Power amplifier module 8002B chip

Description:

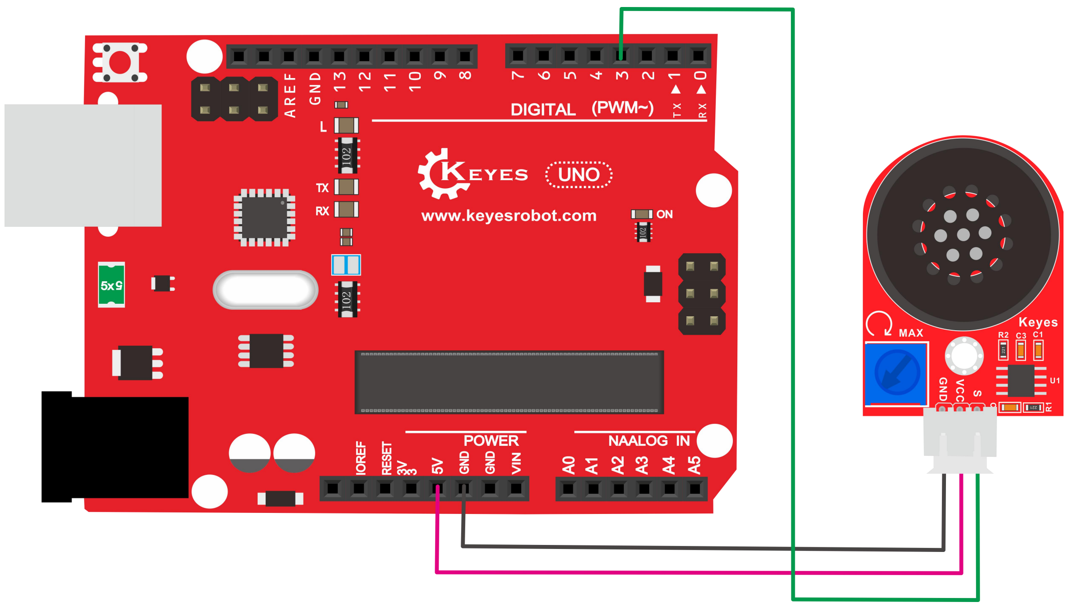
The main components of this module are an adjustable potentiometer, a speaker, and an audio amplifier chip; the main function is: it can amplify the output small audio signal, probably with a magnification of 8.5 times, and can pass through the built-in low-power speaker It can also be used to play music when it is played. As an external amplification device for some music playback devices, please pay attention to first turn the volume to the minimum when using it, and then increase it slowly to prevent burnout of the speaker.
When we connect the single-chip microcomputer for testing, we can input square waves of different frequencies and different durations at the signal end to edit the sound of the speaker.
At the same time, the module is compatible with various microcontroller control boards, such as arduino series microcontrollers. When in use, we can stack a sensor expansion board on the microcontroller. The module is connected with its own wire, and then connected to the sensor expansion board, which is simple and convenient.
At the same time, the module comes with a positioning hole with a diameter of 3mm, which is convenient for you to fix the module to other equipment.

Technical Parameters:

Working voltage: DC 5V

Working current: ≥500mA
Maximum power: 2W
Working temperature: 0-40℃
Size: 49*31*15.6MM
Speaker power: 0.15W
Speaker sound: 80db
Amplification chip: SC8002B

wiring



Test code

```
#define D0 -1
#define D1 262
#define D2 293
#define D3 329
#define D4 349
#define D5 392
#define D6 440
#define D7 494
#define M1 523
#define M2 586
#define M3 658
#define M4 697
#define M5 783
#define M6 879
#define M7 987
#define H1 1045
#define H2 1171
#define H3 1316
```

```
#define H4 1393
#define H5 1563
#define H6 1755
#define H7 1971
//List all the frequencies of the D key
#define WHOLE 1
#define HALF 0.5
#define QUARTER 0.25
#define EIGHTH 0.25
#define SIXTEENTH 0.625
//List all beats
int tune[]=        //List the frequencies according to the numbered musical notation
{
  M3,M3,M4,M5,
  M5,M4,M3,M2,
  M1,M1,M2,M3,
  M3,M2,M2,
  M3,M3,M4,M5,
  M5,M4,M3,M2,
  M1,M1,M2,M3,
  M2,M1,M1,
  M2,M2,M3,M1,
  M2,M3,M4,M3,M1,
  M2,M3,M4,M3,M2,
  M1,M2,D5,D0,
  M3,M3,M4,M5,
  M5,M4,M3,M4,M2,
  M1,M1,M2,M3,
  M2,M1,M1
};
float durt[]=       //List the beats according to the numbered musical notation
{
  1,1,1,1,
  1,1,1,1,
  1,1,1,1,
  1+0.5,0.5,1+1,
  1,1,1,1,
  1,1,1,1,
  1,1,1,1,
  1+0.5,0.5,1+1,
  1,1,1,1,
  1,0.5,0.5,1,1,
  1,0.5,0.5,1,1,
  1,1,1,1,
  1,1,1,1,
  1,1,1,0.5,0.5,
  1,1,1,1,
  1+0.5,0.5,1+1,
};
int length;
int tonepin=3;   //Need to use port 3
void setup()
{
  pinMode(tonepin,OUTPUT);
  length=sizeof(tune)/sizeof(tune[0]);   //Calculated length
}
void loop()
{
  for(int x=0;x<length;x++)
  {
    tone(tonepin,tune[x]);
    delay(500*durt[x]);   //Here is used to adjust the delay according to the beat. The index
of 500 can be adjusted by yourself. In this music, I found that 500 is more appropriate.
    noTone(tonepin);
  }
  delay(2000);
}
```

Test Results

Connect the cable and upload the code. After power on, the horn will sound "Ode to Joy", and the volume of the sound can be adjusted by rotating the potentiometer.