

# Arduino 基础入门套件



保科技

# Arduino介绍篇

## 概 述

### 什么是 Arduino?

Arduino 是一块基于开放原始代码的 Simple i/o 平台，并且具有开发语言和开发环境都很简单、易理解的特点。让您可以快速使用 Arduino 做出有趣的东西。

它是一个能够用来感应和控制现实物理世界的一套工具。 它由一个基于单片机并且开放源码的硬件平台，和一套为 Arduino 板编写程序 的开发环境组成。

Arduino 可以用来开发交互产品，比如它可以读取大量的开关和传感器信号，并且可以控制各式各样的电灯、电机和其他物理设备。Arduino 项目可以是单独的，也可以在运行时和你电脑中运行的程序（例如：Flash, Processing, MaxMSP）进行通讯。Arduino 开源的 IDE 可以免费下载得到。

### 特色描述

- 开放原始码的电路图设计，开发界面免费下载，也可依需求自己修改!!
- 下载程序简单、方便。
- 可简单地与传感器、各式各样的电子元件连接（如：LED 灯、蜂鸣器、按键、光敏电阻等等），做出各种各样有趣的东西。
- 使用高速的微处理控制器(ATMEGA328)。
- 开发语言和开发环境都非常的简单、易理解，非常适合初学者学习。

### 性能描述

- Digital I/O 数字输入/输出端口 0—13。
- Analog I/O 模拟输入/输出端口 0-5。
- 支持 ISP 下载功能。
- 输入电压：接上 USB 时无须外部供电或外部 5V~9V 直流电压输入。
- 输出电压：5V 直流电压输出和 3.3V 直流电压输出和外部电源输入。
- 采用 Atmel Atmega328 微处理控制器。因其支持者众多，已有公司开发出来 32 位的 MCU 平台支持 arduino。
- Arduino 大小尺寸：宽 70mm X 高 54mm。

### 几个比较特殊的端口说明：

**VIN 端口：**VIN 是 input voltage 的缩写，表示有外部电源时的输入端口。如果不使用 USB 供电时，外接电源可以通过此引脚提供电压。（如电池供电，电池正构接 VIN 端口，负构接 GND 端口）。

**AREF:** Reference voltage for the analog inputs (模拟输入的基准电压)。使用 analogReference() 命令调用。

## Arduino C语法介绍

Arduino 语法是建立在 C/C++基础上的，其实也就是基础的 C 语法，Arduino 语法只不过把相关的一些参数设置都函数化，不用我们去了解他的底层，让我们去了解 AVR 单片机（微控制器）的朋友也能轻松上手。那么这里我就简单的注释一下 Arduino 语法。

关键字：

- **if**
- **if...else**
- **for**
- **switch case**
- **while**
- **do... while**
- **break**
- **continue**
- **return**
- **goto**

语法符号：

- **;**
- **{ }**
- **//**
- **/\* \*/**

运算符：

- **=**
- **+**
- **-**
- **\***
- **/**
- **%**
- **==**
- **!=**
- **<**
- **>**
- **<=**
- **>=**
- **&&**
- **||**
- **!**

- ++
- --
- +=
- -=
- \*=
- /=

数据类型：

- **boolean** 布尔类型
- **char** 字符类型
- **byte** 字节类型
- **int** 整数类型
- **unsigned int** 无符号整型
- **long** 长整型
- **unsigned long** 无符号长整型
- **float** 实数类型
- **double**
- **string**
- **array**
- **void**

常量：

- **HIGH | LOW** 表示数字 IO 口的电平，**HIGH** 表示高电平（1），**LOW** 表示低电平（0）。
- **INPUT | OUTPUT** 表示数字 IO 口的方向，**INPUT** 表示输入（高阻态），**OUTPUT** 表示

输出（AVR 能提供 5V 电压 40mA 电流）。

- **true | false** true 表示真（1），false 表示假（0）。

以上为基础 c 语法的关键字和符号，大家可以了解，具体使用可以结合实验的程序。

## 结构

void setup() 初始化变量，管脚模式，调用库函数等  
void loop() 连续执行函数内的语句

## 功能

### 数字 I/O

- **pinMode(pin, mode)** 数字 IO 口输入输出模式定义函数，pin 表示为 0~13，mode 表示为 INPUT 或 OUTPUT。
- **digitalWrite(pin, value)** 数字 IO 口输出电平定义函数，pin 表示为 0~13，value 表示为 HIGH 或 LOW。比如定义 HIGH 可以驱动 LED。
- **int digitalRead(pin)** 数字 IO 口读输入电平函数，pin 表示为 0~13，value 表示为 HIGH 或 LOW。比如可以读数字传感器。

### 模拟 I/O

- **int analogRead(pin)** 模拟 IO 口读函数，pin 表示为 0~5（Arduino Diecimila 为 0~5，Arduino nano 为 0~7）。比如可以读模拟传感器（10 位 AD，0~5V 表示为 0~1023）。
- **analogWrite(pin, value)** PWM 数字 IO 口 PWM 输出函数，Arduino 数字 IO 口标注了 PWM 的 IO 口可使用该函数，pin 表示 3, 5, 6, 9, 10, 11，value 表示为 0~255。比如可用于电机 PWM 调速或音乐播放。

### 时间函数

- **delay(ms)** 延时函数（单位 ms）。
- **delayMicroseconds(us)** 延时函数（单位 us）。

### 数学函数

- **min(x, y)** 求最小值
  - **max(x, y)** 求最大值
  - **abs(x)** 计算绝对值
  - **constrain(x, a, b)** 约束函数，下限 a，上限 b，x 必须在 ab 之间才能返回。
  - **map(value, fromLow, fromHigh, toLow, toHigh)** 约束函数，value 必须在 fromLow 与 toLow 之间和 fromHigh 与 toHigh 之间。
  - **pow(base, exponent)** 开方函数，base 的 exponent 次方。
  - **sq(x)** 平方
  - **sqrt(x)** 开根号
-

## Arduino入门教程--课前准备

Arduino 控制板拿到手后，首先需要在电脑上把驱动装上，这样才可以进行各种实验。

### 第一步, 下载arduino开发环境

( 需要把 Arduino IDE 下载回来 (IDE 就是 arduino 的软件程序开发环境) )

点击下面链接进行下载:

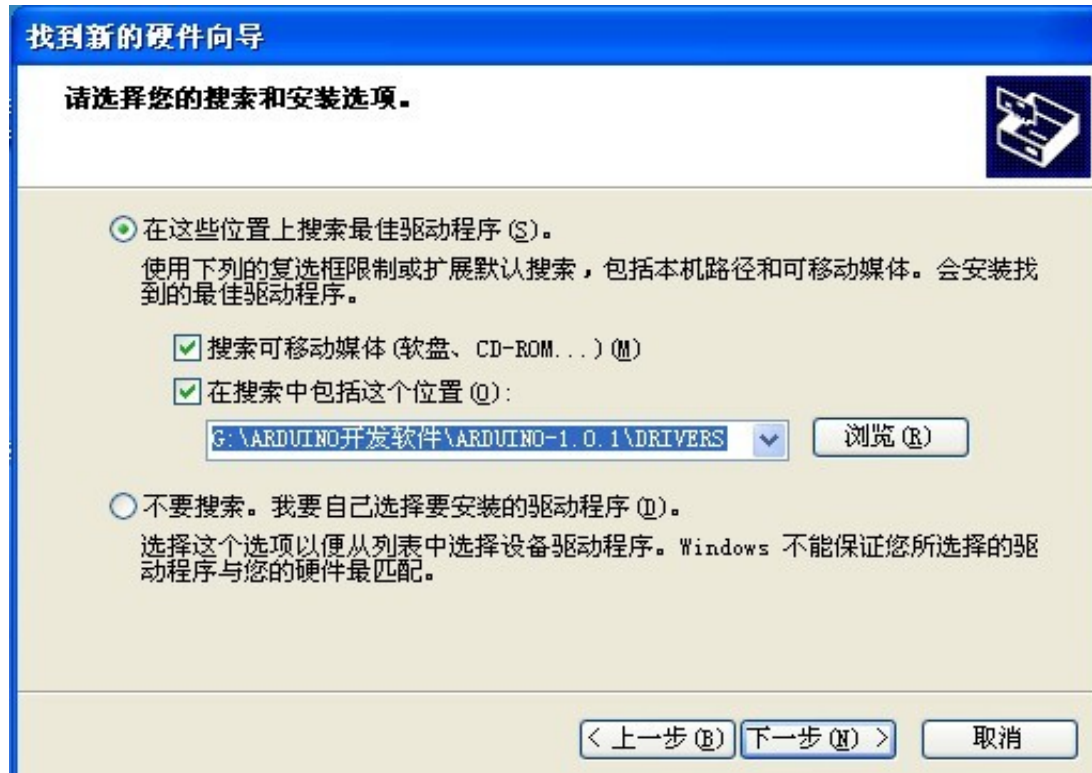
- [Windows](#)
- [Mac OS X](#)
- Linux : [32bit](#) [64bit](#)

### 第二步, 安装

下载后自己找地方解压缩放好就行了

然后将买回来的 arduino 拆了包装, 使用方头 usb 数据线连接到电脑, 本文以 win xp 位为例, 会出现识别到新硬件显示。







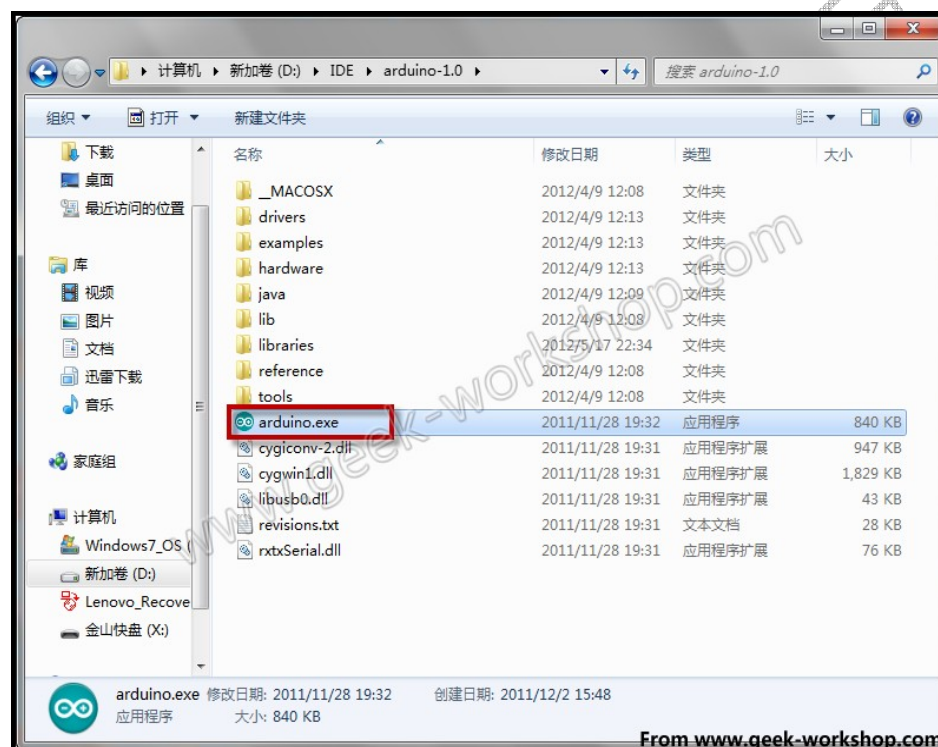
然后点击设备管理



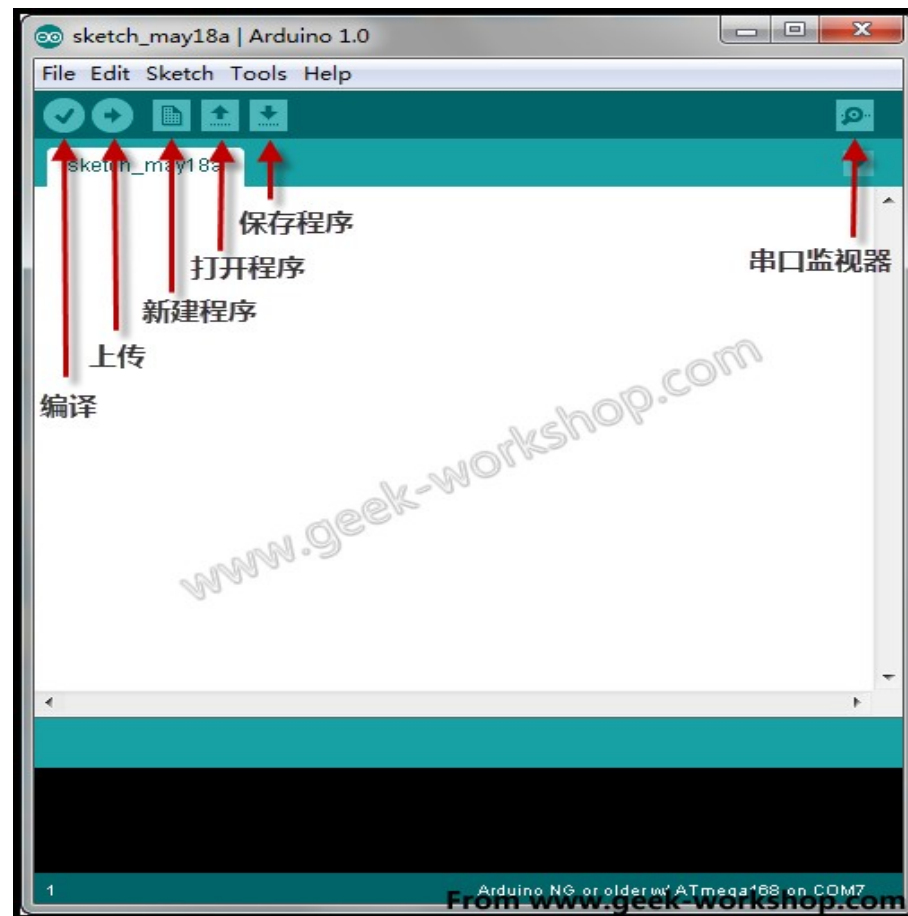
安装好驱动后，就可以显示是 COM 几端口，这里我的显示的是 COM4 端口。

### 第三步，软件的使用

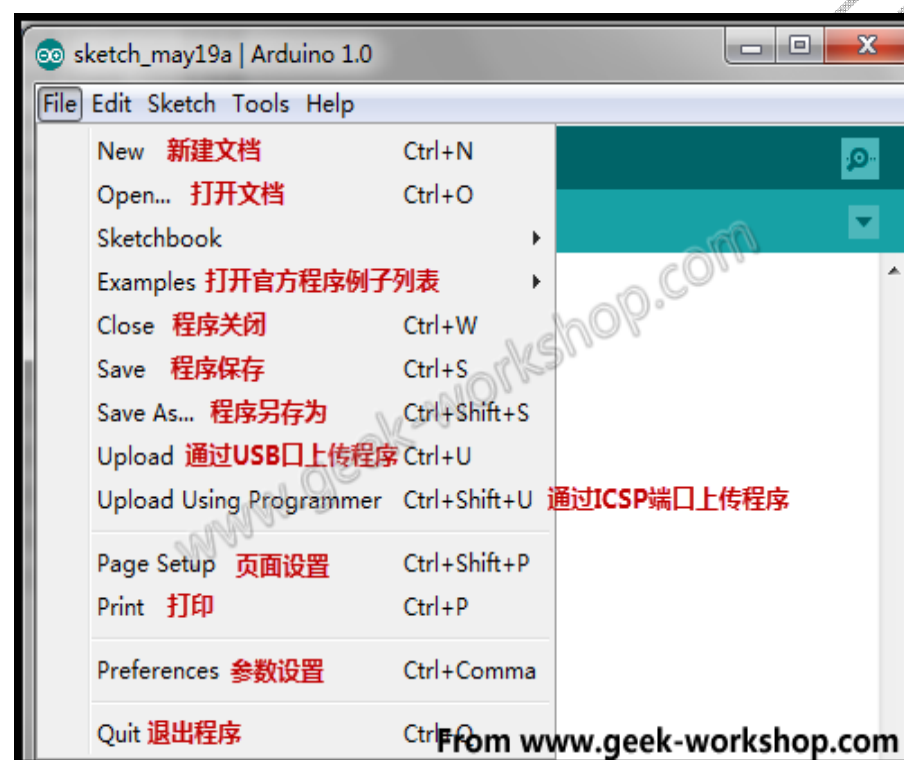
接着我们介绍下 Arduino 的 IDE 界面，以 Arduino 1.0 IDE 为例，首先进入软件目录。然后就可以看到 arduino.exe 文件，双击打开 IDE。



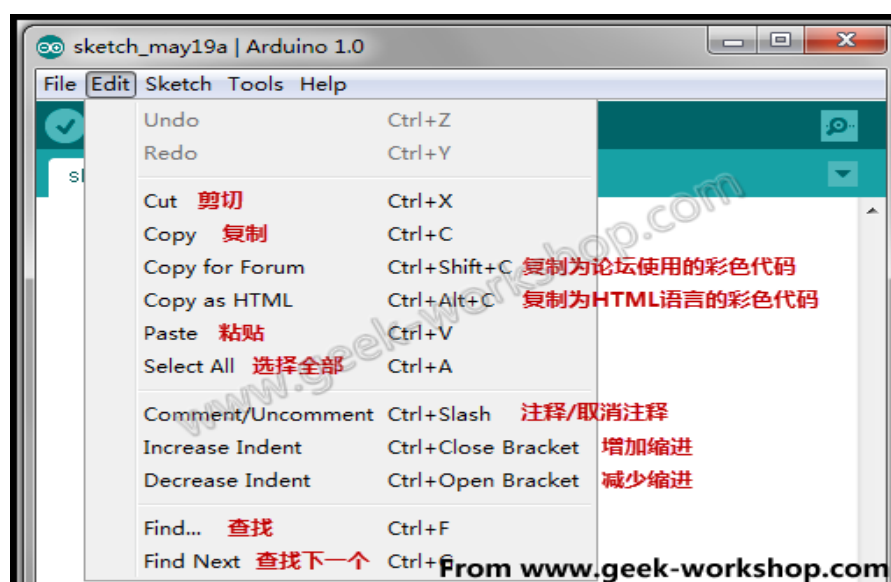




菜单栏上，首先看 File 菜单~介绍如下：



接下来看 Edit 菜单~介绍如下:



Sketch 菜单介绍



Tools 菜单介绍



## 第一章 Hello World!

这一章，我们简单学习一下利用 Arduino IDE 的串口工具，在电脑中显示我们想要显示的内容。

实例代码：

```
void setup()
{
  Serial.begin(9600); // opens serial port, sets data rate to 9600 bps
  Serial.println("Hello World!");
}

void loop()
{
}

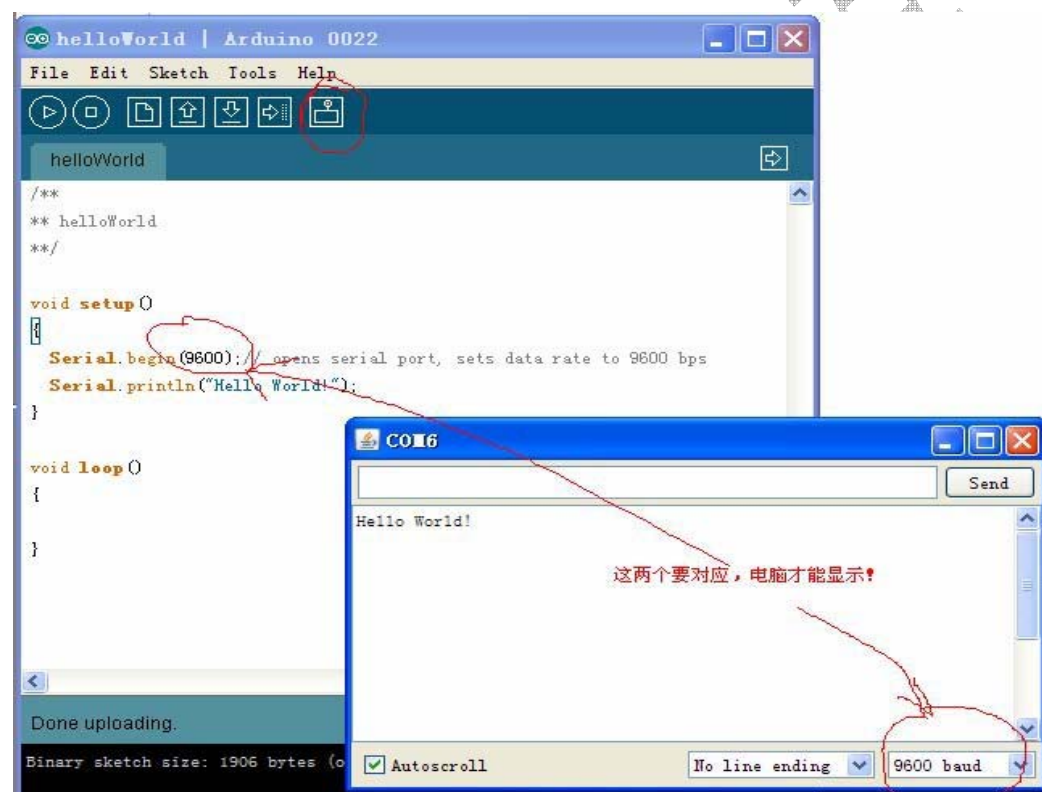
}
```

说明：

`Serial.begin(9600);` 这个函数是为串口数据传输设置每秒数据传输速率，每秒多少位数（波特率）。为了能与计算机进行通信，可选择使用以下这些波特率：“300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 或 115200”。

实验结果与操作：

- 1) 把代码下载到 arduino 控制板。
- 2) 下载成功后，“”先从选项“tool”，选择相应的 arduino 控制板，和对应的“com”口。打开串口工具，在新打开的串口工具窗口的“右下角”选择相应的波特率。

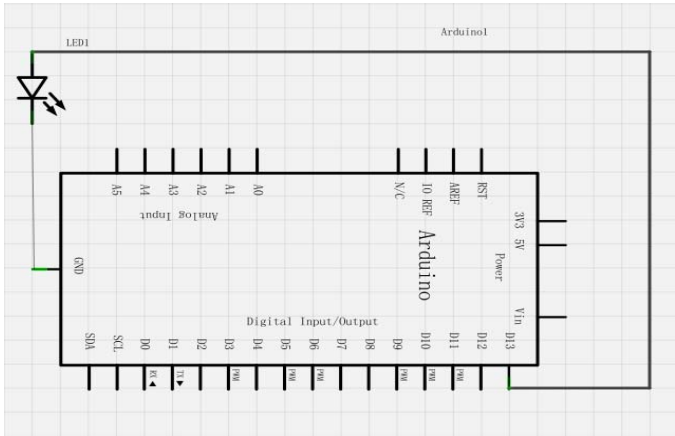


## 第二章 板载LED 闪烁实验

### 板载LED 闪烁实验

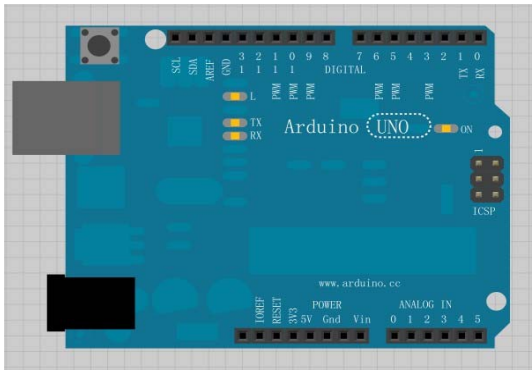
LED 小灯实验是比较基础的实验之一，这次我们利用主板上自带的13脚的LED灯来完成这个实验，我们需要的实验器材除了每个实验都必须的Arduino 控制器和USB 下载线以外，其他的都不用。  
下一步我们按照下面的小灯实验原理图链接实物图，

小灯实验原理图



实物图

实物图



按照上图链接好电路后，就可以开始编写程序了，我们还是让LED 小灯闪烁，点亮1 秒熄灭1 秒。这个程序很简单，就是Arduino 自带的例程里的Blink参考程序如下：

```
int ledPin = 13; //定义数字13 接口
void setup()
{
  pinMode(ledPin, OUTPUT); //定义小灯接口为输出接口
}
void loop()
{
  digitalWrite(ledPin, HIGH); //点亮小灯
  delay(1000); //延时1 秒
  digitalWrite(ledPin, LOW); //熄灭小灯
  delay(1000); // 延时1 秒
}
```

下载完程序就可以看到我们的13脚LED灯在闪烁了，这样我们的小灯闪烁实验就完成了。

## 第三章 LED闪烁实验

### 一、发光二极管介绍

#### 1、什么是发光二极管

发光二极管简称为 LED。由镓 (Ga) 与砷 (AS)、磷 (P) 的化合物制成的二极管，当电子与空穴复合时能辐射出可见光，因而可以用来制成发光二极管，在电路及仪器中作为指示灯，或者组成文字或数字显示。磷砷化镓二极管发红光，磷化镓二极管发绿光，碳化硅二极管发黄光。



它是半导体二极管的一种，可以把电能转化成光能；常简称为 LED。发光二极管与普通二极管一样是由一个 PN 结组成，也具有单向导电性。当给发光二极管加上正向电压后，从 P 区注入到 N 区的空穴和由 N 区注入到 P 区的电子，在 PN 结附近数微米内分别与 N 区的电子和 P 区的空穴复合，产生自发辐射的荧光。不同的半导体材料中电子和空穴所处的能量状态不同。当电子和空穴复合时释放出的能量多少不同，释放出的能量越多，则发出的光的波长越短。常用的是发红光、绿光或黄光的二极管。

#### 2、工作原理

发光二极管的反向击穿电压约 5 伏。它的正向伏安特性曲线很陡，使用时必须串联限流电阻以控制通过管子的电流。限流电阻 R 可用下式计算：

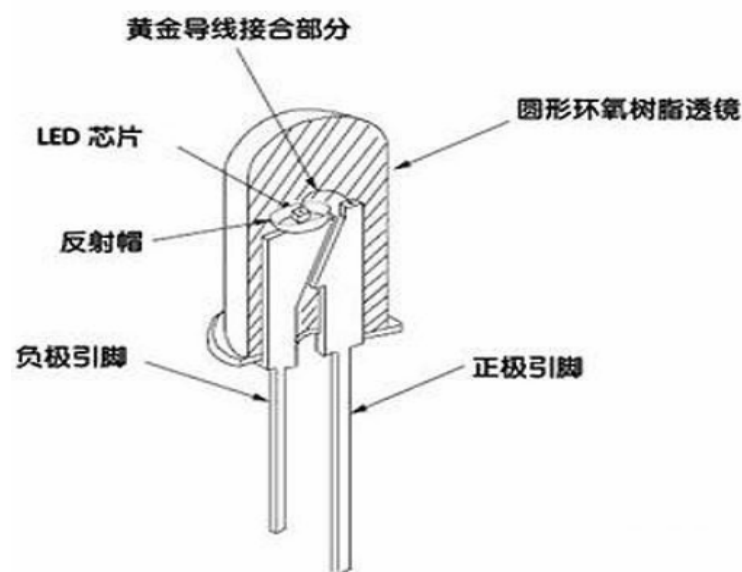
$$R = (E - VF) / I ;$$

式中 E 为电源电压，VF 为 LED 的正向压降，I 为 LED 的一般工作电流。发光二极管的工作电压一般为 1.5~2.0V，其工作电流一般为 10~20mA。所以在 5v 的数字逻辑电路中，可使用 220Ω 的电阻作为限流电阻。

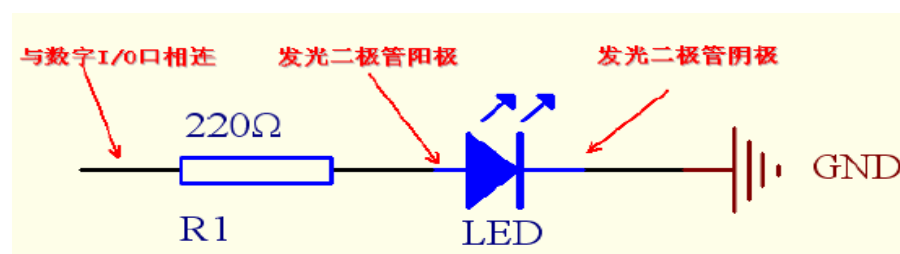


### 3、Led灯的内部结构与连线

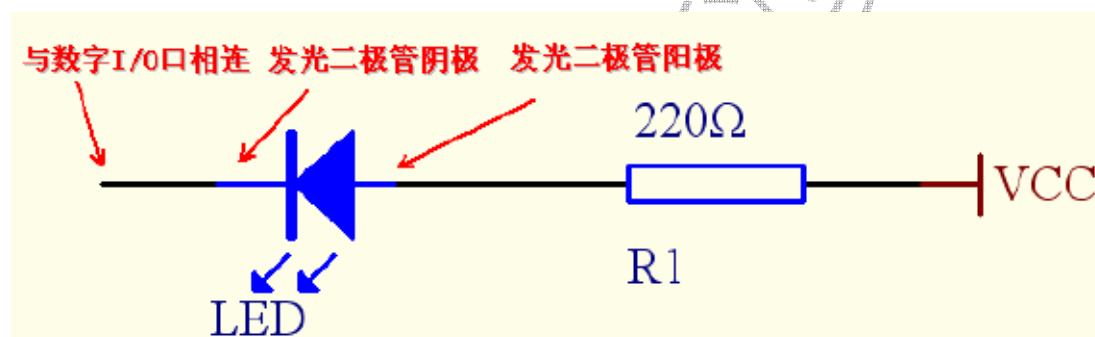
发光二极管的两根引线中较长的一根为正极，应连接电源正极。有的发光二极管的两根引线一样长，但管壳上有一凸起的小舌，靠近小舌的引线是正极。如下图所示：



Led 灯有两种连线方法：当 led 灯的阳极通过限流电阻与板子上的数字 I/O 口相连，数字口输出高电平时，led 导通，发光二极管发出亮光；数字口输出低电平时，led 截止，发光二极管熄灭。如图：

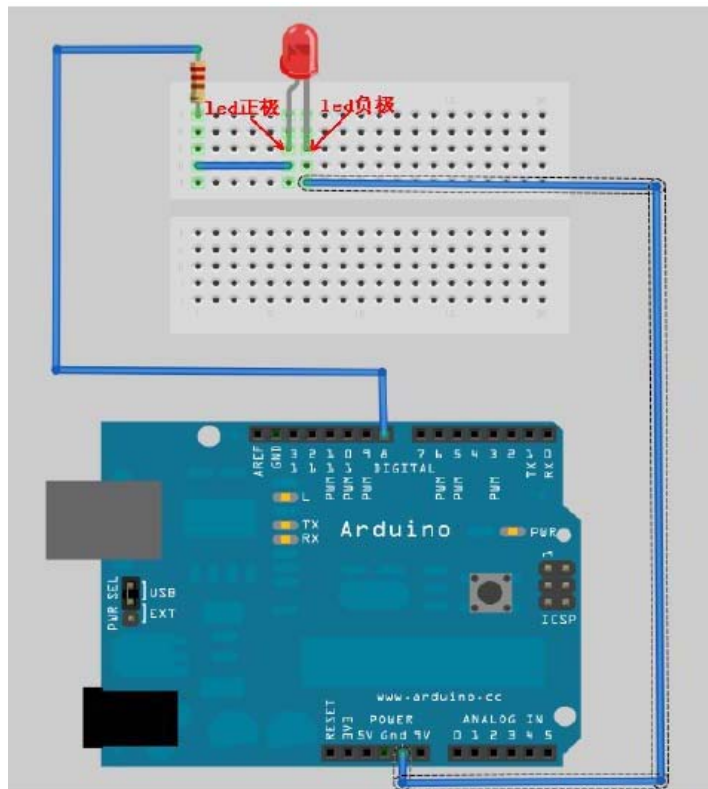


当 led 灯的阴极与板子上的数字 I/O 口相连时，数字口输出高电平，led 截止，发光二极管熄灭；数字口输出低电平，led 灯导通，发光二极管点亮。





本实验选择了接线方法 1 连接发光二极管，将  $220\Omega$  电阻的一端插在 Prototype Shield 扩展板上的第 8 个 digital I/O 口，电阻的另一端插在面包板上，电阻和发光二极管通过导线相连，发光二极管的负端插在面包板上与 GND 相连。具体连接如图：



### 简单的控制一个led灯的闪烁实验

#### 1) 实验器件

Led 灯：1 个  
 $220\Omega$  的电阻：1 个  
多彩面包板实验跳线：若干

#### 2) 实验连线

按照 Arduino 使用介绍将控制板、板子、面包板连接好，下载线插好。  
最后，按照图将发光二极管连接到数字的第 8 引脚。这样我们就完成了实验的连线部分。

#### 3) 实验原理

先设置数字 8 引脚为高电平点亮 led 灯，然后延时 1s，接着设置数字 8 引脚为低电平熄灭 led 灯，再延时 1s。这样使 led 灯亮 1s、灭 1s，在规视上就形成闪烁状态。如果想让 led 快速闪烁，可以将延时时间设置的小一些，但不能过小，过小的话人眼就识别不出来了，看上去就像 led 灯一直在亮着；如果想让 led 慢一点闪烁，可以将延时时间设置的大一些，但不能过大，过大的话就没有闪烁的效果了。

#### 4) 程序代码

程序代码如下：

```
int ledPin=8; //设定控制 LED 的数字 IO 脚
void setup()
{
    pinMode(ledPin,OUTPUT); //设定数字 IO 口的模式，OUTPUT 为输出
}
void loop()
{
    digitalWrite(ledPin,HIGH); //设定 PIN8 脚为 HIGH = 5V 左右
    delay(1000); //设定延时时间，1000 = 1 秒
    digitalWrite(ledPin,LOW); //设定 PIN8 脚为 LOW = 0V
    delay(1000); //设定延时时间，1000 = 1 秒
}
```

从 Arduino 教程中我们可以知道，Arduino 语法是以 setup() 开头，loop() 作为主体的一个程序极架。setup() 是用来初始化变量，管脚模式，调用库函数等等，此函数只运行一次。本程序在 setup() 中用数字 IO 口输入输出模式定义函数 pinMode (pin, mode)，将数字的第 8 引脚设置为输出模式。

loop() 函数是一个循环函数，函数内的语句周而复始的循环执行，本程序在 loop() 中先用数字 IO 口输出电平定义函数 digitalWrite(pin, value)，将数字 8 口定义为高电平，点亮 led 灯；接着调用延时函数 delay(ms)（单位 ms）延时 1000ms，让发光二极管亮 1s；再用数字 IO 口输出电平定义函数 digitalWrite(pin, value)，将数字 8 口定义为低电平，熄灭 led 灯；接着再调用延时函数 delay(ms)（单位 ms）延时 1000ms，让发光二极管熄灭 1s。因为 loop() 函数是一个循环函数，所以这个过程会不断的循环。

#### 5) 下载程序

按照 arduino 教程中的程序下载方法将本程序下载到实验板中。

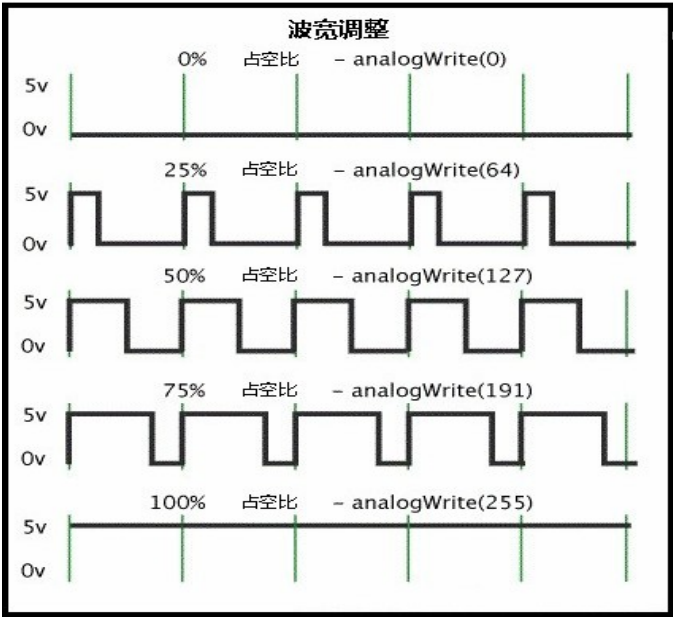
#### 6) 程序功能

将程序下载到实验板后我们可以观察到，发光二极管以 1s 的时间间隔不断的闪烁。

se Width Modulation 就是通常所说的PWM，译为脉冲宽度调制（PWM）是一种对模拟信号电平进行数字编码的方法，由于计算机不能输出模拟电压，只能输出0 或5V 的数字电压值，我们就通过使用高分辨率计数器的占空比被调制的方法来对一个具体模拟信号的电平进行编码。PWM 仍然是数字的，因为在给定的任何时刻，满幅值的直流供电要么是5V (ON)，要么是0V (OFF)。电压或电流源是以一种通(ON)或断(OFF)的重复脉冲序列被加到模拟负载上去的。通的时候即是直流供电被加到负载上的时候，断的时候即是供电被断开的时候。只要带宽足够，任何模拟值都可以使用PWM 进行编码。输出的电压值是通过通和断的时间进行计算的。输出电压=（接通时间/脉冲时间）\*最大电压值

## 第四章 PWM控制实践

脉冲宽度调制（PWM）是一种对模拟信号电平进行数字编码的方法，由于计算机不能输出模拟电压，只能输出0 或5V 的数字电压值，我们就通过使用高分辨率计数器，利用方波的占空比被调制的方法来对一个具体模拟信号的电平进行编码。PWM 信号仍然是数字的，因为在给定的任何时刻，满幅值的直流供电要么是5V (ON)，要么是0V (OFF)。电压或电流源是以一种通(ON)或断(OFF)的重复脉冲序列被加到模拟负载上去的。通的时候即是直流供电被加到负载上的时候，断的时候即是供电被断开的时候。只要带宽足够，任何模拟值都可以使用PWM 进行编码。输出的电压值是通过通和断的时间进行计算的。输出电压=（接通时间/脉冲时间）\*最大电压值



Arduino 端口的输入电压只有两个 0V 与 5V。如我想要 3V 的输出电压怎么办…，也许你会说串联电阻？OK，这个方法是正确的。但是如果我想 1V,3V,3.5V 等等之间来回变动怎么办呢？不可能不停地切换电阻吧。这种情况下…，就需要使用 PWM 了。他是怎么控制的呢，对于 arduino 的数字端口电压输出只有 LOW 与 HIGH 两个开关，对应的就是 0V 与 5V 的电压输出，我们本把 LOW 定义为 0，HIGH 定义为 1。一秒内让 arduino 输出 500 个 0 或者 1 的信号。如果这 500 个全部为 1，那就是完整的 5V，如果全部为 0，那就是 0V。如果 010101010101 这样输出，刚好一半一半，这样输出端口实际的输出电压是 2.5V。这个类似我们放映电影，我们所看的电影并不是完全连续的，它其实是每秒输出 25 张图片，在这种情况下人的肉眼是分辨不出来的，看上去就是连续的了。PWM 也是同样的道理，如果想要不同的电压，就控制 0 与 1 的输出比例就 ok。当然…这和真实的连续输出还是有差别的，单位时间内输出的 0,1 信号越多，控制的就越精确。

在上图中，绿线之间代表一个周期，其值也是 PWM 频率的倒数。换句话说，如果 arduino PWM 的频率是 500Hz，那么两绿线之间的周期就是 2 毫秒。analogWrite() 命令中可以操控的范围为 0-255，analogWrite(255)表示 100% 占空比（常开），analogWrite(127)占空比大约为 50%（一半的时间）。

在 Arduino 语法中，我们使用函数：“analogWrite()”

**analogWrite()**：作用是给端口写入一个模拟值(PWM 波)。可以用来控制 LED 灯的亮度变化，或者以不同的速度驱动马达。当执行 `analogWrite()` 命令后，端口会输出一个稳定的占空比的方波。除非有下一个命令来改变它。PWM 信号的频率大约为 490Hz。

在使用 ATmega168、ATmega328 与 UNO 的 arduino 控制板上，其工作在 3,5,6,9,10,11 端口。Arduino Mega，2560 控制板，可以工作于 2-13 号端口。在更古老的基于 ATmega8 的 arduino 控制板上，`analogWrite()` 命令只能工作于 9,10,11 号端口。在使用 `analogWrite()` 命令前，可以不使用 `pinMode()` 命令把端口定义为输出端口，当然如果定义了更好，这样利于程序语言规范。

#### 语法

**analogWrite(pin, value)**

#### 参数

**Pin**：写入的端口

**value**: 占空比: 在 0-255 之间。

#### 注释与已知问题:

当 PWM 输出与 5,6 号端口的时候，会产生比预期更高的占空比。原因是 PWM 输出所使用的内部时钟，`millis()` 与 `delay()` 两函数也在使用。所以要注意使用 5,6 号端口时，空占比要设置的稍微低一些，或者会产生 5,6 号端口无法输出完全关闭的信号。

## 2) 实验器件

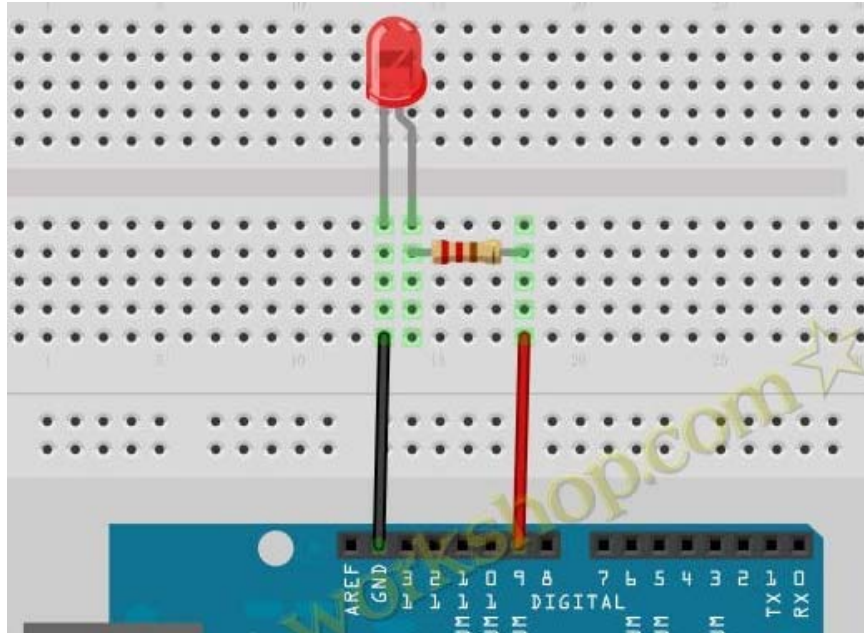
Led 灯：1 个

220Ω 的电阻：1 个

多彩面包板实验跳线：若干

## 3) 实验连线

按照 Arduino 使用介绍将控制板、面包板连接好 最后，按照图将发光二极管通过 220 欧姆电阻连接到数字的第 9 引脚。这样我们就完成了实验的连线部分。



## 5) 程序代码

程序代码如下：

```
int brightness = 0; //定义整数型变量 brightness 与其初始值，此变量用来表示 LED 的亮度。
int fadeAmount = 5; //定义整数型变量 fadeAmount，此变量用来做亮度变化的增减量。

void setup() {
  pinMode(9, OUTPUT); // 设置 9 号口为输出端口：
}

void loop() {

  analogWrite(9, brightness); //把brightness 的值写入 9 号端口

  brightness = brightness + fadeAmount; //改变 brightness 值，使亮度在下次循环发生改变

  if (brightness == 0 || brightness == 255) {
    fadeAmount = -fadeAmount; //在亮度最高与最低时进行翻转
  }

  delay(30); //延时 30 毫秒
}
```

## 5) 下载程序



按照 arduino 教程中的程序下载方法将本程序下载到实验板中。

## 6) 程序功能

将程序下载到实验板后我们可以观察到，通过 PWM 来控制一盏 LED 灯，让它慢慢变亮再慢慢变暗，如此循环。

### PWM

Pulse Width Modulation 脉冲宽度调制，简称脉宽调制。是利用微处理器的数字输出来对模拟电路进行控制的一种非常有效的技术，广泛应用在从测量、通信到功率控制与变换的许多领域中。

脉冲宽度调制（PWM）是一种对模拟信号电平进行数字编码的方法，由于计算机不能输出模拟电压，而只能输出 0V 或 5V 的数字电压值，（0V 为 0；5V 为 1）所以通过高分辨率计数器，利用方波的占空比被调制的方法对一个具体模拟信号的电平进行编码。

但 PWM 信号仍然是数字的，因为在给定的任意时刻，直流供电要么是 5V（数字值为 1），要么是 0V（数字值为 0）。电压或电流源以一种通（ON）、断（OFF）的重复脉冲序列加到模拟负载上，只要带宽足够，任何模拟值都可以使用 PWM 进行编码。

输出的电压值是通过通和断的时间进行计算的，计算公式为：

**输出电压 = （ 接通时间 / 脉冲时间 ） \* 最大电压值**

PWM 的三个基本参数：

- 1、脉冲宽度变化幅度（最小值/最大值）
- 2、脉冲周期（1 秒内脉冲频率个数的倒数）
- 3、电压高度（例如：0V-5V）

Arduino 控制器上有 6 个 PWM 接口分别是数字接口 3、5、6、9、10、11



## 第五章 广告流水灯实验

### 1) 实验器件

- Led 灯：6 个
- 220Ω 的电阻：6 个
- 多彩面包板实验跳线：若干

### 2) 实验连线

按照上述方法将板子和数据线连好。然后按照二极管的接线方法，将六个 LED 灯依次接到数字 1~6 引脚上。如图：

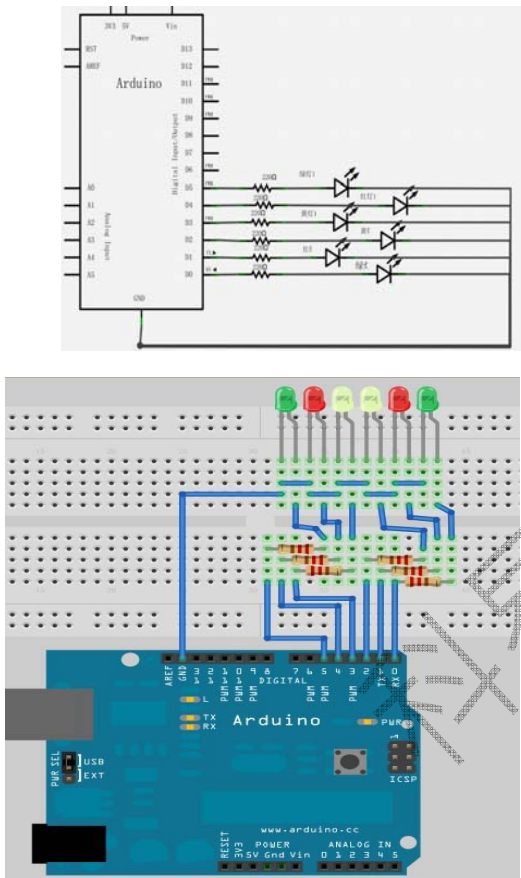


图 1.6 广告灯实验的接线

### 3) 实验原理

在生活中我们经常会看到一些由各种颜色的 led 灯组成的广告灯，广告灯上各个位置上的 led 灯不断的亮灭变化，就形成各种不同的效果。本节实验就是利用 led 灯编程模拟广告灯的效果。

在程序中我们设置 led 灯亮灭的次序和时间，这样就可以组成不同的效果。样式一子程

序：led 首先从左边的绿灯开始间隔 200ms 依次点亮六个 led 灯，如图 1.6，接着从右边的绿灯开始间隔 200ms 依次熄灭六个 led 灯。灯闪烁子程序：六个 led 灯首先全部点亮，接着延时 200ms，最后六个 led 灯全部熄灭，这个过程循环两次就实现了闪烁的效果。样式二子程序设置 k 和 j 的值让中间的两个黄灯亮先亮，接着让挨着两个黄灯两边的红灯亮，最后让两边的绿灯亮；执行一遍后改发 k 和 j 的值让让两边的绿灯先熄灭，接着两边的红灯熄灭，最后中间的两个黄灯熄灭。样式三子程序：设置 k 和 j 的值，让两边的绿灯亮 400ms 后再熄灭， 接着让两边的红灯亮 400ms 后再熄灭，最后让中间的两个黄灯亮 400ms 后再熄灭；执行一遍后改发 k 和 j 的值让两个红灯亮 400ms 后熄灭，接着让两边的绿灯亮 400ms 后熄灭。

#### 4 ) 程序代码

程序代码在广告灯程序文件夹中，双击打开后有一个 led2 文件夹，接着双击打开后可以看见有一个 led2.pde 文件，双击图标即可打开。打开后我们可以看到这是 arduino 编程软件窗口，上面有本实验的程序代码。

程序代码如下：

```
//设置控制 Led 的数字IO 脚
int Led1 = 1;
int Led2 = 2;
int Led3 = 3;
int Led4 = 4;
int Led5 = 5;
int Led6 = 6;
//led 灯花样显示样式 1 子程序
void style_1(void)
{
    unsigned char j;
    for(j=1;j<=6;j++)//每隔 200ms 依次点亮 1~6 引脚相连的 led 灯
    {
        digitalWrite(j,HIGH);//点亮 j 引脚相连的 led 灯
        delay(200);//延时 200ms
    }
    for(j=6;j>=1;j--)//每隔 200ms 依次熄灭 6~1 引脚相连的 led 灯
    {
        digitalWrite(j,LOW);//熄灭 j 引脚相连的 led 灯
        delay(200);//延时 200ms
    }
}
//灯闪烁子程序
void flash(void)
{
    unsigned char j,k;
```



```
for(k=0;k<=1;k++)//闪烁两次
{
    for(j=1;j<=6;j++)//点亮 1~6 引脚相连的 led 灯
        digitalWrite(j,HIGH);//点亮与 j 引脚相连的 led 灯
    delay(200);//延时 200ms
    for(j=1;j<=6;j++)//熄灭 1~6 引脚相连的 led 灯
        digitalWrite(j,LOW);//熄灭与 j 引脚相连的 led 灯
    delay(200);//延时 200ms
}
}
//led 灯花样显示样式 2 子程序
void style_2(void)
{
    unsigned char j,k;
    k=1;//设置 k 的初值为 1
    for(j=3;j>=1;j--)
    {
        digitalWrite(j,HIGH);//点亮灯
        digitalWrite(j+k,HIGH);//点亮灯
        delay(400);//延时 400ms
        k +=2;//k 值加 2
    }
    k=5;//设置k 值为 5
    for(j=1;j<=3;j++)
    {
        digitalWrite(j,LOW);//熄灭灯
        digitalWrite(j+k,LOW);//熄灭灯
        delay(400);//延时 400ms
        k -=2;//k 值减 2
    }
}
//led 灯花样显示样式 3 子程序
void style_3(void)
{
    unsigned char j,k;//led 灯花样显示样式 3 子程序
    k=5;//设置k 值为 5
    for(j=1;j<=3;j++)
    {
        digitalWrite(j,HIGH);//点亮灯
        digitalWrite(j+k,HIGH);//点亮灯
        delay(400);//延时 400ms
        digitalWrite(j,LOW);//熄灭灯
        digitalWrite(j+k,LOW);//熄灭灯
        k -=2;//k 值减 2
    }
}
```

```
    }
    k=3;//设置k 值为
    3 for(j=2;j>=1;j--)
    {
        digitalWrite(j,HIGH);//点亮灯
        digitalWrite(j+k,HIGH);//点亮
        灯delay(400);// 延 时 400ms
        digitalWrite(j,LOW);//熄灭灯
        digitalWrite(j+k,LOW);//熄灭
        灯k +=2;//k 值加 2
    }
}
void setup()
{
    unsigned          char          i;
    for(i=1;i<=6;i++)//依次设置 1~6 个数字引脚为输出模式
        pinMode(i,OUTPUT);//设置第 i 个引脚为输出模式
}
void loop()
{
    style_1();//样式 1
    flash();//闪烁
    style_2();//样式 2
    flash();//闪烁
    style_3();//样式 3
    flash();//闪烁
}
```

程序代码中用到的：**for(i=1;i<=6;i++)//依次设置 1~6 个数字引脚为输出模式**  
**pinMode(i,OUTPUT);//设置第 i 个引脚为输出模式**

这是一个 for 循环。它的一般形式为: for(<初始化>; <条件表达式>; <增量>) 语句; 初始化总是一个赋值语句, 它用来给循环控制发量赋初值; 条件表达式是一个关系表达式, 它决定什么时候退出循环; 增量定义循环控制发量每循环一次后 按什么方式变化。这三个部分之间用";"分开。 例如: for(i=1; i<=10; i++) 语句; 上例中先给 " i " 赋初值 1, 判断 " i " 是否小于等于 10, 若是则执行语句, 之后值增 加 1。再重新判断, 直到条件为假, 即 i>10 时, 结束循环。

### 5 ) 下载程序

按照 arduino 教程中的程序下载方法将本程序下载到实验板中。

### 6 ) 程序功能

将程序下载到实验板后我们可以观察到，六个 led 不断的循环执行样式一子程序—>闪烁子程序—>样式二子程序—>闪烁子程序—>样式三子程序—>闪烁子程序。

在掌握了以上两个程序后，大家可以充分发挥自己的想象，编写出自己想要的 led 灯效果，玩转多彩 led 灯。

## 第六章 交通灯设计实验

### 1) 实验器件

- 红、绿、黄 Led 灯：3 个
- 220 Ω 的电阻：3 个
- 多彩面包板实验跳线：若干
- 面包板：1 个

### 2) 实验连线

按照下图原理图和实物图，将 3 个 LED 灯依次接到数字 10，7，4 脚上。如图：

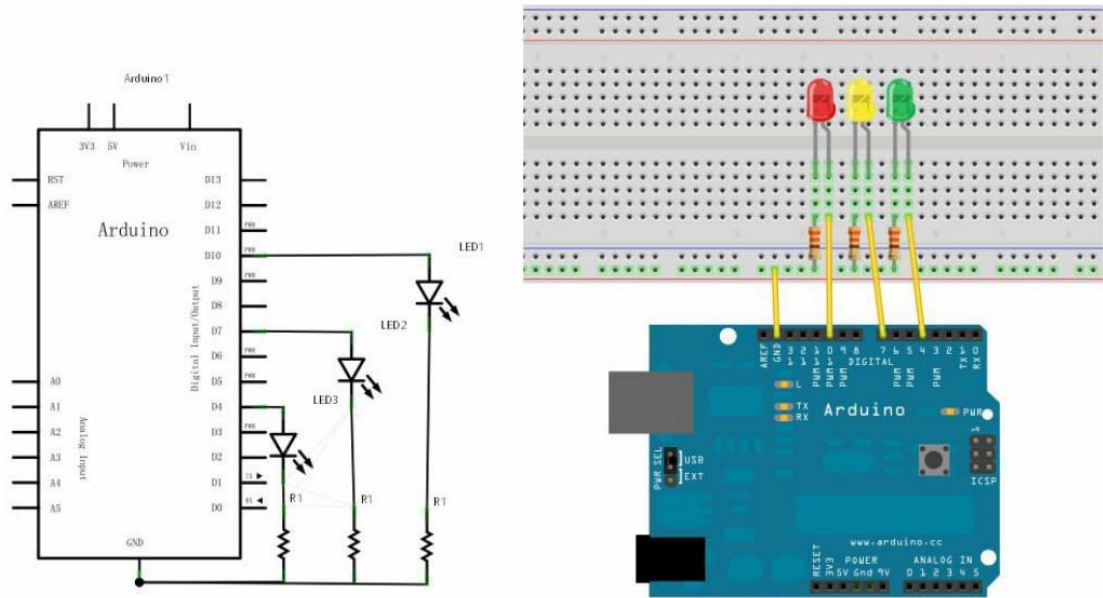


图 1.6 广告灯实验的接线

在这个交通灯模拟实验，红黄绿三色小灯闪烁时间要模拟真实的交通灯，我们使用的 arduino 的 delay() 函数来控制延时时间，这相对于 C 语言就要简单许多了。

### 4) 程序代码

程序代码在“交通灯设计实验”文件夹中，接着双击打开后可以看见有一个 trafficLed.pde 文件，双击图标即可打开。打开后我们可以看到这是 arduino 编程软件窗口，上面有本实验的程序代码。

程序代码如下：

```
int ledred=10; //定义数字接口 10 红灯
int ledyellow=7; //定义数字接口 7 黄灯
```

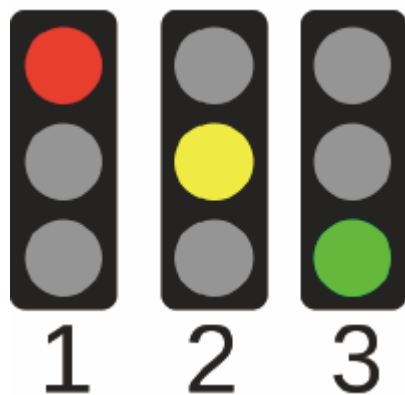
```
int ledgreen=4;    //定义数字接口 4 绿灯
void setup()
{
    pinMode(ledred,OUTPUT);//设置红灯接口为输出接口
    pinMode(ledyellow,OUTPUT);//设置黄灯接口为输出接口
    pinMode(ledgreen,OUTPUT);//设置绿灯接口为输出接口
}
void loop()
{
    digitalWrite(ledred,HIGH);//点亮红灯
    delay(1000);//延时 1000 毫秒 = 1 秒
    digitalWrite(ledred,LOW);//熄灭红灯
    digitalWrite(ledyellow,HIGH);//点亮黄灯
    delay(200);//延时 200 毫秒//
    digitalWrite(ledyellow,LOW);//熄灭黄灯
    digitalWrite(ledgreen,HIGH);//点亮绿灯
    delay(1000);//延时 1000 毫秒
    digitalWrite(ledgreen,LOW);//熄灭绿灯
}
```

## 5) 下载程序

按照 arduino 教程中的程序下载方法将本程序下载到实验板中。

## 6) 程序功能

将程序下载到实验板后，我们可以看到我们自己，设计控制的交通灯了。





## 第七章 蜂鸣器实验

### 一、蜂鸣器介绍

#### 1、认识蜂鸣器

蜂鸣器是一种一体化结构的电子讯响器，采用直流电压供电，广泛应用于计算机、打印机、复印机、报警器、电子玩具、汽车电子设备、电话机、定时器等电子产品中作发声器件。

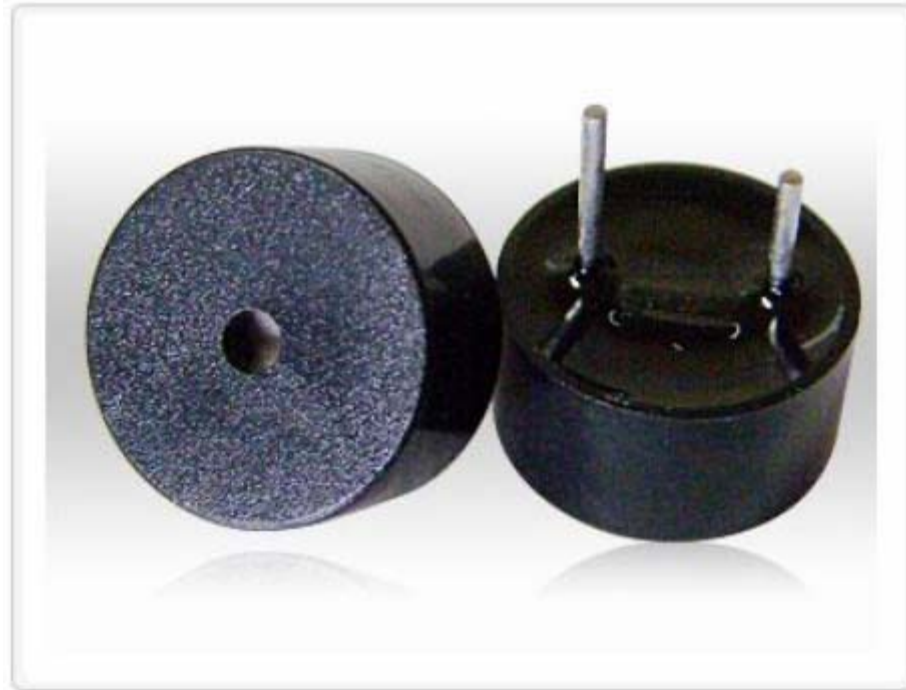
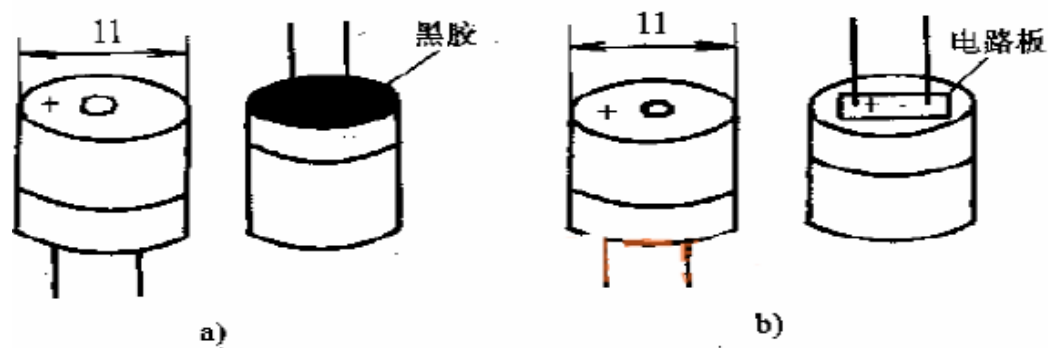


图 2.1 蜂鸣器

按其驱动方式的不同，可分为：有源蜂鸣器（内含驱动线路）和无源蜂鸣器（外部驱动）

教你区分有源蜂鸣器和无源蜂鸣器,现在市场上出售的一种小型蜂鸣器因其体积小(直径只有 11mm)、重量轻、价格低、结构牢靠，而广泛地应用在各种需要发声的电器设备、电子制作和单片机等电路中。有源蜂鸣器和无源蜂鸣器的外观如图 a、b 所示。a)有源 b)无源。



从图 a、b 外观上看，两种蜂鸣器好像一样，但仔细看，两者的高度略有区别，有源蜂鸣器 a，高度为 9mm，而无源蜂鸣器 b 的高度为 8mm。如将两种蜂鸣器的引脚都朝上放置时，可以看出有绿色电路板的一种是无源蜂鸣器，没有电路板而用黑胶封闭的一种是有源蜂鸣器。进一步判断有源蜂鸣器和无源蜂鸣器，还可以用万用表电阻档 R<sub>x1</sub> 档测试：用黑表笔接蜂鸣器 "+" 引脚，红表笔在另一引脚上来回碰触，如果发出咔、咔声的且电阻只有 8Ω (或 16Ω) 的是无源蜂鸣器；如果能发出持续声音的，且电阻在几百欧以上的，是有源蜂鸣器。有源蜂鸣器直接接上额定电源(新的蜂鸣器在标签上都有注明)就可连续发声；而无源蜂鸣器则和电磁扬声器一样，需要接在音频输出电路中才能发声。

按构造方式的不同，可分为：电磁式蜂鸣器和压电式蜂鸣器；

**压电式蜂鸣器** 压电式蜂鸣器主要由多谐振荡器、压电蜂鸣片、阻抗匹配器及共鸣箱、外壳等组成。有的压电式蜂鸣器外壳上还装有发光二极管。多谐振荡器由晶体管或集成电路构成。当接通电源后（1.5~15V 直流工作电压），多谐振荡器起振，输出 1.5~2.5kHz 的音频信号，阻抗匹配器推动压电蜂鸣片发声。压电蜂鸣片由锆钛酸铅或铌镁酸铅压电陶瓷材料制成。在陶瓷片的两面镀上银电极，经极化和老化处理后，再与黄铜片或不锈钢片粘在一起。电磁式蜂鸣器由振荡器、电磁线圈、磁铁、振动膜片及外壳等组成。接通电源后，振荡器产生的音频信号电流通过电磁线圈，使电磁线圈产生磁场。振动膜片在电磁线圈和磁铁的相互作用下，周期性地振动发声。

## 2、工作原理

蜂鸣器发声原理是电流通过电磁线圈，使电磁线圈产生磁场来驱动振动膜发声的，因此需要一定的电流才能驱动它，本实验用的蜂鸣器内部带有驱动电路，所以可以直接使用。当蜂鸣器连接的引脚为高电平时，内部驱动电路导通，蜂鸣器发出声音；当蜂鸣器连接的引脚为低电平，内部驱动电路截止，蜂鸣器不发出声音。

## 3、蜂鸣器的连线

本实验用的蜂鸣器内部带有驱动电路，所以可以直接将蜂鸣器的正极连接到数字口，蜂鸣器的负极连接到 GND 插口中。如下图：

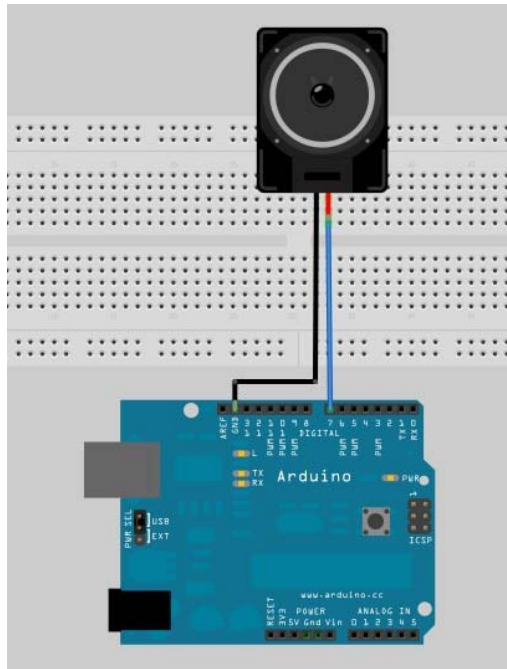


图 2.1 蜂鸣器接线图

## 二、蜂鸣器模拟救护车警笛声音实验

### 1、实验器件

蜂鸣器：1 个  
多彩面包板实验跳线：若干

### 2、实验连线

按照 Arduino 教程将控制板、面包板连接好，下载线插好。然后按照蜂鸣器的接法将蜂鸣器连接到数字 7 口上，连线完毕。

### 3、实验原理

蜂鸣器发出声音的时间间隔不同，频率就不同，所以发出的声音就不同。根据这一原理我们通过改变蜂鸣器发出声音的时间间隔，来发出不同种声音，来模拟各种声音。本程序首先让蜂鸣器间隔 1ms 发出一种频率的声音，循环 80 次；接着让蜂鸣器间隔 2ms 发出另一种频率的声音，循环 100 次。

### 4、程序代码

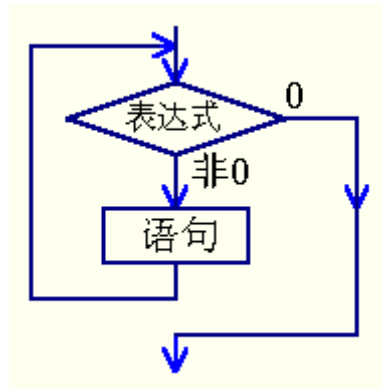
程序代码如下：

```
int buzzer=7;//设置控制蜂鸣器的数字 IO 脚
void setup()
{
    pinMode(buzzer,OUTPUT);//设置数字 IO 脚模式，OUTPUT 为输出
}
void loop()
{
    unsigned char i,j;//定义变量
    while(1)
    {
        for(i=0;i<80;i++)//输出一个频率的声音
        {
            digitalWrite(buzzer,HIGH);//发声音
            delay(1);//延时 1ms
            digitalWrite(buzzer,LOW);//不发声音
            delay(1);//延时 ms
        }
        for(i=0;i<100;i++)//输出另一个频率的声音
        {
            digitalWrite(buzzer,HIGH);//发声音
            delay(2);//延时 2ms
            digitalWrite(buzzer,LOW);//不发声音
            delay(2);//延时 2ms
        }
    }
}
```

在 loop（）中用的 while 也是一个循环语句，一般形式：

**while(表达式)**  
语句

表达式是循环条件，语句是循环体。语义是：计算表达式的值，当值为真(非 0)时， 执行循环体语句。其执行过程可用下图表：



作用：实现“当型”循环。当“表达式”非 0（真）时，执行“语句”。“语句”是被循环执行的程序，称为“循环体”。

## 5、 下载程序

按照 arduino 教程中的程序下载方法将本程序下载到实验板中。

## 6、 程序功能

将程序下载到实验板后我们可以听到，蜂鸣器发出救护车警笛声。  
掌握本程序后，大家可以在程序中自己改发时间间隔，调试出各种频率的声音。

科易互动科技

## 第八章 倾斜开关实验

### 一、倾斜开关介绍

#### 1、了解倾斜开关

本节实验所使用的倾斜开关是内部带有一个金属滚珠的滚珠倾斜开关，如图所示：



图 5.1 SW—200D 倾斜开关

滚珠开关：也叫碰珠开关、摇珠开关、钢珠开关、倾斜开关，倒顺开关、角度传感器。它主要是利用滚珠在开关内随不同倾斜角度的变化，达到触发电路的目的。目前滚珠开关在市场上使用的常用型号有 SW-200D、SW-460、SW-300DA 等，本节使用的是 SW-200D 型号的。这类开关不象传统的水银开关，它功效同水银开关，但没有水银开关的环保及安全等问题。

#### 2、工作原理

观察倾斜开关我们可以发现，倾斜开关的一端为金色寻针，另一端为银色寻针。金色一端为<ON>导通触发端银色一端为<OFF>开路端当受到外力摇晃而达到适当晃动力时或金色一端设置角度低于水平适当角度时导电接脚电气特性会产生短时间导通或持续导通<ON>状态。而当电气特性要恢复开路状态<OFF>时开关设置环境必须为静止，且银色一端设置角度需低于水平 10 度。

#### 3、倾斜开关连线

---



将倾斜开关银色的一端连接到 5V 插口，金色一端连接到模拟口。

## 二、倾斜开关控制led灯的亮灭

### 1、实验器件

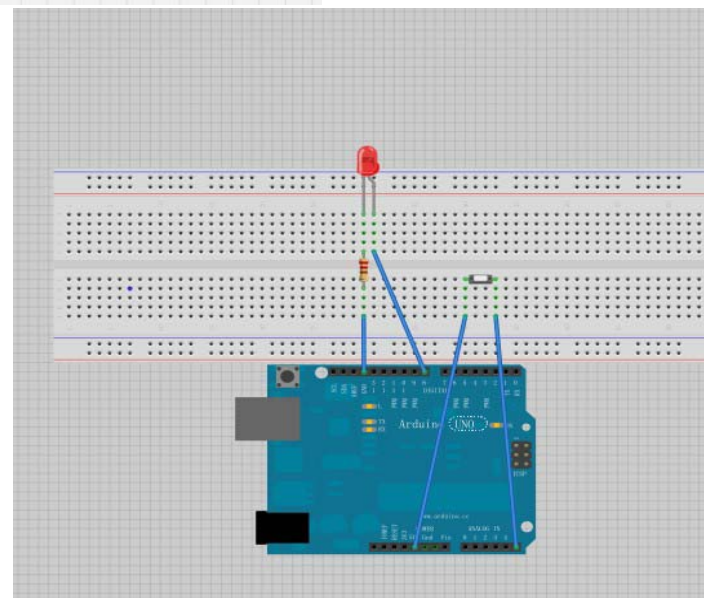
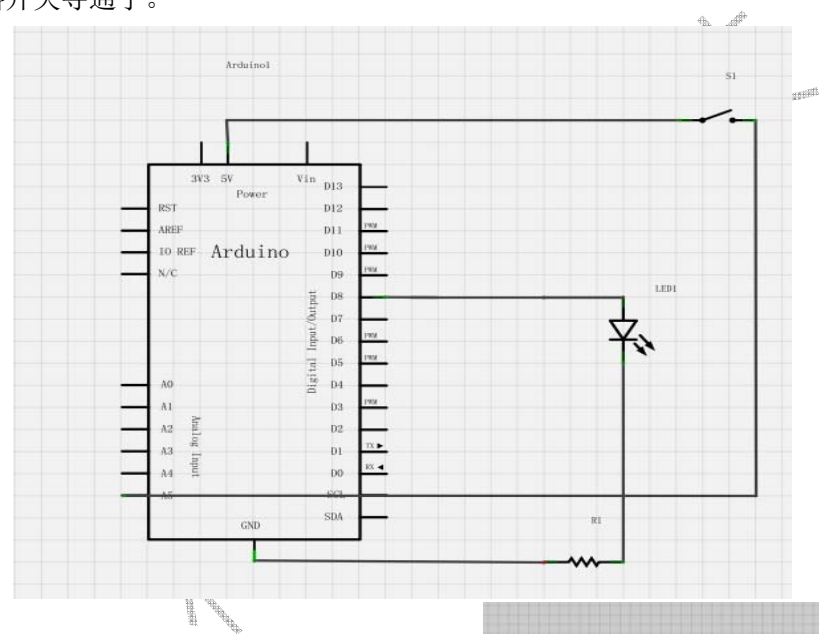
- 倾斜开关模块：1 个
- 多彩面包板实验跳线：若干

### 2、实验连线

按照 Arduino 教程将控制板、面包板连接好，下载线插好。然后将倾斜开关连接到模拟 5 引脚。

### 3、实验原理

当金色一端低于水平位置倾斜，开关寻通，模拟口电压值为 5V 左右（数字二进制表示为 1023），点亮 led 灯。当银色一端低于水平位置倾斜，开关截止，模拟口电压值为 0V 左右（数字二进制表示为 0），熄灭 led 灯。在程序中模拟口电压值是否大于 2.5V 左右（数字二进制表示为 512），即可知道是否倾斜开关寻通了。



#### 4、程序代码

程序代码如下：

```
void setup()
{
    pinMode(8,OUTPUT);//设置数字 8 引脚为输出模式
}
void loop()
{
    int i;//定义变量 i
    while(1)
    {
        i=analogRead(5);//读取模拟 5 口电压值
        if(i>200)//如果大于 512 (2.5V)
        {
            digitalWrite(8,HIGH);//点亮 led 灯
        }
        else//否则
        {
            digitalWrite(8,LOW);//熄灭 led 灯
        }
    }
}
```

#### 5、下载程序

按照 arduino 教程中的程序下载方法将本程序下载到实验板中。

#### 6、程序功能

将程序下载到实验板后大家可以将板子倾斜观察 led 灯的状态。当金色一端低于水平位置倾斜，开关寻通，点亮 led 灯；当银色一端低于水平位置倾斜，开关截止，模拟口电压值为 0V 左右（数字二进制表示为 0），熄灭 led 灯。

掌握本程序后，大家可以按照自己的想法实验，还可以控制其他器件例如蜂鸣器等。

## 第九章 模拟值读取实验

### (1)简述

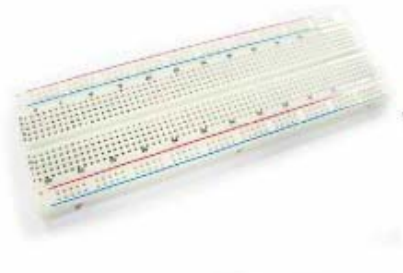
本章我们将使用模拟 I/O 口，Arduino 有模拟口—模拟 5 共计 6 个模拟接口，这 6 个接口也可以算作为接口功能复用，除模拟接口功能以外，这 6 个接口可作为数字接口使用，编号为数字 14—数字 19。

下面我们开始做实验了。

电位器（或者叫滑动电阻）是大家比较熟悉的典型的模拟值输出元件，本实验就用它来完成。



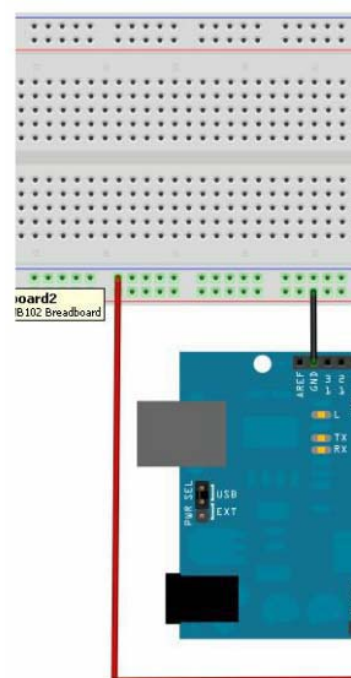
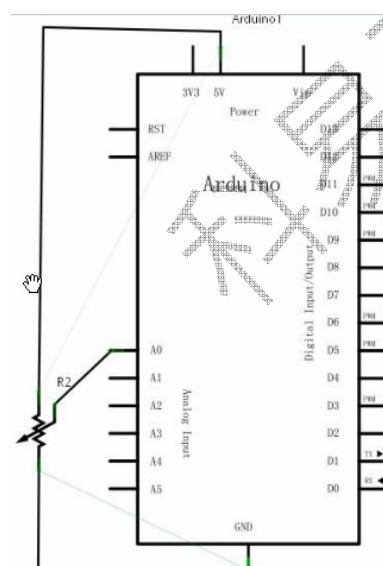
电位计\*1



面包板\*1



面包板跳线\*1 扎



`analogRead()`语句就可以读出模拟口的值，Arduino UNO 控制器是 10 位的 A/D 采集，所以读取的模拟值范围是 0-1023。首先我们在 `void setup()`里面设置波特率，显示数值属于 Arduino 与 PC 机通信，所以 Arduino 的波特率应与 PC 机软件设置的相同才能显示正确的数值，否则将会显示乱码或者不显示。在 Arduino 软件的串口工具监视窗口右下角有一个

可以设置波特率的按钮，选中与程序中设置的波特率语句相同的波特率，Serial.begin();括号中为波特率的值。

实例程序：

```
int potpin = 0; //定义模拟接口 0
int ledpin = 13; //定义数字接口 13
int val = 0; //将定义变量 val，并赋初值 0.

void setup()
{
    pinMode(ledpin,OUTPUT); //设置数字 13 引脚为输出模式
    Serial.begin(9600); //设置波特率 9600
}

void loop()
{
    digitalWrite(ledpin,HIGH); //点亮数字接口 13 的LED
    delay(50); //延时 0.05 秒
    digitalWrite(ledpin,LOW); //熄灭数字接口 13 的LED
    delay(50); //延时 0.05 秒
    val = analogRead(potpin); //读取模拟接口 0 的值，并将其赋给 val
    Serial.println(val); //显示出 val 的值
}
```

程序结果：

每读取一次值，arduino 自带的 LED 小灯就会闪烁一下，下图为读出的值（注意只做参考）：

本实验中，当您旋转电位计旋钮的时候就可以看到屏幕上的数值变化了。这种模拟值读取是我们很常用的功能。因为在很多的传感器，都是模拟值输出，我们读出模拟值后再进行相应的算法处理，就可以应用到我们需要实现的功能里了。



## 第十章 光控声音实验

### 一、光敏电阻介绍

#### 1、认识光敏电阻

光敏电阻又称光导管，常用的制作材料为硫化镉，另外还有硒、硫化铝、硫化铅和硫化铋等材料。这些制作材料具有在特定波长的光照下，其阻值迅速减小的特性。这是由于光照产生的载流子都参与导电，在外加电场的作用下漂移运动，从而使光敏电阻的阻值迅速下降。

实物如图：



图 6.1 光敏电阻

#### 2、工作原理

光敏电阻的工作原理是基于内光电效应。在半导体光敏材料两端装上电极引线，将其封装在带有透明窗的管壳里就极成光敏电阻，为了增加灵敏度，两电极常做成梳状。在有光照射时入射光强，电阻减小，入射光弱，电阻增大。

## 二、光控实验

#### 1、实验器件

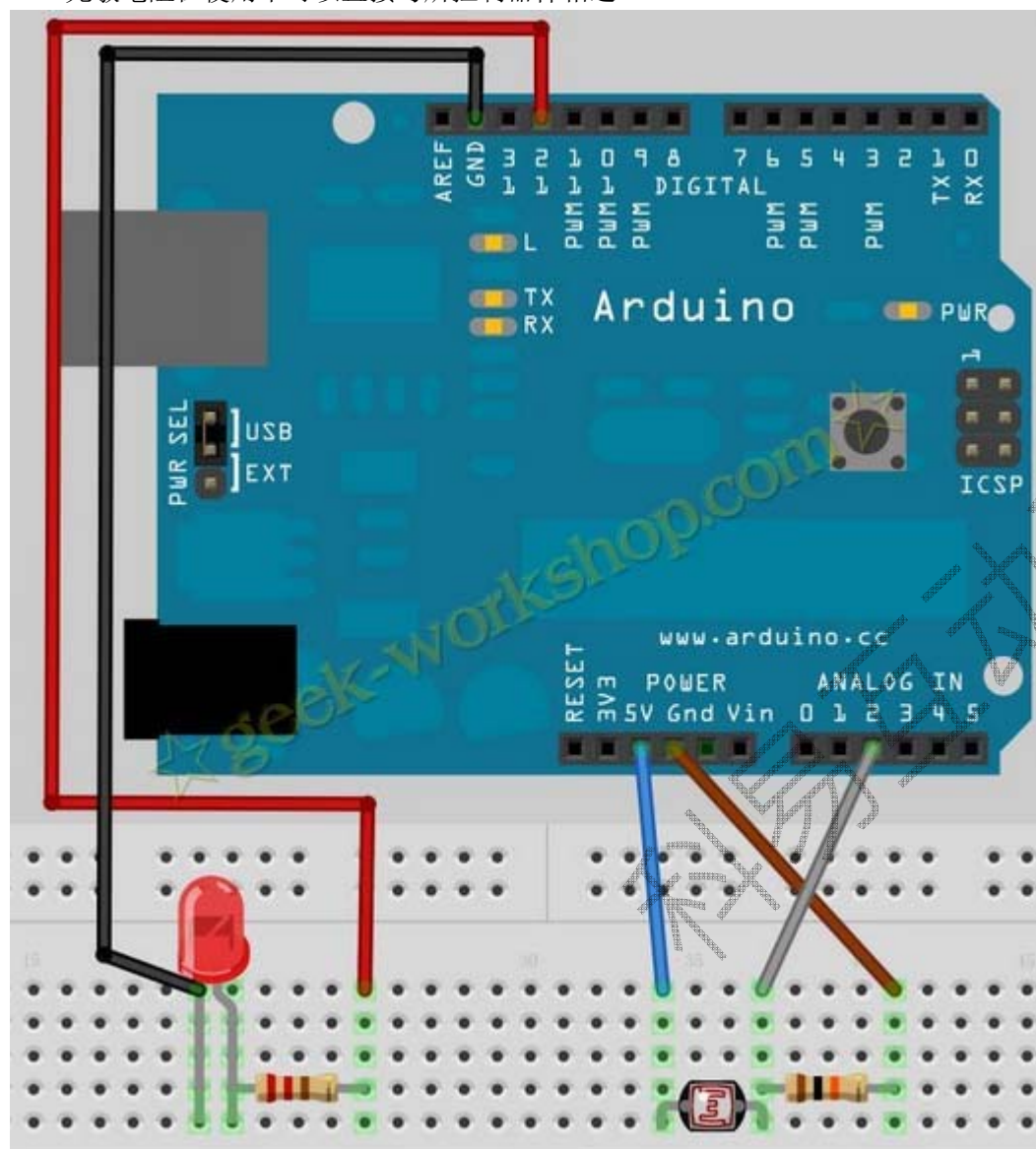
- 光敏电阻：1 个



- 蜂鸣器：1 个
- 10 K 欧电阻：1 个
- 220 欧电阻：1 个
- 多彩面包板实验跳线：若干

## 2、光敏电阻的连线

光敏电阻在使用中可以直接与所控制器件相连。



光敏电阻是根据光强度改变阻值的元件，本章实验我们接着上一章学过的知识，使用模拟口读取模拟值。

### 3、程序代码

程序代码在“光控声音实验”文件夹中.



```
int photocellPin = 2;    //定义变量 photocellsh=2，为电压读取端口。
int ledPin = 12;        //定义变量 ledPin=12，为 led 电平输出端口
int val = 0;            //定义 val 变量的起始值

void setup() {
    pinMode(ledPin, OUTPUT); //使 ledPin 为输出模式
}

void loop() {
    val = analogRead(photocellPin);    //从传感器读取值
    if(val<=512){
//512=2.5V，想让传感器敏感一些的时候，把数值调高，想让传感器迟钝的时候把数值调低。
        digitalWrite(ledPin, HIGH); //当 val 小于 512(2.5V)的时候，led 亮。
    }
    else{
        digitalWrite(ledPin, LOW);
    }
}
```

4、下载程序

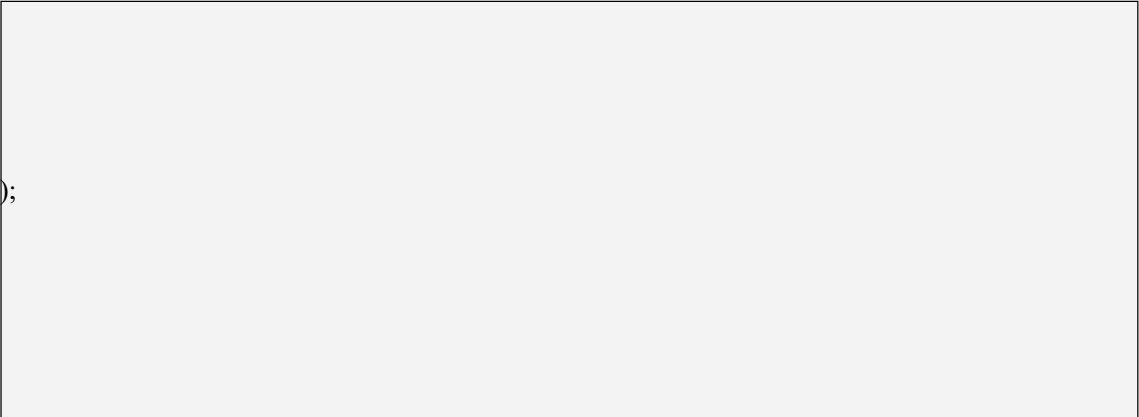
按照 arduino 教程中的程序下载方法将本程序下载到实验板中。

5、程序功能

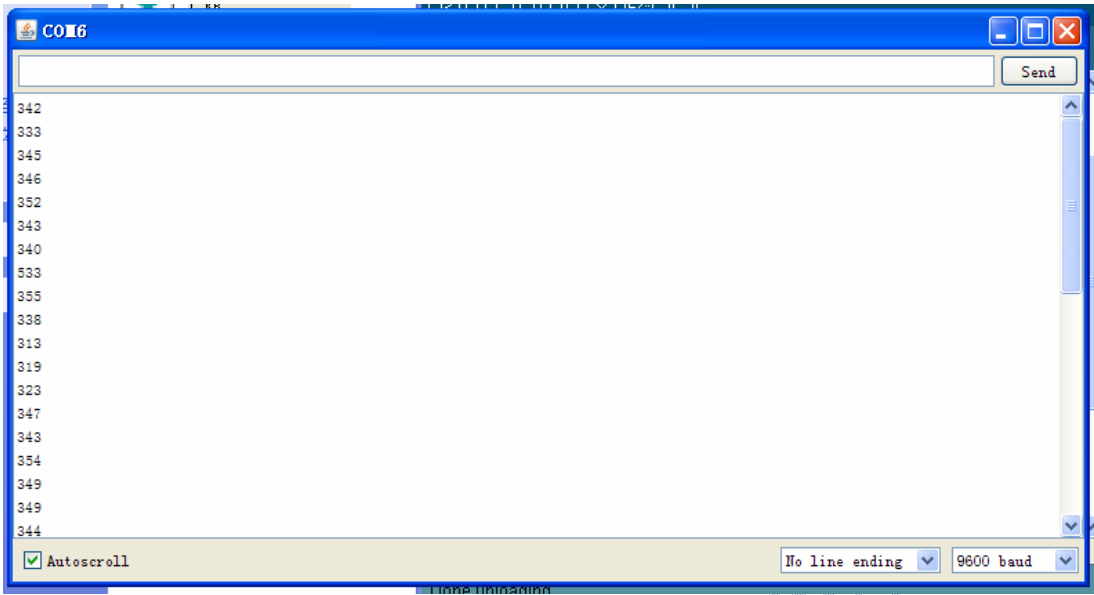
本次实验设计的效果是，当光照正常的时候 led 灯是灭的，当周围变暗时 led 灯变亮。

因为光敏电阻受不同光照影响变化很大，所以本次实验的参数是在 60W 三基色节能灯照射下实验（无日光照射），同样亮度的日光下光敏电阻的阻值会比日光灯下低不少，估计和不同光的波段有关系。不同环境下实验使用的参数不同，大家根据原理进行调整。

```
        delay(1000); //延时 1 秒钟
    }
    else
    {
        digitalWrite(SensorLED,LOW);
        delay(500); //延时 0.5 秒钟
    }
}
}
```



对着声音模块大声说话，声音越大，得出的数值越大。 当超过 400 时，小灯会亮 1 秒钟。

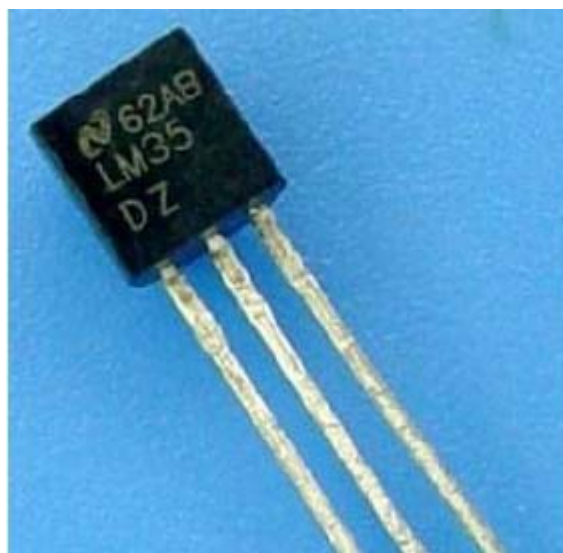


## 第十一章 LM35 温度传感器实验

## 一、温度传感器介绍

## 1、什么是温度传感器？

温度传感器就是利用物质各种物理性质随温度变化的规律把温度转换为电量的传感器。这些呈现规律性变化的物理性质主要有热电阻、热电偶等。温度传感器是温度测量仪表的核心部分，品种繁多。按测量方式可分为接触式和非接触式两大类，按照传感器材料及电子元件特性分为热电阻和热电偶两类。本实验使用的是 LM35 温度传感器。如下图：



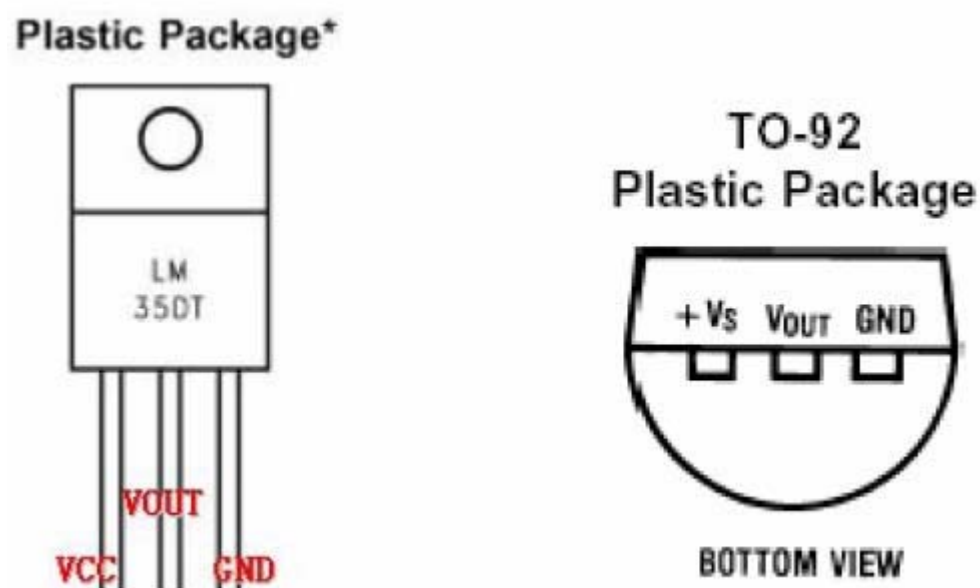
## 2、工作原理

LM35 温度传感器的输出电压不摄氏温标呈线性关系, 0℃, 时输出为 0V, 每升高 1℃, 输出电压增加 10mV。转换公式如下:

$$V_{\text{out\_LM35}}(T) = 10 \text{ mV}/^{\circ}\text{C} \times T^{\circ}\text{C}$$

### 3、LM35 的连线

LM35 的引脚示意图如下:



从实验盒中将温度传感器拿出来可以看到，温度传感器的一面是平的，另一面是半圆的。将平面对着自己，最左边的是 VCC 引脚（接+5v），中间的为 VOUT（电压值输出引脚，接板子上的模拟引脚），最右边的引脚为 GND 引脚（接板子上的 GND）。三个引脚分别接好就可以用了。

## 二、温度报警实验

### 1、实验器件

- LM35 温度传感器模块：1 个
- 多彩面包板实验跳线：若干

### 2、实验连线

首先将实验板连接好；接着按照 LM35 温度传感器连线方法将其连好，将 VOUT 连接到模拟 0 口。这样温度报警实验的电路就连接好了。

### 3、实验原理

从 LM35 温度传感器的工作原理可知，温度每升高 $1^{\circ}\text{C}$ ，vout 口输出的电压就增加 10MV。根据这一原理程序中实时读出模拟 0 口的电压值，由于模拟口读出的电压值使用 0~1023 表示的，即 0V 对应 0、5V 对应 1023。

应用中，我们只需一个 LM35 模块，利用模拟接口，将读取的模拟值转换为实际的温度。

### 4、程序代码

程序代码：

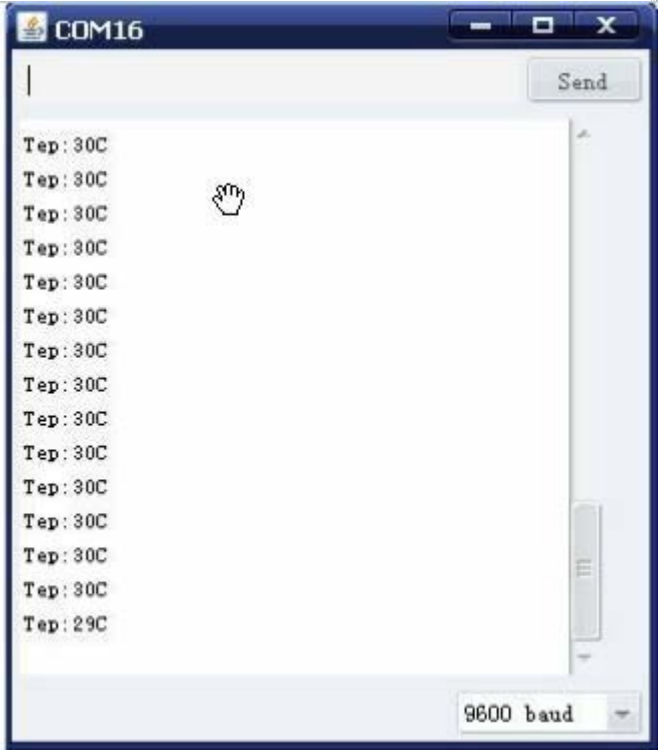
```
int potPin = 0 ;//定义模拟接口 0 连接 LM35 温度传感器
void setup()
{
    Serial.begin(9600);//设置波特率
}
void loop()
{
    int val;//定义变量
    int dat;//定义变量

    val = analogRead(potPin);//读取传感器的模拟值并赋值给 val

    dat = (125*val)>>8 ;//温度计算公式
    Serial.print("Tep : "); //原样输出显示 Tep 字符串代表温度
    Serial.print(dat); //输出显示 dat 的值
    Serial.println("C"); //原样输出显示 C 字符串
    delay(500); //延时 0.5 秒
}
```

5、程序功能

将程序下载到实验板，打开监视器，就可以看到当前的环境温度了。（实际上，温度值有一点点偏差，要根据自己的环境温度修改一下程序，使其完全与自己的环境一致。）



## 第十二章 数码管实验

### 一、数码管介绍

#### 1、认识数码管

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）；



图 3.1 数码管实物图

按能显示多少个“8”可分为 1 位、2 位、4 位等等数码管；



图 3.2 各种数码管

按发光二极管单元连接方式分为共阳极数码管和共阴极数码管。共阳极数码管是指将所



有发光二极管的阳极接到一起形成公共阳极(COM)的数码管。共阳数码管在应用时应将公共端 COM 接到+5V，当某一字段发光二极管的阴极为低电平时，相应字段就点亮。当某一字段的阴极为高电平时，相应字段就不亮。

图 3.3 共阳数码管内部结构

图 3.3 共阳数码管内部结构

共阳数码管是指将所有发光二极管的阳极接到一起形成公共阳极(COM)的数码管。共阳数码管在应用时应将公共端 PWR 接到电源输入 PWR 上，当某一字段发光二极管的阴极为低电平时，相应字段就点亮。当某一字段的阴极为高电平时，相应字段就不亮。

图 3.4 共阴数码管内部结构

图 3.4 共阴数码管内部结构

2 数码管的每一段是由发光二极管组成，所以在使用时跟发光二极管一样，也要连接限流电阻，否则电流过大会烧毁发光二极管。本实验用的是共阳数码管，共阳数码管在应用时应将公共端 COM 接到+5V，当某一字段发光二极管的阴极为低电平时，相应字段就点亮。当某一字段的阴极为高电平时，相应字段就不亮。

### 3、数码管的连线

将限流电阻的一端捏到数字 I/O 中，另一端不数码管的字段引脚相连，剩下六个字段和一个小数点依次按照这种方法接。将公共端 COM 如果是共阳极的就接到+5V，如果是共阴极的就接到 GND。

## 二、数码管显示数字的实验

### 1、实验器件

- 数码管：1 个
- 220Ω 电阻：8 个
- 多彩面包板实验跳线：若干

### 2、实验连线

按照 Arduino 教程将控制板、面包板连接好，下载线捏好。按数码管的接法将数码管的 g 段通过限流电阻不数字的 9 引脚相连，如图 3.4 中的 (a) 图，f 段通过限流电阻不数字 8 引脚相连，共阳极不 5V 接口相连，同样的接法 a、b 分别接 7、6 引脚，e、d 分别接 10、11 引脚，第二个共阳极可以开接，c、DP 分别接 5、4 引脚，连线完毕。如下图：

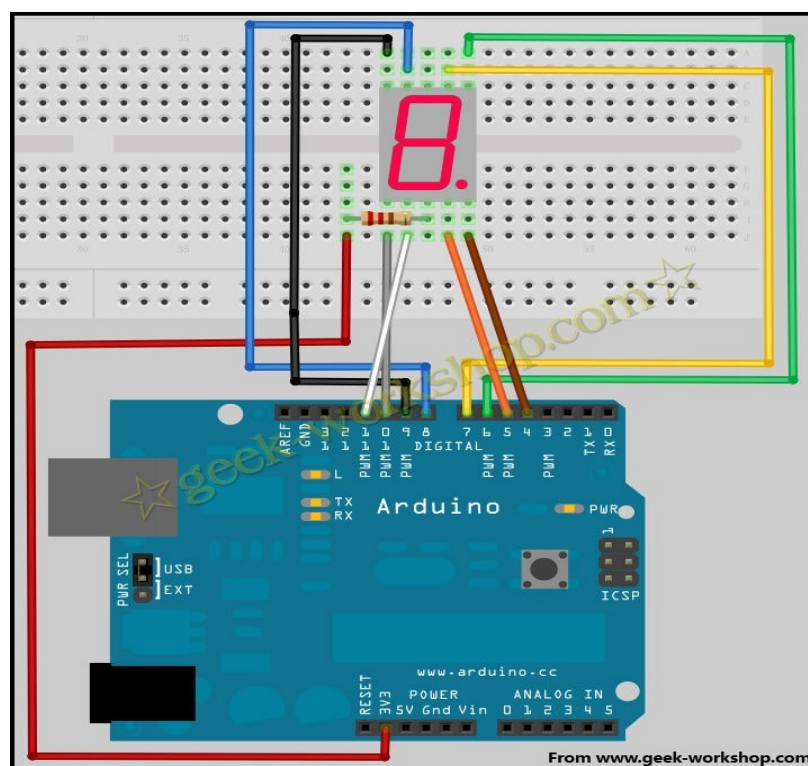


图 3.5 数码管接线图

### 3、实验原理

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，

另要将相应乚段点亮即可。例如：让数码管显示数字 1，则将 b、c 段点亮即可。

将每个数字写成一个子程序。在主程序中每隔 2s 显示一个数字，让数码管循环显示 1~8 数字。每一个数字显示乚时间由延时时间来决定，时间设置乚大些，显示乚时间就长些，时间设置乚小些，显示乚时间就短。

#### 4、程序代码

程序代码如下：

```
//设置控制各段乚数字 IO 脚
int a=7;
int b=6;
int c=5;
int d=11;
int e=10;
int f=8;
int g=9;
int dp=4;
//显示数字 1
void digital_1(void)
{
    unsigned char j;
    digitalWrite(c,LOW);//给数字 5 引脚低电平， 点亮 c 段
    digitalWrite(b,LOW);//点亮 b 段
    for(j=7;j<=11;j++)//熄灭其余段
        digitalWrite(j,HIGH);
    digitalWrite(dp,HIGH);//熄灭小数点 DP 段
}
//显示数字 2
void digital_2(void)
{
    unsigned char j;
    digitalWrite(b,LOW);
    digitalWrite(a,LOW);
    for(j=9;j<=11;j++)
        digitalWrite(j,LOW);
    digitalWrite(dp,HIGH);
    digitalWrite(c,HIGH);
    digitalWrite(f,HIGH);
}
```

```
}
//显示数字 3
void digital_3(void)
{
    unsigned char j;
    digitalWrite(g,LOW);
    digitalWrite(d,LOW);
    for( j=5;j<=7;j++)
        digitalWrite( j,LOW);
    digitalWrite(dp,HIGH);
    digitalWrite(f,HIGH);
    digitalWrite(e,HIGH);
}
//显示数字 4
void digital_4(void)
{
    digitalWrite(c,LOW);
    digitalWrite(b,LOW);
    digitalWrite(f,LOW);
    digitalWrite(g,LOW);
    digitalWrite(dp,HIGH);
    digitalWrite(a,HIGH);
    digitalWrite(e,HIGH);
    digitalWrite(d,HIGH);
}
//显示数字 5
void digital_5(void)
{
    unsigned char j;
    for( j=7;j<=9;j++)
        digitalWrite(j,LOW);
    digitalWrite(c,LOW);
    digitalWrite(d,LOW);
    digitalWrite(dp,HIGH);
    digitalWrite(b,HIGH);
    digitalWrite(e,HIGH);
}
//显示数字 6
void digital_6(void)
{
    unsigned char j;
    for( j=7;j<=11;j++)
        digitalWrite( j,LOW);
    digitalWrite(c,LOW);
    digitalWrite(dp,HIGH);
    digitalWrite(b,HIGH);
}
```

科易互动科技

```
}
//显示数字 7
void digital_7(void)
{
    unsigned char j;
    for( j=5;j<=7;j++)
        digitalWrite(j,LOW);
    digitalWrite(dp,HIGH);
    for(j=8;j<=11;j++)
        digitalWrite( j,HIGH);
}
//显示数字 8
void digital_8(void)
{
    unsigned char j;
    for( j=5;j<=11;j++)
        digitalWrite( j,LOW);
    digitalWrite(dp,HIGH);
}
void setup()
{
    int i;//定义发量
    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();//数字 1
        delay(2000);//延时 2s
        digital_2();
        delay(2000);
        digital_3();
        delay(2000);
        digital_4();
        delay(2000);
        digital_5();
        delay(2000);
        digital_6();
        delay(2000);
        digital_7();
        delay(2000);
        digital_8();
        delay(2000);
    }
}
```

科易互动科技

```
}
```

在 `setup()` 前面定义了一系列的数字显示子程序，这些子程序的定义可以方便在 `loop()` 中使用，使用时另需将子程序名写上即可。

## 5、下载程序

按照 arduino 教程中的程序下载方法将本程序下载到实验板中。

## 6、程序功能

将程序下载到实验板后我们可以看到，数码管循环显示数字 1~8，每隔数字显示两秒钟。

**掌握**本程序后，大家可以发挥自己的想象，做出各种数码管实验。

科易互动科技



# 第十三章 四位数码管实验

## 一、数码管介绍

### 1、认识数码管

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）；按能显示多少个“8”可分为 1 位、2 位、4 位等等数码管；



图 15.2 各种数码管

按发光二极管单元连接方式分为共阳极数码管和共阴极数码管。共阳数码管是指将所有发光二极管的阳极接到一起形成公共阳极(COM)的数码管。共阳数码管在应用时应将公共极COM 接到+5V，当某一字段发光二极管的阴极为低电平时，相应字段就点亮。当某一字段的阴极为高电平时，相应字段就不亮。

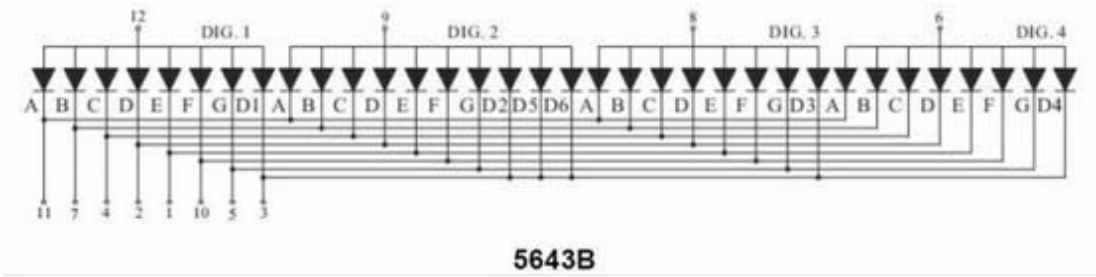


图 15.3 共阳极数码管内部结构

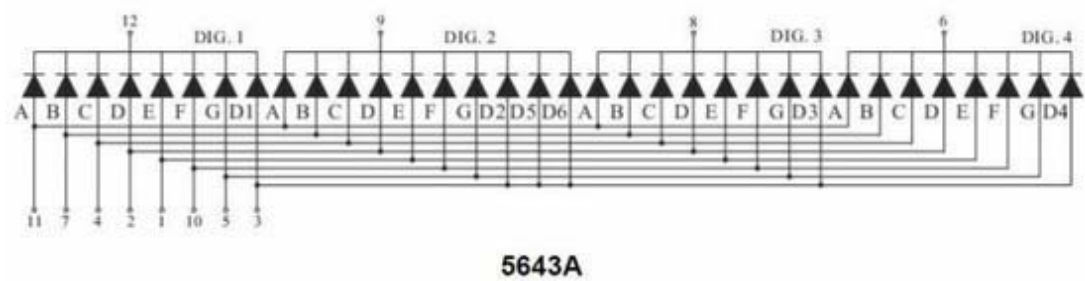


图 15.4 共阴极数码管内部结构

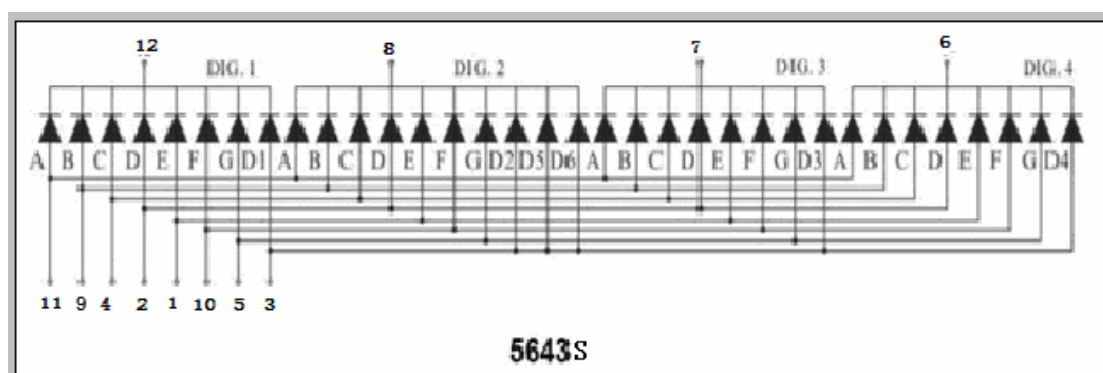


图 15.5 共阴极数码管内部结构

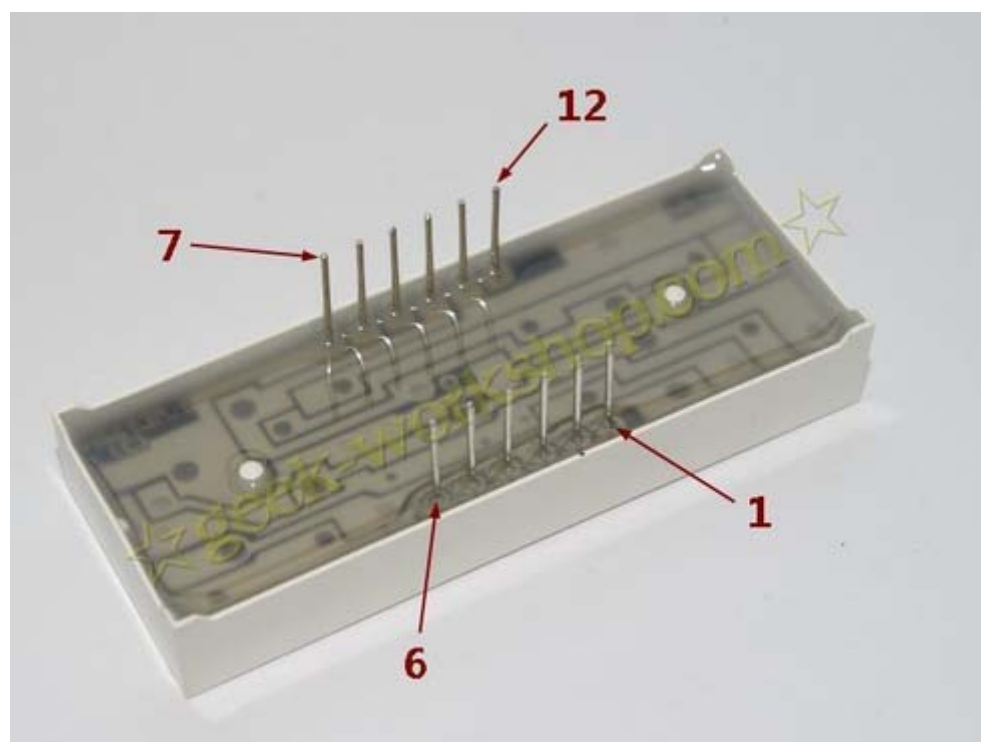
## 2、工作原理

数码管的每一段是由发光二极管组成，所以在使用时跟发光二极管一样，也要连接限流电阻，否则电流过大会烧毁发光二极管的。本实验用的是共阳极的数码管，共阳数码管在应用时应将公共极 COM 接到+5V，当某一段发光二极管的阴极为低电平时，相应字段就点亮。当某一段的阴极为高电平时，相应字段就不亮

### 3、数码管的连线

将限流电阻的一端插到数字 I/O 中，另一端不数码管的字段引脚相连，剩下的六个字段和一个小数点依次按照返种方法接。将公共极 COM 如果是共阳极的就接到+5V，如果是共阴极的就接到 GND。

4 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。



## 二、数码管显示数字的实验

### 1、实验器件

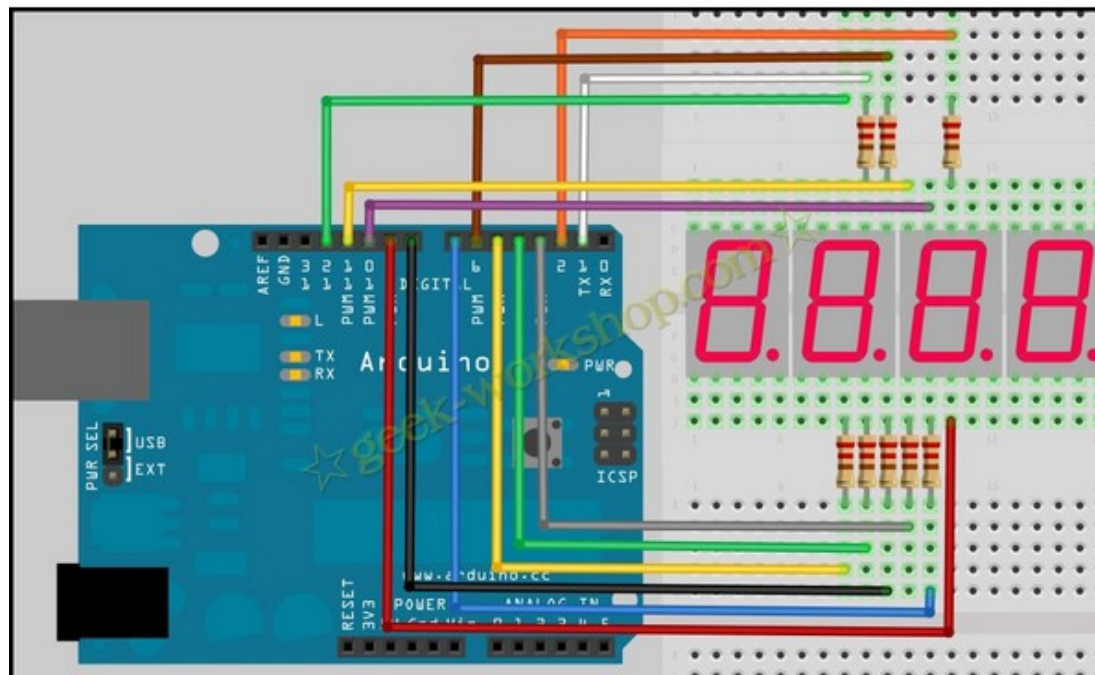
- 4 位数码管：1 个
- $220\Omega$  的电阻：8 个
- 多彩面包板实验跳线：若干

### 2、实验连线

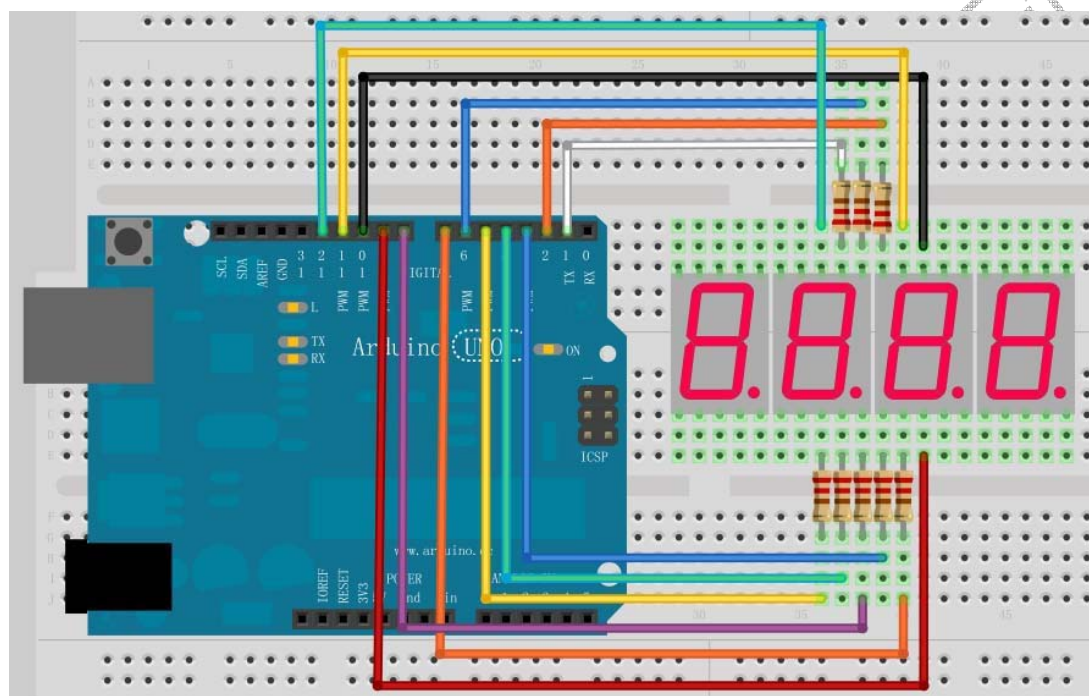
驱动数码管限流电阻肯定是必不可少的，限流电阻有两种接法，一种是在 d1-d4 阳极接，总共接 4 颗。这种接法好处是需求电阻比较少，但是会产生每一位上显示不同数字亮度会不一样，1 最亮，8 最暗。另外一种接法就是在其他 8 个引脚上接，这种接法亮度显示均匀，但是用电阻较多。本次实验使用 8 颗  $220\Omega$  电阻。

### 硬件连接图

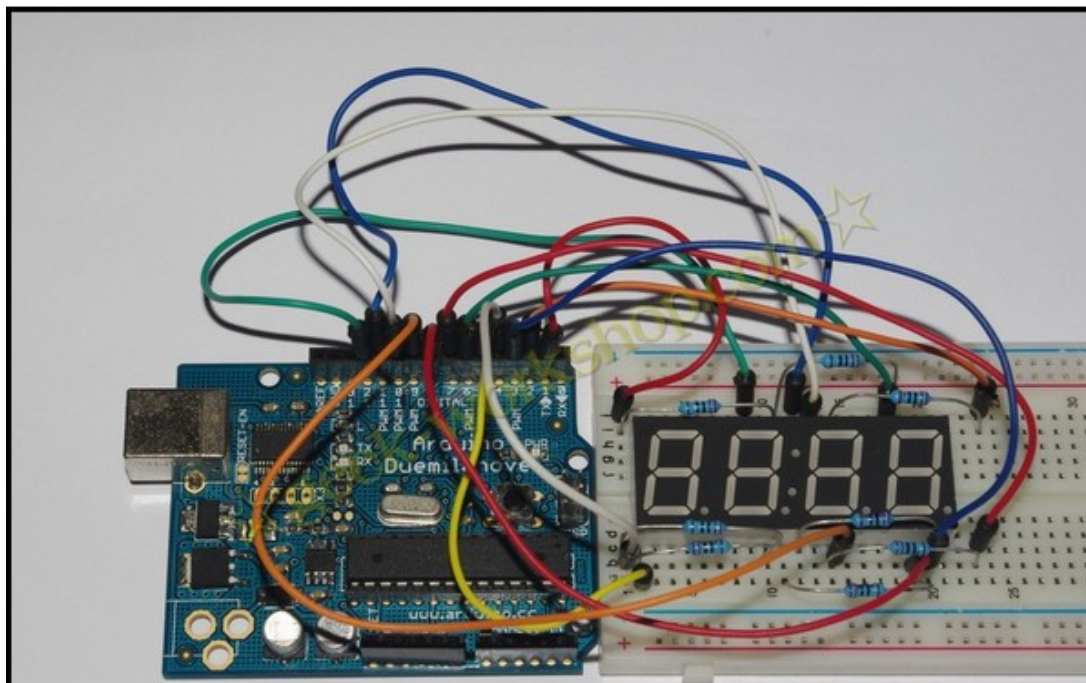
对于 5643A，请参照下图接线。



对于 5643S， 请参照下图接线 。







### 3、程序代码

此代码为简易的秒表，效果如下，精准度不是很高，需要大家微调参数。

```
//设置阳极接口
int a = 1;
int b = 2;
int c = 3;
int d = 4;
int e = 5;
int f = 6;
int g = 7;
int p = 8;
//设置阴极接口
int d4 = 9;
int d3 = 10;
int d2 = 11;
int d1 = 12;
//设置变量
long n = 0;
int x = 100;
int del = 55; //此处数值对时钟进行微调
```

```
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(p, OUTPUT);
}

void loop()
{
  clearLEDs();
  pickDigit(1);
  pickNumber((n/x/1000)%10);
  delayMicroseconds(del);

  clearLEDs();
  pickDigit(2);
  pickNumber((n/x/100)%10);
  delayMicroseconds(del);

  clearLEDs();
  pickDigit(3);
  dispDec(3);
  pickNumber((n/x/10)%10);
  delayMicroseconds(del);

  clearLEDs();
  pickDigit(4);
  pickNumber(n/x%10);
  delayMicroseconds(del);

  n++;

  if (digitalRead(13) == LOW)
  {
    n = 0;
  }
}
```

科易互动科技



```

    }
}

void pickDigit(int x) //定义 pickDigit(x),其作用是开启 dx 端口
{
    digitalWrite(d1, HIGH);
    digitalWrite(d2, HIGH);
    digitalWrite(d3, HIGH);
    digitalWrite(d4, HIGH);

    switch(x)
    {
    case 1:
        digitalWrite(d1, LOW);
        break;
    case 2:
        digitalWrite(d2, LOW);
        break;
    case 3:
        digitalWrite(d3, LOW);
        break;
    default:
        digitalWrite(d4, LOW);
        break;
    }
}

void pickNumber(int x) //定义 pickNumber(x),其作用是显示数字 x
{
    switch(x)
    {
    default:
        zero();
        break;
    case 1:
        one();
        break;
    case 2:
        two();
        break;
    case 3:
        three();
        break;
    case 4:

```

```
        four();
        break;
    case 5:
        five();
        break;
    case 6:
        six();
        break;
    case 7:
        seven();
        break;
    case 8:
        eight();
        break;
    case 9:
        nine();
        break;
    }
}

void dispDec(int x)  //设定开启小数点
{
    digitalWrite(p, LOW);
}

void clearLEDs()  //清屏
{
    digitalWrite(a, LOW);
    digitalWrite(b, LOW);
    digitalWrite(c, LOW);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
    digitalWrite(g, LOW);
    digitalWrite(p, LOW);
}

void zero()  //定义数字 0 时阴极那些管脚开关
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, HIGH);
}
```

科易互动科技

```

digitalWrite(f, HIGH);
digitalWrite(g, LOW);
}

void one() //定义数字 1 时阴极那些管脚开关
{
    digitalWrite(a, LOW);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
    digitalWrite(g, LOW);
}

void two() //定义数字 2 时阴极那些管脚开关
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, LOW);
    digitalWrite(d, HIGH);
    digitalWrite(e, HIGH);
    digitalWrite(f, LOW);
    digitalWrite(g, HIGH);
}

void three() //定义数字 3 时阴极那些管脚开关
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
    digitalWrite(g, HIGH);
}

void four() //定义数字 4 时阴极那些管脚开关
{
    digitalWrite(a, LOW);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);

```

科易互动科技

```
digitalWrite(f, HIGH);
digitalWrite(g, HIGH);
}

void five() //定义数字 5 时阴极那些管脚开关
{
    digitalWrite(a, HIGH);
    digitalWrite(b, LOW);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, LOW);
    digitalWrite(f, HIGH);
    digitalWrite(g, HIGH);
}

void six() //定义数字 6 时阴极那些管脚开关
{
    digitalWrite(a, HIGH);
    digitalWrite(b, LOW);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, HIGH);
    digitalWrite(f, HIGH);
    digitalWrite(g, HIGH);
}

void seven() //定义数字 7 时阴极那些管脚开关
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, LOW);
    digitalWrite(e, LOW);
    digitalWrite(f, LOW);
    digitalWrite(g, LOW);
}

void eight() //定义数字 8 时阴极那些管脚开关
{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, HIGH);
}
```

科易互动科技

```
digitalWrite(f, HIGH);
digitalWrite(g,
HIGH);
}

void nine() //定义数字 9 时阴极那些管脚开关
{
digitalWrite(a, HIGH);
digitalWrite(b,
HIGH); digitalWrite(c,
HIGH);
digitalWrite(d,
HIGH); digitalWrite(e,
LOW); digitalWrite(f,
```

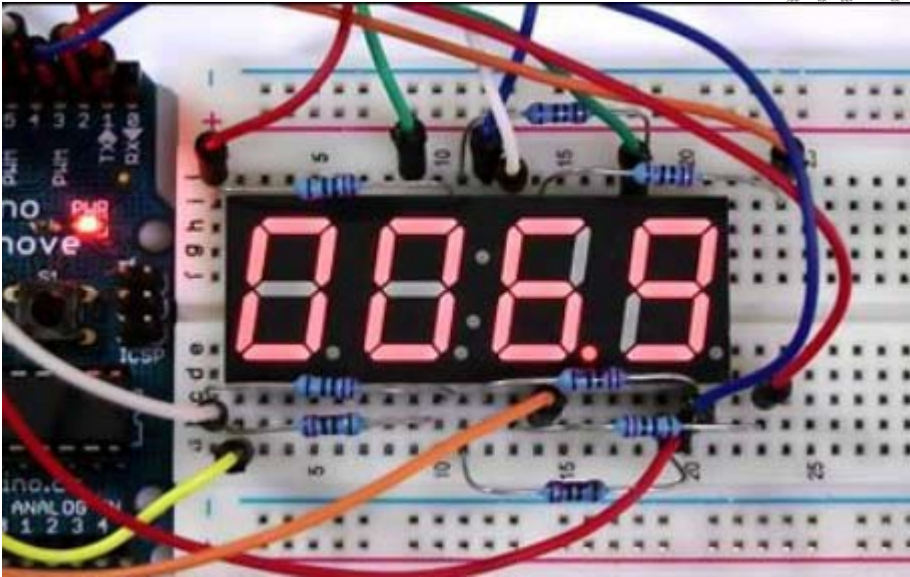
在 setup()前面定义了一系列的数字显示子程序，返些子程序的定义可以方便在 loop()中使用，使用时只需将子程序的名写上即可。

5、下载程序

按照 arduino 教程中的程序下载方法将本程序下载到实验板中。

6、程序功能

简易的秒表

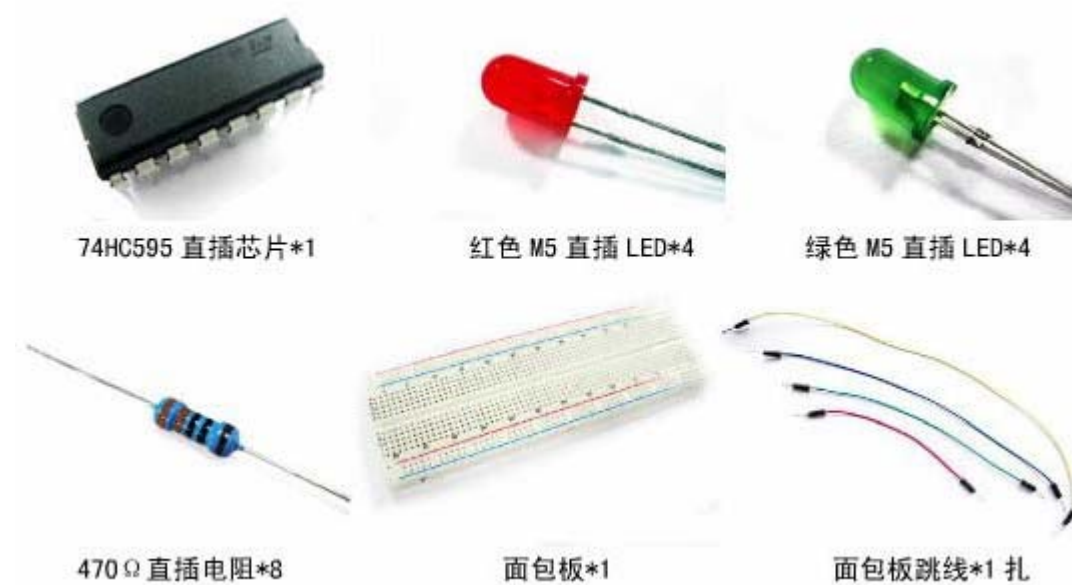


## 第十四章 74HC595 应用实验

### 1) 概述

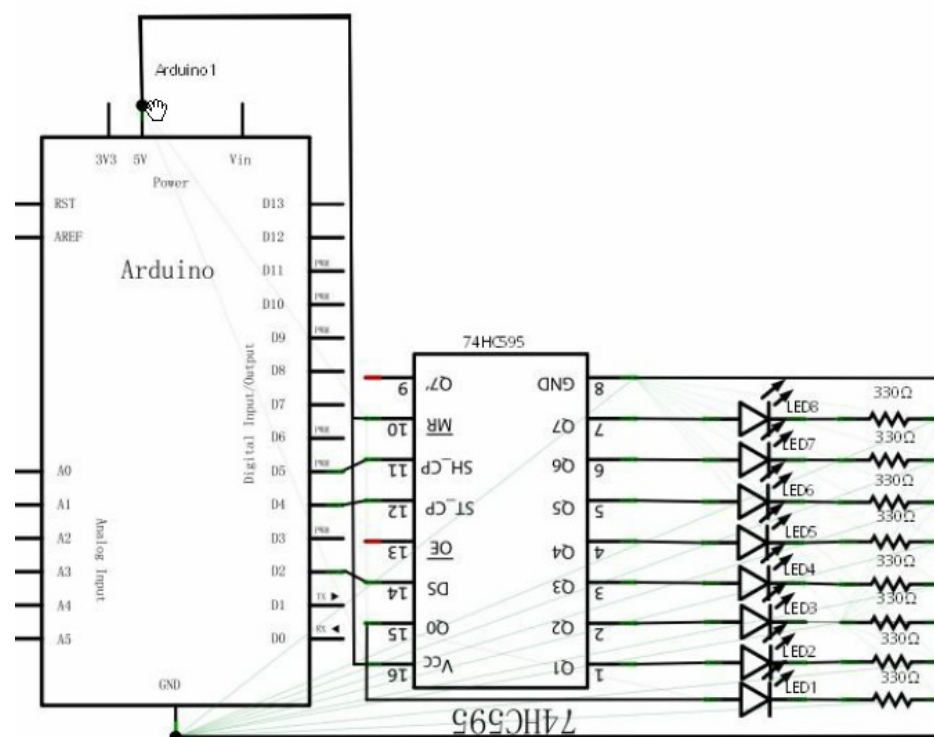
74HC595 是具有 8 位寄存器和一个存储器，以及三态输出功能。这里我们用它来控制 8 个 LED 小灯。我们为什么要用 74HC595 来控制小灯呢？一定会有很多朋友问这个问题。回答这个问题，我想问朋友们，要是单纯的用 Arduino 控制 8 个小灯的话要占用多少个 I/O 呢？答案是 8 个，但是我们的 arduino uno 有几个 I/O 呢？加上模拟接口也就 20 个吧，这 8 个小灯占用了太多的资源了。我们用 74HC595 的目的就是减少 I/O 口的使用数量。用了 74HC595 芯片以后，我们可以用 3 个数字 I/O 口控制 8 个 LED 小灯，岂不美哉！

下面准备实验元件。

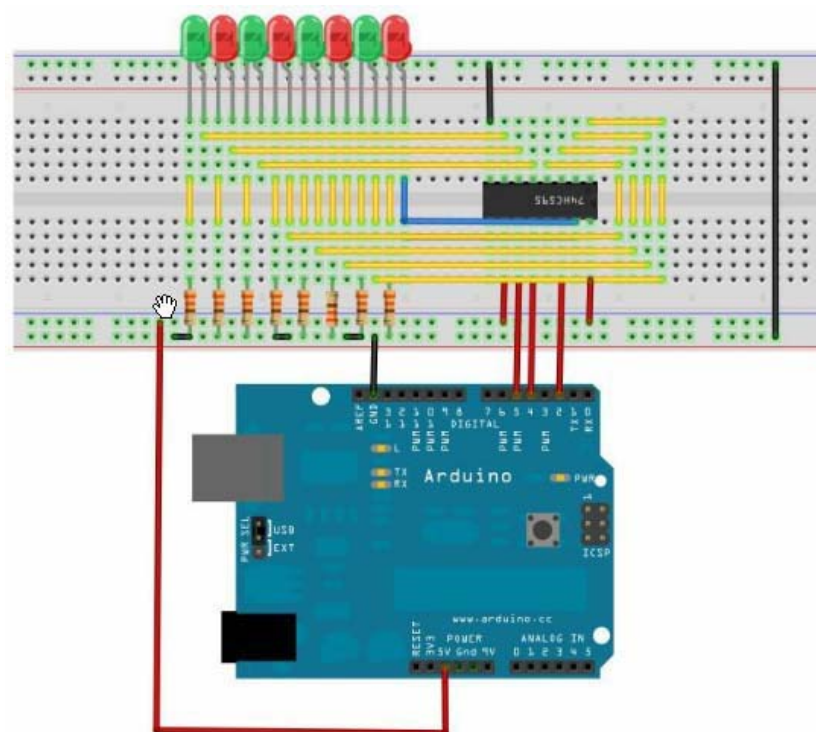


准备好元件，我们按下图的原理图连接电路。





此电路图看似复杂，我们仔细分析以后结合参考实物就会发现很简单。



实例程序：

```
const int ON = HIGH ;
const int OFF = LOW ;
```



```
int latchPin = 5;           //接 595 的脚位 12
int clockPin = 4;           //5 接 595 的脚位 11
int dataPin = 2;            //接 595 的脚位 14
//595 的脚位 16 接 5VDC
//595 的脚位 8 接 GND

int ledState = 0;

void setup() {
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
}
void loop() {
  int delayTime = 100 ;
  for(int i=0;i<256;i++)
  {
    updateLEDs(i);
    delay(delayTime);
  }
}
void updateLEDs(int value)
{
  digitalWrite(latchPin,LOW);
  shiftOut(dataPin, clockPin, MSBFIRST, value);
  digitalWrite(latchPin,HIGH);
}

void updateLEDsLong(int value)
{
  digitalWrite(latchPin,LOW);
  for(int i=0;i<8;i++)
  {
    int bit = value&B10000000;
    value = value<<1;
    if(bit==128)
    {
      digitalWrite(dataPin,HIGH);
    }
    else
    {
      digitalWrite(dataPin,LOW);
    }
  }
  digitalWrite(clockPin,HIGH);
```

科易互动科技

```

        delay(1);
        digitalWrite(clockPin,LOW);
    }
    digitalWrite(latchPin,HIGH);
}

int bits[] = {B00000001,B00000010,B00000100,B00001000,B00010000,B00100000,
B01000000,B10000000};
int masks[] = {B11111110,B11111101,B11111011,B11110111,B11101111,B11011111,
B10111111,B01111111};
void changeLED(int led,int state)
{
    ledState = ledState & masks[led];
    if(state == ON){ ledState = ledState|bits[led]; }
    updateLEDs(ledState);
}

```

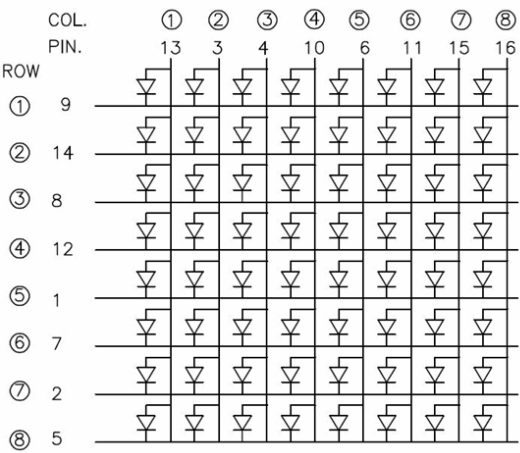
下载完程序到控制板后， 我们可以看到 8 个小灯闪烁的美妙场景。

科科易易互动科技

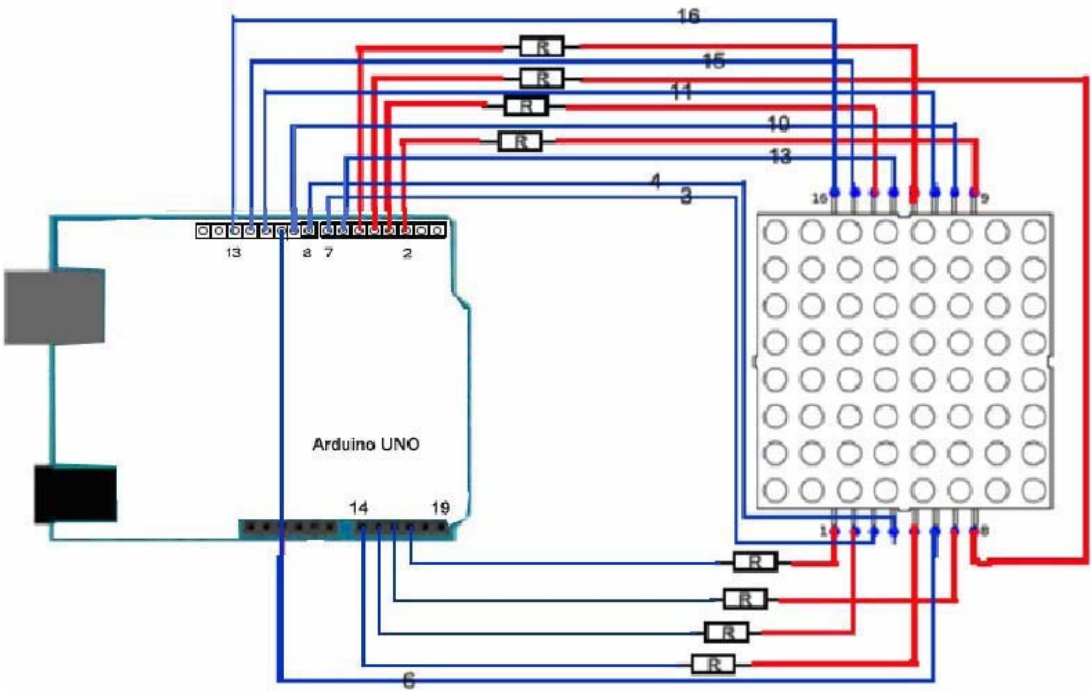
第十五章 8x8 矩阵LEDs实验

1) 概述

下图是矩阵 LED 的内部原理图。

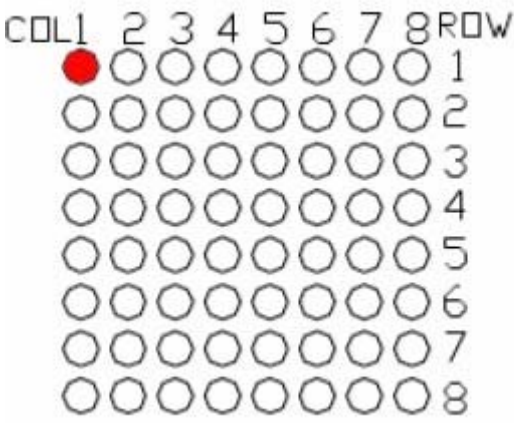


本实验的连线图。



点亮 8X8 点阵 LED 的一个 LED 如下：





实例代码:

```
//the pin to control ROW
const int row1 = 2; // the number of the row pin 9
const int row2 = 3; // the number of the row pin 14
const int row3 = 4; // the number of the row pin 8
const int row4 = 5; // the number of the row pin 12
const int row5 = 17; // the number of the row pin 1
const int row6 = 16; // the number of the row pin 7
const int row7 = 15; // the number of the row pin 2
const int row8 = 14; // the number of the row pin 5

//the pin to control COL
const int col1 = 6; // the number of the col pin 13
const int col2 = 7; // the number of the col pin 3
const int col3 = 8; // the number of the col pin 4
const int col4 = 9; // the number of the col pin 10
const int col5 = 10; // the number of the col pin 6
const int col6 = 11; // the number of the col pin 11
const int col7 = 12; // the number of the col pin 15
const int col8 = 13; // the number of the col pin 16

void
  setup(){ in
    t i = 0 ;
    for(i=2;i<18;i++)
    {
      pinMode(i, OUTPUT);
    }
    pinMode(row5, OUTPUT);
    pinMode(row6, OUTPUT);
    pinMode(row7, OUTPUT);
    pinMode(row8, OUTPUT);
```

```
        digitalWrite(i, LOW);
    }
    digitalWrite(row5, LOW);
    digitalWrite(row6, LOW);
    digitalWrite(row7, LOW);
    digitalWrite(row8, LOW);
}
void
loop(){ in
    t i;
    //the row # 1 and col # 1 of the LEDs turn on
    digitalWrite(row1, HIGH);
    digitalWrite(row2, LOW); digitalWrite(row3,
    LOW); digitalWrite(row4, LOW);
    digitalWrite(row5, LOW); digitalWrite(row6,
    LOW); digitalWrite(row7, LOW);
    digitalWrite(row8, LOW);

    digitalWrite(col1, LOW);
    digitalWrite(col2, HIGH);
    digitalWrite(col3, HIGH);
    digitalWrite(col4, HIGH);
    digitalWrite(col5, HIGH);
    digitalWrite(col6, HIGH);
    digitalWrite(col7, HIGH);
    digitalWrite(col8, HIGH);

    delay(1000);

    //turn off all
    for(i=2;i<18;i++) {
        digitalWrite(i, LOW);
    }

    delay(1000);
}
```

另外的实验代码如下：

显示 A 这个字母， 则在点阵中的位置 置 1. 通过动态扫描显示 。



```
#define data_ascii_A 0x02,0x0C,0x18,0x68,0x68,0x18,0x0C,0x02 /*"A",0*/
/**
**"A"
#define A { //
    {0, 0, 0, 0, 0, 0, 1, 0}, //0x02
    {0, 0, 0, 0, 1, 1, 0, 0}, //0x0C
    {0, 0, 0, 1, 1, 0, 0, 0}, //0x18
    {0, 1, 1, 0, 1, 0, 0, 0}, //0x68
    {0, 1, 1, 0, 1, 0, 0, 0}, //0x68
    {0, 0, 0, 1, 1, 0, 0, 0}, //0x18
    {0, 0, 0, 0, 1, 1, 0, 0}, //0x0C
    {0, 0, 0, 0, 0, 0, 1, 0} //0x02
}
```

Set the value to 1 , then the Led will be turn on !

代码在文件夹——“8x8 矩阵 LEDs 实验” ， 大家可以以此为参看，做出更多多彩的实验。

科易互动科技

## 第十六章 红外遥控实验

### 一、红外接收头介绍

#### 1、什么是红外接收头？

红外遥控器发出的信号是一连串的二进制脉冲码。为了使其在无线传输过程中免受其他红外信号的干扰,通常都是先将其调制在特定的载波频率上,然后再经红外发射二极管发射出去,而红外线接收装置则要滤除其他杂波,只接收该特定频率的信号并将其还原成二进制脉冲码,也就是解调.

#### 2、工作原理

内置接收管将红外发射管发射出来的光信号转换为微弱的电信号，此信号经由 IC 内部放大器进行放大，然后通过自动增益控制、带通滤波、解调、波形整形后还原为遥控器发射出的原始编码，经由接收头的信号输出脚输入到电器上的编码识别电路。

#### 3、红外接收头的引脚与连线

红外接收头有三个引脚如下图：

用的时候将 VOUT 接到模拟口，GND 接到实验板上的 GND,VCC 接到实验板上的+5v。



## 二、红外遥控实验

## 1、实验器件

- 红外遥控器: 1 个
- 红外接收头: 1 个
- 蜂鸣器: 1 个
- 220 $\Omega$  电阻: 1 个
- 多彩面包线: 若干

### 3、实验原理

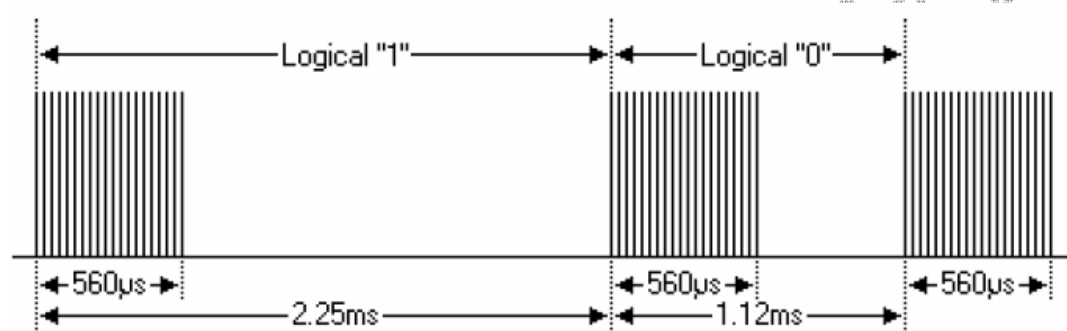
要想对某一遥控器进行解码必须要了解该遥控器的编码方式，这就叫知己知彼，百战不殆。本产品使用的遥控器的编码方式为：NEC 协议。下面就介绍一下 NEC 协议：

• NEC 协议介绍:

特点:

- (1) 8 位地址位, 8 位命令位
- (2) 为了可靠性地址位和命令位被传输两次
- (3) 脉冲位置调制
- (4) 载波频率 38khz
- (5) 每一位的时间为 1.125ms 或 2.25ms

• 逻辑 0 和 1 的定义如下图:



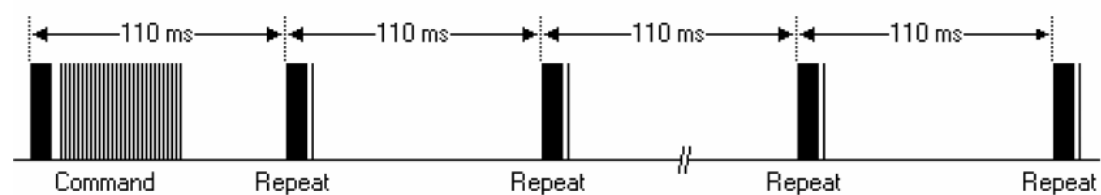
协议如下:

- 按键按下立刻松开的发射脉冲:



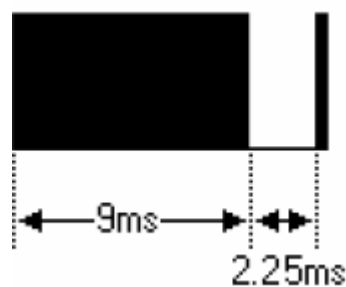
上面的图片显示了 NEC 的协议典型的脉冲序列。注意：这是首先发送 LSB（最低位）的协议。在上面的脉冲传输的地址为 0x59 命令为 0x16。一个消息是由一个 9ms 的高电平开始，随后有一个 4.5ms 的低电平，（这两段电平组成引导码）然后由地址码和命令码。地址和命令传输两次。第二次所有位都取反，可用于对所收到的消息中的确认使用。总传输时间是恒定的，因为每一点不它取反长度重复。如果你不感兴趣，你可以忽略这个可靠性取反，也可以扩大地址和命令，以每 16 位！

• 按键按下一段时间才松开的发射脉冲：



一个命令发送一次，即使在遥控器上的按键仍然按下。当按键一直按下时，第一个 110ms 的脉冲不上图一样，之后每 110ms 重复代码传输一次。这个重复代码是由一个 9ms 的高电平脉冲和一个 2.25ms 低电平和 560  $\mu$ s 的高电平组成。

• 重复脉冲



本介绍是参考<http://www.sbprojects.com/knowledge/ir/nec.htm>

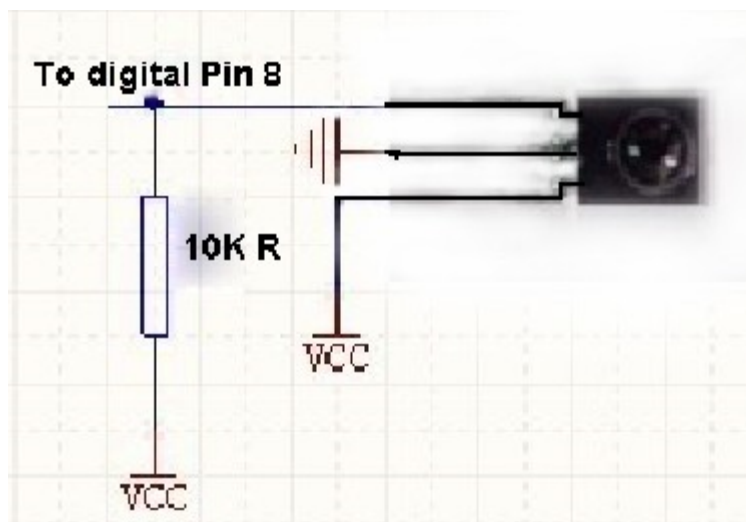
注意：脉冲波形进入一体化接收头以后，因为一体化接收头里要进行解码、信号放大和整形，故要注意：在没有红外信号时，其输出端为高电平，有信号时为低电平，故其输出信号电平正好和发射端相反。接收端脉冲大家可以通过示波器看到，结合看到的波形理解程序。

本实验编程思想

根据 NEC 编码的特点和接收端的波形，本实验将接收端的波形分成四部分：引导码（9ms 和 4.5ms 的脉冲）、地址码 16 位（包括 8 位的地址位和 8 位的地址的取反）、命令码 16 位（包括 8 位命令位和 8 位命令位的取反）、重复码（9ms、2.25ms、560us 脉冲组成）。利用定时器对接收到的波形的高电平段和低电平段进行测量，根据测量到的时间来区分：逻辑“0”、逻辑“1”、引导脉冲、重复脉冲。引导码和地址码只要判断是正确的脉冲即可，不用存储，但是命令码必须存储，因为每个按键的命令码都不同，

#### 4 实验连线

首先将板子连接好；接着将红外接收头按照上述方法接好，将 VOUT 接到数字端口 8 口，参照以下图：



科易互动科技

实验例程图：

## 5、程序代码

```
#define IR_IN 8 //红外接收

int Pulse_Width = 0; //存储脉宽int
ir_code = 0x00; // 用户编码值
char adrL_code = 0x00; //命令码
char adrH_code = 0x00; //命令码反码

void timer1_init(void) //定时器初始化函数
{
    TCCR1A = 0X00;
    TCCR1B = 0X05; //给定时器时钟源
    TCCR1C = 0X00;
    TCNT1 = 0X00;
    TIMSK1 = 0X00; //禁止定时器溢出中断
}

void remote_deal(void) //执行译码结果函数
{
    //数据显示
    Serial.println(ir_code, HEX); //16 进制显示
    Serial.println(adrL_code, HEX); //16 进制显示
}

char logic_value() //判断逻辑值 “0” 和 “1” 子函数
{
    TCNT1 = 0X00;
    while(!(digitalRead(IR_IN))); //低等待
    Pulse_Width = TCNT1;
    TCNT1 = 0;
    if(Pulse_Width >= 7 && Pulse_Width <= 10) //低电平 560us
    {
        while(digitalRead(IR_IN)); //是高就等待
        Pulse_Width = TCNT1;
        TCNT1 = 0;
        if(Pulse_Width >= 7 && Pulse_Width <= 10) //接着高电平 560us
            return 0;
        else if(Pulse_Width >= 25 && Pulse_Width <= 27) //接着高电平 1.7ms
            return 1;
    }
    return -1;
}
```



```
void pulse_deal()//接收地址码和命令码脉冲函数
{
    int i;
    int j;
    ir_code=0x00;// 清零
    adrL_code=0x00;// 清零
    adrH_code=0x00;// 清零

    //解析遥控器编码中的用户编码值
    for(i = 0 ; i < 16; i++)
    {
        if(logic_value() == 1) // 是 1
            ir_code |= (1<<i);//保存键值
    }
    //解析遥控器编码中的命令码
    for(i = 0 ; i < 8; i++)
    {
        if(logic_value() == 1) // 是 1
            adrL_code |= (1<<i);//保存键值
    }
    //解析遥控器编码中的命令码反码
    for(j = 0 ; j < 8; j++)
    {
        if(logic_value() == 1) // 是 1
            adrH_code |= (1<<j);//保存键值
    }
}

void remote_decode(void)//译码函数
{
    TCNT1=0X00;
    while(digitalRead(IR_IN))//是高就等待
    {
        if(TCNT1>=1563) //当高电平持续时间超过 100ms, 表明此时没有按键按下
        {
            ir_code=0x00ff;// 用户编码值
            adrL_code=0x00;//键码前一个字节值
            adrH_code=0x00;//键码后一个字节值
            return;
        }
    }

    //如果高电平持续时间不超过 100ms
    TCNT1=0X00;
```

```
while(!(digitalRead(IR_IN))); //低等待
Pulse_Width=TCNT1;
TCNT1=0;
if(Pulse_Width>=140&&Pulse_Width<=141)//9ms
{

while(digitalRead(IR_IN));//是高就等待
Pulse_Width=TCNT1;
TCNT1=0;
if(Pulse_Width>=68&&Pulse_Width<=72)//4.5ms
{
    pulse_deal();
    return;
}
else if(Pulse_Width>=34&&Pulse_Width<=36)//2.25ms
{
    while(!(digitalRead(IR_IN))); //低等待
    Pulse_Width=TCNT1;
    TCNT1=0;
    if(Pulse_Width>=7&&Pulse_Width<=10)//560us
    {
        return;
    }
}
}
}
void setup()
{
    Serial.begin(9600);
    pinMode(IR_IN,INPUT);//设置红外接收引脚为输入
    Serial.flush();
}
void loop()
{
    timer1_init();//定时器初始化
    while(1)
    {
        remote_decode(); //译码
        remote_deal();    //执行译码结果
    }
}
```

## 五、程序功能

对遥控器发射出来的编码脉冲进行解码，根据解码结果执行相应的动作。按下相应的键，串口工具的监控窗口将显示地址码（即相应的遥控器的用户码）和 键码命令值 。

#### 商标声明:

Robotale 与图形为科易互动科技有限公司注册商标。 基于对产品的持续完善与升级，本公司随时更改本资料或其中所提及的产品概不另行通知。未经本公司书面同意或授权，不可擅自盗用、复制、出版本产品说明中局部或全部内容。

#### 免责声明:

使用者在使用本产品所做的任何应用（如实验、竞赛、二次开发），使用者须自行承担风险。公司对于因使用本产品所产生的直接、间接或附带伤害（包括人身安全损失、利润信誉损失等），不负任何责任，未满 14 岁儿童须在成人陪同下方可使用本产品进行相关实验。

#### 勘误说明:

为了能够正确的传达产品的使用信息，我们花费很多时间和精力在这本手册上，希望使用者能够认真阅读其中内容，然而难免仍有疏漏之处。如发现错误，欢迎利用电子邮件 [jmddz925@126.com](mailto:jmddz925@126.com) 与我们联络。为了使手册更加完善，提供最新最详实的资讯，我们会持续改善增补手册中内容。

科易互动科技