Buzzer

Introduction

Some appliances often buzz when in an electric state , this is actually from a buzzer, and the annoying bell at school is but a larger buzzer. There are two kinds of buzzers. One is active buzzer, the other is passive buzzer. “active” and “passive” don’t mean the common power source, but a buzzer with or without internal oscillators. Active buzzer will buzz as long as you electricity it, but the frequency is fixed. Passive buzzer, buzzer without internal oscillators, will not buzz when electrified internal oscillators, it requires 2~5 kHz square wave to actuate, then wave forms in different frequency can buzz with corresponding sound.

Active buzzer Passive buzzer

Experiment Purpose

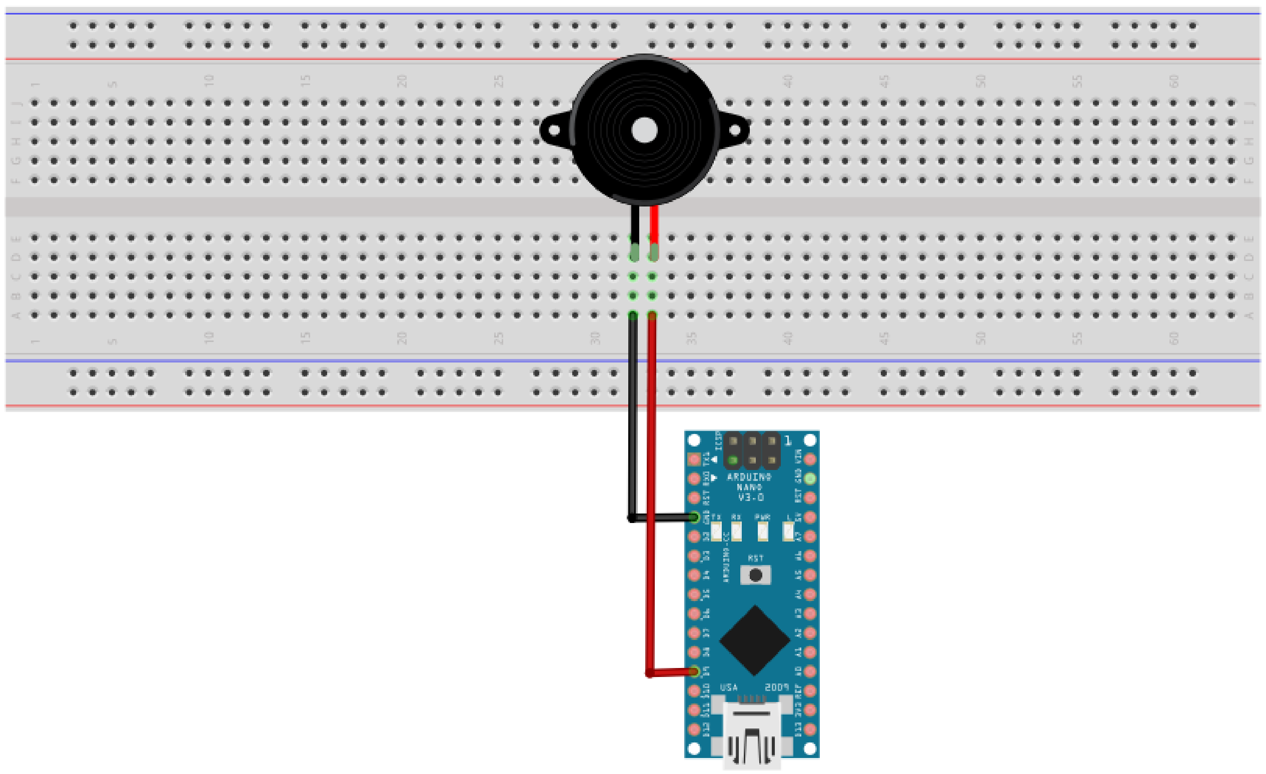
Arduino can be used to create a large number of interactive works, the most common and most common being sound and light displays. We used LEDs in our experiments before. Now we use a buzzer to play the sound. As long as the frequency matches the score, we can hear wonderful music.

Component List

- Arduino Nano Mainboard
- Breadboard
- USB cable
- Active buzzer*1
- Several jumper wires

Wiring of Circuit

Arduino Nano	buzzer
8	+
GND	-



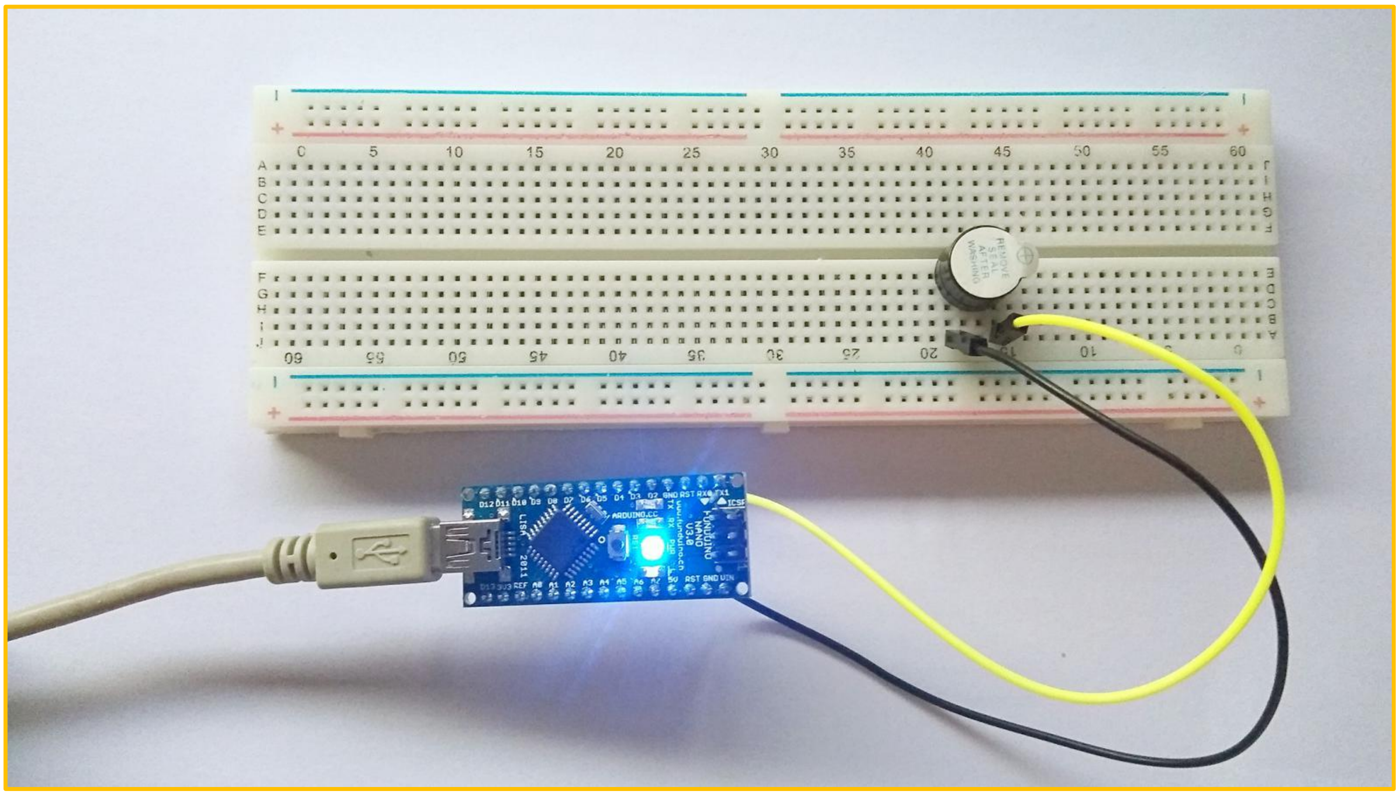
Notice that a buzzer has both a cathode and an anode. We can see the buzzer with two kinds of wiring, red and black, in the right physical diagram below. The connection of circuit and programming are quite simple, the program is similar to the former. Due to the control interface in the buzzer is also digital interface, high and low level from output will control the sound of the buzzer.

Code

```
int buzzer = 8;     // set buzzer out pin
void setup()
{
   pinMode(buzzer,OUTPUT);
}
void loop()
{
   digitalWrite(buzzer,HIGH);
   delay(1000);
   digitalWrite(buzzer,LOW);
   delay(1000);
}
```

Once the program is downloaded, we can hear the buzzer.

Experiment Result



Passive Buzzer working principle

The passive buzzer generates music mainly through the I/O port of the single-chip microcomputer to output different pulse signals of different levels to control the buzzer pronunciation.

For example， if Arduino use12MHzs crystal oscillator, to produce middle tune “Re” sound, it needs 587Hzs audio pulse frequency output .The audio signal pulse cycle is T=1/587=1703.5775us, half cycle time is 852us,the pulse timer needs always count at =852us/1us=852,when it count at 852,the I/O port will reverse the direction, then it get the “Re” sounds in C major scale.

Experiment 1:Alarm Test

Experimental purposes:

Make the buzzer simulate the alarm sound

Experimental principle:

The sound starts with the frequency increasing from 200HZ to 800HZ, then stops for a period of time from 800HZ to 200HZ, and loops Experimental Code Location

“Lesson7-Active&Passive Buzzer\ AlarmSound\AlarmSound.ino”

```
void setup()
{
    pinMode(8,OUTPUT);
}
void loop()
{
    for(int i = 200; i <= 800; i++)   // 200HZ ~ 800HZ
    {
        tone(9,i);
    }
    delay(1000);                     //Max Frequency hold 1s
    for(int i= 800; i >= 200; i--)   // 800HZ ~ 200HZ
    {
        tone(9,i);
        delay(10);
    }
}
```

Firstly,we use a simple procedure to understand how to use the buzzer, and its sound principle. And to drive a buzzer like singing sound,we need make the buzzer issued frequency and duration of the different sound . Cycle is equal to the reciprocal of the frequency, so you can know the time by frequency, and then by calling the delay function or timer to achieve.Similarly the sound duration can also be achieved through the delay function. So the key factor to make the buzzer sing is to know how much time to prolong! Play music with Arduino, you just neded to understand the two concepts of "tone" and "beat."

The tone means the frequency at which a note should be sung.

The beat means how long a note should be sung.

The commonly used method is "look-up table method", this method is complex where you have to find the corresponding frequency of each note (according to the note, the frequency comparison), and then according to the formula converted to the corresponding time (take half cycle), and then through the delay function implementation. Finally by programming to achieve.

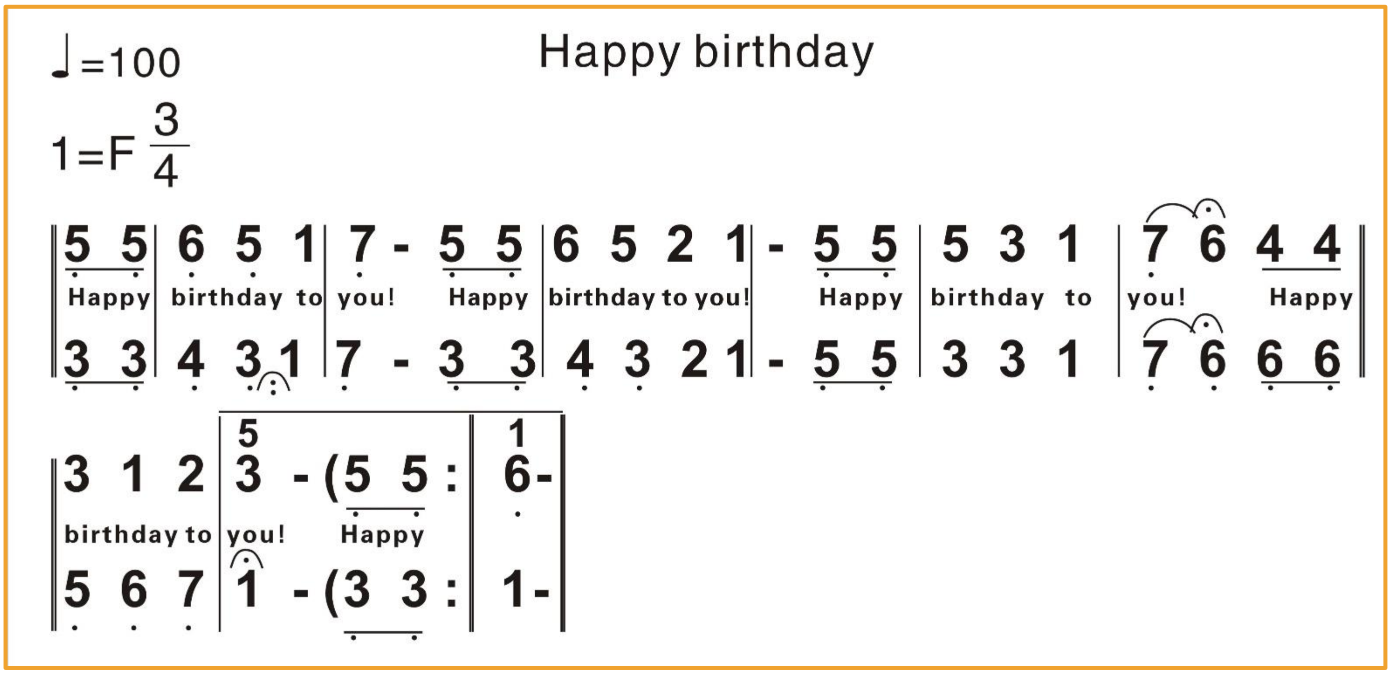
The whole process is like this:

Firstly, according to the score of Happy Birthday song, convert each tone to the corresponding frequency

For example: picture 3.2.11 is the note frequency conversion table, pictue 3.2.12 is the Happy Birthday song score.

	Musical notes	Corresponding frequency （Hz）	Half cycle(us)
Bass	1	261.63	1911.13
	1.5	277.18	1803.86
	2	293.66	1702.62
	2.5	311.13	1607.06
	3	329.63	1516.86
	4	349.23	1431.73
	4.5	369.99	1351.37
	5	392.00	1275.53
	5.5	415.30	1203.94
	6	440.00	1136.36
	6.5	446.16	1120.66
	7	493.88	1012.38
Alto	1	523.25	955.56
	1.5	554.37	901.93
	2	587.33	851.31
	2.5	622.25	803.53
	3	659.26	758.43
	4	698.46	715.86
	4.5	739.99	675.69
	5	783.99	637.76
	5.5	830.61	601.97
	6	880.00	568.18
	6.5	932.33	536.29
	7	987.77	506.19
Treble	1	1046.50	477.78
	1.5	1108.73	450.97
	2	1174.66	425.66
	2.5	1244.51	401.77
	3	1318.51	379.22
	4	1396.91	357.93
	4.5	1479.98	337.84
	5	1567.98	318.88
	5.5	1661.22	300.98
	6	1760.00	284.09
	6.5	1864.66	268.15
	7	1975.53	253.10

The note frequency conversion table



The score of Happy birthday song

Firstly , let’s learn some knowledge about music score and look at the above music score ,the bass is the one which with point under the number, the normal tone without any point .The treble is the one which with point above the number. The bass of tone 5 is 4.5,the treble is 5.5. Other notes are the corresponding truth. There is a “1=F” on the upper left of music score, while the general music score is C ,it is “1=C”.Note, the 1234567(do ,re,mi,fa,so,la,xi,duo) relative is CDEFGAB,not ABCDEFG.So, if the rule is F ,that means 2 is G tone,3 is A tone,……7 is E tone. So, in the situation, the bass 5 corresponds to bass 1.5,the tone need to move to the right or left. If you still don't understand, look at the following:

1 originally corresponding should be c,4 originally should correspond to f.

Then now 1 corresponds to F, which corresponds to 4, then 1.5 corresponds to 4.5, 2 corresponds to 5.

So, bass 5 is actually 4.5, so half cycle is 1803μs.

As to it is based on half-cycle calculations, because the single-chip microcomputer makes a sound by looping resetting the port connected to the buzzer, so it is a half-cycle. Becaues our product is passive buzzer, the active buzzer is full cycle.

Then according to the above reason, converse the tone one by one, achieve it by delay function. Because the frequency of the conversion of each note is different, you need to using multiple delay functions to achieve accurate tone frequencies one by one. But this is too complicated, and the microcontroller itself is not specifically to sing. The delay function has almost the same frequency in order to adapt to each tone, you need to calculate it by yourself, and different songs have different values, so this is the more troublesome issue.

After we know the frequency of the pitch, the next step is to control the playing time of the notes. Each note plays for a certain amount of time so that it can be a beautiful piece of music. The rhythm of notes is divided into one beat, half beat, 1/4 beat, and 1/8 beat. We stipulate that the time of a clap of notes is 1, half a beat for the 0.5;1/4 Pat for the 0.25;1/8 0.125 ...

Firstly,ordinary notes occupy for 1 shots.

Secondly, underlined notes indicate 0.5 beats; two underlines are quarter beats (0.25)

Thirdly, the notes which followed by a point, which means more 0.5 beats, that is 1+0.5.

Fourthly, the notes followed by a "-", which means more than one beat, that is, 1 +1.

So we can give this beat to each note and play it out. As for the conversion of beats to frequency, it also has corresponding list, just follow table two:

Music beat	1/4 beat delay time	Music	1/8 beat delay time
4/4	125ms	4/4	62ms
3/4	187ms	3/4	94ms
2/4	250ms	2/4	125ms

Table 2: Beat and frequency correspondence table

It is also achieved through the delay function,of course there will be errors. The idea of programming is very simple, firstly convert the note frequency and the time you want to sing into the two arrays. Then in the main programming, through the delay function to reach the corresponding frequency . sing it over, stop for a while, and then sing it, all the conversion is complete, we get the following frequency (Table 3) and beat:

Do 262	Re 294	Mi 330	Fa 349	Sol 392	La 440	Si 494	Do_h 523
Si_h 988	Mi_h 659		La_h 880	Sol_h 784	Fa_h698	Re_h 587	

Table 3: Happy Birthday Song beat table

According to the music score, we can get the frequency of the birthday song.

Sol,Sol,La,Sol,Do_h,Si,Sol,Sol,La,Sol,Re_h,Do_h,Sol,Sol,Sol_h,Mi_h,Do_h,Si,La,Fa_h,Fa_h,Mi_h,Do_h,Re_h,Do_hfloat

The beat is as follows:

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

Add beats and frequency to the program and download it to Arduino to play.

Happy Birthday music score beat, view table two rhythm and frequency corresponding table 1 beats time is 187*4 = 748ms

Note: The procedure is shown in the: "Lesson7-Active&Passive Buzzer\Happy_Birthday \Happy_Birthday.ino"

```
#define RGB_RED    11
#define RGB_GREEN  10
#define RGB_BLUE   9

#define Do 262
#define Re 294
#define Mi 330
#define Fa 349
#define Sol 392
#define La 440
#define Si 494
#define Do_h 523
#define Re_h 587
#define Mi_h 659
#define Fa_h 698
#define Sol_h 784
#define La_h 880
#define Si_h 988

int buzzer = 8;   // buzzer pin 9
int length;
// happy birthday Music score

void setColor(int red,int green,int blue)
{
    analogWrite(RGB_RED,red);
    analogWrite(RGB_GREEN,green);
    analogWrite(RGB_BLUE,blue);
}
int scale[] = {Sol, Sol, La, Sol, Do_h, Si, Sol, Sol,
             La, Sol, Re_h, Do_h, Sol, Sol, Sol_h, Mi_h,
             Do_h, Si, La, Fa_h, Fa_h, Mi_h, Do_h, Re_h, Do_h };

// Beats time
float durt[]=
{
    0.5, 0.5, 1, 1, 1, 1+1, 0.5, 0.5,
    1, 1, 1, 1+1, 0.5, 0.5, 1, 1,
    1, 1, 1, 0.5, 0.5, 1, 1, 1, 1+1
};
```

```
void setup()
{
   pinMode(buzzer, OUTPUT);
   // get scale length
   length = sizeof(scale) / sizeof(scale[0]);
   pinMode(RGB_RED,OUTPUT);
   pinMode(RGB_GREEN,OUTPUT);
   pinMode(RGB_BLUE,OUTPUT);
   Serial.begin(9600);
}

void loop()
{
   for(int x = 0; x < length; x++) {
      // Serial.println(scale[x]);
      tone(buzzer, scale[x]);
      setColor(scale[x] - 425, scale[x] - 500, scale[x] - 95);
      // 1= 3/4F so one Beats is  187*4 = 748ms
      delay(748 * durt[x]);
      noTone(buzzer);
   }
   delay(3000);
}
```