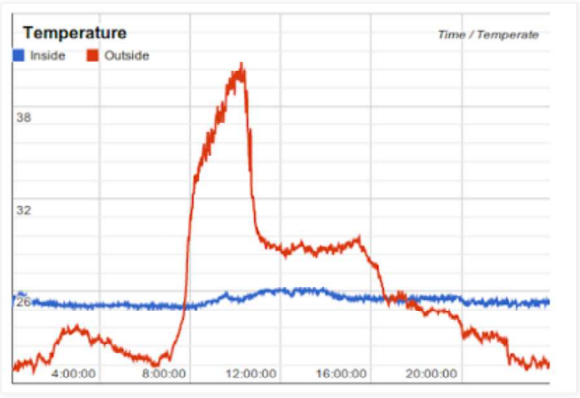
Paul's Projects

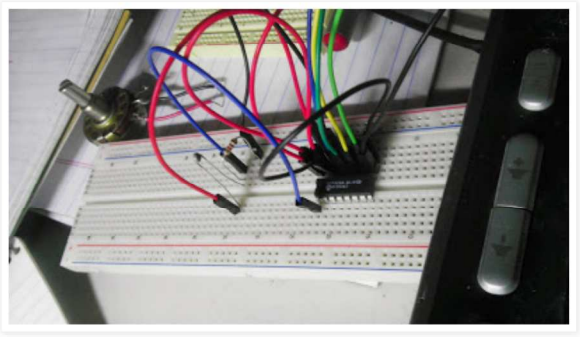
Friday, August 16, 2013

Monitoring Temperatures Using a Raspberry Pi and a MCP3208 ADC with Thermistors



After my Raspbery Pi Internet LED Control project, I wanted to try reading sensors from my room on to the internet. The only sensors I had around were thermistors, a type of resistor that changes resistance based on the temperature.

Thermistors are analog sensors, and the Raspberry Pi does not have an analog-to-digital converters (ADC) built in. An external ADC is needed to measure analog sensors. Most tutorials use the MCP3008 chip for this, but I had the MCP3208, which is just an MCP3008 with higher resolution. I found a great tutorial to measure temperature using the MCP3008 with a more modern temperature sensor.



MCP3208 and thermistor connected

Pinout

MCP 3208 Pin	Pi GPIO Pin #	Pi Pin Name
16 VDD	1	3.3 V
15 VREF	1	3.3 V
14 AGND	6	GND
13 CLK	23	GPIO11 SPI0_SCLK
12 DOUT	21	GPIO09 SPI0_MISO
11 DIN	19	GPIO10 SPI0_MOSI
10 CS	24	GPIO08 CE0
9 DGND	6	GND

Pinout to connect Pi to MCP3208

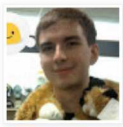
The tutorial code works fine for modern sensors, but I had to adjust it to work with a thermistor. I needed to be able to calculate the resistance of the thermistor using the ADC, so I created a voltage divider with the 1K ohm thermistor and a 1K ohm resistor.

Pages
- Home
- Projects

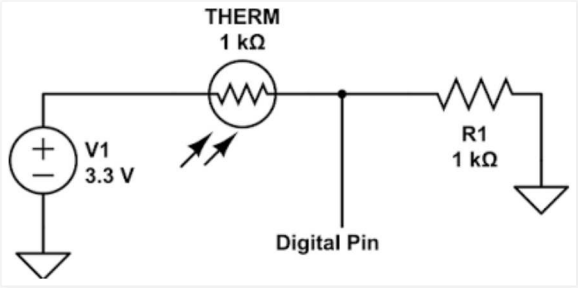
Blog Archive
- ▶ 2014 (4)
- ▼ 2013 (9)
  - ▶ December (1)
  - ▶ September (1)
  - ▼ August (2)
    - Monitoring Temperatures Using a Raspberry Pi and a...
    - Shower Radio Repair
  - ▶ June (2)
  - ▶ March (2)
  - ▶ February (1)
- ▶ 2012 (3)

Networks
- paulschow on Github
- @weirdguy on Twitter
- Paul Schow on Facebook
- Paul Schow on Google+
- Paul Schow on LinkedIn

About Me



Paul Schow
Follow 157
View my complete profile



Voltage divider circuit. I'm using the photoresistor symbol for the thermistor

This circuit allows me to read the voltage and calculate resistance of the thermistor, and is simple to implement in python.

```
value = readadc(adc) #read the adc
volts = (value * 3.3) / 1024 #calculate the voltage
ohms = ((1/volts)*3300)-1000 #calculate the ohms of the thermististor
```

Python code to read thermistor

Unfortunately the reason no one sane uses thermistors anymore is because they have a non-linear relation between temperature and resistance. This means to calculate temperatures from resistance a lookup table has to be used for maximum accuracy. However, there are equations that are close enough for me. I used the Steinhart Hart Equation: Temp = 1/(a + b[ln(ohm)] + c[ln(ohm)]^3) . To complicate it further, I had no idea who made the thermistors I was using, only that they read 1K ohms at 25 degrees celsius. Luckily I found an online thermistor calculator that calculated the values for me. I discovered by comparing values from the calculator to a thermometer in my room that the best curve for my thermistors was curve R (-6.2%/C @ 25C) Mil Ratio X.

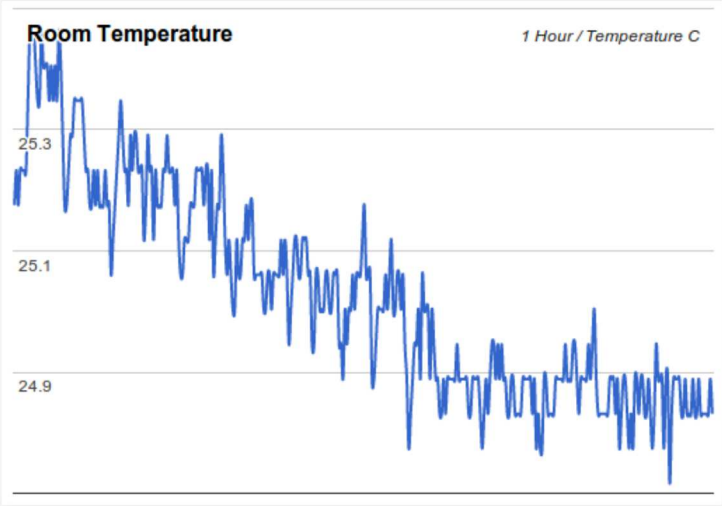
```
#thermistor reading function
def temp_get(adc):
    value = readadc(adc) #read the adc
    volts = (value * 3.3) / 1024 #calculate the voltage
    ohms = ((1/volts)*3300)-1000 #calculate the ohms of the thermististor

    lnohm = math.log1p(ohms) #take ln(ohms)

    #a, b, & c values from http://www.thermistor.com/calculators.php
    #using curve R (-6.2%/C @ 25C) Mil Ratio X
    a =  0.002197222470870
    b =  0.000161097632222
    c =  0.000000125008328

    #Steinhart Hart Equation
    # T = 1/(a + b[ln(ohm)] + c[ln(ohm)]^3)

    t1 = (b*lnohm) # b[ln(ohm)]

    c2 = c*lnohm # c[ln(ohm)]

    t2 = math.pow(c2,3) # c[ln(ohm)]^3

    temp = 1/(a + t1 + t2) #calcualte temperature

    tempc = temp - 273.15 - 4 #K to C
    # the -4 is error correction for bad python math

    #print out info
    print ("%4d/1023 => %5.3f V => %4.1f Ω => %4.1f °K => %4.1f °C from adc %d" %
(value, volts, ohms, temp, tempc, adc))
    return tempc
```

Python function to get ˚C from thermistor

I created a function in python: `temp_get()`. When provided with the value of a pin on the MCP3208, this function calculates and returns the temperature in ˚C.

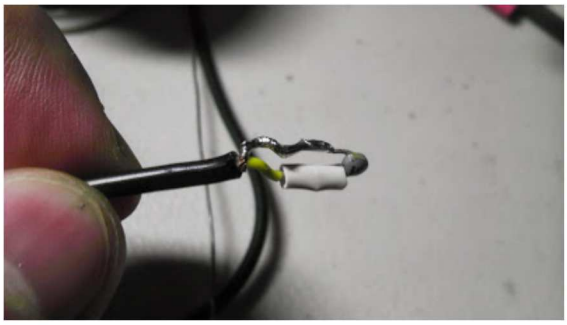
One problem I encountered was python rounding part of the calculation up to 0. The third part of the equation (c[ln(ohm)]^3) was rounded up to 0 instead of the actual value. This threw off all my readings, but I was able to compensate by subtracting 4 from the final value.

Then I created a loop to take temperature readings once a minute and save them in a file, which I can then import to google drive and make a nice graph from.



Recorded room temperature over one hour. Interactive version here

Although this room temperature graph is a cool proof of concept, it's not very exciting or even useful. I added code to record the time of the recording and built a second sensor to go outside.

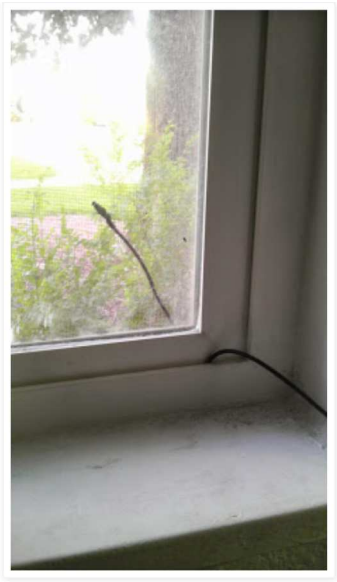


Thermistor attached to cable and soldered in

I picked out a fairly long cable from my parts pile and added a thermistor at the end, then covered it in heat shrink tubing. I measured the resistance in the cable to be 0.1 ohm, not high enough to mess with my readings.

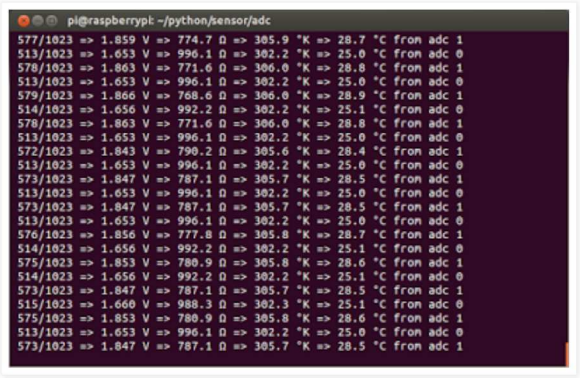
Heat shrinked sensor

Then I took the sensor and placed it under my window so it could read from outside.



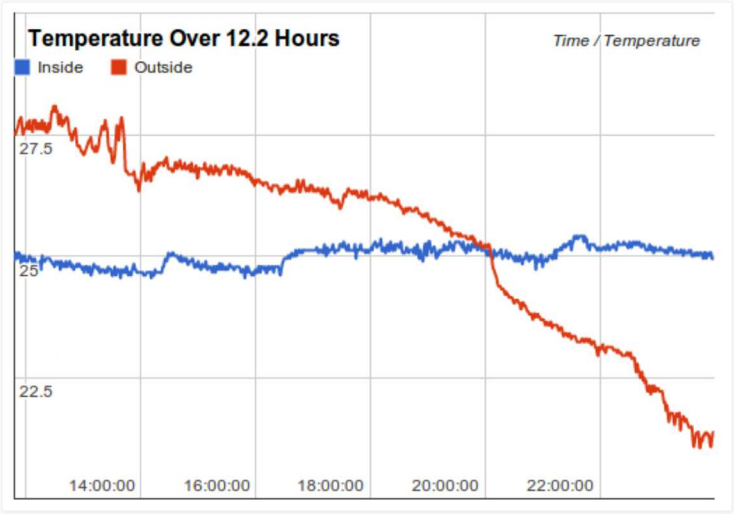
Outdoor sensor in place

I was worried about my window crushing the cable, but it still worked fine. It's not a long-term solution though. The sensor cable is connected to another voltage divider on digital channel 1.



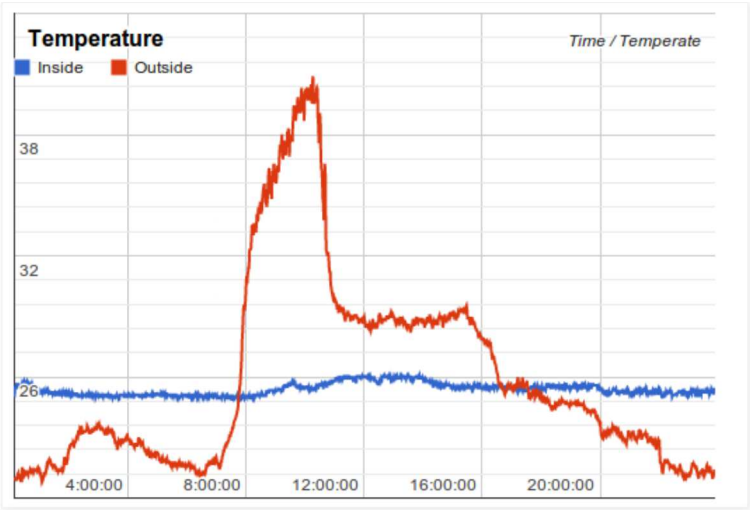
Terminal output of the program

Then I modified the code to record both sensors and the time from `strftime` in the log file.



Recorded outside and inside temperatures over 12.2 hours. Interactive version here

This graph is more exciting, you can see the exact time that the temperature outside gets cooler than the temperature inside. My next goal was to log the temperatures for an entire day.

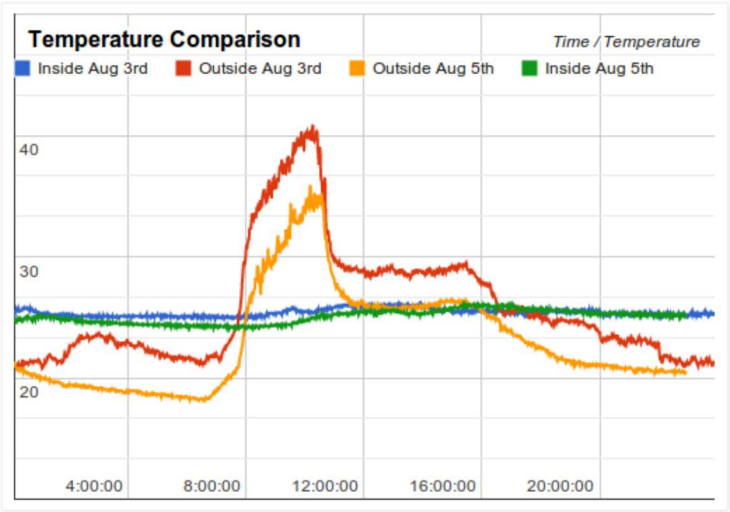


Recorded outside and inside temperatures on August 3rd. Interactive version here

The August 3rd graph is much more interesting, but it has a few anomalies. In the morning around 2:30 the temperature goes up and then goes back down again. This was probably caused by the sprinkler systems outside my window.

More obviously, the huge temperature spike in the morning. No, it does not reach 40˚C when the sun rises only to drop to 28 ˚C a few hours later. This huge spike is caused by the sun hitting the sensor directly and warming it up beyond the actual temperature.

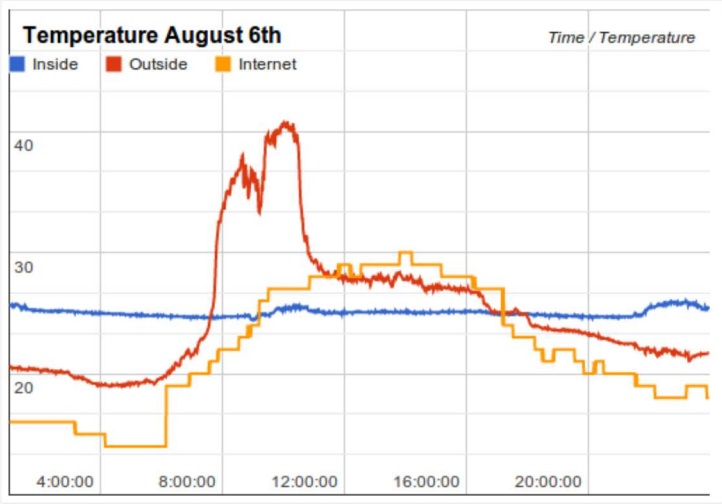
To verify my theories about the analmonies I recorded another day of temperatures.



Comparison of recorded temperatures on August 3rd and August 5th. Interactive version here

When the two graphs are compared, it is easy to see that the 2:30 anomaly is gone, but the sun anomaly is still present.

To verify my theory about the sun anomaly, I needed another source of temperature. Ideally I would have place another sensor outside in the shade all day, but that was not practical for me. Instead I used python-weather-api to get the temperature from weather.com and then record it along with my temperatures.



Comparison of recorded and internet temperatures on August 6th. Interactive version here

The weather.com temperatures verified my sun theory, while still follwing the same general line. Also, the internet temperature gets a lot lower than my recorded temperatures, which indicates that my sensor is still being heated through the window.

There is something strange in the August 6th graph, the sun temperature goes down for a bit before going back up. This was probably caused by a car parking in front of my window and shading the sensor from the sun, then driving off later.

I'm still working on this project, with aims to use the Pi as a webserver and display temperatures and graphs automatically. Maybe I will even build a wireless temperature sensor and put it somewhere shady.

For this project I put the code in a git repository on bitbucket:
https://bitbucket.org/pschow/rpiadctherm/overview

The script for recording the temperatures is basiclogmcp.py

Note: Even with this data my roommates still can't understand why to keep the windows closed during the day and open at night. Oh well.

Posted by Paul Schow at 5:55 PM

3 comments:

Abhiram September 27, 2013 at 6:02 AM

Great Post. Could you eloberate a little on the log portion?

Reply

Paul Schow September 27, 2013 at 4:49 PM

Thanks! The problem is with the part of the equation that multiplies 0.000000125008328 (C) by the natural log of the resistance. In python it always comes out to zero, even though it should actually be something like 7.6687768*10^-7. There's probably a good way to fix it, but I never figured it out.

Reply

John Wayne December 5, 2013 at 9:34 AM

Consider referencing this if you ever go wireless with your sensor. Its a great tutorial using a thermoistor and XRF wireless transcievers!

http://openmicros.org/index.php/articles/79-computer-projects/241-python-thermostat-control

Reply

Enter your comment...

Comment as: Select profile...

Publish Preview

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)