttyS2 ispeed 115200 ospeed 115200 cs8 -parity
root@rk3562-buildroot:/# cat /dev/ttyS2 &
[1] 1656
root@rk3562-buildroot:/# echo talowe_uart_test > /dev/ttyS2
root@rk3562-buildroot:/# talowe_uart_test
root@rk3562-buildroot:/# echo talowe_uart_test > /dev/ttyS2
root@rk3562-buildroot:/# talowe_uart_test
root@rk3562-buildroot:/# echo talowe_uart_test > /dev/ttyS2
root@rk3562-buildroot:/# talowe_uart_test
root@rk3562-buildroot:/# echo talowe_uart_test > /dev/ttyS2
root@rk3562-buildroot:/# talowe_uart_test
```

此处发送后立即被自己接收并打印出来

测试结束后关闭后台

```
killall cat
```

7.17.2. RS232 (UART5 CON6 端子) 测试

串口工具选择波特率为 115200。

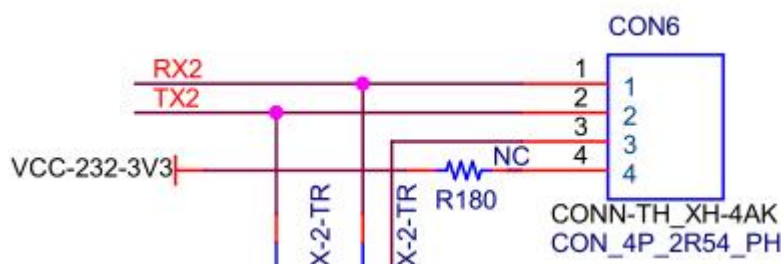
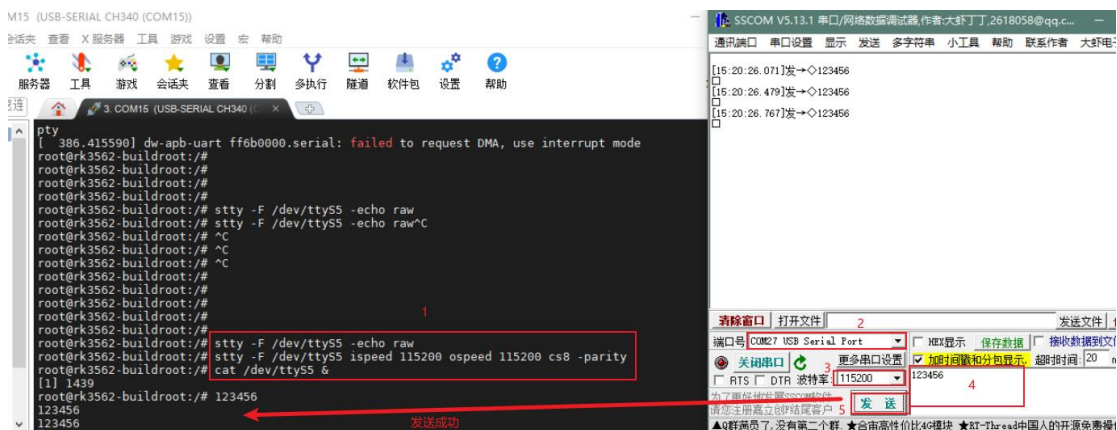


图 62 RS232 的管脚图

测试指令：

```
stty -F /dev/ttyS5 -echo raw #设串口为 raw 模式并关闭回显
stty -F /dev/ttyS5 ispeed 115200 ospeed 115200 cs8 -parity
cat /dev/ttyS5 & #串口 5 后台读取数据
echo talowe_uart_test > /dev/ttyS5 #向串口 5 写入数据
```

设置好串口工具后发送数据，板子能够接收到串口工具所发送的数据：



输入指令板子发送数据，发送后，能够在串口工具中看到板子所发送数据：

```
root@rk3562-buildroot:/# echo talowe_uart_test > /dev/ttyS5
root@rk3562-buildroot:/# echo talowe_uart_test > /dev/ttyS5
root@rk3562-buildroot:/# echo talowe_uart_test > /dev/ttyS5
root@rk3562-buildroot:/# echo talowe_uart_test > /dev/ttyS5
root@rk3562-buildroot:/#
```

[15:21:01.954]收←talowe_uart_test
[15:21:02.338]收←talowe_uart_test
[15:21:02.546]收←talowe_uart_test
[15:21:02.690]收←talowe_uart_test

也可以不使用串口工具，通过短接 RS232 座子的 TX、RX 进行回传测试。

```
root@rk3562-buildroot:/# stty -F /dev/ttyS5 -echo raw
root@rk3562-buildroot:/# stty -F /dev/ttyS5 ispeed 115200 ospeed 115200 cs8 -parity
root@rk3562-buildroot:/# cat /dev/ttyS5 &
[2] 1704
root@rk3562-buildroot:/# echo talowe_uart_test > /dev/ttyS5
talowe_uart_test
root@rk3562-buildroot:/# echo talowe_uart_test > /dev/ttyS5
talowe_uart_test
root@rk3562-buildroot:/# echo talowe_uart_test > /dev/ttyS5
talowe_uart_test
root@rk3562-buildroot:/# echo talowe_uart_test > /dev/ttyS5
talowe_uart_test
```

测试结束后关闭后台

```
killall cat
```

7.17.3. RS485 测试

EVM-RK3562 底板上两路引出的 RS485 串口，其在硬件设计上有些许不同，因此两者测试的过程中会有些许的不同。

(1) RS485_1 测试 (UART3 J18 座子)

把底板的串口 (RS485_A1 以及 RS485_B1) 和 GND 引脚与 RS485 串口模块的相应引脚进行连接 (A-A, B-B, GND-PGND)，配置好串口助手之后运行以下命令进行串口测试。

```
stty -F /dev/ttyS3 -echo raw #设串口为 raw 模式并关闭回显
stty -F /dev/ttyS3 ispeed 115200 ospeed 115200 cs8 -parity
```

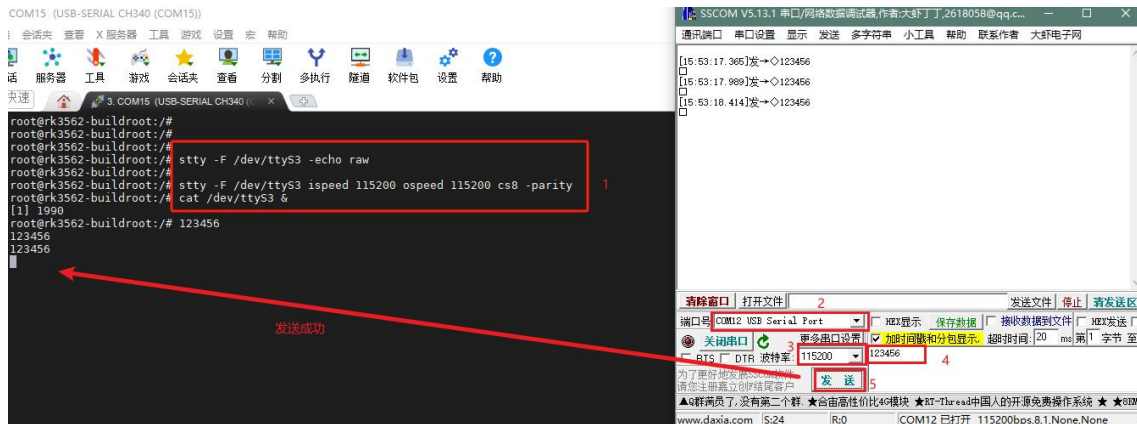
```
cat /dev/ttyS3 &
```

```
echo talowe_uart_test > /dev/ttyS3
```

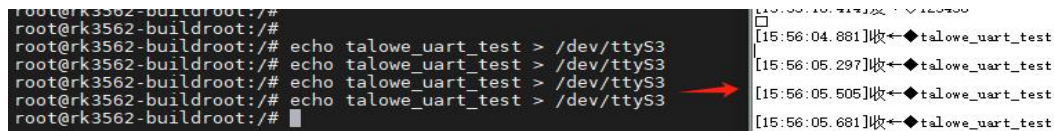
#串口 3 后台读取数据

#向串口 3 写入数据

输入指令后进行 PC 端到开发板的串口通信：



进行开发板到 PC 端的串口通信。

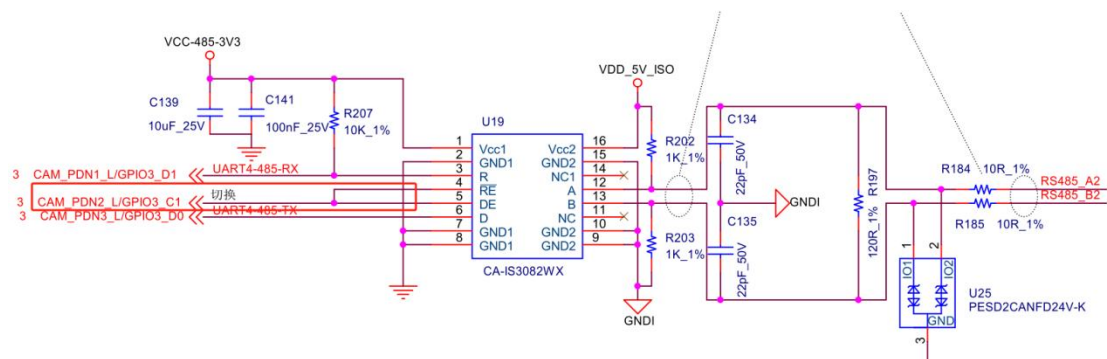


测试结束后关闭后台

killall cat

(2) RS485_2 测试 (UART4 J18 座子)

此处 RS485 在硬件设计上未进行自收发设计，因此在进行 RS485 串口通信的时候，需要通过控制相应的 GPIO 引脚的电平从而去进行收或者发功能的实现。当 GPIO3_C1 为低电平的时候，则只能够进行接收操作，当其处于高电平的时候，则只能够进行发送操作



从原理图中可以看出，通过控制 GPIO3_C1 的电平从而实现数据的收发。

把底板的串口 (RS485_A2 以及 RS485_B2) 和 GND 引脚与 RS485 串口模块的相应引脚进行连接 (A-A, B-B, GND-PGND)，配置好串口助手准备进行串口测试。

通过 USB 转 485/422 连接板子和电脑（A 对 A，B 对 B）。

串口工具选择波特率为：115200

设置好串口工具后发送数据，板子能够接收到串口工具所发送的数据。

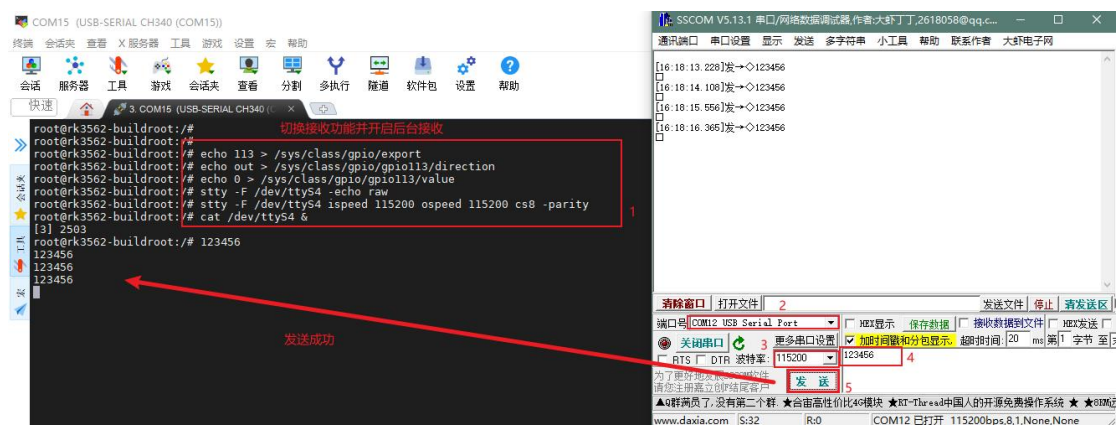
切换 RS485 为接收/发送功能：

```
echo 113 > /sys/class/gpio/export           #导出引脚（只需导出一次）
echo out > /sys/class/gpio/gpio113/direction #设置为输出模式
echo 0 > /sys/class/gpio/gpio113/value       #设置为接收功能
echo 1 > /sys/class/gpio/gpio113/value       #设置为发送功能
```

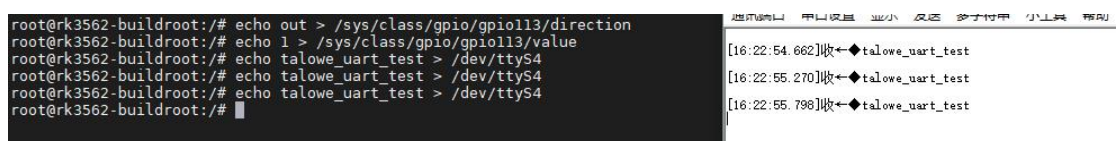
测试指令：

```
stty -F /dev/ttyS4 -echo raw                 #设串口为 raw 模式并关闭回显
stty -F /dev/ttyS4 ispeed 115200 ospeed 115200 cs8 -parity
cat /dev/ttyS4 &                             #串口 4 后台读取数据
echo talowe_uart_test > /dev/ttyS4          #向串口 4 写入数据
```

输入指令后进行 PC 端到开发板的串口通信。



设置为发送模式，并进行开发板到 PC 端的串口通信。



测试结束后关闭后台

```
killall cat
```

7.18. 蜂鸣器测试

通过指令对开发板的 Beep 进行测试，Beep 的 IO 口配置为 pwm 输出，可在命令行下，运行如下命令后，蜂鸣器会鸣叫。

```
echo 0 > /sys/class/pwm/pwmchip2/export       #设置 PWM 为输出
echo 1000000 > /sys/class/pwm/pwmchip2/pwm0/period #设置频率
echo 500000 > /sys/class/pwm/pwmchip2/pwm0/duty_cycle #设置占空比
echo normal > /sys/class/pwm/pwmchip2/pwm0/polarity #设置极性
```

```
echo 1 > /sys/class/pwm/pwmchip2/pwm0/enable #打开蜂鸣器
```

执行以下命令，蜂鸣器会被关闭

```
echo 0 > /sys/class/pwm/pwmchip2/pwm0/enable #关闭蜂鸣器
```

7.19. 烧录口 OTG 测试

7.19.1. 插入 Type-c 转 USB 线测试

```
root@rk3562-buildroot:/# [ 123.669271] dwc3 fe500000.usb: request 00000000cac24019 was not queued to ep
[ 123.671393] android_work: did not send uevent (0 0 0000000000000000)
[ 123.779546] xhci-hcd xhci-hcd.1.auto: xHCI Host Controller
[ 123.779972] xhci-hcd xhci-hcd.1.auto: new USB bus registered, assigned bus number 3
[ 123.780256] xhci-hcd xhci-hcd.1.auto: hcc params 0x0220fe64 hci version 0x110 quirks 0x00000020220100
[ 123.780348] xhci-hcd xhci-hcd.1.auto: irq 95, io mem 0xfe500000
[ 123.780551] xhci-hcd xhci-hcd.1.auto: xHCI Host Controller
[ 123.780878] xhci-hcd xhci-hcd.1.auto: new USB bus registered, assigned bus number 4
[ 123.780914] xhci-hcd xhci-hcd.1.auto: Host supports USB 3.0 SuperSpeed
[ 123.781155] usb usb3: New USB device found, idVendor=1d6b, idProduct=0002, bcdDevice= 5.10
[ 123.781170] usb usb3: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[ 123.781185] usb usb3: Product: xHCI Host Controller
[ 123.781195] usb usb3: Manufacturer: Linux 5.10.198 xhci-hcd
[ 123.781267] usb usb3: SerialNumber: xhci-hcd.1.auto
[ 123.782168] hub 3-0:1.0: USB hub found
[ 123.782244] hub 3-0:1.0: 1 port detected
[ 123.782838] usb usb4: We don't know the algorithms for LPM for this host, disabling LPM.
[ 123.783011] usb usb4: New USB device found, idVendor=1d6b, idProduct=0003, bcdDevice= 5.10
[ 123.783026] usb usb4: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[ 123.783040] usb usb4: Product: xHCI Host Controller
[ 123.783052] usb usb4: Manufacturer: Linux 5.10.198 xhci-hcd
[ 123.783065] usb usb4: SerialNumber: xhci-hcd.1.auto
[ 123.783951] hub 4-0:1.0: USB hub found
[ 123.784039] hub 4-0:1.0: 1 port detected
```

7.19.2. 移除 Type-c 转 USB 线测试

```
[ 78.877845] xhci-hcd xhci-hcd.1.auto: remove, state 4
[ 78.877998] usb usb4: USB disconnect, device number 1
[ 78.879087] xhci-hcd xhci-hcd.1.auto: USB bus 4 deregistered
[ 78.879473] xhci-hcd xhci-hcd.1.auto: remove, state 4
[ 78.879526] usb usb3: USB disconnect, device number 1
[ 78.880972] xhci-hcd xhci-hcd.1.auto: USB bus 3 deregistered
[ 79.121560] android_work: did not send uevent (0 0 0000000000000000)
[ 80.265855] android_work: did not send uevent (0 0 0000000000000000)
```

7.19.3. OTG U 盘识别

插入 U 盘后成功挂载

```
[ 170.808088] usb 3-1: new high-speed USB device number 2 using xhci-hcd
[ 170.953746] usb 3-1: New USB device found, idVendor=0951, idProduct=1666, bcdDevice= 1.10
[ 170.953816] usb 3-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 170.953841] usb 3-1: Product: DataTraveler 3.0
[ 170.953863] usb 3-1: Manufacturer: Kingston
[ 170.953885] usb 3-1: SerialNumber: E0D55E55CCF6E731E9251046
[ 170.956641] usb-storage 3-1:1.0: USB Mass Storage device detected
[ 170.957730] scsi host0: usb-storage 3-1:1.0
[ 171.980740] scsi 0:0:0:0: Direct-Access Kingston DataTraveler 3.0 PMAP PQ: 0 ANSI: 6
[ 171.985078] sd 0:0:0:0: [sda] 60604416 512-byte logical blocks: (31.0 GB/28.9 GiB)
[ 171.985836] sd 0:0:0:0: [sda] Write Protect is off
[ 171.986780] sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
[ 172.109572] sda: sda1
[ 172.114434] sd 0:0:0:0: [sda] Attached SCSI removable disk
[ 172.406886] FAT-fs (sda1): utf8 is not a recommended IO charset for FAT filesystems, filesystem will be ca
[ 172.410684] FAT-fs (sda1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
```

7.20. 摄像头测试

7.20.1. 单路摄像头测试

摄像头接口为底板上的 MIPI_CSI0(J6/J13)，该摄像头接口适配两种摄像头，分别为 GC8034 以及 OV13855 摄像头，将适配的摄像头接入到底板的其中一个

摄像头接口，并烧录固件（rk3562-EVM-LVDS_SingleCAM-v1.0.img）

（1）摄像头识别检测和格式支持查询：

通过查看 rkisp-vir0 下的节点，查找后续用于操作的节点，此处用于操作的节点为/dev/video22。该操作节点根据实际情况来进行更改。（默认为/dev/video22节点）

```
v4l2-ctl --list-devices

rkisp_mainpath (platform:rkisp-vir0):
[ 26.542896] gc8034 4-0037: gc8034_g_mbus_config(2679) enter!
/dev/video22
[ 26.542969] gc8034 4-0037: gc8034_g_mbus_config(2679) enter!
/dev/video23
[ 26.542983] gc8034 4-0037: gc8034_g_mbus_config(2679) enter!
/dev/video24
[ 26.543154] rkCIF-mipi-lvds2: rkCIF_update_sensor_info: stream[0] get remote terminal sensor failed!
/dev/video25
[ 26.543165] stream_cif_mipi_id0: update sensor info failed -19
/dev/video26
[ 26.543211] rkCIF-mipi-lvds2: rkCIF_update_sensor_info: stream[1] get remote terminal sensor failed!
/dev/media2
[ 26.543222] stream_cif_mipi_id1: update sensor info failed -19
```

图 62 GC8034 设备信息

（2）格式支持查询

```
v4l2-ctl --list-formats-ext -d /dev/video22

root@rk3562-buildroot:/# v4l2-ctl --list-formats-ext -d /dev/video22
ioctl: VIDIOC_ENUM_FMT
Type: Video Capture Multiplanar

[0]: 'UYVY' (UYVY 4:2:2)
    Size: Stepwise 32x32 - 3264x2448 with step 8/8
[1]: 'NV16' (Y/CbCr 4:2:2)
    Size: Stepwise 32x32 - 3264x2448 with step 8/8
[2]: 'NV61' (Y/CrCb 4:2:2)
    Size: Stepwise 32x32 - 3264x2448 with step 8/8
[3]: 'NV21' (Y/CrCb 4:2:0)
    Size: Stepwise 32x32 - 3264x2448 with step 8/8
[4]: 'NV12' (Y/CbCr 4:2:0)
    Size: Stepwise 32x32 - 3264x2448 with step 8/8
[5]: 'NM21' (Y/CrCb 4:2:0 (N-C))
    Size: Stepwise 32x32 - 3264x2448 with step 8/8
[6]: 'NM12' (Y/CbCr 4:2:0 (N-C))
    Size: Stepwise 32x32 - 3264x2448 with step 8/8
```

（3）摄像头预览

输入以下命令从而实现摄像头数据预览：

```
gst-launch-1.0 v4l2src device=/dev/video22 ! videoconvert ! video/x-raw,format=NV12,
width=640,height=480 ! autovideosink
```

（4）摄像头拍照

支持对于摄像头数据进行采集，通过以下指令能够进行拍照，并将其数据流保存为jpg文件，存储在根目录下。

```
gst-launch-1.0 v4l2src device=/dev/video22 num-buffers=1 ! video/x-raw, format=NV12,
width=640,height=480 ! mppjpegenc ! filesink location=/pic.jpg
```

```

root@rk3562-buildroot:/#
root@rk3562-buildroot:/# ls -l
total 74
lrwxrwxrwx 1 root root 7 Jul 30 2024 bin -> usr/bin
-rw-r--r-- 1 root root 489 Apr 25 2024 busybox.fragment
lrwxrwxrwx 1 root root 8 Aug 22 2024 data -> userdata
drwxr-xr-x 14 root root 4120 Jan 1 00:00 dev
drwxr-xr-x 25 root root 4096 Jan 1 00:00 etc
drwxr-xr-x 2 root root 4096 Aug 22 2024 info
lrwxrwxrwx 1 root root 7 Jul 30 2024 lib -> usr/lib
lrwxrwxrwx 1 root root 3 Jul 30 2024 lib64 -> lib
lrwxrwxrwx 1 root root 11 Jul 30 2024 linuxrc -> bin/busybox
drwx----- 2 root root 16384 Aug 22 2024 lost+found
drwxr-xr-x 11 root root 4096 Jul 30 2024 media
drwxr-xr-x 5 root root 4096 Aug 22 2024 mnt
drwxr-xr-x 3 root root 1024 Jan 1 00:00 oem
drwxr-xr-x 3 root root 4096 Jul 30 2024 opt
-rw-r--r-- 1 root root 6081 Jan 1 00:18 pic.jpg
dr-xr-xr-x 208 root root 0 Jan 1 00:00 proc
drwxr-xr-x 15 root root 4096 Aug 22 2024 rockchip-test

```

(5) 录制 H264 格式视频

通过以下指令能捕获摄像头数据 100 帧并存储视频文件于根目录下。

```

gst-launch-1.0 v4l2src device=/dev/video22 num-buffers=100 ! video/x-raw,format=NV12,width=640,height=480 ! tee name=t ! queue ! mpph264enc ! queue ! h264parse ! qtmux
x ! filesink location=h264.mp4 t ! queue ! autovideosink

```

```

root@rk3562-buildroot:/# gst-launch-1.0 v4l2src device=/dev/video22 num-buffers=100 ! video/x-raw,format=NV12,width=640,height=480 ! tee name=t ! queue ! mpph264enc ! queue ! h264parse ! qtmux
! filesink location=h264.mp4 t ! queue ! waylandsink
Setting pipeline to PAUSED ...
Using mplane plugin for capture
Pipeline is live and does not need PREROLL ...
Pipeline is PREROLLED ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
[ 2680.403930] rkisp_hw ff3f0000.isp: set isp clk = 594000000Hz
[ 2680.409435] rkCIF-mipi-lyds2: stream[0] start streaming
[ 2680.409549] rockchip-mipi-csi2 mipi2-csi2: stream on, src_sd: 00000000ff837340, sd_name:rockchip-csi2-debug

```

```

root@rk3562-buildroot:/# ls -l
total 510
lrwxrwxrwx 1 root root 7 Jul 30 2024 bin -> usr/bin
-rw-r--r-- 1 root root 489 Apr 25 2024 busybox.fragment
lrwxrwxrwx 1 root root 8 Aug 22 2024 data -> userdata
drwxr-xr-x 14 root root 4120 Jan 1 00:00 dev
drwxr-xr-x 25 root root 4096 Jan 1 00:00 etc
-rw-r--r-- 1 root root 443282 Jan 1 00:30 h264.mp4
drwxr-xr-x 2 root root 4096 Aug 22 2024 info
lrwxrwxrwx 1 root root 7 Jul 30 2024 lib -> usr/lib

```

(6) 播放 H264 格式视频

```

gst-launch-1.0 filesrc location=h264.mp4 ! qtdemux ! queue ! h264parse ! mppvideodec !
autovideosink

```

7.20.2. 双路摄像头测试

若需要同时使用到两路摄像头，在底板上的 MIPI_CSI0(J6/J13)接口，同时插入两个适配的摄像头（GC8034 或 OV13855 摄像头），并烧录固件（rk3562-EVM-LVDS_TwoCAM-v1.0img），便能够同时使用两路摄像头。

(1) 双路摄像头识别检测和格式支持查询：

通过查看 rkisp-vir0、rkisp-vir1 下的节点，查找后续用于操作的节点，此处

用于操作的节点为/dev/video22，以及/dev/video29，该操作节点根据实际情况来进行更改。(默认 J6 接口操作节点为/dev/video22，J13 接口操作节点为/dev/video29)

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

```
[ 57.479660] stream_cif_mipi_id0: open video, entity use_count 3
rkisp_mainpath (platform:rkisp-vir0):
[ 57.479660] stream_cif_mipi_id0: close video, entity use_count 2
/dev/video22
/dev/video23
/dev/video24
/dev/video25
/dev/video26
/dev/media2

rkisp_mainpath (platform:rkisp-vir1):
/dev/video29
/dev/video30
/dev/video31
/dev/video32
/dev/video33
/dev/media3
```

图 64 双路摄像头设备信息

(2) 双路摄像头预览

```
gst-launch-1.0 v4l2src device=/dev/video22 ! videoconvert ! video/x-raw,format=NV12,
width=800,height=800, framerate=30/1 ! waylandsink render-rectangle='<0,0,500,500>' & gst-
launch-1.0 v4l2src device=/dev/video29 ! videoconvert ! video/x-raw,format=NV12,width=8
00,height=800, framerate=30/1 ! waylandsink render-rectangle='<500,0,500,500>' &
```

(3) 结束预览:

预览指令是将指令置于后台持续运行，因此我们若要终止预览需要杀掉该进程，因在预览过程中会显示预览时间，因此此处指令可直接输入并回车运行。

查看正在运行的进程:

```
ps -ef
[ 232.075402] gc8034 4-0037: gc8034_g_mbus_config(2679) enter!
[ 232.075410] gc8034 4-0037: gc8034_g_mbus_config(2679) enter!
[ 232.075420] rockchip-csi2-dphy3: dphy3, data_rate_mbps 1080
[ 232.075435] rockchip-csi2-dphy csi2-dphy0: csi2_dphy_s_stream stream on:1, dphy0,
[ 232.075441] rockchip-csi2-dphy csi2-dphy3: csi2_dphy_s_stream stream on:1, dphy3,
[ 232.075448] ov13855 5-0036: ov13855_s_stream: on: 1, 4224x3136@30
[ 232.075453] gc8034 4-0037: gc8034_s_stream: on: 1, 3264x2448@30
[ 232.143794] rkisp_hw ff3f0000.isp: set isp clk = 594000000Hz
Redistribute latency...
Redistribute latency...
0:00:04.4 / 99:99:99.
```

直接输入 ps -ef 并回车

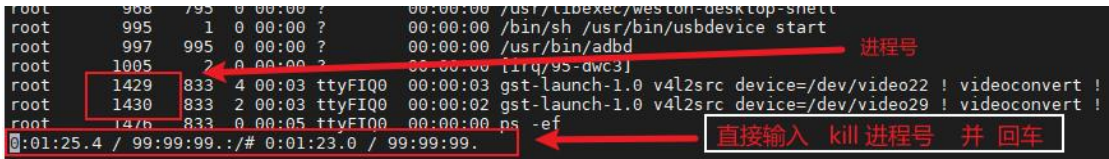
查看正在运行的进程号（以实际为准，此处为 1429 以及 1430）。

```
root 988 795 0 00:00 ? 00:00:00 /usr/libexec/xdm/xsession
root 995 1 0 00:00 ? 00:00:00 /bin/sh /usr/bin/usbdevice start
root 997 995 0 00:00 ? 00:00:00 /usr/bin/adbd
root 1005 2 0 00:00 ? 00:00:00 [irq/95-dwc3]
root 1429 833 4 00:03 ttyFIQ0 00:00:03 gst-launch-1.0 v4l2src device=/dev/video22 ! videoconvert !
root 1430 833 2 00:03 ttyFIQ0 00:00:02 gst-launch-1.0 v4l2src device=/dev/video29 ! videoconvert !
root 1476 833 0 00:05 ttyFIQ0 00:00:00 ps -ef
0:01:25.4 / 99:99:99.:/# 0:01:23.0 / 99:99:99.
```

进程号

结束进程 (kill 进程号)

```
kill 1429
kill 1430
```



```
root 968 793 0 00:00 ? 00:00:00 /usr/libexec/xdm-xdmcp
root 995 1 0 00:00 ? 00:00:00 /bin/sh /usr/bin/usbdevice start
root 997 995 0 00:00 ? 00:00:00 /usr/bin/adbd
root 1005 2 0 00:00 ? 00:00:00 [irq/95-dwc3]
root 1429 833 4 00:03 ttyFIQ0 00:00:03 gst-launch-1.0 v4l2src device=/dev/video22 ! videoconvert !
root 1430 833 2 00:03 ttyFIQ0 00:00:02 gst-launch-1.0 v4l2src device=/dev/video29 ! videoconvert !
root 1476 833 0 00:05 ttyFIQ0 00:00:00 ps -ef
```

0:01:25.4 / 99:99:99.:/# 0:01:23.0 / 99:99:99.

直接输入 kill 进程号 并 回车

摄像头测试结束。

8. 免责声明

本文档提供有关广州眺望电子科技有限公司产品的信息。本文档并未授予任何知识产权的许可，并未以明示或暗示，或以禁止发言或其它方式授予任何知识产权许可。除广州眺望电子科技在其产品的销售条款和条件中声明的责任之外，概不承担任何其它责任。并且，产品的销售和 / 或使用不作任何明示或暗示的担保，包括对产品的特定用途适用性、适销性或对任何专利权、版权或其它知识产权的侵权责任等，均不作担保。广州眺望电子科技产品并非设计用于医疗、救生或维生等用途。广州眺望电子科技可能随时对产品规格及产品描述做出修改，恕不另行通知。

文档所属产品可能包含某些设计缺陷或错误，一经发现将收入勘误表，并因此可能导致产品与已出版的规格有所差异。如客户索取，可提供最新的勘误表。在订购产品之前，请您与我司销售处或分销商联系，以获取最新的规格说明。本文档中提及的含有订购号的文档以及其它文献可通过访问 <https://www.talowe.com/> 获得。

广州眺望电子科技有限公司保留所有权利。