KUMAN
www.kumantech.com

# Active Buzzer


## Overview

In this experiment, you'll use the Raspberry Pi to generate a tone with the active buzzer. Active buzzers differ from passive buzzers in that the contain an internal oscillator that produces the (fixed) output pitch; all they require externally is DC voltage. Active buzzers are used in a wide variety of electronic devices for "beep tones" and error indicators.

## Experimental Materials

```
Raspberry Pi      x1
Breadboard        x1
Active 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.GPIO library, and wiringPi library as described in READ_ME_FIRST.TXT.
2. You can remove the "remove seal after washing" sticker covering the buzzer, if it remains. (The manufacturing process of many electronic components involves a solvent wash to remove solder flux. This sticker protects the speaker during this process. But if you remove the sticker be sure to see the note in *Technical Background*, below, about distinguishing between active and passive buzzers!)
3. 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:

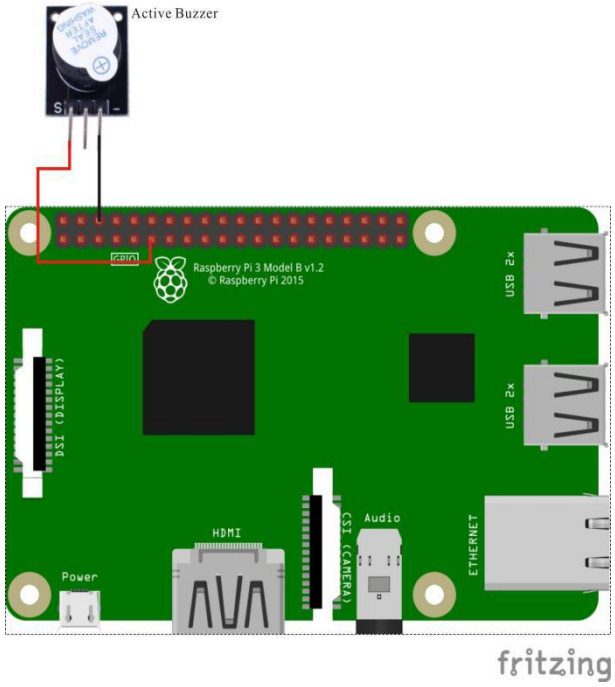
```
cd Code/C
gcc activeBuzzer.c -o activeBuzzer.out -lwiringPi
./activeBuzzer.out
```

If using Python, launch the Python script:

```
cd Code/Python
python activeBuzzer.py
```

4. Make experimental observations. The buzzer beeps, at half-second intervals.

## Wiring Diagram




Active 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
def setup():
   GPIO.setmode(GPIO.BOARD)
```

```
    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()
```

## 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;
}
```

## Technical Background

Note it is not always possible to tell an active from a passive buzzer from external appearances. In Kuman sensor kits, the underside of the passive buzzer module exposes a green circuit board; the active buzzer shows no circuit board and is sealed in vinyl. Given an unknown buzzer found in the field, the most reliable method is to use a multimeter to test the buzzer resistance. Passive buzzers are usually low impedance: 8Ωor 16Ω. Active buzzers, by comparison, have a resistance of several hundred ohms or more. (This active buzzer is rated at +5V DC and <25 mA).