IDO-EVB3568-V2-Ubuntu 系统使用手册

Industio
触 觉 智 能

IDO-EVB3568-V2

Ubuntu系统使用手册

深圳触觉智能科技有限公司

www.industio.cn

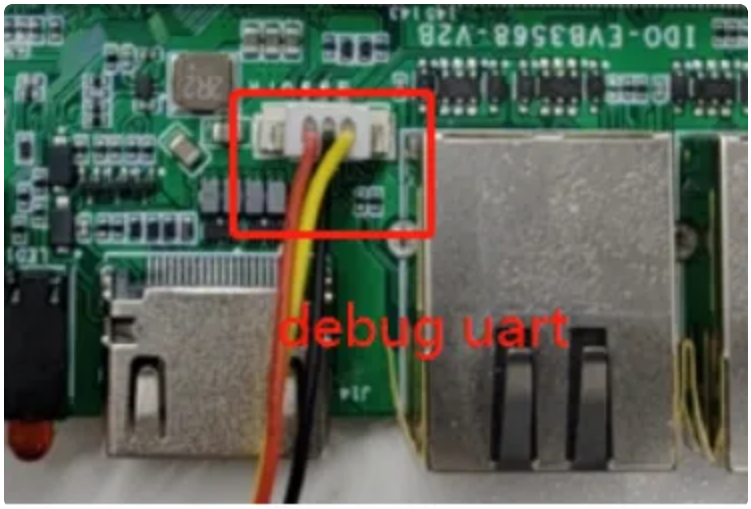
文档修订历史

版本	修订内容	修订	审核	日期
V1.0	创建文档；	谭文学		2023/03/10
v1.1	新增5.10内核音频使用方法	何伟聪		2024/1/11

1、调试

1.1 串口调试

串口调试端口位于J4，通信参数为1500000 8 N 1，电平状态为TTL电平。

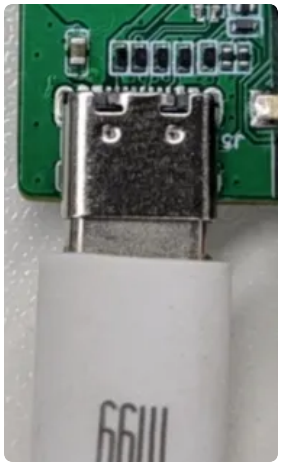


串口调试默认无登录账号密码为：

```
Ubuntu 20.04.3 LTS ido ttyFIQ0

ido login: root (automatic login)

 ___ ____   ___
|_ _|  _ \ / _ \
 | || | | | | | |
 | || |_| | |_| |
|___|____/ \___/

Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 4.19.219 aarch64)


System information as of Mon Feb 21 04:10:17 UTC 2022

System load:   0.48 0.10 0.03   Up time:       0 min
Memory usage:  6 % of 1970MB    IP:
Usage of /:    25% of 14G

Last login: Mon Feb 21 04:10:17 UTC 2022 on ttyFIQ0
root@ido:~#
```

1.2 ADB调试

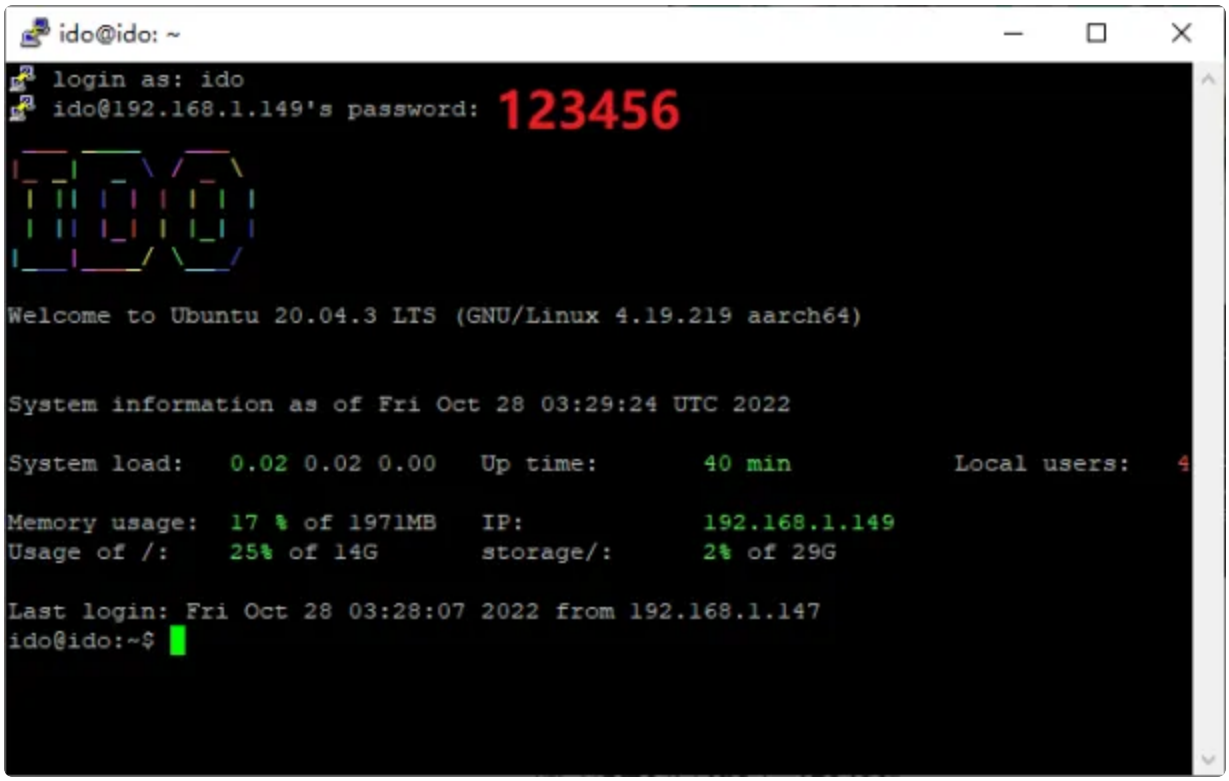
ADB调试端口位于J5，使用TYPE-C线，连接主板的TYPE-C端口和电脑，即可在电脑上使用adb调试。



```
C:\Users\ronnie>adb shell
adb server is out of date.  killing...
* daemon started successfully *
root@ido:/# ls
ls
ahci.ko           dev         lib                 mnt   sbin    tmp
ahci_platform.ko  dmesg       libahci.ko          opt   sdcard  udisk
bin               etc         libahci_platform.ko proc  srv     usr
boot              home        lost+found          root  sys     var
data              insmod.sh   media               run   system  vendor
root@ido:/#
```

1.3 ssh调试

系统默认ssh账号和密码为 **ido @ 123456**。



1.3.1 打开root远程调试

出于安全性考虑，ubuntu系统默认不允许远程root调试。可以通过串口或adb先修改配置文件，打开远程root调试功能。

设置root密码

默认没有root密码，需要先手动设置。

```bash
root@ido:~# passwd root
New password:
Retype new password:
passwd: password updated successfully
root@ido:~#
```

修改ssh配置文件

```bash
root@ido:~# sed -i 's/#PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config
root@ido:~#
```

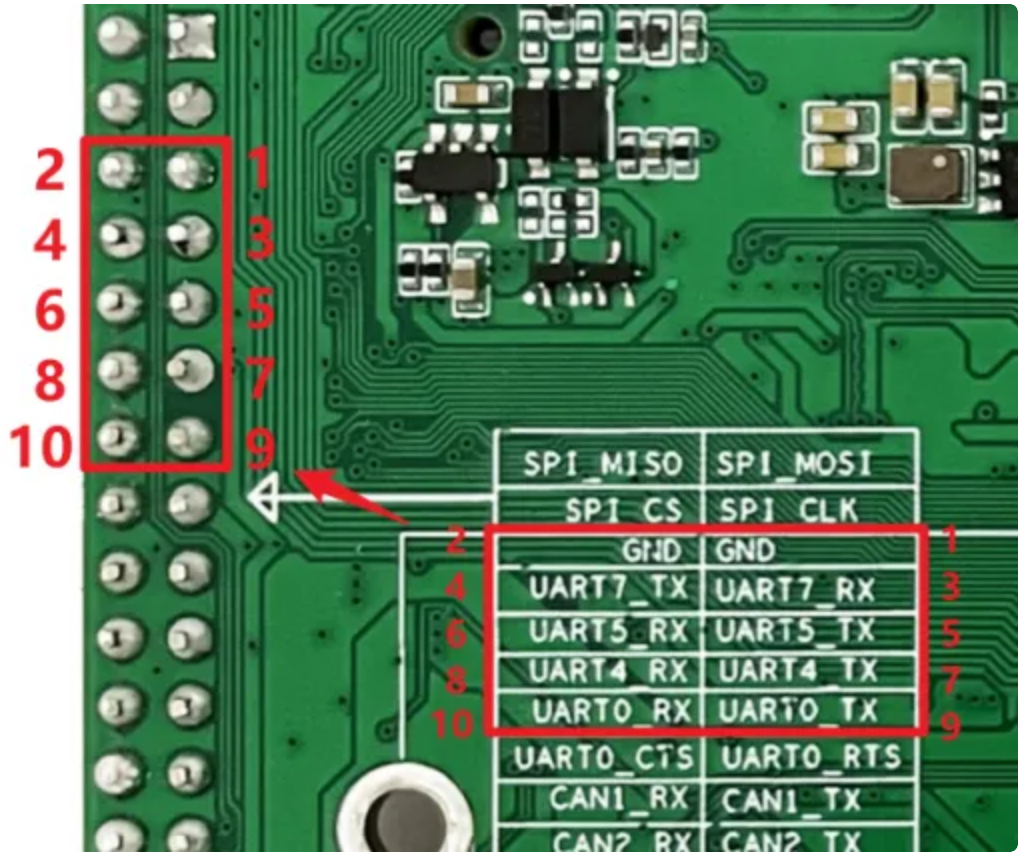
重启ssh服务

```bash
root@ido:~# /etc/init.d/ssh restart
Restarting ssh (via systemctl): ssh.service.
root@ido:~#
```

重启后，即可以远程登录root账户，wincp软件也可以登录root账户，对整个文件系统的文件操作。

2、串口

主板共配置了4路串口（不包括调试串口），其中1路支持流控。



序号	设备节点	默认电平类型	位置	备注
1	/dev/ttyS0	TTL	J24	4线，支持流控
2	/dev/ttyS4	TTL	J24	2线，不支持流控
3	/dev/ttyS5	TTL	J24	2线，不支持流控
4	/dev/ttyS7	TTL	J24	2线，不支持流控

2.1 测试方法

4路使用microcom工具进行简单的收发测试。

需要先安装microcom工具：

Bash

```
root@ido:~# sudo apt-get update
root@ido:~# sudo apt-get install microcom
```

以测试/dev/ttyS0为例：

```Bash
root@ido:~# microcom -s 115200 -p /dev/ttyS0
[  754.636312] of_dma_request_slave_channel: dma-names property of node '/serial@fdd50000' missing or empty
[  754.636443] ttyS0 - failed to request DMA, use interrupt mode
connected to /dev/ttyS0
Escape character: Ctrl-\
Type the escape character to get to the prompt.
```

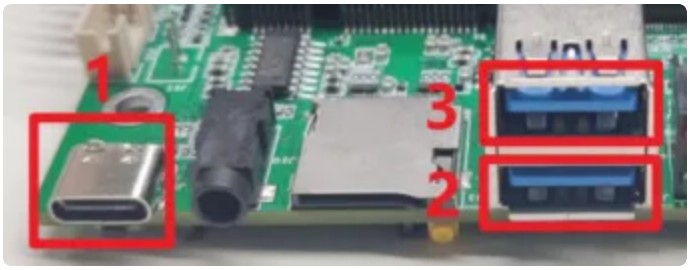
按下键盘任意键会发送对应的字符，而接收的内容会显示在终端。

按【ctrl】和【\】组合键，然后输入quit退出测试。

```Bash
root@ido:~# ls
Desktop
root@ido:~# microcom -s 115200 -p /dev/ttyS0
[  754.636312] of_dma_request_slave_channel: dma-names property of node '/serial@fdd50000' missing or empty
[  754.636443] ttyS0 - failed to request DMA, use interrupt mode
connected to /dev/ttyS0
Escape character: Ctrl-\
Type the escape character to get to the prompt.

Enter command. Try 'help' for a list of builtin commands
-> quit
exiting
```

3、USB

主板共配置了5路USB接口，分别为USB0-5，均为USB-HOST。



编号	名称	类型	位置
1	usb0	USB OTG	J5
2	usb1	host-2.0	J6
3	usb2	host-3.0	J6
4	usb3	host-2.0	J9
5	usb4	host-2.0	J8
6	usb5	host-2.0	J7

USB1默认为device模式，可用于adb调试。如果要切换host模式，执行以下命令：

```Bash
root@ido:~# echo HOST > /dev/otg_mode
```

当要从host切换到device模式，执行以下命令：

```Bash
root@ido:~# echo DEVICE > /dev/otg_mode
```

当USB-HOST插入U盘后，会自动挂载/media/ido/目录下：

```Bash
root@ido:~# ls /media/ido/
KINGSTON
```

3.1 电源控制

默认所有USB-HOST的电源都是开启的，其中USB3-5我们提供了开启/关闭电源的方法。

编写	名称	电源控制节点	位置
4	usb3	/sys/class/leds/usb_fed3_pwr/brightness	J9
5	usb4	/sys/class/leds/usb_fed2_pwr/brightness	J8
6	usb5	/sys/class/leds/usb_fed1_pwr/brightness	J7

打开USB3的电源：

```bash
root@ido:~# echo 255 > /sys/class/leds/usb_fed1_pwr/brightness
```

关闭USB3的电源：

```bash
root@ido:~# echo 0 > /sys/class/leds/usb_fed1_pwr/brightness
```

USB4-5的电源控制方法类似。

4、TF CARD

主板配置了一个TF CARD接口，当TF CARD接口插入TF卡后，会自动挂载到/media/ido/目录下。

```bash
root@ido:~# /media/ido/
3533-3735
root@ido:~#
```

5、以太网

主板配置了2个1000M以太网接口，对应的网络设备节点为eth0和eth1。

5.1 查看以太网IP地址

5.1.1 使用命令查看

系统默认以太网为动态获取IP，当以太网接口插入网线时，会自动获取IP。

```Bash
root@ido:~# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.133  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::3b43:b691:ded5:c497  prefixlen 64  scopeid 0x20<link>
        ether 82:4c:21:62:f5:35  txqueuelen 1000  (Ethernet)
        RX packets 29  bytes 4592 (4.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 43  bytes 4146 (4.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 40

root@ido:~#
```

5.2 设置IP地址

5.2.1 临时设置IP

```Plain Text
ifconfig eth1 192.168.1.123
```

5.2.2 设置静态IP

默认eth0和eth1均为动态获取IP，也可以设置为静态IP。以设置eth1静态IP为192.168.1.10为例。

新建/etc/netplan/00-installer-config.yaml，然后写入如下内容（注意缩进以Tab为单位）：

```Bash
network:
        version: 2
        renderer: NetworkManager
        ethernets:
                eth1:
                        dhcp4: no
                        dhcp6: no
                        addresses: [192.168.1.10/24]
                        gateway4: 192.168.1.1
                        nameservers:
                                addresses: [8.8.8.8, 114.114.114.114]
```

然后重启网络服务，立即生效。

```Bash
root@ido:~# netplan apply
```

此时可以看到eth1的ip变成了192.168.1.10

```Bash
root@ido:~# ifconfig eth1
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.10  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::b458:94ff:fe45:af44  prefixlen 64  scopeid 0x20<link>
        ether b6:58:94:45:af:44  txqueuelen 1000  (Ethernet)
        RX packets 156  bytes 22019 (22.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 266  bytes 27550 (27.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 45

root@ido:~#
```

设备重启后，eth1的静态ip配置依旧有效。

6、WiFi

系统上电默认会打开WiFi，对应的网络设备节点为**wlan0**。

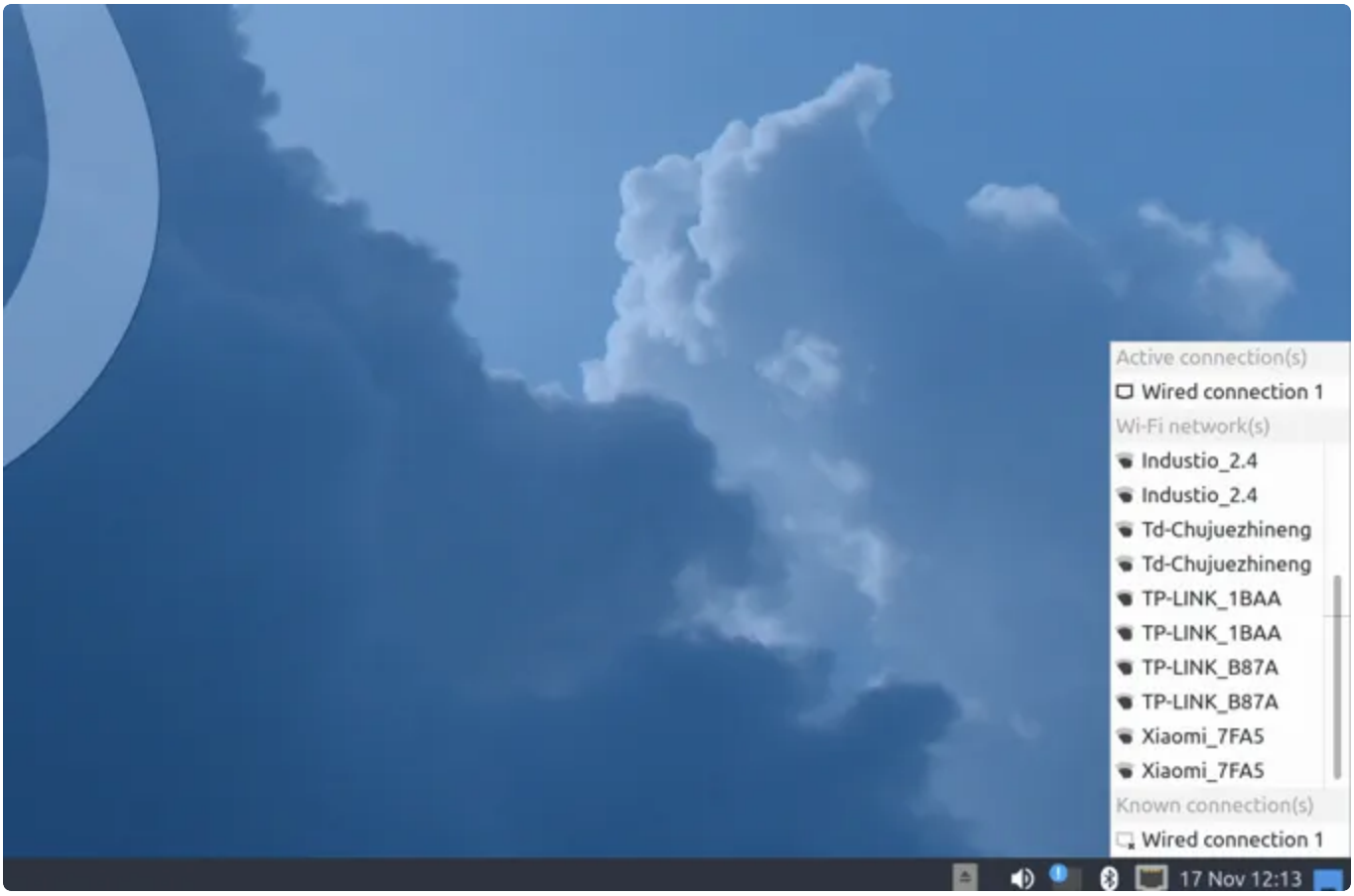
```bash
root@ido:~# ifconfig wlan0
wlan0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 2c:d2:6b:10:ea:4d  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@ido:~#
```

6.1 连接热点

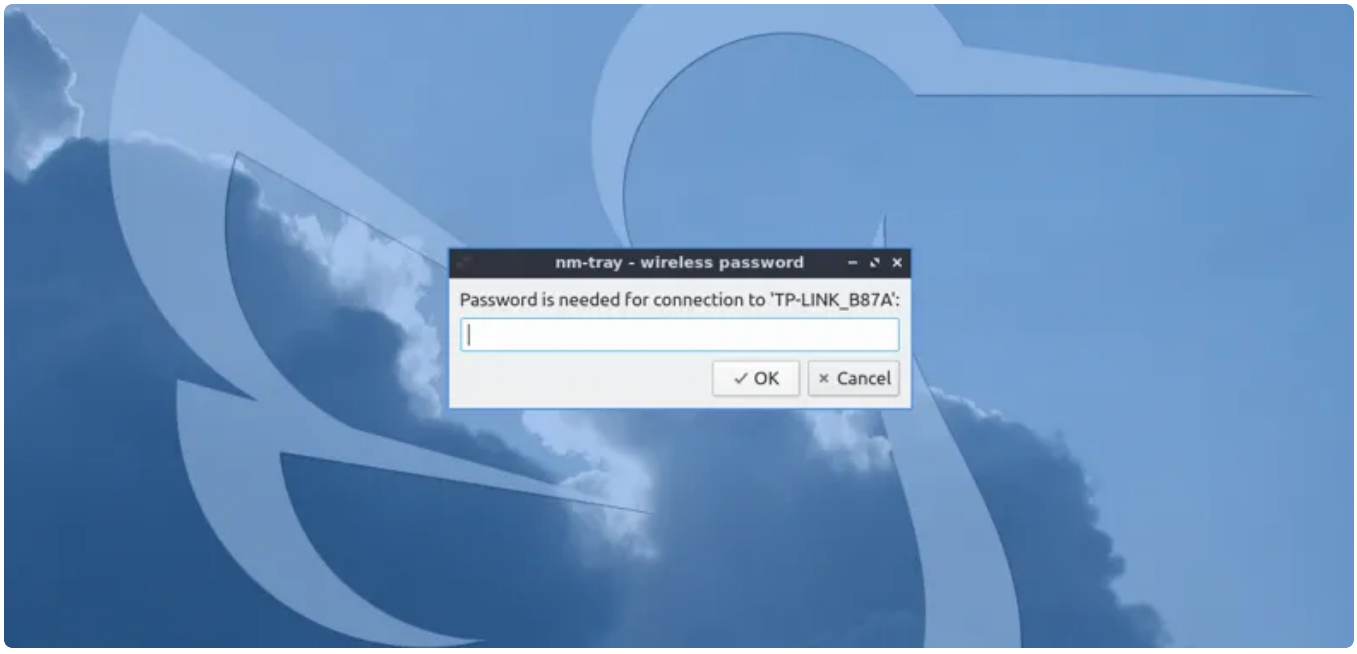
连接热点可以在桌面上操作，也可以使用命令行操作。

6.1.1 方式一

鼠标左键点击桌面右下角的网络图标，即可看到WiFi热点列表：

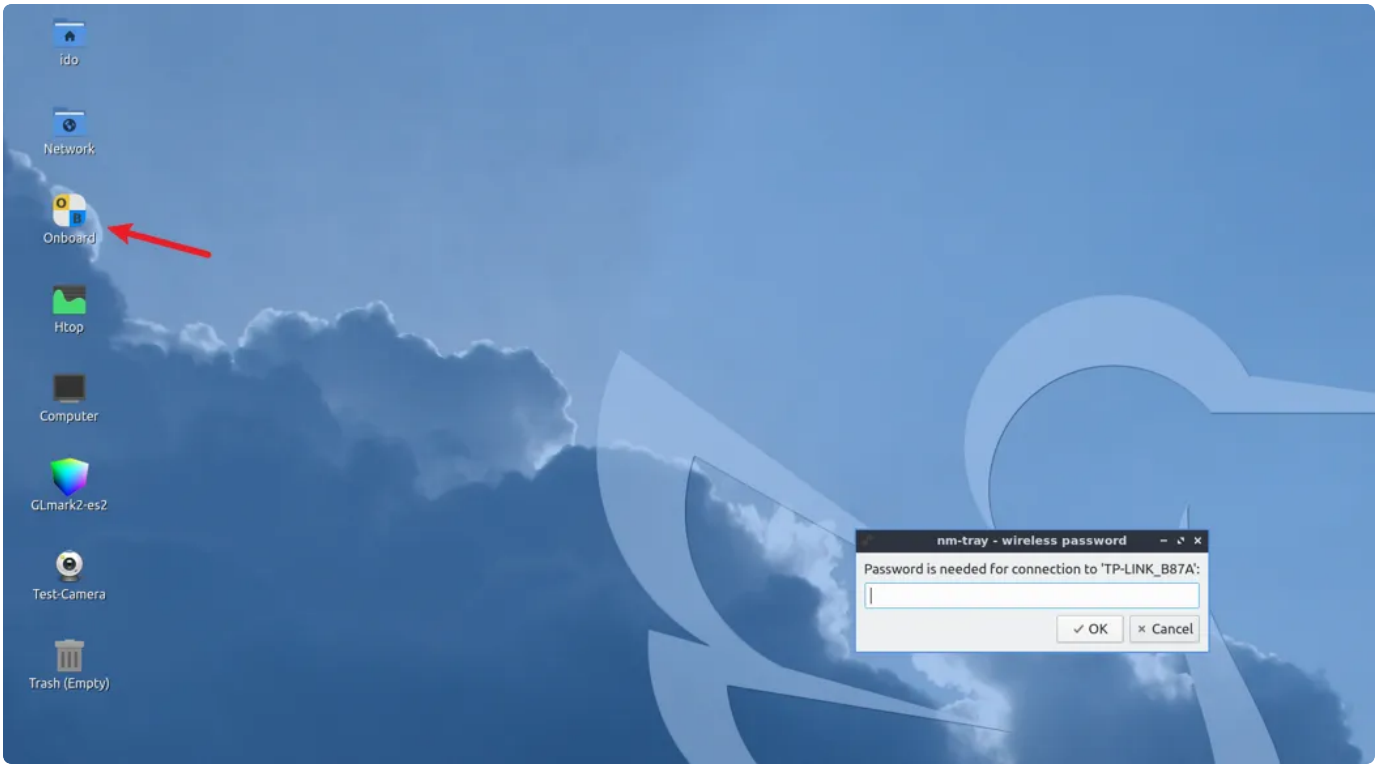


点击要连接的热点，弹出密码输入窗口：

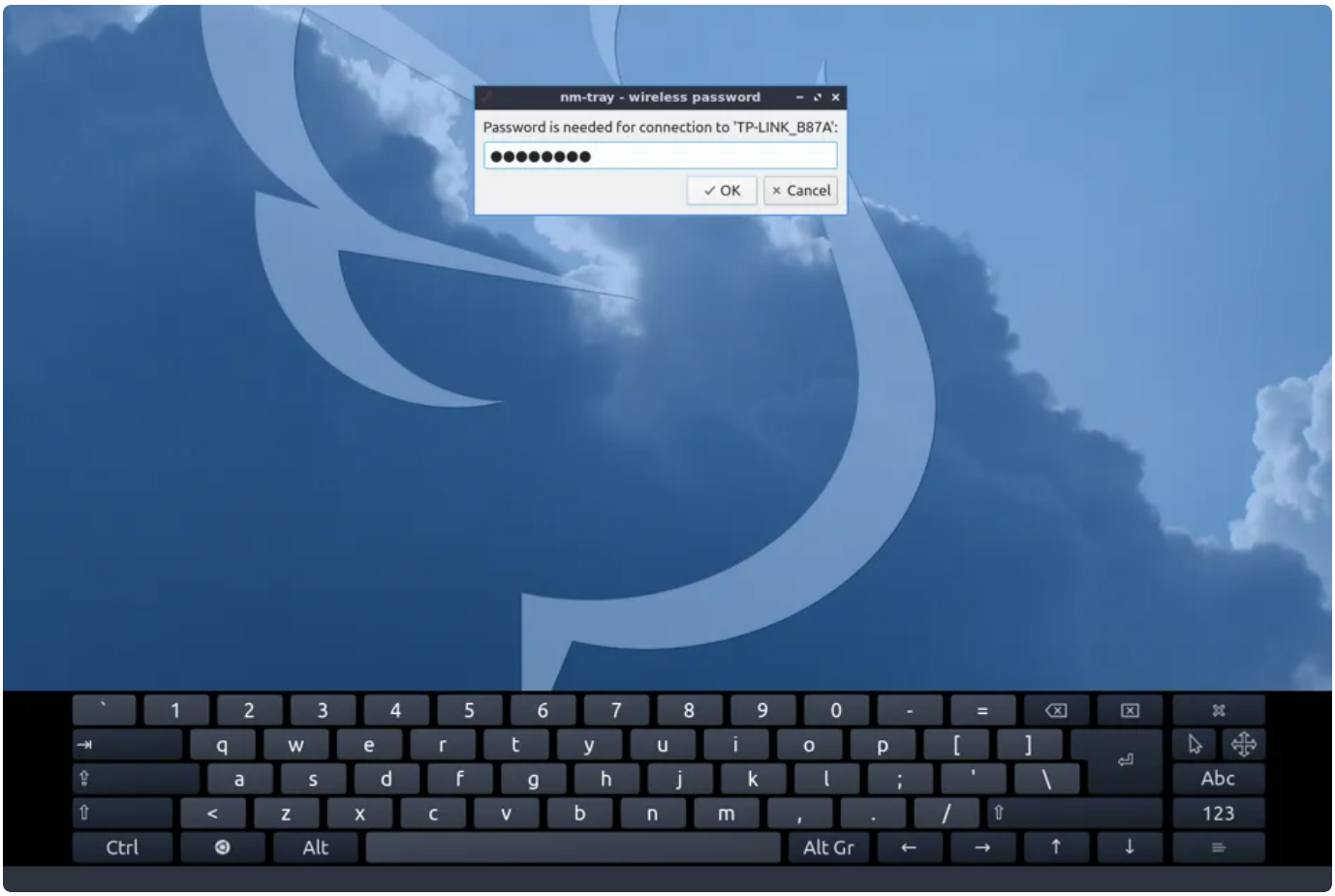


如果有连接键盘，直接输入密码即可；如果没有连接键盘，可以使用系统自带的软键盘。

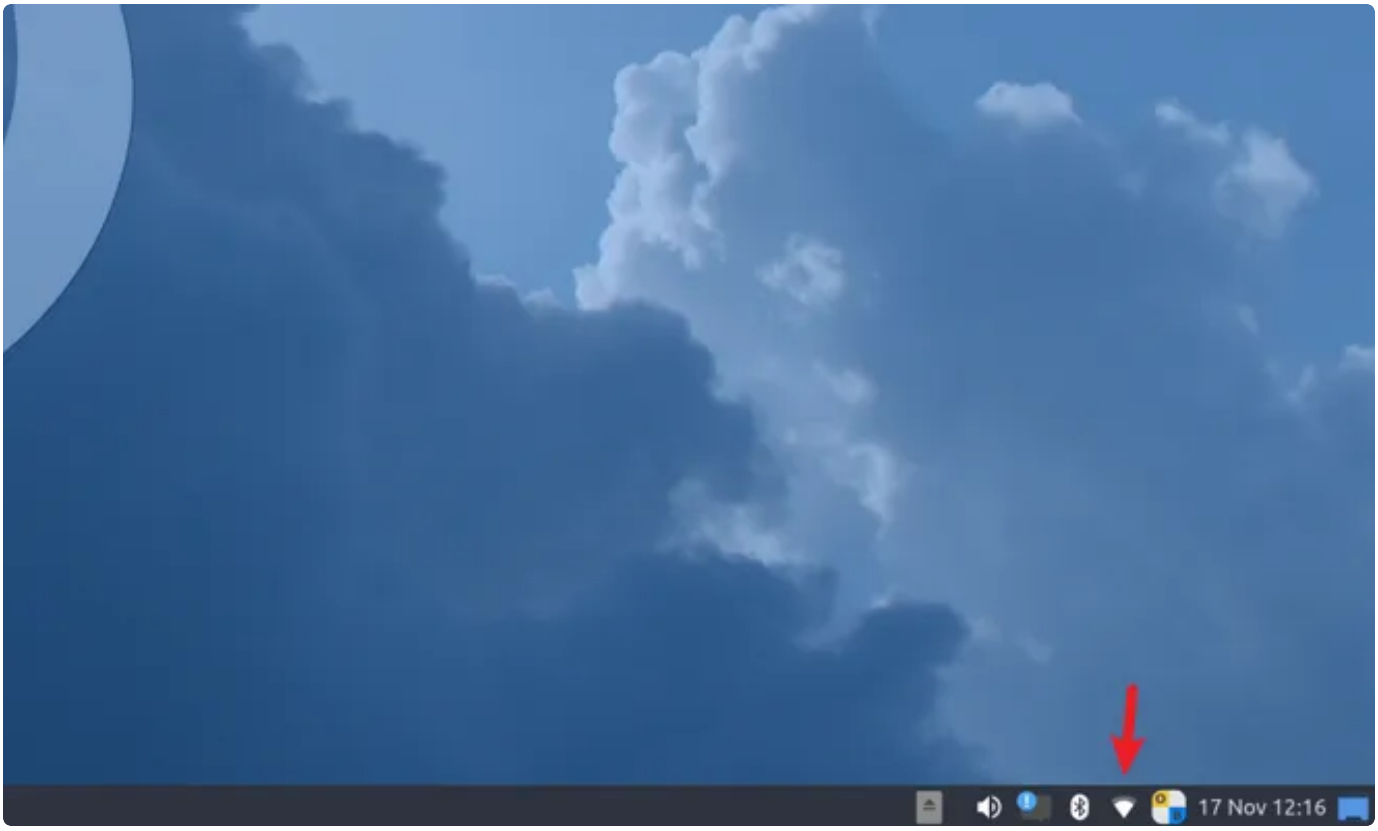
点击桌面的Onboard图标，即可打开系统自带的软键盘



使用软键盘输入密码后，点击【OK】连接热点



连接成功后，桌面右下角的网络图标将改变:



6.1.2 方式二

新建/etc/netplan/01-wifi-sta.yaml，并按照以下格式填写内容：

```Bash
network:
    wifis:
        wlan0:
            dhcp4: true
            access-points:
                "TP-LINK_B87A":
                    password: "12345678"
    version: 2
```

其中TP-LINK_B87A为要连接的热点名称，12345678为连接密码。

修改成功后执行以下命令进行连接：

```Bash
root@ido:~# killall wpa_supplicant
root@ido:~# netplan apply
```

等待几秒钟后，将成功连接WiFi热点。

6.2 查看WiFi的IP地址

使用ifconfig命令可查看连接热点后获取的IP

```Bash
root@ido:~# ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.165  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::984a:9a2f:77b4:e899  prefixlen 64  scopeid 0x20<link>
        ether 2c:d2:6b:10:ea:4d  txqueuelen 1000  (Ethernet)
        RX packets 83  bytes 10479 (10.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 35  bytes 5285 (5.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@ido:~#

```

测试wifi的网络

```Bash
root@ido:~# ping www.baidu.com -I wlan0
PING www.a.shifen.com (14.215.177.39) from 192.168.1.117 p2p0: 56(84) byte
s of data.
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=1 ttl=54 time=17.8 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=2 ttl=54 time=9.30 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=3 ttl=54 time=25.7 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=4 ttl=54 time=42.1 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=5 ttl=54 time=13.1 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=6 ttl=54 time=39.8 ms
64 bytes from 14.215.177.39 (14.215.177.39): icmp_seq=7 ttl=54 time=28.1 ms
```

7.1 连接蓝牙设备

系统开机默认打开蓝牙，对应的网络节点为hci0。

```Bash
root@ido:~# hciconfig
hci0:   Type: Primary  Bus: UART
        BD Address: 70:D5:2B:5B:22:22  ACL MTU: 1021:8  SCO MTU: 255:12
        UP RUNNING
        RX bytes:1665 acl:0 sco:0 events:57 errors:0
        TX bytes:6311 acl:0 sco:0 commands:57 errors:0

root@ido:~#
```

7.1 扫描设备

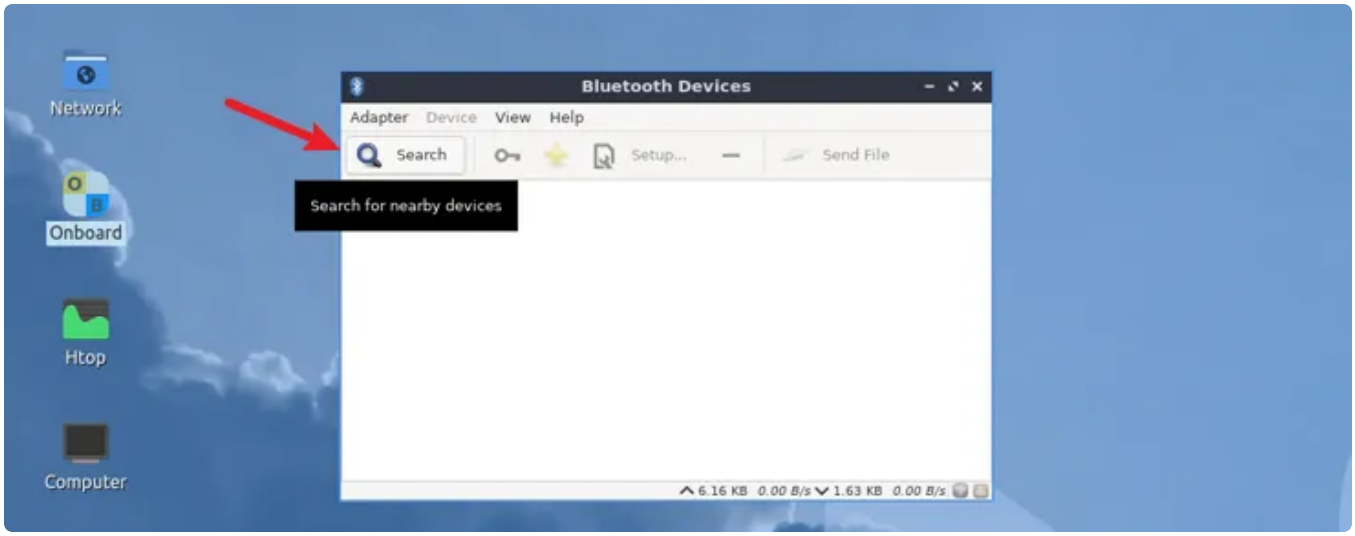
使用hcitool scan或hcitool lescan对附近的设备进行扫描：

```Bash
root@ido:~# hcitool scan
Scanning ...
        94:87:E0:9D:14:12       seeyou
root@ido:~# hcitool lescan
LE Scan ...
48:AA:99:9F:BE:D8 (unknown)
24:CF:91:F8:E8:1B (unknown)
C2:E0:B3:FE:8A:E6 (unknown)
50:94:0D:43:77:86 (unknown)
72:6B:43:77:F4:27 (unknown)
5E:C3:95:C1:D2:F1 (unknown)
1A:1B:A6:E9:55:90 (unknown)
```

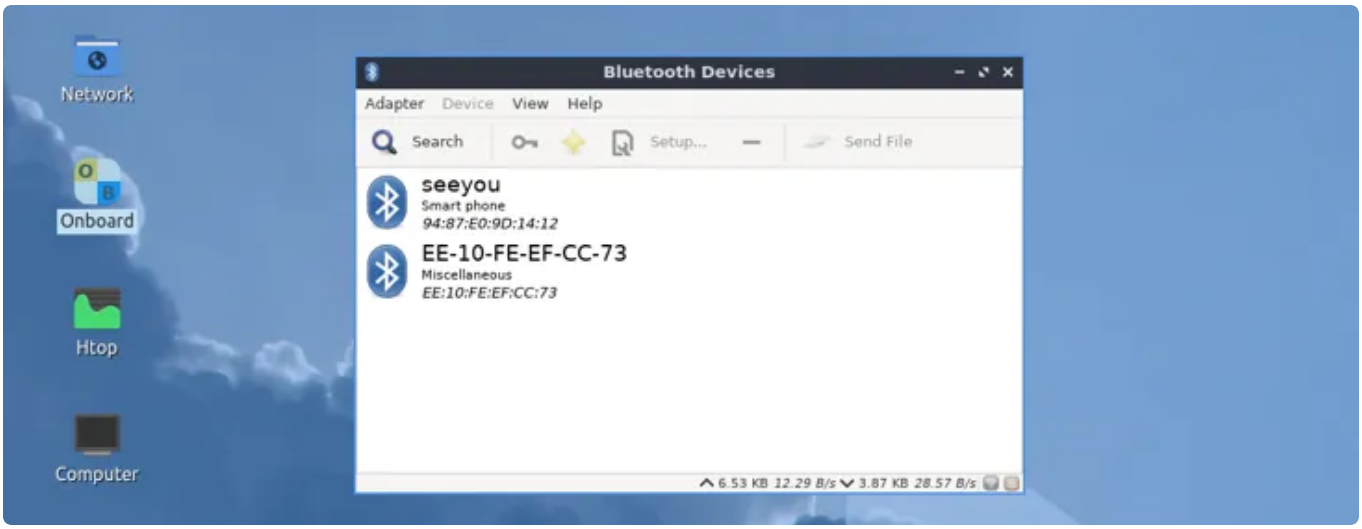
7.2 连接蓝牙设备

将鼠标放到桌面右下角蓝牙图标，右键->Devices

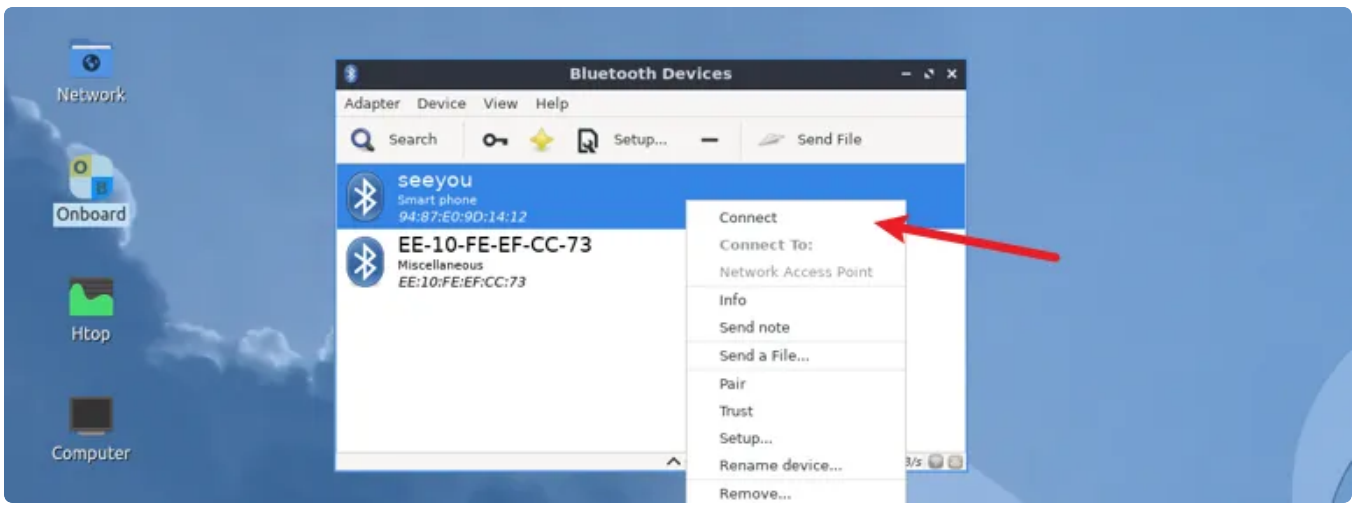
在弹出的窗口中，点击【Search】：



将看到附近的蓝牙设备列表：



选中要连接设备，右键–>Connect，即可连接该设备：



8、4G

主板默认适配EC20模块（4G）和 RG200U-CN（5G），上电前，正确按照模块和SIM卡，上电后，系统会自动进行拨号上网。

序号	模块名称	说明
1	EC20	4G LTE
2	RG200U-CN	支持 5G NSA 和 SA 模式，支持 TDD 和 FDD 两种模式，向下兼容 4G/3G。

拨号成功会产生wwan0网络节点：

```bash
root@ido:~# ifconfig wwan0
wwan0: flags=193<UP,RUNNING,NOARP>  mtu 1500
        inet 10.216.81.159  netmask 255.255.255.192
        inet6 fe80::2ce5:13ff:fe65:7dd  prefixlen 64  scopeid 0x20<link>
        ether 2e:e5:13:65:07:dd  txqueuelen 1000  (Ethernet)
        RX packets 26  bytes 4777 (4.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 118  bytes 8278 (8.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@ido:~#
```

使用ping命令测试4G/5G上网功能是否正常：

```Bash
root@ido:~# ping www.baidu.com -I wwan0
PING www.a.shifen.com (14.215.177.38) from 10.216.81.159 wwan0: 56(84) bytes of data.
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=1 ttl=49 time=52.2 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=2 ttl=49 time=52.7 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=3 ttl=49 time=54.8 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=4 ttl=49 time=63.2 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=5 ttl=49 time=50.3 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=6 ttl=49 time=50.0 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=7 ttl=49 time=68.3 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=8 ttl=49 time=61.2 ms
64 bytes from 14.215.177.38 (14.215.177.38): icmp_seq=9 ttl=49 time=65.1 ms
```

9、音频

9.1 查看声卡设备

```Bash
root@ido:~# aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: rockchiphdmi [rockchip,hdmi], device 0: rockchip,hdmi i2s-hifi-0 [rockchip,hdmi i2s-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: rockchiprk809co [rockchip,rk809-codec], device 0: fe410000.i2s-rk817-hifi rk817-hifi-0 [fe410000.i2s-rk817-hifi rk817-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
root@ido:~#
```

9.2 播放音频

播放到HDMI：

```bash
aplay -D plughw:0,0 /usr/share/sounds/alsa/Rear_Center.wav
```

播放到Lineout：

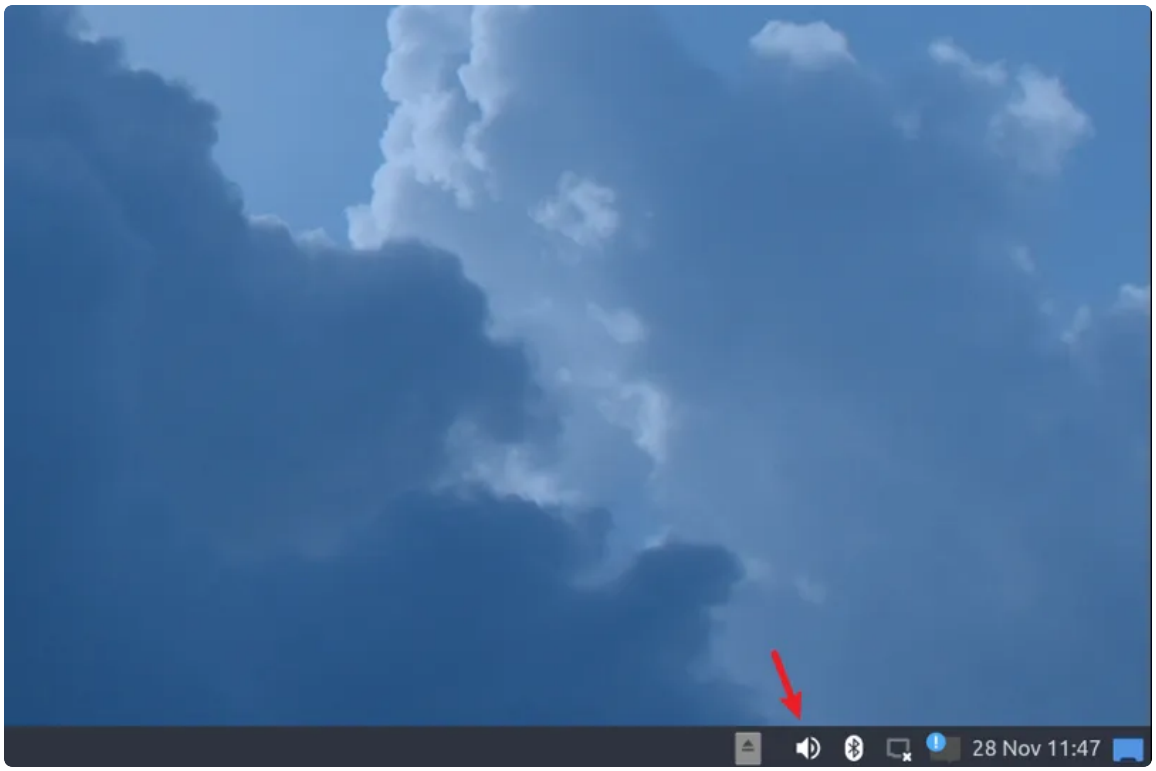
```bash
aplay -D plughw:1,0 /usr/share/sounds/alsa/Rear_Center.wav
```

播放到耳机（需要插入耳机）：

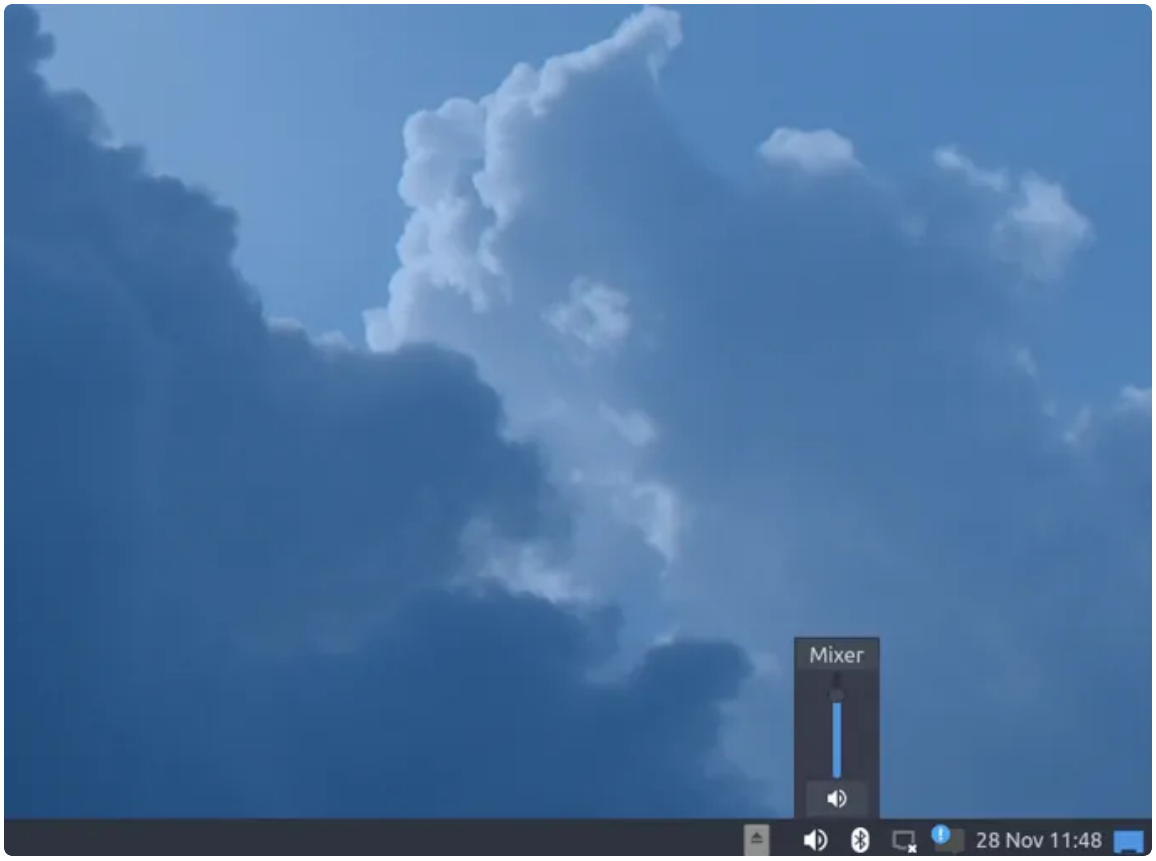
```bash
aplay -D plughw:1,0 /usr/share/sounds/alsa/Rear_Center.wav
```

9.3 音量的调节

鼠标点击桌面右下角的音量图标：



然后滑动鼠标进行音量调节：



当需要静音是，点击静音按钮：

9.4 录音

将麦克风连接到J121。

使用arecord工具可以进行录音测试：

```Bash
root@ido:~# arecord -D hw:1,0 -r 48000 -c 2 -f S16_LE test.wav
Recording WAVE 'test.wav' : Signed 16 bit Little Endian, Rate 48000 Hz, Stereo
^CAborted by signal Interrupt...
root@ido:~#
```

录音完后播放测试：

```Bash
root@ido:~# aplay -D plughw:1,0 ./test.wav
Playing WAVE './test.wav' : Signed 16 bit Little Endian, Rate 48000 Hz, Stereo
root@ido:~#
```

9.5（5.10内核）音频

播放到hdmi

```Bash
aplay -D plug:spk_c0 /usr/share/sounds/alsa/Rear_Center.wav
```

播放到喇叭/耳机

```Bash
aplay -D plug:spk_c1 /usr/share/sounds/alsa/Rear_Center.wav
```

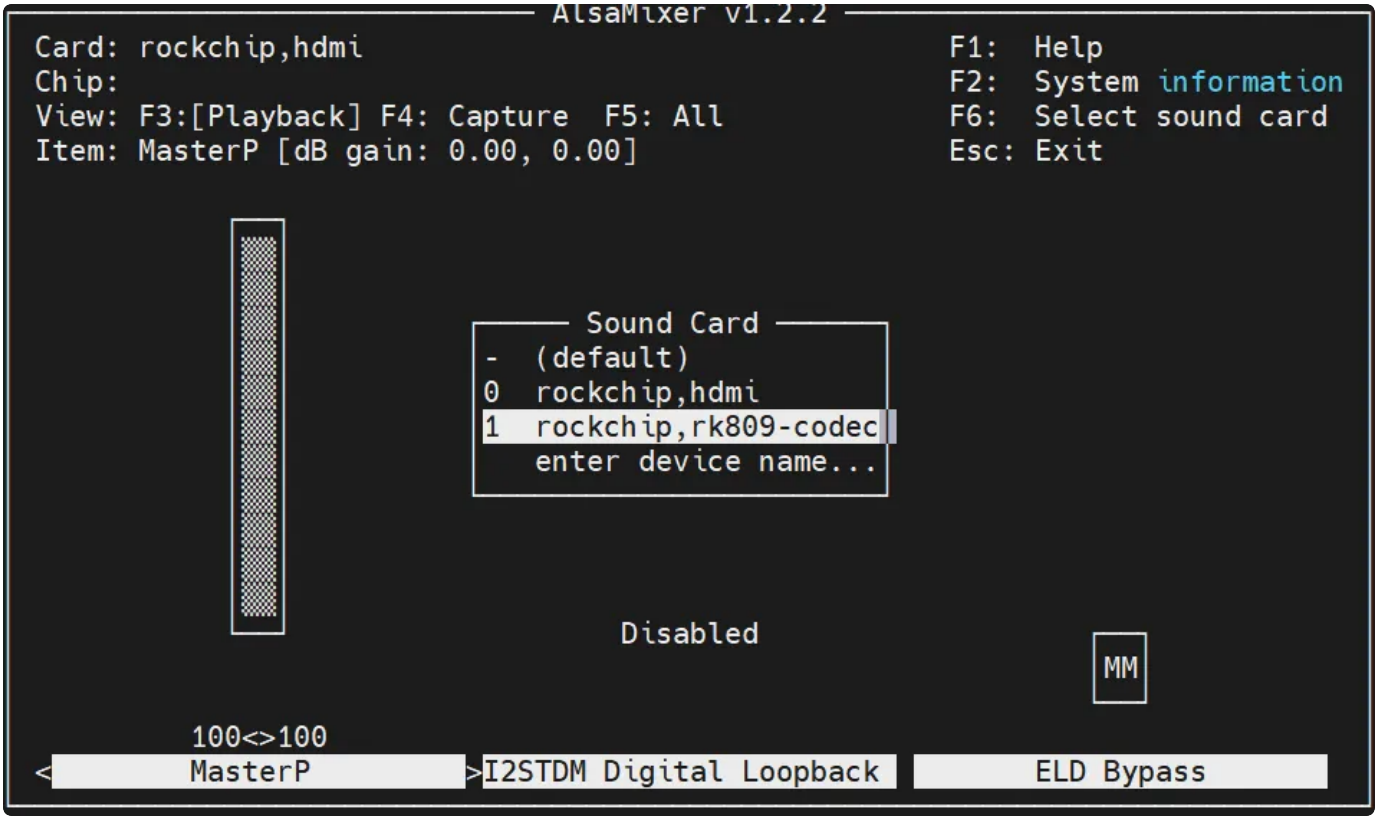
录音

```Bash
arecord -D hw:1,0 -r 48000 -c 2 -f S16_LE test.wav
```

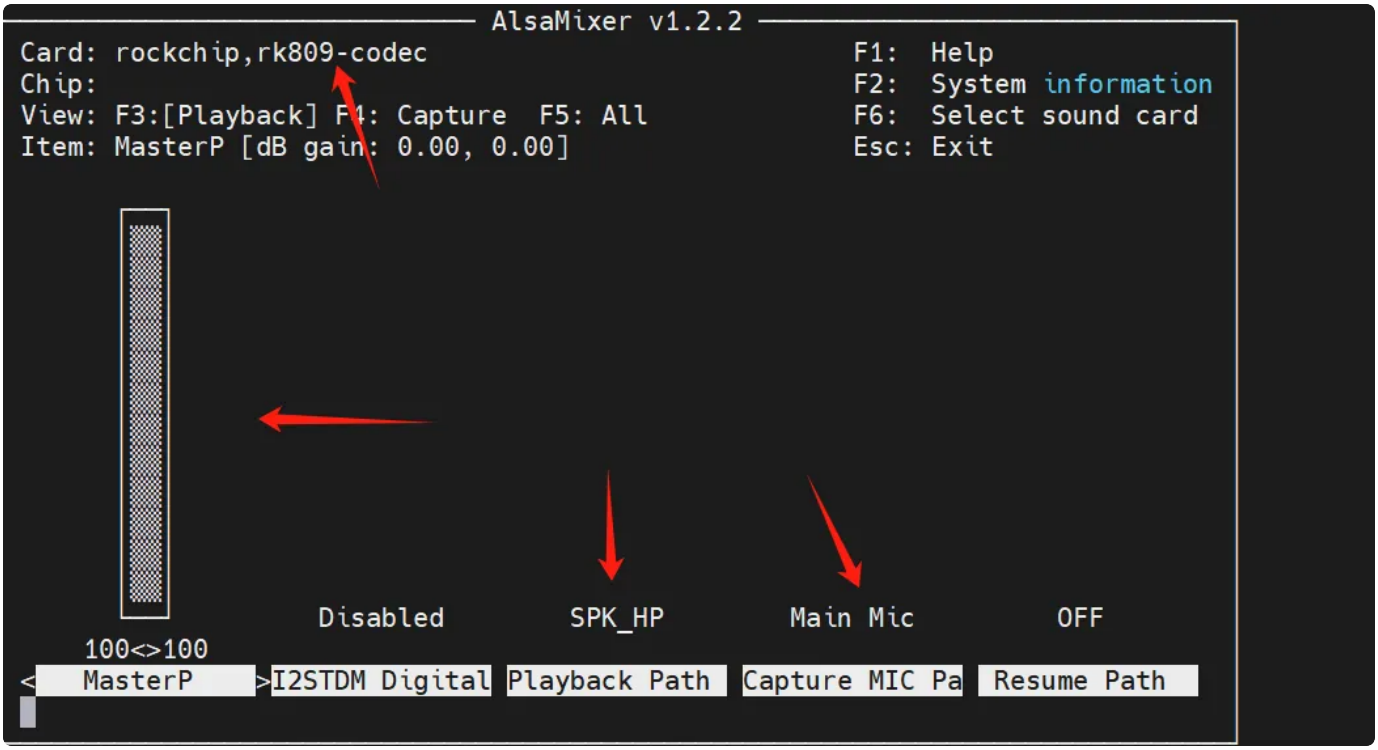
音量调节

```bash
alsamixer
```

进入菜单界面，然后按s，进行选择声卡，这里用声卡1，也就是喇叭播放的声卡

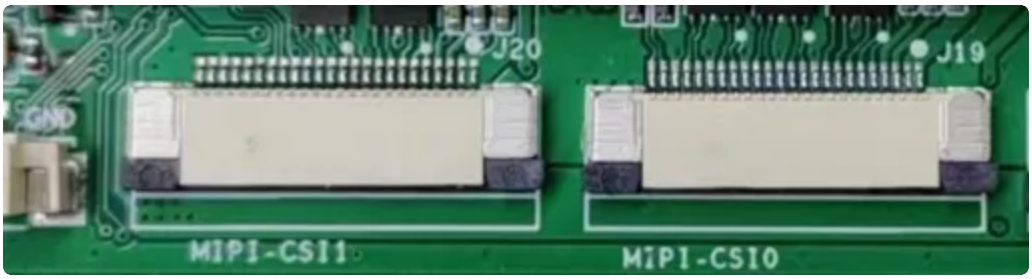


然后就可以进行音量调节了，注意，这里的Playback Path Capture MIC Pa ，有时候播放喇叭没声音，需要设置这里的参数



10、摄像头

主板默认适配OV5648+OV8858摄像头。



10.1 测试

10.1.1 测试摄像头是否存在

```
root@ido:~# media-ctl -p -d /dev/media0
...
- entity 67: rockchip-csi2-dphy0 (2 pads, 2 links)
             type V4L2 subdev subtype Unknown flags 0
             device node name /dev/v4l-subdev2
        pad0: Sink
                [fmt:SBGGR10_1X10/2592x1944@10000/150000 field:none]
                <- "m00_b_ov5648 2-0036":0 [ENABLED]
        pad1: Source
                [fmt:SBGGR10_1X10/2592x1944@10000/150000 field:none]
                -> "rkisp-csi-subdev":0 [ENABLED]

- entity 70: m00_b_ov5648 2-0036 (1 pad, 1 link)
             type V4L2 subdev subtype Sensor flags 0
             device node name /dev/v4l-subdev3
        pad0: Source
                [fmt:SBGGR10_1X10/2592x1944@10000/150000 field:none]
                -> "rockchip-csi2-dphy0":0 [ENABLED]
root@ido:~#
```

10.1.3抓取视频

使用v4l2-ctl工具可以抓取摄像头的视频数据流。

```
root@ido:~# v4l2-ctl --verbose -d /dev/video0 --set-fmt-video=width=1920,height=1080,pixelformat='NV12' --stream-mmap=4 --set-selection=target=crop,flags=0,top=0,left=0,width=1920,height=1080 --stream-to=./out.yuv
VIDIOC_QUERYCAP: ok
VIDIOC_G_FMT: ok
VIDIOC_S_FMT: ok
Format Video Capture Multiplanar:
        Width/Height      : 1920/1080
        Pixel Format      : 'NV12' (Y/CbCr 4:2:0)
        Field             : None
        Number of planes  : 1
        Flags             :
        Colorspace        : Default
        Transfer Function : Default
        YCbCr/HSV Encoding: Default
        Quantization      : Full Range
        Plane 0           :
           Bytes per Line : 1920
           Size Image     : 3110400
VIDIOC_G_SELECTION: ok
VIDIOC_S_SELECTION: ok
                VIDIOC_REQBUFS returned 0 (Success)
                VIDIOC_QUERYBUF returned 0 (Success)
                VIDIOC_QUERYBUF returned 0 (Success)
                VIDIOC_QUERYBUF returned 0 (Success)
                VIDIOC_QUERYBUF returned 0 (Success)
                VIDIOC_QBUF returned 0 (Success)
                VIDIOC_QBUF returned 0 (Success)
                VIDIOC_QBUF returned 0 (Success)
                VIDIOC_QBUF returned 0 (Success)
                VIDIOC_STREAMON returned 0 (Success)
cap dqbuf: 0 seq:      1 bytesused: 3110400 ts: 1384.549991 (ts-monotonic, ts-src-eof)
cap dqbuf: 1 seq:      2 bytesused: 3110400 ts: 1384.616490 delta: 66.499 ms (ts-monotonic, ts-src-eof)
cap dqbuf: 2 seq:      3 bytesused: 3110400 ts: 1384.682975 delta: 66.485 ms (ts-monotonic, ts-src-eof)
cap dqbuf: 3 seq:      4 bytesused: 3110400 ts: 1384.749486 delta: 66.511 ms (ts-monotonic, ts-src-eof)
cap dqbuf: 0 seq:      5 bytesused: 3110400 ts: 1384.816022 delta: 66.536 ms fps: 15.04 (ts-monotonic, ts-src-eof)
cap dqbuf: 1 seq:      6 bytesused: 3110400 ts: 1384.882509 delta: 66.487 ms fps: 15.04 (ts-monotonic, ts-src-eof)
cap dqbuf: 2 seq:      7 bytesused: 3110400 ts: 1384.949025 delta: 66.516 ms fps: 15.04 (ts-monotonic, ts-src-eof)
```

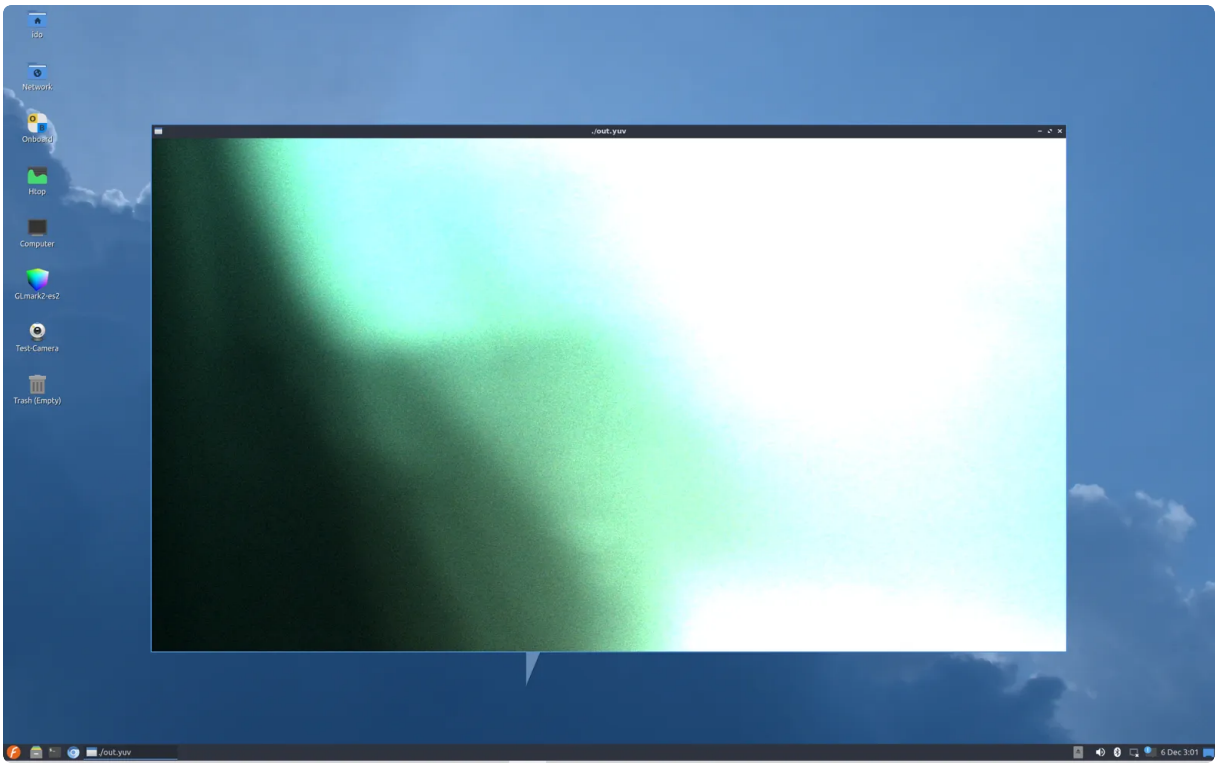
```
cap dqbuf: 3 seq:      8 bytesused: 3110400 ts: 1385.015545 delta: 66.520 ms fps: 15.04 (ts-monotonic, ts-src-eof)
cap dqbuf: 0 seq:      9 bytesused: 3110400 ts: 1385.082051 delta: 66.506 ms fps: 15.04 (ts-monotonic, ts-src-eof)
cap dqbuf: 1 seq:     10 bytesused: 3110400 ts: 1385.148567 delta: 66.516 ms fps: 15.04 (ts-monotonic, ts-src-eof)
cap dqbuf: 2 seq:     11 bytesused: 3110400 ts: 1385.215079 delta: 66.512 ms fps: 15.04 (ts-monotonic, ts-src-eof)
cap dqbuf: 3 seq:     12 bytesused: 3110400 ts: 1385.281594 delta: 66.515 ms fps: 15.04 (ts-monotonic, ts-src-eof)
cap dqbuf: 0 seq:     13 bytesused: 3110400 ts: 1385.348115 delta: 66.521 ms fps: 15.04 (ts-monotonic, ts-src-eof)
cap dqbuf: 1 seq:     14 bytesused: 3110400 ts: 1385.414669 delta: 66.554 ms fps: 15.03 (ts-monotonic, ts-src-eof)
cap dqbuf: 2 seq:     15 bytesused: 3110400 ts: 1385.481133 delta: 66.464 ms fps: 15.04 (ts-monotonic, ts-src-eof)
cap dqbuf: 3 seq:     16 bytesused: 3110400 ts: 1385.547656 delta: 66.523 ms fps: 15.04 (ts-monotonic, ts-src-eof)
cap dqbuf: 0 seq:     17 bytesused: 3110400 ts: 1385.614172 delta: 66.516 ms fps: 15.04 (ts-monotonic, ts-src-eof)
cap dqbuf: 1 seq:     18 bytesused: 3110400 ts: 1385.680680 delta: 66.508 ms fps: 15.04 (ts-monotonic, ts-src-eof)
cap dqbuf: 2 seq:     19 bytesused: 3110400 ts: 1385.747241 delta: 66.561 ms fps: 15.03 (ts-monotonic, ts-src-eof)
cap dqbuf: 3 seq:     20 bytesused: 3110400 ts: 1385.813714 delta: 66.473 ms fps: 15.03 (ts-monotonic, ts-src-eof)
^C
```

按Ctrl-C停止抓取，视频流保存到文件out.yuv。

使用ffplay工具播放抓取的视频流：

```
root@ido:~# ffplay -f rawvideo -video_size 1920x1080 -pix_fmt nv12 ./out.yuv
ffplay version 4.2.4-1ubuntu1.0firefly1 Copyright (c) 2003-2020 the FFmpeg developers
  built with gcc 9 (Ubuntu 9.3.0-17ubuntu1~20.04)
  configuration: --prefix=/usr --extra-version=1ubuntu1.0firefly1 --toolchain=hardened --libdir=/usr/lib/aarch64-linux-gnu --incdir=/usr/include/aarch64-linux-gnu --arch=arm64 --enable-gpl --disable-stripping --enable-avresample --disable-filter=resample --enable-avisynth --enable-gnutls --enable-ladspa --enable-libaom --enable-libass --enable-libbluray --enable-libbs2b --enable-libcaca --enable-libcdio --enable-libcodec2 --enable-libflite --enable-libfontconfig --enable-libfreetype --enable-libfribidi --enable-libgme --enable-libgsm --enable-libjack --enable-libmp3lame --enable-libmysofa --enable-libopenjpeg --enable-libopenmpt --enable-libopus --enable-libpulse --enable-librsvg --enable-librubberband --enable-libshine --enable-libsnappy --enable-libsoxr --enable-libspeex --enable-libssh --enable-libtheora --enable-libtwolame --enable-libvidstab --enable-libvorbis --enable-libvpx --enable-libwavpack --enable-libwebp --enable-libx265 --enable-libxml2 --enable-libxvid --enable-libzmq --enable-libzvbi --enable-lv2 --enable-omx --enable-openal --enable-opencl --enable-opengl --enable-sdl2 --enable-libdc1394 --enable-libdrm --enable-libiec61883 --enable-chromaprint --enable-frei0r --enable-libx264 --enable-libdrm --enable-librga --enable-rkmpp --enable-version3 --disable-libopenh264 --disable-vaapi --disable-vdpau --disable-decoder=h264_v4l2m2m --disable-decoder=vp8_v4l2m2m --disable-decoder=mpeg2_v4l2m2m --disable-decoder=mpeg4_v4l2m2m --disable-muxer='ac3,eac3,mlp,truehd' --disable-encoder='ac3_fixed,ac3,mlp,spdif,truehd' --disable-demuxer='ac3,eac3,mlp,truehd,dts,dtshd' --disable-parser='aac,ac3,mlp' --disable-decoder='ac3,eac3,mlp,dolby_e' --enable-shared --disable-doc
  libavutil      56. 31.100 / 56. 31.100
  libavcodec     58. 54.100 / 58. 54.100
  libavformat    58. 29.100 / 58. 29.100
  libavdevice    58.  8.100 / 58.  8.100
  libavfilter     7. 57.100 /  7. 57.100
  libavresample   4.  0.  0 /  4.  0.  0
  libswscale      5.  5.100 /  5.  5.100
  libswresample   3.  5.100 /  3.  5.100
  libpostproc    55.  5.100 / 55.  5.100
Option -pix_fmt is deprecated, use -pixel_format.
libGL error: failed to create dri screen
libGL error: failed to load driver: rockchip
libGL error: failed to create dri screen
libGL error: failed to load driver: rockchip
[rawvideo @ 0x7f3c000ba0] Estimating duration from bitrate, this may be inaccurate
Input #0, rawvideo, from './out.yuv':
```

```
21    Duration: 00:00:04.00, start: 0.000000, bitrate: 622075 kb/s
22      Stream #0:0: Video: rawvideo (NV12 / 0x3231564E), nv12, 1920x1080, 622
    080 kb/s, 25 tbr, 25 tbn, 25 tbc
23
```



11、RTC

主板包含2个RTC，其中/dev/rtc0为外部RTC（HYM8563），/dev/rtc1为CPU内部的RTC（RK808）。系统默认使用rtc0的时间。

11.1 获取RTC时间

Bash

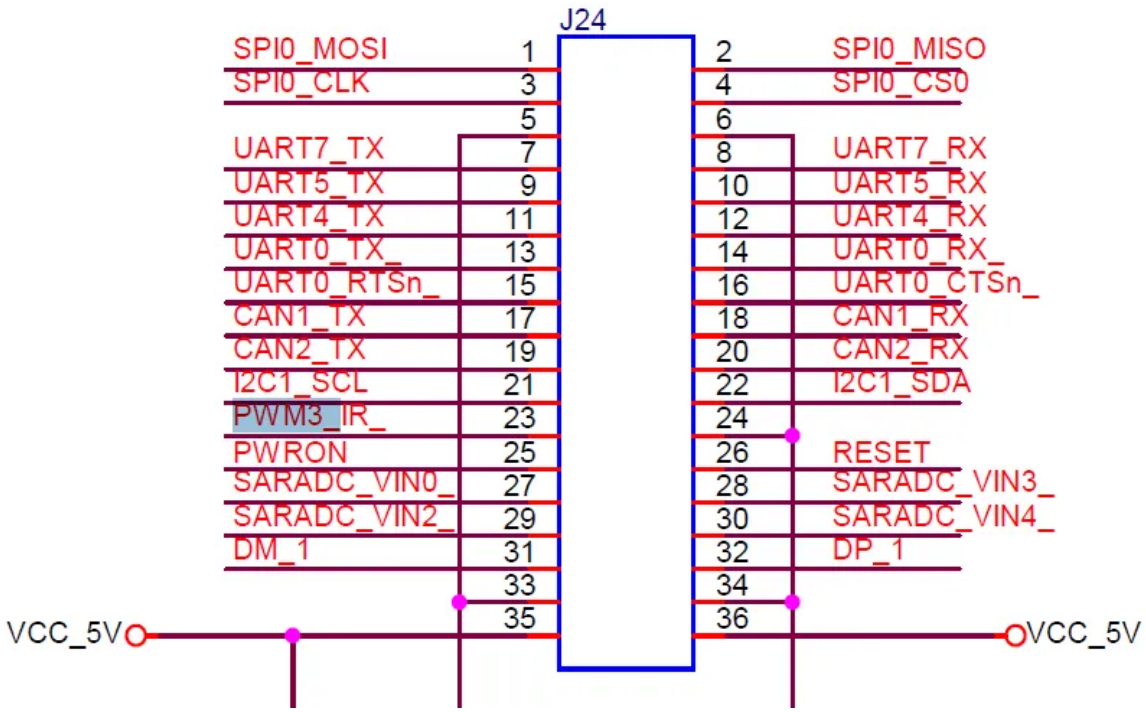
```
1   root@ido:~# hwclock
2   2022-11-10 02:16:23.617474+00:00
```

11.2 设置RTC时间

```bash
root@ido:~# hwclock
2022-11-18 08:30:40.778874+00:00
root@ido:~# date -s '2022-11-10 10:17:00'
Thu Nov 10 10:17:00 UTC 2022
root@ido:~# hwclock -w
root@ido:~# hwclock
2022-11-18 08:31:06.829691+00:00
root@ido:~#
```

12、PWM功能

PWM位于J24双排针。



《双排针图》

序号	定义	电平	说明
23	PWM3_IR_	3.3V	

12.1 测试

```bash
cd /sys/class/pwm/pwmchip0/
```

开启PWM:

```bash
echo 0 > export
cd pwm0
```

设置周期:

```bash
echo 330000> period
```

设置占空比:

```bash
echo 150000 > duty_cycle
```

使能PWM:

```bash
echo 1 > enable
```

失能PWM:

```bash
echo 0 > enable
```

13、开机自启动

默认系统开机会运行/etc/rc.local脚本，将要开机执行的程序放到该脚本中即可。

以设置开机运行/usr/sbin/app为例。

编辑/etc/rc.local（如果不存在则创建，且赋予可执行权限）：

```Bash
#! /bin/sh
export PATH="${PATH:+$PATH:}/usr/sbin:/sbin"

case "$1" in
  start)
                /usr/sbin/app
        ;;
esac

exit 0
```

这样每次开机后，/usr/sbin/app就会执行。

14、屏幕控制

14.1 背光调节

通过修改/sys/class/backlight/backlight/brightness的值，实现背光的调节，范围取0-255，值越大，亮度越高。（/backlight/brightness没有这个） (backlight/brightness没找到)

设置亮度为100：

```Bash
root@ido:~# echo 100 >  /sys/class/backlight/backlight/brightness
root@ido:~#
```

14.2 屏幕旋转

使用xrandr工具可以实现屏幕的旋转。

14.2.1 临时旋转

系统启动后，执行xrandr -o normal,inverted,left,right，可以实现临时旋转屏幕方向，其中normal表示顺时针旋转0度，inverted表示顺时针旋转180度，left表示顺时针旋转270度，right表示顺时针旋转90度。

```bash
root@ido:~# xrandr -o inverted
```

14.2.2 永久旋转

修改/etc/default/xrandr启动文件，可以实现永久旋转。

以旋转180度为例：

```bash
root@ido:~# cat /etc/default/xrandr
#!/bin/sh
        /usr/bin/xrandr -o inverted

root@ido:~#
```

这样修改后，每次重启设备，桌面将旋转180度。

15、按键

主板配置了一个按键SW2，作为电源键使用，对应的设备节点为/dev/input/event0。

系统运行时，短按该按键上报KEY_POWER，并且进入待机状态。

系统待机时，短按该按键，系统恢复正常运行。

系统运行时，长按该按键5秒关机。

系统关机时，短按该按键开机。

16、ADC

主板配置了4路ADC，位于J24的第7.8.9,10引脚，分别记作ADC0、ADC2、ADC3、ADC4，精度为10位。

16.1 ADC转换方法

V = (raw/1024)*1.8v

其中raw为对应设备节点读取的值，范围为0-1023。

序号	编号	设备节点
9	SARADC VINU0_	/sys/bus/iio/devices/iio:device0/in_voltage0_raw
7	SARADC VINU2_	/sys/bus/iio/devices/iio:device0/in_voltage2_raw
10	SARADC VINU3_	/sys/bus/iio/devices/iio:device0/in_voltage3_raw
8	SARADC VINU4_	/sys/bus/iio/devices/iio:device0/in_voltage4_raw

16.2 测试

以测试ADC2为例，其余ADC测试方法类似。

```Bash
root@ido:~# cat /sys/bus/iio/devices/iio:device0/in_voltage3_raw
997
```

设备节点读取的raw值为997，代入到公式计算：

$$V=(997/1024)*1.8v=1.75v$$

即ADC1输入的电压为1.75v。

17、网络优先级设置

主板支持以太网、WiFi和4G/5G三种网络，通过路由表来设置它们的网络优先级。

17.1 查看路由表

```
root@ido:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         _gateway        0.0.0.0         UG    100    0        0 eth0
default         _gateway        0.0.0.0         UG    600    0        0 wlan0
192.168.1.0     0.0.0.0         255.255.255.0   U     100    0        0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     600    0        0 wlan0
```

17.2 设置默认路由

17.2.1 设置WiFi为默认路由

当前以太网和WiFi同时使用，设置WiFi优先：

```
root@ido:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         _gateway        0.0.0.0         UG    100    0        0 eth0
default         _gateway        0.0.0.0         UG    600    0        0 wlan0
192.168.1.0     0.0.0.0         255.255.255.0   U     100    0        0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     600    0        0 wlan0
root@ido:~# route del default dev eth0
root@ido:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         _gateway        0.0.0.0         UG    600    0        0 wlan0
192.168.1.0     0.0.0.0         255.255.255.0   U     100    0        0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     600    0        0 wlan0
```

这样默认路由就是wlan0了，即优先使用WiFi进行数据通信。

17.2.2 设置以太网为默认路由

当前以太网和WiFi同时使用，且WiFi优先：

```
root@ido:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         _gateway        0.0.0.0         UG    600    0        0 wlan0
192.168.1.0     0.0.0.0         255.255.255.0   U     100    0        0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     600    0        0 wlan0
root@ido:~#
```

设置为以太网优先：

```
root@ido:~# route del default dev wlan0
root@ido:~# route add default dev eth0
root@ido:~# route add default gw 192.168.1.1
root@ido:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         0.0.0.0         0.0.0.0         U     0      0        0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     100    0        0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     600    0        0 wlan0
root@ido:~#
```

其他情况按照类似的方法进行处理即可。

18、CAN

CAN位于J24的双排针。共有3路CAN可供使用。

序号	编号	描述
19	CAN1_TX	CAN1
20	CAN1_RX	
17	CAN2_TX	CAN2
18	CAN2_RX	
15	CAN0_TX (I2C_SCL)	CAN0 当作为CAN0时需要关闭I2C1
16	CAN0_RX (I2C_SDA)	

注意：由于开发板未带有CAN芯片，所以这里需要用到转接板进行测试。

18.1测试

测试需要用USB转CAN工具，通过PC与板上CAN通信。

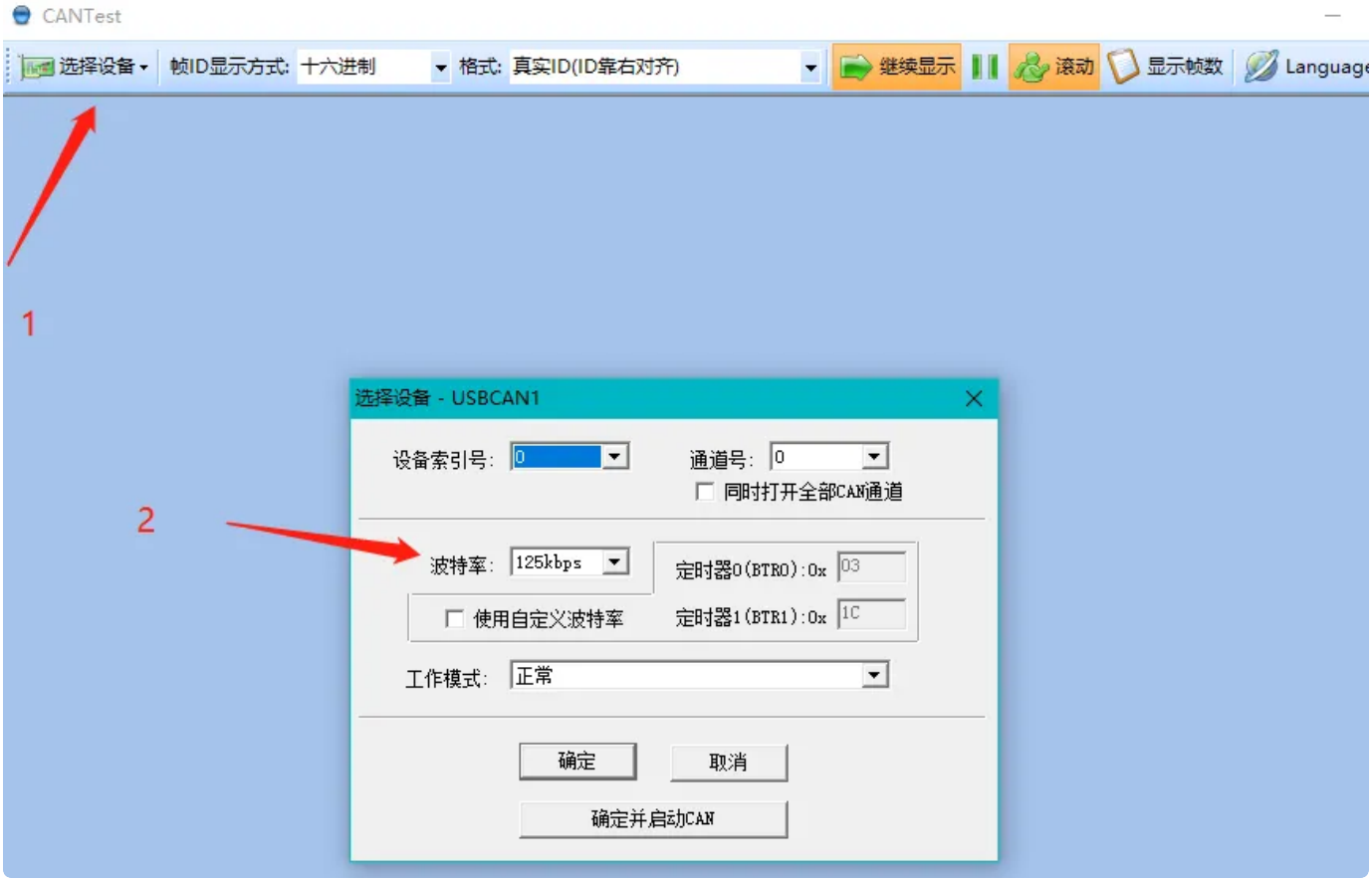
这里以CAN0为例，其余节点测试方法相同：

Bash

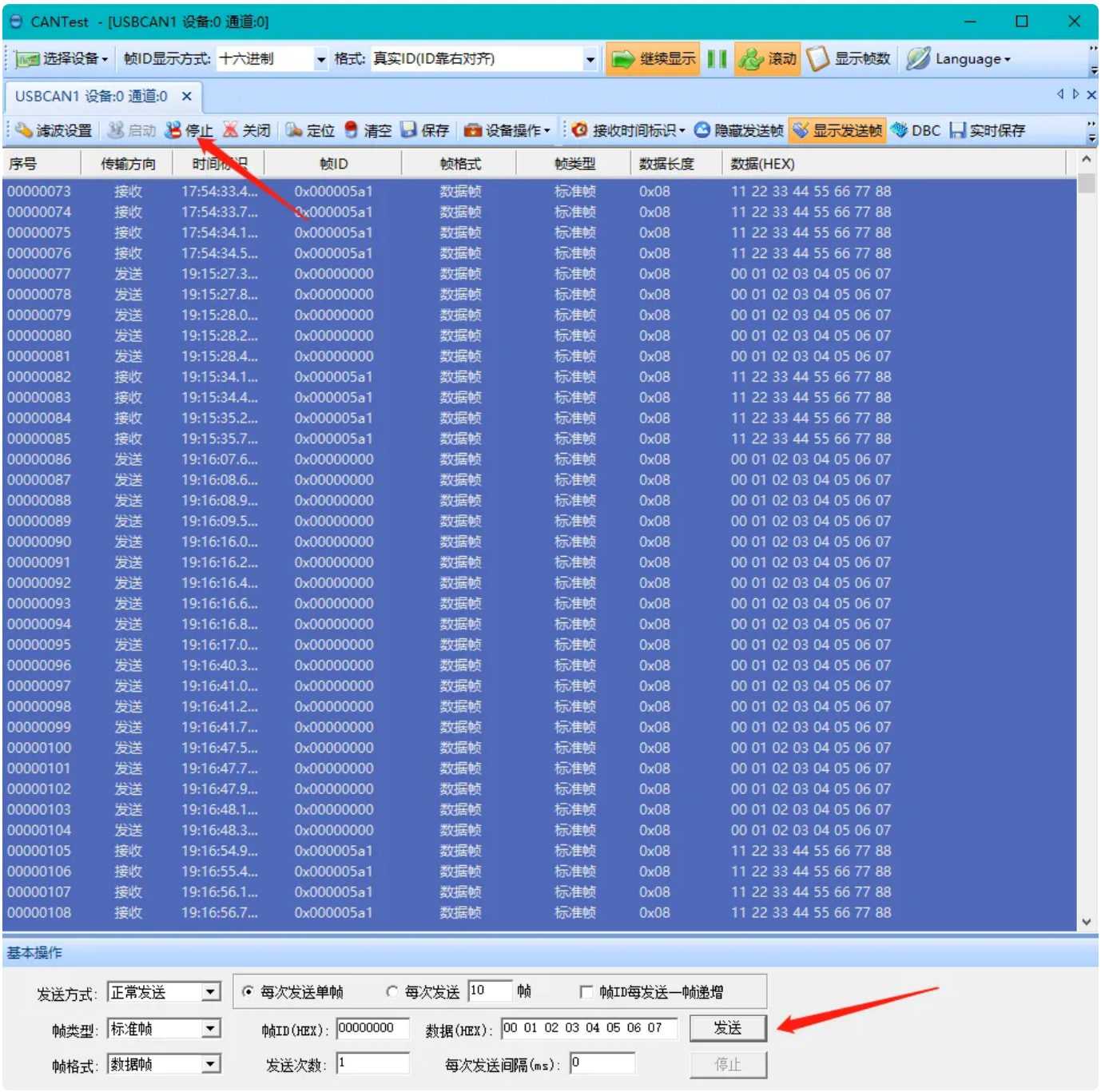
```
//关掉can
ifconfig can0 down

//配置can通信的波特率
ip link set can0 type can bitrate 125000 triple-sampling on

//开启can通信
ifconfig can0 up

//作为接收端接收数据
candump can0

//作为发送端发送数据
cansend can0 5A1#1122334455667788

```

PC软件端的接收与发送：

(1) 选择USBCAN1


CANTest
选择设备
帧ID显示方式: 十六进制
格式: 真实ID(ID靠右对齐)
继续显示
滚动
显示帧数
Language
1
2
选择设备 - USBCAN1
设备索引号: 0
通道号: 0
同时打开全部CAN通道
波特率: 125kbps
定时器0(BTR0): 0x 03
使用自定义波特率
定时器1(BTR1): 0x 1C
工作模式: 正常
确定
取消
确定并启动CAN

（2）启动CAN测试



板端发送过来的数据可以在CANtest上打印出来。