

基于树莓派的 fm 收音机

2023 年 12 月 25 日

1 实验平台

Raspberry Pi 3 Model B+

Linux RaspberryPI 6.2.0-v8+ #4 SMP PREEMPT Wed Mar 15 08:49:52 CST 2023 aarch64
GNU/Linux

Python 3.8.2

TEA5767

5641AS 4 位共阴极数码管（带有 i2c 通讯芯片）

面包板; 杜邦线若干; 3.5mm 耳机一个

2 理论知识

所需要的理论知识主要包括:linux 与 python 的使用, IIC 总线 (又称 i²c 或 i2c 总线) 的配置, 数码管的控制, TEA5767 模块的控制与使用方法。此部分重点介绍 i2c 总线的配置、通信方法, TEA5767 模块的控制原理、方法, 5641AS 数码管的显示方法。

2.1 树莓派与 i2c 总线

树莓派上有一排用于外设控制的针脚。需要用到 i2c-1 针脚。它的引脚映射如表1

将 5641AS 数码管,TEA5767 模块上的针脚与树莓派对应地连接起来, 即 VCC 接 5V,GND 接

| | | | | | |
|-------|------|---|---|-----|--|
| | 3.3V | 1 | 2 | 5V | |
| GPIO2 | SDA1 | 3 | 4 | 5V | |
| GPIO3 | SCL1 | 5 | 6 | GND | |

表 1: i2c 总线用到的针脚

GND,SDA 接 SDA1(GPIO2),SCL 接 SCL1(GPIO3)。为了便于接线, 可使用面包板。连接完成后将树莓派接电源 (通过 micro USB 接口从 USB-A 接口供电), 使用网线连接至电脑。亦可连接树莓派的 wifi 进行控制, ssid 为 RPI_XXX(XXX 为树莓派独有的编号, 是它的 ip 地址最后三位, 例如我使用的树莓派编号是 135, 它的 ip 地址就是 192.168.208.135。

控制方法为: 在主机命令行中输入:

```
ssh root@192.168.208.XXX
```

此时会要求输入密码（以我的为例）：

```
root@192.168.208.135's password: #默认密码是123456。
```

值得注意的是，将模块依序连接好之后还需安装驱动，才能正确识别。指令如下：

```
modprobe i2c_dev
modprobe i2c_bcm2835
```

使用这条指令查看各模块的连接状态：

```
i2cdetect -y 1
```

正确连接并识别的输出结果如下图所示。0x24-0x27 及 0x34-0x37 为数码管的地址，0x24 控制亮度与暗灭，0x60 为 FM 模块 TEA5767 的地址。

```

student@C418-23: ~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
student@C418-23:~$ ssh root@192.168.208.135
root@192.168.208.135's password:
#####
#                      Raspberry Pi3                      #
#                      Desktop                               #
#####
RPi3-aarch64
RaspberryPI:~ # modprobe i2c_dev
RaspberryPI:~ # modprobe i2c_bcm2835
RaspberryPI:~ # i2cdetect -y 1
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
10:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20:  -- -- -- -- 24 25 26 27 -- -- -- -- -- -- -- --
30:  -- -- -- -- 34 35 36 37 -- -- -- -- -- -- -- --
40:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: 60 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70:  -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
RaspberryPI:~ #

```

i2c 总线是由两根线组成的串行通信协议，使用多主从架构。数据通过 SDA Data 和 SCL Clock 传输，主要包括地址、数据、读写命令。

i2c 的通讯过程包括以下几个步骤：

1. 起始条件：主设备通过将 SDA 线从高电平切换到低电平，再将 SCL 线从高电平切换到低电平，来向每个连接的从机发送启动条件。
2. 发送从设备地址：主设备向每个从机发送要与之通信的从机的 7 位或 10 位地址，以及相应的读/写位。

3. 接收应答：每个从设备将主设备发送的地址与其自己的地址进行比较。如果地址匹配，则从设备通过将 SDA 线拉低一位以表示返回一个 ACK 位。如果来自主设备的地址与从机自身的地址不匹配，则从设备将 SDA 线拉高，表示返回一个 NACK 位。

4. 收发数据：主设备发送或接收数据到从设备。

5. 接收应答：在传输完每个数据帧后，接收设备将另一个 ACK 位返回给发送方，以确认已成功接收到该帧。

6. 停止通信：为了停止数据传输，主设备将 SCL 切换为高电平，然后再将 SDA 切换为高电平，从而向从机发送停止条件 1。

通信的过程如图1所示：

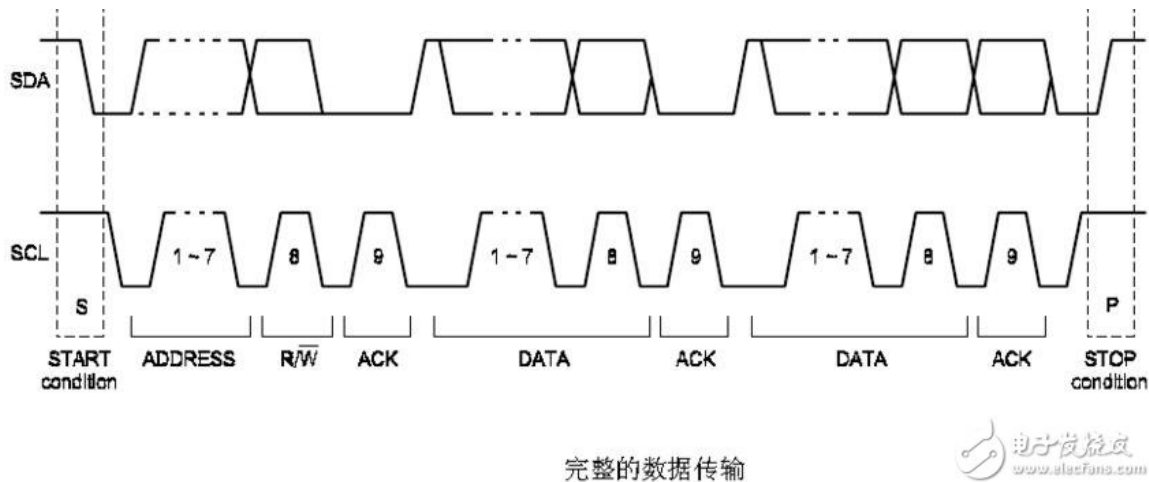


图 1: i2c 通信过程

使用 python 中的 smbus2(System Management Bus) 模块可以方便地使用 i2c 协议进行通讯。这里给出一个例子：

```
from smbus2 import SMBus, i2c_msg
bus = SMBus(1)
address = 0x24 #某个地址
length = 2 #读2字节的数据
block_of_bytes = [0x11,0x22,0x33,0x44,0x55]
read = i2c_msg.read(address,length)
bus.write_byte(address,byte)#写入1字节
write = i2c_msg.write(address,block_of_bytes)#连续写入几个字节
bus.i2c_rdwr(write)
bus.i2c_rdwr(write,read)
```

2.2 TEA5767 模块的使用

TEA5767 是一个 FM 收音模块，它通过 i2c 总线与控制主机通信。它可以一次向寄存器写入 5 个字节，每个字节的每一位都有含义。如表2所示。

更详细的介绍见于[TEA5767 document](#)

| BIT | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 |
|-----|-------------|-------|--------------|------------------|-------------|
| 7 | MUTE | PLL7 | search UP/DN | SW Port2 | PLL ref |
| 6 | search mode | PLL6 | stop level1 | stand-by | De-emphasis |
| 5 | PLL13 | PLL5 | stop level0 | band limit | - |
| 4 | PLL12 | PLL4 | HiLo | xtal | - |
| 3 | PLL11 | PLL3 | mono/stereo | soft-mute | - |
| 2 | PLL10 | PLL2 | mute left | HCC | - |
| 1 | PLL9 | PLL1 | mute right | SNC | - |
| 0 | PLL8 | PLL0 | SW port1 | search indicator | - |

表 2: TEA5767 Registor Function

其中，PLL 为配置锁相环的 14 位 2 进制数。它的计算方法如下：

$$PLL = \frac{4 \times (F_{RF} + F_{IF})}{F_{REF}} \quad (1)$$

F_{RF} is the wanted tuning frequency in Hz 即电台的频率, F_{IF} is the intermediate frequency of 225kHz 即间隔频率, F_{REF} 为基准频率，由 xtal 和 PLL ref 共同控制。

计算出 PLL 后，再将它分割成一个 6 位 2 进制数和一个 8 位二进制数，再结合是否需要自动搜台（自动搜台则 Byte1 前两位为 01，否则为 00）即得到前两个寄存器的值。此处给出一个后三位寄存器取值的例子：

```
Byte3=0b00111000
Byte4=0b00010111
Byte5=0b00000000
```

Byte3 第一位为 0 表示 search down；随后两位为 01 表示 search stop level 为 low，之后一位为 1 表示 High side LO injection，有助于减少噪声；mono/stereo=1 表示强制单声道，随后几位均为 0，左声道和右声道都不静音。

Byte4[3],Byte5[0]=1,0 表示基准频率设定为 32768Hz。这两位的功能是设定基准频率，从 13MHz, 6.5MHz, 32.768kHz 中选取。其余位对收音效果影响不大，[TEA5767 document](#) 的第 14 页起有详细的介绍。

2.3 数码管的使用

5641AS 是共阴极 4 位 7 段数码管，带有一个 i2c 控制芯片实现通信、串转并、译码。正如之前提到的，它的地址是 0x24-0x27 及 0x34-0x37。0x34-0x37 控制数码管的每一位；0x24 控制整个数

码管的亮度与暗灭。它的功能如表3所示。[B2,B1,B0] 组成的 3 位 2 进制数为亮度 (Brightness), 最后一位 ON/OFF 控制亮/灭. 因此在程序的一开始要设定它为 0x41。

```
bus.write_byte(0x24,0x41)
```

| | | | | | | |
|---|----|----|----|-----|--|--------|
| X | B2 | B1 | B0 | 7/8 | | ON/OFF |
|---|----|----|----|-----|--|--------|

表 3: 0x24 Function

0x25-0x27 没有控制数码管的功能，它们的作用是控制其他基于 i2c 总线的外设。
0x34-0x37 分别控制数码管每一位的显示，编码方式服从共阴极数码管的一般规则，即：

| | | | | | | | | | | |
|-------|------|------|------|------|------|------|------|------|------|------|
| 显示数字 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 寄存器取值 | 0x3F | 0x06 | 0x5B | 0x4F | 0x66 | 0x6D | 0x7D | 0x07 | 0x7F | 0x6F |

表 4: 编码表

可以将编码表、地址存入数组中：

```
lut=[0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f]  
addr=[0x34,0x35,0x36,0x37]
```

3 使用 python 编写程序实现 fm 收音机功能

正如之前介绍的，TEA5767 模块具有 i2c 通信功能，可以通过 i2c 总线设定调谐频率、自动搜索功能、搜台方向（向上或向下），stop level, high/low side injection。编写的程序应当具有这些功能：

- 1. 显示目前的调谐频率（即电台频率）。
- 2. 可以自动搜台。搜台方向由输入决定。
- 3. 具备一定的自动控制能力。模块本身的自动搜索能力较有限，但可以通过多次搜索实现更宽的频率搜索范围。

实现这些功能需要 3 个输入：调谐频率；自动搜台 flag；自动搜台方向。

使用两个模块实现：setnum（设定显示的数值）和 setfreq(控制 TEA5767)。将程序封装成模块的好处是便于代码复用，并便于相互调用。

3.1 频率显示功能

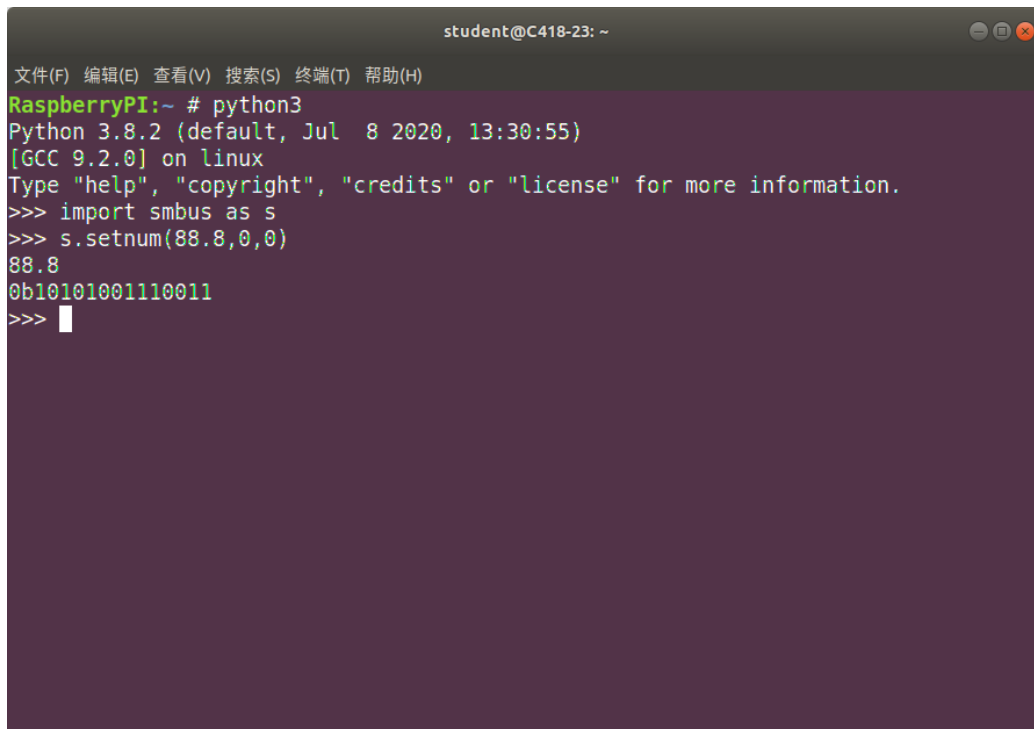
从外部传入三个参数:f,auto,direc 分别表示频率、自动搜台、搜台方向. 先将浮点数 f 分为整数部分和小数部分，再通过 i2c 总线控制某 address 的数码管显示相应数字，并在个位显示小数点。最后，调用频率设定模块。

查阅[资料](#)可知，显示小数点的方法是给当前编码加上 0x80.

参考代码如下:

```
import time
from smbus2 import SMBus, i2c_msg
bus = SMBus(1)
def setnum(f,auto,direc):
    bus.write_byte(0x24,0x41)
    s=f
    n1=int(s)
    n2=round((s-float(n1))*10) % 10
    res = s
    for i in range(3):
        bus.write_byte(addr[2-i],lut[n1 % 10])
        #向对应地址写入频率的每一位数字
        if i ==0 :
            bus.write_byte(addr[2],0x80+lut[n1 % 10])#个位还需显示小数点
            n1 = n1 //10
    bus.write_byte(addr[3],lut[n2])
    print(res) #便于调试
    setfreq(res,auto,direc)
```

成功运行的结果:



```
student@C418-23: ~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
RaspberryPI:~ # python3
Python 3.8.2 (default, Jul 8 2020, 13:30:55)
[GCC 9.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import smbus as s
>>> s.setnum(88.8,0,0)
88.8
0b10101001110011
>>> 
```

3.2 频率设定模块

此部分的功能主要包括：将输入的频率 f 转换为配置锁相环的 14 位 2 进制数 (PLL) 依据输入的搜台模式 (sm)，搜台方向 (direction)，设置 Byte1-Byte5 中对应某位的值 (0/1) 配置其余位，包括晶振、stop level 等，使收音机能正常收台，并且输出信噪比尽可能高（设置 HiLo=1 可显著地减少噪声）。

配置 TEA5767 模块的方式是向它的地址一次性写入 5 字节 (block of bytes)。

参考代码如下：

```
def setfreq(f,sm,direction):
    cof=32768
    freq=f
    freq14bit = int (4*(freq * 1000000 + 225000) / cof)
    freqH = freq14bit >> 8
    freqL = freq14bit & 0xff
    print(bin(freq14bit))#将f转换为PLL并输出
    up=direction << 7
    if sm :searchflag=1 << 6
    else: searchflag=0
    data=[0 for i in range(5)]
    data[0]=freqH & 0x3f | searchflag
    #位运算，若searchflag为1则data[0]的次高位为1；否则为0
    data[1]=freqL
    data[2]=0b00111000 | up
    #位运算，若direction为1则data[2]的最高位为1；否则为0
    data[1]=freqL
    data[3]=0b00010111
    data[4]=0b00000000
    try:
        write=i2c_msg.write(0x60,data)#prepare write block of bytes
        bus.i2c_rdwr(write)#finish write
    except:
        pass
```

3.3 显示自动搜索完成的频率

显然，自动搜索完成的频率与一开始输入的频率是不同的。为了使数码管显示最终的频率，应当从 TEA5767 的地址中读出对应寄存器的值，转换为浮点数后调用 setnum 再次显示。

值得注意的是，Python 多次使用 i2c 总线通讯会占用树莓派的较多资源，造成卡顿。因此，在两次通讯之间暂停 0.5s 是合理的，也便于观察自动搜索的过程。暂停 0.5s 需要使用到 time 模块中的 time.sleep() 函数。参考代码如下：

```

time.sleep(0.5)
read = i2c_msg.read(0x60,2)#prepare to read
bus.i2c_rdwr(read)#read 2 bytes
s = read.__bytes__()#转换为字节
pll = int.from_bytes(s, byteorder = 'big', signed =
    False)#转换为int类型
pll = pll & 0x3fff#最高2位不是PLL, 置为0
frequency =float(( pll * cof / 4 - 225000) /
    1000000)#转换为频率 (浮点数)
setnum(frequency,sm,direction)#显示

```

3.4 自动控制功能

TEA5767 本身具有自动搜台功能, 但搜索范围有限, 会出现搜台一次仍然搜不到台的情况。为了改善这个情况, 可以在 setfreq 的最后调用 setnum, 在 setnum 中设置递推的出口, 即搜索 2 次后便不再反复搜索。只需要对 setnum 和 setfreq 程序做一些小小的更改, 以 sm 为计数器, 每次调用 setnum 前加 1, setnum 中判断它大于 3 时停止相互调用即可。

整体修改完的程序如下:

```

import time
from smbus2 import SMBus, i2c_msg
bus = SMBus(1)
lut=[0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f]
addr=[0x34,0x35,0x36,0x37]
def setnum(f,auto,direc):
    ...
    if auto < 3 : setfreq(res,auto,direc)
    # auto>=3 means that auto search process is finished, just need
    to display the fatal frequency
def setfreq(f,sm,direction):
    ...
    if sm or direction :
        sm += 1
        setnum(frequency,sm,direction)
        print(sm,direction)

```

4 功能测试

使用时先在保存程序 (我给它命名为 smbus.py) 的文件夹打开终端, 输入


```
python3
>>> import smbus as s
```

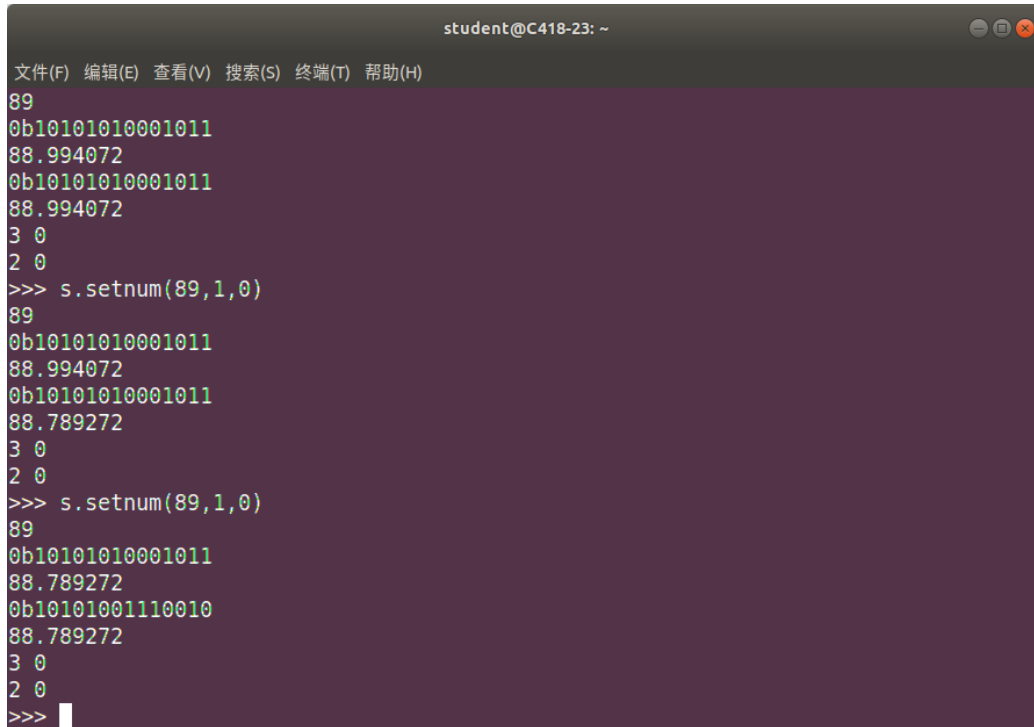
控制:

```
>>> s.setnum(<frequency>,<search_mode>,<search_direction>)
```

以下是一些成功运行的例子:

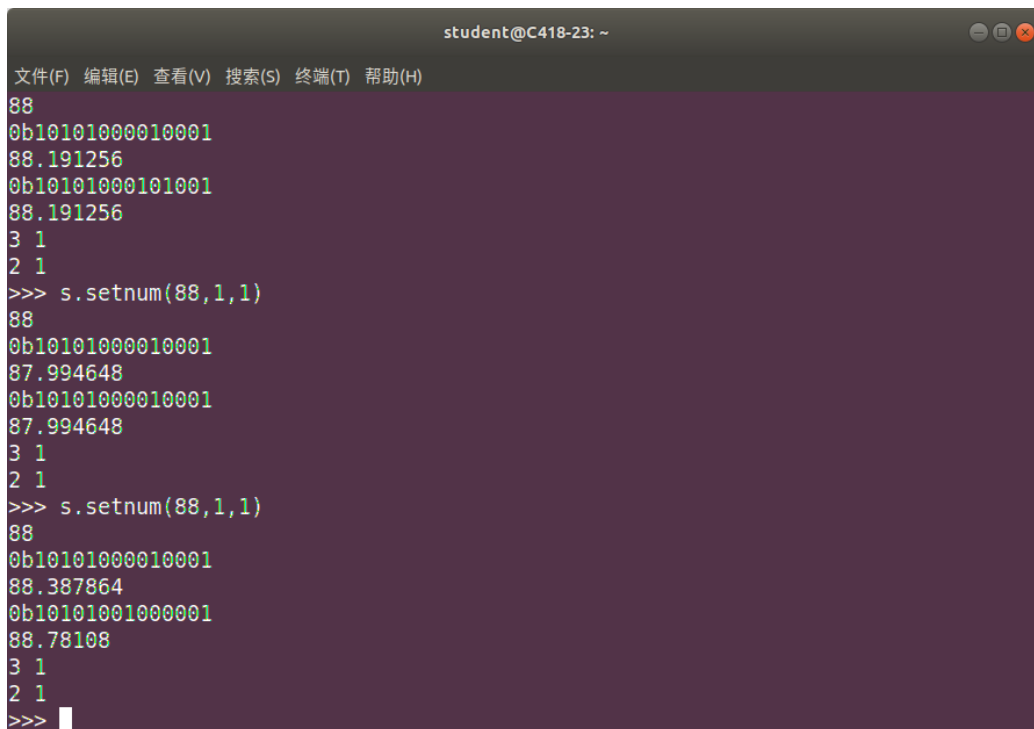


图 2: 直接设定频率



```
student@C418-23: ~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
89  
0b10101010001011  
88.994072  
0b10101010001011  
88.994072  
3 0  
2 0  
>>> s.setnum(89,1,0)  
89  
0b10101010001011  
88.994072  
0b10101010001011  
88.789272  
3 0  
2 0  
>>> s.setnum(89,1,0)  
89  
0b10101010001011  
88.789272  
0b10101001110010  
88.789272  
3 0  
2 0  
>>>
```

图 3: 向下自动搜台



```
student@C418-23: ~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
88  
0b10101000010001  
88.191256  
0b10101000101001  
88.191256  
3 1  
2 1  
>>> s.setnum(88,1,1)  
88  
0b10101000010001  
87.994648  
0b10101000010001  
87.994648  
3 1  
2 1  
>>> s.setnum(88,1,1)  
88  
0b10101000010001  
88.387864  
0b10101001000001  
88.78108  
3 1  
2 1  
>>>
```

图 4: 向上自动搜台