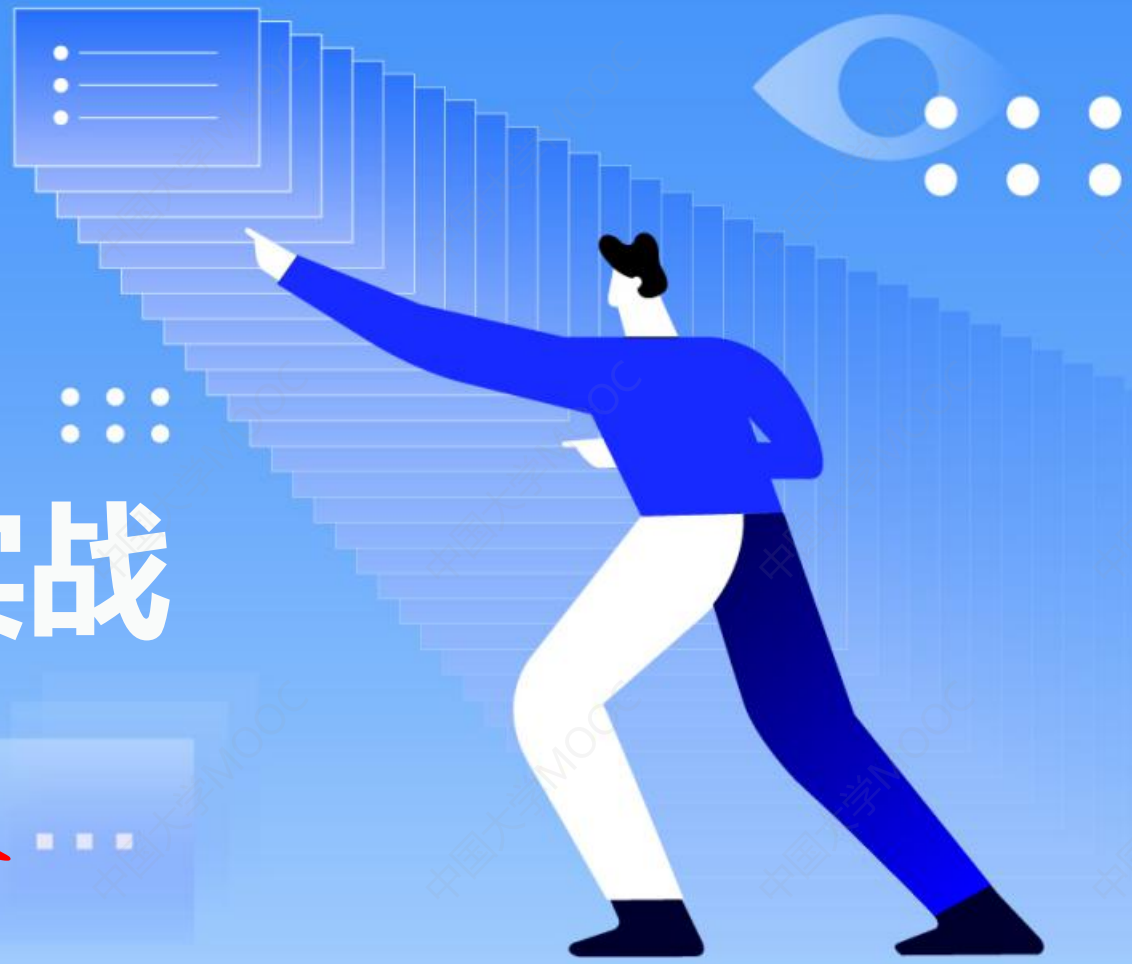
电子技术实习

——树莓派实战

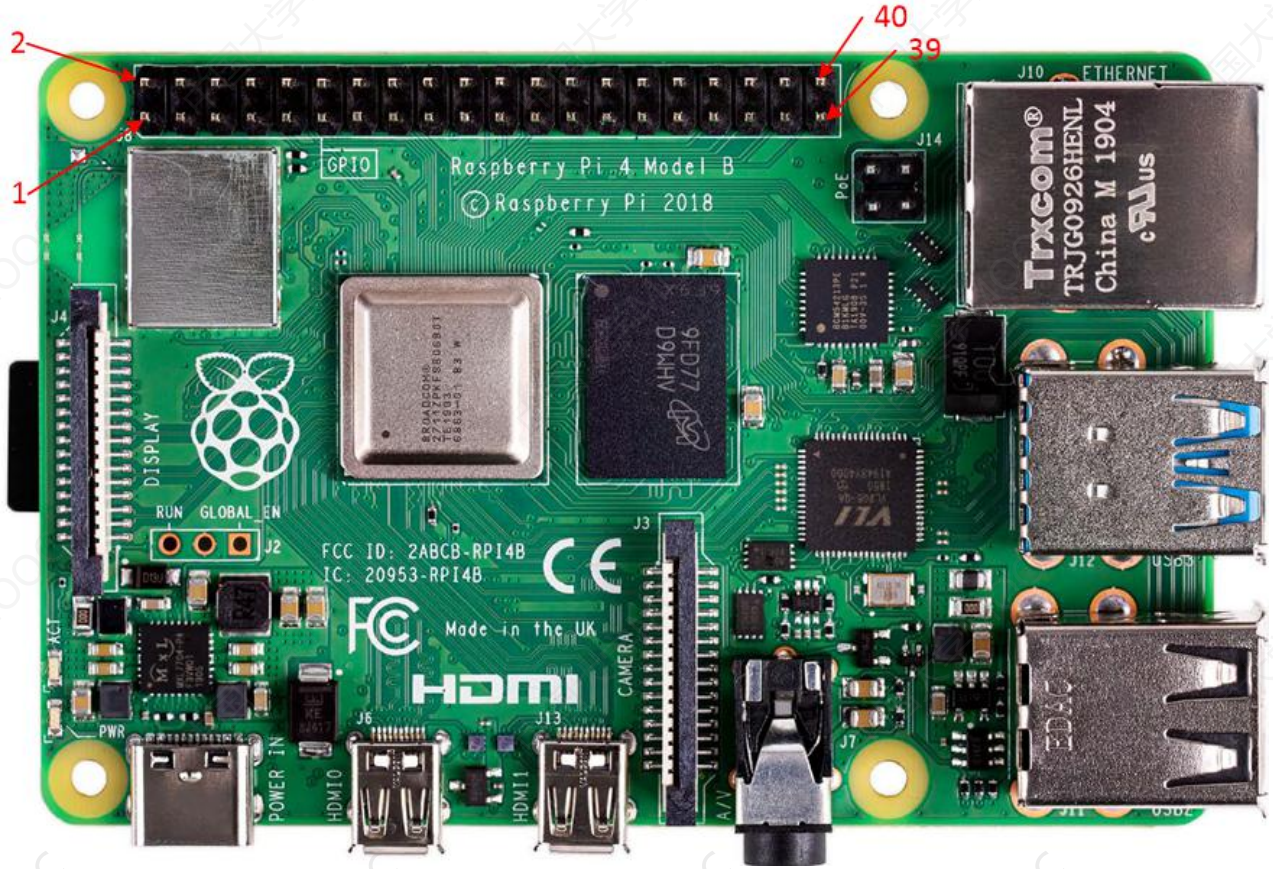


PWM波形及双色LED实验

RaspberryPi

中国大学MOOC

1、树莓派 Pi 4B介绍 - GPIO



树莓派 40Pin 引脚对照表

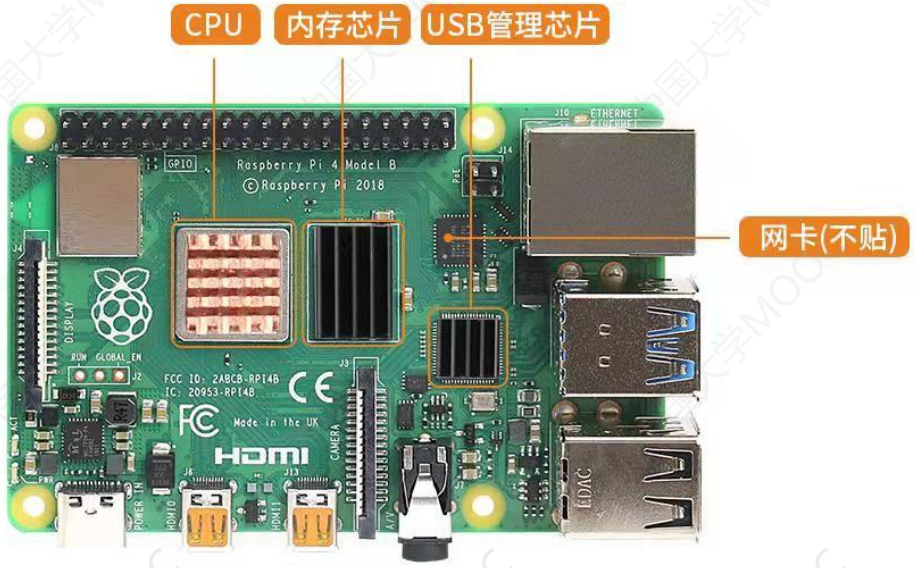
wiringPi 编码	BCM 编码	功能名	物理引脚 BOARD编码		功能名	BCM 编码	wiringPi 编码
		3.3V	1	2	5V		
8	2	SDA.1	3	4	5V		
9	3	SCL.1	5	6	GND		
7	4	GPIO.7	7	8	TXD	14	15
		GND	9	10	RXD	15	16
0	17	GPIO.0	11	12	GPIO.1	18	1
2	27	GPIO.2	13	14	GND		
3	22	GPIO.3	15	16	GPIO.4	23	4
		3.3V	17	18	GPIO.5	24	5
12	10	MOSI	19	20	GND		
13	9	MISO	21	22	GPIO.6	25	6
14	11	SCLK	23	24	CE0	8	10
		GND	25	26	CE1	7	11
30	0	SDA.0	27	28	SCL.0	1	31
21	5	GPIO.21	29	30	GND		
22	6	GPIO.22	31	32	GPIO.26	12	26
23	13	GPIO.23	33	34	GND		
24	19	GPIO.24	35	36	GPIO.27	16	27
25	26	GPIO.25	37	38	GPIO.28	20	28
		GND	39	40	GPIO.29	21	29

1、树莓派 Pi 4B介绍 - 散热片

北京科技大学 University of Science and Technology Beijing

散热片粘贴详解

新引入的高速USB3.0管理芯片的发热量不低, 同时与CPU、内存较为接近, 因此建议粘贴散热片为整体散热. 树莓派4B相比3B+，网卡芯片发热量有所减少，尺寸更小, 因周围元件较多, 担心造成短路, 所以不建议网卡使用散热片.



树莓派 40Pin 引脚对照表

wiringPi 编码	BCM 编码	功能名	物理引脚 BOARD编码		功能名	BCM 编码	wiringPi 编码
		3.3V	1	2	5V		
8	2	SDA.1	3	4	5V		
9	3	SCL.1	5	6	GND		
7	4	GPIO.7	7	8	TXD	14	15
		GND	9	10	RXD	15	16
0	17	GPIO.0	11	12	GPIO.1	18	1
2	27	GPIO.2	13	14	GND		
3	22	GPIO.3	15	16	GPIO.4	23	4
		3.3V	17	18	GPIO.5	24	5
12	10	MOSI	19	20	GND		
13	9	MISO	21	22	GPIO.6	25	6
14	11	SCLK	23	24	CE0	8	10
		GND	25	26	CE1	7	11
30	0	SDA.0	27	28	SCL.0	1	31
21	5	GPIO.21	29	30	GND		
22	6	GPIO.22	31	32	GPIO.26	12	26
23	13	GPIO.23	33	34	GND		
24	19	GPIO.24	35	36	GPIO.27	16	27
25	26	GPIO.25	37	38	GPIO.28	20	28
		GND	39	40	GPIO.29	21	29

1、树莓派 Pi 4B介绍 - GPIO

北京科技大学 University of Science and Technology Beijing

GPIO基本和扩展功能（BOARD编址）

板载接口编号：GPIO

1、通用IO ：所有的IO都可以作为GPIO使用

2、串口 ：8、10

3、SPI口 ：19、21、23、24、26

4、I2C口 ：3、5、27、28

5、1-Wire接口 ：所有GPIO可以成单总线使用。

wiringPi编码中 7

树莓派 40Pin 引脚对照表

wiringPi 编码	BCM 编码	功能名	物理引脚 BOARD编码		功能名	BCM 编码	wiringPi 编码
		3.3V	1	2	5V		
8	2	SDA.1	3	4	5V		
9	3	SCL.1	5	6	GND		
7	4	GPIO.7	7	8	TXD	14	15
		GND	9	10	RXD	15	16
0	17	GPIO.0	11	12	GPIO.1	18	1
2	27	GPIO.2	13	14	GND		
3	22	GPIO.3	15	16	GPIO.4	23	4
		3.3V	17	18	GPIO.5	24	5
12	10	MOSI	19	20	GND		
13	9	MISO	21	22	GPIO.6	25	6
14	11	SCLK	23	24	CE0	8	10
		GND	25	26	CE1	7	11
30	0	SDA.0	27	28	SCL.0	1	31
21	5	GPIO.21	29	30	GND		
22	6	GPIO.22	31	32	GPIO.26	12	26
23	13	GPIO.23	33	34	GND		
24	19	GPIO.24	35	36	GPIO.27	16	27
25	26	GPIO.25	37	38	GPIO.28	20	28
		GND	39	40	GPIO.29	21	29

1、树莓派 Pi 4B介绍 - GPIO

北京科技大学 University of Science and Technology Beijing

GPIO基本功能：

通用IO：GPIO 通用型之输入输出的简称，其接脚可以供使用者由程控自由使用，PIN脚依现实考量可作为通用输入（GPI）或通用输出（GPO）或通用输入与输出（GPIO）。一个引脚可以用于输入、输出或其他特殊功能，有专用寄存器用来选择这些功能。

1、输入，可以通过读取某个寄存器来确定引脚电位的高低；

2、输出，可以通过写入某个寄存器来让这个引脚输出高电位或者低电位；

3、其他特殊功能，另外的寄存器来控制它们。

树莓派 40Pin 引脚对照表

wiringPi 编码	BCM 编码	功能名	物理引脚 BOARD编码		功能名	BCM 编码	wiringPi 编码
		3.3V	1	2	5V		
8	2	SDA.1	3	4	5V		
9	3	SCL.1	5	6	GND		
7	4	GPIO.7	7	8	TXD	14	15
		GND	9	10	RXD	15	16
0	17	GPIO.0	11	12	GPIO.1	18	1
2	27	GPIO.2	13	14	GND		
3	22	GPIO.3	15	16	GPIO.4	23	4
		3.3V	17	18	GPIO.5	24	5
12	10	MOSI	19	20	GND		
13	9	MISO	21	22	GPIO.6	25	6
14	11	SCLK	23	24	CE0	8	10
		GND	25	26	CE1	7	11
30	0	SDA.0	27	28	SCL.0	1	31
21	5	GPIO.21	29	30	GND		
22	6	GPIO.22	31	32	GPIO.26	12	26
23	13	GPIO.23	33	34	GND		
24	19	GPIO.24	35	36	GPIO.27	16	27
25	26	GPIO.25	37	38	GPIO.28	20	28
		GND	39	40	GPIO.29	21	29

2、树莓派 Pi 4B介绍 - GPIO输出

北京科技大学 University of Science and Technology Beijing

输出，可以通过写入某个寄存器来让这个引脚输出高电位或者低电位

TTL电平：

输出：0：＜0.4V　　输入：0：≤0.8V

1：＞2.4V　　1：≥2.0V

树莓派对应输入和输出

1：高电平 - 3.3V

0：低电平 - 0V

树莓派 40Pin 引脚对照表

wiringPi 编码	BCM 编码	功能名	物理引脚 BOARD编码		功能名	BCM 编码	wiringPi 编码
		3.3V	1	2	5V		
8	2	SDA.1	3	4	5V		
9	3	SCL.1	5	6	GND		
7	4	GPIO.7	7	8	TXD	14	15
		GND	9	10	RXD	15	16
0	17	GPIO.0	11	12	GPIO.1	18	1
2	27	GPIO.2	13	14	GND		
3	22	GPIO.3	15	16	GPIO.4	23	4
		3.3V	17	18	GPIO.5	24	5
12	10	MOSI	19	20	GND		
13	9	MISO	21	22	GPIO.6	25	6
14	11	SCLK	23	24	CE0	8	10
		GND	25	26	CE1	7	11
30	0	SDA.0	27	28	SCL.0	1	31
21	5	GPIO.21	29	30	GND		
22	6	GPIO.22	31	32	GPIO.26	12	26
23	13	GPIO.23	33	34	GND		
24	19	GPIO.24	35	36	GPIO.27	16	27
25	26	GPIO.25	37	38	GPIO.28	20	28
		GND	39	40	GPIO.29	21	29

2、GPIO输出 – PWM及参数

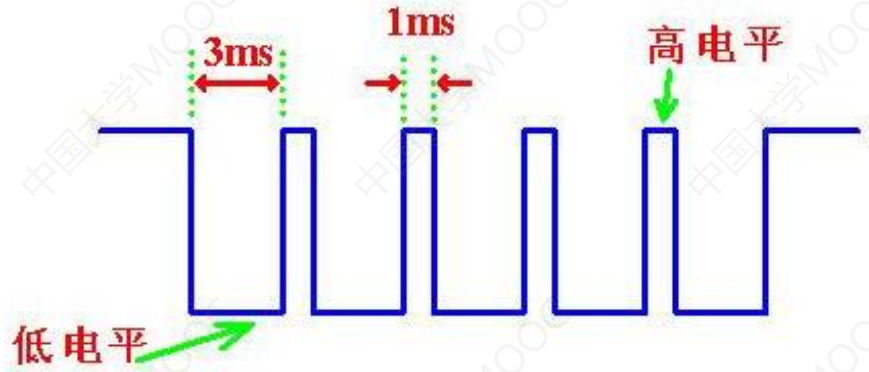
PWM——Pulse Width Modulation 脉冲宽度调制

脉冲——波形

宽度——占空比

调制——可调

输出高电平一段时间后，输出低电平，周而复始。



PWM参数：

频　率：每一秒钟多少个周期

占空比：每个周期内，高电平占的百分比

占空比可以调，就实现了PWM的调制

频　率：1s/4ms=250Hz

占空比：1ms/4ms=25%

2、GPIO输出 – PWM作用

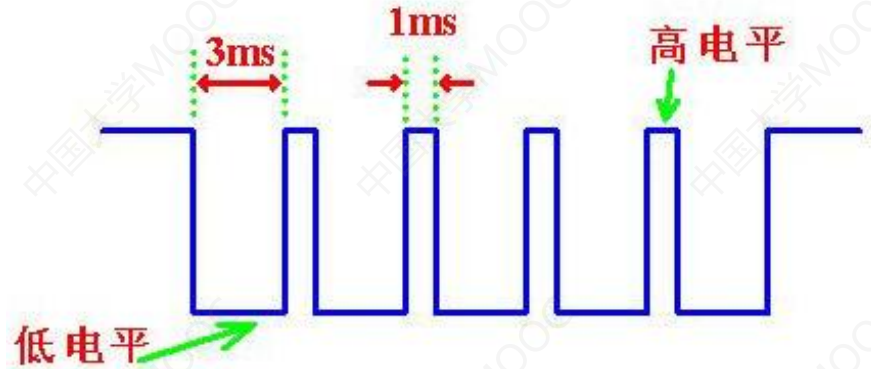
占空比——计算平均电压

3.3V*25%=0.825V

如果是周期为4秒，占空比为25%：

将这样的电压控制一个LED会是什么效果呢？

微观效果显示：亮一秒，灭3秒



频　率：1s/4ms=250Hz

占空比：1ms/4ms=25%

如果周期变短，频率加大到250Hz，占空比为25%：

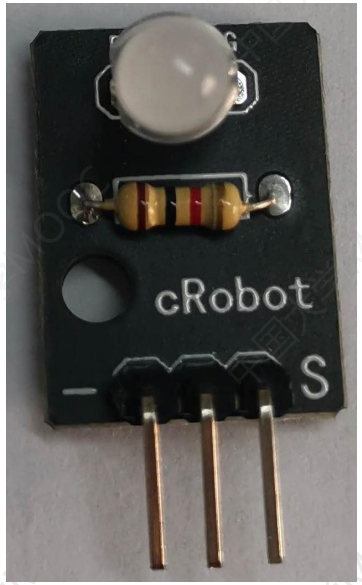
LED又会是什么效果呢？

宏观效果显示：亮度变暗【通过占空比可以调节LED的亮度】

3、双色LED实验-实验内容

1、模块介绍

双色LED灯，又名双基色LED灯，是指模块只能显示2种颜色，一般是红色和绿色，可以有三种状态：灭、颜色1亮和颜色2亮。根据颜色组合的不同，分为红蓝双色、黄蓝双色、红绿双色等。



2、实验组件

（1）Raspberry Pi主板*1

（2）树莓派电源适配器*1

（3）40P软排线*1

（4）双色LED模块*1

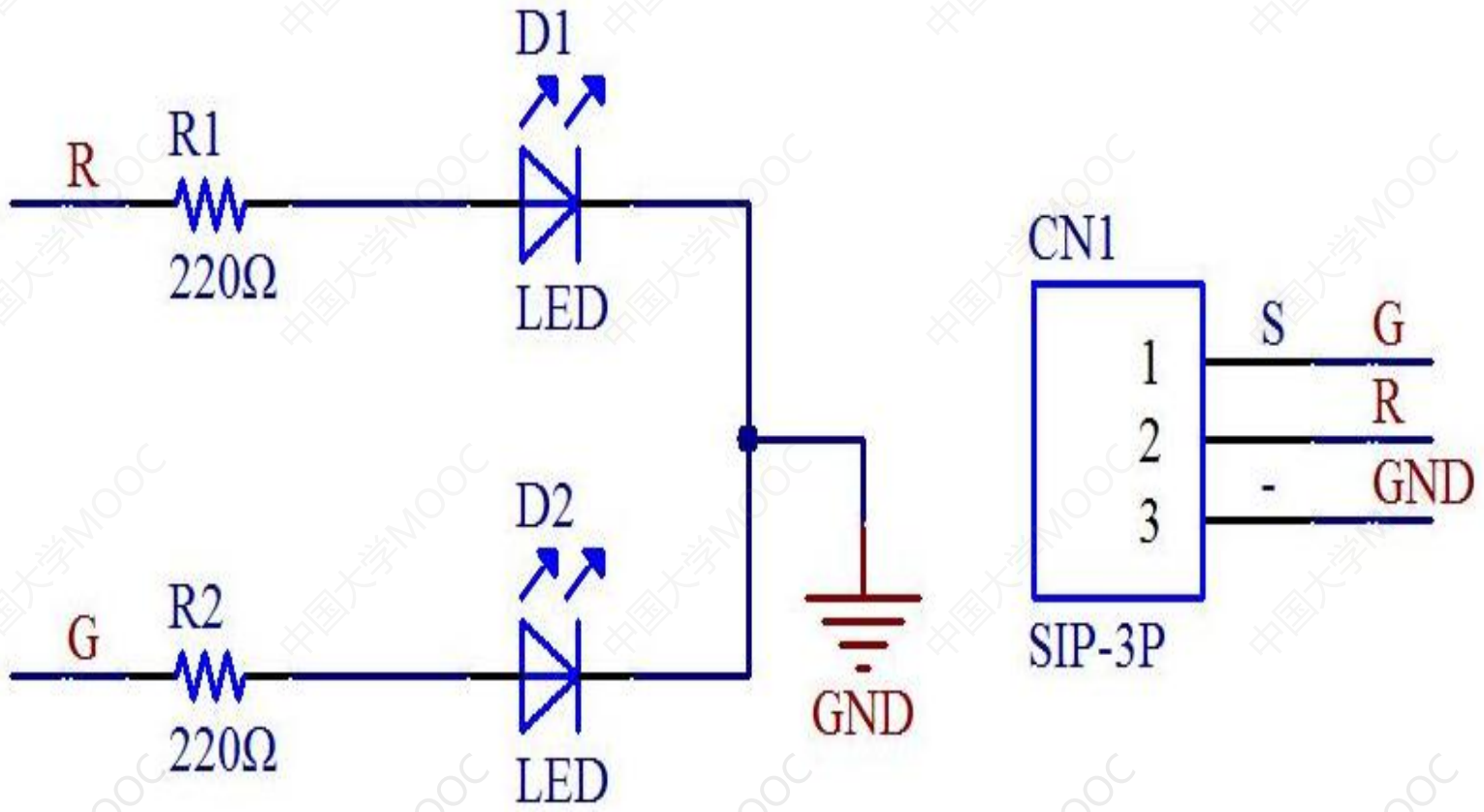
（5）面包板*1

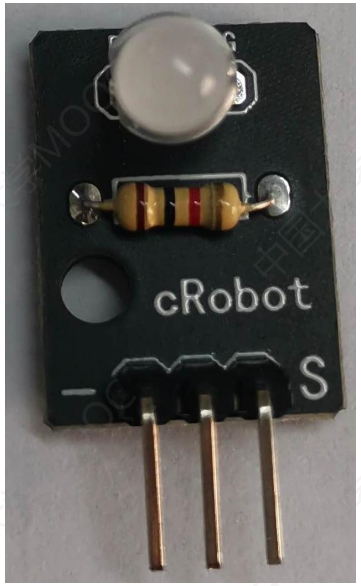
（6）跳线若干

3、双色LED实验-实验原理

北京科技大学 University of Science and Technology Beijing

双色LED模块电路原理图

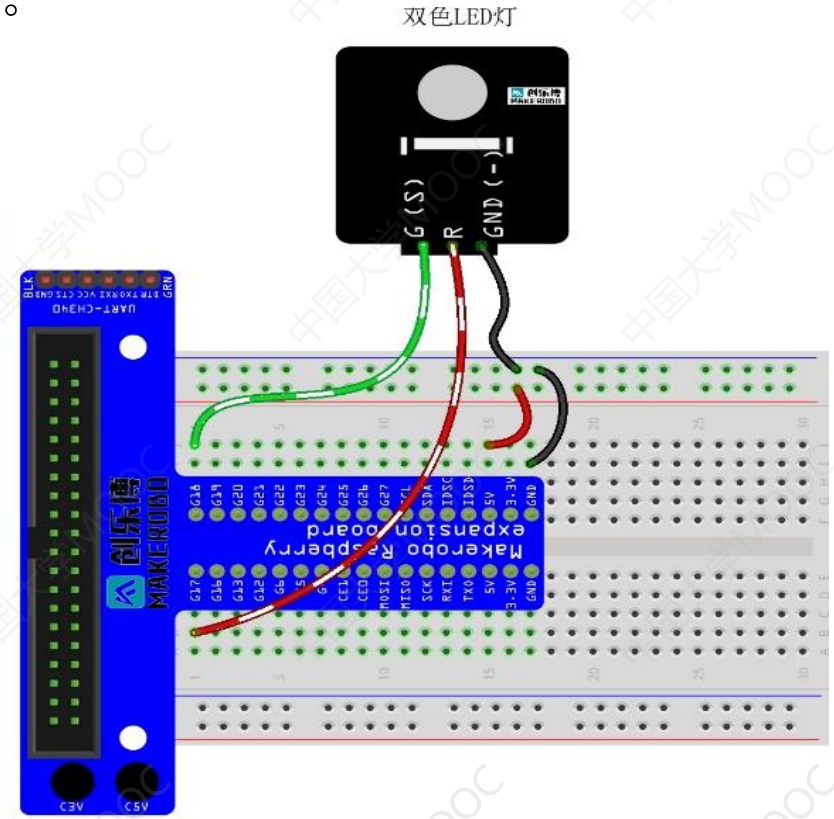




3、双色LED实验-实验要求

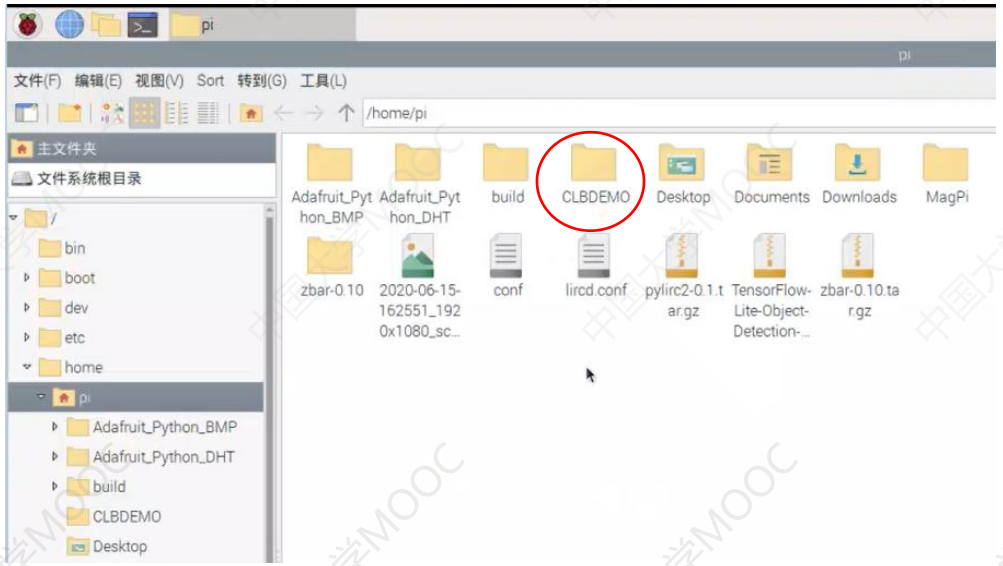
将引脚G（绿色）和中间管脚R（红色）连接 Raspberry Pi的GPIO接口上，对Raspberry Pi进行编程控制，将LED的颜色从红色变为绿色，然后使用PWM混合成其他颜色。

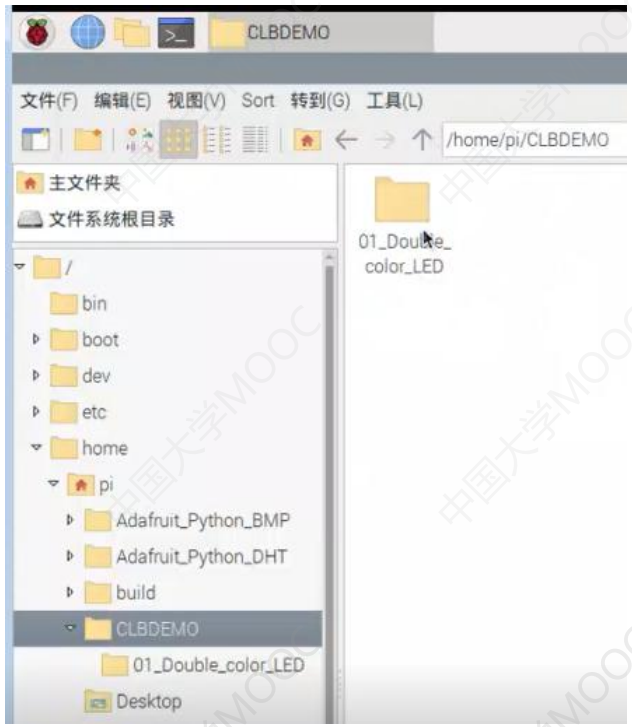
树莓派	T型转接板	双色LED模块
GPIO1	GPIO18	G（S）
GPIO0	GPIO17	R（中间）
GND	GND	GND



3、双色LED实验-实验步骤

1. 建立电路；
2. Mu IDE软件编写程序；
3. 运行代码，观察并记录实验现象；
4. 运行完成后，使用Ctrl+C退出程序。





注意：

运行完成后一定要退出程序运行，否则该程序会一直在后台运行，从而干扰其他程序运行！

3、双色LED实验- Python代码

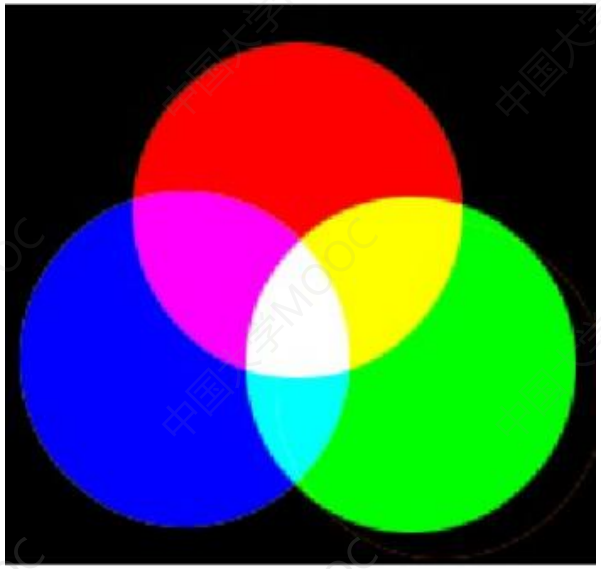

```
import RPi.GPIO as GPIO
import time

colors = [0xFF00, 0x00FF, 0x0FF0, 0xF00F] # 颜色列表
```

补充：RGB色彩模式

RGB色彩模式是工业界的一种颜色标准，是通过对红(R)、绿(G)、蓝(B)三个颜色通道的变化以及它们相互之间的叠加来得到各种颜色。

在电脑中，RGB各有256级亮度，用数字表示为从0、1、2...直到255。注意虽然数字最高是255，但0也是数值之一，0表示没有刺激量，255表示刺激量达最大值。R、G、B均为255时就合成了白光，R、G、B均为0时就形成了黑色。



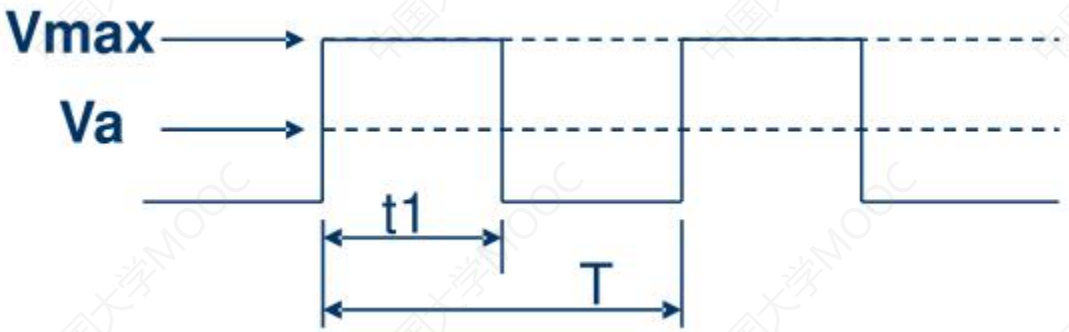
中国大学MOOC

3、双色LED实验- Python代码


```
makerobo_pins = (11, 12)          # PIN管脚字典
GPIO.setmode(GPIO.BOARD)  # 采用实际的物理管脚给GPIO口
GPIO.setwarnings(False)           # 去除GPIO口警告
GPIO.setup(makerobo_pins, GPIO.OUT)       # 设置Pin模式为输出模式
GPIO.output(makerobo_pins, GPIO.LOW)      # 设置Pin管脚为低电平(0V)关闭LED
p_R = GPIO.PWM(makerobo_pins[0], 2000)   # 设置频率为2KHz
p_G = GPIO.PWM(makerobo_pins[1], 2000)   # 设置频率为2KHz
```

补充：占空比

占空比：是指一串理想脉冲序列中，正脉冲的持续时间与脉冲总周期的比值。调整LED通过电流和不通过电流的时间比来控制，由于人眼有视觉暂留特性，所以只要频率比较高是看不出来闪烁的。当然通过电流比不通过电流的时间比例越大，LED也就越亮。



$$D=(t1/T)\times 100\%$$

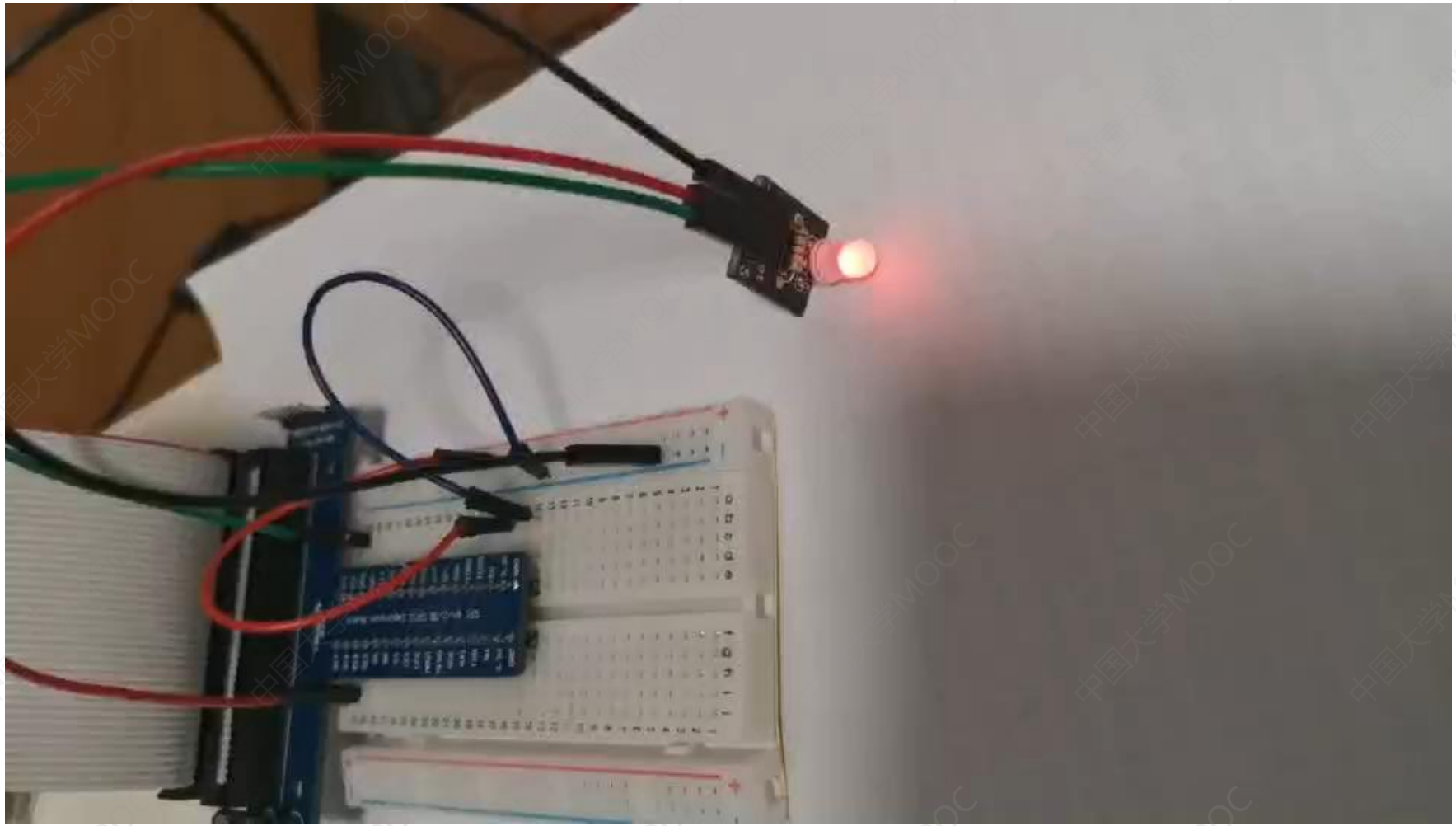
3、双色LED实验- Python代码


```
# 初始化占空比为0(led关闭)
p_R.start(0)
p_G.start(0)
def makerobo_pwm_map(x, in_min, in_max, out_min, out_max):
        return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min
def makerobo_set_Color(col):
        R_val = col  >> 8
        G_val = col & 0x00FF
        # 把0-255的范围同比例缩小到0-100之间
        R_val = makerobo_pwm_map(R_val, 0, 255, 0, 100)
        G_val = makerobo_pwm_map(G_val, 0, 255, 0, 100)
        p_R.ChangeDutyCycle(R_val)     # 改变占空比
        p_G.ChangeDutyCycle(G_val)     # 改变占空比
# 调用循环函数
def makerobo_loop():
         while True:
                 for col in colors:
```

3、双色LED实验- Python代码


```
                    makerobo_set_Color(col)
                    time.sleep(0.5)
# 释放资源
def makerobo_destroy():
            p_G.stop()
            p_R.stop()
            GPIO.output(makerobo_pins, GPIO.LOW)   # 关闭所有LED
            GPIO.cleanup()                         # 释放资源
# 程序入口
if __name__ == "__main__":
        try:
                makerobo_loop()       # 调用循环函数
        except KeyboardInterrupt:     # 当按下Ctrl+C时，将执行destroy()子程序。
                makerobo_destroy()    # 释放资源
```

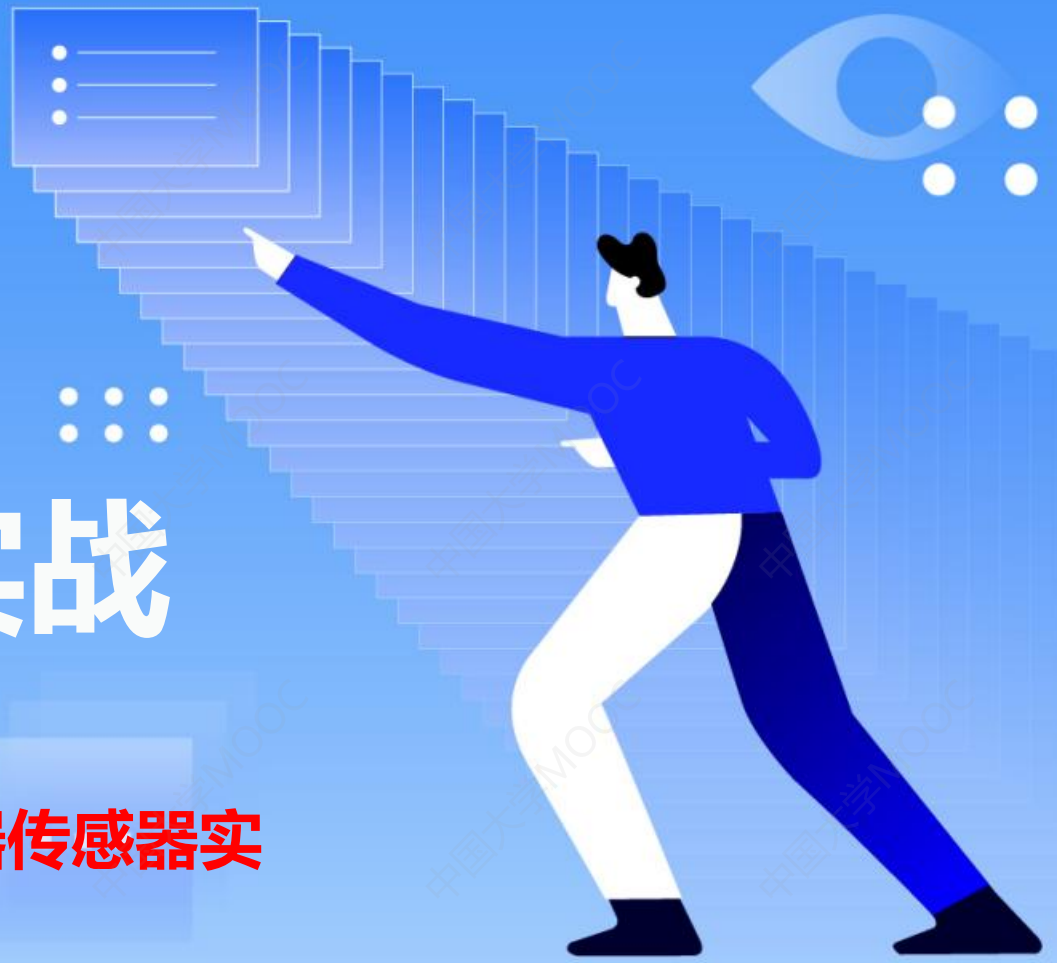
3、双色LED实验-实验效果



中国大学MOOC

电子技术实习

——树莓派实战



实验12 PCF8591模数转换器传感器实验

中国大学MOOC

实验内容

北京科技大学 University of Science and Technology Beijing

PCF8591简介：

PCF8591是一款单芯片，单电源，低功耗8位CMOS数据采集设备 。

- 四个模拟输入（AIN0, AIN1, AIN2, AIN3）
- 一个模拟输出（AOUT）
- 一个串行I²C总线接口（SDA,SLC）。
- 三个地址A0,A1和A2用于对硬件地址进行编程（允许使用多达8个连接到I²C总线的设备）在PCF8591器件上输入输出的地址、控制和数据信号都是通过双线双向I²C总线以串行方式进行传输

SYMBOL	PIN	DESCRIPTION
AINO	1	analog inputs (A/D converter)
AIN1	2	
AIN2	3	
AIN3	4	
A0	5	hardware address
A1	6	
A2	7	
V_{SS}	8	negative supply voltage
SDA	9	I²C-bus data input/output
SCL	10	I²C-bus clock input
OSC	11	oscillator input/output
EXT	12	external/internal switch for oscillator input
AGND	13	analog ground
V_{REF}	14	voltage reference input
AOUT	15	analog output (D/A converter)
V_{DD}	16	positive supply voltage

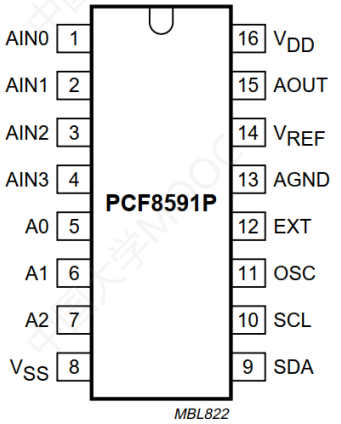


Fig.2 Pinning diagram (DIP16).

A/D/A转换 PCF8591是AIN端口输入模拟电压，然后PCF8591将转换后的数字量通过I²C总线发送给单片机；或是单片机通过I²C总线给一个数字量，然后PCF8591通过AOUT端口将模拟电压输出。

实验内容

PCF8591：1、器件地址

在I²C总线协议中，在启动条件之后的第一个字节。每一个I²C器件都有一个器件地址，来区分不同的I²C设备，下面是PCF8591的地址:

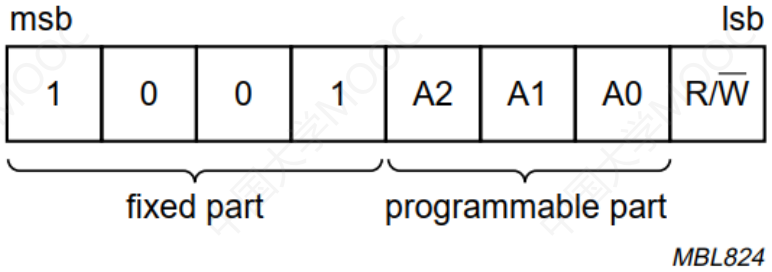
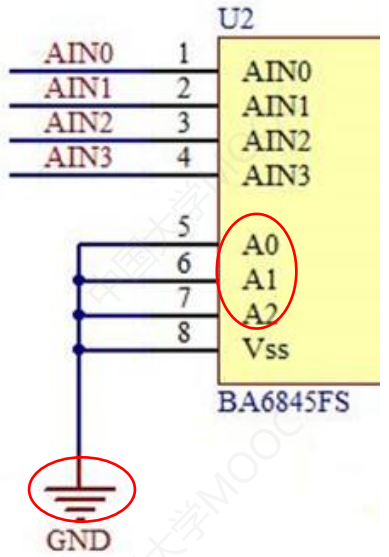


Fig.4 Address byte.

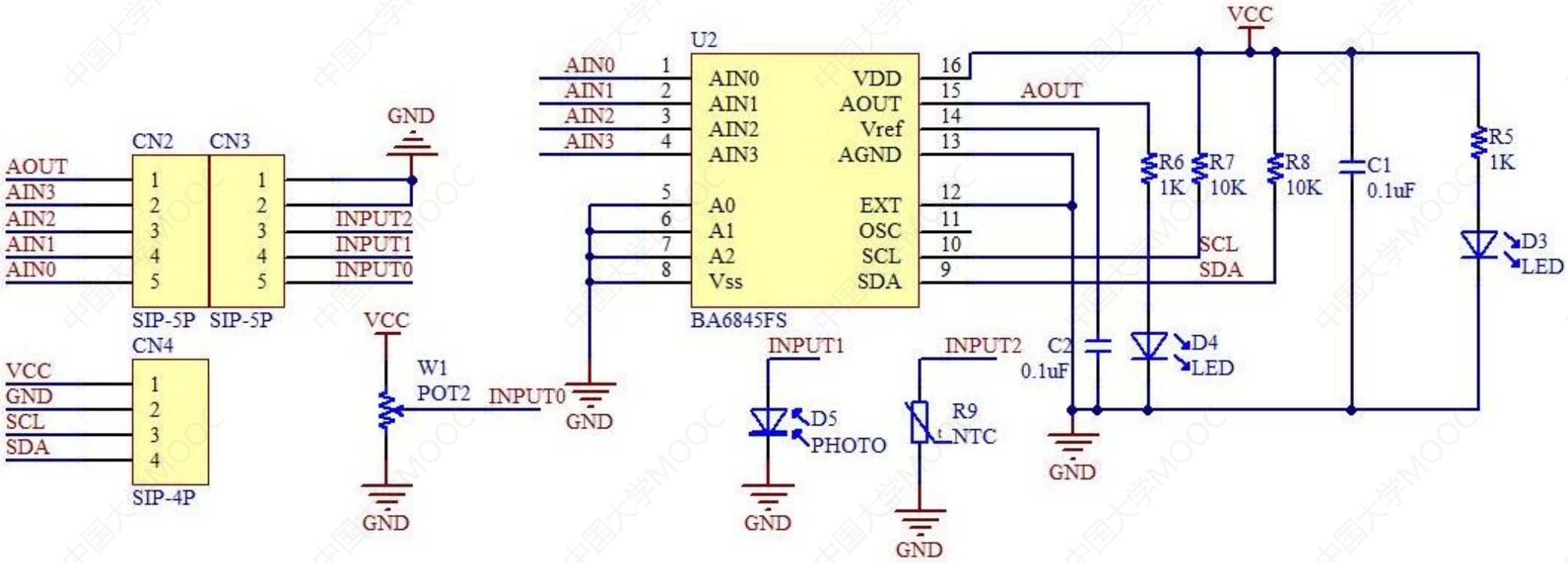


- 它的地址是由1001（固定部分）和A2A1A0（可编程部分）组成的，可编程部分必须按照地址引脚A0,A1,A2进行设置。
- 地址字节的最后一位是读/写，它设置了数据传输的方式：0表示下一个字节往总线上写数据， 1 表示下一个字节从总线上读取数据。
- 在原理图中可以看出，A2A1A0均为0，所以7位器件地址（1001000）为0x48。

实验原理

北京科技大学 University of Science and Technology Beijing

PCF8591模数转换器模块电路原理图：

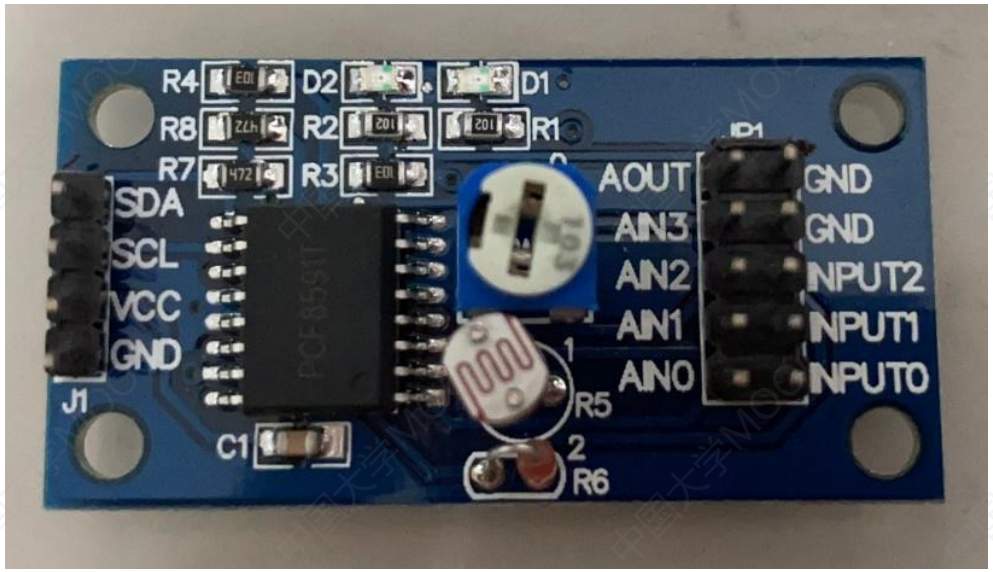


本实验中，AIN0（模拟输入0）端口用于接收来自电位计模块的模拟信号，AOUT（模拟输出）用于将模拟信号输出到双色LED模块，以便改变LED的亮度。

实验内容

PCF8591模数转换器（AD/DA转换器）传感器模块：

该模块右侧有一排跳线帽，例：AIN0处插上跳线帽，相当于把模块中的电位器接入电路中；另外，AIN1接入光敏电阻，AIN2接入热敏电阻。左侧为通讯总线，为IIC总线通讯接口。

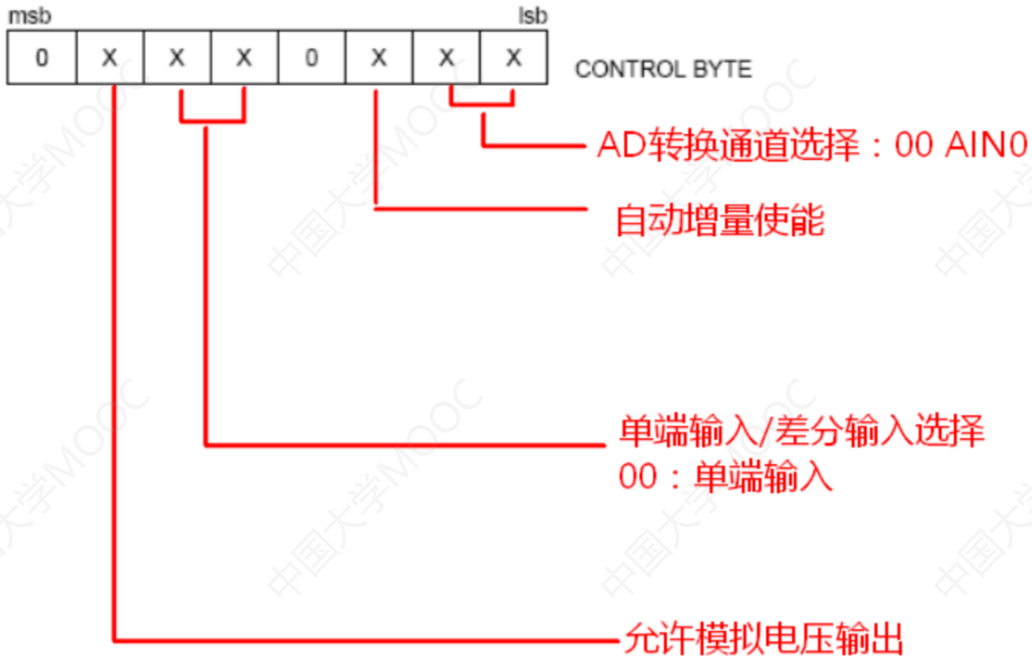


PCF8591器件的地址、控制和数据通过两线双向I2C总线传输。器件功能包括多路复用模拟输入、片上跟踪和保持功能、8位模数转换和8位数模拟转换。最大转换速率取决于I2C总线的最高速率。

实验内容

PCF8591：2、控制字格式

发送到PCF8591的第二个字节将被存储在其控制寄存器中，并且需要控制器件功能。



- 最高位默认为 0；
- 第 6 位是选择是否允许模拟电压输出，在DA转换时设置为1，AD转换时设置为0或1均可；
- 第5/4位是选择模拟电压输出方式，一般选择00单端输入方式，其他几种方式可以查阅技术手册；
- 第3位默认为 0；
- 第2位是自动增量使能位，如果自动增量（auto-increment）标志置1，每次A/D 转换后通道号将自动增加；
- 第1/0位是在AD转换时选择哪一个通道输入的电压转换为数字量；

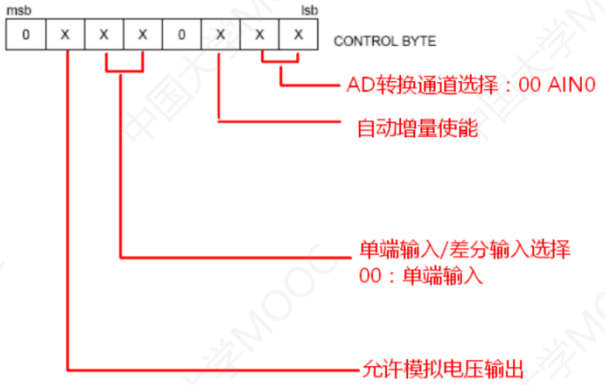
实验内容

PCF8591：4、AD转换应用开发

- AD的位数：表明这个AD共有2^n个刻度，8位AD，输出的刻度是0~255。 8591就是8位精度的，因此它digtal Read的数据在0-255之间。

- 分辨率：就是AD能够分辨的最小的模拟量变化。假设5.10V的系统用8位的AD采样，那么它能分辨的最小电压就是5.10/255=0.02V。

PCF8591模块代码：




```
# 说明：这是一个PCF8591模块的程序。
#       警告：模拟输入不能超过3.3V!
# 在这个程序中，我们使用电位计进行
# 的LED灯，你可以导入这个程序到另一
# import PCF8591 as ADC
# ADC.Setup(Address)  # 通过 sudo i
# ADC.read(channal) # 通道选择范围为0-3
# ADC.write(Value)  # 值的范围为：0-255
##########################################################
import smbus
import time

# 对应比较旧的版本如RPI V1 版本，则 "bus = smbus.SMBus(0)"
bus = smbus.SMBus(1)

#通过 sudo i2cdetect -y -1 可以获取到IIC的地址
def setup(Addr):
    global address
    address = Addr

# 读取模拟量信息
def read(chn): #通道选择，范围是0-3之间
    try:
        if chn == 0:
            bus.write_byte(address,0x40)#0100 0000
        if chn == 1:
            bus.write_byte(address,0x41)#0100 0001
        if chn == 2:
            bus.write_byte(address,0x42)#0100 0010
        if chn == 3:
            bus.write_byte(address,0x43)#0100 0011
        bus.read_byte(address) # 开始进行读取转换
    except Exception as e:
        print ("Address: %s" % address)
        print (e)
    return bus.read_byte(address)
```

实验内容

PCF8591：4、AD转换应用开发

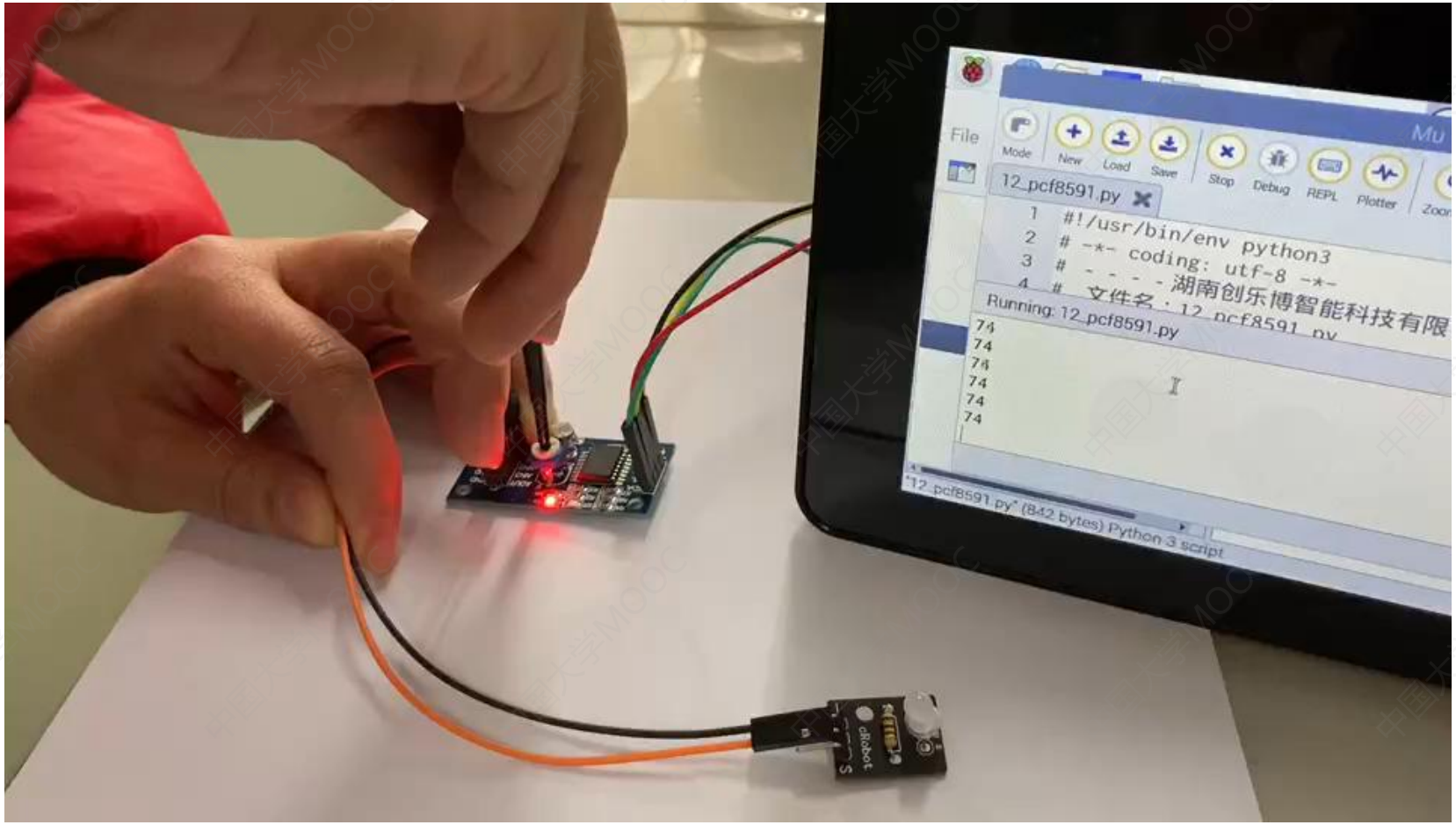
- AD的位数：表明这个AD共有2^n个刻度，8位AD，输出的刻度是0~255。 8591就是8位精度的，因此它digtal Read的数据在0-255之间。

- 分辨率：就是AD能够分辨的最小的模拟量变化。假设5.10V的系统用8位的AD采样，那么它能分辨的最小电压就是5.10/255=0.02V。

PCF8591模块代码：

```
43
44  # 模块输出模拟量控制，范围为0-255
45  def write(val):
46      try:
47          temp = val # 将数值赋给temmp 变量
48          temp = int(temp) # 将字符串转换为整型
49          # 在终端上打印temp以查看，否则将注释掉
50          bus.write_byte_data(address, 0x40, temp)
51      except Exception as e:
52          print ("Error: Device address: 0x%2X" % address)
53          print (e)
54
55  if __name__ == "__main__":
56      setup(0x48)
57      while True:
58          print ('AIN0 = ', read(0))
59          print ('AIN1 = ', read(1))
60          tmp = read(0)
61          tmp = tmp*(255-125)/255+125 # 低于125时LED不会亮，所以请将"0-255"转换为"125-2
62          write(tmp)
63  #       time.sleep(0.3)
64
```

实验效果


File
Mode
New
Load
Save
Stop
Debug
REPL
Plotter
12_pcf8591.py
#!/usr/bin/env python3
-*- coding: utf-8 -*-
Running: 12_pcf8591.py
74
74
74
74
74
74
"12_pcf8591.py" (842 bytes) Python 3 script