How hot is it?

This describes how to connect a DS18B20 waterproof temperature sensor to your Raspberry Pi. Since it's waterproof, you can hang it out the window or dip it in an aquarium or similar, although be careful not to expose your Raspberry Pi to the elements since that is definitely not waterproof!

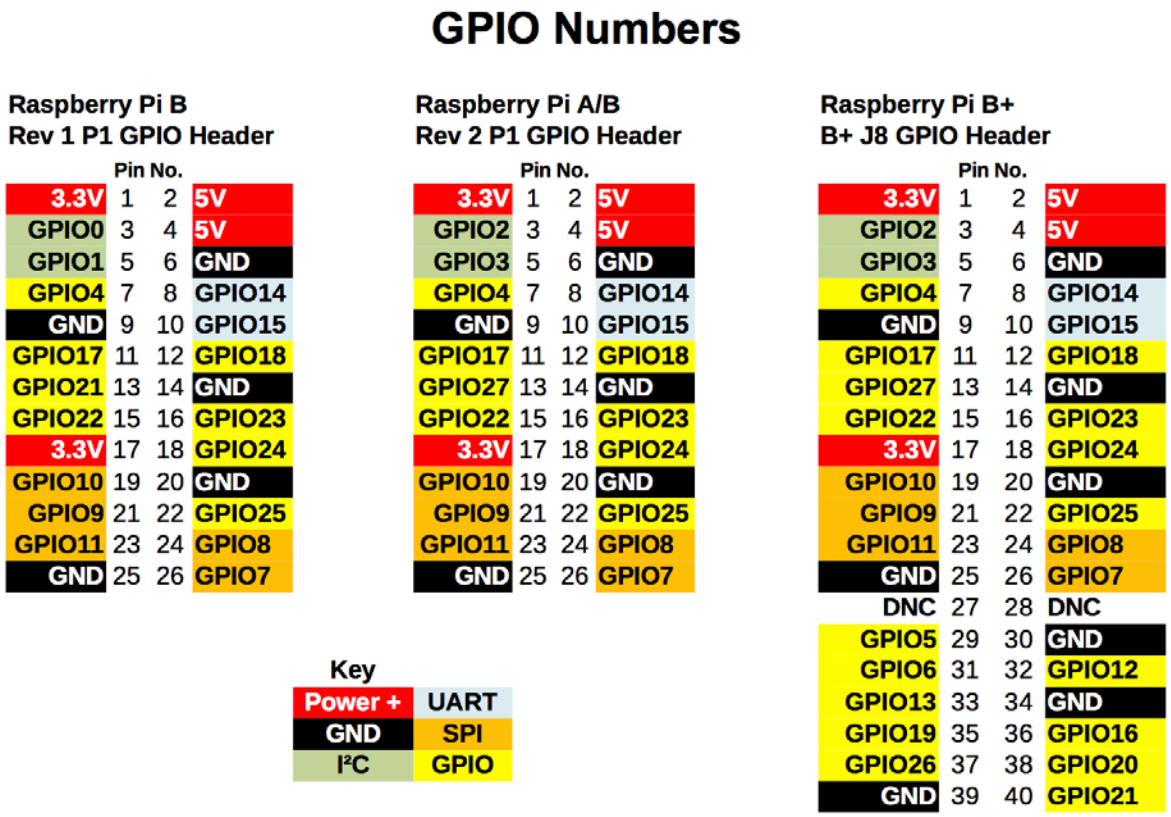
The DS18B20 sensor

The sensor has three wires:

- Red – VCC – This powers the device
- Yellow – DATA – The data all comes in on one wire, using the "1-wire protocol"
- Black – GND – This is the earth

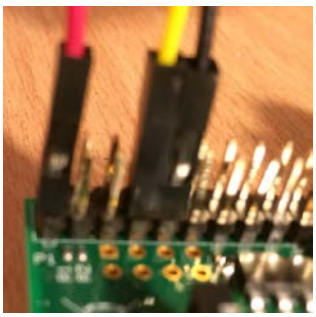
The sensor needs 3v power on the red wire, which it can take from any of the 3v pins on the Pi, i.e. pin 1 or pin 17 (see the pin diagram for the various Pi versions below – note that the A+ has the same layout as the B+). The black wire can be connected to any of the GND pins, e.g. 6, 9, 14, etc. The yellow wire can be connected to any of the GPIO pins, e.g. 3, 5, 7, etc.

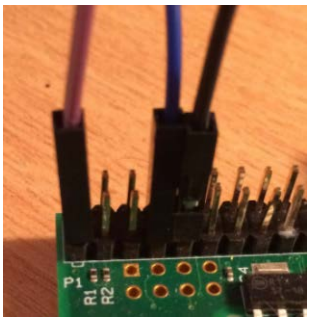
In order to keep this as simple as possible, we'll use pin 1 for power, pin 7 (GPIO4) for data and pin 9 for ground: all pins are on the top left, and all are in the same position on all versions of the Pi.



One other thing we need to do is to include what's known as a pull-up resistor across the power and data wires. The reason for this is that a pin can be HIGH or LOW or FLOATING – by adding a pull-up

resistor, we force the pin to be HIGH when no input is present. This is probably not too important for now, but you can read up on it if you're interested.





The sensors I provided have female connectors on the end of the wires, with a 4.7K Ohm pull-up resistor across the red and yellow wires. Note that I ran out of the correct colour connector wires, so if yours has different colours, they are:

- Purple – treat this as red - VCC
- Blue – treat this as yellow - DATA
- Black – unchanged – GND

Now, with your Raspberry Pi disconnected from power, slip the red wire's connector over pin 1 (top left), the yellow over pin 7 (3 pins further down) and the black over pin 9 (the next pin down).

Now go ahead and power up your Pi.

Making sure it works

The Linux kernel in Raspbian has the 1-wire protocol built in, so you don't have to do anything special to read the data. However, you do have to enable it.

Edit the file ***/etc/modules*** to enable 1-wire support by adding the following lines to the bottom:

```
w1-gpio
w1-therm
```

Next add the following line to the bottom of ***/boot/config.txt***, to tell the system that we want to use GPIO4 (pin 7) for data:

```
dtoverlay=w1-gpio,gpiopin=4
```

Finally, reboot:

```
sudo reboot
```

When the Pi has restarted, display the contents of the /sys/bus/w1/devices directory:

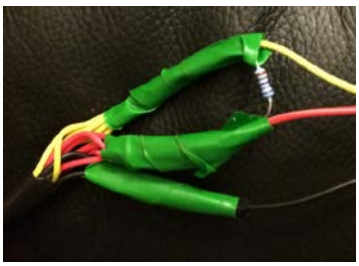
```
brendan@minnow ~ $ ls /sys/bus/w1/devices
28-0000060886df  w1_bus_master1
```

The folder with the name that starts with "28-" followed by a bunch of hexadecimal digits is the folder containing the interesting data. This is where the sensor's data are stored.

Note that each sensor has a unique ID (the bunch of hex digits in the folder name) and if you have a number of sensors connected, they will each have a unique folder name:

```
brendan@beespi ~ $ ls /sys/bus/w1/devices
28-000005faa303  28-000005faee71  28-000005fb858d
28-000005fae41b  28-000005fb27bd  w1_bus_master1
```

In fact, if you have a number of sensors connected, you simply connect all the red wires together, all the yellows together and all the blacks together, and treat them as though you had a single wire of each colour, as you can see in the picture:



Reading the temperature

The 28-XXXXXXXXX folder contains the current temperature reading from the sensor. It also indicates if there has been an error in the most recent reading:

```
brendan@minnow ~ $ ls /sys/bus/w1/devices
28-0000060886df  w1_bus_master1
brendan@minnow ~ $ cat /sys/bus/w1/devices/28-0000060886df/w1_slave
fc 00 4b 46 7f ff 04 10 a6 : crc=a6 YES
fc 00 4b 46 7f ff 04 10 a6 t=15750
```

The w1_slave file in the sensor folder contains a checksum and whether the reading is value on the first line, and the temperature on the second line.

- Checksum = a6
- Reading is valid: YES
- Temperature: 15.750°C

Now what?

First you want to create a way to capture the data: the simplest way is to read the file, verify that the first line ends in YES, and then grab the temperature value from the second line. You could do this in Python or PHP or Node.js or in any of a number of ways.

Next, decide how you want to show it to the users. For example, you could display the current temperature on a web page, perhaps using a nice pretty widget:

You could store readings from every 15 minutes, and then display them as a graph showing the temperature changes over the time you've recorded. This could mean that you write to a database such as SQLite3, storing the data and time and temperature. You can then read the data from the database and display as a graph.

You could sound an alarm if the temperature fell outside some standard values.

As you can see, there are plenty of applications you can try.