高中通用技术课程

Arduino 电子控制技术教程

学习　探索　设计　实践

北京师范大学第二附属中学

前言

Arduino 是当今最流行的电子创意开发平台之一。无论是 IT 界的专业人士、研究人员，还是广大的技术爱好者和“创客”，都可以将 Arduino 作为手中的创新利器，进行对声、光、电、机的控制，完成从互动媒体艺术到机器人、物联网、科学仪器等无数的项目，做到创意无限，精彩多多！

本教程是北师大二附中通用技术必修课“控制与设计”和选修课《Arduino 电子技术》的校本教材，也可作为各种科技活动、研究性学习活动的参考资料。教程采用编者独创的“基于互联网社区的技术学习模式”，提倡“做中学”，在实践中掌握 Arduino 相关知识。这一学习模式特别强调互联网开源社区的作用，吸收了其中大量优秀案例和资源。学生学习一段时间后，就可以无障碍的自主利用互联网上浩如烟海的资源来实现自己的创意，学到自己想学的东西，避免了传统模式下“老师教什么，学生会什么；老师有什么，学生做什么”的问题。

本教程采用的学习模式可以概括为“抄”，“懂”，“改”三个字。“抄”是指将教程或互联网社区中的经典项目调试运行成功，形成对技术项目的感性认识；“懂”就是根据项目附带的注释或说明，弄懂项目运行的全过程，把感性认识上升到理性认识；“改”的过程实际上是活学活用所学知识，实现自己创意的过程。全过程模拟了专业人员在实际工作中快速学习的过程。

希望大家通过本课程的学习，不仅学会 Arduino 相关的知识和技能，快速实现独特的创意；也能掌握这一技术学习模式，为今后的长远发展打下坚实的基础。创客的大门向每一个人敞开，让我们一起走上快乐的创新之路！

创客格言

别再玩推特了，赶快站起来制作吧。

Leave tweeter, go make something.

别总是陷于空想，否则你什么都做不出来。

Don’t get so caught up in thinking that you never do anything.

量要量两次，切只切一次。

Measure twice. Cut once.

不要买便宜的螺丝起子。

Never buy cheap screwdrivers.

千万别在疲劳的时候制作东西。

Never work when you are tired.

别用手指确认热熔胶是否冷却了。它还没有。

Never use your finger to check if the hot glue has cooled down. It hasn’t.

如果借来的工具是干净的，干干净净的还回去。如果工具借来时是脏的，也要干干净净的还回去。

If you borrow a clean tool, return it clean. If you borrow a dirty tool, return it clean.

理论上，理论与实践应该是一样的。但在实践中，理论与实践是不一样的。

In theory, there is no difference between theory and practice. In practice, there is.

不要害怕拆卸东西。在学习修理东西的过程中，克服这种恐惧往往是最难的。

Don’t be afraid to take things apart. Overcoming that fear is often the hardest part of fixing things.

先做好规划再抄家伙干活。

Pick your tool after you plan your job.

目录

第一章 Arduino 简介

1.1 Arduino 套件简介

1 Arduino 介绍

Arduino 是一块简单易学的电子创意设计平台，它让你可以快速做出有趣的东西。Arduino 可以配合一些电子元件使用，例如 LED 灯、蜂鸣器、按键、光敏电阻等等。Arduino 配套软件基于“开放”原则，可以让您免费下载使用，开发出更多令人惊奇的互动作品。

2 设备参数及组成

官方说明：

采用 Atmel 微处理控制器。Arduino 大小尺寸：宽 70mm X 高 54mm。

数字输入/输出端共 0~13，注意 0 和 1（RX TX）用于和电脑通信，不要占用。3，5，6，9，10，11 口可输出 8bit (0-255)PWM 信号

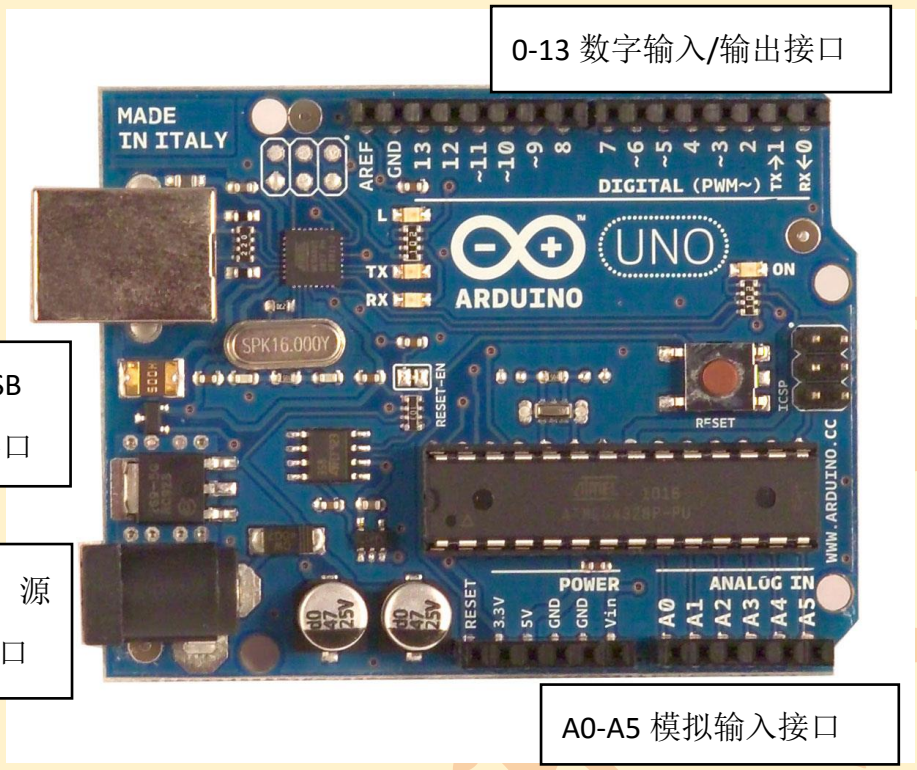
模拟输入端共 0~5。可以检测 0-5V 信号，10bit 量化（0-1023）

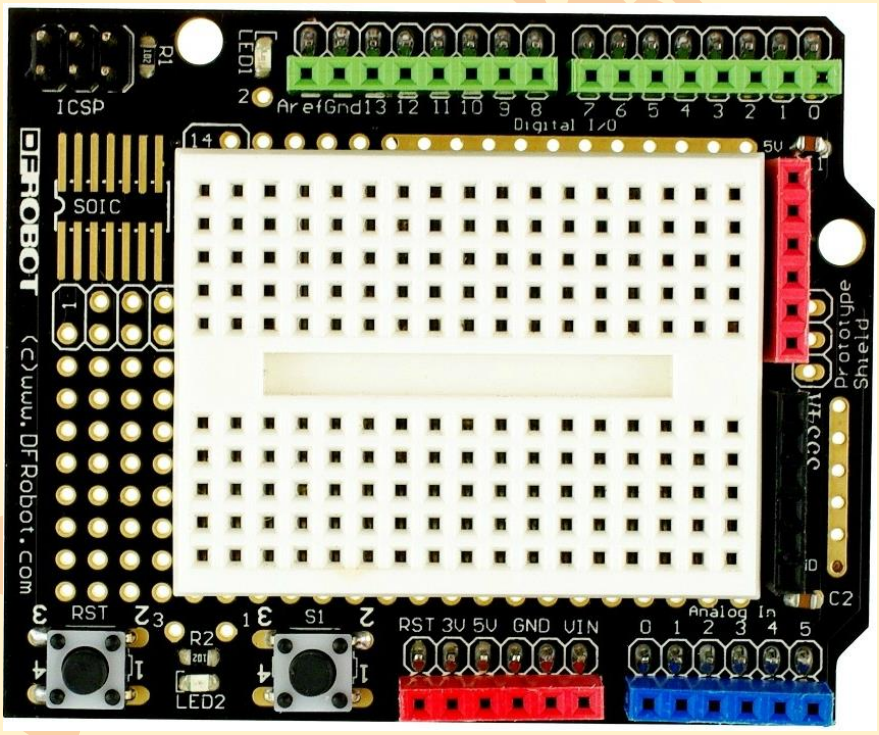
输入电压：

（1）USB 供电。左侧为 USB 接口和电源接口。我们在使用时，将 USB 线分别接在 Arduino 和电脑 USB 口上。供电电压 5V。

（2） 外部 7V~12V 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


0-13 数字输入/输出接口
USB
接口
电 源
接口
A0-A5 模拟输入接口
MADE
IN ITALY
DIGITAL (PWM~)
UNO
ARDUINO
POWER
ANALOG IN
WWW.ARDUINO.CC


ICSP
SOIC
DFROBOT
(c)www.DFRobot.com
Digital I/O
Prototype
Shield
RST
S1
LED2
Analog In

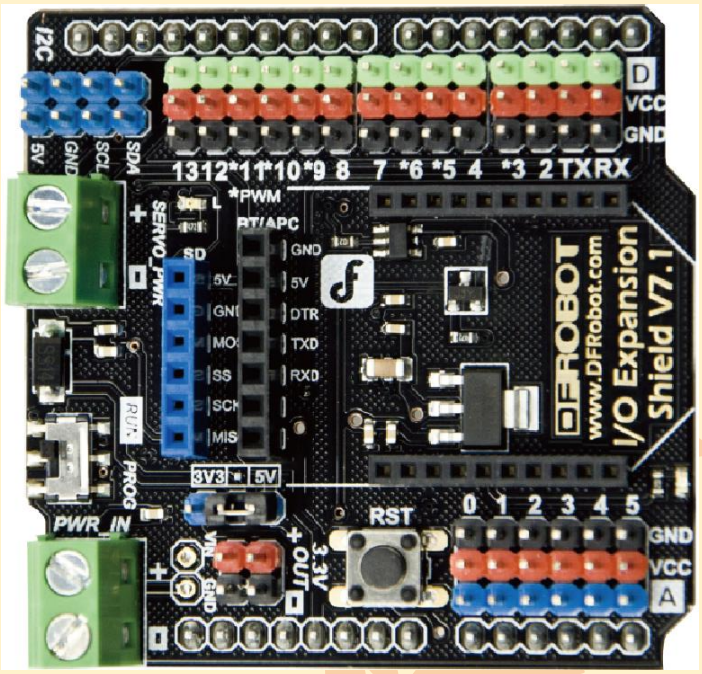
0-13 数字输入/输出接口

0-13 数字输入/输出接口

红正（5V，VCC），黑负（GND，0V，地）

A 为模拟输入口，一般接传感器

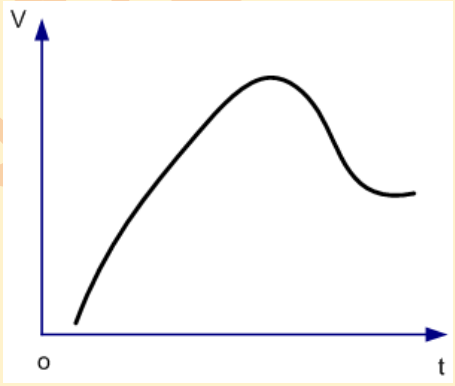
D 为数字输入输出口，一般接被控对象，也可接开关型传感器



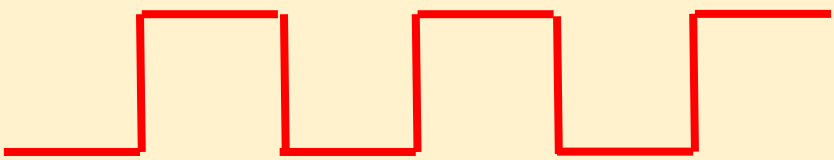
A0-A5 模拟输入接口

小知识

“数字”信号只有 2 种状态：高电平（电源正极电压，这里为 5V）低电平（电源负极电压，这里为 0V）。而“模拟”信号是“连续”的，在 Arduino 上可以是 0-5V 中任何数值。



模拟信号



数字信号

在 Arduino 上，Digital（数字）端口常用于“输出”。它只能输出数字信号，即 0V 或 5V。

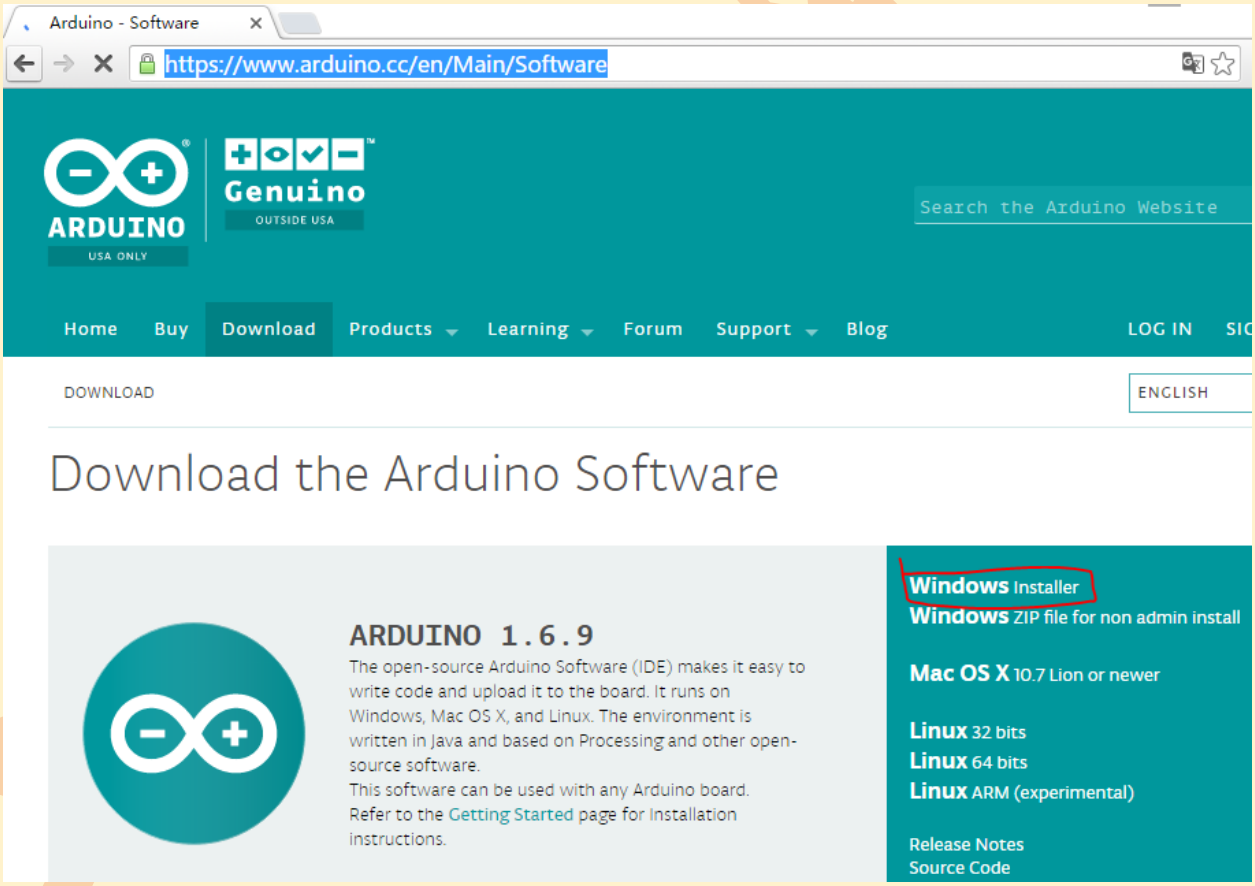
因此我们可以把它看作是一个特殊的电源，能够根据我们的需要输出 0V 或 5V 电压。

Analog In 用于“输入”，也就是检测模拟信号。这里我们可以把它看作是一个量程为 5V 的电压表。

1.2 准备工作：Arduino 初始设置

1.2.1 软件安装

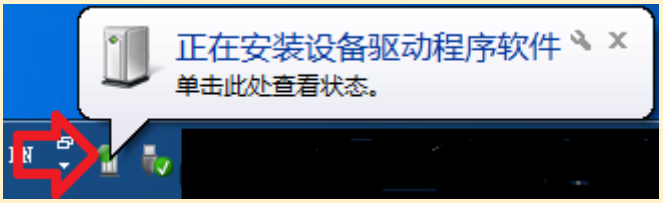
打开 https://www.arduino.cc/en/Main/Software 网站。下载 Windows Installer 安装包。



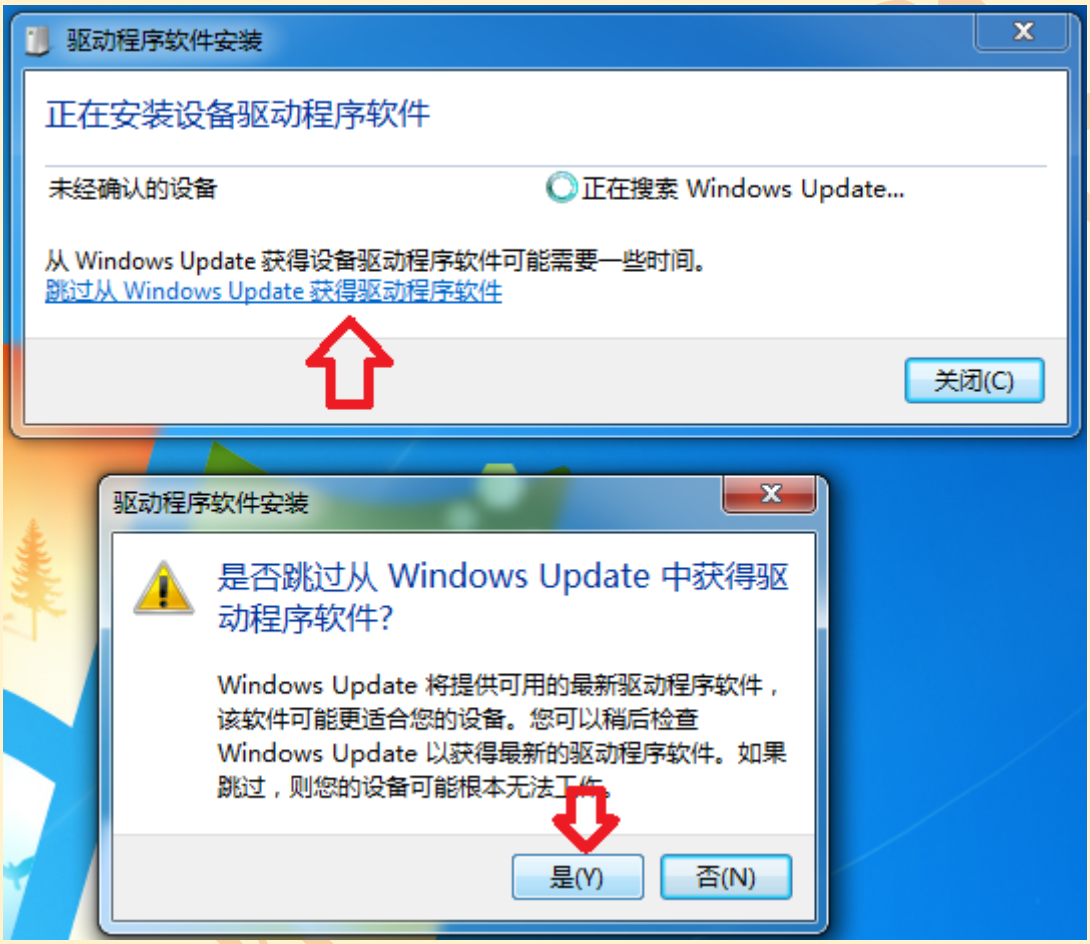
然后用默认选项安装即可。

1.2.2 硬件安装

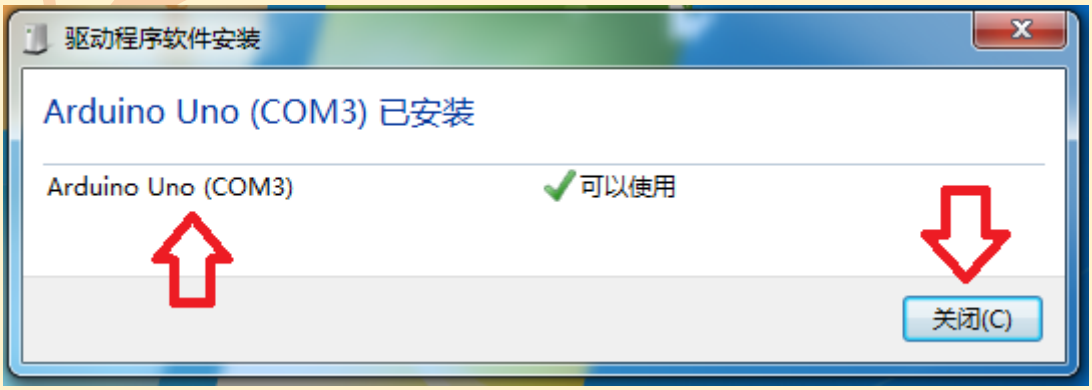
插入 USB 线，Arduino 控制板上的电源指示灯会被点亮，电脑右下角会出现:



点击这个图标：出现下面的对话框，然后点击“跳过从 Windows Update 获得驱动程序软件”，然后点“是”



然后会从本机安装，完成后显示下面对话框，记住它安装到 COM 几，点关闭。



第二章 Arduino 数字输出

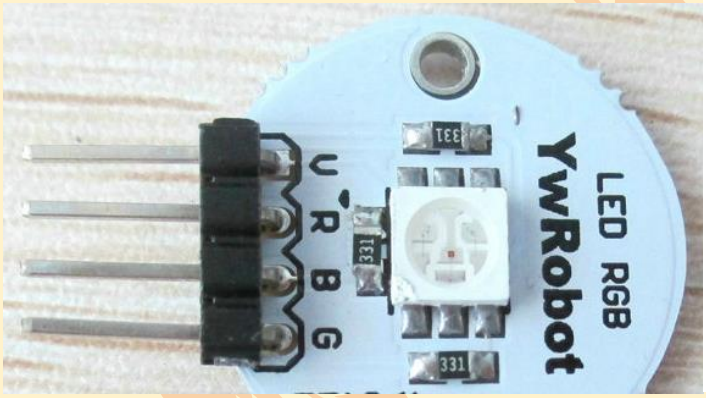
本章主要学习如何用 Arduino 控制一个“被控对象”的开关，即输出数字量。

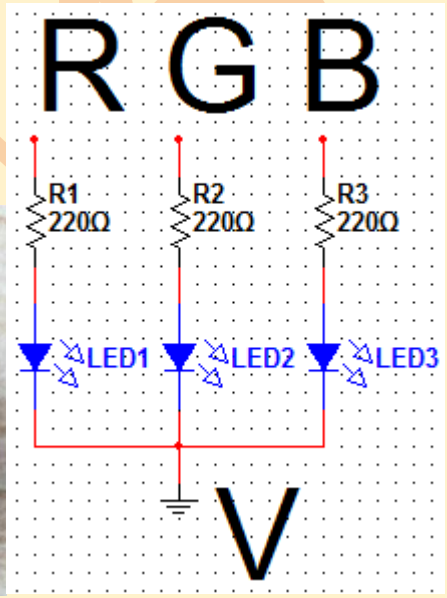
2.1 多彩 led 灯实验

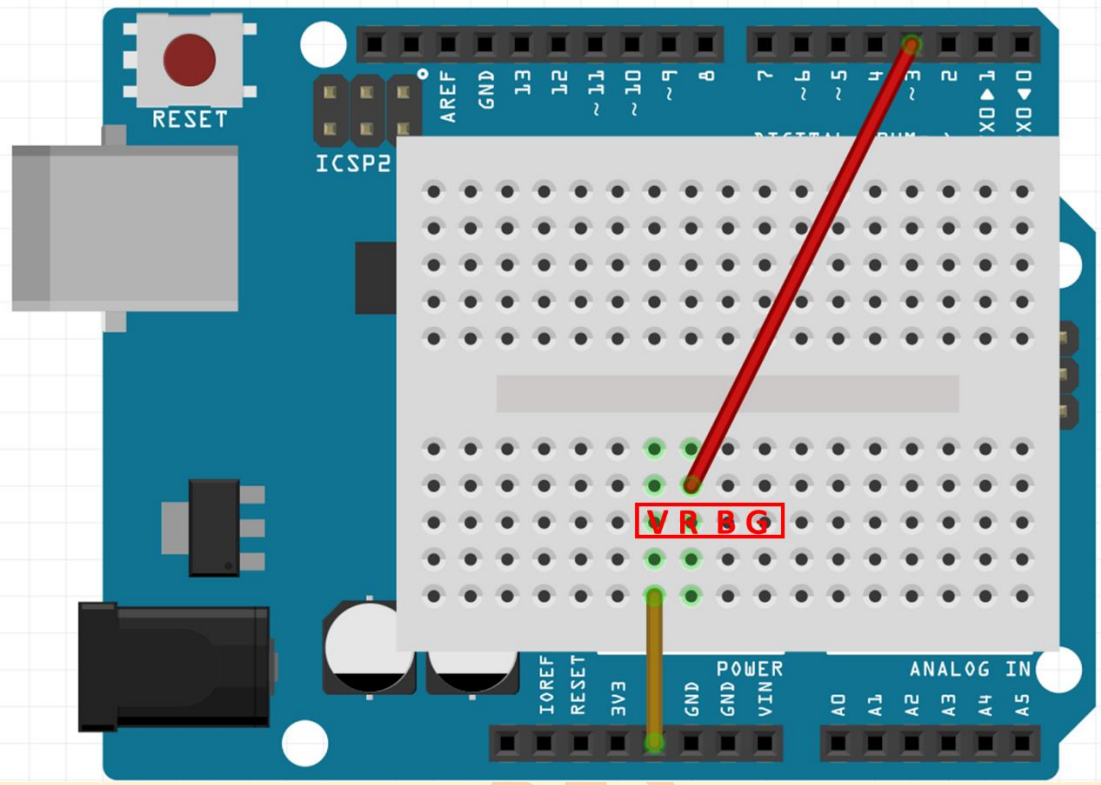
1 第一个程序：闪烁 led 灯

元件清单及电路连接

全彩 LED 模块 1 个，导线若干。





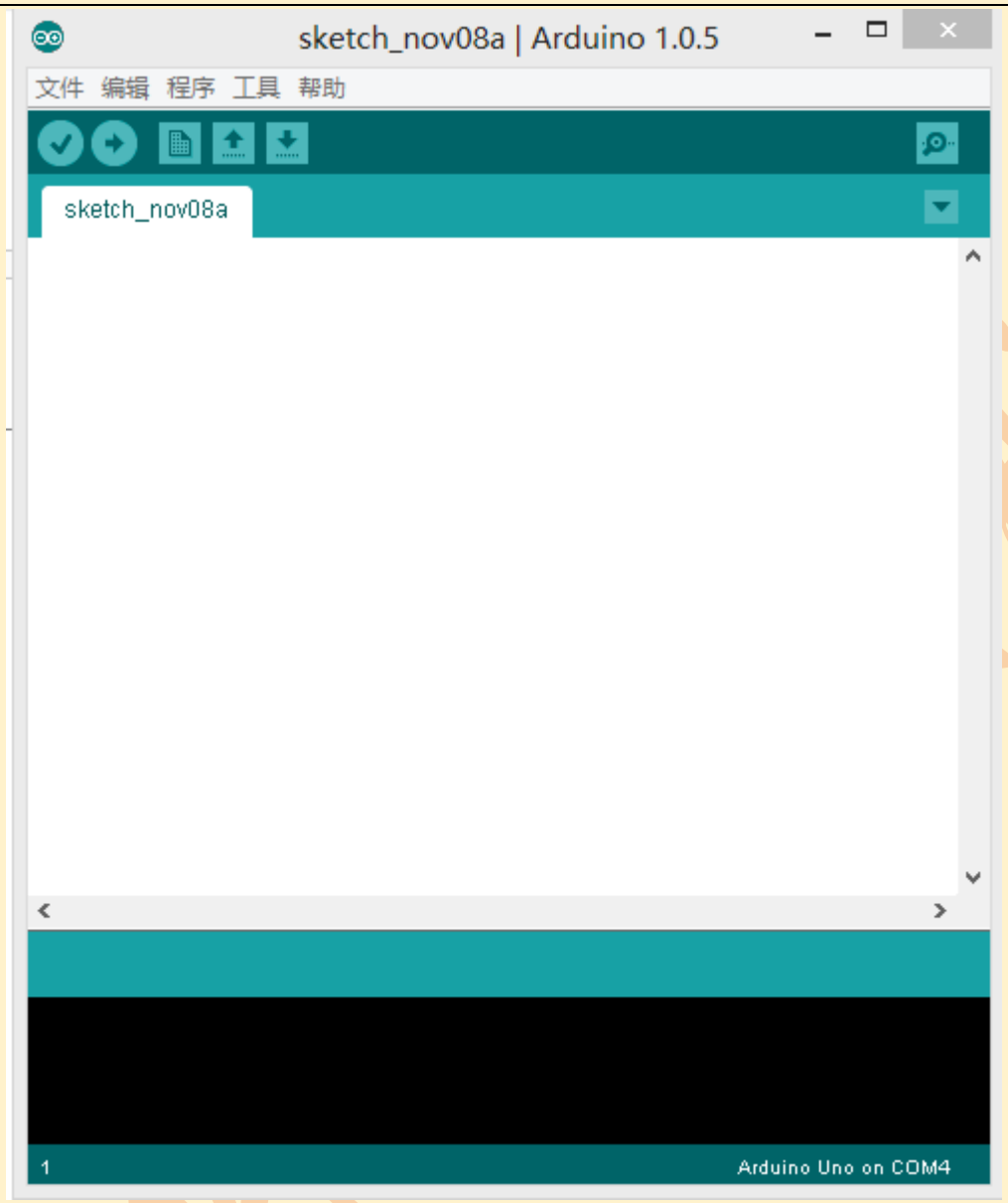


该元件可以接在面包板上。注意不要将它们接在同一列，因为同一列的 5 个孔是连通的。

从原理图可知，三个 LED 阴极是接在一起的（V 脚），该点接电源电源负（0V，GND）。R，G，B 脚的电压决定了对应 LED 是否点亮。本项目中只用 1 个灯，我们可以任选一个管脚进行操作。例如用红色 LED 就选择 R，将该管脚连接到 Digital I/O 第 3 口。

打开 arduino 开发环境

打开 arduino 文件夹，双击打开 arduino.exe，会出现如下界面：



Arduino 开发编译环境很简洁，从左到右功能键功能描述如下：

校验/Verify：验证程序是否正确

下载/Upload：把程序下载到 Arduino

新建/New,打开/Open,保存/Save 三个按钮：新建，打开，保存

最右边串口监视器/Serial Monitor：查看串口数据

打开软件后，我们就可以在窗口的空白处编写程序了。

编程方式 1：程序代码（将以下文字复制到开发软件中）

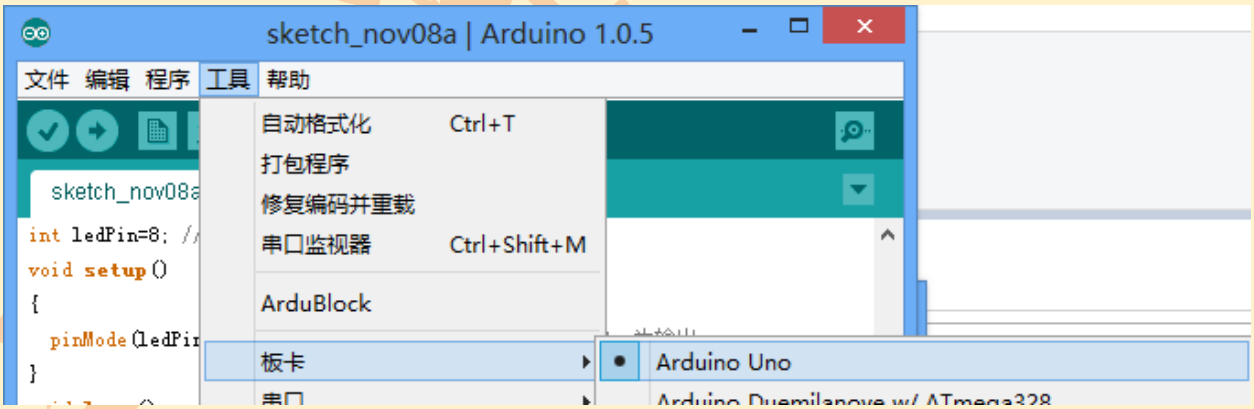
```
int ledPin=3; //设定控制 LED 的数字 IO 脚
void setup()
```

```
{
   pinMode(ledPin,OUTPUT);//设定数字 IO 口的模式，OUTPUT　为输出
}
void loop()
{
   digitalWrite(ledPin,HIGH); //设定 PIN3 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
}
```

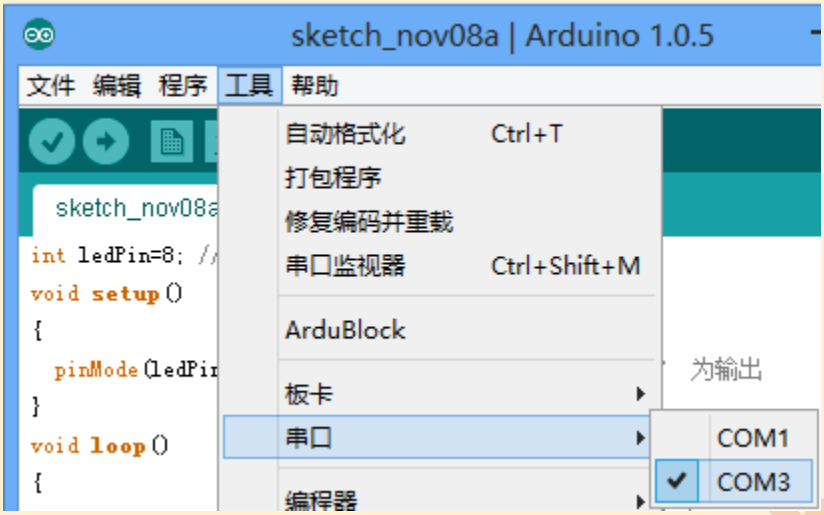
提示：工具菜单中的“自动格式化”可以帮助你整理好程序的外观。

程序下载

下载程序前需要先将板子型号和 com 口选好。点击工具/Tools->板卡/Board 选择开发板型号

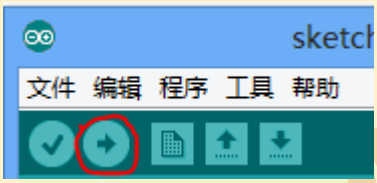


这里我们使用的是 Arduino UNO 控制板，所以点击第一个即可。接下来选择串口，如图：



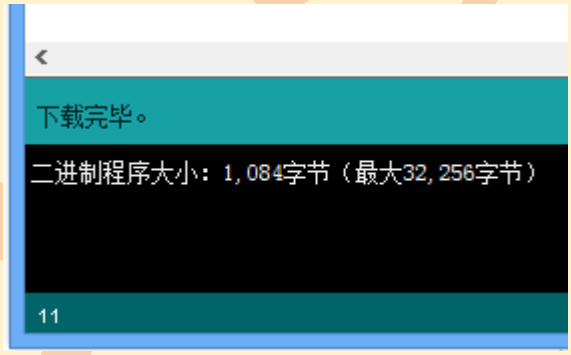
这里还有个技巧：如果只有 2 个 COM 口，直接选择数字比较大的那个。这样板子型号和 COM 口就选好了。

点击 arduino 软件上的下载按钮，如图：



点击之后下载按钮发成橙色，软件下方出现 Uploading to I/O Board，同时板子上标有 TX 和 RX 的灯会亮， 这是绝对禁止拔下 Arduino！

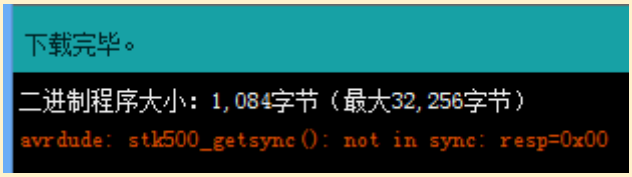
程序下载完毕后，下载按钮恢复原来的颜色，下面出现 Done Uploading，如图：



如果没有显示 Done Uploading，而是出现了红色的字，表示下载失败，可以检查一下 USB 线是否连接好、电源开关是否打开、COM 口是否选对等等。如果出现上图，表示程序下载成功了，如果你看到 13 脚下方的 L 灯亮 1s、灭 1s 的在闪烁，恭喜你，你的 Arduino 板开始工作啦！

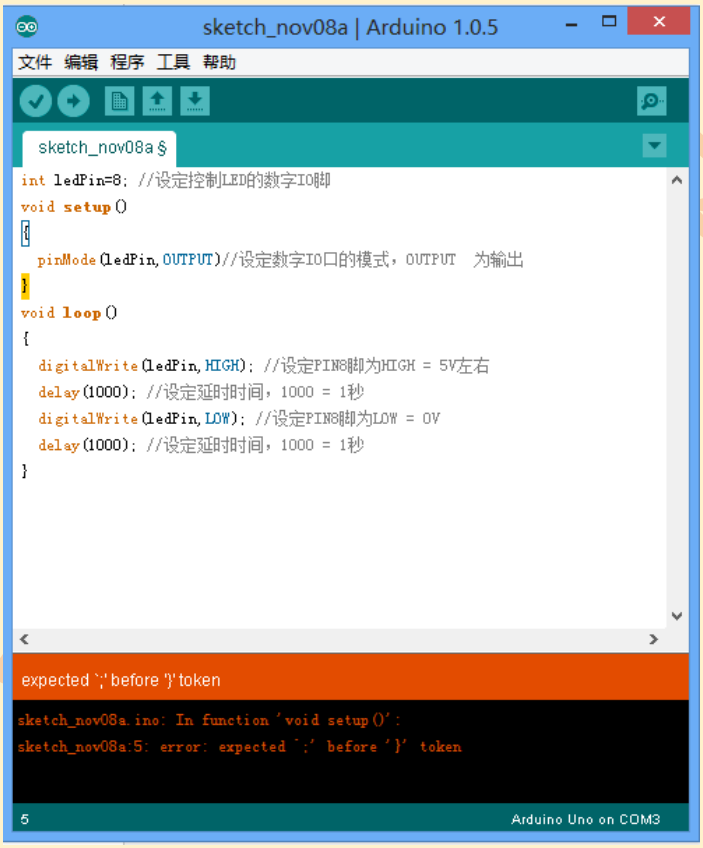
常见错误排查

（1）下载程序失败。这样的错误通常提示：



这样的错误一般是由于 COM 口或板子型号出了问题。重新检查设置。如果还不行，拔掉 Arduino 并关闭编程软件并重新开始。

（2）语法错误。例如，将程序中 pinMode（ledPin,OUTPUT）后面的分号去掉，点击编译按钮，编译完成后会出现如下图所示状态：



1 处告诉我们是因为在“}”附近缺少分号而出现的错误。2 处用文字告诉我们错误是出现在 void setup（）的一个“}”附近。3 处用黄颜色将“}”覆盖，表示错误就在这附近。从程序中看到错误确实在大括号附近，将分号添上后就会编译成功。以后编写程序出现错误时，就可以通过看下面信息栏里的提示调试程序。

实验结果

LED 灯亮 1 秒，灭 1 秒不断闪烁。

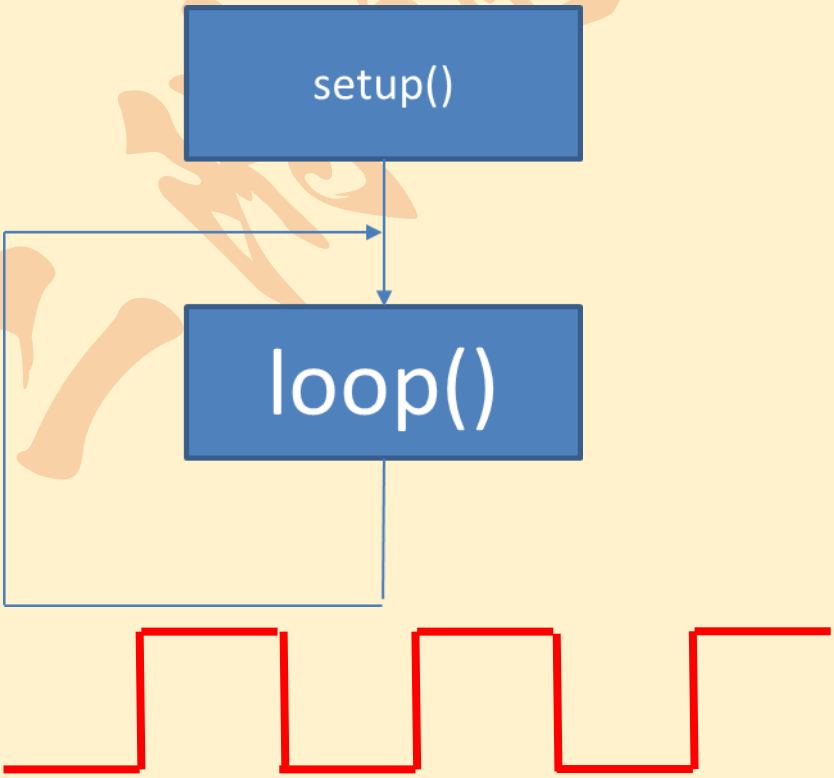
工作原理分析

（1）程序部分

Arduino 上的 C++程序包含 setup()初始化和 loop()循环执行 2 段程序。先执行 setup()的内容，完毕后反复执行 loop()。

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 3 引脚设置为输出模式。

loop()包含反复执行的内容，编写一个周期即可循环执行。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 3 口定义为高电平 5V；接着调用延时函数 delay(ms)（单位 ms）延时 1000ms； 再用数字 IO 口输出电平定义函数 digitalWrite(pin, value)，将数字 3 口定义为低电平；接着再调用延时函数 delay(ms) 单位 ms）延时 1000ms 因为 loop()函数是一个循环函数，所以这个过程会不断的循环，使 3 口电压形成一个 5V 方波。



常用语句示例：（注意大小写）

delay（2000）；　　延时 2000 毫秒

pinMode(3,OUTPUT);　　数字 3 口设为输出，常写在 setup()里

digitalWrite（3,HIGH);　　3 口输出 5V 电压

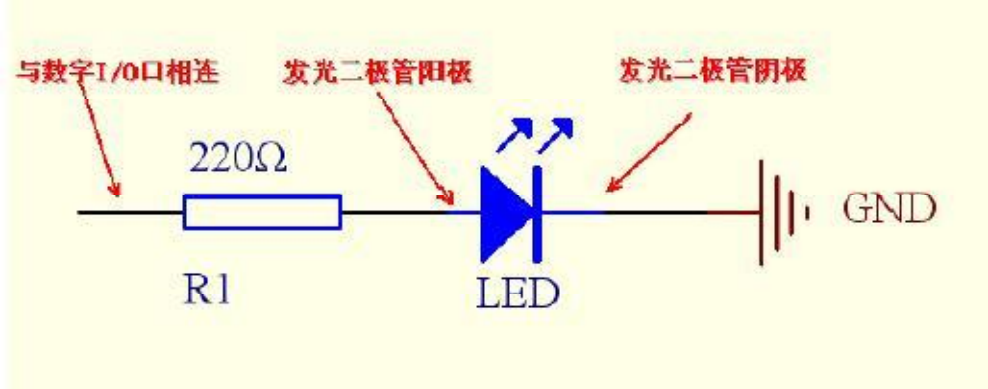
digitalWrite（7,LOW);　　7 口输出 0V 电压

注意：用 digitalWrite 需在初始化中用 pinMode 设为输出

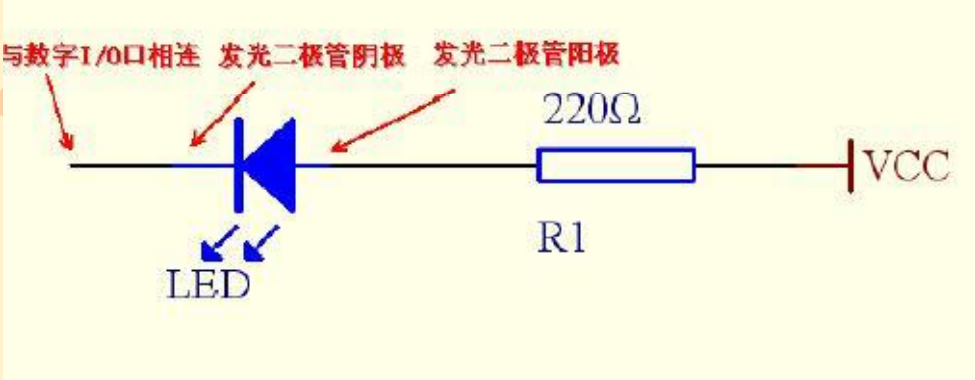
（2）电路部分

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。如下图：

VCC=5V（电源正极），GND=0V（电源负极）



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。



在这个电路中，数字 3 引脚为高电平 5V 时 led 灯熄灭，然后延时 1s，数字 3 引脚为低电平 0V led 灯点亮，再延时 1s。这样使 led 灯亮 1s、灭 1s，在视觉上就形成闪烁状态。如果想让 led 快速闪烁，可以将延时时间设置的小一些，如果想让 led 慢一点闪烁，可以将

延时时间设置的大一些。

注意：我们一般不使用 Digital 0，1 口，否则会无法写入程序。

2 自主设计：交通灯控制系统

循环亮灯：绿灯亮持续 8 秒，然后绿灯灭，黄灯亮持续 2 秒，然后黄灯灭红灯亮持续 8 秒，再回到开始（绿灯亮）。

提示：

程序注意大小写。R,G,B 分别接 3 个数字口（如 3，4，5）。

setup()里用 3 个 pinMode（X,OUTPUT）初始化。

loop()编写一个周期，用 digitalWrite 控制亮灭，delay 控制时间。

红绿同时亮可以合成黄色。

思考:

（1） 如何编写: 变灯前闪烁 2 次。

（2） 参照下一节“蜂鸣器”中的内容，在变灯时发声提示，发声时间 1 秒。

2.2 其他可用“数字输出”控制的元件

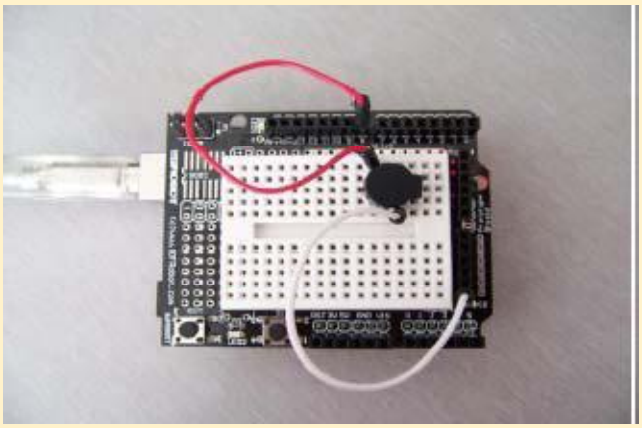
2.2.1 蜂鸣器

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。

连线

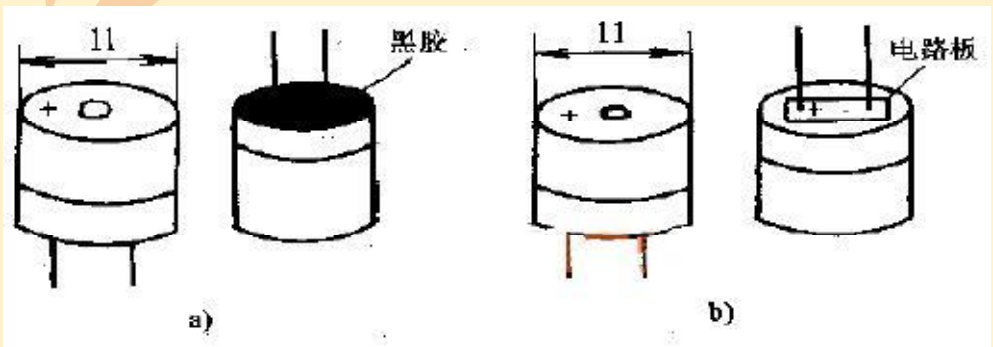
蜂鸣器发声原理是电流通过电磁线圈，使电磁线圈产生磁场来驱动振动膜发声的，因此需要一定的电流才能驱动它，本实验用的蜂鸣器内部带有驱动电路，所以可以直接使用。当与蜂鸣器连接的引脚为高电平时，内部驱动电路导通，蜂鸣器发出声音；当与蜂鸣器连接的引脚为低电平，内部驱动电路截止，蜂鸣器不发出声音。

将蜂鸣器的正极（长腿）连接到数字口，蜂鸣器的负极（短腿）连接到 GND 插口中。如下图，当数字口设为高电平时，蜂鸣器应该发声。



小知识

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郡朝上放置时，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 Rxl 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，如果触发出咔、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。

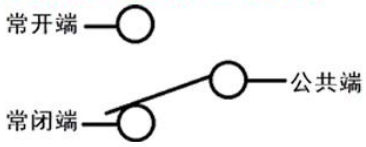
多谐振荡器由晶体管或集成电路极成。当接通电源后（1.5~15V 直流工作电压）,多谐振荡器起振,输出 1.5~2.5kHZ 的音频信号，阻抗匹配器推动压电蜂鸣片发声压电蜂鸣片由锆钛酸铅或铌镁酸铅压电陶瓷材料制成。在陶瓷片的两面镀上银电极，经极化和老化处理后，再与黄铜片或不锈钢片粘在一起。

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。

2.2.2 继电器

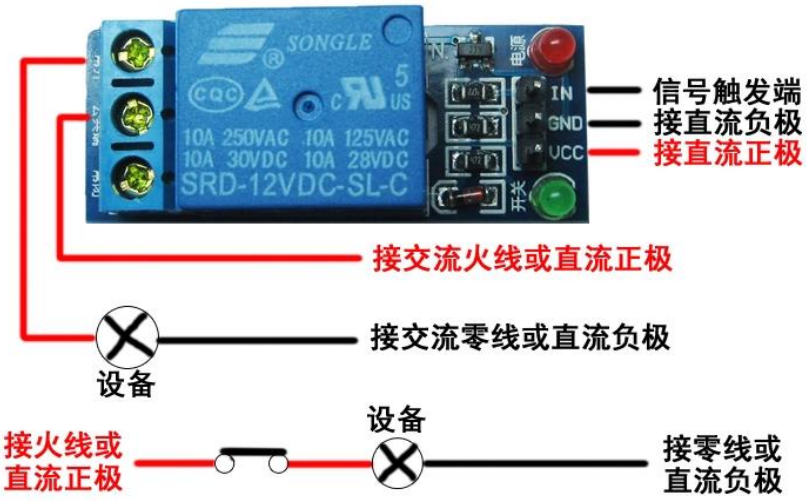
继电器可用来间接控制大功率的电器,我们可将它看作是一个由 arduino 控制的开关。连线时，将 VCC，GND 分别和 arduino 上的 5V，GND 相连；IN 和 arduino 的 Digital 输出端口相连，就可以参照上一节的办法控制继电器的开关了。将大功率负载接在接线柱上，就实现了 arduino 对此负载的控制。

继电器开关输出说明：



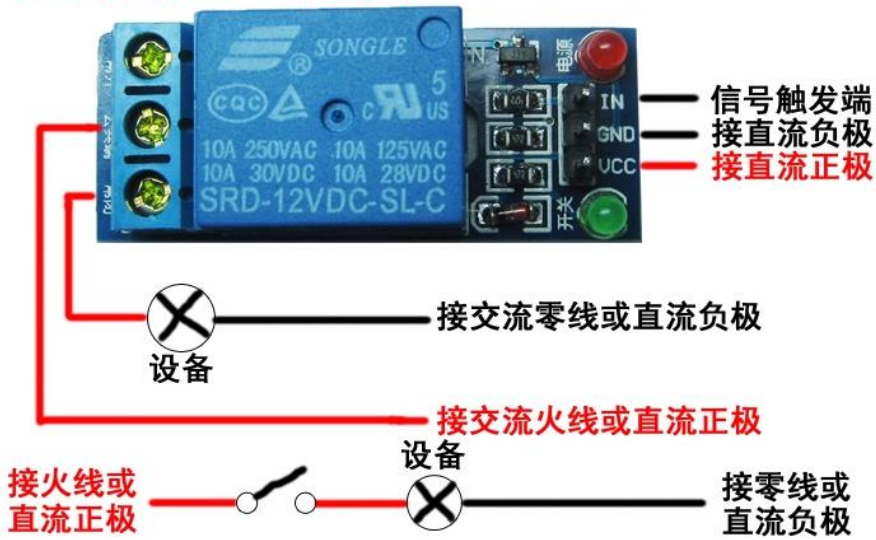
继电器线圈没有电压时，继电器没有吸合，公共端与常闭端接通，当有电压时，继电器吸合，公共端与常开端接通

接常开端的电路接法：



当信号触发端有高电平触发时，继电器吸合，相当于开闭合，此时电路接通，设备将有电将正常工作。

接常闭端的电路接法：



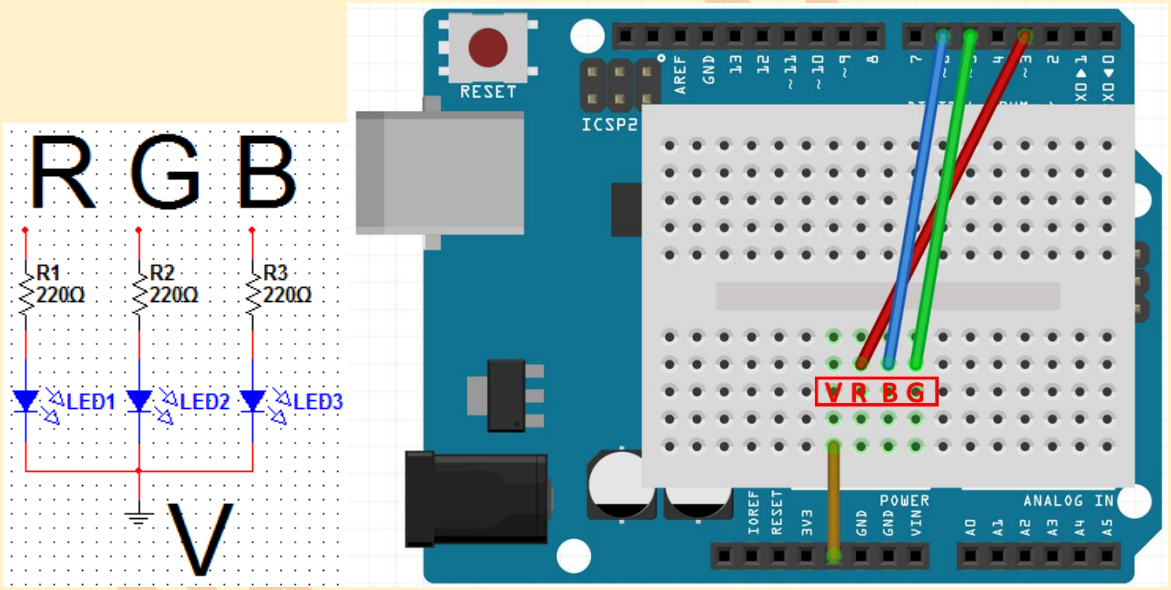
当信号触发端有高电平触发时，继电器吸合，相当于开由原来接通变为断开，此时设备将断电停止工作。

第三章 Arduino 模拟输出

3.1 呼吸灯

元件清单及电路连接

全彩 LED 包含红，绿，蓝三个 LED，用共阴联接法，内部已经串联电阻，因此无需额外的电阻。V 为共阴，接 GND。R，G，B 分别为红，绿，蓝控制端，电压有效值越高越亮。呼吸灯中， R 接在第 3 脚。



程序代码

```
int ledPin=3; //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()
{
  for(int i=0;i<=255;i++)//循环，i 初始为 0，每次+1，一直增加到 255 为止
```

```
    {
      analogWrite(ledPin,255-i); //写入亮度值（0 增加到 255）
      delay(5);
    }
    for(int i=255;i>=0;i--)//循环，初始为 255，每次-1，一直减小到 0 为止
    {
      analogWrite(ledPin,255-i); //写入亮度值（255 减小到 0）
      delay(5);
    }
    delay(100);
  }
```

实验原理

PWM（Pulse-width modulation）脉宽调制

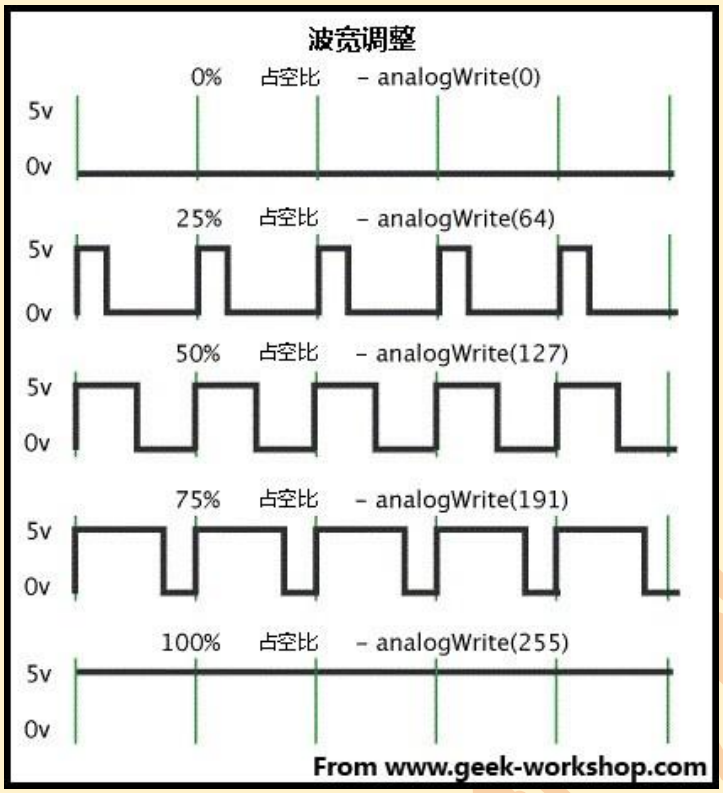
PWM 是使用数字手段来控制模拟输出的一种手段。使用数字控制产生占空比不同的方波（一个不停在开与关之间切换的信号)来控制模拟输出。

以本次实验来看，端口的输入电压只有两个 0V 与 5V。如想要 3V 的输出电压怎么办？有同学说串联电阻，对滴，这个方法是正确滴。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。

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，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，输出端口就感觉是 2.5V 的电压输出了。这个和咱们放映电影是一个道理，咱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 的输出比例控制就 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。

在下图中，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）。



analogWrite 其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.在我们使用的 UNO 板上，其工作在 3,5,6,9,10,11 端口；Arduino Mega 控制板，可以工作于 2-13 号端口。

语法：analogWrite(pin, value)

参数：pin:写入的端口，value:占空比:在 0-255 之间。（如果不在该范围，会自动除以 255 取余数）

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。

思考：如何实现：红黄绿青蓝紫循环变色。

全彩 LED 由 RGB（红绿蓝）三种 LED 组成。用红绿蓝三种光可合成不同颜色的光，如：R=255,G=0,B=0 为红光，R=255,G=255,B=0 为黄光，依次类推。

算法：

	R	G	B
红	255	0	0
黄	255	255	0
绿	0	255	0
青	0	255	255
蓝	0	0	255
紫	255	0	255
下一循环:红	255	0	0

示例 C 程序：（R 接 3，G 接 5，B 接 6）

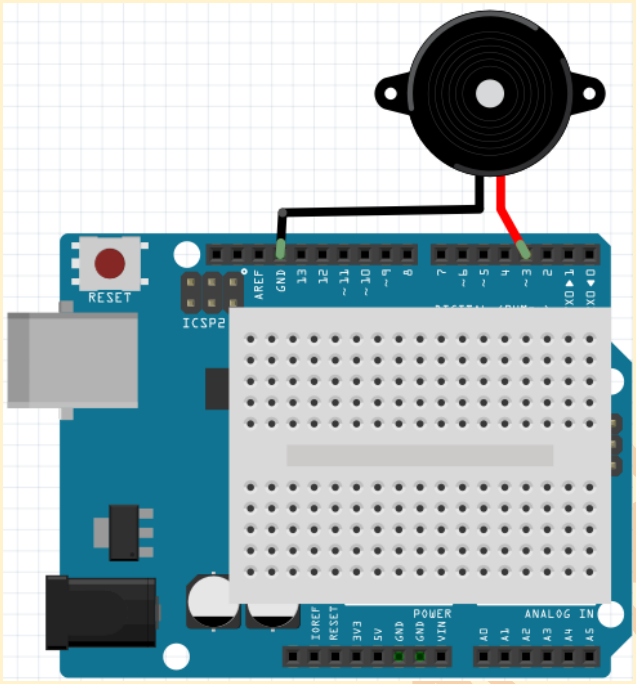
```
int r=3;
int g=5;
int b=6;
int a=10;

void setup(){
  pinMode(3,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  analogWrite(r,0);//共阳：初始值为红
  analogWrite(g,255);
  analogWrite(b,255);
}
```

```
void loop(){

  for(int i=0;i<=255;i++){ //红变黄
    analogWrite(g,255-i);
    delay(a);
  }
  for(int i=255;i>=0;i--){ //黄变绿
    analogWrite(r,255-i);
    delay(a);
  }
  for(int i=0;i<=255;i++){ //绿变青
    analogWrite(b,255-i);
    delay(a);
  }
  for(int i=255;i>=0;i--){ //青变蓝
    analogWrite(g,255-i);
    delay(a);
  }
  for(int i=0;i<=255;i++){ //蓝变紫
    analogWrite(r, 255-i);
    delay(a);
  }
  for(int i=255;i>=0;i--){ //紫变红
    analogWrite(b,255-i);
    delay(a);
  }
}
```

3.2 简单音乐制作

元件清单及电路连接

adruino uno 一块，面包线若干条，小喇叭一个。音量太大可串联 220 欧电阻。



工作原理

首先讲下简单的乐理知识，知道音乐是怎么演奏出来的自然就可以通过代码来进行编排了。

1.演奏单音符 一首乐曲有若干音符组成，一个音符对应一个频率。我们知道到相对应的频率 让 arduino 输出到蜂鸣器 蜂鸣器就会放出相应的声音，这里有个表供大家参考：

音调 音符	1 .	2 .	3 .	4 .	5 .	6 .	7 .
A	221	248	278	294	330	371	416
B	248	278	294	330	371	416	467
C	131	147	165	175	196	221	248
D	**147**	**165**	**175**	**196**	**221**	**248**	**278**
E	165	175	196	221	248	278	312
F	175	196	221	234	262	294	330
G	196	221	234	262	294	330	371

音调 音符	1	2	3	4	5	6	7
A	441	495	556	589	661	742	833
B	495	556	624	661	742	833	935
C	262	294	330	350	393	441	495
D	**294**	**330**	**350**	**393**	**441**	**495**	**556**
E	330	350	393	441	495	556	624
F	350	393	441	495	556	624	661
G	393	441	495	556	624	661	742

音调 音符	. 1	. 2	. 3	. 4	. 5	. 6	. 7
A	882	990	1112	1178	1322	1484	1665
B	990	1112	1178	1322	1484	1665	1869
C	525	589	661	700	786	882	990
D	**589**	**661**	**700**	**786**	**882**	**990**	**1112**
E	661	700	786	882	990	1112	1248
F	700	786	882	935	1049	1178	1322
G	786	882	990	1049	1178	1322	1484

2.音符的演奏时间 我们知道了音符是如何演奏出来的，下一步就是控制音符的演奏时间。每个音符都会播放一定的时间，这样才能构成一首优美的曲子，而不是生硬的一个调的把所有的音符一股脑的都播放出来。如何确定每个音符演奏的单位时间呢？我们知道，音符节奏分为一拍、半拍、1/4 拍、1/8 拍，我们规定一拍音符的时间为 1；半拍为 0.5；1/4 拍为 0.25；1/8 拍为 0.125……，所以我们可以为每个音符赋予这样的拍子播放出来，音乐就成了。

我们看看如何将简谱翻译成对应频率和拍子。以葫芦娃为例：

先看下左上角 1=D 这里，用的是 D 调，好那我们就看《音符频率表》中的 D 行（红色部分），

第一个音符是 1，时间是一拍=1，第二个音符 6（没有点），时间也是一拍=1，第三个音符 5，因为有下划线所以是半拍=0.5，……以此类推。

第四 0 这里要注意下，这里是没有声音，拍子=1 拍。

第五，再接着看到第一句歌词葫芦娃 这个娃的音是 3—，这表示是两拍，后面每加一个“-”，表示拍子+1（1+1），本例中最多是加到 4.

第六，第二句歌词葫芦娃这个娃，3•带个点，点的意思是去 3 的拍子的一半，即 3•的拍子是 1+0.5

第七，大家可能会问那弧线怎么表示，这在音乐中属于连音，操作上更复杂了，本例没有做连音的处理。

所以规律就是时间上单个音符没有下划线，就是一拍（1），有下划线是半拍（0.5），两个下划线是四分之一拍（0.25），有“—”=前面音符的拍子+1；频率上就是按照音符是否带点，

点在上还是在下到表中查找就可以了。

至此原理清楚，随便拿个简谱来我们都可以翻译成代码了。

程序代码

```
//每种声音对应的频率
#define D0 -1
#define D1 294
#define D2 330
#define D3 370
#define D4 393
#define D5 441
#define D6 495
#define D7 556

#define DL1 147
#define DL2 165
#define DL3 190
#define DL4 196
#define DL5 221
#define DL6 248
#define DL7 278

#define DH1 589
#define DH2 661
#define DH3 700
#define DH4 786
#define DH5 882
#define DH6 990
#define DH7 1120
```

```
//乐谱：音阶
int tune[] =
{
   DH1,D6,D5,D6,D0,
   DH1,D6,D5,DH1,D6,D0,D6,
   D6,D6,D5,D6,D0,D6,
   DH1,D6,D5,DH1,D6,D0,

   D1,D1,D3,
   D1,D1,D3,D0,
   D6,D6,D6,D5,D6,
   D5,D1,D3,D0,
   DH1,D6,D6,D5,D6,
   D5,D1,D2,D0,
   D7,D7,D5,D3,
   D5,
   DH1,D0,D6,D6,D5,D5,D6,D6,
   D0,D5,D1,D3,D0,
   DH1,D0,D6,D6,D5,D5,D6,D6,
   D0,D5,D1,D2,D0,
   D3,D3,D1,DL6,
   D1,
   D3,D5,D6,D6,
   D3,D5,D6,D6,
   DH1,D0,D7,D5,
   D6,
};
//乐谱：拍数。例如：四分音符为 1 拍，二分为 2，八分为 0.5.
```

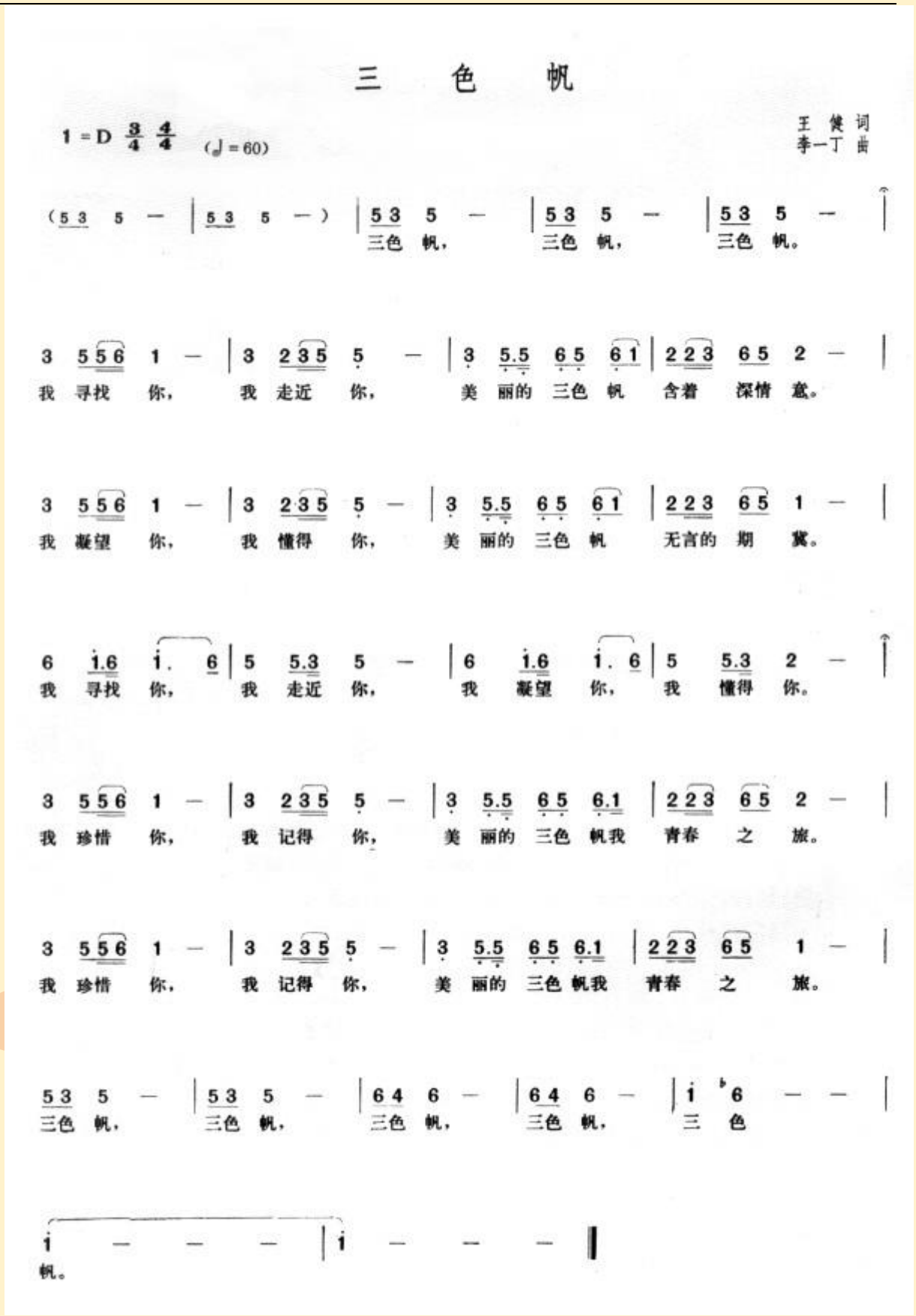
```
float duration[]=
{
   1,1,0.5,0.5,1,
   0.5,0.5,0.5,0.5,1,0.5,0.5,
   0.5,1,0.5,1,0.5,0.5,
   0.5,0.5,0.5,0.5,1,1,

   1,1,1+1,
   0.5,1,1+0.5,1,
   1,1,0.5,0.5,1,
   0.5,1,1+0.5,1,
   0.5,0.5,0.5,0.5,1+1,
   0.5,1,1+0.5,1,
   1+1,0.5,0.5,1,
   1+1+1+1,
   0.5,0.5,0.5+0.25,0.25,0.5+0.25,0.25,0.5+0.25,0.25,
   0.5,1,0.5,1,1,
   0.5,0.5,0.5+0.25,0.25,0.5+0.25,0.25,0.5+0.25,0.25,
   0.5,1,0.5,1,1,
   1+1,0.5,0.5,1,
   1+1+1+1,
   0.5,1,0.5,1+1,
   0.5,1,0.5,1+1,
   1+1,0.5,0.5,1,
   1+1+1+1
};
int tonePin=3;//蜂鸣器的端口
void setup()
{
```

```
    pinMode(tonePin,OUTPUT);//设置蜂鸣器的 pin 为输出模式
}

void loop()
{
    int length = sizeof(tune)/sizeof(tune[0]);//固定格式，查出 tone 序列里有多少个音符
    for(int x=0;x<length;x++)//循环音符的次数
    {
        tone(tonePin,tune[x]);//此函数依次播放 tune 序列里的数组，即每个音符
        delay(400*duration[x]);//每个音符持续的时间，即节拍 duration，400 是调整时间的越大，曲子速度越慢，越小曲子速度越快，自己掌握吧
        noTone(tonePin);//停止当前音符，进入下一音符
    }
    delay(2000);//等待 2 秒后重新播放
}
```

思考

1.前面示例程序中有个音不准，请改正。

2.编写三色帆校歌乐曲，简谱如下：


三 色 帆
1=D 3/4 4/4 (♩=60)
王 健 词
李一丁 曲
(5 3 5 — | 5 3 5 —) | 5 3 5 — | 5 3 5 — | 5 3 5 — |
三色 帆, 三色 帆, 三色 帆。
3 5 5 6 1 — | 3 2 3 5 5 — | 3 5.5 6 5 6 1 | 2 2 3 6 5 2 — |
我 寻找 你, 我 走近 你, 美 丽的 三色 帆 含着 深情 意。
3 5 5 6 1 — | 3 2 3 5 5 — | 3 5.5 6 5 6 1 | 2 2 3 6 5 1 — |
我 凝望 你, 我 懂得 你, 美 丽的 三色 帆 无言的 期 冀。
6 1.6 i. 6 | 5 5.3 5 — | 6 1.6 i. 6 | 5 5.3 2 — |
我 寻找 你, 我 走近 你, 我 凝望 你, 我 懂得 你。
3 5 5 6 1 — | 3 2 3 5 5 — | 3 5.5 6 5 6.1 | 2 2 3 6 5 2 — |
我 珍惜 你, 我 记得 你, 美 丽的 三色 帆我 青春 之 旅。
3 5 5 6 1 — | 3 2 3 5 5 — | 3 5.5 6 5 6.1 | 2 2 3 6 5 1 — |
我 珍惜 你, 我 记得 你, 美 丽的 三色 帆我 青春 之 旅。
5 3 5 — | 5 3 5 — | 6 4 6 — | 6 4 6 — | i ♭6 — — |
三色 帆, 三色 帆, 三色 帆, 三色 帆, 三 色
i — — — | i — — — ‖
帆。

参考乐谱：

```
int tune[] =
{
  D5,D3,D5,D5,D3,D5,D5,D3,D5,
  D3,D5,D5,D6,D1,D3,D2,D3,D5,DL5,DL3,DL5,DL5,DL6,DL5,DL6,D1,D2,D2,D3,D6,D5,D2,
  D3,D5,D5,D6,D1,D3,D2,D3,D5,DL5,DL3,DL5,DL5,DL6,DL5,DL6,D1,D2,D2,D3,D6,D5,D1,
  D6,DH1,D6,DH1,D6,D5,D5,D3,D5,D6,DH1,D6,DH1,D6,D5,D5,D3,D2,
  D3,D5,D5,D6,D1,D3,D2,D3,D5,DL5,DL3,DL5,DL5,DL6,DL5,DL6,D1,D2,D2,D3,D6,D5,D2,
  D3,D5,D5,D6,D1,D3,D2,D3,D5,DL5,DL3,DL5,DL5,DL6,DL5,DL6,D1,D2,D2,D3,D6,D5,D1,
  D5,D3,D5,D5,D3,D5,D6,D4,D6,D6,D4,D6,DH1,D5,DH1
};
float duration[]=
{
  0.5,0.5,2,0.5,0.5,2,0.5,0.5,2,
  1,0.5,0.25,0.25,2,1,0.5,0.25,0.25,2,1,0.5,0.25,0.5,0.5,0.5,0.5,0.5,0.25,0.25,0.5,0.5,2,
  1,0.5,0.25,0.25,2,1,0.5,0.25,0.25,2,1,0.5,0.25,0.5,0.5,0.5,0.5,0.5,0.25,0.25,0.5,0.5,2,
  1,0.75,0.25,1,0.5,1,0.75,0.25,2,1,0.75,0.25,1,0.5,1,0.75,0.25,2,
  1,0.5,0.25,0.25,2,1,0.5,0.25,0.25,2,1,0.75,0.25,0.5,0.5,0.75,0.25,0.5,0.25,0.25,0.5,0.5,2,
  1,0.5,0.25,0.25,2,1,0.5,0.25,0.25,2,1,0.75,0.25,0.5,0.5,0.75,0.25,0.5,0.25,0.25,0.5,0.5,2,
  0.5,0.5,2,0.5,0.5,2,0.5,0.5,2,0.5,0.5,2,1,3,8,
};
```

同时必须修改播放速度。

2.在上述基础上，每播放一个音符闪一次灯。

参考答案（LED 接 digital 6 口）

```
void loop()
{
  int length = sizeof(tune)/sizeof(tune[0]);//固定格式，查出 tone 序列里有多少个音符
  for(int x=0;x<length;x++)//循环音符的次数
  {
    tone(tonePin,tune[x]);//此函数依次播放 tune 序列里的数组，即每个音符
```

```
        digitalWrite(6,LOW);
        delay(400*duration[x]);//每个音符持续的时间，即节拍 duration，400 是调整时间
的
         //越大，曲子速度越慢，越小曲子速度越快，自己掌握吧
         digitalWrite(6,HIGH);
        delay(400*duration[x]);
        noTone(tonePin);//停止当前音符，进入下一音符
      }
      delay(2000);//等待 2 秒后重新播放
    }
```

第四章 Arduino 模拟输入与传感器

4.1 简单科学仪器设计

1 模拟信号与传感技术

“天然的”物理量大多数是连续的，包括电阻，电压，光照度，加速度，声音等。我们如何测量这些信号，并用于自动控制呢？

前面我们学习的都是“输出”某个信号，或者说“控制“某个元件。测量模拟信号需要把它“输入”到 arduino 中，这里需要用到传感技术。

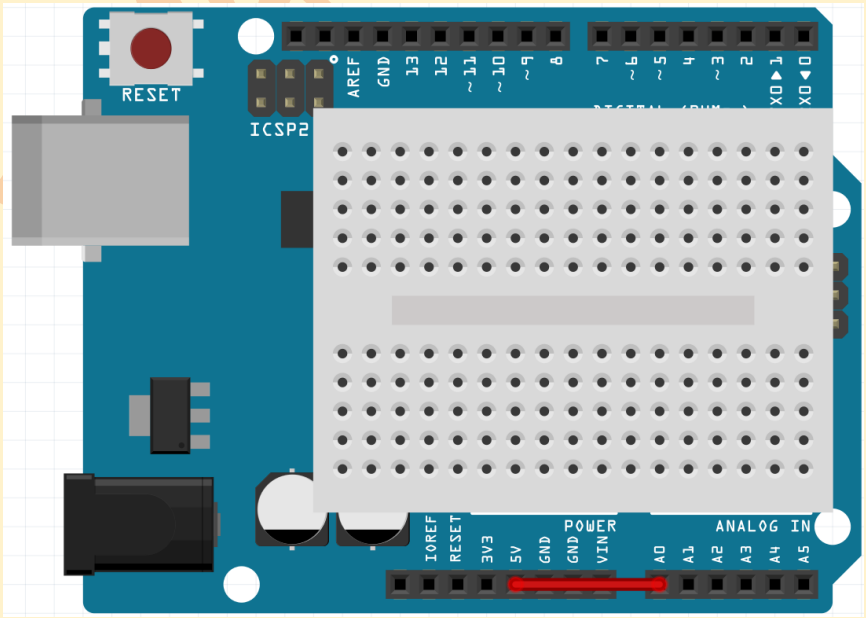
2 电压表

元件清单

Arduino，面包板，导线

电路连接

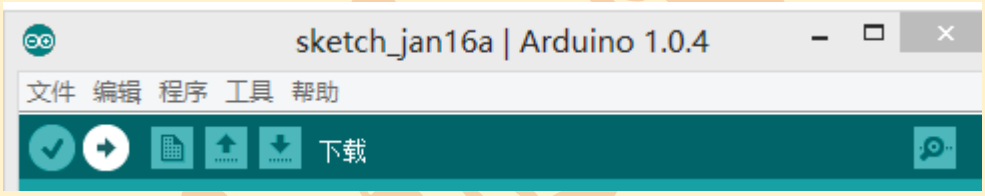
在 analog In 0 口上接一根线，把它连接到待测端口，如 5V、GND、3V.



程序代码

```
void setup()
{
  Serial.begin(9600); //打开串口通信，速率 9600bps
}
void loop()
{
   int a=analogRead(0); //读取 A0 电压，储存在整型变量 a 中，量程 0-5V
   Serial.println(a); //从串口将 a 发送给电脑显示
   delay(100);
}
```

程序功能



写入程序，打开最右侧的按钮“串口监视器”，可以看到电压读数。将 Analog 0 口分别和 5V,3.3V,GND（0V）连接，可以看到数值变化。

实验原理

思考：

系统首先运行哪个语句？（　　）哪个语句只运行一次？（　　）

A　　Serial.begin(9600);

B　　int a=analogRead(0);

C　　Serial.println(a);

D　　delay(100);

观察可得，电压值和输出值是成（ ）比的。比例系数为（ ）。

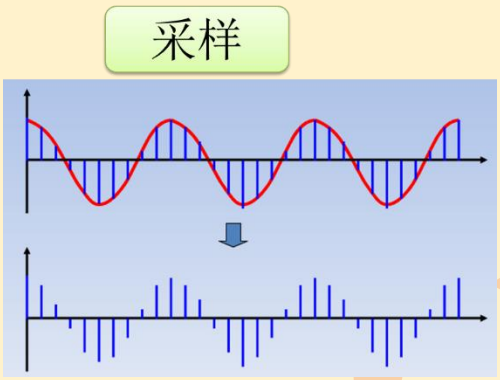
Arduino 将电压数字化的过程，分为三步：采样，量化，编码。

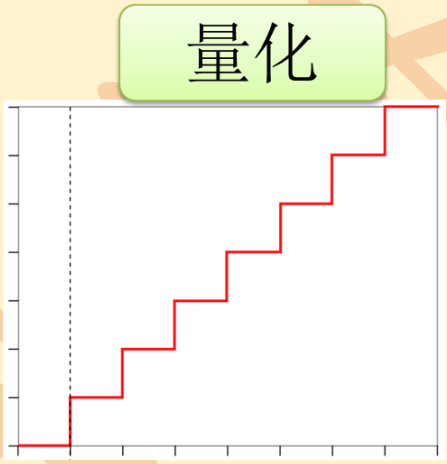
采样：信号的时间轴离散化

量化：信号的幅度轴离散化

编码：按一定格式记录采样和量化后的数据

Analog 端口可看作电压表，0-5V 量程，对应输出为 0-1023。





思考：

本系统的采样率（ ）Hz，（ ）bit 量化。

思考：

比例系数的意义是什么？

如何让系统直接显示电压。提示：考虑比例系数。

3 欧姆表

我们已经学会测量电压了。如何测量未知电阻？

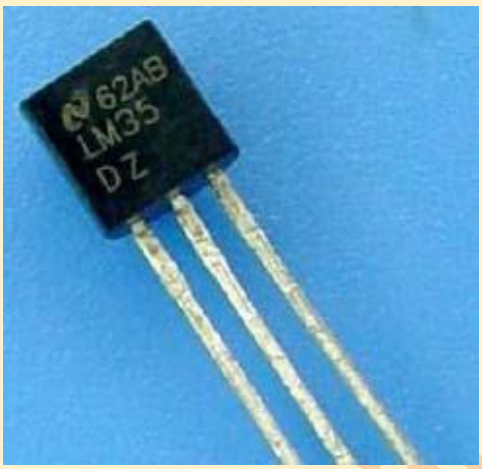
提示：单片机上资源有：已知电源（5V），Analog 端口（可以测量电压），大家用过的定值电阻。根据初中电学的知识可以设计出测量未知电阻的方法。

完成电路连接，推导公式，然后修改程序，完成未知电阻测量。

4.2 温度传感实验

1 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体。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：

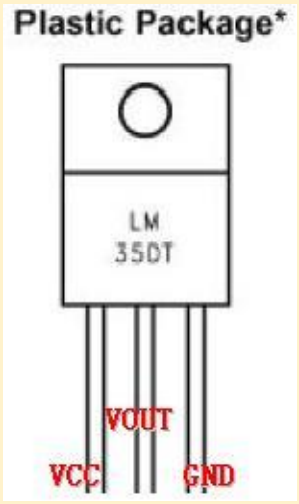


工作原理

LM35 温度传感器的输出电压与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：

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

连线

LM35 的引脚示意图如下： 

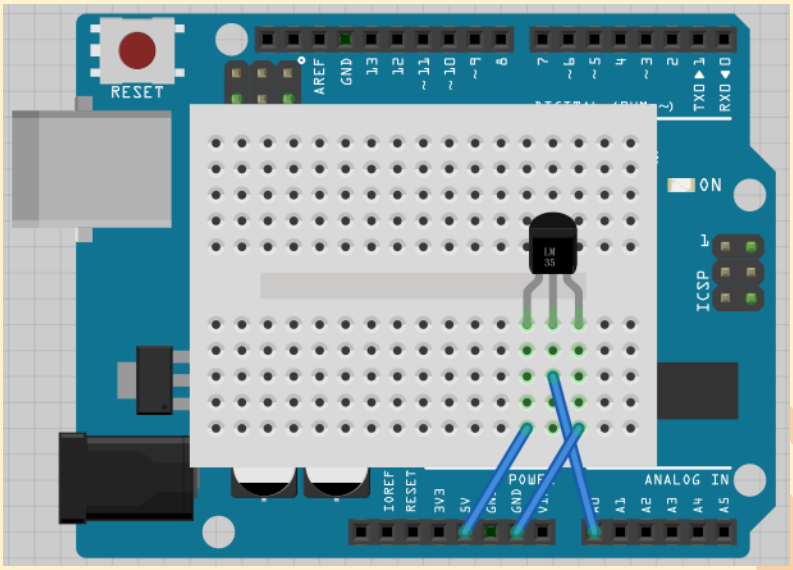
从实验盒中将温度传感器拿出来可以看到，温度传感器的一面是平的，另一面是半圆的。将平面对着自己，最左边的是 VCC 引脚（接+5v），中间的为 VOUT（电压值输出引脚，接板子上的模拟引脚），最右边的引脚为 GND 引脚（接板子上的 GND）。三个引脚分别接好就可以用了。

2 电子温度计

元件清单

LM35 温度传感器：1 个　多彩面包板实验跳线：若干

电路连接



程序代码

```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
   Serial.println(analogRead(0)/1024.0*5.0/0.01);
   delay(2000);
}
```

程序功能

将程序下载到实验板，打开右上角串口监视器/Serial Monitor 按钮，可以看到 LM35 输出数值。用手加热 LM35，可以看到数值变化。

3 温度报警实验

元件清单

LM35 温度传感器：1 个　　LED 灯盘 1 个

多彩面包板实验跳线：若干

电路连接

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； V 为共阳，接 5V，R 接 3，G 接 5，B 接 6。这样温度报警实验的电路就连接好了。

程序代码

程序代码：（建议更改温度阈值方便用手测试）

```
int b=6;
int g=5;
int r=3;
void setup()
{
  pinMode(g,OUTPUT);
  pinMode(b,OUTPUT);
  pinMode(r,OUTPUT);
  Serial.begin(9600);
}
void loop()
{
  double i=analogRead(0)/1024.0*5.0/0.01;//读取温度
  Serial.println(i);
```

```
    if(i<27) //温度低于 27
    {
      digitalWrite(b,HIGH);//共阳灯板，蓝灯亮
      digitalWrite(g,LOW);
      digitalWrite(r, LOW);
    }
    else if(i>=30) //温度高于 30
    {
      digitalWrite(r, HIGH);//红灯亮
      digitalWrite(g, LOW);
      digitalWrite(b, LOW);
    }
    else //温度在 27-30 之间
    {
      digitalWrite(g, HIGH);//绿灯亮
      digitalWrite(r, LOW);
      digitalWrite(b, LOW);
    }
    delay(500);
  }
```

程序功能

将程序下载到实验板，看看那个 LED 灯亮着，就知道你的环境温度是在那个温度段了。

4 彩色喷头

有一种喷头可以根据温度的变化改变颜色，防止用户被冷水冲感冒或被烫伤。最合适的温度可以设置。

要求：用渐变色显示温度。阈值在程序中定义，温度过低时显示蓝色。随着温度过渡到合适，颜色从蓝色过渡到青色直到绿色。随着温度从合适升到偏高，颜色从绿色过渡到黄色直到红色。

参考答案：

```
int RED=3; //设定控制 LED 的数字 IO 脚
int GREEN=5;
int BLUE=6;
double set=27;

void setup()
{
   Serial.begin(9600);
   pinMode(RED,OUTPUT);//设定数字 IO 口的模式，OUTPUT 为输出
   pinMode(BLUE,OUTPUT);
   pinMode(GREEN,OUTPUT);
}
void loop()
{

      double tmp=analogRead(0)*0.488;
      Serial.println(tmp);
      Serial.println(set);
      Serial.println("-----");

      if(tmp>set+3)
      {
         work(255,0,0);
      }
      else if(tmp<set+3 && tmp>set)
      {
         work((tmp-set)*255/3,(set+3-tmp)*255/3,0);
      }
      else if(tmp<set && tmp>set-3)
```

```
        {
            work(0,(tmp-set+3)*255/3,(set-tmp)*255/3);
        }
        else
        {
            work(0,0,255);
        }
    }

    void work(int r,int g,int b)
    {
        analogWrite(RED, r);
        analogWrite(GREEN, g);
        analogWrite(BLUE, b);
        delay(100);
    }
```

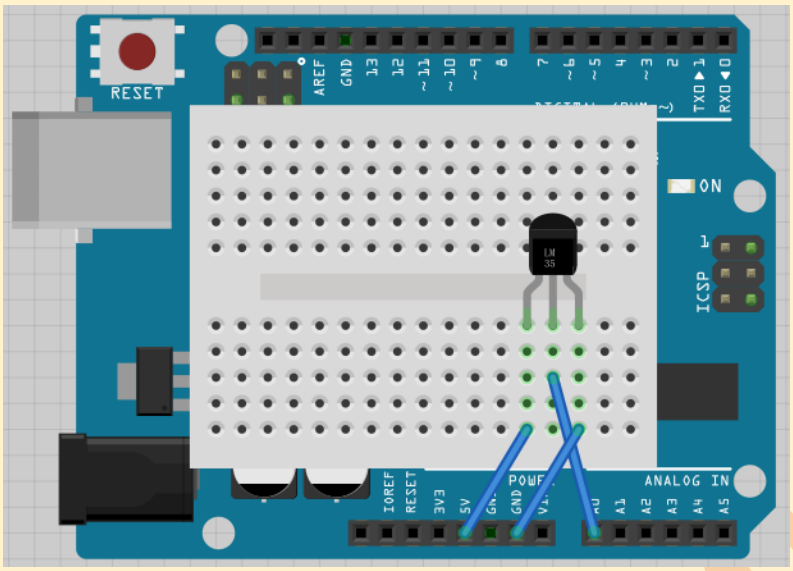
5 物联网检测系统

元件清单

LM35 温度传感器：1 个　多彩面包板实验跳线：若干 网络扩展板 W5100 一个面包板等

电路连接

W5100 插在 Arduino 底板上，面包板叠在 W5100 上。



程序代码

```
#include <SPI.h>
#include <Ethernet.h>
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
EthernetServer server(80);

void setup() {
    Serial.begin(9600);
  Ethernet.begin(mac);
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
}

void loop() {
  EthernetClient client = server.available();
  if (client) {
    Serial.println("new client");
    boolean currentLineIsBlank = true;
```

```
while (client.connected()) {
    if (client.available()) {
        char c = client.read();
        Serial.write(c);
        if (c == '\n' && currentLineIsBlank) {
            client.println("HTTP/1.1 200 OK");
            client.println("Content-Type: text/html");
            client.println("Connection: close");
            client.println();

            client.println("<!DOCTYPE HTML>");
            client.println("<html>");
            client.println("<meta http-equiv=\"refresh\" content=\"5\">");
            int sensorReading == analogRead(0);
            client.print("analog input is ");
            client.print(sensorReading);
            client.println("<br />");
            client.println("</html>");
            break;
        }
        if (c == '\n') {
            currentLineIsBlank = true;
        }
        else if (c != '\r') {
            currentLineIsBlank = false;
        }
    }
}
delay(1);
```

```
        client.stop();
        Serial.println("client disonnected");
    }
}
```

程序功能

将程序下载到实验板，打开右上角串口监视器/Serial Monitor 按钮，可以看到 server is at 192.168.XX.XX

在浏览器中输入后面的 IP 地址，即可看到传感器输出。5 秒自动刷新，用手加热 LM35，可以看到数值变化。

思考

如何让网页直接显示温度。

4.3 模块化传感器

1 简介

传感器是将非电物理量转化成电学量的元件。这里以光敏传感器为例：

一般的接线方法：

端口	用途	接线
VCC	传感器电源正极	接 Arduino 电源正极（5V）
GND	传感器电源负极	接 Arduino 电源负极（GND）

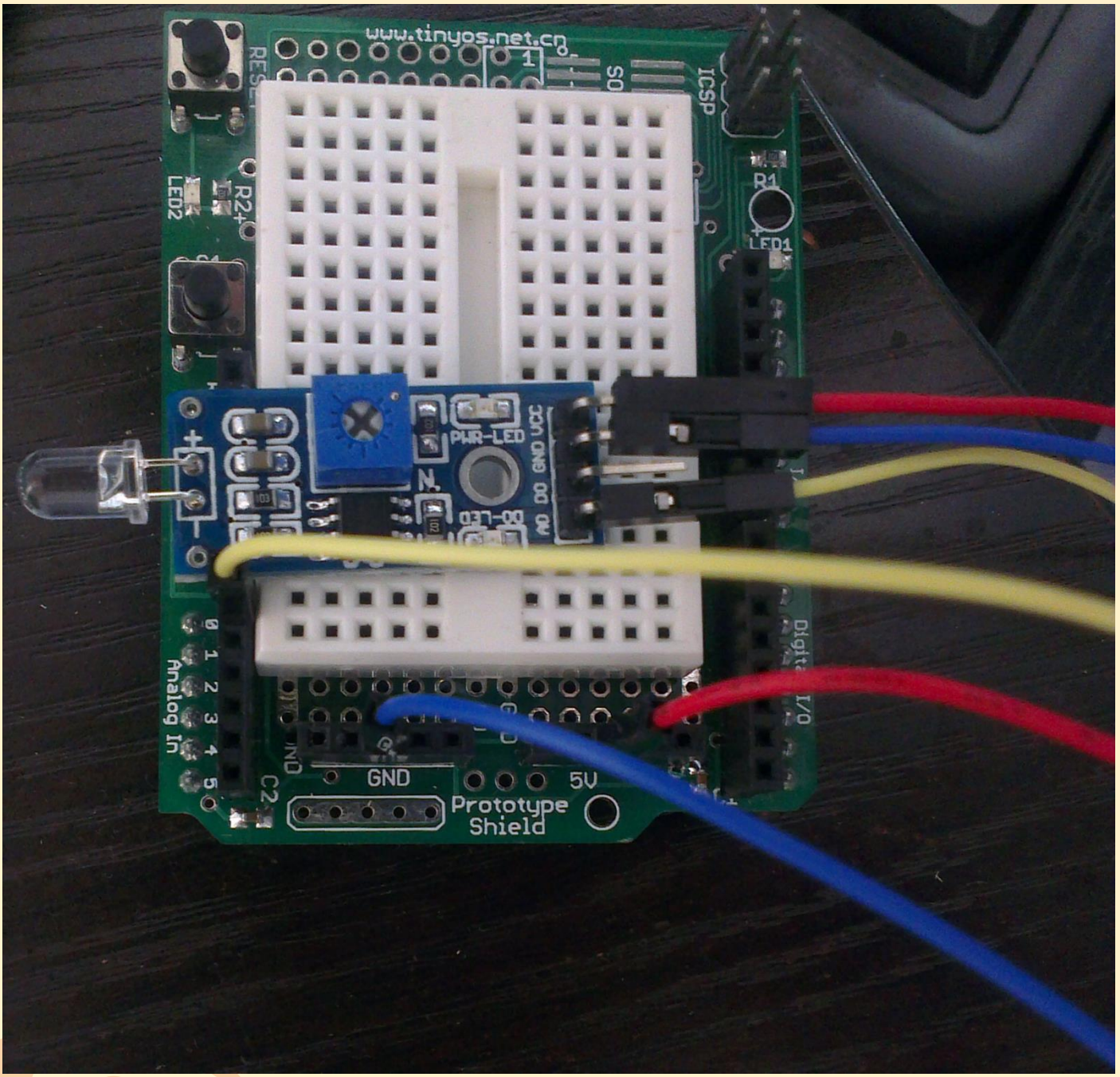
AO	模拟输出。输出 0V-5V，光照越强电压越高。	接 Arduino Analog In 端口，用“电压表”检测输出
DO	数字输出。光强达不到设定阈值时，输出高电平（5V），当外界环境光强超过设定阈值时，输出低电平(0V)	接 Arduino Analog In 端口，用“电压表”检测输出

2 范例：搭建光控开关

1 先断开 arduino 和计算机的连接。不要带电插拔电路！

2 用一头针一头孔的导线将光敏传感器的 VCC 连接到 arduino 的 5V 端口，同样将传感器的 GND 连接到 arduino 的 GND 端口，将传感器的 AO 连接到 Arduino analog In 的 0 口。（如图，这里红线连接传感器 VCC 和 Arduino 5V，蓝线连接传感器和 arduino 的 GND，黄线连接传感器 AO 和 arduino analog in 0）。

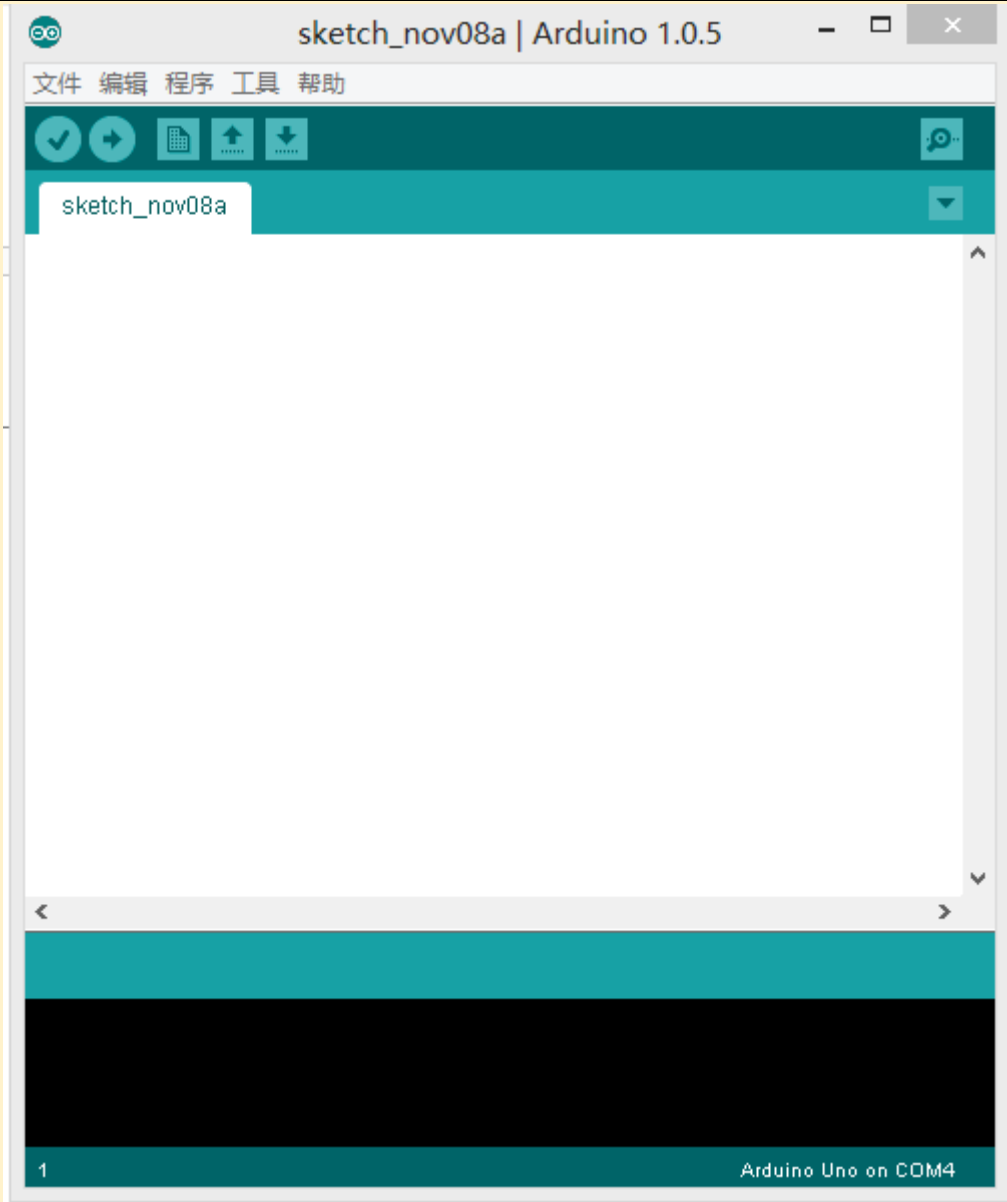
反复检查，千万不要接反！否则会烧毁元件！



3 连接继电器。继电器的 VCC 和 GND 连接同上，继电器的 In 连接到 arduino 的 Digital I/O 8 口。

4 将 arduino 连接到计算机上，此时传感器和继电器上的红灯都会亮起，如不亮则迅速断开与计算机的连接，再次检查连线。

5 打开电脑桌面上的 arduino 程序，出现下面的 Arduino 界面。



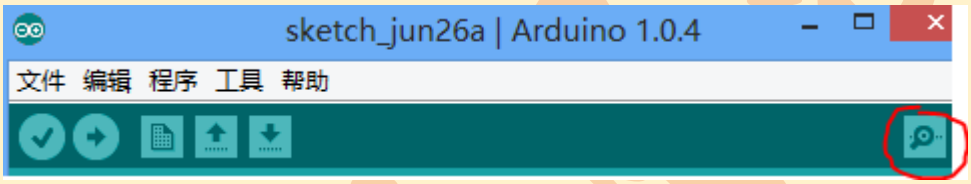
6 将下面的程序复制到编辑器中。//后面是说明。

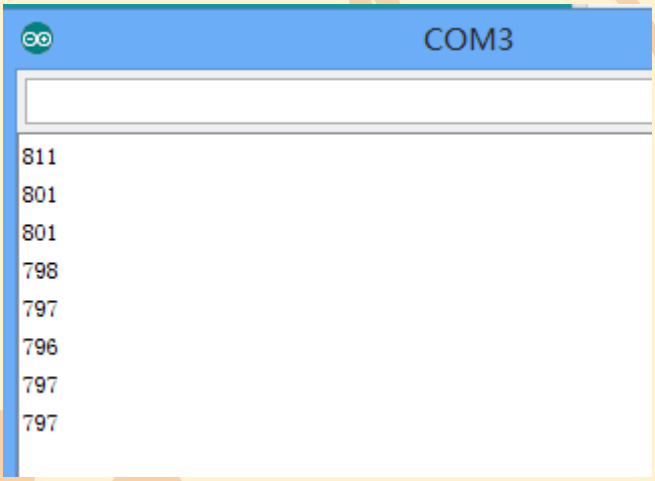
```
void setup()//此段程序在最开始运行一次，进行初始化
{
   Serial.begin(9600);  //初始化 arduino 和计算机的连接，可用于显示数值
   pinMode(8,OUTPUT);//声明 Digital I/O 8 口为输出（电源）
}

void loop()//此段程序将自动反复执行
{
```

```
    int a=analogRead(0); //声明整型变量 a，读取 analog In 0 口电压存储在 a 中
    Serial.println(a);//在电脑上显示 a 的值（0 口电压），0-1023 对应 0-5V
    if(a>900)//如果 0 口电压大于阈值(光线太暗)
      digitalWrite(8,1); //打开第 8 口上的开关
    else //否则
    digitalWrite(8,0); //断开第 8 口上的开关
    delay(500);//延时 0.5 秒
}
```

7 把程序上传到 arduino 中。

打开串口监视器，能看到显示的数值，这是传感器输出的电压。0-1023 对应 0-5V。





用手遮住传感器，该值超过程序中的阈值，继电器动作，会发出声音，状态指示灯改变。

如果将交流照明电路接到继电器的接线端口上，就可以控制它的开关了。

练习

1.完成上述案例后，将 AO 口上的线改接 DO，重复实验观察现象。

2.完成第 1 题后，改用红外避障传感器，程序不变，自己参照上述内容接线并观察现象。

常用模块化传感器参考

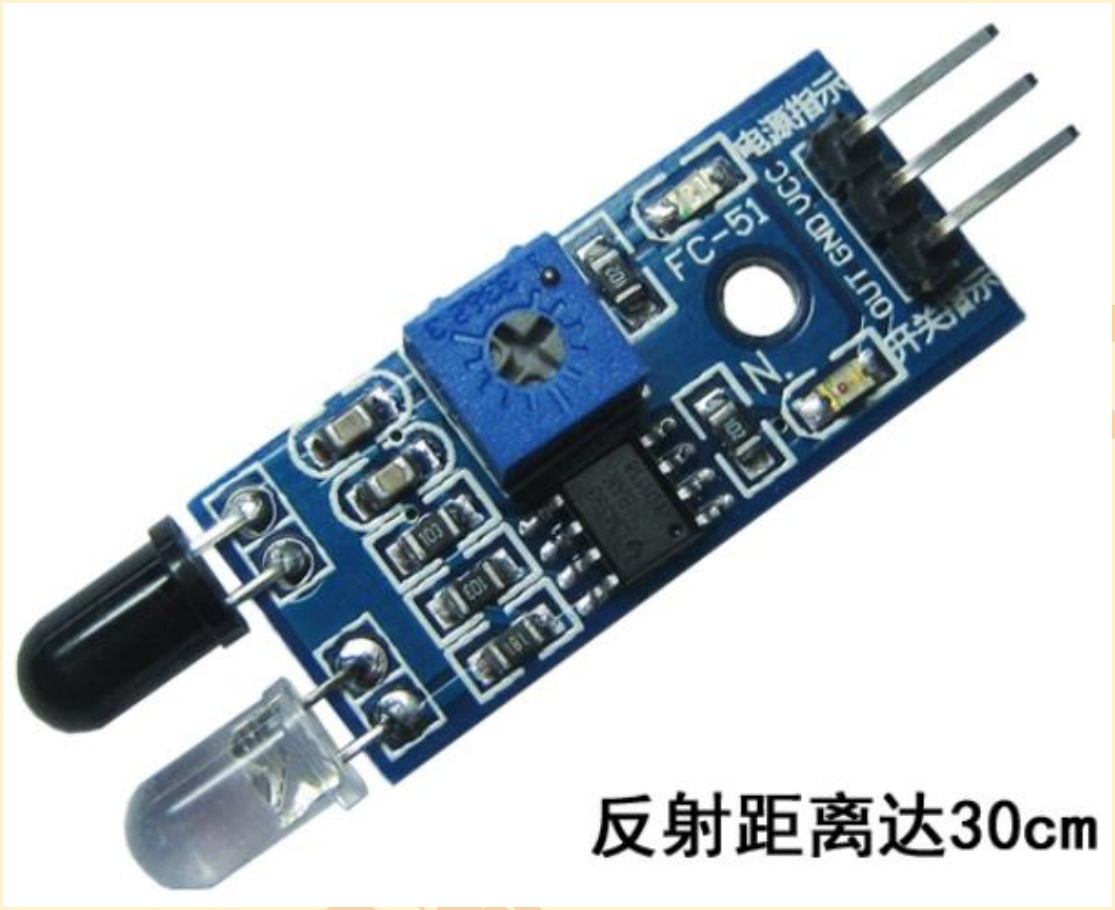
光敏传感器

用途：

光线亮度检测,光线亮度传感器，具有方向性，只感应传感器正前方的光源，用于寻光效果更佳。

与光敏电阻比较，方向性比较好，可以感知固定方向的光源；灵敏度可调（图中蓝色数字电位器调节），工作电压 3.3V-5V；输出形式：DO 数字开关量输出（0 和 1）和 AO 模拟电压输出。模块在无光条件或者光强达不到设定阈值时，DO 口输出高电平，当外界环境光强超过设定阈值时，模块 D0 输出低电平。设有固定螺栓孔，方便安装。

红外避障传感器



1

该模块检测距离 2～30cm，检测角度 35°，检测距离可以通过电位器进行调节，顺时针调电位器，检测距离增加；逆时针调电位器，检测距离减少。（非特殊情况，请勿随意调节电位器）.目标的反射率和形状是探测距离的关键。其中黑色探测距离最小,白色最大;小面积物体距离小,大面积距离大。

模块接口说明:

VCC 外接 3.3V-5V 电压（可以直接与 5v 单片机和 3.3v 单片机相连）

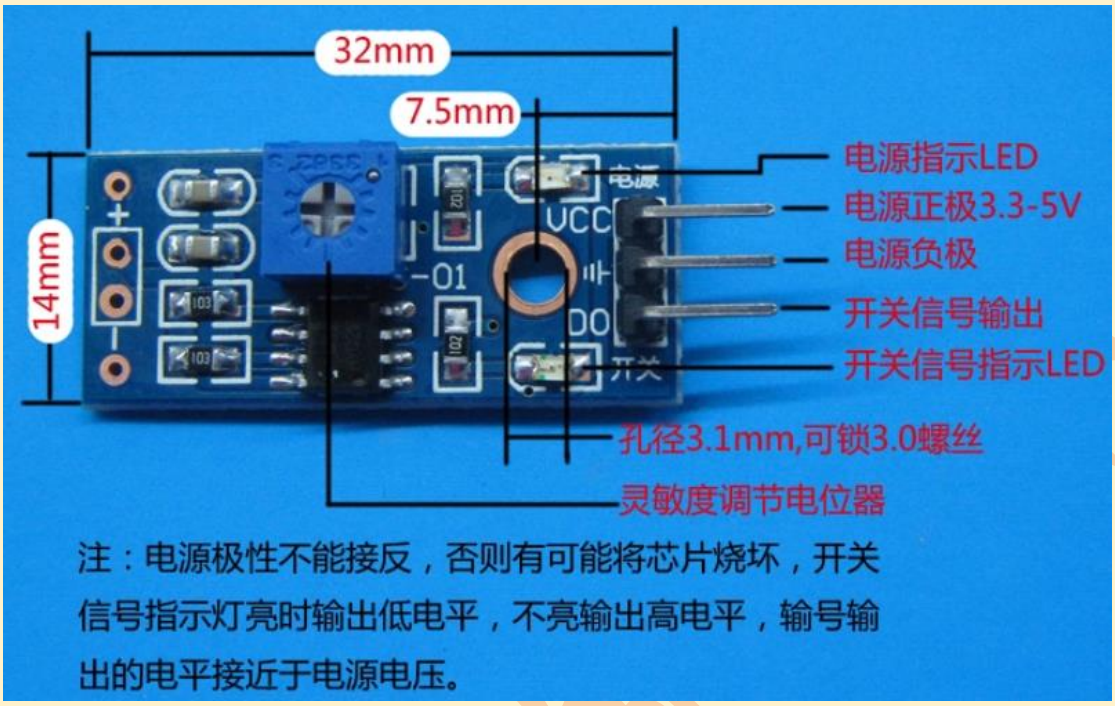
GND 外接 GND

OUT 小板数字量输出接口（0 和 1）

当电源接通时，红色电源指示灯点亮； 当模块检测到前方障碍物信号时，电路板上绿色指示灯点亮电平，同时 OUT 端口持续输出低电平信号。

具有 3mm 的螺丝孔，便于固定、安装；电路板尺寸：3.2CM*1.4CM。

磁敏传感器



工作电压 3.3V-5V。数字开关量输出（0 和 1），电流超过 15mA。

设有固定螺栓孔，方便安装，尺寸：3.2cm x 1.4cm

干簧管需要和磁铁配合使用，在感应到有一定的磁力的时候，会呈导通状态，模块输出低电平，无磁力时，呈断开状态，输出高电平，干簧管与磁铁的感应距离在 1.5cm 之内超出不灵敏或会无触发现象。模块 DO 输出端可以单片机 I/O 口直接相连。

4.5 倾斜开关实验

1 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：

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发化，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。这类开关功效同水银开关，但没有水银开关的环保及安全等问题。

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导针，另一端为银色导针。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。

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。

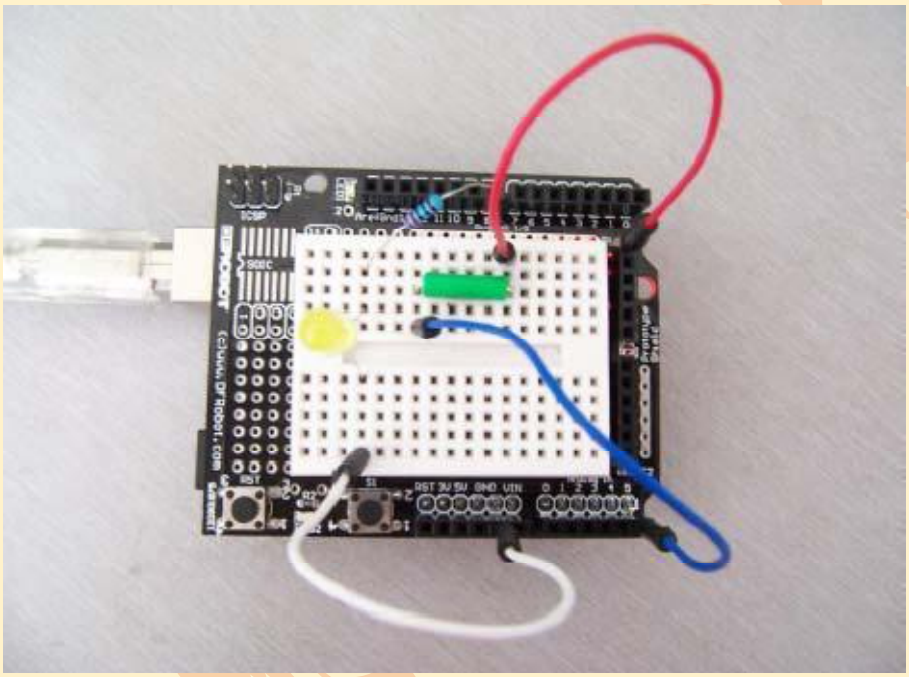
2 倾斜开关控制 led 灯的亮灭

元件清单

倾斜开关：1 个　Led 灯：1 个　220Ω 电阻：1 个　多彩面包板实验跳线：若干

电路连接

按照 Arduino 教程将控制板、Prototype shield 板子、面包板连接好，下载线插好。然后将 led 灯连接到数字 8 引脚，倾斜开关连接到模拟 5 引脚。如图：



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
    i=analogRead(5);//读取模拟 5 口电压值储存到 i
        if(i>1000)//如果大于 1000（接近 5V）
        {
            digitalWrite(8,HIGH);//点亮 led 灯
        }
        else//否则
        {
            digitalWrite(8,LOW);//熄灭 led 灯
        }
}
```

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，开关导通，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。

实验原理

当金色一端低于水平位置倾斜，开关导通，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 4.9V 左右（数字二进制表示为 1000），即可知道是否倾斜开关导通了。

第五章　Arduino“库”的应用

5.1 红外遥控实验

本项目使用 IRremote 库。IRremote 库包含了接收、发送红外遥控信号的功能，使用前必须先将 IRremote 文件夹复制到 Arduino 文件夹下的 libraries 文件夹里，然后重新启动 Arduino 程序，在代码中必须有引用库的语句：#include <IRremote.h>。IRremote 目录里有相应资料可供参考。

元件清单

红外遥控器：1 个　　红外接收头：1 个　　多彩面包线：若干

电路连接

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 12 口，这样就完成了电路部分的连接。本项目使用的 LED 灯为 Arduino 第 13 脚自带的。

红外接收头有三个引脚如下图：

用的时候将 VOUT 接到数字口，GND 接到实验板上的 GND,VCC 接到实验板上的+5v。

程序代码

```
#include <IRremote.h>//告诉系统我们使用了 IRremote 库
decode_results results; //定义保存结果的变量
```

```
int LED_PIN=13;   //LED 灯在数字口 13
int IR_PIN=12;    //红外接收管接在 12 脚
IRrecv irrecv(IR_PIN);    //初始化红外分析程序

void setup()
{
  Serial.begin(9600);  //初始化串口通信，一会儿可以在电脑上看到解码结果
  irrecv.enableIRIn(); //  初始化红外接收程序
  pinMode(LED_PIN,OUTPUT);
}

void loop() {
  if (irrecv.decode(&results)) {    //如果红外接收头接收到了信息，那么…
    Serial.println(results.value, HEX);  //显示当前按键对应的代码

    if(results.value==0xFFA25D)        //如果按下的是关键，那么…
      digitalWrite(LED_PIN,LOW);          //关灯
    if(results.value==0xFFE21D)        //如果按下的是开键，那么…
      digitalWrite(LED_PIN,HIGH);           //开灯

    irrecv.resume(); // Receive the next value    //重置红外接收器以便下一次工作
  }
}
```

程序功能

被控制的灯在底层的板子上。按下遥控器第一行红色开关键，灯熄灭，按第一行绿色键灯亮。

实验原理

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.

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、解调发、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。

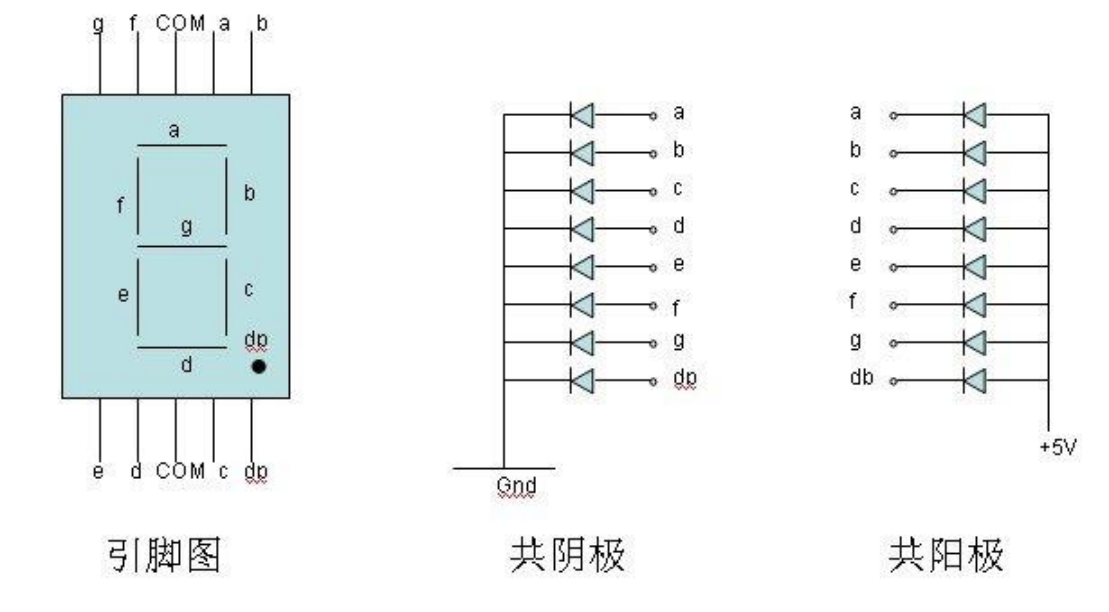
软件工作原理见注释。我们可以获取其他按键代码，扩展更多功能。

5.2 数码管实验

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；

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。



将数字 I/O 和数码管的字段引脚相连，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。公共极上需要接限流电阻，否则电流过大会烧毁发光二极管的。

元件清单

数码管：1 个　220Ω的电阻：1 个　多彩面包板实验跳线：若干

电路连接

按照 Arduino 教程将控制板、Prototype shield 板子、面包板连接好，下载线插好

数码管的端口连接参见程序。COM 端接 5V（共阳），注意需要接限流电阻。

程序代码

```
 //设置控制各段的数字 IO 脚
#include <display.h>
display dis(5,6,7,8,9,10,11,4);//a,b,c,d,e,f,g,dp 管脚

void setup()
{
```

```
}
void loop()
{
   dis.set(1);//数字 1
   delay(2000);//延时 2s
   dis.set(2);//数字 2
   delay(2000);
   dis.set(3);
   delay(2000);
   dis.set(4);
   delay(2000);
   dis.set(5);
   delay(2000);
   dis.set(6);
   delay(2000);
   dis.set(7);
   delay(2000);
   dis.set(8);
   delay(2000);
}
```

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。

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，只要将相应的段点亮即可。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

一个数字显示的时间由延时时间来决定，时间设置的大些，显示的时间就长些，时间设置的小些，显示的时间就短。

本程序使用了老师开发的 Display 库。在 Arduino\library\display 中有相关的代码。使用时，#include <display.h>这句是引入“库”，display dis(5,6,7,8,9,10,11,4);这句对数码管进行初始化，设定各个端口。在 loop()中，使用 dis.set(1);就可以显示 1，依此类推。

练习

结合前面的交通灯项目，制作包含倒计时的交通灯。

5.3 红外遥控数码管

元件清单

红外遥控器：1 个　红外接收头：1 个　多彩面包线：若干 数码管：1 个

220Ω 电阻：1 个

电路连接

参照数码管实验（简化方式）连接数码管；接着将红外接收头按照上述方法接好，将 VOUT 接到数字 12 口。将这两个实验的程序结合即可得到本实验的程序。

程序代码

```
#include <IRremote.h>//告诉系统我们使用了 IRremote 库
#include <display.h>
display dis(5,6,7,8,9,10,11,4); //a,b,c,d,e,f,g,dp 管脚

decode_results results; //定义保存结果的变量
int IR_PIN=12;     //红外接收管接在 12 脚
IRrecv irrecv(IR_PIN);     //初始化红外分析程序
```

```
void setup()
{
  Serial.begin(9600);  //初始化串口通信，一会儿可以在电脑上看到解码结果
  irrecv.enableIRIn(); // 初始化红外接收程序
}
void loop()
{
  if (irrecv.decode(&results)) {    //如果红外接收头接收到了信息，那么…
    Serial.println(results.value, HEX);  //显示当前按键对应的代码

    if(results.value==0xFF30CF)          //如果按下的是 1 键，那么…
      dis.set(1);
    if(results.value==0xFF18E7)
      dis.set(2);
    if(results.value==0xFF7A85)
      dis.set(3);
    if(results.value==0xFF10EF)
      dis.set(4);
    if(results.value==0xFF38C7)
      dis.set(5);
    if(results.value==0xFF5AA5)
      dis.set(6);
    if(results.value==0xFF42BD)
      dis.set(7);
    if(results.value==0xFF4AB5)
      dis.set(8);
    irrecv.resume(); // Receive the next value     //重置红外接收器以便下一次工作
  }
```

```
}
```

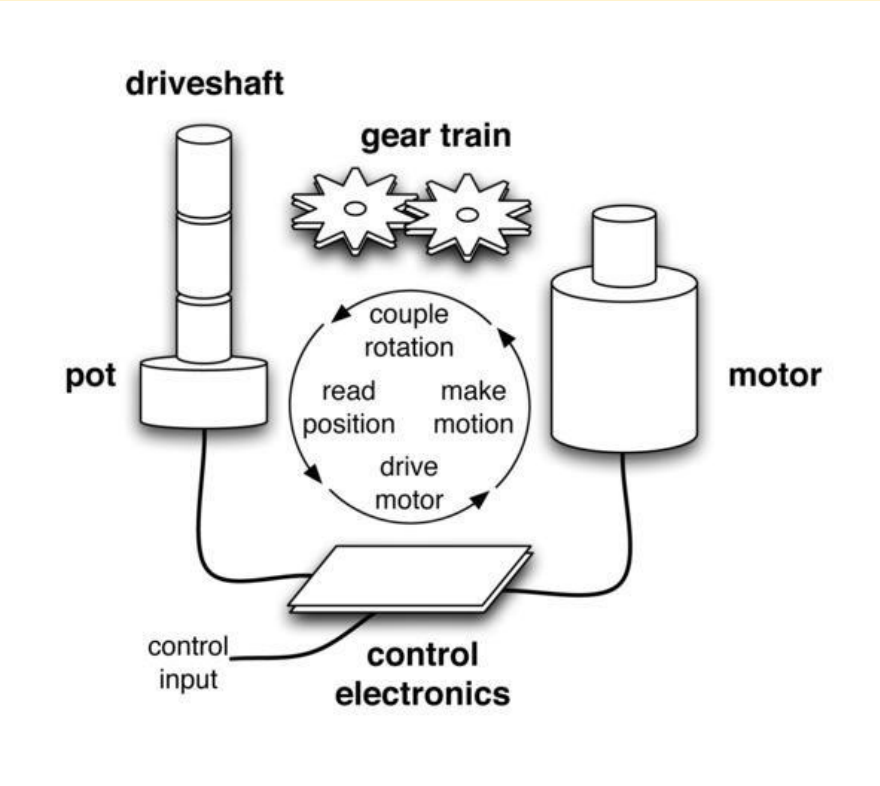
第六章 Arduino 在机电一体化中的应用

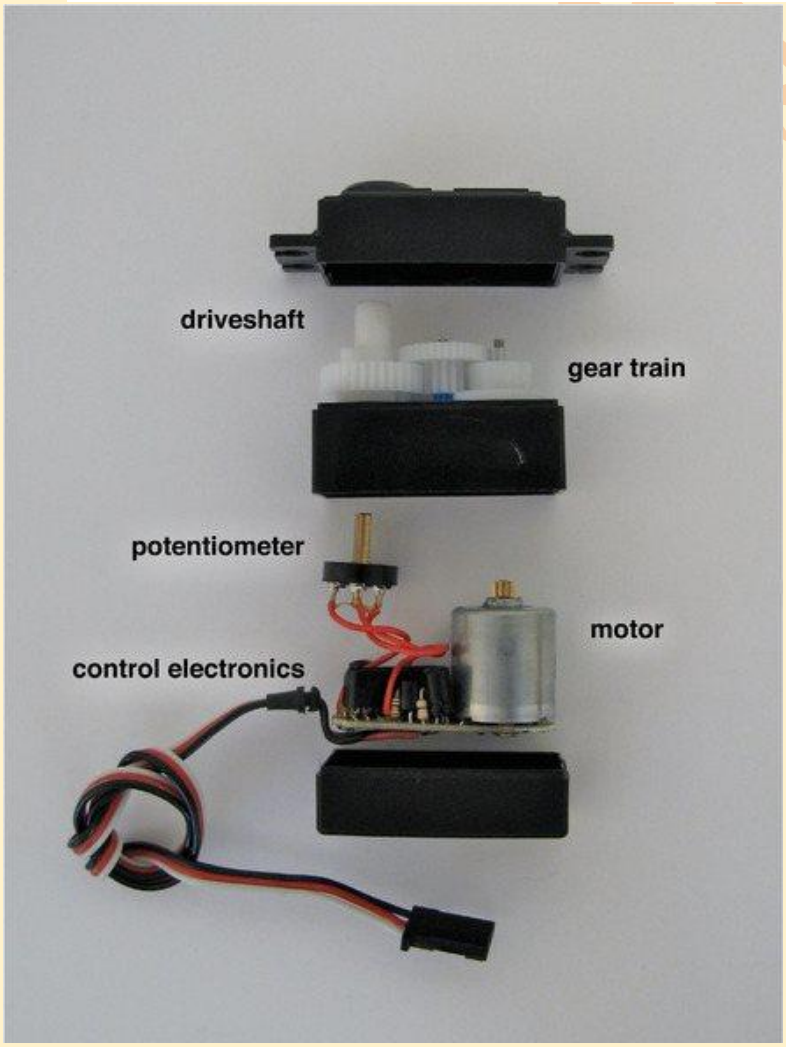
6.1 舵机基础知识

基本原理与组成

舵机常用在机器人技术，控制玩具汽车和飞机等。舵机的旋转不像普通电机那样只是古板的转圈圈，它可以根据你的指令旋转到 0 至 180 度之间的任意角度然后精准的停下来。如果你想让某个东西按你的想法运动，舵机可是个不错的选择。

舵机是个糅合了多项技术的科技结晶体，它由直流电机、减速齿轮组、传感器和控制电路组成，是一套自动控制装置。对于舵机而言，位置检测器是它的输入传感器，舵机转动的位置一变，位置检测器的电阻值就会跟着变。通过控制电路读取该电阻值的大小，就能根据阻值适当调整电机的速度和方向，使电机向指定角度旋转。下图显示的是一个标准舵机的部件分解图和舵机闭环反馈控制的工作过程。


driveshaft
gear train
pot
motor
couple rotation
read position
make motion
drive motor
control input
control electronics


driveshaft
gear train
potentiometer
motor
control electronics

舵机选型

图：大扭力/微型/标准舵机

● 舵机的形状和大小多到让人眼花缭乱，但大致可以如图所示分类。最右边身材不错的是常见的标准舵机，中间两个小不点是体积最小的微型舵机，左边的魁梧的那个是体积最大的大扭力舵机。它们都是同样的三线控制，因此你可以根据需求换个大个的或小个的。

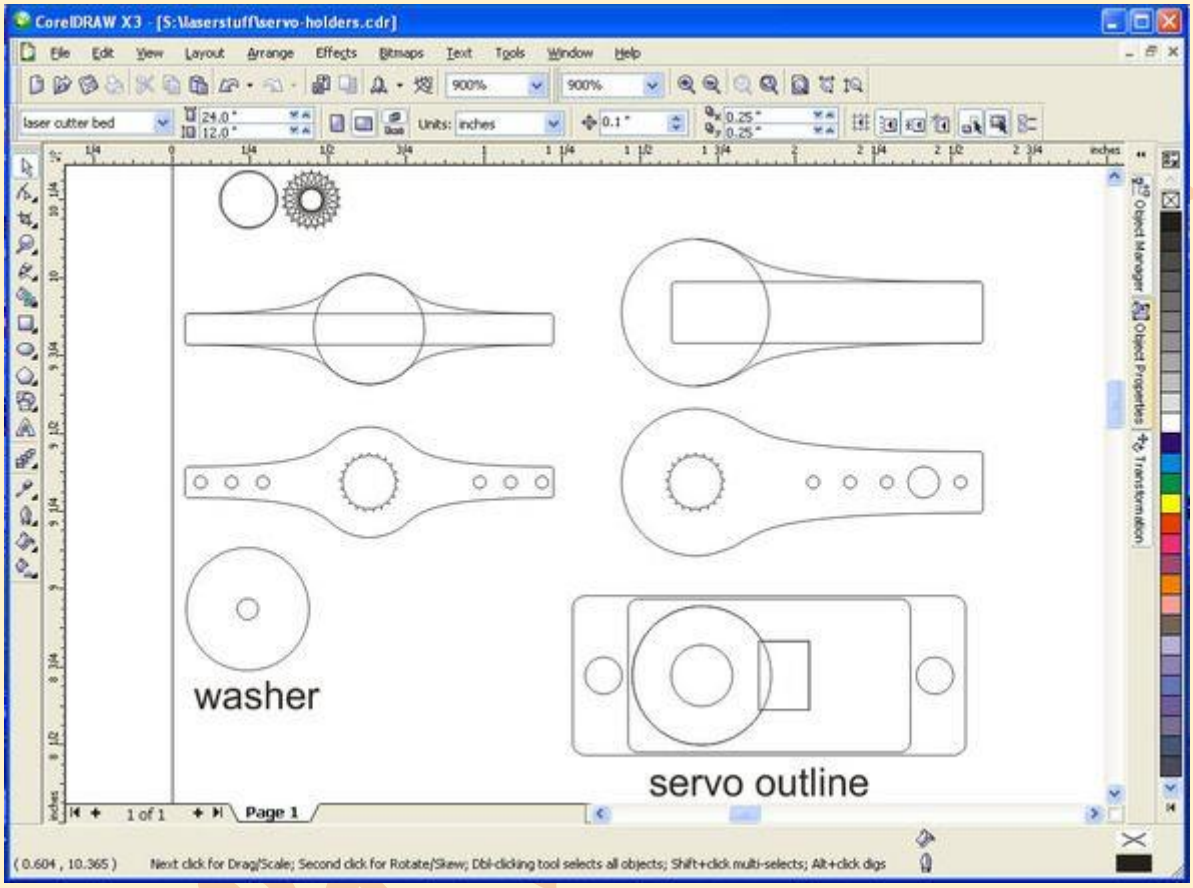
● 除了大小和重量，舵机还有两个主要的性能指标：扭力和转速，这两个指标由齿轮组和电机所决定。扭力，通俗讲就是舵机有多大的劲儿。在 5V 的电压下，标准舵机的扭力是 5.5 千克/厘米（75 盎司/英寸），转速很容易理解，就是指从一个位置转到另一个位置要多长时间。在 5V 电压下，舵机标准转度是 0.2 秒移动 60 度。总之，和我们人一样，舵机的个子越大，转的就越慢但也越有劲儿。确定做什么之后，选择哪种大小的舵机（标准型、微型、绞盘型）就是小 case 了。

机械连接

想在你的项目中用上舵机，就要满足两个条件：一是需要个能把舵机固定到基座上的支架，二是得有个能将驱动轴和物体连在一起的连接装置。支架一般舵机上就有，而且带有拧螺丝用的安装孔。如果你仅仅是测试的话，用点儿热熔胶或者双面泡沫胶带就能轻松的固定住舵机。

怎样连接驱动轴呢，你会发现舵机都附带了一些有孔的小东西，这就是舵盘，它可以套在驱动轴，臂上打上了些小孔。你只要用连接棒或者线把物体连到孔上，就可以将舵机的旋转运动变成物体的直线运动了，当然了，选用不同的舵盘或固定孔就能产生不同的运动啦。

图示的是几种不同的舵盘。前面 4 个白色的是舵机附带的舵盘，右边四个是用激光切割机切割塑料得到的 DIY 舵盘。最右边的 2 个是舵盘和支架的组合，如果你想实现两个舵机的组合运动，把这个舵盘的支架固定到另一个舵机的支架上就 OK 了。

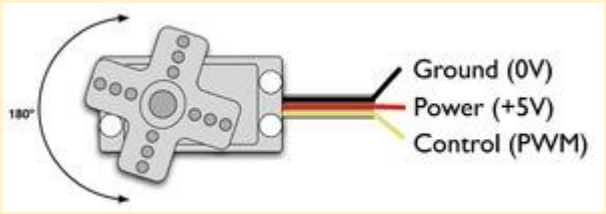


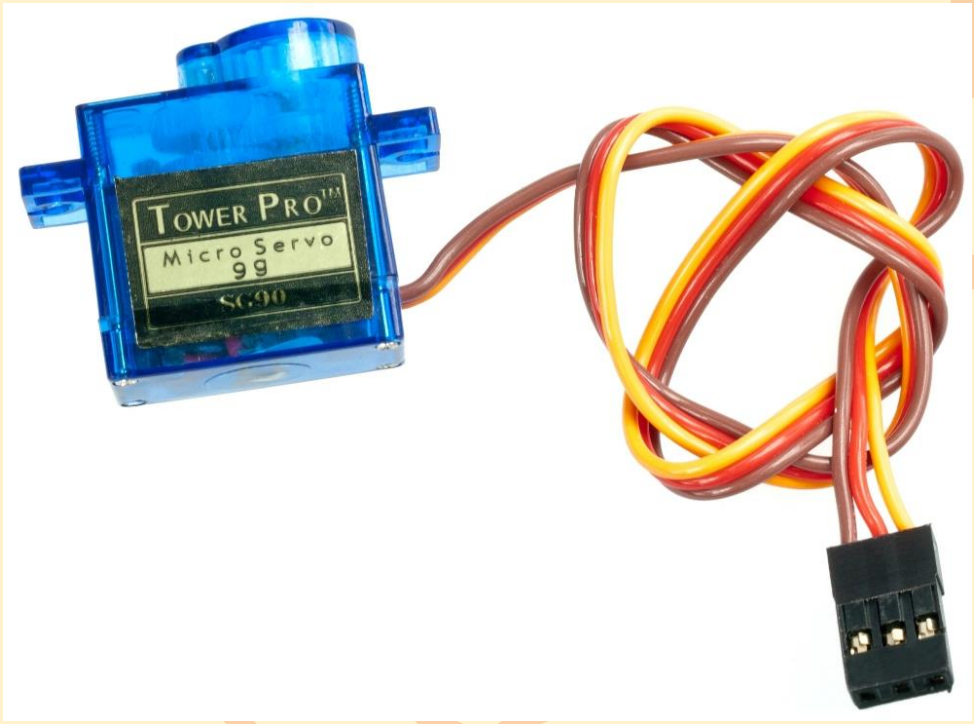
6.2 舵机控制入门

元件清单

Arduino UNO 1 块，传感器扩展板或面包板一块，舵机一个。如果用面包板，还需要两头针杜邦线 3 条。

电路连接





舵机有一个三线的接口。黑色（或棕色）的线是接地线，红线接+5V 电压，黄线（或是白色或橙色）接控制信号端。将 5V 线连接到单片机 5V 或 VCC 端口，将接地线连接到单片机 GND 口，将控制线连接到单片机任意数字（Digital I/O）端口（除 0，1 外）。本例中连接到第 9 脚上。

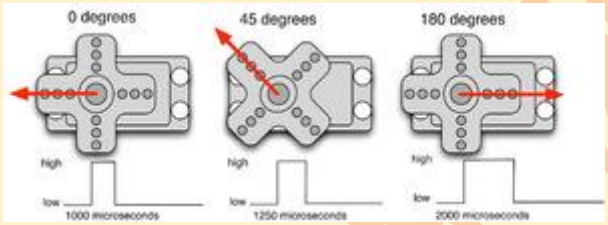
特别提示：VCC GND 切勿接反！！

程序代码

```
#include <Servo.h> //启用舵机库
Servo myservo;   // 声明一个舵机变量，变量名 myservo
void setup() //初始化程序，只在最开始执行一次
{
```

```
    myservo.attach(9);   // 舵机接在 D9 口
}
void loop() //执行完 setup 后，loop 将会循环执行
{
        myservo.write(45);// 命令舵机旋转 45 度（允许 0-180）
        delay(2000);//暂停 2 秒
        myservo.write(135);// 命令舵机旋转 135 度（允许 0-180）
        delay(2000);//暂停 2 秒
}
```

实验原理



使用舵机必备程序：首先启动舵机库

#include <Servo.h>

然后声明舵机类型的变量（用几个就声明几次，并为不同的舵机取不同的名字）。

Servo 舵机名字;

在初始化 setup 程序中，为每个舵机指定连接的端口，方法：

舵机名字.attach(连接端口)；

在 loop 中实现控制：

舵机名字.write(旋转角度)；

此处角度只能是 0-180.

思考

遥控激光炮设计：用 1 个舵机，1 个 5V 激光二极管，红外遥控器和接收管，arduino 开发板完成遥控激光炮。

（1） 实现：可以按键控制激光是否发射；可以按键使激光头旋转。

（2）拓展任务：用 2 个舵机实现旋转和俯仰发射。

提示：

激光管直接接 5V 点亮，注意有极性，红线为正。舵机注意接线正确。

Attach 命令必须写在 void setup()里，write 命令写在合适的地方

6.3 MiniQ 小车控制

1 小车组装

工具：什锦改锥，导线，烙铁套装

材料：小车散件

过程：参考下面页面

http://www.dfrobot.com.cn/goods-468.html

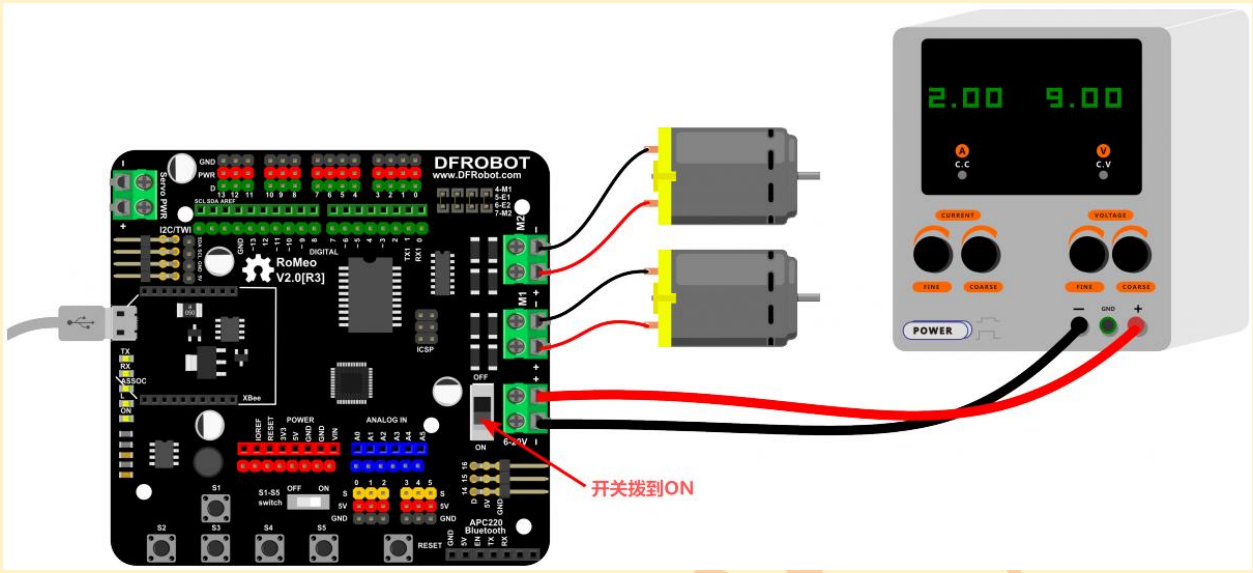
2 Romeo 控制程序

完整内容参考下面的链接。

http://wiki.dfrobot.com.cn/index.php/(SKU:DFR0225)RoMeo_V2

RoMeo 上集成了 2 路电机驱动，这是为了让机器人爱好者节约大量制作硬件的时间，而把开发重点放在软件上。电机驱动电路采用 L298 芯片，峰值电流可达 2A。当使用电机驱动时，又会涉及到电源供电问题，如下图所示的接线方法，M_VIN 接正极，GND 接负极。还有就是记得把开关拨到"ON"端。

外接电源取 6-9V 为宜。



引脚	功能
4	电机 1 方向控制
5	电机 1PWM 控制
6	电机 2PWM 控制
7	电机 2 方向控制

电机驱动电路控制端使用短路跳线选通，用的时候接通，不用就断开。

演示代码：

```
int E1 = 5;       //定义 M1 使能端
int E2 = 6;       //定义 M2 使能端
int M1 = 4;       //定义 M1 控制端
int M2 = 7;       //定义 M1 控制端
void stop(void){                    //停止
        digitalWrite(E1,LOW);
        digitalWrite(E2,LOW);
}

void advance(char a,char b){              //前进
        analogWrite (E1,a);               //PWM 调速
```

```
        digitalWrite(M1,HIGH);
        analogWrite (E2,b);
        digitalWrite(M2,HIGH);
}
void back_off (char a,char b) {                 //后退
        analogWrite (E1,a);
        digitalWrite(M1,LOW);
        analogWrite (E2,b);
        digitalWrite(M2,LOW);
}
void turn_L (char a,char b) {                   //左转
        analogWrite (E1,a);
        digitalWrite(M1,LOW);
        analogWrite (E2,b);
        digitalWrite(M2,HIGH);
}
void turn_R (char a,char b) {                   //右转
        analogWrite (E1,a);
        digitalWrite(M1,HIGH);
        analogWrite (E2,b);
        digitalWrite(M2,LOW);
}

void setup(void) {
    int i;
    for(i=4;i<=7;i++)
    pinMode(i, OUTPUT);
    Serial.begin(19200);          //设置串口波特率
}
```

```
void loop(void) {
    if(Serial.available()>0){
        char val = Serial.read();
        if(val!=-1){
            switch(val){
                case 'w'://前进
                        advance (100,100);     //PWM 调速
                        break;
                case 's'://后退
                        back_off (100,100);
                        break;
                case 'a'://左转
                        turn_L (100,100);
                        break;
                case 'd'://右转
                        turn_R (100,100);
                        break;
                }
            delay(40);
         }
        else stop();
    }
}
```

程序功能：串口输入"w","s","a","d"，电机会有相应的动作。

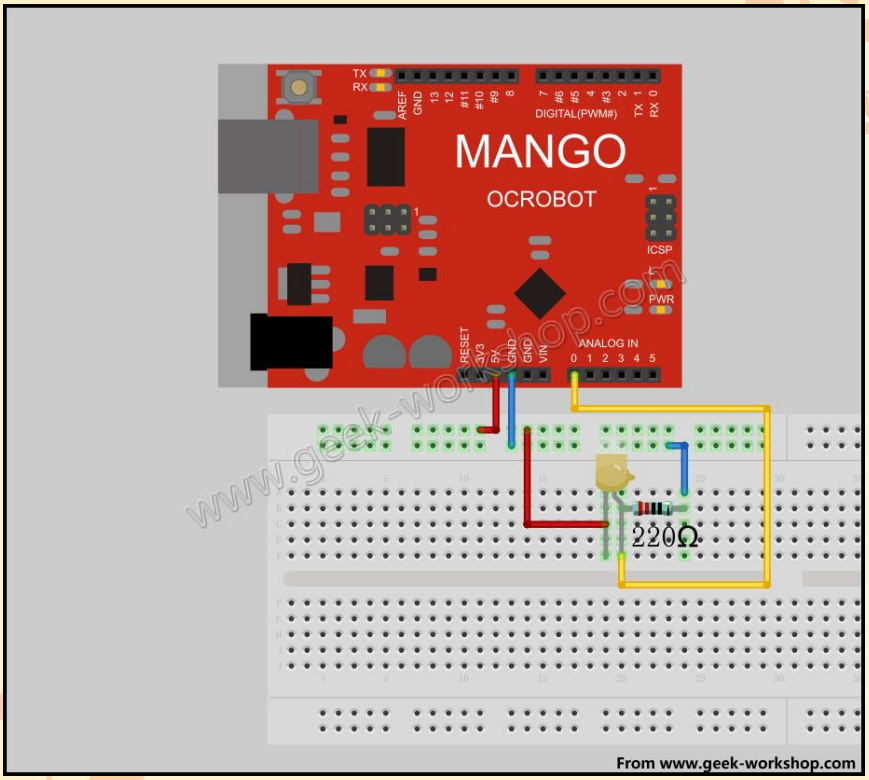
第七章 Arduino 其他应用

7.1 Arduino 光频闪波形计

元件清单

光敏三极管 1 个，220Ω 的电阻：2 个，多彩面包板实验跳线：若干

电路连接



采集光敏三极管的光强度数据。实物图：

程序代码

Arduino 程序：

```
void setup() {
    Serial.begin(9600);              //使用 9600 速率进行串口通讯
}
 void loop() {
   int n = analogRead(A0);       //读取 A0 口的值

 Serial.println(n);                          //串口输出光强度数据
 }
```

串口绘图程序： https://code.google.com/p/serialchart/

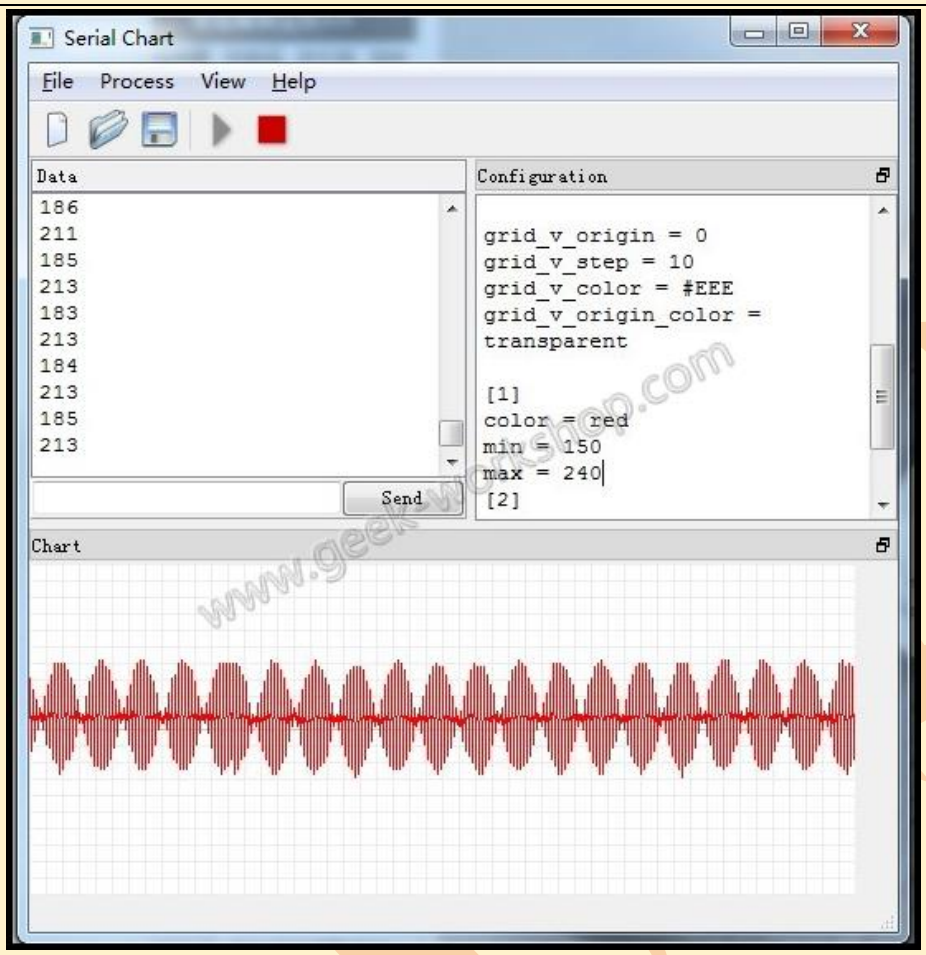
下面程序填写到串口绘图程序的右边窗口内

[_setup_]

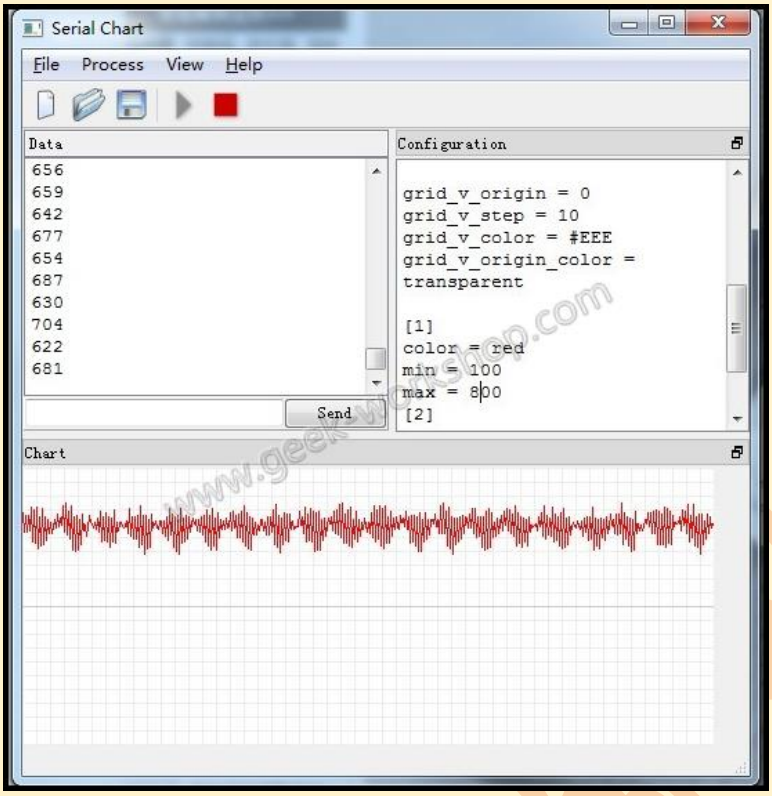
```
port=COM3
baudrate=9600

width=500
height=201
background_color = white

grid_h_origin = 100
grid_h_step = 10
grid_h_color = #EEE
grid_h_origin_color = #CCC

grid_v_origin = 0
grid_v_step = 10
grid_v_color = #EEE
grid_v_origin_color = transparent

[1]
color = red
min = 0
max = 1024
```

程序功能和实验原理

Arduino 通过串口发送光强度数据。SerialChart 程序的前两句就是定义串口号和波特率 ，最后三句的意思是用红色的线条绘制，最大范围是 1024 最小范围是 0 根据自己的需要来调整这些值，使线条在画面中间方便观察。



这张图是钨丝灯的频闪图，很明显，钨丝灯和市电的频率相同，跟着市电的频率闪烁。当然这种闪烁人眼是看不出来的。



这个是飞利浦节能灯的频闪图。

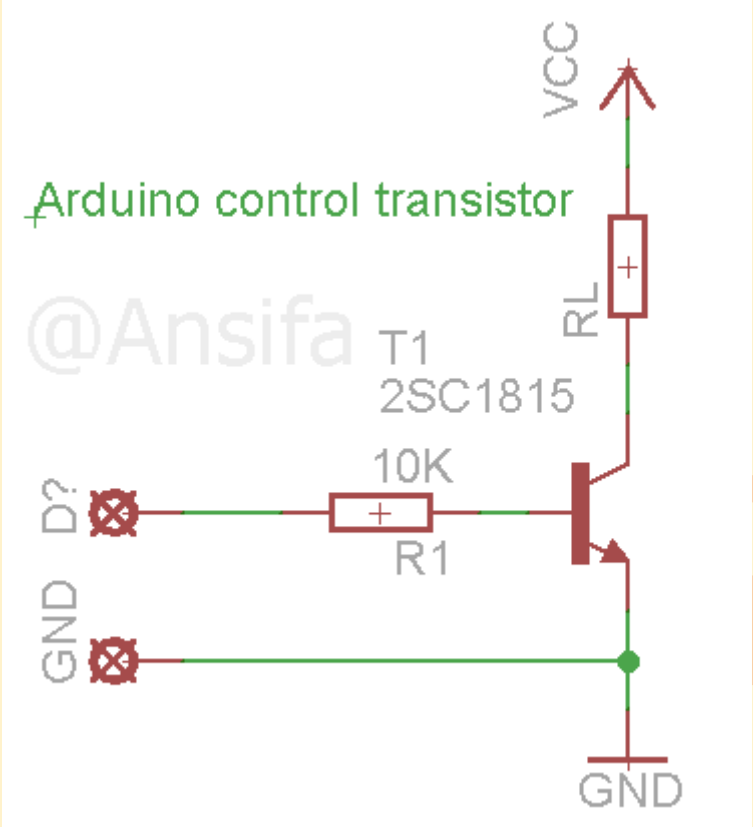
更换传感器，可以得到各种物理量的波形。该软件仅支持 com1-com9 这几个逻辑 com 口。所以您的 arduino com 口号大于 10 的话就要对 COM 口号进行修改。

7.2 Arduino 控制大功率负载解决方案

使用 Arduino 时候，经常需要控制大电流或者高压负载。但是 Arduino 只有最大 5v，20mA 的引脚输出。必须要通过扩流电路。扩流的方法有很多。下面介绍几种方法，适合在不同场合下面使用。

注意：下述电路图的负载均用电阻符号代替，符号标志是电子学的负载符号 RL，就是 R（load）的意思。

小功率 NPN 三极管扩流（适用于扩展后负载电压 5v 以下，负载电流建议<1A）。Arduino 输出引脚直接连电阻驱动三极管基极。



复杂度：★★☆☆☆；成本：★☆☆☆☆；可扩流倍数：★☆☆☆☆；

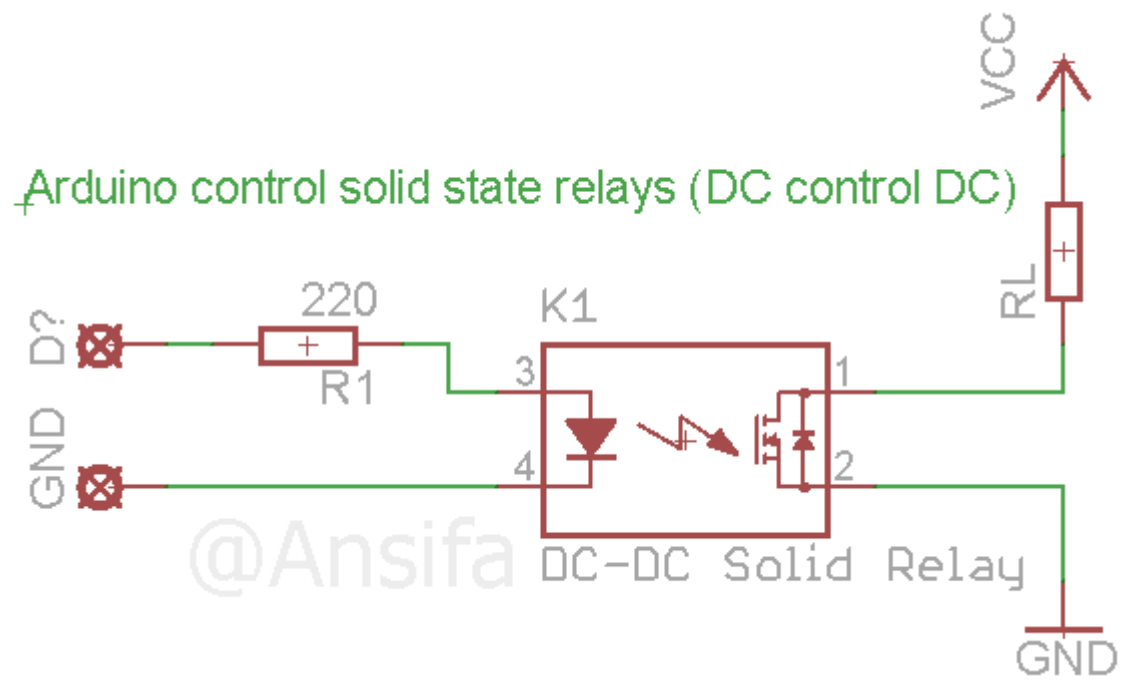
优点：简单方便，成本低。开关频率上限直接由三极管决定，可以做的很高；

缺点：受控大电流和 Arduino 直接连通，所以外置驱动电源不建议超过 5v，以免外置电源的电压通过 Q1 倒灌到 Arduino 引脚引起 Arduino 烧坏。

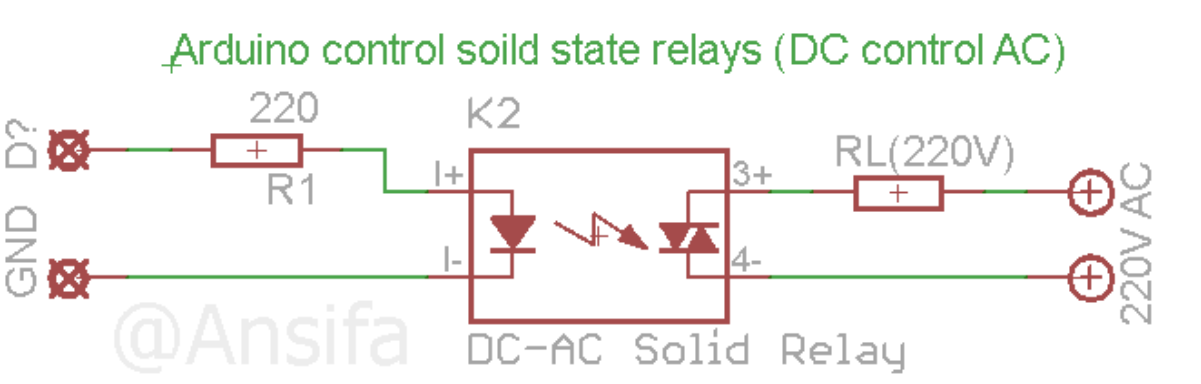
选材：三极管 Q1 可以选用小功率的 NPN 三极管。推荐型号有 2SC1815，2N2222，8050，2SD882 等(点元件名看对应数据手册，下同)；基极电阻 R1 必不可少，否则会导致 Arduino 因为引脚负载过大而发热甚至烧毁。R1 阻值在 100Ω~10k 之间均可，推荐值 1kΩ。所有电阻功率无要求，贴片 0805 以上，直插 1/8w 以上的就行，以下所有电路均使用这种电阻规格。

2、固态继电器（光耦）扩流（适用于 220V 交流直接控制，或者大功率直流控制，建议用于负载电流 0.2A~40A 间）

使用现成的固态继电器可以很方便的被 Arduino 控制。对于 Arduino 来说，驱动固态继电器就像驱动一个 LED 那么简单。



直流控制直流



直流控制交流

复杂度：★☆☆☆☆；成本：★★★★★；可扩流倍数：★★★★★；

优点：使用最简单，抗干扰能力最强，无电磁干扰。可以控制交流电/直流电，并且可以控制很大电流的负载。

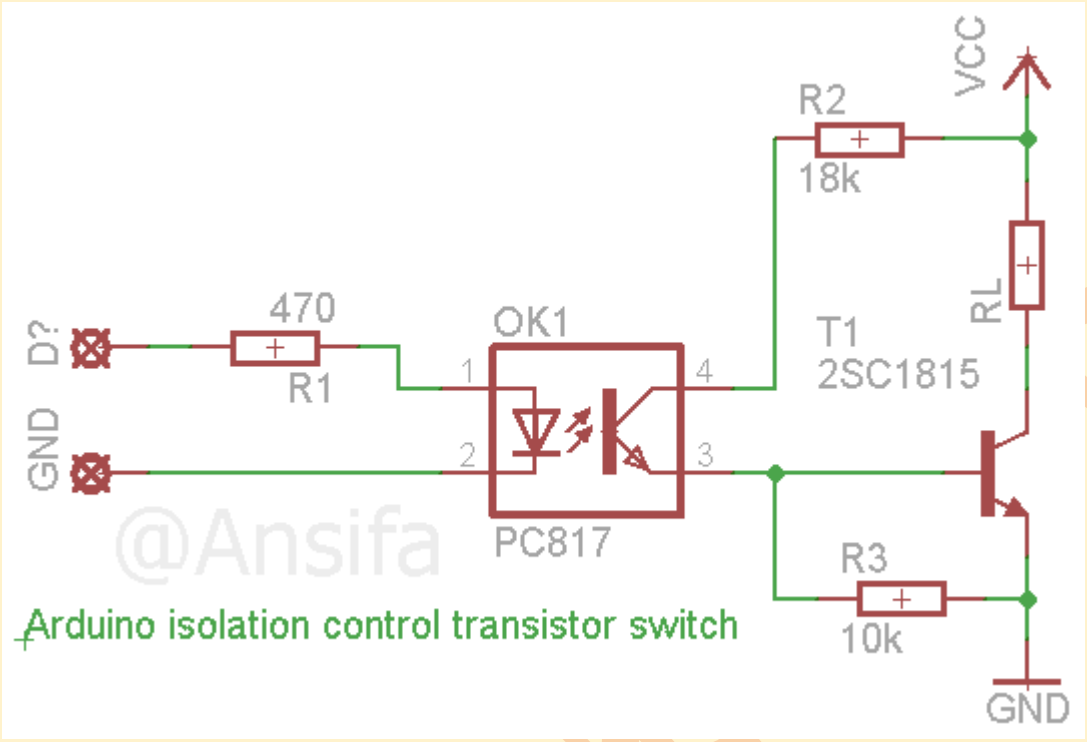
缺点：成本很高

选材：注意的是，固态继电器有两种：直流控制交流固态继电器/直流控制直流固态继电器。它们的受控端有本质的区别，不能混用。直流控交流的交流是用可控硅进行开关的，而直流控直流用的是三极管或者场效应管进行开关。下面会对两种继电器仿制进行介绍的（见 3）。

3、带光耦隔离 NPN 三极管扩流（适用于大范围负载电压，建议用于负载电流 1A~5A

间）

通过光耦加扩流做出一个控制/受控隔离的直流控制器。完全等效于直流固态继电器。



复杂度：★★★★☆；成本：★★☆☆☆；可扩流倍数：★★★★★；

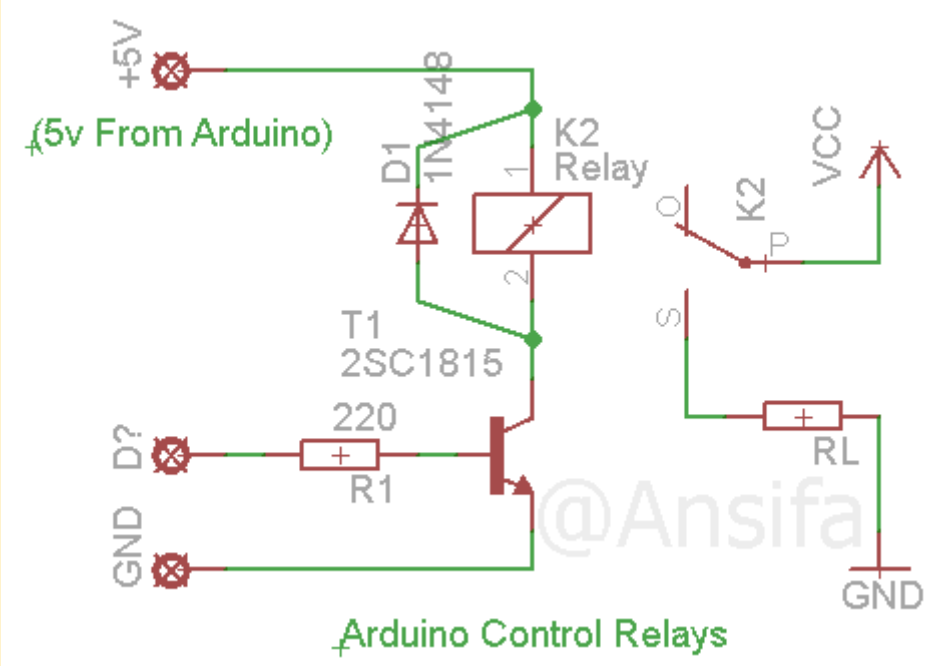
优点：控制电流小，等效于驱动一只 LED。受控大电流和 Arduino 控制板完全电气隔离，即使受控部分发生事故烧毁了，也不会影响到 Arduino 主板。

缺点：电路比直接扩流复杂，成本稍高。

选材：三极管可以选择 2SD882，2SD669A，TIP122(5A 达林顿管)等。光耦使用 PC817 等廉价光耦即可。

4、继电器扩流（适用于低速，对受控端开关电阻有要求的场合，建议用于负载电流 0~3A 间）

用一个小功率三极管扩流，然后控制一个 5v 的继电器。



复杂度：★★★★☆；成本：★★★☆☆；可扩流倍数：★★★★★；

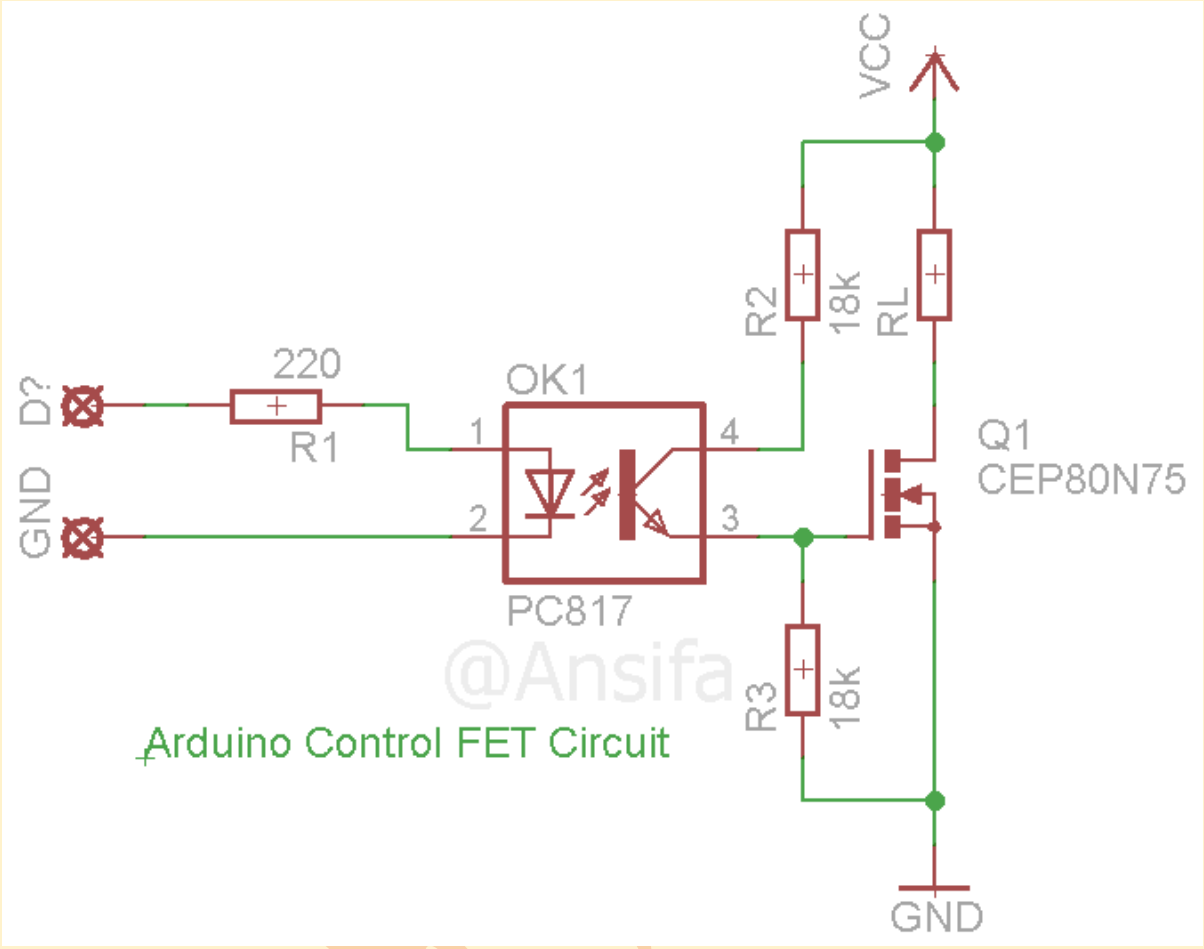
优点：扩流电流大，并且由于继电器是机械闭合触点，闭合电阻基本为零，不像固态继电器或者三极管，有正向压降；适用于对闭合电阻要求高的场合，比如受控端是开关 0~0.7v 的信号。使用三极管或者固态继电器就不能工作了，只能使用继电器。

缺点：低速，每秒最快只能开关几次；机械开关使用寿命短，开关频率高的话，很快就会坏掉。成本高，电路也不简单；开关电流大，需要充足的电源供给继电器吸合。有较强的空间电磁干扰（EMI），会对高速数字电路（USB，串口，视频等）或者小信号模拟电路（音频信号线，仪器测量输入线）造成干扰。必须做好屏蔽措施；并且继电器断开时候会产生反向高压，必须处理反压。

选材：继电器必须选用 5v 控制的，因为 Arduino 只有 5v。并且继电器吸合电流必须小于 200mA，不能影响 Arduino 使其工作电压不稳。如果不能满足的话，可以尝试继电器级联，即小继电器拖动大继电器。

5、场效应管扩流（适用于大负载直流电流，建议用于负载电流 5A~100A 间）

用场效应管代替三极管扩流。由于场效应管属于电压控制型器件，输入电流极小。与三极管扩流相比，可以获得更快的开关速度和更小的输入电流，并且可以控制很大的直流电流（比如 10~50A）。用 Arduino 驱动的话。驱动电路跟 1、3 几乎一样，只是换了场管。



复杂度：★★★★☆；成本：★★★★☆；可扩流倍数：★★★★★；

优点：控制电流小，等效于驱动一只 LED。受控大电流和 Arduino 控制板完全电气隔离，即使受控部分发生事故烧毁了，也不会影响到 Arduino 主板。有最高的控制速度，并且电流也可以做的非常大。

缺点：电路比较复杂，场效应管成本比三极管更高。

选材：场效应管可以使用普通的 N 沟道增强型场效应管（N-Channel Enhancement Mode Field Effect Transistor）。常用型号型号有：CEP80N75（75V,80A,75W），IXGQ240N30P(IGBT 管,240A,300V,500W)，IRF630(9A,200V,75W)等。

7.3 Yeelink 物联网开发

注册

http://www.yeelink.net/developer/doc/11

控制 LED 灯（注意要接限流电阻）

http://www.yeelink.net/developer/doc/42

远程显示温度

http://www.yeelink.net/developer/doc/48

远程控制 LED 程序:

```
/*
 Yeelink sensor client power switch example
 */

#include <SPI.h>
#include <Ethernet.h>
#include <Wire.h>
#include <math.h>

byte buff[2];

// for yeelink api
#define  APIKEY           "dc7d1c98898fa2e4517xxxxxxxxx" // 此处替换为你自己的
API KEY
#define DEVICEID          3933 // 此处替换为你的设备编号
#define SENSORID1          5578 // 此处替换为你的传感器编号


// assign a MAC address for the ethernet controller.
byte mac[] = {
   0x00, 0x1D, 0x72, 0x82, 0x35, 0x9D};
// initialize the library instance:
```

```
EthernetClient client ;
char server[] = "api.yeelink.net";      // name address for yeelink API

unsigned long lastConnectionTime = 0;                // last time you connected to the server,
in milliseconds
boolean lastConnected = false;                       // state of the connection last time
through the main loop
const unsigned long postingInterval = 3*1000; // delay between 2 datapoints, 30s
String returnValue = "";
boolean ResponseBegin = false;

void setup() {
   pinMode(5, OUTPUT);
   Wire.begin();
   // start serial port:
   Serial.begin(57600);

   // start the Ethernet connection with DHCP:
   if (Ethernet.begin(mac) == 0) {
      Serial.println("Failed to configure Ethernet using DHCP");
      for(;;)
         ;
   }
   else {
      Serial.println("Ethernet configuration OK");
   }
}

void loop() {
```

```
// if there's incoming data from the net connection.
// send it out the serial port.    This is for debugging
// purposes only:

if (client.available()) {
    char c = client.read();
    // Serial.print(c);
    if (c == '{')
        ResponseBegin = true;
    else if (c == '}')
        ResponseBegin = false;

    if (ResponseBegin)
        returnValue += c;
}
if (returnValue.length() !=0 && (ResponseBegin == false))
{
    Serial.println(returnValue);

    if (returnValue.charAt(returnValue.length() - 1) == '1') {
        Serial.println("turn on the LED");
        digitalWrite(5, HIGH);
    }
    else if(returnValue.charAt(returnValue.length() - 1) == '0') {
        Serial.println("turn off the LED");
        digitalWrite(5, LOW);
    }
    returnValue = "";
}
```

```
    // if there's no net connection, but there was one last time
    // through the loop, then stop the client:
    if (!client.connected() && lastConnected) {
        Serial.println();
        Serial.println("disconnecting.");
        client.stop();
    }

    // if you're not connected, and ten seconds have passed since
    // your last connection, then connect again and send data:
    if(!client.connected() && (millis() - lastConnectionTime > postingInterval)) {
        // read sensor data, replace with your code
        //int sensorReading = readLightSensor();
        Serial.print("yeelink:");
        //get data from server
        getData();
    }
    // store the state of the connection for next time through
    // the loop:
    lastConnected = client.connected();
}


// this method makes a HTTP connection to the server and get data back
void getData(void) {
    // if there's a successful connection:
    if (client.connect(server, 80)) {
        Serial.println("connecting...");
```

```
    // send the HTTP GET request:

    client.print("GET /v1.0/device/");
    client.print(DEVICEID);
    client.print("/sensor/");
    client.print(SENSORID1);
    client.print("/datapoints");
    client.println(" HTTP/1.1");
    client.println("Host: api.yeelink.net");
    client.print("Accept: *");
    client.print("/");
    client.println("*");
    client.print("U-ApiKey: ");
    client.println(APIKEY);
    client.println("Content-Length: 0");
    client.println("Connection: close");
    client.println();
    Serial.println("print get done.");

}
else {
    // if you couldn't make a connection:
    Serial.println("connection failed");
    Serial.println();
    Serial.println("disconnecting.");
    client.stop();
}
// note the time that the connection was made or attempted:
lastConnectionTime = millis();
```

```
}
```

远程显示温度程序

```
#include <Ethernet.h>
#include <WiFi.h>
#include <SPI.h>
#include <yl_data_point.h>
#include <yl_device.h>
#include <yl_w5100_client.h>
#include <yl_wifi_client.h>
#include <yl_messenger.h>
#include <yl_sensor.h>
#include <yl_value_data_point.h>
#include <yl_sensor.h>

//this example reads data from a lm35dz sensor, convert value to degree Celsius
//and then post it to yeelink.net

//replace 2633 3539 with ur device id and sensor id
yl_device ardu(4136);   //此处替换为你的设备编号
yl_sensor therm(5895, &ardu);//此处替换为你的传感器编号
//replace first param value with ur u-apikey
yl_w5100_client client;
yl_messenger messenger(&client, "dc7d1c98898fa2e45xxxxxxx", "api.yeelink.net");     //
此处替换为你自己的 API KEY

const int THERM_PIN    = A0;
```

```
float lm35_convertor(int analog_num)
{
    return analog_num * (5.0 / 1024.0 * 100);
}

void setup()
{
    Serial.begin(9600);              //for output information
    byte mac[] = {
       0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xAA};
    Ethernet.begin(mac);
}

void loop()
{
    int v = analogRead(THERM_PIN);
    Serial.println(lm35_convertor(v));
    yl_value_data_point dp(lm35_convertor(v));
    therm.single_post(messenger, dp);
    delay(1000);
}
```

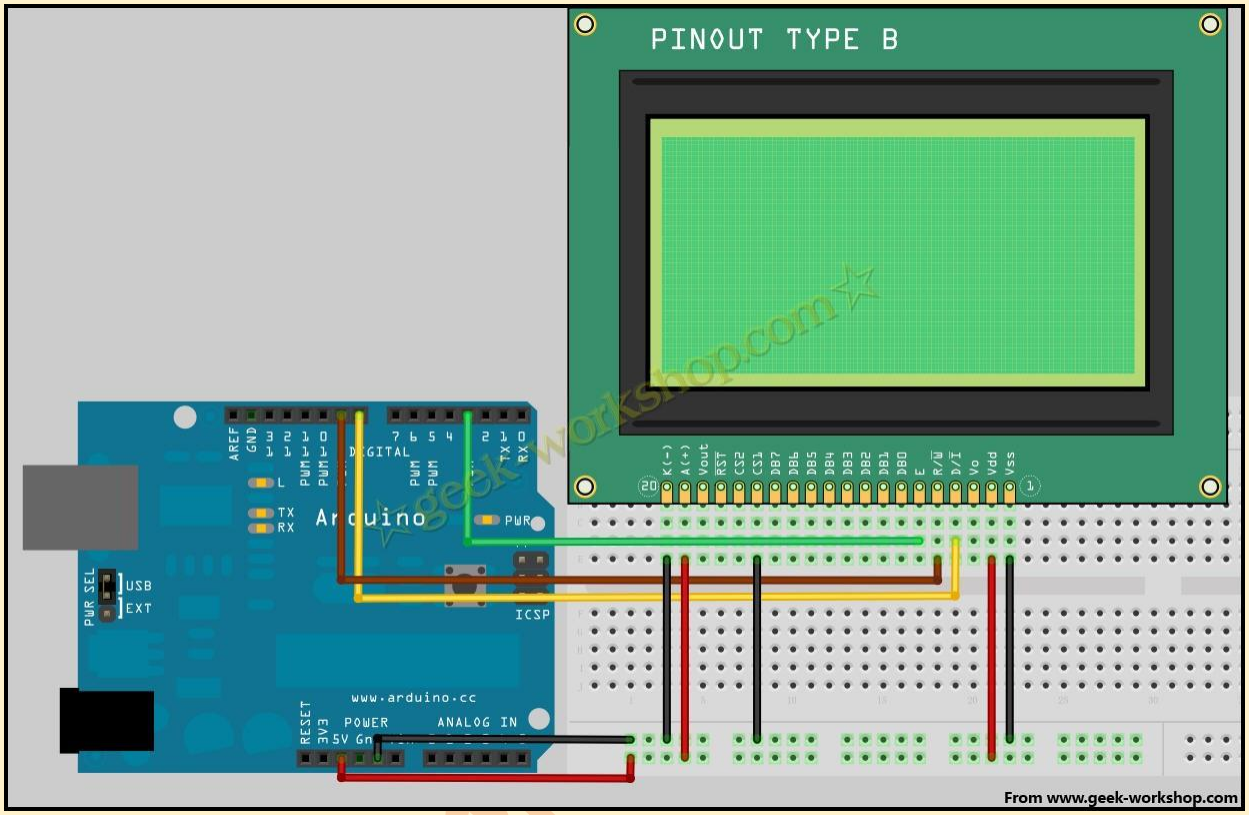
7.4 超声测距装置

元件清单

LCD12864 液晶屏，SR04 超声探头，导线，arduino 及扩展板

电路连接

不同的 12864 管脚说明可能有所差异，此时可以参考其相对位置。



程序代码

```
/*
LCD    Arduino
PIN1 = GND
PIN2 = 5V
RS(CS) = 8;
RW(SID)= 9;
EN(CLK) = 3;
PIN15 PSB = GND;
*/
#include "SR04.h"
```

```
#include "LCD12864RSPI.h"

#define AR_SIZE( a ) sizeof( a ) / sizeof( a[0] )
#define TRIG_PIN 4
#define ECHO_PIN 5

unsigned char show1[]="distance:";
SR04 sr04 = SR04(ECHO_PIN,TRIG_PIN);

void setup()
{
LCDA.Initialise(); //  屏幕初始化
delay(100);
Serial.begin(9600);
}

void loop()
{
delay(1000);
   LCDA.CLEAR();//清屏
   int a=sr04.Distance();

   char b[20];
   unsigned char* c;

   String myString="";
   myString.concat(a); //用 concat 合并
   myString.toCharArray(b,20);
   c=(unsigned char*)b;
```

```
  LCDA.DisplayString(1,1,show1,AR_SIZE(show1));//第三行第二格开始，显示文字 geek-workshop
  LCDA.DisplayString(2,1,c,4);

  }
```

附录 1 基础电子元件及仪表使用

元件检测

数字万用表是电工、电子中常用的仪表，用电池驱动、液晶显示，可用来测量交直流电压、电流、电阻、电容，测试线路通断、二极管三极管参数等，功能很强大。在使用时需注意以下几点：

（1）测量前，要检查表笔是否可靠接触，是否正确连接、是否绝缘良好等，以避免电击。

（2）测量时，请勿输入超过规定的极限值，以防电击和损坏仪表。

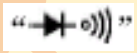
（3）在测量高于 60V 直流、40V 交流电压时，应小心谨慎，防止触电。

（4）选择正确的功能，谨防误操作！换功能时，表笔要离开测试点。

（5）不允许表笔插在电流端子测量电压！

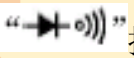
（6）显示为“1　　”代表超出量程。

下面我们结合具体元件分别讲解万用表的使用。

测量项目	红表笔位置	黑表笔位置	功能开关/档位	读取结果
导线	VΩ	COM	“ ” 	两个表笔接导线两端，导线完好则应发声。注意改变导线形状复测
电阻	VΩ	COM	“Ω”档，选择量程	两表笔分别接被测电阻的一个管脚，可以读出电阻数值，单位为Ω，KΩ或MΩ（依量程而定，如 200KΩ档位的单位是KΩ）。“1　　”表明已超过量程
二极管	VΩ	COM	“ ” 	红黑表笔分别接一根管脚。如果红笔接正黑笔接负，显示正向压降；如果反之则显示“1　　”。如正反接都无显示则说明损坏。
电压	VΩ	COM	“V⎓”选择合适量程	将两个表笔接触到试点，红黑表笔之间的电压与极性（负数代表黑笔电压大于红笔）显示在屏幕上

导线

导线一般由铜或铝制成，也有用银线所制（导电性好），表面常有绝缘层。

导线常见的问题是内芯损坏导致断路，因此在使用前必须用万用表检测。通断检测的方法：将黑表笔插入“COM”插孔，红表笔插入“VΩ”插孔，将功能开关转至“  ”挡。将表笔连接到待测线路的两点，如果电阻值很低（导线完好），则内置蜂鸣器发声。测试时可以稍微改变导线形状多次测试，以避免漏查接触不良的导线。

注意：根据我们的经验，坏导线是项目失败的主要原因之一。检查导线必不可少。

电阻

（1）电阻的功能

功能 1：分压，限流，将电能转化为热能。这些功能与电阻本身的参数密切相关。相应的原理大家已经非常清楚了。

功能 2：传感器，包含光敏电阻、热敏电阻等。光敏电阻是一种半导体器件，有光照时电阻很小，无光照时电阻很大。热敏电阻是对温度敏感的半导体元件，主要特征是随着外界环境温度的变化，其阻值会相应发生较大改变。包括正温度系数热敏电阻（PTC）和负温度系数热敏电阻（NTC）。

用万用表测量电阻的步骤：将黑表笔插入“COM”插孔，红表笔插入“VΩ”插孔。将功能开关转至“Ω”档，选择量程，如果事先对被测电阻范围没有概念，应将开关调至最高的档位。然后将两表笔跨接在被测电阻上读出数值。单位为Ω，KΩ或 MΩ（依量程而定，如 200KΩ档位的单位是 KΩ）。

注意：如 LCD 显示：1　　表明已超过量程范围，须将量程开关转至高一档，当测量电阻超过 1MΩ以上时，读数需几秒时间才能稳定。禁止带电操作。电阻测量注意正确使用Ω，KΩ，MΩ单位，前面

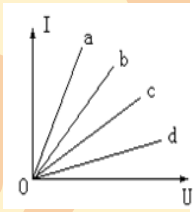
的数字应在[1,1000)范围内。避免出现 15000Ω　0.015MΩ 这样的情况。（应为 15KΩ）。电阻阻值不是随意的，在设计电路时必须根据表格寻找合适的电阻。

电阻的标称阻值系列

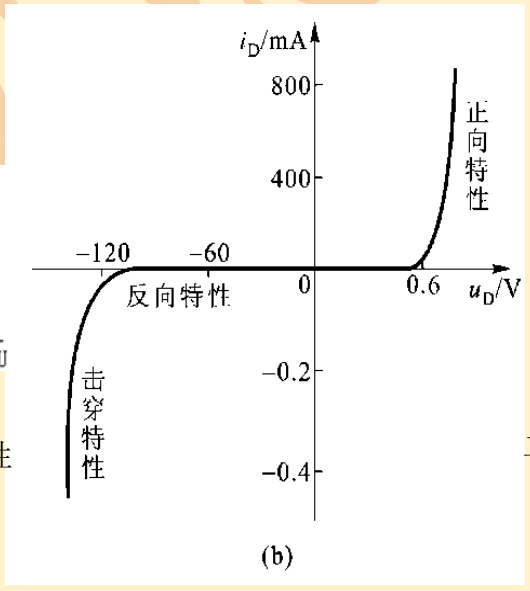
阻值系列	允许误差	偏差等级	电阻标称值											
E24	± 5%	I	1.0	1.1	1.2	1.3	1.5	1.6	1.8	2.0	2.2	2.4	2.7	3.0
			3.3	3.6	3.9	4.3	4.7	5.1	5.6	6.2	6.8	7.5	8.2	9.1
E12	± 10%	II	1.0		1.2		1.5		1.8		2.2		2.7	
			3.3		3.9		4.7		5.6		6.8		8.2	
E6	±20%	III	1.0				1.5				2.2			
			3.3		3.9		4.7		5.6		6.8		8.2	

二极管

二极管是有 2 个管脚的半导体元件。常见的种类有普通二极管和发光二极管(LED)。



电阻的伏安特性



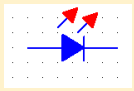
二极管伏安特性

观察二极管伏安特性曲线，并和电阻伏安特性比较。

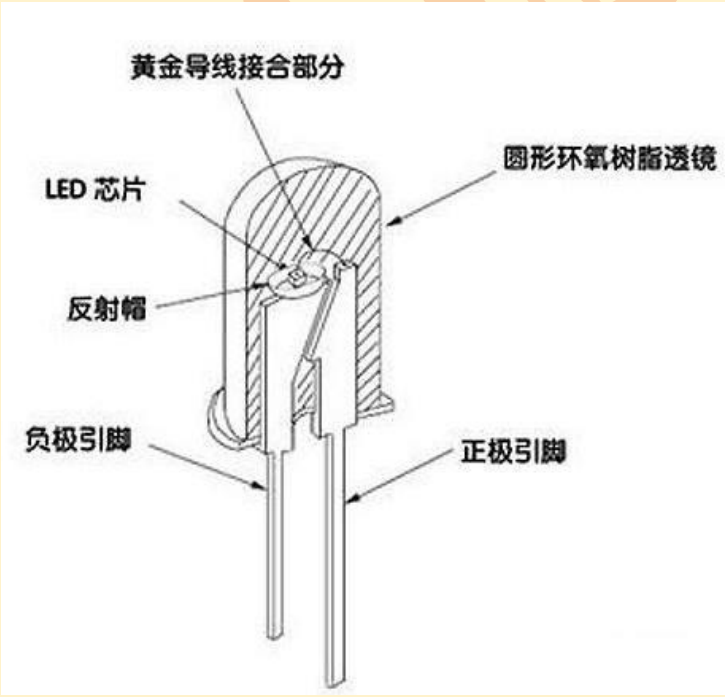
二极管在正向电压（硅管 0.6V-0.7V，发光二极管在 1.7-3.6V）作用下电阻很小，处于导通状态，相当于一只接通的开关，正常工作时两端电压基本恒定，此电压称为正向压降

（硅管为 0.7V，发光二极管根据类型不同，1.7V-3.6V 不等）。在反向电压作用下，电阻很大，处于截止状态，如同一只断开的开关。

思考：根据伏安特性曲线， LED 能否直接接在 5V 电源上？ LED 反接能否发光?

二极管类型	发光二极管
符号	
实物	
极性	长腿为正，短腿为负
正向压降	1.7V-3.6V 不等，取决于颜色和类型
用途	指示、照明等

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：



二极管的万用表测试方法：将黑表笔插入“COM”插孔，红表笔插入“VΩ”插孔，将

功能开关转至  挡。测量分为两步：1.正向测量：红表笔接被测二极管正极，黑表笔负极，即显示二极管正向压降；2.反向测量：将红表笔接到被测二极管负极，黑表笔正极，显示“1　　”。如果测试结果与此不符，说明二极管是坏的。

拓展

晶体二极管，简称二极管（diode）；它只往一个方向传送电流的电子零件。它是一种具有 1 个零件号接合的 2 个端子的器件，具有按照外加电压的方向，使电流流动或不流动的性质。晶体二极管为一个由 p 型半导体和 n 型半导体形成的 PN 结。

P 型半导体（P 指 positive，带正电的）：由单晶硅通过特殊工艺掺入少量的三价元素组成，会在半导体内部形成带正电的空穴；N 型半导体（N 指 negative，带负电的）：由单晶硅通过特殊工艺掺入少量的五价元素组成，会在半导体内部形成带负电的自由电子。

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；常简写为 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。发光二极管其核心是 PN 结。因此它同样具有一般 P-N 结的 I-N 特性，即正向导通，反向截止、击穿特性。

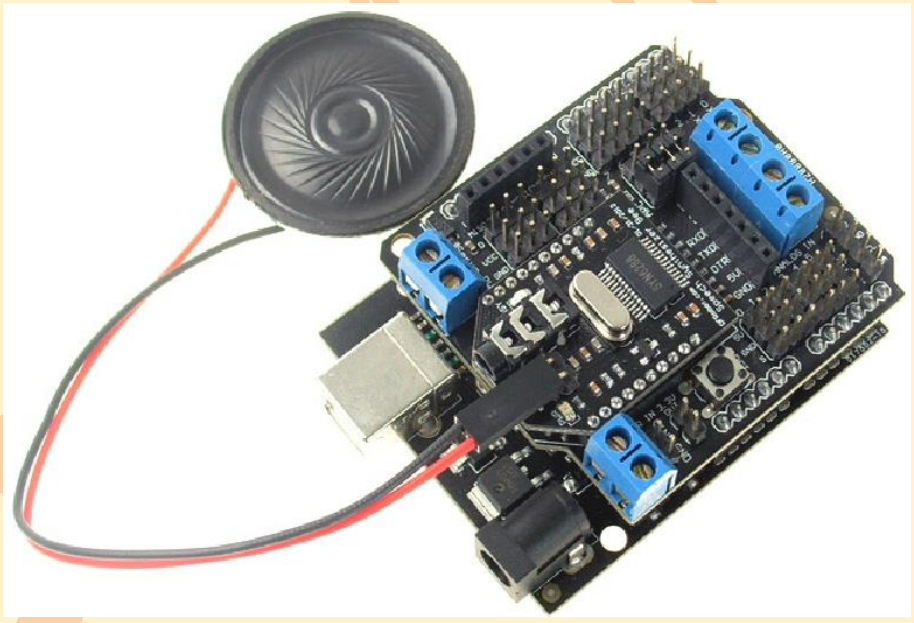
电压测量

将黑表笔插入“COM”插孔，红表笔插入“VΩ”插孔，将功能开关转至“V⎓”档，选择合适量程；将测试表笔接触到试点，红表笔所接的该点电压与极性（负数代表黑笔电压大于红笔）显示在屏幕上。

附录 2 语音合成

元件清单及电路连接

Arduino UNO（不带面包板），传感器扩展板，语音合成模块，喇叭



注意语音合成模块先不要插上，先写入程序再连接它，不要接反。

产品特性

支持 GB2312 、 GBK 、 BIG5 和 UNICODE 内码格式的文本；

- 清晰、自然、准确的中文语音合成效果；可合成任意的中文文本，支持英文字母的合成；
- 具有智能的文本分析处理算法，可正确识别数值、号码、时间日期及常用的度量衡符号；
- 具备很强的多音字处理和中文姓氏处理能力；
- 支持多种文本控制标记，提升文本处理的正确率；
- 每次合成的文本量最多可达 200 字节；
- 支持多种控制命令，包括：合成、停止、暂停合成、继续合成、改变波特率等；
- 支持休眠功能，在休眠状态下可降低功耗；支持多种方式查询芯片工作状态；
- 支持串行数据通讯接口，支持三种通讯波特率：9600bps ， 19200bps 、38400bps ；
- 支持 16 级音量调整；播放文本的前景音量和播放背景音乐的背景音量可分开控制；
- 可通过发送控制标记调节词语语速，支持 6 级词语语速调整；
- 芯片内固化有多首和弦音乐、提示音效和针对某些行业领域的常见语音提示音；
- 内部集成 19 首声音提示音， 23 首和弦提示音， 15 首背景音乐；
- 最终产品提供 SSOP 贴片封装形式；体积业内最小；
- 芯片各项指标均满足室外严酷环境下的应用；

程序代码

```
#include "Syn6288.h"
Syn6288 syn;

uint8_t text1[]={0xbb,0xb6,0xd3,0xad,0xc0,0xb4,
0xb5,0xbd,0xb1,0xb1,0xca,0xa6,0xb4,0xf3,
0xb6,0xfe,0xb8,0xbd,0xd6,0xd0,0xb8,0xdf,0xd6,0xd0,0xb2,0xbf,};

void setup()
```

```
{
   Serial.begin(9600);
}
void loop()
{
   syn.play(text1,sizeof(text1),1);//合成 text1，背景音乐 1
}
```

程序功能

语音: 欢迎来到北师大二附中高中部。

计算机中汉字的表示也是用二进制编码，同样是人为编码的。根据应用目的的不同，汉字编码分为外码、交换码、机内码等。

1.外码（输入码）

外码也叫输入码，是用来将汉字输入到计算机中的一组键盘符号。常用的输入码有拼音码、五笔字型码、自然码、表形码、认知码、区位码和电报码等，一种好的编码应有编码规则简单、易学好记、操作方便、重码率低、输入速度快等优点，每个人可根据自己的需要进行选择。在后面的章节中，重点介绍智能全拼输入法和五笔字型输入法。

2.交换码(国标码)

计算机内部处理的信息，都是用二进制代码表示的，汉字也不例外。而二进制代码使用起来是不方便的，于是需要采用信息交换码。中国标准总局 1981 年制定了中华人民共和国国家标准 GB2312--80《信息交换用汉字编码字符集--基本集》，即国标码。

区位码是国标码的另一种表现形式，把国标 GB2312--80 中的汉字、图形符号组成一个 94×94 的方阵，分为 94 个“区”，每区包含 94 个“位”，其中“区”的序号由 01 至 94，“位”的序号也是从 01 至 94。94 个区中位置总数=94×94=8836 个，其中 7445 个汉字和图形字符中的每一个占一个位置后，还剩下 1391 个空位，这 1391 个位置空下来保留备用。

3.机内码

根据国标码的规定，每一个汉字都有了确定的二进制代码，在微机内部汉字代码都用机内码，在磁盘上记录汉字代码也使用机内码。

程序中的十六进制代码就是机内码，这里为了方便常用十六进制代替二进制代码。