tilostat

SENSING AND IOT COURSEWORK

Matilda Supple

Dyson School of Design Engineering | MEng Design Engineering

Module Exam

Module code and Name	DE4-SIOT Sensing & IoT
Student CID	01112643
Assessment date	10th Jan 2019

Presentation URL (publicly accessible link):
https://github.com/tillysupple/tilostat/tree/master/Video

Webpage Code & Data Analysis (publicly accessible link):
https://github.com/tillysupple/tilostat

Data:
https://docs.google.com/spreadsheets/d/1TQulUsmQa88eC1JoYR_MXuvi-K9uQxDGPhP59njFBBo/edit?usp=sharing

Coursework 1: Sensing

Introduction

Summary

Using a combination of physical sensors and data from public APIs this system senses the temperature of a family home living room and collects other relevant data that may influence it. The APIs used are the Open Weather Map API, to access current weather data, and the Nest Learning Thermostat API, allowing access to read information from specific Nest devices.

Objectives

The objective of this project is to identify key trends between outside temperature, Living room temperature and Nest temperature to improve the usage of the Nest device.

A key aim is to present the information gathered in a clear, user friendly and attractive way to the user as, currently Nest users are only presented with limited insights as to their home temperature trends.

Sensing Set up

Overview

The DS18B20 waterproof sensor. This sensor was chosen as it is accurate within 0.05°C of the true temperature. It also has an analogue to digital converter so it could be easily integrated with the digital Raspberry Pi GPIO pins.

Raspberry Pi

A Raspberry Pi was purchased, Raspbian was installed and it was connected to the internet. The IP address was found which then allowed remote access from a laptop via an ssh connection. This increased the efficiency of iterating the code and made the implementation easier.

It was necessary to continually run the scripts over seven days. Using a laptop to run these was not feasible as, if being used for other work simultaneously, the data could be tampered with or risk becoming disconnected. The Raspberry Pi provided a good solution, hosting and running the scripts at regular intervals without being turned off or touched over this period.

Sensing Circuit

The circuit includes the DS18B20 sensor and a 4.7k ohm resistor, connected to the Raspberry Pi ground, 3v3 power supply and BCM 4 pins in the configuration shown in Diagram 1.

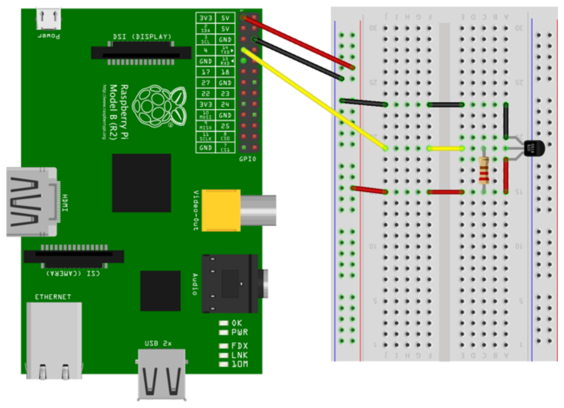


Diagram 1: Circuit and Raspberry Pi GPIO Pin Layout

Following set up, the 1-Wire protocol was then enabled on the Raspberry Pi and the ID was found to query the sensor.

The Raspberry Pi and sensor circuit were located on shelf in the centre of living room to remain untampered by the occupants however also accurately reflected the temperature of the room and the changes. Ideally this system would be mounted to a wall and work without needing to be plugged in to a mains supply.

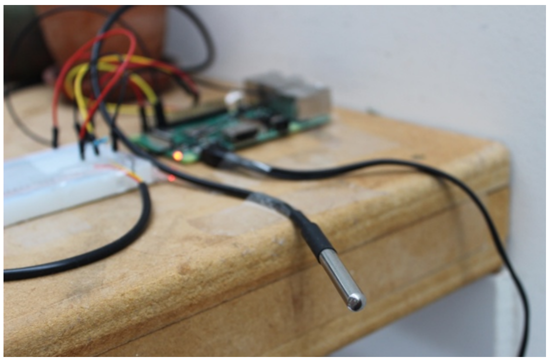


Diagram 1: Circuit and Raspberry Pi GPIO Pin Layout

Data Sources

Sensor Data

When the script `Data_collection.py` runs it opens the device file on the Raspberry Pi and reads the value from the temperature sensor. This data is then processed the in the `read_temp_raw` function to output the relevant temperature value.

Weather Data

The next section of the code calls the OpenWeatherMap API. To set this up a developer account was made and an API key was received and added to the `URL` that calls by geographical location, in this case the latitude and longitude of the house where the sensor is located. When called it outputs a JSON containing the weather data at the specified location. In the script an http request is sent to get the data from the URL It then interprets the data from the JSON using the `'main'` key.

The outside temperature and weather conditions are then found and the temperature is converted from Kelvin to Celsius two remain consistent with the data received from the sensor.

Nest Data

To access data from the Nest thermostat a developer account was created and connected to the account associated with the device for security. A client was then created, specifying the permissions it required. The permissions required for this projected were reading temperature and energy data however in the future it would also require the ability to write, to adjust settings based on the data acquired.

Figure 1: Nest Thermostat

The nest thermostat was then added to the account and a device authorization PIN was obtained as well as a `Client ID` and `Client Secret`. When authorising the device with the PIN a JSON is downloaded to the device, containing an access token for future calls.

The nest module was installed on the raspberry pi and imported in the script. The script then uses the nest module to call the API with the `access token`, `Client Secret` and `Client ID` and read the data. The specific values desired were then separated into variables through the '`device' key`'. The nest temperature and the target temperature are outputted as well as humidity, the eco-setting and how long until it will take to heat to the target temperature.

The last part of the code outputs a time stamp so that each time the code is run the data and time are recorded.

Data Storage and Collection

Google Sheets

For reliability Google Sheets was used to store the data. As it is cloud based it reduced the risk of data loss compared to if stored on the Raspberry Pi SD card.

First a spreadsheet was created, with the correct headers in place. A new project was then created in the Google Developers Console and the Google Drive API was enabled. To access the API a JSON was obtained containing the `client ID`, `client email`, and a `private key.` The spreadsheet was then shared with the client email from the JSON.

On the Raspberry Pi, `oath2client` and `PyOpenSSL` were installed and `gspread` was imported in the script. At the end of the script the variables from each of the data sources are appended to the spreadsheet. Every time this script is run a new line updates.

Scheduling

`Cron` scheduling was used to run the script at specific times. Crontab is a file on the Raspberry Pi which contains the schedule of cron entries to be run. This was edited to run every 5 minutes, each time appending to the google sheet.

5 minutes was selected as the OpenWeatherMap API updates every 10 minutes. Sampling at double this frequency complies with Nyquist Sampling theory and ensured that the data was accurately recorded.

System Overview

In summary this system uses a Raspberry Pi to run a script that makes calls to the Nest, Google Sheets, and OpenWeatherMap APIs every 5 minutes along with recording temperature data from. As it collected it is published to a Google Drive Spreadsheet row by row to be Analysed.

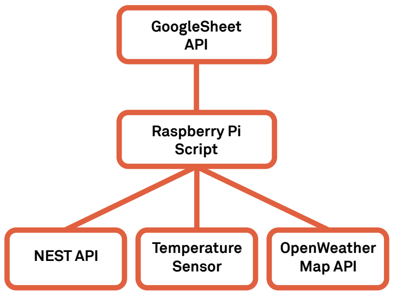


Figure 2: System Links

Coursework 2: Internet of Things

Data Visualisation Platform

Output

At present the data collected is able to be visualised through a web application, currently hosted on GitHub. The webpage also presents attractive and understandable data analytics as well as key insights obtained from the data that, if relayed to a user, could improve the usage of the Nest Thermostat and tailor it better to the individual rooms of the house.

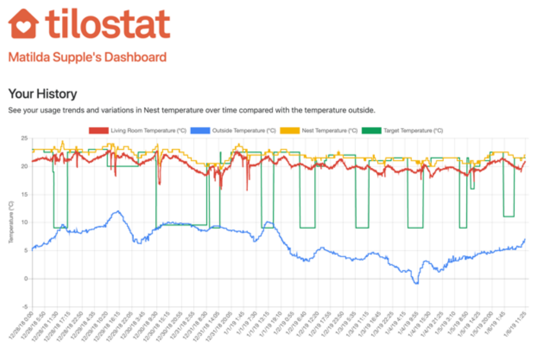

tilostat
Matilda Supple's Dashboard
Your History
See your usage trends and variations in Nest temperature over time compared with the temperature outside.
Living Room Temperature (°C)
Outside Temperature (°C)
Nest Temperature (°C)
Target Temperature (°C)
Temperature (°C)

Figure 3: The Web Application

System Overview

The webpage runs from a repository on a GitHub account. This contains an html file called `Index.html` which is the default page shown on the website. It contains html code that specifies the layout and design of the page and runs a script, also in the repository, called `Charts.js`. to load charts and graphs onto the page. `Charts.js` is a JavaScript file that uses the Chart.js library to present and animate data in different forms. The data used in the JavaScript file is loaded from a Jupyter Notebook called `Data Analyis.ipynb` that produces lists of data to be plotted based on analysis done previously.

Set Up

A new repository called tillysupple.github.io/tilostat was created and cloned to the GitHub Desktop Application. It was then placed in a file on the computer and could be worked on locally, pushing major changes to the official web page periodically. A `bootstrap` template was then downloaded an added to this file and an `Index.html` file was created and linked to the bootstrap stylesheet. Basic html headings, text and canvas' were inserted as well as a grid layout. All code was written using the Visual Studio Code editor.

The `Charts.js` file was written in JavaScript using the Chart.js library. It contains all 5 main plots, each related to an element id `document.getElementById()`, specified in the `Index.html` file so that it is uploaded to the correct canvas on the webpage. In the html file the chart.js library is linked as well as JQuery so that it can access the JavaScript file information.

The chart data was loaded from a Jupyter Notebook called `Data Analyis.ipynb`. In this notebook the collected data is read from the csv with the argument `parse_dates` to be interpreted by the Pandas Module, as shown in figure 4.

It was then cleaned using `fillna()` and visualised at the specified frequency of 300s (5 minutes) at which it was collected using `asfreq()`. The data was split by week, by day and by hour and averages were collected to understand trends. Correlation was found as well as outliers, peak detection and overall averages.

	Inside Temperature (°C)	Outside Temperature (°C)	Weather Description	Nest Temperature (°C)	Target Temperature (°C)
Date_Time					
2018-12-28 00:05:00	20.937	5.37	mist	22.5	23.0
2018-12-28 00:10:00	20.937	5.37	mist	22.5	23.0
2018-12-28 00:15:00	21.000	5.37	mist	22.5	23.0
2018-12-28 00:20:00	20.937	5.36	mist	22.5	23.0
2018-12-28 00:25:00	20.937	5.36	mist	22.5	23.0

Figure 4: First 5 rows of the csv

Data Analytics

Raw Data:

The raw data collected was plotted using the Pandas python module, this spanned over 11 days shown in Figure 4. Over this time clear trends in temperature were displayed, for example the outside temperature began to decline as the days progressed. It also showed the inconsistency of the nest learning thermostat in setting target temperatures. The large fluctuations are the result of improper use and therefore learning. This information is useful to the user as they can reset their device.

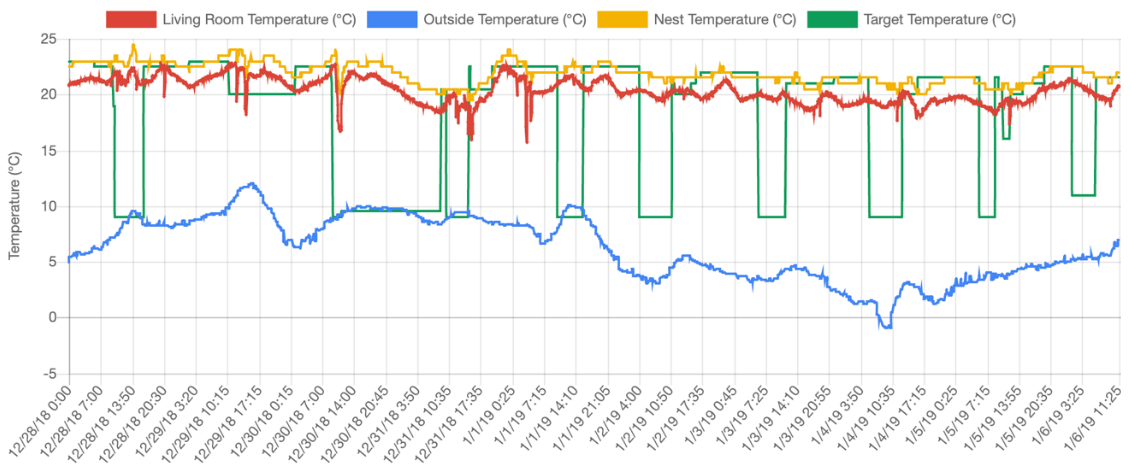


Figure 5: Raw Data Plot

The plot also clearly shows how the house temperature fluctuates, represented by both the living room temperature sensor and the nest thermostat. Interesting peaks and dips are caused by the sudden temperature changes when a door or window is opened. From this data you can detect the number of times the door is left open. This was implemented with the `Detect_outlier()` function.

Lastly this graph shows a consistent difference between the living room recorded temperatures and the nest recorded temperatures. This suggests that if the user wants the nest to match their living room conditions more accurately it should be repositioned. It also gives an indication as to how well insulated the living room is. In this case there is a large window that could be responsible for this temperature difference.

Analysis 1: Average Temperatures by Day

The second plot presents the user with their weekly averages, shown in Figure 5. Here a weekly average temperature of 22°C was found on the Nest and 21°C in the living room using the `Average_temperature()` function. This shows the temperature has stayed consistent despite changes in outside temperature.

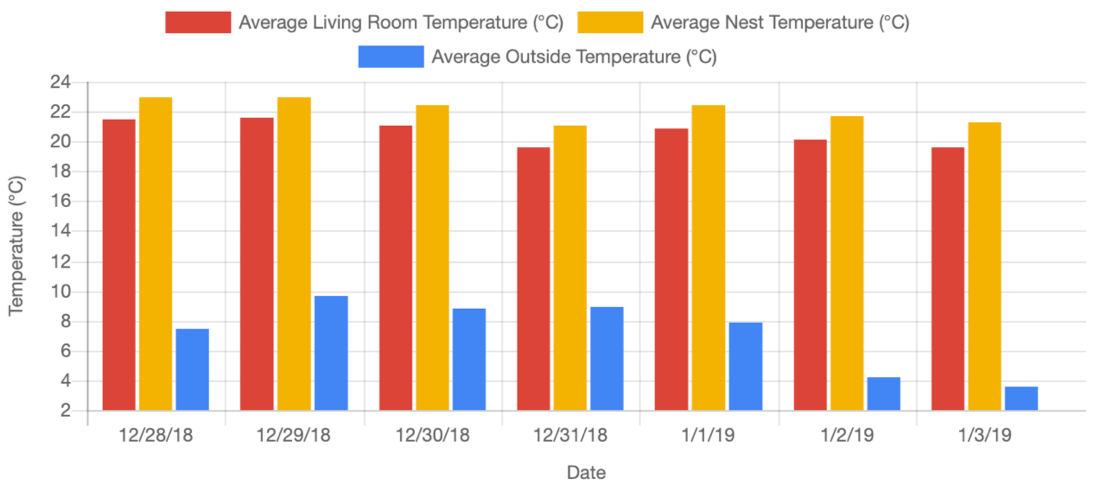
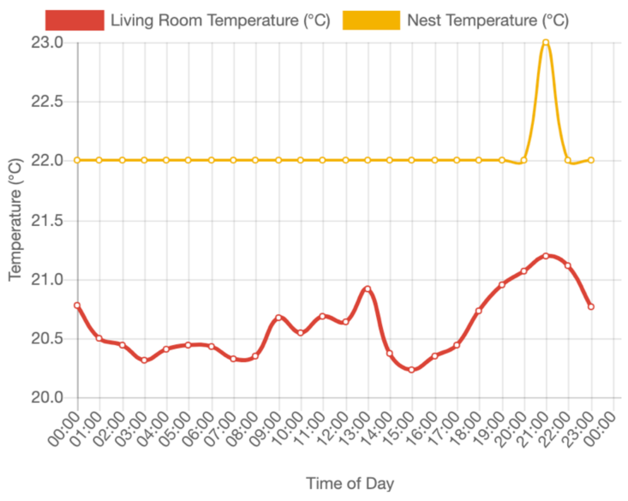


Figure 6: Average daily temperatures

This information helps the user understand their most comfortable temperature but could also be used to remind users that if they reduce their averages they could make financial savings.

Analysis 2: Nest Efficiency

The average hourly Nest temperatures and hourly living room temperatures were plotted to display the average difference between them over a day, shown in figure 6 and 7. This showed an average difference of approximately 2°C. This was displayed alongside a chart showing the percentage of the time over the week that the Nest temperature matched the Living room temperature using the `Average_difference()` and `Same_values()` functions.



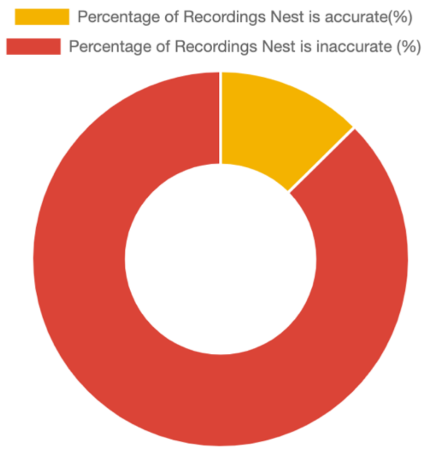


Figure 7 and 8: Nest Efficiency

It shows that the Nest rarely gives an accurate representation of the main room in the house. The nest could therefore be recalibrated in this case or the if the user requires a specific temperature in their living room they should set their Nest thermostat to 2°C higher than their desired temperature.

Analysis 3: The Affect of Outside Temperature of Nest Efficiency

To investigate the reasoning behind the difference between recorded living room temperatures and Nest recorded temperatures, the average daily difference in temperature was plotted against the average daily outside temperature, shown in figure 7.

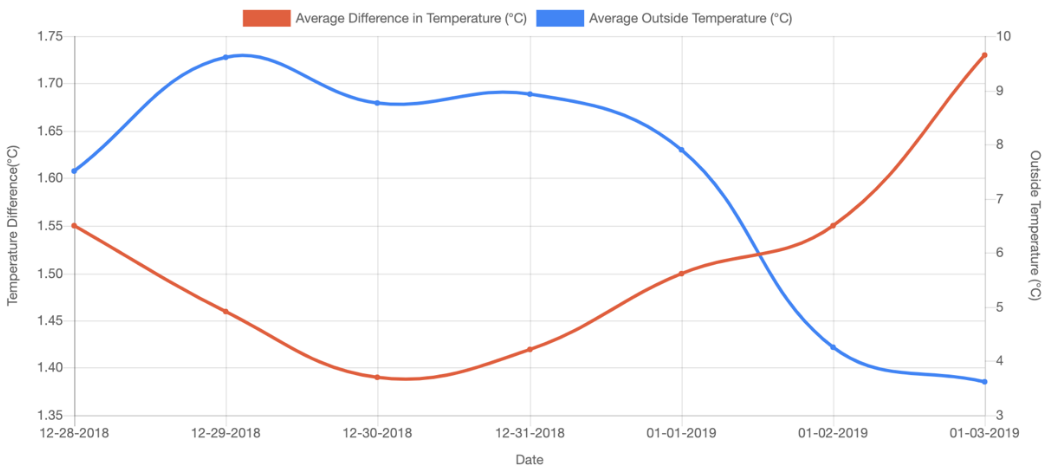


Figure 9: Average Reading difference against Average outside Temperature by day

Over the seven days it showed that the lower the temperature outside, the greater the difference between the Nest and the Living room recorded temperatures.
The correlation between difference in temperature and outside temperature was found to be -0.8550863. This strong negative correlation shows how the room responds to the outside conditions, giving an indication as to how well the room is insulated.

Future Implementation

From the analysis there were some key insights that could lead to further development of a useful system. This system would include a webpage/app and a hardware component.

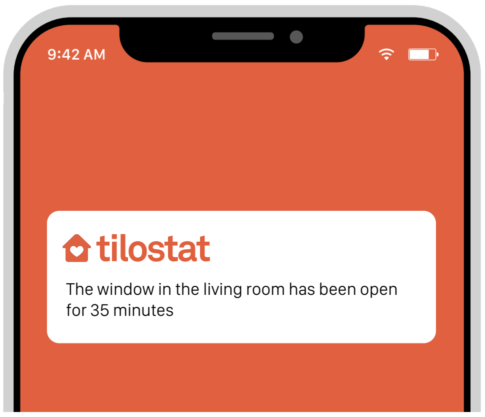
The Hardware

The hardware component would be similar to the current Raspberry Pi and temperature sensor circuit configuration, however wireless (using batteries), smaller and mountable. This could be mass produced to be less expensive and a user could purchase multiple for each room in the house. The data would be sent to a cloud storage service, similar to GoogleSheets, with an additional column for each room for location based temperature data analysis through the app.

The App

The app would update in real-time giving users stats about their habits and information about their individual rooms. This could be implemented through calling the GoogleSheets API and reading the data directly as the CSV updates and then analysing and plotting the data in JavaScript, instead of Jupyter notebook to be directly uploaded to the website as it comes in.

Currently the app only allows users to observe their trends however using the Nest API's write functionality they could be able to change their nest settings through the app based on the data they observe. The app could also be mobile based to allow the user to turn off the heating on the go, but also provide notifications. Examples of notification could be example the door has been left open for a significant time or the temperature in the Living room is higher than average.



The window in the living room has been open for 35 minutes

Figure 10: Example of App Notifications

Potential Impact

Small Scale Applications

This system could be useful when setting up the nest thermostat initially, seeing how the location of the thermostat effects the representation of different rooms.

Another application could be to understand how well insulated different rooms are. The differences in temperature around the house could be recorded for a week and analysed and thermally insulating methods (eg. Double glazing, filling floor gaps) could then installed accordingly.

Possible Future Applications

The most interesting information gathered from this project is the correlation between difference in temperature between the nest and the living room and the outside temperature. This could be used to create a function that calculates the temperature that the Nest should be set to obtain the desired living room temperature based on the weather forecast. The desired living room temperature would be acquired from the calculated averages in the Data Analysis section.

Discussion

Currently the interface on the Nest gives the user limited understanding as to their temperature trends or how the device is learning. Presenting the user with the tilostat application, that presents the data in a user friendly, clear format with key insights, would be beneficial to Nest Thermostat users and improve the functionality of the device.

It is however, important to consider the time frame of the recorded data being 11 days. This project would gain more validation if left to record over months.

The temperature values are very dependent on the season. Therefore it would be expected to have completely different results in Summer. If this app were continued it would present different dashboards depending on the month and also incorporate the Nest cooling functionality.