Active Buzzer

Overview

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

Experimental Materials

RaspberryPi \*1

Breadboard \*1

Active buzzer \*1

Dupont Line

Ready to 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

1. 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 DC 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.



1. Characteristic parameters

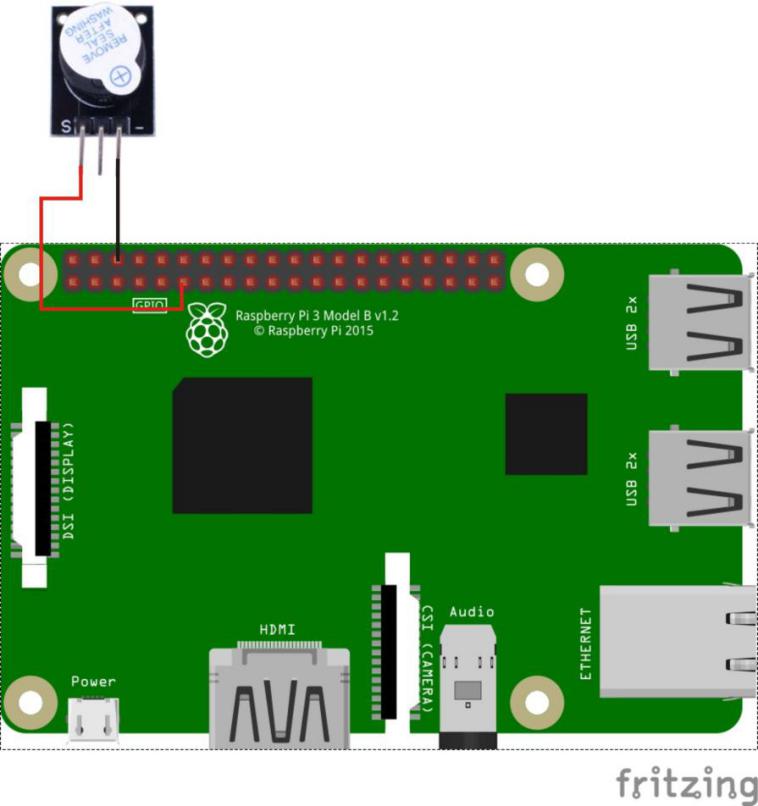
◆Sound type: beep Continuously

◆Rated voltage: DC 5V

◆Current: less than 25 mA;

◆Material: plastic, electronic parts;

Wiring diagram



Sample code

1、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()

1. 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;

}

Experimental phenomena

The active buzzer beeps at intervals of 0.5 seconds.