EE351 微机原理与微系统 期中实验报告

12110748 李璐

lab1：树莓派的安装、远程登录和基础知识学习

1234567890ertyuixcvbnhjsdfgh

基本要求：

1. 总结实验1-实验9，完成实验报告。
2. 包含所有完成的实验基本功能，解释实验原理
3. 解释编程的思路，附带程序源代码

加分项：

1. 报告图文并茂，结构清晰，有条理
2. 程序的原理图，框图
3. 实验视频演示
4. 增加新功能
5. 将实验两两融合，形成系统

avatar

```
12345
```

实验介绍

实验原理

编译部分

Lab2：双色LED实验

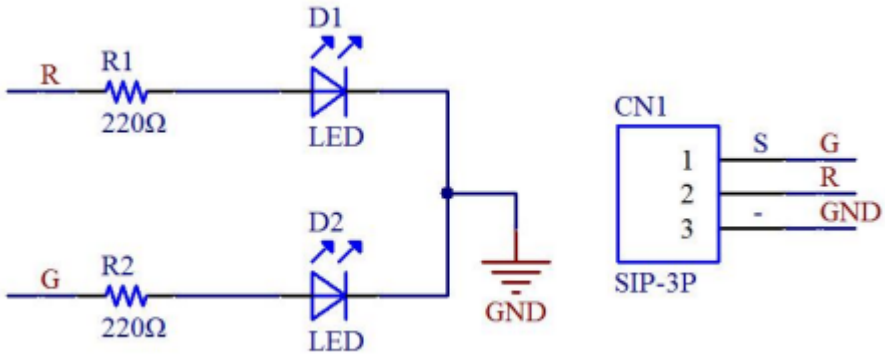
实验介绍

本次实验里，我主要了解如何在树莓派上调用不同的接口，并学习了单片机学习最基础的点灯实验，并实现了LED灯红绿交替闪烁。

双色发光二极管（LED）能够通过将高电压接入不同的引脚，发出一般红色和绿色两种不同颜色的光。其一次只能有一个引脚接受电压，实际应用常作为设备的指示灯。

实验原理

将引脚 S（绿色）和中间管脚（红色）连接到Raspberry Pi 的 GPIO0（11）和GPIO1（12）上,引脚-连接到树莓派的 GND 上，利用树莓派里的MU软件进行python编程开发，实现LED灯红绿交替闪烁。



编译部分

目标：实现LED灯红绿交替闪烁

```
import RPi.GPIO as GPIO
import time

#设置GPIO口的模式和输出的端口
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11,GPIO.OUT)
GPIO.setup(12,GPIO.OUT)
#防止代码修改后运行，由于端口重复定义警告
GPIO.setwarnings(False)

def loop():
    while True:
        #绿灯灭，红灯亮
        GPIO.output(11,False)
        GPIO.output(12,True)
        time.sleep(0.5)
        #绿灯亮，红灯灭
        GPIO.output(11,True)
        GPIO.output(12,False)
        time.sleep(0.1)

if __name__ == '__main__':
    loop()
```

Lab3：轻触按键开关实验

实验介绍

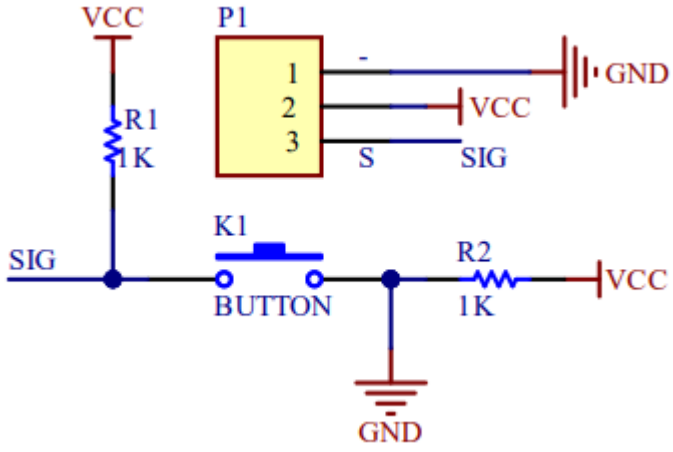
在本次实验中，我学习了轻触开关的使用，并结合lab2点灯实验，实现了按键改变灯的显示状态。

轻触开关模块是最常见的开关模块，内部有一个轻触开关（按键开关）。引脚-接GND，中间引脚接VCC。按下按键时，S脚输出为低电平；松开按键时，S脚输出为高电平。

实验原理

在本实验中，我将使用轻触开关作为树莓派的输入设备。首先，需要将树莓派的某个 GPIO口设置为输入模式，然后通过该 GPIO口检测轻触开关的S引脚。当S引脚为低电平时，表示按键被按下；当S引脚为高电平时，表示按键被松开。

同时，我通过开关控制lab2所用双色LED模块的不同显示状态。开关的原理图如下：



编译部分

基础目标：实现轻触开关控制lab2中LED灯亮起

扩展目标：

- 开关控制状态切换：

1. 按一下按键，LED红灯亮起； 2 再次按一下按键，LED红灯闪烁；
2. 再次按一下按键，LED绿灯亮；
3. 再次按一下按键，LED绿灯闪烁。

- 抖动消除

```
import RPi.GPIO as GPIO
import time

LED_1 = 11
LED_2 = 12
BUTTON = 13

#设置GPIO口的模式和输出的端口
GPIO.setmode(GPIO.BOARD)
GPIO.setup(LED_1,GPIO.OUT)
GPIO.setup(LED_2,GPIO.OUT)
GPIO.setup(BUTTON,GPIO.IN,pull_up_down=GPIO.PUD_UP)
#防止代码修改后运行，由于端口重复定义警告
GPIO.setwarnings(False)

#定义不同状态的LED显示，红/绿，是否闪烁
def control_led(color,blink):
    if color == 'r':
        GPIO.output(LED_1,GPIO.HIGH)
        GPIO.output(LED_2,GPIO.LOW)
        if blink:
            for _ in range(2):
                GPIO.output(LED_1,GPIO.LOW)
                time.sleep(0.5)
                GPIO.output(LED_1,GPIO.HIGH)
                time.sleep(0.5)
    elif color == 'g':
        GPIO.output(LED_1,GPIO.LOW)
        GPIO.output(LED_2,GPIO.HIGH)
        if blink:
        time.sleep(0.5)
        for _ in range(2):
                GPIO.output(LED_2,GPIO.LOW)
                time.sleep(0.5)
                GPIO.output(LED_2,GPIO.HIGH)
                time.sleep(0.5)

#判断按键次数，4次一周期循环切换状态
def check_button():
    count = 0
    while True:
        if GPIO.input(BUTTON) == GPIO.LOW:
            count += 1
            if count == 1:
                control_led('r',False)
            elif count == 2:
                control_led('r',True)
            elif count == 3:
                control_led('g',False)
            elif count == 4:
                control_led('g',True)
                count = 0
        #消除抖动
```

```
            while GPIO.input(BUTTON) == GPIO.LOW:
                continue

    if __name__ == '__main__':
        check_button()
```

lab4：PCF8591模数转换器实验

实验介绍

在本次实验中，我学习了模数转换器PCF8591的使用，并利用上面的电位器、热敏电阻和光敏电阻控制lab2模块所用LED的亮度。

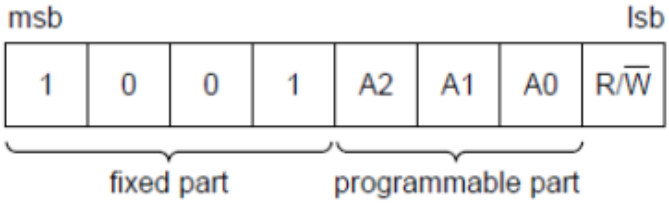
PCF8591是一款功能丰富的模拟输入/输出设备，广泛用于各种嵌入式系统中。它有以下主要特点：

1. 四个模拟输入：PCF8591可以同时连接四个模拟传感器或信号源，从而实现一次采集多个模拟输入信号的能力。
2. 一个模拟输出：除了输入功能外，PCF8591也具有模拟输出功能，能够输出模拟信号，用于操控其他模拟设备。
3. 串行I^2C总线接口：通过I^2C总线接口，PCF8591可以方便地与其他设备进行通信，实现数据的收发和控制指令的传输。
4. 可编程地址：通过地址引脚A0、A1和A2进行编程，可以使得多达8个PCF8591设备同时连接到同一个I^2C总线，而无需额外的硬件开关或选择器。
5. 最大转换率：PCF8591的最大转换率受限于I^2C总线的最大速度。因为I^2C总线的速度通常受到系统时钟频率和其他设备的影响，所以最大转换率可能会因系统不同而有所差异。

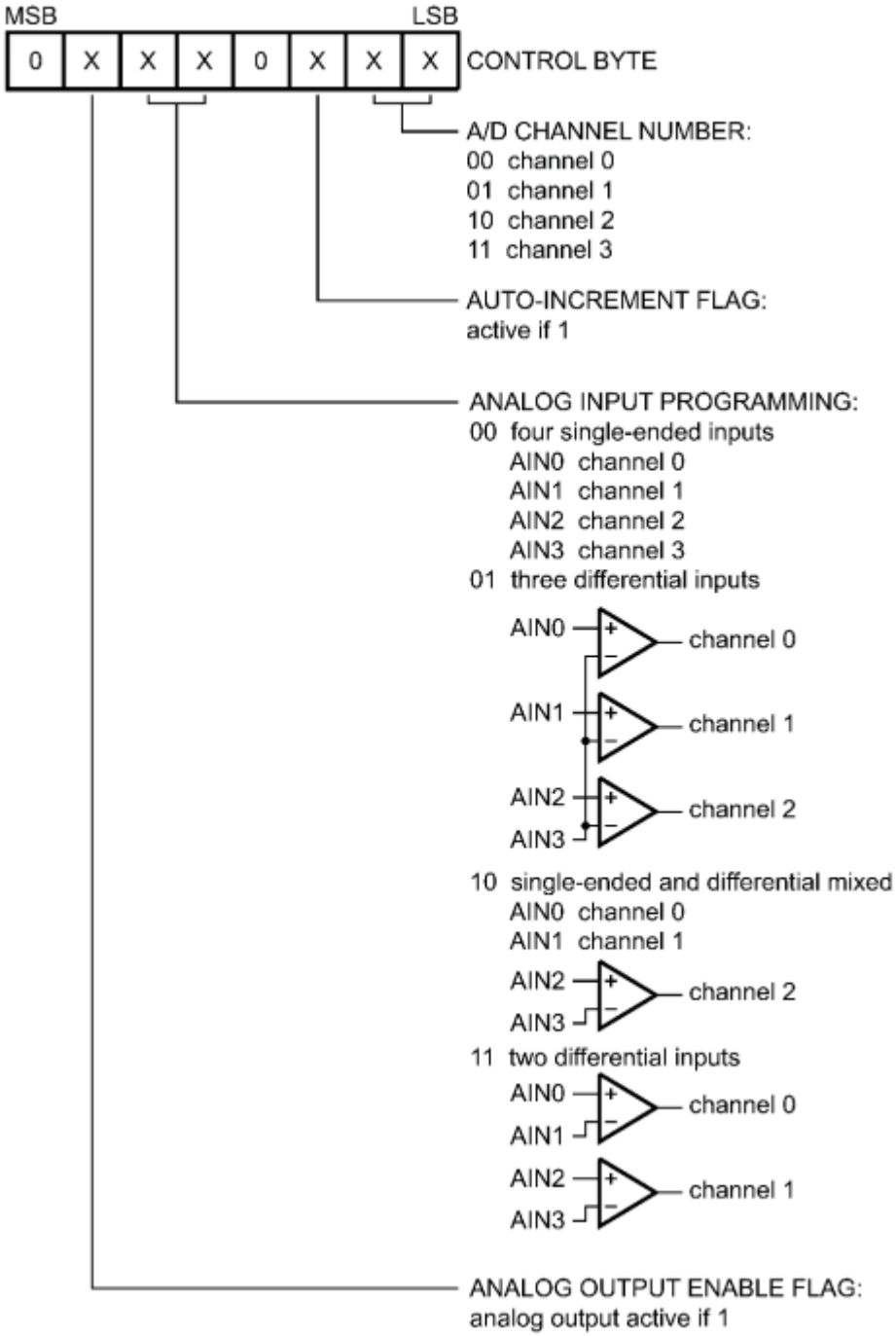
通过这些特点，PCF8591 能够灵活地应用于各种嵌入式系统中，实现模拟信号的采集、输出和处理。

实验原理

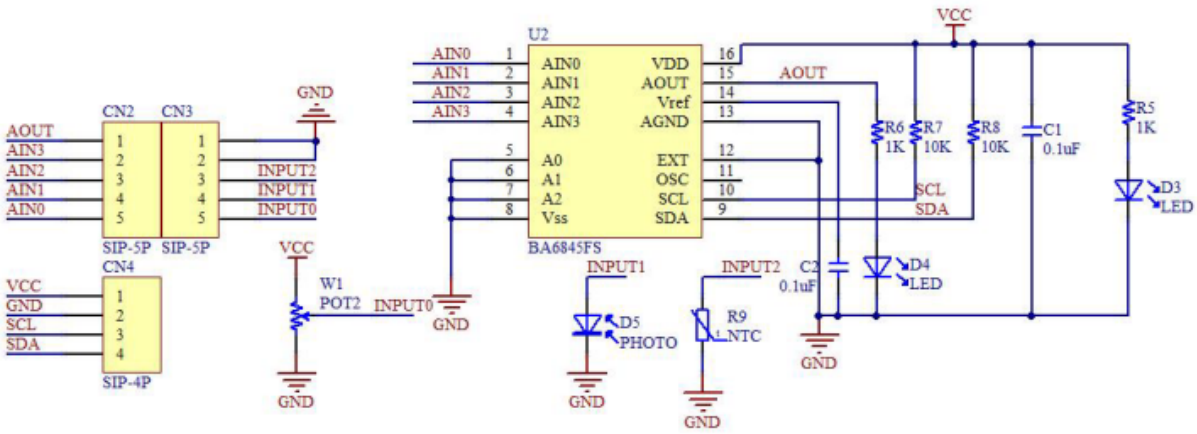
在I2C总线系统中，每个PCF8591设备通过发送有效地址来激活。地址由固定部分和可编程部分组成，可编程部分需要根据地址引脚A0，A1，A2进行设置。地址字节的最后一位是读/写位，用于设置数据传输的方向。



传输到PCF8591设备的第二个字节将被存储在其控制寄存器中，用于配置设备功能。控制寄存器的高半字节用于使能模拟输出，并将模拟输入编程为单端或差分输入。下半字节选择一个模拟输入通道。



在本实验中，AIN0端口用于接收来自电位计模块的模拟信号，AOUT端口用于将模拟信号输出到双色LED模块，以改变LED的亮度。此外，PCF8591模块还带有光电二极管和负温度系数（NTC）热敏电阻，可以实现光强和温度感知。



总的来说，PCF8591设备在I2C总线系统中实现了模拟输入输出的功能，并能够应用于光敏和热敏电阻数据的采集与控制。

编译部分

基础目标：实现利用PCF8591上的电位器控制lab2中的双色LED灯亮度

扩展目标： 实现光敏电阻（环境光照度）和热敏电阻（环境温度）控制LED灯亮度

```
import smbus
import time

bus = smbus.SMBus(1)
PCF8591_ADDRESS = 0x48
OFFSET = 0x00

def setup():
    # 在PCF8591上设置通道0的增益为1，并且偏移量为 0
    bus.write_byte_data(PCF8591_ADDRESS, 0x00, 0x03)
    bus.write_byte_data(PCF8591_ADDRESS, 0x01, OFFSET)

def read_channel(channel):
    # 设置并读取通道
    bus.write_byte_data(PCF8591_ADDRESS, channel, OFFSET)
    value = bus.read_byte_data(PCF8591_ADDRESS, channel)
    return value - OFFSET

def write_channel(channel, value):
    # 设置要写入的通道
    bus.write_byte_data(PCF8591_ADDRESS, channel, value)

if __name__ == "__main__":
    setup()
    while True:
        # 读取并写入通道，控制 LED 的亮度
        # 0是热敏电阻，1是光敏电阻，2是电位器
        brightness = read_channel(2)
```

```
        write_channel(1, brightness)
        time.sleep(2)
```

lab5：模拟温度传感器实验

实验介绍

在上次实验里，我已经了解如何通过模数转换器感知温度的变化，而在这次实验里，我将学习一个新的更加精确的测温模块 —— NTC热敏电阻。

温度感测模块提供易于使用的传感器，它带有模拟和数字输出。该温度模块使用NTC（负温度系数）热敏电阻来检测温度变化，其对温度感应非常灵敏。NTC热敏电阻电路相对简单，价格低廉，组件精确，可以轻松获取项目的温度数据，因此广泛应用于各种温度的感测与补偿中。 简而言之，NTC热敏电阻将随温度变化传递为电阻变化，利用这种特性，我们可以通过测量电阻网络(例如分压器)的电压来检测室内/环境温度。

实验原理

具体温度计算步骤如下：

1. 通过函数 ADC.read(0) 取得传感器模拟输出通过A/D转化后的数字值：

   analogVal = ADC.read(0)

2. 利用上面的值计算热敏电阻的原始模拟电压值Vr ：

   $V_r = \frac{5 \times \text{float(analogVal)}}{255}$

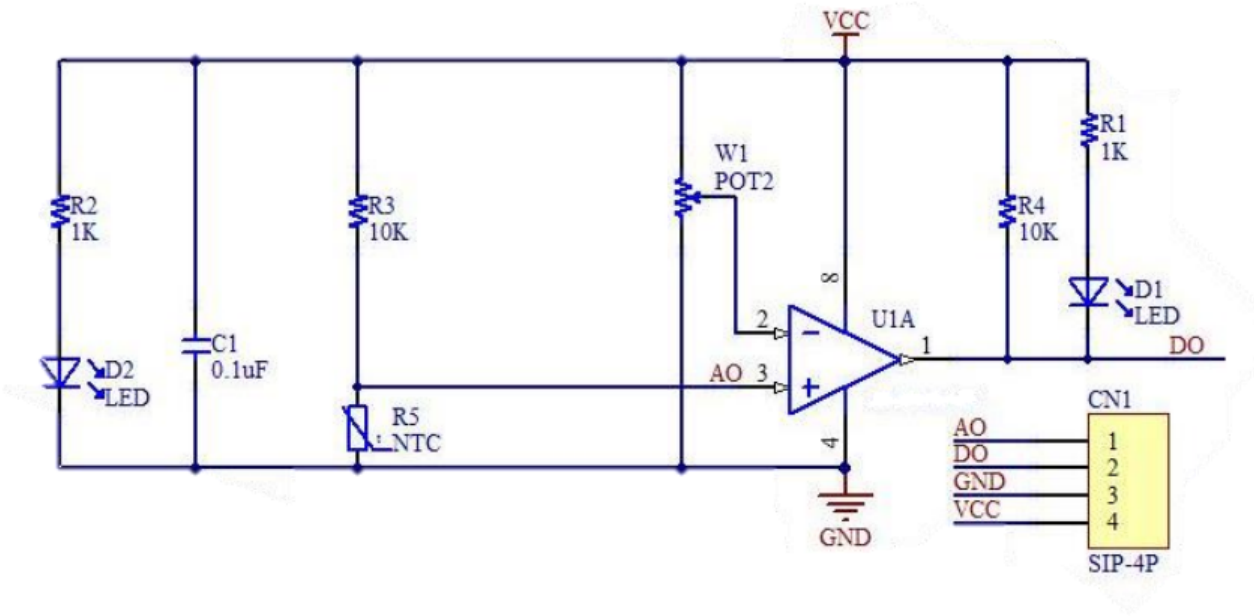
3. 从下面的电路图可知，R3与R5串联电流相等，R3电阻阻值为10K，所以热敏电阻值:

   $R_t = \frac{10000 \times V_r}{5 - V_r}$

4. 根据Steinhart-Hart 方程

   $R_t=R_0 \times exp[B \times (\frac{1}{T}-\frac{1}{T_0})]$

且 $\begin{cases} R_0 = 10000\ T_0 = 298.15K\ B = 3950\ T_{Celsius} = T - 273.15 - 13\ \end{cases}$



编译部分

```
import time
import math
import smbus

ADDRESS = 0x48
CHANNEL = 0x40

class PCF8591:
    def __init__(self):
        self.bus = smbus.SMBus(1)

    def read(self, channel):
        self.bus.write_byte(ADDRESS, CHANNEL + channel)
        self.bus.read_byte(ADDRESS)
        return self.bus.read_byte(ADDRESS)

def compute_temperature(analogVal):
    Vr = 5 * float(analogVal) / 255
    Rt = 10000 * Vr / (5 - Vr)
    R0 = 10000
    T0 = 298.15
    B = 3950
    inv_T = 1/T0 + 1/B * math.log(Rt/R0)
    T = 1/inv_T
    T_Celsius = T - 273.15-13
    return T_Celsius

def main():
    pcf = PCF8591()

    try:
```

```
        while True:
            analogVal = pcf.read(0)
            temperature = compute_temperature(analogVal)
            print(f"Measured T: {temperature:.2f}°C")
            time.sleep(1)
    except KeyboardInterrupt:
        print("Program stopped.")

if __name__ == "__main__":
    main()
```

lab6：超声波传感器实验

lab7：蜂鸣器实验

lab8：PS2操纵杆实验

lab9：红外遥控实验