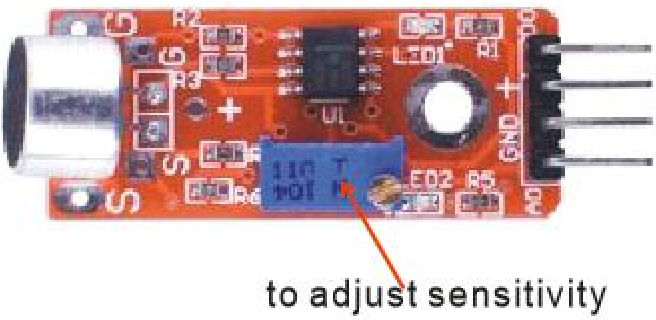
www.kumantech.com

Big Sound Sensor



Overview

The Big Sound Sensor packages a sound-sensitive, capacitive electret microphone inside a convenience circuit that reports two outputs. As acoustic waves vibrate the thin electret film, they generate a small voltage, allowing the intensity of the sound to be reported as an analog output. At the same time, a digital output indicates whether that measured volume exceeds a particular user-adjustable threshold. Simple sound detection has many applications in automation, security, and novelty and entertainment technologies. In this experiment, you'll use your Raspberry Pi and the analog-to-digital converter to monitor both outputs of the sound sensor.

The Big Sound Sensor is very similar to the Small Sound Sensor, except it contains a larger microphone, which makes it more sensitive to a broader range of sounds and therefore able to detect quieter noises. The wiring diagrams, experimental procedures and source code of both sound sensor experiments are the same.

Experimental Materials

Raspberry Pi	x1
Breadboard	x1
Big sound sensor	x1
ADC0832	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. Install the ADC0832 analog/digital converter IC and the sound sensor on your breadboard, and use Dupont jumper wires to connect them to each other and your Raspberry Pi as illustrated in the Wiring Diagram below.

3. Execute the sample stored in this experiment's subfolder.
   If using C, compile and execute the C code:

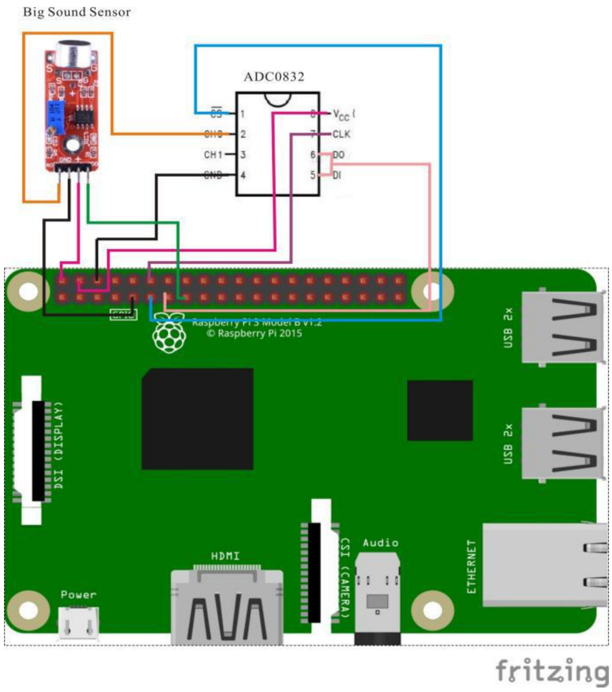
```
cd Code/C
gcc soundSensor.c -o soundSensor.out -lwiringPi
./soundSensor.out
```

   If using Python, launch the Python script:

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

4. Make experimental observations. The command line interface of the Raspberry Pi displays the current measured sound intensity (from the analog signal run through the ADC). When that intensity exceeds the threshold value determined by the onboard potentiometer, the Raspberry Pi also displays "voice in!" To change the sensitivity of that second measurement, change the position of the potentiometer dial.

Wiring Diagram



AD0382 pin position:

CS	↔	Raspberry Pi pin 11
CLK	↔	Raspberry Pi pin 12
DI	↔	Raspberry Pi pin 13
D0	↔	Raspberry Pi pin 13
CH0	↔	Sound Sensor pin A0
VCC	↔	Raspberry Pi +5V
GND	↔	Raspberry Pi GND

Sound Sensor position:

A0	↔	ADC0382 Pin CH0
D0	↔	Raspberry Pi Pin 15
GND	↔	Raspberry Pi GND
"+"	↔	Raspberry Pi +5V

Sample Code

Python Code

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
import ADC0832
import time

MIC_DO_PIN = 15

def init():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(MIC_DO_PIN, GPIO.IN,
pull_up_down=GPIO.PUD_UP)
    ADC0832.setup()
```

```
def loop():
    while True:
        global digitalVal
        digitalVal = GPIO.input(MIC_DO_PIN)
        if(digitalVal == 0):
            print 'DO is %d' % digitalVal
            print "voice in..."
            print 'Current analog value is %d'%
ADC0832.getResult(0)
        else:
            pass

if __name__ == '__main__':
    init()
    try:
        loop()
    except KeyboardInterrupt:
        ADC0832.destroy()
        print 'The end !'
```

C Code

```
#include <wiringPi.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <stdlib.h>

#define     ADC_CS    0
#define     ADC_CLK   1
#define     ADC_DIO   2

#define  Sound_DO_Pin   3

typedef unsigned char uchar;
typedef unsigned int uint;


uchar get_ADC_Result(void)
{
   uchar i;
   uchar dat1=0, dat2=0;
```

```
    digitalWrite(ADC_CS, 0);
    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,1);   delayMicroseconds(2);
    digitalWrite(ADC_CLK,1);   delayMicroseconds(2);

    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,1);   delayMicroseconds(2);

    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,0);   delayMicroseconds(2);
    digitalWrite(ADC_CLK,1);
    digitalWrite(ADC_DIO,1);    delayMicroseconds(2);
    digitalWrite(ADC_CLK,0);
    digitalWrite(ADC_DIO,1);    delayMicroseconds(2);

    for(i=0;i<8;i++)
    {
        digitalWrite(ADC_CLK,1);   delayMicroseconds(2);
        digitalWrite(ADC_CLK,0);    delayMicroseconds(2);

        pinMode(ADC_DIO, INPUT);
        dat1=dat1<<1 | digitalRead(ADC_DIO);
    }

    for(i=0;i<8;i++)
    {
        dat2 = dat2 | ((uchar)(digitalRead(ADC_DIO))<<i);
        digitalWrite(ADC_CLK,1);   delayMicroseconds(2);
        digitalWrite(ADC_CLK,0);    delayMicroseconds(2);
    }

    digitalWrite(ADC_CS,1);
    pinMode(ADC_DIO, OUTPUT);
    return(dat1==dat2) ? dat1 : 0;
}

int main(void)
{
    uchar digitalVal = 1;
    uchar analogVal = 0;
```

```
    if(wiringPiSetup() == -1)
    {
        printf("setup wiringPi failed!\n");
        return -1;
    }

    pinMode(ADC_CS,  OUTPUT);
    pinMode(ADC_CLK, OUTPUT);
    pinMode(Sound_DO_PIN, INPUT);
    pullUpDnControl(Sound_DO_PIN, PUD_UP);
    printf("Please speak into the sensor...\n");

    while(1)
    {
        printf("Current analog value is %d.\n",
get_ADC_Result());

        if(!(digitalVal = digitalRead(Sound_DO_PIN)))
        {
            printf("D0 is %d.\n", digitalVal);
            printf("Voice in...");
        }

        delay(200);

    }

    return 0;
}
```