Keeping cool

Using a Raspberry PI to create a networked temperature sensor

By Dwight Hubbard

dwight.hubbard@gmail.com

How this got started

We wanted to ensure our cat was safe during our pit stops on road trips.

Next Up

Choosing the Hardware

Raspberry Pi Hardware Connections

The Raspberry Pi has a General Purpose Input Output (GPIO) pins that can be used to connect to electronics.

These pins allow connecting electronics directly to the system.

This is really powerful

Raspberry Pi GPIO requirements

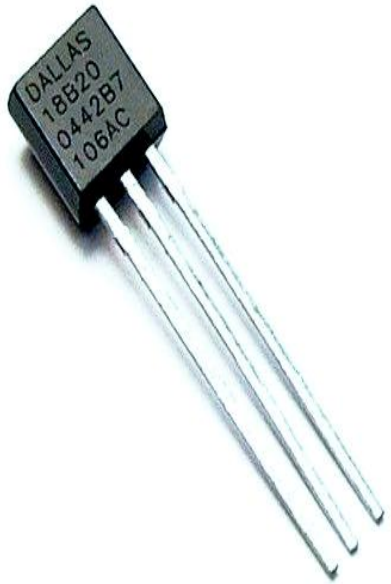
When choosing electronics to connect to the Raspberry PI GPIO pins there are 3 simple things to look for:

- ☐ Component must operate at 3.3 volts
- ☐ Component must be digital
- ☐ Documentation or software (driver) to interpret the signals from the component into something useful.

The Temp Sensor

Maxim/Dallas DS18B20

- Operates at 3.3 Volts
  - Supports 3.0Volts to 5.5Volts
- Digital
  - Uses a serial (digital) protocol
- Driver - Built into the Raspbian distribution.

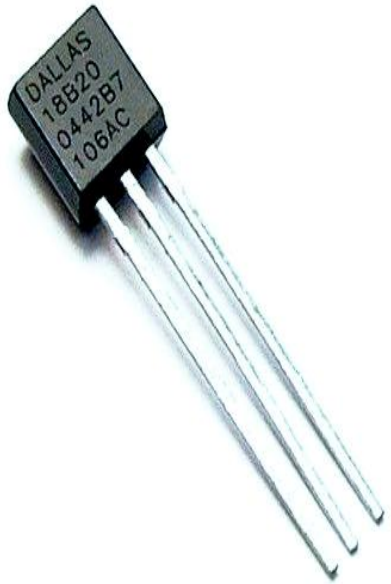


Temp Sensor Continued

The sensor has 2 sides that look different

2 of the wires are for power

1 wire sends the temperature data



Next Up

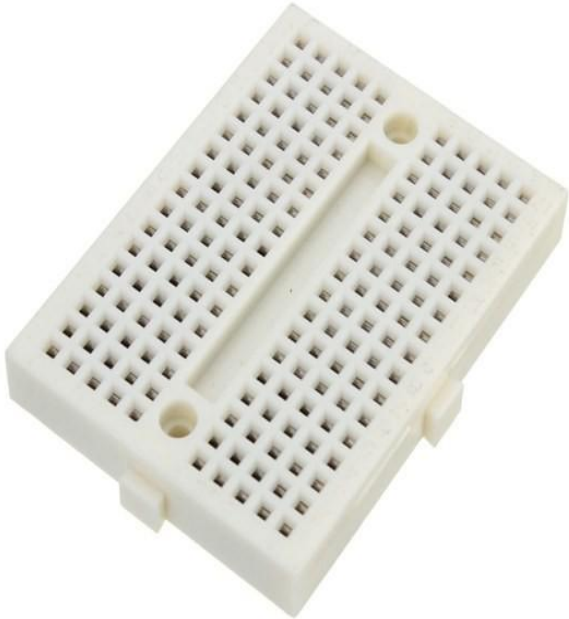
Hooking up the Hardware

Other Things

Breadboard - A prototyping board

Wires - To connect the Raspberry Pi GPIO pins to the Breadboard.

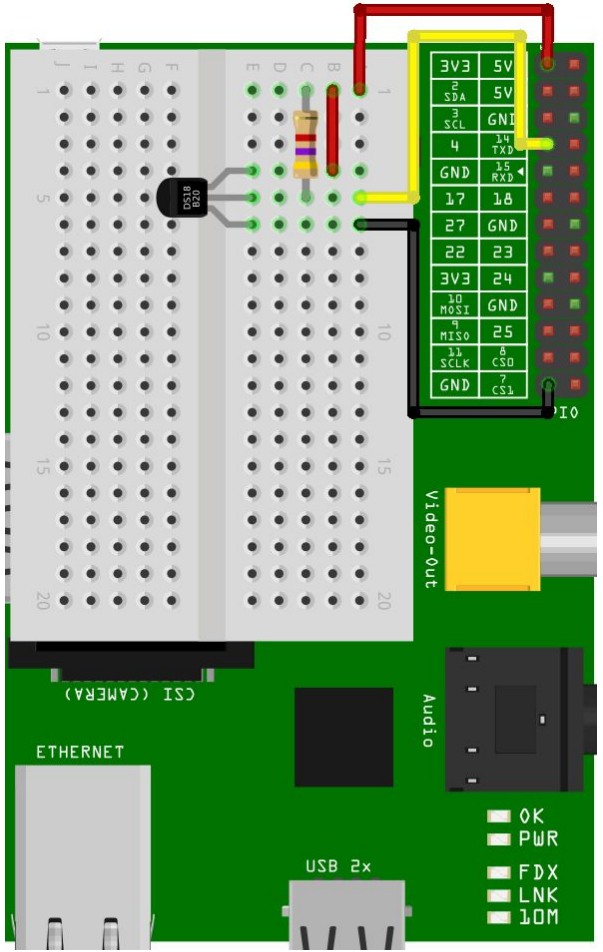
4.7k resistor



Wiring the sensor to the Raspberry Pi

To connect the sensor we do the following:

1. Plug the sensor into the breadboard
2. Connect the 3v GPIO pin to the breadboard
3. Connect the 3v row to the sensor power pin
4. Connect a GPIO GND pin to the row with the sensor GND Pin
5. Connect the 4.7k resistor to the 3v row and the row with the middle sensor pin
6. Connect the middle pin to the Pin labeled #4



Next Up

Lessons Learned from writing the first software.

First software attempt

The first version of the sensor used the sqlite python module for logging the temperature data.

SD-Cards like the ones in the Raspberry Pi can only handle a limited number of writes.

Left running, and sd-card failed after about a month.

Lesson learned

Take into account the limited lifespan of the sd-card when writing the software.

Switch to Redis for Logging

- Stores data to RAM memory
- Writes to disk/sd-card are configurable
- Can replicate the data to another computer over a network without writing to the sd-card
- Easy to learn if you know Python

Next Up
Setting Up

Setting up the Raspberry Pi with our software.

Setting up the Software

Everything this example uses is either part of the Raspbian OS or installable using the python pip tool.

```
$ pip install redislite redis-collections bottle
```

Next Up Writing the Code

Now we have the sensor all wired up now time to write some code to use it.

Data Logger

```
from redislite import StrictRedis
from redis_collections import List
from time import sleep

DEVICE_ID = '28-000006b63824'
def read_temp_c():
    device_file = '/sys/bus/w1/devices/' + DEVICE_ID + '/w1_slave'
    with open(device_file) as device:
        for line in device:
            if 't=' in line:
                return float(line.split('=')[-1])/1000

temp_readings = List(redis=StrictRedis('/var/lib/example1.rdb'), key='temp:'+DEVICE_ID)
while True:
    current_temp = read_temp_c()
    temp_readings.append(current_temp)
    print('Temp C:', current_temp)
    sleep(1)
```

Web Interface

```
from bottle import route, run
from redislite import StrictRedis
from redis_collections import List

DEVICE_ID = '28-000006b63824'
temp_readings = List(redis=StrictRedis('/var/lib/example1.rdb'), key='temp:'+DEVICE_ID)

@route('/')
def current_temp():
        return ‘%d:%f’ % (len(temp_readings), temp_readings[-1])

@route('/average_temp/<seconds>')
def average_temp(seconds=3600):
        seconds=int(seconds)
        return str(sum(temp_readings[-seconds:])/len(temp_readings[-seconds:]))

run(host='10.10.10.10', port=8080, debug=True)
```

Next Up Networking Sensors

So lets see how we can run more than one of these

Networked Data Logger

```
from redislite import StrictRedis
from redis_collections import List
from time import sleep

DEVICE_ID = '28-000006b63824'
def read_temp_c():
    device_file = '/sys/bus/w1/devices/' + DEVICE_ID + '/w1_slave'
    with open(device_file) as device:
        for line in device:
            if 't=' in line:
                return float(line.split('=')[-1])/1000

temp_readings = List(redis=StrictRedis('/var/lib/example2.rdb', serverconfig={'port': '8002', 'requirepass': 'secret'}),
key='temp:'+DEVICE_ID)
while True:
    current_temp = read_temp_c()
    temp_readings.append(current_temp)
    print('Temp C:', current_temp)
    sleep(1)
```

Replicated Web Interface

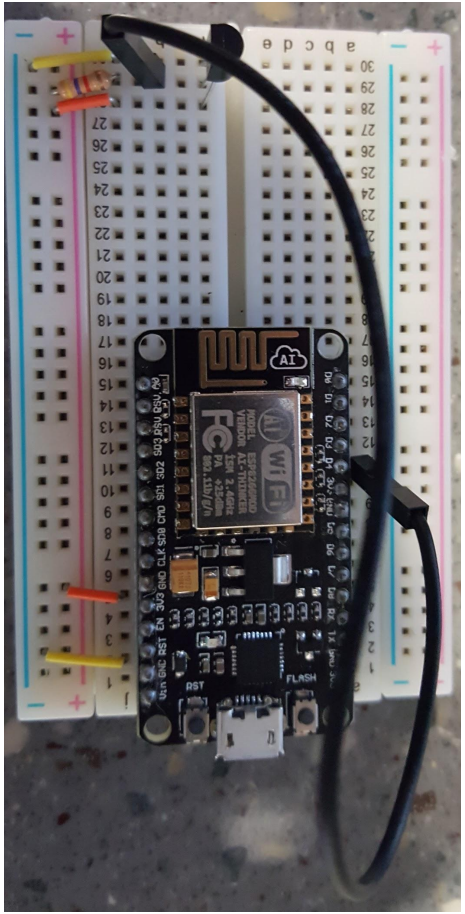
```
from bottle import route, run
from redislite import StrictRedis
from redis_collections import List

redis_connection = StrictRedis('/tmp/temp.rdb', serverconfig={'slaveof': 'notebook.local 8002', 'masterauth': 'secret'})
temp_readings = List(redis=redis_connection, key='temp_readings')

@route('/current_temp')
def current_temp():
        return str(temp_readings[-1])

@route('/average_temp/')
@route('/average_temp/<seconds>')
def average_temp(seconds=3600):
        seconds=int(seconds)
        readings=list(temp_readings)[-seconds:]
        return str(sum(readings)/len(readings))
run(host='0.0.0.0', port=8080, debug=True)
```

Next Up
Adding more

Taking things further

There are a lot of different options and tradeoffs when implementing Internet of Things projects with Python.

This photo is an example of the next generation of this project which uses a wifi enabled, battery powered microcontroller powered by micropython.



More Information

Component vendor tutorials
Adafruit - https://learn.adafruit.com/category/raspberry-pi
SparkFun - https://learn.sparkfun.com/tutorials/tags/raspberry-pi

Python on Microcontrollers
Micropython - http://micropython.org/