

# 实验11 linux室温计

---

## 实验目的

---

1. 了解嵌入式Linux板卡一般情况；
2. 熟悉RPi/pcDuino的供电、调试串口等的接线方式；
3. 复习Linux启动过程；
4. 掌握Linux的以太网和WiFi配置；
5. 掌握Linux的SSH配置；
6. 掌握PC上的SSH软件；
7. 掌握嵌入式板卡和PC建立文件共享的方式；
8. 寻找和安装交叉编译环境；
9. 熟悉嵌入式板卡的Linux下的编程环境；
10. 了解远程访问嵌入式板卡图形桌面的方式；
11. 熟练掌握嵌入式Linux应用程序访问GPIO的方式。

## 实验器材

---

### 硬件

- 树莓派1块；
- 5V/1A电源1个；
- microUSB线1根；
- USB-TTL串口线1根；
- 以太网线一根；
- HDMI显示器；
- HDMI线；
- USB键盘/鼠标；
- USB Hub。

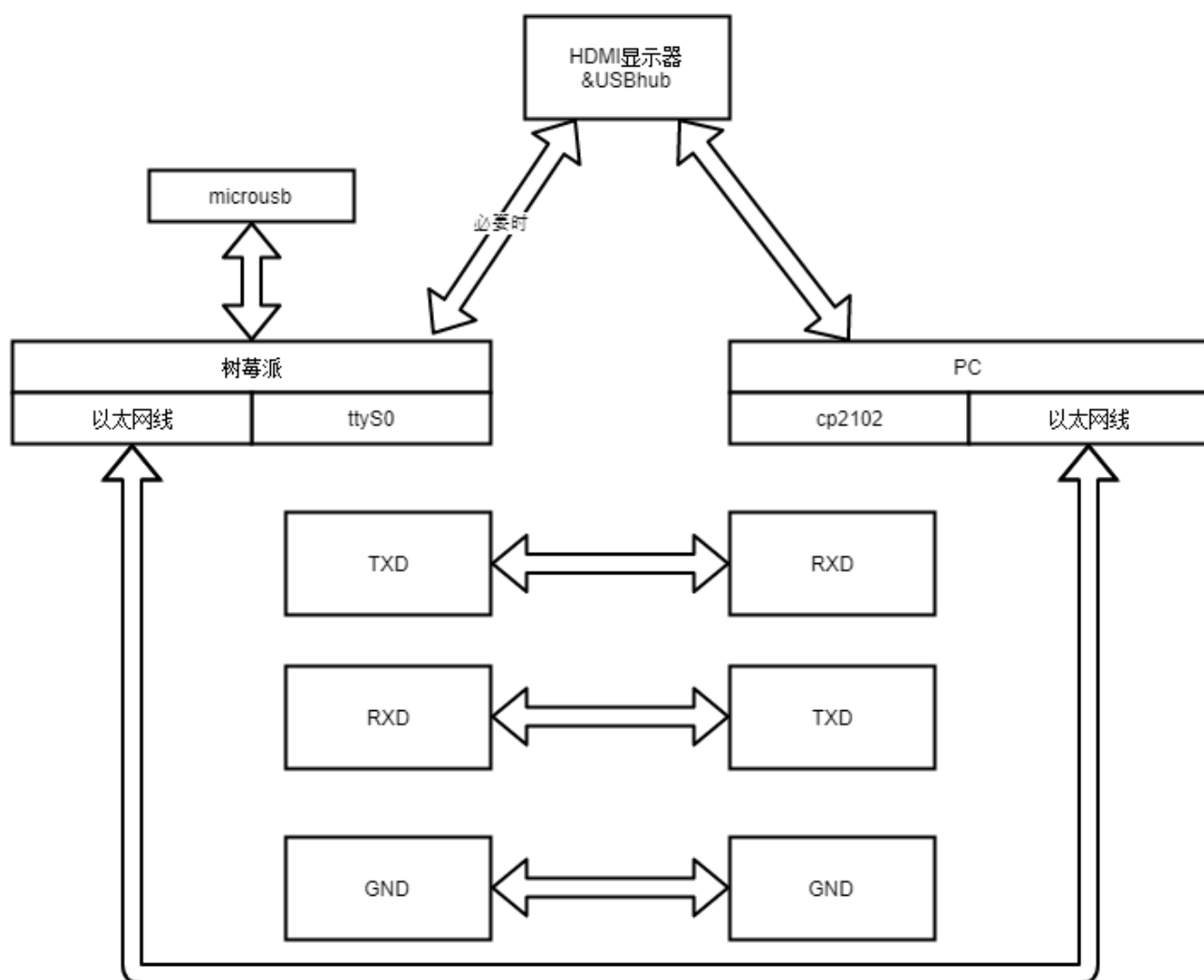
### 软件

- PC上的USB-TTL串口线配套的驱动程序；
- PC上的串口终端软件putty；
- PC上的SSH软件putty。
- PC上的SFTP软件Filezilla&FlashFXP
- 交叉编译软件。

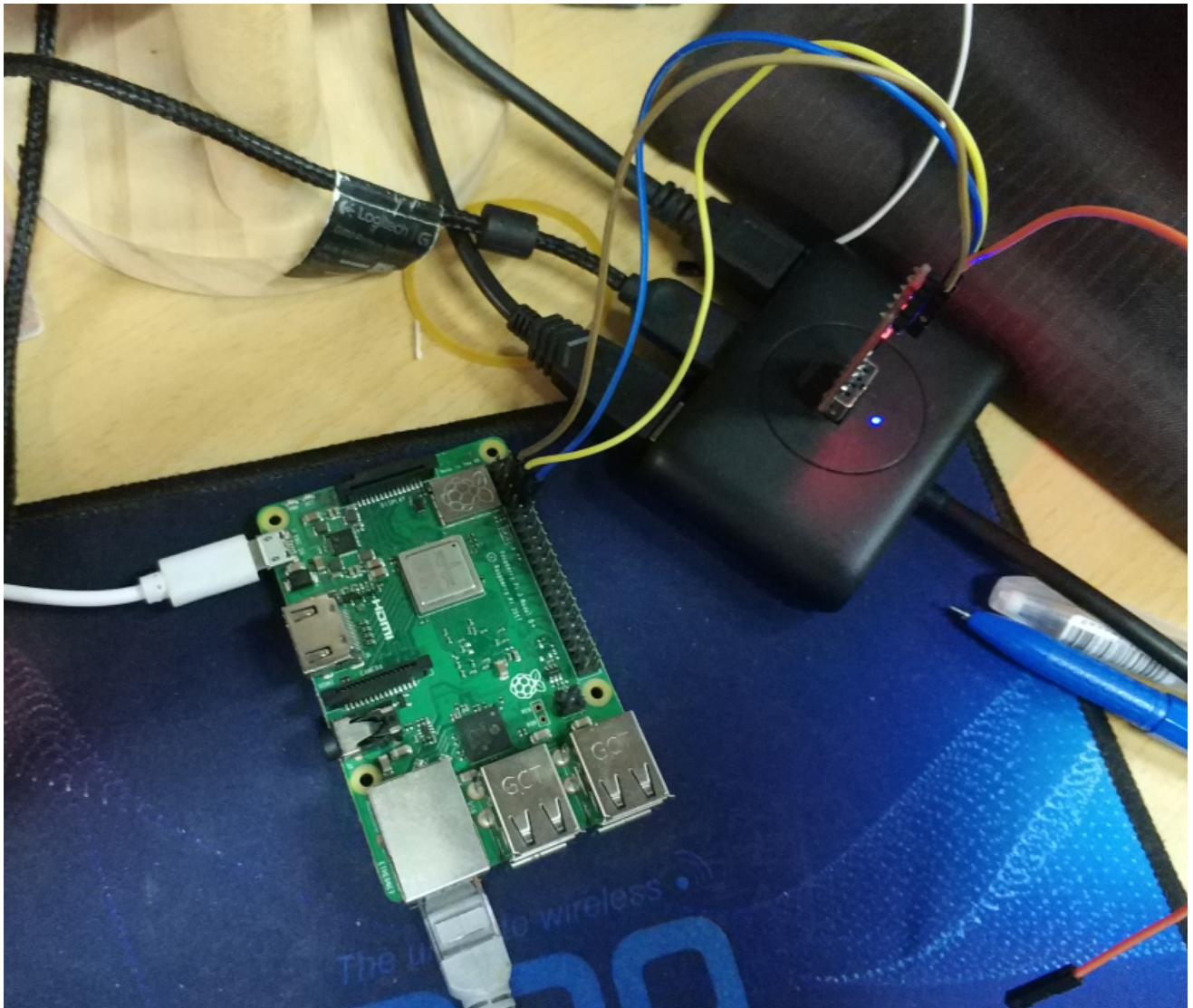
## 实验接线

---

### 示意图



实际图



## 实验步骤

1~3见上述内容

### 4.通过cp2102串口连接树莓派时的输出文字

```
1 pi@raspberrypi:~$ sudo reboot
2
3 [ 193.530690] reboot: Restarting system
4
5 [ 7.367174] Under-voltage detected! (0x00050005)
6
7
8
9
10 Raspbian GNU/Linux 10 raspberrypi ttyS0
11
12
13
14 raspberrypi login: pi
15
```

```
16
17 密码:
18
19 上一次登录: 三 5月 27 11:59:02 CST 2020tty1 上
20
21 Linux raspberrypi 4.19.97-v7+ #1294 SMP Thu Jan 30 13:15:58 GMT 2020 armv7l
22
23
24
25 The programs included with the Debian GNU/Linux system are free software;
26
27 the exact distribution terms for each program are described in the
28
29 individual files in /usr/share/doc/*/copyright.
30
31
32
33 Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
34
35 permitted by applicable law.
36
37 pi@raspberrypi:~$
38
```

pi@raspberrypi:~\$ sudo reboot

[ 193.530690] reboot: Restarting system

[ 7.367174] Under-voltage detected! (0x00050005)

Raspbian GNU/Linux 10 raspberrypi ttyS0

raspberrypi login: pi

密码:

上一次登录: 三 5月 27 11:59:02 CST 2020tty1 上

Linux raspberrypi 4.19.97-v7+ #1294 SMP Thu Jan 30 13:15:58 GMT 2020 armv7l

The programs included with the Debian GNU/Linux system are free software;

the exact distribution terms for each program are described in the

individual files in /usr/share/doc/\*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent

permitted by applicable law.

pi@raspberrypi:~\$

截图如下:

```
pi@raspberrypi:~$ sudo reboot
[ 193.530690] reboot: Restarting system
[ 7.367174] Under-voltage detected! (0x00050005)

Raspbian GNU/Linux 10 raspberrypi ttyS0

raspberrypi login: pi

密码:

上一次登录: 三 5月 27 11:59:02 CST 2020tty1 上
Linux raspberrypi 4.19.97-v7+ #1294 SMP Thu Jan 30 13:15:58 GMT 2020 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

pi@raspberrypi:~$ █
```

## 5.通过linux获得硬件数据并截屏给出

- 通过 `cat /proc/cpuinfo` 获得CPU型号

```
pi@raspberrypi:~$ cat /proc/cpuinfo

processor       : 0
model name     : ARMv7 Processor rev 4 (v7l)
BogoMIPS      : 38.40
Features       : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
vfpd32 lpae evtstrm crc32
CPU implementer : 0x41
CPU architecture: 7
CPU variant    : 0x0
CPU part       : 0xd03
CPU revision   : 4

processor       : 1
model name     : ARMv7 Processor rev 4 (v7l)
BogoMIPS      : 38.40
Features       : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
vfpd32 lpae evtstrm crc32
CPU implementer : 0x41
CPU architecture: 7
CPU variant    : 0x0
CPU part       : 0xd03
CPU revision   : 4
```

剩下两个处理器的信息与前两个完全一致，不再给出截图

```
Hardware       : BCM2835
Revision       : a020d3
Serial        : 000000006e1ec980
Model         : Raspberry Pi 3 Model B Plus Rev 1.3
```

- 通过简单的shell脚本可以得到树莓派的所有硬件时钟频率

```
1 for src in arm core h264 isp v3d uart pwm emmc pixel vec hdmi dpi ; do \
2     echo -e "$src:\t$(vcgencmd measure_clock $src)" ; \
3 done
```

```
pi@raspberrypi:~$ for src in arm core h264 isp v3d uart pwm emmc pixel vec hdmi
dpi ; do \
>   echo -e "$src:\t$(vcgencmd measure_clock $src)" ; \
> done
arm:    frequency(45)=600000000
core:   frequency(1)=250000000
h264:   frequency(28)=0
isp:    frequency(42)=250000000
v3d:    frequency(43)=250000000
uart:   frequency(22)=48000000
pwm:    frequency(25)=0
emmc:   frequency(47)=200000000
pixel:  frequency(29)=154000000
vec:    frequency(10)=0
hdmi:   frequency(9)=163683000
dpi:    frequency(4)=0
```

- 通过 `free -h` 指令获取内存大小

```
pi@raspberrypi:~$ free -h
```

	total	used	free	shared	buff/cache	available
Mem:	926Mi	128Mi	611Mi	6.0Mi	186Mi	739Mi
Swap:	99Mi	0B	99Mi			

## 6. 给出网络配置参数

我通过 wlan 给树莓派连接网络，通过 `ifconfig wlan0` 指令来查看连接状态

```
pi@raspberrypi:~$ ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 10.181.166.219  netmask 255.255.192.0  broadcast 10.181.191.255
    inet6 fe80::d1c3:2d37:b231:45a0  prefixlen 64  scopeid 0x20<link>
    ether b8:27:eb:4b:9c:d5  txqueuelen 1000  (Ethernet)
    RX packets 15715  bytes 4474738 (4.2 MiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 67  bytes 9292 (9.0 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

在PC端ping树莓派ip 10.181.176.46 (因为中间断连过一次, 所以树莓派ip变了), 可以ping通

```
C:\Users\Rookie>ping 10.181.176.46

正在 Ping 10.181.176.46 具有 32 字节的数据:
来自 10.181.176.46 的回复: 字节=32 时间=3ms TTL=62
来自 10.181.176.46 的回复: 字节=32 时间=2ms TTL=62
来自 10.181.176.46 的回复: 字节=32 时间=2ms TTL=62
来自 10.181.176.46 的回复: 字节=32 时间=2ms TTL=62

10.181.176.46 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 2ms, 最长 = 3ms, 平均 = 2ms
```

在树莓派端ping百度, 可以ping通

```
pi@raspberrypi:~$ ping www.baidu.com

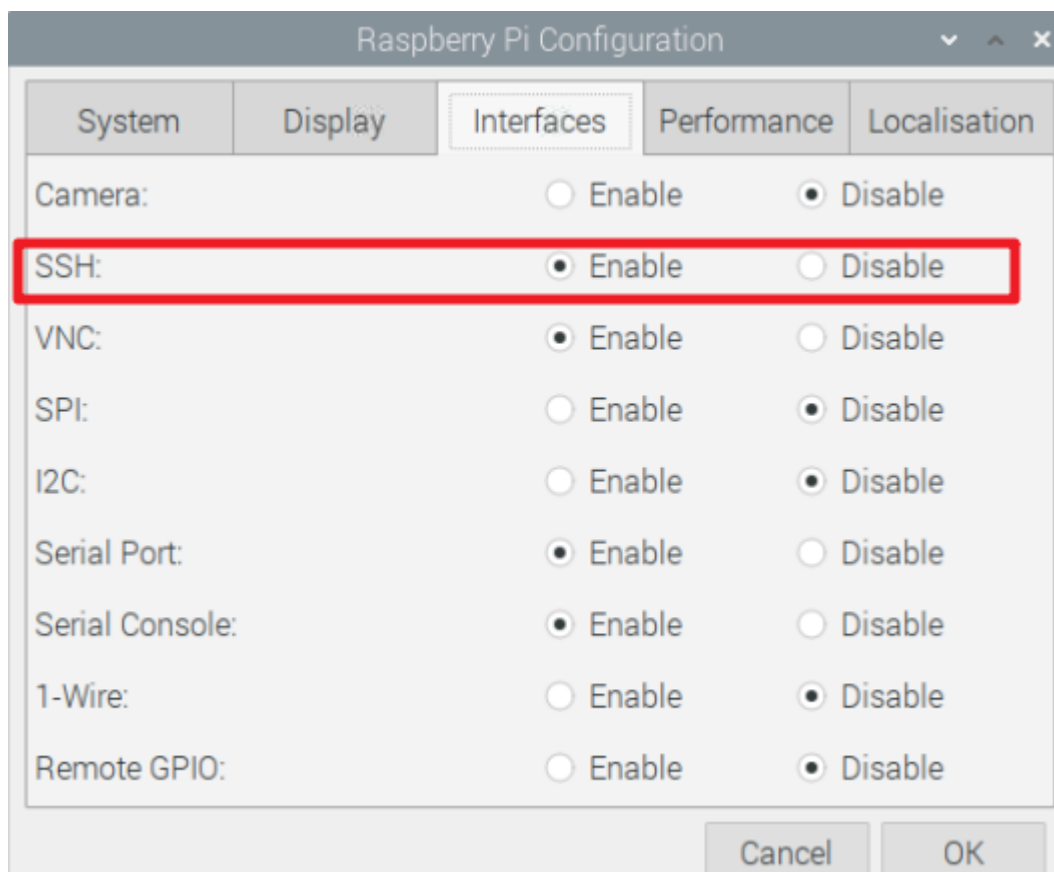
PING www.a.shifen.com (36.152.44.95) 56(84) bytes of data.

64 bytes from 36.152.44.95 (36.152.44.95): icmp_seq=1 ttl=55 time=8.84 ms
64 bytes from 36.152.44.95 (36.152.44.95): icmp_seq=2 ttl=55 time=8.91 ms
64 bytes from 36.152.44.95 (36.152.44.95): icmp_seq=3 ttl=55 time=10.2 ms
64 bytes from 36.152.44.95 (36.152.44.95): icmp_seq=4 ttl=55 time=9.99 ms
```

## 7.给出SSH配置文件

在configuration中打开ssh即可, 没有进行额外的配置





## 8.多个登录时，如何看到不同端口的登录

在终端输入 `w` 指令，可以看到不同端口的登录情况

```
pi@raspberrypi:~ $ w
 17:14:50 up 7 min,  4 users,  load average: 0.00, 0.07, 0.06
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU WHAT
pi        ttyS0    -               17:07    3:19   0.74s  0.46s -bash
pi        tty1     -               17:07    7:23   0.57s  0.44s -bash
pi        tty7     :0              17:07    7:23   1.52s  0.41s /usr/bin/lxsess
pi        pts/0    222.205.21.247  17:09    0.00s  0.45s  0.03s w
pi@raspberrypi:~ $
```

00:05:51 Connected SSH/22

其中：

- ttyS0是串口登录
- pts/0是ssh登录

尝试用 `write` 从串口向SSH发信息：

- 串口端：

```
pi@raspberrypi:~$ write pi pts/0
write: write: you have write permission turned off.

hello ssh, this is serial

^Cpi@raspberrypi:~$
```

- SSH端:

```
pi@raspberrypi:~$
Message from pi@raspberrypi on ttyS0 at 15:58 ...

hello ssh, this is serial

EOF
```

## 9.SAMBA配置文件内容并解释

```
1 [share] # 该共享的共享名
2     comment = share
3     path = /home/pi/test # 共享路径
4     guest ok = yes
5     available = yes # 设置该共享目录是否可用
6     browseable = yes # 设置该共享目录是否可显示
7     public = yes # 指明该共享资源是否能给游客帐号访问
8     writable = yes # 指定了这个目录缺省是否可写
9     read only = no
```

## 10.给出各种方式传递文件的过程并比较

### SAMBA

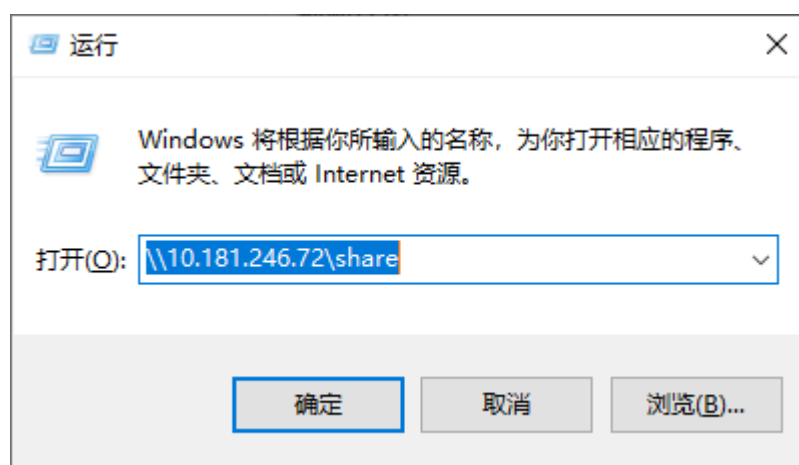
1. 在树莓派上安装SAMBA相关软件

```
pi@raspberrypi:~$ sudo apt-get install samba samba-common-bin
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
samba 已经是最新版 (2:4.9.5+dfsg-5+deb10u1+rpi1)。
samba-common-bin 已经是最新版 (2:4.9.5+dfsg-5+deb10u1+rpi1)。
升级了 0 个软件包，新安装了 0 个软件包，要卸载 0 个软件包，有 174 个软件包未被
升级。
pi@raspberrypi:~$
```

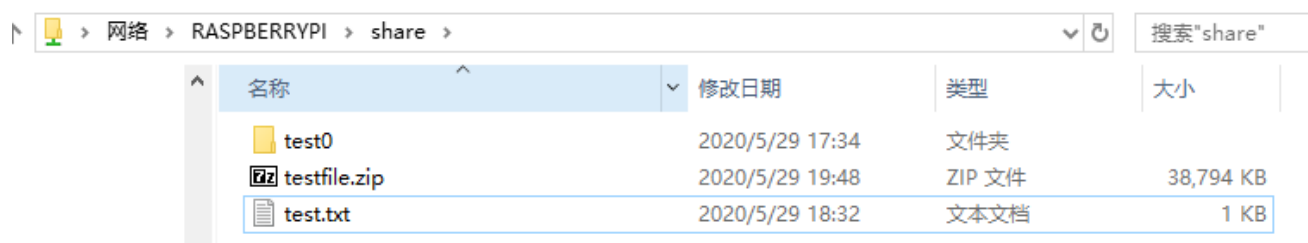
2. 在 /home/pi 中新建文件夹 test 作为共享路径
3. 通过 `sudo chmod 777 /home/android/test`，允许owner之外的用户写入
4. 如9中配置SAMBA
5. 通过 `sudo service smbd restart` 重启SAMBA

6. `sudo smbpasswd -a pi` 添加samba共享用户

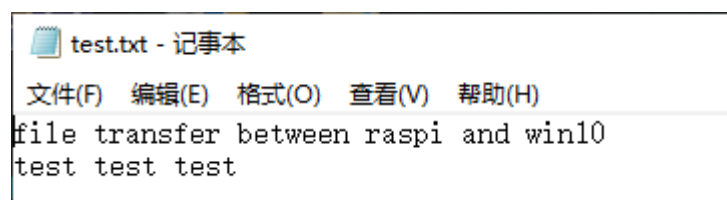
7. 在pc端进入共享文件夹如下图：

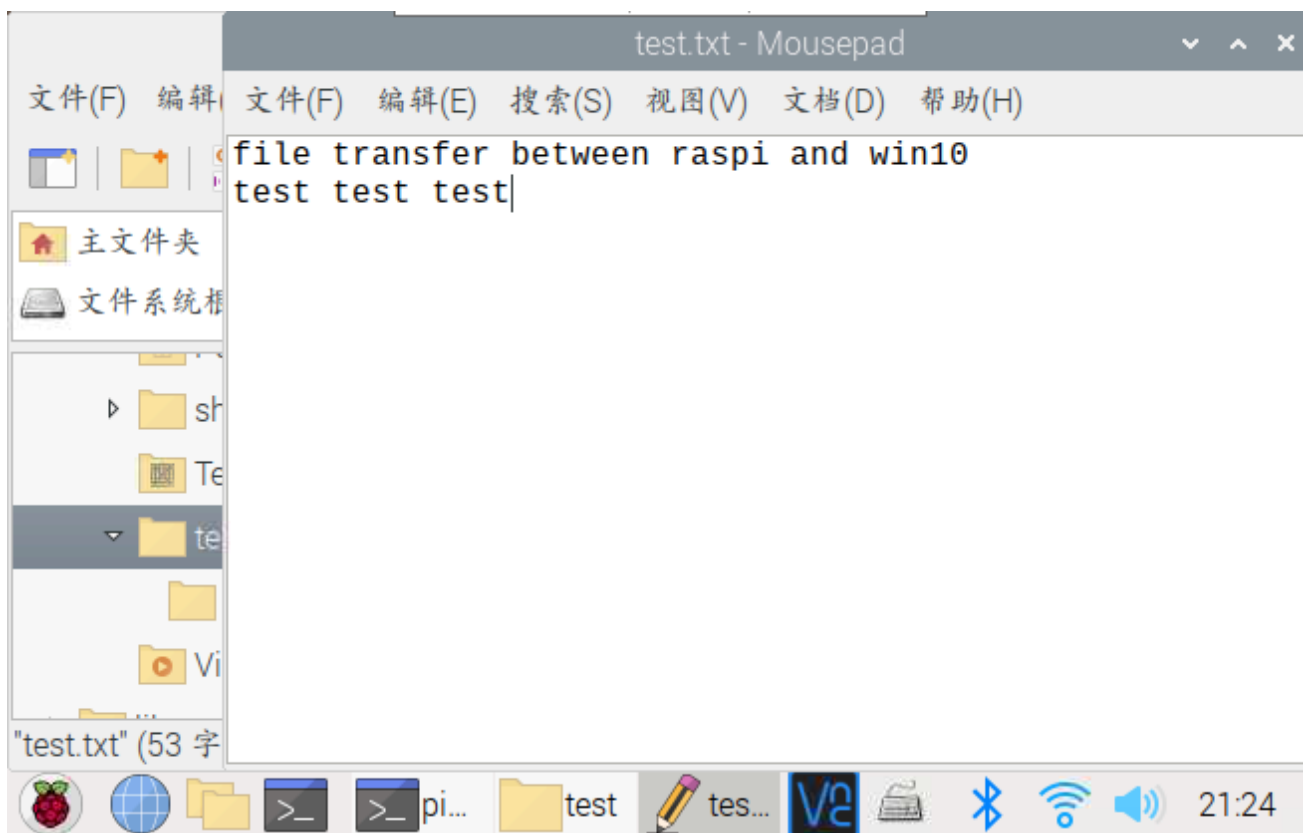


8. 想本地路径一样将文件拖入即可



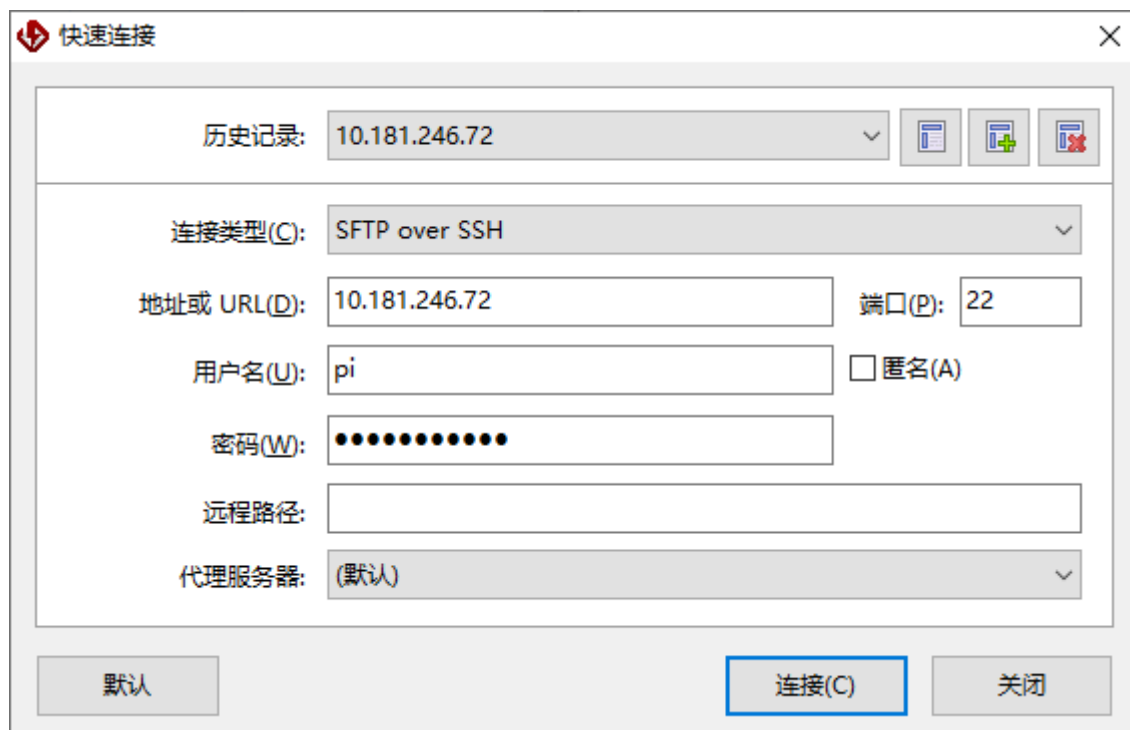
9. 打开文件发现与pc端内容相同，证明传输成功





## SFTP

1. 这里采用FlashFXP来进行SFTP文件传输，输入ip等必要信息并建立链接



2. 可以看到pc通过FlashFXP成功连接到树莓派，如下图：



/home/pi/

名称	大小	修改时间	属性
上级目录			
.bluej	4 KB	2020/5/22 12:57:45	drwxr-xr-x
.cache	4 KB	2020/2/14 0:37:52	drwxr-xr-x
.config	4 KB	2020/5/29 16:19:13	drwx-----
.gnupg	4 KB	2020/2/14 0:31:52	drwx-----
.java	4 KB	2020/5/22 12:52:58	drwxr-xr-x
.local	4 KB	2020/2/14 0:03:48	drwxr-xr-x
.pki	4 KB	2020/2/14 0:37:54	drwx-----
.pp_backup	4 KB	2020/5/29 15:44:16	drwx-----
.scim	4 KB	2020/5/22 13:24:25	drwx-----
.thumbnails	4 KB	2020/5/29 16:19:17	drwx-----
.vnc	4 KB	2020/5/29 16:31:55	drwx-----
Desktop	4 KB	2020/5/29 16:19:17	drwxr-xr-x
Documents	4 KB	2020/2/14 0:32:13	drwxr-xr-x
Downloads	4 KB	2020/5/22 12:52:05	drwxr-xr-x
MagPi	4 KB	2020/2/14 0:03:49	drwxr-xr-x
Music	4 KB	2020/2/14 0:32:13	drwxr-xr-x
Pictures	4 KB	2020/2/14 0:32:13	drwxr-xr-x
Public	4 KB	2020/2/14 0:32:13	drwxr-xr-x
share	4 KB	2020/5/29 17:04:13	drwxrwxrwx
Templates	4 KB	2020/2/14 0:32:13	drwxr-xr-x

7 个文件, 22 个文件夹, 共计 29 项, 已选定 1 项 (0 字节)

10.181.246.72

<implicit>, 压缩: none.

[18:27:42] [R] Auth Type: Password

[18:27:42] [R] 身份验证成功

[18:27:42] [R] SSH 连接打开

[18:27:42] [R] 已建立连接对象: OpenSSH\_7.9p1 Raspbian-10+deb10u2 (SFTP v3)

[18:27:42] [R] SFTP 连接就绪

[18:27:43] [R] 获取目录列表中.....

[18:27:43] [R] 列表完成: 2 KB 耗时 0.07 秒 (2.8 KB/s)

[18:29:19] [R] 目录更改进度: /home/pi/Desktop/

[18:29:19] [R] 获取目录列表中.....

[18:29:19] [R] 列表完成: 267 字节 耗时 0.04 秒 (0.3 KB/s)

3. 通过SFTP向树莓派传输 test.txt 文件, 可以看到成功传输

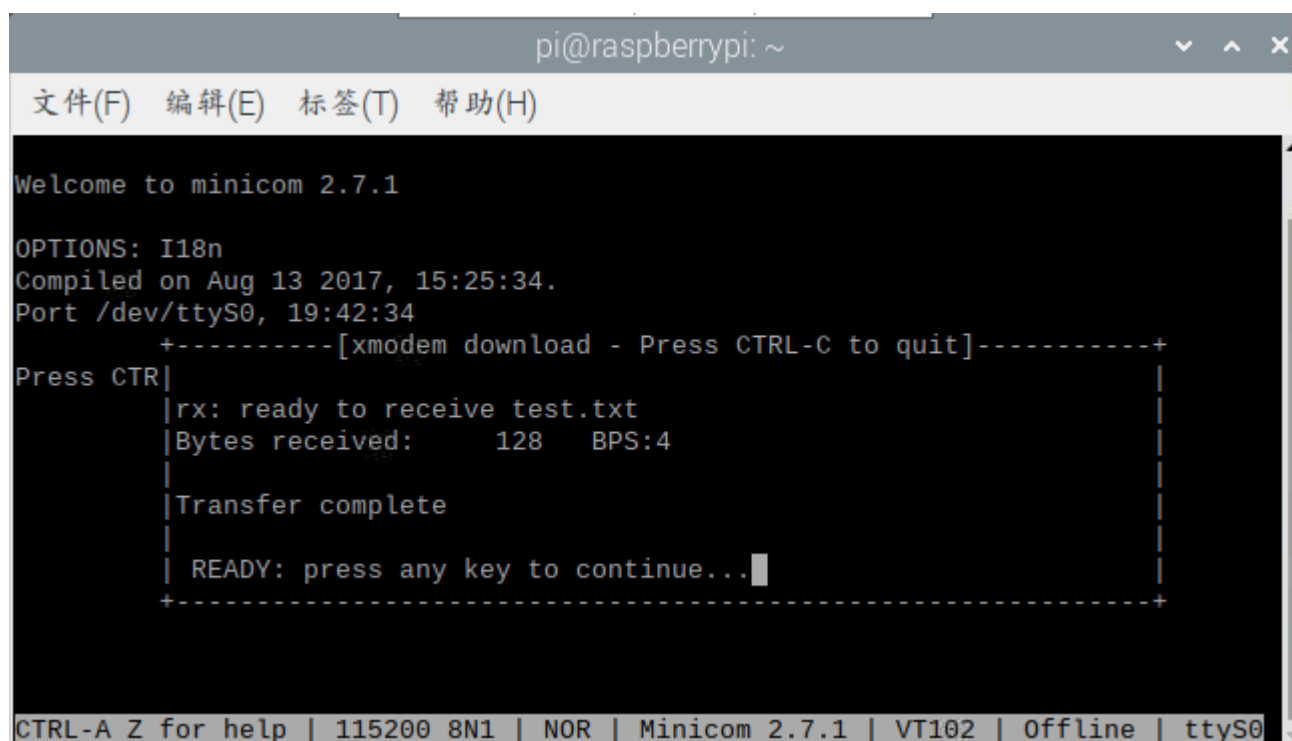
/home/pi/Desktop/

名称	大小	修改时间	属性
上级目录			
smb.conf	8 KB	2020/5/29 16:19:17	-rw-r--r--
test.txt	53	2020/5/29 18:32:02	-rw-r--r--

## Xmodem

1. `sudo apt-get install minicom` 安装minicom
2. 因为我用的串口是 `ttys0` 所以通过 `minicom -D /dev/ttys0` 打开minicom->ctrl+a->z->r->Xmodem
3. 在pc端通过putty连接串口，并通过Xmodem发送文件

Xmodem尝试好几次才成功一次，不知道是不是因为TIMEOUT时间设得太短了，传输test.txt成功结果如下：



```
pi@raspberrypi: ~
文件(F) 编辑(E) 标签(T) 帮助(H)

Welcome to minicom 2.7.1

OPTIONS: I18n
Compiled on Aug 13 2017, 15:25:34.
Port /dev/ttys0, 19:42:34
+-----[xmodem download - Press CTRL-C to quit]-----+
Press CTRL|
|rx: ready to receive test.txt|
|Bytes received: 128 BPS:4|
|Transfer complete|
| READY: press any key to continue...|
+-----+

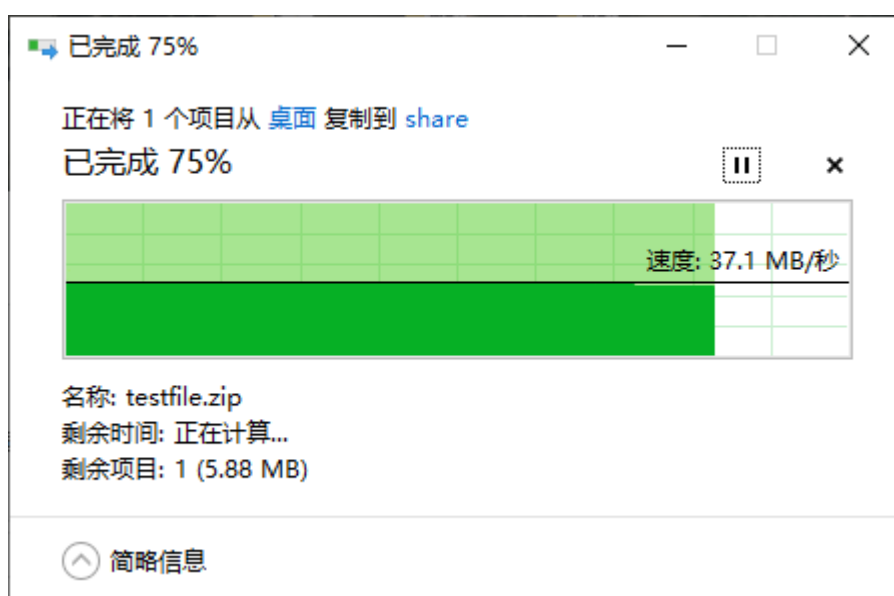
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | ttys0
```

因为Xmodem有校验，一般情况下，显示Transfer complete即为传输成功

## 比较

为了比较以上几种方法，需要传输一个稍大点的文件进行测试，这里我准备了一个39.05MB的压缩文件进行测试

- SAMBA：没有精确的时间，但从截图可以看到传输速度极快，达到了37.1MB/S，传输过程几乎在瞬间完成



- SFTP：传输耗时：9秒，速度4.38MB/s

```
[18:46:17] [R] 正在上传: /home/pi/Desktop/testfile.zip
[18:46:26] 上传: testfile.zip 39.05 MB 耗时 9 秒 (4.38 MB/s)
[18:46:26] [R] 获取目录列表中.....
[18:46:26] [R] 列表完成: 461 字节 耗时 0.03 秒 (0.5 KB/s)
[18:46:26] 传输队列已完成
[18:46:26] 已传输 1 个文件 (39.05 MB) 耗时 9 秒 (4.34 MB/s)
```

- Xmodem：使用Xmodem传输压缩文件，尝试了多次都失败了，原因可能是因为文件太大。这也体现看Xmodem相当不稳定也相当慢。

综上：

- 就传输速度而言SAMBA->SFTP->Xmodem三者传输速度依次降低
- 就易用性而言SFTP最方便，SAMBA其次，Xmodem极其难用，尝试多次才成功
- 就安全性而言，SAMBA安全性一般。SFTP经过SSH连接加密，安全可靠，Xmodem安全性一般

## 11.给出交叉编译的环境

在win10下通过vmware启动ubuntu虚拟机，并通过 `apt-get install gcc-arm-linux-gnueabi libncurses5-dev` 安装交叉编译环境

```
rookie@rookie-virtual-machine:~$ sudo apt-get install gcc-arm-linux-gnueabi libncurses5-dev
[sudo] rookie 的密码：
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
libncurses5-dev 已经是最新版 (6.0+20160213-1ubuntu1)。
gcc-arm-linux-gnueabi 已经是最新版 (4:5.3.1-1ubuntu1)。
下列软件包是自动安装的并且现在不需要了：
  binutils-aarch64-linux-gnu cpp-5-aarch64-linux-gnu cpp-aarch64-linux-gnu
  g++-4.8 gcc-5-aarch64-linux-gnu-base lib32asan0 lib32gcc-4.8-dev
  lib32stdc++-4.8-dev lib32stdc++-5-dev libasan2-arm64-cross
  libatomic1-arm64-cross libc6-arm64-cross libc6-dev-arm64-cross
  libgcc-5-dev-arm64-cross libgcc1-arm64-cross libgomp1-arm64-cross
  libitm1-arm64-cross libstdc++-4.8-dev libstdc++6-arm64-cross
  libubsan0-arm64-cross libx32asan0 libx32gcc-4.8-dev libx32stdc++-4.8-dev
  libx32stdc++-5-dev linux-libc-dev-arm64-cross
使用 'sudo apt autoremove' 来卸载它(它们)。
升级了 0 个软件包，新安装了 0 个软件包，要卸载 0 个软件包，有 330 个软件包未被升级。
rookie@rookie-virtual-machine:~$
```

## 12.给出交叉编译程序的情况，并证明是ARM/MIPS可执行文件

1. 编写浮点数运算程序

2. 交叉编译浮点数运算程序，并将a.out传输到树莓派上

```
rookie@rookie-virtual-machine:~$ cd 桌面
rookie@rookie-virtual-machine:~/桌面$ arm-linux-gnueabi-gcc test.c
rookie@rookie-virtual-machine:~/桌面$
```

3. 在树莓派上运行交叉编译程序

```
pi@raspberrypi:~/Downloads $ chmod 775 a.out
pi@raspberrypi:~/Downloads $ ./a.out
2.000000
pi@raspberrypi:~/Downloads $
```

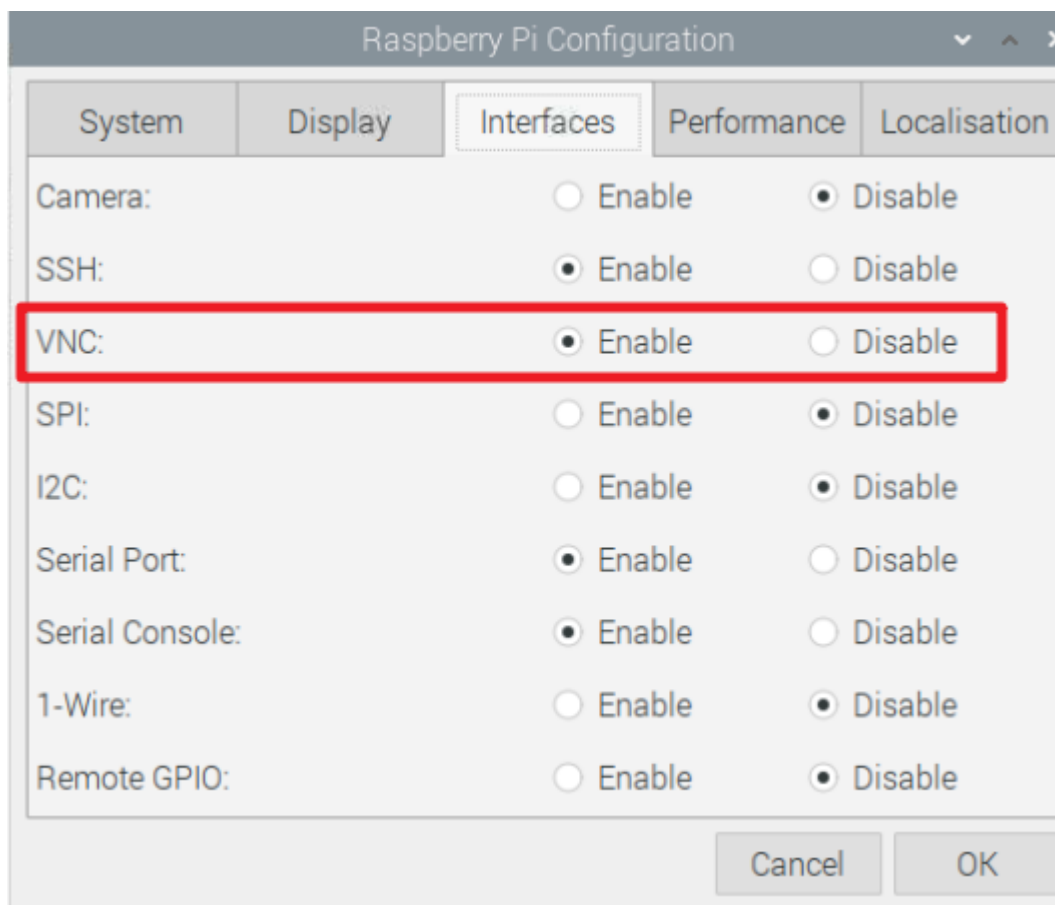
## 13.尝试远程桌面

尝试VNCviewer远程桌面，下载地址：<https://www.realvnc.com/en/connect/download/viewer/>

1. 树莓派端

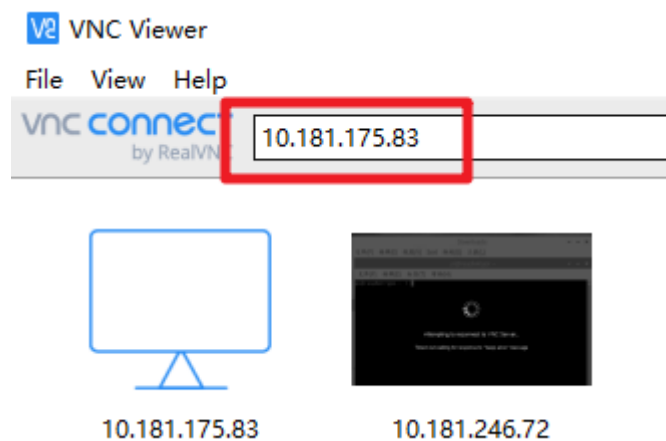
在configuration中开启VNC

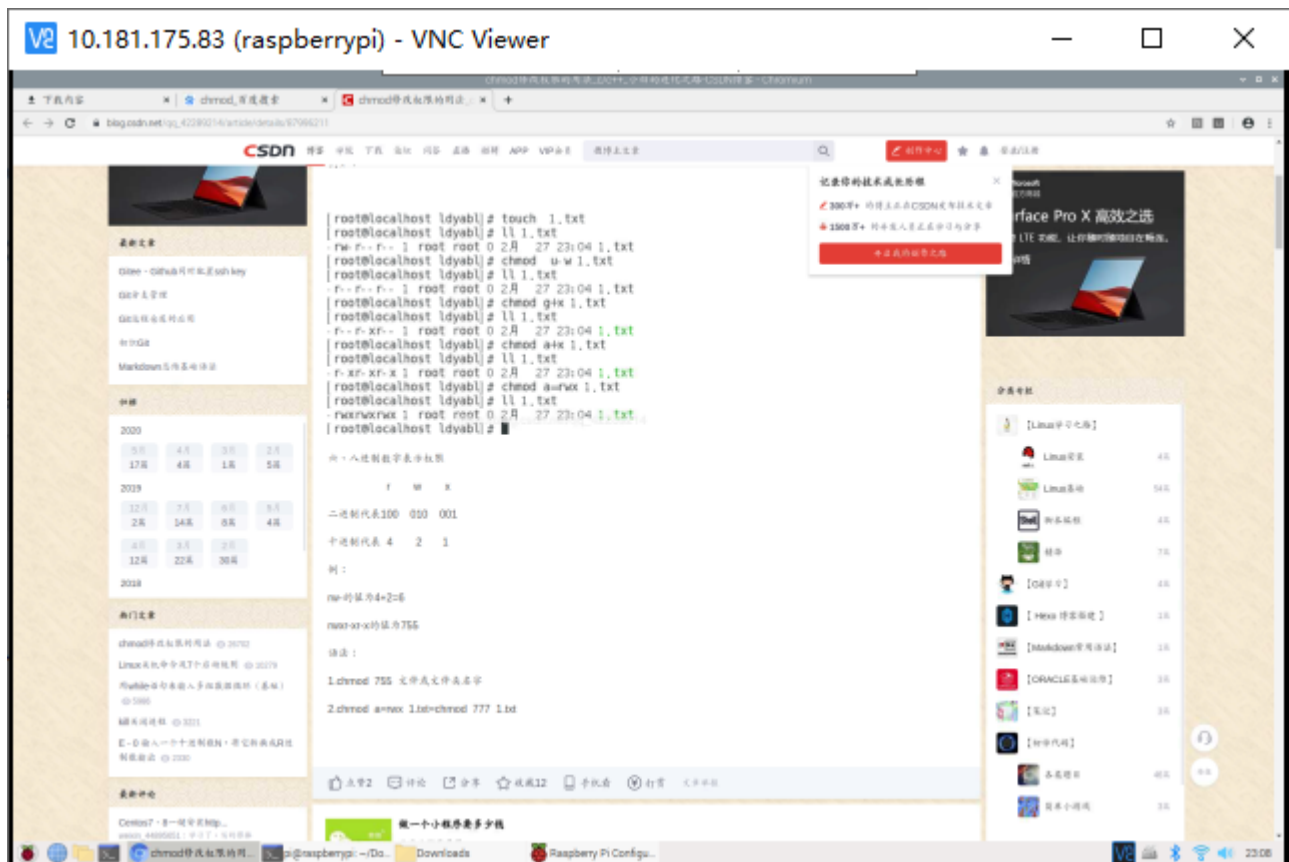




## 2. 电脑端

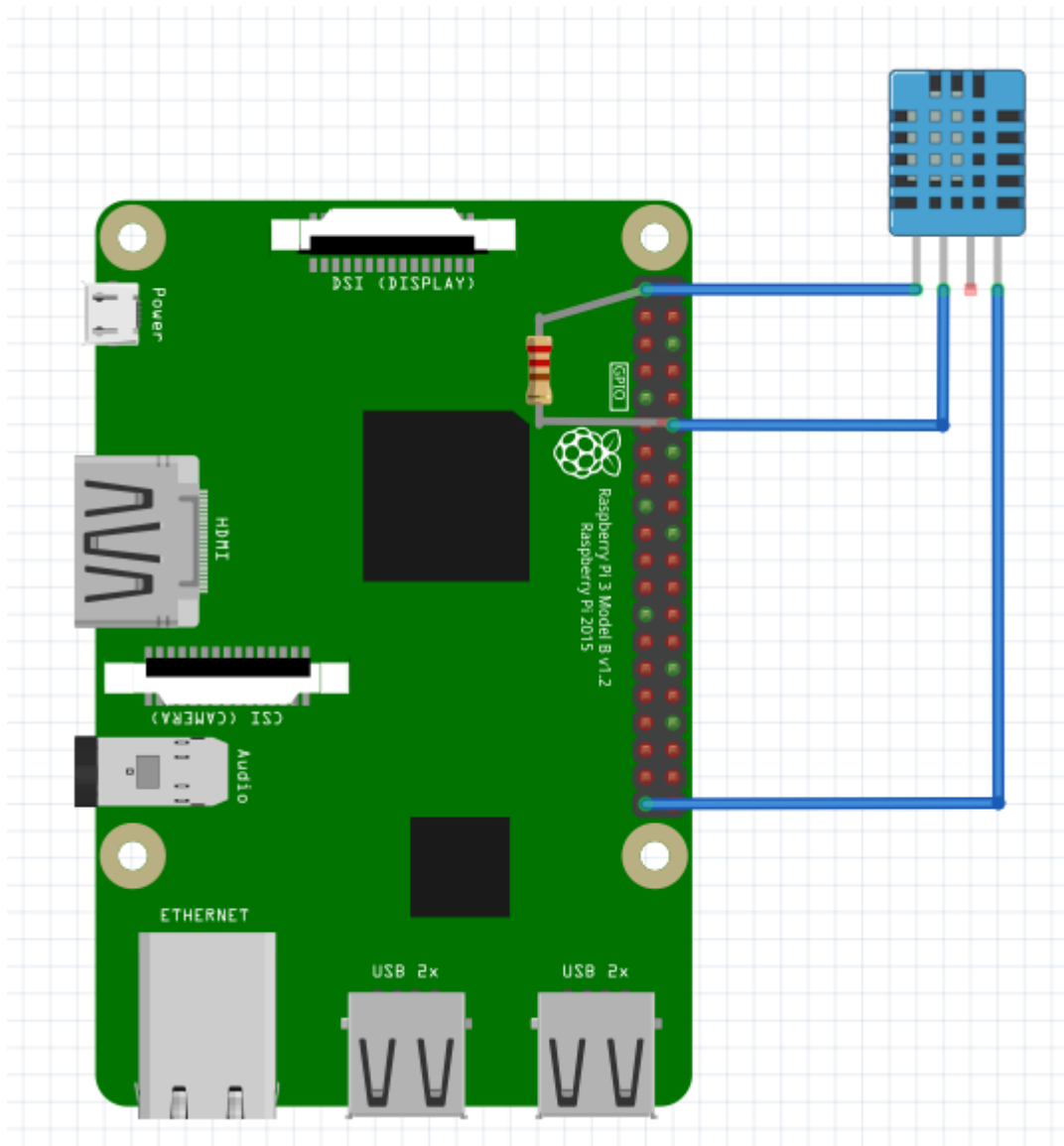
打开VNCviewer，并输入树莓派的ip，随后输入用户名和密码，与ssh连接时相同，即可访问远程桌面



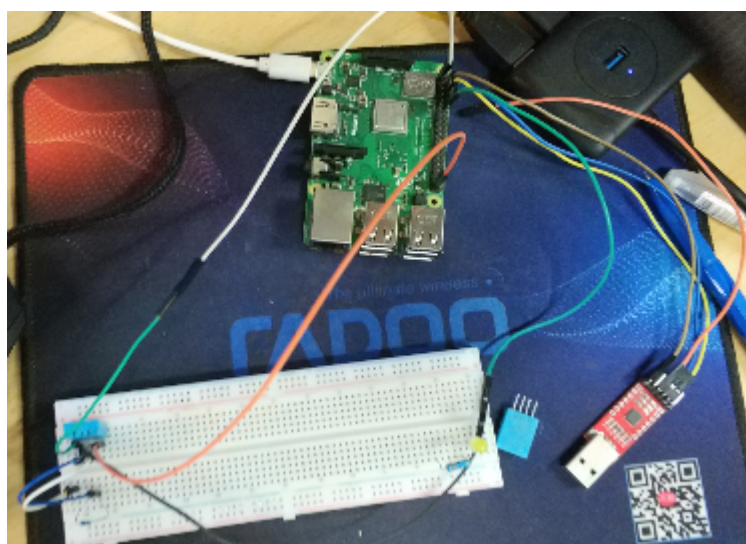


## 14.DHT11连接示意图

- 示意图



- 实际接线图



## 15.读取DHT11的代码

- 用python读取DHT11，代码及注释如下

```
1 import RPi.GPIO as GPIO
2 import time
3
4 CH = 18 # GPIO引脚18
5 GPIO.setmode(GPIO.BCM)
6
7 TIMEOUT_THRESH = 100
8 HIGH_LOW_THRESH = 8
9
10 def bit_to_num(arr):
11     num = 0
12     for (i, bit) in enumerate(arr):
13         num += bit * 2 ** (7-i)
14     return num
15
16 while 1:
17     time.sleep(1)
18     data = []
19     timeout_flag = 0
20     # 主机发开始信号
21     GPIO.setup(CH, GPIO.OUT)
22     GPIO.output(CH, GPIO.LOW)
23     time.sleep(0.025)
24     # 拉高
25     GPIO.output(CH, GPIO.HIGH)
26     GPIO.setup(CH, GPIO.IN)
27     # 等待DHT11拉低回复
28     cnt = 0
29     while GPIO.input(CH) == GPIO.HIGH:
30         cnt += 1
31         if cnt > TIMEOUT_THRESH:
32             timeout_flag = 1
33             break
34     # 等待DHT11拉高回复
35     cnt = 0
36     while GPIO.input(CH) == GPIO.LOW:
37         cnt += 1
38         if cnt > TIMEOUT_THRESH:
39             timeout_flag = 1
40             break
41     # 等待DHT11拉低回复
42     cnt = 0
43     while GPIO.input(CH) == GPIO.HIGH:
44         cnt += 1
45         if cnt > TIMEOUT_THRESH:
46             timeout_flag = 1
47             break
48     # 判断是否超时
49     if timeout_flag == 1:
50         print('RESET ERROR! TIMEOUT!')
51         continue
52     # 读取40bit的数据
53     timeout_flag = 0 # 超时标记
```

```

54     for i in range(40):
55         # 等待高电平数据到来
56         cnt = 0
57         while GPIO.input(CH) == GPIO.LOW:
58             cnt += 1
59             if cnt > TIMEOUT_THRESH:
60                 timeout_flag = 1
61                 break
62         if timeout_flag == 1:
63             break
64         # 等待高电平数据结束
65         cnt = 0
66         while GPIO.input(CH) == GPIO.HIGH:
67             cnt += 1
68             if cnt > TIMEOUT_THRESH:
69                 timeout_flag = 1
70                 break
71         if timeout_flag == 1:
72             break
73         # 判断是1或0
74         if cnt < HIGH_LOW_THRESH:
75             data.append(0)
76         else:
77             data.append(1)
78     if len(data) < 40:
79         print('READ ERROR! TIMEOUT!')
80         continue
81
82     humidity_integer = bit_to_num(data[0:8])
83     humidity_decimal = bit_to_num(data[8:16])
84     temperature_integer = bit_to_num(data[16:24])
85     temperature_decimal = bit_to_num(data[24:32])
86     checksum = bit_to_num(data[32:40])
87
88     if humidity_integer + humidity_decimal + temperature_integer +
temperature_decimal != checksum:
89         print('CHECK ERROR!')
90     else:
91         print('temp:', temperature_integer, 'humi:', humidity_integer)
92
93     GPIO.cleanup()

```

- 结果，成功率似乎比较低，要提升成功率还要进一步修改阈值，此外杜邦线不稳定也是影响成功率的一大因素

```
pi@raspberrypi:~/Desktop $ python3 LAB11.py
temp: 28 humi: 58
temp: 28 humi: 58
temp: 28 humi: 58
temp: 28 humi: 58
temp: 28 humi: 58
temp: 28 humi: 57
temp: 28 humi: 57
temp: 28 humi: 57
READ ERROR! TIMEOUT!
READ ERROR! TIMEOUT!
READ ERROR! TIMEOUT!
READ ERROR! TIMEOUT!
READ ERROR! TIMEOUT!
temp: 28 humi: 57
temp: 28 humi: 57
READ ERROR! TIMEOUT!
temp: 28 humi: 57
READ ERROR! TIMEOUT!
temp: 28 humi: 57
temp: 28 humi: 57
temp: 28 humi: 57
CHECK ERROR!
temp: 28 humi: 56
temp: 28 humi: 56
```