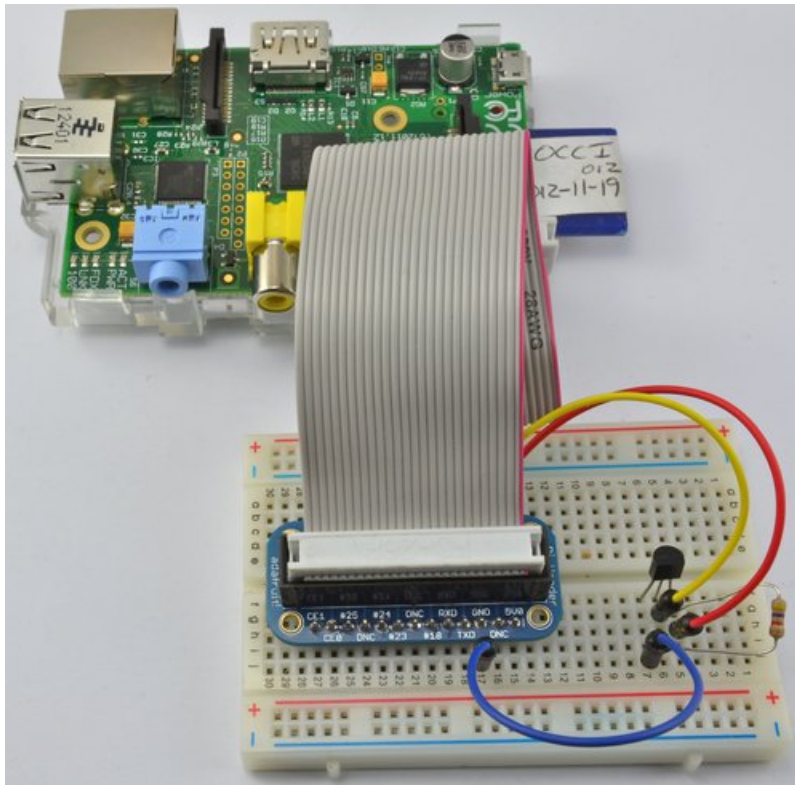


Adafruit's Raspberry Pi Lesson 11. DS18B20 Temperature Sensing

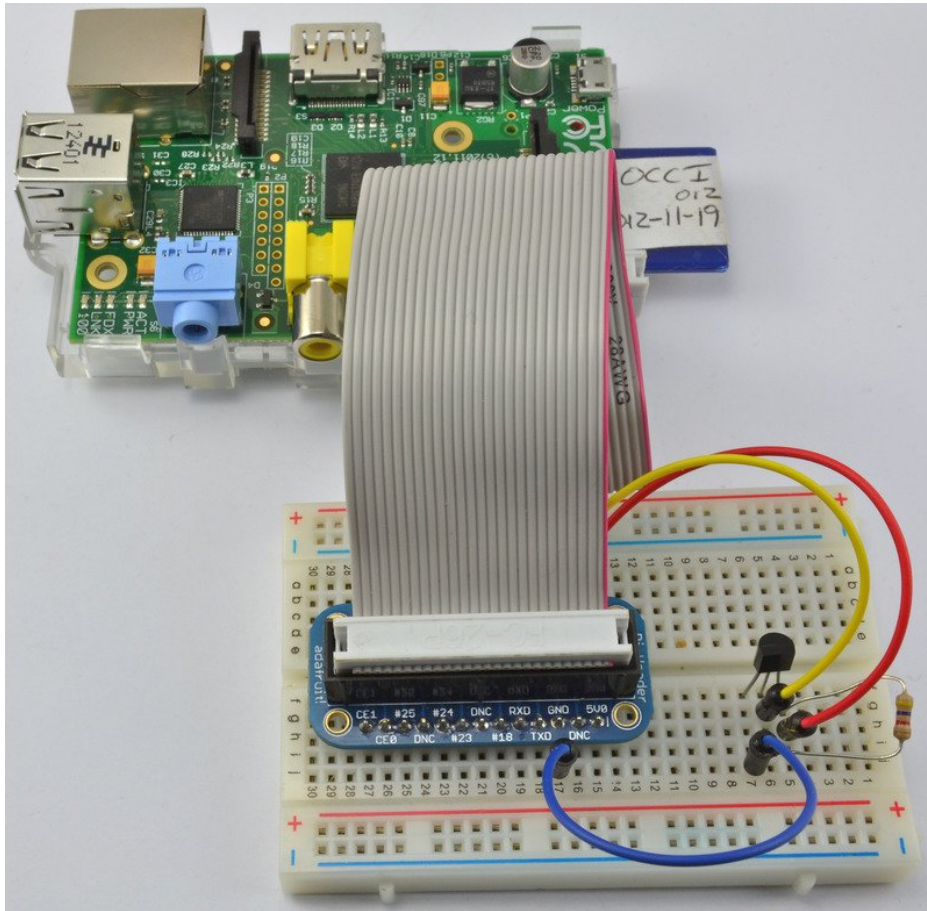
Created by Simon Monk



Last updated on 2019-05-07 08:44:58 PM UTC

Overview

The Raspbian distribution 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.

Hardware

The breadboard layout for the DS18B20 transistor form factor (TO-92) part and waterproof corded model is shown below. We provide example for the older 20-pin and modern 40-pin Raspberry Pi connectors. This setup will work on all models of Raspberry Pi which have been released except for the Pi compute modules which do not have breakout headers.

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

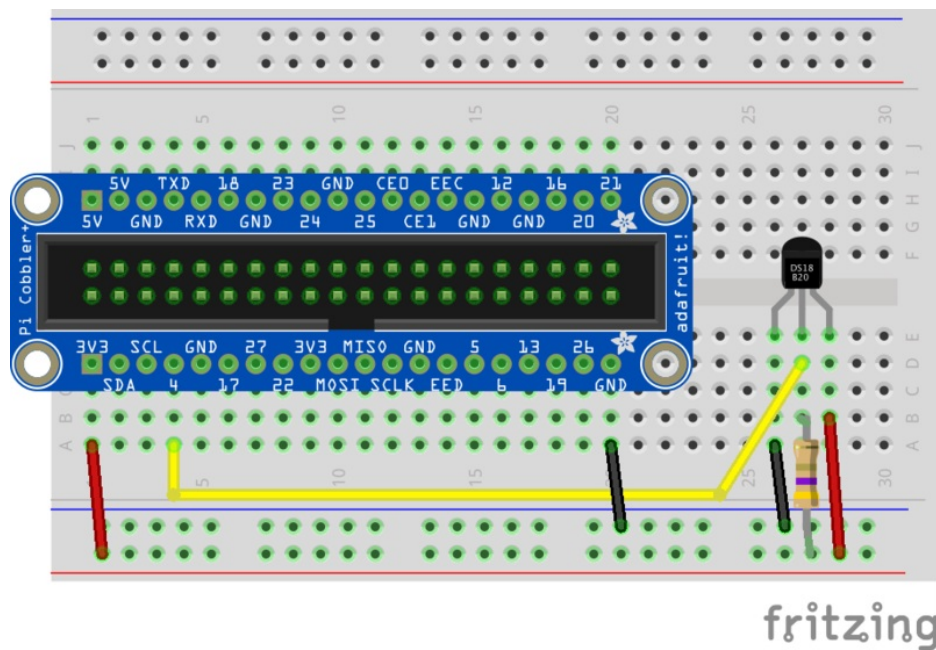
The resistor is used as a 'pullup' for the data-line, and is required to keep the data transfer stable and happy

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.

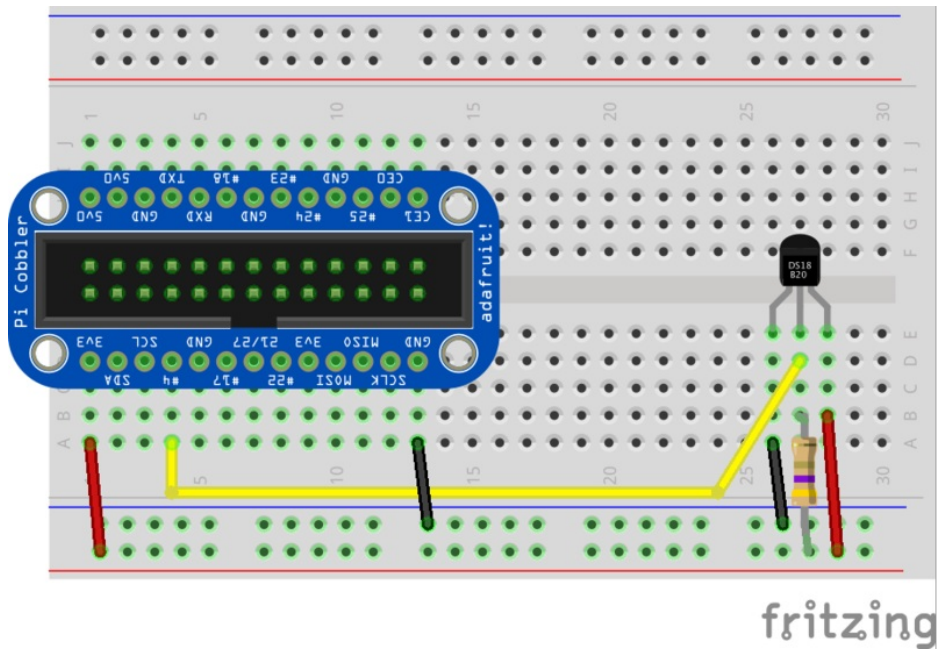


Despite both being temperature sensors, the DS18B20+ is totally different than the TMP36. You cannot use a TMP36 for this tutorial!

40-Pin (A, B, B+ and Zero) Cobbler Plus Schematic

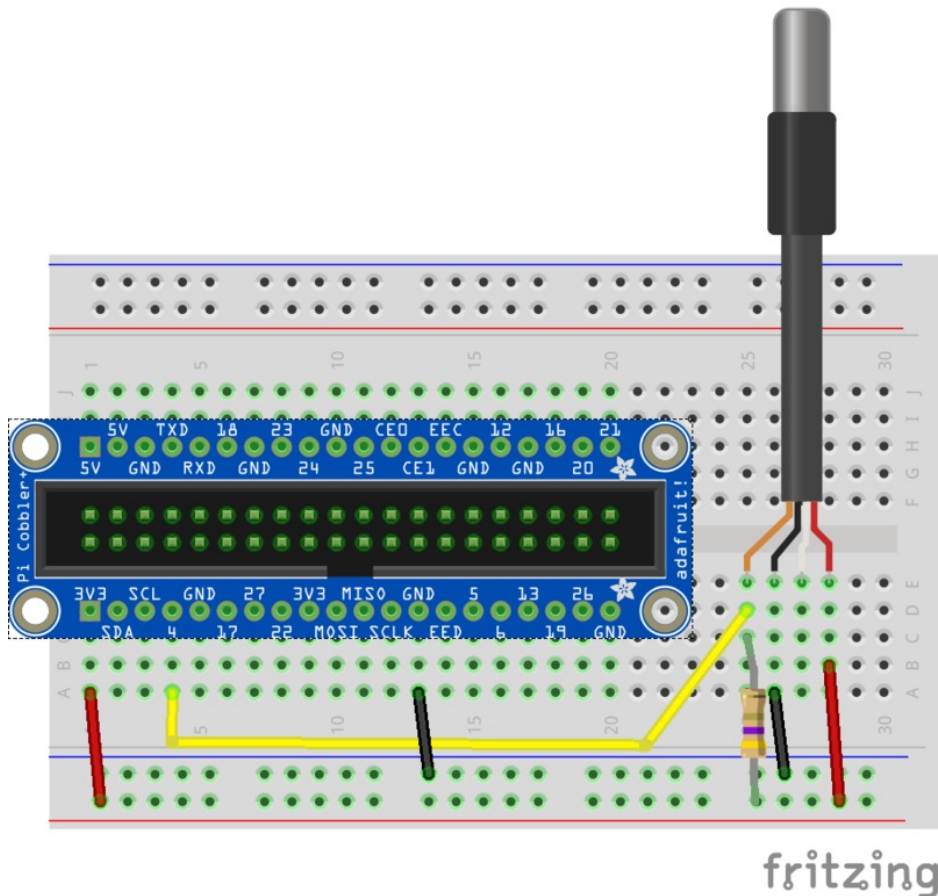


20-Pin (Raspberry Pi Rev 1 and Rev 2) Cobbler Schematic

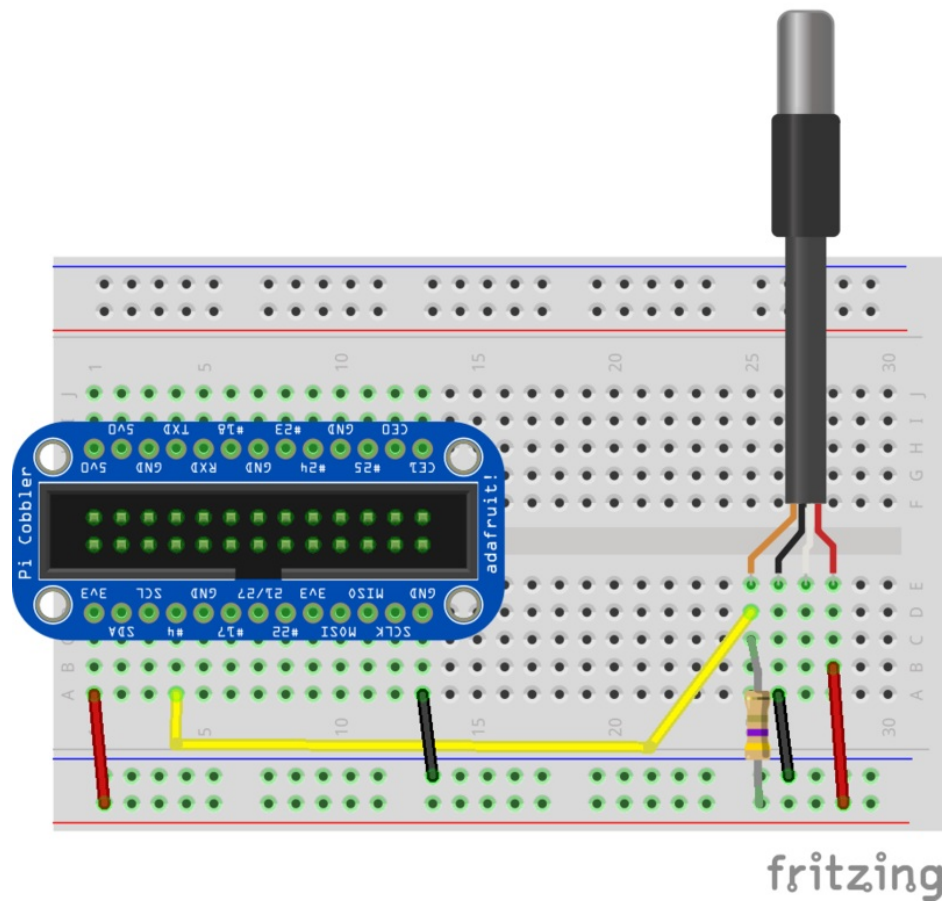


Waterproof 40-Pin [black]

If you are using the [waterproof version of the DS18B20](https://adafruit.it/C4p) (<https://adafruit.it/C4p>) then the device has three leads, red, black and yellow. The bare copper screening lead that does not need to be connected.



Waterproof 20-Pin [black]



Hightemp Waterproof [white]

If you are using the "high temperature (<https://adafruit.it/C4q>)" 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

DS18B20

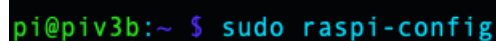
Although the DS18B20 just looks like a regular transistor, there is actually quite a lot going on inside.

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 includes an interface to read those messages. Once we enable 1-Wire using the 'raspi-config' tool on your Raspberry Pi the correct kernel modules will be loaded on subsequent reboots.

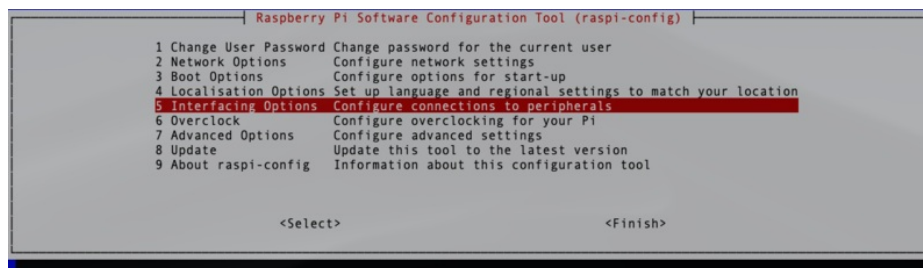
Enable 1-Wire

```
sudo raspi-config
```

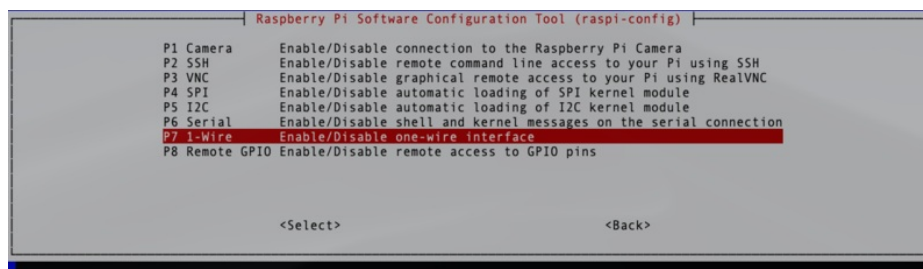


```
pi@piv3b:~ $ sudo raspi-config
```

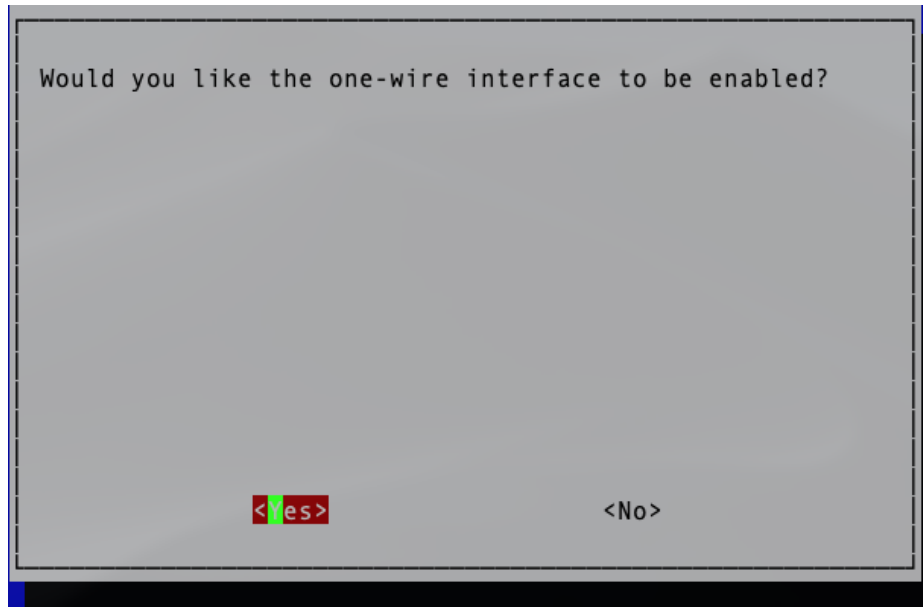
Select Interfacing Options



Select 1-Wire



Select Yes



Exit out of raspi-config and reboot your Pi so the 1-wire kernel modules load up.

```
sudo reboot
```

```
pi@piv3b:~ $ sudo reboot
```

Verify that the 1-Wire kernel modules have loaded on the next boot. You should see something like the below output when running the **lsmod** command.

```
lsmod | grep -i w1_
```

```
pi@piv3b:~ $ lsmod | grep -i w1_
w1_therm          16384  0
w1_gpio           16384  0
wire              45056  2 w1_gpio,w1_therm
```

Software

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

Temporarily unable to load content:

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

```
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 — 80x24
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

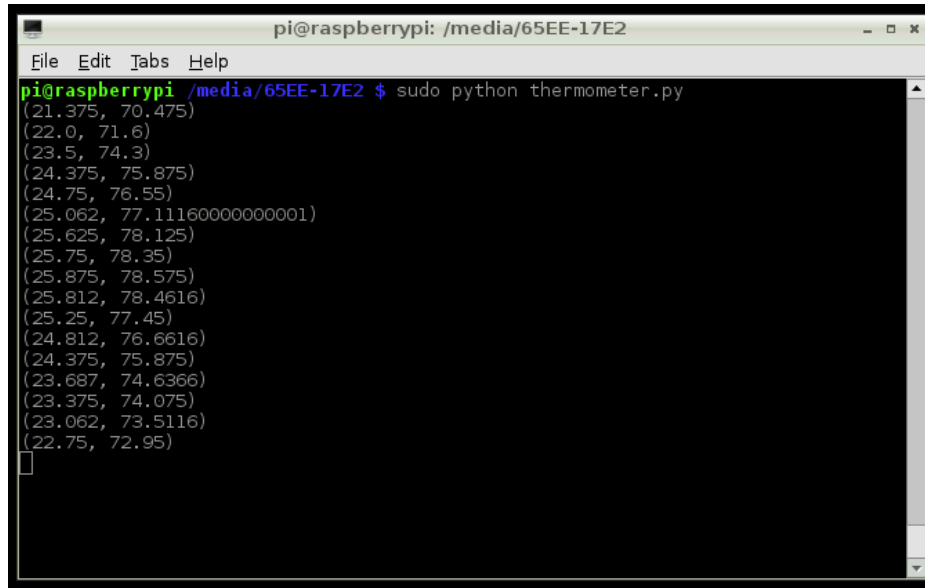
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

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:

A terminal window titled 'pi@raspberrypi: /media/65EE-17E2' displays the output of the 'sudo python thermometer.py' command. The output consists of 18 lines of coordinate pairs (x, y) representing temperature readings. The x-values range from approximately 21.375 to 25.812, and the y-values range from approximately 70.475 to 78.4616. The readings show a general upward trend followed by a slight decrease.

```
pi@raspberrypi /media/65EE-17E2 $ sudo python thermometer.py
(21.375, 70.475)
(22.0, 71.6)
(23.5, 74.3)
(24.375, 75.875)
(24.75, 76.55)
(25.062, 77.11160000000001)
(25.625, 78.125)
(25.75, 78.35)
(25.875, 78.575)
(25.812, 78.4616)
(25.25, 77.45)
(24.812, 76.6616)
(24.375, 75.875)
(23.687, 74.6366)
(23.375, 74.075)
(23.062, 73.5116)
(22.75, 72.95)

```

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