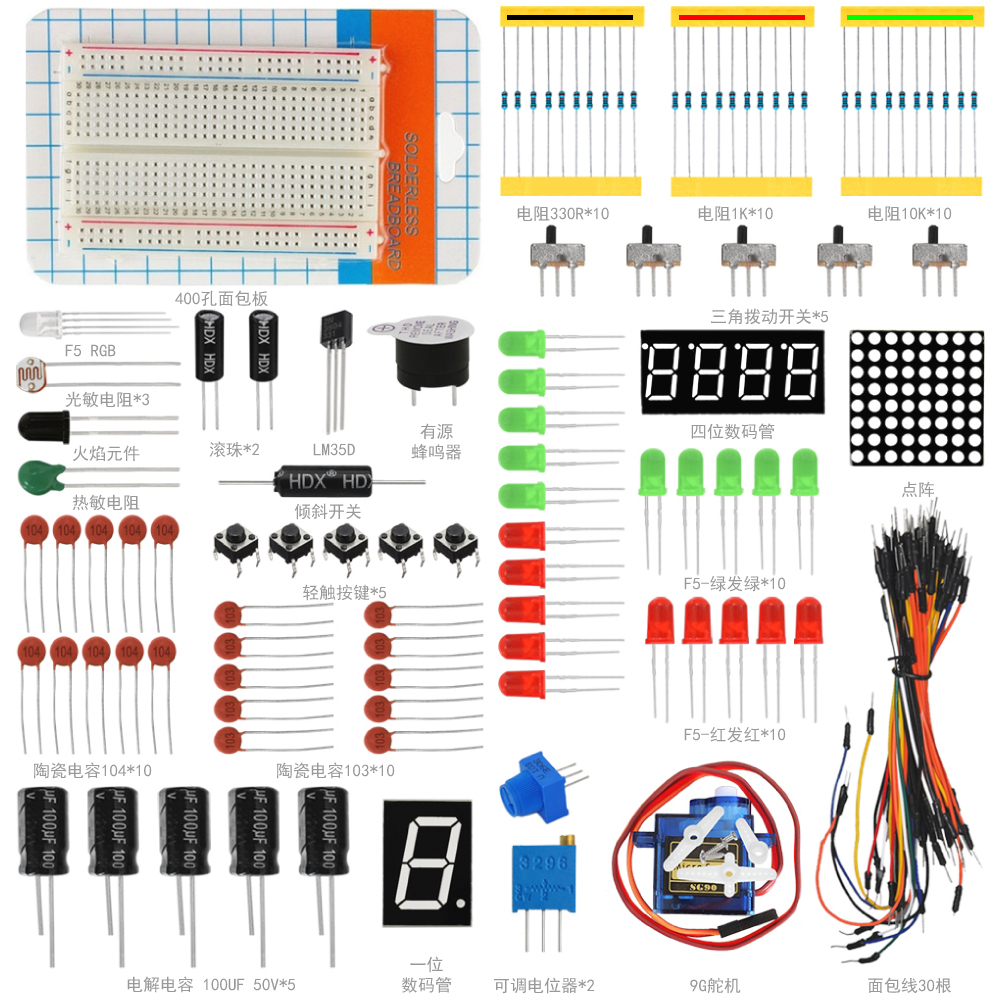
**Arduino电子爱好者通用元件包套件503A**

****

**目录**

[1、说明 3](#_Toc2403)

[2、清单 3](#_Toc22069)

[3、Arduino IDE和驱动的安装 6](#_Toc10241)

[4、Arduino IDE的使用方法 12](#_Toc10568)

[5、实验课程 14](#_Toc8989)

[实验一 LED 闪烁实验 14](#_Toc22416)

[实验二 呼吸灯实验 15](#_Toc17296)

[实验三 广告灯实验 17](#_Toc22004)

[实验四 按键控制LED实验 18](#_Toc1105)

[实验五 抢答器实验 20](#_Toc20112)

[实验六 电位器调控灯光亮度实验 23](#_Toc7681)

[实验七 感光灯实验 24](#_Toc23186)

[实验八 有源蜂鸣器实验 25](#_Toc17379)

[实验九 火焰报警实验 27](#_Toc3028)

[实验十 拨码开关控制LED实验 29](#_Toc17794)

[实验十一 LM35检测温度 30](#_Toc19560)

[实验十二 魔术光杯实验 32](#_Toc8848)

[实验十三 热敏电阻传感器实验 35](#_Toc2938)

[实验十四 一位数码管显示实验 36](#_Toc19145)

[实验十五 8\*8点阵显示实验 40](#_Toc10790)

[实验十六 四位数码管显示数字实验 43](#_Toc20957)

[实验十七 舵机控制实验 50](#_Toc28738)

[实验十八 震动传感器实验 53](#_Toc24489)

[6、 相关资料链接 54](#_Toc20516)

# 1、说明

这个套件包含我们玩单片机时使用到的常用元件，如不同阻值的电阻、不同颜色的LED灯、不同容值的电容、传感器、显示器等。有了这款配置好的元件包之后，玩单片机就不需要再去买什么零星的电子物料了。它适用于各种单片机和树莓派。我们还会根据这些元件，提供一些基于Arduino开发板的一些学习课程，包含接线方法，测试代码等，让你对这些电子元件和Arduino开发板有个初步的了解。

# 2、清单

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编码 | 名称 | 描述 | 数量 | 图片 |
| 1 | LED | F5-红发红-短 | 10 |  |
| 2 | LED | F5-绿发绿-短 | 10 |  |
| 3 | LED | F5-全彩RGB透明共阴 | 1 | KE0096  D-503 |
| 4 | 电阻 | 碳膜色环 1/4W 1% 330R 编带 | 10 | **84848484848484848484** |
| 5 | 电阻 | 碳膜色环 1/4W 1% 1K 编带 | 10 | **84848484848484848484** |
| 6 | 电阻 | 碳膜色环 1/4W 1% 10K 编带 | 10 | **84848484848484848484** |
| 7 | 点阵 | 20\*20MM 1.9MM 红色 共阳 | 1 | **8484** |
| 8 | 数码管 | 一位0.56英寸共阴红 | 1 | **84** |
| 9 | 数码管 | 四位0.36英寸共阴红 3461AH | 1 | **84** |
| 10 | 可调电位器 | 3296W 10k 103 | 1 | KE0093 A-503 (4) |
| 11 | 可调电位器 | 1. MU 103   （三针直排） | 1 | **84** |
| 12 | 蜂鸣器 | 有源 12\*9.5MM 5V 普通分体 2300Hz | 1 | **84** |
| 13 | 轻触按键 | 6\*6\*5MM 插件 | 5 |  |
| 14 | 拨动开关 | SS12D00 3MM柄高 三脚/两档 | 5 |  |
| 15 | 传感器元件 | LM35DZ | 1 | **84** |
| 16 | 传感器元件 | 5516 亮电阻5-10KΩ 暗电阻0.2MΩ | 3 | **848484** |
| 17 | 传感器元件 | 红外接收 5MM 火焰 | 1 | **84** |
| 18 | 滚珠开关 | HDX-2801 两脚一样 | 2 | **8484** |
| 19 | 舵机 | SG90 9G 23\*12.2\*29mm 蓝色 辉盛(环保） | 1 | **FWE** |
| 20 | 热敏电阻 | 5MM 103 阻值 10K 绿色 插件 | 1 |  |
| 21 | 传感器元件 | SW-200D 振动开关 | 1 | KE0093 A-503 (4) |
| 22 | 陶瓷电容 | 100NF 104 2.54 | 10 |  |
| 23 | 陶瓷电容 | 10NF 103 2.54 | 10 |  |
| 24 | 电解电容 | 100UF 50V 8\*12MM 插件 | 5 | KE0093 A-503 (4)KE0093 A-503 (4)KE0093 A-503 (4)KE0093 A-503 (4)KE0093 A-503 (4) |
| 25 | 面包板 | ZY-60 400孔白色（纸卡包装） | 1 |  |
| 26 | 面包线 | 面包板连接线30根 | 1 |  |
| 27 | 电阻卡 | 100\*70MM | 1 | 电阻卡 |

# 3、Arduino IDE和驱动的安装

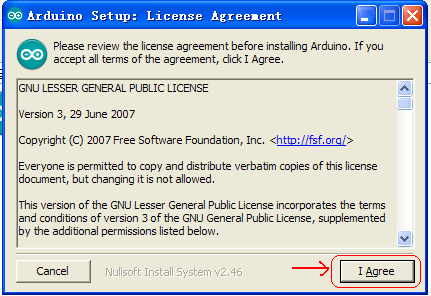
当我们拿到开发板时，首先我们要安装Arduino IDE和驱动，相关文件我们可以在官网上找到，以下链接是包含各种系统、各种版本的Arduino IDE和驱动任你选择。

<https://www.arduino.cc/en/Main/OldSoftwareReleases#1.5.x>

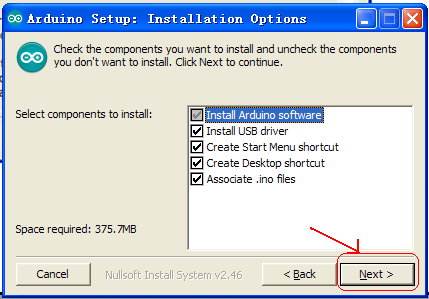
下面我们介绍下Arduino-1.5.6 版本IDE在Windows系统的安装方法。

下载下来的文件是一个arduino-1.5.6-r2-windows.zip的压缩文件夹，解压出来到硬盘。

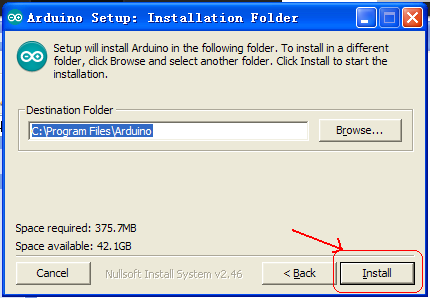
双击Arduino-1.5.6 .exe文件

****

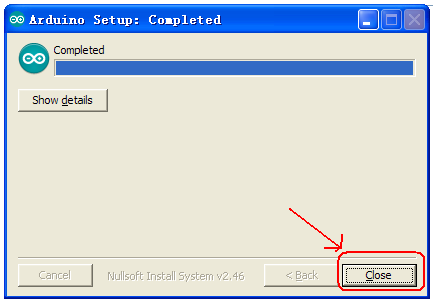
然后

****

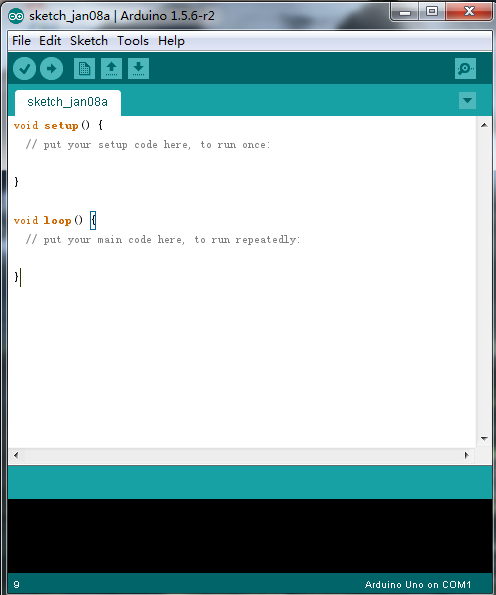
然后

****

等待安装完成.点击close，安装完成。

****

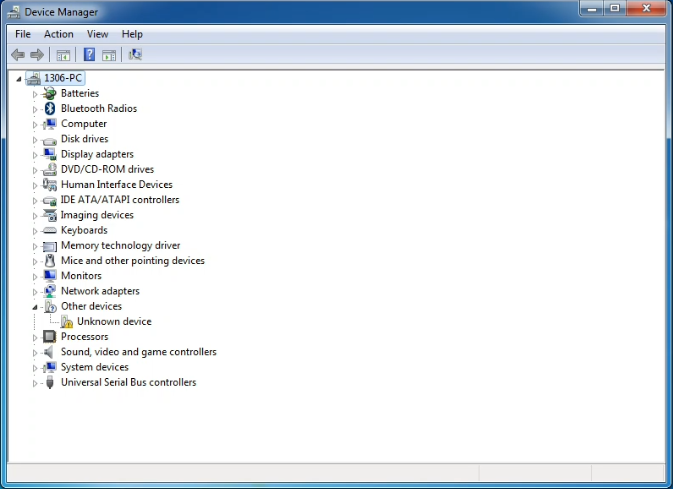
1.5.6版本安装后的样子。



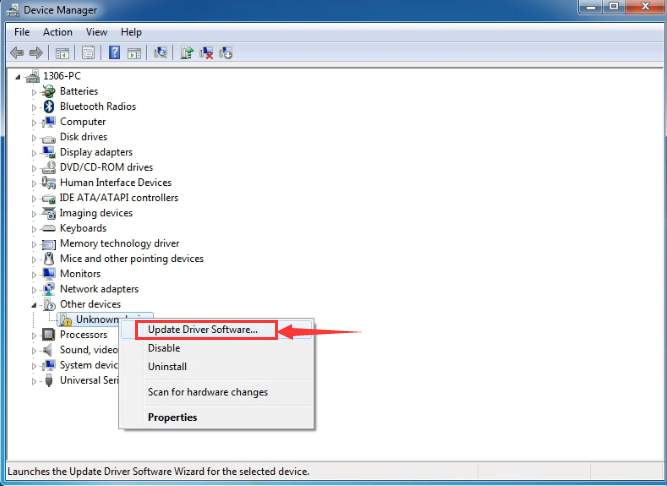
接下来是开发板驱动的安装，这次我们安装的是Keyes UNO R3 开发板的驱动，Keyes 2560 R3 开发板安装驱动方法和这个类似，驱动文件可以用同一个文件。

不同的系统，安装驱动的方法也有一些细小的区别，下面我们介绍在WIN 7系统安装驱动的方法。

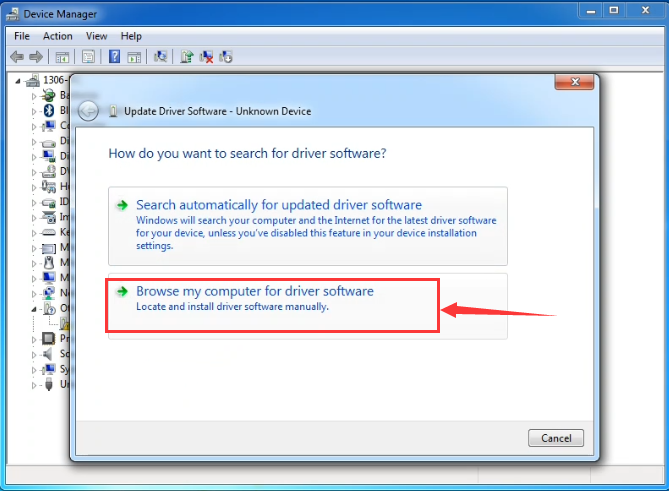
第一次Keyes UNO R3 开发板连接电脑时，点击计算机--属性--设备管理器，显示如下图。



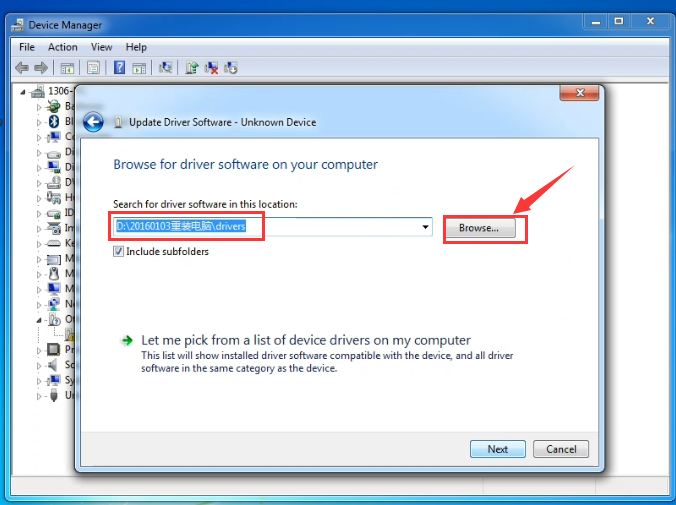
点击 Unknown device 安装驱动，如下图。



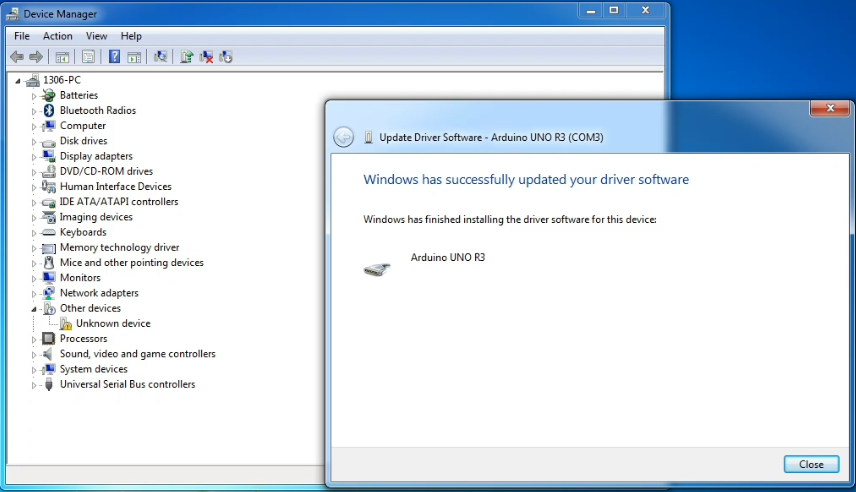
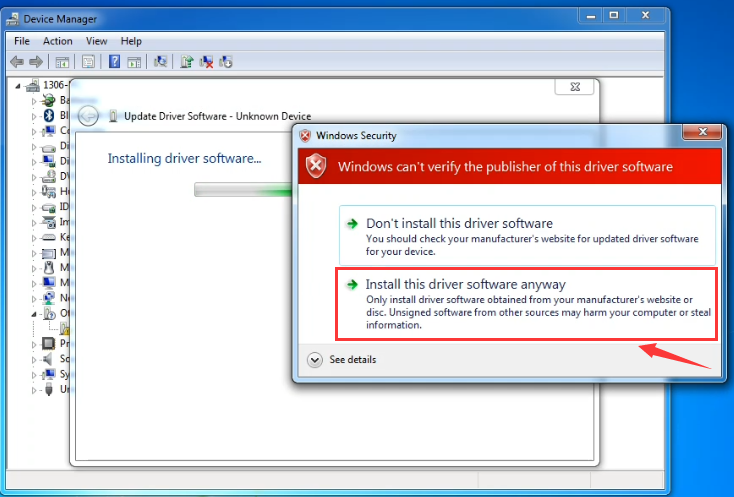
进入下图，选择



找到Arduino安装位置的drivers文件夹

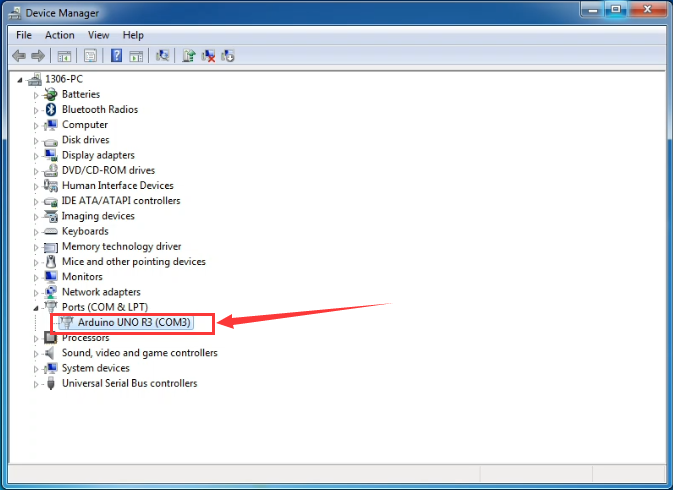


点击“Next”，今天下图选择，开始安装驱动



安装驱动完成，出现下图点击Close。

这样驱动就装好了。点击计算机--属性--设备管理器，我们可看见如下图。



# 4、Arduino IDE的使用方法

Keyes UNO R3 开发板的USB驱动安装成功之后，我们可以在Windows设备管理器中找到相应的串口。

下面示范第一个程序的烧写，串口监视器中显示“Hello World！”。

测试代码为：

int val;

int ledpin=13;

void setup()

{

Serial.begin(9600);

pinMode(ledpin,OUTPUT);

}

void loop()

{

val=Serial.read();

if(val=='R')

{

digitalWrite(ledpin,HIGH);

delay(500);

digitalWrite(ledpin,LOW);

delay(500);

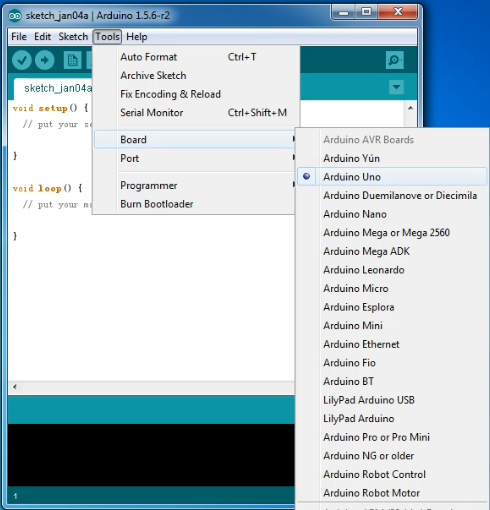
Serial.println("Hello World!");

}

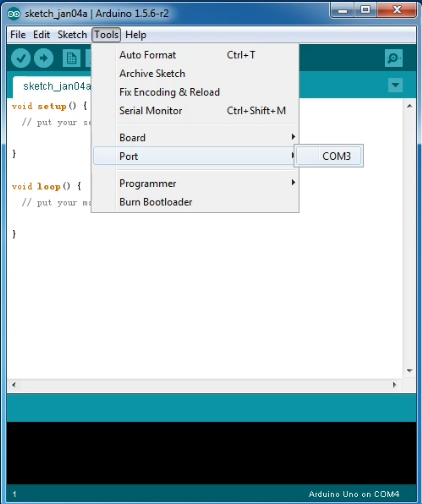
}

我们打开Arduino 的软件，编写一段程序让Keyes UNO R3 开发板接受到我们发的指令就显示“Hello World！”字符串；我们再借用一下Keyes UNO R3 开发板上的 D13 的指示灯，让Keyes UNO R3 开发板接受到指令时指示灯闪烁一下，再显示“Hello World！”。

打开Arduino 的软件，设置板，如下。



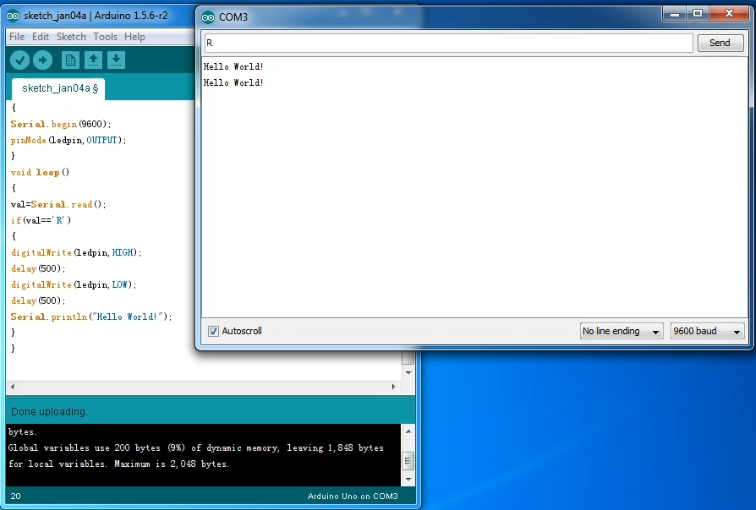
设置COM端口，如下



点击编译程序，检查程序是否错误；点击上传程序；Keyes UNO R3 开发板设置OK后右下脚显示如下图，和设备管理器中显示一致。



上传成功，输入R，点击发送，Keyes UNO R3 开发板上的 D13 的指示灯闪烁一次，串口监视器中显示 Hello World! 如下图



那么恭喜你，你的第一个程序已经成功了！！！

# 5、实验课程

## 实验一 LED 闪烁实验

**实验说明**

LED 闪烁实验是比较基础的实验之一，上一个“ Hello World！”实验里已经利用到了Arduino 自带的LED，这次我们利用其他I/O 口和外接直插LED 灯来完成这个实验。

**实验器材**

开发板\*1

USB线\*1

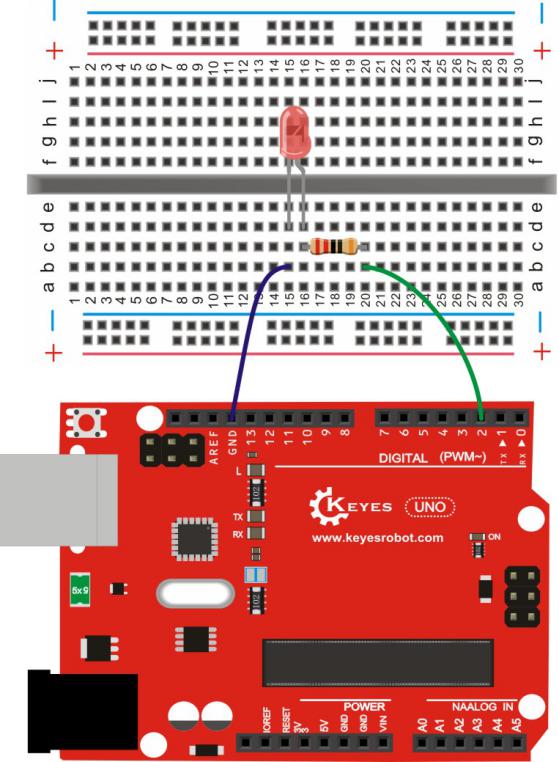
LED\*1

330Ω 电阻\*1

面包板\*1

面包板连接线若干

**接线图**

****

**测试代码**

int led = 2; //定义数字口2

void setup()

{

  pinMode(led, OUTPUT);     //设置led为输出

}

void loop()

{

  digitalWrite(led, HIGH);   //开启led

  delay(1000); //延迟1秒

  digitalWrite(led, LOW);    //关闭led

  delay(1000);//延迟1秒

}

**测试结果**

下载完程序就可以看到我们的IO口外接小灯在闪烁了，这样我们的实验现象为LED不停闪烁，间隔大约为1秒。

## 实验二 呼吸灯实验

**实验说明**

上一课程中我们只是控制LED的亮和灭，那么我们可以怎么控制LED的亮度呢？本课程中我们把LED接到PWM口中，然后通过改变PWM数值，调节LED亮度，使LED逐渐变亮，和逐渐变暗，从而达到呼吸灯的效果。

**实验器材**

开发板\*1

USB线\*1

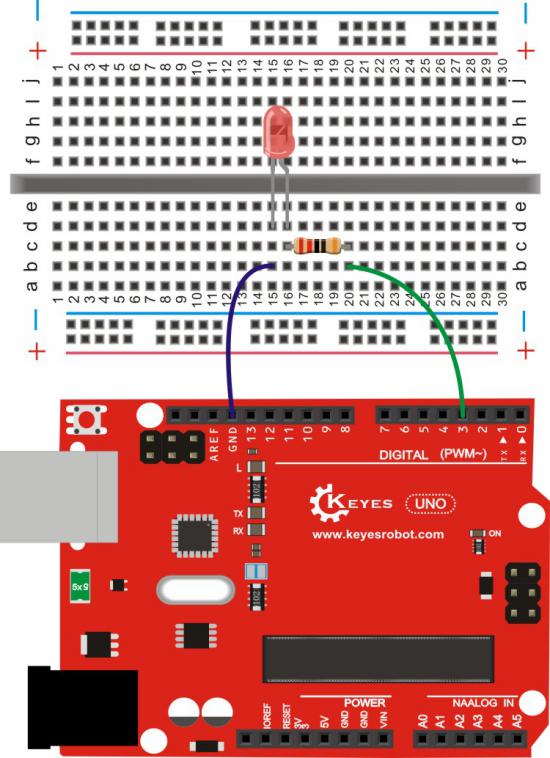
LED\*1

330Ω 电阻\*1

面包板\*1

面包板连接线若干

**接线图**

****

**测试代码**

int ledPin = 3; // 定义数字口3

void setup()

{

pinMode(ledPin, OUTPUT);// 将ledPin设置为输出

}

void loop()

{

for (int a=0; a<=255;a++)// 设置使LED逐渐变亮

{

analogWrite(ledPin,a); // 开启led,调节亮度，范围是0-255，在255时led最亮

delay(10); // 延迟0.01S

}

for (int a=255; a>=0;a--) // 设置使LED逐渐变暗

{

analogWrite(ledPin,a); // 开启led,调节亮度，范围是0-255，在255时led最亮

delay(10); // 延迟0.01秒

}

delay(1000);// 延迟1秒

}

**测试结果**

下载完程序就可以看到我们的IO口外接小灯显示出呼吸灯的效果，小灯先逐渐变亮，后逐渐变暗，循环交替。

## 实验三 广告灯实验

**实验说明**

在生活中我们经常会看到一些由各种颜色的led灯组成的广告牌，广告牌上各个位置上癿led灯不断的变话,形成各种效果。本节实验就是利用led灯编程模拟广告灯效果。

**实验器材**

开发板\*1

USB线\*1

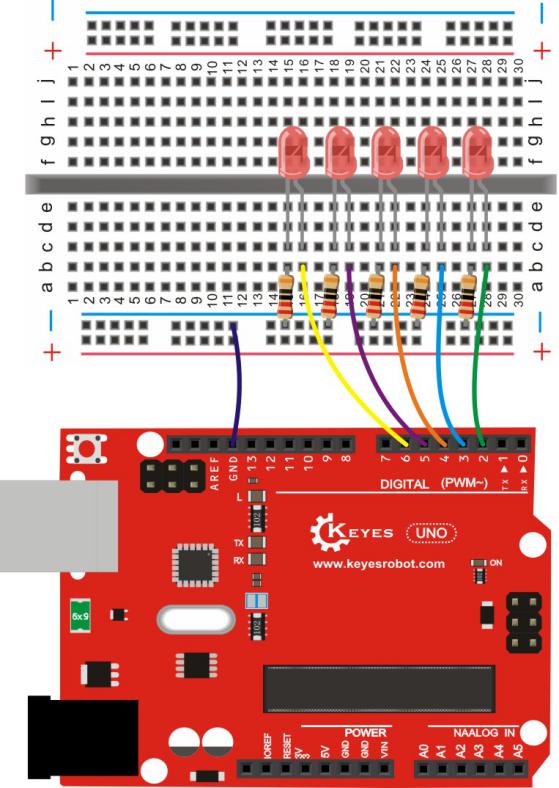
LED\*5

330Ω 电阻\*5

面包板\*1

面包板连接线若干

**接线图**

****

**测试代码**

int BASE = 2 ; //第一个 LED 接的 I/O 口

int NUM = 5; //LED 的总数

void setup()

{

for (int i = BASE; i < BASE + NUM; i ++)

{

pinMode(i, OUTPUT); //设定数字I/O口为输出

}

}

void loop()

{

for (int i = BASE; i < BASE + NUM; i ++)

{

digitalWrite(i, HIGH); //设定数字I/O口输出为"高"，即逐渐开灯

delay(200); //延迟

}

for (int i = BASE; i < BASE + NUM; i ++)

{

digitalWrite(i, LOW); //设定数字I/O口输出为"低"，即逐渐关灯

delay(200); //延迟

}

}

**测试结果**

下载完程序就可以看到我们的IO口外接小灯先逐渐变亮，然后逐渐变暗，循环交替。

## 实验四 按键控制LED实验

**实验说明**

I/O 口的意思即为INPUT 接口和OUTPUT 接口，到目前为止我们设计的小灯实验都还只是应用到Arduino 的I/O 口的输出功能，这个实验我们来尝试一下使用Arduino的I/O 口的输入功能即为读取外接设备的输出值，我们用一个按键和一个LED 小灯完成一个输入输出结合使用的实验，让大家能简单了解I/O 的作用。

**实验器材**

开发板 \*1

USB线\*1

LED\*1

轻触按键\*1

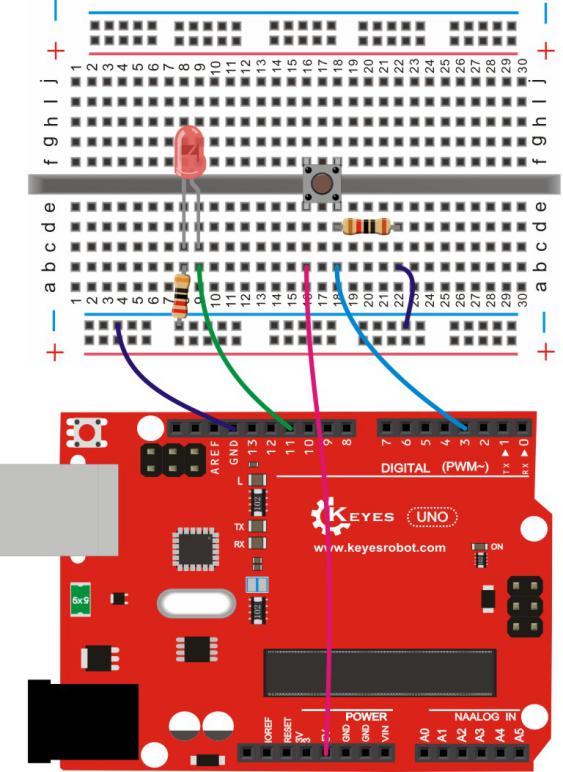
330Ω 电阻\*1

10KΩ 电阻\*1

面包板\*1

面包板连接线若干

**接线图**

****

**测试代码**

int ledPin = 11; //定义数字口11

int inputPin = 3; //定义数字口3

void setup()

{

pinMode(ledPin, OUTPUT); //将ledPin设置为输出

pinMode(inputPin, INPUT); //将inputPin设置为输入

}

void loop()

{

int val = digitalRead(inputPin);

//设置数字变量val，读取到数字口3的数值，并赋值给 val

if (val == LOW) //当val为低电平时，LED变暗

{

digitalWrite(ledPin, LOW); // LED变暗

}

else

{

digitalWrite(ledPin, HIGH); // LED亮起

}

}

**测试结果**

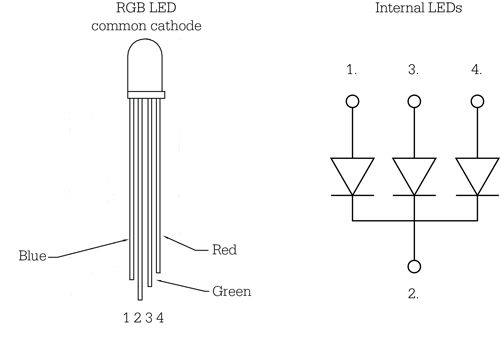
下载完程序，上电后，当按键按下时小灯亮起，否则小灯不亮。

## 实验五 抢答器实验

**实验说明**

完成上面的实验以后相信已经有很多朋友可以独立完成这个实验了，我们可以将上面的按键控制小灯的实验扩展成4个按键对应3 个小灯，占用7个数字I/O 接口。为方便接线，我们把3个小灯用一个RGB灯代替。RGB灯可通过 R、 G、 B三个引脚的PWM电压输入可以调节三种基色（红/蓝/绿）的强度从而实现全彩的混色效果。

本实验中我们利用4个按键控制3个PWM口，控制RGB模块发光颜色从而达到抢答器的效果。RGB灯接口说明如下图。



**实验器材**

开发板\*1

USB线\*1

RGB灯\*1

轻触按键\*4

10KΩ 电阻\*4

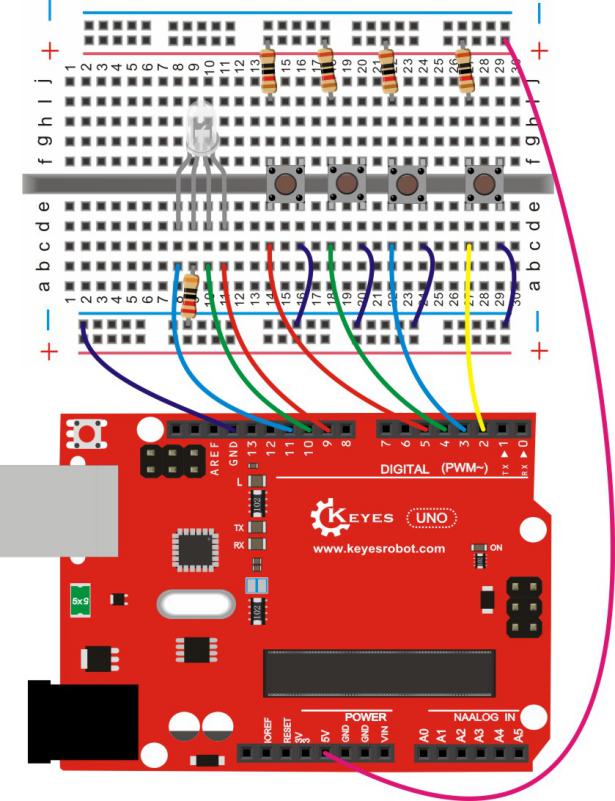
330Ω 电阻\*1

面包板\*1

面包板连接线若干

杜邦线若干

**接线图**

****

**测试代码**

int redled=9;

int greenled=10;

int blueled=11;

int redpin=5;

int greenpin=4;

int bluepin=3;

int restpin=2;

int red;

int green;

int blue;

void setup()

{

pinMode(redled,OUTPUT);

pinMode(greenled,OUTPUT);

pinMode( blueled,OUTPUT);

pinMode(redpin,INPUT);

pinMode(greenpin,INPUT);

pinMode(bluepin,INPUT);

}

void loop()

{

red=digitalRead(redpin);

green=digitalRead(greenpin);

blue=digitalRead(bluepin);

if(red==LOW)RED\_YES();

if(green==LOW)GREEN\_YES();

if(blue==LOW)BLUE\_YES();

}

void RED\_YES()

{

while(digitalRead(restpin)==1)

{

color(255, 0, 0);

}

clear\_led();

}

void GREEN\_YES()

{

while(digitalRead(restpin)==1)

{

color(0, 255, 0);

}

clear\_led();

}

void BLUE\_YES()

{

while(digitalRead(restpin)==1)

{

color(0, 0, 255);

}

clear\_led();

}

void clear\_led()

{

color(0, 0, 0);

}

void color (unsigned char red, unsigned char green, unsigned char blue) //颜色控制函数

{

analogWrite(redled, red);

analogWrite(greenled,green);

analogWrite(blueled, blue);

}

**测试结果**

下载完程序，上电后，一个简单的抢答器就做好了，我们根据RGB灯显示的颜色判断是谁抢答成功。在复位后。RGB灯关闭。

## 实验六 电位器调控灯光亮度实验

**实验说明**

在第二课程中我们直接通过PWM口控制灯的亮度，从而达到呼吸灯的效果。在这课程中我们通过一个电位器，利用电位器调节PWM值，从而控制灯的亮度。

**实验器材**

开发板\*1

USB线\*1

LED\*1

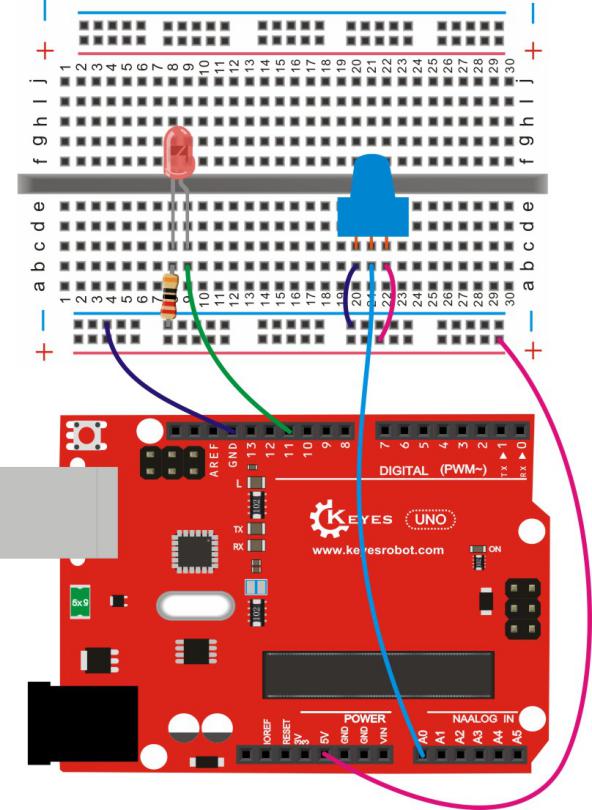
220Ω 电阻\*1

可调电位器\*1

面包板\*1

面包板连接线若干

**接线图**

****

**测试代码**

int ledpin=11;//定义数字接口11（PWM 输出）

void setup()

{

pinMode(ledpin,OUTPUT);//定义数字接口11 为输出

Serial.begin(9600);//设置波特率为9600

}

void loop()

{

int val=analogRead(0);//读取模拟口A0口的值

val = map(val, 0, 1023, 0, 255);//从0-1023映射到0-255

Serial.println(val);//显示val 变量

analogWrite(ledpin,val);// 打开LED 并设置亮度

delay(100);//延时0.1 秒

}

**测试结果**

下载完程序后。我们可以通过旋转可调电位器控制小灯的亮度，打开串口监视器，设置波特率为9600，就可看到调节LED亮度的PWM值。

## 实验七 感光灯实验

**实验说明**

完成以上的各种实验后，我们对Arduino 的应用也应该有一些认识和了解了，在基本的数字量输入输出和模拟量输入以及PWM 的产生都掌握以后，我们就可以开始进行一些传感器的应用了。

本次实验我们先进行一个较为简单的光敏电阻的使用实验。光敏电阻既然是可以根据光强改变阻值的元件，自然也需要模拟口读取模拟值了，本实验可以借鉴电位器调控灯光亮度实验，将电位计换做光敏电阻实现当光强不同时LED 小灯的亮度也会有相应的变化。

**实验器材**

开发板\*1

USB线\*1

LED\*1

330Ω 电阻\*1

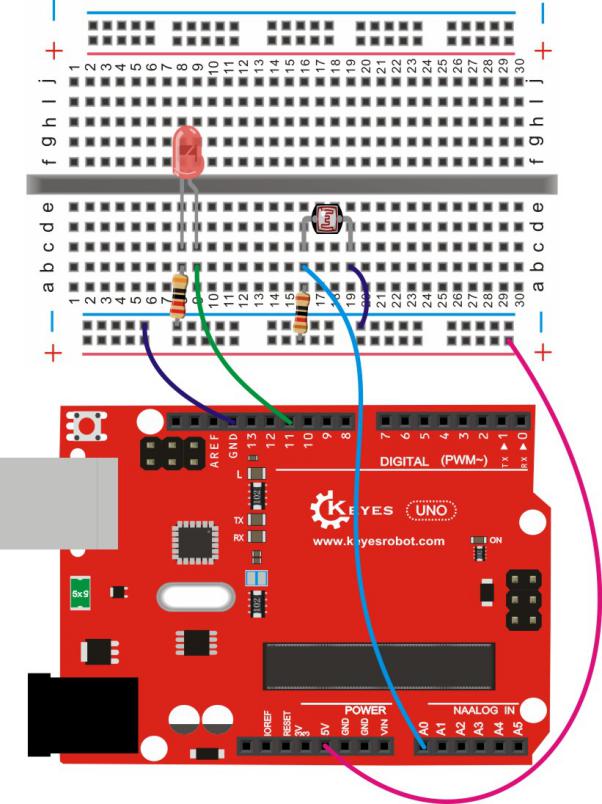
10KΩ 电阻\*1

光敏电阻\*1

面包板\*1

面包板连接线若干

**接线图**

****

**测试代码**

int ledpin=11;//定义数字接口11（PWM 输出）

void setup()

{

pinMode(ledpin,OUTPUT);//定义数字接口11 为输出

Serial.begin(9600);//设置波特率为9600

}

void loop()

{

int val=analogRead(0);//读取模拟口A0口的值

Serial.println(val);//显示val 变量

val = map(val, 0, 1023, 0, 255);//从0-1023映射到0-255

analogWrite(ledpin,255-val);// 打开LED 并设置亮度

delay(10);//延时0.01 秒

}

**测试结果**

下载完程序后，光敏电阻感应到灯光越亮，小灯越暗；光敏电阻感应到灯光越暗，小灯越亮。打开串口监视器，设置波特率为9600，就可看到光敏电阻感应到外界光强所得的模拟值。

## 实验八 有源蜂鸣器实验

**实验说明**

蜂鸣器可分为有源蜂鸣器和无源蜂鸣器两种。本课程中主要用到了有源蜂鸣器，有源蜂鸣器内部有一简单的振荡电路，能将恒定的直流电转化成一定频率的脉冲信号。实验中中我们只需要给蜂鸣器输入一个高电平信号，蜂鸣器响起。

**实验器材**

开发板\*1

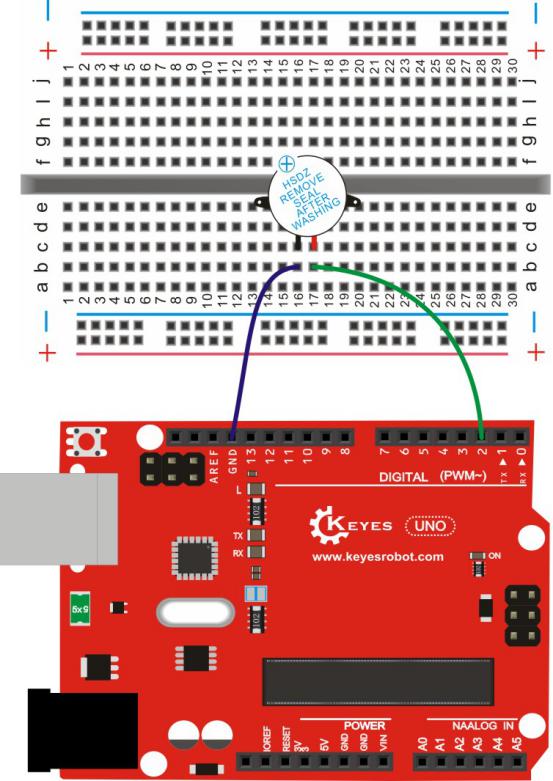
USB线\*1

有源蜂鸣器\*1

面包板\*1

面包板连接线若干

**接线图**

****

**测试代码**

int buzzer = 2; //定义数字口2

void setup()

{

  pinMode(buzzer, OUTPUT);     //设置buzzer为输出

}

void loop()

{

  digitalWrite(buzzer, HIGH);   //开启buzzer

  delay(1000); //延迟1S

  digitalWrite(buzzer, LOW);    //关闭buzzer

  delay(1000);//延迟1S

}

**测试结果**

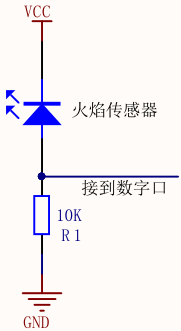
下载完程序后，我们可以听到蜂鸣器响1秒，停止响起1秒，循环交替。

## 实验九 火焰报警实验

**实验说明**

火焰传感器是机器人专门用来搜寻火源的传感器，本传感器对火焰特别灵敏。火焰传感器利用红外线对火焰非常敏感的特点，使用特制的红外线接收管来检测火焰，然后把火焰的亮度转化为高低变化的电平信号。

实验中，我们把火焰的亮度转化为高低变化的电平信号输入到UNO板中，然后控制蜂鸣器的响起。火焰传感器的短引线端为负极，长引线端为正极。按照下图将负极接到5V接口中，然后将正极和10K电阻相连，电阻的另一端接到GND接口中，最后从火焰传感器的正极端所在列接入一根跳线，跳线的另一端接在数字口中。



**实验器材**

开发板\*1

USB线\*1

有源蜂鸣器\*1

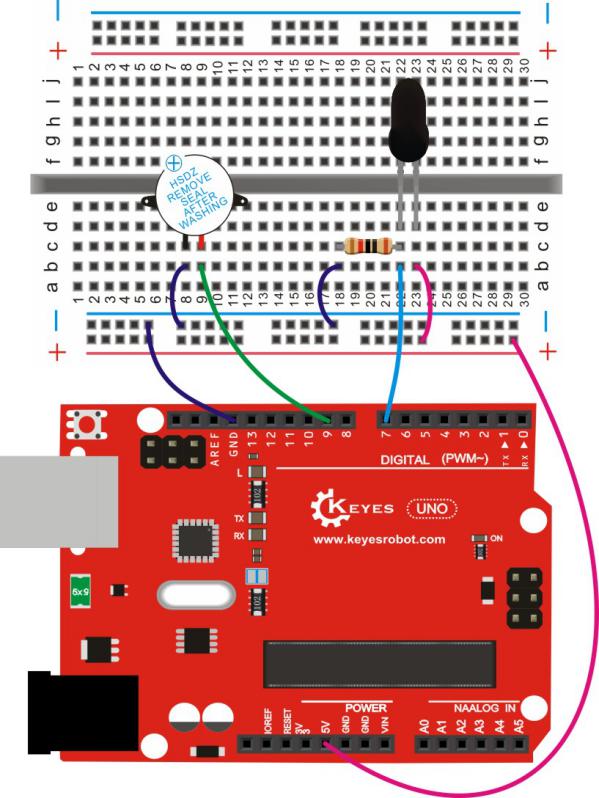
火焰传感器\*1

10KΩ 电阻\*1

面包板\*1

面包板连接线若干

**接线图**

****

**测试代码**

int flame=7;//定义火焰接口为数字7 接口

int Beep=9;//定义蜂鸣器接口为数字9 接口

void setup()

{

pinMode(Beep,OUTPUT);//定义Beep为输出接口

pinMode(flame,INPUT);//定义flame为输入接口

}

void loop()

{

int val=digitalRead(flame);//读取火焰传感器

if(val==HIGH)//当数字口7为高电平时蜂鸣器鸣响

{

digitalWrite(Beep,HIGH);

}else

{

digitalWrite(Beep,LOW);

}

delay(500);

}

**测试结果**

下载完程序后，我们可以模拟在有火焰时报警的情况，在没有火焰时一切正常，当有火焰时立刻报警做出提示。

## 实验十 拨码开关控制LED实验

**实验说明**

拨码开关是很常用且易用的开关元件，相信大家都了解过，这里就不介绍了。本实验中我们用一个拨码开关控制两个LED灯的亮和灭。

**实验器材**

开发板 \*1

USB线\*1

拨码开关\*1

LED\*2

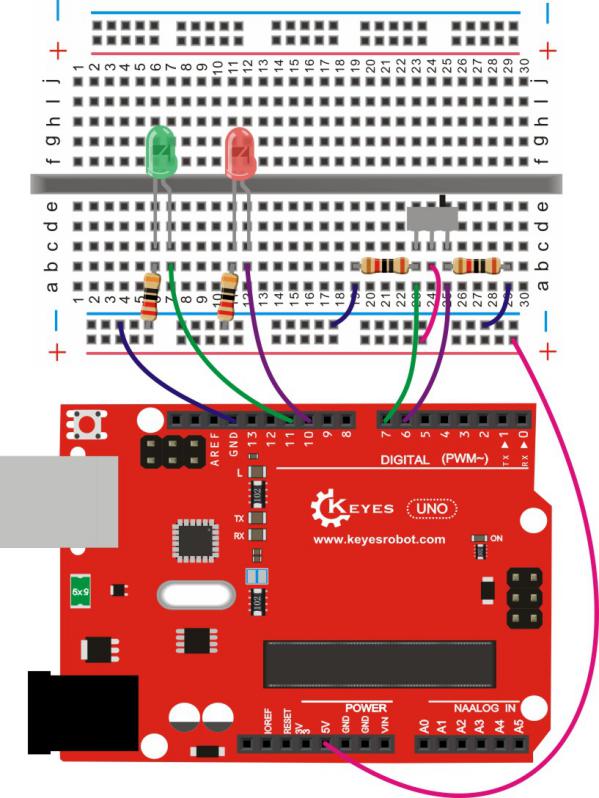
220Ω 电阻\*2

10KΩ 电阻\*2

面包板\*1

面包板连接线若干

**接线图**

****

**测试代码**

int redled = 10; //定义数字口10

int greenled = 11; //定义数字口11

int redpin = 6; //定义数字口6

int greenpin = 7; //定义数字口7

void setup()

{

pinMode(redled,OUTPUT);

pinMode(greenled, OUTPUT);

pinMode(redpin, INPUT);

pinMode(greenpin,INPUT);

}

void loop()

{

int val = digitalRead(redpin);

if (val == HIGH)

{

digitalWrite(redled, HIGH);

}

else

{

digitalWrite(redled,LOW);

}

int val1 = digitalRead(greenpin);

if (val1 == HIGH)

{

digitalWrite(greenled, HIGH);

}

else

{

digitalWrite(greenled,LOW);

}

}

**测试结果**

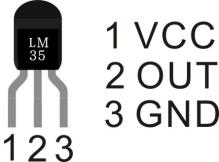
按照接线图接线，上传好代码，上电后，拨码开关拨到左边，绿色LED亮起，红色LED熄灭，拨码开关拨到右边，绿色LED熄灭，红色LED亮起。

## 实验十一 LM35检测温度

**实验说明**

LM35 是很常用且易用的温度传感器元件。实验中我们将LM35 温度传感器接到开发板上，通过算法可将读取的模拟值转换为实际的温度，并在Arduino IDE的串口监视器上显示该温度值。

实验时，需特别注意LM35的方向，如若接反，会把LM35传感器烧毁，接口方向如下。



**实验器材**

开发板 \*1

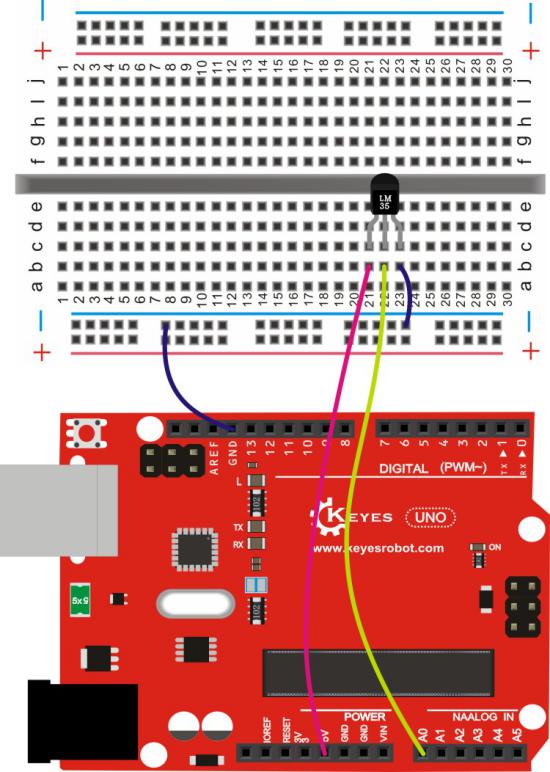
USB线\*1

LM35DZ\*1

面包板\*1

面包板连接线若干

**接线图**

****

**测试代码**

void setup()

{

Serial.begin(9600);//设置波特率

}

void loop()

{

int val; //定义数字变量val

int dat;//定义数字变量dat

val=analogRead(0);//将val设置为读取到的A0的数值

dat=(500 \* val) /1024; //计算出当前温度数字dat

Serial.print("Temp:"); //显示 Temp:

Serial.print(dat); //显示计算的温度值

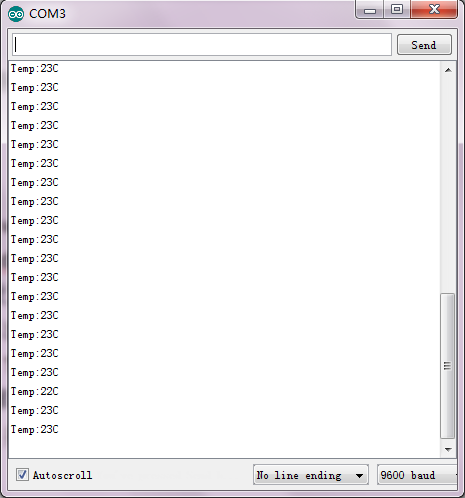
Serial.println("C");//显示C，并自动换行

delay(500); //延迟0.5S

}

**测试结果**

按照上图接好线，上传好代码，上电后，我们可以在软件的串口监视器中看到当前环境中的温度值，如下图。



## 实验十二 魔术光杯实验

**实验说明**

倾斜开关的工作原理是当开关一端低于水平位置倾斜，开关寻通；当另一端低于水平位置倾斜 ，开关停止。魔术光杯实验原理是利用 PWM 调光的原理，两个LED的亮度发生变化。

这个实验中倾斜开关提供数字信号，触发 PWM 的调节，通过程序的设计，我们就能看到类似于两组装满光的杯子倒来倒去的效果了。

**实验器材**

开发板\*1

USB线\*1

LED\*2

倾斜开关\*2

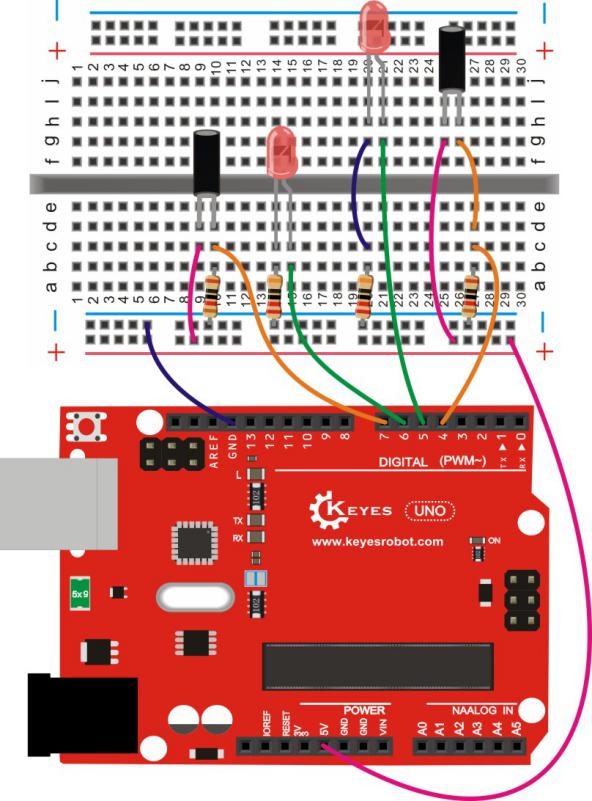
330Ω 电阻\*2

10KΩ 电阻\*2

面包板\*1

面包板连接线若干

**接线图**

****

**测试代码**

int LedPinA = 5; //定义数字口5

int LedPinB = 6; //定义数字口6

int ButtonPinA = 7;//定义数字口7

int ButtonPinB = 4;//定义数字口4

int buttonStateA = 0;

int buttonStateB = 0;

int brightnessA = 0;

int brightnessB= 255;

void setup()

{

Serial.begin(9600);//设置波特率

pinMode(LedPinA, OUTPUT);//数字口5设置为输出

pinMode(LedPinB, OUTPUT);//数字口6设置为输出

pinMode(ButtonPinA, INPUT);//数字口7设置为输入

pinMode(ButtonPinB, INPUT);//数字口4设置为输入

}

void loop()

{

buttonStateA = digitalRead(ButtonPinA);//读取数字口7的数值赋值给buttonStateA

if (buttonStateA == HIGH && brightnessA != 255)

//当buttonStateA为高电平且brightnessA不为255

{

brightnessA ++;//brightnessA加1

delay(10);//延迟0.01S

}

if (buttonStateA == LOW && brightnessA != 0)

//当buttonStateA为低电平且brightnessA不为0

{

brightnessA --;//brightnessA减1

delay(10);//延迟0.01S

}

analogWrite(LedPinB, brightnessA);//将brightnessA赋值为给PWM口6

Serial.print(brightnessA);//显示brightnessA数值

Serial.print(" ");

buttonStateB = digitalRead(ButtonPinB);//读取数字口4的数值赋值给buttonStateB

if (buttonStateB == HIGH && brightnessB != 0)

//当buttonStateB为高电平且brightnessA不为0

{

brightnessB --;//brightnessB减1

delay(10);//延迟0.01S

}

if (buttonStateB == LOW && brightnessB != 255)

//当buttonStateB为低电平且brightnessA不为255

{

brightnessB++;//brightnessB加1

delay(10);//延迟0.01S

}

analogWrite(LedPinA, brightnessB); //将brightnessB赋值为给PWM口5

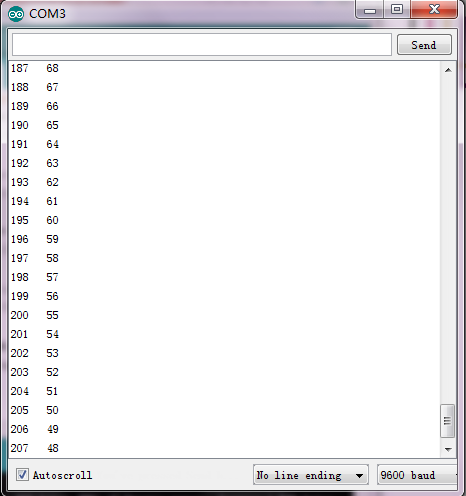
Serial.println(brightnessB);//显示brightnessB数值，并自动换行

delay(5);

}

**测试结果**

按照上图接好线，烧录好代码，上电后，将两个倾斜开关同时倾斜一边， 一个LED逐渐变暗，同时另一个逐渐变亮，最终一个LED完全熄灭，一个LED最亮；在串口监视器中看到对应具体数值变化，如下图。当倾斜另一边中，现象一样，方向相反。



## 实验十三 热敏电阻传感器实验

**实验说明**

热敏电阻能够实时感知周边环境温度的变化，随着温度变化，热敏电阻也发生变化。实验中，我们搭配好电路，把温度变化转换成电压变化，将对应的电压输入到Arduino UNO的模拟口上，并在串口监视器上显示出对应的模拟值。

**实验器材**

开发板\*1

USB线\*1

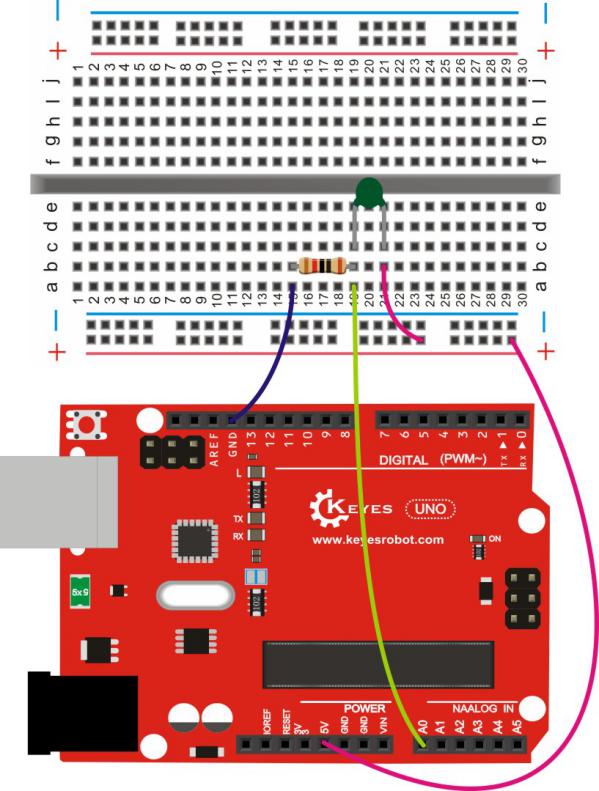
热敏电阻\*1

10KΩ 电阻\*1

面包板\*1

面包板连接线若干

**接线图**

****

**测试代码**

void setup()

{

Serial.begin(9600); //Set serial baud rate to 9600 bps

}

void loop()

{

int val;

val=analogRead(0);//Read rotation sensor value from analog 0

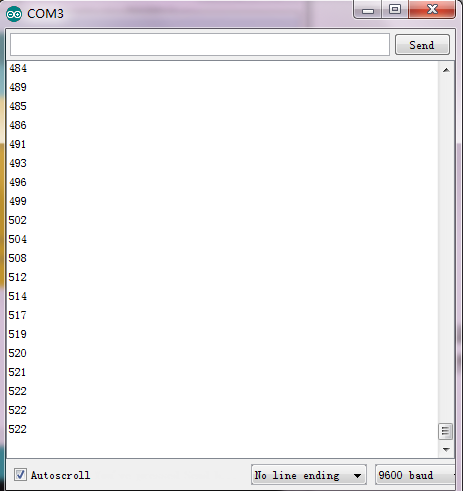
Serial.println(val,DEC);//Print the value to serial port

delay(100);

}

**测试结果**

按照上图接好线，上传好代码，上电后，我们就可以看串口监视器上看到代表当前温度的模拟值。当温度升高，电阻减小，模拟值增大；当人体对准温度电阻呼气时，温度升高，显示如下图。



## 实验十四 一位数码管显示实验

**实验说明**

数码管是一种半导体发光器件，其基本单元是发光二极管。数码管按段数分为七段数码管和八段数码管，八段数码管比七段数码管多一个发光二极管单元（多一个小数点显示），本实验所使用的是八段数码管。数码管共有七段显示数字的段，还有一个显示小数点的段。当让数码管显示数字时，只要将相应的段点亮即可。

**实验器材**

开发板 \*1

USB线\*1

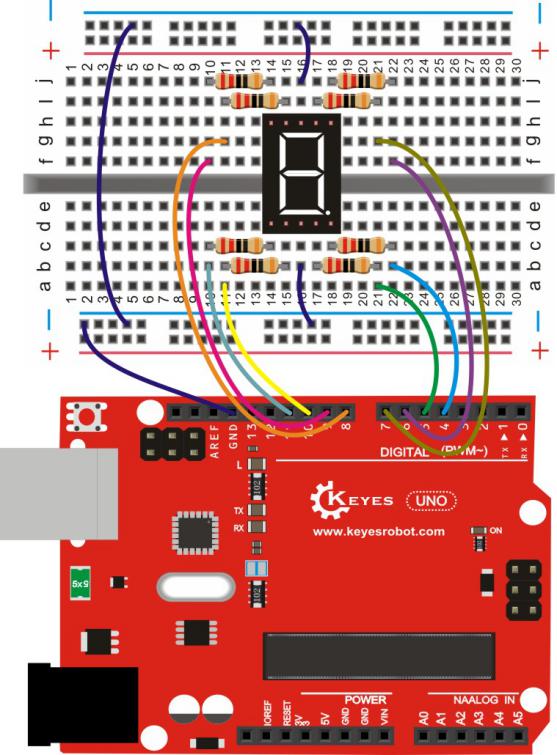
一位数码管\*1

330Ω 电阻\*8

面包板\*1

面包板连接线若干

**接线图**

****

**测试代码**

//设置控制各段的数字IO 脚

int a=7;//定义数字接口7 连接a 段数码管

int b=6;// 定义数字接口6 连接b 段数码管

int c=5;// 定义数字接口5 连接c 段数码管

int d=10;// 定义数字接口11 连接d 段数码管

int e=11;// 定义数字接口10 连接e 段数码管

int f=8;// 定义数字接口8 连接f 段数码管

int g=9;// 定义数字接口9 连接g 段数码管

int dp=4;// 定义数字接口4 连接dp 段数码管

void digital\_1(void) //显示数字1

{

unsigned char j;

digitalWrite(c,HIGH);//给数字接口5 引脚高电平，点亮c 段

digitalWrite(b,HIGH);//点亮b 段

for(j=7;j<=11;j++)//熄灭其余段

digitalWrite(j,LOW);

digitalWrite(dp,LOW);//熄灭小数点DP 段

}

void digital\_2(void) //显示数字2

{

unsigned char j;

digitalWrite(b,HIGH);

digitalWrite(a,HIGH);

for(j=9;j<=11;j++)

digitalWrite(j,HIGH);

digitalWrite(dp,LOW);

digitalWrite(c,LOW);

digitalWrite(f,LOW);

}

void digital\_3(void) //显示数字3

{

unsigned char j;

digitalWrite(g,HIGH);

digitalWrite(d,HIGH);

for(j=5;j<=7;j++)

digitalWrite(j,HIGH);

digitalWrite(dp,LOW);

digitalWrite(f,LOW);

digitalWrite(e,LOW);

}

void digital\_4(void) //显示数字4

{

digitalWrite(c,HIGH);

digitalWrite(b,HIGH);

digitalWrite(f,HIGH);

digitalWrite(g,HIGH);

digitalWrite(dp,LOW);

digitalWrite(a,LOW);

digitalWrite(e,LOW);

digitalWrite(d,LOW);

}

void digital\_5(void) //显示数字5

{

unsigned char j;

for(j=7;j<=9;j++)

digitalWrite(j,HIGH);

digitalWrite(c,HIGH);

digitalWrite(d,HIGH);

digitalWrite(dp,LOW);

digitalWrite(b,LOW);

digitalWrite(e,LOW);

}

void digital\_6(void) //显示数字6

{

unsigned char j;

for(j=7;j<=11;j++)

digitalWrite(j,HIGH);

digitalWrite(c,HIGH);

digitalWrite(dp,LOW);

digitalWrite(b,LOW);

}

void digital\_7(void) //显示数字7

{

unsigned char j;

for(j=5;j<=7;j++)

digitalWrite(j,HIGH);

digitalWrite(dp,LOW);

for(j=8;j<=11;j++)

digitalWrite(j,LOW);

}

void digital\_8(void) //显示数字8

{

unsigned char j;

for(j=5;j<=11;j++)

digitalWrite(j,HIGH);

digitalWrite(dp,LOW);

}

void setup()

{

int i;//定义变量

for(i=4;i<=11;i++)

pinMode(i,OUTPUT);//设置4～11 引脚为输出模式

}

void loop()

{

while(1)

{

digital\_1();//显示数字1

delay(2000);//延时2s

digital\_2();//显示数字2

delay(1000); //延时1s

digital\_3();//显示数字3

delay(1000); //延时1s

digital\_4();//显示数字4

delay(1000); //延时1s

digital\_5();//显示数字5

delay(1000); //延时1s

digital\_6();//显示数字6

delay(1000); //延时1s

digital\_7();//显示数字7

delay(1000); //延时1s

digital\_8();//显示数字8

delay(1000); //延时1s

}

}

**测试结果**

下载完程序后，数码管循环显示1～8 数字。

## 实验十五 8\*8点阵显示实验

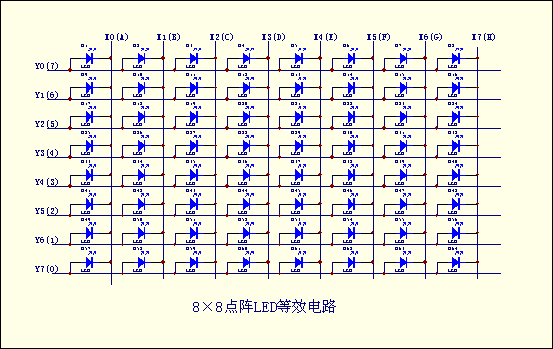
**实验说明**

点阵在我们生活中很常见，很多都有用到他，比如LED广告显示屏，电梯显示楼层，公交车报站等等。

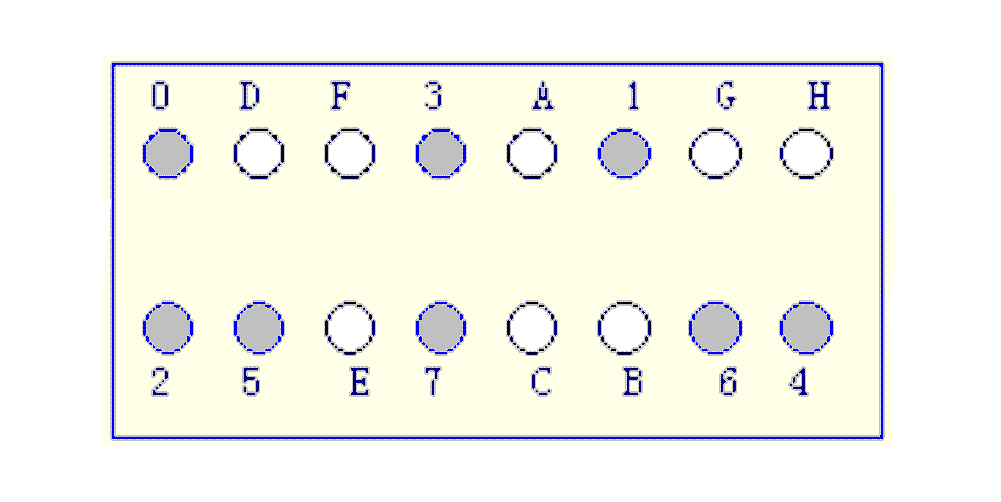
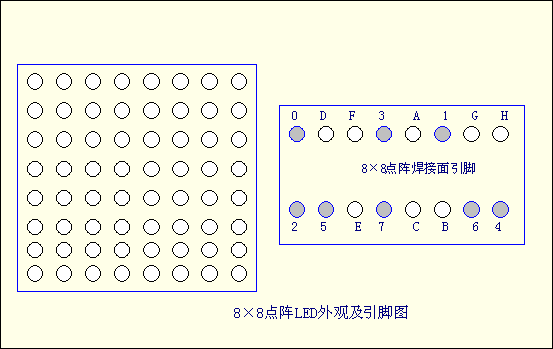
8\*8点阵共由64个发光二极管组成，且每个发光二极管是放置在行线和列线的交叉点上，当对应的某一行置高电平，某一列置低电平，则相应的二极管就亮；如要将第一个点点亮，则7脚接高电平A脚接低电平，则第一个点就亮了；如果要将第一行点亮，则第7脚要接高电平，而A、B、C、D、E、F、G、H这些引脚接低电平，那么第一行就会点亮；如要将第一列点亮，则第A脚接低电平，而0、1、2、3、4、5、6、7接高电平，那么第一列就会点亮。

在本课程中，我们只是让点阵输出一个“0”。

8\*8点阵原理图



8\*8点阵实物图



**实验器材**

开发板\*1

USB线\*1

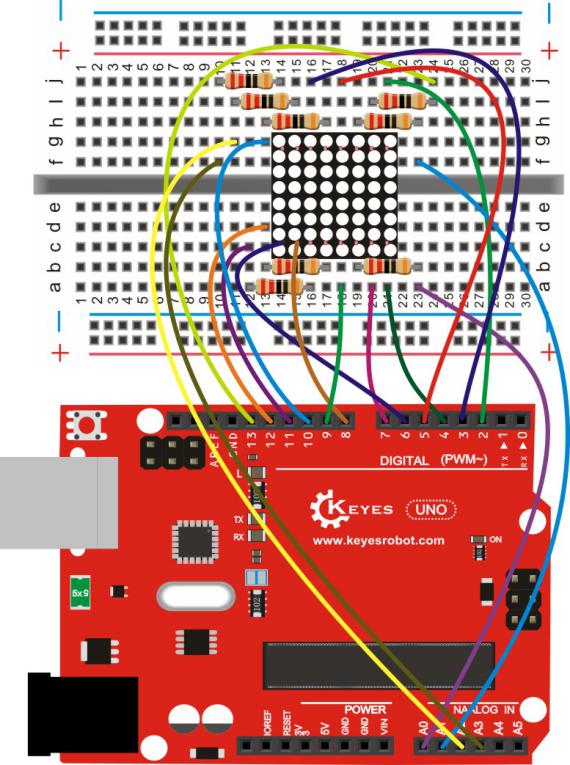
8\*8点阵\*1

220Ω 电阻\*8

面包板\*1

面包板连接线若干

**接线图**

****

**测试代码**

//定义了一个数组，用来存放“0”字的字模

unsigned char Text[]={0x00,0x1c,0x22,0x22,0x22,0x22,0x22,0x1c};

void Draw\_point(unsigned char x,unsigned char y)//画点函数

{

clear\_();

digitalWrite(x+2, HIGH);

digitalWrite(y+10, LOW);

delay(1);

}

void show\_num(void)//显示函数，最终还是调用了画点函数。

{

unsigned char i,j,data;

for(i=0;i<8;i++)

{

data=Text[i];

for(j=0;j<8;j++)

{

if(data & 0x01)Draw\_point(j,i);

data>>=1;

}

}

}

void setup(){

int i = 0 ;

for(i=2;i<18;i++)

{

pinMode(i, OUTPUT);

}

clear\_();

}

void loop()

{

show\_num();

}

void clear\_(void)//清除屏幕

{

for(int i=2;i<10;i++)

digitalWrite(i, LOW);

for(int i=0;i<8;i++)

digitalWrite(i+10, HIGH);

}

**测试结果**

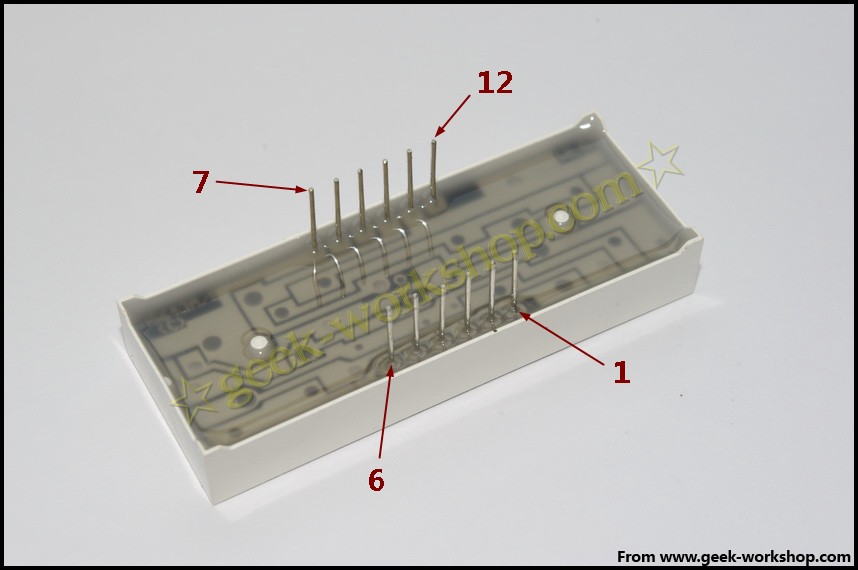
下载完程序后，点阵上显示数字“0”。

## 实验十六 四位数码管显示数字实验

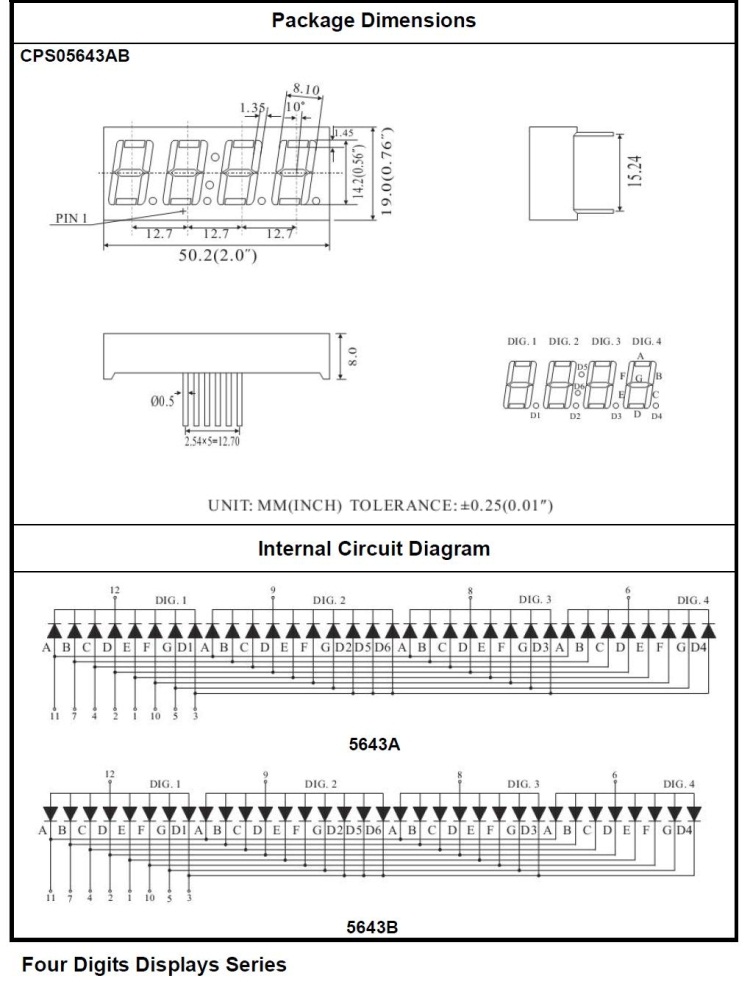
**实验说明**

在实验十五中我们使用开发板驱动一个一位数码管，本实验我们使用开发板驱动一个共阴四位数码管。驱动数码管限流电阻肯定是必不可少的，限流电阻有两种接法，一种是在d1-d4阴极接，总共接4颗。这种接法好处是需求电阻比较少，但是会产生每一位上显示不同数字亮度会不一样，1最亮，8最暗。另外一种接法就是在其他8个引脚上接，这种接法亮度显示均匀，但是用电阻较多。本次实验使用8颗220Ω电阻。

四位数码管总共有12个引脚，小数点朝下正放在面前时，左下角为1,其他管脚顺序为逆时针旋转。左上角为最大的12号管脚。



四位数码管原理图如下



**实验器材**

开发板\*1

USB线\*1

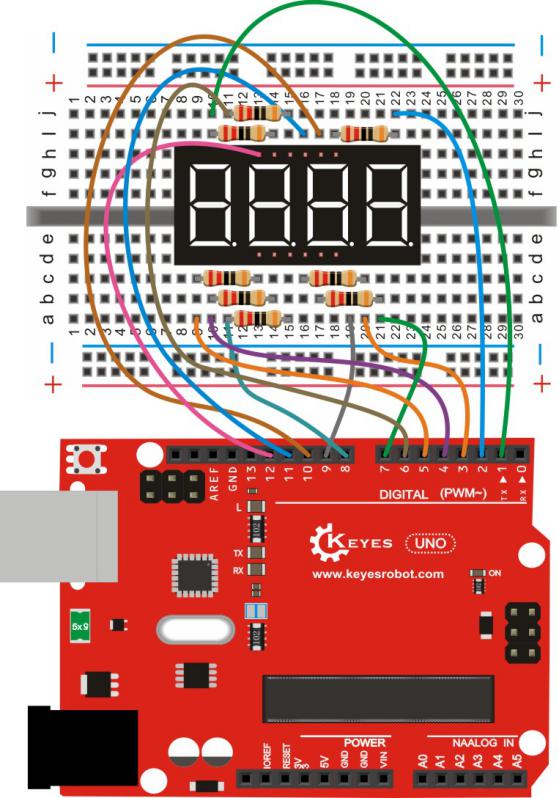
四位数码管\*1

330Ω 电阻\*8

面包板\*1

面包板连接线若干

**接线图**

****

**测试代码**

int a = 1;

int b = 2;

int c = 3;

int d = 4;

int e = 5;

int f = 6;

int g = 7;

int dp = 8;

int d4 = 9;

int d3 = 10;

int d2 = 11;

int d1 = 12;

// set variable

long n = 1230;

int x = 100;

int del = 55; // fine adjustment for clock

void setup()

{

pinMode(d1, OUTPUT);

pinMode(d2, OUTPUT);

pinMode(d3, OUTPUT);

pinMode(d4, OUTPUT);

pinMode(a, OUTPUT);

pinMode(b, OUTPUT);

pinMode(c, OUTPUT);

pinMode(d, OUTPUT);

pinMode(e, OUTPUT);

pinMode(f, OUTPUT);

pinMode(g, OUTPUT);

pinMode(dp, OUTPUT);

}

/////////////////////////////////////////////////////////////

void loop()

{

int a=0;

int b=0;

int c=0;

int d=0;

unsigned long currentMillis = millis();

while(d>=0)

{

while(millis()-currentMillis<1000)

{

Display(1,a);

Display(2,b);

Display(3,c);

Display(4,d);

}

currentMillis = millis();

d++;

if (d>9)

{

c++;

d=0;

}

if (c>9)

{

b++;

c=0;

}

if (b>9)

{

a++;

b=0;

}

if (a>9)

{

a=0;

b=0;

c=0;

d=0;

}

}

}

///////////////////////////////////////////////////////////////

void WeiXuan(unsigned char n)//

{

switch (n)

{

case 1:

digitalWrite(d1, LOW);

digitalWrite(d2, HIGH);

digitalWrite(d3, HIGH);

digitalWrite(d4, HIGH);

break;

case 2:

digitalWrite(d1, HIGH);

digitalWrite(d2, LOW);

digitalWrite(d3, HIGH);

digitalWrite(d4, HIGH);

break;

case 3:

digitalWrite(d1, HIGH);

digitalWrite(d2, HIGH);

digitalWrite(d3, LOW);

digitalWrite(d4, HIGH);

break;

case 4:

digitalWrite(d1, HIGH);

digitalWrite(d2, HIGH);

digitalWrite(d3, HIGH);

digitalWrite(d4, LOW);

break;

default :

digitalWrite(d1, HIGH);

digitalWrite(d2, HIGH);

digitalWrite(d3, HIGH);

digitalWrite(d4, HIGH);

break;

}

}

void Num\_0()

{

digitalWrite(a, HIGH);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, HIGH);

digitalWrite(e, HIGH);

digitalWrite(f, HIGH);

digitalWrite(g, LOW);

digitalWrite(dp, LOW);

}

void Num\_1()

{

digitalWrite(a, LOW);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, LOW);

digitalWrite(e, LOW);

digitalWrite(f, LOW);

digitalWrite(g, LOW);

digitalWrite(dp, LOW);

}

void Num\_2()

{

digitalWrite(a, HIGH);

digitalWrite(b, HIGH);

digitalWrite(c, LOW);

digitalWrite(d, HIGH);

digitalWrite(e, HIGH);

digitalWrite(f, LOW);

digitalWrite(g, HIGH);

digitalWrite(dp, LOW);

}

void Num\_3()

{

digitalWrite(a, HIGH);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, HIGH);

digitalWrite(e, LOW);

digitalWrite(f, LOW);

digitalWrite(g, HIGH);

digitalWrite(dp, LOW);

}

void Num\_4()

{

digitalWrite(a, LOW);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, LOW);

digitalWrite(e, LOW);

digitalWrite(f, HIGH);

digitalWrite(g, HIGH);

digitalWrite(dp, LOW);

}

void Num\_5()

{

digitalWrite(a, HIGH);

digitalWrite(b, LOW);

digitalWrite(c, HIGH);

digitalWrite(d, HIGH);

digitalWrite(e, LOW);

digitalWrite(f, HIGH);

digitalWrite(g, HIGH);

digitalWrite(dp, LOW);

}

void Num\_6()

{

digitalWrite(a, HIGH);

digitalWrite(b, LOW);

digitalWrite(c, HIGH);

digitalWrite(d, HIGH);

digitalWrite(e, HIGH);

digitalWrite(f, HIGH);

digitalWrite(g, HIGH);

digitalWrite(dp, LOW);

}

void Num\_7()

{

digitalWrite(a, HIGH);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, LOW);

digitalWrite(e, LOW);

digitalWrite(f, LOW);

digitalWrite(g, LOW);

digitalWrite(dp, LOW);

}

void Num\_8()

{

digitalWrite(a, HIGH);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, HIGH);

digitalWrite(e, HIGH);

digitalWrite(f, HIGH);

digitalWrite(g, HIGH);

digitalWrite(dp, LOW);

}

void Num\_9()

{

digitalWrite(a, HIGH);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, HIGH);

digitalWrite(e, LOW);

digitalWrite(f, HIGH);

digitalWrite(g, HIGH);

digitalWrite(dp, LOW);

}

void Clear() // clear the screen

{

digitalWrite(a, LOW);

digitalWrite(b, LOW);

digitalWrite(c, LOW);

digitalWrite(d, LOW);

digitalWrite(e, LOW);

digitalWrite(f, LOW);

digitalWrite(g, LOW);

digitalWrite(dp, LOW);

}

void pickNumber(unsigned char n)// select number

{

switch (n)

{

case 0: Num\_0();

break;

case 1: Num\_1();

break;

case 2: Num\_2();

break;

case 3: Num\_3();

break;

case 4: Num\_4();

break;

case 5: Num\_5();

break;

case 6: Num\_6();

break;

case 7: Num\_7();

break;

case 8: Num\_8();

break;

case 9: Num\_9();

break;

default: Clear();

break;

}

}

void Display(unsigned char x, unsigned char Number)// take x as coordinate and display number

{

WeiXuan(x);

pickNumber(Number);

delay(1);

Clear() ; // clear the screen

}

**测试结果**

下载完程序后，数码管首先显示“0000”数值，显示跳动，每跳动一下数码管显示数值加1。当显示数值为超过“9999”后，显示数值再次变为“0000”，循环显示。

## 实验十七 舵机控制实验

**实验说明**

舵机是一种位置伺服的驱动器，主要是由外壳、电路板、无核心马达、齿轮与位置检测

器所构成。舵机有很多规格，但所有的舵机都有外接三根线，分别用棕、红、橙三种颜

色进行区分，由于舵机品牌不同，颜色也会有所差异，棕色为接地线，红色为电源正极

线，橙色为信号线。



舵机的转动的角度是通过调节PWM（脉冲宽度调制）信号的占空比来实现的，标准PWM

（脉冲宽度调制）信号的周期固定为20ms（50Hz），理论上脉宽分布应在1ms到2ms

之间，但是，事实上脉宽可由0.5ms 到2.5ms 之间，脉宽和舵机的转角0°～180°相

对应。有一点值得注意的地方，由于舵机牌子不同，对于同一信号，不同牌子的舵机旋

转的角度也会有所不同。



**实验器材**

开发板\*1

USB线\*1

舵机\*1

面包线若干

**接线图**

****

**测试代码**

程序A：

int servopin=9;//定义数字接口9 连接伺服舵机信号线

int myangle;//定义角度变量

int pulsewidth;//定义脉宽变量

int val;

void servopulse(int servopin,int myangle)//定义一个脉冲函数

{

pulsewidth=(myangle\*11)+500;//将角度转化为500-2480 的脉宽值

digitalWrite(servopin,HIGH);//将舵机接口电平至高

delayMicroseconds(pulsewidth);//延时脉宽值的微秒数

digitalWrite(servopin,LOW);//将舵机接口电平至低

delay(20-pulsewidth/1000);

}

void setup()

{

pinMode(servopin,OUTPUT);//设定舵机接口为输出接口

Serial.begin(9600);//连接到串行端口，波特率为9600

Serial.println("servo=o\_seral\_simple ready" ) ;

}

void loop()//将0 到9 的数转化为0 到180 角度，并让LED 闪烁相应数的次数

{

val=Serial.read();//读取串行端口的值

if(val>='0'&&val<='9')

{

val=val-'0';//将特征量转化为数值变量

val=val\*(180/9);//将数字转化为角度

Serial.print("moving servo to ");

Serial.print(val,DEC);

Serial.println();

for(int i=0;i<=50;i++) //给予舵机足够的时间让它转到指定角度

{

servopulse(servopin,val);//引用脉冲函数

}

}

}

程序B：

#include <Servo.h>

Servo myservo;//定义舵机变量名

void setup()

{

myservo.attach(9);//定义舵机接口（9、10 都可以，缺点只能控制2 个）

}

void loop()

{

myservo.write(90);//设置舵机旋转的角度

}

注意：在上传程序前，要把Servo文件夹放到 编译器安装目录下的

\Arduino\libraries里。不然编译不过。

例如我的：C:\Program Files\Arduino\libraries

**测试结果**

程序A 结果：

在串口监视器中输入数字点击发送，舵机转动到所对应的角度数的位置，并将角度打印显示到屏幕上。

程序B结果：

舵机自己转动到90度位置。

## 实验十八 震动传感器实验

**实验说明**

实验中我们主要用SW-200D 振动开关做一个震动传感器。SW-200D 振动开关也是个倾斜开关，我们用它控制一个LED灯的亮灭，当开关检测到震动时，开关内的滚珠左右摇摆，从而控制LED灯一闪一灭。实验时我们只需要观察外接LED灯的状态就可知道是否有震动。

**实验器材**

开发板 \*1

USB线\*1

LED\*1

SW-200D 振动开关\*1

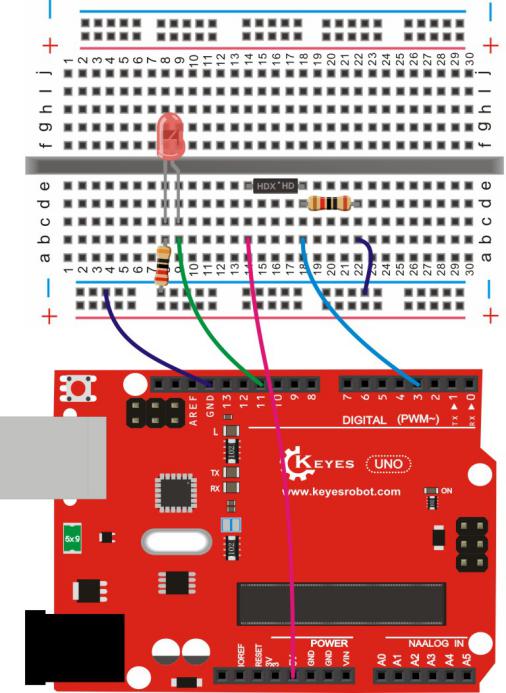
330Ω 电阻\*1

10KΩ 电阻\*1

面包板\*1

面包板连接线若干

**接线图**

****

**测试代码**

int ledPin = 11; //定义数字口11

int inputPin = 3; //定义数字口3

void setup()

{

pinMode(ledPin, OUTPUT); //将ledPin设置为输出

pinMode(inputPin, INPUT); //将inputPin设置为输入

}

void loop()

{

int val = digitalRead(inputPin);

//设置数字变量val，读取到数字口3的数值，并赋值给 val

if (val == LOW) //当val为低电平时，LED变暗

{

digitalWrite(ledPin, LOW); // LED变暗

}

else

{

digitalWrite(ledPin, HIGH); // LED亮起

}

}

**测试结果**

下载完程序，上电后，当检测到震动时，外接LED灯无规则的亮灭。

# 相关资料链接

<https://pan.baidu.com/s/1pMjQ6Pd>