Arduino 学习手册

修订历史

版本	日期	原因
V1.0	2013/09/16	第一次发布
V1.1	2013/10/25	修改一些文字错误

请认准以下店铺购买：
http://www.ILovemcu.taobao.com
http://www.52dpj.taobao.com
http://www.epic-mcu.taobao.com

1.ARDUINO 简介

1.1 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 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。

1.2 为什么要选择 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。

1.3 ARDUINO 最流行的版本 UNO 的配置

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。

基本性能配置：

- ✓ 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/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。

2 ARDUINO 开发板的初次使用

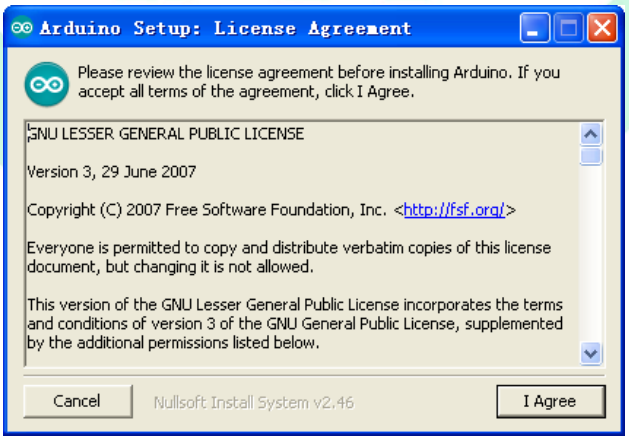
2.1 ARDUINO 开发环境简介

Arduino 的开发环境地址：

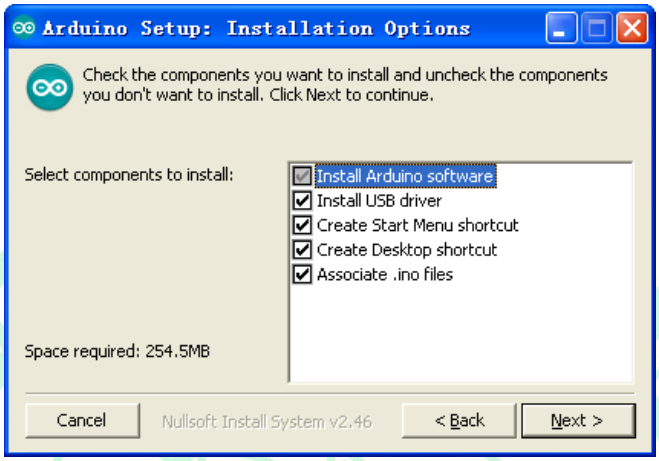
- ✓ 在 Arduino 独家整理资料包\1.开发环境\ arduino-1.0.5-windows.exe 找到
- ✓ 在 Arduino 官网下载最新开发环境，地址 http://arduino.cc/en/Main/Software

2.2 ARDUINO 开发环境安装

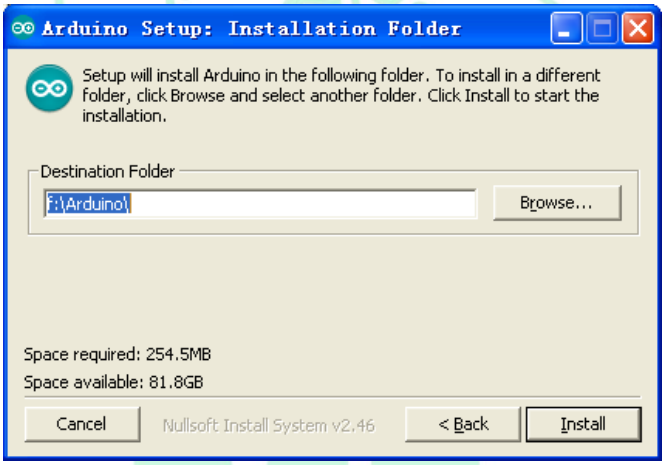
第一步:打开安装包 arduino-1.0.5-windows.exe



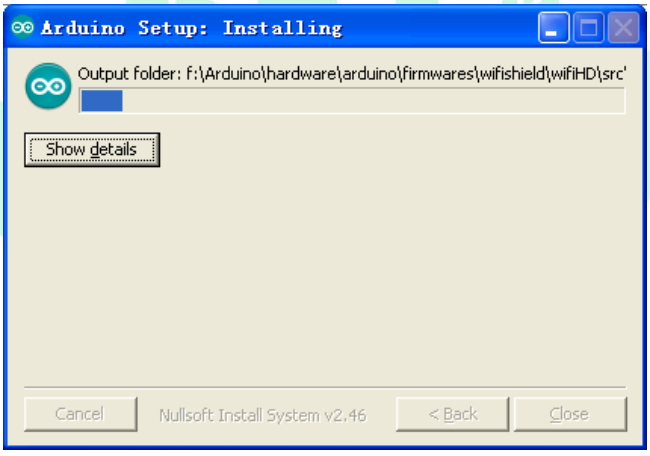
第二步：点击 I Agree



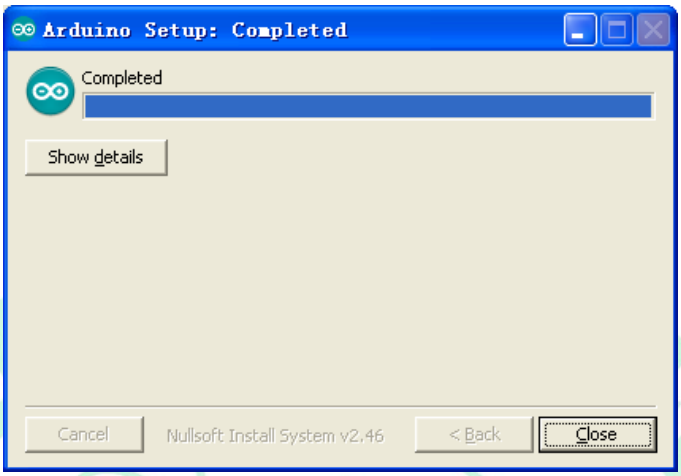
第三步：点击 NEXT



第四步：选择自己的安装路径，点击 Install



第五步：等待安装完成

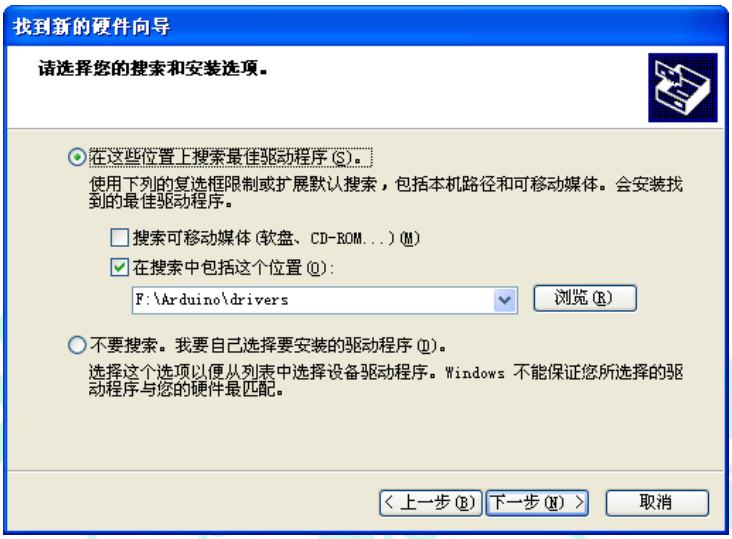


第六步：点击 Close

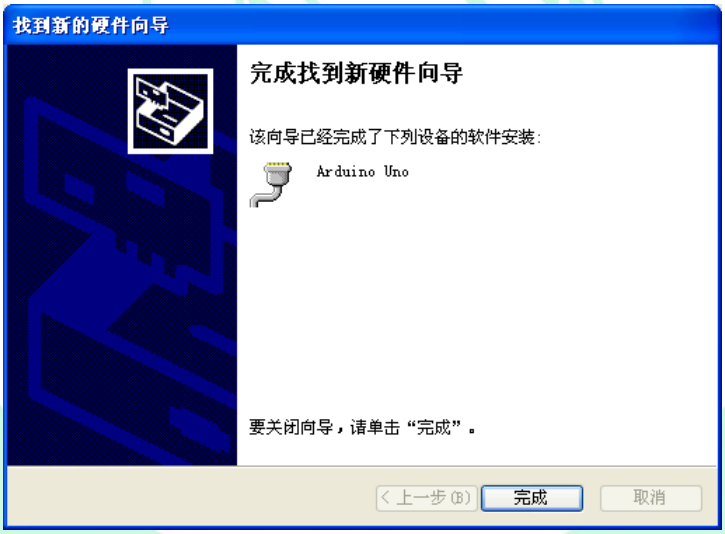
2.3 ARDUINO 驱动程序安装

第一步：使用配套的 USB 线连接电脑和 Arduino 开发板。

第二步：选择 从列表或指定位置安装（高级），点击 下一步



第三步：选择 在搜索中包括这个位置 ， 选择 Arduino 开发环境安装目录下的 drivers 点击 下一步

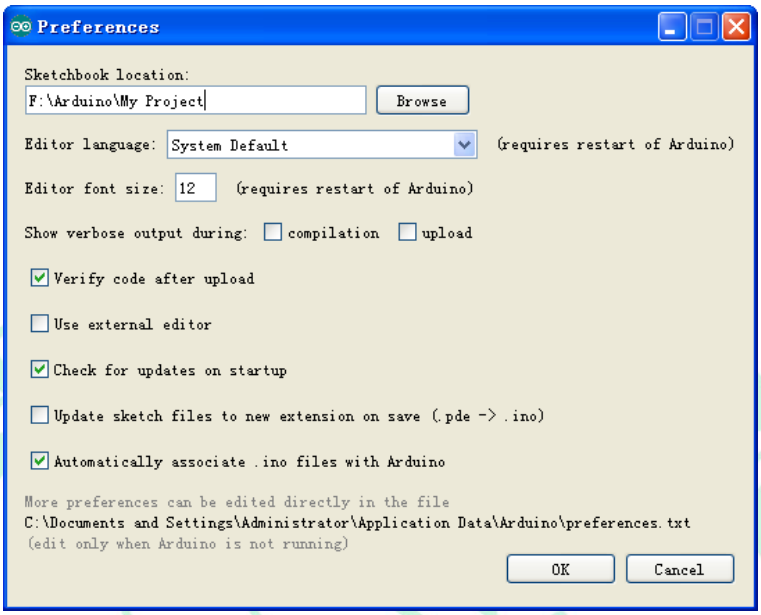


第四部：点击 完成 结束安装。

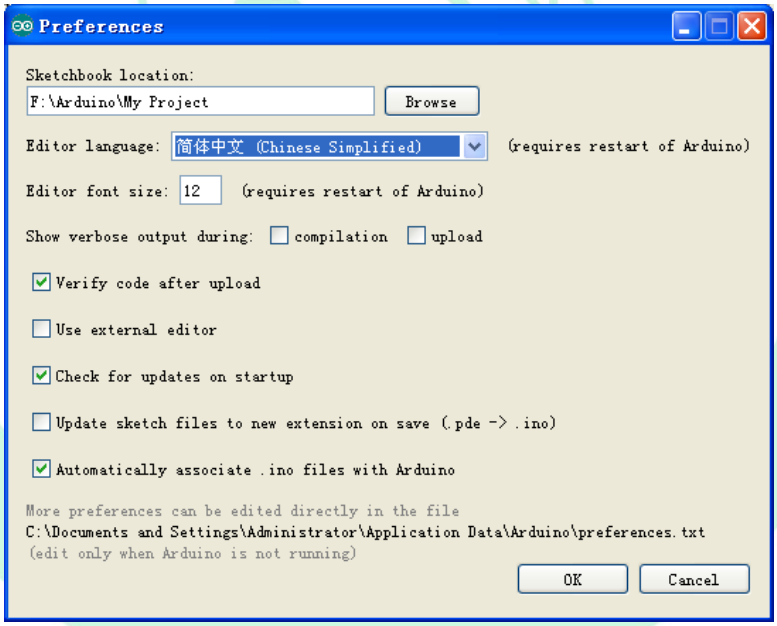
2.4 开发环境使用

1) 打开桌面上的 Arduino 开发环境快捷方式

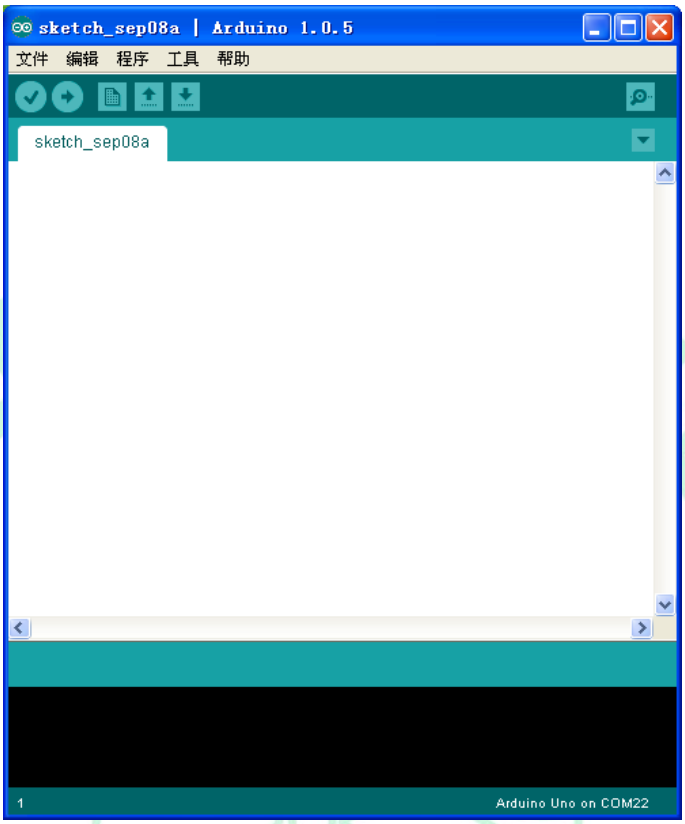
2) 选择 file 菜单下的 Preferenced



3) 选择简体中文开发环境，点 OK

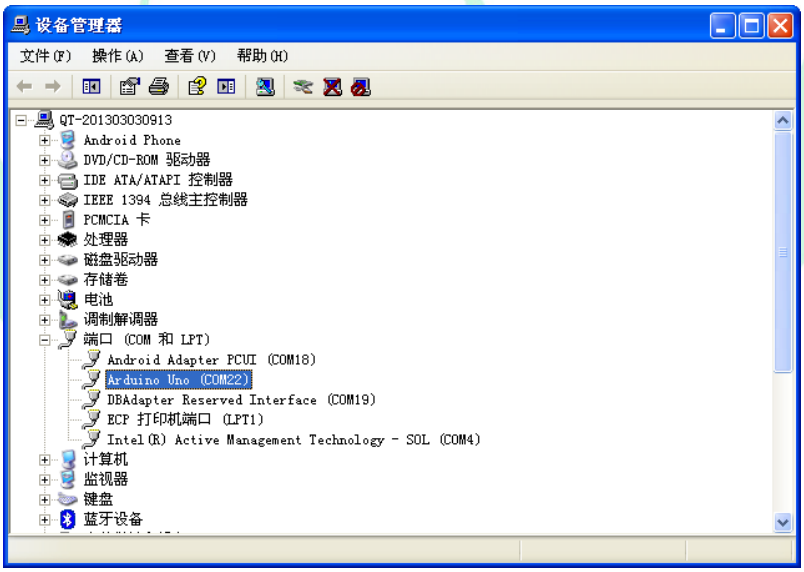


4) 关闭开发环境，重新打开，切换到中文界面，是不是舒服多了？(*^__^*) 嘻嘻……

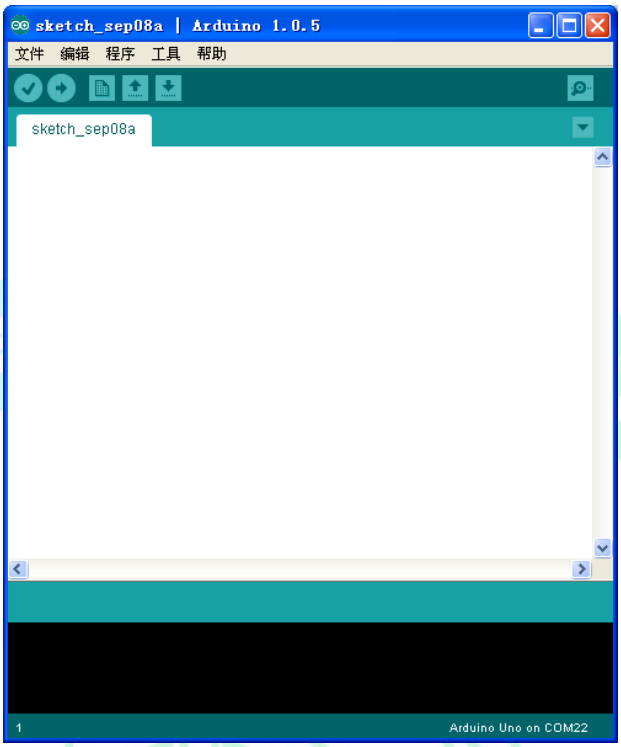


5) 在 工具 菜单下找到 板卡 ，选择自己的板子 ARDUINO UNO

6) 选择串口，串口可以在 我的电脑 右键 设备管理器 查看,

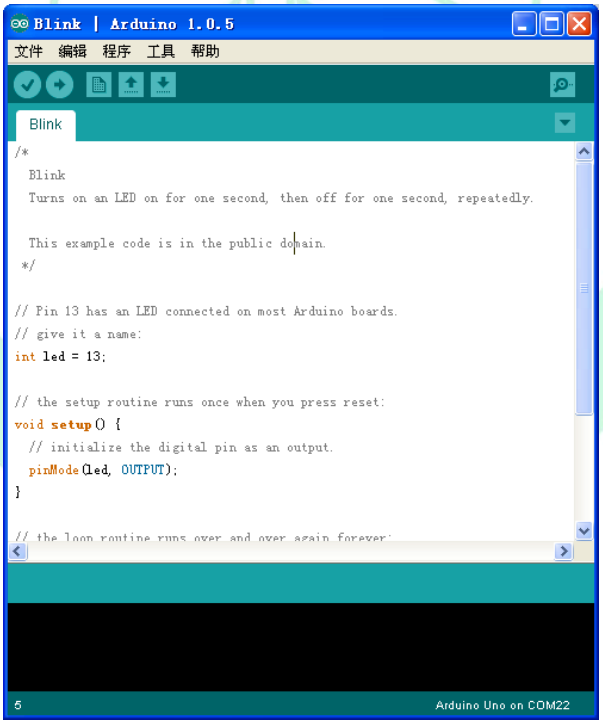


7) 我的电脑是 COM22，在 工具 菜单 串口 中选择 COM22
右下角出现 Arduino Uno on COM22



8) 打开一个例程测试板子是否运转正常

文件 –> 示例 -> 01.Basics -> Blink



9) 点击 编译 点击 下载，
看开发板上的 LED 以 1 秒的频率闪烁。（恭喜您开发板的初步操作已经掌握）
您可以去测试其它的系统自带例程，或者我们提供的例程了！

3 ARDUINO 例程讲解

例程讲解声明：

1. 软件源程序存放在 Arduino 独家整理资料包\3.例程 内，本手册不再一一粘贴出来，浪费篇幅增加大家观看教程难度。
2. 实验原理/代码讲解部分把小编认为程序最有看点或者最重要的部分载录出来重点讲解，其他部分请查看代码中文注释。所有代码注释均采用中文，让大家看程序更容易。
3. 实验原理图和连接图使用 Fritzing（Arduino 独家整理资料包\5.Arduino 面包板连线绘图软件\fritzing.2013.07.27.pc.zip）绘制。
4. 绘制好的电路连接底图存放在 Arduino 独家整理资料包\6.例程连线图原理图源文件 中，公开开放给大家，可以用 Fritzing 软件打开观看高清图纸。
5. 例程不断更新增加，请经常光顾我们店铺或者我们提供的下载链接，下载最新的实验手册。

3.1 LESSON1 HELLO WORLD

做为程序员编写所有程序的第一课，Hello World！是必须的一个环节，这一讲我们讲解一下如何使用 Arduino 的串口编写一句“Hello World！”，然后用 Arduino 发送给 PC 机。

后缀名为 ino 的为 Arduino 的项目文件，例如 LESSON1.ino 双击打开即可。

理论学习：

该程序中用到 Arduino 程序里面最常见的几个函数：

- ✓ void setup() 该函数用于编写 Arduino 的初始化内容，本例程内设置通讯波特率就放在这里，对于通讯的设置只许设置一次就可以了，所以把 Serial.begin(9600) 放在这里。
- ✓ Serial.begin(9600) 用于设置串口通讯的波特率，这里设置为 9600
- ✓ void loop() 该函数是 Arduino 的主程序部分，编写 Arduino 的核心代码。这里通过串口向电脑发送一个字符串，字符串内容为“Hello World！”
- ✓ Serial.println("Hello World!") 用于填写需要发送的字符串的内容。
- ✓ delay(1000); 这是毫秒延时函数，延时 1000 毫秒，即 1 秒

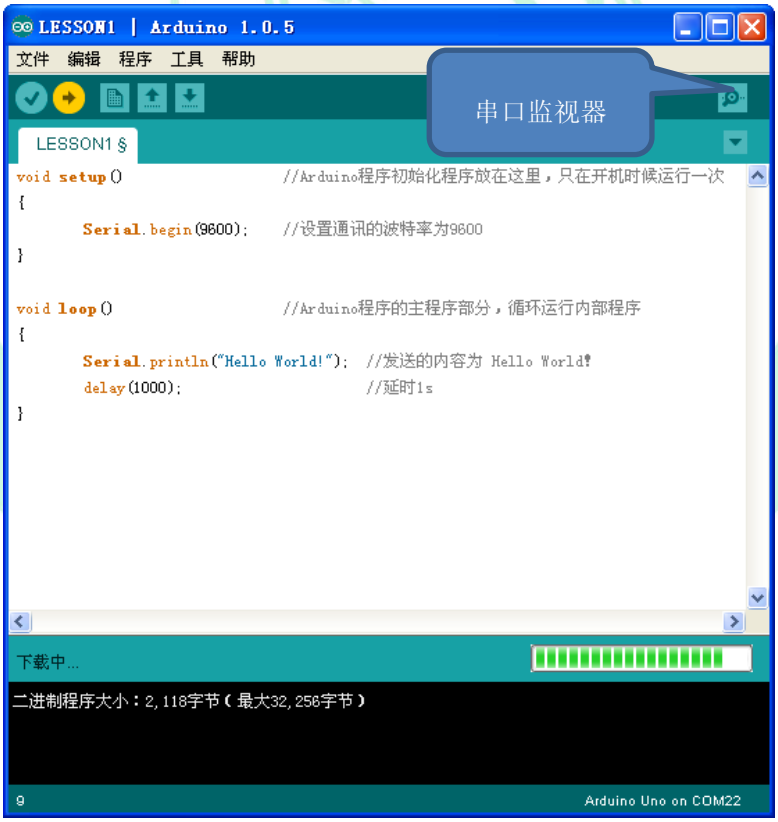


图 3.1.1 代码示例

把程序烧录进 Arduino 板子后，点击右上方的 串口监视器 ，设置监视器的波特率为 9600（在监控界面右下角），

我们发现每过 1 秒钟接收到一条 Hello World！

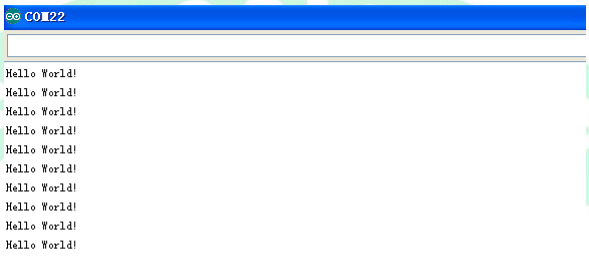


图 3.1.2 串口显示界面

3.2 LED 闪烁实验

本实验来自小编对系统自带实验 Blink 的优化。

实验现象：板载 LED 1 秒钟闪烁。

理论学习：

实验用到的新函数：

- ✓ #define led 13 //用于设置板子上的 13 引脚的名字为 led
  //比用系统源程序里面的 int led = 13，少了 2 个字节的开销
- ✓ pinMode(led, OUTPUT); //用于设置 LED 引脚为输出引脚
- ✓ digitalWrite(led, HIGH); //设置 LED 引脚输出高电平，点亮 LED
- ✓ digitalWrite(led, LOW); //设置 LED 引脚输出低电平,熄灭 LED

图 3.2.1 代码示例

3.3 按键控制 LED 亮灭

实验现象：本实验通过一个按键来控制一个发光二极管的亮灭。按键按一下 LED 点亮，再按一下 LED 熄灭。

备注：LED 长引脚为正极，短引脚为负极。

理论学习：

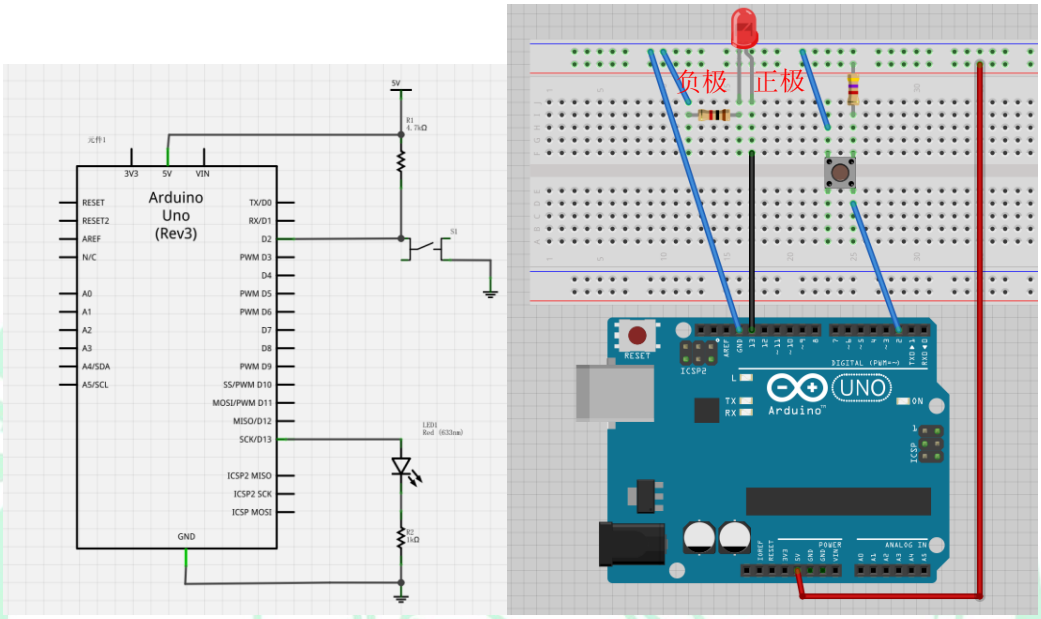
该实验主要难点：

- 如何扫描按键的状态？
- 如何防抖动？

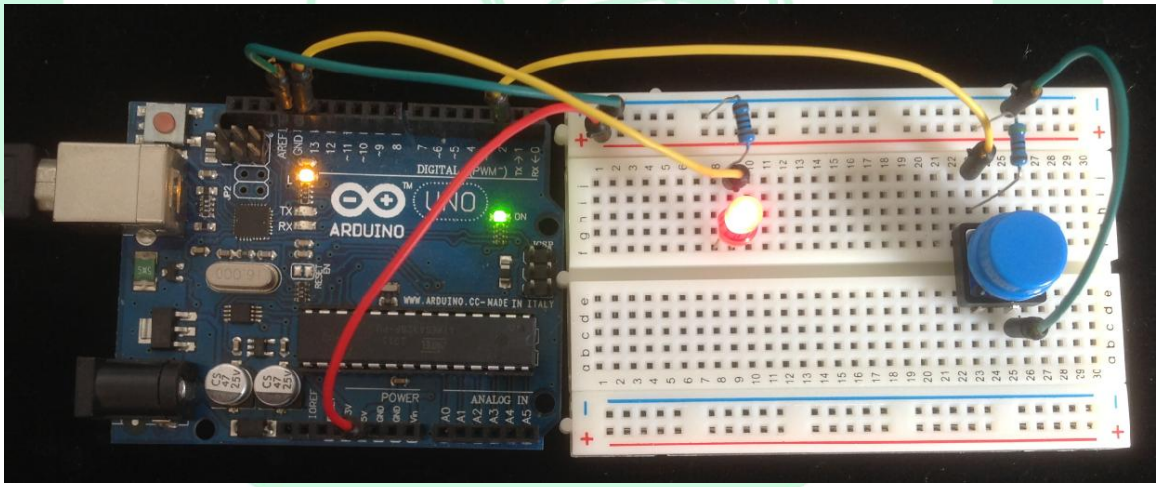
原理讲解：

- ✓ **按键检测原理：**通过把 Arduino 的数字 IO 设置为输入状态来监控按键是否按下，当按键未按下时候因为有上拉电阻存在，读到的电平为 HIGH，当按键按下时候因为按键引脚接地，所以读回来的电平为 LOW，由此判断按键是否按下。
- ✓ **按键去抖动原理：**因为人手的机械动作使按键按下时候会产生大概 20ms 左右的按键抖动，如果 Arduino 在这 20ms 内去检测 IO 口的电平很可能会检测出来不稳定的信号。因此，在检测到低电平后延迟 20ms 再次检测可以起到软件去抖动的作用。
  程序中用到的 delay(20); 就是这个作用

✓ **松手检测原理**：所谓的松手检测原理更简单，就是程序无限循环等待按键松开，即让程序不停的去检测 IO 状态，如果恢复了 HIGH 就跳出循环。
程序中这样写的 while(digitalRead(KEY) == 0);



3.3.1 原理图和连接图



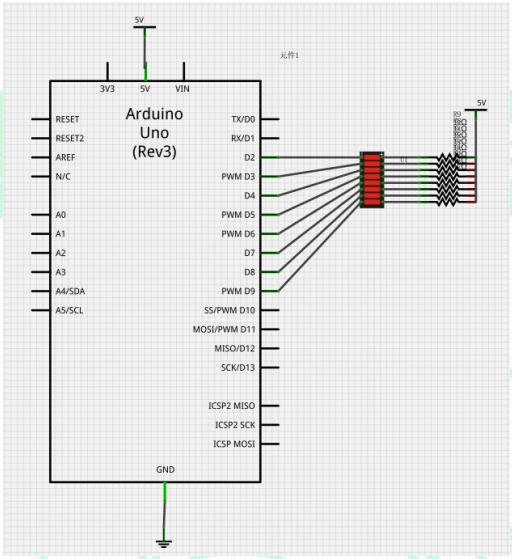
3.3.2 实际效果图

3.4 广告流水灯模块实验

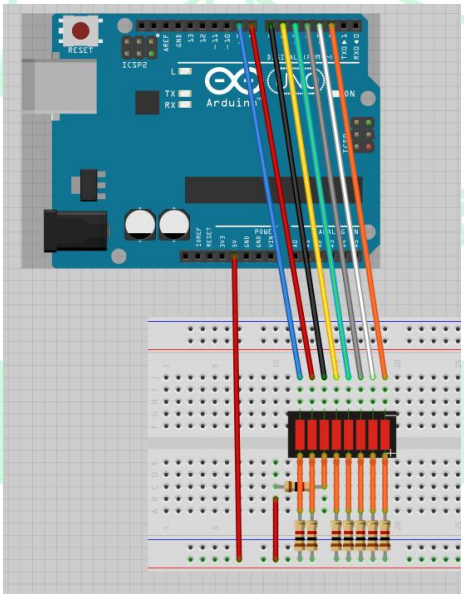
实验现象：按照自己的想法让 LED 模块显示各种效果。本程序 4 种流水效果，其它流水效果大家可以发挥自己的想象编写。

理论学习：

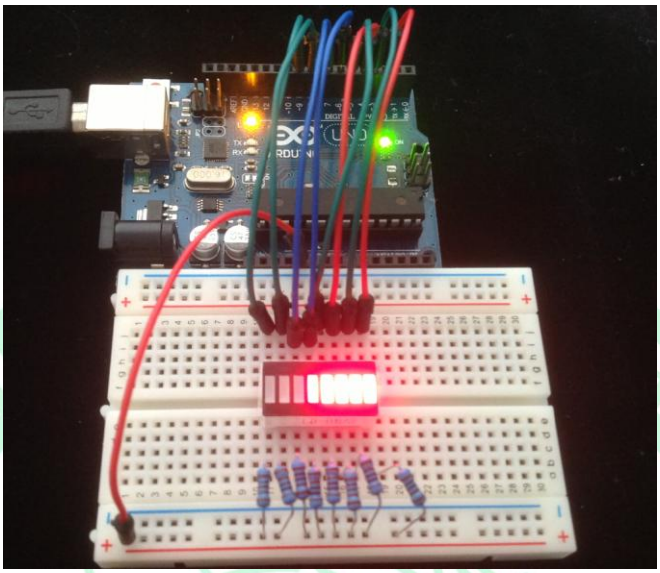
- ✓ LED 模块相当于 8 个 LED 的一个组合体，带文字的一面是阳极，另外一面是阴极。
- ✓ 本实验采用共阳极，IO 控制阴极模式。



3.4.1 原理图



3.4.2 连接图



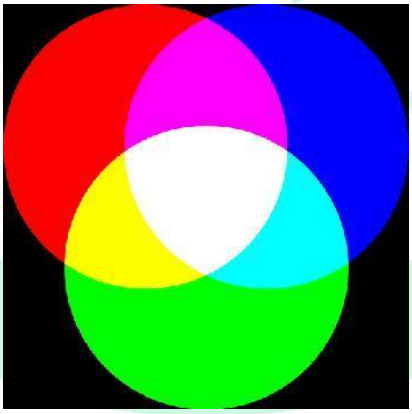
3.4.3 实物效果图

3.5 RGB LED 七彩跳变

实现现象：本实验使用 UNO 驱动一颗 RGB 三基色 LED 产生七色光的变化。

备注：引脚最长的为共阳的正极，剩下 3 个分别为红色、绿色、蓝色。

理论学习：三基色组合七色光原理：



红色+绿色=黄色

绿色+蓝色=青色

红色+蓝色=紫色

红色+绿色+蓝色=白色

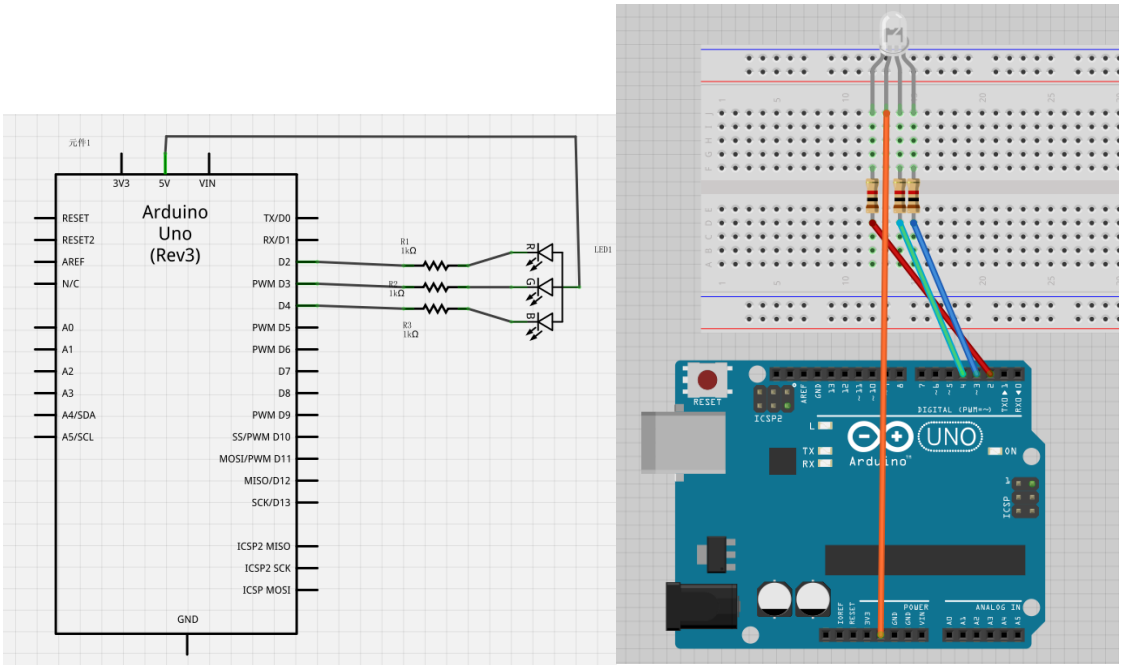
总共由 3 种基色“红绿蓝”组合出来七色光“红绿蓝青紫黄白”。

大家明白了三基色的原理就很容易的去编写这个例程了。我们想让哪个颜色 LED 点亮就把对应控制引脚的 IO 设置为 LOW 就可以了。（因为这个 LED 是共阳极的哦!）

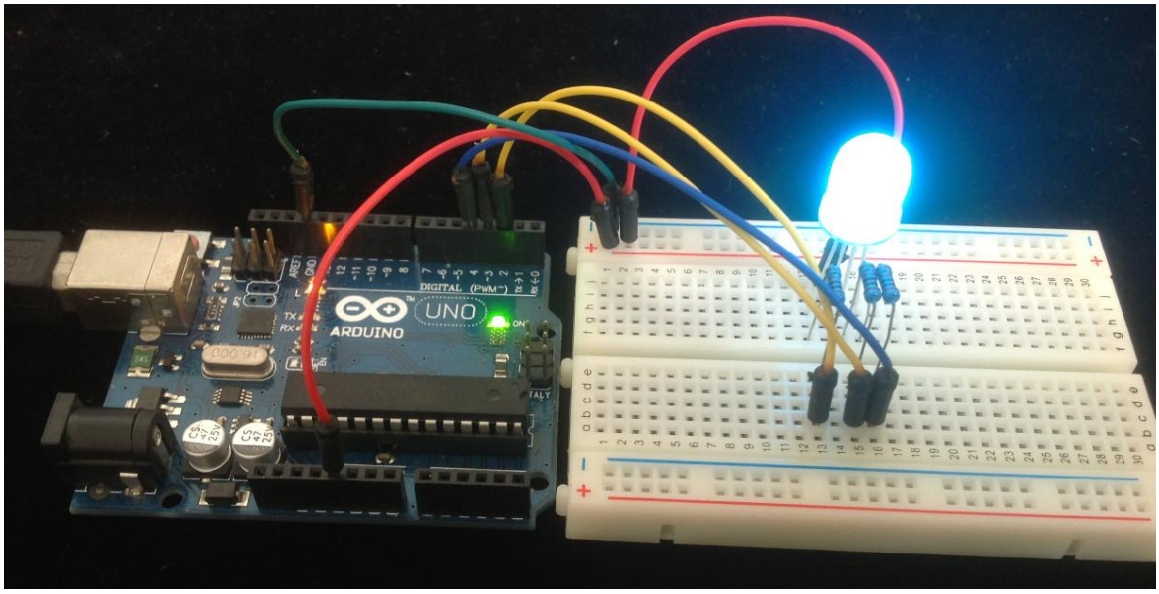
试验中使用 switch 语句来选择颜色

```
enum{Color_R,Color_G,Color_B,Color_RG,Color_RB,Color_GB,Color_RGB};                    //枚举所有颜色
//红，绿，蓝，黄，紫，青，白
void Change_Color(unsigned char data_color)        //选择颜色函数
{
    switch(data_color)
    {
        case Color_R:                                          //红色
            digitalWrite(LED_R,LOW);
            digitalWrite(LED_G,HIGH);
            digitalWrite(LED_B,HIGH);
            break;
        case Color_G:                                          //绿色
            digitalWrite(LED_R,HIGH);
            digitalWrite(LED_G,LOW);
            digitalWrite(LED_B,HIGH);
            break;
        case Color_B:                                          //蓝色
            digitalWrite(LED_R,HIGH);
            digitalWrite(LED_G,HIGH);
            digitalWrite(LED_B,LOW);
            break;
        case Color_RG:                                         //黄色
            digitalWrite(LED_R,LOW);
            digitalWrite(LED_G,LOW);
            digitalWrite(LED_B,HIGH);
            break;
        case Color_RB:                                         //紫色
            digitalWrite(LED_R,LOW);
            digitalWrite(LED_G,HIGH);
            digitalWrite(LED_B,LOW);
            break;
        case Color_GB:                                         //青色
            digitalWrite(LED_R,HIGH);
            digitalWrite(LED_G,LOW);
            digitalWrite(LED_B,LOW);
            break;
        case Color_RGB:                                        //白色
            digitalWrite(LED_R,LOW);
            digitalWrite(LED_G,LOW);
            digitalWrite(LED_B,LOW);
            break;
        default:
            break;
    }
```

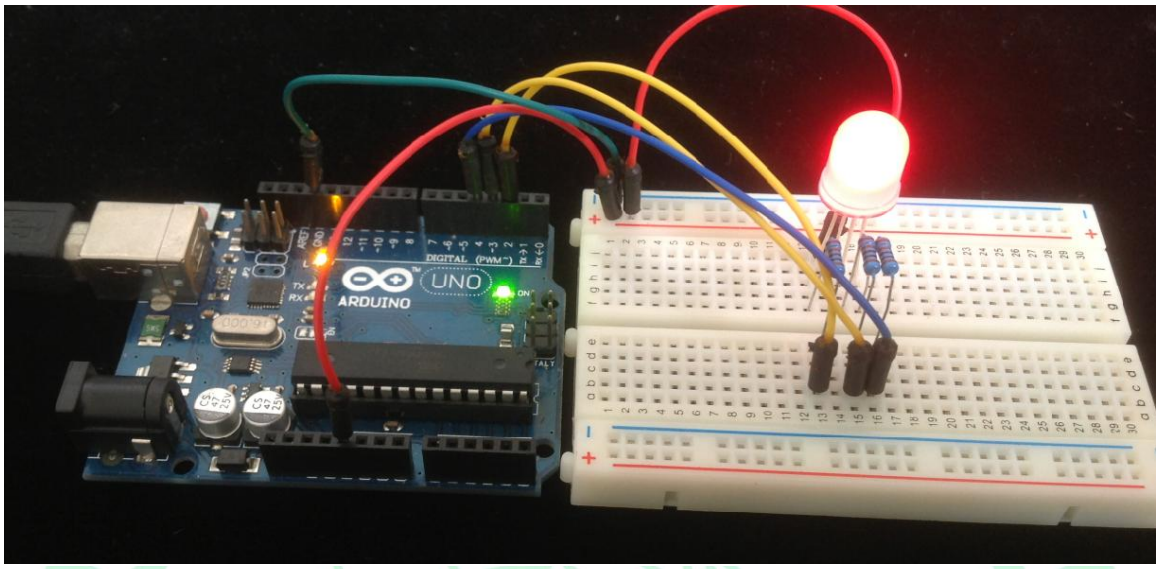
```
}
```



3.5.1 原理图和连接图



3.5.2 实际效果图（七彩跳变）



3.5.3 实际效果图（七彩跳变）

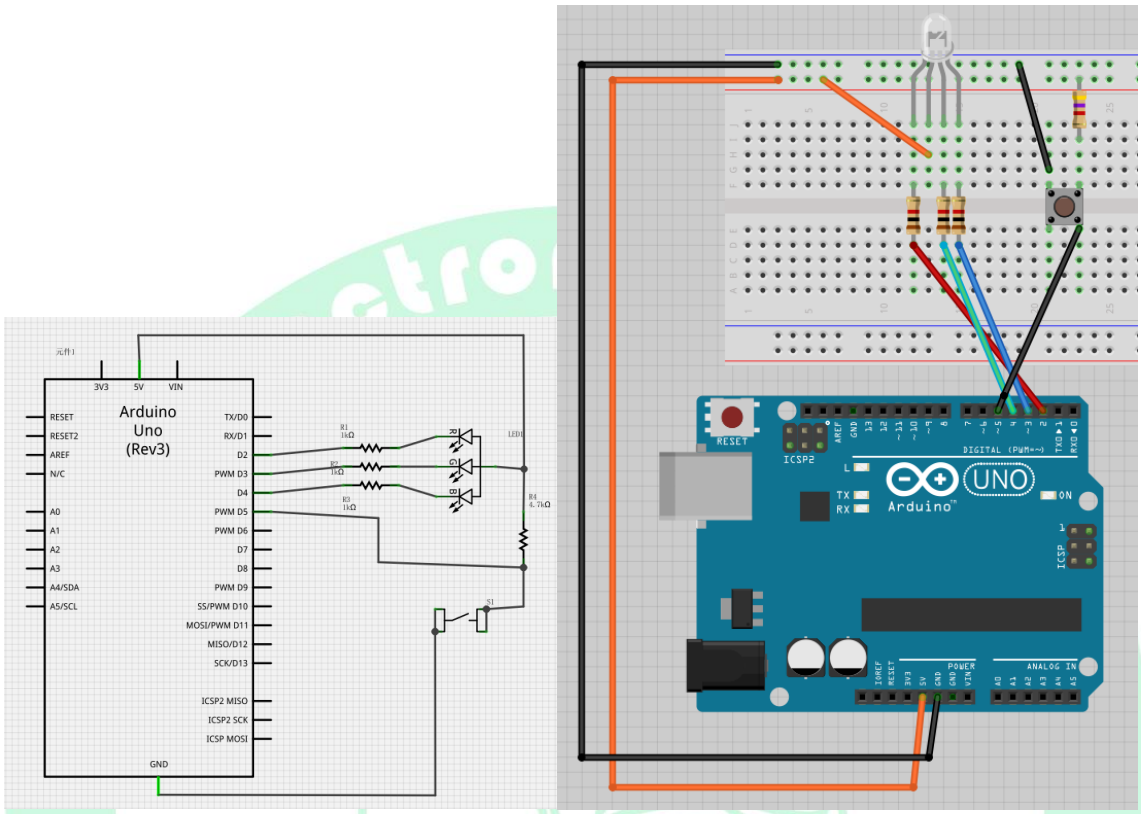
3.6 通过按键切换 LED 颜色

实验现象：按键每次按下 LED 切换一种颜色，依次为红，绿，蓝，黄，紫，青，白；

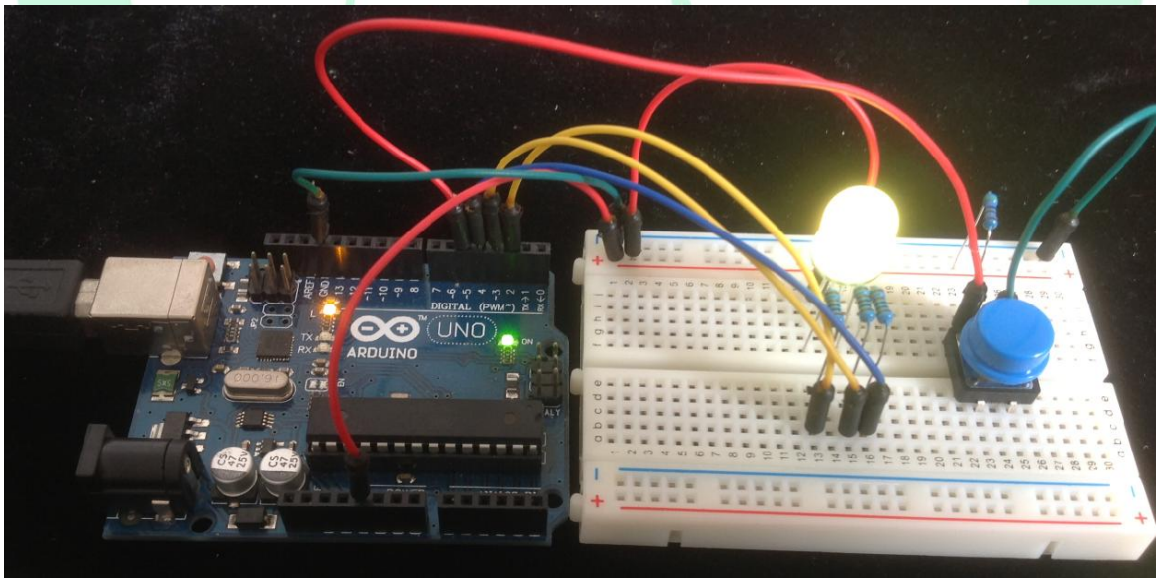
理论学习：

- ✓ 学会枚举语句的应用
  enum{Color_R,Color_G,Color_B,Color_RG,Color_RB,Color_GB,Color_RGB};
  相当于 Color_R = 0,Color_G = 1,……Color_RGB = 6,使用枚举语句使程序更易懂，阅读起来更简单舒服。
  这样在选择颜色时候，例如选择白色 Change_Color(Color_RGB);就可以了，如果写 Change_Color(6)当然也能看懂，但是需要花费很大的精力。
- ✓ 因为要 0-6 这 7 种颜色循环切换，因此当切换到 白色 时候要跳回 红色。
  程序中使用以下语句切换：

```
Color_Value++;                    //颜色变量+1
if(Color_Value == Color_RGB+1)          //当颜色变量 == 7
{
    Color_Value = Color_R;          //颜色变量清零，切换为红色
}
```



3.6.1 原理图和连接图



3.6.2 实际效果图

3.7 PWM 调光

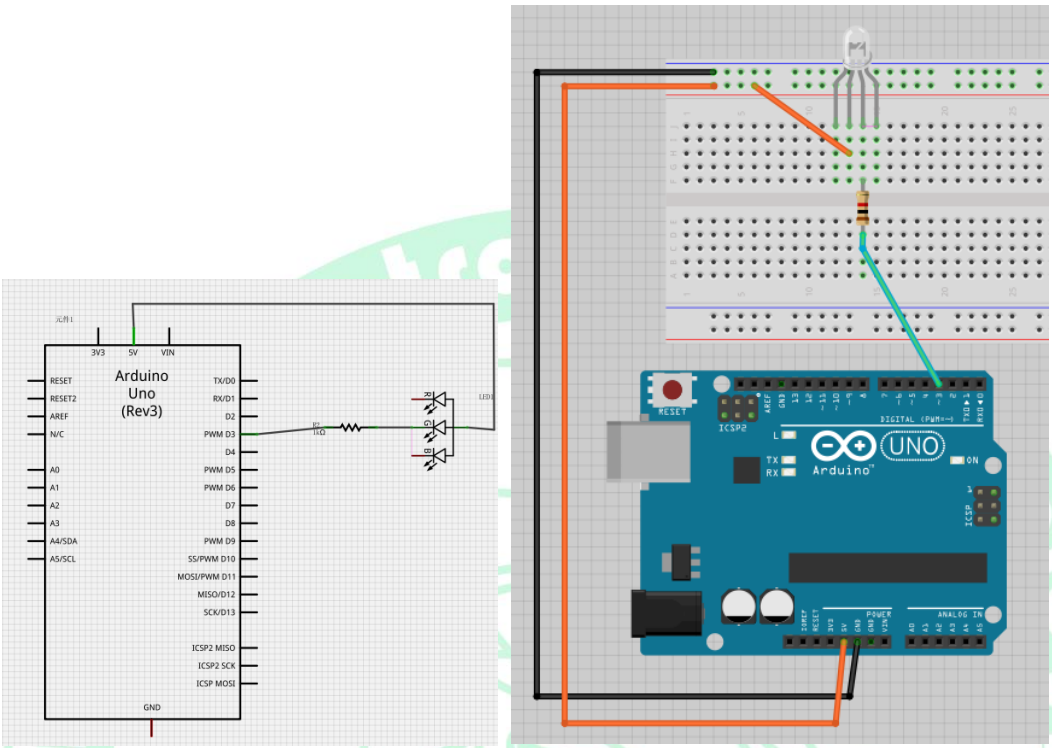
实验现象：LED 逐渐点亮，然后逐渐熄灭。

备注：因为 RGB LED 采用 10mm 的 LED，观察渐变效果更明显，因此这里使用 RGB LED 的其中一个颜色来测试。

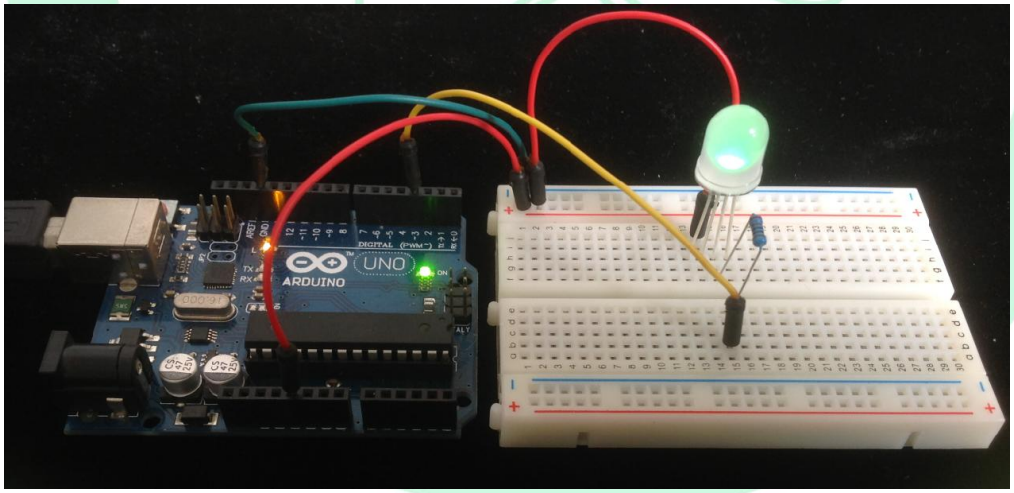
理论学习：

- ✓ 学会 PWM 的使用：Pulse Width Modulation 脉冲宽度调制，简称脉宽调制。是利用微处理器的数字输出来对模拟电路进行控制的一种非常有效的技术，广泛应用在从测量、通信到功率控制与变换的许多领域中。
  脉冲宽度调制（PWM）是一种对模拟信号电平进行数字编码的方法，由于计算机不能输出模拟电压，而只能输出 0V 或 5V 的数字电压值，（0V 为 0；5V 为 1）所以通过高分辨率计数器，利用方波的占空比被调制的方法对一个具体模拟信号的电平进行编码。 但 PWM 信号仍然是数字的，因为在给定的任意时刻，直流供电要么是 5V(数字值为 1)，要么是 0V（数字值为 0）。电压或电流源以一种通(ON)、断(OFF)的重复脉冲序列加到模拟负载上，只要带宽足够，任何模拟值都可以使用 PWM 进行编码。
- ✓ 输出的电压值是通过通和断的时间进行计算的，计算公式为：
  输出电压=（ 接通时间/ 脉冲时间 ）* 最大电压值
- ✓ PWM 的三个基本参数：
  1、脉冲宽度变化幅度（最小值/最大值）
  2、脉冲周期（1 秒内脉冲频率个数的倒数）
  3、电压高度（例如：0V-5V）
  Arduino UNO 控制器上有 6 个 PWM 接口分别是数字接口 3、5、6、9、10、11
- ✓ Arduino 中的设置 PWM 的语句：

```
analogWrite（pin,value）；
// pin：用于输入数值的引脚。
//value：占空比：0（完全关闭）到 255（完全打开）之间。
```



3.7.1 原理图和连接图



3.7.2 实际效果图

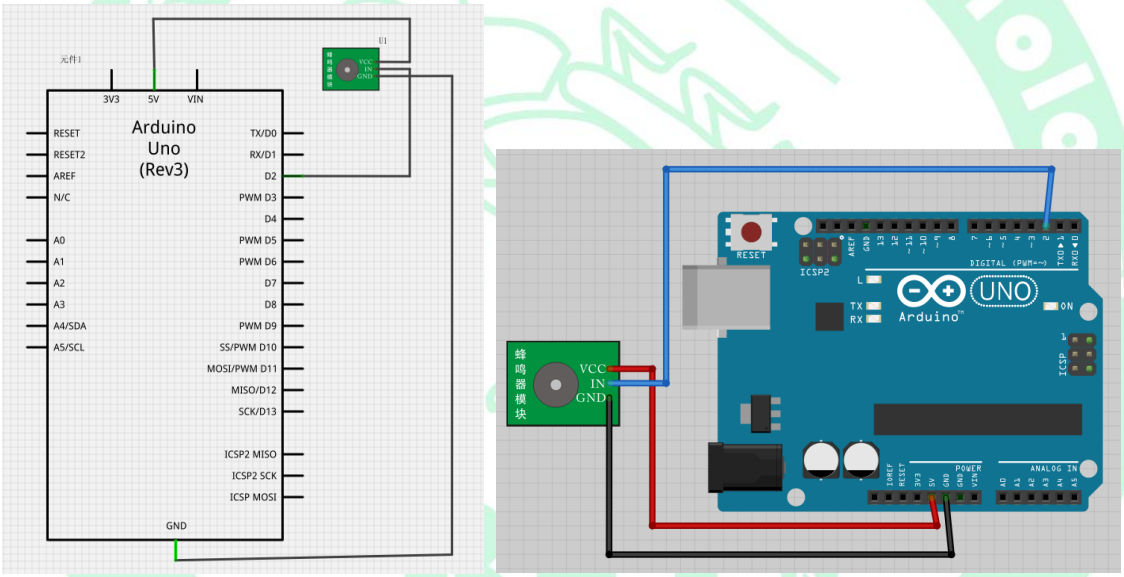
3.8 蜂鸣器模块实验

实验现象：蜂鸣器发出滴滴的响声。

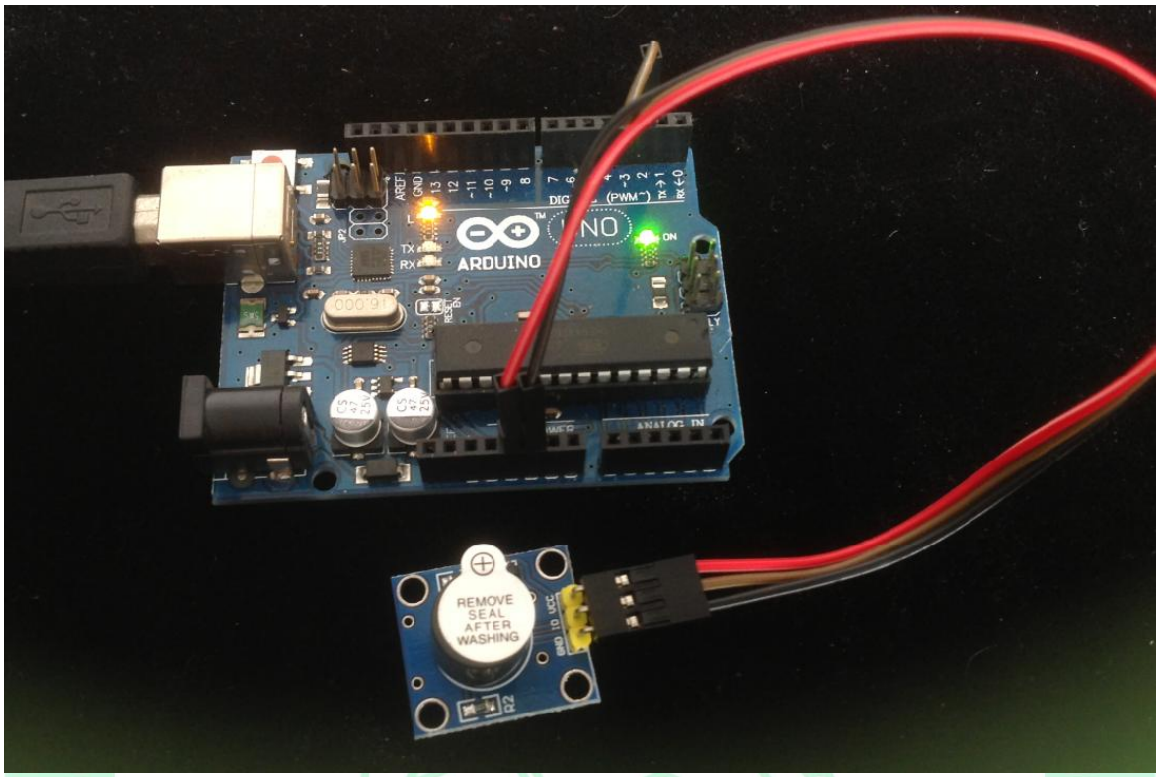
备注：如果想让蜂鸣器声音大一点，可以撕掉蜂鸣器上的贴纸。

理论学习：

- ✓ 有源蜂鸣器与无源蜂鸣器的区别：
  注意：这里的“源”不是指电源，而是指震荡源。
  也就是说，有源蜂鸣器内部带震荡源，所以只要一通电就会叫。
  而无源内部不带震荡源，所以如果用直流信号无法令其鸣叫。必须用 2K-5K 的方波去驱动它。
  套餐内提供的是有源蜂鸣器。
- ✓ 有源蜂鸣器的驱动电路：可以使用 NPN 三极管驱动，也可以通过 PNP 三极管驱动，
  NPN 三极管驱动时候当控制引脚给高电平时候蜂鸣器响，低电平不响。
  PNP 三极管驱动时候当控制引脚给低电平时候蜂鸣器响，高电平不响。
  套餐内提供的蜂鸣器模块是 NPN 三极管驱动的。



3.8.1 原理图和连接图



3.8.2 实际效果图

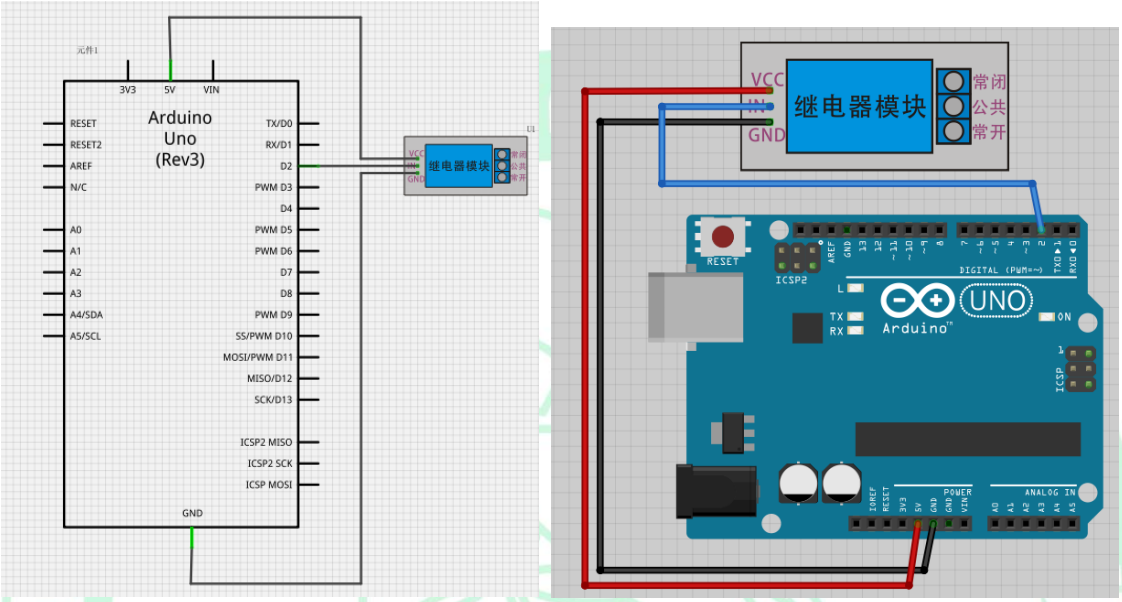
3.9 继电器模块实验

实验现象：继电器 2 秒吸合，2 秒关闭。可以直观从继电器模块上的 D2 指示灯查看状态。

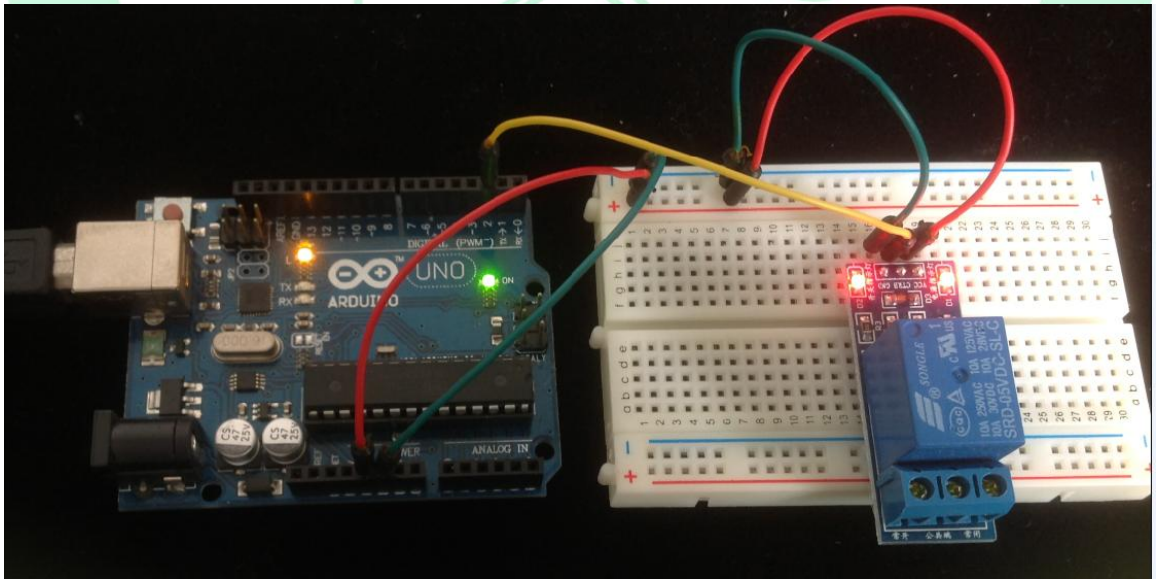
理论学习：

- ✓ 电磁继电器一般由铁芯、线圈、衔铁、触点簧片等组成的。只要在线圈两端加上一定的电压，线圈中就会流过一定的电流，从而产生电磁效应，衔铁就会在电磁力吸引的作用下克服返回弹簧的拉力吸向铁芯，从而带动衔铁的动触点与静触点（常开触点）吸合。当线圈断电后，电磁的吸力也随之消失，衔铁就会在弹簧的反作用力返回原来的位置，使动触点与原来的静触点（常闭触点）释放。这样吸合、释放，从而达到了在电路中的导通、切断的目的。对于继电器的“常开、常闭”触点，可以这样来区分：继电器线圈未通电时处于断开状态的静触点，称为“常开触点”；处于接通状态的静触点称为“常闭触点”。继电器一般有两股电路，为低压控制电路和高压工作电路。

✓ 继电器一般是低压控制高压的一种装置，顾一般需要通过三极管隔离驱动，本套餐内模块使用 NPN 的三极管驱动，当控制引脚给高电平时候继电器常开端吸合/D2 点亮，低电平时候，继电器常开端断开/D2 熄灭。



3.9.1 原理图和连接图



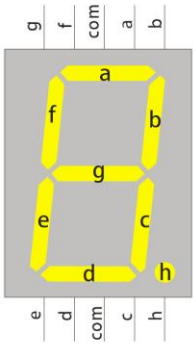
3.9.2 实际效果图

3.10 一位共阴数码管实验

实验现象：数码管依次显示 0-9。

理论学习：

- ✓ 数码管是一种半导体发光器件，其基本单元是发光二极管。按发光二极管单元连接方式可分为共阳极数码管和共阴极数码管。共阳数码管是指将所有发光二极管的阳极接到一起形成公共阳极(COM)的数码管，共阳数码管在应用时应将公共极 COM 接到+5V，当某一字段发光二极管的阴极为低电平时，相应字段就点亮，当某一字段的阴极为高电平时，相应字段就不亮。共阴数码管是指将所有发光二极管的阴极接到一起形成公共阴极(COM)的数码管，共阴数码管在应用时应将公共极 COM 接到地线 GND 上，当某一字段发光二极管的阳极为高电平时，相应字段就点亮，当某一字段的阳极为低电平时，相应字段就不亮
- ✓ 套餐内给用户配备有一个一位共阴数码管。
- ✓ 共阴数码管的引脚示意图：



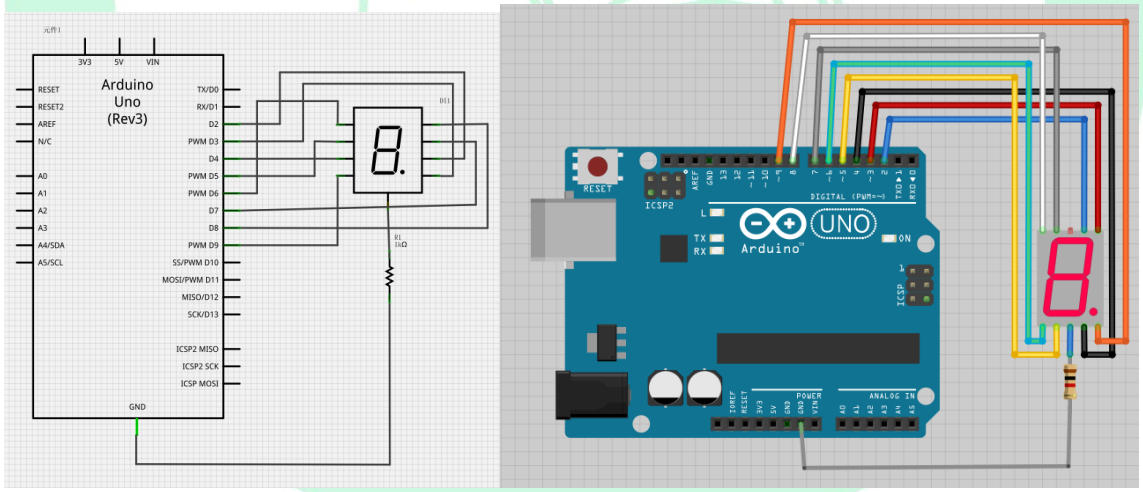
- ✓ 显示原理：如果想点亮 a 段的 LED，需要 a 引脚提供高电平，com 口提供低电平。建议在 COM 引脚串联一个电阻用于分压限流。
- ✓ 共阴数码管显示 0-9 时候的码值：

	h	g	f	e	d	c	b	a
0	0	0	1	1	1	1	1	1
1	0	0	0	0	0	1	1	0
2	0	1	0	1	1	0	1	1
3	0	1	0	0	1	1	1	1
4	0	1	1	0	0	1	1	0
5	0	1	1	0	1	1	0	1

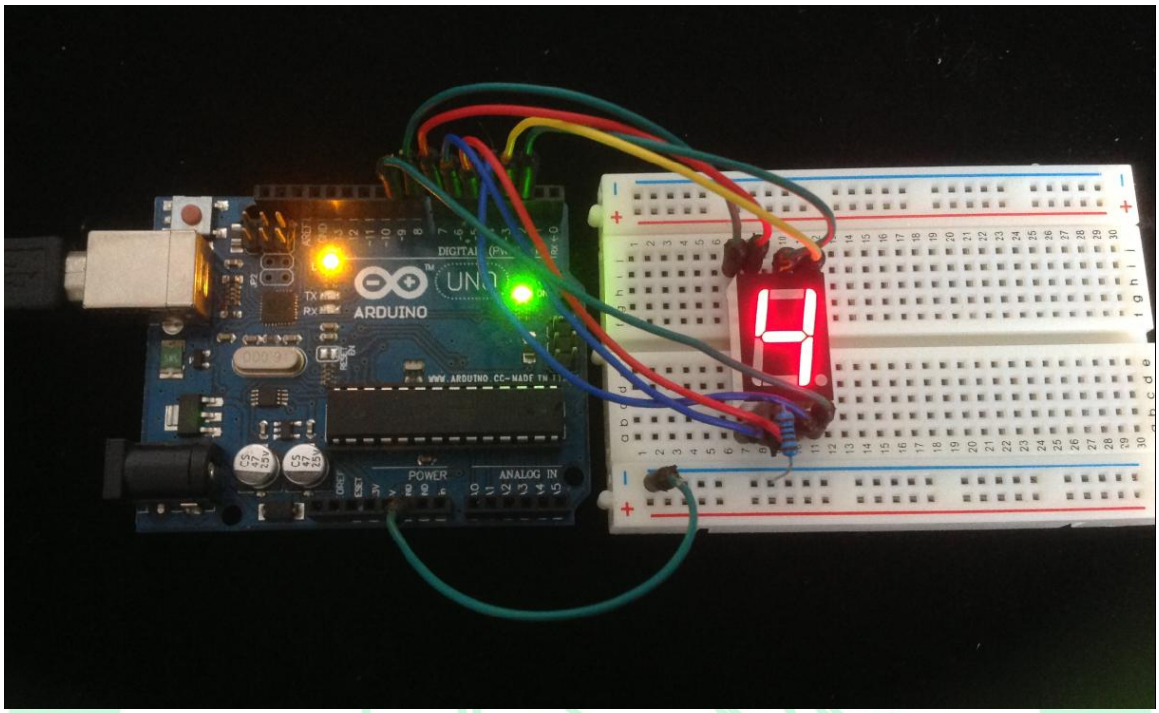
6	0	1	1	1	1	1	0	1
7	0	0	0	0	0	1	1	1
8	0	1	1	1	1	1	1	1
9	0	1	1	0	1	1	1	1

将以上存放在数组中

```
unsigned char table[10][8] =
{
{0,  0,  1,  1,  1,  1,  1,  1},          //0
{0,  0,  0,  0,  0,  1,  1,  0},          //1
{0,  1,  0,  1,  1,  0,  1,  1},          //2
{0,  1,  0,  0,  1,  1,  1,  1},          //3
{0,  1,  1,  0,  0,  1,  1,  0},          //4
{0,  1,  1,  0,  1,  1,  0,  1},          //5
{0,  1,  1,  1,  1,  1,  0,  1},          //6
{0,  0,  0,  0,  0,  1,  1,  1},          //7
{0,  1,  1,  1,  1,  1,  1,  1},          //8
{0,  1,  1,  0,  1,  1,  1,  1}           //9
};
```



3.10.1 原理图和连接图



3.10.2 实际效果图

3.11 四位共阴数码管实验

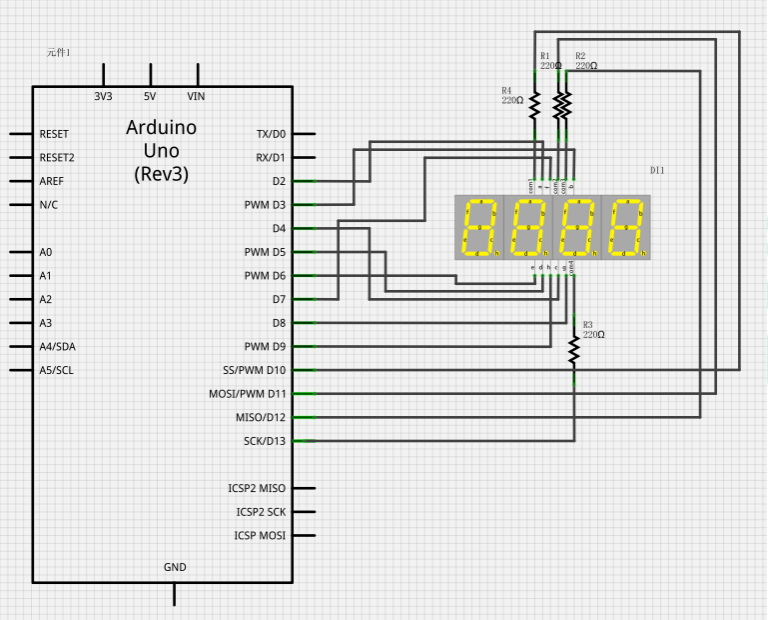
实验现象：数码管 1-4 位分别显示 1、2、3、4。

理论学习：

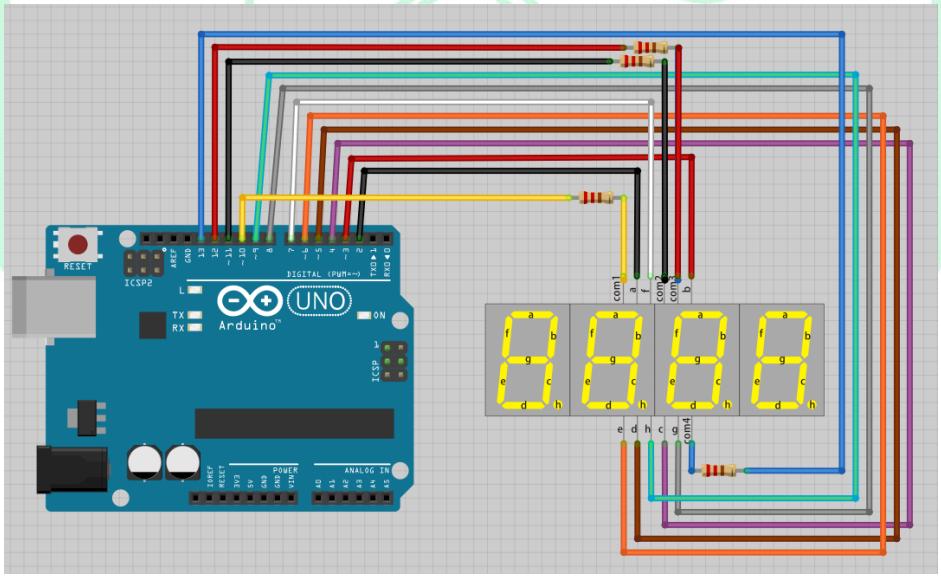
- ✓ 数码管动态扫描：动态显示的特点是将所有位数码管的段选线并联在一起，由位选线控制是哪一位数码管有效。这样一来，就没有必要每一位数码管配一个锁存器，从而大大地简化了硬件电路。选亮数码管采用动态扫描显示。所谓动态扫描显示即轮流向各位数码管送出字形码和相应的位选，利用发光管的余辉和人眼视觉暂留作用，使人的感觉好像各位数码管同时都在显示。
  通俗点说：动态扫描就是一位一位显示数码管的速度加快，速度达到几个 ms 以内。
- ✓ 动态扫描时候去除余晖（显示模糊）：在动态扫描时候切换位选时候先把段选的输出清零，然后再切换位选，这样可以有效去除余晖。程序中 void Display(unsigned char com,unsigned char num)函数中最前面的程序即为去除余晖的代码：

```
digitalWrite(SEG_A,LOW);              //去除余晖
digitalWrite(SEG_B,LOW);
digitalWrite(SEG_C,LOW);
digitalWrite(SEG_D,LOW);
```

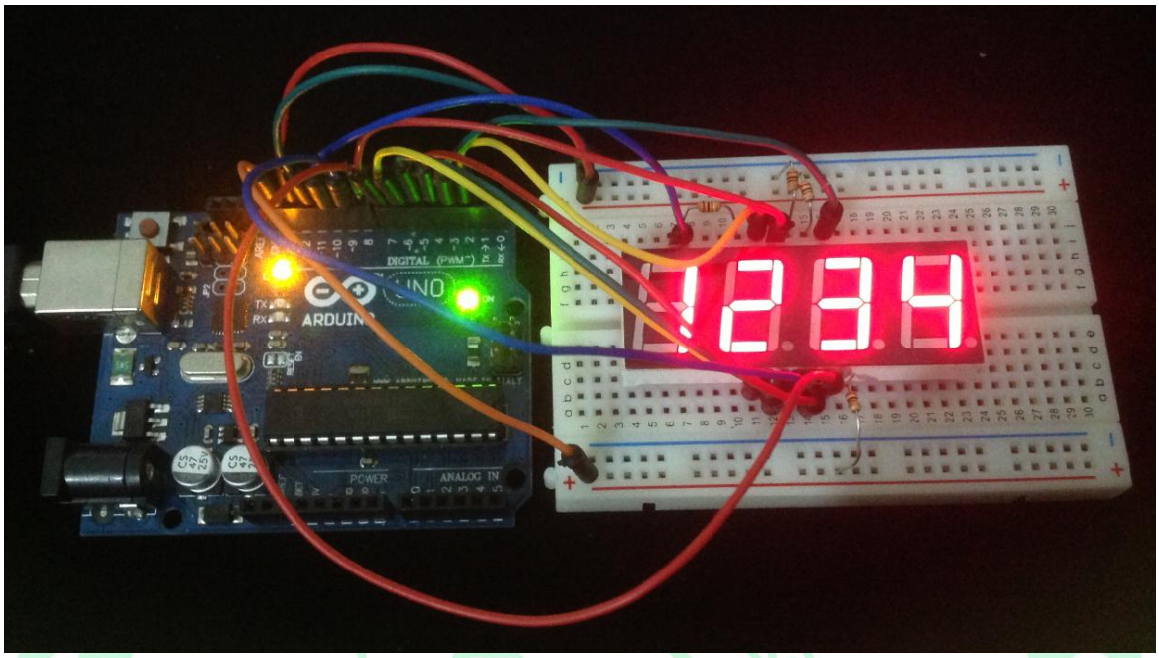
```
digitalWrite(SEG_E,LOW);
digitalWrite(SEG_F,LOW);
digitalWrite(SEG_G,LOW);
digitalWrite(SEG_H,LOW);
```



3.11.1 原理图



3.11.2 连接图



3.11.3 实物效果图

3.12 步进电机驱动实验

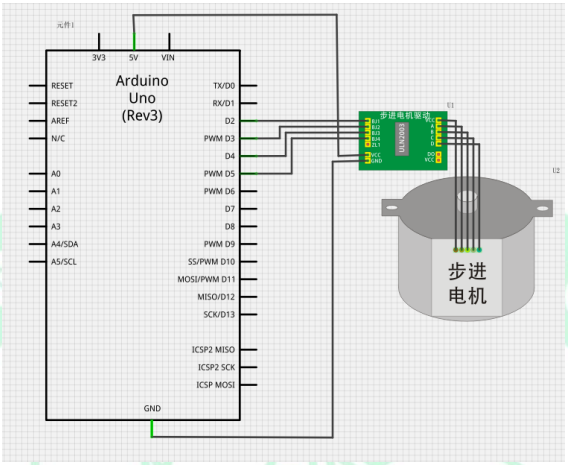
实验现象：步进电机逆时针旋转。

理论学习：

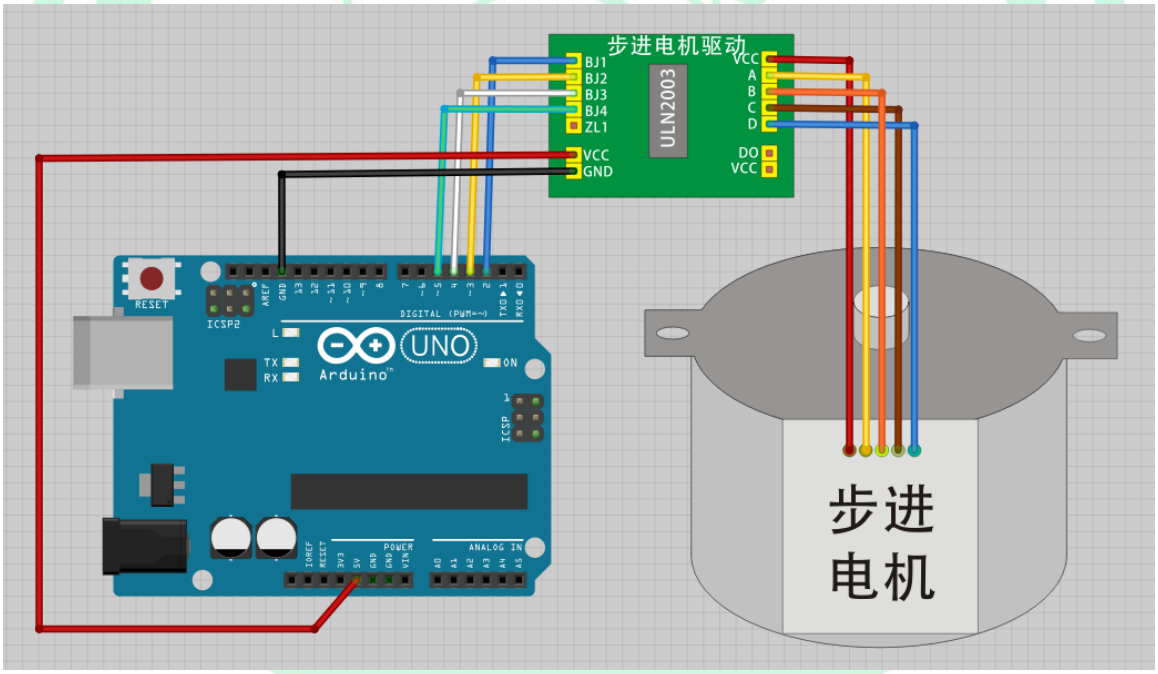
- ✓ 28BYJ-48 步进电机空载耗电在 50mA 以下，带 64 倍减速器，输出力矩比较大，可以驱动重负载，极适合开发板使用。注意：此款步进电机带有 64 倍减速器，与不带减速器的步进电机相比，转速显得较慢，为方便观察，可在输出轴处粘上一片小纸板。其中红色线为 VCC，其余 4 个为 4 个相位。
- ✓ 使用 ULN2003 达林顿驱动芯片驱动步进电机，板载 4 个 LED，可以指示相位状态。
- ✓ 步进电机相位控制，如果选择相位 A，单片机给驱动板 BJ1 引脚高电平，其它 BJ2/BJ3/BJ4 引脚低电平，达林顿管对应引脚输出电平会翻转，达林顿管输出低电平时候配合 VCC，可以驱动 A 相位。代码示例：

```
void Phase_A()
{
    digitalWrite(A1,HIGH);     //A1引脚高电平
    digitalWrite(B1,LOW);
    digitalWrite(C1,LOW);
    digitalWrite(D1,LOW);
}
```

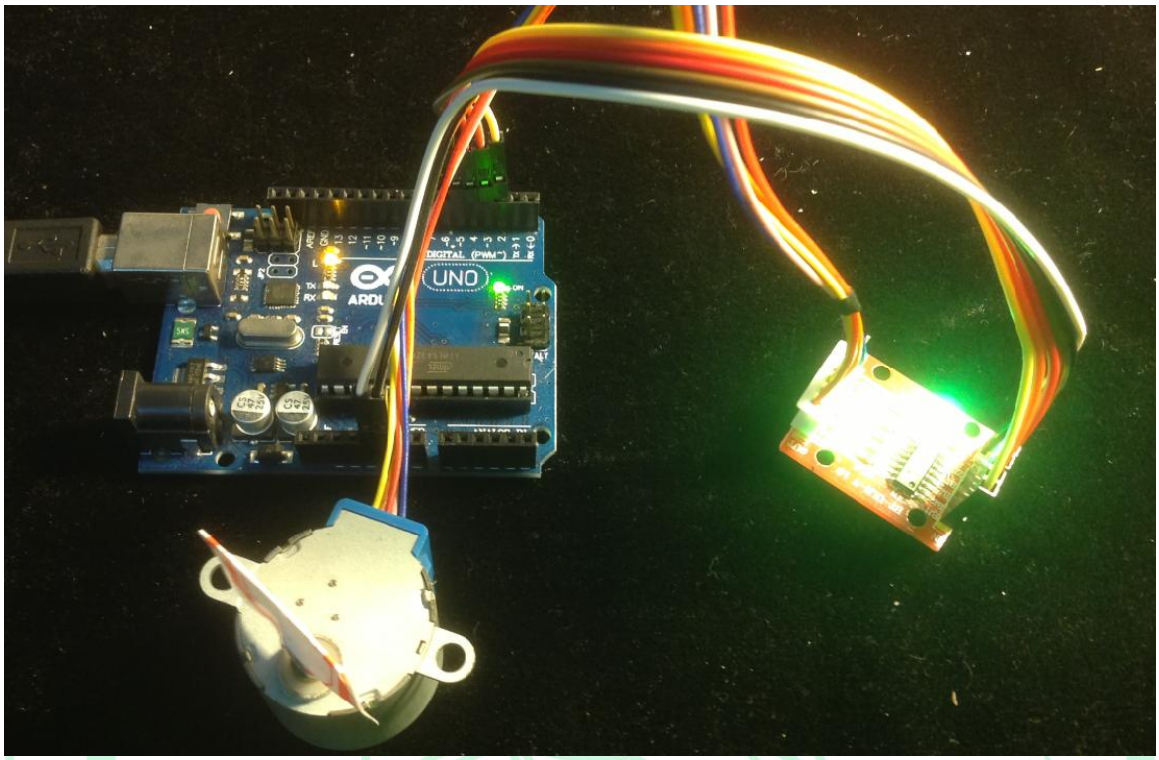
原理图和连接图：



3.12.1 原理图



3.12.2 连接图



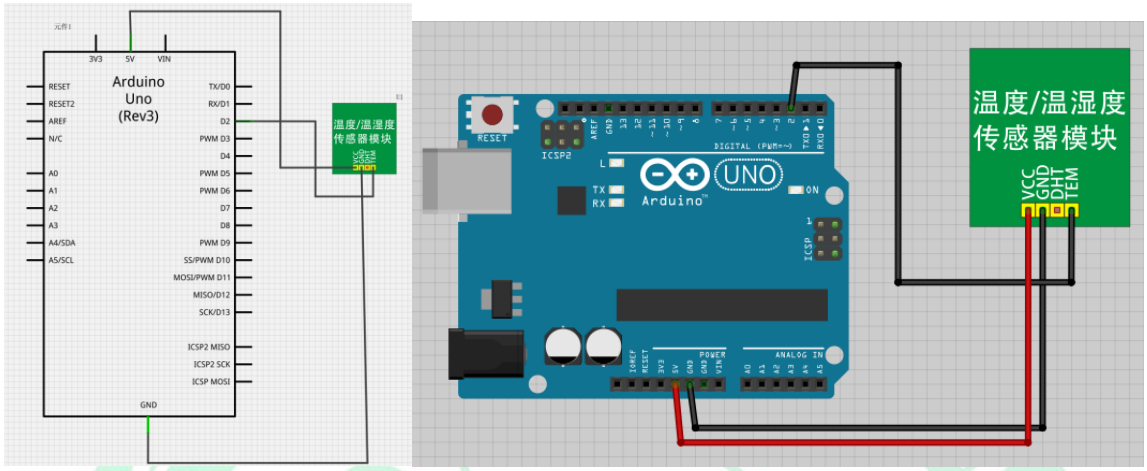
3.12.3 实物效果图

3.13 温度传感器 DS18B20 实验

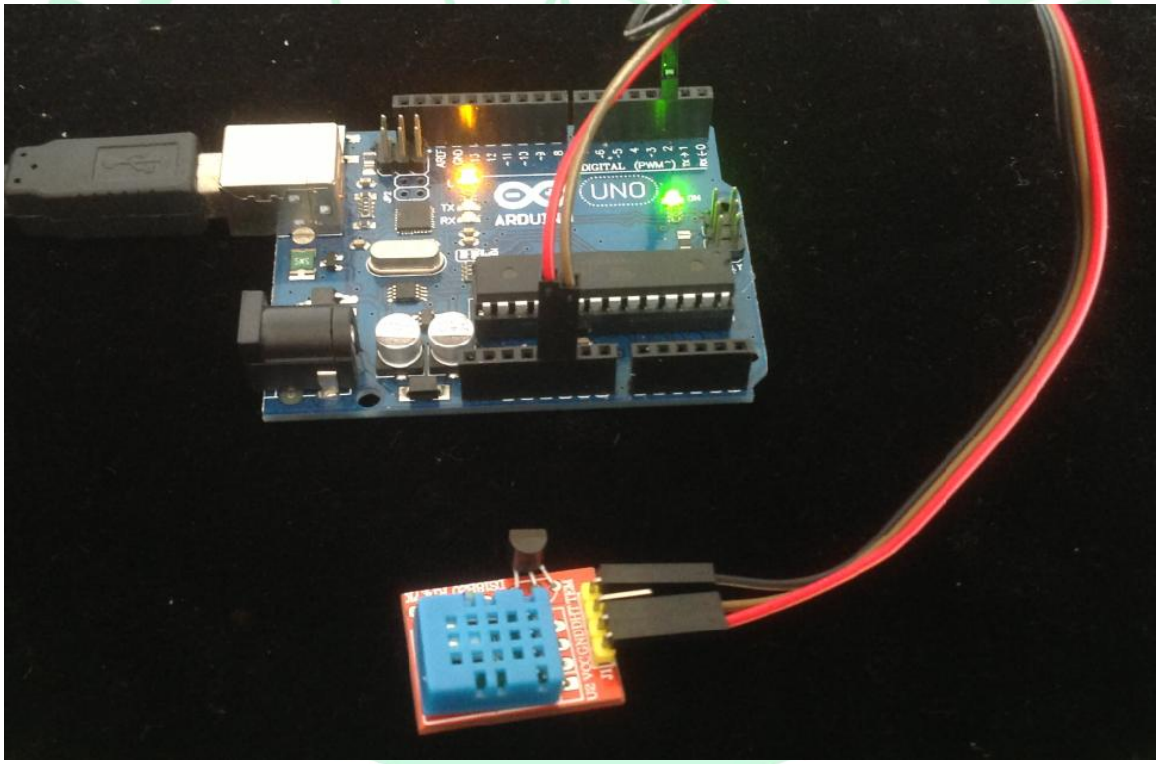
实验现象：Arduino 读取温度传感器 DS18B20 的值，然后计算出当前温度，通过串口发送给 PC，PC 串口调试助手显示。

理论学习：

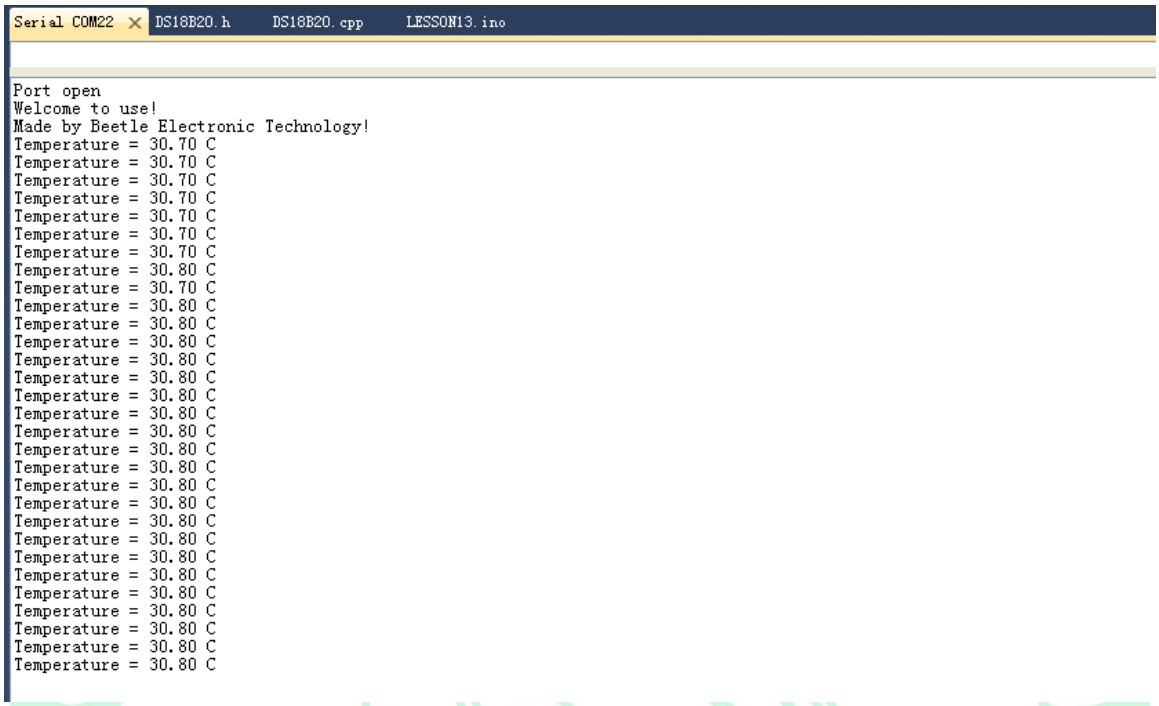
- ✓ DS18B20 数字温度传感器采用单总线模式，和 MCU 通讯仅需要一个 IO，测温范围-55~125℃
- ✓ 温度数字量转换时间 200ms（典型值），即每次读取的时间间隔最好大于 200ms
  其它细节的指标请查看 DS18B20 手册（Arduino 独家整理资料包\7.芯片资料\DS18B20 中文资料）
- ✓ 程序中调用头文件DS18B20.h后，可以使用 Get_temp();函数读取温度变量值，读取出来的温度值以unsigned int型格式返回，该数值为温度值的10倍，因此需要除以10切换位温度值。标志位flag_temper 可以确定当前温度是正值还是负值，如果flag_temper == 0,说明温度为正值，如果flag_temper == 1,则说明温度为负值，显示时候需要显示负号。



3.13.1 原理图和连接图



3.13.2 实物效果图



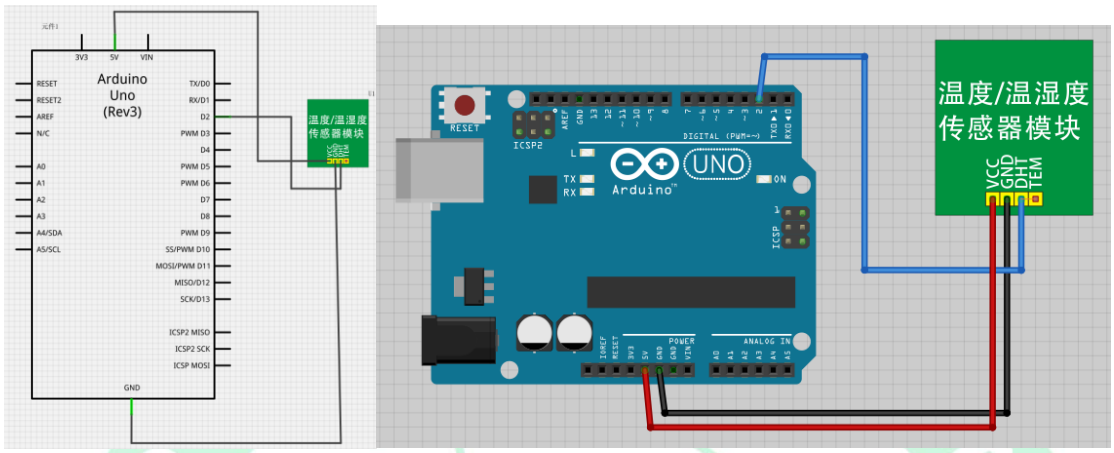
3.13.3 串口显示界面

3.14 湿度传感器 DHT11 实验

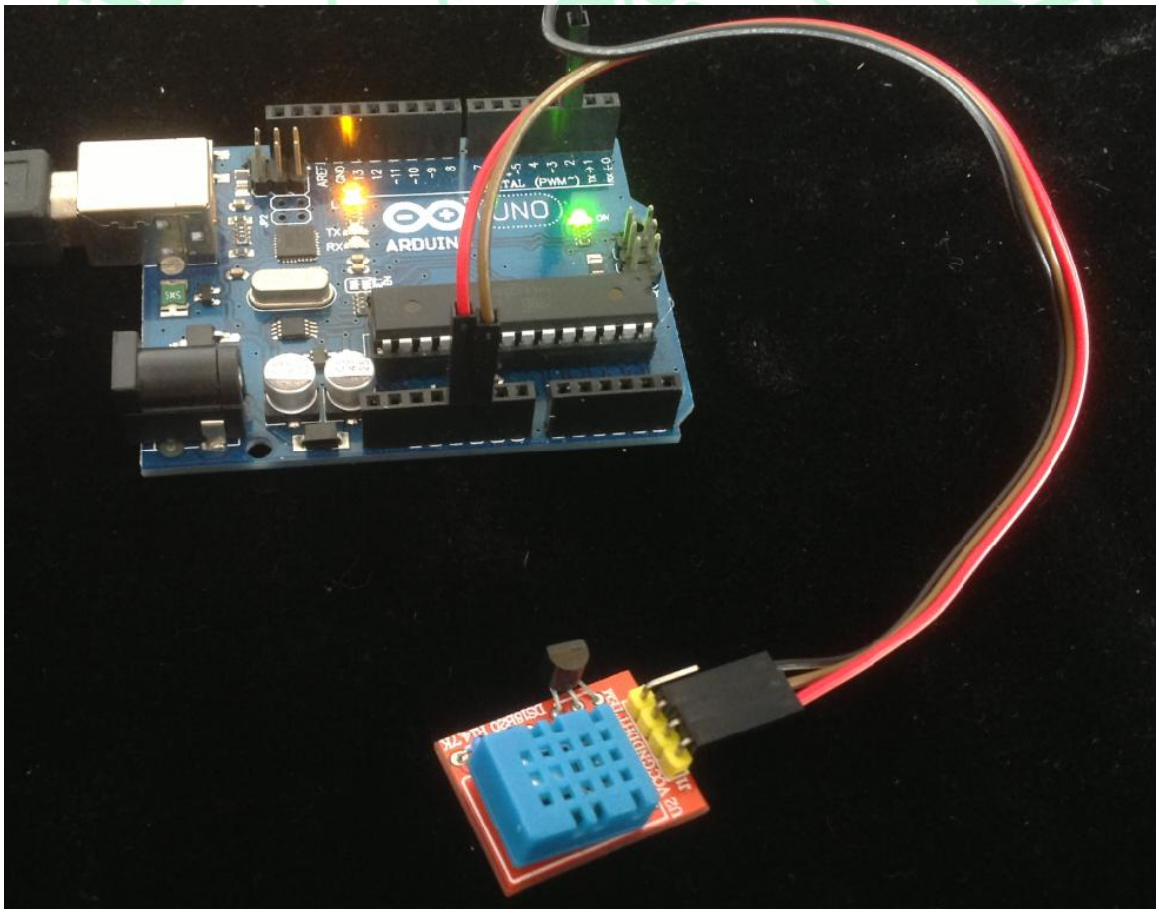
实验现象：读取 DHT11 温湿度传感器的值,然后发送给 PC 通过串口显示出来。

理论学习：

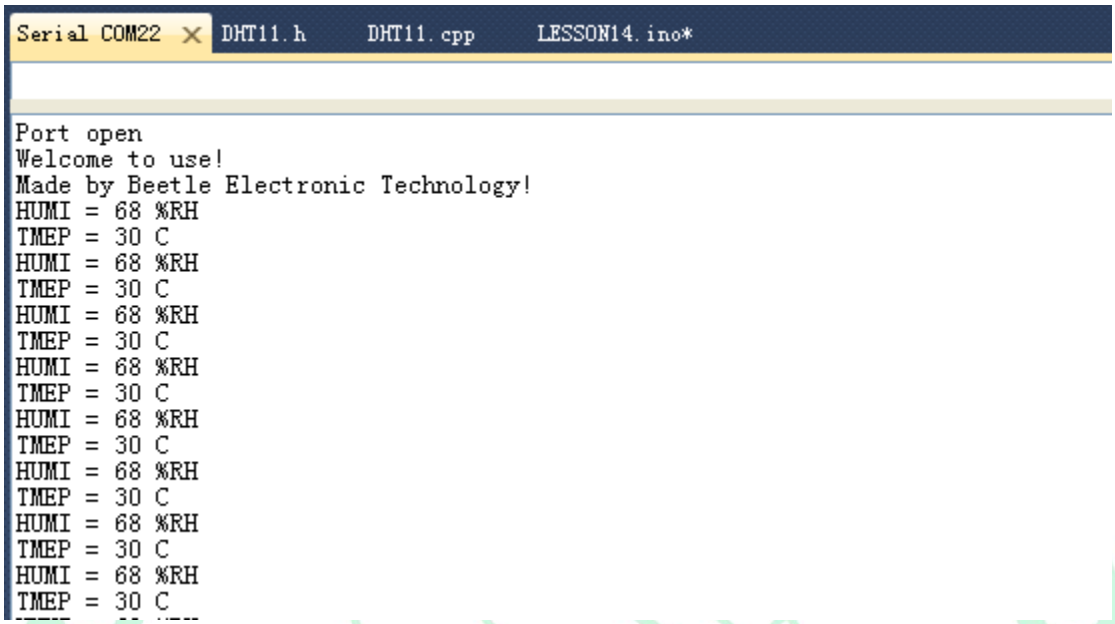
- ✓ DHT11 是一款单总线的温湿度传感器。
  供电电压： 3.3~5.5V DC
  输 出： 单总线数字信号
  测量范围： 湿度 20-90%RH， 温度 0~50℃
  测量精度： 湿度+-5%RH， 温度+-2℃
  分 辨 率： 湿度 1%RH， 温度 1℃
- ✓ DHT11 传感器的初始化时序和读取 0/1 的时序要求请参考 DHT11 说明书（Arduino 独家整理资料包\7.芯片资料\DHT11 说明书）
- ✓ 本程序通过调用头文件DHT11.h内的DHT11_Read();来读取当前的温湿度值，湿度值存放在HUMI_Buffer_Int中，温度值存放在TEM_Buffer_Int中，然后通过通过串口发送出来。



3.14.1 原理图和连接图



3.14.2 实物效果图



3.14.3 串口显示界面

3.15 实时时钟 DS1302 实验

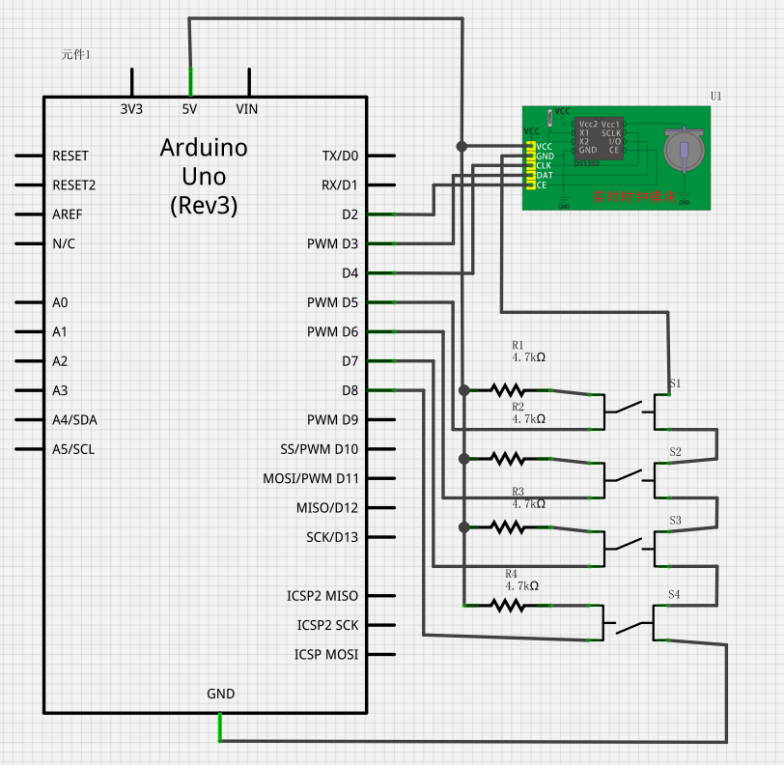
实验现象：

- ✓ 通过 DS1302 的读取获取年/月/日/星期/时/分/秒的值，然后串口显示在 PC 上。
- ✓ 可以通过 4 个按键设定当前的时间。
- ✓ 按键 1（设置）：按一下进入设置状态。再按一下退出设置状态。
- ✓ 按键 2（切换）：按一下切换一种设置变量，分别可以设置年/月/日/星期/时/分/秒
- ✓ 按键 3（增一）：可以对当前的设置变量进行加一操作。
- ✓ 按键 4（减一）：可以对当前的设置变量进行减一操作。

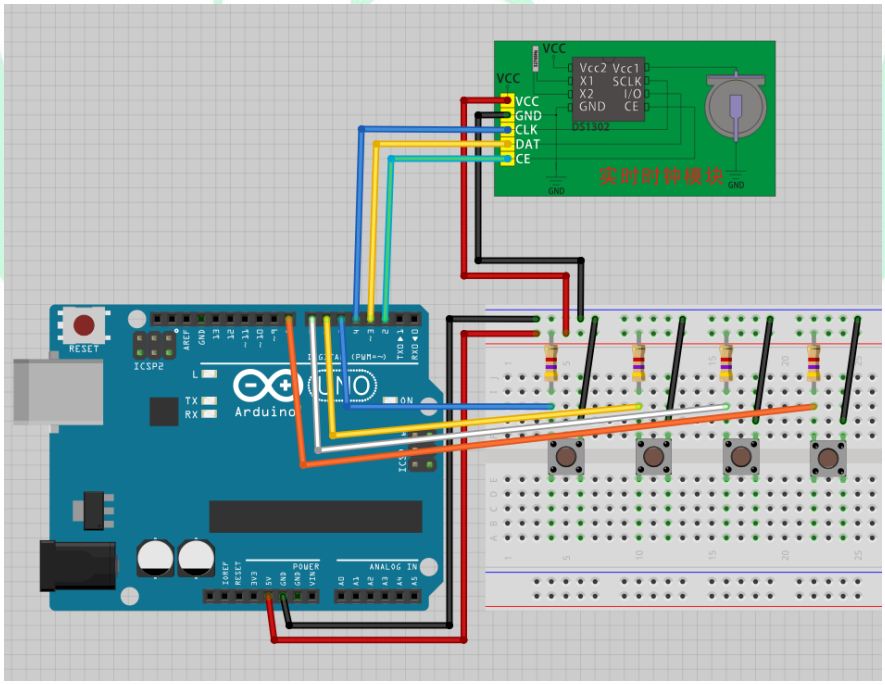
理论学习：

- ✓ DS1302 是美国 DALLAS 公司推出的一种高性能、低功耗的实时时钟芯片，附加 31 字节静态 RAM，采用 SPI 三线接口与 CPU 进行同步通信，并可采用突发方式一次传送多个字节的时钟信号和 RAM 数据。实时时钟可提供秒、分、时、日、星期、月和年，一个月小与 31 天时可以自动调整，且具有闰年补偿功能。工作电压宽达 2.5～5.5V。采用双电源供电（主电源和备用电源），可设置备用电源充电方式，提供了对后背电源进行涓细电流充电的能力。细节参数请查看 DS1302 中文资料(Arduino 独家整理资料包\7.芯片资料\DS1302 中文资料)。
- ✓ 程序通过调用头文件DS1302.h里面的DS1302_GetTime(&DS1302Buffer)函数获取当前的实时时钟，存放在结构体变量DS1302Buffer中，然后通过Display_RTCC()函数把实时时钟通过串口发送出来。

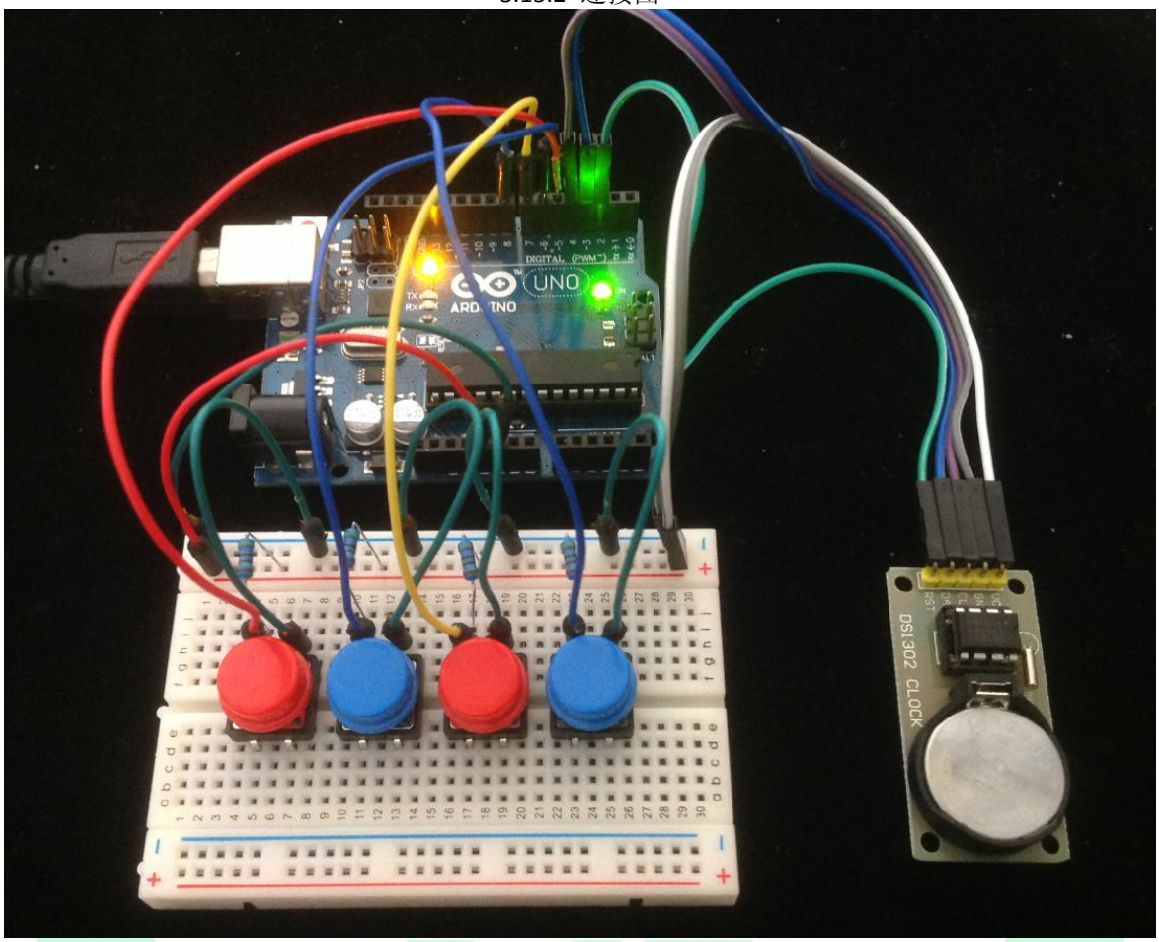
- ✓ Scan_KEY();语句用于扫描4个按键的按键动作。
- ✓ switch(KEY_NUM)语句用于执行4个按键的功能。



3.15.1 原理图



3.15.2 连接图



3.15.3 实物效果图

3.15.4 串口显示界面

3.16 烟雾传感器 MQ-2 实验

实验现象：烟雾传感器把烟雾浓度值转换为模拟电压值输出，Arduino 通过 AO 引脚读取模拟电压值，通过串口发送给 PC,当模拟电压值超过模块的比较器设定值的时候，模块 DO 引脚输出低电平，Arduino 监控到 DO 引脚拉低信号时候，板载 LED 点亮报警,串口输出 Alarm 信号。报警的阀值可以通过调节板载的电位器调整大小。

理论学习：

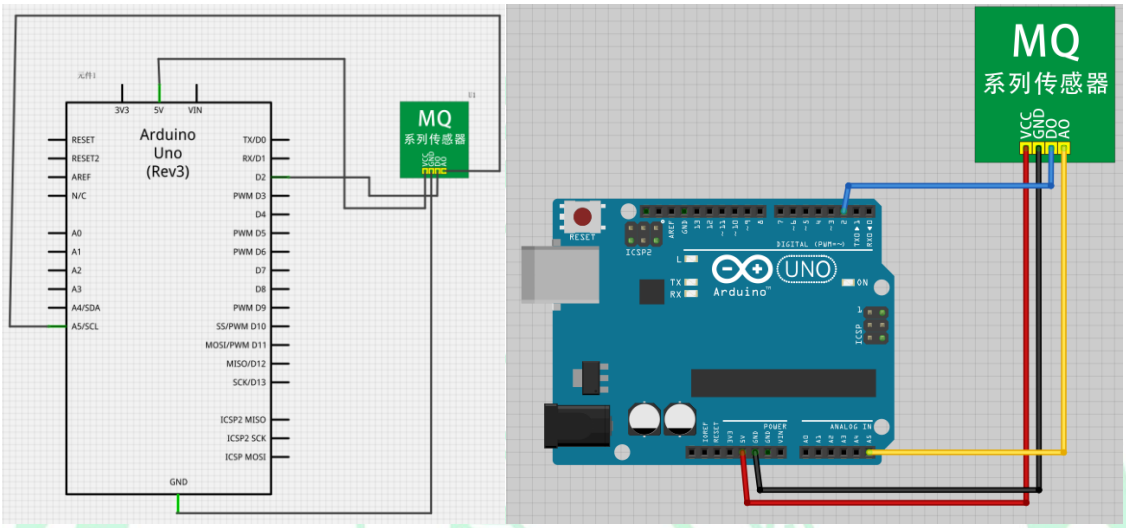
- ✓ MQ 系列传感器常见型号：
  MQ-2 烟雾传感器
  MQ-3 酒精传感器
  MQ-4 甲烷、天然气传感器
  MQ-5 煤气传感器
  MQ-6 液化气体传感器
  MQ-7 一氧化碳传感器
  MQ-8 氢气传感器
  MQ-9 一氧化碳传感器
- ✓ 模块使用说明:

1.VCC/GND 之间加上 5v 电压;

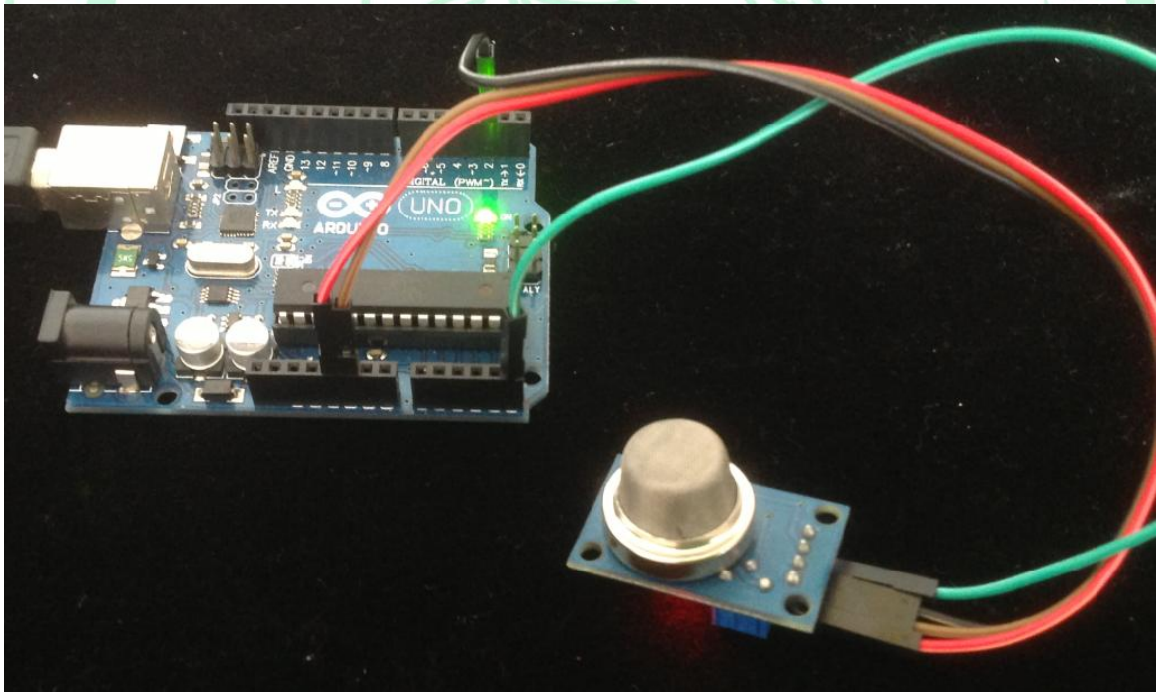
2.等待大概 30s 传感器预热后，读取模拟输出 AO 引脚电压;

3.通过调节电位器,改变 LED 报警阀值.顺时针变大,逆时针减小；

4.当浓度值大于设定值时候，模块上 LED 指示灯点亮,同时 D0 引脚输出低电平。



3.16.1 原理图和连接图



3.16.2 实物效果图

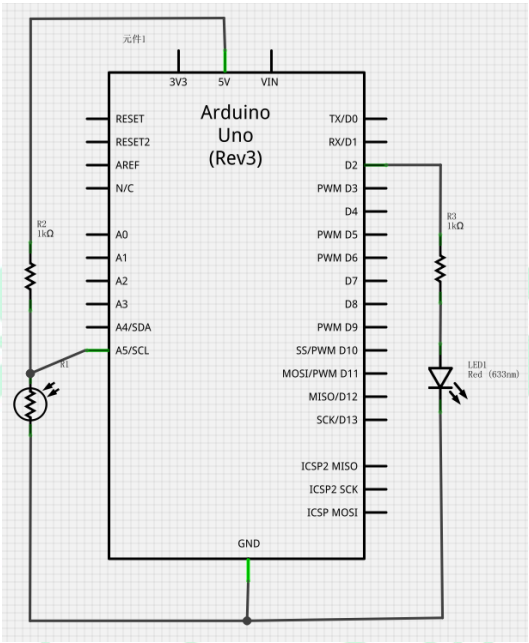
3.16.3 串口显示界面

3.17 光控 LED 实验

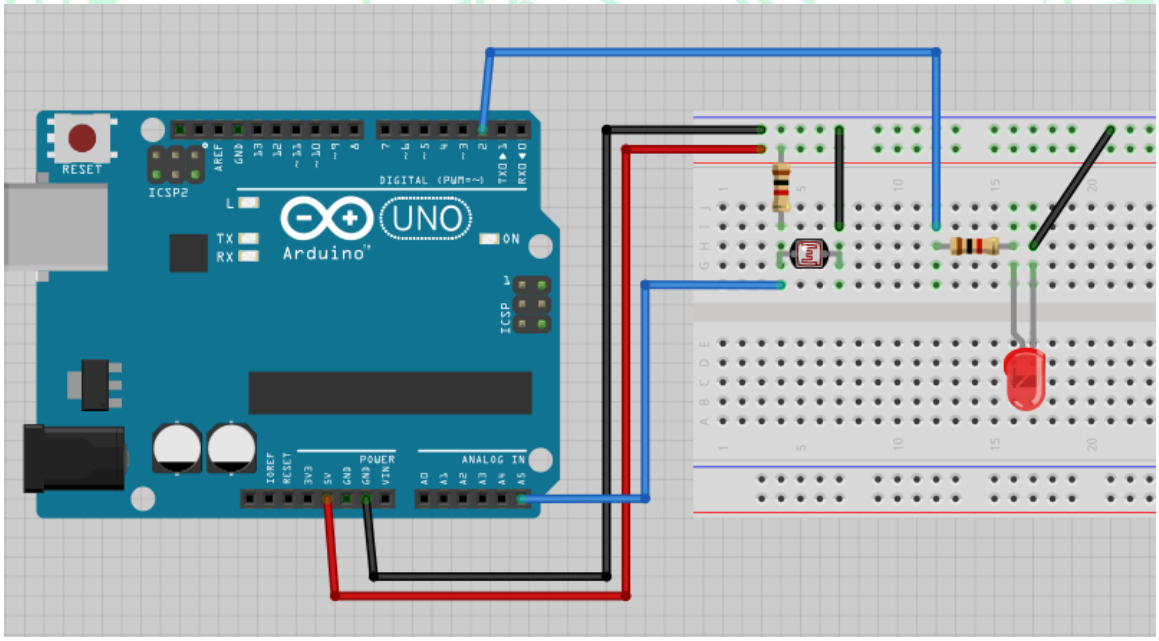
实验现象：当室内光线充足时候 LED 关闭，当室内光线变暗时候点亮 LED。

理论学习：

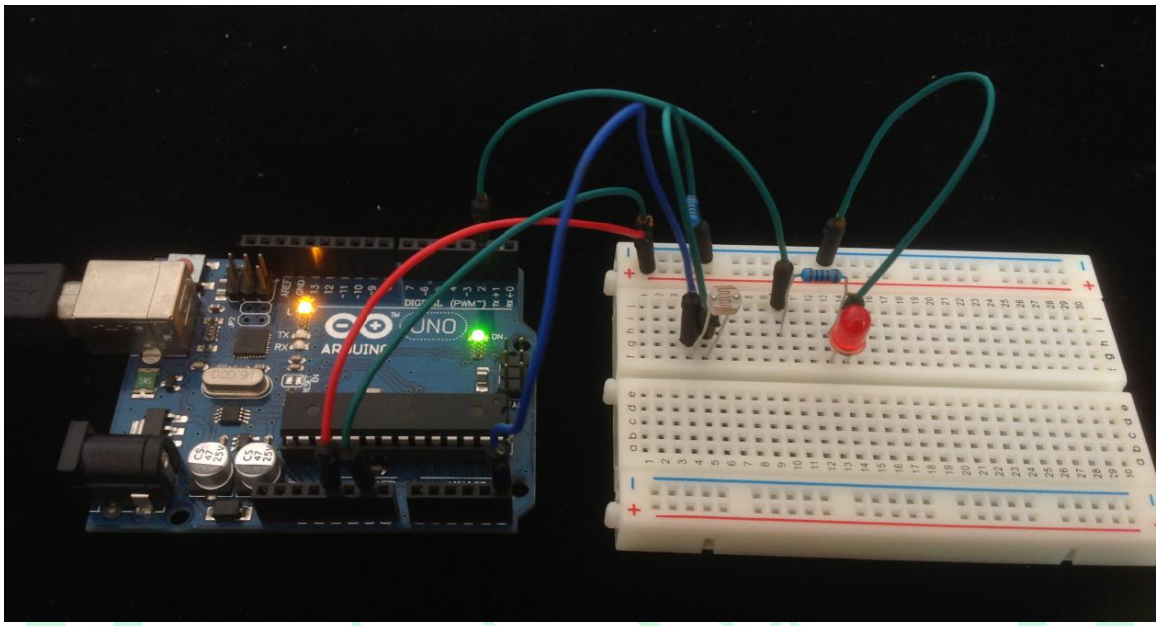
- ✓ 光敏电阻的特性是光敏电阻的阻值随着光照强度的变大而减小。换句话来说，光线越亮电阻越小，光线越暗电阻越大。
- ✓ 本实验选用 1K 电阻和光敏电阻串联，根据电阻分压原理，光线越暗，光敏电阻两端的电压越大。
- ✓ 本实验通过 A5 引脚检测光敏电阻两端的电压值来检测光强。



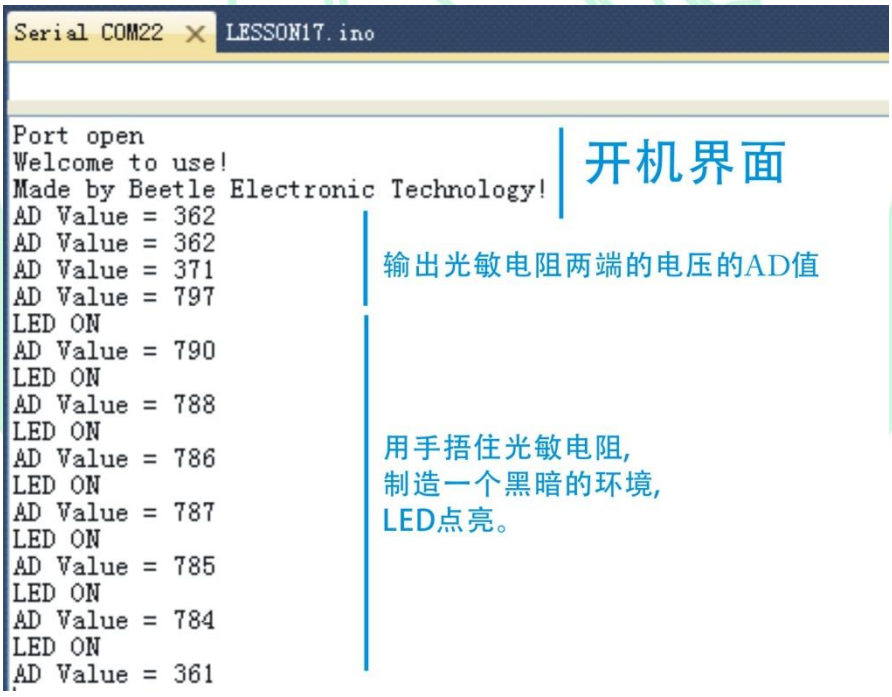
3.17.1 原理图



3.17.2 连接图



3.17.3 实际效果图



3.17.4 串口显示界面

3.18 9 克舵机

实验现象：通过调整电位器来改变舵机旋转的角度。

（因为电脑 USB 供电电流不充足，建议下载完程序拔下 USB 线，使用电源适配器供电来测试该程序。）

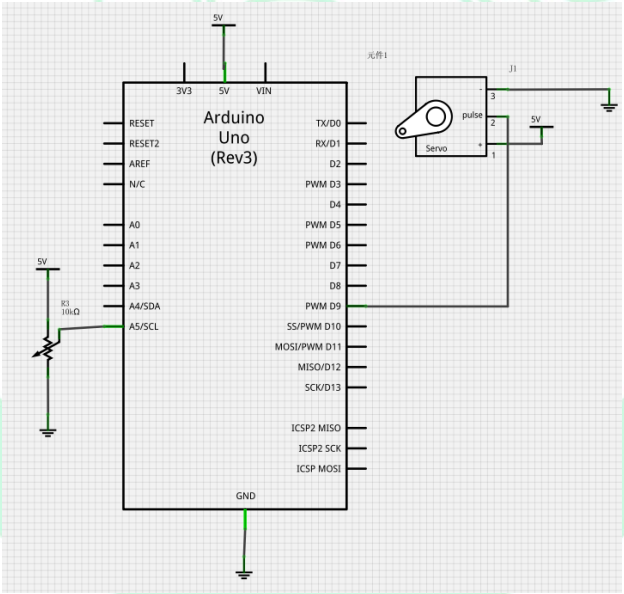
理论学习：

- ✓ 标配 9g 舵机引脚，黄色为信号线，红色为电源，棕色为地线。
- ✓ 舵机是一种位置伺服的驱动器，具有闭环控制系统的机电结构，由小型直流电机、变速齿轮组、可调电位器、控制板等部件组成由于可以方便地控制舵机旋转的角度（舵角，但是舵角一般不超过 180°），因此，舵机在要求角度不断变化的控制系统中得到了广泛应用。
- ✓ 舵机在工作中，控制器发出脉冲宽度调制（PWM）信号给舵机，获得直流偏置电压。舵机内部有一个基准电路，产生周期为 20ms，宽度为 1.5ms 的基准信号，将获得的直流偏置电压与电位器的电压比较，获得电压差输出到电机驱动芯片，驱动芯片根据电压差的正负控制电机的正反转。
- ✓ 舵机转动的角度是通过调节 PWM 信号的占空比来实现的，标准 PWM 信号的周期固定为 20ms，理论上脉宽（脉冲的高电平部分）范围在 1ms~2ms 之间，但实际上脉宽可以在 0.5ms~2.5ms 之间，脉宽和舵机的转角 0°~180°相对应。如以脉宽为 0.5ms~2.5ms 范围控制舵机的角度转动，转动范围为 0°~180°。
- ✓ 小型舵机的工作电压一般为 4.8V 或 6V，转速也不是很快，所以假如更改角度控制脉宽太快时，舵机可能反应不过来。如果需要更快速的反应，就需要更高的转速了。要精确的控制舵机，其实没有那么容易，很多舵机的位置等级有 1024 个，那么，如果舵机的有效角度范围为 180°的话，其控制的角度精度是可以达到 180°/1024≈0.18°，如果假定脉宽为 0.5ms~2.5ms 范围，则要求的脉宽控制精度为(2.5-0.5)ms/1024≈2μs。
- ✓ 舵机分别用 0.5ms~2.5ms 之间的脉宽来对应 0 到 180°左右的角度，且转动的角度与脉宽呈线性关系，则舵机每转动 1°，对应的脉宽为（2.5-0.5）ms/180°，该值除不尽，因此，用一个除不尽的脉冲宽度控制舵机转动，显然转动角度的精度很难控制，为此，实验中以接近 2.5ms 且能够整除 180 的值最为脉宽的变化范围，则取脉宽的范围为 0.5ms~2.48ms，此时，舵机每转动 1°，则脉宽变化（2.48-0.5）ms/180=11μs。因此，定义脉宽与转动角度之间的关系为：
  pulsewidth=(angle*11)+500
- ✓ 根据这个公式可以写出来设置转动角的函数，然后写出第一个例程。

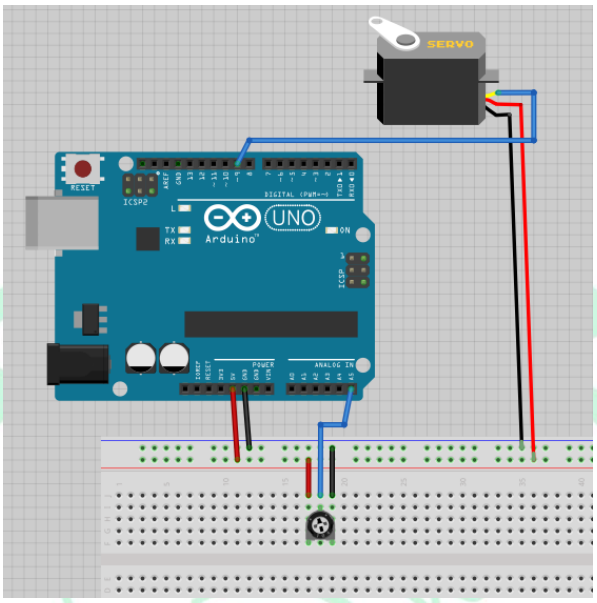
```
void pulse(int angle)
{
    pulsewidth=int ((angle*11)+500);
    digitalWrite(PWM_pin,HIGH);
    delayMicroseconds(pulsewidth);
    digitalWrite(PWM_pin,LOW);
    delay(20-pulsewidth/1000);
}
```

- ✓ 当然还有更简单的办法：调用伺服电机控制 lib，

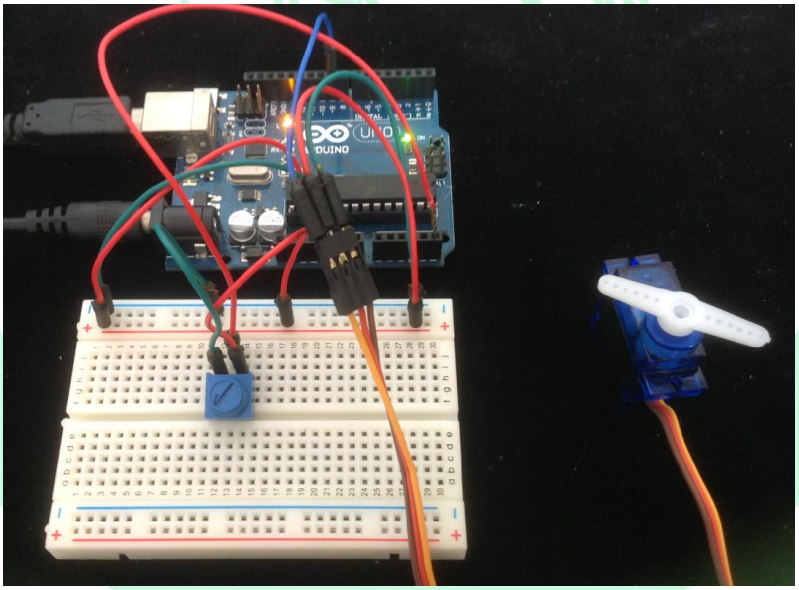
```
#include <Servo.h> //引入 lib
Servo myservo;   // 创建一个伺服电机对象
#define potpin   A5   // 设定连接可变电阻的模拟引脚
int val;      // 创建变量，储存从模拟端口读取的值（0 到 1023）
void setup()
{
  myservo.attach(9);   // 9 号引脚输出电机控制信号
                       //仅能使用 9、10 号引脚
}
void loop()
{
  val = analogRead(potpin);
 // 读取来自可变电阻的模拟值（0 到 1023 之间）
  val = map(val, 0, 1023, 0, 179);      // 利用“map”函数缩放该值，得到伺服电机需
要的角度（0 到 180 之间）
  myservo.write(val);      // 设定伺服电机的位置
  delay(15);               // 等待电机旋转到目标角度
}
```



3.18.1 原理图



3.18.2 连接图



3.18.3 实物效果图

3.19 红外遥控实验

实验现象：当按下遥控器上某个按键，串口输出该按键的名称。

理论学习：

- ✓ 本实验使用红外接收头 VS1838+红外遥控器。VS1838 使用 NEC 码编码格式，
- ✓ NEC 码格式：

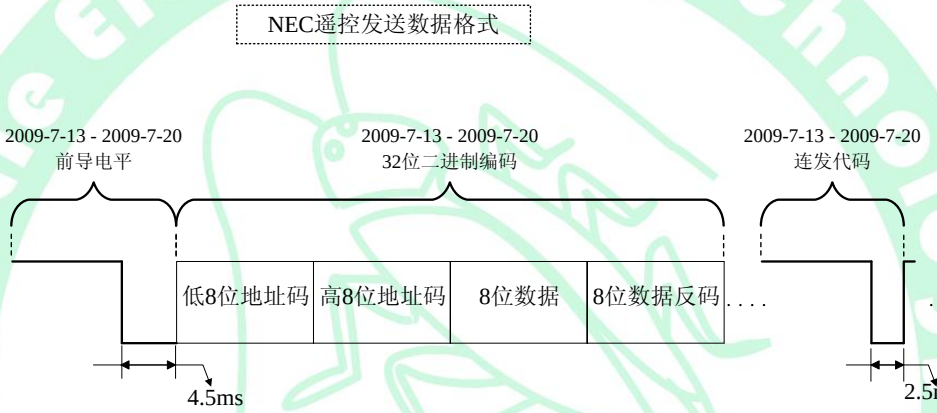
1.使用 38 kHz 载波频率

2.引导码间隔是 9 ms + 4.5 ms

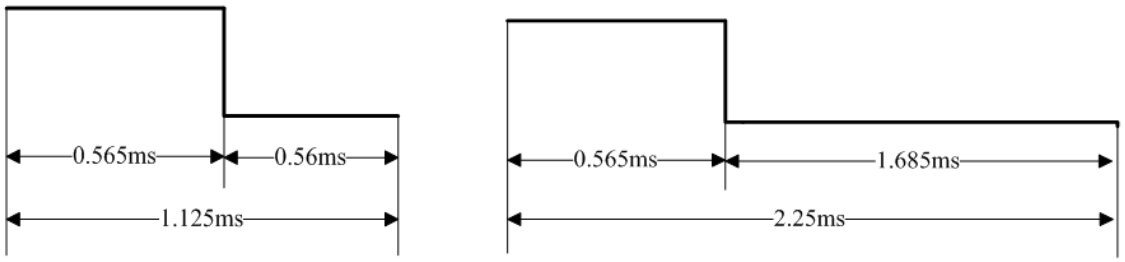
3.使用 16 位客户代码

4.使用 8 位数据代码和 8 位取反的数据代码

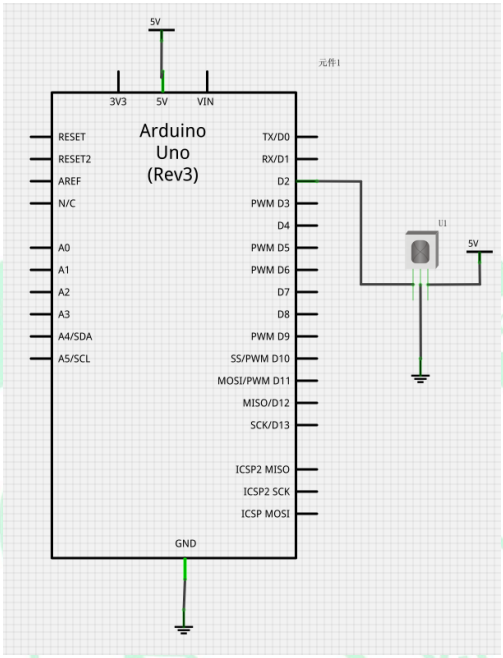
下面的波形是从红外接收头上得到的波形：（调制信号转变成高低电平了）



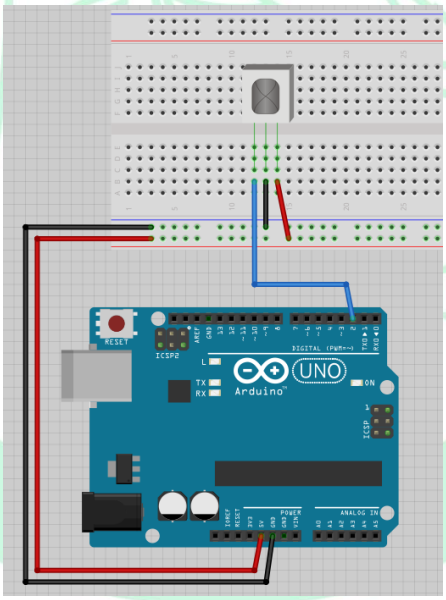
5 二进制 0 和 1 的表示方法：



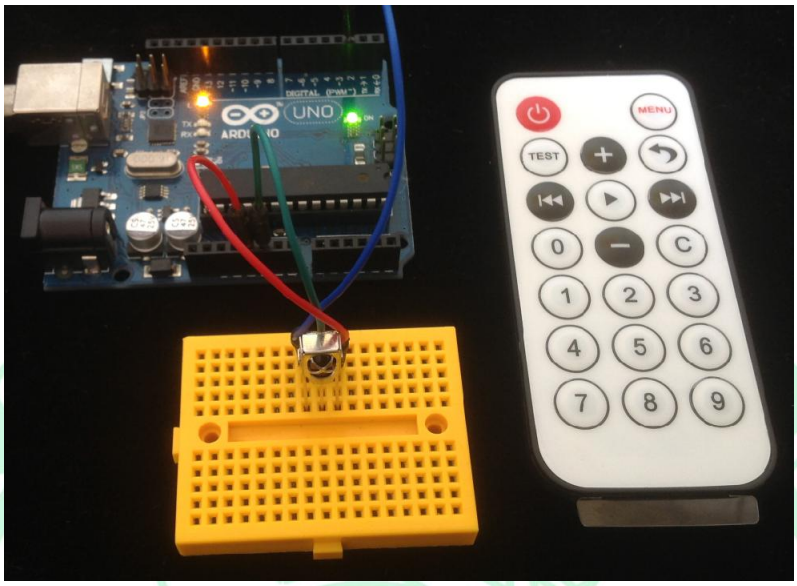
遥控发射码“0” 遥控发射码“1”



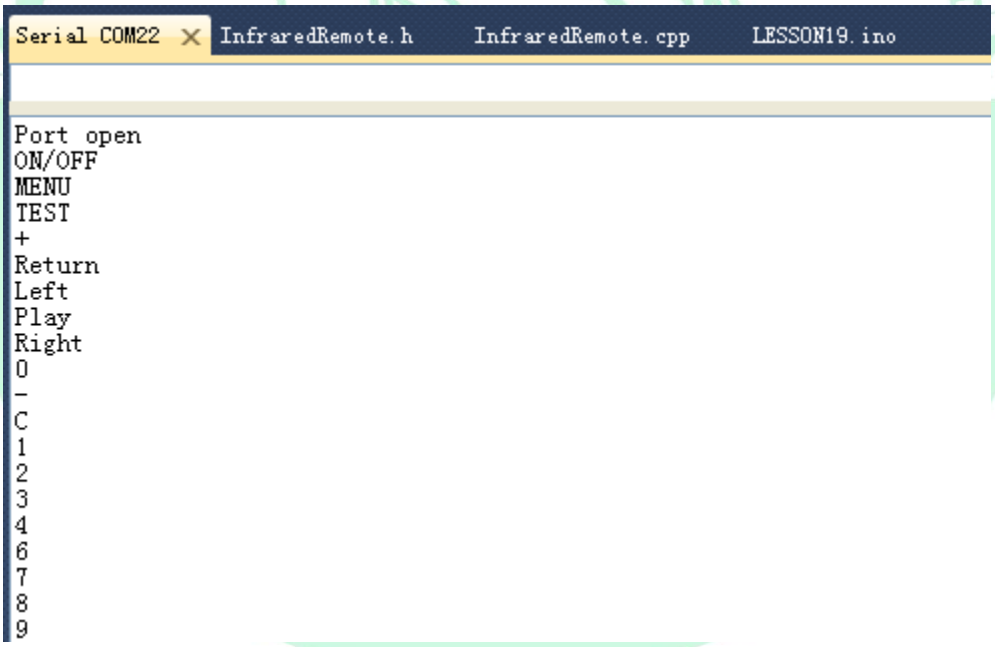
3.19.1 原理图



3.19.2 连接图



3.19.3 实物效果图



3.19.4 串口显示界面

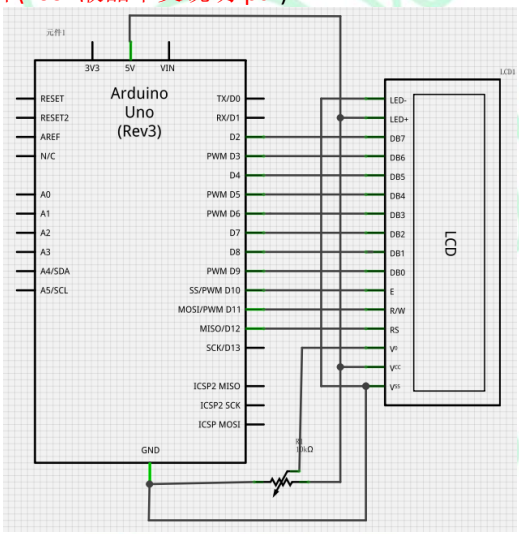
3.20 液晶 LCD1602 实验

实验现象：

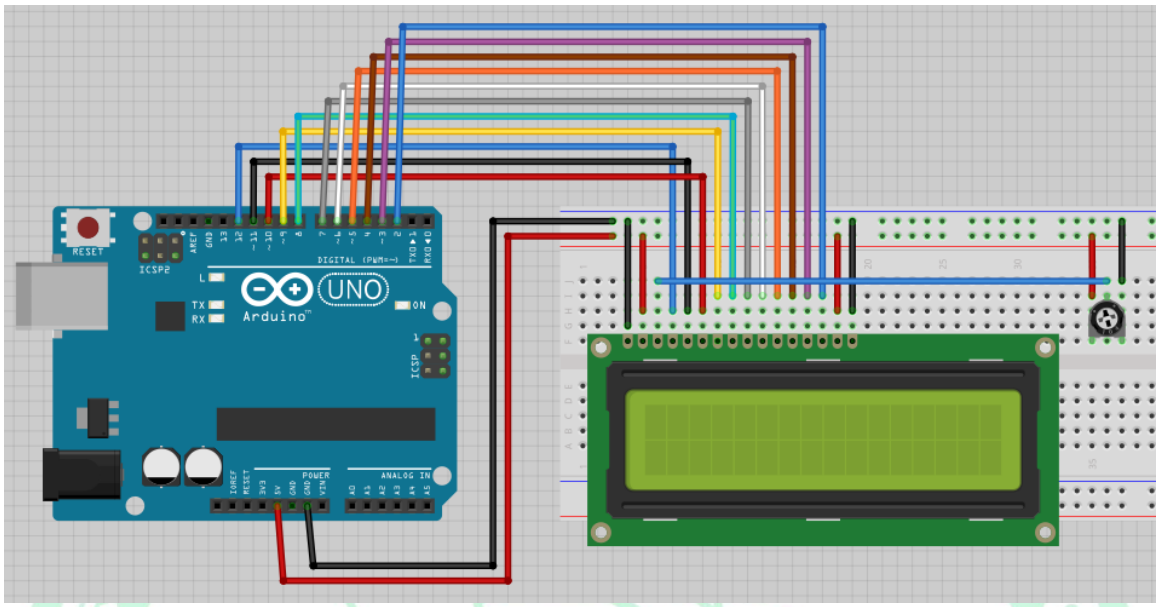
- ✓ 液晶第一行显示 Welcome to use !
- ✓ 液晶第二行前 10 位通过数字加引号方式显示 0-9 数字
- ✓ 液晶第二行后 6 位通过直接写 ASCII 码方式写数字 0-5
- ✓ 液晶的对比度可以通过调节对比度电位器改变。

理论学习：

- ✓ LCD1602 液晶是一款通用简易液晶，可以显示 ASCII 码的英文字母、数字和标点符号（不能显示中文）。
- ✓ LCD1602 总共可以显示 2 行*16 个字符。本程序调用库函数 LCD1602.h 的 Init_LCD1602();对程序进行初始化，然后设置液晶指针位置 LCD1602_write_com(0x80);其中 0x80 地址对应液晶的第一行第一个列，之后每列地址加一，第一行最后一列的地址位 0x8f。
  第二行的地址从 0x80+0x40 开始，最后一列的地址位 0x80+0x4f。
- ✓ 用户可以通过LCD1602_write_data函数写单个字符，或者通过LCD1602_write_word写字符串。
- ✓ LCD1602液晶的详细指标和时序图请参考 1602液晶中文说明.pdf(Arduino独家整理资料包\7.芯片及模块资料\1602液晶中文说明.pdf)



3.20.1 原理图



3.20.2 连接图

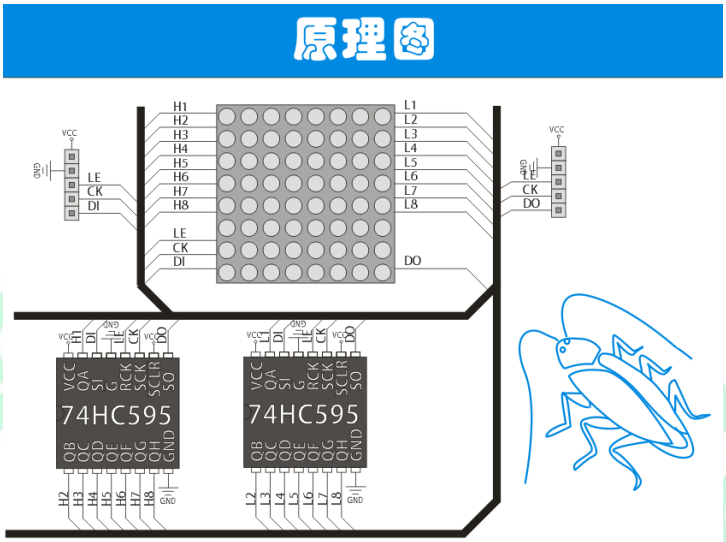
3.20.3 实物效果图

3.21 8*8 点阵模块静止显示

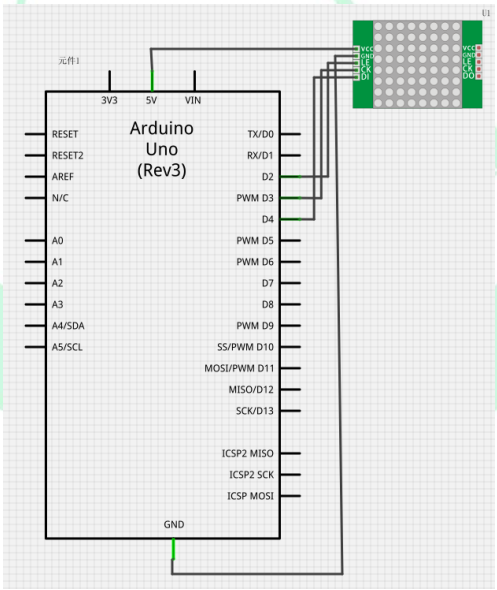
实验现象：点阵显示一颗心，过一段时间显示一个箭头。

理论学习：

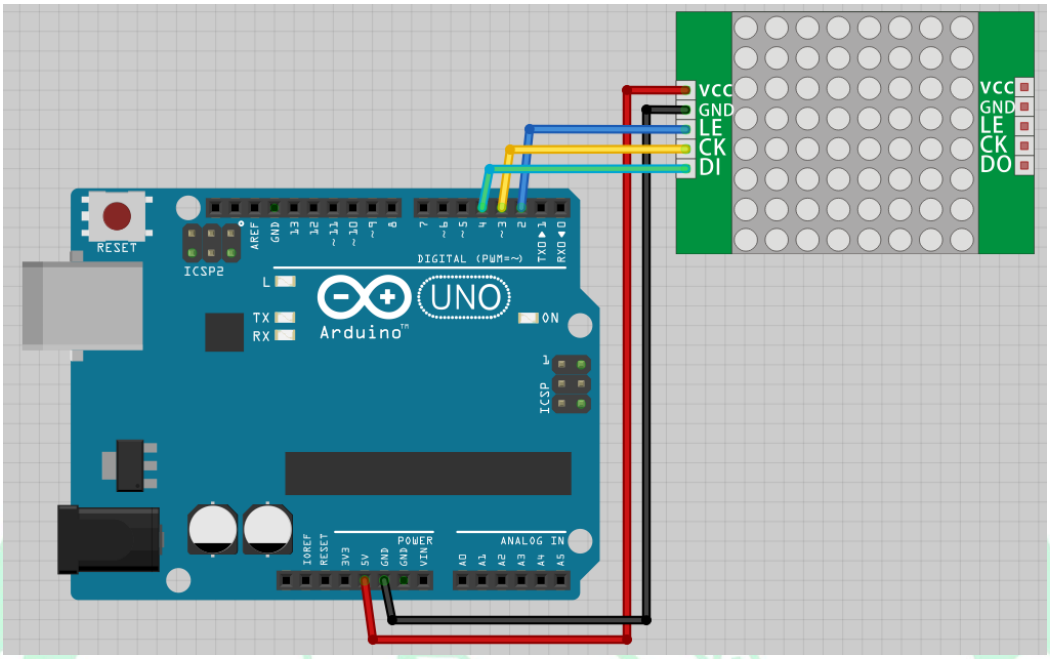
- ✓ 点阵模块原理图：



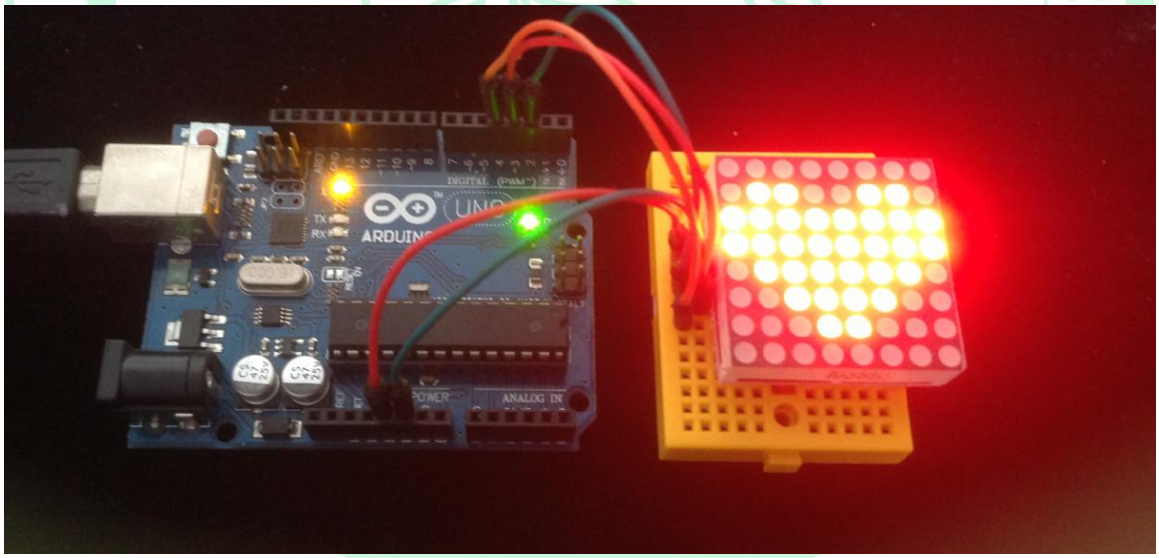
- ✓ 该模块点阵的行和列分别有一个 74HC595 控制，两个 74HC595 芯片串联起来，只用三个 IO 就可以控制 8*8 点阵。
- ✓ 其中 DI 引脚用于给 74HC595 发送数据。
- ✓ CK 引脚用于给 74HC595 提供时钟。
- ✓ LE 引脚用于锁存输出 74HC595 的数据。
- ✓ DO 引脚用于级联下一级点阵的 DI 输入。
- ✓ 显示原理：当 74HC595 选通点阵的某一列（即给选通列低电平）时候，给控制行的 74HC595 发送一字节的数据（高电平点亮）。



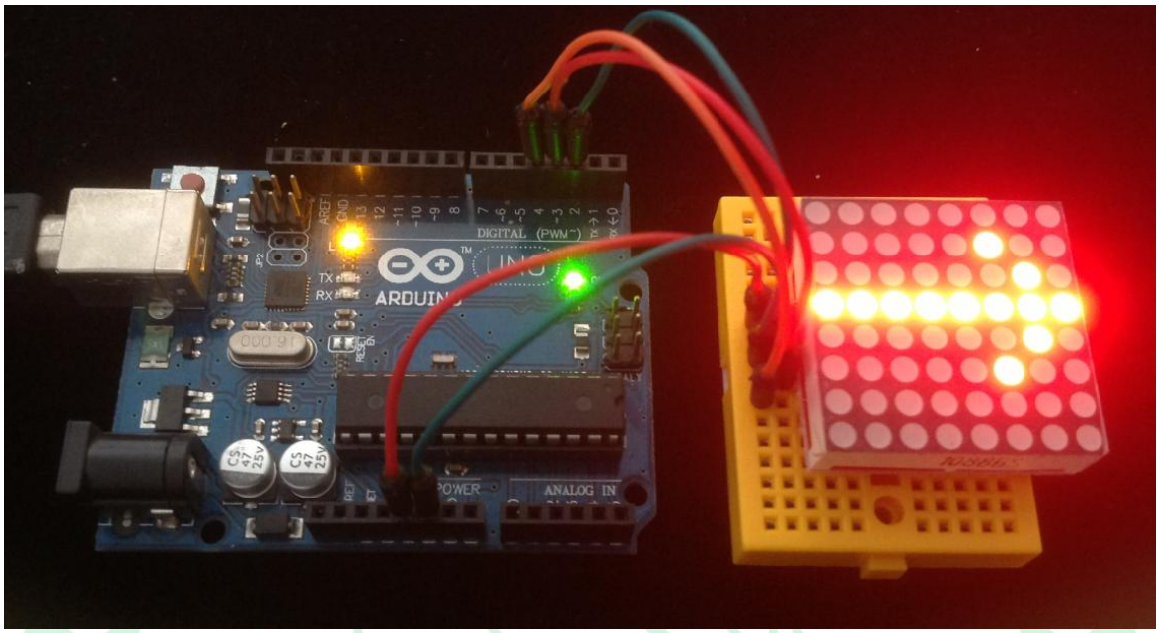
3.21.1 原理图



3.21.2 连接图



3.21.3 实物效果图 1



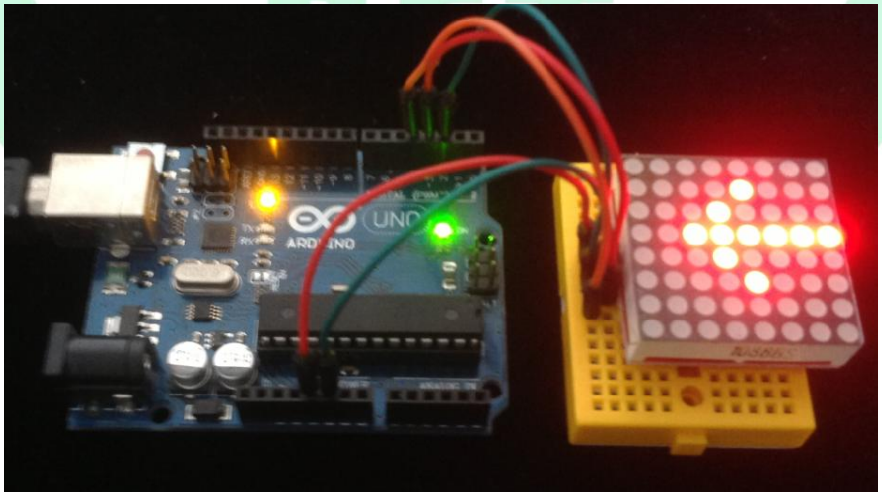
3.21.4 实物效果图 2

3.22 8*8 点阵模块左移显示

实验现象：一个箭头图标从右向左移动。

理论学习：利用每个瞬间播放不同的图片的原理，让人眼觉得图片在从右向左移动。

原理图和连接图：同例 21



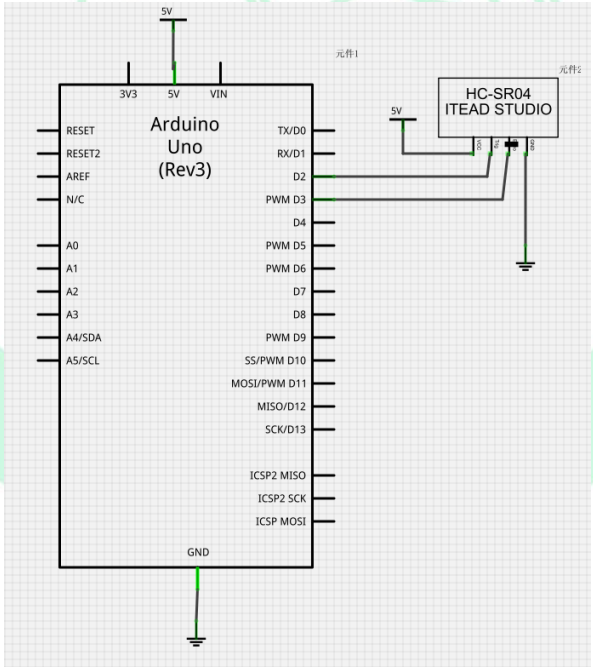
3.22.1 实物效果图

3.23 超声波测距

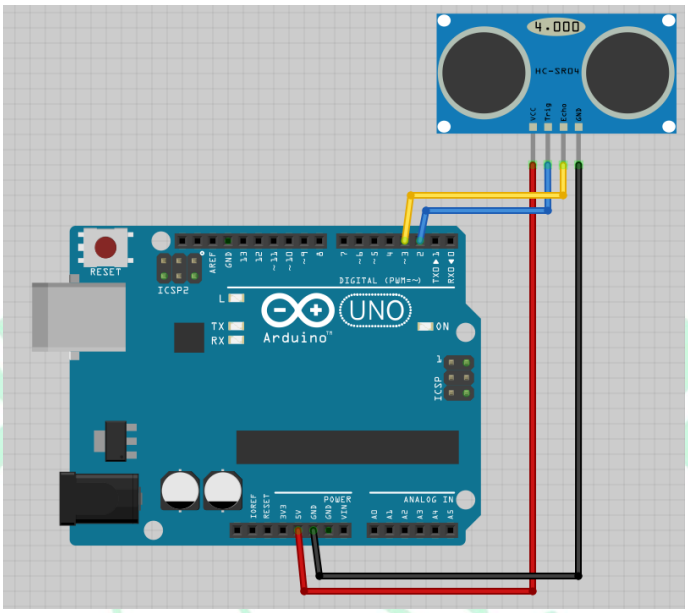
实验现象：串口输出超声波模块和格挡物体之间的距离，单位为 cm

理论学习：

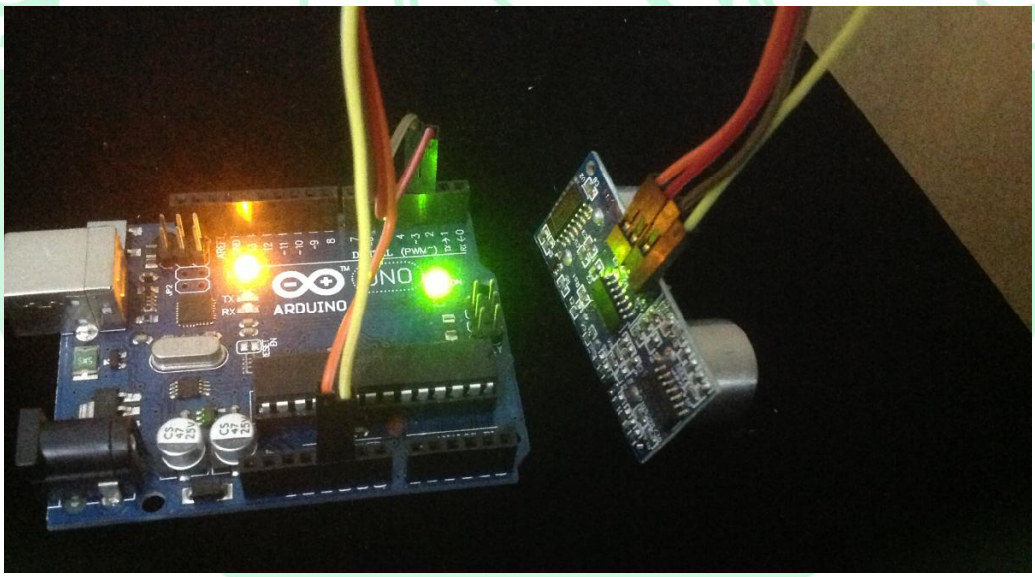
- ✓ 超声波测距原理：
  (1)采用 trig 触发测距，给至少 10us 的高电平信号;
  (2)模块自动发送 8 个 40khz 的方波，自动检测是否有信号返回；
  (3)有信号返回，通过 echo 输出高电平，高电平持续的时间就是距离的 2 倍；
  (4) 超声波从发射到返回的时间．测试距离=(高电平时间*声速(340M/S))/2
- ✓ 测距程序核心代码 pulseIn(pin, value) 函数：读取一个引脚的脉冲（HIGH 或 LOW）。例如，如果 value 是 HIGH，pulseIn()会等待引脚变为 HIGH，开始计时，再等待引脚变为 LOW 并停止计时。返回脉冲的长度，单位微秒。如果在指定的时间内无脉冲函数返回。此函数的计时功能由经验决定，长时间的脉冲计时可能会出错。计时范围从 10 微秒至 3 分钟。（1 秒=1000 毫秒=1000000 微秒），请注意单位为 us。
  pin:你要进行脉冲计时的引脚号。
  value:要读取的脉冲类型，HIGH 或 LOW。



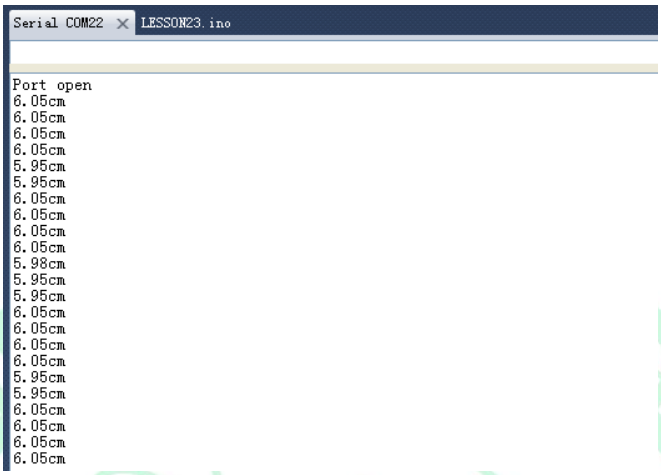
3.23.1 原理图



3.23.2 连接图



3.23.3 实物效果图



3.23.4 串口显示界面

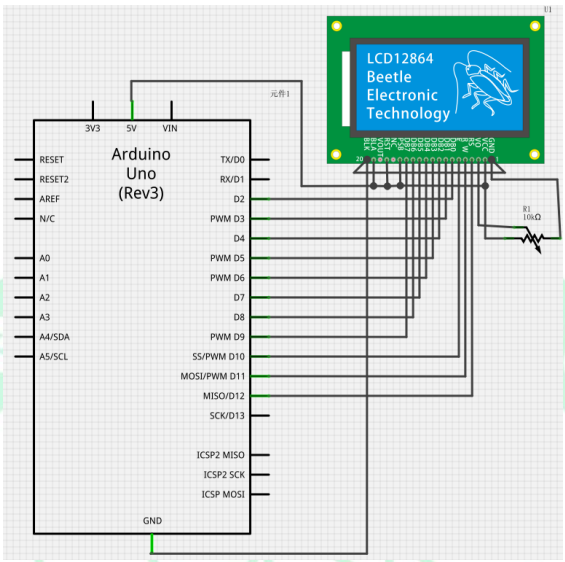
3.24 液晶 LCD12864 并行模式

实验现象：

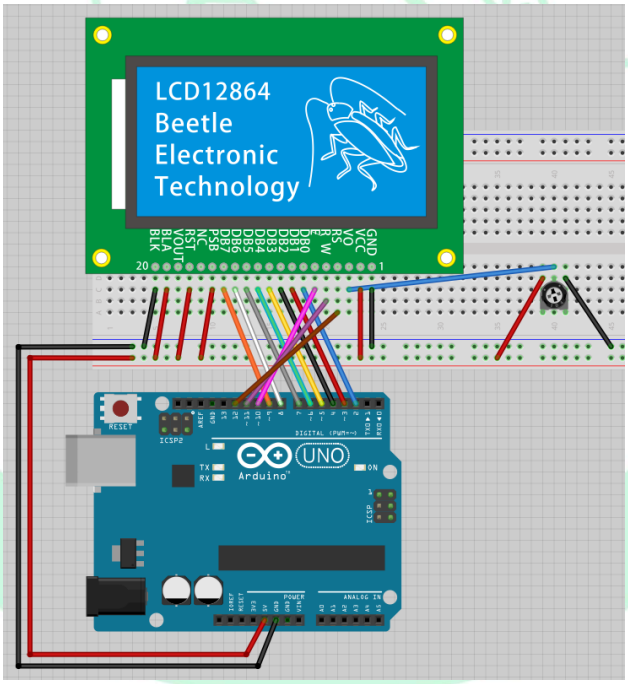
- ✓ 利用图形模式显示出 128*64 象素的图片
- ✓ 利用普通模式显示英文字符数字
- ✓ 利用 LCD12864 内部带的中文字库，显示出中文字

理论学习：

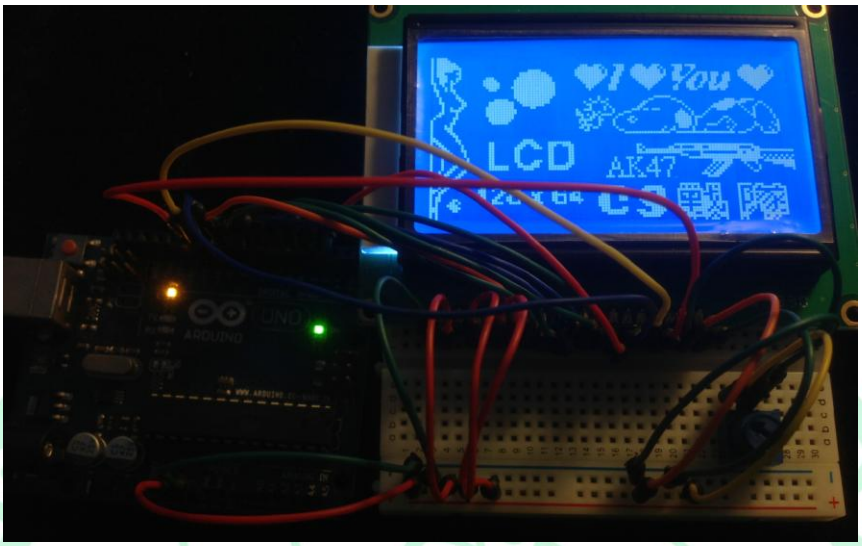
- ✓ 本店的 LCD12864 是带中文字库的液晶显示屏，其内核是使用 ST7920 芯片（Arduino 独家整理资料包\7.芯片及模块资料\中文字库液晶芯片 ST7920.pdf），可以显示英文字母/数字/标点符号/汉字/中文标点的一款液晶显示屏。
- ✓ 本液晶可以显示 4 行数据，每行有 8 个地址，每个地址可以显示一个汉字或者 2 个英文字符。
- ✓ 本液晶分绘图模式和普通模式两个模式，绘图模式中可以直接操作绘图空间显示出 128*64 象素尺寸的图片。普通模式直接调用 ST7920 芯片内字库进行显示。
- ✓ 并行模式用到 13 个 IO，其中 3 个控制 IO，8 个数据 IO。



3.24.1 原理图

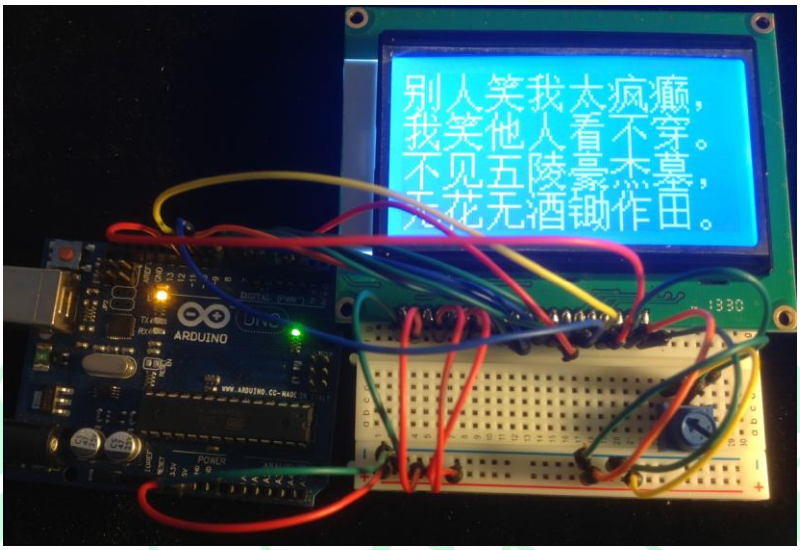


3.24.2 连线图



3.24.3 绘图模式

3.24.4 普通模式写字符



3.24.5 普通模式写汉字

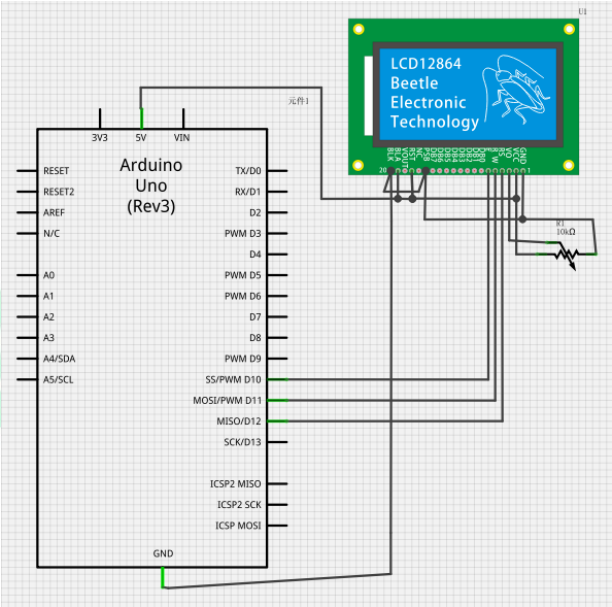
3.25 液晶 LCD12864 串行模式

实验现象：

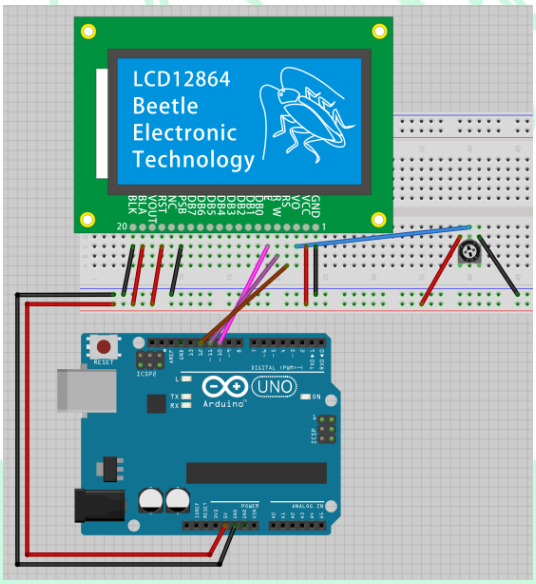
- ✓ 利用图形模式显示出 128*64 象素的图片
- ✓ 利用普通模式显示英文字符数字
- ✓ 利用 LCD12864 内部带的中文字库，显示出中文字
- ✓ 感受串行模式和并行模式的优缺点

理论学习:

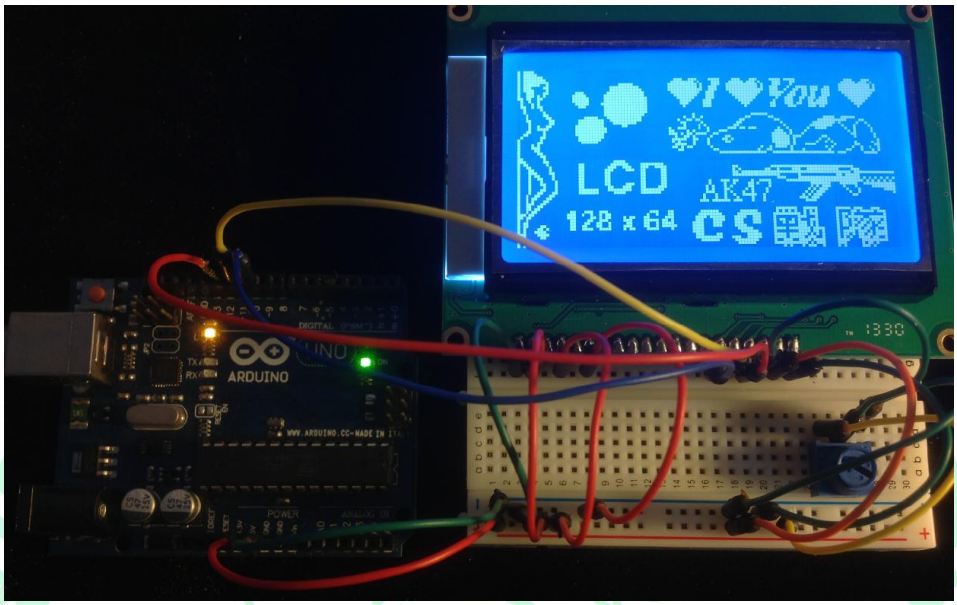
- ✓ 将液晶的 PSB 引脚就低电平会使能串行模式。
- ✓ 串行模式只用 3 个 IO 控制，比并行模式少用了 8 个 IO
- ✓ 串行模式因为比并行的少用 8 个 IO，因此刷屏速率上就有明显差异，比并行要慢的多，绘图模式尤为明显。



3.25.1 原理图

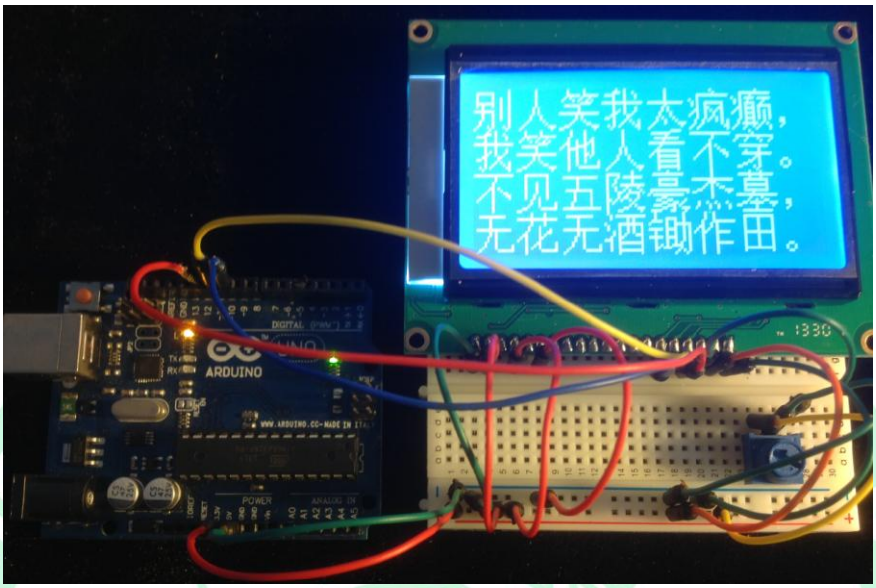


3.25.2 连接图



3.25.3 绘图模式

3.25.4 普通模式显示字符



3.25.5 普通模式显示汉字

3.26 16*16 点阵模块静止显示

实验现象：静止显示一个汉字“强”。

理论学习：

✓ 行选择有 2 个 74HC138 组合成的 4-16 译码器来选择，

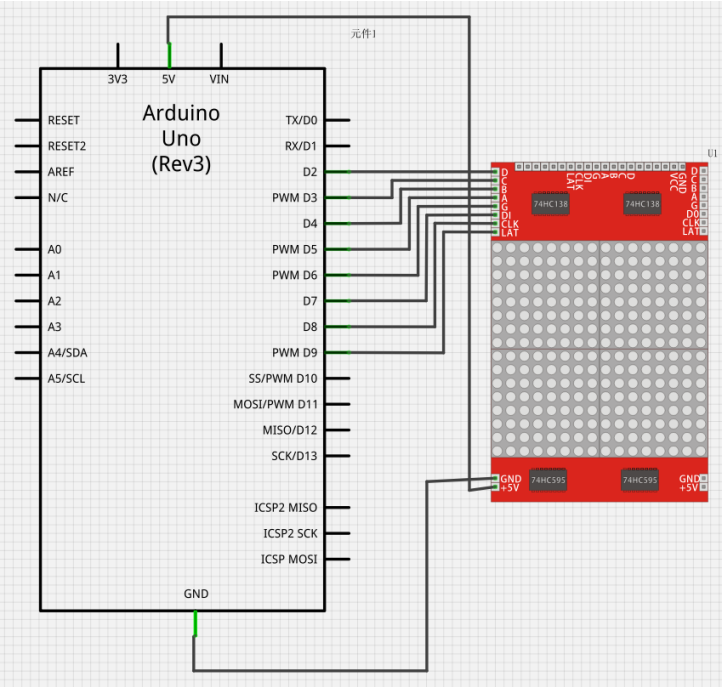
真值表

D C B A	选通行
0 0 0 0	行 0
0 0 0 1	行 1
0 0 1 0	行 2
0 0 1 1	行 3
0 1 0 0	行 4
0 1 0 1	行 5
0 1 1 0	行 6
0 1 1 1	行 7
1 0 0 0	行 8
1 0 0 1	行 9
1 0 1 0	行 10
1 0 1 1	行 11
1 1 0 0	行 12
1 1 0 1	行 13

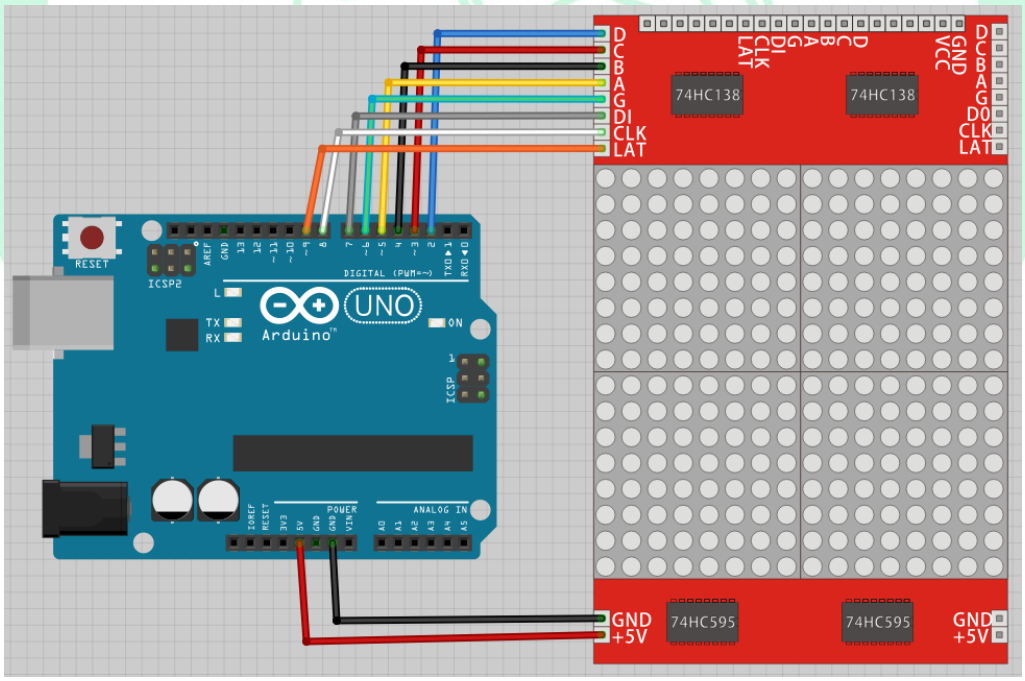
1 1 1 0　行 14
1 1 1 1　行 15

被选通的行，LED 的阳极接通。

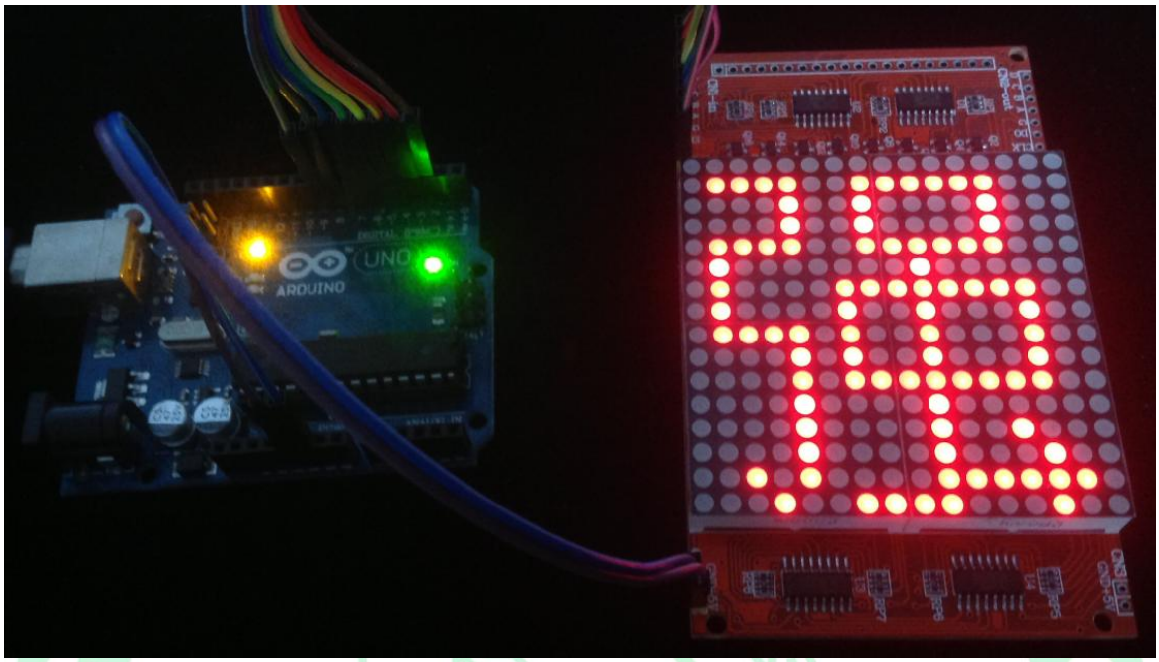
- ✓ 2 列输出由两片 74HC595 级联而成，通过 SPI 信号把串行数据转换为并行数据。当某列输出信号为高电平时候，该列 LED 阴极为高电平，所以选通行与该列交叉点的 LED 不亮。相反，列输出信号为低电平时候，该列的 LED 阴极为低，所以选通行与该列交叉点的 LED 点亮
- ✓ 选通一行后，74HC595 输出该行数据。总共 16 行依次循环，动态扫描。使 16*16 的点阵显示出来需要的文字或者图形。
- ✓ 显示的汉字的字模是通过字模软件取出来的。（Arduino 独家整理资料包\8.小软件\点阵 LED 字模生成工具）
- ✓ 取模软件设置如下：



3.26.1 原理图



3.26.2 连接图



3.26.3 实物效果图

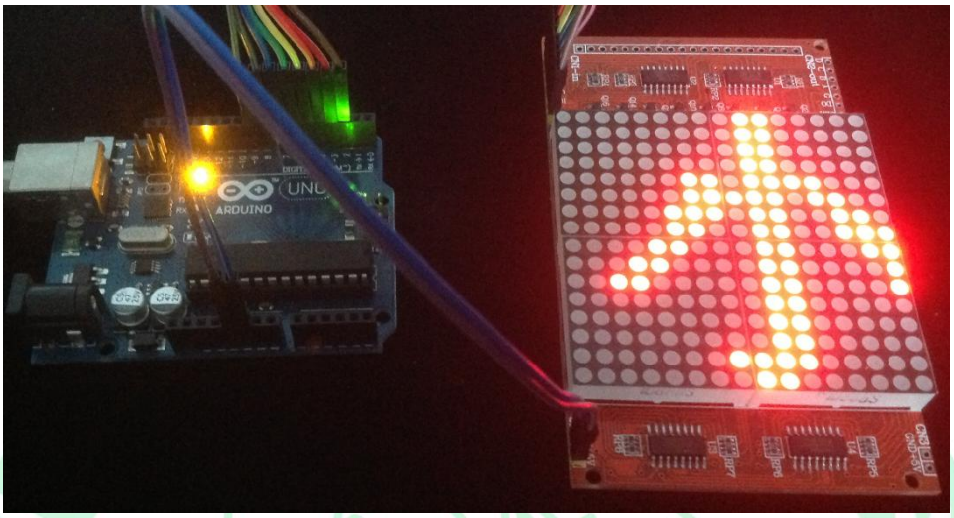
3.27 16*16 点阵模块左移显示

实验现象：“小强电子设计”5 个字左移显示。

理论学习：

- ✓ 当学会静态显示时候，我们可以静下来思考一下，如果想左移显示只要把新的数据不停的存放在显示缓冲区中就可以。最大的问题就是如何把取的字模值经过数据换算转换成需要显示的区域和内容。

原理图和连接图：同例 26



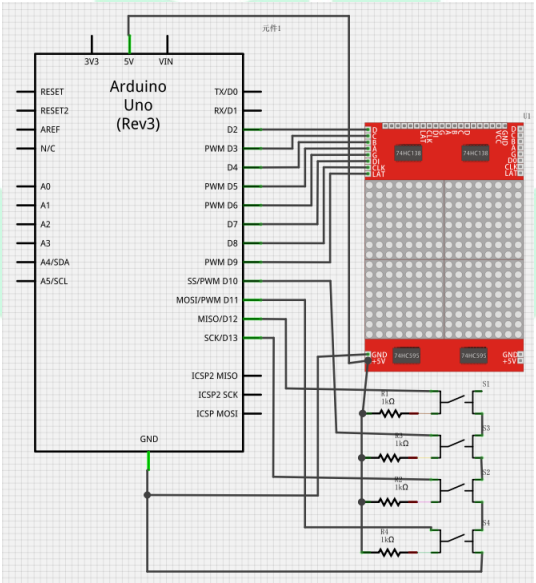
3.28.1 实物效果图

3.28 16*16 点阵模块贪吃蛇游戏

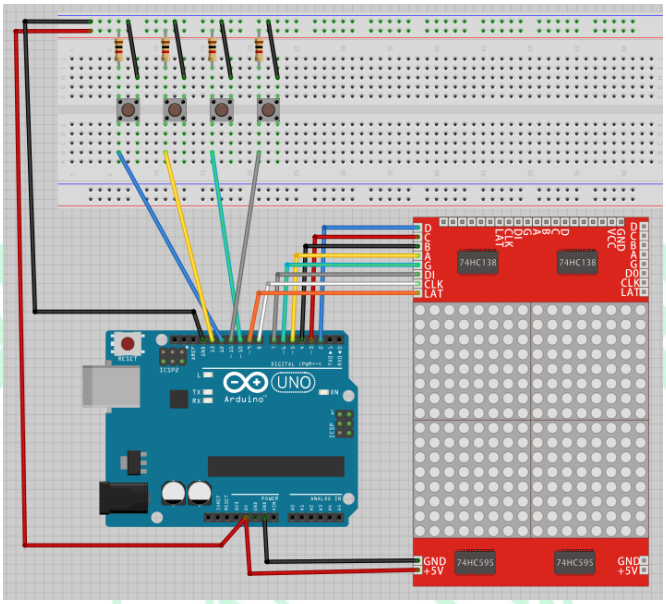
实验现象：可通过 4 个按键用点阵屏玩贪食蛇游戏，其中向右按键可做为游戏开始按键。

理论学习:

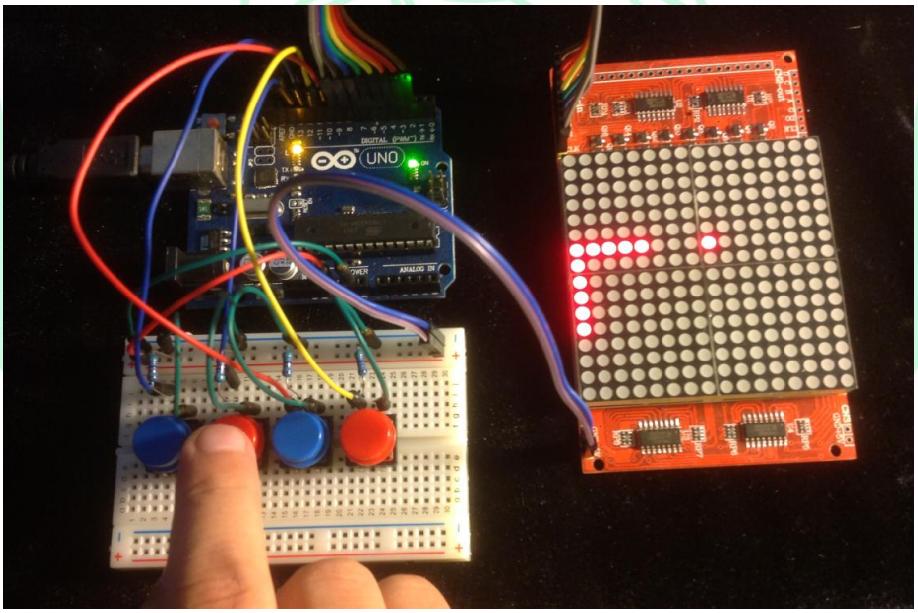
- ✓ 通过程序自习体会，如何用随即函数随即出一个数字，然后在点阵上显示出来。
- ✓ 学会怎么判断蛇身碰到边框或者自身。
- ✓ 学会怎么判断吃到食物，并且吃到食物后蛇身长度加一。



3.28.1 原理图



3.28.2 连接图



3.28.1 贪吃蛇游戏

3.29 卫星 GPS 定位

实验现象：

- ✓ Arduino 接收到 GPS 模块发来的信息直接转发给 PC。
- ✓ Arduino 把 GPS 信息进行处理解析为有用信息，然后发送给 PC。

理论学习：

- ✓ GPS 模块是接收卫星的信息然后用串口发送出来的装置，本店 GPS 采用 U-BLOX 模组，自带可充电后备电池（以支持温启动或热启动，后备电池在主电源断电后，可以维持半小时左右的 GPS 接收数据保存）；
- ✓ 模块通过串口与外部系统连接，串口波特率：4800、9600（默认）、38400、57600 等不同速率;
- ✓ 兼容 5V/3.3V 单片机系统，可以非常方便的与您的产品进行连接。
- ✓ 最常用指令解析：

➢ $GPGGA（GPS 定位信息，Global Positioning System Fix Data）

$GPGGA 语句的基本格式如下（其中 M 指单位 M，hh 指校验和，CR 和 LF 代表回车换行，下同）：

$GPGGA,(1),(2),(3),(4),(5),(6),(7),(8),(9),M,(10),M,(11),(12)*hh(CR)(LF)

(1) UTC 时间，格式为 hhmmss.ss；
(2) 纬度，格式为 ddmm.mmmmm（度分格式）；
(3) 纬度半球，N 或 S（北纬或南纬）；
(4) 经度，格式为 dddmm.mmmmm（度分格式）；
(5) 经度半球，E 或 W（东经或西经）；
(6) GPS 状态，0=未定位，1=非差分定位，2=差分定位；
(7) 正在使用的用于定位的卫星数量（00~12）
(8) HDOP 水平精确度因子（0.5~99.9）
(9) 海拔高度（-9999.9 到 9999.9 米）
(10) 大地水准面高度（-9999.9 到 9999.9 米）
(11) 差分时间（从最近一次接收到差分信号开始的秒数，非差分定位，此项为空）
(12) 差分参考基站标号（0000 到 1023，首位 0 也将传送，非差分定位，此项为空)

举例如下：

$GPGGA,023543.00,2308.28715,N,11322.09875,E,1,06,1.49,41.6,M,-5.3,M,,*7D

➢ $GPRMC（推荐定位信息，Recommended Minimum Specific GPS/Transit Data）

$GPRMC 语句的基本格式如下：

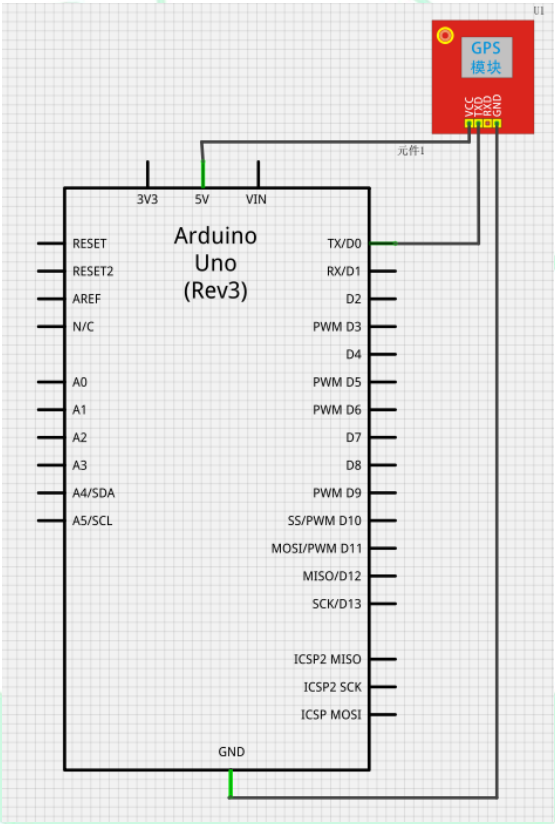
$GPRMC,(1),(2),(3),(4),(5),(6),(7),(8),(9),(10),(11),(12)*hh(CR)(LF)

(1) UTC 时间，hhmmss（时分秒）
(2) 定位状态，A=有效定位，V=无效定位
(3) 纬度 ddmm.mmmmm（度分）
(4) 纬度半球 N（北半球）或 S（南半球）

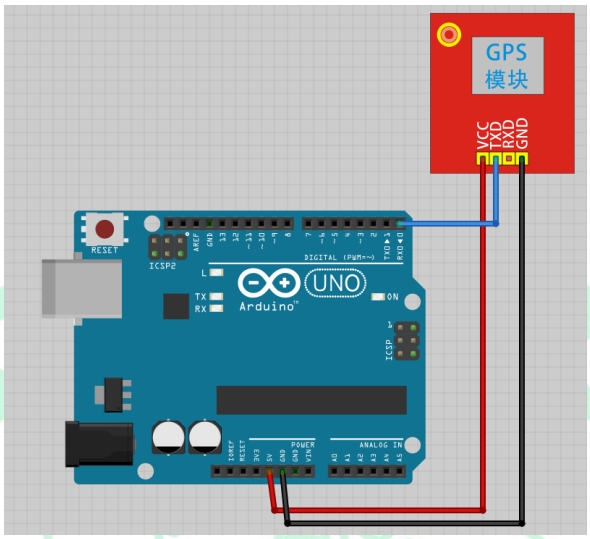
(5) 经度 dddmm.mmmmm（度分）
(6) 经度半球 E（东经）或 W（西经）
(7) 地面速率（000.0~999.9 节）
(8) 地面航向（000.0~359.9 度，以真北方为参考基准）
(9) UTC 日期，ddmmyy（日月年）
(10) 磁偏角（000.0~180.0 度，前导位数不足则补 0）
(11) 磁偏角方向，E（东）或 W（西） (12) 模式指示（A=自主定位，D=差分，E=估算，N=数据无效）
举例如下：

$GPRMC,023543.00,A,2308.28715,N,11322.09875,E,0.195,,240213,,,A*78

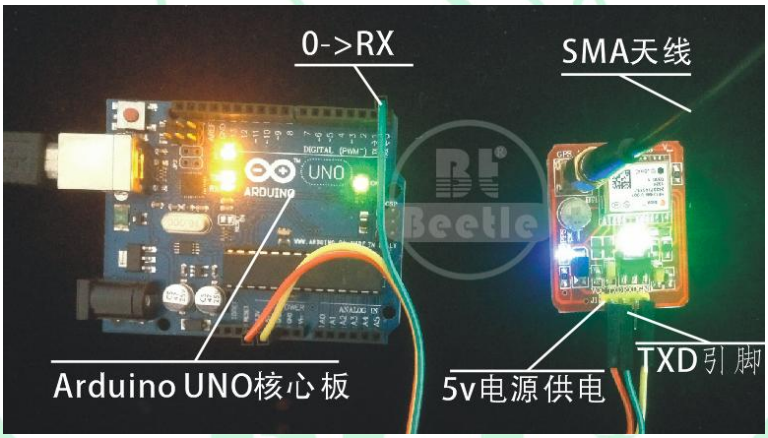
- 其它指令解析请参考 GPS 手册。



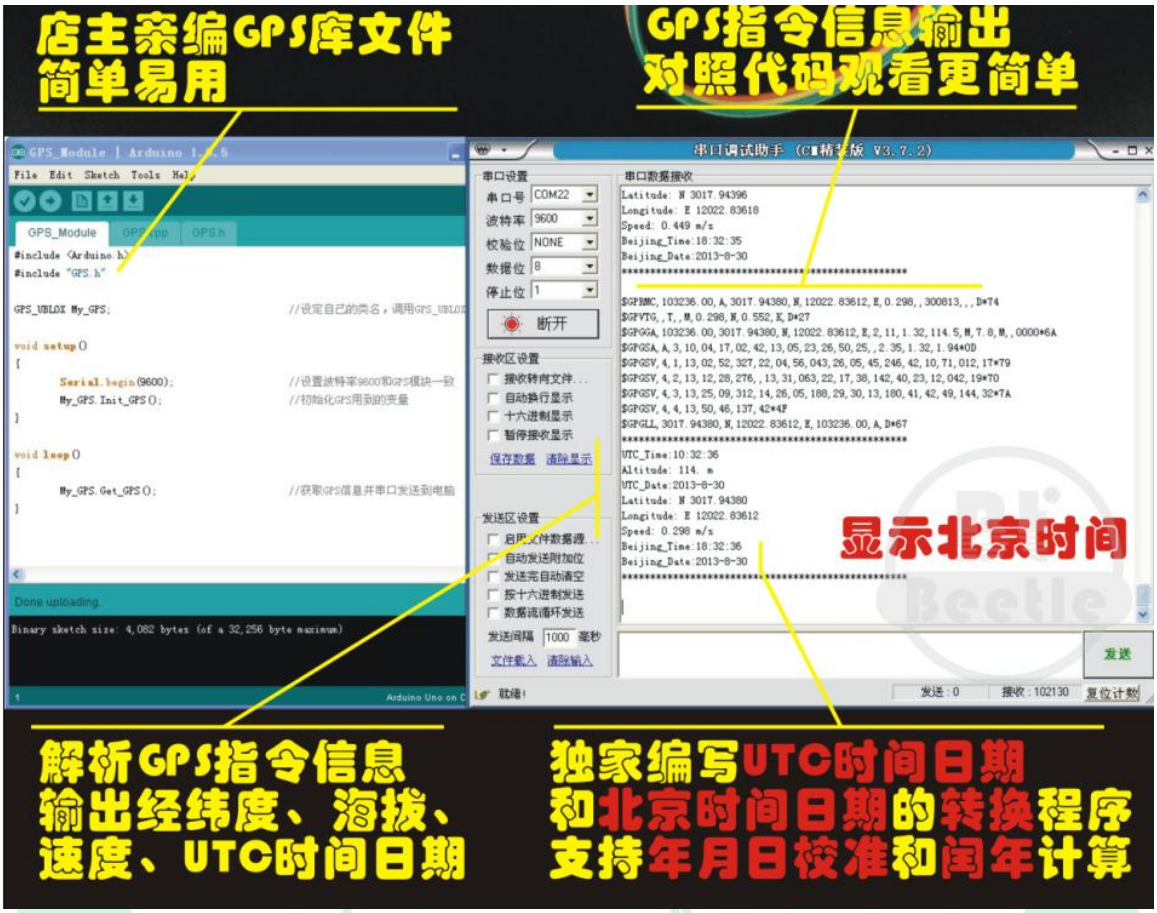
3.29.1 原理图



3.29.2 连接图



3.29.3 实物效果图



3.29.4 串口输出图

3.30 手机蓝牙控制 ARDUINO 开关灯

实验现象：使用安卓手机控制继电器的开关。

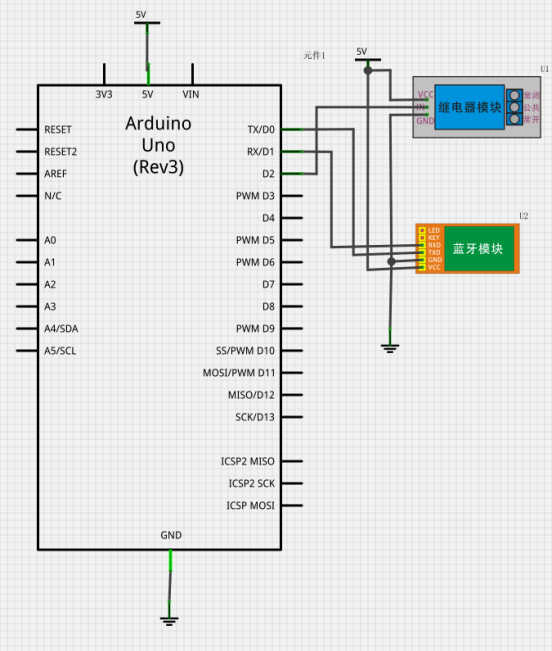
理论学习：

- ✓ 本实验采用本店开发的 BT-HC05 蓝牙模块，该模块主从一体，可以软件设置是主机还是从机。（发货默认从机，波特率 9600）。
- ✓ 模块详细指令手册，参考资料包（Arduino 独家整理资料包\7.芯片及模块资料\蓝牙）
- ✓ 实际上我们出货时候的设置就可以直接拿来用了：
  1. 安卓手机端安装蓝牙调试助手（Arduino 独家整理资料包\7.芯片及模块资料\蓝牙\蓝牙串口助手 v1.97.apk）
  2. 把蓝牙模块按照连接图接线。
  3. 打开蓝牙调试助手。

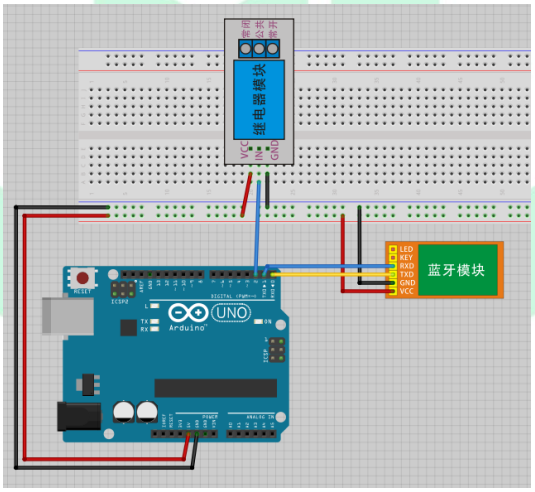
4. 按照以下图片操作：

5. 输入控制字符串：

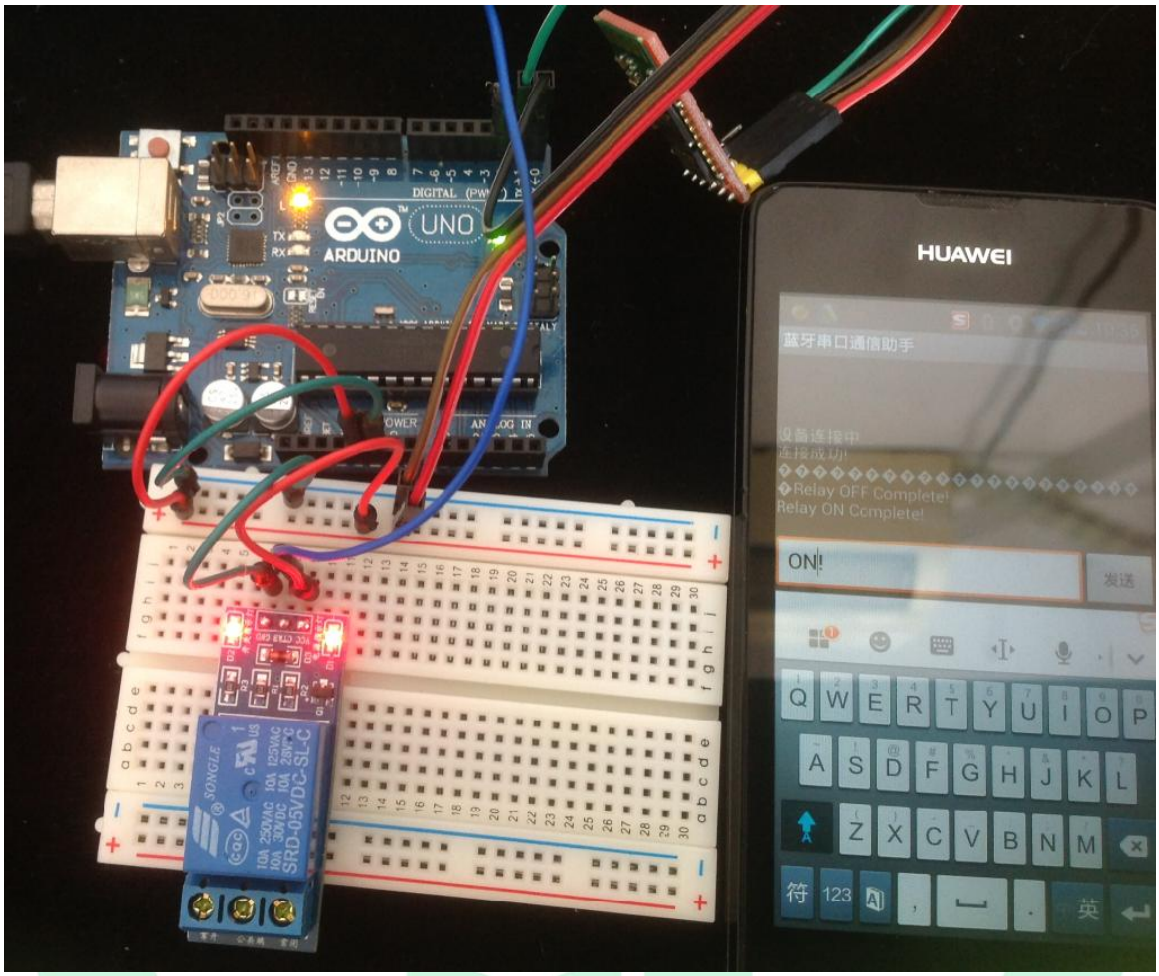
ON! 指令会打开继电器（指示灯 D2 点亮）。
OFF! 指令会关闭继电器（指示灯 D2 熄灭）。
（很简单的就可以手机控制家里的家电了哦(*^__^*)）
（有兴趣的同学可以研究一下软件的键盘模式，把指令设置成按键，谁用谁知道 O(∩_∩)O~）



3.30.1 原理图



3.30.2 连接图



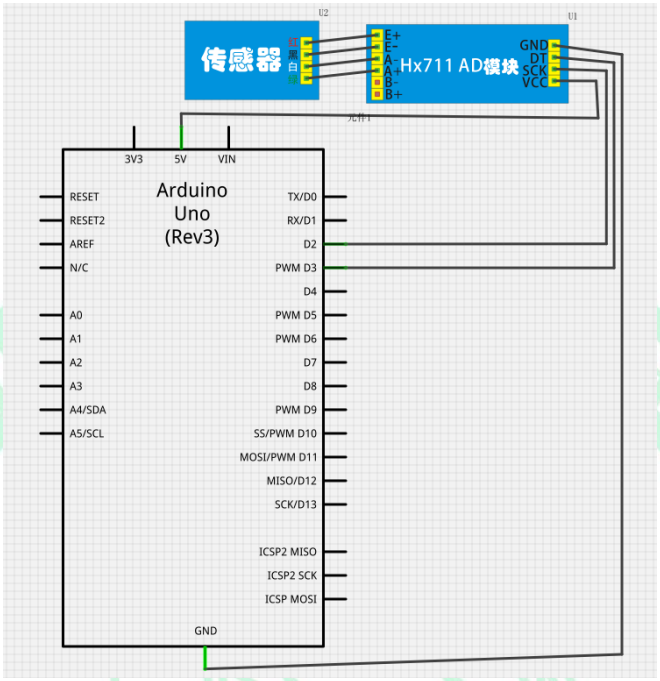
3.30.3 实物效果图

3.31 ARDUINO 电子秤制作

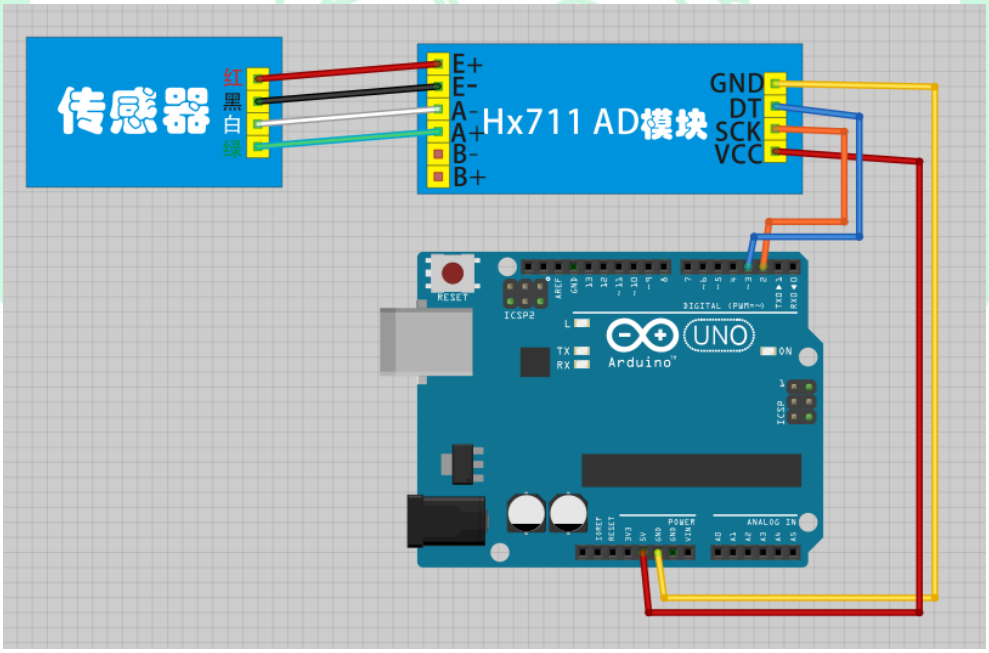
实验现象：手提电子秤挂 100g 砝码，串口输出 100g（误差±1g）。

理论学习：

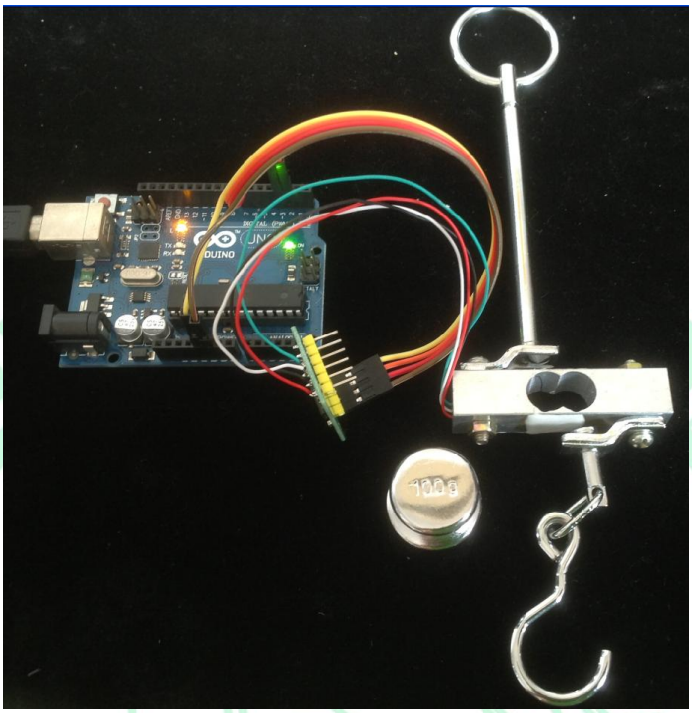
- ✓ HX711 AD 模块是高精度 24bit AD，内部带 128 倍增益，可方便测量 mV 电压。
- ✓ 3Kg 压力拉力传感器分辨率 1mV/V (例如传感器供电 5v，传感器最大输出 5mV)。
- ✓ Arduino 把接收到的 HX711 的串行数据，转换为砝码的重量，通过串口输出显示。



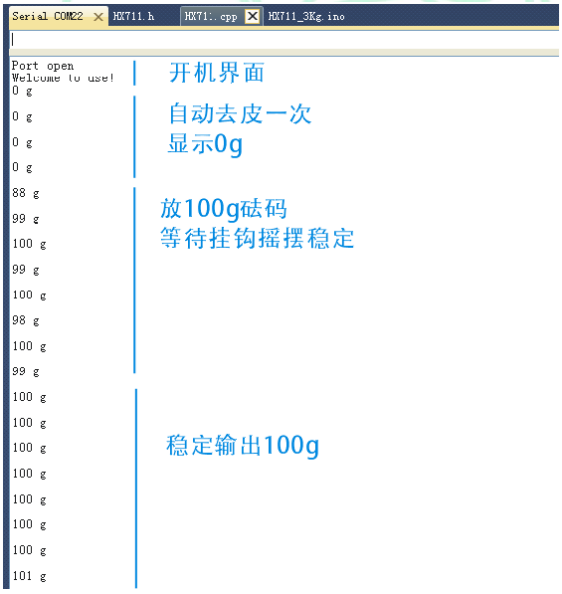
3.31.1 原理图



3.31.2 连接图



3.31.3 实物效果图



3.31.4 串口显示界面

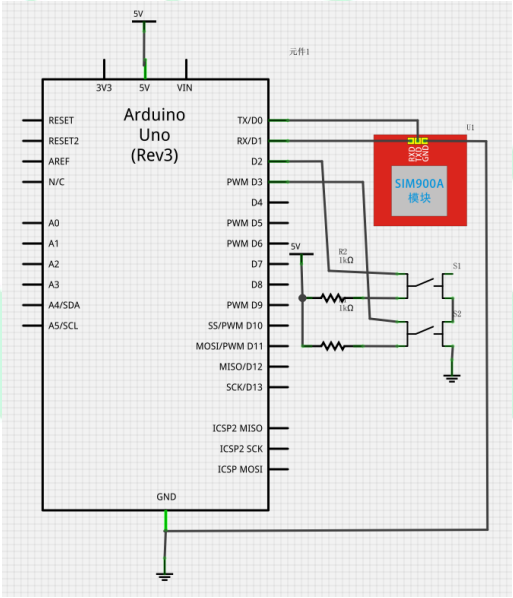
3.32 GSM 模块 SIM900A 拨打电话

实验现象：按下按键 1 后，SIM900 向指定手机号拨打电话。按下按键 2，挂断电话。

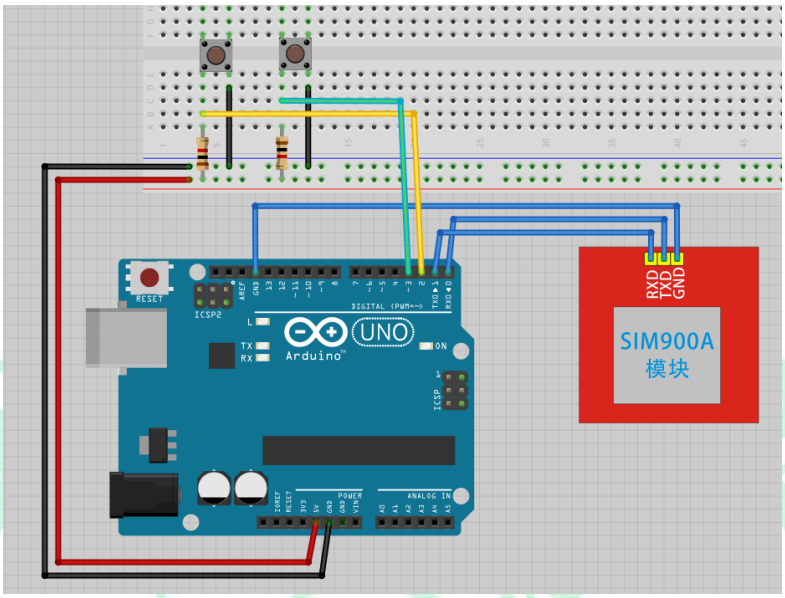
理论学习：

- ✓ BT-MINI-SIM900A 是 本店开发的一款高性能工业级 GSM/GPRS 模块（开发板）。BT-MINI-SIM900 模块板载 SIMCOM 公司的工业级双频 GSM/GPRS 模块：SIM900A，工作频段双频：900/1800Mhz，可以低功耗实现语音、SMS（短信，不支持彩信）、数据和传真信息的传输。BT-MINI-SIM900A 模块使用串口通讯，支持 5v 电源适配器供电或者手机电池供电。
- ✓ ATD 拨打电话指令。
- ✓ ATH 挂断电话指令。
- ✓ 每条指令后面要加回车（\r\n）。
- ✓ 模块使用顺序：

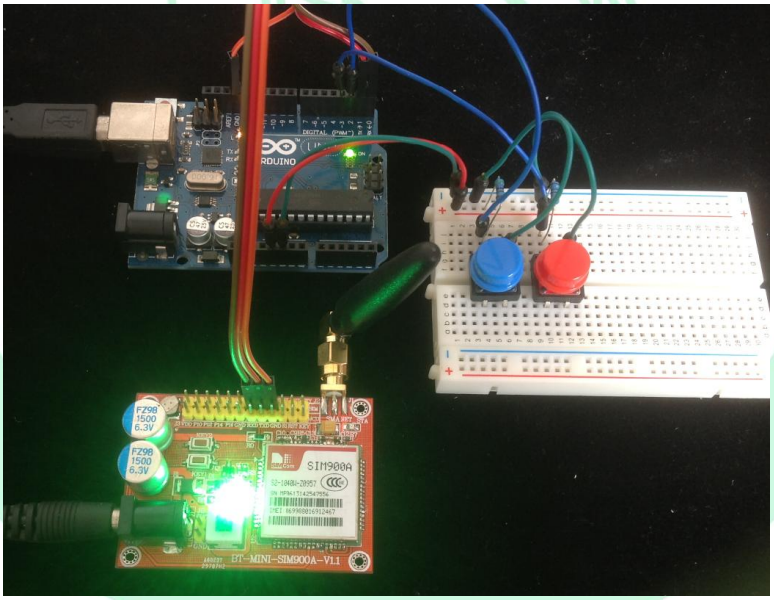
1. 将一张移动手机卡插到模块背面的 SIM 卡插槽。
2. 使用标配的 5V2A 电源给 SIM900A 模块供电。
   按下 SIM900A 模块上的 KEY2 按键，给模块开机。等待 LED 指示灯 STA 注册到网络（64ms 亮/3000ms 灭）
3. 下载例程到 Arduino（一定要在未连接 RX/TX 引脚时候下载程序，否则会下载程序不成功）。
4. 按照连接图连接各导线。
5. 按面包板上按键 1，拨打指定手机号码。
6. 按面包板上按键 2，挂断电话。



3.32.1 原理图



3.32.2 连接图



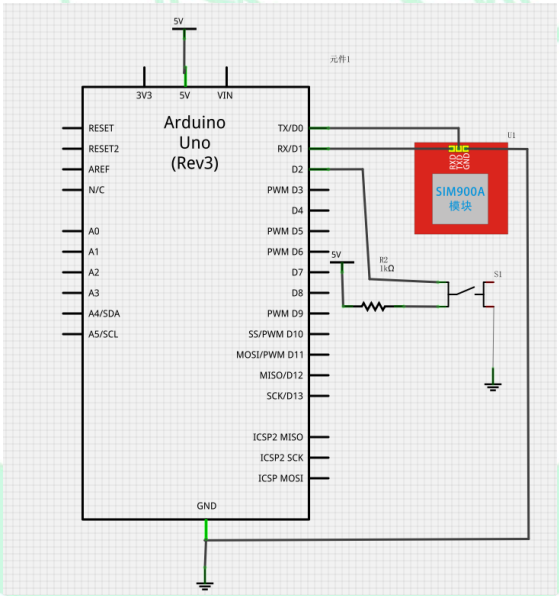
3.32.3 实物效果图

3.33 GSM 模块 SIM900A 发送短信

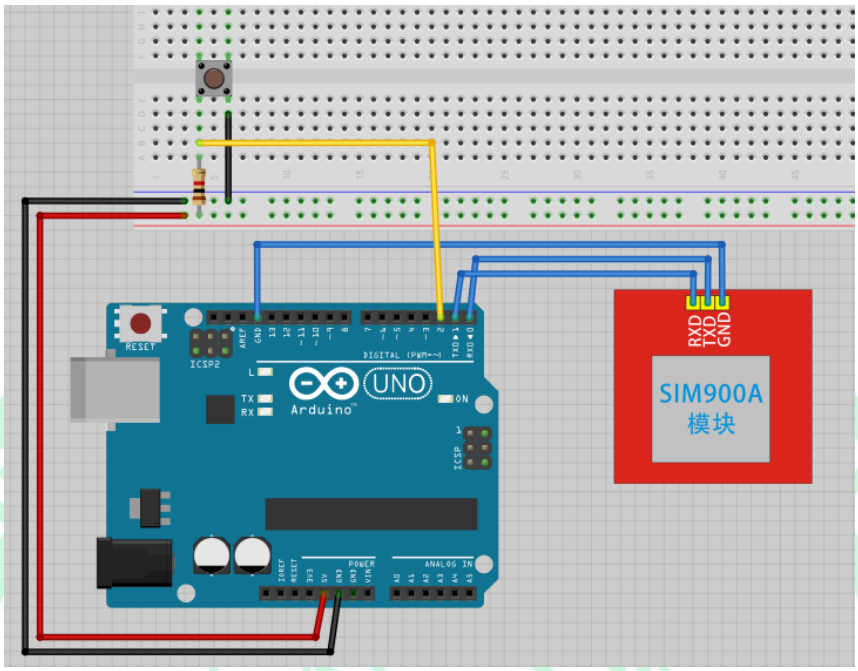
实验现象：按下面包板上按键后，向指定手机发送一条短信，短信内容为“Hello World！”。

理论学习：

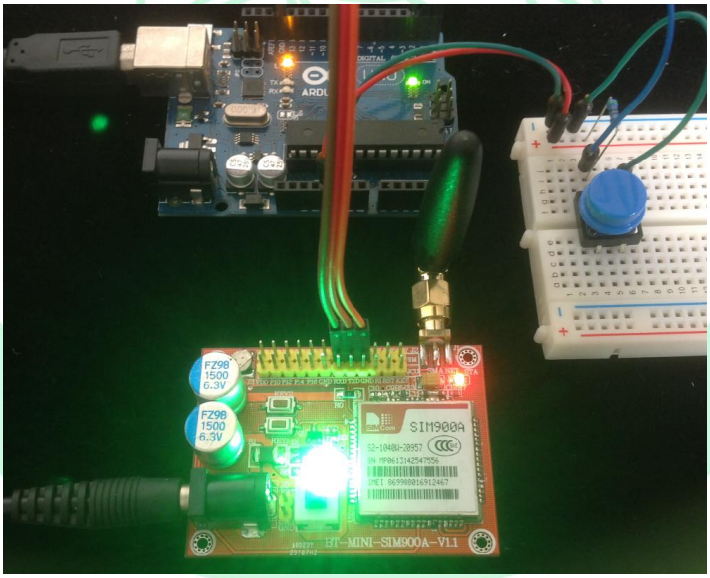
- ✓ AT+CSCS=”GSM”设置为缺省字符集
- ✓ AT+CMGF，用于设置短消息模式，SIM900A 支持 PDU 模式和文本（TEXT）模式等 2 种模式，发送：AT+CMGF=1，即可设置为文本模式。
  本短信发送英文短信，因此使用 AT+CMGF=1
- ✓ AT+CMGS,用于发送短信,在"GSM"字符集下,最大可以发送 180 个字节的英文字符，在"UCS2"字符集下，最大可以发送 70 个汉字（包括字符/数字）。
- ✓ 模块使用顺序：
  1.将一张移动手机卡插到模块背面的 SIM 卡插槽。
  2．使用标配的 5V2A 电源给 SIM900A 模块供电。
  3．按下 SIM900A 模块上的 KEY2 按键，给模块开机。等待 LED 指示灯 STA 注册到网络（64ms 亮/3000ms 灭）
  4．下载例程到 Arduino（一定要在未连接 RX/TX 引脚时候下载程序，否则会下载程序不成功）。
  5．按照连接图连接各导线。
  6．按面包板上按键后，向指定手机发送一条短信。



3.33.1 原理图



3.33.2 连接图



3.33.3 实物效果图