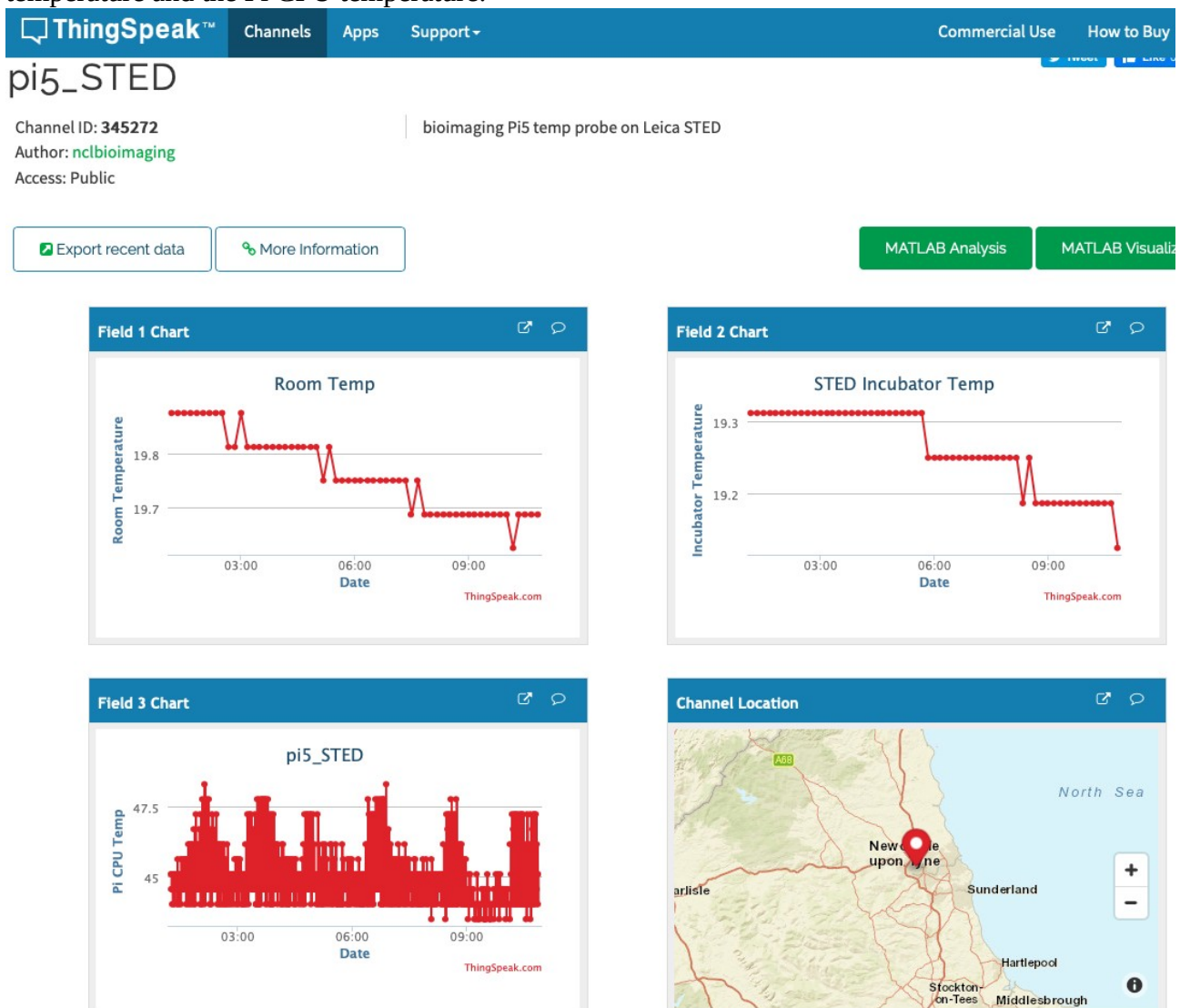


Temperature Recorder for Room and Incubator Monitoring

This document describes how to build a Raspberry Pi with two temperature probes that report to ThingSpeak, a website that will plot the data for you using inbuilt or matlab functions, and also send an email if the temperature goes above a set point. We use these to monitor room temperature for microscopes where the stability of the temperature affects the performance of the microscope. We also use the second probe to monitor the microscope incubator for live cell imaging, to ensure temperatures remain stable throughout experiments. An example screenshot of recordings from one of our systems is shown below, with the location of the probes, the room temperature, incubator temperature and the Pi CPU temperature:



Requirements

Hardware:

I purchased all bar the soldering iron from thepihut.com.

- Raspberry Pi – I built these with Pi2 and Pi3 boards, but Pi Zero or Pi4 should work. Buying the bundled version including the case, SD card and power adapter is simplest. <https://thepihut.com/collections/raspberry-pi-kits-and-bundles/products/raspberry-pi-starter-kit>
- 2 x Waterproof DS18B20 temperature probes. You can use cheaper non waterproof version, but this is pre-wired and easier to use. <https://thepihut.com/products/waterproof-ds18b20-digital-temperature-sensor-extras>
- 1 x Resistor, 4.7 kOhm (comes with the above linked temp probe)
- Wiring to attach resistors in-line, e.g. <https://thepihut.com/collections/adafruit-cables/products/premium-female-male-extension-jumper-wires-20-x-6>
- Soldering iron (Aldi special works fine!)
- Shrink tube, e.g. <https://thepihut.com/products/heat-shrink-pack>

Software:

- Pi NOOBS software works fine and comes with the Pi bundle as listed above and allows you to install the OS Raspbian <https://www.raspberrypi.org/downloads/raspberry-pi-os/> which is perfectly fine for what we're doing.
- A ThingSpeak account (free), www.thingspeak.com . Set up the account and create one Channel for each Pi you are setting up, noting the Channel ID. In the Channel, click on the API tab and make a copy of the write API code.
- Optional- an email account with known server addresses.

Notes:

- 1 wire sensors only work on GPIO4, so need to have them serially connected, sharing an in line resistor. It is possible to remap another GPIO for 1 wire, but a bit more complicated to do so and I never tried it.
- A ThingSpeak account can have multiple pages/ channels- we have one set up of reach room that has a Pi in it. Each channel can report multiple datasources, eg. the three temperatures we export here.
- Build each Pi reasonably secure with a unique username and password.
- The python script has a lot of commented out lines for checking it all worked- you can uncomment these by removing the hashes at the start to check you have your system set up right.

Method:

1. Build a Pi with the SD card and get the MAC address registered with your local IT people so that you can talk across the ethernet, and also ask them to give it a static IP address I can't add more to this as every workplace is different for what requirements the IT Dept. want from you to give access. Then ensure you have the IP address for the Pi. It is far easier to do this by plugging a mouse, keyboard and monitor I at first to get these values in my experience. I haven't listed these hardware as required as once they are built, they aren't required.

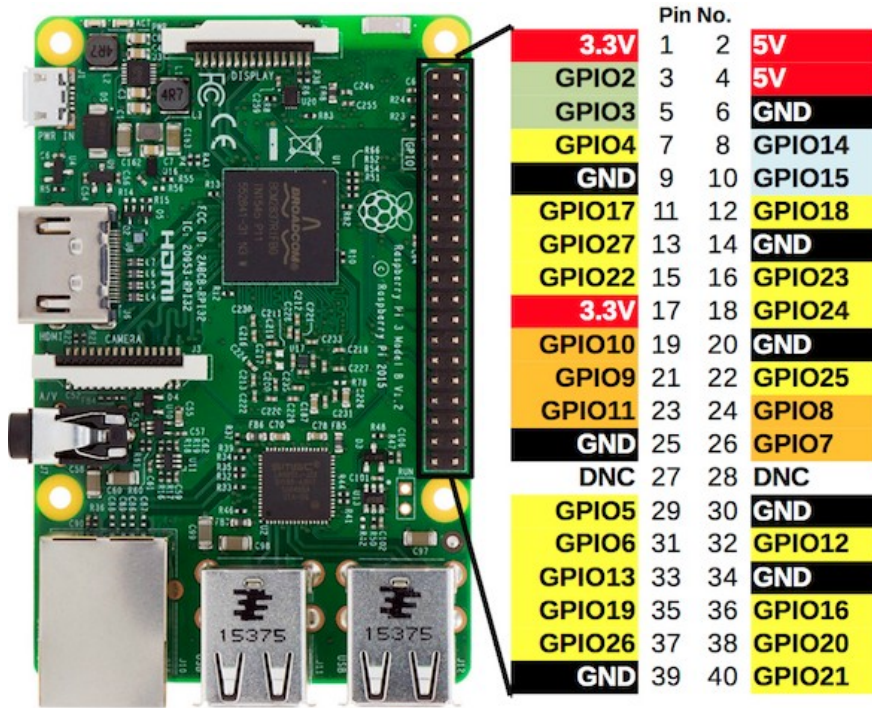
2. Build your temperature probes. There is a nice overview here:

<http://www.reuk.co.uk/wordpress/raspberry-pi/ds18b20-temperature-sensor-with-raspberry-pi/>

Solder attachments to allow power and ground connections to the GPIO and the data wires to GPIO4 with a 4.7 kOhm resistor soldered between the power and data wires.

1. Black wire → Ground
2. Red wire → 3.3V power
3. White (or yellow) wire → Data

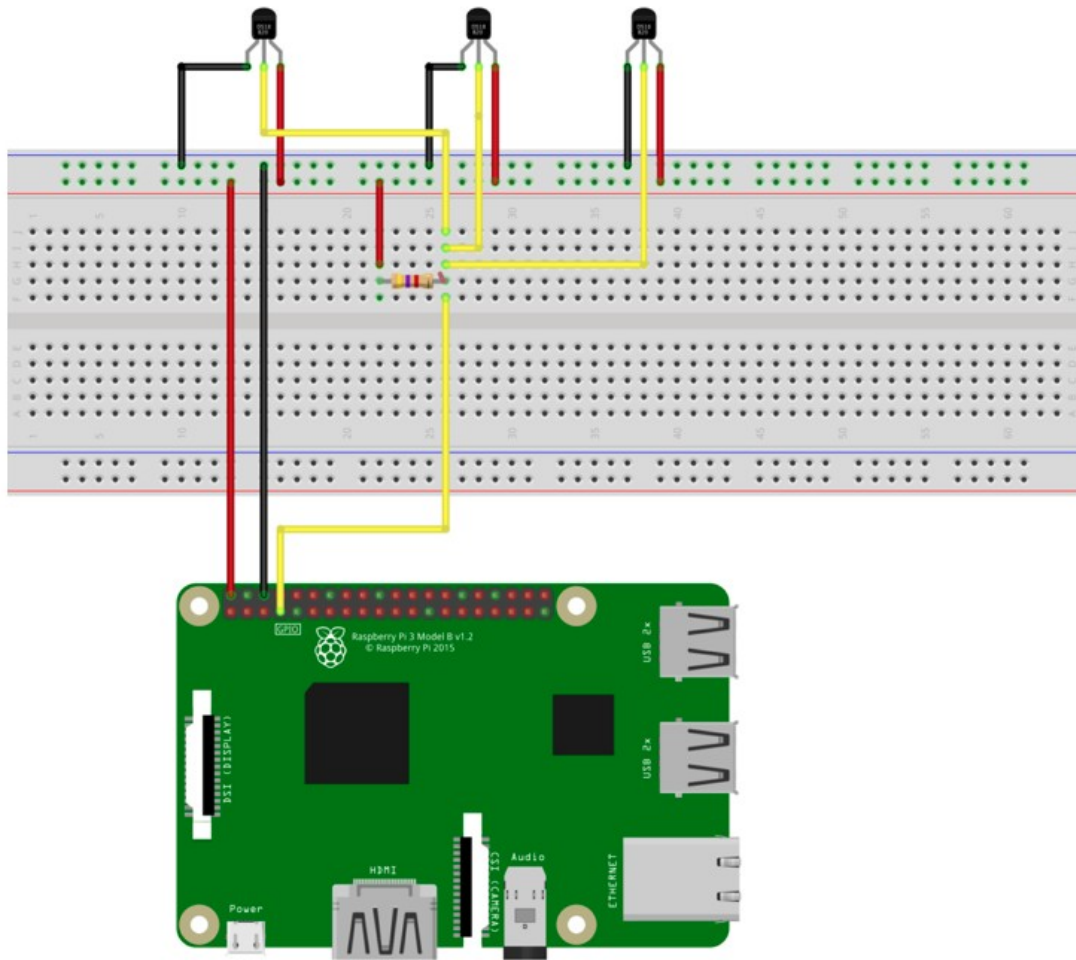
The GPIO layout on the Pi is as follows:



	Pin No.		
	1	2	3.3V
			5V
GPIO2	3	4	5V
GPIO3	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7
DNC	27	28	DNC
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GND	39	40	GPIO21

This can be done with a breadboard instead of soldering, eg. this image from

<https://raspberrypiautomation.com/connect-multiple-ds18b20-temperature-sensors-to-a-raspberry-pi/>



Need photo of two probes here ideally

3. Once soldering is complete (and cooled), startup the Pi and open ssh if you want to remotely access the Pi (recommended for checking on any problems down the line), easily done via the Desktop: <https://www.raspberrypi.org/documentation/remote-access/ssh/>
4. Ensure the GPIO4 pin is set to read from 1 wire sensors:
At the command prompt, type:

```
sudo nano /boot/config.txt
```

This opens the config file in a simple editor. Scroll to the bottom of the file and add the line:

```
dtoverlay=w1-gpio
```

Press Ctrl and X to save the amended file and reboot to ensure the Pi knows what that GPIO is for.

5. Copy the python script to a folder on your Pi, eg via an email attachment or mapping a network drive or plugging a USB stick in. This can be copied to your home folder. I created a thingspeak folder and copied the .py file there, e.g.: /home/piusername/thingspeak/all_ds18b20_andCPU_to_thingspeak_withEmail.py
6. Make the script file read write and execute for all users. To report the CPU temp, it needs to run with highest privileges I think. Do this by typing the following line at a command prompt, replacing with the path to your script:

```
Sudo chmod 777  
/home/piusername/thingspeak/all_ds18b20_andCPU_to_thingspeak_withEmail.py
```

7. Now edit the python script. You will see it needs the following parameters changing:
 1. For email:
 - email addresses for from and to (*Lines 35 and 36*)
 - email message header and body (*Lines 40 and 42*)
 - email host, port, server (*Lines 46 to 50*)
 - email login method and password. In this script it is setup for an Exchange server at our Uni- you will need to ask your IT Dept. for what this should be (or Googling it will probably be quicker). (*Lines 51 to 53*)
 2. Decide how frequently you want to report to ThingSpeak by changing the sleep value. Default is every 120 seconds (*Line 61*)
 3. Add the key value for your ThingSpeak channel: this is the API write code as mentioned in Software above. (*Line 62*)
 4. In the Thermometer definition, you need to set:
 1. MaxT. This is the room temperature in oC at which an email will be sent (*Line 79*)
 2. TimeBetweenEmails. This is the time in minutes it will wait before sending another email after the initial temperature breach, assuming the temperature is still high. It is set to 24h (1439 minutes). I did have it shorter, but if a temp goes too high, the account was being flooded with emails. (*Line 80*)
 3. The filepath for where the Temperature probes store their data. To find these, at a command prompt, type:

```
ls -l /sys/bus/w1/devices/
```

You will receive a list of what is in that directory. If your connections are good, you should have two folders with alphanumerical names like 28-0000xxxxxxx. In this script we have a room probe with ID 28-0000070fb272 and an incubator probe with ID 28-0000070feeaa. In the filepaths, change the folder name to match your two probes, ensuring you leave the subfolder /w1_slave in the path.

Which is which you ask? Place one probe in your grubby paw and see if it's temperature changes. To do this, at the command prompt, report from one of the probes directories:

```
cat /sys/bus/w1/devices/28-0000070feeaa/w1_slave
```

(ensuring you use your own probe ID for the directory!) This will give two lines output, the end of the second line has the temperature in oC multiplied by 1000. By retyping the same command (use the up arrow key to retype it automatically) you can report again- watch to see if it rises. If it doesn't, you guessed wrong hopefully. Swap to report the other probe using it's filepath and the cat command and check again. Once you know which is which, label the probes with a bit of tape for future ref with their ID and also which you have made them (room or incubator).

Back in the py script, make the probe you have chosen for the room the path for tempfile (Line 83) and the incubator probe filepath for tempfile2 (Line 116).

8. The script is set to report the room temperature to field1 in your ThingSpeak channel, the incubator temperature to Field2 and the CPU temp to Field3. These can be changed if desired in the line defining params (Line 125).
9. Now save the py script and make a backup of it somewhere else (in case your SD card dies).
10. Ensure the script continues to run in the event of a power outage etc. by adding it to cron.
<https://www.raspberrypi.org/documentation/linux/usage/cron.md>

At the command prompt type:

```
Sudo crontab -e
```

Then add these lines at the bottom of cron, where the path to py script is the location of the python script.:

```
@hourly /path to py script  
@reboot /path to py script
```

Everything should now be set. Restarting the Pi should now automatically start the python script after rebooting. It should then report to the ThingSpeak channel you set up. You can play lost with the visualisations in there, and choose whether to make public or not (I recommend as you can easily show users what the temperature is when they are working on the system). The disadvantage to this is there is no long term storage. ThingSpeak has a limit to how many datapoint it will store, so it doesn't work for historical comparisons beyond about a week or so with the intervals we have used here.