**智能小车**

周珂 车永祥

**总述：**

**上上周我们组完成了智能小车的拼装，模仿现有代码实现了前进，后退，和转弯。**

**本周我们的重点放在Arduino的代码学习上和编写上。**

**代码学习：**

**Arduino编程语言，Arduino UNO板，逻辑电路**

**完成的例程有：蜂鸣器报警，黑线循迹，红外避障实验（基础）**

**Arduino开发语言**

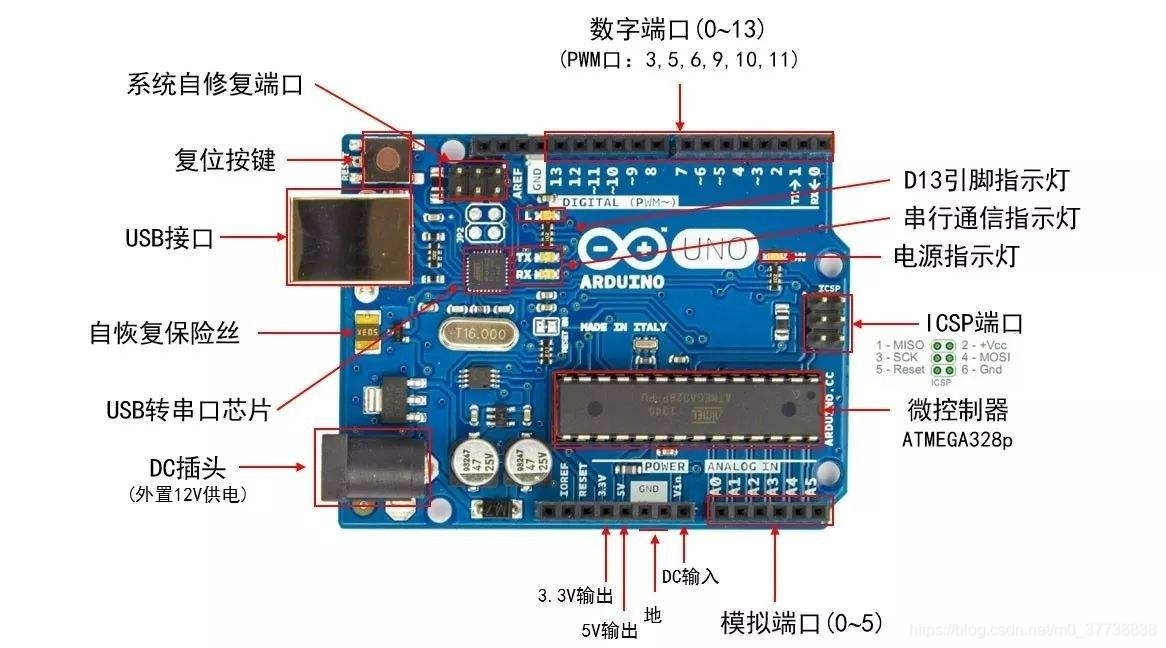
Arduino使用C/C++编写程序，虽然C++兼容C语言，但这是两种语言，C语言是一种面向过程的编程语言，C++是一种面向对象的编程语言。早期的Arduino核心库使用C语言编写，后来引进了面向对象的思想，目前最新的Arduino核心库采用C与C++混合编写而成。

通常我们说的Arduino语言，是指Arduino核心库文件提供的各种应用程序编程接口（Application Programming Interface，简称API）的集合。这些API是对更底层的单片机支持库进行二次封装所形成的。例如，使用AVR单片机的Arduino的核心库是对AVR-Libc（基于GCC的AVR支持库）的二次封装。

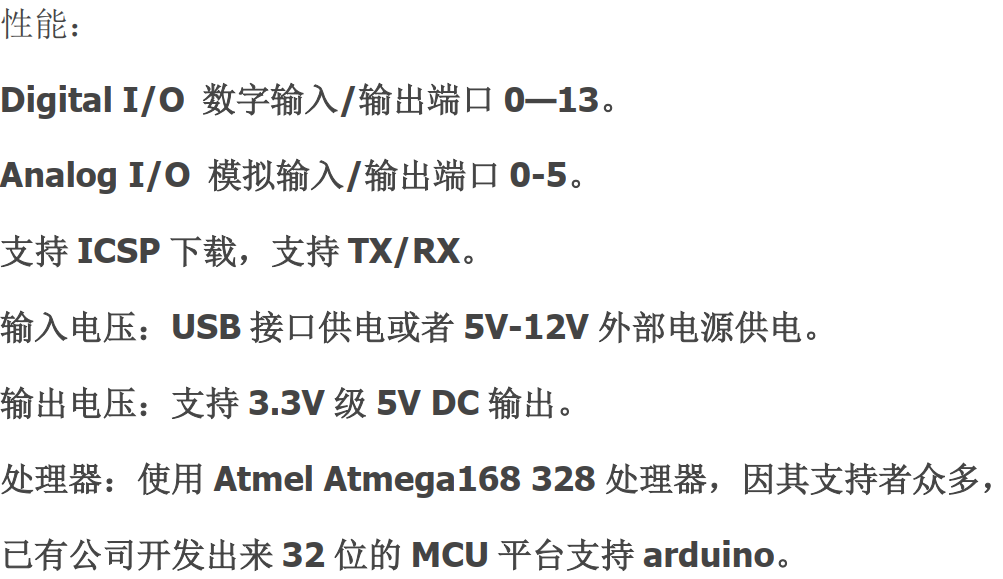
传统开发方式中，需要通过配置多个寄存器来实现相应功能，而在Arduino中，繁杂的寄存器被封装成简单的API，能进行直观控制，增强程序的可读性的同时也提高了开发效率。

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。

1. **Arduino UNO板**



1. **Arduino单片机**



1. **Arduino板上几个特殊的端口**

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。

1. Arduino编程

（1）常量设置（C语言部分）

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）。

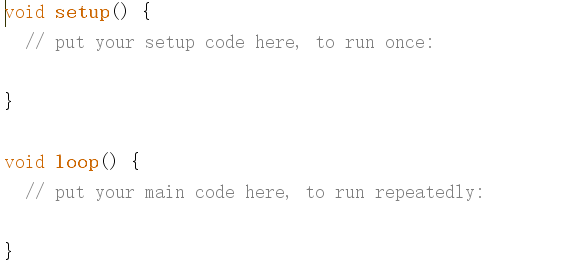
（2）Arduino语言

结构:

声明变量及接口名称（ int val;int ledPin=13;）。

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);）。

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。



功能：

1. 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。

1. 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。

1. 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，

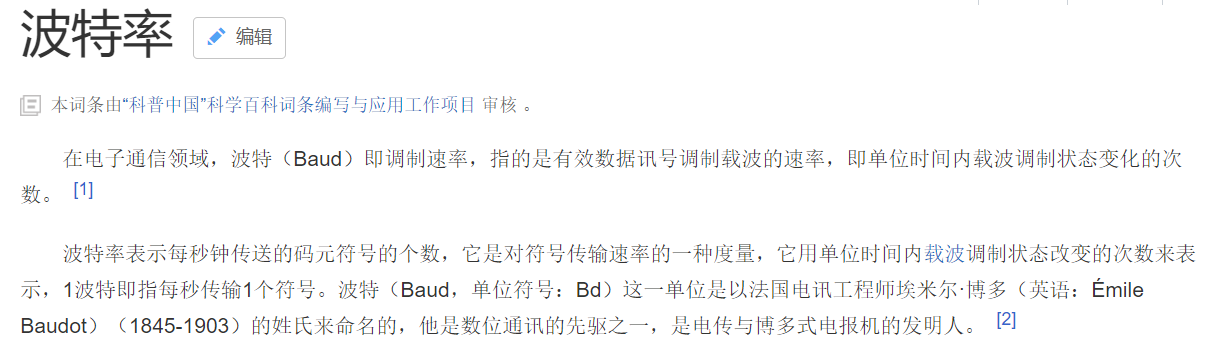
dataPin 为数据口， clockPin 为时钟口， bitOrder 为数据传输方向（MSBFIRST 高位在前， LSBFIRST 低位在前）， value 表示所要传送的数据（ 0～255），另外还需要一个 IO 口做74HC595 的使能控制。

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。

1. 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
2. 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。
3. 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。

1. 理解与解释



* 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。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(接口名称, HIGH 或 LOW)——将数字接口值至高或低。digitalWrite(pin, value)将数位脚位指定为开或关。脚位必须先通过 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*

* digitalRead（接口名称） ——读出数字接口的值。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*

* 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。

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）*

* 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。

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*

* delay()——延时一段时间， delay(1000)为一秒。

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*

* 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
* Serial.read()——读取持续输入的数据。
* 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)。

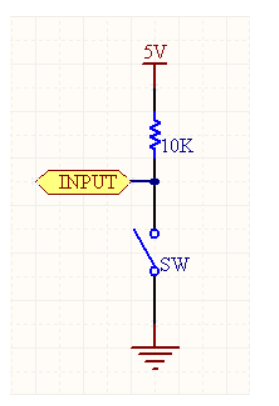
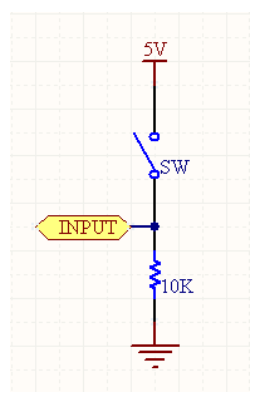
Serial.println(数据，数据的进制)——从串行端口输出数据，跟随一个回车和一个换行符。这个函数所取得的值与 Serial.print()一样

* 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*

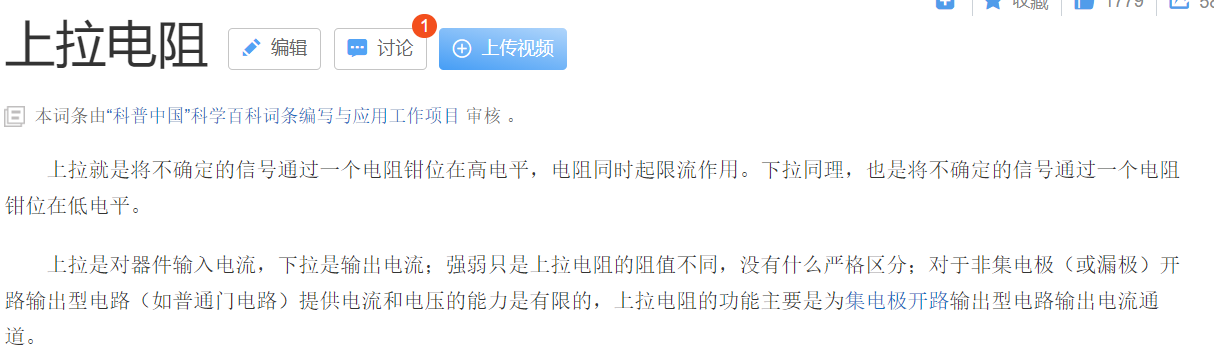
1. 数字输入

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。

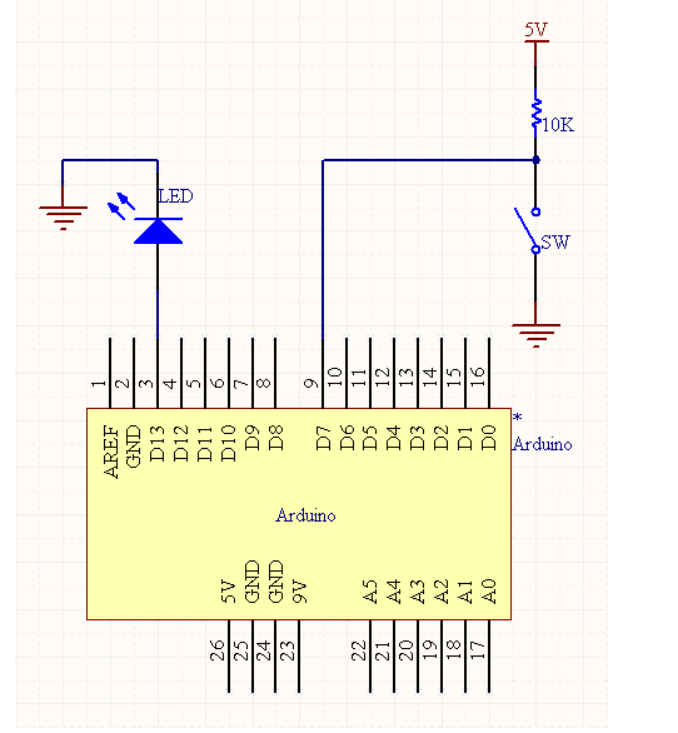
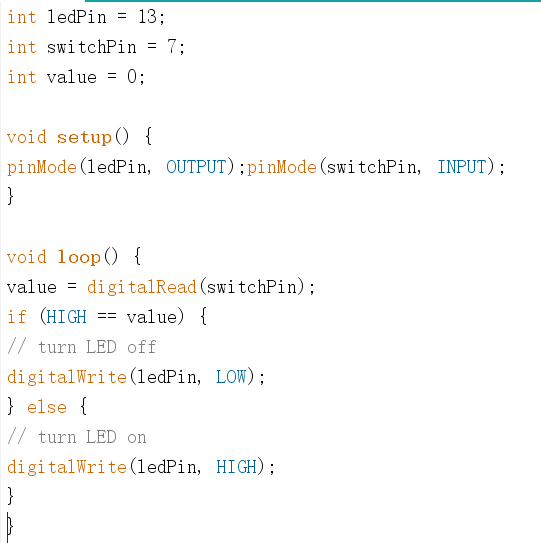
1. 在正逻辑电路中，开关一端接电源，另一端则通过一个 10K 的下拉电阻接地，输入信号从开关和电阻间引出。当开关断开的时候，输入信号被电阻“拉”向地，形成低电平（ 0V）；当开关接通的时候，输入信号直接与电源相连，形成高电平。对于经常用到的按压式开关来讲，就是按下为高，抬起为低

 B.

1. 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。



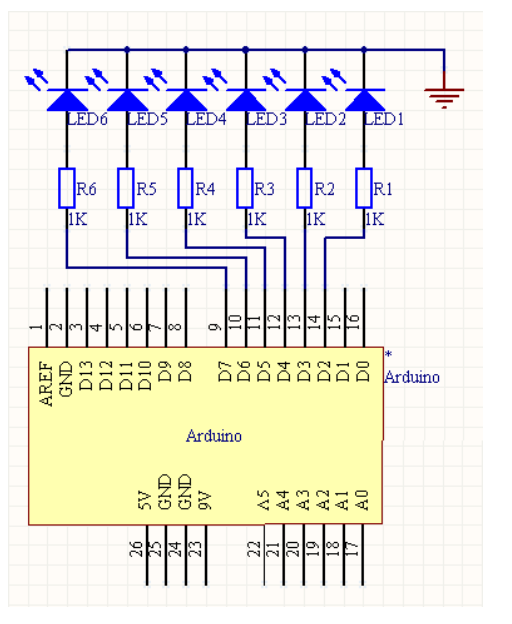
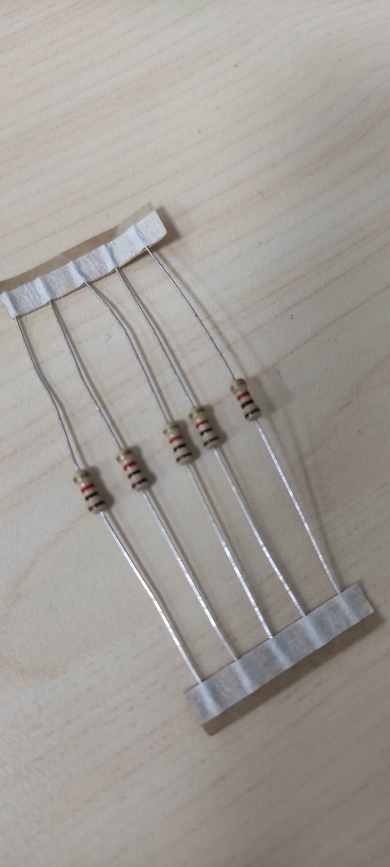
验证方式：

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

1. 数字输出

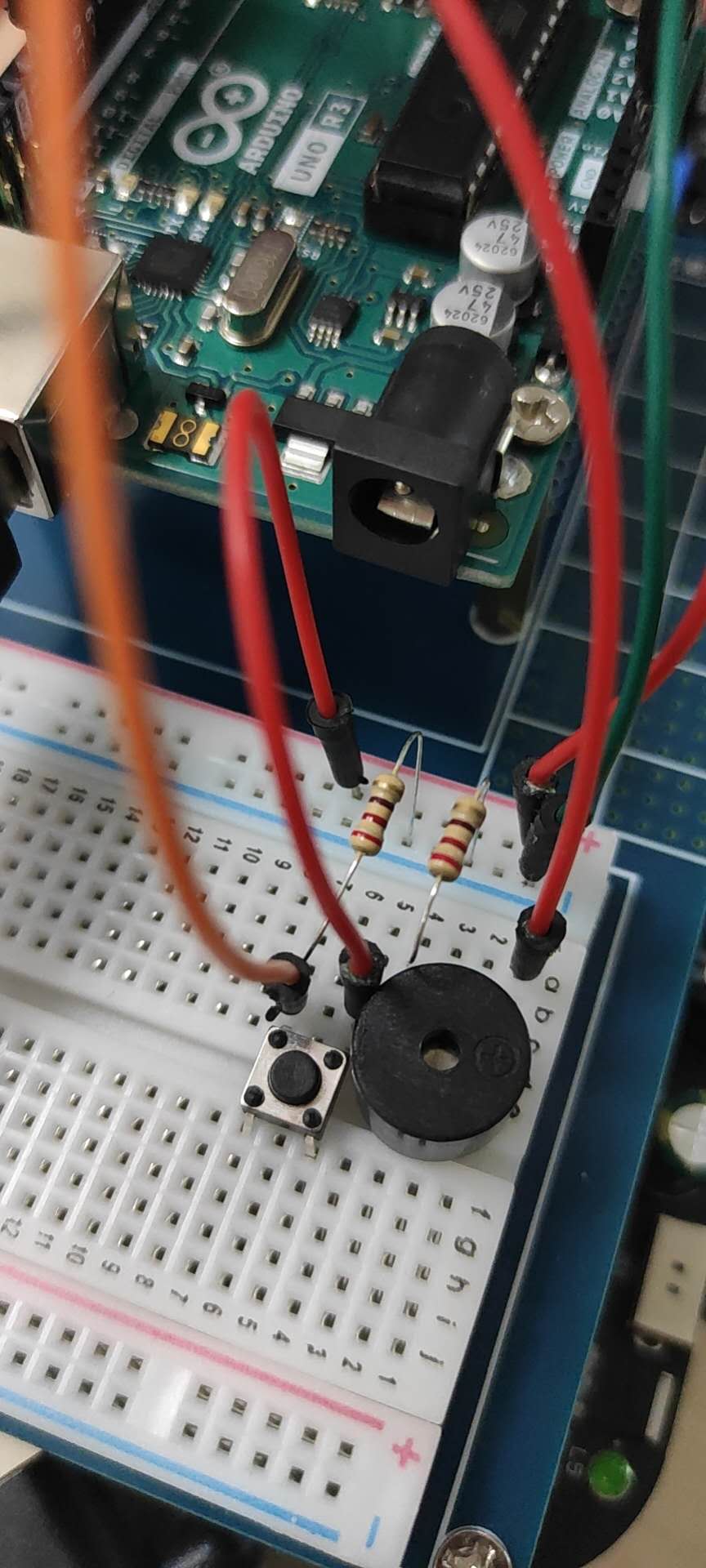
Arduino 的数字 I/O 被分成两个部分，其中每个部分都包含有 6 个可用的 I/O 管脚，即管脚 2 到管脚 7 和管脚 8 到管脚 13。除了管脚 13上接了一个 1K 的电阻之外，其他各个管脚都直接连接到 ATmega 上。我们可以利用一个 6 位的数字跑马灯，来对 Arduino 数字 I/O 的输出功能进行验证，以下是相应的原理图：

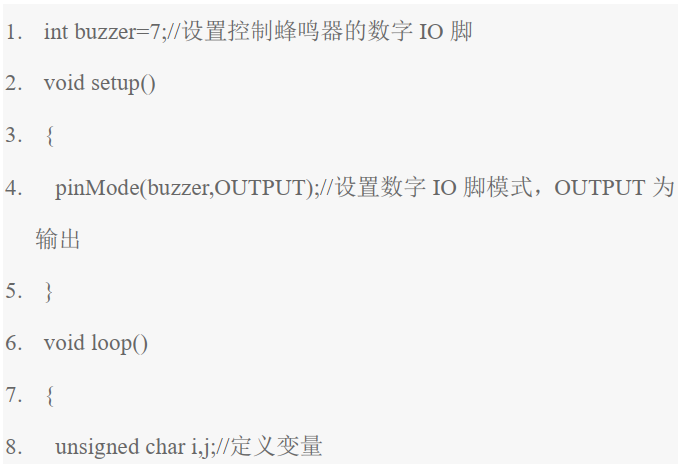
 

电路中在每个 I/O 管脚上加的那个 1K 电阻被称为限流电阻，由于发光二极管在电路中没有等效电阻值，使用限流电阻可以使元件上通过的电流不至于过大，能够起到保护的作用。

1. 完成实例代码讲解

例：蜂鸣器实验







A． 智能小车按键启动和蜂鸣器报警

1. 关键函数说明

接受到高电平的时候，蜂鸣器发出声音，低电平时不发出声音

void keysacn()

{

int val;

val=digitalRead(key);//读取数字7 ，将电平值赋给val

while(!digitalRead(key))//当按键没被按下时，一直循环

{

val=digitalRead(key);//此句可省略，可让循环跑空

}

while(digitalRead(key))//当按键被按下时

{

delay(10); //延时10ms

val=digitalRead(key);//读取数字7 电平值赋给val

if(val==HIGH) //第二次判断按键是否被按下

{

digitalWrite(beep,HIGH); //蜂鸣器响

while(!digitalRead(key)) //判断按键是否被松开

digitalWrite(beep,LOW); //蜂鸣器停止

}

else

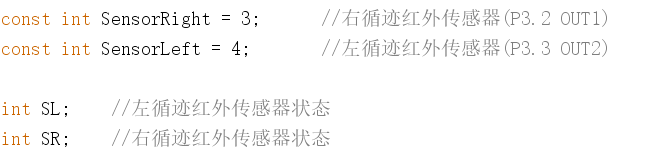
digitalWrite(beep,LOW);//蜂鸣器停止

}

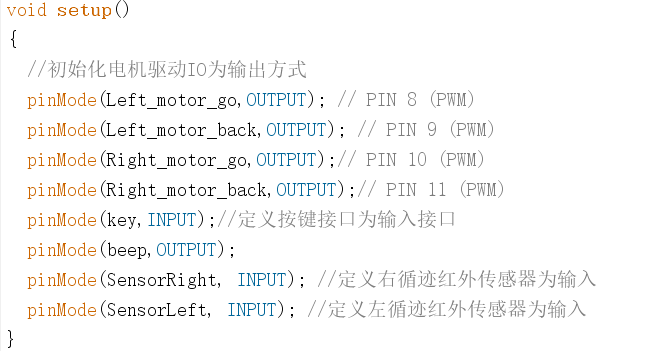
}

1. 智能小车循迹
2. 关键代码说明

小车循迹是由地面感应灯实现的，具体思路是感应到黑色的线时，让感应灯熄灭，在其他颜色的地面上让感应灯发亮，这个调节是靠物理手动调节实现的，在代码里面只是将红外感应信号传了回去。



设置左循迹红外传感器信号和右循迹红外传感器信号。



将左右循迹传感器作为输入输出。

