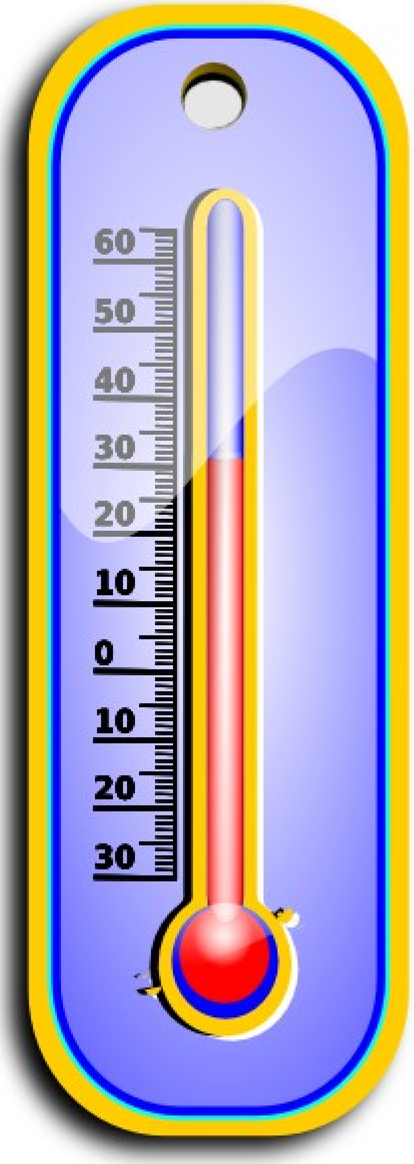
Raspberry PI 'How-To' Series

AM2302-DHT11/22 Temperature Sensors



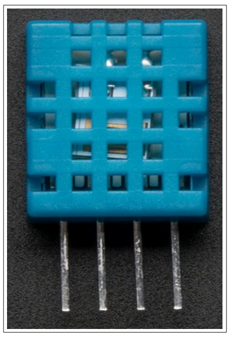
Written by: *Sopwith*
Revision 1.0
February 8, 2015
sopwith@ismellsmoke.net

Introduction

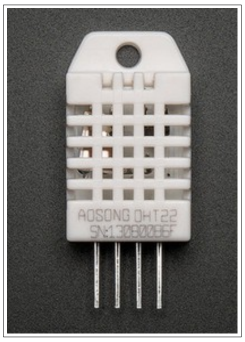
A very common use of the Raspberry Pi is interfacing with sensors. A favorite project among many Pi enthusiasts is to build a weather station. There are many digital temperature/humidity sensors on the market that work with the PI.

I wrote up a previous '*How-To*' document for the Aosong AM2315 temperature/humidity sensor. You can find it here: http://sopwith.ismell smoke.net/?p=46

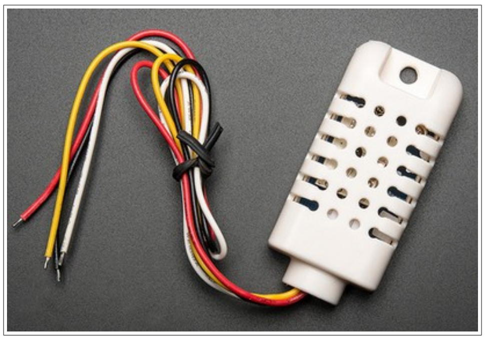
In this '*How-To*' I will walk through the process of interfacing the AM2302, DHT11, and DHT22 temperature sensors with the Pi. These sensors are available from *Adafruit*, *Sparkfun*, and other vendors.



DHT11



DHT22



AM2302

All three of the sensors work exactly the same. The only differences are form factor and accuracy. The DHT11 is the cheapest of the bunch, but it also the least accurate. If you need more precision use the DHT22 or AM2302. The only difference here is the packaging. The AM2302 is in a rugged plastic container and it connects with wires instead of pins.

The datasheets for the devices can be found here:

DHT11 DHT22 AM2302

Challenges

There are a few important points to know about these devices. One of the good things is they only need three wires to operate. V+, Gnd, and a single data pin. This makes them very easy to wire up.

The bad news here is the devices use a non-standard digital interface protocol. The single data wire requires the Pi to send a 'read' request to the sensor; then the sensor sends back the temperature/humidity data in 41 pulses. A long pulse indicates a binary '1' and a short pulse is a binary '0'. It is not important for you to understand the specific protocol details.

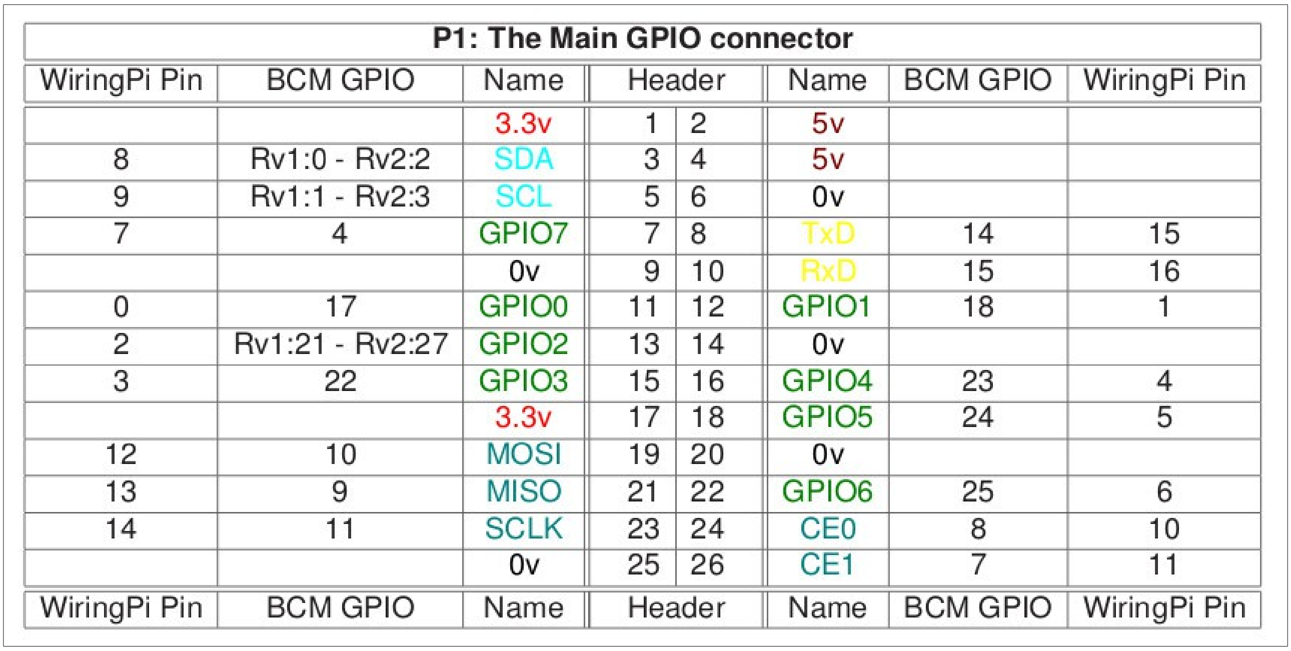
What is important to know is the Pi is not well equipped to interface with these types of devices. Why? Because the Linux operating system on the Pi is not a 'real-time' OS. Unlike the Arduino, the Pi uses a multi-process and multi-threaded operating system that time-slices to make it appear as if many processes are running at the same time. This means the Pi spends a lot of time switching between different processes.

The Arduino is much closer to a 'real-time' device since it focuses on a single specific task and does it very fast. In short, the Arduino and similar devices are able to accurately transmit and receive binary pulses from sensors with very precise timing. The Pi is not able to do this well.

The takeaway here is the fact a lot of work has to go in to writing software for the Pi that works with fast devices like these temperature sensors. In fact, raw Python code is completely unable to communicate with these devices because it is not fast enough. For this reason, most of the code that 'talks' to these devices is written in C and Python programs are wrapped around a C interface. These interfaces are not perfect and you will sometimes receive errors when reading these devices.

Wiring

There are good tutorials on how to wire up one of these temperature devices to a Pi on the *Adafruit* and *Sparkfun* web sites and other places. Below is a great diagram of the Pi GPIO pins that can guide you where to connect the temperature sensor data pin.



P1: The Main GPIO connector							
WiringPi Pin	BCM GPIO	Name	Header		Name	BCM GPIO	WiringPi Pin
		3.3v	1	2	5v		
8	Rv1:0 - Rv2:2	SDA	3	4	5v		
9	Rv1:1 - Rv2:3	SCL	5	6	0v		
7	4	GPIO7	7	8	TxD	14	15
		0v	9	10	RxD	15	16
0	17	GPIO0	11	12	GPIO1	18	1
2	Rv1:21 - Rv2:27	GPIO2	13	14	0v		
3	22	GPIO3	15	16	GPIO4	23	4
		3.3v	17	18	GPIO5	24	5
12	10	MOSI	19	20	0v		
13	9	MISO	21	22	GPIO6	25	6
14	11	SCLK	23	24	CE0	8	10
		0v	25	26	CE1	7	11
WiringPi Pin	BCM GPIO	Name	Header		Name	BCM GPIO	WiringPi Pin

Courtesy of http://wiringpi.com

You can safely connect the sensor to any of the pins shown in green above (GPIO 0-7). When you use the software in this 'How-To', you will refer to the data pin by the *BCM GPIO* number listed in the column next to '*Name*' above. **NOTE:** There are some differences in the pin-outs of the different Pi revisions so make sure you are connecting to the correct pins on your device.

For example, I connected my AM2302 sensor to the Pi GPIO1 pin. For software purposes this is known as Pin-18.

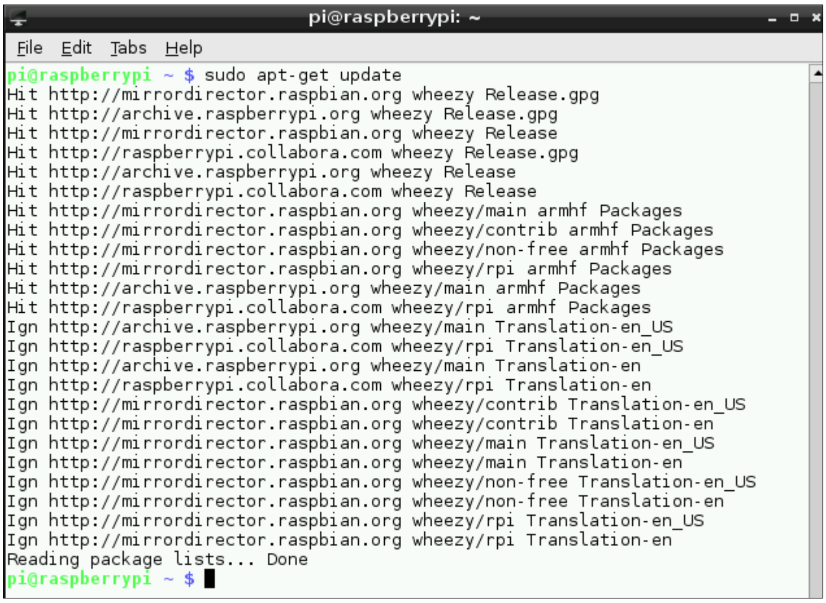
Step-by-Step

Let's get to work getting your sensor up and running.

1) Be sure your Pi has all the latest patches

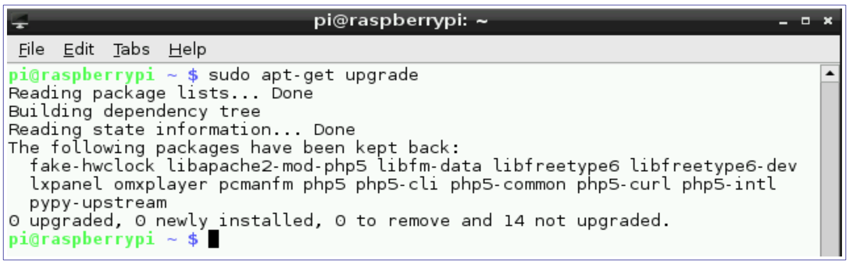
Using a terminal window enter the following commands:

$ sudo apt-get update



```
pi@raspberrypi ~ $ sudo apt-get update
Hit http://mirrordirector.raspbian.org wheezy Release.gpg
Hit http://archive.raspberrypi.org wheezy Release.gpg
Hit http://mirrordirector.raspbian.org wheezy Release
Hit http://raspberrypi.collabora.com wheezy Release.gpg
Hit http://archive.raspberrypi.org wheezy Release
Hit http://raspberrypi.collabora.com wheezy Release
Hit http://mirrordirector.raspbian.org wheezy/main armhf Packages
Hit http://mirrordirector.raspbian.org wheezy/contrib armhf Packages
Hit http://mirrordirector.raspbian.org wheezy/non-free armhf Packages
Hit http://mirrordirector.raspbian.org wheezy/rpi armhf Packages
Hit http://archive.raspberrypi.org wheezy/main armhf Packages
Hit http://raspberrypi.collabora.com wheezy/rpi armhf Packages
Ign http://archive.raspberrypi.org wheezy/main Translation-en_US
Ign http://raspberrypi.collabora.com wheezy/rpi Translation-en_US
Ign http://archive.raspberrypi.org wheezy/main Translation-en
Ign http://raspberrypi.collabora.com wheezy/rpi Translation-en
Ign http://mirrordirector.raspbian.org wheezy/contrib Translation-en_US
Ign http://mirrordirector.raspbian.org wheezy/contrib Translation-en
Ign http://mirrordirector.raspbian.org wheezy/main Translation-en_US
Ign http://mirrordirector.raspbian.org wheezy/main Translation-en
Ign http://mirrordirector.raspbian.org wheezy/non-free Translation-en_US
Ign http://mirrordirector.raspbian.org wheezy/non-free Translation-en
Ign http://mirrordirector.raspbian.org wheezy/rpi Translation-en_US
Ign http://mirrordirector.raspbian.org wheezy/rpi Translation-en
Reading package lists... Done
pi@raspberrypi ~ $ █
```

$ sudo apt-get upgrade

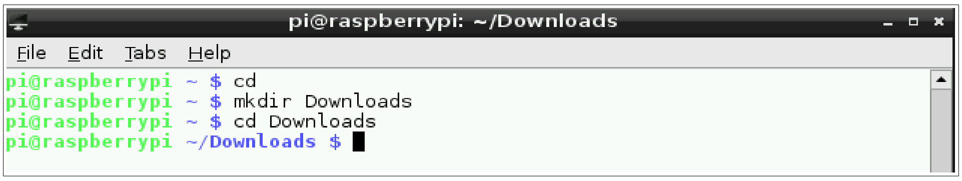


```
pi@raspberrypi ~ $ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages have been kept back:
  fake-hwclock libapache2-mod-php5 libfm-data libfreetype6 libfreetype6-dev
  lxpanel omxplayer pcmanfm php5 php5-cli php5-common php5-curl php5-intl
  pypy-upstream
0 upgraded, 0 newly installed, 0 to remove and 14 not upgraded.
pi@raspberrypi ~ $ █
```

As you can see in the above screen shots, my Pi is current with patches. It may take some time for these commands to complete on your system. Be patient. Also, be sure to reboot your Pi after an upgrade.

2) Create a folder for downloads

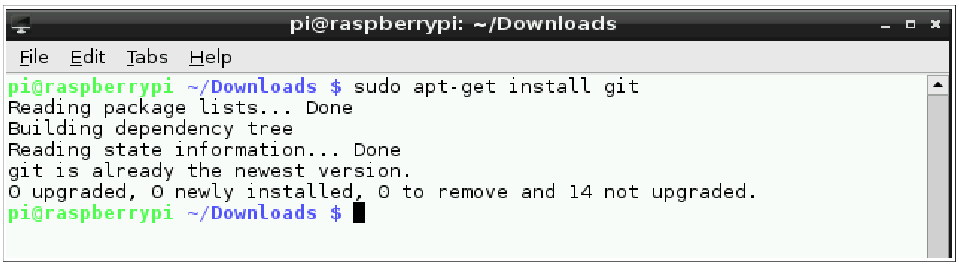
I like to keep my Pi's tidy. I place all downloads in a folder named '*Downloads*.' From your terminal window, make sure you are in your */home* (e.g. /home/pi) directory.
$ cd
$ mkdir Downloads
$ cd Downloads



```
pi@raspberrypi ~ $ cd
pi@raspberrypi ~ $ mkdir Downloads
pi@raspberrypi ~ $ cd Downloads
pi@raspberrypi ~/Downloads $ █
```

3) Install git

Make sure you have the latest version of git.
$ sudo apt-get install git

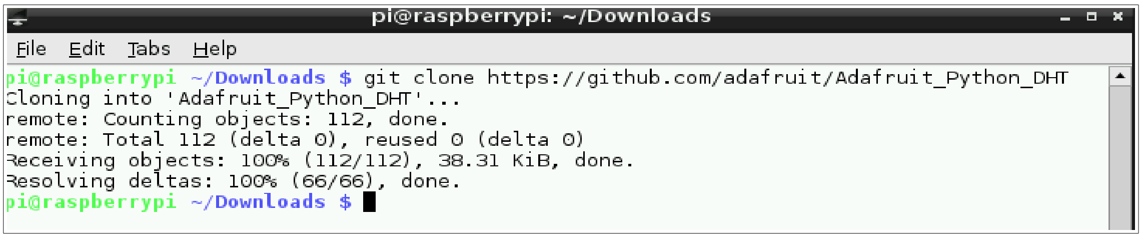


```
pi@raspberrypi ~/Downloads $ sudo apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 14 not upgraded.
pi@raspberrypi ~/Downloads $ █
```

While you are at it, make sure the Python header files are on you Pi.
$ sudo apt-get install python-dev.
(No screen shot provided.)

4) Download the *Adafruit DHT22* library

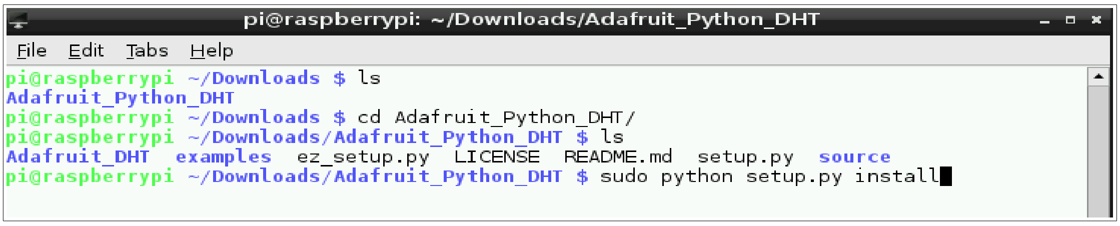
From your terminal window, make sure you are in your *Downloads* directory.
$ git clone https://github.com/Adafruit/Adafruit_Python_DHT



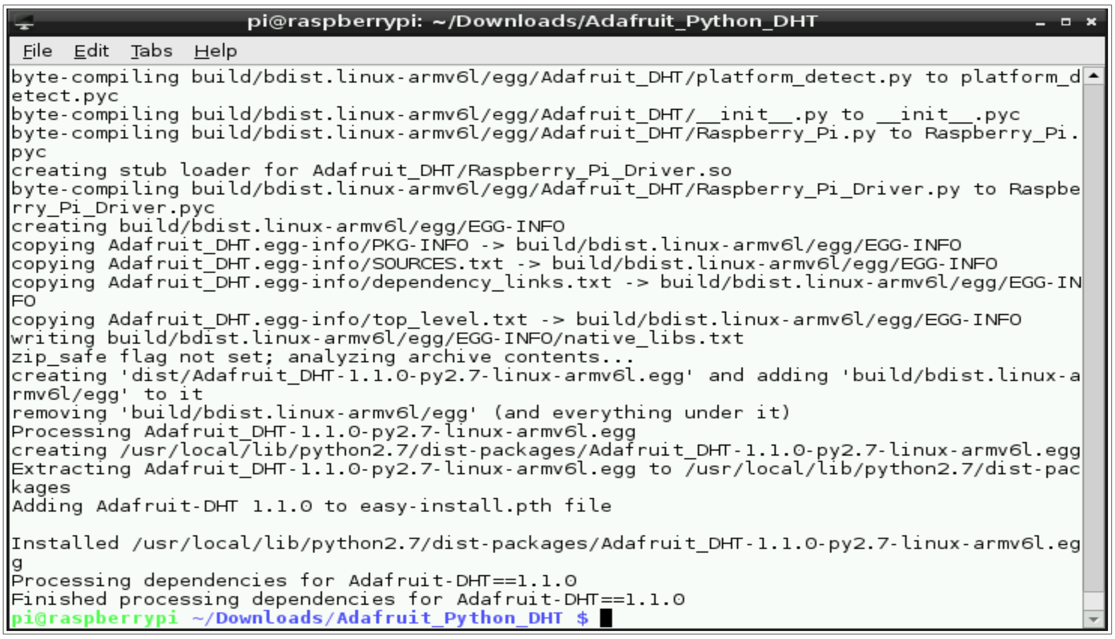
```
pi@raspberrypi ~/Downloads $ git clone https://github.com/adafruit/Adafruit_Python_DHT
Cloning into 'Adafruit_Python_DHT'...
remote: Counting objects: 112, done.
remote: Total 112 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (112/112), 38.31 KiB, done.
Resolving deltas: 100% (66/66), done.
pi@raspberrypi ~/Downloads $ █
```

5) Install the *Adafruit DHT22* library

$ cd Adafruit_Python_DHT
$ sudo python setup.py install



```
pi@raspberrypi ~/Downloads $ ls
Adafruit_Python_DHT
pi@raspberrypi ~/Downloads $ cd Adafruit_Python_DHT/
pi@raspberrypi ~/Downloads/Adafruit_Python_DHT $ ls
Adafruit_DHT  examples  ez_setup.py  LICENSE  README.md  setup.py  source
pi@raspberrypi ~/Downloads/Adafruit_Python_DHT $ sudo python setup.py install
```



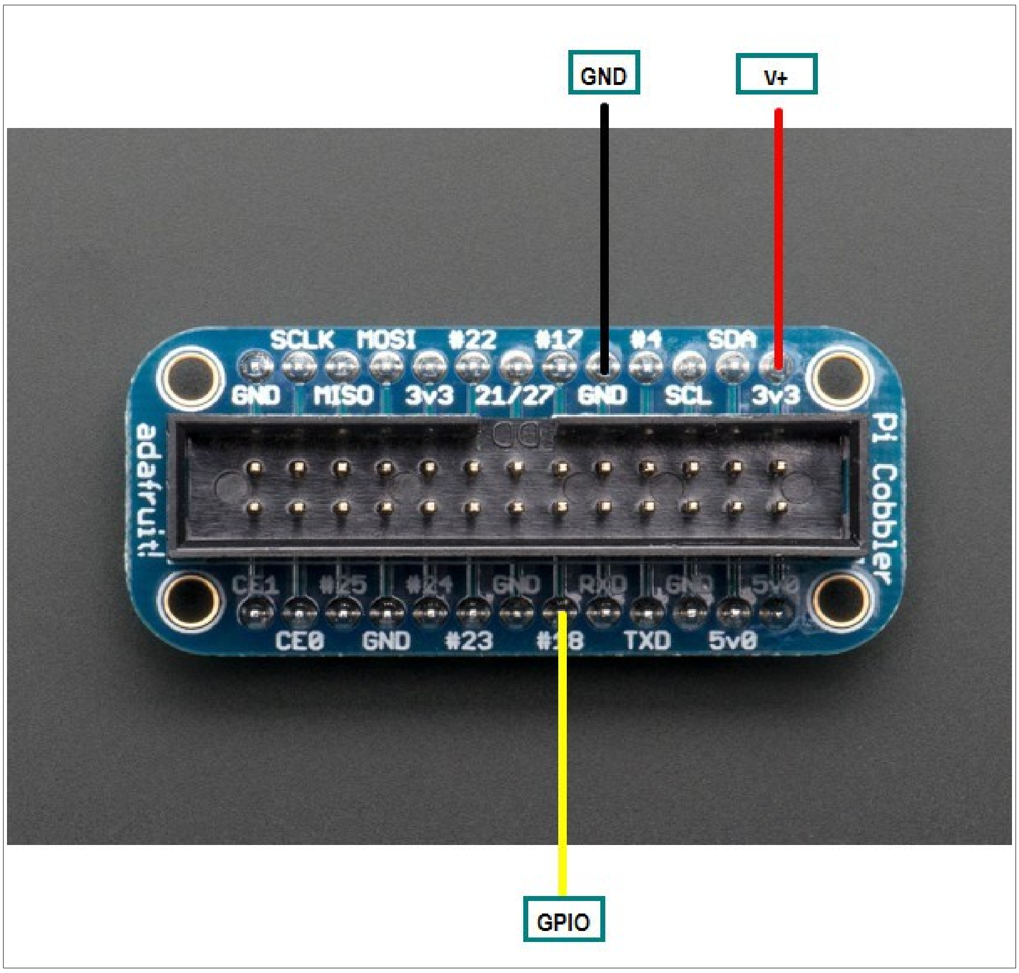
```
byte-compiling build/bdist.linux-armv6l/egg/Adafruit_DHT/platform_detect.py to platform_d
etect.pyc
byte-compiling build/bdist.linux-armv6l/egg/Adafruit_DHT/__init__.py to __init__.pyc
byte-compiling build/bdist.linux-armv6l/egg/Adafruit_DHT/Raspberry_Pi.py to Raspberry_Pi.
pyc
creating stub loader for Adafruit_DHT/Raspberry_Pi_Driver.so
byte-compiling build/bdist.linux-armv6l/egg/Adafruit_DHT/Raspberry_Pi_Driver.py to Raspbe
rry_Pi_Driver.pyc
creating build/bdist.linux-armv6l/egg/EGG-INFO
copying Adafruit_DHT.egg-info/PKG-INFO -> build/bdist.linux-armv6l/egg/EGG-INFO
copying Adafruit_DHT.egg-info/SOURCES.txt -> build/bdist.linux-armv6l/egg/EGG-INFO
copying Adafruit_DHT.egg-info/dependency_links.txt -> build/bdist.linux-armv6l/egg/EGG-IN
FO
copying Adafruit_DHT.egg-info/top_level.txt -> build/bdist.linux-armv6l/egg/EGG-INFO
writing build/bdist.linux-armv6l/egg/EGG-INFO/native_libs.txt
zip_safe flag not set; analyzing archive contents...
creating 'dist/Adafruit_DHT-1.1.0-py2.7-linux-armv6l.egg' and adding 'build/bdist.linux-a
rmv6l/egg' to it
removing 'build/bdist.linux-armv6l/egg' (and everything under it)
Processing Adafruit_DHT-1.1.0-py2.7-linux-armv6l.egg
creating /usr/local/lib/python2.7/dist-packages/Adafruit_DHT-1.1.0-py2.7-linux-armv6l.egg
Extracting Adafruit_DHT-1.1.0-py2.7-linux-armv6l.egg to /usr/local/lib/python2.7/dist-pac
kages
Adding Adafruit-DHT 1.1.0 to easy-install.pth file

Installed /usr/local/lib/python2.7/dist-packages/Adafruit_DHT-1.1.0-py2.7-linux-armv6l.eg
g
Processing dependencies for Adafruit-DHT==1.1.0
Finished processing dependencies for Adafruit-DHT==1.1.0
pi@raspberrypi ~/Downloads/Adafruit_Python_DHT $
```

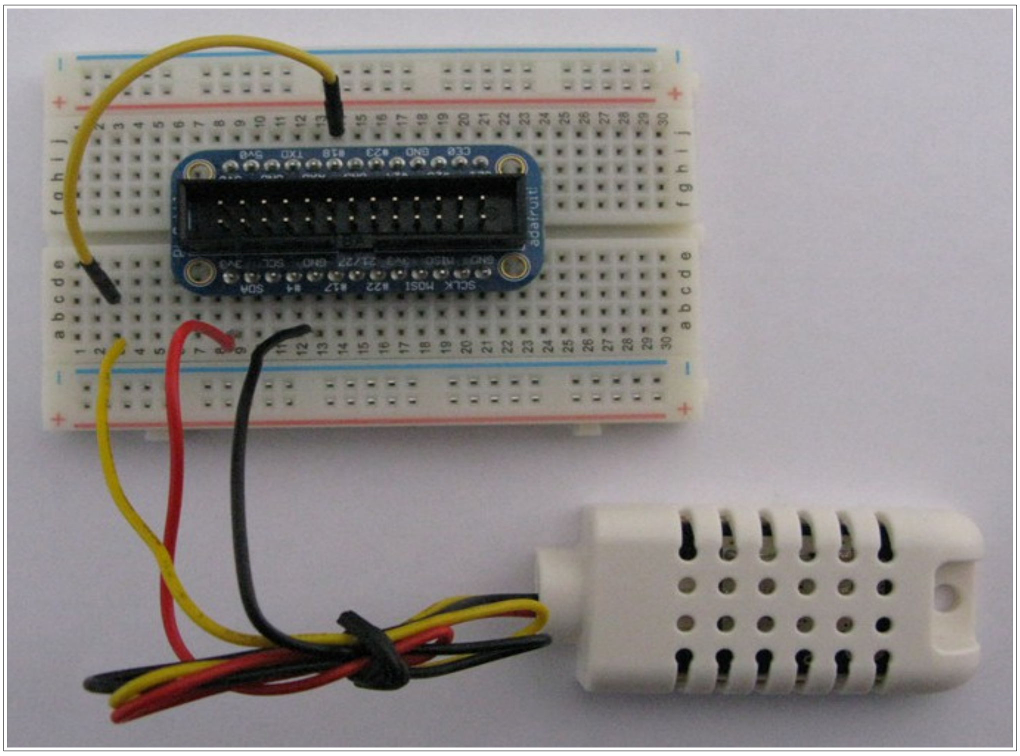
6) Wire up your sensor

If you have not already done so, connect your sensor to your Pi. One of the best investments you can make is to purchase a PiCobbler from *Adafruit*. This simple and inexpensive tool makes it easy to wire up your sensors.

In the photos below you can see that I wired up my AM2302 sensor to my PiCobbler. The red wire connects to the 3.3V pin, the black wire connects to ground, and the yellow GPIO/Data wire is connected to GPIO port 18. If your AM2302 has a white wire it is not used. Notice I used a jumper wire to connect the data pin. This was done to make the photo clearer.

NOTE: The AM2302 contains an internal pullup resistor so there is no need to install one when using this device. This is *not* true with the DHT22 – you need a pullup resistor.


GND
V+
SCLK
MOSI
#22
#17
#4
SDA
GND
MISO
3v3
21/27
GND
SCL
3v3
adafruit!
Pi Cobbler
CE1
#25
#24
GND
RXD
GND
5v0
CE0
GND
#23
#18
TXD
5v0
GPIO



If you are using the DHT11/22 device then you will need to use a 10K pullup resistor to ensure the device works properly. The below image is from *Adafruit* and it show how to wire your DHT11/22 device.

Facing the front of the device (grid side), the left pin (1) is for 5/3.3 volt power. Pin-2 is the data pin and this should be connected to a Pi GPIO pin. Pin-3 is not used. Pin-4 should be connected to ground.

The resistor is connected between Pin-2 of the device (data pin) and the power source. This is shown clearly in the diagram below.

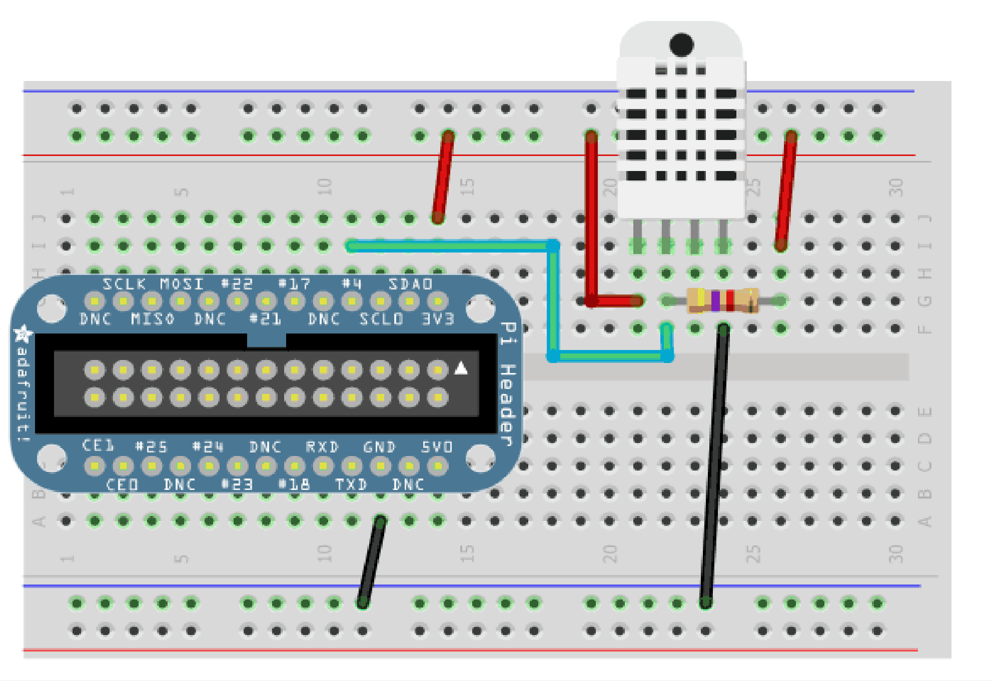
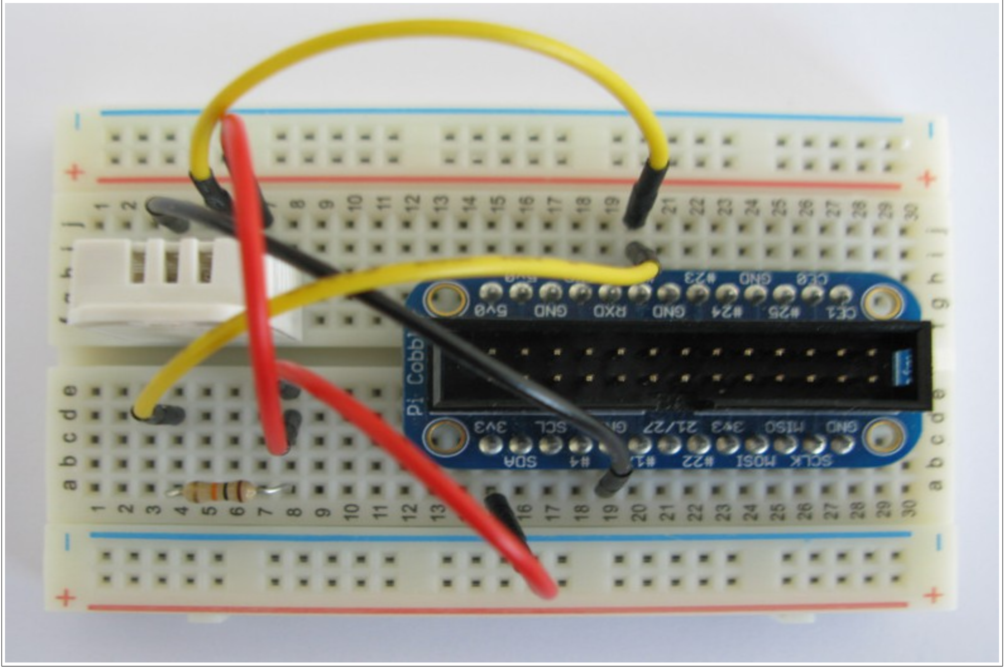

SCLK MOSI #22 #17 #4 SDA0
DNC MISO DNC #21 DNC SCL0 3V3
adafruit
Pi Header
CE1 #25 #24 DNC RXD GND 5V0
CE0 DNC #23 #18 TXD DNC
1
5
10
15
20
25
30
J
I
H
G
F
E
D
C
B
A

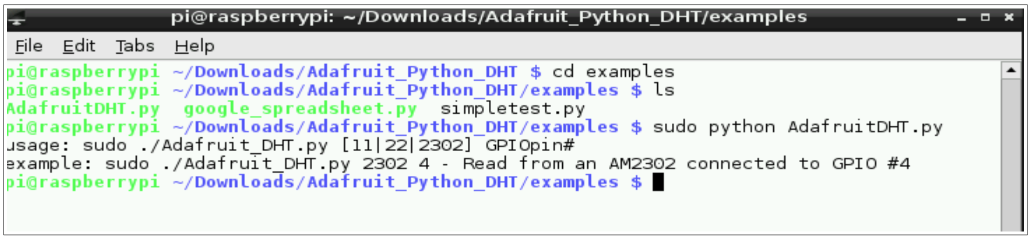
Image courtesy of *Adafruit*

The image below shows the breadboard I wired up with a DHT22 sensor. You can see I used jumper wires, but the wiring is exactly the same as shown in the *Adafruit* diagram except I used a different GPIO pin.



7) Test the device

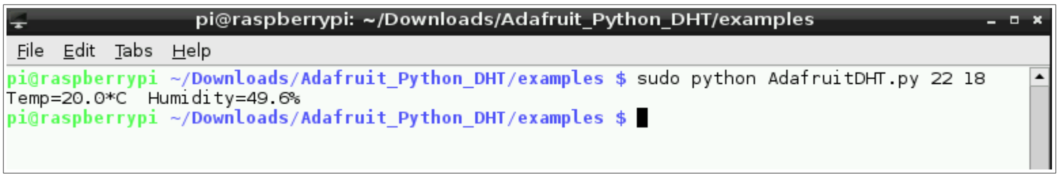
The *Adafruit* DHT22 library includes a test program which is quite handy.



```
pi@raspberrypi ~/Downloads/Adafruit_Python_DHT $ cd examples
pi@raspberrypi ~/Downloads/Adafruit_Python_DHT/examples $ ls
AdafruitDHT.py  google_spreadsheet.py  simpletest.py
pi@raspberrypi ~/Downloads/Adafruit_Python_DHT/examples $ sudo python AdafruitDHT.py
usage: sudo ./Adafruit_DHT.py [11|22|2302] GPIOpin#
example: sudo ./Adafruit_DHT.py 2302 4 - Read from an AM2302 connected to GPIO #4
pi@raspberrypi ~/Downloads/Adafruit_Python_DHT/examples $
```

As you can see in the above screen shot, I changed directories into the *Adafruit_Python_DHT/examples* folder and listed the files there. Then I ran the *AdafruitDHT.py* example program without any command line arguments so I can see the usage output.

This program expects two (2) command line arguments. 1) the device type (11|22|2302), and 2) the GPIO pin your device is connected to. I have a DHT22 connected to GPIO port 18 so I will run the program with the parameters 22 (DHT22) and 18 (GPIO port).



```
pi@raspberrypi ~/Downloads/Adafruit_Python_DHT/examples $ sudo python AdafruitDHT.py 22 18
Temp=20.0*C  Humidity=49.6%
pi@raspberrypi ~/Downloads/Adafruit_Python_DHT/examples $
```

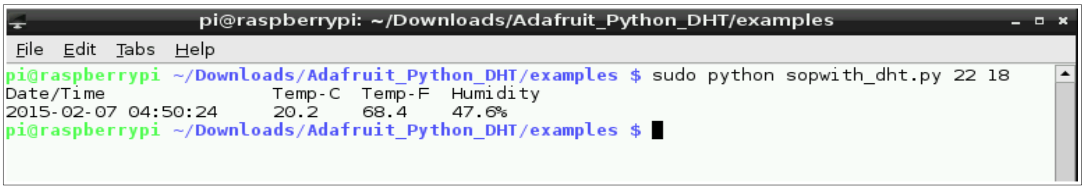
My device is working properly! Great. Due to the challenges describes earlier in this document, there may be times when the application does not return any data. In my experience, this happens very rarely – but you need to be aware of it. Also, the datasheet states the device will not respond to a read request unless 2 seconds have passed since the previous read. If you need to read the temperature more than once every two seconds then these devices will not meet the needs of your project.

NOTE: Once you get a response from your sensor you no longer need the *Adafruit_Python_DHT* folder. The library is now installed on you Pi and is available for use by any Python scripts. Copy the *AdafruitDHT.py* script to another folder for later use.

8) Hack some code

Once you have your device working, no doubt you will want to customise some Python code to make it do what you want. The great thing about the open-source community is the huge codebase that is free to use and modify, and the willingness of others to help you.

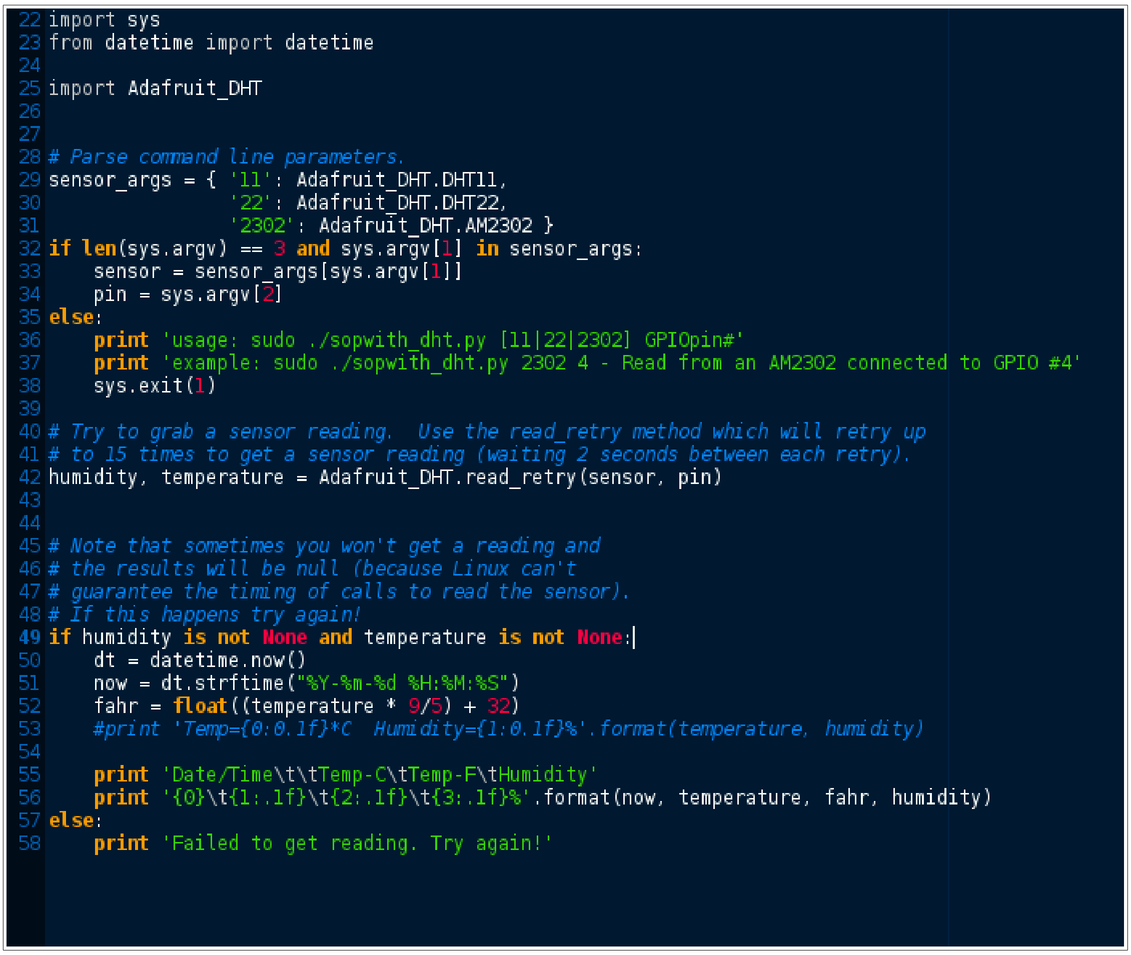
I modified the *AdafruitDHT.py* code to include the date and time of the reading and also provide a Fahrenheit temperature. The output of my code is shown below.



```
pi@raspberrypi ~/Downloads/Adafruit_Python_DHT/examples $ sudo python sopwith_dht.py 22 18
Date/Time               Temp-C  Temp-F  Humidity
2015-02-07 04:50:24     20.2    68.4    47.6%
pi@raspberrypi ~/Downloads/Adafruit_Python_DHT/examples $
```

This code is useful if you want to write the output of your sensor readings to a file. It is tab delimited so the output file can be opened into a nicely formatted worksheet in LibreOffice Calc. I encourage you to learn the basics of Python. It is a marvelous and easy programming language.

For those that are interested, the below Python code is the *AdafruitDHT.py* example script that I modified. I am sure you will agree, this code is quite simple to understand with basic Python skills.



```
22 import sys
23 from datetime import datetime
24
25 import Adafruit_DHT
26
27
28 # Parse command line parameters.
29 sensor_args = { '11': Adafruit_DHT.DHT11,
30                 '22': Adafruit_DHT.DHT22,
31                 '2302': Adafruit_DHT.AM2302 }
32 if len(sys.argv) == 3 and sys.argv[1] in sensor_args:
33     sensor = sensor_args[sys.argv[1]]
34     pin = sys.argv[2]
35 else:
36     print 'usage: sudo ./sopwith_dht.py [11|22|2302] GPIOpin#'
37     print 'example: sudo ./sopwith_dht.py 2302 4 - Read from an AM2302 connected to GPIO #4'
38     sys.exit(1)
39
40 # Try to grab a sensor reading.  Use the read_retry method which will retry up
41 # to 15 times to get a sensor reading (waiting 2 seconds between each retry).
42 humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
43
44
45 # Note that sometimes you won't get a reading and
46 # the results will be null (because Linux can't
47 # guarantee the timing of calls to read the sensor).
48 # If this happens try again!
49 if humidity is not None and temperature is not None:
50     dt = datetime.now()
51     now = dt.strftime("%Y-%m-%d %H:%M:%S")
52     fahr = float((temperature * 9/5) + 32)
53     #print 'Temp={0:0.1f}*C  Humidity={1:0.1f}%'.format(temperature, humidity)
54
55     print 'Date/Time\t\tTemp-C\tTemp-F\tHumidity'
56     print '{0}\t{1:.1f}\t{2:.1f}\t{3:.1f}%'.format(now, temperature, fahr, humidity)
57 else:
58     print 'Failed to get reading. Try again!'
```

Lines 1-21

These are not shown but they only contain the *Adafruit* Copyright information.

Lines 22-25

These are the import declarations needed to access other Python modules.

Lines 29-38

This code handles the command line arguments and the usage screen.

Line 42

This line calls into a C code module to read the sensor. Notice this code will try to read the sensor 15 times in 2-second intervals until it gets a good reading. This means you may be waiting up to 30 seconds to get your results.

Line 49

This important line checks to see if valid data was sent from the sensor.

Lines 50-52

This is code I added. It determines and formats the current system time and it converts the Celsius temperature value to Fahrenheit.

Line 53
This line is the original print statement that I commented out.

Line 55
Prints an output header row. Comment this out if you do not want a header.

Line 56
Prints the tab-delimited output including the time and Fahrenheit reading.

Line 58
Advises if there was an error.

Summary

Hopefully, this 'How-To' has helped you get your sensors up and running. Congratulations! If you have any problems be sure to send me an Email and I will help you in any way I can. Also, be sure to visit my Blog once in a while. I am building a modular weather station kit designed to be used in the education community.

Oh – and one more thing. Share your Pi adventures with a kid.

Sopwith
08-Feb-2015
London, UK
http://sopwith.ismellsmoke.net
sopwith@ismellsmoke.net

NOTE: This document was created entirely on a Raspberry PI using LibreOffice *Writer*. The screen shots were captured with *Scrot.* The screen shots were edited with the *Gimp*. Windows not required.