arduino 学习笔记

arduino 学习笔记 1 - 什么是 arduino?

要了解 arduino 就先要了解什么是单片机，arduino 平台的基础就是 AVR 指令集的单片机。

1、什么是单片机？它与个人计算机有什么不同？

一台能够工作的计算机要有这样几个部份构成：中央处理单元 CPU（进行运算、控制）、随机存储器 RAM（数据存储）、存储器 ROM（程序存储）、输入/输出设备 I/O（串行口、并行输出口等）。在个人计算机（PC）上这些部份被分成若干块芯片，安装在一个被称之为主板的印刷线路板上。而在单片机中，这些部份全部被做到一块集成电路芯片中了，所以就称为单片（单芯片）机，而且有一些单片机中除了上述部份外，还集成了其它部份如模拟量/数字量转换（A/D）和数字量/模拟量转换（D/A）等。

2、单片机有什么用？

实际工作中并不是任何需要计算机的场合都要求计算机有很高的性能，一个控制电冰箱温度的计算机难道要用酷睿处理器吗？应用的关键是看是否够用，是否有很好的性能价格比。如果一台冰箱都需要用酷睿处理起来进行温度控制，那价格就是天价了。

慧净电子---ARDUINO 模块化创新视频教程

单片机通常用于工业生产的控制、生活中与程序和控制有关（如：电子琴、冰箱、智能空调等）的场合。

下图就是一个 Atmega328P-PU 单片机，基于 AVR 指令集的 8 位处理器，频率 20MHz，存储器空间 32KB。

什么是 Arduino？

Arduino 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 板你可以选择自己去手动组装或是购买已经组装好的；Arduino 开源的 IDE

慧净电子---ARDUINO 模块化创新视频教程

可以免费下载得到。

Arduino 的编程语言就像似在对一个类似于物理的计算平台进行相应的连线，它基于处理多媒体的编程环境。

为什么要使用 Arduino？

有很多的单片机和单片机平台都适合用做交互式系统的设计。例如：Parallax Basic Stamp， Netmedia's BX-24， Phidgets， MIT's Handyboard 和其它等等提供类似功能的。 所有这些工具，你都不需要去关心单片机编程繁琐的细节，提供给你的是一套容易使用的工具包。 Arduino 同样也简化了同单片机工作的流程，但同其它系统相比 Arduino 在很多地方更具有优越性，特别适合老师，学生和一些业余爱好者们使用：

- 便宜 — 和其它平台相比，Arduino 板算是相当便宜了。最便宜的 Arduino 版本可以自己动手制作，即使是组装好的成品，其价格也不会超过 200 元。

- 跨平台 — Arduino 软件可以运行在 Windows，Macintosh OSX，和 Linux 操作系统。大部分其它的单片机系统都只能运行

在 Windows 上。

- 简易的编程环境 — 初学者很容易就能学会使用 Arduino 编程环境，同时它又能为高级用户提供足够多的高级应用。对于老师们来说，一般都能很方便的使用 Processing 编程环境，所以如果学生学习过使用 Processing 编程环境的话，那他们在使用 Arduino 开发环境的时候就会觉得很相似很熟悉。

- 软件开源并可扩展 — Arduino 软件是开源的，对于有经验的程序员可以对其进行扩展。Arduino 编程语言可以通过 C++库进行扩展，如果有人想去了解技术上的细节，可以跳过 Arduino 语言而直接使用 AVR C 编程语言（因为 Arduino 语言实际上是基于 AVR C 的）。类似的，如果你需要的话，你也可以直接往你的 Arduino 程序中添加 AVR-C 代码。

- 硬件开源并可扩展 — Arduino 板基于 Atmel 的 ATMEGA8 和 ATMEGA168/328 单片机。Arduino 基于 Creative Commons 许可协议，所以有经验的电路设计师能够根据需求设计自己的模块，可以对其扩展或改进。甚至是对于一些相对没有什么经验的用户，也可以通过制作试验板来理解 Arduino 是怎么工作的，省钱又省事。

慧净电子---ARDUINO 模块化创新视频教程

Arduino 基于 AVR 平台，对 AVR 库进行了二次编译封装，把端口都打包好了，寄存器啦、地址指针之类的基本不用管。大大降低了软件开发难度，适宜非专业爱好者使用。优点和缺点并存，因为是二次编译封装，代码不如直接使用 AVR 代码编写精练，代码执行效率与代码体积都弱于 AVR 直接编译。

性能：

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 ICSP 下载，支持 TX/RX。

输入电压：USB 接口供电或者 5V-12V 外部电源供电。

输出电压：支持 3.3V 级 5V DC 输出。

处理器：使用 Atmel Atmega168 328 处理器，因其支持者众多，已有公司开发出来 32 位的 MCU 平台支持 arduino。

目前 arduino 的控制板最新的为 Arduino Uno，如下图：

慧净电子---ARDUINO 模块化创新视频教程

国内使用比较多的为 Arduino Duemilanove 2009，主要原因是 Uno 的 usb 控制芯片封装方式改变，制造成本上升，其他变化不大，性价比还是 Arduino Duemilanove 2009 比较好。

因其开源特性，生产 arduino 控制板的厂商众多，同样的 Duemilanove 2009 就有很多种颜色。

对于一些对电路板大小要求比较严格的地方，arduino 团队提供了 arduino Nano，此板体积做的非常小。如下图：

慧净电子---ARDUINO 模块化创新视频教程

arduino 板子上基本端口如图描述，对几个比较特殊的端口下面详细说明下：

VIN 端口：VIN 是 input voltage 的缩写，表示有外部电源时的输入端口。

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

ICSP：也有称为 ISP（In System Programmer)，就是一种线上即时烧录，目前比较新的芯片都支持这种烧录模式，包括大家常听说的 8051 系列的芯片，也都慢慢采用这种简便的烧录方式。我们都知道传统的烧录方式，都是将被烧录的芯片，从线路板上拔起，有的焊死在线路板上的芯片，还得先把芯片焊接下来才能烧录。为了解决这

种问题，发明了 ICSP 线上即时烧录方式。只需要准备一条 R232 线（连接烧录器），以及一条连接烧录器与烧录芯片针脚的连接线就可以。电源的+5V，GND，两条负责传输烧录信息的针脚，再加上一个烧录电压针脚，这样就可以烧录了。

arduino 学习笔记 2

通过 Arduino 编译器查看串口数据

最简单的例子：

```
void setup()
{
  Serial.begin(9600); //  打开串口，设置波特率为 9600 bps
}
void loop()
{
      int val;
      val=analogRead(5);//传感器接到模拟口 5，数值根据自己的需要可变
      Serial.println(val,DEC);//从串口发送字符串并换行
```

```
    delay(100);
}
```

如果电路安装正确，按照示例代码运行、上传，然后点击编译器的 Serial Monitor 按钮，就可以看到从代码定义的输入口（这儿是模拟口 5）获取的数据了。

arduino 学习笔记 3 arduino 语言

- Arduino 语言是建立在 C/C++基础上的，其实也就是基础的 C 语言，Arduino 语言只不过把 AVR 单片机（微控制器）相关的一些参数设置都函数化，不用我们去了解他的底层，让我们不了解 AVR 单片机（微控制器）的朋友也能轻松上手。

  在与 Arduino DIYER 接触的这段时间里，发现有些朋友对 Arduino 语言还是比较难入手，那么这里我就简单的注释一下 Arduino 语言（本人也是半罐子水，有错的地方还请各位指正）。

  /*************基础 C 语言*************/

  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**
- **byte** 字节类型
- **int**
- **unsigned int**
- **long**

- **unsigned long**
- **float**
- **double**
- **string**
- **array**
- **void**

数据类型转换：

- **char()**
- **byte()**
- **int()**
- **long()**
- **float()**

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 语言的关键字和符号，有 c 语言基础的都应该了解其含义，这里也不作过多的解释。

/*************Arduino 语言*************/

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。

扩展 I/O

- **shiftOut**(dataPin, clockPin, bitOrder, value)　SPI 外部 IO 扩展函数，通常使用带 SPI 接口的 74HC595 做 8 个 IO 扩展，dataPin 为数据口，clockPin 为时钟口，bitOrder 为数据传输方向（**MSBFIRST** 高位在前，**LSBFIRST** 低位在前），value 表示所要传送的数据（0～255），另外还需要一个 IO 口做 74HC595 的使能控制。
- unsigned long **pulseIn**(pin, value)　脉冲长度记录函数，返回时间参数（us），pin 表示为 0～13，value 为 HIGH 或 LOW。比如 value 为 HIGH，那么当 pin 输入为高电平时，开始计时，当 pin 输入为低电平时，停止计时，然后返回该时间。

时间函数

- unsigned long **millis**()　返回时间函数（单位 ms），该函数是指，当程序运行就开始计时并返回记录的参数，该参数溢出大概需要 50 天时间。
- **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

三角函数

- **sin**(rad)
- **cos**(rad)

- **tan**(rad)

随机数函数

- **randomSeed**(seed)　随机数端口定义函数，seed 表示读模拟口 analogRead(pin)函数　。
- long **random**(max)　随机数函数，返回数据大于等于 0，小于 max。
- long **random**(min, max)　随机数函数，返回数据大于等于 min，小于 max。

外部中断函数

- **attachInterrupt**(interrupt, , mode)　外部中断只能用到数字 IO 口 2 和 3，interrupt 表示中断口初始 0 或 1，表示一个功能函数，mode：**LOW** 低电平中断，**CHANGE** 有变化就中断，**RISING** 上升沿中断，**FALLING** 下降沿中断。
- **detachInterrupt**(interrupt)　中断开关，interrupt=1 开，interrupt=0 关。

中断使能函数

- **interrupts**() 使能中断
- **noInterrupts**() 禁止中断

串口收发函数

- **Serial.begin**(speed) 串口定义波特率函数，speed 表示波特率，如 9600，19200 等。
- int **Serial.available**() 判断缓冲器状态。
- int **Serial.read**() 读串口并返回收到参数。
- **Serial.flush**() 清空缓冲器。
- **Serial.print**(data) 串口输出数据。
- **Serial.println**(data) 串口输出数据并带回车符。

/**********/

/**********Arduino 语言库文件**********/

官方库文件

- **EEPROM** - EEPROM 读写程序库
- **Ethernet** - 以太网控制器程序库
- **LiquidCrystal** - LCD 控制程序库
- **Servo** - 舵机控制程序库
- **SoftwareSerial** - 任何数字 IO 口模拟串口程序库
- **Stepper** - 步进电机控制程序库
- **Wire** - TWI/I2C 总线程序库
- **Matrix** - LED 矩阵控制程序库
- **Sprite** - LED 矩阵图象处理控制程序库

慧净电子---ARDUINO 模块化创新视频教程

非官方库文件

- **DateTime** - a library for keeping track of the current date and time in software.
- **Debounce** - for reading noisy digital inputs (e.g. from buttons)
- **Firmata** - for communicating with applications on the computer using a standard serial protocol.
- **GLCD** - graphics routines for LCD based on the KS0108 or equivalent chipset.
- **LCD** - control LCDs (using 8 data lines)
- **LCD 4 Bit** - control LCDs (using 4 data lines)
- **LedControl** - for controlling LED matrices or seven-segment displays with a MAX7221 or MAX7219.
- **LedControl** - an alternative to the Matrix library for driving multiple LEDs with Maxim chips.
- **Messenger** - for processing text-based messages from the computer
- **Metro** - help you time actions at regular intervals
- **MsTimer2** - uses the timer 2 interrupt to trigger an action every N milliseconds.

- **OneWire** - control devices (from Dallas Semiconductor) that use the One Wire protocol.
- **PS2Keyboard** - read characters from a PS2 keyboard.
- **Servo** - provides software support for Servo motors on any pins.
- **Servotimer1** - provides hardware support for Servo motors on pins 9 and 10
- **Simple Message System** - send messages between Arduino and the computer
- **SSerial2Mobile** - send text messages or emails using a cell phone (via AT commands over software serial)
- **TextString** - handle strings
- **TLC5940** - 16 channel 12 bit PWM controller.
- **X10** - Sending X10 signals over AC power lines

/***/

arduino 学习笔记 4　数据类型

有多种类型的变量，如下所述

boolean　布尔

char　　字符

byte　　字节

int 整数

unsigned int 无符号整数

long 长整数

unsigned long 无符号长整数

float 浮点

double 双字节浮点

string 字符串

array 数组

arduino 学习笔记 5 Arduuino 复合运算符

+= , -= , *= , /=

Description 描述

Perform a mathematical operation on a variable with another constant or variable. The += (et al) operators are just a convenient shorthand for the expanded syntax, listed below.

对一个变量和另一个参数或变量完成一个数学运算。+=（以及其他）可以缩短语法长度。

Syntax 语法

x += y; // equivalent to the expression x = x + y; // 等价于 x = x + y;

x -= y; // equivalent to the expression x = x - y; // 等价于 x = x - y;

x *= y; // equivalent to the expression x = x * y; // 等价于 x = x * y;

x /= y; // equivalent to the expression x = x / y; // 等价于 x = x / y;

Parameters 参数

x: any variable type

x：任何变量类型

y: any variable type or constant

y：任何变量类型或常数

Examples 范例

```
x = 2;
x += 4;      // x now contains 6          // x 现在为 6
x -= 3;      // x now contains 3          // x 现在为 3
x *= 10;     // x now contains 30         // x 现在为 30
x /= 2;      // x now contains 15         // x 现在为 15
```

Syntax 语法

x++; // increment x by one and returns the old value of x

// 将 x 的值加 1 并返回原来的 x 的值。 ++x; // increment x by one and returns the new value of x // 将 x 的值加 1 并返回现在的 x 的值。

x-- ; // decrement x by one and returns the old value of x // 将 x 的值减 1 并返回原来的 x 的值。

--x ; // decrement x by one and returns the new value of x // 将 x 的值减 1 并返回现在的 x 的值。

Parameters 参数

x: an integer or long (possibly unsigned)

x：一个整数或长整数（可以无符号）

Returns 返回

The original or newly incremented / decremented value of the variable.

返回变量原始值或增加/消耗后的新值。

Examples 范例

```
x = 2;
y = ++x;      // x now contains 3, y contains 3          // x 现在为 3，y 为 3
y = x--;      // x contains 2 again, y still contains 3          // x 现在仍然为 2，y 将为 3
```

arduino 学习笔记 6 Arduino 基础

在学语言之间，还要做的一个功课就是要明白程序的构架，这个也同样简单，大体可分为几个部分。

1、声明变量及接口名称（int val;int ledPin=13;）。

2、setup()——函数在程序开始时使用，可以初始化变量、接口模式、启用库等（例如：pinMode(ledPin,OUTUPT);）。

3、loop（）——在 setup()函数之后，即初始化之后，loop() 让你的程序循环地被执行。使用它来运转 Arduino。

接下来就开始学习一下几个基本函数。

1、pinMode(接口名称,OUTPUT 或 INPUT)将——接口定义为输入或输出接口，用在 setup()函数里。

2、digitalWrite(接口名称, HIGH 或 LOW)——将数字接口值至高或低。

3、digitalRead（接口名称）——读出数字接口的值。

4、analogWrite(接口名称，数值)——给一个接口写入模拟值（PWM 波）。对于 ATmega168 芯片的 Arduino（包括 Mini 或 BT），该函数可以工作于 3, 5, 6, 9, 10 和 11 号接口。老的 ATmega8 芯片的 USB 和 serial Arduino 仅仅支持 9, 10 和 11 号接口。

5、analogRead(接口名称)——从指定的模拟接口读取值，Arduino 对该模拟值进行 10-bit 的数字转换，这个方法将输入的 0-5 电压值转换为 0 到 1023 间的整数值。

6、delay()——延时一段时间，delay(1000)为一秒。

7、Serial.begin(波特率)——设置串行每秒传输数据的速率（波特率）。在同计算机通讯时，使用下面这些值：300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 或 115200。你也可以在任何时候使用其它的值，比如，与 0 号或 1 号插口通信就要求特殊的波特率。用在 setup()函数里

8、Serial.read()——读取持续输入的数据。

9、Serial.print(数据，数据的进制)——从串行端口输出数据。Serial.print(数据)默认为十进制等于 Serial.print(数据，DEC)。

10、Serial.println(数据，数据的进制)——从串行端口输出数据，跟随一个回车和一个换行符。这个函数所取得的值与 Serial.print()一样。

以上几个函数是常用基本函数，还有很多以后会慢慢学习

arduino 学习笔记 7 函数

输入输出函数

Arduino 内含了一些处理输出与输入的切换功能，相信已经从书中程式范例略知一二。

pinMode(pin, mode)

将数位脚位(digital pin)指定为输入或输出。

范例 ：

pinMode(7,INPUT); // 将脚位 7 设定为输入模式

digitalWrite(pin, value)

将数位脚位指定为开或关。脚位必须先透过 pinMode 明示为输入或输出模式 digitalWrite 才能生效。

范例 ：

```
digitalWrite(8,HIGH); //将脚位 8 设定输出高电位
```

```
int digitalRead(pin)
```

将输入脚位的值读出，当感测到脚位处于高电位时时回传 HIGH，否则回传 LOW。

范例 ：

```
val = digitalRead(7); // 读出脚位 7 的值并指定给 val
```

```
int analogRead(pin)
```

读出类比脚位的电压并回传一个 0 到 1023 的数值表示相对应的 0 到 5 的电压值。

范例 ：

val = analogRead(0); //读出类比脚位 0 的值并指定给 val 变数

analogWrite(pin, value)

改变 PWM 脚位的输出电压值，脚位通常会在 3、5、6、9、10 与 11。Value 变数范围 0-255，例如：输出电压 2.5 伏特（V），该值大约是 128。

范例 :

analogWrite(9,128); // 输出电压约 2.5 伏特（V）

unsigned long pulseIn(pin, value)

设定读取脚位状态的持续时间，例如使用红外线、加速度感测器测得某一项数值时，在时间单位内不会改变状态。

范例 :

time = pulsein(7,HIGH); // 设定脚位 7 的状态在时间单位内保持为 HIGH

shiftOut(dataPin, clockPin, bitOrder, value)

把资料传给用来延伸数位输出的暂存器，函式使用一个脚位表示资料、一个脚位表示时脉。bitOrder 用来表示位元间移动的方式（LSBFIRST 最低有效位元或是 MSBFIRST 最高有效位元），最后 value 会以 byte 形式输出。此函式通常使用在延伸数位的输出。

范例 ：

shiftOut(dataPin, clockPin, LSBFIRST, 255);

时间函数

控制与计算晶片执行期间的时间

unsigned long millis()

回传晶片开始执行到目前的毫秒

范例:

duration = millis()-lastTime; // 表示自"lastTime"至当下的时间

delay(ms)

暂停晶片执行多少毫秒

范例:

delay(500); //暂停半秒（500 毫秒）

delay Microseconds(us)

暂停晶片执行多少微秒

范例:

delayMicroseconds(1000); //暂停 1 豪秒

数学函式

三角函数以及基本的数学运算

min(x, y)

回传两数之间较小者

范例：

```
val = min(10,20); // 回传 10
```

max(x, y)

回传两数之间较大者

范例：

```
val = max(10,20); // 回传 20
```

abs(x)

回传该数的绝对值，可以将负数转正数。

范例：

```
val = abs(-5); // 回传 5
```

```
constrain(x, a, b)
```

判断 x 变数位于 a 与 b 之间的状态。x 若小于 a 回传 a；介于 a 与 b 之间回传 x 本身；大于 b 回传 b

范例：

```
val = constrain(analogRead(0), 0, 255); // 忽略大于 255 的数
```

```
map(value, fromLow, fromHigh, toLow, toHigh)
```

将 value 变数依照 fromLow 与 fromHigh 范围，对等转换至 toLow 与 toHigh 范围。时常使用于读取类比讯号，转换至程式所需要的范围值。

例如：

```
val = map(analogRead(0),0,1023,100, 200); // 将 analog0 所读取到的讯号对等转换至 100 - 200 之间的数值。
```

```
double pow(base, exponent)
```

回传一个数(base)的指数(exponent)值。

范例：

```
double x = pow(y, 32); // 设定 x 为 y 的 32 次方
```

```
double sqrt(x)
```

回传 double 型态的取平方根值。

范例：

```
double a = sqrt(1138); // 回传 1138 平方根的近似值
33.73425674438
```

```
double sin(rad)
```

回传角度（radians）的三角函数 sine 值。

范例：

```
double sine = sin(2); // 近似值 0.90929737091
```

```
double cos(rad)
```

回传角度（radians）的三角函数 cosine 值。

范例：

```
double cosine = cos(2); //近似值-0.41614685058
```

```
double tan(rad)
```

回传角度（radians）的三角函数 tangent 值。

范例：

```
double tangent = tan(2); //近似值-2.18503975868
```

乱数函式

产生乱数

```
randomSeed(seed)
```

事实上在 Arduino 里的乱数是可以被预知的。所以如果需要一个真正的乱数，可以呼叫此函式重新设定产生乱数种子。你可以使用乱数当作乱数的种子，以确保数字以随机的方式出现，通常会使用类比输入

当作乱数种子，藉此可以产生与环境有关的乱数（例如：无线电波、宇宙雷射线、电话和萤光灯发出的电磁波等）。

范例：

```
randomSeed(analogRead(5)); // 使用类比输入当作乱数种子
```

```
long random(max)
```

```
long random(min, max)
```

回传指定区间的乱数，型态为 long。如果没有指定最小值，预设为 0。

范例：

```
long randnum = random(0, 100); // 回传 0 - 99 之间的数字
```

```
long randnum = random(11);          // 回传 0 -10 之间的数字
```

序列通讯

你可以在第五章看见一些使用序列埠与电脑交换讯息的范例，以下是函式解释。

Serial.begin(speed)

你可以指定Arduino从电脑交换讯息的速率，通常我们使用9600 bps。当然也可以使用其他的速度，但是通常不会超过 115,200 bps（每秒位元组）。

范例：

Serial.begin(9600);

Serial.print(data)

Serial.print(data, encoding)

经序列埠传送资料，提供编码方式的选项。如果没有指定，预设以一般文字传送。

范例：

Serial.print(75);　　// 列印出 "75"

Serial.print(75, DEC); //列印出 "75"

Serial.print(75, HEX); // "4B" (75 的十六进位)

Serial.print(75, OCT); // "113" (75 in 的八进位)

Serial.print(75, BIN); // "1001011" (75 的二进位)

Serial.print(75, BYTE); // "K" (以 byte 进行传送，显示以 ASCII 编码方式)

Serial.println(data)

Serial.println(data, encoding)

与 Serial.print()相同，但会在资料尾端加上换行字元（ ）。意思如同你在键盘上打了一些资料后按下 Enter。

范例：

```
Serial.println(75);          //列印出"75 "

Serial.println(75, DEC); //列印出"75 "

Serial.println(75, HEX); // "4B "

Serial.println(75, OCT); // "113 "

Serial.println(75, BIN); // "1001011 "

Serial.println(75, BYTE); // "K "
```

int Serial.available()

回传有多少位元组（bytes）的资料尚未被 read()函式读取，如果回传值是 0 代表所有序列埠上资料都已经被 read()函式读取。

范例：

```
int count = Serial.available();
```

```
int Serial.read()
```

读取 1byte 的序列资料

范例：

```
int data = Serial.read();
```

```
Serial.flush()
```

有时候因为资料速度太快，超过程式处理资料的速度，你可以使用此函式清除缓冲区内的资料。经过此函式可以确保缓冲区(buffer)内的资料都是最新的。

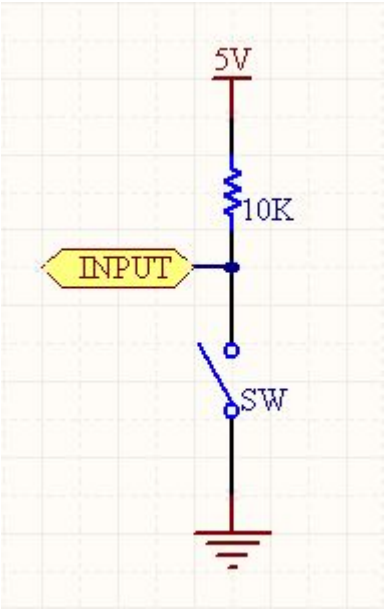
范例：

```
Serial.flush();
```

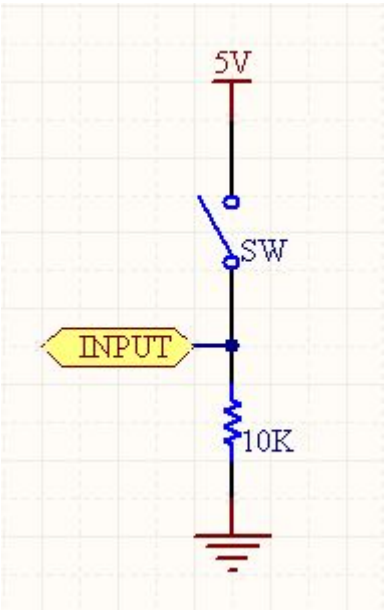
arduino 学习笔记 8 数字输入

在数字电路中开关（switch）是一种基本的输入形式，它的作用是保持电路的连接或者断开。Arduino 从数字 I/O 管脚上只能读出高电平（5V）或者低电平（0V），因此我们首先面临到的一个问题就是如何将开关的开/断状态转变成 Arduino 能够读取的高/低电平。解决的办法是通过上 /下拉电阻，按照电路的不同通常又可以分为正逻辑（Positive Logic）和负逻辑（Inverted Logic）两种。

在正逻辑电路中，开关一端接电源，另一端则通过一个 10K 的下拉电阻接地，输入信号从开关和电阻间引出。当开关断开的时候，输入信号被电阻“拉”向地，形成低电平（0V）；当开关接通的时候，输入信号直接与电源相连，形成高电平。对于经常用到的按压式开关来讲，就是按下为高，抬起为低。

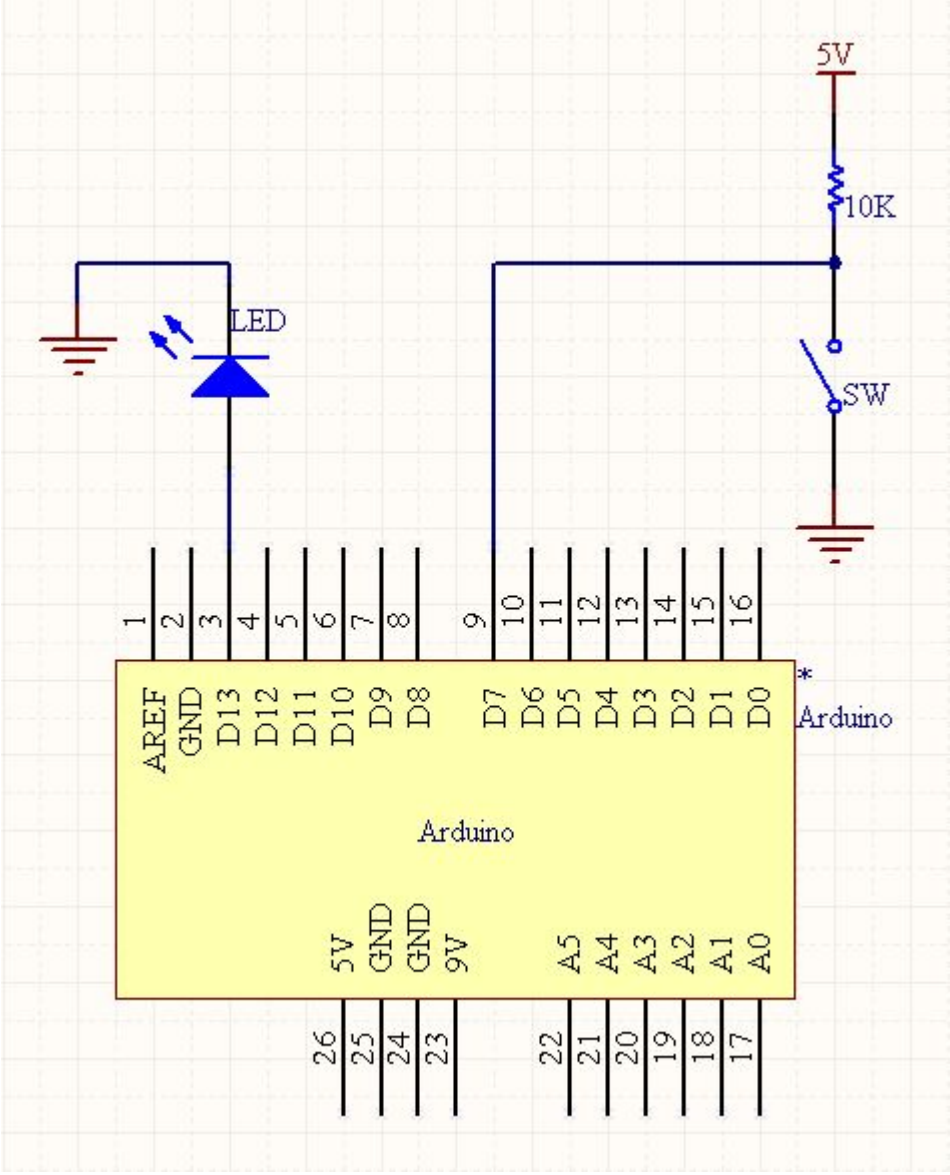


在负逻辑电路中，开关一端接地，另一端则通过一个 10K 的上拉电阻接电源，输入信号同样也是从开关和电阻间引出。当开关断开时，输入信号被电阻“拉”向电源，形成高电平（5V）；当开关接通的时候，输入信号直接与地相连，形成低电平。对于经常用到的按压式开关来讲，就是按下为低，抬起为高。



慧净电子---ARDUINO 模块化创新视频教程

为了验证 Arduino 数字 I/O 的输入功能，我们可以将开关接在 Arduino 的任意一个数字 I/O 管脚上（13 除外），并通过读取它的接通或者断开状态，来控制其它数字 I/O 管脚的高低。本实验采用的原理图如下所示，其中开关接在数字 I/O 的 7 号管脚上，被控的发光二极管接在数字 I/O 的 13 号 管脚上：



相应的代码为：

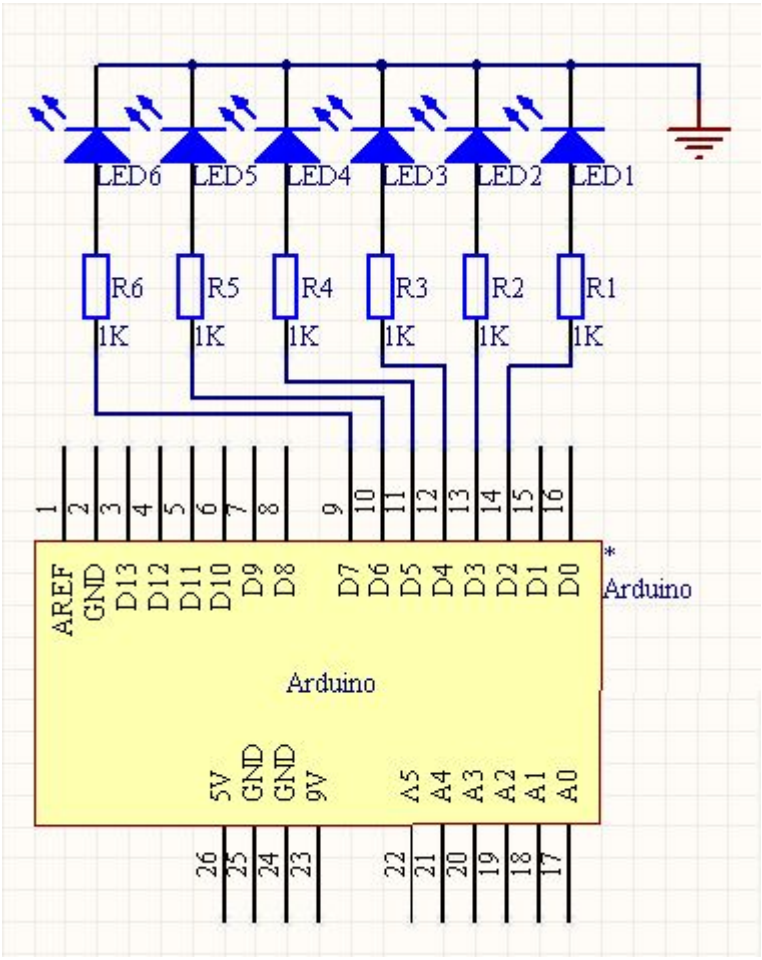
```
int ledPin = 13;
int switchPin = 7;
int value = 0;

void setup() {
pinMode(ledPin, OUTPUT);
pinMode(switchPin, INPUT);
}

void loop() {
value = digitalRead(switchPin);
if (HIGH == value) {
    // turn LED off
    digitalWrite(ledPin, LOW);
} else {
    // turn LED on
    digitalWrite(ledPin, HIGH);
}
}
```

由于采用的是负逻辑电路，开关按下时用 digitalRead()函数读取到的值为 LOW，此时再用 digitalWrite()函数将发光二极管所在的管脚置

为高，点亮发光二极管。同理，当开关抬起时，发光二极管将被熄灭，这样我们就实现了用开关来控制发光二极管的功能。

arduino 学习笔记 9 Arduino 的数字输出

Arduino 的数字 I/O 被分成两个部分，其中每个部分都包含有 6 个可用的 I/O 管脚，即管脚 2 到管脚 7 和管脚 8 到管脚 13。除了管脚 13 上接了一个 1K 的电阻之外，其他各个管脚都直接连接到 ATmega 上。我们可以利用一个 6 位的数字跑马灯，来对 Arduino 数字 I/O 的输出功能进行验证，以下是相应的原理图：



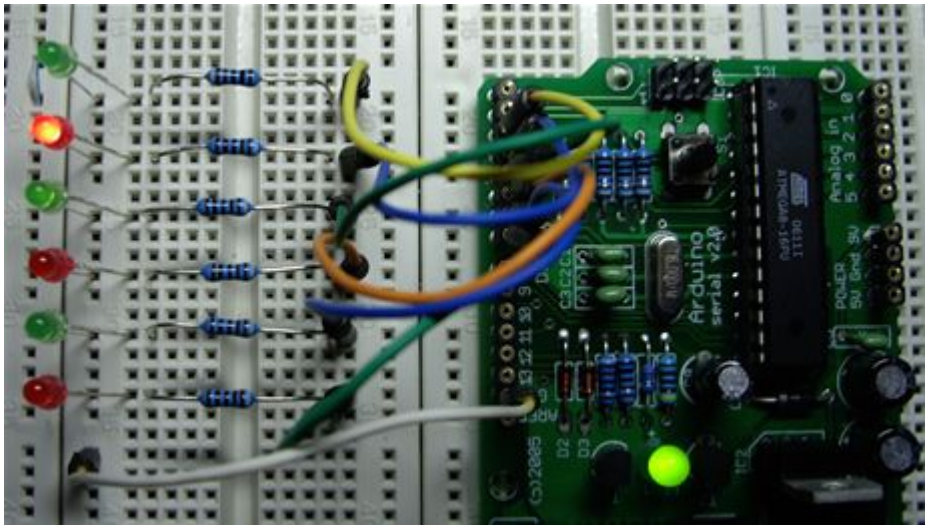
电路中在每个 I/O 管脚上加的那个 1K 电阻被称为限流电阻，由于发光二极管在电路中没有等效电阻值，使用限流电阻可以使元件上通过的电流不至于过大，能够起到保护的作用。

该工程对应的代码为：

```
int BASE = 2;
int NUM = 6;
int index = 0;

void setup()
{
for (int i = BASE; i < BASE + NUM; i ++) {
    pinMode(i, OUTPUT);
}
}

void loop()
{
for (int i = BASE; i < BASE + NUM; i ++) {
    digitalWrite(i, LOW);
}
digitalWrite(BASE + index, HIGH);
```

```
index = (index + 1) % NUM;
delay(100);
}
```

下载并运行该工程，连接在 Arduino 数字 I/O 管脚 2 到管脚 7 上的发光二极管会依次点亮 0.1 秒，然后再熄灭：



这个实验可以用来验证数字 I/O 输出的正确性。Arduino 上一共有十二个数字 I/O 管脚，我们可以用同样的办法验证其他六个管脚的正确性，而这只需要对上述工程的第一行做相应的修改就可以了：

```
int BASE = 8;
```

arduino 学习笔记 10 Arduino 的串口输入

串行通信是在实现在 PC 机与微控制器进行交互的最简单的办法。之前的 PC 机上一般都配有标准的 RS-232 或者 RS-422 接口来实现串行通信，但现在这种情况已经发生了一些改变，大家更倾向于使用

USB 这样一种更快速但同时也更加复杂的方式来实现串行通信。尽管在有些计算机上现在已经找不到 RS-232 或者 RS-422 接口了，但我们仍可以通过 USB/串口或者 PCMCIA/串口这样的转换器， 在这些设备上得到传统的串口。

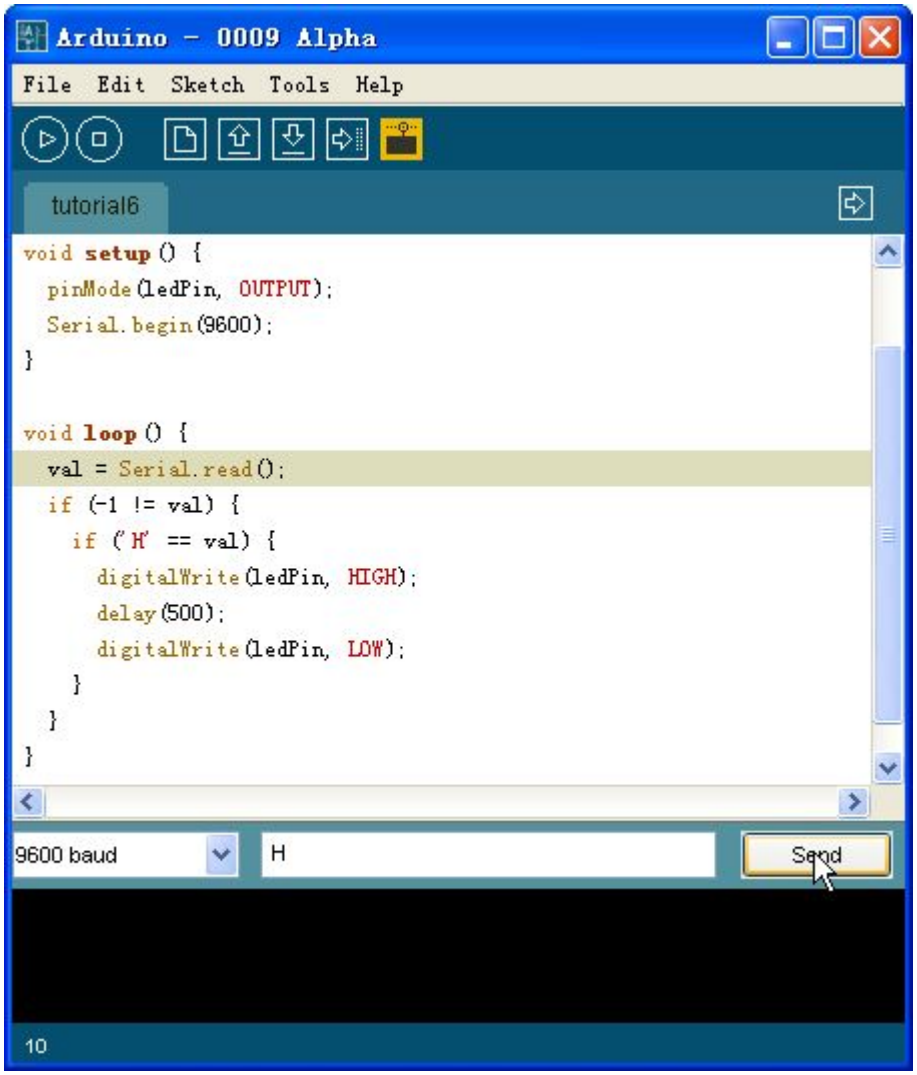
通过串口连接的 Arduino 在交互式设计中能够为 PC 机提供一种全新的交互方式，比如用 PC 机控制一些之前看来非常复杂的事情，像声音和视频等。很多场合中都要求 Arduino 能够通过串口接收来自于 PC 机的命令，并完成相应的功能，这可以通过 Arduino 语言中提供的 Serial.read()函数来实现。

在这一实验中我们同样不需要任何额外的电路，而只需要用串口线将 Arduino 和 PC 机连起来就可以了，相应的 Arduino 工程代码为：

```
int ledPin = 13;
int val;

void setup() {
pinMode(ledPin, OUTPUT);
Serial.begin(9600);
}

void loop() {
```

```
val = Serial.read();
if (-1 != val) {
    if ('H' == val) {
      digitalWrite(ledPin, HIGH);
      delay(500);
      digitalWrite(ledPin, LOW);
    }
}
}
```

把工程下载到 Arduino 模块中之后，在 Arduino 集成开发环境中打开串口监视器并将波特率设置为 9600，然后向 Arduino 模块发送字符 H，如下图所示：



该工程运行起来之后会不断调用 Serial.read()函数从串口获得数据。Arduino 语言提供的这个函数是不阻塞的，也就是说不论串口上是否真的有数据到达，该函数都会立即返回。Serial.read()函数每次只读取一个字节的数据，当串口上有数据到达的时候，该函数的返回值为到达的数 据中第一个字符的 ASCII 码；当串口上没有数据到达的时候，该函数的返回值则为-1。

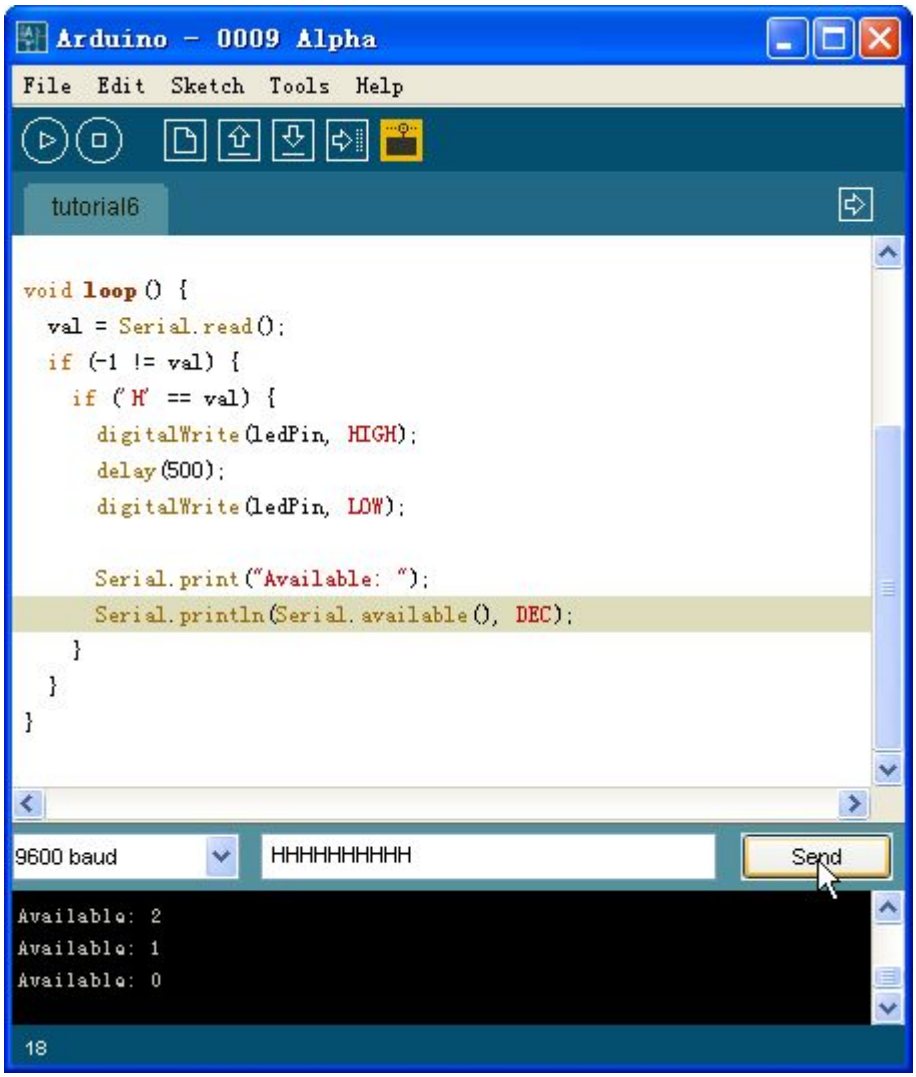
Arduino 语言的参考手册中没 有对 Serial.read()函数做过多的说明，我的一个疑问是如果 PC 机一次发送的数据太多，Arduino 是否提供

相应的串口缓存功能来保证数据不会丢失？Arduino 语言中提供的另外一个函数 Serial.available()或许能够帮助我们用实验来进行验证：

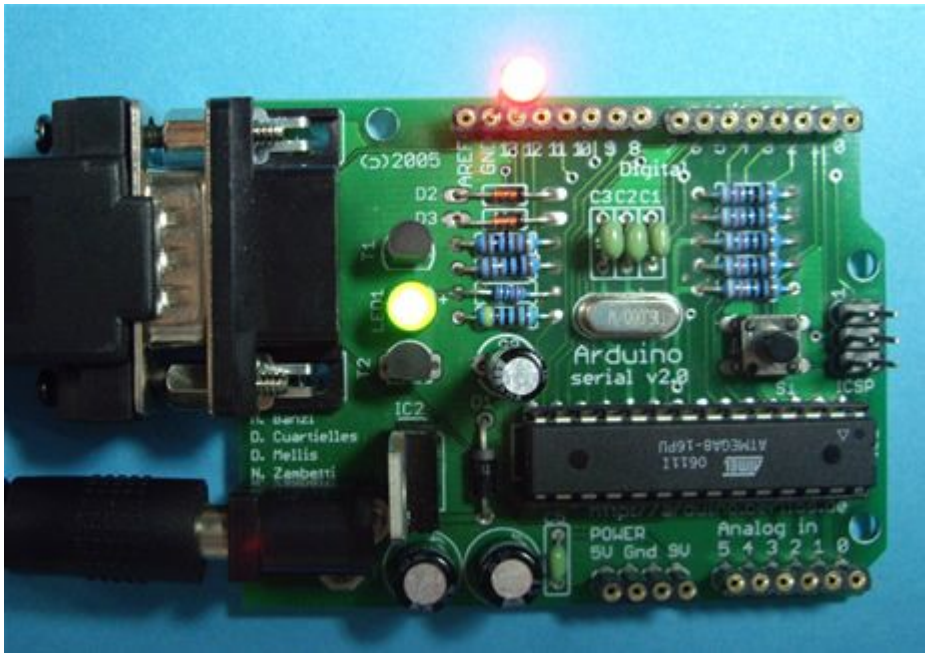
```
int ledPin = 13;
int val;

void setup() {
pinMode(ledPin, OUTPUT);
Serial.begin(9600);
}

void loop() {
val = Serial.read();
if (-1 != val) {
    if ('H' == val) {
      digitalWrite(ledPin, HIGH);
      delay(500);
      digitalWrite(ledPin, LOW);

Serial.print("Available: ");
      Serial.println(Serial.available(), DEC);
    }
```

```
}
}
```

函数 Serial.available()的功能是返回串口缓冲区中当前剩余的字符个数，按照 Arduino 提供的该函数的说明，串口缓冲区中最多能缓冲 128 个字节。我们可以一次给 Arduino 模块发送多个字符，来验证这一功能：



在这一实验中，每当 Arduino 成功收到一个字符 H，连接在数字 I/O 端口管脚 13 上的发光二极管就会闪烁一次：



arduino 学习笔记 11 Arduino 的串口输出

在许多实际应用场合中我们会要求在 Arduino 和其它设备之间实现相互通信，而最常见通常也是最简单的办法就是使用串行通信。在串行通信中，两个设备之间一个接一个地来回发送数字脉冲，它们之间必须严格遵循相应的协议以保证通信的正确性。

在 PC 机上上最常见的串行通信协议是 RS-232 串行协议，而在各种微控制器（单片机）上采用的则是 TTL 串行协议。由于这两者的电平有很大的不同，因此在实现 PC 机和微控制器的通信时，必须进行相应的转换。完成 RS-232 电平和 TTL 电平之间的转换一般采用专

用芯片，如 MAX232 等，但在 Arduino 上是用相应的电平转换电路来完成的。

根据 Arduino 的原理图我们不难看出，ATmega 的 RX 和 TX 引脚一方面直接接到了数字 I/O 端口的 0 号和 1 号管脚，另一方面又通过电平转换电路接到了串口的母头上。因此，当我们需要用 Arduino 与 PC 机通信时，可以用串口线将两者连接起来；当我们需要用 Arduino 与微控制器（如另一块 Arduino）通信时，则可以用数字 I/O 端口的 0 号和 1 号管脚。

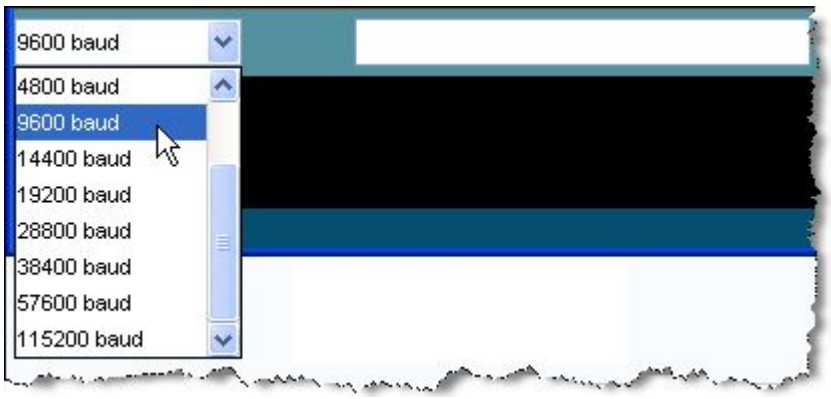
串行通信的难点在于参数的设置，如波特率、数据位、停止位等，在 Arduino 语言可以使用 Serial.begin()函数来简化这一任务。为了实现数据的发送，Arduino 则提供了 Serial.print()和 Serial.println()两个函数，它们的区别在于后者会在请求发送的数 据后面加上换行符，以提高输出结果的可读性。

在这一实验中没有用到额外的电路，我们只需要用串口线将 Arduino 和 PC 机连起来就可以了，相应的代码为：

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

void loop() {
```

```
  Serial.println("Hello World!");
  delay(1000);
}
```

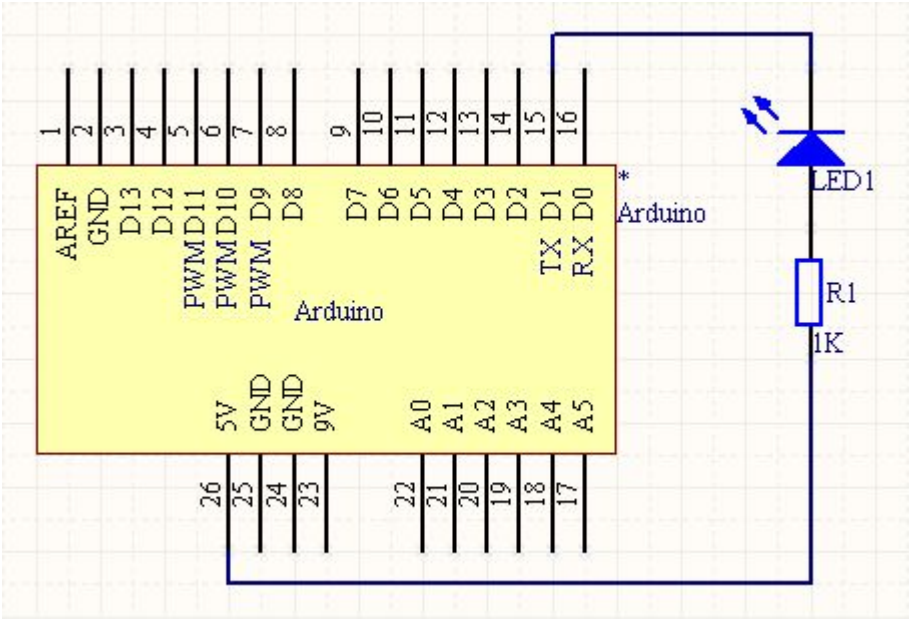
在将工程下载到 Arduino 模块中之后，在 Arduino 集成开发环境的工具栏中单击“Serial Monitor”控制，打开串口监视器：



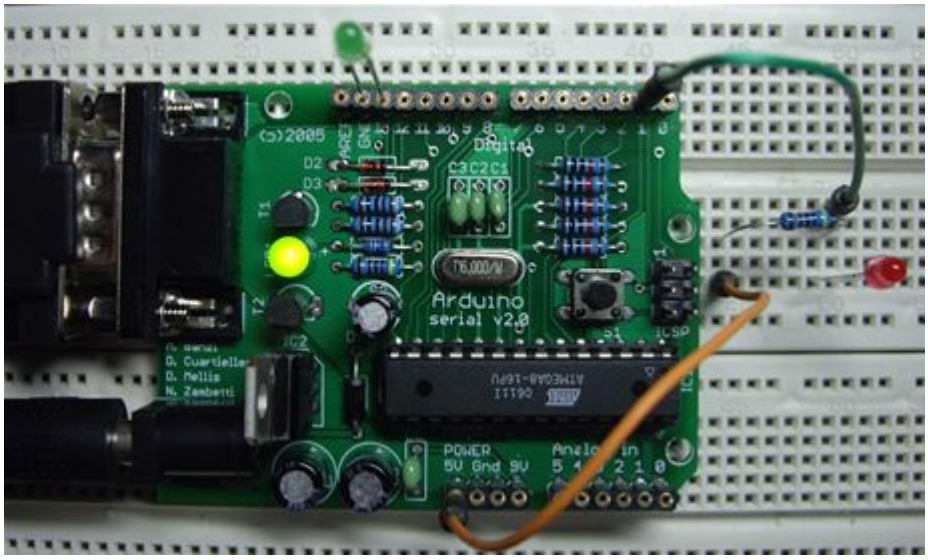
接着将波特率设置为 9600，即保持与工程中的设置相一致：

如果一切正常，此时我们就可以在 Arduino 集成开发环境的 Console 窗口中看到串口上输出的数据了：

为了检查串口上是否有数据发送，一个比较简单的办法是在数字 I/O 端口的 1 号管脚（TX）和 5V 电源之间接一个发光二极管，如下面的原理图所示：



这样一旦 Arduino 在通过串口向 PC 机发送数据时，相应的发光二极管就会闪烁，实际应用中这是一个非常方便的调试手段;-)



arduino 学习笔记 12 PWM（脉冲宽度调制）

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square

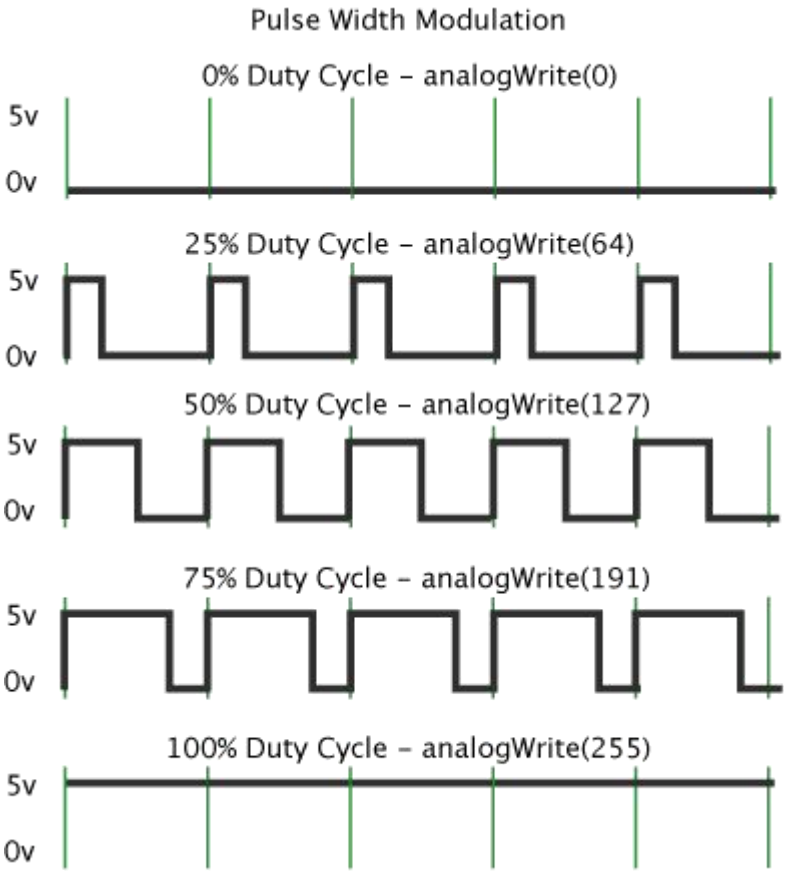
wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width. To get varying analog values, you change, or modulate, that pulse width. If you repeat this on-off pattern fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and 5v controlling the brightness of the LED.

脉冲宽度调制或 PWM，是通过数字均值获得模拟结果的技术。数字控制被用来创建一个方波，信号在开和关之间切换。这种开关模式通过改变“开”时间段和“关”时间段的比值完全模拟从开（5 伏特）和关（0 伏特）之间的电压。“开时间“的周期称为脉冲宽度。为了得到不同的模拟值，你可以更改，或调节脉冲宽度。如果你重复这种开关模式速度足够快，其结果是一个介于 0 和 5V 之间的稳定电压用以控制 LED 的亮度。

In the graphic below, the green lines represent a regular time period. This duration or period is the inverse of the PWM frequency. In other words, with Arduino's PWM frequency at about 500Hz, the green lines would measure 2 milliseconds each. A call to analogWrite() is on a scale of 0 - 255, such that analogWrite(255)

requests a 100% duty cycle (always on), and analogWrite(127) is a 50% duty cycle (on half the time) for example.

下图中，绿色线表示一个固定的时间期限。此持续时间或周期是 PWM 的频率的倒数。换言之，Arduino 的 PWM 频率约为 500Hz，每个绿线之间表示 2 毫秒。一个 analogWrite（）的调用区间为 0- 255，例如 analogWrite（255）需要 100％占空比（常开），和 analogWrite（127）是 50％占空比（上一半的时间）。



Once you get this example running, grab your arduino and shake it back and forth. What you are doing here is essentially mapping time across the space. To our eyes, the movement blurs each LED

blink into a line. As the LED fades in and out, those little lines will grow and shrink in length. Now you are seeing the pulse width.

一旦你运行这个例子中，抓住你的 Arduino 来回摇晃。你这么做的实质上是时间跨越时空的映射。对我们的眼睛，每个运动模糊成一条线的 LED 闪烁。由于 LED 消失和缩小，那些小行的长度会增长和收缩。现在你就可以看到脉冲宽度。

arduino 学习笔记 13 Arduino 的模拟输入

Description 介绍

Reads the value from the specified analog pin. The Arduino board contains a 6 channel (8 channels on the Mini and Nano, 16 on the Mega), 10-bit analog to digital converter. This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023. This yields a resolution between readings of: 5 volts / 1024 units or, .0049 volts (4.9 mV) per unit. The input range and resolution can be changed using analogReference().

It takes about 100 microseconds (0.0001 s) to read an analog input, so the maximum reading rate is about 10,000 times a second.

从指定的模拟引脚读取值。Arduino 主板有 6 个通道（Mini 和 Nano 有 8 个，Mega 有 16 个），10 位 AD（模数）转换器。这意味着输入电压 0-5 伏对应 0-1023 的整数值。这就是说读取精度为：5 伏/1024

个单位，约等于每个单位 0.049 伏（4.9 毫伏）。输入范围和进度可以通过 analogReference()进行修改。

模拟输入的读取周期为 100 微秒（0.0001 秒），所以最大读取速度为每秒 10,000 次。

Syntax 语法

analogRead(pin)

Parameters 参数

pin: the number of the analog input pin to read from (0 to 5 on most boards, 0 to 7 on the Mini and Nano, 0 to 15 on the Mega)

pin：读取的模拟输入引脚号（大多数主板是 0-5，Mini 和 Nano 是 0-7，Mega 是 0-15）

Returns 返回值

int (0 to 1023)

整数型 int（0 到 1023）

Note 备注

If the analog input pin is not connected to anything, the value returned by analogRead() will fluctuate based on a number of factors (e.g. the values of the other analog inputs, how close your hand is to the board, etc.).

如果模拟输入引脚没有连接到任何地方，analogRead()的返回值也会因为某些因素而波动（如其他模拟输入，你的手与主板靠的太近）

Example 例子

```
int analogPin = 3;         // potentiometer wiper (middle terminal)
connected to analog pin 3
                                  // outside leads to ground and +5V
int val = 0;                  // variable to store the value read
void setup()
{
  Serial.begin(9600);               //   setup serial
}
void loop()
{
  val = analogRead(analogPin);       // read the input pin
  Serial.println(val);                  // debug value
}
```

arduino 学习笔记 14 Arduino 的模拟输出

慧净电子---ARDUINO 模块化创新视频教程

Description 介绍

Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds. After a call to **analogWrite()**, the pin will generate a steady square wave of the specified duty cycle until the next call to **analogWrite()** (or a call to **digitalRead()** or **digitalWrite()** on the same pin). The frequency of the PWM signal is approximately 490 Hz.

将模拟值（PWM 波）输出到管脚。可用于在不同的光线亮度调节发光二极管亮度或以不同的速度驱动马达。调用 analogWrite()后，该引脚将产生一个指定占空比的稳定方波，直到下一次调用 analogWrite()（或在同一引脚调用 digitalRead()或 digitalWrite()）。 PWM 的信号频率约为 490 赫兹。

On most Arduino boards (those with the ATmega168 or ATmega328), this function works on pins 3, 5, 6, 9, 10, and 11. On the Arduino Mega, it works on pins 2 through 13. Older Arduino boards with an ATmega8 only support analogWrite() on pins 9, 10, and 11. You do not need to call pinMode() to set the pin as an output before calling analogWrite().

在大多数 Arduino 板（带有 ATmega168 或 ATmega328），这个函数工作在引脚 3，5，6，9，10 和 11。在 ArduinoMega，它适用于 2-13 号引脚。老的带有 ATmega8 的 Arduino 板只支持 9，10，11

引脚上使用。你并不需要在调用 analogWrite()之前为设置输入引脚而调用 pinMode()。

The *analogWrite* function has nothing whatsoever to do with the analog pins or the *analogRead* function.

这个 analogWrite 方法与模拟引脚或者 analogRead 方法毫不相干

Syntax 语法

analogWrite(pin, value)

Parameters 参数

pin: the pin to write to.

pin：输出的引脚号

value: the duty cycle: between 0 (always off) and 255 (always on).

value：占用空：从 0（常关）到 255（常开）

Returns 返回值

nothing

Notes and Known Issues 备注和已知问题

The PWM outputs generated on pins 5 and 6 will have higher-than-expected duty cycles. This is because of interactions with the millis() and delay() functions, which share the same internal timer used to generate those PWM outputs. This will be noticed mostly on low duty-cycle settings (e.g 0 - 10) and may

result in a value of 0 not fully turning off the output on pins 5 and 6.

引脚 5 和 6 的 PWM 输出将产生高于预期的占空比。这是因为 millis()和 delay()函数，它们共享同一个内部定时器用于产生 PWM 输出所产生的相互作用。这提醒我们引脚 5 和 6 在多数低占空比的设置（如 0- 10）的情况下 0 数值的结果并没有完全关闭。

Example 例子

Sets the output to the LED proportional to the value read from the potentiometer.

```
int ledPin = 9;          // LED connected to digital pin 9
int analogPin = 3;       // potentiometer connected to analog pin 3
int val = 0;             // variable to store the read value
void setup()
{
  pinMode(ledPin, OUTPUT);     // sets the pin as output
}
void loop()
{
  val = analogRead(analogPin);     // read the input pin
```

analogWrite(ledPin, val / 4); // analogRead values go from 0 to 1023, analogWrite values from 0 to 255

arduino 学习笔记 15 制作第一个电路 单 led 闪烁

做任何一个电路之前，一定要先了解电路中原件的参数，其工作电压，工作电流等。

第一个实验咱们用到的 LED 从网上查找资料得知，其工作电压一般为 1.5-2.0V，工作电流一般为 10-20ma，反向击穿电压为 5V。控制板逻辑电路供电为 5V。根据以上参数假设 LED 工作电压选用 1.7，工作电流选用 15ma，限流电阻=（总电压-LED 电压）/电流，所以限流电阻=（5-1.7）/0.015=220Ω。

首先需要从 arduino 官方网站下载其编译软件，地址是 http://arduino.cc/en/Main/Software

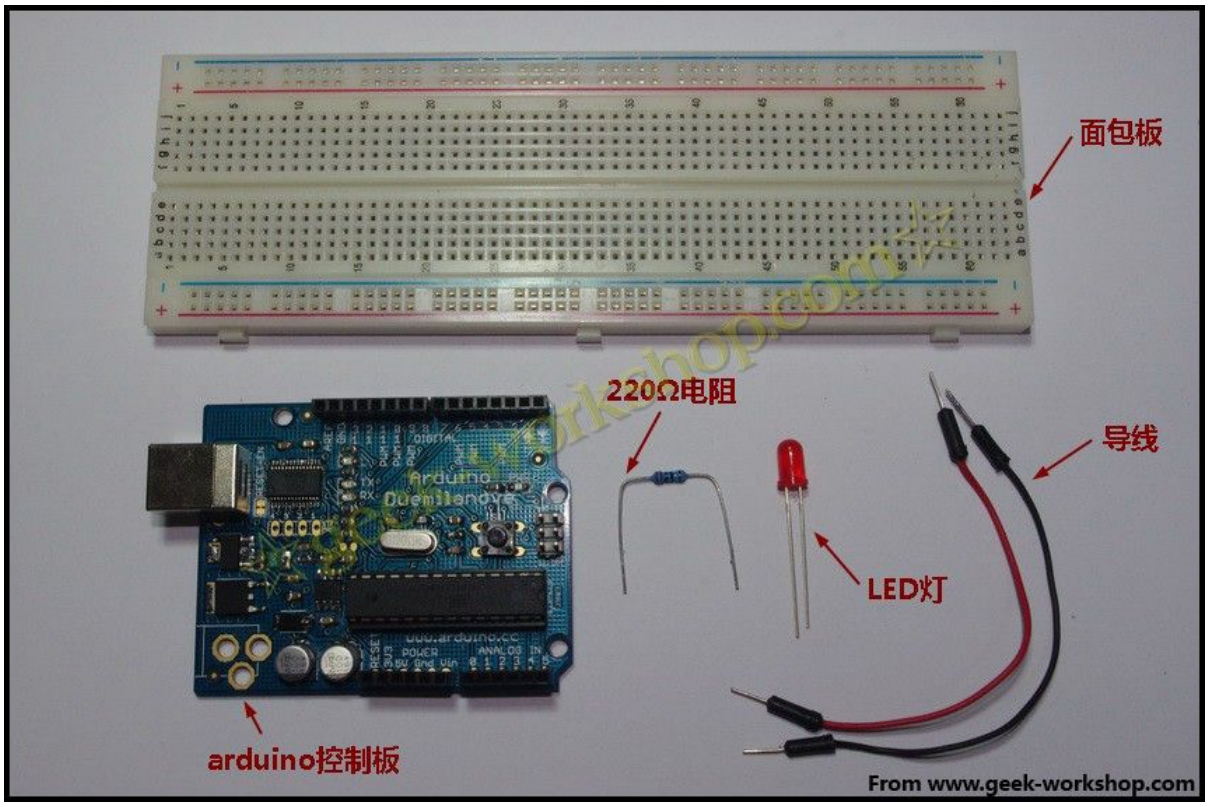
笔者使用的平台为 win7 32 位，如果大家使用的是其他平台，按照对应的下载就可以。

下载回来的软件包解压缩后的目录结构如下图，arduino.exe 是程序的启动文件，driver 目录是控制板 usb 芯片驱动，usb 接上控制板后如果要寻找驱动，把目录指定到这里就可以。

慧净电子---ARDUINO 模块化创新视频教程

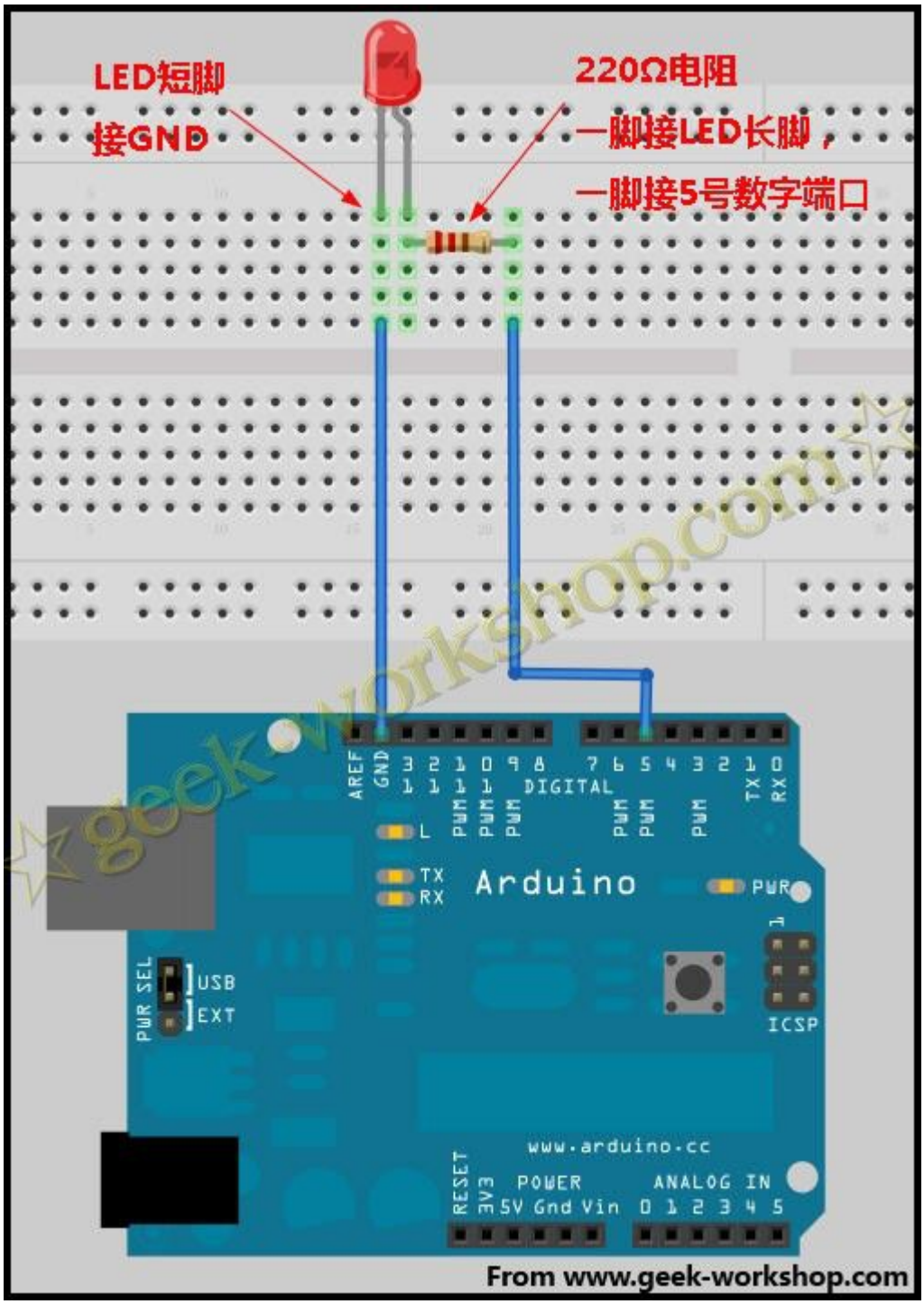
本次实验所用到的器材为一个面包板，一个 LED，一个 220Ω的电阻，几根导线，如下图：

慧净电子---ARDUINO 模块化创新视频教程



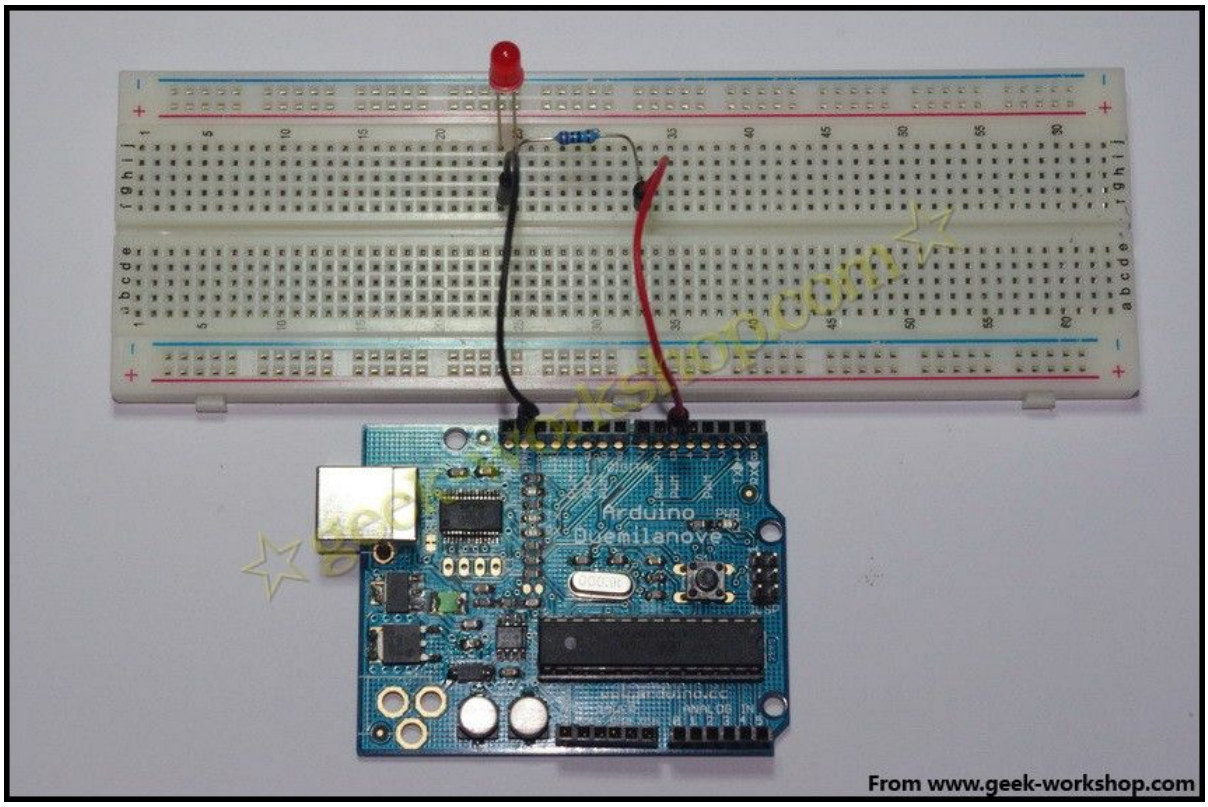
本次实验的连接方法如下图，LED 两个针脚有一长一短，短的是连接 GND，长的是连接正极。在 LED 的长引脚前，需要添加一个 220Ω的限流电阻。连接数字 5 号接口。

慧净电子---ARDUINO 模块化创新视频教程



实物连接如下图：

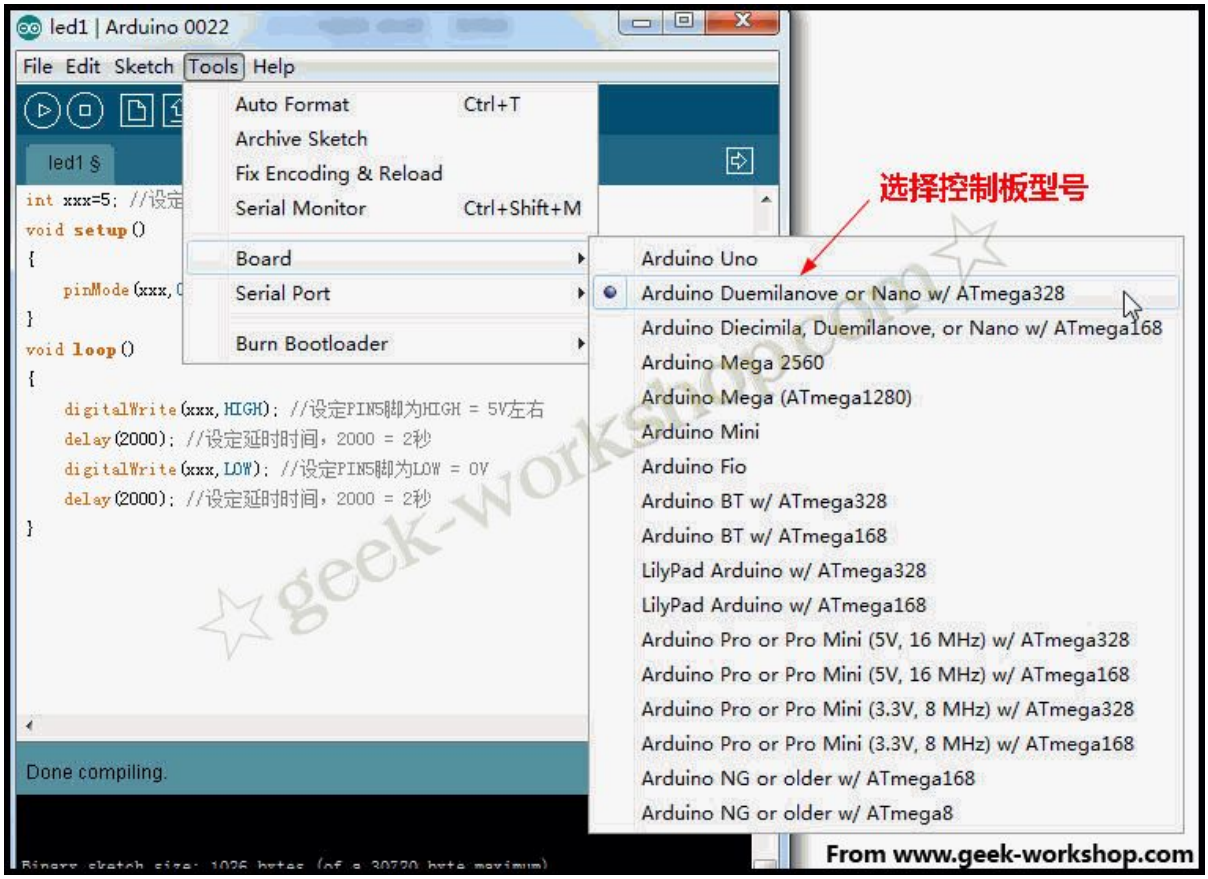
慧净电子---ARDUINO 模块化创新视频教程



通过面包板把个个电子器件连接好以后，接上 USB 线，设置好控制板型号、端口号。程序语言先不详解，大家先照猫画虎对着抄，后面通过各个实验，将对不同命令进行详解。

编写程序前，先需要选择控制板的型号。如下图：

慧净电子---ARDUINO 模块化创新视频教程



控制板型号选择好后，选择串口位置，笔者电脑的串口为 COM3：

串口具体是多少号可以到设备管理中进行查看，如下图：

慧净电子---ARDUINO 模块化创新视频教程

先把程序复制进去：

```
1.  int ledPin=5; //设定控制 LED 的数字 IO 脚
2.  void setup()
3.  {
4.      pinMode(ledPin,OUTPUT);//设定数字 IO 口的模式，
    OUTPUT 为输出
5.  }
6.  void loop()
```

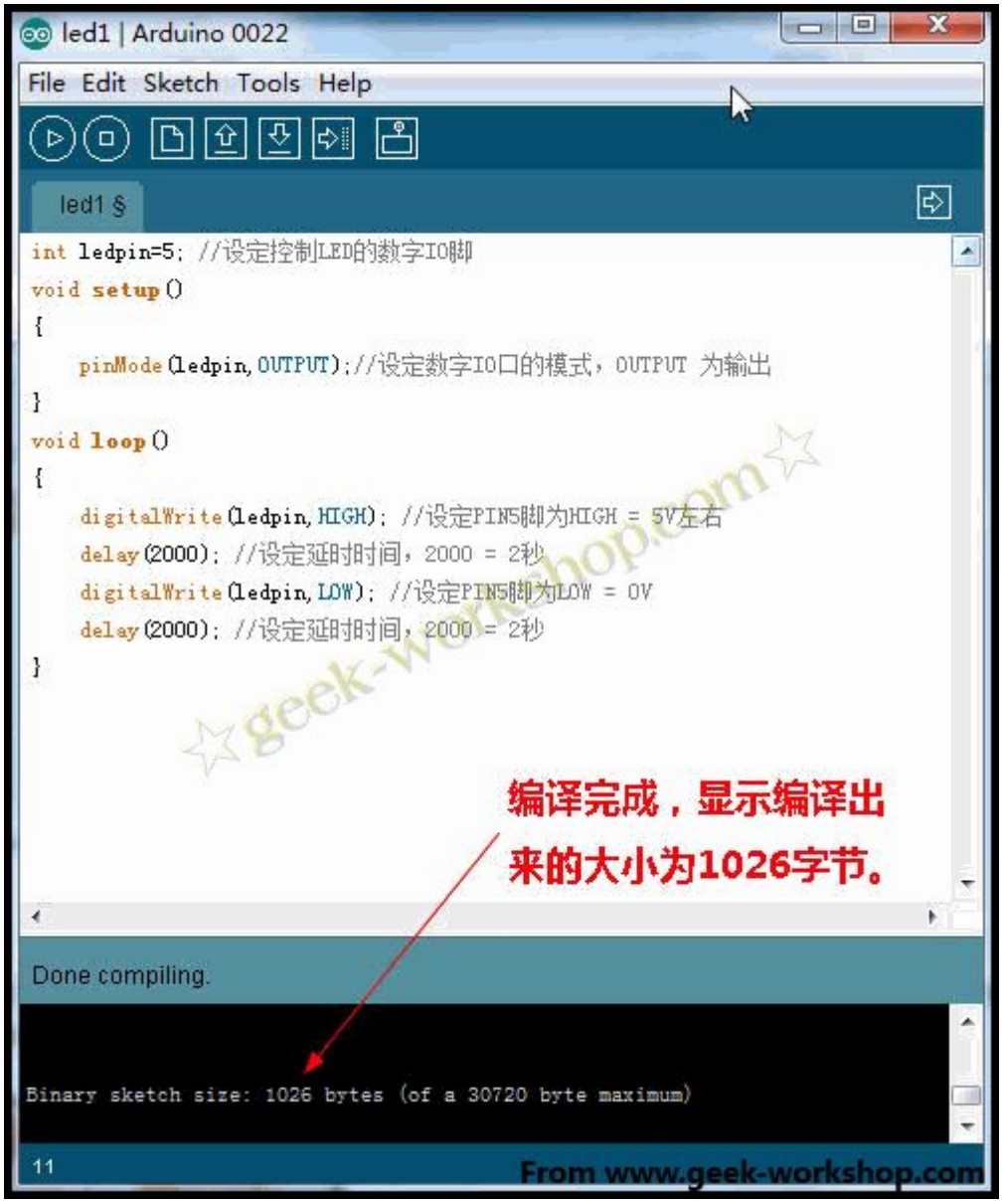
```
7.  {
8.      digitalWrite(ledPin,HIGH); //设定 PIN5 脚为 HIGH = 5V 左
    右
9.      delay(2000); //设定延时时间，2000 = 2 秒
10.     digitalWrite(ledPin,LOW); //设定 PIN5 脚为 LOW = 0V
11.     delay(2000); //设定延时时间，2000 = 2 秒
12. }
```

复制代码

图中 int;void setup 等褐色的文字是系统命令，OUTPUT 等蓝色文字是命令的功能开关，黑色文字是变量。

程序写好以后点击编译按钮进行编译。

编译完成后会显示出来编译后的文件大小，本次编译出来的程序大小为 1026 字节。

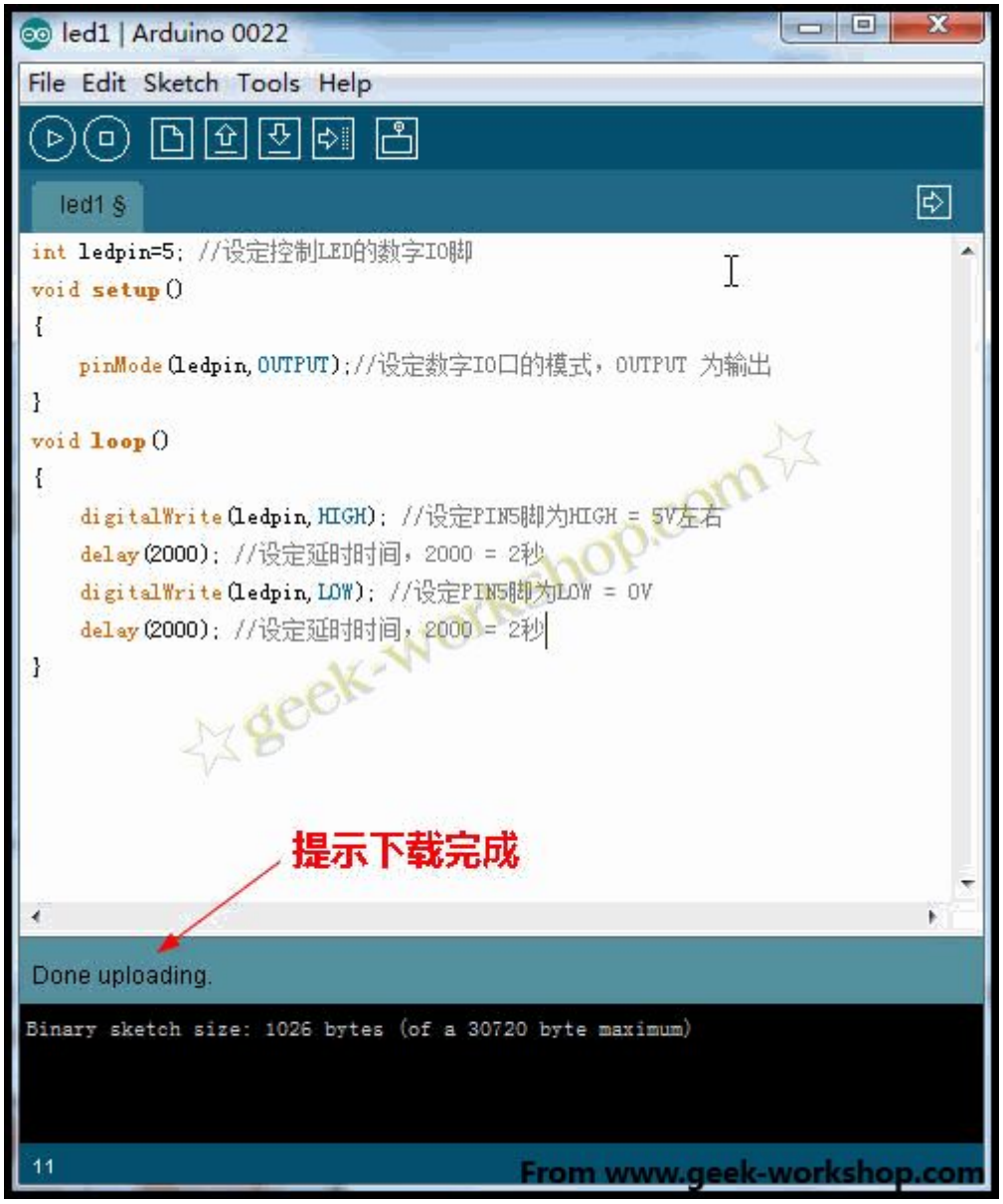


然后把编译好的程序下载到 arduino 控制板上，点击下载按钮。

慧净电子---ARDUINO 模块化创新视频教程

下载完成后会有提示

慧净电子---ARDUINO 模块化创新视频教程



把所有的 ledpin 换成 xxx 试试，一样可以滴~~~ledpin 只是自己定义的一个名字，作用只是方便识别辨认。

慧净电子---ARDUINO 模块化创新视频教程

本次实验效果如下，闪烁着光芒的灯。。。

总结：

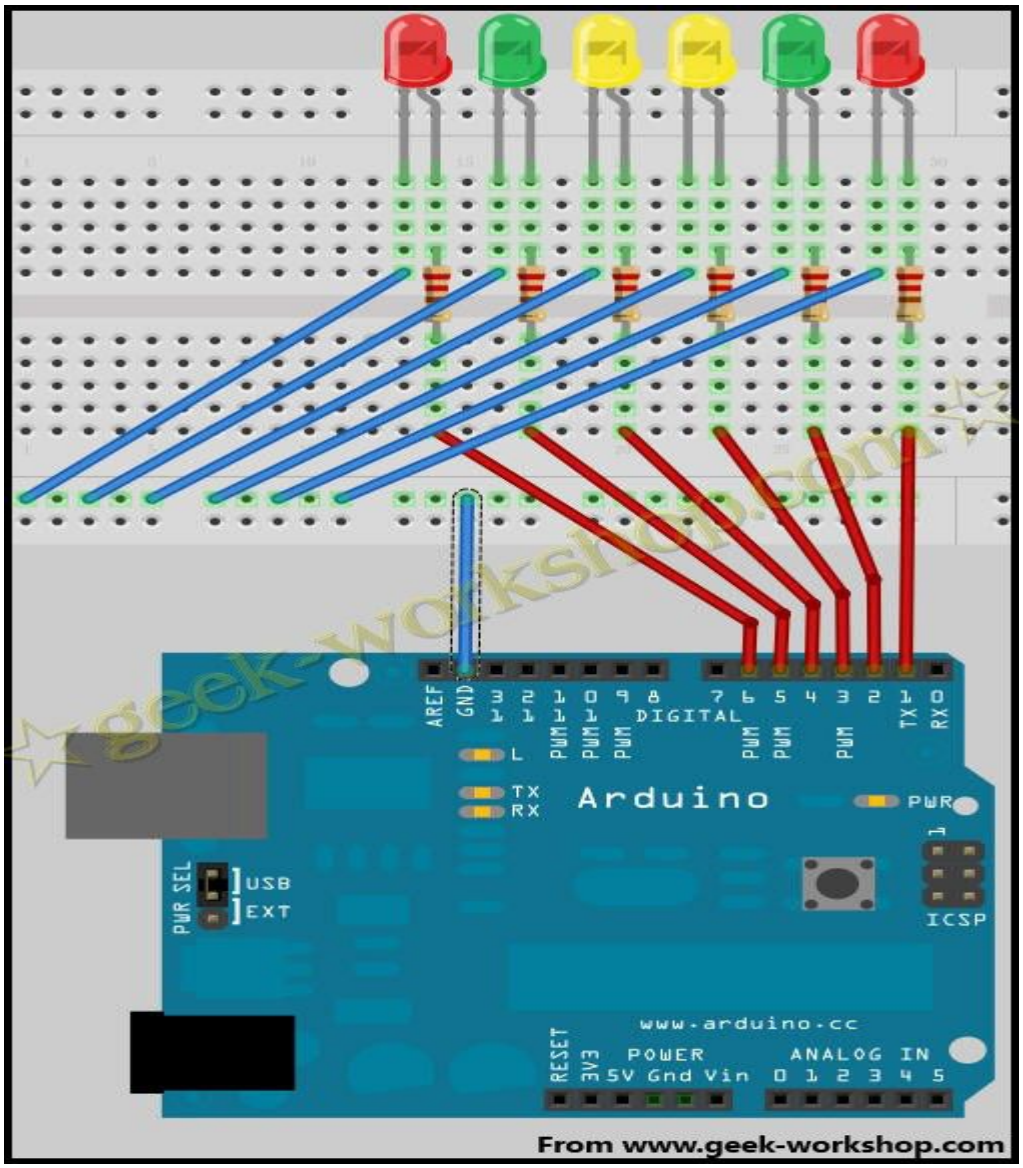
int;void setup 等褐色文字是系统命令，OUTPUT 等蓝色文字是系统命令的功能开关，黑色文字是变量。

在"int ledpin=5"中;设置了 LED 的数字 IO 脚，ledpin 仅仅是 5 号数字端口自定义出来的名字，变成 xxx 等都可以。对于多脚 IO 操作的程序中，为每一个引脚定义名字是有必要性的，程序复杂后方便功能识别。

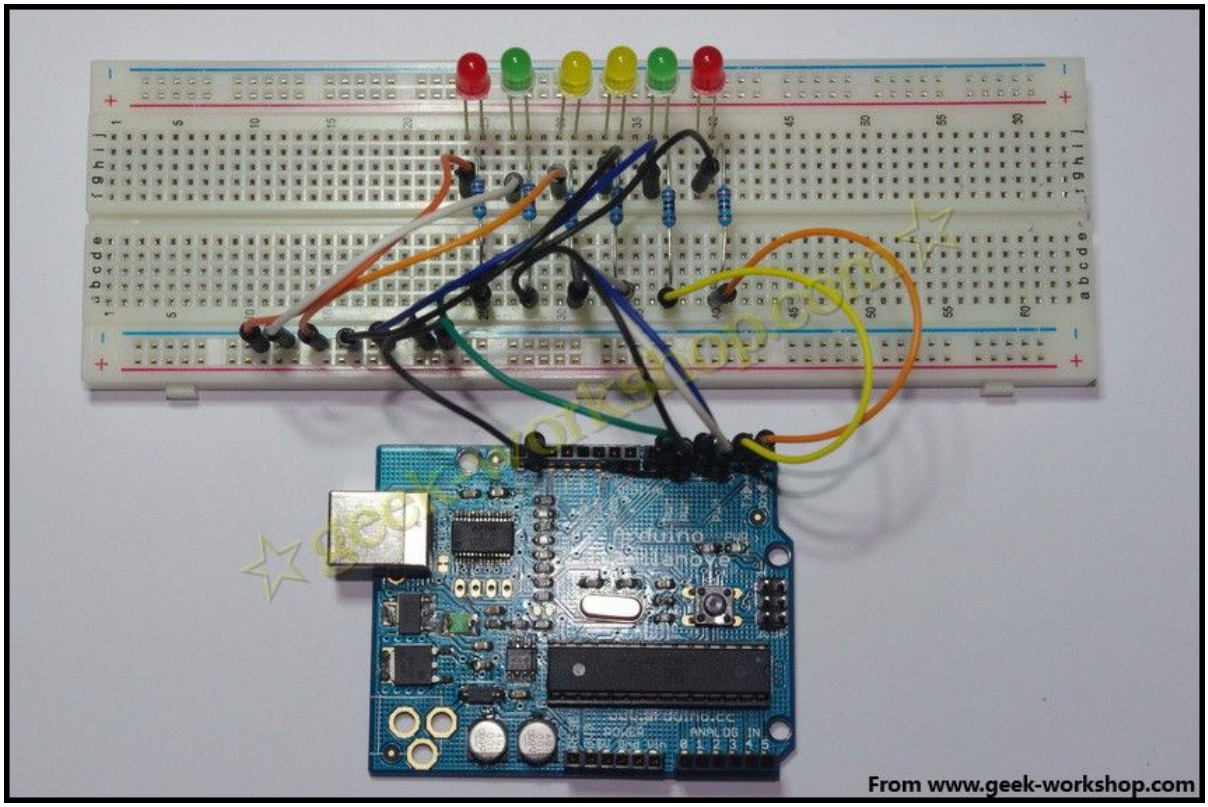
arduino 学习笔记 16 六灯闪烁实验

通过上一节学习，知道了怎样让一个 led 进行闪烁，下面的实验会对六个 led 进行控制，先看硬件连接图。

慧净电子---ARDUINO 模块化创新视频教程



慧净电子---ARDUINO 模块化创新视频教程



按照上面的硬件连接方法接好后，咱们来测试两段程序，看看其中的差别。通过这两段程序介绍一下 arduino 的语言轮廓。

```
1.  //设置控制 Led 的数字 IO 脚
2.  int Led1 = 1;
3.  int Led2 = 2;
4.  int Led3 = 3;
5.  int Led4 = 4;
6.  int Led5 = 5;
7.  int Led6 = 6;
8.  //led 灯花样显示样式 1 子程序
```

```
9.  void style_1(void)
10. {
11.   unsigned char j;
12.   for(j=1;j<=6;j++)
13.   {
14.     digitalWrite(j,HIGH);
15.     delay(200);
16.   }
17.   for(j=6;j>=1;j--)
18.   {
19.     digitalWrite(j,LOW);
20.     delay(200);
21.   }
22. }
23. void setup()
24. {
25.   unsigned char i;
26.   for(i=1;i<=6;i++)//依次设置 1~6 个数字引脚为输出模式
27.     pinMode(i,OUTPUT);//设置第 i 个引脚为输出模式
28. }
29. void loop()
30. {
```

```
31.   style_1();//样式 1
32.
33. }
```

复制代码

```
1.   //设置控制 Led 的数字 IO 脚
2.   int Led1 = 1;
3.   int Led2 = 2;
4.   int Led3 = 3;
5.   int Led4 = 4;
6.   int Led5 = 5;
7.   int Led6 = 6;
8.   //led 灯花样显示样式 1 子程序
9.   void style_1(void)
10. {
11.   unsigned char j;
12.   for(j=1;j<=6;j++)
13.      digitalWrite(j,HIGH);
14.      delay(200);
15.
16.   for(j=6;j>=1;j--)
17.   {
18.      digitalWrite(j,LOW);
```

```
19.      delay(200);
20.   }
21. }
22. void setup()
23. {
24.   unsigned char i;
25.   for(i=1;i<=6;i++)//依次设置 1~6 个数字引脚为输出模式
26.      pinMode(i,OUTPUT);//设置第 i 个引脚为输出模式
27. }
28. void loop()
29. {
30.   style_1();//样式 1
31.
32. }
```

复制代码

通过下载测试，发现第一段程序是 led 1-6 逐个点亮，然后从 6-1 再逐个熄灭如此循环。

第二段程序是 6 个灯同时亮，然后再 6-1 逐个熄灭如此循环。下面对产生不同效果的代码进行分析。

下面这段代码代表的 for 语句循环的是，是把 j 点亮，然后再延迟 200

毫秒，然后再循环。形成的效果就是 6 个灯相隔 200 毫秒逐步被点亮。

```
1.    for(j=1;j<=6;j++)
2.    {
3.        digitalWrite(j,HIGH);
4.        delay(200);
5.    }
```

复制代码

下面这段代码其实是不规范写法，for 命令表达要求一定要有{}循环，如果没有标出{}，编译时就会自动对下一句加上{}。如果代码量很大，出问题是查找起来会非常辛苦。

```
1.    for(j=1;j<=6;j++)
2.          digitalWrite(j,HIGH);
3.        delay(200);
```

复制代码

正确的写法应该是下面这个样子

```
1.  for(j=1;j<=6;j++)  {
2.        digitalWrite(j,HIGH);
3.    }
4.        delay(200);
```

复制代码

六个灯逐个被点亮，然后再延时 200 毫秒进入下一句。因为六灯逐个

点亮的速度非常快，所以看上去像一起亮的。

void（无类型）在 arduino 中是数据类型的一种，通常用来代表一个事件。如果控制过程比较简单 void 一般无需定义，直接使用

```
1.  void setup()
2.  {
3.    // ...
4.  }
5.
6.  void loop()
7.  {
8.    // ...
9.  }
```

复制代码

代表事件的开始与事件的循环。

如果控制过程比较复杂，一般就要设置多个子事件，把复杂的过程进行分解，每一个子事件定义为一个 void 数据。

把以下代码上传上去，看看 led 灯是如何工作的。

```
1.  //设置控制 Led 的数字 IO 脚
2.  int Led1 = 1;
```

```
3.  int Led2 = 2;
4.  int Led3 = 3;
5.  int Led4 = 4;
6.  int Led5 = 5;
7.  int Led6 = 6;
8.  //led 灯花样显示样式 1 子程序
9.  void style_1(void)
10. {
11.   unsigned char j;
12.   for(j=1;j<=6;j++)//每隔 200ms 依次点亮 1~6 引脚相连的
    led 灯
13.   {
14.     digitalWrite(j,HIGH);//点亮 j 引脚相连的 led 灯
15.     delay(200);//延时 200ms
16.   }
17.   for(j=6;j>=1;j--)//每隔 200ms 依次熄灭 6~1 引脚相连的 led
    灯
18.   {
19.     digitalWrite(j,LOW);//熄灭 j 引脚相连的 led 灯
20.     delay(200);//延时 200ms
21.   }
22. }
```

```
23. //灯闪烁子程序
24. void flash(void)
25. {
26.   unsigned char j,k;
27.   for(k=0;k<=1;k++)//闪烁两次
28.   {
29.     for(j=1;j<=6;j++)//点亮 1~6 引脚相连的 led 灯
30.       digitalWrite(j,HIGH);//点亮与 j 引脚相连的 led 灯
31.     delay(200);//延时 200ms
32.     for(j=1;j<=6;j++)//熄灭 1~6 引脚相连的 led 灯
33.       digitalWrite(j,LOW);//熄灭与 j 引脚相连的 led 灯
34.     delay(200);//延时 200ms
35.   }
36. }
37. //led 灯花样显示样式 2 子程序
38. void style_2(void)
39. {
40.   unsigned char j,k;
41.   k=1;//设置 k 的初值为 1
42.   for(j=3;j>=1;j--)
43.   {
44.     digitalWrite(j,HIGH);//点亮灯
```

```
45.     digitalWrite(j+k,HIGH);//点亮灯
46.     delay(400);//延时 400ms
47.     k +=2;//k 值加 2
48.  }
49.  k=5;//设置 k 值为 5
50.  for(j=1;j<=3;j++)
51.  {
52.     digitalWrite(j,LOW);//熄灭灯
53.     digitalWrite(j+k,LOW);//熄灭灯
54.     delay(400);//延时 400ms
55.     k -=2;//k 值减 2
56.  }
57. }
58. //led 灯花样显示样式 3 子程序
59. void style_3(void)
60. {
61.  unsigned char j,k;//led 灯花样显示样式 3 子程序
62.  k=5;//设置 k 值为 5
63.  for(j=1;j<=3;j++)
64.  {
65.     digitalWrite(j,HIGH);//点亮灯
66.     digitalWrite(j+k,HIGH);//点亮灯
```

```
67.      delay(400);//延时 400ms
68.      digitalWrite(j,LOW);//熄灭灯
69.      digitalWrite(j+k,LOW);//熄灭灯
70.      k -=2;//k 值减 2
71.   }
72.   k=3;//设置 k 值为 3
73.   for(j=2;j>=1;j--)
74.   {
75.      digitalWrite(j,HIGH);//点亮灯
76.      digitalWrite(j+k,HIGH);//点亮灯
77.      delay(400);//延时 400ms
78.      digitalWrite(j,LOW);//熄灭灯
79.      digitalWrite(j+k,LOW);//熄灭灯
80.      k +=2;//k 值加 2
81.   }
82. }
83. void setup()
84. {
85.   unsigned char i;
86.   for(i=1;i<=6;i++)//依次设置 1~6 个数字引脚为输出模式
87.      pinMode(i,OUTPUT);//设置第 i 个引脚为输出模式
88. }
```

```
89. void loop()
90. {
91.   style_1();//样式 1
92.   flash();//闪烁
93.   style_2();//样式 2
94.   flash();//闪烁
95.   style_3();//样式 3
96.   flash();//闪烁
97. }
```

复制代码

上面的代码分为 4 个子事件，样式 1，样式 2，样式 3 和闪烁。

闪烁：1-6 号灯亮，延时 200ms，1-6 号灯熄灭，延时 200ms。

样式 1:1-6 号灯逐个点亮，然后 6-1 逐个熄灭。效果视频如下：

样式 2：3,4 号灯先亮，然后 2，5 号再亮，最后 1,6 号两。接着 1，

6 号灯熄灭，再 2，5 号熄灭，最后 3,4 号熄灭。效果视频如下：

样式三：3,4 号灯亮，然后 3,4 号熄灭 2，5 号亮，然后 2,5 号熄灭 1，6 号亮，再 1，6 号熄灭 2,5 号亮，最后 2,5 号熄灭 3,4 号亮。

通过以上 4 种事件的组合，就可以做出来各种效果。

arduino 学习笔记 17 蜂鸣器实验

本次实验所用的为下图所示的这种小型无源蜂鸣器

慧净电子---ARDUINO 模块化创新视频教程

基于：慧净 ARDUINO 智能机器人---视频教程下载网址：WWW.HJMCU.COM　WWW.HLMCU.COM

通过上网查询参数，得到其工作电压为 5V，和 arduino 控制板数字端口输出电压一致，所以不需要接电阻。可直接接上使用。

先简单介绍一下这种小型蜂鸣器。

小型蜂鸣器因其体积小(直径只有 6mm)、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。这种蜂鸣器分有源蜂鸣器与无源蜂鸣器

下图为有源蜂鸣器

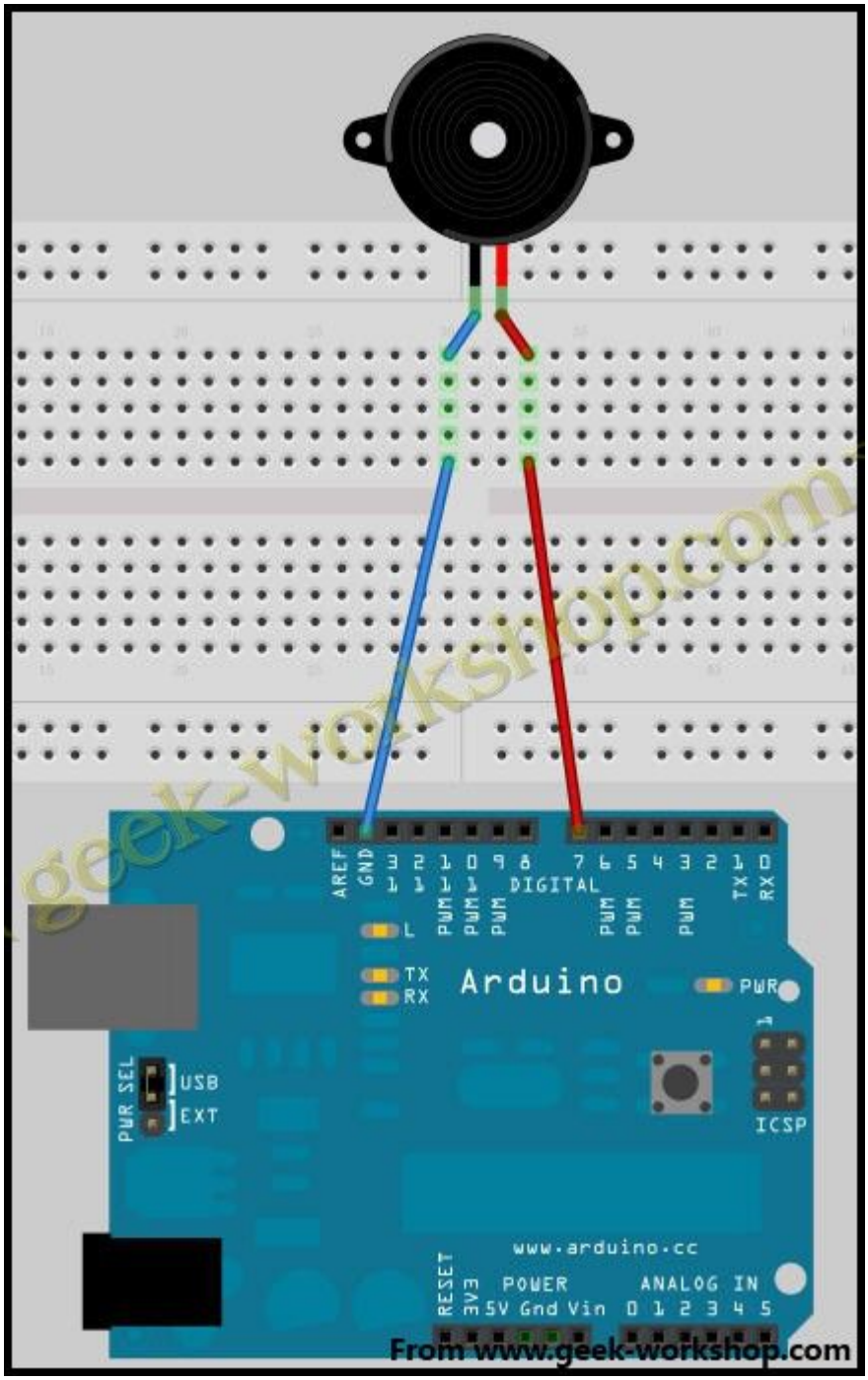
下图为无源蜂鸣器

从外观上看，两种蜂鸣器好像一样，如果将蜂鸣器引脚朝上时，可以看到，有绿色电路板的是一种无源蜂鸣器，没有电路板而使用黑胶密封的是一种有源蜂鸣器。从外观上并不能绝对的区分出有源与无源，最可靠的做法除了查看产品的参数手册以外，还有就是使用万用表测试蜂鸣器电阻，只有 8Ω或者 16Ω的是无源蜂鸣器，电阻在几百欧以上的是有源蜂鸣器。

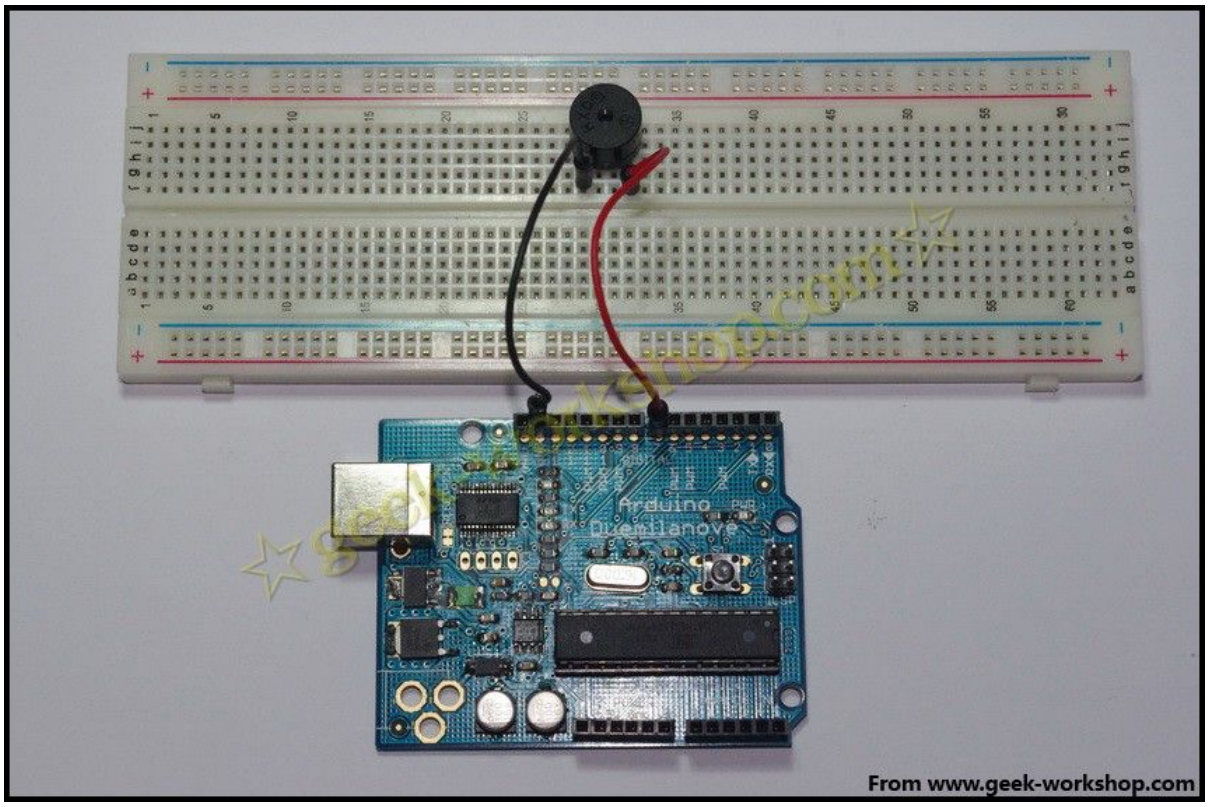
有源蜂鸣器直接接上额定电源（新的蜂鸣器在标签上都有注明）就可以连续发声，而无源蜂鸣器则和电磁扬声器一样，需要接在音频输出电路中才能发声。

慧净电子---ARDUINO 模块化创新视频教程

简单介绍完蜂鸣器以后先看一下硬件连接示意图



慧净电子---ARDUINO 模块化创新视频教程



把下面的代码上传到 arduino 控制板上，看看实验结果。

```
1.  int buzzer=7;//设置控制蜂鸣器的数字 IO 脚
2.  void setup()
3.  {
4.    pinMode(buzzer,OUTPUT);//设置数字 IO 脚模式，OUTPUT 为
    输出
5.  }
6.  void loop()
7.  {
8.    unsigned char i,j;//定义变量
```

```
9.    while(1)
10.   {
11.     for(i=0;i<80;i++)//输出一个频率的声音
12.     {
13.       digitalWrite(buzzer,HIGH);//发声音
14.       delay(1);//延时 1ms
15.       digitalWrite(buzzer,LOW);//不发声音
16.       delay(1);//延时 ms
17.     }
18.     for(i=0;i<100;i++)//输出另一个频率的声音
19.     {
20.       digitalWrite(buzzer,HIGH);//发声音
21.       delay(2);//延时 2ms
22.       digitalWrite(buzzer,LOW);//不发声音
23.       delay(2);//延时 2ms
24.     }
25.   }
26. }
```

复制代码

第一个频率的声音为 1 毫秒发声 1 毫秒不发声。1 秒等于 1000 毫秒，2 毫秒为一个周期。得出频率为 500 赫兹。

第二个频率的声音为 2 毫秒发声 2 毫秒不发声，4 毫秒为一个周期。得出频率为 250 赫兹。

一个事件的循环就是 500 赫兹的声音响 80 毫秒，然后 250 赫兹的声音响 100 毫秒。如此循环下去。

while()函数

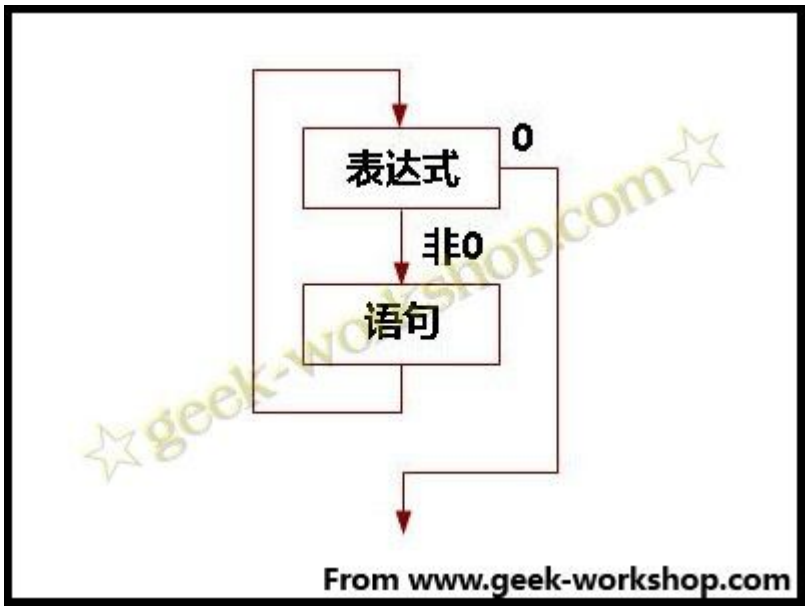
本次试验使用到了 while()函数

在 loop()中用 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）是，执行“语句”。“语句”是被循环执行的程序，称为“循环体”。

arduino 学习笔记 18 数码管实验

数码管介绍

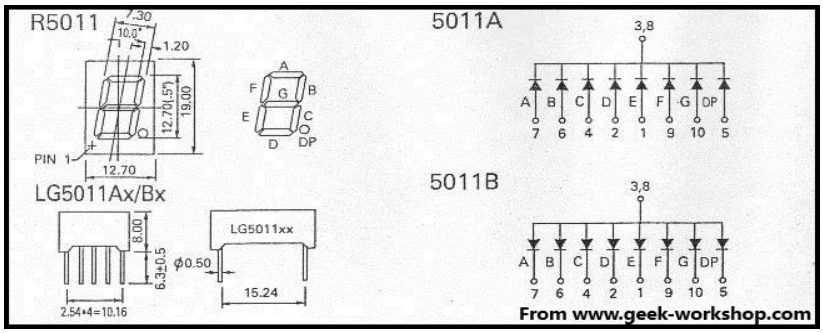
数码管是一种半导体发光器件，其基本单元是发光二极管。数码管按段数分为七段数数码管和八段数数码管，八段数数码管比七段数数码管多一个发光二极管单元（多一个小数点显示）

发光二极管单元连接方式分为共阳数码管和共阴数码管。共阳数码管是指将所有发光二极管的阳极连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 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。共阴数码管则更好相反，阴极连接到一起形成了公共阴极，阳极是独立分开的。

先来看一下本次实验使用的数码管。

通过查询型号 LG5011BSR 得知其为 0.5"单联共阳数码管，下面是其引脚图。

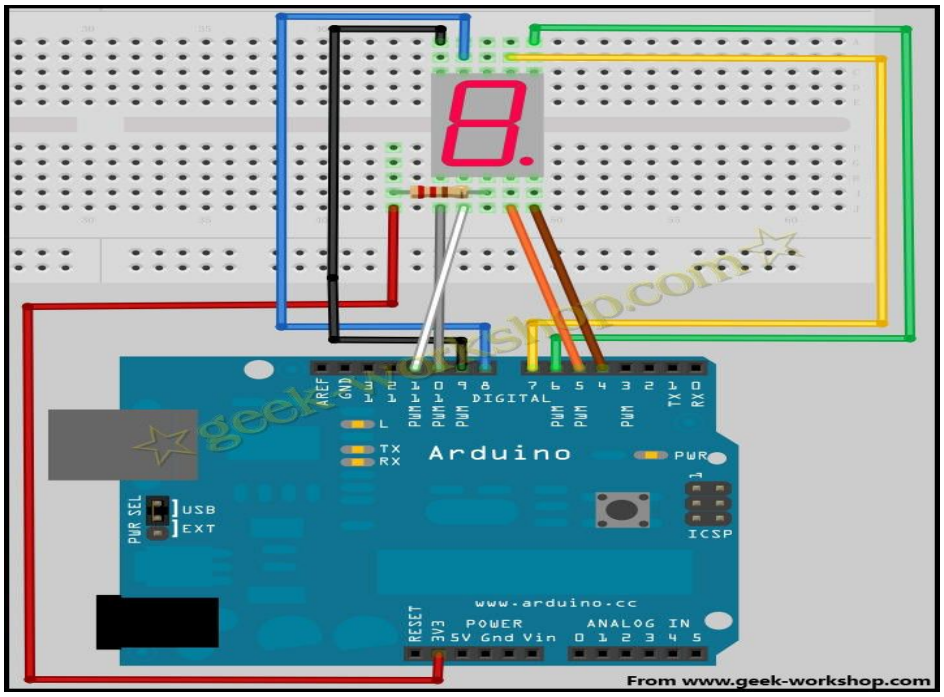


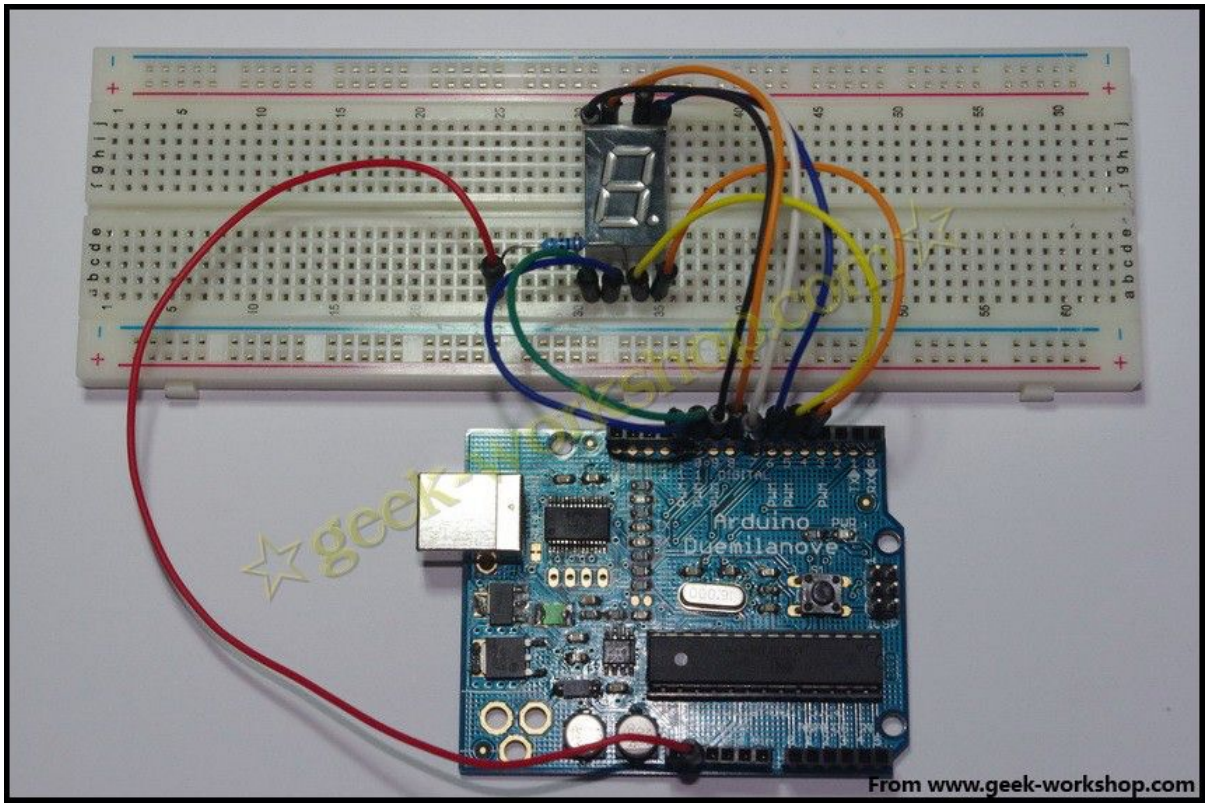
查看其背后，四个角分别有 2 个原点和 5，10 两个标记。分别表示了 1,6,5,10 针脚。

数码管和发光二极管一样，需要添加限流电阻，因为网上没有查到资料说明该数码管的击穿电压是多大。所以供给电源电压宁可小不可大，所以选择 220Ω限流电阻，和 3.3V 供电。

线路连接图如下

慧净电子---ARDUINO 模块化创新视频教程





把下面的代码编译后下载到控制板上，看看效果~

```
1.  //设置控制各段的数字 IO 脚，具体几号引脚对应哪一段，来源
    为数码管官方引脚图。
2.  int a=7;
3.  int b=6;
4.  int c=5;
5.  int d=11;
6.  int e=10;
7.  int f=8;
8.  int g=9;
9.  int dp=4;
10.
11. //显示数字 1
12. void digital_1(void)
13. {
14.   unsigned char j;
15.   digitalWrite(c,LOW);//给数字 5 引脚低电平，点亮 c 段
16.   digitalWrite(b,LOW);//点亮 b 段
17.   for(j=7;j<=11;j++)//熄灭其余段
18.     digitalWrite(j,HIGH);
19.   digitalWrite(dp,HIGH);//熄灭小数点 DP 段
20. }
21. //显示数字 2
```

```
22. void digital_2(void)
23. {
24.   unsigned char j;
25.   digitalWrite(b,LOW);
26.   digitalWrite(a,LOW);
27.   for(j=9;j<=11;j++)
28.     digitalWrite(j,LOW);
29.   digitalWrite(dp,HIGH);
30.   digitalWrite(c,HIGH);
31.   digitalWrite(f,HIGH);
32. }
33. //显示数字 3
34. void digital_3(void)
35. {
36.   unsigned char j;
37.   digitalWrite(g,LOW);
38.   digitalWrite(d,LOW);
39.   for(j=5;j<=7;j++)
40.     digitalWrite(j,LOW);
41.   digitalWrite(dp,HIGH);
42.   digitalWrite(f,HIGH);
43.   digitalWrite(e,HIGH);
```

```
44. }
45. //显示数字 4
46. void digital_4(void)
47. {
48.   digitalWrite(c,LOW);
49.   digitalWrite(b,LOW);
50.   digitalWrite(f,LOW);
51.   digitalWrite(g,LOW);
52.   digitalWrite(dp,HIGH);
53.   digitalWrite(a,HIGH);
54.   digitalWrite(e,HIGH);
55.   digitalWrite(d,HIGH);
56. }
57. //显示数字 5
58. void digital_5(void)
59. {
60.   unsigned char j;
61.   for(j=7;j<=9;j++)
62.      digitalWrite(j,LOW);
63.   digitalWrite(c,LOW);
64.   digitalWrite(d,LOW);
65.   digitalWrite(dp,HIGH);
```

```
66.   digitalWrite(b,HIGH);
67.   digitalWrite(e,HIGH);
68. }
69. //显示数字 6
70. void digital_6(void)
71. {
72.   unsigned char j;
73.   for(j=7;j<=11;j++)
74.      digitalWrite(j,LOW);
75.   digitalWrite(c,LOW);
76.   digitalWrite(dp,HIGH);
77.   digitalWrite(b,HIGH);
78. }
79. //显示数字 7
80. void digital_7(void)
81. {
82.   unsigned char j;
83.   for(j=5;j<=7;j++)
84.      digitalWrite(j,LOW);
85.   digitalWrite(dp,HIGH);
86.   for(j=8;j<=11;j++)
87.      digitalWrite(j,HIGH);
```

```
88. }
89. //显示数字 8
90. void digital_8(void)
91. {
92.   unsigned char j;
93.   for(j=5;j<=11;j++)
94.      digitalWrite(j,LOW);
95.   digitalWrite(dp,HIGH);
96. }
97. void setup()
98. {
99.   int i;//定义变量
100.  for(i=4;i<=11;i++)
101.     pinMode(i,OUTPUT);//设置 4～11 引脚为输出模式
102. }
103. void loop()
104. {
105.  while(1)
106.  {
107.     digital_1();//数字 1
108.     delay(2000);//延时 2s
109.     digital_2();
```

```
110.      delay(2000);
111.      digital_3();
112.      delay(2000);
113.      digital_4();
114.      delay(2000);
115.      digital_5();
116.      delay(2000);
117.      digital_6();
118.      delay(2000);
119.      digital_7();
120.      delay(2000);
121.      digital_8();
122.      delay(2000);
123.   }
124.}
```

复制代码

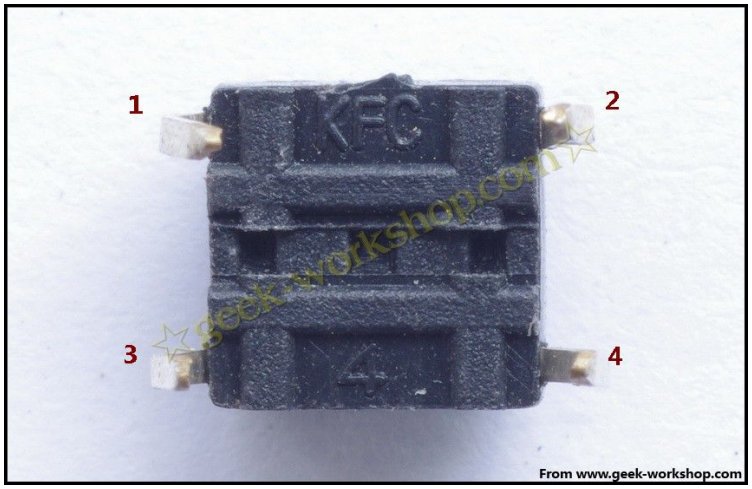
本次试验的效果为数码管 1,2,3,4,5,6,7,8 这样子循环显示。

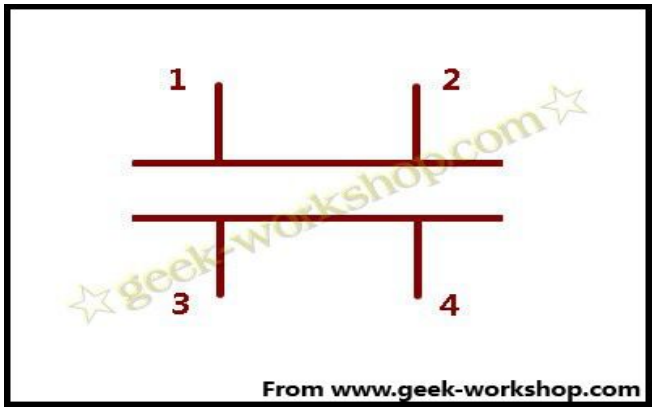
慧净电子---ARDUINO 模块化创新视频教程

arduino 学习笔记 19 按键实验

按键是一种常用的控制电器元件，常用来接通或断开电路，从而达到控制电机或者其他设备运行的开关。按键的外观多种多样，本次实验使用的是这种微型按键，6mm 的，如下图。

此种按键有 4 个脚，从背面看是这样子的。





在按键没有按下去的时候 1，2 号脚相连，3,4 号脚相连。按键按下去的时候，1,2,3,4 号脚就全部接通。

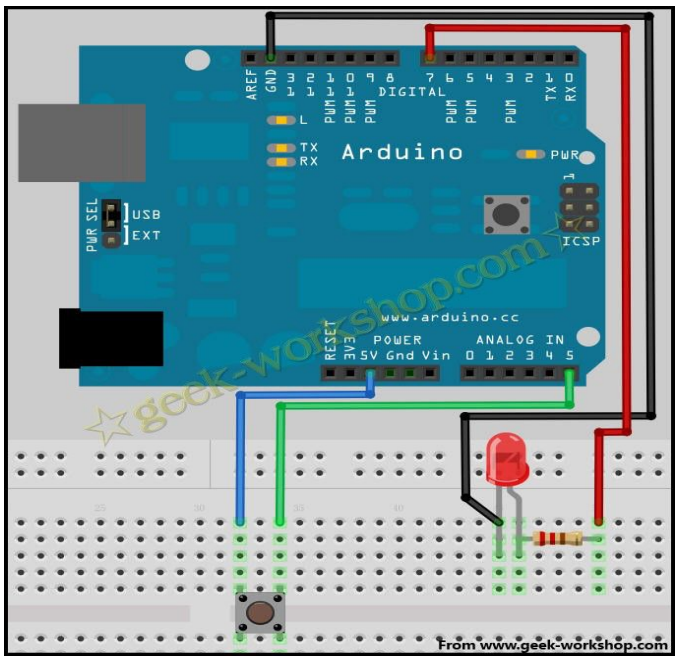
本次实验使用按键来控制 led 的亮或者灭。

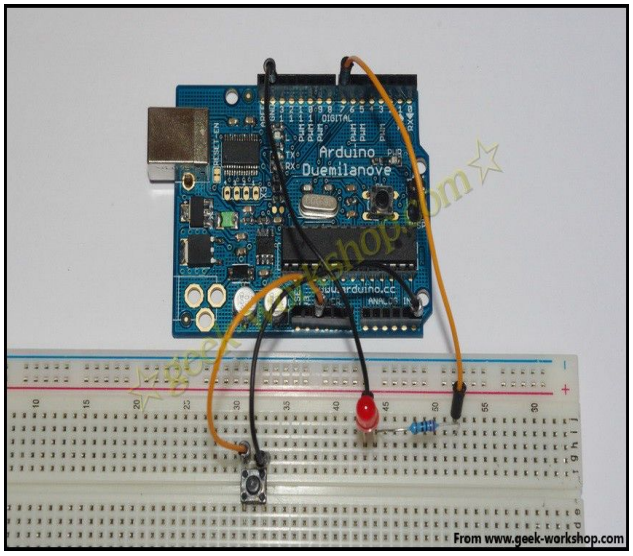
一般情况是直接把按键开关串联在 led 的电路中来开关，这种应用情况比较单一。

这次实验通过间接的方法来控制，按键接通后判断按键电路中的输出电压，如果电压大于 4.88V，就使给 LED 电路输出高电平，反之就输出低电平。使用逻辑判断的方法来控制 LED 亮或者灭，此种控制方

法应用范围较广。

本次连接方法如下图。按键开关两段一端连接 5V 接口，一端连接模拟 5 号口。LED 长针脚串联 220Ω电阻连接数字 7 号口，短针脚连接 GND。





把下面的代码上传到 arduino 控制板上，看看效果。

```
1.  int key=7;//设置 LED 的数字 IO 脚
2.  void setup()
3.  {
4.    pinMode(key,OUTPUT);//设置数字 IO 引脚为输出模式
5.  }
6.  void loop()
7.  {
8.    int i;//定义变量
9.    while(1)
10.  {
11.     i=analogRead(5);//读取模拟 5 口电压值
12.     if(i>1000)//如果电压值大于 1000（即 4.88V）
13.       digitalWrite(key,HIGH);//设置第七引脚为高电平，点亮
    led 灯
14.     else
15.       digitalWrite(key,LOW);//设置第七引脚为低电平，熄灭 led
    灯
16.  }
17. }
```

复制代码

本次实验使用到 analogRead()这个新命令。

analogRead()作用是读取模拟口的数值。默认是把 0-5V 的输入电压分成 1024 份，每一份大约为 0.049V，其数值在 0-1023 之间。

在本次程序代码中的读取数值如果大于 512 则给 LED 输出高电平，所对应的电压也就为大于 2.5V。

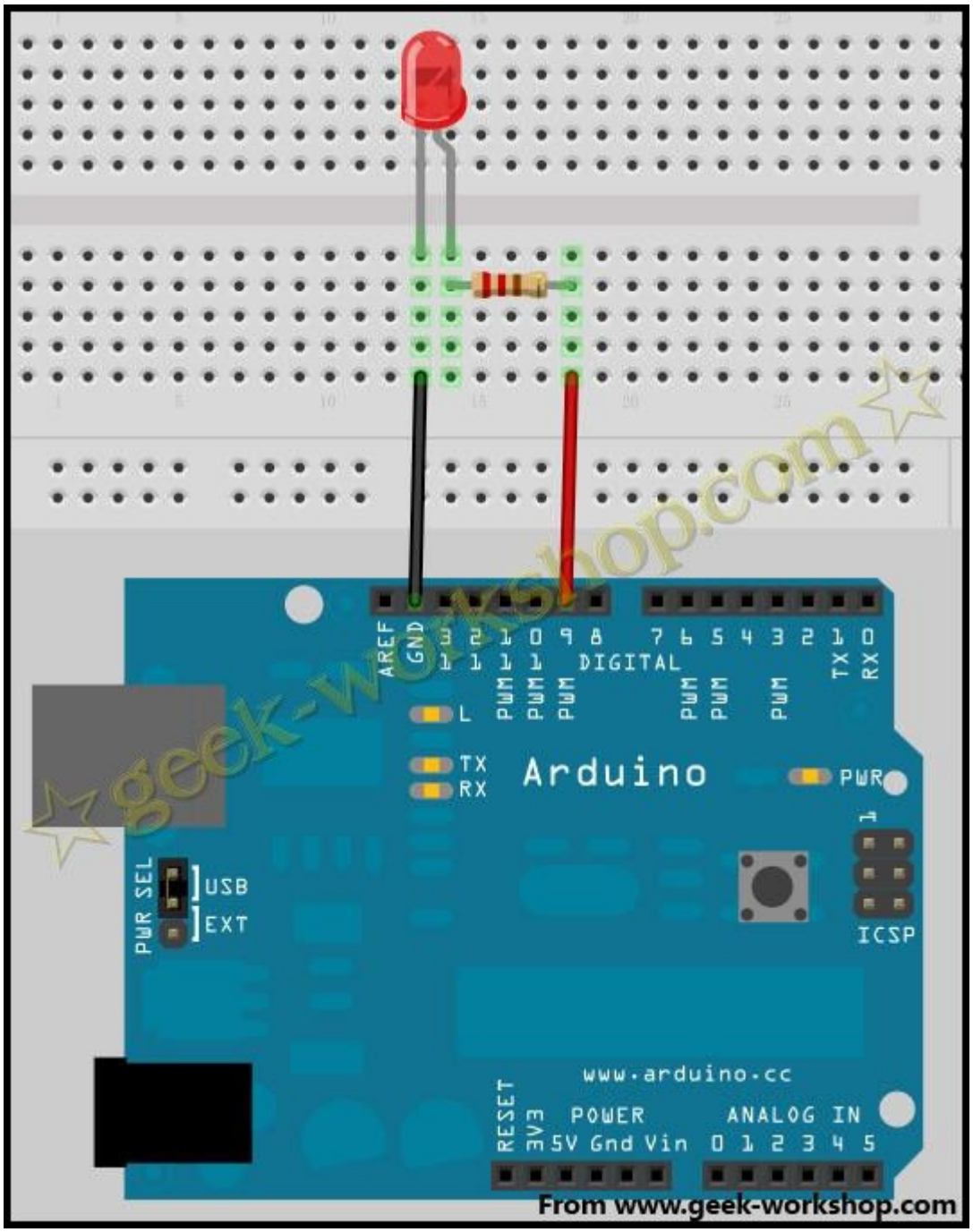
analogRead()命令输入的范围与分辨率可以使用 analogReference()命令进行改动。

刚开始本实验选用的判断标准是 512，也就是 2.5V。但是有网友按照教程的方法进行试验发现有问题，有时不需要按按钮灯就会自己亮。根据多次试验与分析后，确定其为各种干扰所致。比如感应电流等等不少都是大于 2.5V 的，所以为了提高准确度，只能提高判断的电压，本次实验就是提高到 1000（4.88V）。人体自身也带电，早中晚还个不一样。下面的实验就是把模拟 5 号口判断标准定位 512，用手去触摸模拟 5 号口导线就可以点亮 LED。

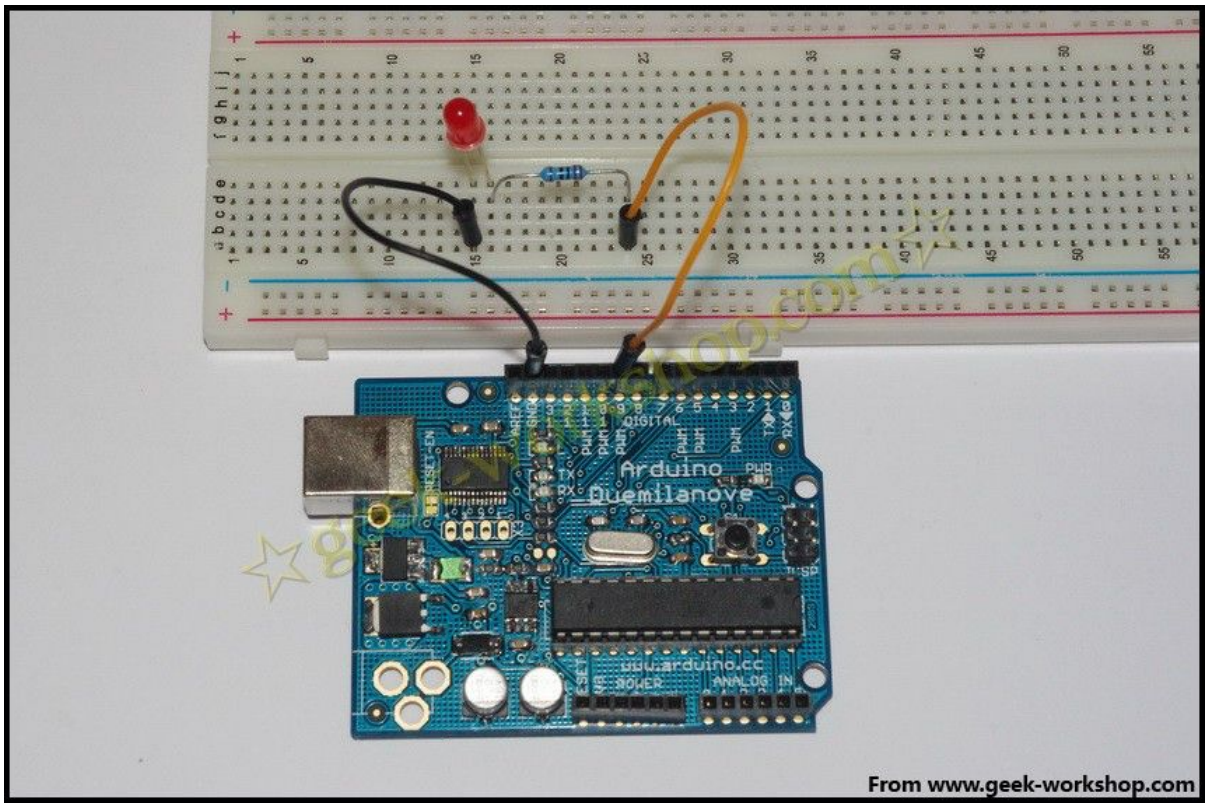
arduino 学习笔记 20 PWM 控制 LED 实验 PWM 讲解

慧净电子---ARDUINO 模块化创新视频教程

本次实验通过 PWM 来控制一盏 LED 灯，让它慢慢变亮再慢慢变暗，如此循环。下面是接线图：



慧净电子---ARDUINO 模块化创新视频教程




```
1.  /*
2.  本次实验演示如何通过 analogWrite()命令使连接 9 号端口的
    LED 灯亮度逐渐变化
3.    */
4.  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。
5.  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。
6.
7.  void setup()  {
8.
```

```
9.    pinMode(9, OUTPUT);//  设置 9 号口为输出端口:
10. }
11.
12. void loop()  {
13.
14.   analogWrite(9, brightness);//把 brightness 的值写入 9 号端
    口
15.
16.   brightness = brightness + fadeAmount;//改变 brightness 值，
    使亮度在下一次循环发生改变
17.
18.   if (brightness == 0 || brightness == 255) {
19.      fadeAmount = -fadeAmount ; //在亮度最高与最低时进行
    翻转
20.   }
21.
22.   delay(30); //延时 30 毫秒
23. }
```

复制代码

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.

在使用 ATmega168 与 ATmega328 的 arduino 控制板上，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。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。

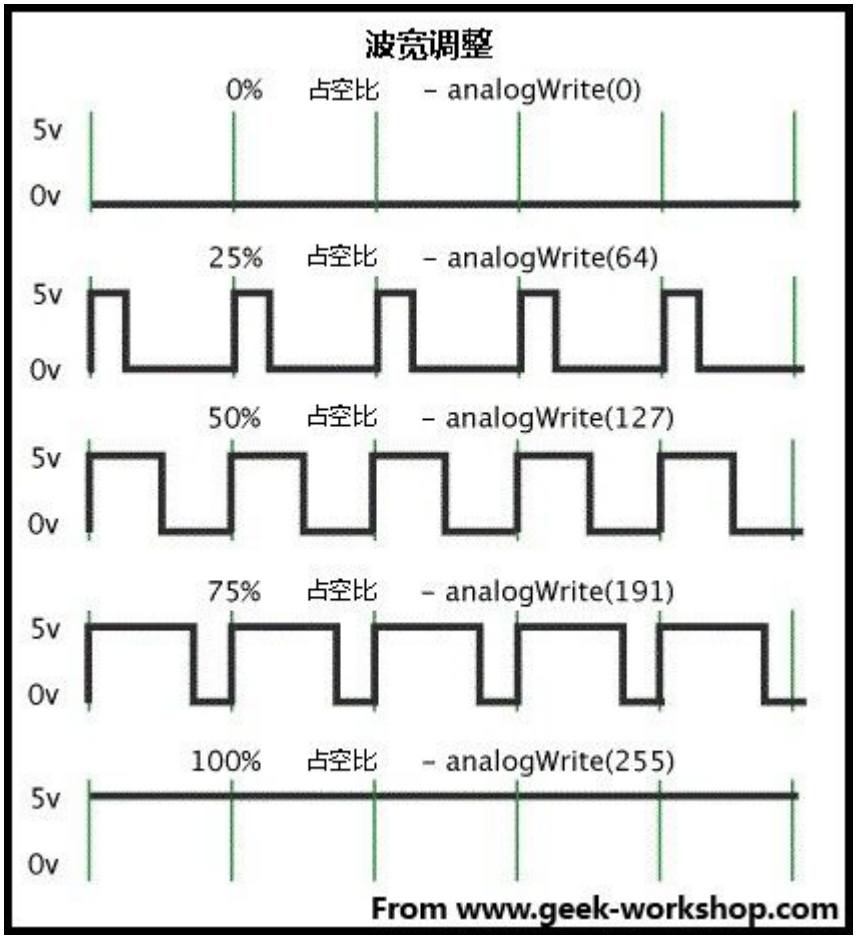
PWM（Pulse-width modulation）脉宽调制

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。额~~这个说的太专业了，还是说的通俗点。

以本次实验来看，端口的输入电压只有两个 0V 与 5V。如我我想要 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。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，咱本把 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，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）。



传统方法实现 PWM

除了使用 analogWrite()命令实现 PWM，还可以通过传统方法来控制电平的开关时间来设置。

请看如下代码

```
1.  void setup()
2.  {
3.    pinMode(13, OUTPUT);//设定 13 号端口为输出
4.  }
5.
6.  void loop()
7.  {
8.    digitalWrite(13, HIGH);
9.    delayMicroseconds(100); //  大约 10%占空比的 1KHz 方波
10.   digitalWrite(13, LOW);
11.   delayMicroseconds(900);
12. }
```

复制代码

这种方法的的优点是他可以使用任意数字端口做输出端口。而且可以自己随意设定占空比与频率。一个主要的缺点是任何中断都会影响时

钟，这样就会导致很大的抖动，除非你禁用中断。第二个却就是 CPU 在处理输出的时候，就无法做其他事情了。

上面的代码用到了一个新的命令

delayMicroseconds()

其作用是产生一个延时，计量单位是微秒，1000 微秒=1 毫秒。目前 delayMicroseconds()最大值为 16383。如果值大于 1000，推荐使用 delay() 命令。

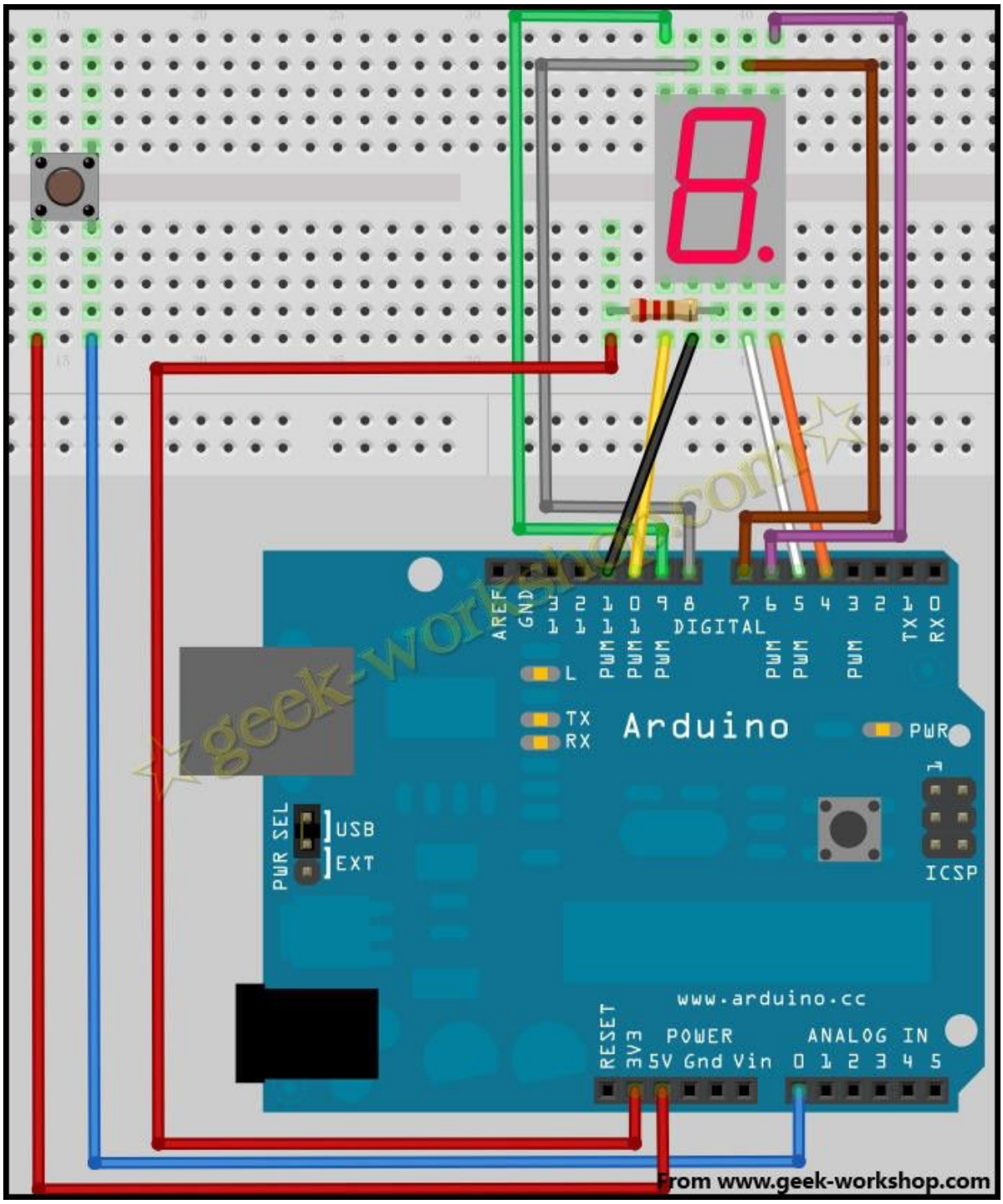
arduino 学习笔记 21 数字骰子实验

前几次做了数码管实验和按键控制 LED 的实验，通过实验大家已经学会了两种器件的基本用法。这次使用数码管与按键进行组合，做一个简易数字骰子。

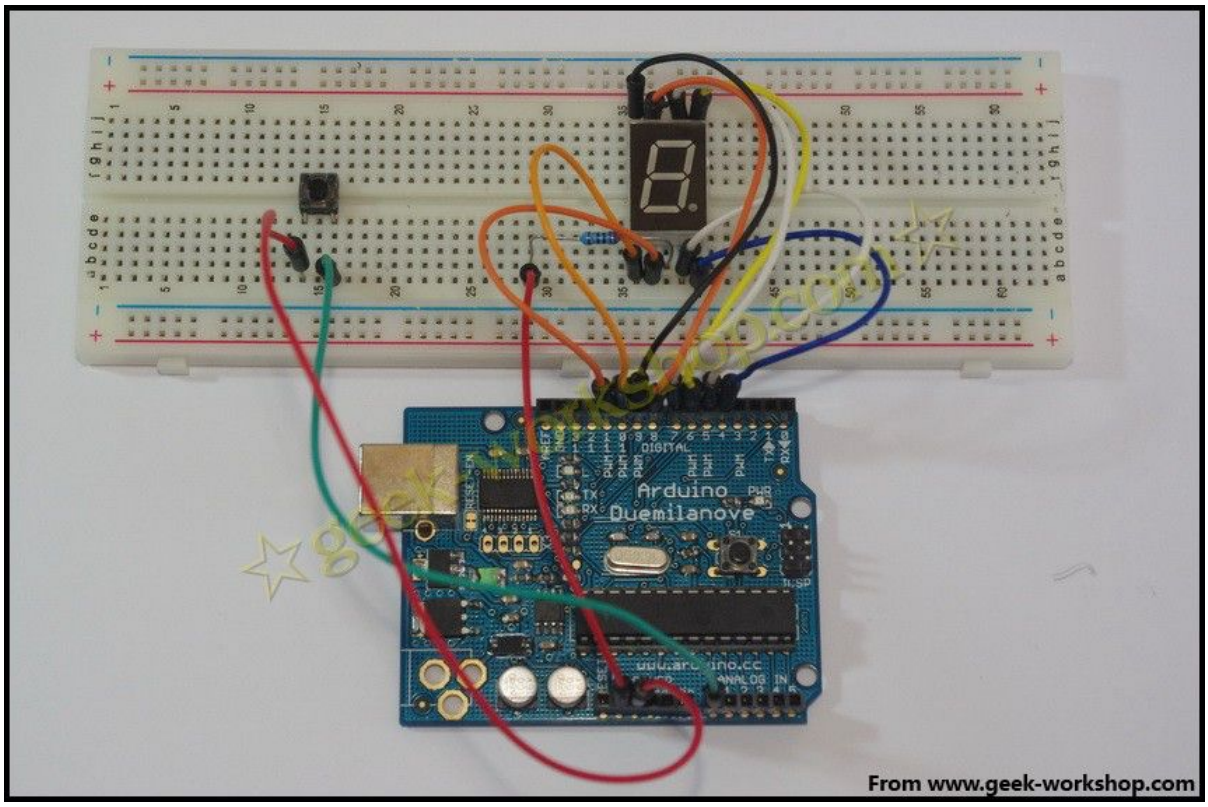
数字筛子的基本原理是数码管快速在 1-8 之间循环跳动，当按下按键时，数码管定格在当前的数字上，不再跳动。

先看一下接线图

慧净电子---ARDUINO 模块化创新视频教程



慧净电子---ARDUINO 模块化创新视频教程




```
1.  //设置控制各段的数字 IO 脚
2.  int a=7;
3.  int b=6;
4.  int c=5;
5.  int d=11;
6.  int e=10;
7.  int f=8;
8.  int g=9;
9.  int dp=4;
10.
11. //显示数字 1
12. void digital_1(void)
```

```
13. {
14.   unsigned char j;
15.   digitalWrite(c,LOW);//给数字 5 引脚低电平，点亮 c 段
16.   digitalWrite(b,LOW);//点亮 b 段
17.   for(j=7;j<=11;j++)//熄灭其余段
18.     digitalWrite(j,HIGH);
19.   digitalWrite(dp,HIGH);//熄灭小数点 DP 段
20. }
21. //显示数字 2
22. void digital_2(void)
23. {
24.   unsigned char j;
25.   digitalWrite(b,LOW);
26.   digitalWrite(a,LOW);
27.   for(j=9;j<=11;j++)
28.     digitalWrite(j,LOW);
29.   digitalWrite(dp,HIGH);
30.   digitalWrite(c,HIGH);
31.   digitalWrite(f,HIGH);
32. }
33. //显示数字 3
34. void digital_3(void)
```

```
35. {
36.   unsigned char j;
37.   digitalWrite(g,LOW);
38.   digitalWrite(d,LOW);
39.   for(j=5;j<=7;j++)
40.      digitalWrite(j,LOW);
41.   digitalWrite(dp,HIGH);
42.   digitalWrite(f,HIGH);
43.   digitalWrite(e,HIGH);
44. }
45. //显示数字 4
46. void digital_4(void)
47. {
48.   digitalWrite(c,LOW);
49.   digitalWrite(b,LOW);
50.   digitalWrite(f,LOW);
51.   digitalWrite(g,LOW);
52.   digitalWrite(dp,HIGH);
53.   digitalWrite(a,HIGH);
54.   digitalWrite(e,HIGH);
55.   digitalWrite(d,HIGH);
56. }
```

```
57. //显示数字 5
58. void digital_5(void)
59. {
60.   unsigned char j;
61.   for(j=7;j<=9;j++)
62.      digitalWrite(j,LOW);
63.   digitalWrite(c,LOW);
64.   digitalWrite(d,LOW);
65.   digitalWrite(dp,HIGH);
66.   digitalWrite(b,HIGH);
67.   digitalWrite(e,HIGH);
68. }
69. //显示数字 6
70. void digital_6(void)
71. {
72.   unsigned char j;
73.   for(j=7;j<=11;j++)
74.      digitalWrite(j,LOW);
75.   digitalWrite(c,LOW);
76.   digitalWrite(dp,HIGH);
77.   digitalWrite(b,HIGH);
78. }
```

```
79. //显示数字 7
80. void digital_7(void)
81. {
82.   unsigned char j;
83.   for(j=5;j<=7;j++)
84.      digitalWrite(j,LOW);
85.   digitalWrite(dp,HIGH);
86.   for(j=8;j<=11;j++)
87.      digitalWrite(j,HIGH);
88. }
89. //显示数字 8
90. void digital_8(void)
91. {
92.   unsigned char j;
93.   for(j=5;j<=11;j++)
94.      digitalWrite(j,LOW);
95.   digitalWrite(dp,HIGH);
96. }
97. void setup()
98. {
99.      int i;
100.     for(i=4;i<=11;i++)
```

```
101.     {
102.     pinMode(i,OUTPUT);//设置 4~11 口为输出模式
103.     }
104.}
105.void loop()
106.{
107.    while(1)
108.    {
109.         digital_1();//显示数字 1
110.         while(analogRead(0)>1000);//如果读到模拟 0 口的值
    1000 则说明有按键按下
111.         delay(100);//延时 200ms
112.         digital_2();
113.         while(analogRead(0)>1000);
114.         delay(100);
115.         digital_3();
116.         while(analogRead(0)>1000);
117.         delay(100);
118.         digital_4();
119.         while(analogRead(0)>1000);
120.         delay(100);
121.         digital_5();
```

```
122.          while(analogRead(0)>1000);
123.          delay(100);
124.          digital_6();
125.          while(analogRead(0)>1000);
126.          delay(100);
127.          digital_7();
128.          while(analogRead(0)>1000);
129.          delay(100);
130.          digital_8();
131.          while(analogRead(0)>1000);
132.          delay(100);
133.    }
134.}
```

复制代码

arduino 学习笔记 22 光控 LED 实验

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。

光敏电阻的工作原理基于内光电效应。在半导体光敏材料的两端装上电极引线，将其封装在带有透明窗的管壳里就构成光敏电阻，为了增加灵敏度，两电极常做成梳状。在有光照射时，射入的光强，电阻减小，射入的光弱，电阻增大。

下图就是一个光敏电阻

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。

实验前先测量一下当前环境下光敏电阻的亮阻值与暗阻值

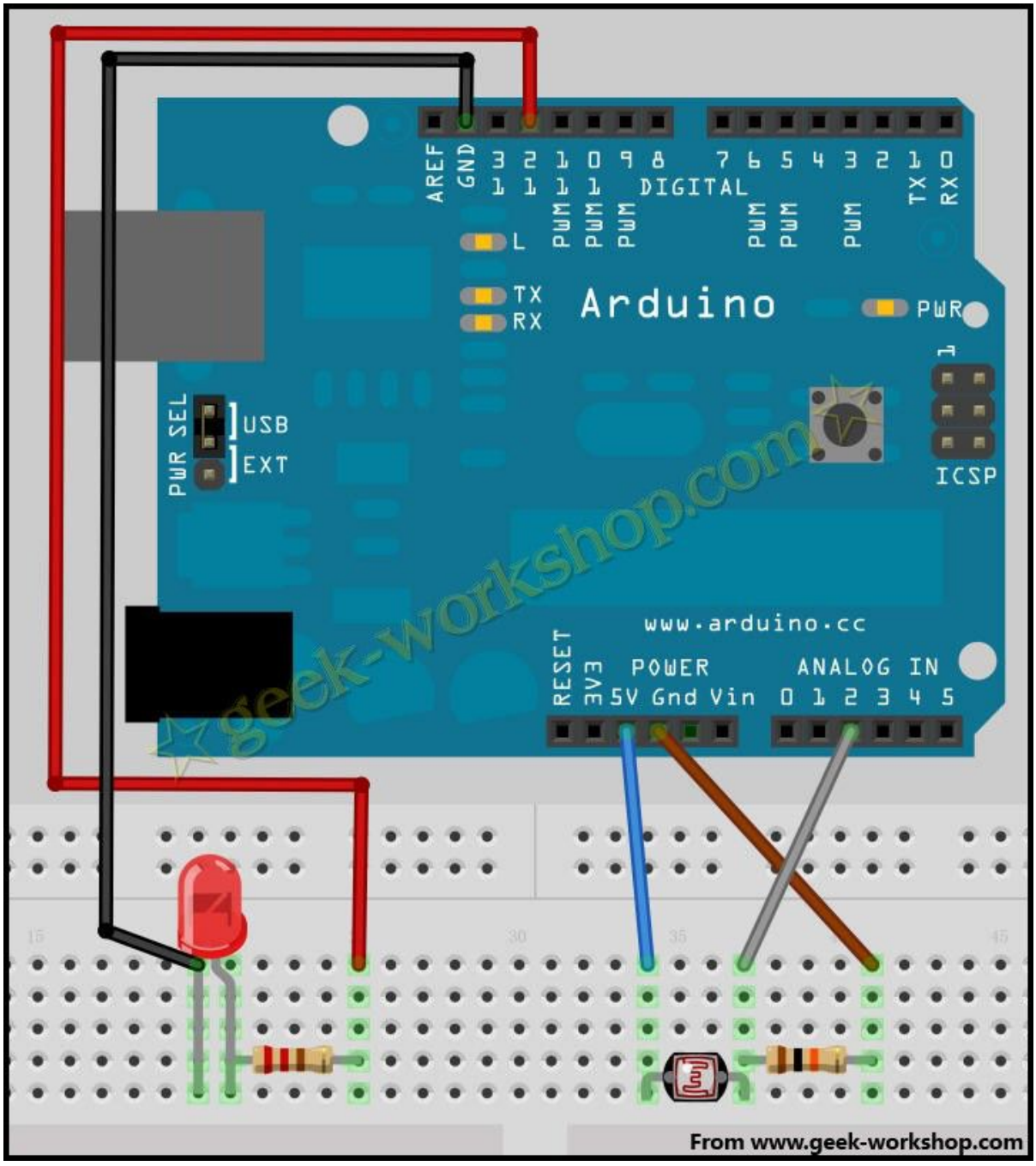
下图是测出来的 LED 亮阻值，为 9.1KΩ

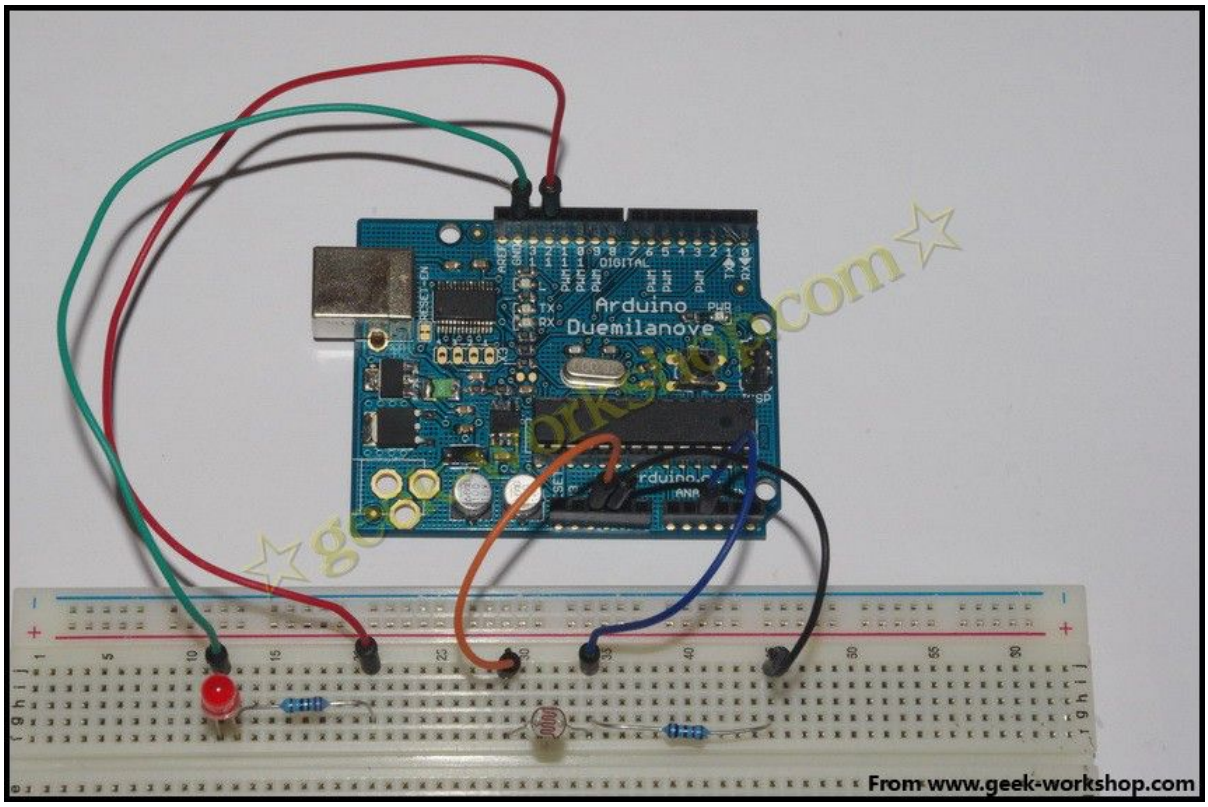
下图是测出来的 LED 暗阻值，为 32.4KΩ

慧净电子---ARDUINO 模块化创新视频教程

硬件连接图如下

慧净电子---ARDUINO 模块化创新视频教程





根据测出来的亮阻 9.1KΩ，暗阻 32.4 欧姆。选定分压电阻为 10KΩ。因为当有遮挡物的后，阻值会变大。假设亮阻为 10KΩ（对于光敏电阻来说，与测量出来的 9.1KΩ差别不大，计算起来更加方便了），分压阻值为 10K 欧姆。模拟 2 号口所测量的触发电压为 10KΩ分压电阻的，在 5V 电源供电下，亮与暗转换的触发电压为 5×10÷（10+10）=2.5V。当光线越暗，光敏电阻的阻值也就越大，分压两端电压也就越小。所以触发条件就为≤2.5V。（不同光照条件下触发电压不同，请根据实验环境进行调整。）

程序代码如下

```
1.  int photocellPin = 2;     //定义变量 photocellsh=2，为电压读
    取端口。
2.  int ledPin = 12;    //定义变量 ledPin=12，为 led 电平输出端口
3.  int val = 0;        //定义 val 变量的起始值
4.
5.
6.  void setup() {
7.    pinMode(ledPin, OUTPUT);  //使 ledPin 为输出模式
8.  }
9.
10. void loop() {
11.   val = analogRead(photocellPin);     //从传感器读取值
12.   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。
13.     digitalWrite(ledPin, HIGH); //当 val 小于 512(2.5V)的时候，
    led 亮。
14.   }
15.   else{
16.     digitalWrite(ledPin, LOW);
17.   }
18. }
```

复制代码

arduino 学习笔记 23 1602 液晶实验

本次试验使用 arduino 直接驱动 1602 液晶显示文字

1602 液晶在应用中非常广泛，最初的 1602 液晶使用的是 HD44780 控制器，现在各个厂家的 1602 模块基本上都是采用了与之兼容的 IC，所以特性上基本都是一致的。

1602LCD 主要技术参数

显示容量为 16×2 个字符；

芯片工作电压为 4.5～5.5V；

工作电流为 2.0mA（5.0V）；

模块最佳工作电压为 5.0V；

字符尺寸为 2.95×4.35（W×H）mm。

1602 液晶接口引脚定义

编号	符号	引脚说明	编号	符号	引脚说明
1	VSS	电源地	9	D2	Date I/O
2	VDD	电源正极	10	D3	Date I/O
3	VL	液晶显示偏压信号	11	D4	Date I/O
4	RS	数据/命令选择端（V/L）	12	D5	Date I/O
5	R/W	读/写选择端（H/L）	13	D6	Date I/O
6	E	使能信号	14	D7	Date I/O
7	D0	Date I/O	15	BLA	背光源正极
8	D1	Date I/O	16	BLK	背光源负极

From www.geek-workshop.com

接口说明：

1、两组电源 一组是模块的电源 一组是背光板的电源 一般均使用 5V 供电。本次试验背光使用 3.3V 供电也可以工作。

2、VL 是调节对比度的引脚，串联不大于 5KΩ的电位器进行调节。本次实验使用 1KΩ的电阻来设定对比度。其连接分高电位与低电位接法，本次使用低电位接法，串联 1KΩ电阻后接 GND。

3、RS 是很多液晶上都有的引脚 是命令/数据选择引脚 该脚电平为高时表示将进行数据操作；为低时表示进行命令操作。

4、RW 也是很多液晶上都有的引脚 是读写选择端 该脚电平为高是表示要对液晶进行读操作；为低时表示要进行写操作。

5、E 同样很多液晶模块有此引脚 通常在总线上信号稳定后给一正脉冲通知把数据读走，在此脚为高电平的时候总线不允许变化。

6、D0—D7 8 位双向并行总线，用来传送命令和数据。

7、BLA 是背光源正极，BLK 是背光源负极。

1602 液晶的基本操作分以下四种：

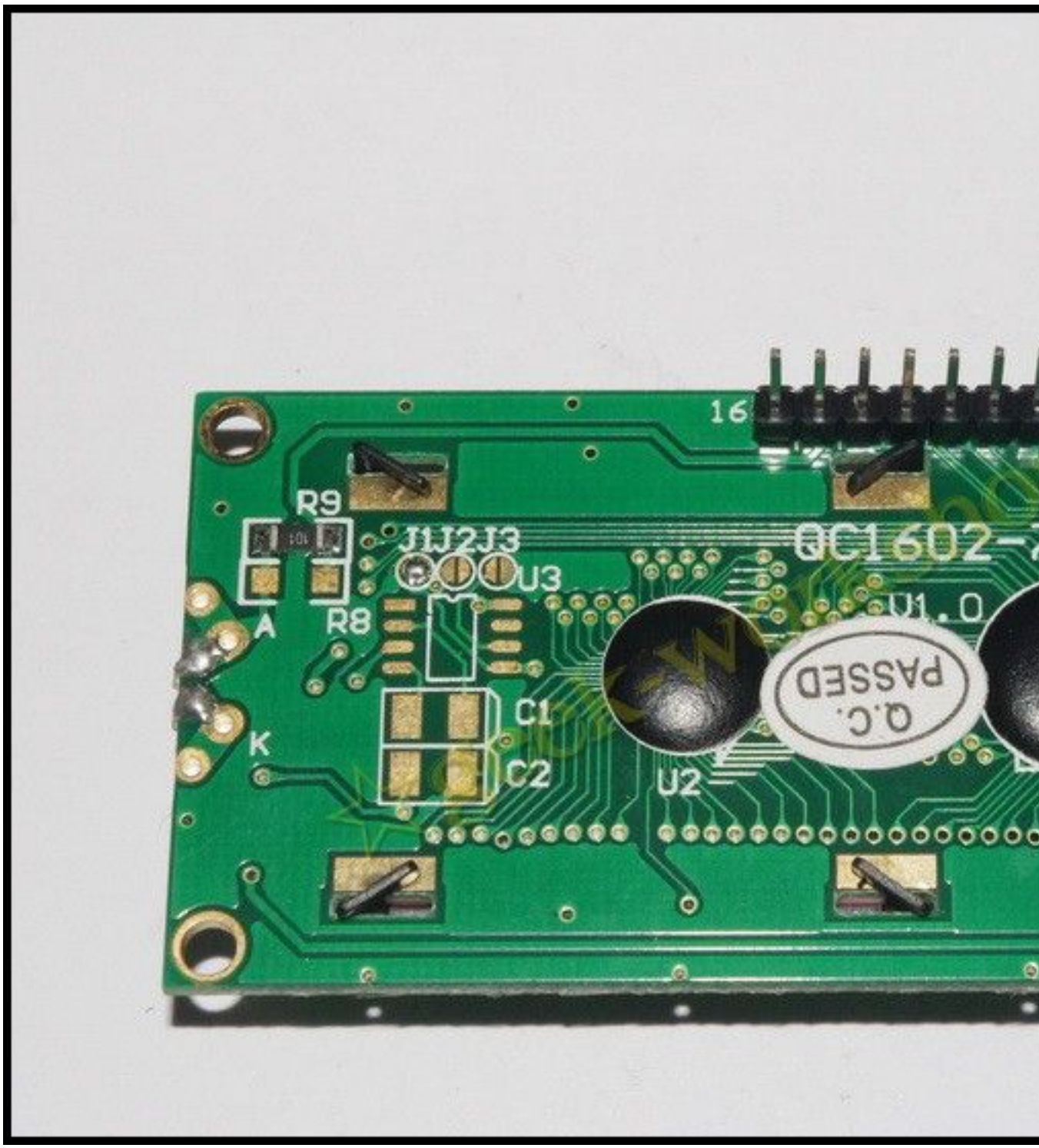
读状态	输入	RS=L，R/W=H，E=H	输出	D0~
写指令	输入	RS=L，R/W=L，D0～D7=指令码，E=高脉冲	输出	无
读数据	输入	RS=H，R/W=H，E=H	输出	D0~
写数据	输入	RS=H，R/W=L，D0～D7=数据，E=高脉冲	输出	无

Fron

下图就是 1602 液晶实物图

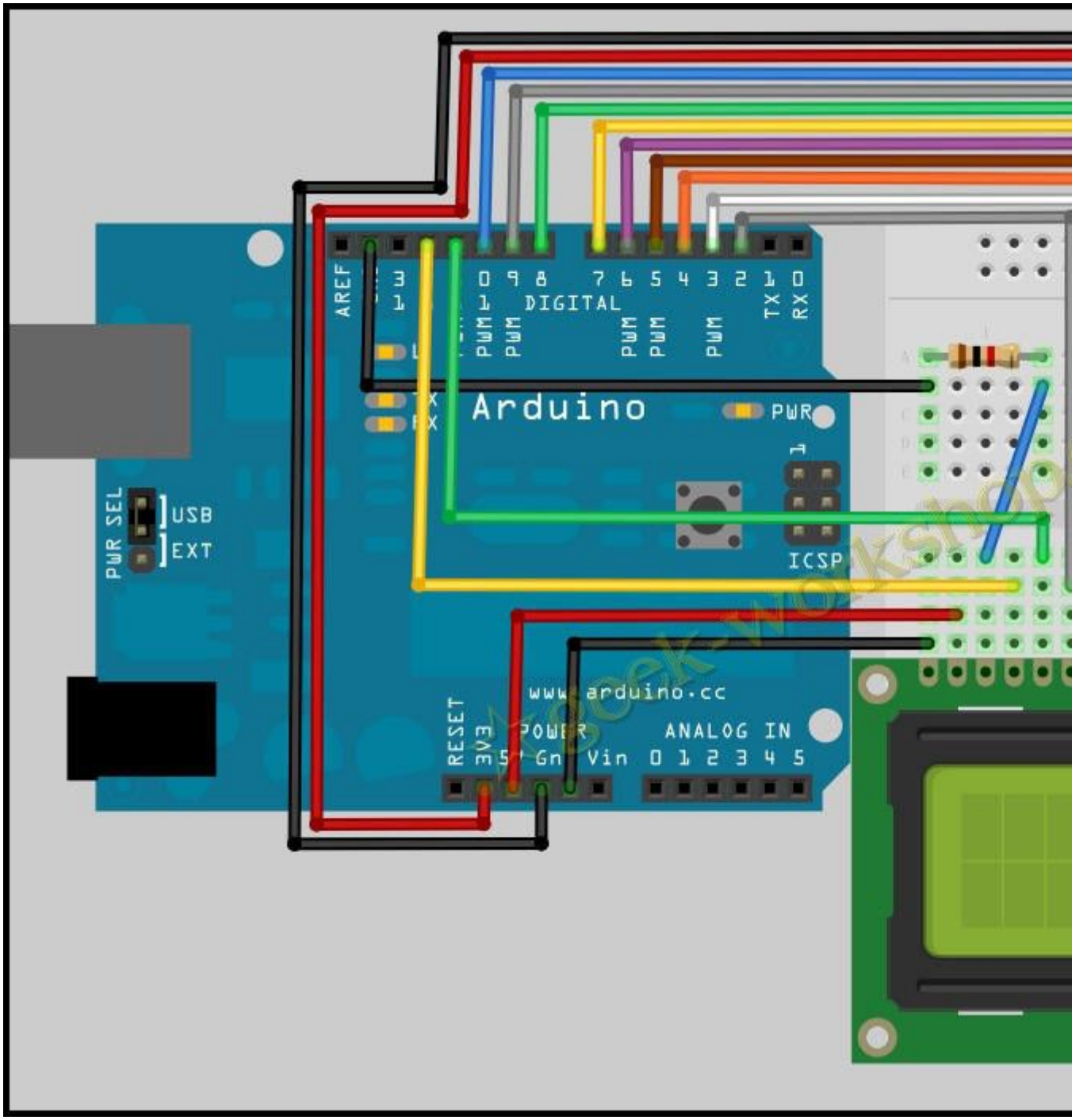
慧净电子---ARDUINO 模块化创新视频教程

慧净电子---ARDUINO 模块化创新视频教程

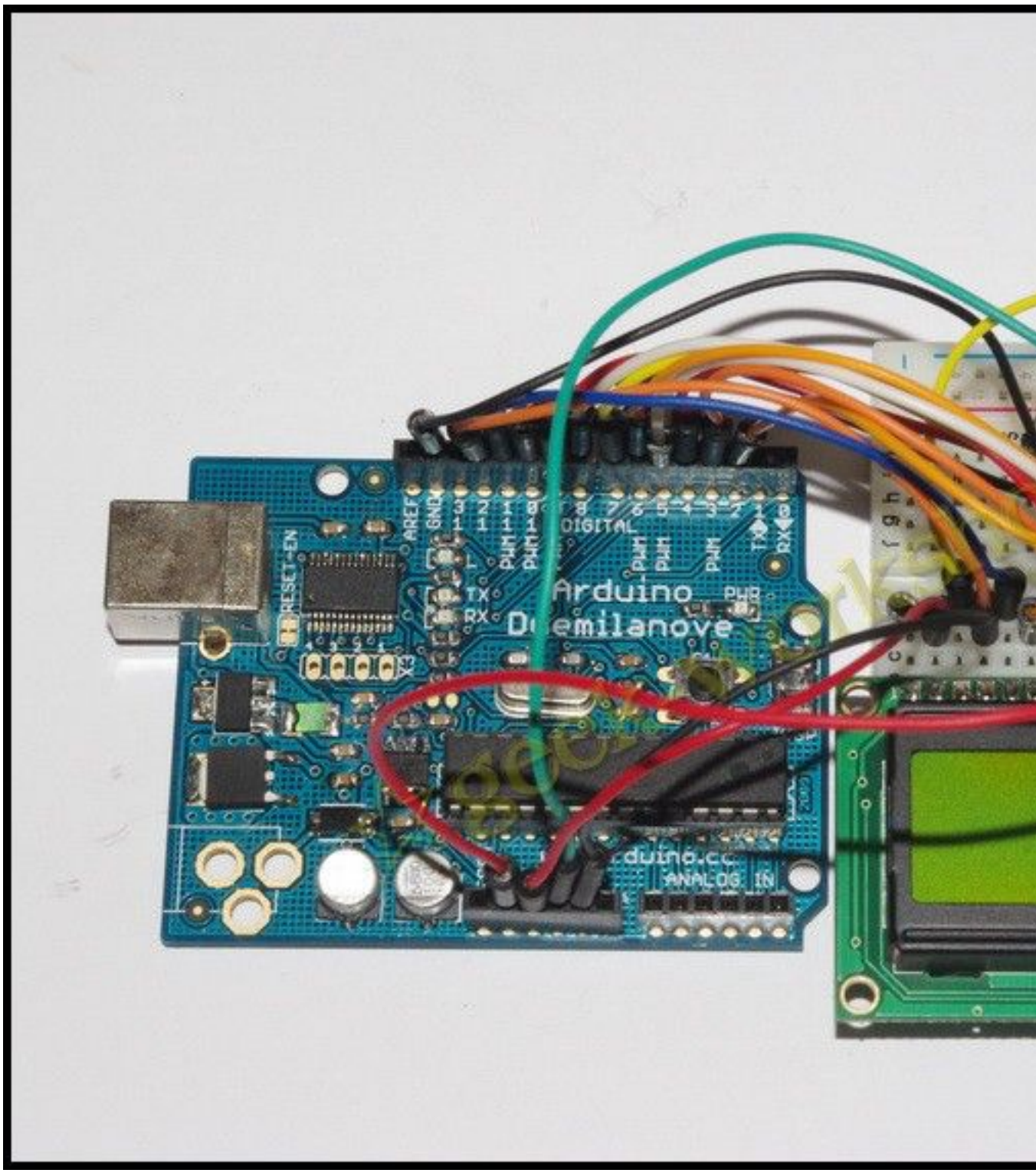


1602 直接与 arduino 通信，根据产品手册描述，分 8 位连接法与 4 位连接法，咱们先使用 8 位连接法进行实验。硬件连接方式如下图

慧净电子---ARDUINO 模块化创新视频教程



慧净电子---ARDUINO 模块化创新视频教程



代码如下

```
1.  int DI = 12;
2.  int RW = 11;
3.  int DB[] = {3, 4, 5, 6, 7, 8, 9, 10};//使用数组来定义总
    线需要的管脚
4.  int Enable = 2;
5.
6.  void LcdCommandWrite(int value) {
7.  // 定义所有引脚
8.  int i = 0;
9.  for (i=DB[0]; i <= DI; i++) //总线赋值
10. {
11.      digitalWrite(i,value & 01);//因为 1602 液晶信号识别
    是 D7-D0(不是 D0-D7)，这里是用来反转信号。
12.      value >>= 1;
13. }
14. digitalWrite(Enable,LOW);
15. delayMicroseconds(1);
16. digitalWrite(Enable,HIGH);
17. delayMicroseconds(1);    // 延时 1ms
18. digitalWrite(Enable,LOW);
19. delayMicroseconds(1);    // 延时 1ms
20. }
```

```
21.
22. void LcdDataWrite(int value) {
23. // 定义所有引脚
24. int i = 0;
25. digitalWrite(DI, HIGH);
26. digitalWrite(RW, LOW);
27. for (i=DB[0]; i <= DB[7]; i++) {
28.       digitalWrite(i,value & 01);
29.       value >>= 1;
30. }
31. digitalWrite(Enable,LOW);
32. delayMicroseconds(1);
33. digitalWrite(Enable,HIGH);
34. delayMicroseconds(1);
35. digitalWrite(Enable,LOW);
36. delayMicroseconds(1);    // 延时 1ms
37. }
38.
39. void setup (void) {
40. int i = 0;
41. for (i=Enable; i <= DI; i++) {
42.       pinMode(i,OUTPUT);
```

```
43. }
44. delay(100);
45. // 短暂的停顿后初始化 LCD
46. // 用于 LCD 控制需要
47. LcdCommandWrite(0x38);     // 设置为 8-bit 接口,2 行显示,
    5x7 文字大小
48. delay(64);
49. LcdCommandWrite(0x38);     // 设置为 8-bit 接口,2 行显示,
    5x7 文字大小
50. delay(50);
51. LcdCommandWrite(0x38);     // 设置为 8-bit 接口,2 行显示,
    5x7 文字大小
52. delay(20);
53. LcdCommandWrite(0x06);     // 输入方式设定
54.                                          // 自动增量,
    没有显示移位
55. delay(20);
56. LcdCommandWrite(0x0E);     // 显示设置
57.                                          // 开启显示
    屏，光标显示，无闪烁
58. delay(20);
59. LcdCommandWrite(0x01);     // 屏幕清空,光标位置归零
```

```
60. delay(100);
61. LcdCommandWrite(0x80);     // 显示设置
62.                                                  // 开启显示
    屏，光标显示，无闪烁
63. delay(20);
64. }
65.
66. void loop (void) {
67.     LcdCommandWrite(0x01);     // 屏幕清空，光标位置归
    零
68.     delay(10);
69.     LcdCommandWrite(0x80+3);
70.     delay(10);
71.     // 写入欢迎信息
72.     LcdDataWrite('W');
73.     LcdDataWrite('e');
74.     LcdDataWrite('l');
75.     LcdDataWrite('c');
76.     LcdDataWrite('o');
77.     LcdDataWrite('m');
78.     LcdDataWrite('e');
79.     LcdDataWrite(' ');
```

```
80.     LcdDataWrite('t');
81.     LcdDataWrite('o');
82.     delay(10);
83.     LcdCommandWrite(0xc0+1);     // 定义光标位置为第二
   行第二个位置
84.     delay(10);
85.     LcdDataWrite('g');
86.     LcdDataWrite('e');
87.     LcdDataWrite('e');
88.     LcdDataWrite('k');
89.     LcdDataWrite('-');
90.     LcdDataWrite('w');
91.     LcdDataWrite('o');
92.     LcdDataWrite('r');
93.     LcdDataWrite('k');
94.     LcdDataWrite('s');
95.     LcdDataWrite('h');
96.     LcdDataWrite('o');
97.     LcdDataWrite('p');
98.     delay(5000);
99.     LcdCommandWrite(0x01);     // 屏幕清空，光标位置归
   零
```

```
100.     delay(10);
101.     LcdDataWrite('I');
102.     LcdDataWrite(' ');
103.     LcdDataWrite('a');
104.     LcdDataWrite('m');
105.     LcdDataWrite(' ');
106.     LcdDataWrite('h');
107.     LcdDataWrite('o');
108.     LcdDataWrite('n');
109.     LcdDataWrite('g');
110.     LcdDataWrite('y');
111.     LcdDataWrite('i');
112.     delay(3000);
113.     LcdCommandWrite(0x02); //设置模式为新文字替换老文
   字，无新文字的地方显示不变。
114.     delay(10);
115.     LcdCommandWrite(0x80+5); //定义光标位置为第一行第
   六个位置
116.     delay(10);
117.     LcdDataWrite('t');
118.     LcdDataWrite('h');
119.     LcdDataWrite('e');
```

```
120.     LcdDataWrite(' ');
121.     LcdDataWrite('a');
122.     LcdDataWrite('d');
123.     LcdDataWrite('m');
124.     LcdDataWrite('i');
125.     LcdDataWrite('n');
126.     delay(5000);
127.}
```

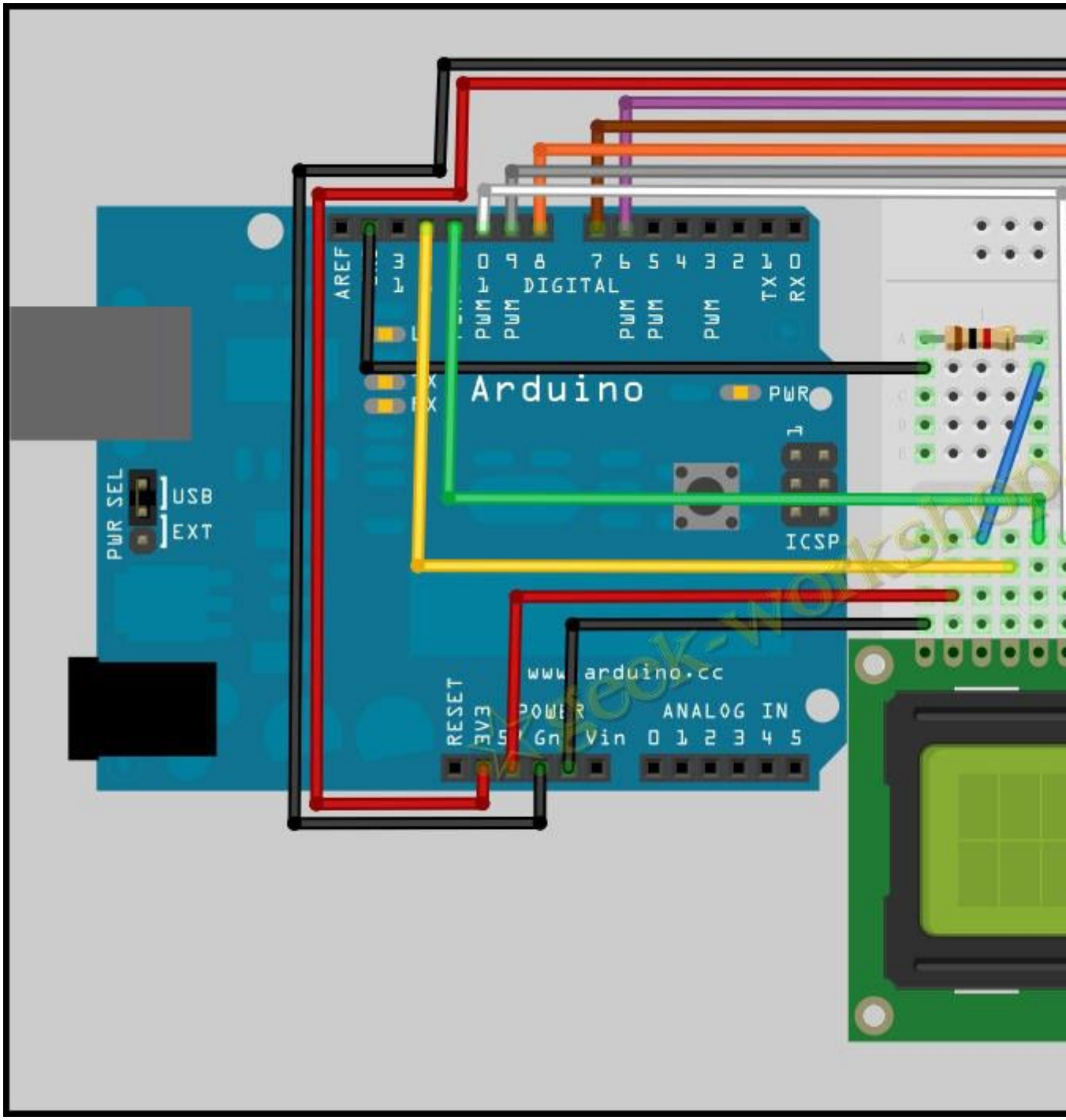
复制代码

4 位接法

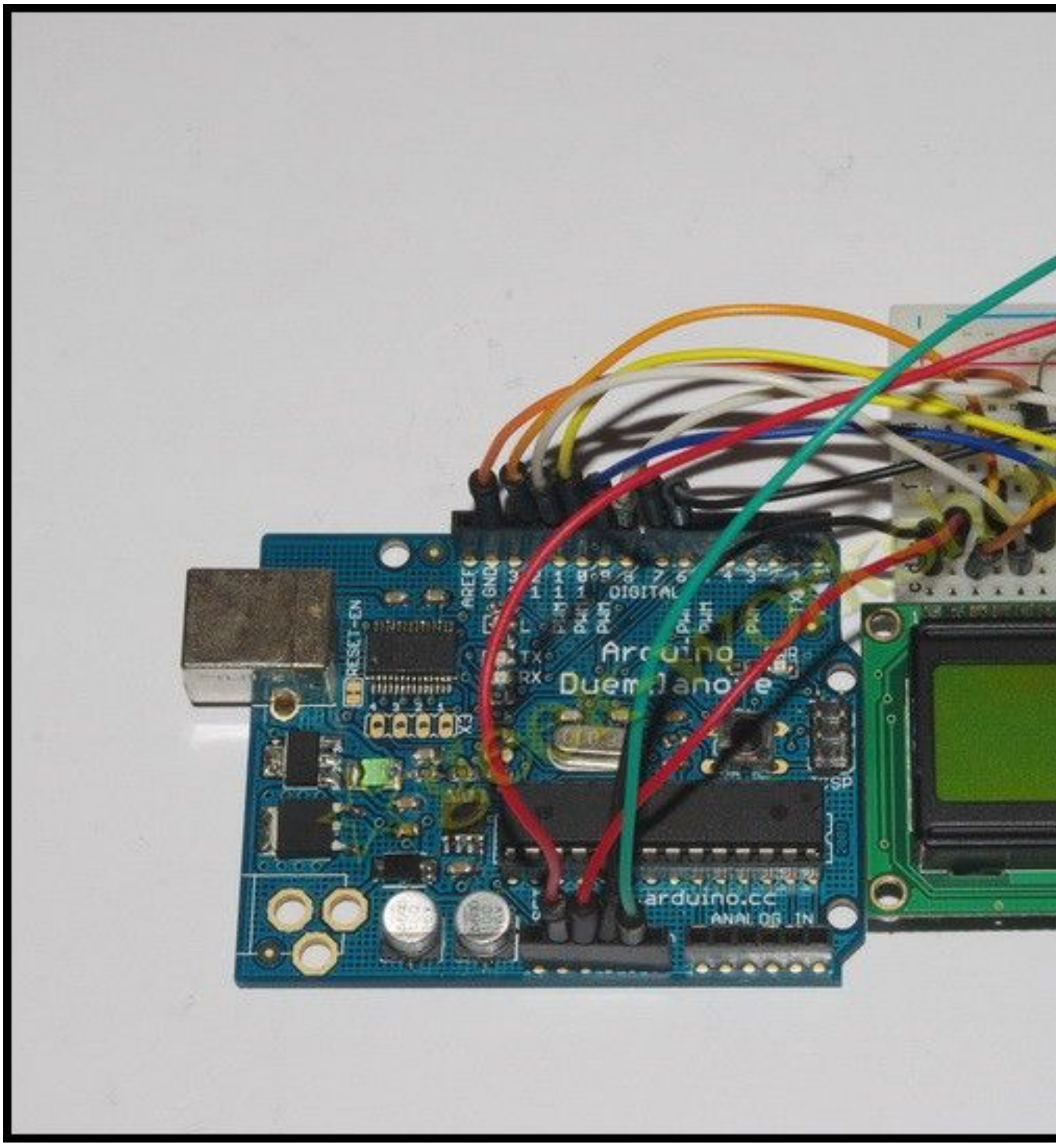
在正常使用下，8 位接法基本把 arduino 的数字端口占满了，如果想要多接几个传感器就没有端口了，这种情况下怎么处理呢，咱们可以使用 4 位接法。

4 位接法的硬件连接方法如下图

慧净电子---ARDUINO 模块化创新视频教程



慧净电子---ARDUINO 模块化创新视频教程



硬件接好后把下面的代码上传到控制板上，看看效果。

```
1.  int LCD1602_RS=12;
```

```
2.  int LCD1602_RW=11;
3.  int LCD1602_EN=10;
4.  int DB[] = { 6, 7, 8, 9};
5.  char str1[]="Welcome to";
6.  char str2[]="geek-workshop";
7.  char str3[]="this is the";
8.  char str4[]="4-bit interface";
9.
10. void LCD_Command_Write(int command)
11. {
12. int i,temp;
13. digitalWrite( LCD1602_RS,LOW);
14. digitalWrite( LCD1602_RW,LOW);
15. digitalWrite( LCD1602_EN,LOW);
16.
17. temp=command & 0xf0;
18. for (i=DB[0]; i <= 9; i++)
19. {
20.      digitalWrite(i,temp & 0x80);
21.      temp <<= 1;
22. }
23.
```

```
24. digitalWrite( LCD1602_EN,HIGH);
25. delayMicroseconds(1);
26. digitalWrite( LCD1602_EN,LOW);
27.
28. temp=(command & 0x0f)<<4;
29. for (i=DB[0]; i <= 10; i++)
30. {
31.       digitalWrite(i,temp & 0x80);
32.       temp <<= 1;
33. }
34.
35. digitalWrite( LCD1602_EN,HIGH);
36. delayMicroseconds(1);
37. digitalWrite( LCD1602_EN,LOW);
38. }
39.
40. void LCD_Data_Write(int dat)
41. {
42. int i=0,temp;
43. digitalWrite( LCD1602_RS,HIGH);
44. digitalWrite( LCD1602_RW,LOW);
45. digitalWrite( LCD1602_EN,LOW);
```

```
46.
47. temp=dat & 0xf0;
48. for (i=DB[0]; i <= 9; i++)
49. {
50.       digitalWrite(i,temp & 0x80);
51.       temp <<= 1;
52. }
53.
54. digitalWrite( LCD1602_EN,HIGH);
55. delayMicroseconds(1);
56. digitalWrite( LCD1602_EN,LOW);
57.
58. temp=(dat & 0x0f)<<4;
59. for (i=DB[0]; i <= 10; i++)
60. {
61.       digitalWrite(i,temp & 0x80);
62.       temp <<= 1;
63. }
64.
65. digitalWrite( LCD1602_EN,HIGH);
66. delayMicroseconds(1);
67. digitalWrite( LCD1602_EN,LOW);
```

```
68. }
69.
70. void LCD_SET_XY( int x, int y )
71. {
72.     int address;
73.     if (y ==0)      address = 0x80 + x;
74.     else                address = 0xC0 + x;
75.     LCD_Command_Write(address);
76. }
77.
78. void LCD_Write_Char( int x,int y,int dat)
79. {
80.     LCD_SET_XY( x, y );
81.     LCD_Data_Write(dat);
82. }
83.
84. void LCD_Write_String(int X,int Y,char *s)
85. {
86.       LCD_SET_XY( X, Y );      //设置地址
87.       while (*s)                    //写字符串
88.       {
89.           LCD_Data_Write(*s);
```

```
90.            s ++;
91.        }
92. }
93.
94. void setup (void)
95. {
96.     int i = 0;
97.     for (i=6; i <= 12; i++)
98.      {
99.          pinMode(i,OUTPUT);
100.      }
101.     delay(100);
102.     LCD_Command_Write(0x28);//4 线 2 行 5x7
103.     delay(50);
104.     LCD_Command_Write(0x06);
105.     delay(50);
106.     LCD_Command_Write(0x0c);
107.     delay(50);
108.     LCD_Command_Write(0x80);
109.     delay(50);
110.     LCD_Command_Write(0x01);
111.     delay(50);
```

```
112.
113.}
114.
115.void loop (void)
116.{
117.      LCD_Command_Write(0x01);
118.      delay(50);
119.      LCD_Write_String(3,0,str1);//第 1 行，第 4 个地址起
120.      delay(50);
121.      LCD_Write_String(1,1,str2);//第 2 行，第 2 个地址起
122.      delay(5000);
123.      LCD_Command_Write(0x01);
124.      delay(50);
125.      LCD_Write_String(0,0,str3);
126.      delay(50);
127.      LCD_Write_String(0,1,str4);
128.      delay(5000);
129.
130.}
```

复制代码

4 位接法实验效果如下

慧净电子---ARDUINO 模块化创新视频教程

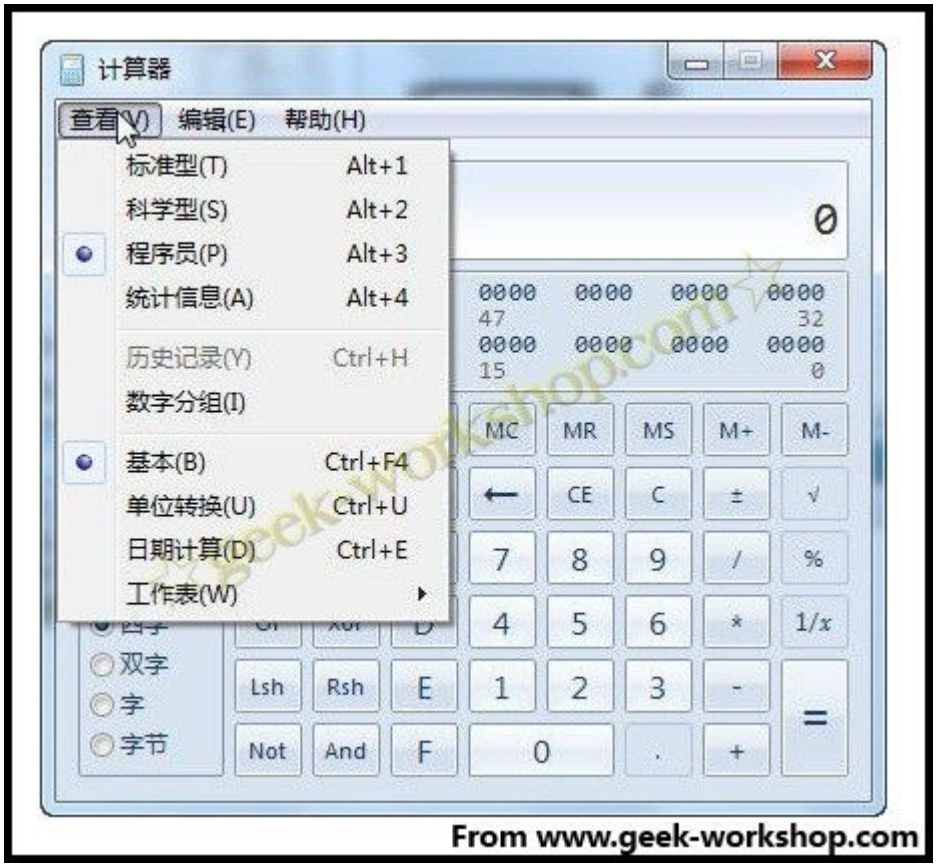
http://player.youku.com/player.php/sid/XMjg3MDE5MDIw/v.swf

这里我们讲解一下最关键的部分，就是 LCD 的控制命令。

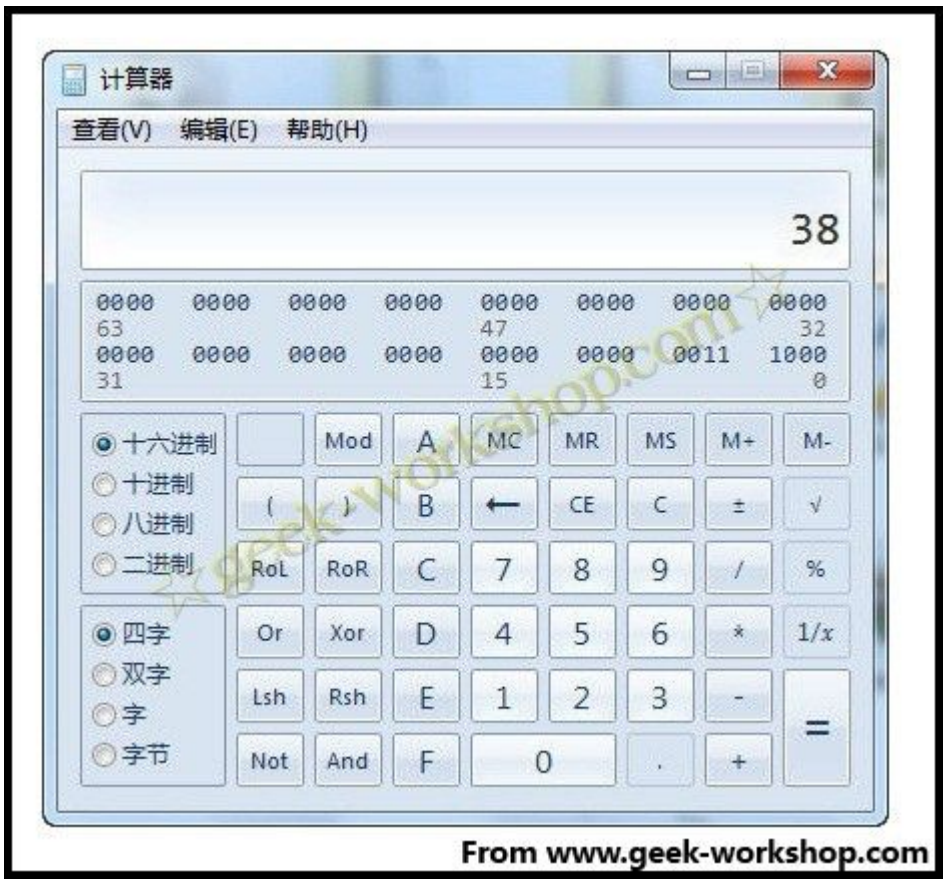
在上面两段代码中，我们常常可以遇到 0x01,0x38 这种参数。这些参数代表什么呢？

在 C/C++语言中，0x38 代表的是十六进制的数值"38"，"0x"的意思就是十六进制。

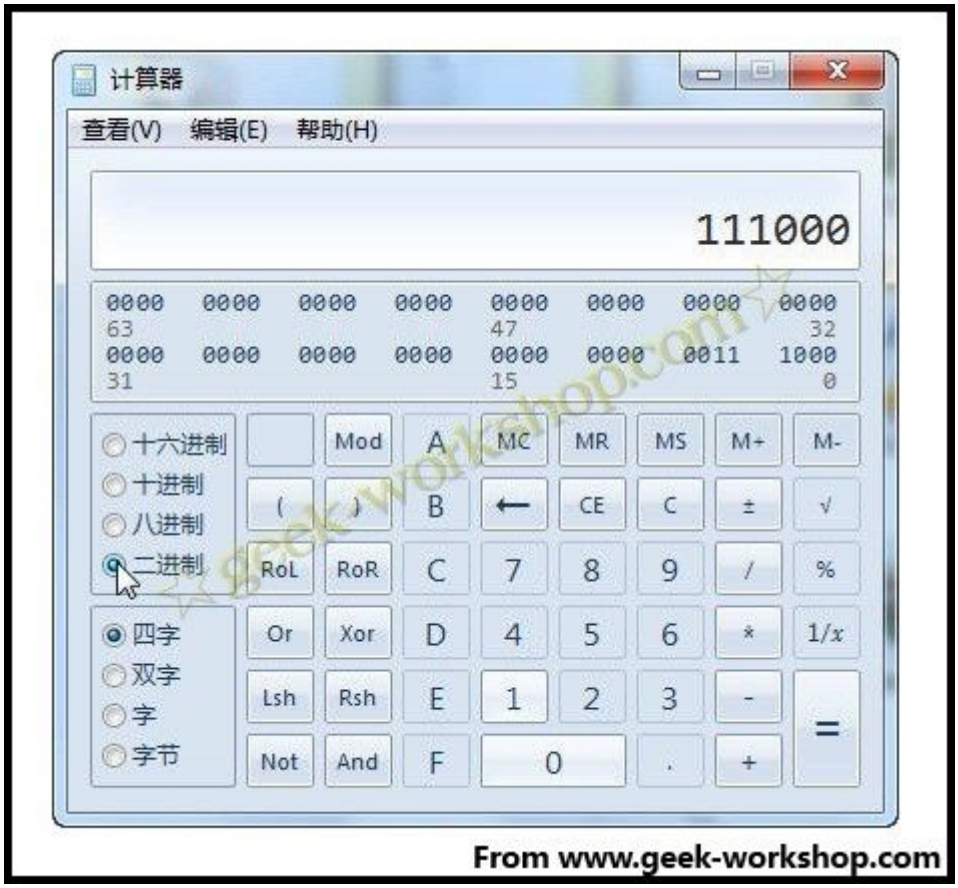
先打开 win7 下的计算器，选择“程序员”“基本”，



然后咱们选择“十六进制”，输入“38”，



然后再点击“二进制”。这时十六进制的“38”就会转换为二进制下的数值“111000”。



以 8 位控制法接 LCD 是，对应的控制信息就是“00111000”

端口	D7	D6	D5	D4	D3	D2	D1	D0
信号	0	0	1	1	1	0	0	0
开关	关	关	开	开	开	关	关	关

同理，也可以把二进制的控制信息，逆运算为十六进制的。

有的产品说明书写的控制命令是"38H"

这里说明一下，一般情况下

十六进制 前缀 0x 后缀 h

十进制 后缀 D

八进制 后缀 Q

二进制 后缀 B

但是不同的程序语言，对于十六进制的表达方式不完全相同，在 arduino 下，表达十六进制数值“38”只能使用“0x38”而不能用“38H”

最后放三个附件，是三个不同厂家的 1602 LCD 手册，供大家深入研究。

 1602 手册.pdf （634.17 KB, 下载次数: 3）

 1602.pdf （233 KB, 下载次数: 1）

 SMC1602A.pdf （255.88 KB, 下载次数: 1）

arduino 学习笔记 24 温度传感器实验

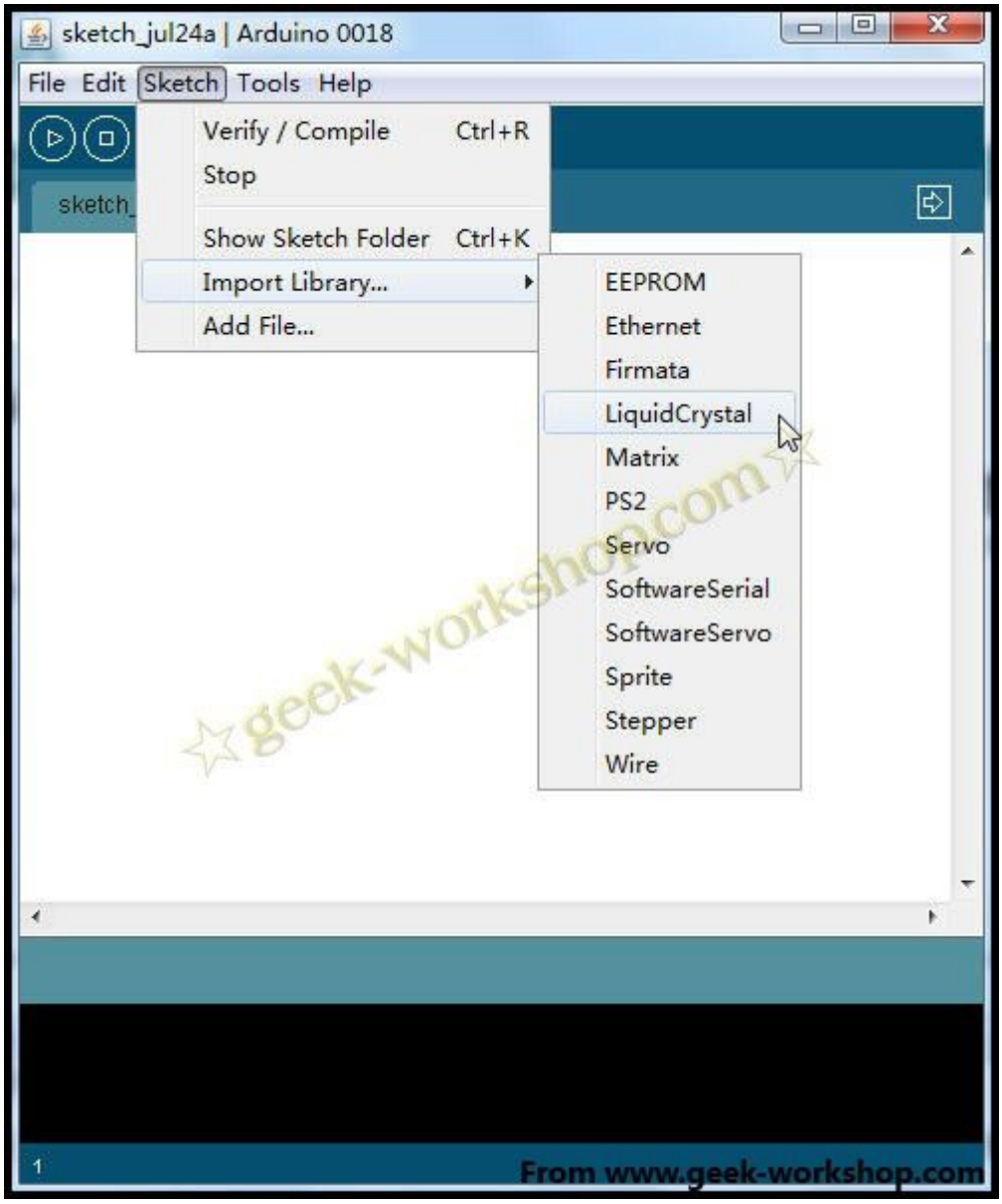
通过上一次的实验，我们学会了 1602 液晶的使用。下面我们做一个简单的温度传感器，通过一个 LM35 温度传感器读取室温后，使温度显示在 1602 液晶屏幕上。

慧净电子---ARDUINO 模块化创新视频教程

上次我们驱动液晶是使用的手工编写代码，这次我们直接使用 arduino 自带的 LiquidCrystal 库来进行驱动，此库文件允许 arduino 控制板控制基于 Hitachi HD44780 或与之相兼容芯片大部分的液晶，可以工作于 4bit 或者 8bit 状态。

下图为我们所使用的 arduino 的 LiquidCrystal 库文件位置，只有这里显示的库文件，arduino 才可以调用。

慧净电子---ARDUINO 模块化创新视频教程

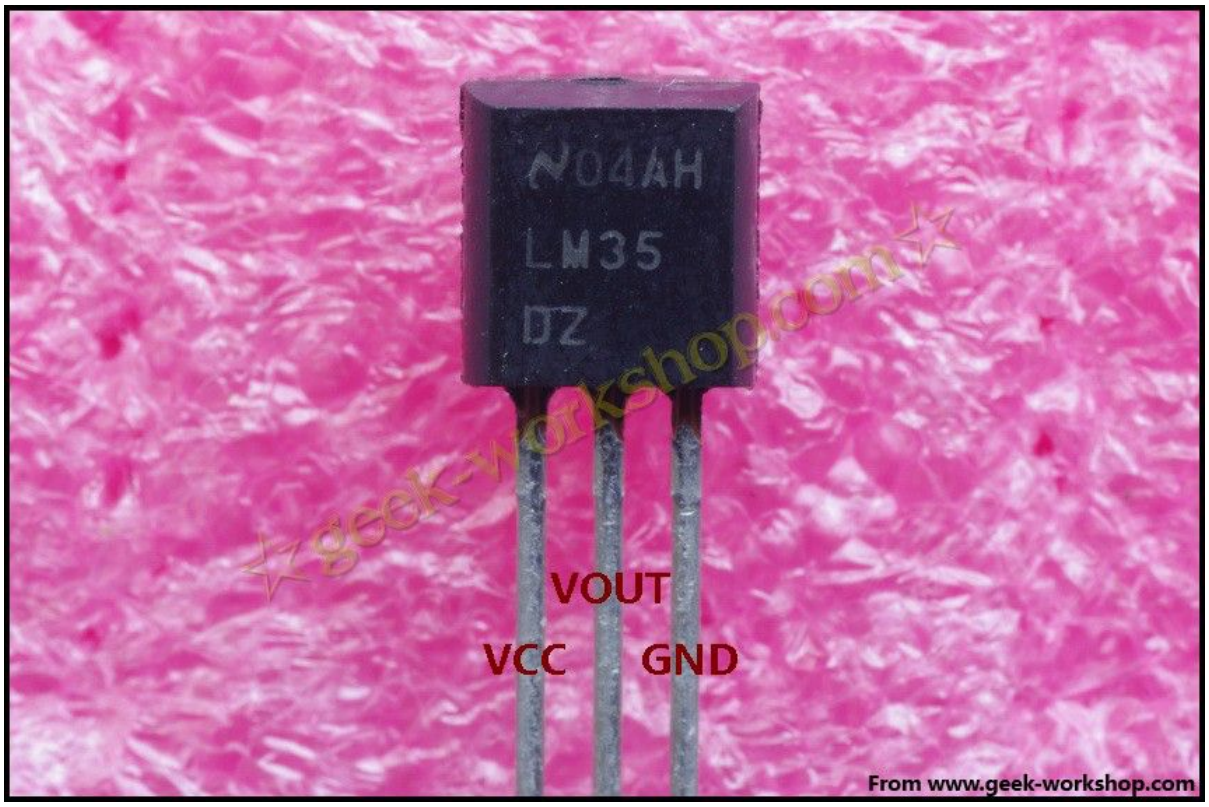


什么是温度传感器？

温度传感器就是利用物质随温度变化特性的规律，把温度转换为电量的传感器。按照测量方式可以分为接触式和非接触式两大类，按照传感器材料以及元件特性分为热电阻传感器和热电偶传感器两类。白光烙铁头使用的是热电偶传感器，本次试验使用的 LM35 是热电阻传感

器。

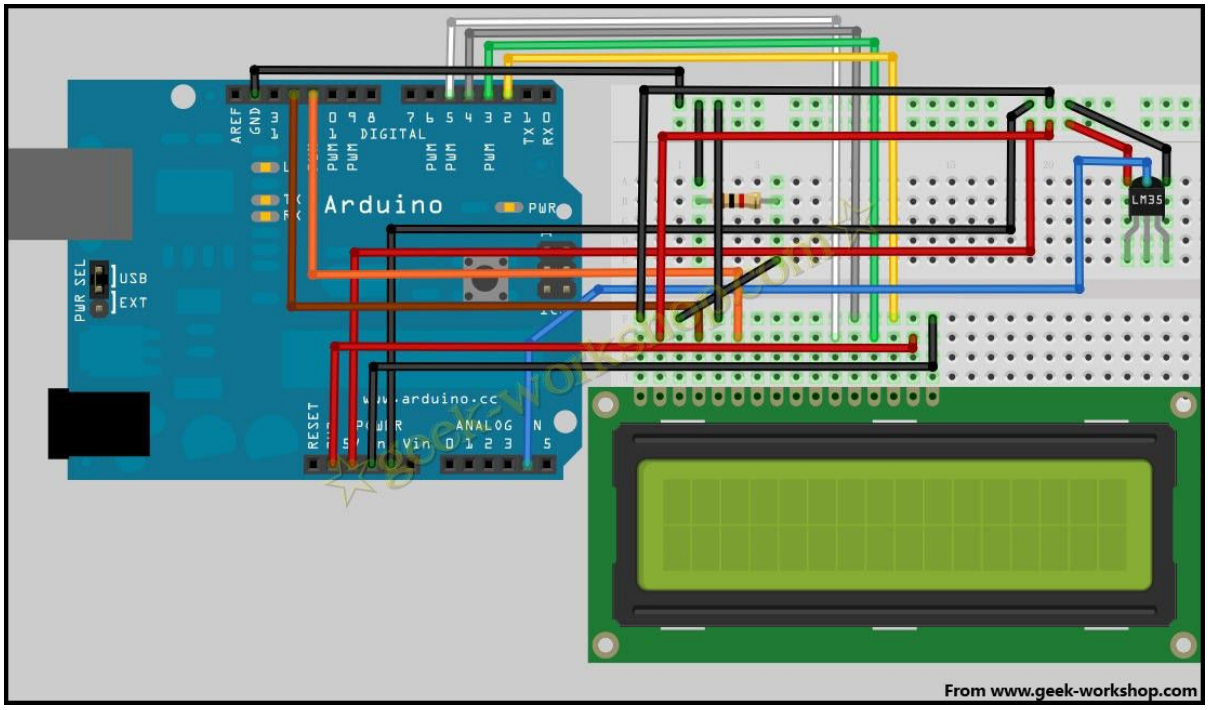
LM35 温度传感器实物如下图：



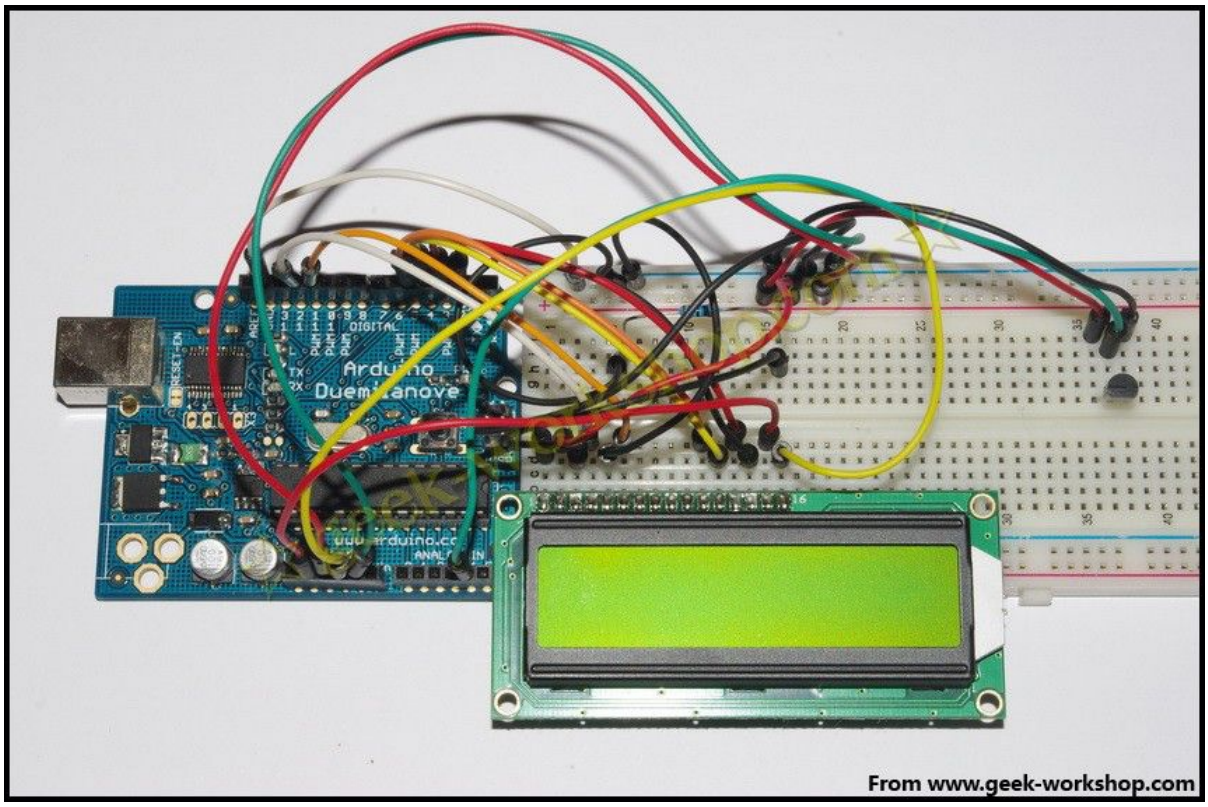
LM35 使用非常普遍，他使用内部补偿机制，输出可以从 0℃开始。封装为 T0992，工作电压 4—30V。而且在上述电压范围内，芯片的工作电流不超过 60ua。根据产品使用手册，得知 LM35 传感器的输出电压与摄氏温度呈线性关系，0℃时输出为 0V，每升高 1℃，输出电压增加 10mv。

慧净电子---ARDUINO 模块化创新视频教程

下图为实验硬件连接方式



慧净电子---ARDUINO 模块化创新视频教程



代码如下

```
1.  #include <LiquidCrystal.h>    //调用 arduino 自带的
    LiquidCrystal 库
2.
3.  LiquidCrystal lcd(12, 11, 5, 4, 3, 2);//设置接口
4.
5.  int potPin = 4;                    //设置模拟口 4 为 LM35 的信
    号输入端口
6.  float temperature = 0;              //设置 temperature 为浮
    点变量
```

```
7.  long val=0;                          //设置 val 为长整数变量
8.
9.  void setup()
10. {
11.   lcd.begin(16, 2);  //初始化 LCD
12. lcd.print("LM35 Thermometer");  //使屏幕显示文字 LM35
    Thermometer
13. delay(1000); //延时 1000ms
14. }
15.
16. void loop ()
17. {
18.
19. val = analogRead(potPin);              //val 变量为从 LM35 信
    号口读取到的数值
20. temperature = ((val+1)*0.0048828125*1000);          //把读
    取到的 val 转换为温度数值的 10 倍
21. lcd.clear(); //清屏
22. lcd.print("LM35 Thermometer"); //使屏幕显示文字 LM35
    Thermometer
23. lcd.setCursor(0, 1) ; //设置光标位置为第二行第一个位置
24. lcd.print((long)temperature / 10);    //显示温度整数位
```

```
25. lcd.print(".");     //显示小数点
26. lcd.print( (long)temperature % 10); //显示温度小数点后一位
27. lcd.print((char)223); //显示 o 符号
28. lcd.print("C"); //显示字母 C
29.
30. delay(2000);                    //延时 2 秒，这里也就是刷新速
    度。
31.
32. }
```

复制代码

实验效果如下，为了使温度发生变化，用一个装了热水的塑料杯，放在传感器旁边。

关于 LiquidCrystal 库使用的更详细方法，请查看 arduino 官方介绍。

http://arduino.cc/en/Reference/LiquidCrystal

附件为 LM35 的产品说明书


LM35 产品说明书.pdf (303.72 KB, 下载次数: 0)

arduino 学习笔记 25 ADXL345 加速度传感器实验

前两天我们做了温度传感器实验，大家一定还有印象。今天我们来研究另外一种传感器加速度传感器。

什么是加速度传感器

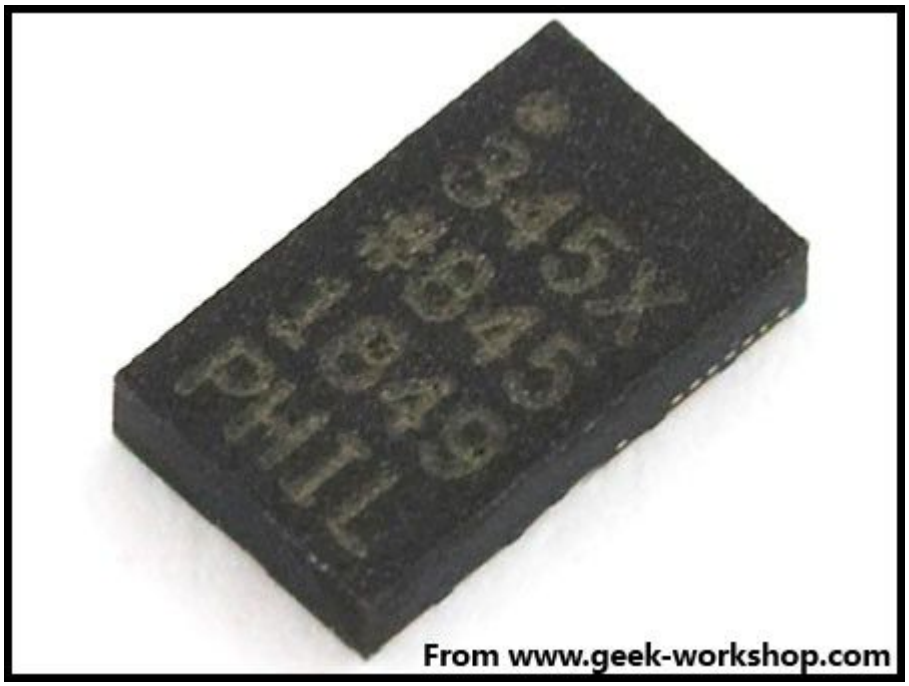
加速度传感器，作用是测量在加速过程中产生的力。最基本的如咱们平常所熟悉的是重力加速度，大小是 1g。

加速度传感器一般用于什么地方

通过测量由重力引起的加速度，你可以计算出设备相对于水平面的倾斜角度。通过分析动态加速度，你可以分析出设备的移动方式。自平衡车中就是使用加速度传感器与陀螺仪进行卡尔曼滤波进行姿态矫正。

本次试验使用的 ADXL345 数字传感器，通过 I2C 或者 SPI 接口直接输出数字信号。在 1g 的加速度下，输出数值为 256.

慧净电子---ARDUINO 模块化创新视频教程

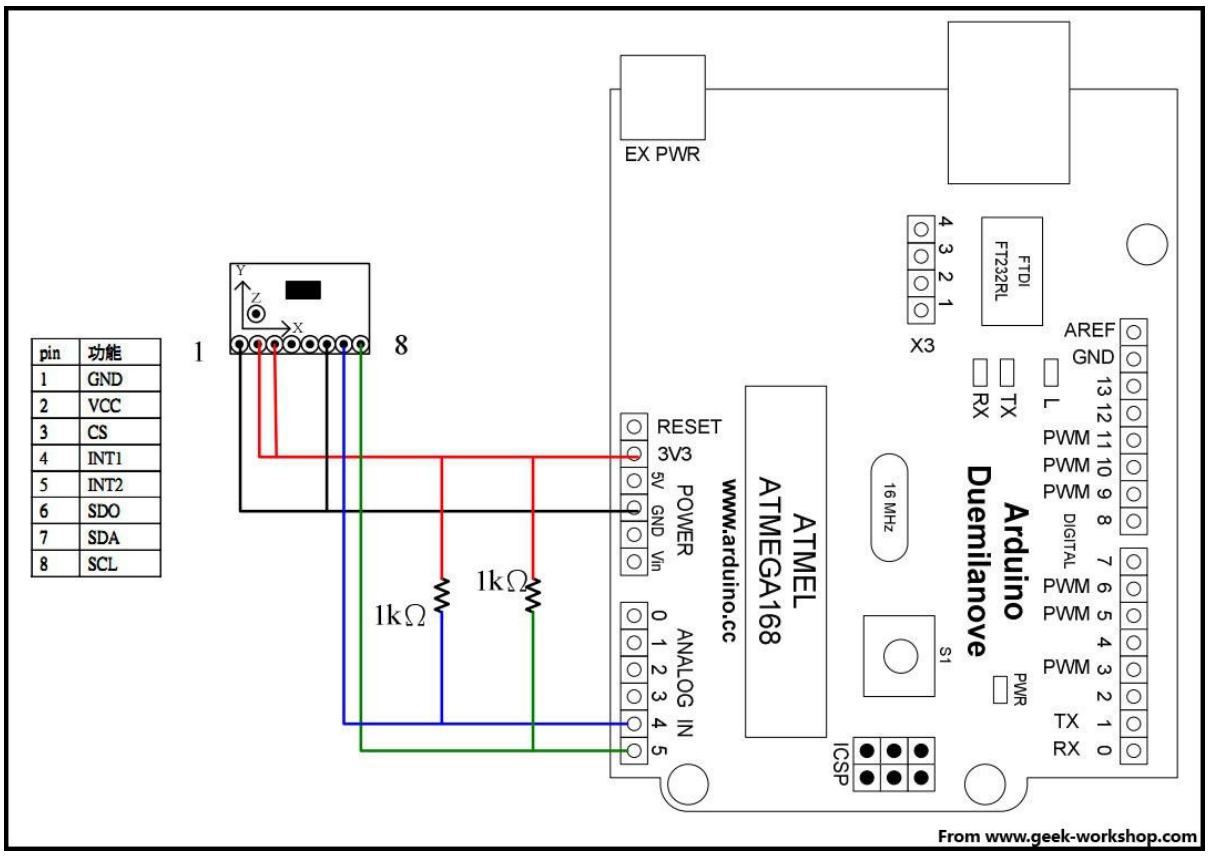


2011-7-28 22:56:15 上传

下载附件 (24.89 KB)

下面是硬件连接图

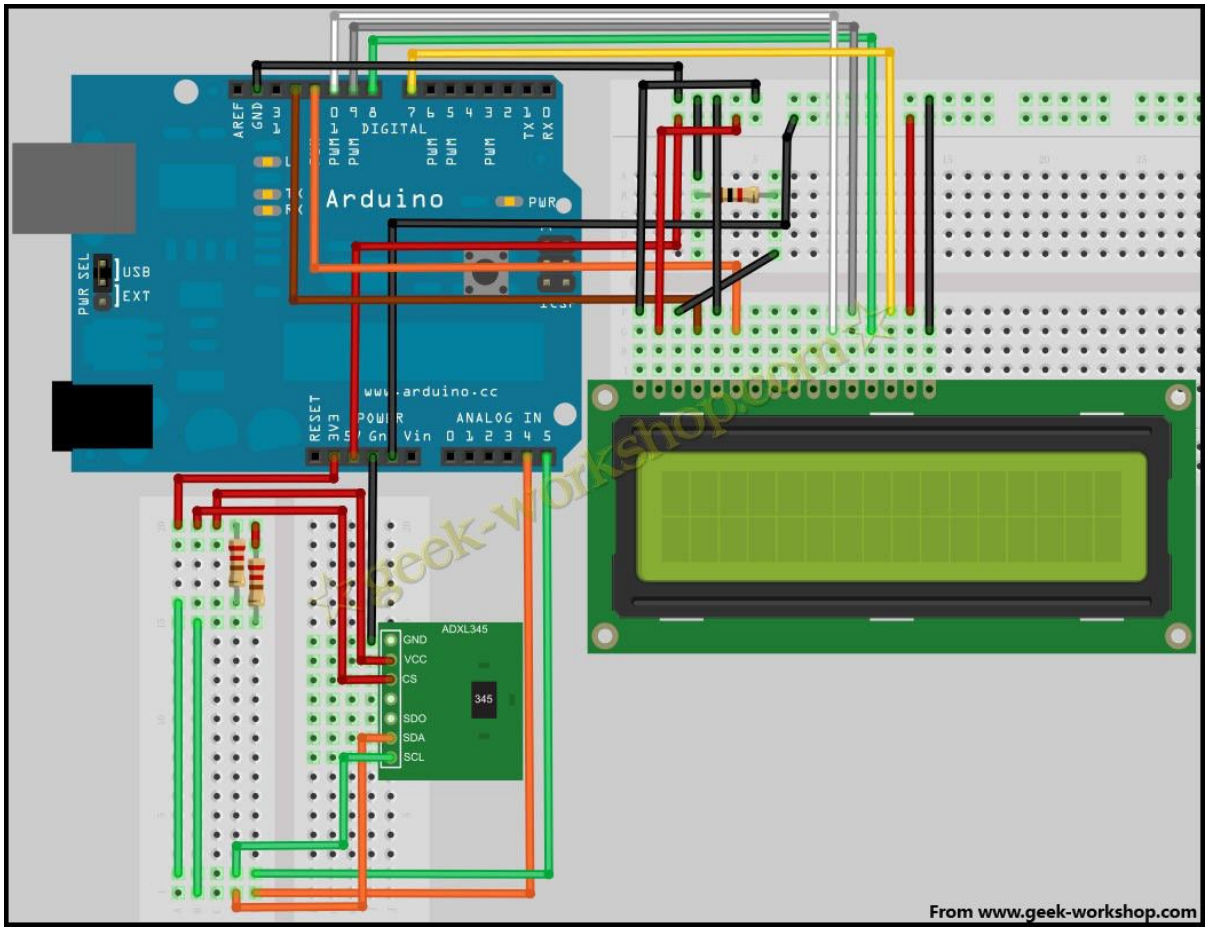
慧净电子---ARDUINO 模块化创新视频教程



2011-7-29 22:16:48 上传

下载附件 (112.73 KB)

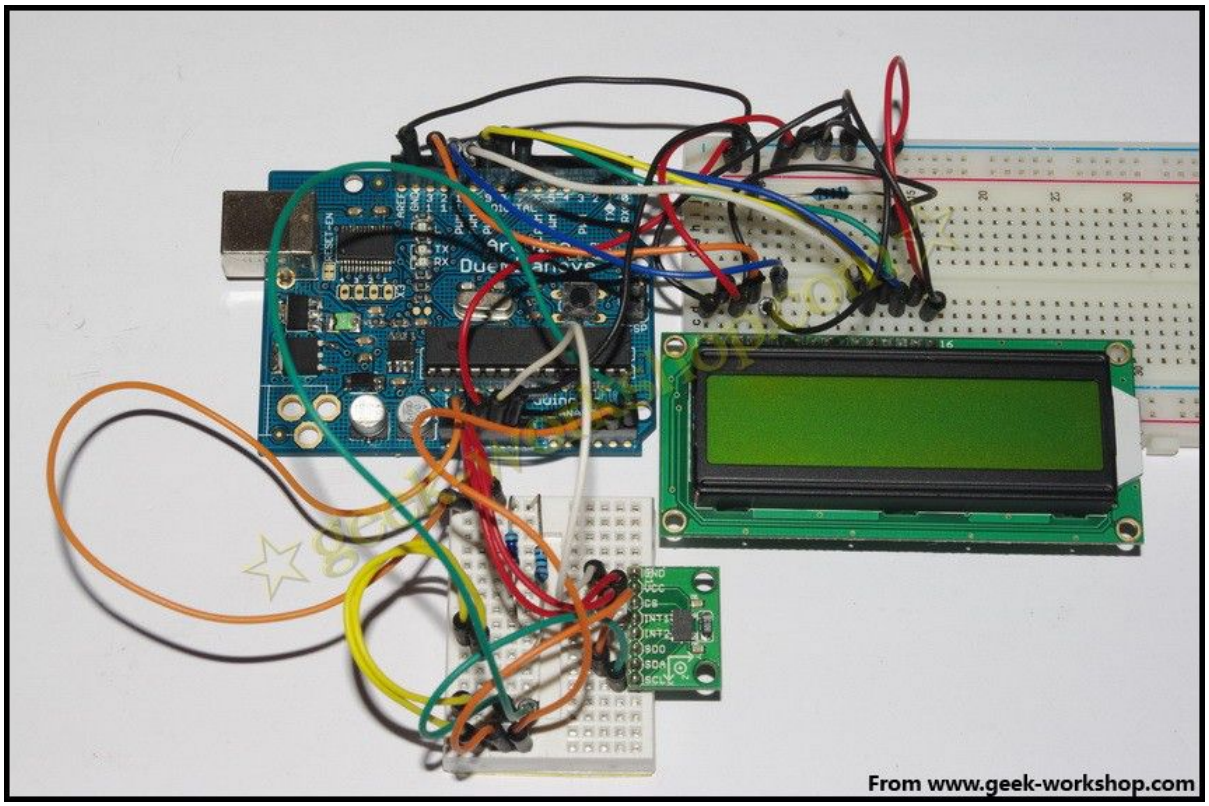
慧净电子---ARDUINO 模块化创新视频教程



2011-7-29 15:58:16 上传

下载附件 (164.15 KB)

慧净电子---ARDUINO 模块化创新视频教程



2011-7-28 22:56:16 上传

下载附件 (203.65 KB)

下面是代码

```
1. #include <Wire.h>      //调用 arduino 自带的 I2C 库
2. #include <LiquidCrystal.h>      //调用 arduino 自带的
   LiquidCrystal 库
3.
```

```
4. #define Register_ID 0
5. #define Register_2D 0x2D
6. #define Register_X0 0x32
7. #define Register_X1 0x33
8. #define Register_Y0 0x34
9. #define Register_Y1 0x35
10. #define Register_Z0 0x36
11. #define Register_Z1 0x37
12.
13. LiquidCrystal lcd(12, 11, 10, 9, 8, 7);//设置接口
14.
15. int ADXAddress = 0xA7>>1;     //转换为 7 位地址
16. int reading = 0;
17. int val = 0;
18. int X0,X1,X_out;
19. int Y0,Y1,Y_out;
20. int Z1,Z0,Z_out;
21. double Xg,Yg,Zg;
22.
23. void setup()
24. {
25.     lcd.begin(16, 2);     //初始化 LCD
```

```
26.      delay(100);
27.      Wire.begin();     //初始化 I2C
28.      delay(100);
29.      Wire.beginTransmission(ADXAddress);
30.      Wire.send(Register_2D);
31.      Wire.send(8);
32.      Wire.endTransmission();
33. }
34.
35. void loop()
36. {
37.      Wire.beginTransmission(ADXAddress);
38.      Wire.send(Register_X0);
39.      Wire.send(Register_X1);
40.      Wire.endTransmission();
41.      Wire.requestFrom(ADXAddress,2);
42.      if(Wire.available()<=2);
43.      {
44.        X0 = Wire.receive();
45.        X1 = Wire.receive();
46.        X1 = X1<<8;
47.        X_out = X0+X1;
```

```
48.     }
49.
50.     Wire.beginTransmission(ADXAddress);
51.     Wire.send(Register_Y0);
52.     Wire.send(Register_Y1);
53.     Wire.endTransmission();
54.     Wire.requestFrom(ADXAddress,2);
55.     if(Wire.available()<=2);
56.     {
57.       Y0 = Wire.receive();
58.       Y1 = Wire.receive();
59.       Y1 = Y1<<8;
60.       Y_out = Y0+Y1;
61.     }
62.
63.     Wire.beginTransmission(ADXAddress);
64.     Wire.send(Register_Z0);
65.     Wire.send(Register_Z1);
66.     Wire.endTransmission();
67.     Wire.requestFrom(ADXAddress,2);
68.     if(Wire.available()<=2);
69.     {
```

```
70.         Z0 = Wire.receive();
71.         Z1 = Wire.receive();
72.         Z1 = Y1<<8;
73.         Z_out = Y0+Y1;
74.       }
75.
76.       Xg = X_out/256.00;//把输出结果转换为重力加速度 g,
    精确到小数点后 2 位。
77.       Yg = Y_out/256.00;
78.       Zg = Z_out/256.00;
79.       lcd.clear(); //清屏
80.       lcd.print("X="); //使屏幕显示文字 X=
81.       lcd.print(Xg);
82.       lcd.setCursor(8, 0);
83.       lcd.print("Y=");
84.       lcd.print(Yg);
85.       lcd.setCursor(0, 1);
86.       lcd.print("Z=");
87.       lcd.print(Zg);
88.       delay(500);      //延时 0.5 秒，刷新频率这里进行调整
89.
90. }
```

复制代码

试验效果如下

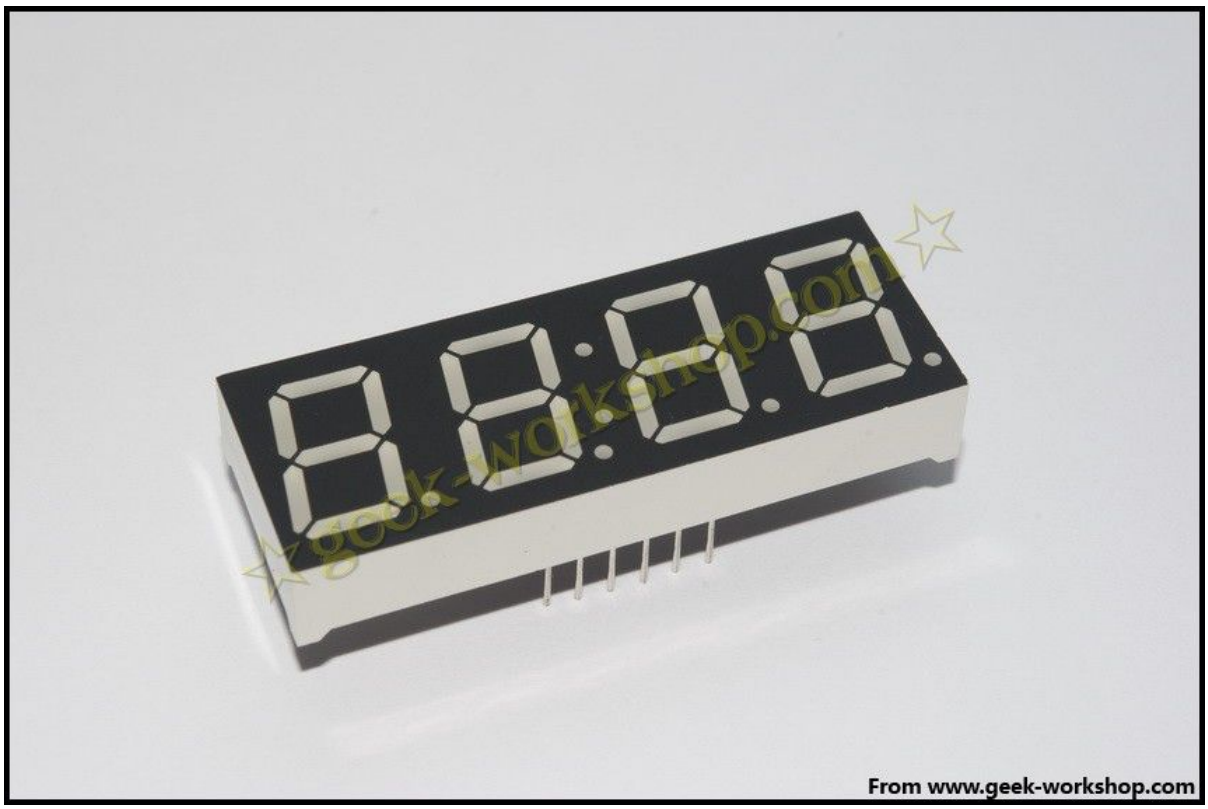
附件为 ADXL345 中文手册

ADXL345 中文 PDF.pdf

arduino 学习笔记 26　　4 位数码管实验

这次我们进行的实验是使用 arduino 驱动一块共阳四位数码管。驱动数码管限流电阻肯定是必不可少的，限流电阻有两种接法，一种是在 d1-d4 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 220Ω 电阻（因为没有 100Ω 电阻，所以使用 220Ω 的代替，100 欧姆亮度会比较高）。

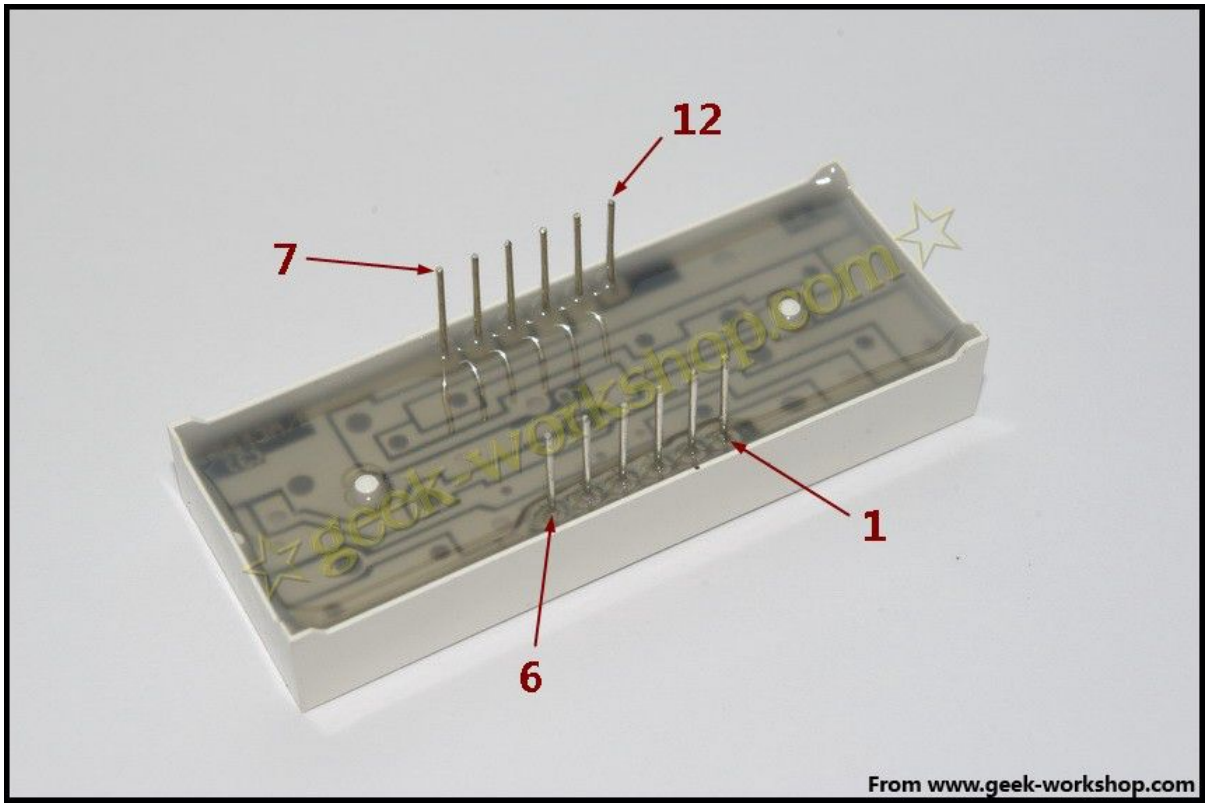
下图为 4 位数码管



23　分钟前　上传

下载附件 (118.5 KB)

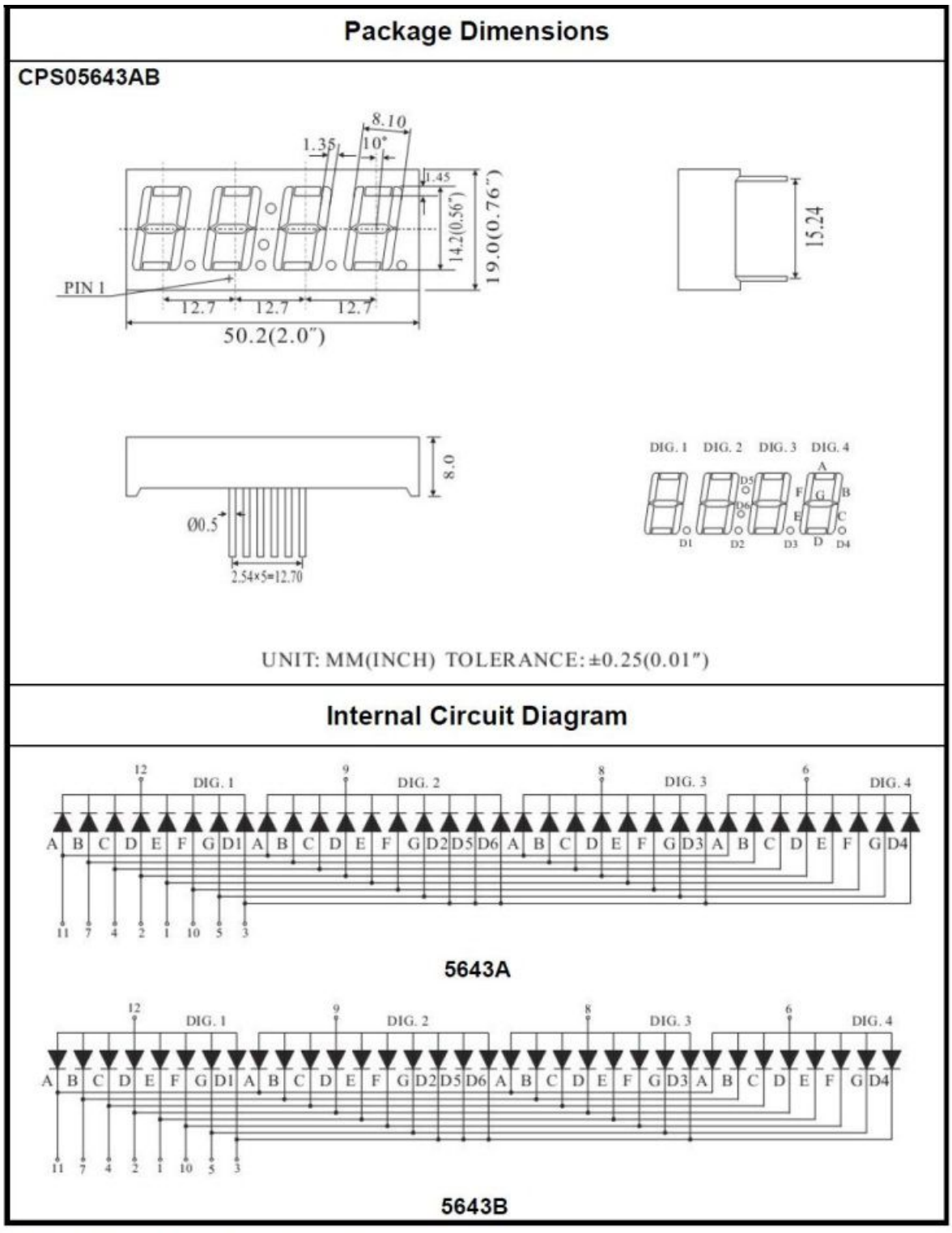
4 位数码管总共有 12 个引脚，小数点朝下正放在面前时，左下角为 1，其他管脚顺序为逆时针旋转。左上角为最大的 12 号管脚。



23　分钟前　上传

下载附件 (62.87 KB)

下图为数码管的说明手册

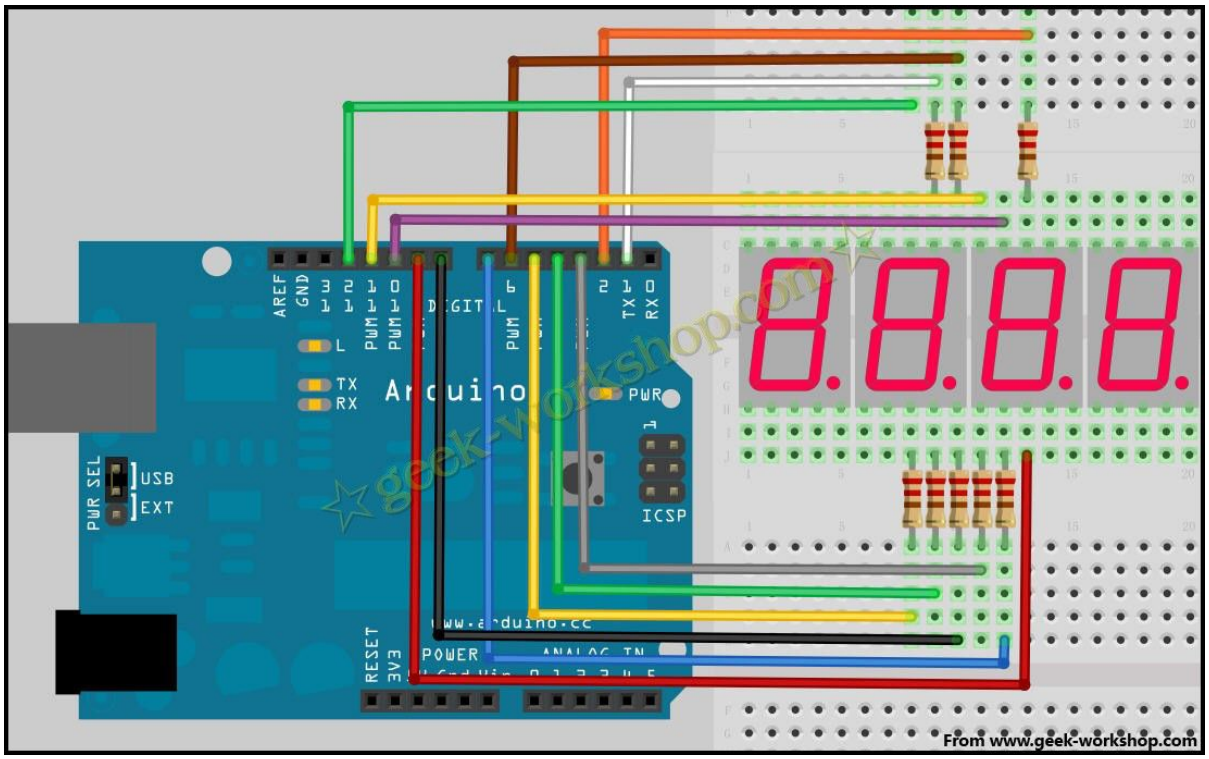


Four Digits Displays Series

23 分钟前 上传

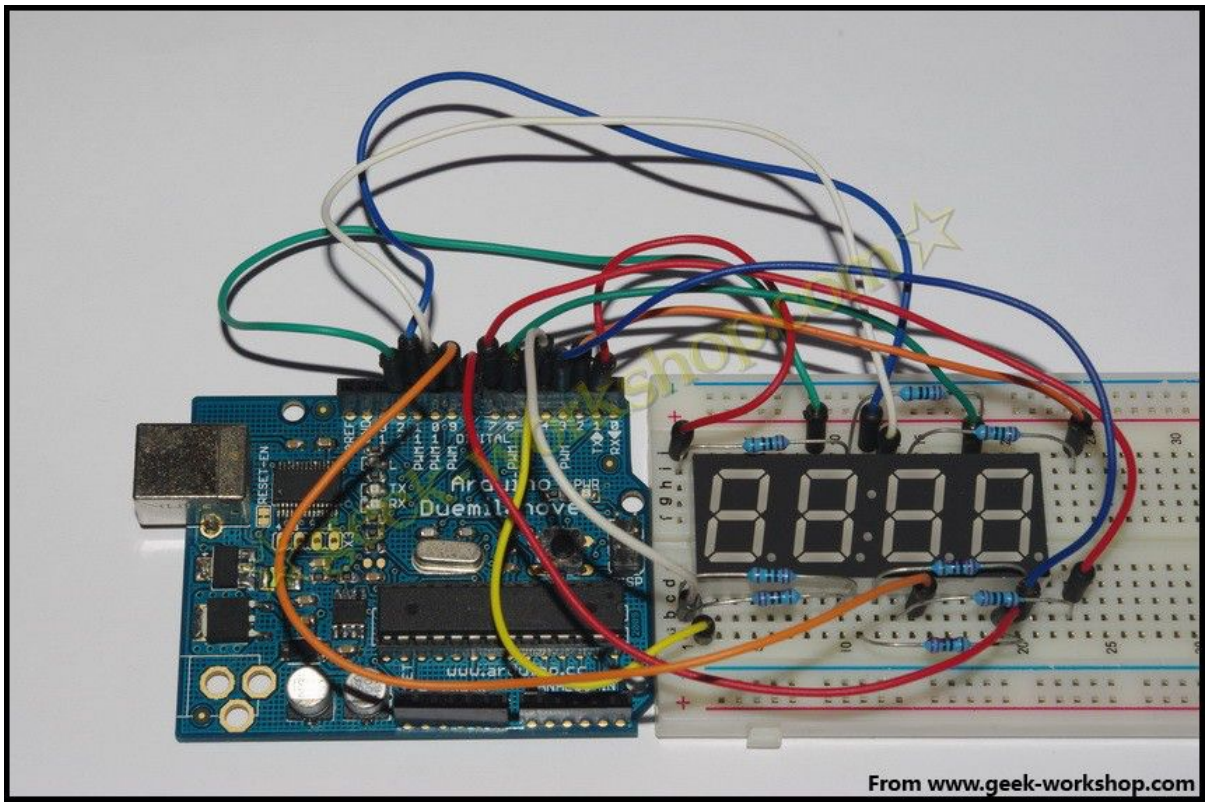
下载附件 (143.54 KB)

下面是硬件连接图



23　分钟前　上传

下载附件 (154.12 KB)



18　分钟前　上传

下载附件 (202.16 KB)

把下面的代码复制下载到控制板中，看看效果。

```
1. //设置阴极接口
2. int a = 1;
3. int b = 2;
4. int c = 3;
```

```
5. int d = 4;
6. int e = 5;
7. int f = 6;
8. int g = 7;
9. int p = 8;
10. //设置阳极接口
11. int d4 = 9;
12. int d3 = 10;
13. int d2 = 11;
14. int d1 = 12;
15. //设置变量
16. int del = 100;
17. int buttoncount = 0;
18. int loopcount = 0;
19.
20. void setup()
21. {
22.     pinMode(d1, OUTPUT);
23.     pinMode(d2, OUTPUT);
24.     pinMode(d3, OUTPUT);
25.     pinMode(d4, OUTPUT);
26.     pinMode(a, OUTPUT);
```

```
27.     pinMode(b, OUTPUT);
28.     pinMode(c, OUTPUT);
29.     pinMode(d, OUTPUT);
30.     pinMode(e, OUTPUT);
31.     pinMode(f, OUTPUT);
32.     pinMode(g, OUTPUT);
33.     pinMode(p, OUTPUT);
34.     digitalWrite(a, HIGH);
35.     digitalWrite(b, HIGH);
36.     digitalWrite(c, HIGH);
37.     digitalWrite(d, HIGH);
38.     digitalWrite(e, HIGH);
39.     digitalWrite(f, HIGH);
40.     digitalWrite(g, HIGH);
41.     digitalWrite(p, HIGH);
42. }
43.
44. void loop()
45. {
46.     roulette(4);     //轮转效果
47.     delay(100);
48.     zigzag(2);       //Z 字型效果
```

```
49.     delay(100);
50.     circles(4);      //圈状效果
51.     delay(100);
52. }
53.
54. void pickDigit(int x)     //定义 pickDigit(x),其作用是
    开启 dx 端口
55. {
56.     digitalWrite(d1, LOW);
57.     digitalWrite(d2, LOW);
58.     digitalWrite(d3, LOW);
59.     digitalWrite(d4, LOW);
60.
61.     switch(x)
62.     {
63.     case 1:
64.       digitalWrite(d1, HIGH);
65.       break;
66.     case 2:
67.       digitalWrite(d2, HIGH);
68.       break;
69.     case 3:
```

```
70.       digitalWrite(d3, HIGH);
71.       break;
72.     default:
73.       digitalWrite(d4, HIGH);
74.       break;
75.     }
76. }
77.
78. void clearLEDs()     //清屏
79. {
80.     digitalWrite(a, HIGH);
81.     digitalWrite(b, HIGH);
82.     digitalWrite(c, HIGH);
83.     digitalWrite(d, HIGH);
84.     digitalWrite(e, HIGH);
85.     digitalWrite(f, HIGH);
86.     digitalWrite(g, HIGH);
87.     digitalWrite(p, HIGH);
88. }
89.
90. void roulette(int x)     //设置轮转效果
91. {
```

```
92.     loopcount = 0;
93.     while (loopcount < x)
94.     {
95.       digitalWrite(a, LOW);
96.       pickDigit(1);
97.       delay(del);
98.       pickDigit(2);
99.       delay(del);
100.        pickDigit(3);
101.        delay(del);
102.        pickDigit(4);
103.        delay(del);
104.        digitalWrite(a, HIGH);
105.        digitalWrite(b, LOW);
106.        delay(del);
107.        digitalWrite(b, HIGH);
108.        digitalWrite(c, LOW);
109.        delay(del);
110.        digitalWrite(c, HIGH);
111.        digitalWrite(d, LOW);
112.        delay(del);
113.        pickDigit(3);
```

```
114.          delay(del);
115.          pickDigit(2);
116.          delay(del);
117.          pickDigit(1);
118.          delay(del);
119.          digitalWrite(d, HIGH);
120.          digitalWrite(e, LOW);
121.          delay(del);
122.          digitalWrite(e, HIGH);
123.          digitalWrite(f, LOW);
124.          delay(del);
125.          clearLEDs();
126.          loopcount++;
127.        }
128.    }
129.
130.    void zigzag(int x)     //设置 Z 字形效果
131.    {
132.        loopcount = 0;
133.        while(loopcount < x)
134.        {
135.          digitalWrite(a, LOW);
```

```
136.          pickDigit(1);
137.          delay(del);
138.          pickDigit(2);
139.          delay(del);
140.          pickDigit(3);
141.          delay(del);
142.          pickDigit(4);
143.          delay(del);
144.          digitalWrite(a, HIGH);
145.          digitalWrite(b, LOW);
146.          delay(del);
147.          digitalWrite(b, HIGH);
148.          digitalWrite(g, LOW);
149.          delay(del);
150.          pickDigit(3);
151.          delay(del);
152.          pickDigit(2);
153.          delay(del);
154.          pickDigit(1);
155.          delay(del);
156.          digitalWrite(g, HIGH);
157.          digitalWrite(e, LOW);
```

```
158.          delay(del);
159.          digitalWrite(e, HIGH);
160.          digitalWrite(d, LOW);
161.          delay(del);
162.          pickDigit(2);
163.          delay(del);
164.          pickDigit(3);
165.          delay(del);
166.          pickDigit(4);
167.          delay(del);
168.          digitalWrite(d, HIGH);
169.          digitalWrite(c, LOW);
170.          delay(del);
171.          digitalWrite(c, HIGH);
172.          digitalWrite(g, LOW);
173.          delay(del);
174.          pickDigit(3);
175.          delay(del);
176.          pickDigit(2);
177.          delay(del);
178.          pickDigit(1);
179.          delay(del);
```

```
180.          digitalWrite(g, HIGH);
181.          digitalWrite(f, LOW);
182.          delay(del);
183.          clearLEDs();
184.          loopcount++;
185.        }
186.    }
187.
188.    void circles(int x)     //设置圈状效果
189.    {
190.        loopcount = 0;
191.        while (loopcount < x)
192.        {
193.          digitalWrite(a, LOW);
194.          digitalWrite(b, LOW);
195.          digitalWrite(f, LOW);
196.          digitalWrite(g, LOW);
197.          pickDigit(1);
198.          delay(250);
199.          digitalWrite(a, HIGH);
200.          digitalWrite(b, HIGH);
201.          digitalWrite(f, HIGH);
```

```
202.        digitalWrite(c, LOW);
203.        digitalWrite(d, LOW);
204.        digitalWrite(e, LOW);
205.        pickDigit(2);
206.        delay(250);
207.        digitalWrite(a, LOW);
208.        digitalWrite(b, LOW);
209.        digitalWrite(f, LOW);
210.        digitalWrite(c, HIGH);
211.        digitalWrite(d, HIGH);
212.        digitalWrite(e, HIGH);
213.        pickDigit(3);
214.        delay(250);
215.        digitalWrite(a, HIGH);
216.        digitalWrite(b, HIGH);
217.        digitalWrite(f, HIGH);
218.        digitalWrite(c, LOW);
219.        digitalWrite(d, LOW);
220.        digitalWrite(e, LOW);
221.        pickDigit(4);
222.        delay(250);
223.        clearLEDs();
```

```
224.            loopcount++;
225.        }
226.    }
```

复制代码

上面代码效果如下，为 3 种花样组成。

再把下面代码复制下载到控制板中，看看效果。

```
1. //设置阴极接口
2. int a = 1;
3. int b = 2;
4. int c = 3;
5. int d = 4;
6. int e = 5;
7. int f = 6;
8. int g = 7;
9. int p = 8;
10. //设置阳极接口
11. int d4 = 9;
12. int d3 = 10;
13. int d2 = 11;
```

```
14. int d1 = 12;
15. //设置变量
16. long n = 0;
17. int x = 100;
18. int del = 55;     //此处数值对时钟进行微调
19.
20. void setup()
21. {
22.     pinMode(d1, OUTPUT);
23.     pinMode(d2, OUTPUT);
24.     pinMode(d3, OUTPUT);
25.     pinMode(d4, OUTPUT);
26.     pinMode(a, OUTPUT);
27.     pinMode(b, OUTPUT);
28.     pinMode(c, OUTPUT);
29.     pinMode(d, OUTPUT);
30.     pinMode(e, OUTPUT);
31.     pinMode(f, OUTPUT);
32.     pinMode(g, OUTPUT);
33.     pinMode(p, OUTPUT);
34. }
35.
```

```
36. void loop()
37. {
38.     clearLEDs();
39.     pickDigit(1);
40.     pickNumber((n/x/1000)%10);
41.     delayMicroseconds(del);
42.
43.     clearLEDs();
44.     pickDigit(2);
45.     pickNumber((n/x/100)%10);
46.     delayMicroseconds(del);
47.
48.     clearLEDs();
49.     pickDigit(3);
50.     dispDec(3);
51.     pickNumber((n/x/10)%10);
52.     delayMicroseconds(del);
53.
54.     clearLEDs();
55.     pickDigit(4);
56.     pickNumber(n/x%10);
57.     delayMicroseconds(del);
```

```
58.
59.     n++;
60.
61.     if (digitalRead(13) == HIGH)
62.     {
63.       n = 0;
64.     }
65. }
66.
67. void pickDigit(int x)    //定义 pickDigit(x),其作用是
    开启 dx 端口
68. {
69.     digitalWrite(d1, LOW);
70.     digitalWrite(d2, LOW);
71.     digitalWrite(d3, LOW);
72.     digitalWrite(d4, LOW);
73.
74.     switch(x)
75.     {
76.     case 1:
77.       digitalWrite(d1, HIGH);
78.       break;
```

```
79.     case 2:
80.       digitalWrite(d2, HIGH);
81.       break;
82.     case 3:
83.       digitalWrite(d3, HIGH);
84.       break;
85.     default:
86.       digitalWrite(d4, HIGH);
87.       break;
88.     }
89. }
90.
91. void pickNumber(int x)     //定义 pickNumber(x),其作用
    是显示数字 x
92. {
93.     switch(x)
94.     {
95.     default:
96.       zero();
97.       break;
98.     case 1:
99.       one();
```

```
100.          break;
101.        case 2:
102.          two();
103.          break;
104.        case 3:
105.          three();
106.          break;
107.        case 4:
108.          four();
109.          break;
110.        case 5:
111.          five();
112.          break;
113.        case 6:
114.          six();
115.          break;
116.        case 7:
117.          seven();
118.          break;
119.        case 8:
120.          eight();
121.          break;
```

```
122.        case 9:
123.          nine();
124.          break;
125.        }
126.    }
127.
128.    void dispDec(int x)     //设定开启小数点
129.    {
130.        digitalWrite(p, LOW);
131.    }
132.
133.    void clearLEDs()     //清屏
134.    {
135.        digitalWrite(a, HIGH);
136.        digitalWrite(b, HIGH);
137.        digitalWrite(c, HIGH);
138.        digitalWrite(d, HIGH);
139.        digitalWrite(e, HIGH);
140.        digitalWrite(f, HIGH);
141.        digitalWrite(g, HIGH);
142.        digitalWrite(p, HIGH);
143.    }
```

```
144.
145.   void zero()     //定义数字 0 时阴极那些管脚开关
146.   {
147.       digitalWrite(a, LOW);
148.       digitalWrite(b, LOW);
149.       digitalWrite(c, LOW);
150.       digitalWrite(d, LOW);
151.       digitalWrite(e, LOW);
152.       digitalWrite(f, LOW);
153.       digitalWrite(g, HIGH);
154.   }
155.
156.   void one()     //定义数字 1 时阴极那些管脚开关
157.   {
158.       digitalWrite(a, HIGH);
159.       digitalWrite(b, LOW);
160.       digitalWrite(c, LOW);
161.       digitalWrite(d, HIGH);
162.       digitalWrite(e, HIGH);
163.       digitalWrite(f, HIGH);
164.       digitalWrite(g, HIGH);
165.   }
```

```
166.
167.    void two()     //定义数字 2 时阴极那些管脚开关
168.    {
169.        digitalWrite(a, LOW);
170.        digitalWrite(b, LOW);
171.        digitalWrite(c, HIGH);
172.        digitalWrite(d, LOW);
173.        digitalWrite(e, LOW);
174.        digitalWrite(f, HIGH);
175.        digitalWrite(g, LOW);
176.    }
177.
178.    void three()     //定义数字 3 时阴极那些管脚开关
179.    {
180.        digitalWrite(a, LOW);
181.        digitalWrite(b, LOW);
182.        digitalWrite(c, LOW);
183.        digitalWrite(d, LOW);
184.        digitalWrite(e, HIGH);
185.        digitalWrite(f, HIGH);
186.        digitalWrite(g, LOW);
187.    }
```

```
188.
189.    void four()     //定义数字 4 时阴极那些管脚开关
190.    {
191.        digitalWrite(a, HIGH);
192.        digitalWrite(b, LOW);
193.        digitalWrite(c, LOW);
194.        digitalWrite(d, HIGH);
195.        digitalWrite(e, HIGH);
196.        digitalWrite(f, LOW);
197.        digitalWrite(g, LOW);
198.    }
199.
200.    void five()     //定义数字 5 时阴极那些管脚开关
201.    {
202.        digitalWrite(a, LOW);
203.        digitalWrite(b, HIGH);
204.        digitalWrite(c, LOW);
205.        digitalWrite(d, LOW);
206.        digitalWrite(e, HIGH);
207.        digitalWrite(f, LOW);
208.        digitalWrite(g, LOW);
209.    }
```

```
210.
211.   void six()     //定义数字 6 时阴极那些管脚开关
212.   {
213.       digitalWrite(a, LOW);
214.       digitalWrite(b, HIGH);
215.       digitalWrite(c, LOW);
216.       digitalWrite(d, LOW);
217.       digitalWrite(e, LOW);
218.       digitalWrite(f, LOW);
219.       digitalWrite(g, LOW);
220.   }
221.
222.   void seven()     //定义数字 7 时阴极那些管脚开关
223.   {
224.       digitalWrite(a, LOW);
225.       digitalWrite(b, LOW);
226.       digitalWrite(c, LOW);
227.       digitalWrite(d, HIGH);
228.       digitalWrite(e, HIGH);
229.       digitalWrite(f, HIGH);
230.       digitalWrite(g, HIGH);
231.   }
```

```
232.
233.    void eight()     //定义数字 8 时阴极那些管脚开关
234.    {
235.        digitalWrite(a, LOW);
236.        digitalWrite(b, LOW);
237.        digitalWrite(c, LOW);
238.        digitalWrite(d, LOW);
239.        digitalWrite(e, LOW);
240.        digitalWrite(f, LOW);
241.        digitalWrite(g, LOW);
242.    }
243.
244.    void nine()     //定义数字 9 时阴极那些管脚开关
245.    {
246.        digitalWrite(a, LOW);
247.        digitalWrite(b, LOW);
248.        digitalWrite(c, LOW);
249.        digitalWrite(d, LOW);
250.        digitalWrite(e, HIGH);
251.        digitalWrite(f, LOW);
252.        digitalWrite(g, LOW);
253.    }
```

复制代码

这次的代码为简易的秒表，效果如下，精准度不是很高，需要大家微调参数。

arduino 学习笔记 27 DHT11 数字温湿度传感器的使用

概述

DHT11 数字温湿度传感器是一款含有已校准数字信号输出的温湿度复合传感器。它应用专用的数字模块采集技术
和温湿度传感技术，确保产品具有极高的可靠性与卓越的长期稳定性。传感器包括一个电阻式感湿元件和一个 NTC 测
温元件，并与一个高性能 8 位单片机相连接。因此该产品具有品质卓越、超快响应、抗干扰能力强、性价比极高等优
点。每个 DHT11 传感器都在极为精确的湿度校验室中进行校准。校准系数以程序的形式储存在 OTP 内存中，传感器内
部在检测信号的处理过程中要调用这些校准系数。单线制串行接口，使系统集成变得简易快捷。超小的体积、极低的
功耗，信号传输距离可达 20 米以上，使其成为各类应用甚至最为苛刻的应用场合的最佳选则。
DHT11 数字温湿度传感器模块为 3 针 PH2.0 封装。连接方便。

性能描述

1. 供电电压：3-5.5V
2. 供电电流：最大 2.5Ma
3. 温度范围：0-50℃ 误差±2℃
4. 湿度范围：20-90%RH 误差±5%RH
5. 响应时间：1/e(63%) 6-30s
6. 测量分辨率分别为 8bit（温度）、8bit（湿度）
7. 采样周期间隔不得低于 1 秒钟
8. 模块尺寸：30x20mm

传感器的时序

DATA 用于微处理器与 DHT11 之间的通讯和同步,采用单总线数据格式,一次通讯时间 4ms 左右,数据分小数部分和

整数部分,具体格式在下面说明,当前小数部分用于以后扩展,现读出为零.操作流程如下:

一次完整的数据传输为 40bit,高位先出。

数据格式:

8bit 湿度整数数据+8bit 湿度小数数据

+8bi 温度整数数据+8bit 温度小数数据

+8bit 校验和

数据传送正确时校验和数据等于“8bit 湿度整数数据+8bit 湿度小数数据+8bi 温度整数数据+8bit 温度小数数据”所得结果的末 8 位。

用户 MCU 发送一次开始信号后,DHT11 从低功耗模式转换到高速模式,等待主机开始信号结束后,DHT11 发送响应信号,

送出 40bit 的数据,并触发一次信号采集,用户可选择读取部分数据.从模式下,DHT11 接收到开始信号触发一次温湿度采集,如果没有接收到主机发送开始信号,DHT11 不会主动进行温湿度采集.采集数据后转换到低速模式。

模块的使用

将 DHT11 模块接到 Arduino 传感器扩展板的模拟口 0

代码如下：

```
#define DHT11_PIN 0 // ADC0 接到模拟口 0
byte read_dht11_dat()
{
 byte i = 0;
 byte result=0;
 for(i=0; i< 8; i++){
 while(!(PINC & _BV(DHT11_PIN))); // wait for 50us
 delayMicroseconds(30);
 if(PINC & _BV(DHT11_PIN))
 result |=(1<<(7-i));
```

```
 while((PINC & _BV(DHT11_PIN))); // wait '1' finish
 }
 return result;
}
void setup()
{
 DDRC |= _BV(DHT11_PIN);
 PORTC |= _BV(DHT11_PIN);
 Serial.begin(19200);
 Serial.println("Ready");
}
void loop()
{
 byte dht11_dat[5];
 byte dht11_in;
 byte i;
 // start condition
 // 1. pull-down i/o pin from 18ms
 PORTC &= ~_BV(DHT11_PIN);
 delay(18);
 PORTC |= _BV(DHT11_PIN);
 delayMicroseconds(40);
```

```
DDRC &= ~_BV(DHT11_PIN);
delayMicroseconds(40);
dht11_in= PINC & _BV(DHT11_PIN);
if(dht11_in){
Serial.println("dht11 start condition 1 not met");
return;
}
delayMicroseconds(80);
dht11_in = PINC & _BV(DHT11_PIN);
if(!dht11_in){
Serial.println("dht11 start condition 2 not met");
return;
}
delayMicroseconds(80);
// now ready for data reception
for (i=0; i<5; i++)
dht11_dat[i] = read_dht11_dat();
DDRC |= _BV(DHT11_PIN);
PORTC |= _BV(DHT11_PIN);
byte dht11_check_sum =
dht11_dat[0]+dht11_dat[1]+dht11_dat[2]+dht11_dat[3];
// check check_sum
```

```
 if(dht11_dat[4]!= dht11_check_sum)
 {
 Serial.println("DHT11 checksum error");
 }
 Serial.print("Current humdity = ");
 Serial.print(dht11_dat[0], DEC);
 Serial.print(".");
 Serial.print(dht11_dat[1], DEC);
 Serial.print("% ");
 Serial.print("temperature = ");
 Serial.print(dht11_dat[2], DEC);
 Serial.print(".");
 Serial.print(dht11_dat[3], DEC);
 Serial.println("C ");
 delay(2000);
}
```

编译代码后下载到 Arduino 中，打开串口助手即可看见实际测量的温度与湿度。

慧净电子---ARDUINO 模块化创新视频教程

4、我们的产品收入一部分是赠送给慈善机构的,以免影响到你的善心.大家好,才是真的好（双方好评）。

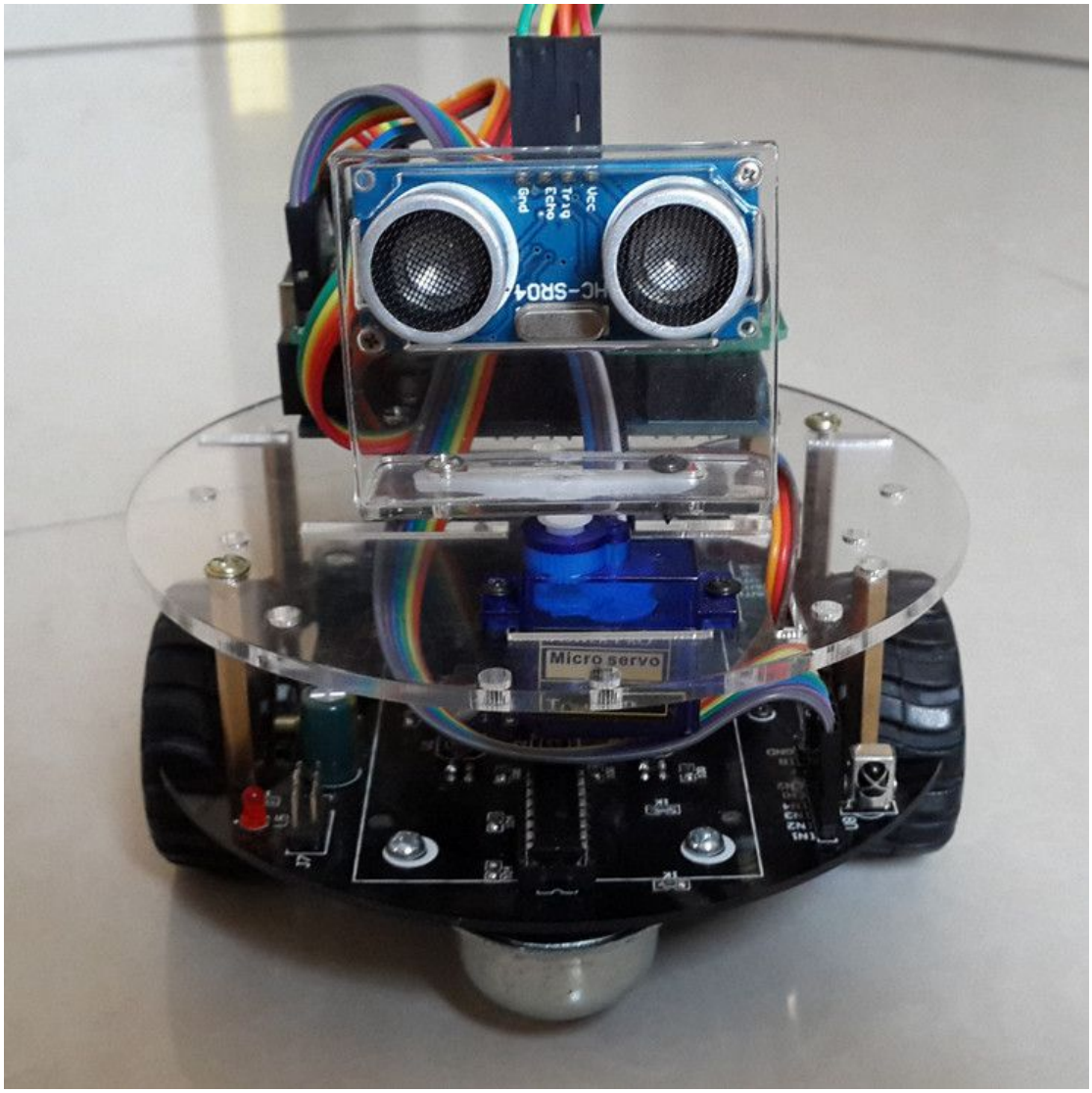
下面是有缘人看的，谢谢理解
善有善报，恶有恶报，不是不报，时候未到。
从古至今，阴司放过谁，大家得多行善。
行善积德，爱护动物，哪怕小蚂蚁也是生命。
可改变命运，可心想事成，有利保佑子孙后代更昌盛。
学习弟子规，教我们如何做人，看和谐拯救危机，教我们看宇宙。
看为什么不能吃它们，教我们慈悲心，看因果轮回纪录，教我们懂得因果报应。
切勿造恶，种瓜得瓜种豆得豆，一切都有过程，待成熟之时，福德或果报自来找你。

慧净
2008 年 8 月 8 日

慧净电子---ARDUINO 模块化创新视频教程

推荐你使用慧净ARDUINO智能机器人

网站：WWW.HJMCU.COM



慧净ARDUINO智能机器人可以蓝牙手机控制，可以超声波避障，等功能，只要你想得到，智能机器人就能做到的。