Raspberry Pi Temperature Sensor

Tutorial by Andrew Oakley
Public Domain 2 July 2016
www.cotswoldjam.org

Getting started

This tutorial shows you how to make a temperature sensor which shines red when hot, and blue when cold.

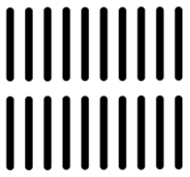
```
Words you will see on the screen, or that you need to type in, are
highlighted like this. At the end of each line, press Enter.
```

Your tutor should have prepared your Raspberry Pi for you and given you a bag of components. Alternatively, you can download the files from: cotswoldjam.org/tutorials

The electronics

A DS18B20 waterproof temperature sensor. It has three wires at one end, and a metal sensor on the other end. This version is waterproof and the sensor can be dipped in water. A smaller, cheaper, non-waterproof version is also available.

A mini breadboard. This has a top and bottom section. All the points along each column in each section are joined, so we can make electrical connections.



Three resistors. Check the coloured stripes - the two identical ones are 220 Ohms and the odd one out (with a blue stripe) is 4700 Ohms. The more Ohms, the more resistance to electricity. Resistors can be placed either way around.

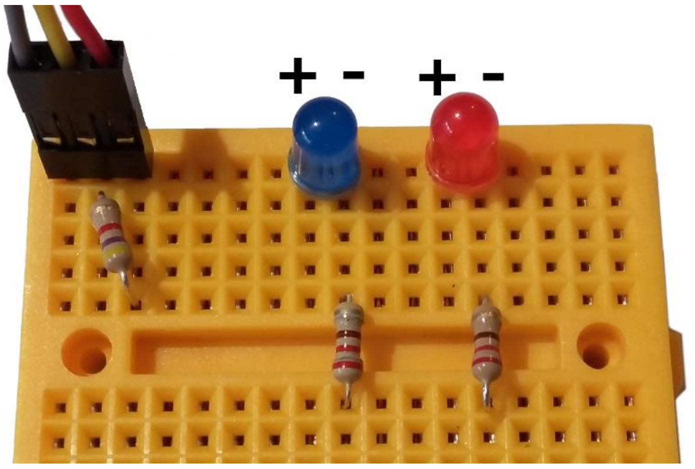
Red and blue LEDs. The short leg is the negative leg. There is also a flat side to the rim of the bulb; this will also indicate which side is negative.

LEDs also need a small resistor to prevent them burning out. We will use a 220 Ohm resistor for each LED.

Jumper wires. You should have six male-to-female jumpers (sticky-outy at one end, inny at the other end) and one male-to-male jumper (outy at both ends).

Your wires may be of many different colours.

The circuit



Place the components on the breadboard like this.

The LEDs have their negative connector (short leg, flat side) to the right, on the same column as the 220 Ohm resistors.

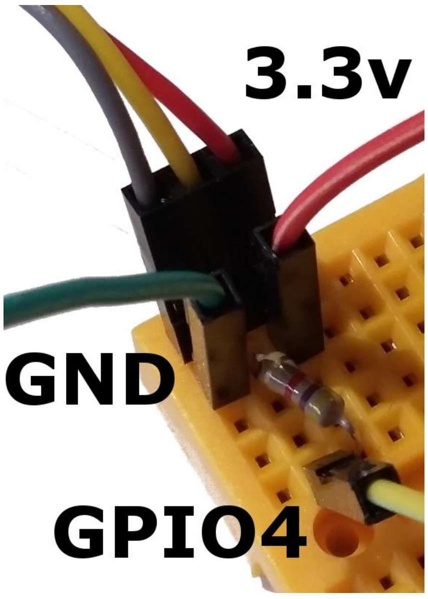
The temperature sensor has three connectors:

- Grey (sometimes green or black) - Negative or Ground - should be on the left
- Yellow (sometimes white) - Data - should be in the middle
- Red (sometimes brown) - Positive - should be on the right

The 4700 Ohm resistor crosses the columns between the temperature sensor's data connector (centre), and the positive connector (right).

Next, connect three male-to-female jumpers between the temperature sensor and the Raspberry Pi's GPIO pins.

Your jumper leads may be different colours. The author of this document is heavily colour-blind. What matters is that the wires go to the right places. Ignore the colours.

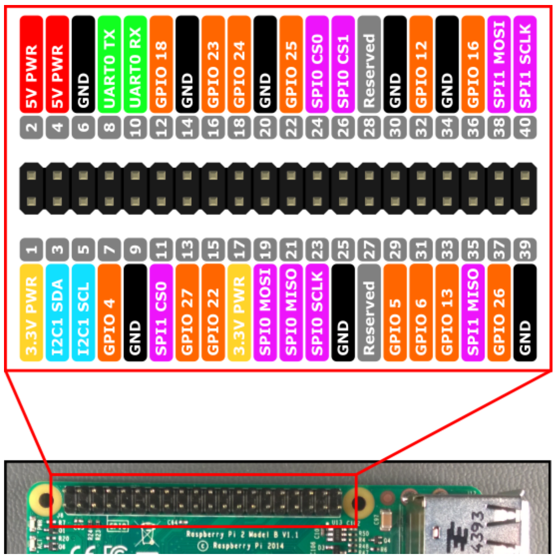


Ground (negative) goes to pin 6 (GND).

Data (middle connector) goes to pin 7 (GPIO4).

3.3v (positive) goes to pin 1.

Note that pin numbers ("board numbers") and GPIO numbers ("BCM numbers" - Broadcom, the processor maker) are not the same. Check the diagram.



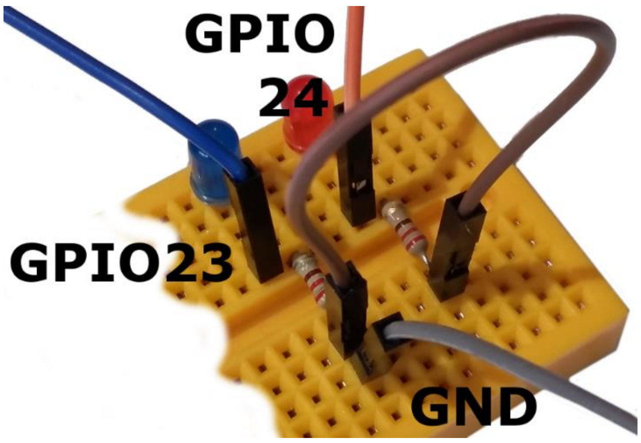
5V PWR	5V PWR	GND	UART0 TX	UART0 RX	GPIO 18	GND	GPIO 23	GPIO 24	GND	GPIO 25	SPI0 CS0	SPI0 CS1	Reserved	GND	GPIO 12	GND	GPIO 16	SPI1 MOSI	SPI1 SCLK
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39
3.3V PWR	I2C1 SDA	I2C1 SCL	GPIO 4	GND	SPI1 CS0	GPIO 27	GPIO 22	3.3V PWR	SPI0 MOSI	SPI0 MISO	SPI0 SCLK	GND	Reserved	GPIO 5	GPIO 6	GPIO 13	SPI1 MISO	GPIO 26	GND

Now connect the jumpers between the LEDs and the Raspberry Pi.

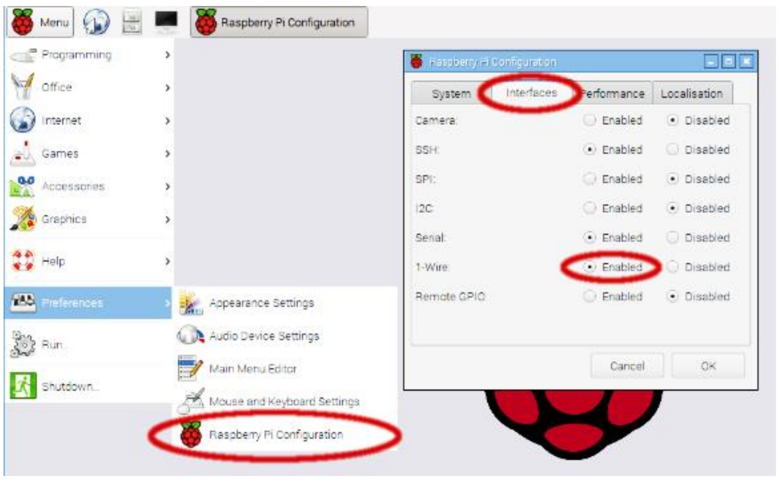
A male-to-female jumper goes from the positive leg of the blue LED to pin 16 (GPIO23)

Another male-to-female jumper goes from the positive leg of the red LED to pin 18 (GPIO24)

A male-to-male jumper crosses between the resistors in the bottom section, and finally, a male-to-female jumper goes from one of the resistors in the bottom section, to pin 14 (GND).



Configuring the Pi for 1-Wire Sensors



Camera: Enabled [✓ Disabled]
SSH: [✓ Enabled] Disabled
SPI: Enabled [✓ Disabled]
I2C: Enabled [✓ Disabled]
Serial: [✓ Enabled] Disabled
1-Wire: [✓ Enabled] Disabled
Remote GPIO: Enabled [✓ Disabled]

The temperature sensor uses a 1-Wire interface (positive and negative connectors, plus 1 data connector). We need to turn 1-Wire on.

From the Raspberry Pi desktop, click **Menu** - **Preferences** - **Raspberry Pi Configuration**.

From the Configuration program, click the **Interfaces** tab at the top. Then find the **1-Wire** row and click **Enabled**. If it's already enabled, that's fine. If it wasn't already enabled, you will be asked to reboot the machine - make sure you reboot.

First program - taking the temperature

From the menu, select **Programming** - **Python 3**. Then use **File**, **New Window** to create a new program and type it in. Alternatively you can load the ready-made program using **File**, **Open**, then scroll sideways and double-click the **python** folder, then double-click the **temperature** folder, then click **temp1.py** and **Open**.

When typing in the program, *make sure you put spaces where they're needed* at the start of lines. For example, two spaces before **temp=readtemp.readtemp()** . Spaces must line up.

```
import gpiozero, readtemp
from time import sleep
while True:
  temp=readtemp.readtemp()
  print ( "Temp: {}c".format(temp) )
  sleep(0.1)
```

Run the program by selecting **Run** menu, **Run Module**. You should see some temperature readings! Try warming up the sensor by holding it tightly. Stop the program by holding down the **CTRL key** and pressing **C**.

import teaches the computer about new things, using small programs that other people have written. We use the word "library" to decribe this. **gpiozero** is a library that makes the GPIO pins easy to use. **readtemp** is a library that makes the temperature sensor easy to use. We also import the **sleep** command from the **time** library.

while True: repeats a section forever (keep going while… always!)

temp=readtemp.readtemp() reads the temperature from the sensor, and places the value in Celcius into the variable named **temp** . Variables are like boxes which can hold numbers or words.

print outputs information to the screen. In this case, a temperature reading.

sleep waits for a number of seconds. We leave a gap of 0.1 of a second between readings. However, readings also take a moment, so the loop doesn't go too fast.

Second program - lighting the LEDs

Change the program, or load in the **temp2.py** file:

```
import gpiozero, readtemp
from time import sleep

cold=22
hot=32
blueled=gpiozero.PWMLED(23)
redled=gpiozero.PWMLED(24)
blueled.on()
redled.on()

while True:
  temp=readtemp.readtemp()

  hotness=(temp-cold)/(hot-cold)
  if hotness>1:
    hotness=1
  if hotness<0:
    hotness=0
  print ( "Temp: {}c - Hotness {}".format(temp,hotness) )

  blueled.value=1-hotness
  redled.value=hotness
  sleep(0.1)
```

Run the program by selecting **Run** menu, **Run Module**. The red and blue LEDs will change their brightness depending on the temperature! Stop the program with **CTRL-C** and try changing the values of **cold** and **hot** . Body temperature is about 37, an ice cube is 0 and a typical room temperature is 22 degrees celcius.

gpiozero.PWMLED tells the computer that we have an LED connected, and that we will change the brightness using Pulse Width Modulation - flashing it on and off very quickly!

We set the brightness to a level (**hotness**) depending on the temperature between **cold** and **hot** . If the temperature is **hot** or above, only red lights up (**hotness** is set to 1). If it's **cold** or below, only blue (**hotness** is set to 0). If it's in between, the brightness of blue and red will be dim or bright depending on temperature - that's the maths that does **(temp-cold)** divided by **(hot-cold)**. Try working out the maths with a pencil or a calculator!

So **hotness** is always a decimal number between 0 (cold) and 1 (hot). We use a neat maths trick for the blue LED to get the "reverse" of the brightness; 1 minus hotness. When **hotness** is 0.8, the blue LED's brightness is 1-0.8 = 0.2 .

Advanced programmers

Have a look at the **readtemp.py** library to find out how we get the readings from the temperature sensor. Find the file **/sys/bus/w1/devices/28-***something***/w1_slave** and output the readings from the terminal using **cat** .