第 5 章

Arduino 便捷的模拟/数字转换

自然界所有的信号都是模拟的，例如温度、光照强度、压力等。单片机只能处理数字离散信号，因而需要一种将模拟信号转化为数字信号形式的方法。将模拟信号转换成数字信号的电路，称为模数转换器，简称 A/D 转换器或 ADC(Analog to Digital Converte)，A/D 转换的作用是将时间连续、幅值也连续的模拟量转换为时间离散、幅值也离散的数字信号。

标准 MCS-51 单片机芯片内部不具备 A/D 转换功能，需要外接 AD0809 等 A/D 转换芯片，而 AVR 系列单片机都配备 A/D 转换子系统。

Arduino 将复杂的 AVR 单片机 A/D 转换过程做了很好的封装，只要使用 analogRead()即能便捷地进行 A/D 转换。

5.1 Arduino UNO 板上的 A/D 转换

在 Arduino UNO 开发板上，有一排标着 A0～A5 的引脚具有模拟信号输入的功能。Arduino 中的片内 A/D 转换设备通过逐次逼近法测量输入电压的大小，得到输入电压，转换成 0～1023 的数字值。

5.1.1 逐次逼近模数转换技术

将模拟信号转换为数字信号一般分为 4 个步骤，即取样、保持、量化和编码。逐次逼近转换过程和用天平称物重非常相似，从最重的砝码开始试放，与被称物体进行比较，若物体重于砝码，则该砝码保留，否则移去。再加上第二个次重砝码，由物体的重量是否大于砝码的重量决定第二个砝码是留下还是移去。照此一直加到最小一个砝码为止。将所有留下的砝码重量相加，就得到该物体的重量。

ATmega328 采用逐次逼近技术，包括一个数模转换器、一个控制器和一个比较器来执行模数转换。从最高有效位开始向下到最低有效位，控制器逐个开启每一位并且生成一个模拟信号，在数模转换器的帮助下，将生成的模拟信号与原始输入模拟信号进行对比。基于对比结果，控制器更换或者离开当前位并且使能下一个最高有效位。该过程一直持续直到生成所有有效位的对比结果。

5.1.2 Arduino UNO 上的 A/D 引脚

Arduino UNO 板包含一个 6 通道(Mini 和 Nano 有 8 个通道,Mega 有 16 个通道)、10 位模拟/数字转换器,它将 0～5V 的输入电压映射为 0～1023 的整数值。读数之间的关系是:5V/1024 单位,或 0.0049V(4.9mV)每单位。输入范围和精度可以使用 analogReference()改变。它需要大约 100μs(0.0001s)来读取模拟输入,所以最大的阅读速度是 10000 次每秒,或 4.88mV 每单位。

在 Arduino UNO 板下方 ANALOG IN 区域有 6 个 A/D 转换输入引脚,分别标记为 A0、A1、A2、A3、A4、A5,这些模拟引脚可作为数字引脚 D14～D19 使用,如图 5-1 所示。

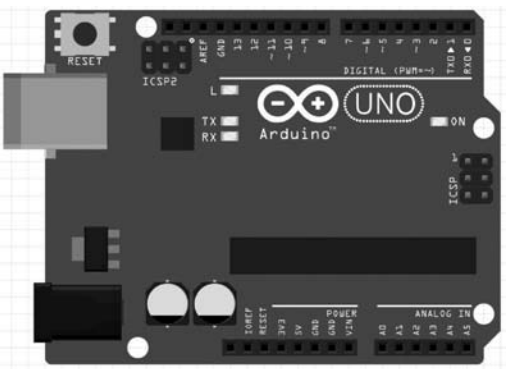


图 5-1 Arduino UNO 的 6 个 A/D 转换引脚

5.1.3 Arduino 中的 A/D 转换语句

Arduino 把 ATmega 单片机中的 10 位 A/D 转换时需要的复杂寄存器设置做了很好的封装,用户只需使用 analogRead(0)、analogRead(1)命令就可以读出 A0、A1 引脚的模拟电平值了。

1. analogRead(pin)

从指定的模拟引脚读取数据值。

返回值:整数型 int(0～1023)。

参数:pin 为 A/D 引脚标号,Arduino UNO 为 0～5,Arduino nano 为 0～7,Mega 为 0～15。

2. analogReference (uint8_t type)

配置模式引脚的参考电压。输入范围和精度可以使用 analogReference()改变,配置用于模拟输入的基准电压(即输入范围的最大值)。函数 analogRead()在读取模拟值之后,根据参考电压将模拟值转换到[0,1023]区间。

参数 type 有如下几种选择。

(1) DEFAULT:基准电压,默认为 5V(Arduino UNO 板为 5V)或 3.3V(Arduino

Nano 板为 3.3V)。

(2) INTERNAL：低功耗模式。在 ATmega168 和 ATmega328 上以 1.1V 为基准电压，在 ATmega8 上以 2.56V 为基准电压。

- INTERNAL1V1：以 1.1V 为基准电压(Arduino Mega)。
- INTERNAL2V56：以 2.56V 为基准电压(Arduino Mega)。

(3) EXTERNAL：以 AREF 引脚(0～5V)的电压作为基准电压，但是要低于 5V。

如果模拟输入引脚没有连入电路，则由 analogRead()返回的值将根据多项因素(例如，你的手靠近电路板等)而产生波动。

注意：如果使用 AREF 引脚上的电压作为基准电压，那么在调用 analogRead()前，必须先设置参考类型为 EXTERNAL；否则，外接的参考电压将会损坏内部基准电压源，这可能会损坏 Arduino 板上的单片机。

不要在 AREF 引脚上使用任何小于 0V 或超过 5V 的外部电压，以免损坏单片机。

5.1.4 A/D 转换器主要技术参数

1. 分辨率

A/D 转换器的分辨率以输出二进制(或十进制)数的位数来表示，它表明 A/D 转换器对输入信号的分辨能力。

分辨率＝参考电压/(总元素－1)，总元素＝2^n，n 为位数。

假设用 4 位(0～15)来表示量化等级，分辨率＝5V/(16－1)＝333mV，对于 Arduino 中的 ATmega328 来说，为 10 位(0～1023)转换精度，分辨率＝5V/1023＝4.88mV。

量化级越高，模拟信号转换时的精度就越高，在将模拟信号转化为数字信号时，为什么我们不使用当前技术为数字系统提供的最大比特数呢？因为必须考虑到该数据系统的计算能力以及所需要的系统分辨率。

ATmega328 的 A/D 系统配备一个 10 位分辨率的逐步逼近式转换器，在给定时间只能对一个 A/D 转换器通道进行转换。

2. 转换误差

转换误差通常是以输出误差的最大值形式给出。它表示 A/D 转换器实际输出的数字量和理论上的输出数字量之间的差别，常用 LSB(最低有效位)的倍数表示。例如，给出相对误差≤±LSB/2，这就表明实际输出的数字量和理论上应得到的输出数字量之间的误差小于最低位的半个字。

ATmega328 的转换误差是±2 最低有效位(LSB)绝对精度，即意味着±9.76mV 的分辨率。

3. 转换时间

转换时间是指 A/D 转换器从转换控制信号到来开始，到输出端得到稳定的数字信号为止所经过的时间。

ATmega328 的 A/D 转换需要 13 个 A/D 时钟转换时间，最高采样速率为 76.9kSPS。

Arduino 系统封装采用了较高的分频比，使最高采样速度降为 10kSPS。当应用需要高速取样时，可以采取直接操控寄存器方式实现，详见第 11 章。

5.2 A/D 转换基本实验

5.2.1 读取模拟引脚上的模拟值并显示出来

微课视频 2 读取模拟引脚上的模拟值并显示出来

1. 电路原理

通常一个电位器有 3 个引脚，其中两个连接到电阻材料，第三个引脚(通常在中间)被连接到可旋转并接触电阻材料的任何位置的滑动触点。当电位器旋转时，滑动触点和一个引脚之间的电阻增加，而和另一个引脚间的电阻减小。当滑动触点移向底端时，滑动触点(图 5-2 中带箭头的线)和地之间具有较低的电阻，而连接到 5V 的一端有更高的电阻。随着滑动触点向下移动，模拟引脚上的电压将减小(最小为 0V)。滑动触点向上移动将产生相反的效果，引脚上的电压将增加(最大为 5V)。电路图及实物连接图如图 5-2 所示。

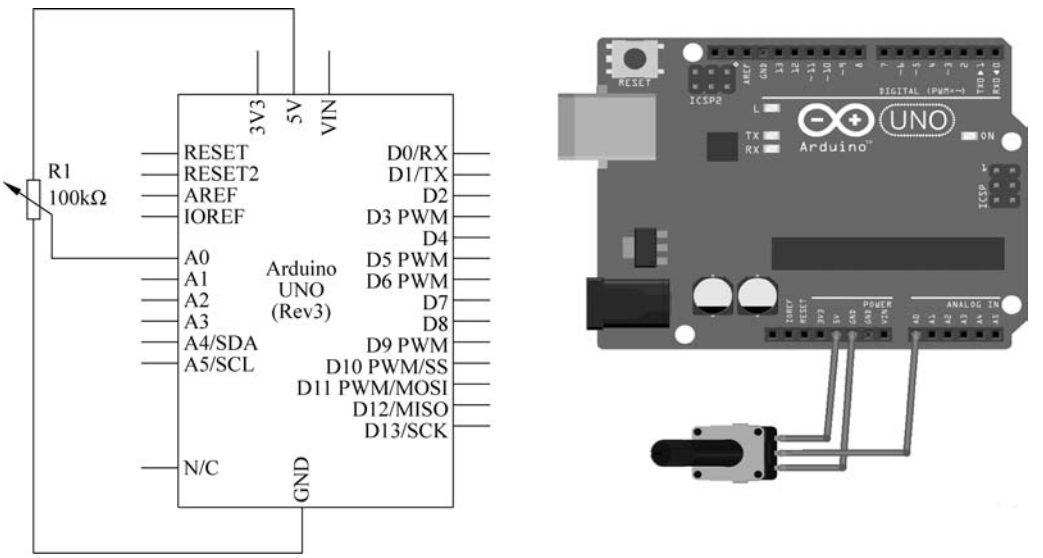


图 5-2 电路图及实物连接图

将电位器中间滑动端接至 Arduino UNO 板的 A0 端，使用 analogRead()语句就可以读出模拟口的值，Arduino UNO 控制器进行的是 10 位的 A/D 采集，所以读取的模拟值范围是 0～1023。

2. 程序说明

首先，在 void setup()中设置波特率。在 Arduino 软件的串口工具监视窗口右下角有一个可以设置波特率的按钮，选中与程序中设置的波特率相同的波特率，Serial. begin()的括号中为波特率的值。

串口监视器中显示的数值来源于 Arduino 与 PC 的通信，因此 Arduino 的波特率应与 PC 软件设置的波特率一致，才能显示正确的数值，否则将会显示乱码或者不显示。

```
/*************************************************************
 * 程序 5-1: 读取模拟引脚上的模拟值并显示
 *************************************************************/
int potpin = 0 ;                          //定义模拟接口 0
int ledpin = 13 ;                         //定义数字接口 13
int val = 0 ;                             //定义变量 val,并赋初值 0
void setup()
{
    //设置数字 13 引脚为输出模式
    pinMode(ledpin,OUTPUT);
    Serial.begin(9600);                   //设置波特率 9600b/s
}
void loop()
{
    digitalWrite(ledpin,HIGH);            //点亮数字接口 13 的 LED
    delay(50);                            //延时 0.05s
    digitalWrite(ledpin,LOW);             //熄灭数字接口 13 的 LED
    delay(50);                            //延时 0.05s
    //读取模拟接口 0 的值,并将其赋给 val
    val = analogRead(potpin);
    Serial.println(val);                  //显示 val 的值
}
```

3. 运行结果

每读取一次值，Arduino 自带的 LED 小灯就会闪烁一下，可以通过旋转来改变读数。通过 Arduino IDE 中的“工具”菜单打开串口监视器，读取模拟量值，如图 5-3 所示。

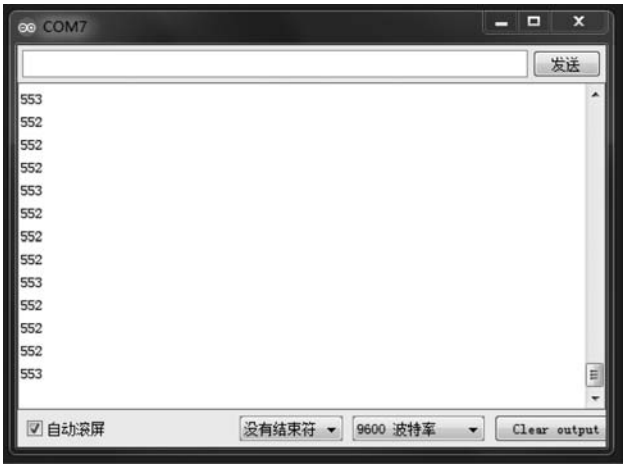


图 5-3　串口监视器的输出

5.2.2 使用 A/D 转换器进行按键输入判别

1. 实验电路原理

电路与实物连接图如图 5-4 和图 5-5 所示。

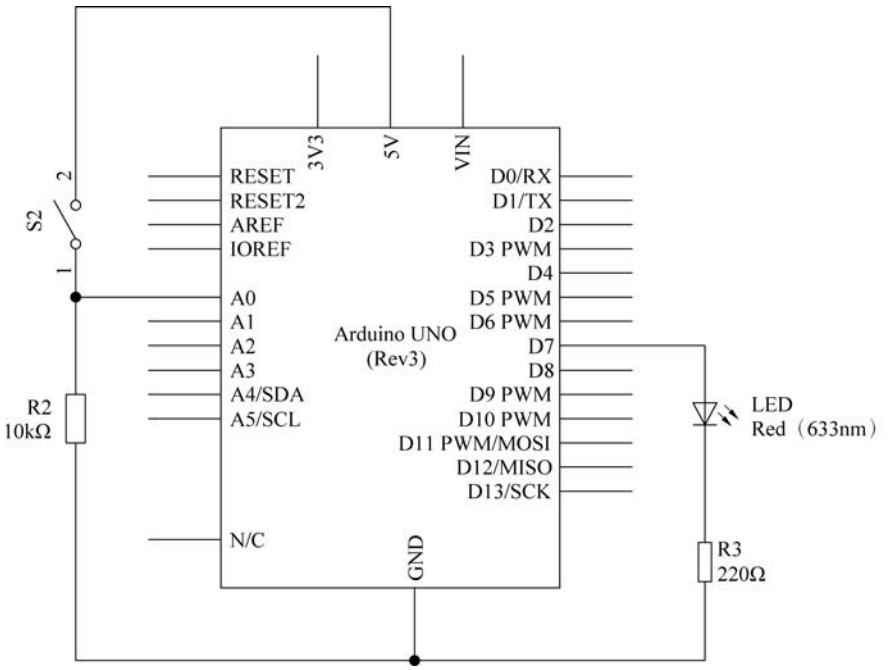


图 5-4 A/D 转换器用作按键判别的电路连接

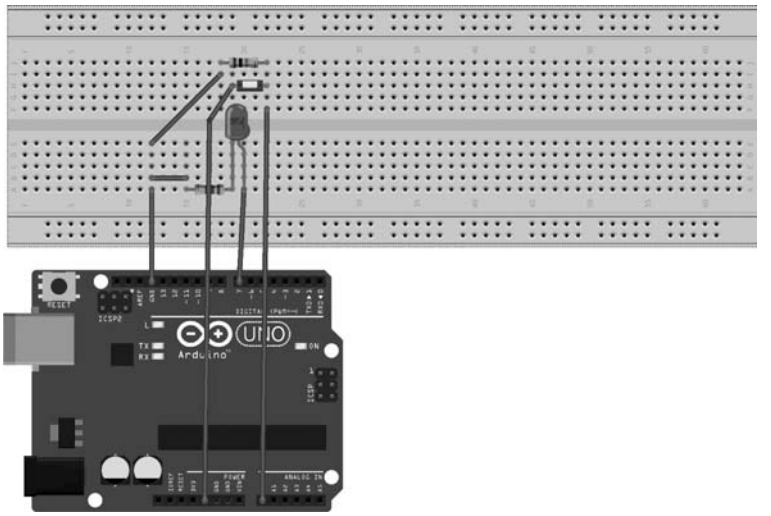


图 5-5 实物连接图

电路连接关系是 Arduino UNO 上的 A0 接按键开关；LED 接 Arduino UNO 上的 DT。

2. 程序说明

使用 A/D 转换器作为按键输入具有一定的抗干扰性。当按键没有被按下时，模拟口电压为 0V，LED 灯熄灭；当按键被按下时，模拟口电压值为 5V，所以只要判断电压值是否大于 2.5V，如果大于 2.5V，则可以知道按键被按下，LED 灯点亮。

```
/*************************************************************
 * 程序 5-2: A/D 转换器用于按键输入判别
 *************************************************************/
int analogPin = 0;
int val = 0;
int led = 7;
void setup()
{
    pinMode(led,OUTPUT);
    pinMode(14,INPUT);
    Serial.begin(9600);
}
void loop()
{
    val = analogRead(analogPin);
    Serial.println(val);
    if(val > 127)
        digitalWrite(led,HIGH);
    else
        digitalWrite(led,LOW);
    delay(200);
}
```

3. 运行结果

将程序下载到 Arduino UNO 开发实验板后，当按下按键时，LED 熄灭；松开按键，LED 灯点亮，串口监视显示如图 5-6 所示，软件施密特触发器实物实现如图 5-7(a)、(b)所示。

5.2.3 使用 A/D 转换器读取键盘值的抢答器

1. 抢答器实验要求

两名选手各有一个按键，比赛开始后进行抢答，谁先按下按键，对应的灯就会亮起来。裁判可根据亮灯情况提示参赛选手答题，本次结束后，裁判按下按键 3 清除亮灯(使亮灯都熄灭)。

这个实验要求使用 analogRead()判定按键是否按下。

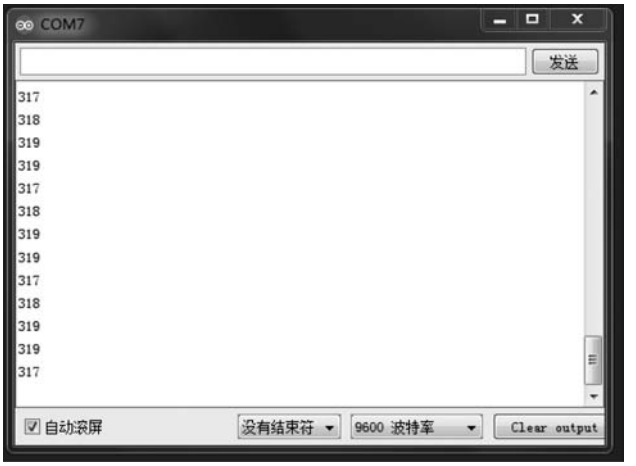
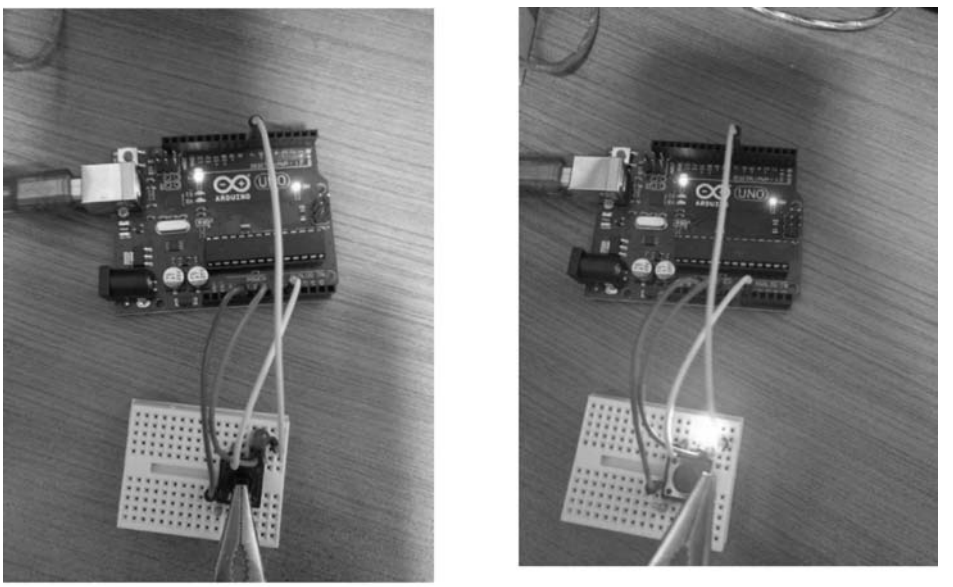


图 5-6　串口监视器的输出



(a) 按下按键　　(b) 松开按键

图 5-7　施密特触发器实物

2. 电路原理

按键 1 —>模拟 2 口(A2)
按键 2 —>模拟 3 口(A3)
按键 3 —>模拟 4 口(A4)
红 LED —>数字第 8 引脚(D8)
绿 LED —>数字第 7 引脚(D7)
蜂鸣器 —>数字第 5 引脚(D5)

电路原理图如图 5-8 所示，实物连接图如图 5-9 所示。

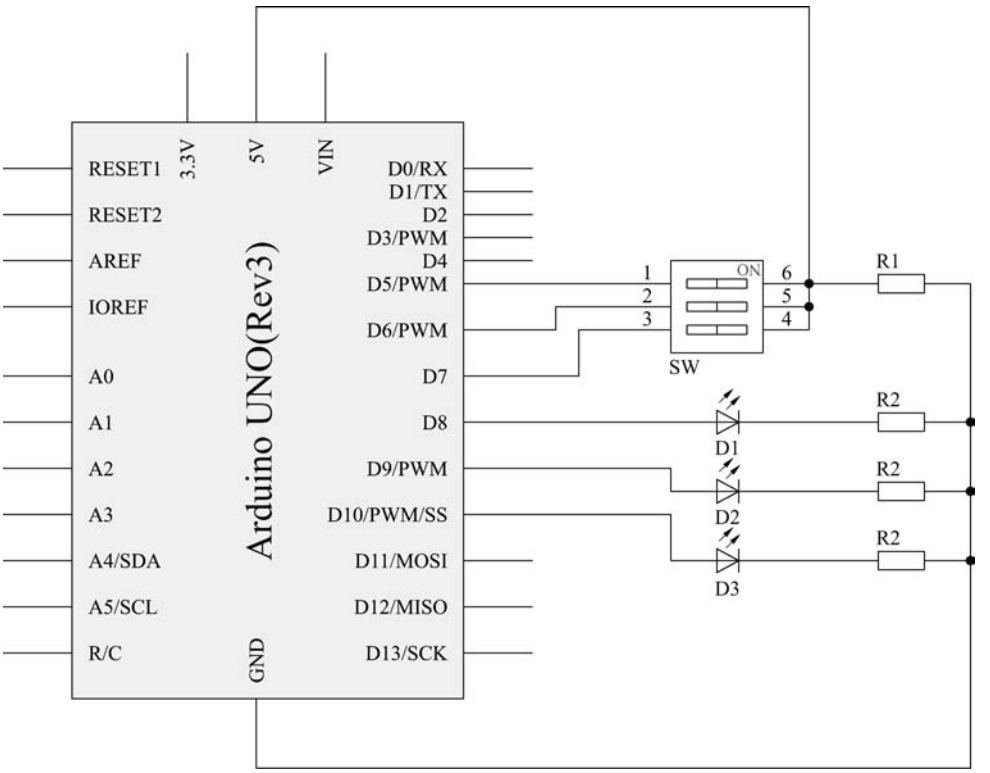


图 5-8　抢答器的电路原理图

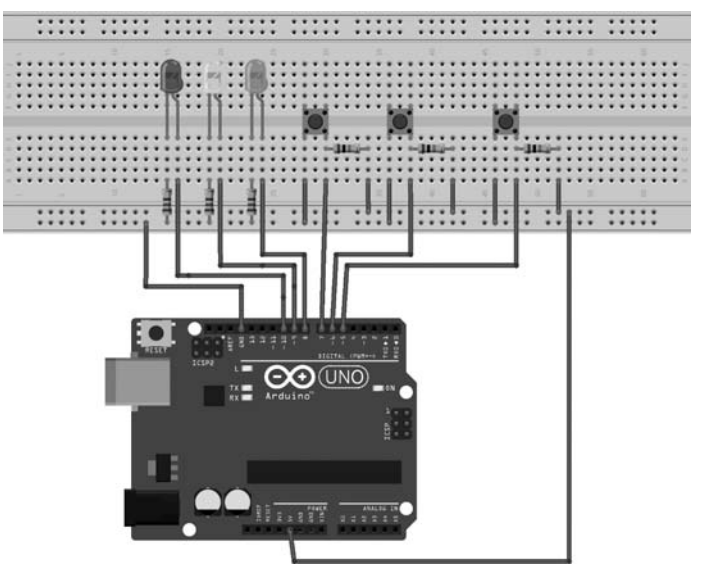


图 5-9　抢答器的实物接线图

3. 程序说明

按键在保持按下和保持弹起这两种状态下,按键引脚电压值是有变化的。因此,我们可以依次读取模拟口 1～模拟口 3 的电压值,根据读出的电压值来判断按键是否被按下。本实验采用高电平有效方式连接。

用万用表测量可知,当没有按键按下时,模拟口电压值为 0.0V 左右。当有按键按下时,模拟口的电压值为 5.0V 左右。所以,我们可以认为当模拟口的电压值小于 1V(数字二进制表示为 204)时,没有按键按下,当模拟口的电压值大于 4V(数字二进制表示为 818)时,有按键按下。

```
/*************************************************************
 * 程序 5-3: 使用 A/D 读取键盘值的抢答器
 *************************************************************/
int RedLed = 8;                    //定义第 8 引脚连接红灯
int GreenLed = 7;                  //定义第 7 引脚连接绿灯
int i;                             //定义变量 i
int j = 0;                         //定义变量 j
void buzzer()                      //蜂鸣器发出"嘀"声音子程序
{
    for(i = 0; i < 80; i++)
    {
        digitalWrite(5,HIGH);      //发声音
        delay(1);                  //延时 1ms
        digitalWrite(5,LOW);       //不发声音
        delay(1);                  //延时 ms
    }
}
void key_scan()                    //按键扫描子程序
{
    int key_1,key_2,key_3;         //定义变量
    key_1 = analogRead(2);         //读模拟引脚 1 电压值
    key_2 = analogRead(3);         //读模拟引脚 2 电压值
    key_3 = analogRead(4);         //读模拟引脚 3 电压值
    //Serial.println(key_1,key_2,key_3);
    Serial.println(key_1);
    Serial.println(key_2);
    Serial.println(key_3);
    /* 如果各按键电压值都小于 204(即模拟值 1V),判断没有按键按下 */
    if(key_1 < 204&&key_2 < 204&&key_3 < 204)
    {
        return;                    //跳出本子函数
    }
    /* 如果按键 1 电压值都大于 818(即模拟值 4V),判断按键 1 被按下 */
    if(key_1 > 818)
    {
```

```
36          delay(10);                          //由于有抖动,所以延时 100ms 再一次判断
37          /* 如果按键 1 电压值都大于 818(即模拟值 4V),判断按键 1 确实被按下 */
38          if(key_1 > 818)
39          {
40              buzzer();                       //蜂鸣器发出声音
41              digitalWrite(RedLed,HIGH);      //红灯亮
42              digitalWrite(GreenLed,LOW);     //绿灯灭
43          }
44          else                                //否则认为是抖动干扰,不做任何动作
45          {
46                return;                       //跳出本子函数
47          }
48      }
49  /* 如果按键 2 电压值都大于 818(即模拟值 4V),判断按键 2 被按下 */
50      if(key_2 > 818)
51      {
52          delay(10);                          //存在抖动,延时 100ms 再一次判断
53          /* 如果按键 2 电压值都大于 818(即模拟值 4V),判断按键 2 确实被按下 */
54          if(key_2 > 818)
55          {
56              buzzer();                       //蜂鸣器发出声音
57              digitalWrite(RedLed,LOW);       //红灯灭
58              digitalWrite(GreenLed,HIGH);    //绿灯亮
59          }
60          else                                //否则认为是抖动干扰,不做任何动作
61          {
62              return;                         //跳出本子函数
63          }
64      }
65      /* 如果按键 3 电压值都大于 818(即模拟值 4V),则可以判断按键 3 被按下 */
66      if(key_3 > 818)
67      {
68          delay(10);                          //存在抖动,延时 100ms 再一次判断
69          /* 如果按键 3 电压值都大于 818(即模拟值 4V),判断按键 3 确实被按下 */
70          if(key_3 > 818)
71          {
72              buzzer();                       //蜂鸣器发出声音
73              digitalWrite(RedLed,LOW);       //红灯灭
74              digitalWrite(GreenLed,LOW);     //绿灯灭
75          }
76          else                                //否则认为是抖动干扰,不做任何动作
77          {
78              return;                         //跳出本子函数
79          }
80      }
81  }
```

```
82  void setup()
83  {
84      Serial.begin(9600);
85      for(i=5; i<=8; i++)
86      {
87          pinMode(i,OUTPUT);                    //引脚 5～8 设置为输出模式
88      }
89  }
90  void loop()
91  {
92      key_scan();                               //循环扫描按键
93  }
```

4. 运行结果

按键 1 和按键 2 是抢答按键,按键 3 是清除按键。如果按键 1 先被按下,则蜂鸣器发出提示音,红灯亮,绿灯灭;如果按键 2 先被按下,则蜂鸣器发出提示音,绿灯亮,红灯灭;如果按键 3 被按下,则蜂鸣器发出提示音,将红灯和绿灯都熄灭。实物效果如图 5-10 所示。

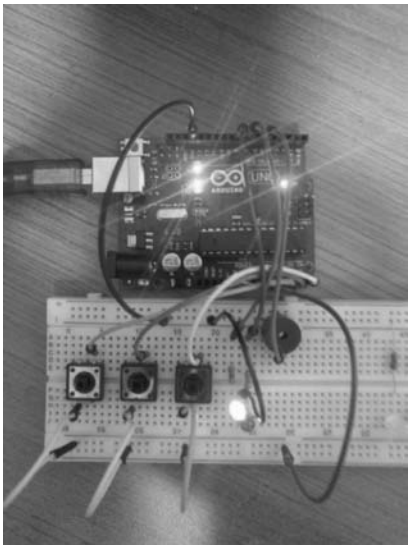


图 5-10　抢答器的实现

需要注意的是,按键可能存在抖动干扰,为了更加准确地判断是否有按键被按下,在第一次判断有按键按下之后,延时 10ms 躲避抖动,然后进行第二次判断。另外,从模拟口读出的电压值是用二进制表示的。