基于 Arduino UNO 的课题研究记录

一、 光敏电阻的照度曲线的测量

1. 实验概述

在拿到光敏电阻的时候，我就自然而然地想要测量其电阻随光照强度的变化曲线。但光敏电阻返回的是电压值，这个电压值可以用来衡量光敏电阻在某一特定光照强度下的阻值，却无法给出光照强度的数值。为了测量光照强度的数值，我另外购买了 BH1750FVI 数字式环境光传感器，该传感器可以通过串口给出环境光照强度。考虑到 BH1750FVI 感受到的光强与光敏电阻感受到的光强可能存在一定差别，定量的照度曲线肯定是无法实现了，但定性的照度曲线还是可以通过选择合适的光源来得到的。

2. 实验过程及结果

本实验选用三色 LED 模块（红光）为光源，通过改变模拟输入数值的大小，来改变光源的亮度（类似于呼吸灯的制作）。那么，实验的第一步，就是测量光源的亮度与模拟输入数值之间的关系。电路连接图如图 1 所示，为了减弱环境光的影响，实验是在晚上，关灯，并用不透明塑料盒盖住实验装置的条件下进行的，结果如表 1 所示。可以看到，在模拟输入数值以 30 为单位从 30 变到 180 的过程中，光照强度几乎是完美的线性变化。但总体来看，光照强度随模拟数值的变化率有逐渐减小的趋势。可能的原因是在光照强度较弱时，光线几乎都被束缚在了盒子内，传到外界的光线很少；而随着光照强度的增强，从盒子外壁可以看到明显的

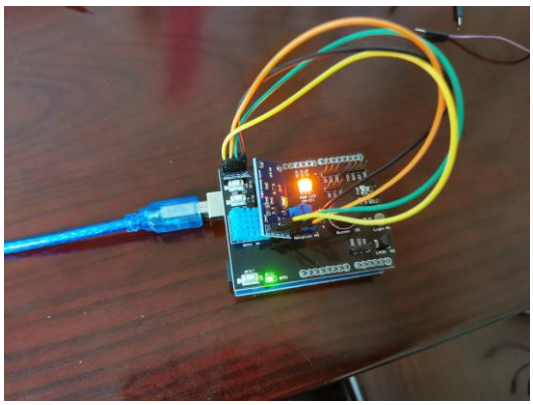

图 1. BH1750FVI 电路连接图

红色光晕，表明有大量光线“逃”到了盒子外面，导致盒内光照强度减小。

模拟输入	0	30	60	90	120	150	180	210	240
光照强度（lx）	0	10	19	27	36	45	54	62	70

表 1. 光照强度与模拟输入数值的关系

接下来拆除 BH1750FVI 模块，接入光敏电阻模块，关灯，盖上塑料盒，以 10 为单位改变 LED 的模拟输入数值，读取光敏电阻返回的电压值。可以看到，随着光照强度的增大，光敏电阻返回的电压值也在逐渐增大，表明光敏电阻返回的是电路内电阻（设为 r0）两端的电压。这是因为光敏电阻一般是用半导体材料制作的，光照强度越大，产生的电子空穴数也就越多，电阻相应减小。由于使用的是 arduino 自带的 5V 电源，可将电源电压默认为 5V。那么，由公式$U(I)=\frac{r_0}{r_0+R(I)}U_0$可求出 R（I）（以 r0 为单位，这里假设 r0 不变）。同时根据第一步实验得到的结果，可用 40-180 模拟输入数值替代光强，并以 10 为单位（设其对应光强为 I0），由此作出光敏电阻的照度曲线，如图 2 所示。

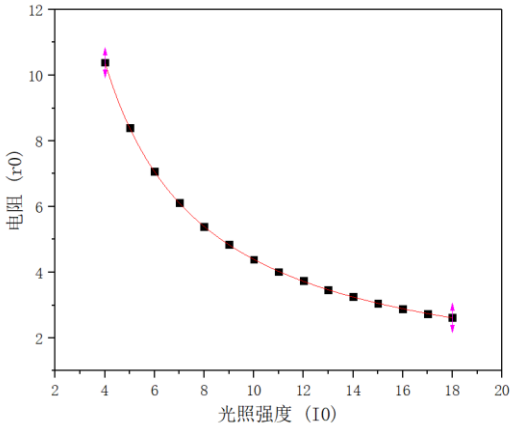

图 2. 光敏电阻的照度曲线

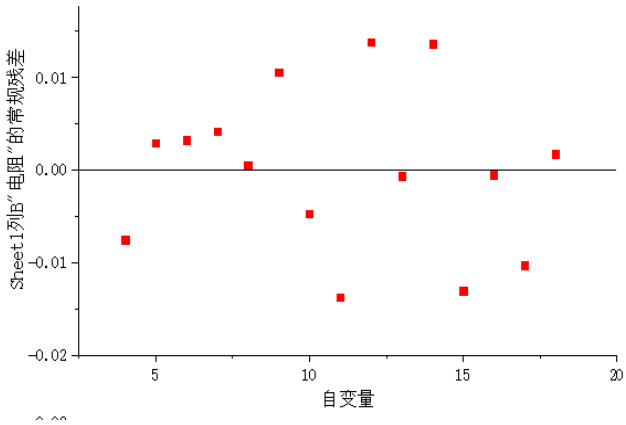

图 3. 拟合残差图

用公式$y=Ax^{-1}+B$拟合，R 平方达到 5 个 9，拟合残差图如图 3 所示。

实验结果表明，在一般的光照强度下（可近似表达成不高于 54lx，不低于

10lx)，光敏电阻的阻值与光照强度成反比例函数关系。

3. 实验代码

① BH1750FVI 部分：

```
#include <Wire.h>
#define ADDRESS_BH1750FVI 0x23
#define ONE_TIME_H_RESOLUTION_MODE 0x20

#define LED_R 9
#define LED_B 10
#define LED_G 11

#define LED_ON HIGH
#define LED_OFF LOW

int Value =30;
int state = 0;

byte highByte = 0;
byte lowByte = 0;
unsigned int sensorOut = 0;
unsigned int illuminance = 0;

void setup()
{
  Wire.begin();
  Serial.begin(115200);
  pinMode(LED_R,OUTPUT);
  pinMode(LED_B,OUTPUT);
  pinMode(LED_G,OUTPUT);
}

void loop()
{
  Wire.beginTransmission(ADDRESS_BH1750FVI);
  Wire.write(ONE_TIME_H_RESOLUTION_MODE);
  Wire.endTransmission();

  delay(180);

  Wire.requestFrom(ADDRESS_BH1750FVI, 2);
  highByte = Wire.read();    // get the high byte
  lowByte = Wire.read(); // get the low byte
```

```
    sensorOut = (highByte<<8)|lowByte;
    illuminance = sensorOut/1.2;
    Serial.print(illuminance);
    Serial.println(" lux");
    analogWrite(LED_R,Value);
    analogWrite(LED_B,0);
    analogWrite(LED_G,0);
    delay(1000);
}
```

② 光敏电阻部分：

```
#define LED_R 9
#define LED_B 10
#define LED_G 11
#define LIGHT A1

int val = 0;
int value = 0;
int value2 = 210;

void setup()
{
    pinMode(LED_R,OUTPUT);
    pinMode(LED_B,OUTPUT);
    pinMode(LED_G,OUTPUT);
    Serial.begin(9600);
}

void loop()
{
    val = analogRead(LIGHT);
    value = map(val,0,1023,0,5000);
    analogWrite(LED_R,value2);
    analogWrite(LED_B,0);
    analogWrite(LED_G,0);
    delay(2000);
    Serial.println(value);
}
```

二、 温控风扇的制作

1. 实验概述

实验目标是制作一个随着温度变化改变档位（转速）的电风扇，为此需要有

一个温度传感器，配有扇叶的直流电机以及直流电机驱动模块。实验中用LCD1602 液晶屏实时显示温度以及电风扇档位。

2. 实验过程及结果

电路连接图及实验结果如图 4 所示。最开始我用的是 LM35 温度传感器，但在实验过程中我发现，LM35 测得的温度值波动十分剧烈。随着电机转速的提高，LCD1602 的亮度会减弱，同时 LM35 返回的温度值会提高。在室温 24℃的条件下，5 档电风扇工作时显示温度可达到41℃。通过查阅资料，得知 arduino 的 5Vpower 引脚的工作电流为 200mA，而电机的最大工作电流甚至已经超过了这个数值，所以当电机处于高转速状态时，LCD1602 以及 LM35 分到的电流都会相应减小，最终导致 LCD1602 亮度减小以及 LM35 测得的温度不准（或者说计算公式发生了变化）。这一问题可以通过三种途径解决，一是更换板子，二是采用外部电源（有待商榷），三是更换温度传感器（实际上并不会解决问题，但至少能改变受影响程度）。我这里采用的是第三种方法。将 LM35 替换为 DHT11，温度的波动范围大致为 2~3℃，实验效果的提升可以说是相当显著了，如图 5 所示。不过用 LM35 也有一个好处，那就是风扇的档位会随着显示温度的波动不断变化，可以作为实现“温控”的凭据。

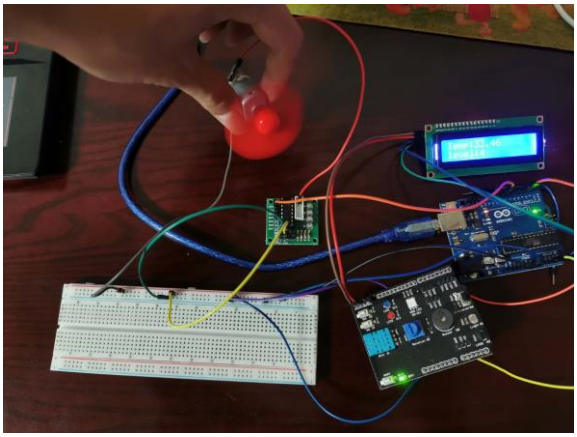

图 4. 温控风扇效果图（LM35）

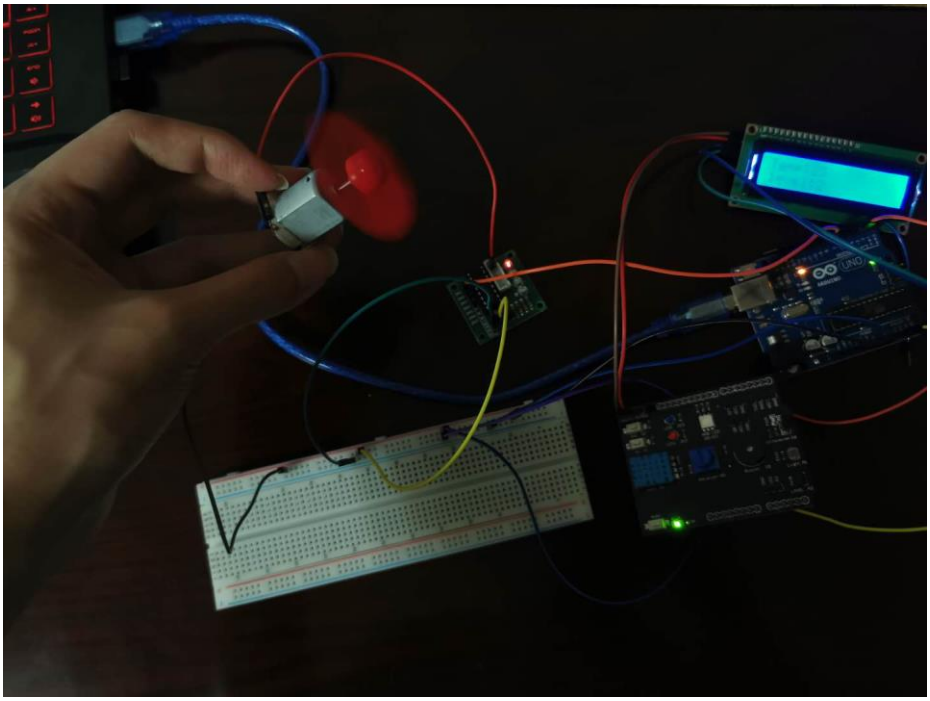

图 5. 温控风扇效果图（DHT11）

3. 实验代码

DHT11：

```
#include <LiquidCrystal_I2C.h:
#include <Wire.h>
#include <dht11.h>
dht11 DHT11;
#define DHT11PIN 4
#define U 9

int LEVEL = 0;

LiquidCrystal_I2C lcd(0x27,16,2);

void setup()
{
  Serial.begin(9600);
  lcd.init();
  lcd.backlight();
}

void loop()
{
  int chk = DHT11.read(DHT11PIN);
  lcd.setCursor(0,0);
```

```
lcd.print("Temp:");        //输出字符到 LCD1602 上
lcd.print(DHT11.temperature);
if(DHT11.temperature<=15)
{
  lcd.setCursor(0,1);
  lcd.print("level:");
  lcd.print(0);
  LEVEL = 0;
}
if(DHT11.temperature>15 & DHT11.temperature<=20)
{
  lcd.setCursor(0,1);
  lcd.print("level:");
  lcd.print(1);
  LEVEL = 100;
}
if(DHT11.temperature>20 & DHT11.temperature<=25)
{
  lcd.setCursor(0,1);
  lcd.print("level:");
  lcd.print(2);
  LEVEL = 150;
}
if(DHT11.temperature>25 & DHT11.temperature<=30)
{
  lcd.setCursor(0,1);
  lcd.print("level:");
  lcd.print(3);
  LEVEL = 180;
}
if(DHT11.temperature>30 & DHT11.temperature<=35)
{
  lcd.setCursor(0,1);
  lcd.print("level:");
  lcd.print(4);
  LEVEL = 210;
}
if(DHT11.temperature>35)
{
  lcd.setCursor(0,1);
  lcd.print("level:");
  lcd.print(5);
  LEVEL = 250;
}
```

```
  analogWrite(U,LEVEL);
  delay(2000);
}
```

三、　用手机控制档位的风扇的制作

1. 实验概述

和第二个实验类似，这个实验是想通过蓝牙模块控制风扇的转速，相当于是对蓝牙模块的一个实际应用。

2. 实验过程及结果

电路连接及效果如图 6 所示，这里使用的是 HC05 蓝牙模块。手机端通过输入数字 1、2、3、4 或 5，可以改变风扇的转速。但实验过程中发现当 arduino 接上 HC05 后，程序将无法烧录，同时返回错误：avrdude: stk500_getsync(): not in sync: resp=0x00。通过查阅资料，得知这个错误语句的意思是 arduino 无法识别 0、1 号引脚（即 arduino 的数据接受和输出端引脚）所接的扩展板（也就是 HC05）。故这里需要先烧录程序再接入 HC05 蓝牙模块。之后打开手机上安装好的蓝牙串口助手 APP，连接蓝牙模块，连接成功后发送数字，电机按照正常档位工作。但随后又出现了第二个问题，那就是 APP 与蓝牙模块之间的连接自动断开了，而且无法重新连上。我更换了几个 APP，都会出现同样的问题。这个问题我还没有解决，或许是 arduino UNO 本身不支持 HC05 模块。所以本实验只实现了手机端对风扇的驱动，没有实现连续控制。

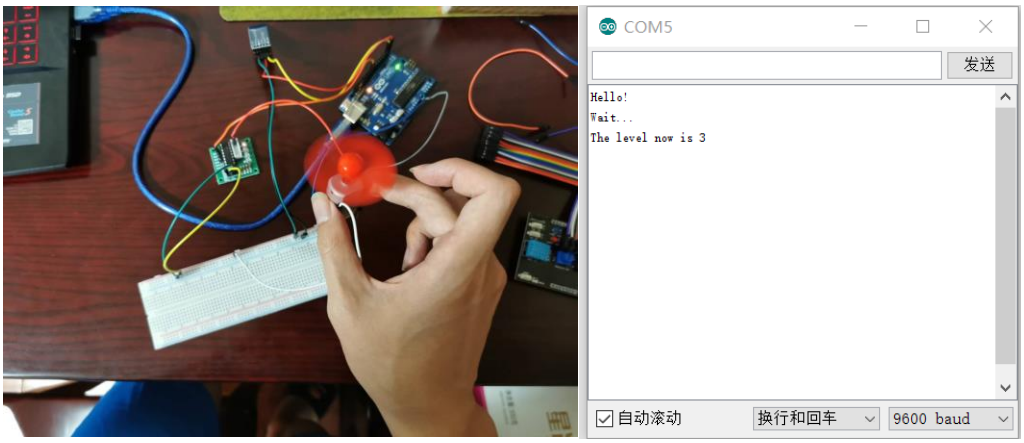

图 6. 蓝牙模块控制电风扇

3. 实验代码

```
#define U 9

int LEVEL = 0;

void setup()
{
   Serial.begin(9600);
   Serial.println("Hello!");
   Serial.println("Wait...");
}

void loop()
{
   while(Serial.available())
   {
     char c = Serial.read();
     if(c=='1')
     {
       LEVEL = 100;
       Serial.write("The level now is 1");
     }
     else if(c == '2')
     {
       LEVEL = 150;
       Serial.write("The level now is 2");
     }
     else if(c == '3')
     {
       LEVEL = 180;
```

```
      Serial.write("The level now is 3");
    }
    else if(c == '4')
    {
      LEVEL = 210;
      Serial.write("The level now is 4");
    }
    else if(c == '5')
    {
      LEVEL = 255;
      Serial.write("The level now is 5");
    }
    analogWrite(U,LEVEL);
  }
}
```