Project 11 - Using the Raspberry Pi to measure temperature

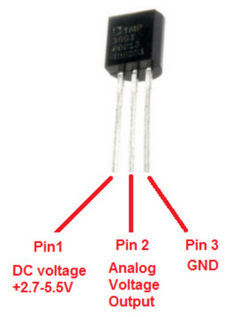
Outline

This application note shows the hardware and Python code required to measure temperature using the Raspberry Pi. Any of the Raspberry Pi models can be used including B, B+ and the Raspberry Pi 2.

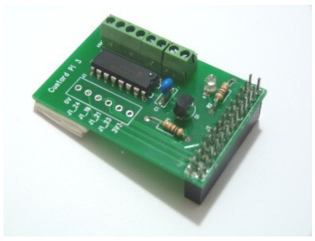
Hardware

TMP36: We are going to use this simple 3-pin device which outputs a voltage proportional to the temperature. For a given voltage V (in mV) the temperature t in deg C is given by:

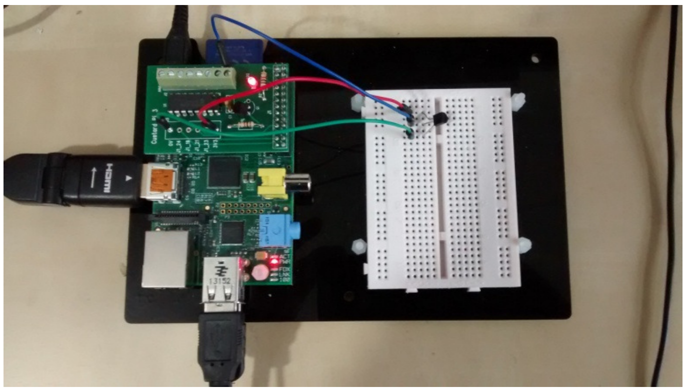
t = (V-500)/10



Custard Pi 3: The Raspberry Pi GPIO does not have any analogue inputs so we are going to use the Custard Pi 3 which has 8 analogue inputs.



For ease of assembly, this project uses a Custard Pi base and a prototyping board. The assembled hardware is shown below.



The red wire is soldered to the 3.3V via hole on the CPi3 and goes to pin 1 of the TMP36. The green wire goes from 0V via hole on the CPi3 and goes to pin 3 of the TMP36. The middle pin (pin 2) of the TMP36 goes to the input labelled 1.

Software

To make life easier, we are going to use a routine called cpi3x.py to read the voltage from any of the 8 inputs available on the Custard Pi 3. This has been provided by SF Innovations and is available on their website under the 'Downloads' tab. A full listing of cpi3x is also provided in appendix A.

By using the function 'readanalog(y)' the program for reading temperature is very simple. The Python code below reads the voltage on channel 0, converts it to a temperature and prints the result.

```
import RPi.GPIO as GPIO
from time import sleep
import cpi3x
GPIO.setmode(GPIO.BOARD)

read=cpi3x.readanalog

while True:

    V = read(0)
    V = round(V,3)
    print V
    t = (1000*V-500)/10
    print "Ch 0 temperature = ", t

    print
    sleep (1)
```

(Download this code from here: http://www.sf-innovations.co.uk/downloads)

The value V is rounded off to 3 decimal places and then converted to temperature. (V is in volts, so we first multiply by 1000 to convert it to mV). This routine will run continuously with a 1 second pause between measurements until terminated by CTRL C.

It should be possible also to run the TMP36 on a long cable (a few metres) and get reasonable results as the voltage output is just under 1V at room temperature and will not be affected too much by noise. However the supply voltage may be an issue, so a 100uF capacitor placed close to the TMP36 across pins 1 and 3 (the supply pins) will help stabilise this.

Ideas for advanced projects

Log the measured temperature to a file along with real time to find how this varies over the day.

Build a 2- channel temperature logger to measure inside and outside temperature. (Pot the TMP36 and capacitor inside a potting box to seal it from moisture).

Use the temperature measurement and a heater connected to the Mains Switch Widget to keep the temperature in the room within a specified range.

Notes:

The Custard Pi 3, Custard Base and Mains Switch Widget are available from amazon.co.uk and directly from SF Innovations. The code presented here can be downloaded from the SF Innovations website.

Appendix A - cpi3x

The function 'readanalog(y)' in this routine returns a value in volts for channel y.

(Download this code from here: http://www.sf-innovations.co.uk/downloads)

```
#1/usr/bin/env python
#Program to read an analogue input on Custard Pi 3
#www.sf-innovations.co.uk

import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)

GPIO.setup(24, GPIO.OUT)     #pin 24 is chip enable
GPIO.setup(23, GPIO.OUT)     #pin 23 is clock
GPIO.setup(19, GPIO.OUT)     #pin 19 is data out
GPIO.setup(21, GPIO.IN, pull_up_down = GPIO.PUD_UP)      #pin 21 is data in

#set pins to default state
GPIO.output(24, True)
GPIO.output(23, False)
GPIO.output(19, True)

#set address channels 0 to 7
#1st bit selects single/differential
#2nd bit channel address
#3rd bit channel address
#4th bit channel address
#5th bit 1 bit delay for data
#6th bit 1st null bit of data

def readanalog(y):
    GPIO.output(19, True)
    #select the right channel
    if y == 0:
        word= [1 ,1, 0, 0, 0, 1, 1]     #set channel 0
    if y == 1:
        word= [1 ,1, 0, 0, 1, 1, 1]     #set channel 1
    if y == 2:
        word= [1 ,1, 0, 1, 0, 1, 1]     #set channel 2
    if y == 3:
        word= [1 ,1, 0, 1, 1, 1, 1]     #set channel 3
    if y == 4:
        word= [1 ,1, 1, 0, 0, 1, 1]     #set channel 4
    if y == 5:
        word= [1 ,1, 1, 0, 1, 1, 1]     #set channel 5
    if y == 6:
        word= [1 ,1, 1, 1, 0, 1, 1]     #set channel 6
    if y == 7:
        word= [1 ,1, 1, 1, 1, 1, 1]     #set channel 7
    if y == 8:
        word= [1 ,0, 0, 0, 0, 1, 1]     #set diff ch0 +ve ch1 -ve
    if y == 9:
        word= [1 ,0, 0, 0, 1, 1, 1]     #set diff ch0 -ve ch1 +ve
    if y == 10:
        word= [1 ,0, 0, 1, 0, 1, 1]     #set diff ch2 +ve ch3 -ve
    if y == 11:
        word= [1 ,0, 0, 1, 1, 1, 1]     #set diff ch2 -ve ch3 +ve
    if y == 12:
```

```
    word= [1 ,0, 1, 0, 0, 1, 1]     #set diff ch4 +ve ch5 -ve
if y == 13:
    word= [1 ,0, 1, 0, 1, 1, 1]     #set diff ch4 -ve ch5 +ve
if y == 14:
    word= [1 ,0, 1, 1, 0, 1, 1]     #set diff ch6 +ve ch7 -ve
if y == 15:
    word= [1 ,0, 1, 1, 1, 1, 1]     #set diff ch6 -ve ch7 +ve

GPIO.output(24, False)  #enable chip
anip=0  #clear variable

#clock out 7 bits to select channel
for x in range (0,7):
    GPIO.output(19, word[x])
    time.sleep(0.01)
    GPIO.output(23, True)
    time.sleep(0.01)
    GPIO.output(23, False)

 #clock in 11 bits of data
for x in range (0,12):
    GPIO.output(23,True)    #set clock hi
    time.sleep(0.01)
    bit=GPIO.input(21)      #read input
    time.sleep(0.01)
    GPIO.output(23,False)   #set clock lo
    value=bit*2**(12-x-1)   #work out value of this bit
    anip=anip+value         #add to previous total
    #print (bit,value,anip)
GPIO.output(24, True)       #disable chip

volt = anip*2.5/4096        #use ref voltage of 2.5 to work out voltage
#volt = ("%.2f" %round(volt,2))  #round to 2 decimal places
#print "voltage ch", y, volt #print to screen
return volt
```