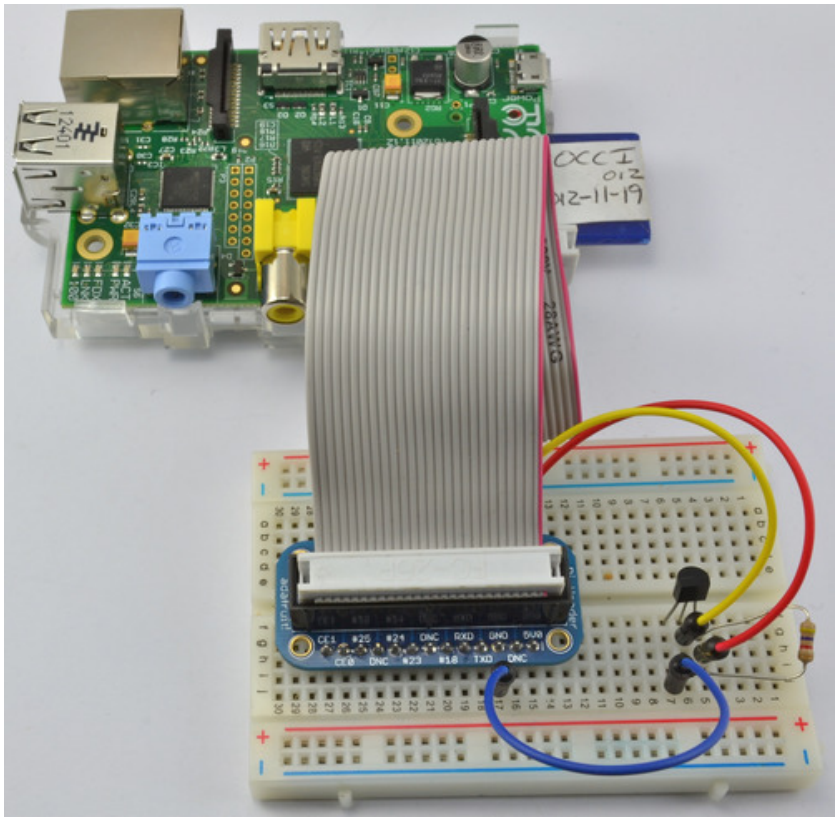
adafruit learning system

Adafruit's Raspberry Pi Lesson 11. DS18B20 Temperature Sensing

Created by Simon Monk



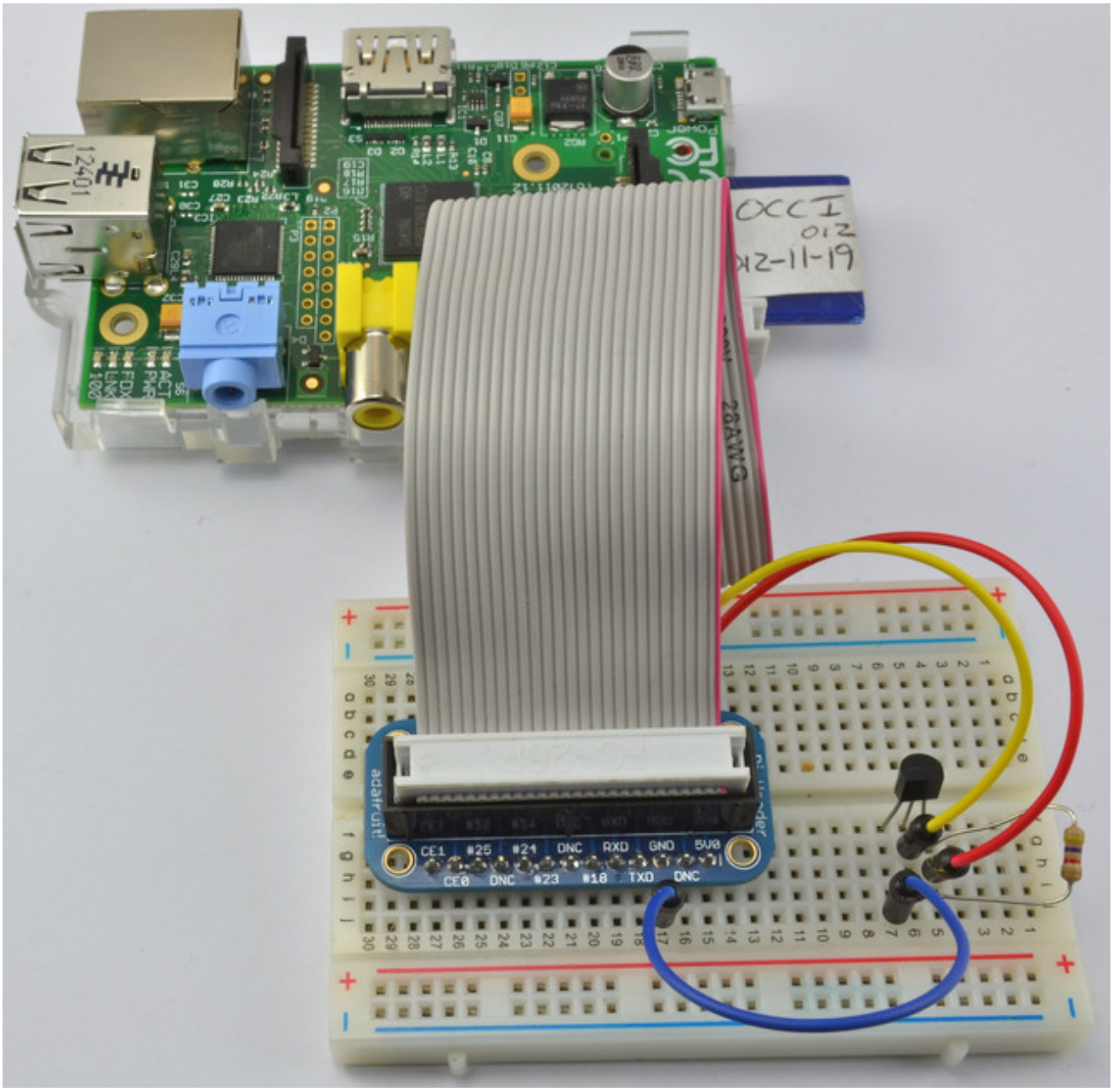
Last updated on 2015-04-09 03:47:48 PM EDT

Guide Contents

Guide Contents 2
Overview 3
Other Code Libraries 4
Parts 5
Hardware 8
DS18B20 12
Add OneWire support 12
Software 14
Configure and Test 17
Adding more sensors 17

Overview

The Occidentalis Linux distribution for Raspberry Pi (and Raspbian as of Dec 2012) includes support for the DS18B20 1-wire temperature sensor. These sensors come in a small three pin package like a transistor and are accurate digital devices.



In this lesson, you will learn how to use a DS18B20 with the Raspberry Pi to take temperature readings.

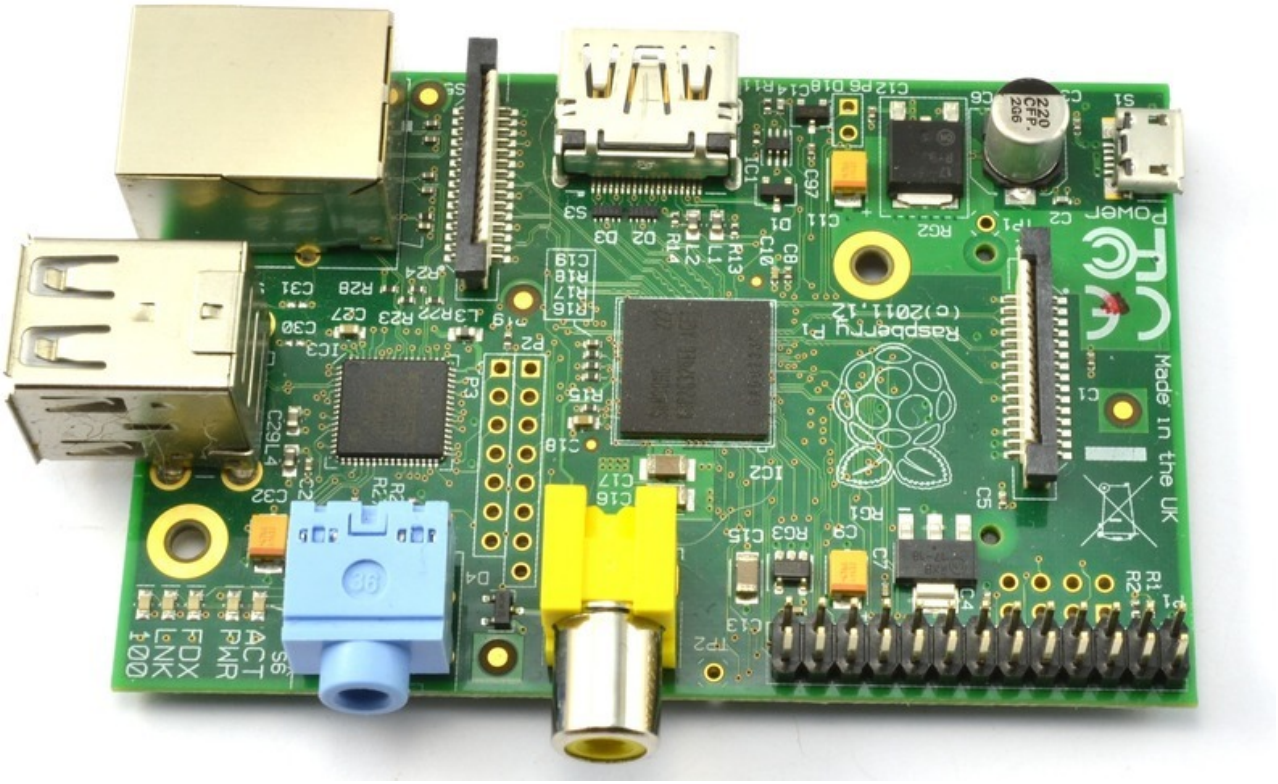
Since the Raspberry Pi has no ADC (Analog to Digital Converter), it cannot directly use an analog temperature sensor like the TMP36, making the DS18B20 a good choice for temperature sensing.

Other Code Libraries

If you are a python-fancier, there's also a nice DS18B20 library that works with raspberry pi over at https://github.com/timofurrer/ds18b20 (http://adafru.it/dpK) (written by Timo!)

Parts

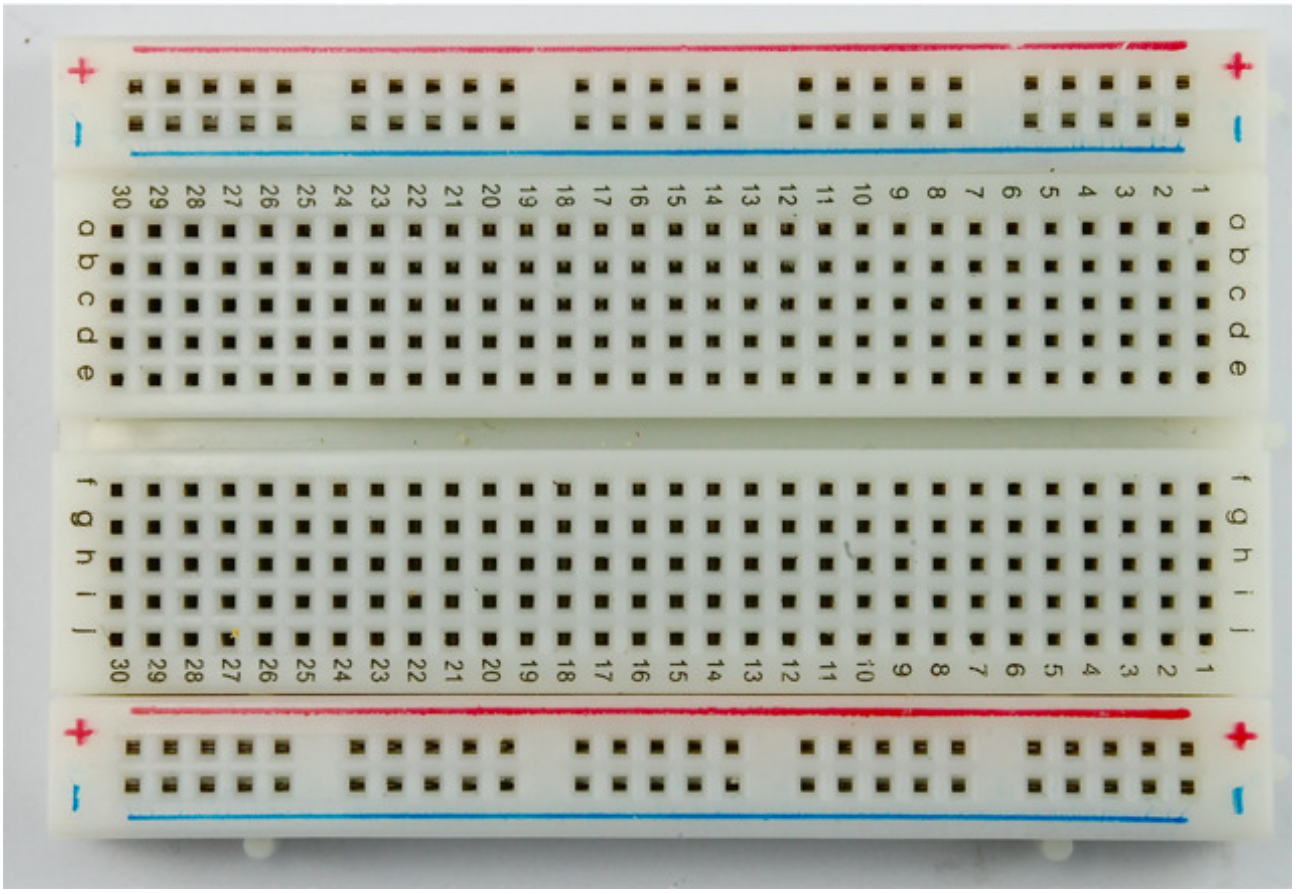
To build the project described in this lesson, you will need the following parts.



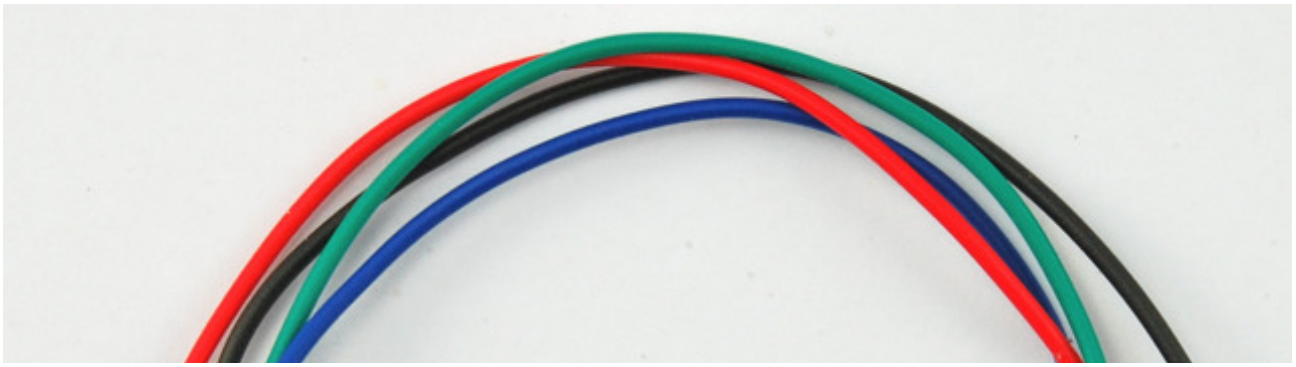
Raspberry Pi

DS18B20 Digital temperature sensor + extras, or waterproof or high temperature versions (see products sidebar)

You'll also need a 4.7K or 10K ohm resistor, it comes with the sensor when purchased from Adafruit



Half-sized breadboard



Jumper wire pack

Pi Cobbler

Hardware

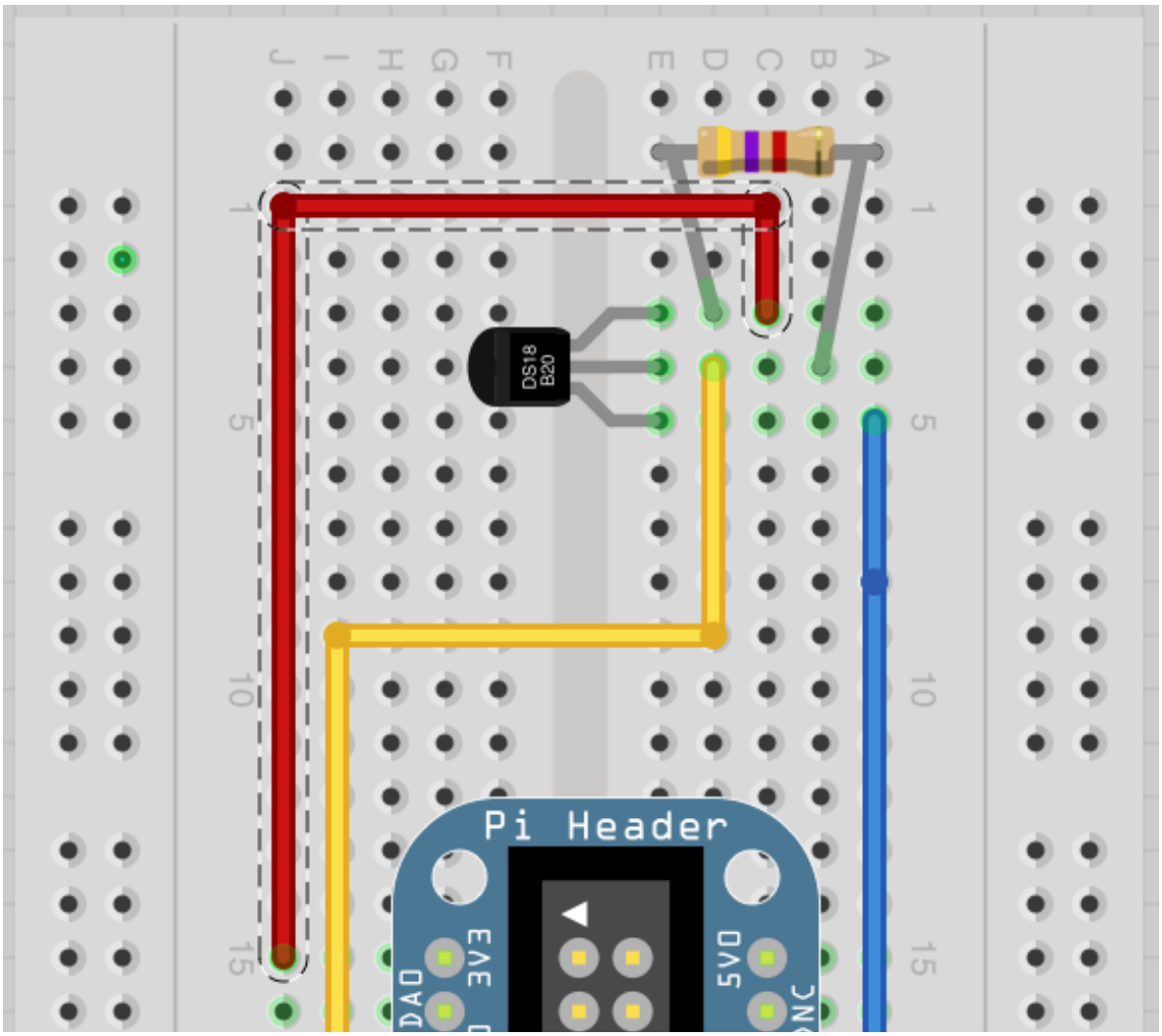
The breadboard layout for just the basic DS18B20 is shown below.

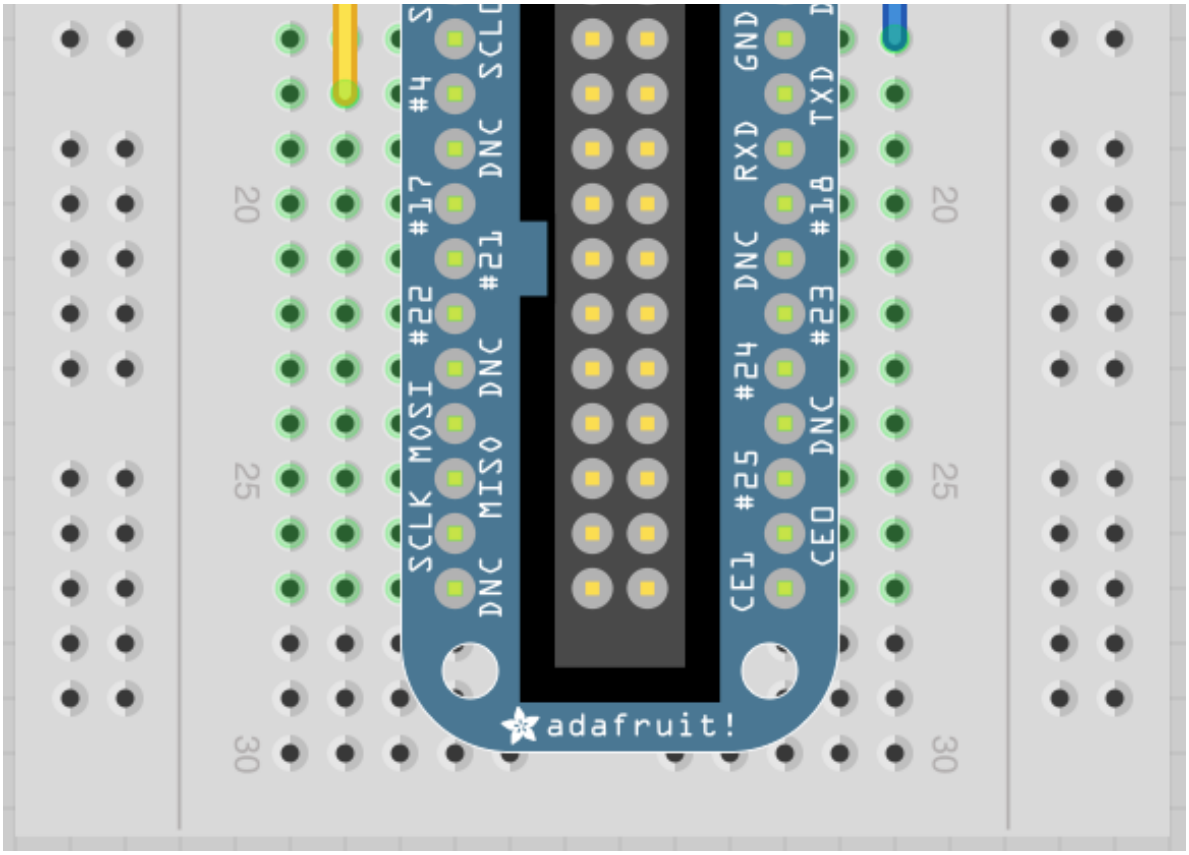
The DS18B20 "1-wire" sensors can be connected in parallel - unlike nearly any other sensor sold! All sensors should share the same pins, but you only need one 4.7K resistor for all of them

The resistor is used as a 'pullup' for the data-line, and is required to keep the data transfer stable and happy

Be careful to get the DS18B20 the right way around. The curved edge should be to the left as shown in the figure below. If you put it the wrong way around, it will get hot and then break.

> Despite both being temperature sensors, the DS18B20+ is totally different than the TMP36. You cannot use a TMP36 for this tutorial!

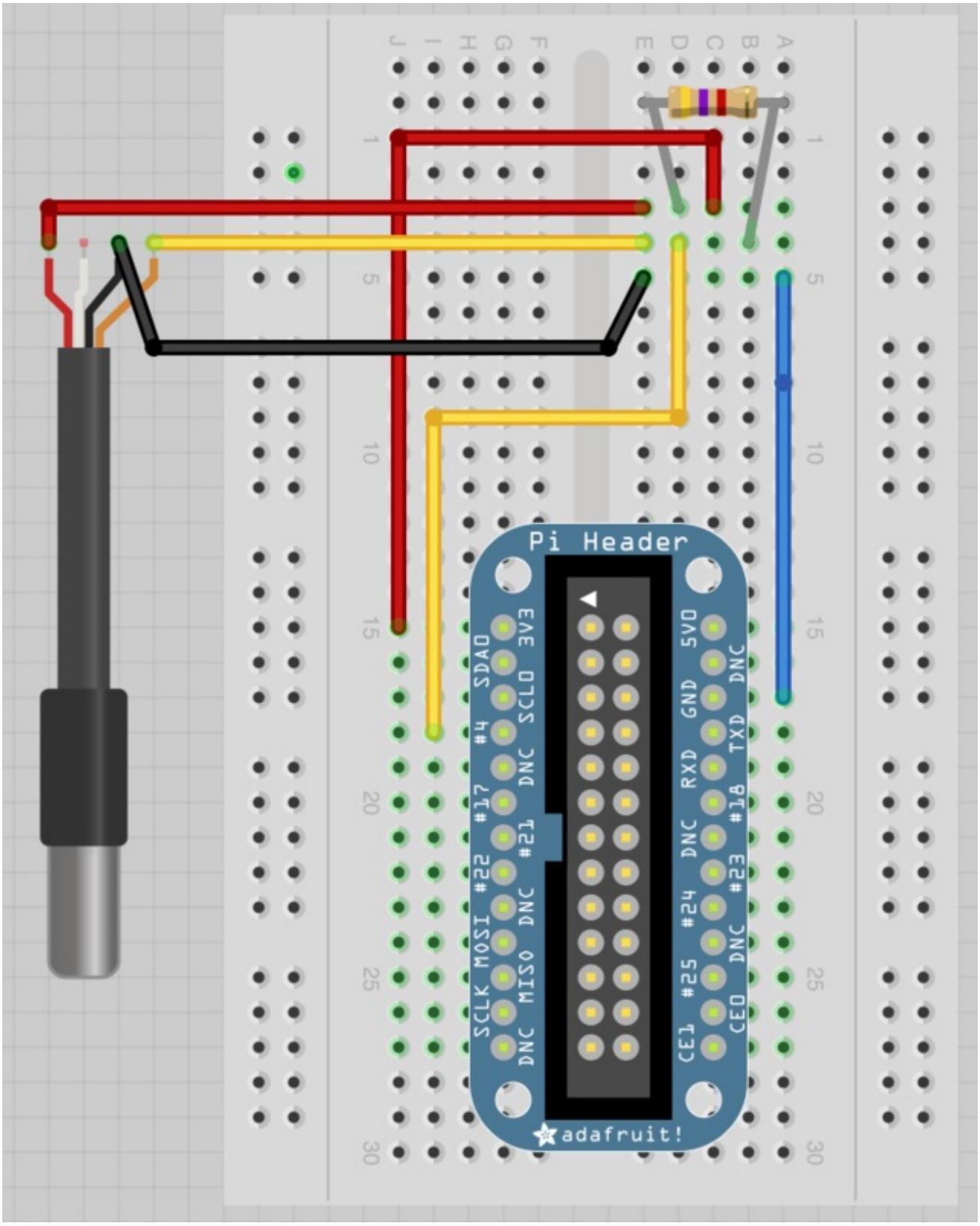


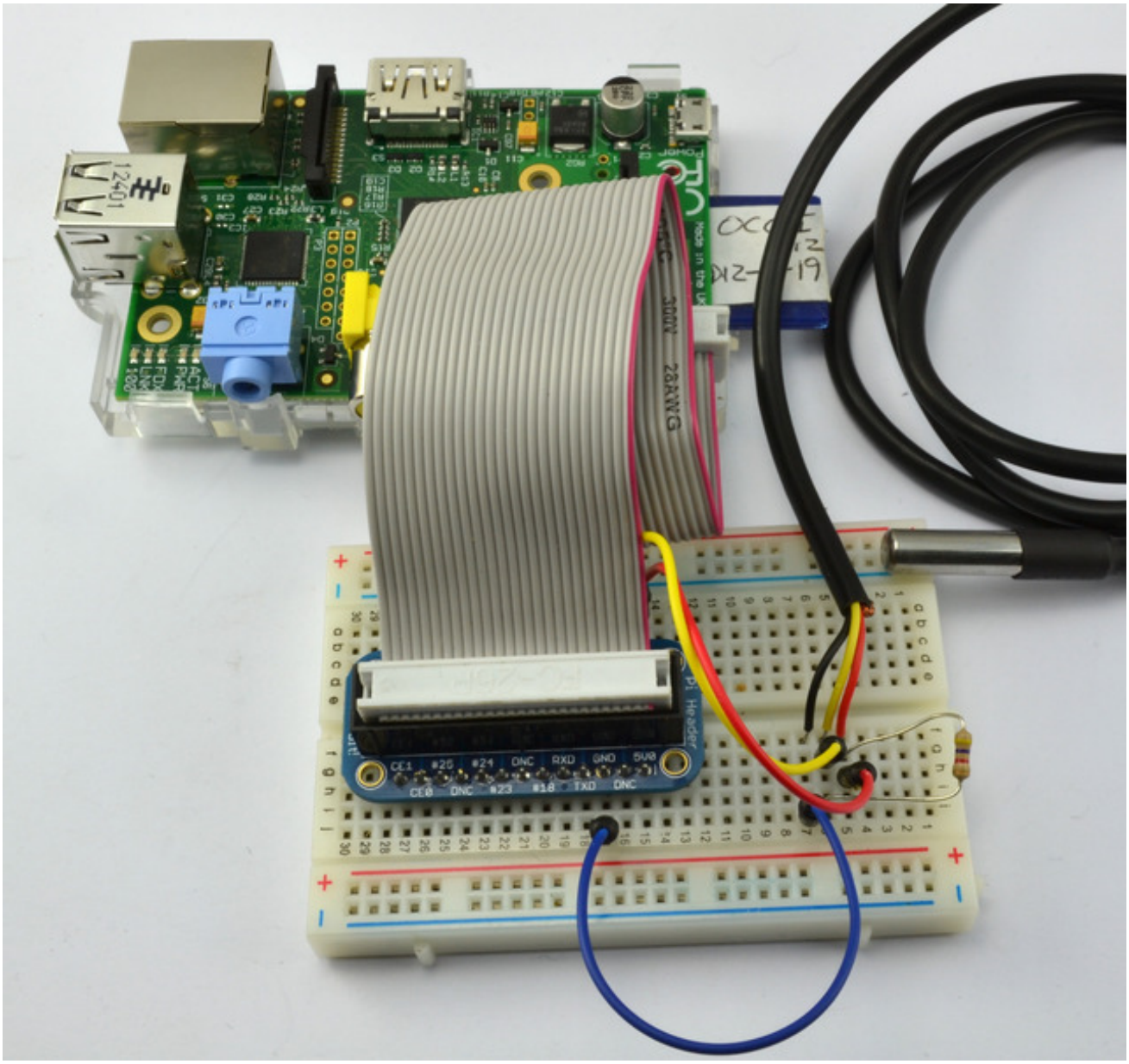


If you are using the waterproof version of the DS18B20 then the device has three leads, red, black and yellow. The bare copper screening lead that does not need to be connected.

If you are using the "high temperature" version of the DS18B20 we sell, connect Orange Stripe to 3.3V, White connects to ground and Blue Stripe is data, pin #4.

You still need a ~4.7K-10K resistor from data to 3.3V


Pi Header
3V3
SDA0
#4
DNC
#17
#22
#21
DNC
SCLK
MOSI
DNC
MISO
DNC
5V0
GND
RXD
TXD
DNC
#18
#23
DNC
#24
DNC
#25
CE1
CE0
adafruit!


12401
300V 28AWG
Pi Header
CE1 #25 #24 DNC RXD GND 5V0
CE0 DNC #23 #18 TXD DNC

DS18B20

Although the DS18B20 just looks like a regular transistor, there is actually quite a lot going on inside.

The chip includes the special 1-wire serial interface as well as control logic and the temperature sensor itself.

Its output pin sends digital messages and Raspbian/Occidentalis includes an interface to read those messages. You can experiment with the device from the command line or over SSH (see Lesson 6 (http://adafru.it/aWc)), before we run the full program.

Add OneWire support

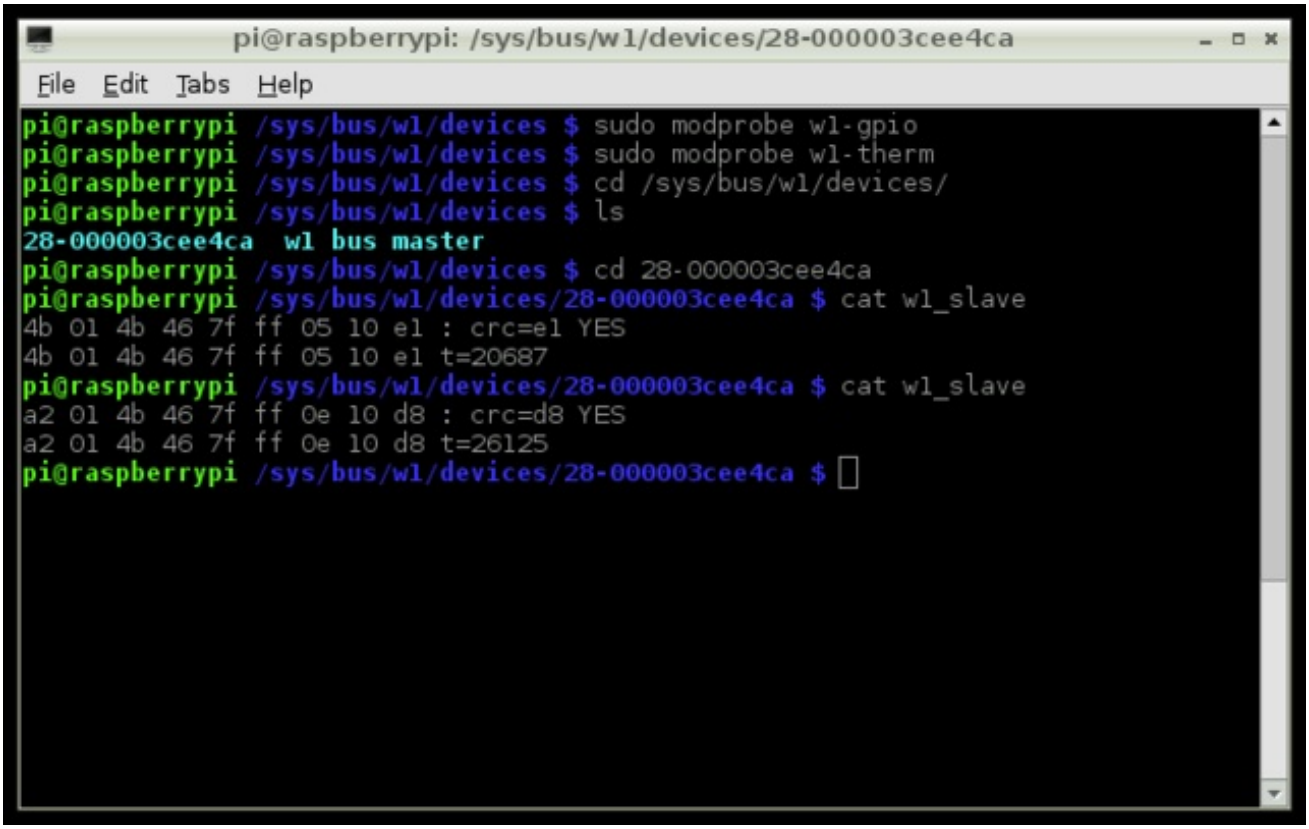
Start by adding the following line to **/boot/config.txt**

You can edit that file with nano by running **sudo nano /boot/config.txt** and then scrolling to the bottom and typing it there

```
dtoverlay=w1-gpio
```

Then reboot with **sudo reboot**. When the Pi is back up and you're logged in again, type the commands you see below into a terminal window. When you are in the 'devices' directory, the directory starting '28-' may have a different name, so cd to the name of whatever directory is there.

```
sudo modprobe w1-gpio
sudo modprobe w1-therm
cd /sys/bus/w1/devices
ls
cd 28-xxxx (change this to match what serial number pops up)
cat w1_slave
```



The interface is a little unreliable, but fortunately it tells us if there is a valid temperature to read. It's like a file, so all we have to do is read

The response will either have YES or NO at the end of the first line. If it is yes, then the temperature will be at the end of the second line, in 1/000 degrees C. So, in the example above, the temperature is actually read as 20.687 and then 26.125 degrees C.

If you have more than one Sensor connected, you'll see multiple **28-xxx** files. Each one will have the unique serial number so you may want to plug one in at a time, look at what file is created, and label the sensor!

Software

The Python program deals with any failed messages and reports the temperature in degrees C and F every second.

```
import os
import glob
import time

os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')

base_dir = '/sys/bus/w1/devices/'
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'

def read_temp_raw():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines

def read_temp():
    lines = read_temp_raw()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string) / 1000.0
        temp_f = temp_c * 9.0 / 5.0 + 32.0
        return temp_c, temp_f

while True:
 print(read_temp())
 time.sleep(1)
```

The program starts by issuing the 'modprobe' commands that are needed to start the interface running.

The next three lines, find the file from which the messages can be read.

A problem has been reported with occasional hangs when reading the temperature file when using Raspbian. If you find you have the same problem, try replacing the function read_temp_raw with the

code below. You will also need to add a line at the top of the file 'import subprocess'.

```
def read_temp_raw():
 catdata = subprocess.Popen(['cat',device_file], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
 out,err = catdata.communicate()
 out_decode = out.decode('utf-8')
 lines = out_decode.split('\n')
 return lines
```

Reading the temperature takes place in two functions, read_temp_raw just fetches the two lines of the message from the interface. The read_temp function wraps this up checking for bad messages and retrying until it gets a message with 'YES' on end of the first line. The function returns two values, the first being the temperature in degrees C and the second in degree F.

You could if you wished separate these two as shown in the example below:

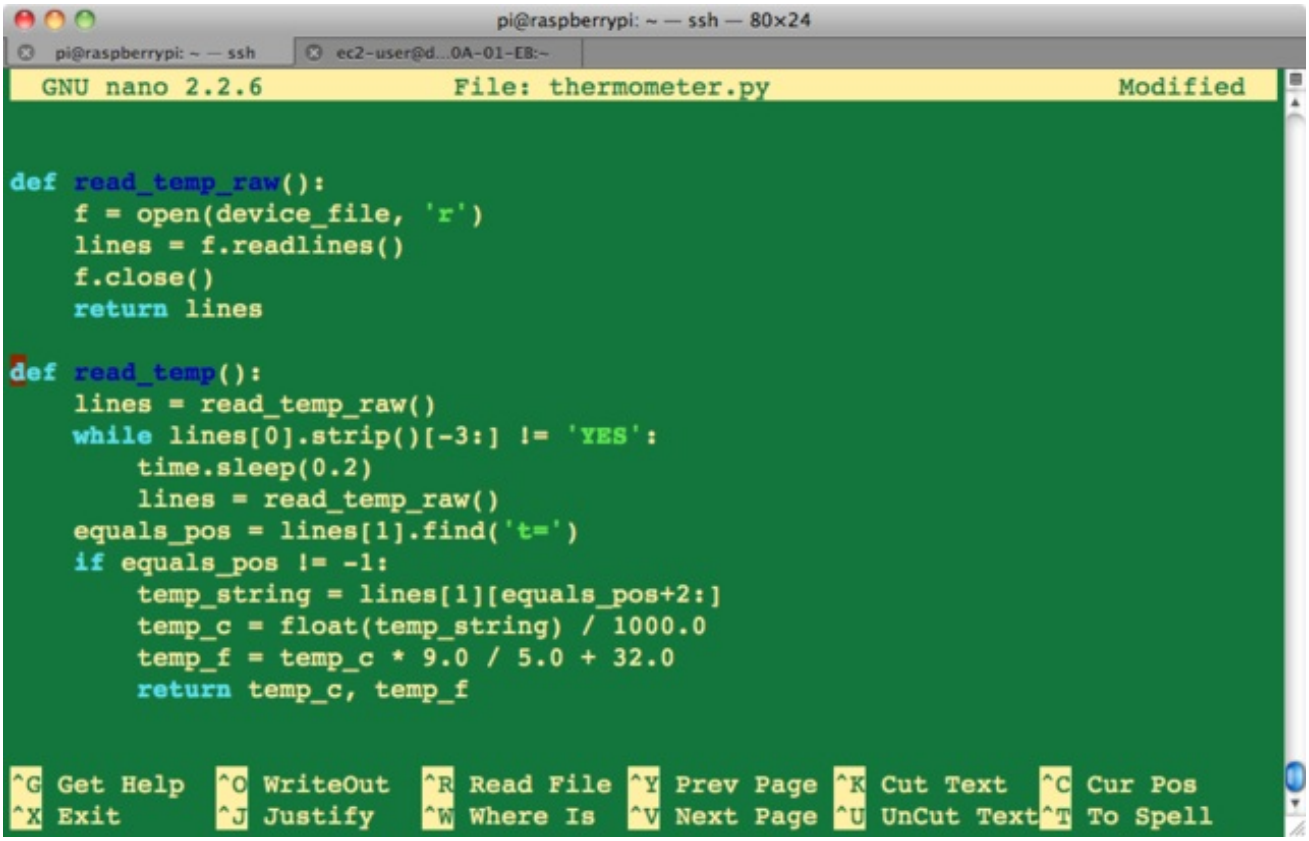
```
deg_c, deg_f = read_temp()
```

The main loop of the program simply loops, reading the temperature and printing it, before sleeping for a second.

To upload the program onto your Raspberry Pi, you can use SSH to connect to the Pi, start an editor window using the line:

```
nano thermometer.py
```

and then paste the code above, before saving the file with CTRL-x and Y.

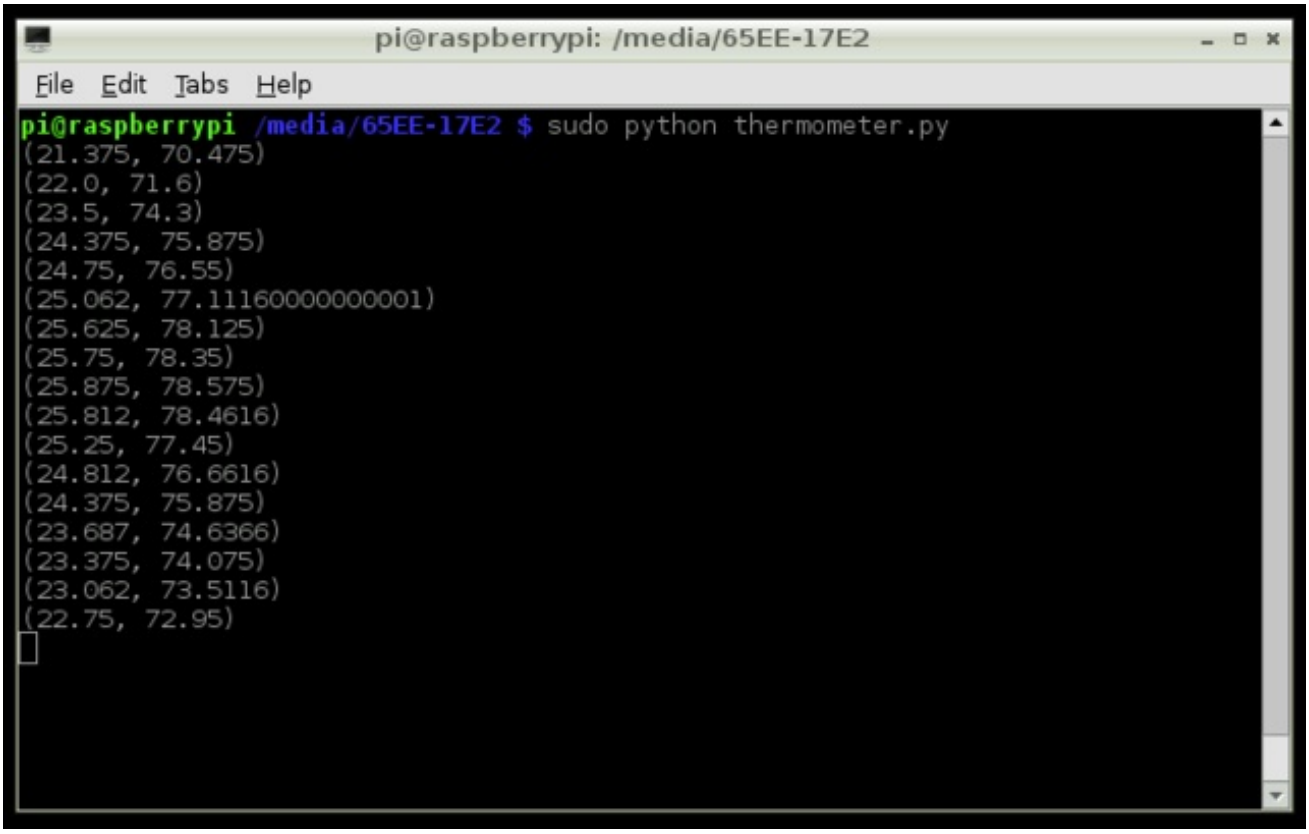

pi@raspberrypi: ~ — ssh — 80×24
pi@raspberrypi: ~ — ssh
ec2-user@d...0A-01-E8:~
GNU nano 2.2.6 File: thermometer.py Modified
def read_temp_raw():
f = open(device_file, 'r')
lines = f.readlines()
f.close()
return lines
def read_temp():
lines = read_temp_raw()
while lines[0].strip()[-3:] != 'YES':
time.sleep(0.2)
lines = read_temp_raw()
equals_pos = lines[1].find('t=')
if equals_pos != -1:
temp_string = lines[1][equals_pos+2:]
temp_c = float(temp_string) / 1000.0
temp_f = temp_c * 9.0 / 5.0 + 32.0
return temp_c, temp_f
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

Configure and Test

The program must be run as super user, so type the following command into the terminal to start it:

```
sudo python thermometer.py
```

If all is well, you will see a series of readings like this:



Try putting your finger over the sensor to warm it up.

Adding more sensors

You can add additional DS18B20 sensors in parallel - connect all the sensors' VCC, data and ground pins together. Use a single 4.7K resistor. You will see multiple **/sys/bus/w1/devices/28-nnnnn** directories, each one having the unique serial number as the directory name. The python example code only works for one sensor right now so you will have to adapt it if you want it to read from different sensors at once