第3章

I/O的高级应用

掌握数字 I/O 和模拟 I/O 的基本操作方法后，就可以完成很多 Arduino 制作了。除此之外，Arduino 还提供了一些 I/O 口的高级操作。

3.1 调声函数

调声函数 tone()主要用于 Arduino 连接蜂鸣器或扬声器发声，其实质是输出一个频率可调的方波，以此驱动蜂鸣器或扬声器振动发声。

tone()

可以让指定引脚产生一个占空比为 50%的指定频率的方波。

语法

```
tone(pin, frequency)
tone(pin, frequency, duration)
```

参数

pin：需要输出方波的引脚。

frequency：输出的频率，unsigned int 型。

duration：方波持续的时间，单位为毫秒。如果没有该参数，Arduino 将持续发出设定的音调，直到改变发声频率或者使用 noTone()函数停止发声。

返回值

无。

tone()和 analogWrite()函数都可以输出方波，不同的是 tone()函数输出方波的占空比固定(50%)，调节的是方波的频率；而 analogWrite()函数输出的频率固定(约 490 Hz)，调节的是方波的占空比。

需要注意的是，同一时间 tone()函数仅能作用一个引脚，如果有多个引脚需要使用 tone()函数，那必须先使用 noTone()函数停止之前已经使用了 tone()函数的引脚，再使用 tone()函数开启下一个指定引脚的方波输出。

noTone()

停止指定引脚上的方波输出。

语法

```
noTone(pin)
```

参数

pin：需要停止方波输出的引脚。

返回值

无。

下面将使用 tone()函数驱动蜂鸣器播放曲子。

3.1.1　蜂鸣器发声

无源蜂鸣器模块(图 3-1)是一种一体化结构的电子讯响器，采用直流电压供电，广泛应用于计算机、报警器、电子玩具等电子设备中。

无源蜂鸣器发声需要有外部振荡源，即一定频率的方波。不同频率的方波输入，会产生不同的音调。接下来我们要利用这种特性，用 tone()函数输出不同的频率的方波，实现 Arduino 播放简单的曲子。

如果使用的是蜂鸣器模块，则直接连接到扩展板即可；如果使用的是独立的扬声器或者蜂鸣器，可按图 3-2 所示方式连接。

在示例程序中使用了两个数组 melody[]和 noteDurations[]记录整个曲谱，然后遍历这两个数组实现输出曲子的功能。

可以在 Arduino IDE 菜单“文件→示例→02. Digital→toneMelody”打开以下程序：

图 3-1　蜂鸣器模块

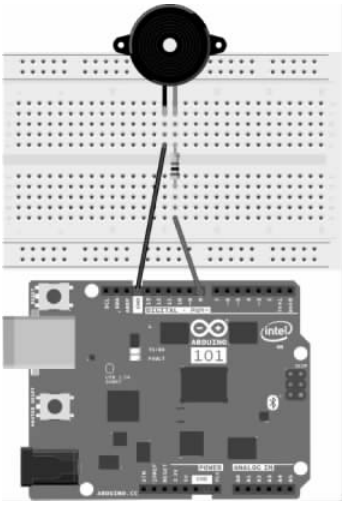

图 3-2　蜂鸣器模块使用连接示意图

```
/*
melody
Plays a melody
This example code is in the public domain.
http://arduino.cc/en/Tutorial/Tone
*/

#include "pitches.h"

//记录曲子的音符
int melody[] = {
  NOTE_C4, NOTE_G3,NOTE_G3, NOTE_A3, NOTE_G3,0, NOTE_B3, NOTE_C4};

//音符持续时间 4 为四分音符, 8 为八分音符
int noteDurations[] = {
  4, 8, 8, 4,4,4,4,4 };

void setup() {
  //遍历整个曲子的音符
  for (int thisNote = 0; thisNote < 8; thisNote++) {

//noteDurations[]数组中存储的是音符的类型
//需要将其换算为音符持续时间,方法如下
    //音符持续时间 = 1 000ms / 音符类型
    //例如,四分音符 = 1000/4,八分音符 = 1000/8
    int noteDuration = 1000/noteDurations[thisNote];
```

```
        tone(8, melody[thisNote],noteDuration);

        //为了能辨别出不同的音调,需要在两个音调间设置一定的延时
        //增加 30 % 延时时间是比较合适的
        int pauseBetweenNotes = noteDuration * 1.30;
        delay(pauseBetweenNotes);
        //停止发声
        noTone(8);
      }
    }

    void loop() {
        //程序并不重复,因此这里为空
    }
```

使用以上程序驱动蜂鸣器,还需要一个定义了音调对应频率的头文件 pitches.h,其中记录了不同频率的对应的音调,在 toneMelody 中即是调用了这些定义。如果是通过示例程序打开的该程序,则会在选项卡中看到这个头文件(图 3-3)。

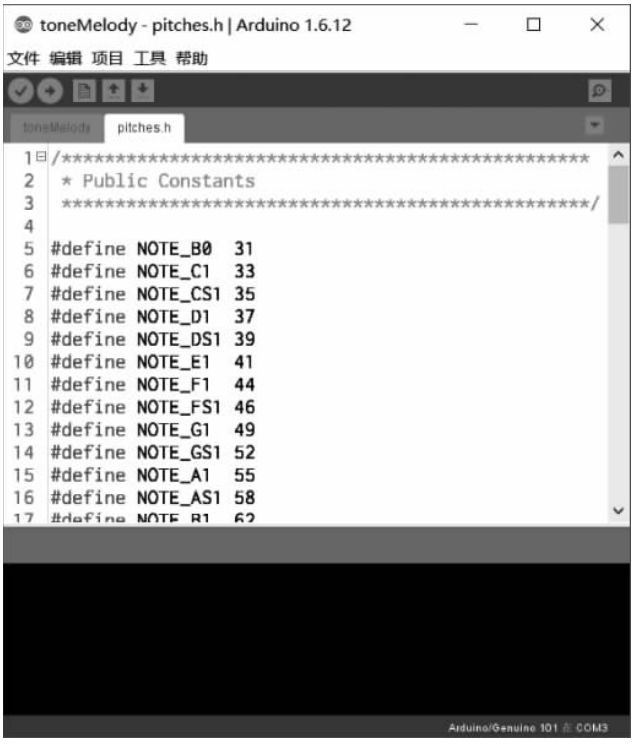

图 3-3　pitches.h 文件

如果是新建的相关程序，要调用这些音调定义，则需要先建立一个名为 pitches. h 的头文件。如图 3-4 所示，在 IDE 中，单击串口监视器下方的小三角，选择新建标签，并在下方的输入框中键入新文件名 pitches. h，并单击“好”按钮。

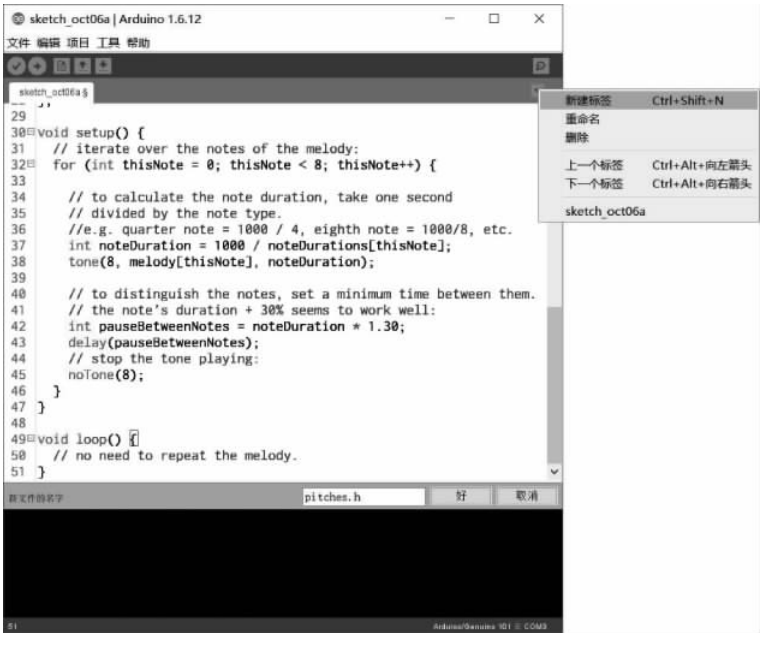

图 3-4　向项目中添加新文件

单击“好”按钮后，IDE 会在项目文件夹中新建一个名为 pitches. h 的文件，并打开这个文件。然后需要将 pitches. h 的内容写入该文件，可以在 Arduino 安装目录中找到 pitches. h 这个文件（Arduino 安装目录\examples\02. Digital\toneMelody\ pitches. h）。

单击“上传”按钮，在检查无误后，编译器即会编译主程序及 pitches. h 文件。上传成功后，即可听到蜂鸣器或扬声器发出的声音了。

还可以试着使用按键和其他的一些传感器结合蜂鸣器制作一个 Arduino 电子琴，按下不同的按键或者触发不同的传感器，让蜂鸣器发出各种不同的音调。

3.1.2　简易电子琴

通过无源蜂鸣器和按键的组合可以制作一个简易的电子琴。当按下不同的按键时，蜂鸣器就会发出不同的音调，达到模拟琴键的效果。

如图 3-5 所示项目中，使用了 7 个按键分别连接到 7 个引脚，并给每个引脚加上了 10kΩ 下拉电阻以稳定引脚上的电平。Arduino 通过依次检查各按键的状态来控制 10 号引

脚输出方波，驱动蜂鸣器发出各种不同的音调。

1. 所需材料

Genuino 101、无源蜂鸣器、按键 7 个、连接线若干。

2. 连接示意图

简易电子琴项目的连接示意图如图 3-5 所示。

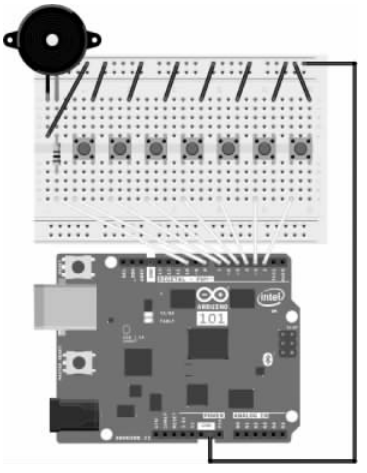

图 3-5　简易电子琴项目连接示意图

示例程序代码如下：

```
/*
使用蜂鸣器与按键制作简易电子琴
*/
#include "pitches.h"

void setup() {
  //初始化按键模块连接引脚
  pinMode(2,INPUT);
  pinMode(3,INPUT);
  pinMode(4,INPUT);
  pinMode(5,INPUT);
  pinMode(6,INPUT);
  pinMode(7,INPUT);
  pinMode(8,INPUT);
}
```

```
void loop() {
  //依次读出各按键模块状态
  //如果按键被按下,则发出相应的音色
  if(digitalRead(2)){
    tone(10, NOTE_C5,10);                    //Do(523Hz)
  }
  if(digitalRead(3)){
    tone(10, NOTE_D5,10);                    //Re (587Hz),
  }
  if(digitalRead(4)){
    tone(10, NOTE_E5, 10);                   //Mi(659Hz)
  }
  if(digitalRead(5)){
    tone(10, NOTE_F5, 10);                   //Fa(698Hz)
  }
  if(digitalRead(6)){
    tone(10, NOTE_G5, 10);                   //So(784Hz)
  }
  if(digitalRead(7)){
    tone(10, NOTE_A5, 10);                   //La(880Hz)
  }
  if(digitalRead(8)){
    tone(10, NOTE_B5, 10);                   //Si(988Hz)
  }
}
```

编译并上传程序后,即可以用这个自制的电子琴演奏一个简单的曲子了。还可以将按键换成触摸模块或者其他数字传感器以获得不同的体验效果。

3.2 脉冲宽度测量函数

Arduino 提供了 pulseIn()函数,用于检测引脚上脉冲信号的宽度。

pulseIn()

检测指定引脚上的脉冲信号宽度。

例如,当要检测高电平脉冲时,pulseIn()函数会等待指定引脚输入的电平变高,当变高后开始记时,直到输入电平变低,停止计时。pulseln()函数会返回这个脉冲信号持续的时间,即这个脉冲的宽度。

函数还可以设定超时时间。如果超过设定时间仍未检测到脉冲,则会退出 pulseIn()函

数并返回 0。当没有设定超时时间时，pulseIn()函数会默认 1s 的超时时间。

语法

```
pulseIn(pin, value)
pulseIn(pin, value, timeout)
```

参数

pin：需要读取脉冲的引脚。

value：需要读取的脉冲类型，HIGH 或 LOW。

timeout：超时时间，单位为微秒(μs)，数据类型为无符号长整型。

返回值

返回脉冲宽度，单位为微秒(μs)，数据类型为无符号长整型。如果在指定时间内没有检测到脉冲，则返回 0。

接下来将学习利用 pulseIn()函数与超声波传感器完成测距工作。

超声波是频率高于 20 000Hz 的声波，它指向性强，能量消耗缓慢，在介质中传播的距离较远，因而经常用于测量距离。超声波传感器型号众多，这里为大家介绍一款常见的超声波传感器。

1. SR04 超声波传感器

SR04(图 3-6)是利用超声波特性检测距离的传感器。其带有两个超声波探头，分别用作发射和接收超声波。其测量范围为 3～450cm。

2. 工作原理

如图 3-7 所示，超声波发射器向某一方向发射超声波，在发射的同时开始计时，超声波在空气中传播，途中碰到障碍物就立即返回来，超声波接收器收到反射波就立即停止计时。声波在空气中的传播速度约为 340m/s，根据计时器记录的时间 t，就可以计算出发射点距障碍物的距离 s，即：$s=340\text{m/s}\times t/2$。这就是所谓的时间差测距法。

图 3-6　SR04 超声波传感器

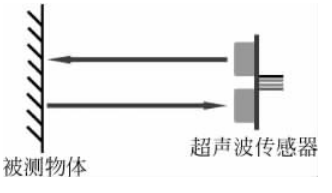

图 3-7　超声波发射接收示意图

3. 引脚

SR04 超声波模块有 4 个引脚，各功能见表 3-1。

表 3-1 SR04 引脚

引　脚	说　明
Vcc	电源 5V
Trig	触发引脚
Echo	回馈引脚
Gnd	接地引脚

4. 驱动方法

如图 3-8 所示，使用 Arduino 的数字引脚给 SR04 的 Trig 引脚至少 10μs 的高电平信号，触发 SR04 模块测距功能。

图 3-8 Arduino 发送触发信号

如图 3-9 所示，触发后，模块会自动发送 8 个 40kHz 的超声波脉冲，并自动检测是否有信号返回。这步会由模块内部自动完成。

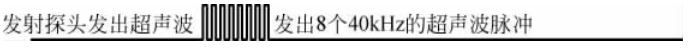

图 3-9 超声波发出超声波脉冲

如图 3-10 所示，如有信号返回，Echo 引脚会输出高电平，高电平持续的时间就是超声波从发射到返回的时间。此时能使用 pulseIn()函数获取测距的结果，并计算出距被测物的实际距离。

图 3-10 超声波返回测距结果

5. 连接示意图

如图 3-11 所示，本示例将超声波模块的 Trig 引脚连接到 Arduino 的 2 号引脚，Echo 引脚连接到 Arduino 的 3 号引脚。

示例程序代码如下：

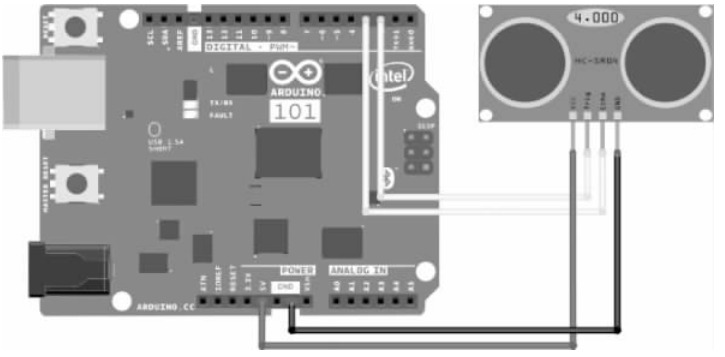

图 3-11　超声波测距连接示意图

```
/*
SR04 超声波传感器驱动
串口显示检测距离
*/

//设定 SR04 连接的 Arduino 引脚
const int TrigPin = 2;
const int EchoPin = 3;
float distance;

void setup()
{   //初始化串口通信及连接 SR04 的引脚
    Serial.begin(9600);
    pinMode(TrigPin, OUTPUT);
    //要检测引脚上输入的脉冲宽度,需要先设置为输入状态
    pinMode(EchoPin, INPUT);
    Serial.println("Ultrasonic sensor:");
}

void loop()
{
    //产生一个 10μs 的高脉冲去触发 TrigPin
    digitalWrite(TrigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(TrigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(TrigPin, LOW);
    //检测脉冲宽度,并计算出距离
```

```
    distance = pulseIn(EchoPin, HIGH)/ 58.00;
    Serial.print(distance);
    Serial.print("cm");
    Serial.println();
    delay(1000);
}
```

编译并上传程序，然后打开串口监视器，并将超声波传感器对向需要测量的物体，即可看到当前超声波传感器距物体的距离，如图 3-12 所示。

图 3-12　超声波测距结果

环境温度、湿度等对声波的传输速度也有影响，可以尝试结合其他的温/湿度传感器校正超声波传感器测出的数据，以得到更准确的测量结果。

3.3　外部中断

程序运行中时常需要监控一些事件的发生，如对某一传感器的检测结果做出反应。使用轮询的方式检测，效率较低，等待时间较长；而使用中断方式检测，可以到达实时检测的效果。

如图 3-13 所示，中断程序可以看作是一段独立于主程序之外的程序，当中断触发时，控制器会暂停当前正在运行的主程序，而跳转去运行中断程序，中断程序运行完后，会再回到之前主程序暂停的位置，继续运行主程序。如此即可达到实时响应处理事件的效果。

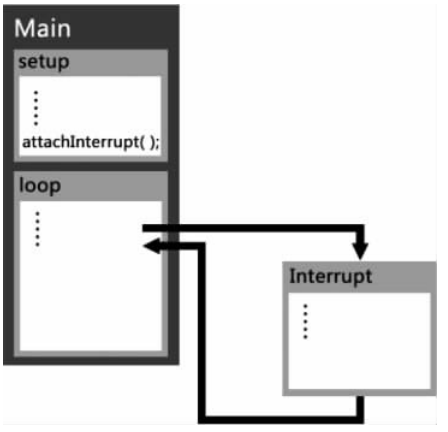

图 3-13　中断结构

3.3.1　外部中断的使用

外部中断是由外部设备发起请求的中断。要使用外部中断，需要了解中断引脚的位置，根据外部设备选择中断模式，编写一个中断触发后执行的中断函数。

1. 中断引脚与中断编号

在不同型号的 Arduino 控制器上，中断引脚的位置也不同，只有中断信号发生在带外部中断功能的引脚上，Arduino 才能捕获这个中断信号并做出响应。

在 Genuino 101 上的中断使用，和其他 Arduino 控制器有些许不同。其他 Arduino 只有部分引脚支持外部中断，且中断编号和引脚号是不一样的；而 Genuino 101 所有引脚都支持外部中断，中断编号即是引脚编号。

2. 中断模式

还需要了解设备触发外部中断的输入信号类型，以设置中断模式。中断模式也就是中断触发的方式，大多数 Arduino 支持表 3-2 中的四种中断触发方式。

表 3-2　可用中断触发模式

模　式	说　明
LOW	低电平触发
CHANGE	电平变化触发，高电平变低电平，低电平变高电平
RISING	上升沿触发，低电平变高电平
FALLING	下降沿触发，高电平变低电平

需要注意的是，Genuino 101 上只有 2，5，7，8，10，11，12，13 引脚支持 CHANGE 中断模式。

3. 中断函数

除了设置中断模式外，还需要编写一个响应中断的处理程序——中断函数，当中断触发后，Arduino 即会运行这个函数。该函数不能带任何参数，且返回类型为空，如：

```
void Hello()
{
    flag = true;
}
```

当中断触发后，Arduino 即会执行这个函数中的语句。

这些准备工作做好后，还需要在 Setup()中使用 attachInterrupt()函数对中断引脚进行初始化配置，以开启 Arduino 的外部中断功能，其方法如下。

(1) attachInterrupt(pin, ISR, mode)

参数

pin：中断引脚；

ISR：中断函数名；

mode：中断模式。

例如：

```
attachInterrupt(2, Hello,LOW);
```

该语句会开启 Genuino 101 的 2 号引脚(中断编号 0)的外部中断功能，并指定下降沿时触发该中断。当 2 号引脚上电平由高变低后，该中断会被触发，Arduino 即会运行 Hello()函数中的语句。

如果不需要使用外部中断了，可以用中断分离函数 detachInterrupt()来关闭中断功能。

(2) detachInterrupt(pin)

禁用外部中断。

参数 pin：需要禁用中断的引脚。

3.3.2 外部中断触发蜂鸣器警报实验

这里制作一个防盗报警装置。装置放在需要看守的物体旁边，通过数字红外障碍传感器检测前方是否有物体，如果没有检测到物体就触发蜂鸣器报警。

数字红外障碍传感器(图 3-14)是一种通过红外光反射来检测障碍物的传感器。

图 3-14　数字红外障碍传感器

模块会发出调制过的 38kHz 红外光,红外光经障碍物反射后由一体化接收头接收。当检测范围内有障碍物时,模块输出低电平;无障碍物时,模块输出高电平。

在编写这个中断程序前,先要清楚适合项目的中断触发方式,这里将红外障碍传感器连接到 Genuino 101 的 2 号引脚上,并将其设为电平改变触发。当电平由低变高时,说明物体被拿走,触发 8 号引脚连接的蜂鸣器报警;当放回物体后,电平由高变低,蜂鸣器停止报警。

示例程序代码如下:

```
/*
Arduino 外部中断的使用
外部中断触发警报声
*/

//默认无遮挡,蜂鸣器发声
volatile boolean RunBuzzer = true;

void setup()
{
  Serial.begin(9600);
  //初始化外部中断
  //当 2 号引脚输入的电平由高变低时,触发中断函数 warning
  attachInterrupt(2, warning, CHANGE);
}

void loop()
{
  if(RunBuzzer)
  {
    tone(8,1000);
  }
  else
  {
    noTone(8);
  }
}

//中断函数
```

```
void warning ()
{
  RunBuzzer = ! RunBuzzer;
}
```

上传该程序后，会听到蜂鸣器发出警报声，用手或其他物体遮挡住红外障碍传感器，警报声便会停止。

要注意的是，需要在中断函数中改变的变量，要使用 volatile 定义避免编译器优化造成程序运行异常。