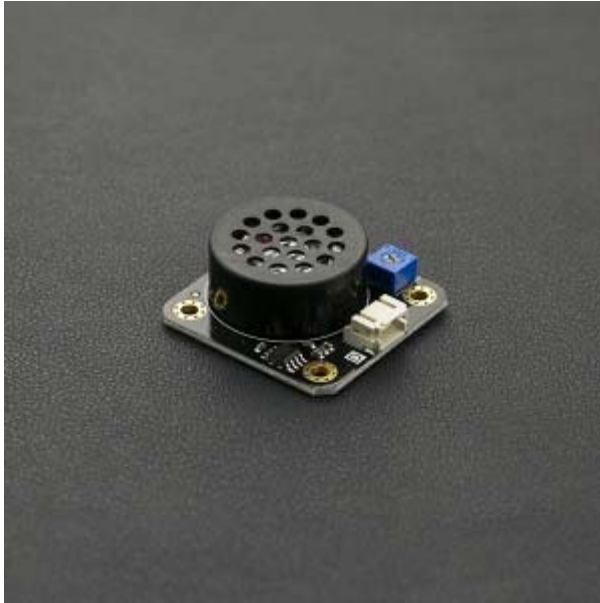




DFRobot Speaker v1.0 SKU: FIT0449



Contents

- [1 Introduction](#)
- [2 Specifications](#)
- [3 Pin Definitions](#)
- [4 Connection Diagram](#)
 - [4.1 Sample Code](#)

Introduction

This speaker module is based on the 8002 speaker amplifier IC. It can be used as a buzzer and also to play high quality audio. Volume output is controlled via the potentiometer.

The module has a compact footprint, simple interface, wide voltage input range, long service life and various other advantages. It can be used with an Arduino I/O expansion shield for interactive, sound related creations.

Specifications

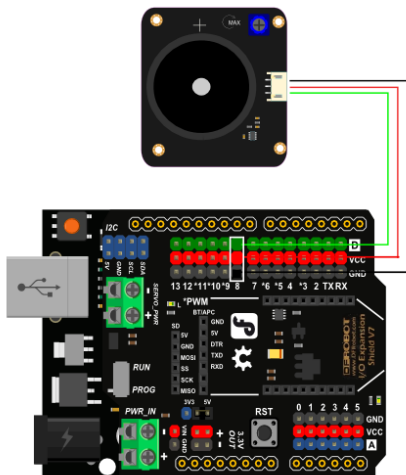
- Operating voltage: 2.0 - 5.5V
- Interface Type: Digital
- Module size: 40mm x 40mm/ 1.57 inches x 1.57 inches
- Support Gravity interface
- Power rating (Speaker component): 0.5W
- Impedance (Speaker component): 8 ohm

Pin Definitions

1. Ground (GND)
2. Power supply (VCC)
3. Input signal (Signal)



Connection Diagram



Speaker speaker with amplifier module connection diagram

Note: The speaker signal input can be connected to any digital pin

Sample Code

```
/******  
Super Mario Music  
*****/  
  
#define NOTE_B0 31  
#define NOTE_C1 33  
#define NOTE_CS1 35  
#define NOTE_D1 37  
#define NOTE_DS1 39  
#define NOTE_E1 41  
#define NOTE_F1 44  
#define NOTE_FS1 46  
#define NOTE_G1 49  
#define NOTE_GS1 52  
#define NOTE_A1 55  
#define NOTE_AS1 58  
#define NOTE_B1 62  
#define NOTE_C2 65  
#define NOTE_CS2 69  
#define NOTE_D2 73  
#define NOTE_DS2 78  
#define NOTE_E2 82  
#define NOTE_F2 87  
#define NOTE_FS2 93  
#define NOTE_G2 98  
#define NOTE_GS2 104  
#define NOTE_A2 110  
#define NOTE_AS2 117  
#define NOTE_B2 123
```

```
#define NOTE_C3 131
#define NOTE_CS3 139
#define NOTE_D3 147
#define NOTE_DS3 156
#define NOTE_E3 165
#define NOTE_F3 175
#define NOTE_FS3 185
#define NOTE_G3 196
#define NOTE_GS3 208
#define NOTE_A3 220
#define NOTE_AS3 233
#define NOTE_B3 247
#define NOTE_C4 262
#define NOTE_CS4 277
#define NOTE_D4 294
#define NOTE_DS4 311
#define NOTE_E4 330
#define NOTE_F4 349
#define NOTE_FS4 370
#define NOTE_G4 392
#define NOTE_GS4 415
#define NOTE_A4 440
#define NOTE_AS4 466
#define NOTE_B4 494
#define NOTE_C5 523
#define NOTE_CS5 554
#define NOTE_D5 587
#define NOTE_DS5 622
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_FS5 740
#define NOTE_G5 784
#define NOTE_GS5 831
#define NOTE_A5 880
```

```
#define NOTE_AS5 932
#define NOTE_B5 988
#define NOTE_C6 1047
#define NOTE_CS6 1109
#define NOTE_D6 1175
#define NOTE_DS6 1245
#define NOTE_E6 1319
#define NOTE_F6 1397
#define NOTE_FS6 1480
#define NOTE_G6 1568
#define NOTE_GS6 1661
#define NOTE_A6 1760
#define NOTE_AS6 1865
#define NOTE_B6 1976
#define NOTE_C7 2093
#define NOTE_CS7 2217
#define NOTE_D7 2349
#define NOTE_DS7 2489
#define NOTE_E7 2637
#define NOTE_F7 2794
#define NOTE_FS7 2960
#define NOTE_G7 3136
#define NOTE_GS7 3322
#define NOTE_A7 3520
#define NOTE_AS7 3729
#define NOTE_B7 3951
#define NOTE_C8 4186
#define NOTE_CS8 4435
#define NOTE_D8 4699
#define NOTE_DS8 4978

// notes in the melody:
int melody[] = {
```

```

NOTE_E4, NOTE_E4, NOTE_E4, NOTE_C4, NOTE_E4, NOTE_G4, NOTE_G3,
NOTE_C4, NOTE_G3, NOTE_E3, NOTE_A3, NOTE_B3, NOTE_AS3, NOTE_A3, NOTE_G3, NOTE
_E4, NOTE_G4, NOTE_A4, NOTE_F4, NOTE_G4, NOTE_E4, NOTE_C4, NOTE_D4, NOTE_B3,
NOTE_C4, NOTE_G3, NOTE_E3, NOTE_A3, NOTE_B3, NOTE_AS3, NOTE_A3, NOTE_G3, NOTE
_E4, NOTE_G4, NOTE_A4, NOTE_F4, NOTE_G4, NOTE_E4, NOTE_C4, NOTE_D4, NOTE_B3,
NOTE_G4, NOTE_FS4, NOTE_E4, NOTE_DS4, NOTE_E4, NOTE_GS3, NOTE_A3, NOTE_C4, NO
TE_A3, NOTE_C4, NOTE_D4, NOTE_G4, NOTE_FS4, NOTE_E4, NOTE_DS4, NOTE_E4, NOTE_
C5, NOTE_C5, NOTE_C5,
NOTE_G4, NOTE_FS4, NOTE_E4, NOTE_DS4, NOTE_E4, NOTE_GS3, NOTE_A3, NOTE_C4, NO
TE_A3, NOTE_C4, NOTE_D4, NOTE_DS4, NOTE_D4, NOTE_C4,
NOTE_C4, NOTE_C4, NOTE_C4, NOTE_C4, NOTE_D4, NOTE_E4, NOTE_C4, NOTE_A3, NOTE_
G3, NOTE_C4, NOTE_C4, NOTE_C4, NOTE_C4, NOTE_D4, NOTE_E4,
NOTE_C4, NOTE_C4, NOTE_C4, NOTE_C4, NOTE_D4, NOTE_E4, NOTE_C4, NOTE_A3, NOTE_
G3
};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
8,4,4,8,4,2,2,
3,3,3,4,4,8,4,8,8,8,4,8,4,3,8,8,3,
3,3,3,4,4,8,4,8,8,8,4,8,4,3,8,8,2,
8,8,8,4,4,8,8,4,8,8,3,8,8,8,4,4,4,8,2,
8,8,8,4,4,8,8,4,8,8,3,3,3,1,
8,4,4,8,4,8,4,8,2,8,4,4,8,4,1,
8,4,4,8,4,8,4,8,2
};

void setup() {
// iterate over the notes of the melody:
for (int thisNote = 0; thisNote < 98; thisNote++) {

// to calculate the note duration, take one second
// divided by the note type.
//e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
int noteDuration = 800/noteDurations[thisNote];
tone(8, melody[thisNote],noteDuration);

```

```
// to distinguish the notes, set a minimum time between them.
// the note's duration + 30% seems to work well:
int pauseBetweenNotes = noteDuration * 1.30;
delay(pauseBetweenNotes);
// stop the tone playing:
noTone(8);
}
}

void loop() {
// no need to repeat the melody.
}
```

If you want to melody to play repeatedly, move the setup part of the code to the main loop of the program.

For any questions/advice/cool ideas to share, please visit the [DFRobot Forum](#) or email techsupport@dfrobot.com