Passive Buzzer

Overview

This course will use the Raspberry Pi to control the passive buzzer to beep.

Experimental Materials

RaspberryPi \*1

Breadboard \*1

Passive buzzer \*1

Dupont Line

Preparatory work

1. Install python interpreter in your Raspberry Pi system

2. Install the RPi.GPIO library in your Raspberry Pi system

3. Install the wiringPi library in your Raspberry Pi system

Refer to the attached "Installing a Python Interpreter and Corresponding Libraries in a Raspberry Pi System" for details.

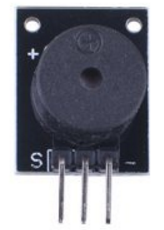
Product Description

Brief Introduction

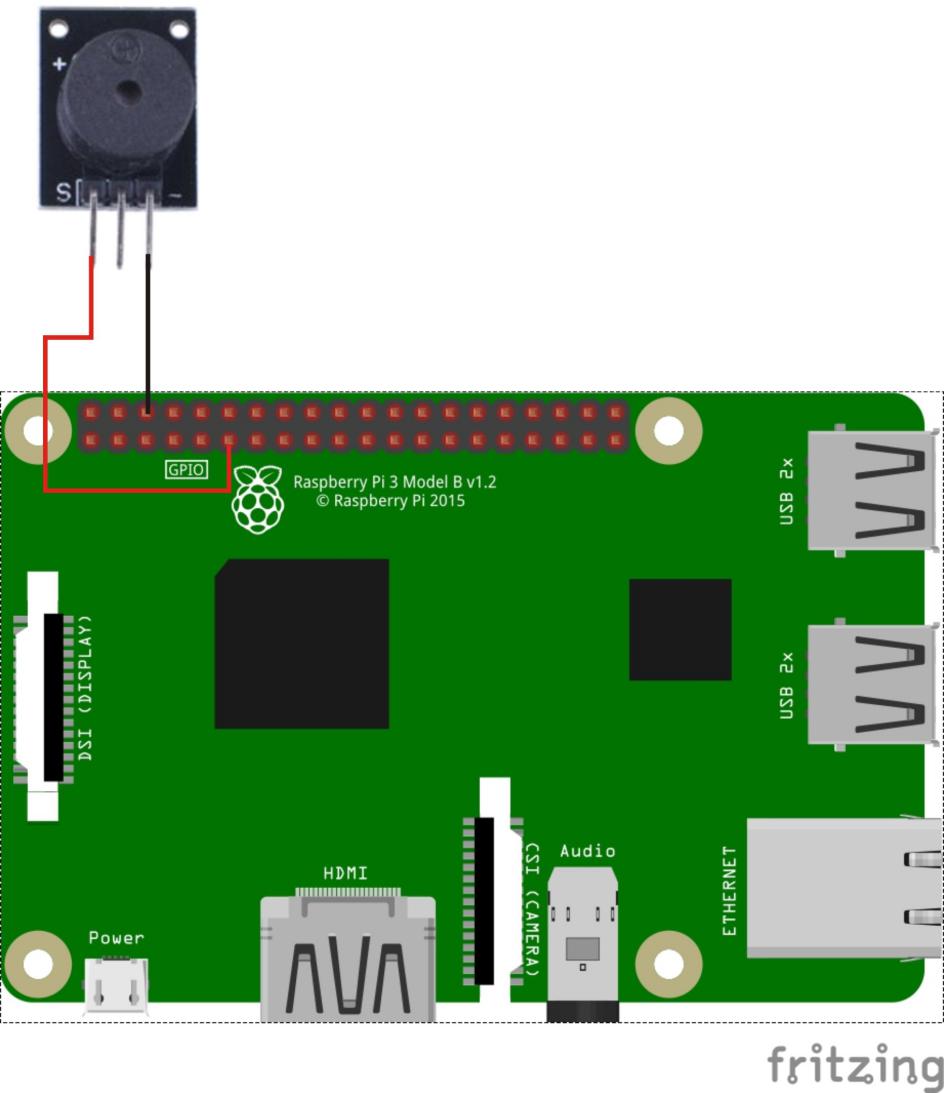
● Application: Active Buzzer is widely used in computers, printers, copiers, alarms, electronic toys, telephones and other electronic products.

● the differences between the Active Buzzer and Passive Buzzer: From exterior, the two buzzer seem to be the same. If the buzzer pin is faced up, there is a green circuit board on passive buzzer, on the contrary, active buzzer has no circuit board but is sealed by vinyl. In appearance, it is not absolutely possible to distinguish between active and passive. The most reliable method is not only to check the parameter manual of the product, but also to use a multimeter to test the buzzer resistance. Passive buzzer is 8Ωor 16Ω. Active buzzer are with a resistance of several hundred ohms or more.

● Function: The active buzzer can beep continuously by directly connecting with the rated power supply. And the passive buzzer is the same as the electromagnetic speaker, which needs to be connected to the audio output circuit to beep.



Wiring diagram



Sample code

1、python code

#!/usr/bin/env python

import RPi.GPIO as GPIO

import time

BuzzerPin = 11 # pin11

SPEED = 1

# List of tone-names with frequency

TONES = {"c6":1047,

"b5":988,

"a5":880,

"g5":784,

"f5":698,

"e5":659,

"eb5":622,

"d5":587,

"c5":523,

"b4":494,

"a4":440,

"ab4":415,

"g4":392,

"f4":349,

"e4":330,

"d4":294,

"c4":262}

# Song is a list of tones with name and 1/duration. 16 means 1/16

SONG = [

["e5",16],["eb5",16],

["e5",16],["eb5",16],["e5",16],["b4",16],["d5",16],["c5",16],

["a4",8],["p",16],["c4",16],["e4",16],["a4",16],

["b4",8],["p",16],["e4",16],["ab4",16],["b4",16],

["c5",8],["p",16],["e4",16],["e5",16],["eb5",16],

["e5",16],["eb5",16],["e5",16],["b4",16],["d5",16],["c5",16],

["a4",8],["p",16],["c4",16],["e4",16],["a4",16],

["b4",8],["p",16],["e4",16],["c5",16],["b4",16],["a4",4]

]

def setup():

GPIO.setmode(GPIO.BOARD) # Numbers GPIOs by physical location

GPIO.setup(BuzzerPin, GPIO.OUT)

def playTone(p,tone):

# calculate duration based on speed and tone-length

duration = (1./(tone[1]\*0.25\*SPEED))

if tone[0] == "p": # p => pause

time.sleep(duration)

else: # let's rock

frequency = TONES[tone[0]]

p.ChangeFrequency(frequency)

p.start(0.5)

time.sleep(duration)

p.stop()

def run():

p = GPIO.PWM(BuzzerPin, 440)

p.start(0.5)

for t in SONG:

playTone(p,t)

def destroy():

GPIO.output(BuzzerPin, GPIO.HIGH)

GPIO.cleanup() # Release resource

if \_\_name\_\_ == '\_\_main\_\_': # Program start from here

setup()

try:

run()

GPIO.cleanup()

except KeyboardInterrupt:

destroy()

1. C code

#include <wiringPi.h>

#include <softTone.h>

#include <stdio.h>

#define BuzPin 0

#define CL1 131

#define CL2 147

#define CL3 165

#define CL4 175

#define CL5 196

#define CL6 221

#define CL7 248

#define CM1 262

#define CM2 294

#define CM3 330

#define CM4 350

#define CM5 393

#define CM6 441

#define CM7 495

#define CH1 525

#define CH2 589

#define CH3 661

#define CH4 700

#define CH5 786

#define CH6 882

#define CH7 990

int song\_1[] =

{CM3,CM5,CM6,CM3,CM2,CM3,CM5,CM6,CH1,CM6,CM5,CM1,CM3,CM2,CM2,CM3,CM5,CM2,CM3,CM3,CL6,CL6,CL6,CM1,CM2,CM3,CM2,CL7,CL6,CM1,CL5};

int beat\_1[] = {1,1,3,1,1,3,1,1,1,1,1,1,1,1,3,1,1,3,1,1,1,1,1,1,1,2,1,1,1,1,1,1,1,1,3};

int song\_2[] =

{CM1,CM1,CM1,CL5,CM3,CM3,CM3,CM1,CM1,CM3,CM5,CM5,CM4,CM3,CM2,CM2,CM3,CM4,CM4,CM3,CM2,CM3,CM1,CM1,CM3,CM2,CL5,CL7,CM2,CM1};

int beat\_2[] = {1,1,1,3,1,1,1,3,1,1,1,1,1,1,3,1,1,1,2,1,1,1,3,1,1,1,3,3,2,3};

int main(void)

{

int i, j;

if(wiringPiSetup() == -1)

{

printf("setup wiringPi failed !");

return -1;

}

if(softToneCreate(BuzPin) == -1)

{

printf("setup softTone failed !");

return -1;

}

while(1)

{

printf("music is being played...\n");

for(i=0;i<sizeof(song\_1)/4;i++)

{

softToneWrite(BuzPin, song\_1[i]);

delay(beat\_1[i] \* 500);

}

for(i=0;i<sizeof(song\_2)/4;i++)

{

softToneWrite(BuzPin, song\_2[i]);

delay(beat\_2[i] \* 500);

}

}

return 0;

}

Experimental phenomena

Passive buzzer can play music.