KUMAN
www.kumantech.com

Analog Temperature Sensor

Overview

The analog temperature sensor is a *thermistor*, a crystalline semiconductor with a resistance that varies non-linearly with temperature. Thermistors are highly temperature-sensitive, and have small a simple structure, long life, and small size, so are widely used in industrial, agricultural, and domestic applications for temperature and pressure measurement and control. The thermistor in this analog sensor has an operating range of -55°C to +125°C with a sensitivity of 0.5°C. As an analog device, reported voltage increases as temperature increases, although since the relationship is non-linear it can be difficult to calculate the temperature on a linear scale (Kelvin, Celcius) directly from the voltage output. In this experiment, you'll use the ADC0832 to convert the sensor's analog signal to a digital temperature value you can read with your Raspberry Pi, and then illuminate an LED light when that temperature value exceeds a specific threshold.

Experimental Materials

Raspberry Pi	x1
Breadboard	x1
Analog temperature 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.GPIO library, and wiringPi library as described in READ_ME_FIRST.TXT.

2. Install the ADC0832 analog/digital converter IC, analog temperature 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 code stored in this experiment's subfolder.
   If using C, compile and execute the C code:

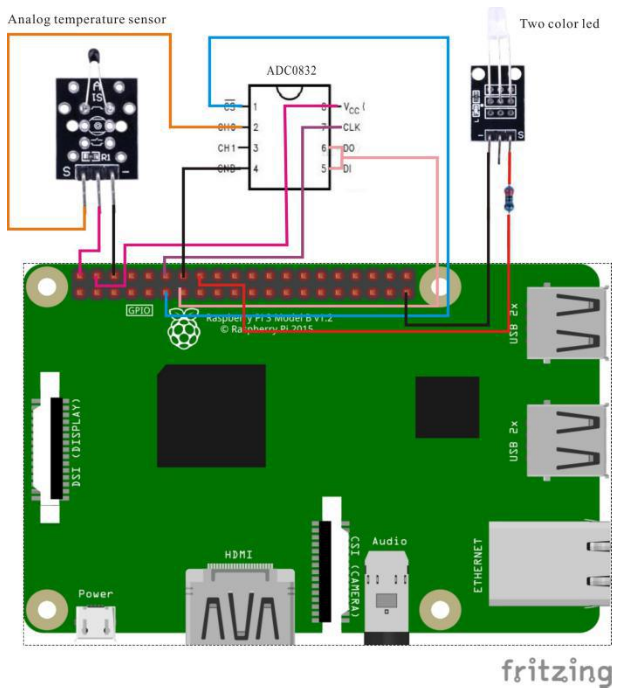
   ```
   cd Code/C
   gcc analogTemp.c -o analogTemp.out -lm -lwiringPi
   ./analogTemp.out
   ```

   If using Python, launch the Python script:

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

4. Make experimental observations. Slowly approach the sensor with an open flame (a match or candle) to increase the local temperature. The LED illuminates. Remove the heat source and the LED should extinguish. If you wish to calibrate your LED to other heat sources—such as body-temperature, when you gently squeeze the thermistor with your fingertips—you can experiment with different values of the *threshold* constant in your sample code. (In the sample code below, *threshold* has a default value of 200.)

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	↔	Analog Temperature Sensor Pin "S"
VCC	↔	Raspberry Pi +5V
GND	↔	Raspberry Pi GND

Analog Hall pin position:

"S"	↔	ADC0382 pin CH0
"+"	↔	Raspberry Pi +5V
"-"	↔	Raspberry Pi GND

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
import math

LedPin = 16
threshold = 200
def init():
   ADC0832.setup()
       GPIO.setmode(GPIO.BOARD)
       GPIO.setup(LedPin, GPIO.OUT)
       GPIO.output(LedPin, GPIO.LOW)

def loop():
   while True:
      analogVal = ADC0832.getResult(0)
      print 'analogVal = %d' % analogVal
      if(analogVal < threshold):
          GPIO.output(LedPin, True)
      else:
          GPIO.output(LedPin, False)
      time.sleep(0.2)

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

C Code

```
/*
 * compile with -lm for math library
 * gcc analogTempSensor.c -lwiringPi -lm
 */

#include <wiringPi.h>
#include <stdio.h>
#include <math.h>

typedef unsigned char uchar;
typedef unsigned int uint;

#define     ADC_CS    0
#define     ADC_CLK   1
#define     ADC_DIO   2
#define     LedPin    4
#define      threshold   200
uchar get_ADC_Result(uint channel)
{
   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,channel);   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);

    return(dat1==dat2) ? dat1 : 0;
}

int main(void)
{
    uchar analogVal;
    double Vr, Rt, temp;

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

    pinMode(ADC_CS,  OUTPUT);
    pinMode(ADC_CLK, OUTPUT);
    pinMode(LedPin, OUTPUT);

    while(1)
    {
        pinMode(ADC_DIO, OUTPUT);

        analogVal = get_ADC_Result(0);
        printf("analogVal = %d.\n", analogVal);
```

```
      if(analogVal < threshold)
      {
         digitalWrite(LedPin, HIGH);
      }
      else
      {
         digitalWrite(LedPin, LOW);
      }

      delay(1000);
   }

   return 0;
}
```