# 蜂鸣器播放音乐

蜂鸣器是一种电子发声元器件，可以发出"beep"的声音。采用直流电压供电，广泛应用于计算机、打印机、复印机、报警器、电子玩具、汽车电子设备、电话机、定时器等电子产品中作发声器件。

蜂鸣器分为有源蜂鸣器和无源蜂鸣器两种。

有源蜂鸣器：内部带有震荡源，只要一通电就会发出固定频率的声音。



无源蜂鸣器：内部不带震荡源，需要使用2 kHz到5 kHz的脉冲信号驱动发声，声音频率可变。



从外观上来看两种蜂鸣器形状相似，有源蜂鸣器底部有密封，无源蜂鸣器底部可看到电路板。两种蜂鸣器都标注有正负极。

## 连接图

蜂鸣器正极接开发板数字引脚8，负极接开发板GND，连接非常简单。



## 程序

所用函数：

**tone()**

* 描述：在指定引脚产生指定频率的方波，可用于蜂鸣器或扬声器播放音调。
* 函数原型：
	+ tone(pin, frequency)
	+ tone(pin, frequency, duration)
* 参数：
	+ pin：产生音调的引脚。
	+ 音调频率，单位赫兹。
	+ 音调持续时间(毫秒单位)，此参数可选。
* 返回值：没有

**noTone()**

* 描述：停止产生方波。noTone(pin)用来停止由tone(pin, frequency)产生的音调。
* 函数原型：noTone(pin)
* 参数：
	+ pin：需要停止产生音调的引脚。
* 返回值：没有。

下面的代码可以播放《天空之城》：

#define NOTE\_D0 -1

#define NOTE\_D1 294

#define NOTE\_D2 330

#define NOTE\_D3 350

#define NOTE\_D4 393

#define NOTE\_D5 441

#define NOTE\_D6 495

#define NOTE\_D7 556

#define NOTE\_DL1 147

#define NOTE\_DL2 165

#define NOTE\_DL3 175

#define NOTE\_DL4 196

#define NOTE\_DL5 221

#define NOTE\_DL6 248

#define NOTE\_DL7 278

#define NOTE\_DH1 589

#define NOTE\_DH2 661

#define NOTE\_DH3 700

#define NOTE\_DH4 786

#define NOTE\_DH5 882

#define NOTE\_DH6 990

#define NOTE\_DH7 112

//以上部分是定义是把每个音符和频率值对应起来，其实不用打这么多，但是都打上了，后面可以随意编写

#define WHOLE 1

#define HALF 0.5

#define QUARTER 0.25

#define EIGHTH 0.25

#define SIXTEENTH 0.625

//这部分是用英文对应了拍子，这样后面也比较好看

int tune[] =

{

 NOTE\_D0, NOTE\_D0, NOTE\_D0, NOTE\_D6, NOTE\_D7, NOTE\_DH1, NOTE\_D7, NOTE\_DH1, NOTE\_DH3, NOTE\_D7, NOTE\_D7, NOTE\_D7, NOTE\_D3, NOTE\_D3,

 NOTE\_D6, NOTE\_D5, NOTE\_D6, NOTE\_DH1, NOTE\_D5, NOTE\_D5, NOTE\_D5, NOTE\_D3, NOTE\_D4, NOTE\_D3, NOTE\_D4, NOTE\_DH1,

 NOTE\_D3, NOTE\_D3, NOTE\_D0, NOTE\_DH1, NOTE\_DH1, NOTE\_DH1, NOTE\_D7, NOTE\_D4, NOTE\_D4, NOTE\_D7, NOTE\_D7, NOTE\_D7, NOTE\_D0, NOTE\_D6, NOTE\_D7,

 NOTE\_DH1, NOTE\_D7, NOTE\_DH1, NOTE\_DH3, NOTE\_D7, NOTE\_D7, NOTE\_D7, NOTE\_D3, NOTE\_D3, NOTE\_D6, NOTE\_D5, NOTE\_D6, NOTE\_DH1,

 NOTE\_D5, NOTE\_D5, NOTE\_D5, NOTE\_D2, NOTE\_D3, NOTE\_D4, NOTE\_DH1, NOTE\_D7, NOTE\_D7, NOTE\_DH1, NOTE\_DH1, NOTE\_DH2, NOTE\_DH2, NOTE\_DH3, NOTE\_DH1, NOTE\_DH1, NOTE\_DH1,

 NOTE\_DH1, NOTE\_D7, NOTE\_D6, NOTE\_D6, NOTE\_D7, NOTE\_D5, NOTE\_D6, NOTE\_D6, NOTE\_D6, NOTE\_DH1, NOTE\_DH2, NOTE\_DH3, NOTE\_DH2, NOTE\_DH3, NOTE\_DH5,

 NOTE\_DH2, NOTE\_DH2, NOTE\_DH2, NOTE\_D5, NOTE\_D5, NOTE\_DH1, NOTE\_D7, NOTE\_DH1, NOTE\_DH3, NOTE\_DH3, NOTE\_DH3, NOTE\_DH3, NOTE\_DH3,

 NOTE\_D6, NOTE\_D7, NOTE\_DH1, NOTE\_D7, NOTE\_DH2, NOTE\_DH2, NOTE\_DH1, NOTE\_D5, NOTE\_D5, NOTE\_D5, NOTE\_DH4, NOTE\_DH3, NOTE\_DH2, NOTE\_DH1,

 NOTE\_DH3, NOTE\_DH3, NOTE\_DH3, NOTE\_DH3, NOTE\_DH6, NOTE\_DH6, NOTE\_DH5, NOTE\_DH5, NOTE\_DH3, NOTE\_DH2, NOTE\_DH1, NOTE\_DH1, NOTE\_D0, NOTE\_DH1,

 NOTE\_DH2, NOTE\_DH1, NOTE\_DH2, NOTE\_DH2, NOTE\_DH5, NOTE\_DH3, NOTE\_DH3, NOTE\_DH3, NOTE\_DH3, NOTE\_DH6, NOTE\_DH6, NOTE\_DH5, NOTE\_DH5,

 NOTE\_DH3, NOTE\_DH2, NOTE\_DH1, NOTE\_DH1, NOTE\_D0, NOTE\_DH1, NOTE\_DH2, NOTE\_DH1, NOTE\_DH2, NOTE\_DH2, NOTE\_D7, NOTE\_D6, NOTE\_D6, NOTE\_D6, NOTE\_D6, NOTE\_D7

};//这部分就是整首曲子的音符部分，用了一个序列定义为tune，整数

float duration[] =

{

 1, 1, 1, 0.5, 0.5, 1 + 0.5, 0.5, 1, 1, 1, 1, 1, 0.5, 0.5,

 1 + 0.5, 0.5, 1, 1, 1, 1, 1, 1, 1 + 0.5, 0.5, 1, 1,

 1, 1, 0.5, 0.5, 0.5, 0.5, 1 + 0.5, 0.5, 1, 1, 1, 1, 1, 0.5, 0.5,

 1 + 0.5, 0.5, 1, 1, 1, 1, 1, 0.5, 0.5, 1 + 0.5, 0.5, 1, 1,

 1, 1, 1, 0.5, 0.5, 1, 0.5, 0.25, 0.25, 0.25, 0.5, 0.5, 0.5, 0.5, 0.25, 0.5, 1,

 0.5, 0.5, 0.5, 0.5, 1, 1, 1, 1, 1, 0.5, 0.5, 1 + 0.5, 0.5, 1, 1,

 1, 1, 1, 0.5, 0.5, 1.5, 0.5, 1, 1, 1, 1, 1, 1,

 0.5, 0.5, 1, 1, 0.5, 0.5, 1.5, 0.25, 0.5, 1, 1, 1, 1, 1,

 1, 1, 1, 1, 1, 1, 1, 1, 0.5, 0.5, 1, 1, 0.5, 0.5,

 1, 0.5, 0.5, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

 0.5, 0.5, 1, 1, 0.5, 0.5, 1, 0.5, 0.25, 0.5, 1, 1, 1, 1, 0.5, 0.5

};//这部分是整首曲子的节拍部分，也定义个序列duration，浮点（数组的个数和前面音符的个数是一样的，一一对应么）

int length;//这里定义一个变量，后面用来表示共有多少个音符

int tonePin = 5; //蜂鸣器的pin

void setup()

{

 pinMode(tonePin, OUTPUT); //设置蜂鸣器的pin为输出模式

 length = sizeof(tune) / sizeof(tune[0]); //这里用了一个sizeof函数， 可以查出tone序列里有多少个音符

}

void loop()

{

 for (int x = 0; x < length; x++) //循环音符的次数

 {

 tone(tonePin, tune[x]); //此函数依次播放tune序列里的数组，即每个音符

 delay(400 \* duration[x]); //每个音符持续的时间，即节拍duration，是调整时间的越大，曲子速度越慢，越小曲子速度越快，自己掌握吧

 noTone(tonePin);//停止当前音符，进入下一音符

 }

 delay(5000);//等待5秒后，循环重新开始

}