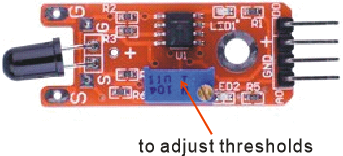
Flame Sensor  


Introduction

This course will use the Raspberry Pi to capture the signal from the flame sensor and controls the LED light on and off based on the captured signal.

Experimental Materials

Raspberry Pi x1

Breadboard x1

Flame sensor x1

ADC0832 x1

LED (3-pin) x1

Resistor (330Ω) 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 ADC0832 analog/digital converter IC, flame sensor, three-pin LED and resistor 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. Note you will connect only two of the three pins on the LED.
3. Execute the sample stored in this experiment’s subfolder.

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

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

1. Make experimental observations. When the flame sensor detects the flame and the intensity of the flame reaches the set threshold, the current flame sensor will be printed out in the command line interface of the Raspberry Pi system after the conversion of the ADC，when this value meet the set condition, Led will be on.

Brief Introduction

The flame sensor (infrared receiver transistor) is a sensor specially used by robots to search for fire sources. The flame sensor uses the characteristics that infrared rays are very sensitive to the flame, using a special infrared receiver tube to detect the flame and converts the brightness of the flame into a voltage signal, which is input to the central processor, and the CPU makes corresponding program processing according to the change of the signal.

**Characteristic Parameters**

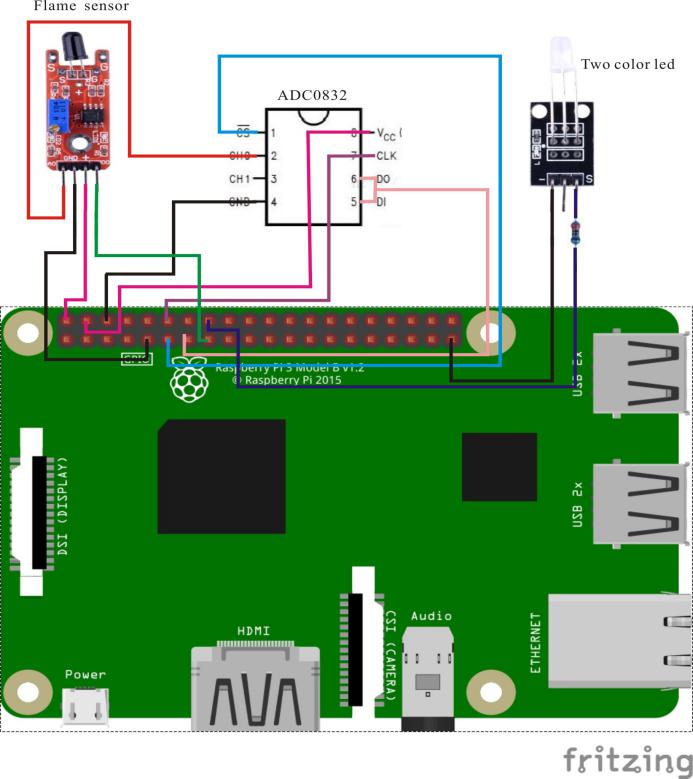
◆ The flame sensor is most sensitive to the flame and also responds to ordinary light. It is generally used for flame alarm purposes.

◆ The sensor and the flame must be kept at a distance so as not to damage the sensor at high temperatures

◆The larger the flame, the farther the test distance.

◆The working voltage is 3.3v -5V

Wiring Diagram



ADC0382 pin position:

CS ↔ Raspberry Pi Pin 11

CLK ↔ Raspberry Pi Pin 12

DI ↔ Raspberry Pi Pin 13

D0 ↔ Raspberry Pi Pin 13

CH0 ↔ Flame Sensor Pin A0

VCC ↔ Raspberry Pi +5V

GND ↔ Raspberry Pi GND

Flame Sensor pin position:

A0 ↔ ADC0382 Pin CH0

D0 ↔ Raspberry Pi Pin 15

GND ↔ Raspberry Pi GND

"+" ↔ Raspberry Pi +5V

LED pin position:

"S" ↔ Raspberry Pi Pin 16(through resistor)

"-" ↔ Raspberry Pi GND

Sample Code

Python Code

#!/usr/bin/env python

import RPi.GPIO as GPIO

import ADC0832

import time

Flame\_DO\_Pin = 15

LedPin = 16

thresholdVal = 150

def init():

GPIO.setmode(GPIO.BOARD)

GPIO.setup(Flame\_DO\_Pin, GPIO.IN, pull\_up\_down=GPIO.PUD\_UP)

GPIO.setup(LedPin, GPIO.OUT)

ADC0832.setup()

def loop():

while True:

global digitalVal, analogVal

digitalVal = GPIO.input(Flame\_DO\_Pin)

if(digitalVal == 1):

print 'DO is %d' % digitalVal

analogVal = ADC0832.getResult(0)

print 'Current analog value is %d'% analogVal

if(analogVal < thresholdVal):

GPIO.output(LedPin, GPIO.HIGH)

time.sleep(0.2)

else:

GPIO.output(LedPin, GPIO.LOW)

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 Flame\_DO\_Pin 3

#define LedPin 4

#define thresholdVal 150

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 !");

return -1;

}

pinMode(ADC\_CS, OUTPUT);

pinMode(ADC\_CLK, OUTPUT);

pinMode(Flame\_DO\_Pin, INPUT);

pullUpDnControl(Flame\_DO\_Pin, PUD\_UP);

pinMode(LedPin, OUTPUT);

while(1)

{

if((digitalVal = digitalRead(Flame\_DO\_Pin)))

{

printf("Do is %d.\n", digitalVal);

analogVal = get\_ADC\_Result();

printf("Current analog value is %d.\n", analogVal);

if(analogVal < thresholdVal)

{

digitalWrite(LedPin, HIGH);

}

delay(200);

}

else

{

digitalWrite(LedPin, LOW);

}

}

return 0;

}