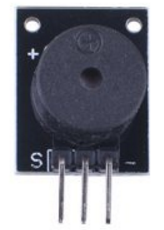
Passive Buzzer  


Overview

In this experiment, you’ll use the Raspberry Pi to generate a tone with the passive buzzer. Passive buzzers differ from active buzzers in that, like (electromagnetic) loudspeakers, they require a changing external signal (AC) to drive their tone. (An active speaker generates its own tone, at a fixed frequency; passive speakers can generate different tones depending on the frequency of the alternating current delivered to them.) Passive buzzers are used as tone generators in a wide variety of applications: toys, instruments, telephones, status indicators, etc.

Experimental Materials

Raspberry Pi x1

Breadboard x1

Passive Buzzer x1

Dupont jumper wires

Experimental Procedure

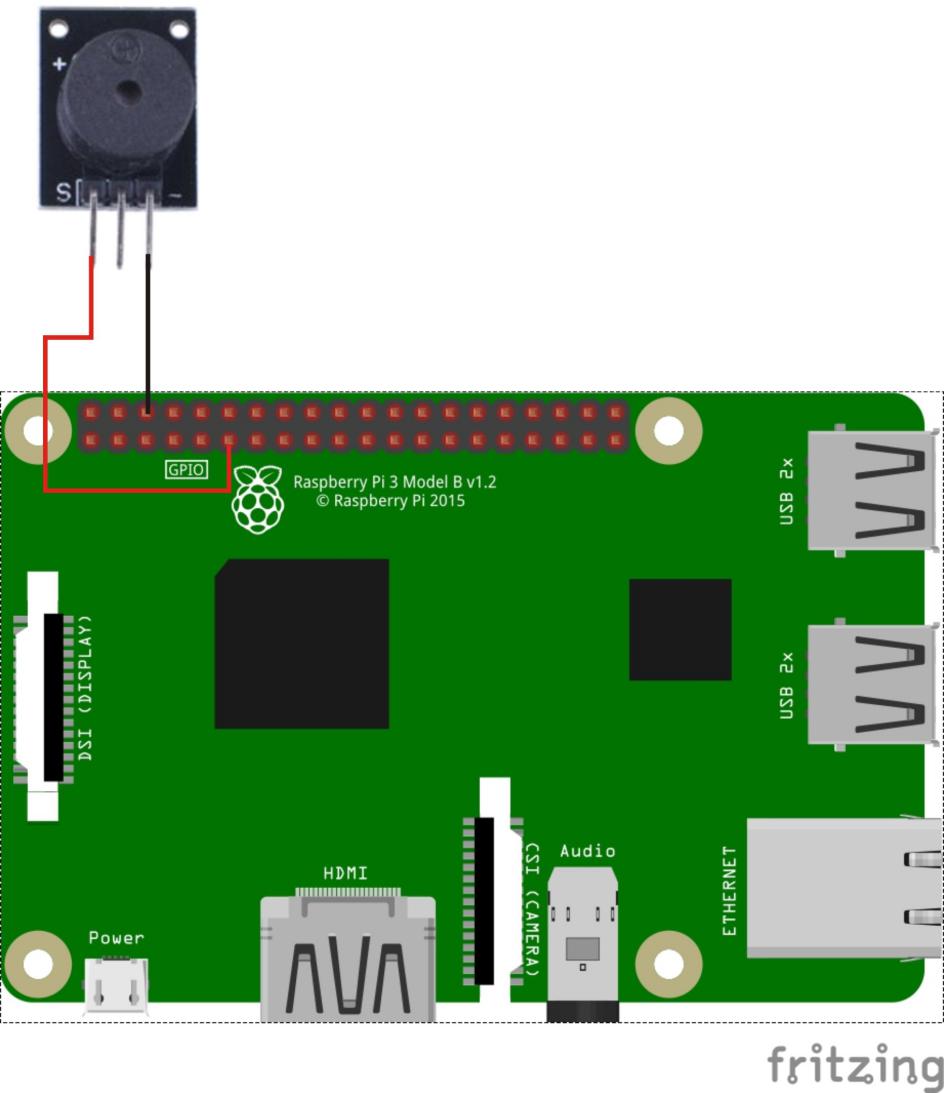
1. If you have not done so already, prepare your development system by installing the Python interpreter, RPi.GIO library, and wiringPi library as described in READ\_ME.TXT.
2. Install the active buzzer in your breadboard, and use Dupont jumper wires to connect it to your Raspberry Pi as illustrated in the Wiring Diagram below. Execute the sample stored in this experiment’s subfolder.

If using C, compile and execute the C code:  
cd Code/C  
gcc passiveBuzzer.c -o passiveBuzzer.out –lwiringPi  
./passiveBuzzer.out

If using Python, launch the Python script:  
cd Code/Python  
python passiveBuzzer.py

1. Make experimental observations. The buzzer plays a brief tune defined in the source code.

Wiring Diagram



Passive Buzzer pin position:

"S" ↔ Raspberry Pi pin 11

"-" ↔ Raspberry Pi GND

Sample Code

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()

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;

}