activeBuzzer

Overview

This class will use an active buzzer to make sounds.

Experimental Materials:

Raspberry Pi *1

T-type expansion board *1

Active buzzer *1

Breadboard*1

Some DuPont lines

Product description:

Active buzzer

● Application: Widely used in computers, printers, copiers, alarms, electronic toys, telephones and other electronic products

● Function: The active buzzer can be connected to the rated power supply directly (the new buzzer has been marked on the label) and it can be continuously sounded. The passive buzzer needs to be connected to the audio output circuit just like the electromagnetic speaker. Can only sound.

Technical Parameters:

Product Name : Electronic Alarm Buzzer;

Sound-making Type : Continuous Sound;

Body Size: 12 x 9.5mm / 0.47" x 0.37" (D*T);

Pin Pitch : 7mm / 0.27";

Terminals: 2 terminal;

Rated Voltage : DC 5V;

Current: less than 25mA;

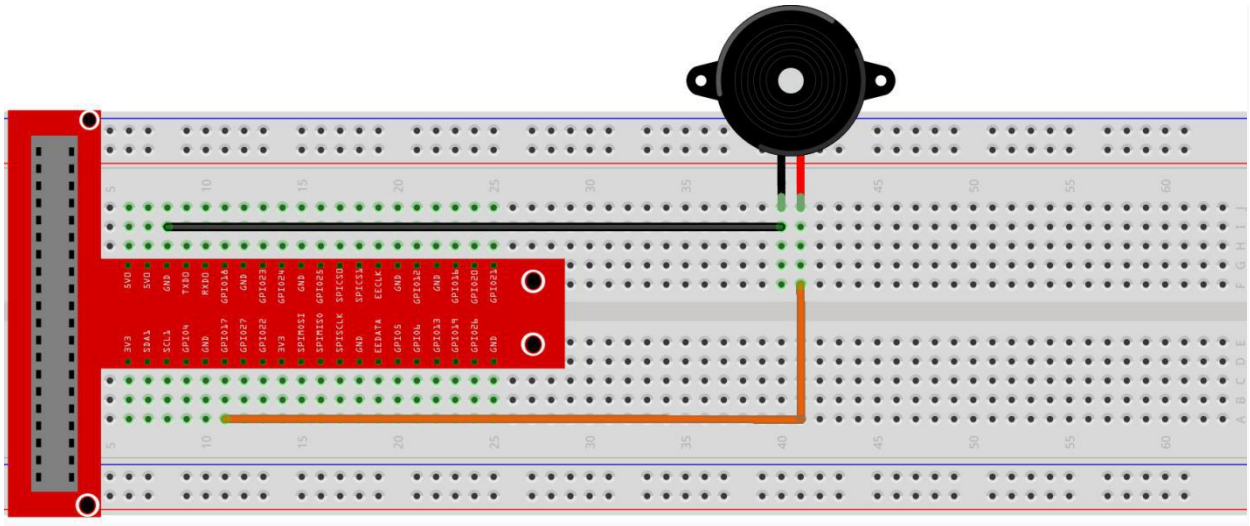
Frequency: 2300Hz(around);

Material : Plastic, electronic parts;

Color: Black;

Active buzzer experiment:

Wiring diagram:



C Code：

```
#include <wiringPi.h>
#include <stdio.h>

#define BuzzerPin      0

int main(void)
{
    if(wiringPiSetup() == -1)
    {
        printf("setup wiringPi failed !");
        return -1;
    }

    pinMode(BuzzerPin,  OUTPUT);
    while(1)
    {
            digitalWrite(BuzzerPin, HIGH);
            delay(500);
            digitalWrite(BuzzerPin, LOW);
            delay(500);
    }
    return 0;
}
```

Python code：

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
import time

BuzzerPin = 11    # pin11

def setup():
    GPIO.setmode(GPIO.BOARD)       # Numbers GPIOs by physical location
    GPIO.setup(BuzzerPin, GPIO.OUT)
    GPIO.output(BuzzerPin, GPIO.LOW)

def loop():
    while True:
        GPIO.output(BuzzerPin, GPIO.HIGH)
        time.sleep(0.5)
        GPIO.output(BuzzerPin, GPIO.LOW)
        time.sleep(0.5)

def destroy():
    GPIO.output(BuzzerPin, GPIO.LOW)
    GPIO.cleanup()                     # Release resource

if __name__ == '__main__':     # Program start from here
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        destroy()
```

Experimental results:

In the directory where the code file is located, execute the following command

C：

```
gcc -Wall -o activeBuzzer activeBuzzer.c -lwiringPi
sudo ./activeBuzzer
```

Python:

python buzzer_active.py

After the instruction is executed, the active buzzer sounds and then stops and continues to cycle