Inferring Occupancy Patterns through Environmental Sensor Data

Michael Rumley

School of Computing Science, Newcastle University, UK

Abstract

Being able to determine the occupancy of a room is vital in implementing smart-building policies. Using an occupancy sensor is a common way of doing this, however occupancy data can be prone to false readings. This project attempts to measure the occupancy of a building using other data; environmental readings such as CO2, temperature, brightness etc, and compares how closely correlated they are to actual occupancy data.

Keywords: Smart Buildings, Occupancy Detection, Environmental Sensors

1. Introduction

1.1 Motivation and Rationale

In lieu of any architectural advancements, to innovate in the modern building industry one must focus on making a building 'smart'- using sensors and actuators in accordance with the internet of things to increase a building's efficiency.

Being able to accurately determine the occupancy status of a room is an important concept in the design and development of smart buildings. If patterns of occupancy of a room or area is known then specific policies can be implemented to enhance the user experience, increasing comfort and decreasing energy usage. Determining occupancy data can be achieved with the use of an occupancy sensor, however of interest in this project is the concept of using data form a variety of environmental sensors, and the effects that occupants would have upon said data.

Also of interest to me in this project are the security implications of being able to track occupancy using publicly identifiable data. Depending on the accuracy of using an environmental metric to measure occupancy, additional methods can be investigated and implemented to obfuscate such information from potential attackers.

1.2 Aim

To investigate, implement and evaluate a way of visualising the environmental data of a smart building, and use this data in accordance with the occupancy data to assess the accuracy and viability of using an environmental metric to measure occupancy.

1.3 Objectives

The above aim can be broken down into the following 4 objectives:

1. Investigate existing approaches to measuring occupancy detection
2. Extract data from Urban Science Building API
3. Plot data and compare to occupancy sensor data
4. Evaluate results and assess accuracy of using environmental data

1.4 Paper Structure

The contents of each section of this report are as follows:

Background: An introduction to the notion of Smart Buildings, Occupancy Detection and environmental sensors.

Implementation: This section discusses the design decisions that were made in development of this project, as well as information on the key software involved and how it was utilised. Then a full explanation of the code is presented.

Evaluation: This section presents and analyses the results collected and provides insight into the causes of any discrepancies that may arise.

Conclusions/Further Work: This section assesses whether the project was successful or not, incorporating personal reflections on the experience as well as comparing the final product to the initial aims and objectives. Further extensions to the project are then proposed that could provide interesting results should additional features be implemented.

2. Background Research

This section will aim to provide information about the major concepts that underpin the work that has been completed in this project. It should be noted that the research undertaken for this section was done in parallel with the research for another module, CSC8205 Research Skills, and as such a longer report with more information on different sensor types can be read as the report submission for that module.

2.1 Smart Buildings

A smart building can be defined as a building controlled by a Building Automation System (BAS), specifically a data- rich one [1]. A BAS centralises control of a building's systems, usually heating, ventilation, air conditioning (HVAC) and lighting, but also can include CCTV, elevators, access control, plumbing etc. Converting a regular building into a smart one carries with it several benefits, primarily the reduction in energy usage and operating costs, but also the enhancement of the user experience and comfort as well as general building efficiency. A typical smart building uses a wireless communication network to communicate between different devices and appliances, as well as the building's outside surroundings, using sensors, actuators and controllers, with sensors reporting changes in data to the controller that then enact various smart building policies that have been programmed(either general policies for the whole building programmed by a building manager or more personal rules by individual users) through a central unit and are carried out by actuators that perform physical actions [2]. Smart buildings typically use two-way smart meters to provide bidirectional communication between a utility usage and end consumer; providing real-time data that can be accessed either on-site or remotely.

2.2 Occupancy Detection

Being able to determine the occupancy status of a building is vital to implementing any smart building policies. For example, if a room is unoccupied, then there is no need to light or heat it and as such the overall energy consumption of the building would fall. The problem of locating a user within a building is known as the occupancy detection problem and is still an open problem in computer science [3]. The most common solution to this problem is by using Passive Infrared Sensors (PIR), that work by using a photoelectric material sensitive to infrared and a Fresnel lens that splits the infrared range into distinct zones, so that when a temperature change is detected between zones a reading is produced [4]. Therefore this kind of sensor measures movement rather than occupancy and would be prone to false readings from external forces such as passers-by or wind.

Another way of measuring occupancy is to use Radio Frequency Identification (RFID) [5], which provides a continuous reading unlike PIR, although with significantly higher installation costs, since all users would need to be given a tag containing a circuit and antennae that would broadcast data to a RFID reader. RFID's short detection range leads to several problems in existing occupancy detection solutions.

Finally an extremely accurate way of measuring occupancy in a room is through using video cameras. This method not only provides occupancy data but detailed information on the identities of each occupant, which then generates problems regarding security and privacy. To maximise accuracy the videos would have to be analysed by hand, which is a tedious and time-consuming process. Solutions have been proposed [6] that use specialised software to identify 'blobs' in camera footage, but they have a high initial cost, and so video cameras are only typically used in occupancy detection to obtain ground truth data.

2.3 Environmental Sensors

An increasingly popular way of measuring occupancy is by combining several environmental sensors into a hybrid sensor. The data is then used to train machine learning models and neural networks to provide occupancy predictions. The environmental sensors relevant to this project are as follows:

- CO2 Sensor

The most common method of detecting carbon dioxide is to use a Non-Dispersive Infrared (NDIR) sensor. An NDIR sensor works by detecting the absorption of the CO2 by diffusing the gaseous environment into a light tube and using a detector to measure the attenuation of the different wavelengths that make up the gas (CO2 has a sensing wavelength of around 4.26 µm [7])

- Temperature Sensor

Most methods of sensing temperature rely on measuring a property of a material that varies with changes in temperature, most commonly a standard thermometer that uses a liquid (typically mercury) as a working fluid that expands as it warms up, so the temperature can be measured by calculating the volume of the liquid. In the context of HVAC systems, temperature is measured using a thermostat, which rely on thermistors or resistance temperature detectors for temperature data since they have no moving parts. Thermistors work by changing resistance when subjected to a change in temperature [8].

- Humidity Sensor

Humidity is measured using a hygrometer, and is expressed as a relative value, the ratio of water vapour present in air to the amount needed for saturation at a particular temperature expressed as a percentage [9]. Humidity is directly tied to temperature, since cold air can hold less water than hot air (this means the same amount of water vapour will produce a higher relative humidity in cold air). A hygrometer can be either resistive (a change in electrical resistance of a material) or capacitive (a change in the dielectric constant).

2.4 Urban Sciences Building (USB)

The USB is an ongoing research project by Newcastle University, with the aim of making urban centres more sustainable for future generations. It contains facilities for research and education in computing, energy, water, digital urban sensing and infrastructure. As well as containing Newcastle University’s School of Computing Science, the building contains over 4000 sensors, representing the largest open platform of urban sensing data in the United Kingdom [10].

3. Implementation

This section will describe the decisions that were made in the design and implementation of this project. Once an appropriate dataset was chosen, the major steps involved were extracting the data, formatting the data to a format that could be accurately graphed, and finally displaying the data in a user-friendly manner. The dataset that was chosen was the USB dataset, due to the large quantity and variety of data being generated. All the programming tasks in the project were completed in the python programming language due to the ability to write complex methods with relatively little code, as well as the existence of several python libraries that provide useful functions.

3.1 Extracting data from the API (requests library)

Whilst the data collected by the USB can be accessed through API [11], the volume of data, as well as the regularity at which it is updated, means that obtaining a dataset small enough to graph is a challenge. The python requests library [12] has been used to make get requests from the API.

```
requests.get(https://api.usb.urbanobservatory.ac.uk//api/
                    v2.0a/sensors/entity)
```

The above will return a requests object containing a paginated list of all entities (an entity describes a location – each room in the USB has at least one entity) in existence, however in this form the data is unusable. It was decided that a user could specify the room number and sensor type, and then the readings over the last 24 hours are returned for both the specified metric and the occupancy sensor. The API allows for those variables to be input as parameters, which are taken by the requests library as additional arguments:

```
...params={'meta:roomNumber':roomNo,'metric':metric})
```

Now that the returned object has been considerably shortened (ideally there would only be one entity returned – in bigger rooms with multiple entities an assumption is made that only the first entity is considered), the JSON data in the requests object is decoded.

Using the structure of an entity object, the next step is to extract the ID of the timeseries associated with the required metric to access the historical data times and values (the structure of an entity is explained below)

```
Entity {
      Feed {
            Timeseries {
                  TimeseriesID
                  TimeseriesEntity {
                        value
                        time
```

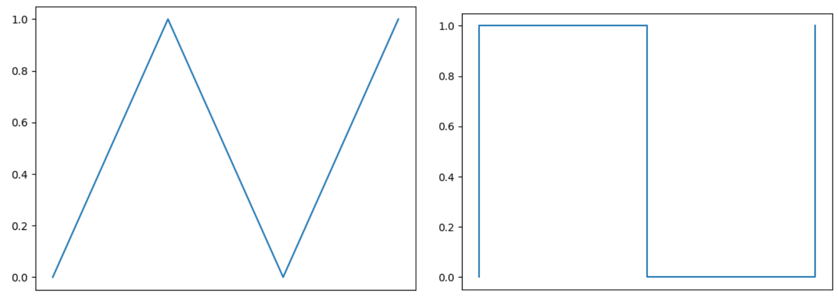
An entity will include an array of feeds, one for each sensor present (different entities will have different amounts of data available). The purpose of the metric parameter in the initial get request is to cut the amount of feeds down, as with the entities per room only the first result will be considered if dealing with multiple sensors collecting the same value. Within a feed there is an array of Timeseries objects; the first one being the most recent data. However the only TimeseriesEntity object contained here is the most recent reading, so an additional get request must be made using the TimeseriesID to obtain historic data;

```
requests.get(https://api.usb.urbanobservatory.ac.uk//api/
       v2.0a/sensors/timeseries/TimeseriesID/historic)
```

The above request returns the associated Timeseries object as detailed above, as well as a ‘historic’ object, containing an array called ‘values’, an array of TimeseriesEntity objects representing all values recorded over the last 24 hours (different time scales can be obtained but the default range of 24 hours sufficed here). This process is repeated for both the user-requested metric as well as the occupancy sensor, which is plotted on all graphs and used to find a correlation figure. For each TimeseriesEntity the value and time are appended to their own separate arrays to be used later in graphing. While the requested metric can be lifted directly, the occupancy sensor data needs some manipulation due to the difference in the way the 2 sensors collect data.

3.2 Augmentation of Occupancy Data

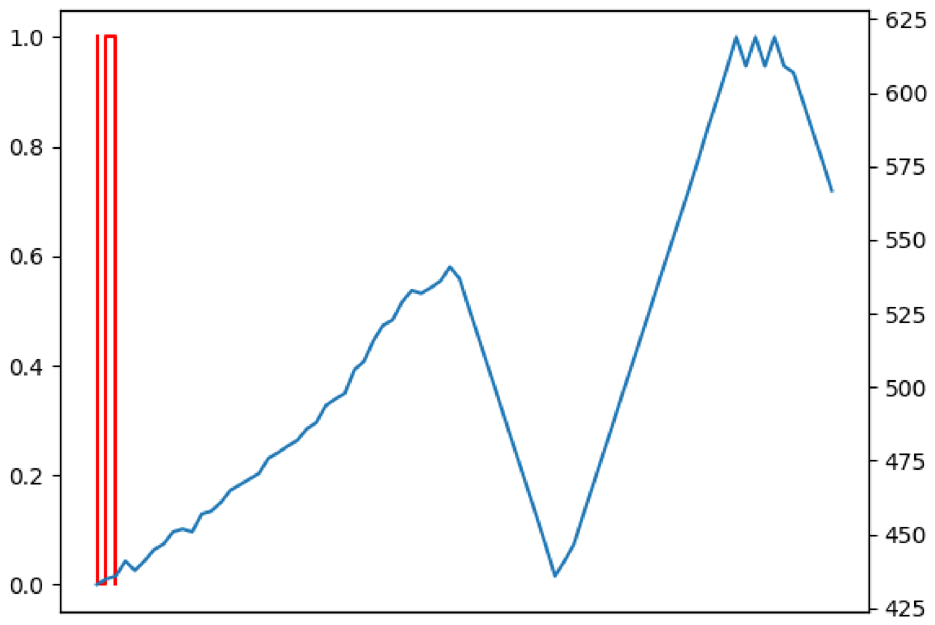
As mentioned in the background research, occupancy sensors and environmental sensors collect data in different ways. While an environmental sensor provides continuous data, transmitting a reading at regular time intervals, an occupancy sensor only provides a reading when a change in occupancy is detected. A value of 1 indicates the room has moved from being unoccupied to occupied, whereas a value of 0 indicates the reverse. Displaying these values graphically does not represent the data correctly (as shown below)



Unmodified Occupancy Data vs Modified Occupancy Data (roomNo:3.032)

Representing such binary data on a line graph like this does not account for the values between the readings- the value will only ever be 0 or 1. To remedy this problem, each time value is duplicated, and a value is added. For a value of 1, a value of 0 is added beforehand, whereas for a value of 0 a value of 1 is added. By doing this a more accurate graph is produced that correctly represents the data.

Now that the values have been correctly represented, attention must now shift to the time values. By pulling the time values from the API, the times are represented as strings, which means there is no sense of scale. While this is not an issue in representing the environmental metric, since there will always be the same amount of time values for a given period, in the occupancy data proves to be more of a problem. Since there would typically be less individual occupancy time readings, a graph is produced like the one below.



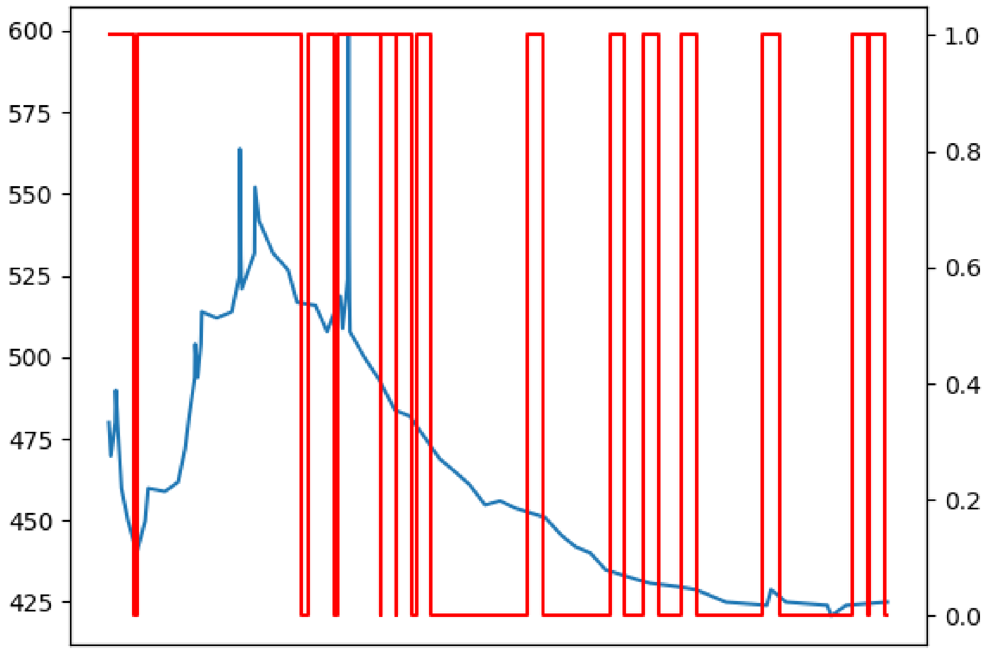
Occupancy data plotted against an environmental metric where time is represented as a string (roomNo:3.032,metric:CO2)

3.3 Normalisation of Dates (datetime library)

The datetime module [13] provides classes to manipulate date and time. By converting from a string to an ‘aware’ datetime object using `strptime()` an integer representation of said object can then be obtained using `timestamp()`.

```
d= datetime.strptime('2019-05-14T04:19:48.783Z',
                      '%Y-%m-%dT%H:%M:%S.%fZ')
            >> 2019-05-14 04:19:48.783000
                 d.timestamp()
              >> 1557803988.783
```

These values are then added to their own array and are used as the x axis data when plotting the occupancy data. By doing this the data is accurately presented as points in time. Finally the decision was made to extend the occupancy data to the range of the environmental data to provide a better-looking graph and a more accurate interpolation function in later stages of the project.



Occupancy Data plotted against an environmental metric where time is represented as an integer(roomNo:3.031,metric:CO2)

3.4 Correlation

The next part of this project is using the gathered data to calculate the correlation between the occupancy data and the requested metric. As explored earlier, there are more time/value pairs for the metric than the occupancy, so extra values must be interpolated to accurately correlate the datasets.

3.4.1 Normalisation of Values (sklearn library)

The sklearn library [14], in particular the preprocessing package, provides classes to change data into a representation to be used later. The MinMaxScaler method is used to scale data in a given range, which is fitted to and then transformed upon a dataset.

```
data
>> [[-1, 2], [-0.5, 6], [0, 10], [1, 18]]
MinMaxScaler (). fit(data). transform(data)
>> [[0.0.], [0.25 0.25], [0.5 0.5], [1.   1.]]
```

By normalising all 4 of the arrays in this project, all the values fall between 0 and 1, which prepares them for additional manipulation.

3.4.2 Interpolate Additional Occupancy Data (scipy library)

Scipy [15] is a collection of mathematical algorithms, and the interpolation subpackage is used in this project to approximate a function based on the existing 1-dimensional fixed data points in the occupancy time and values array. This function can then be applied to the larger set of metric times to produce an estimate of an occupancy value at each metric time.

```
f = interp1d(OccTimes,OccValues)
InterpolatedOccValues= f(metricTimes)
```

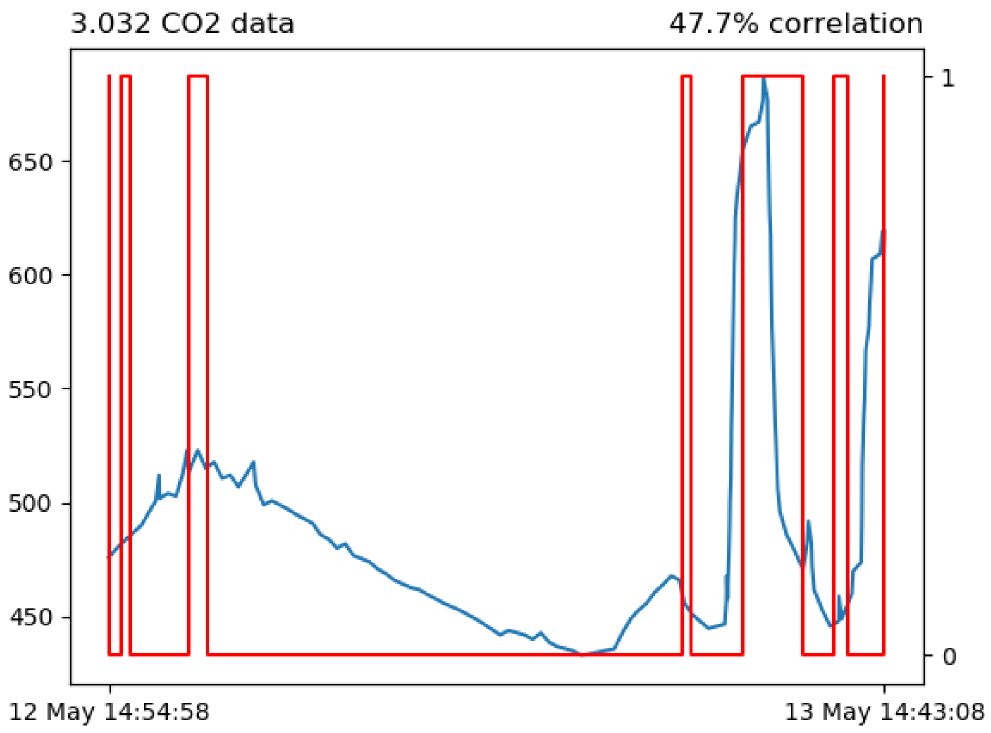
3.4.3 Calculate Correlation value (numpy library)

Numpy [16] is a library concerned with operations on arrays. The numpy.corrcoef function is used in this project to calculate the Pearson product-moment correlation coefficient of the 2 sequences of values.

```
a= [1,4,6]
b= [1,2,3]
corrcoef(a,b)
>> [[1. ,0.99339927], [0.99339927, 1.]]
```

3.5 Representing Data Graphically (matplotlib library)

Matplotlib [17] is a 2D plotting library that allows for generation of graphs. At it's most basic level, a pyplot module in matplotlib provides a state-machine interface to an object-oriented plotting library. Passing 2 arrays to `pyplot.plot()` will plot the 2 datasets against each other, with the state machine automatically generating figures and axes to achieve a visually satisfying representation of the data. The graph can be further modified by specifying the axes, and in the case of this project 2 different sets of axes are generated using the `twinx()` method, which instantiates a new set of axes with an invisible x axis and an independent y axis parallel to the original one. From there, additional parameters can be programmed to change labels, colours, titles etc.



Final graphical output with all editing options (roomNo:3.032, metric: CO2)

4. Evaluation

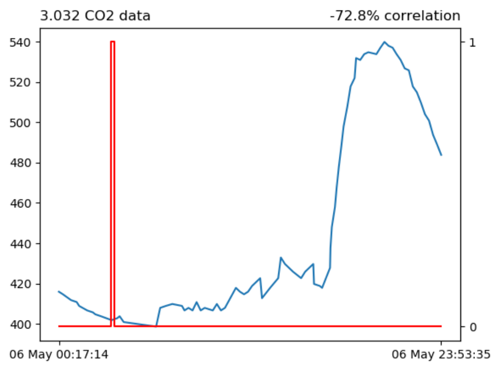
4.1 Correlation between Occupancy and Environmental Metrics

To calculate a correlation between the data generated by the occupancy sensor and the data generated by a particular environmental sensor, I recorded the correlation value as calculated using the methodology detailed in the previous section, obtaining a value for each day of the week from the 6th May to the 12th May 2019 for each of the 5 different rooms(G.069,1.017,2.015,3.032,4.036) I chose in the building. By calculating an average for each room over the week of data collection (henceforth final correlation) and then an average over the 5 rooms (henceforth average correlation), a final value was reached. A full set of results can be found in Appendix A.

4.1.1 CO2

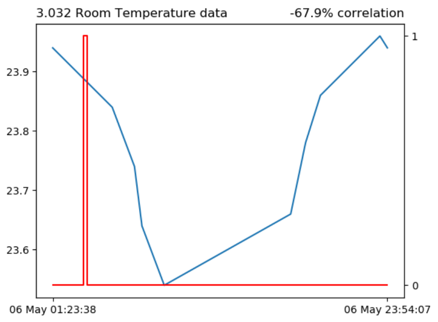
Average Correlation: 17.7%

The average correlation of CO2 is the highest of the 3 final values recorded, however it is still low, due to strong negative and positive correlations practically negating each other. Logically, CO2 levels and Occupancy will be positively corelated (as all final correlations recorded here are)- occupants will exhale carbon dioxide through respiration and increase the CO2 content in a room the more of them there are. Rooms G.069 and 4.036 best show this trend, both having a final correlation of around 25% and only having negligible negative correlation values. At it's peak, correlation values of between 50 and 60% were recorded, such as the value of 60.7% in Room 2.015 on 10/5. This was an active day for this room, with hundreds of values in both datasets and an accurate correlation calculated. Conversely, the largest negative correlations, such as the particularly egregious -72.8% correlation value for Room 3.032, tended to arise when there were very few occupancy values recorded for that day, and changes in CO2 data were caused by other sources, such as it's natural occurrence in air.

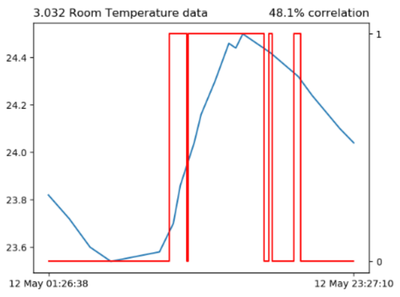


4.1.2 Temperature

Average Correlation: 13.1%



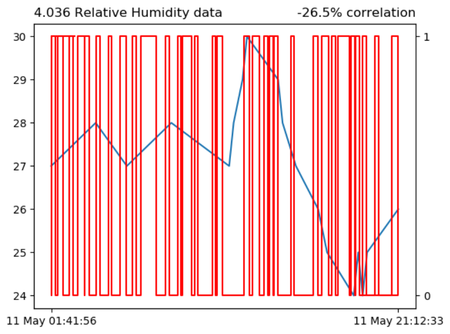
The calculation of an average value for Temperature/Occupancy correlation was affected by only having data for 3 of the 5 rooms, as well as the unpredictable behaviour of Room 3.032. Whilst the other rooms, G.069 and 4.036, behaved as one would expect, generally producing values in the 40-60% range across the week and low values (1% and 9.1% respectively) on a Sunday where a low occupancy is assumed, 3.032 performs almost entirely contrary to those expectations. The room produces strong negative correlation values all week (between -25 and -68%) and then a strong positive correlation on the Sunday (48.1%). In a similar fashion to the CO2 results previously, lower correlation is generally found when there is little occupancy, as with the Sunday values in the other 2 rooms. However in the case of Room 3.032 on 6/5, a single change in occupancy (if the room had remined empty all day the result would differ), as well as the slight dip in temperature, leads to the large negative correlation recorded. The more regular use of the room on 12/5 leads to a positive correlation as expected, but has little impact on the overall correlation due to the existance of 6 negative values from the other days.

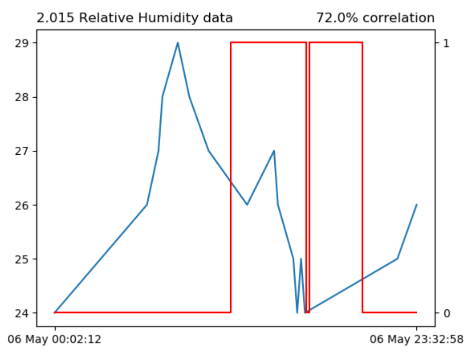


4.1.3 Humidity

Average Correlation: 4.7%

These results follow the trend of the rest of the project, once again consisting of moderate positive and negative correlations cancelling each other out to produce the low average correlation. While it was assumed that the humidity results would themselves correlate with the temperature data, considering warmer air would hold more water and then produce a higher relative humidity value, the





highest final correlation of 7.9% is achieved by Room 3.032, the worst scoring room with regards to temperature. The humidity values have a smaller range than the other data, the lowest values in the -35% to -40% occurring at the weekend in Room 2.015 (the temperature data for this room is unavailable so a comparison cannot be made here). Large discrepancies between temperature and humidity occur in Room G.069 on 8/5 and 10/5, and in Room 4.036 on 9/5 and 11/5. Unlike the other 2 metrics in this project, days with a lot of changes in the occupancy leads to a negative correlation, whilst when the occupancy dataset is small the correlation becomes more positive.

4.2 General Evaluation of System

Ultimately, the system does fulfil the initial objectives, providing a graphical representation of data accessed through the API of the Newcastle University Urban Science Building. In terms of data visualisation, my program provides a similar purpose to what is available online with the 3D data viewer [18] and begins to assess the relationship between sets of data. Since the occupancy sensor only provides a binary reading as to if a room is occupied or not, and environmental data will change based upon the number of people in a room, the correlation values calculated here may not be the most accurate. A room with 100 people in will contain more CO2 and be warmer than a room with one person, but an occupancy sensor considers both rooms occupied. This explains why in my results, generally rooms where the number of values in the occupancy dataset exceed the number of values in the environmental metric (representing a consistently used room), the higher the correlation, whereas lower occupancy values lead to larger negative correlations since while the changes in environmental data are caused by external factors outside the scope of this project, the sporadic occupancy values skew the dataset.

5. Conclusion

5.1 Fulfilment of Objectives

This project was split into 4 major objectives:

‘Investigate existing approaches to measuring occupancy detection’

Whilst I fulfilled this objective by performing an extensive review of existing methods of occupancy detection, and it proved useful in writing the literature review in the research skills module, the method of occupancy detection was chosen for me by the case study I chose to collect data from.

‘Extract data from Urban Science Building API’

I used the python requests library and the USB API to extract data. *‘Plot data and compare to occupancy sensor data’*

The collected data is presented graphically and compared to the occupancy data using the methodology detailed in Section 3.

‘Evaluate results and assess accuracy of using environmental data’

The results are evaluated in Section 4. Popular environmental metrics and rooms were used for evaluation, but the program contains functionality for a user to specify their own parameters for testing.

The overall project aim was to *‘…investigate, implement and evaluate a way of visualising the environmental data of a smart building…’*, and in this respect I feel I have provided a competent solution to this problem. The program has all the functionality I intended it to, with ample room to build additional functionalities and enhance existing ones.

5.2 Personal Reflection

When reflecting upon the work that I have undertook in this project, the only point of comparison I have in terms of scale and depth of work is the Project and Dissertation in Computing Science (CSC3095) that I completed in the 2017/18 academic year about particle systems in video games. Whilst the structure of the project is similar (a research into the project domain before producing a piece of software that must then be evaluated), the subject matter is completely different. Whilst in third year I was undertaking 2 game-development themed modules and working on my dissertation in parallel, the academic content feeding dissertation progress, as well as regularly meeting with project supervisors, this year my style of working became much more independent. Other than being briefly mentioned in the module on System Security (CSC8102), all knowledge of Occupancy Detection, Environmental Sensing and smart buildings was gained through my own research, further developing my research skills. The requirement of having the research component of the dissertation be a separate piece of work was very beneficial and allowed me to research more in depth than I would if it was just a section of the final dissertation. Furthermore, by deciding to work in python for the implementation of my software, a language I had brief experience with prior to university but only picked back up this year for this project, I now consider myself competent in another major programming language and generating original programs in it. This is unlike my project last year, where I was adding features to a C++ library, a language I was already proficient in. Therefore my programming skills also developed.

Despite previous experience with a Project and Dissertation module, I still felt that I spent too long deciding on a topic to cover and was left with a lot of work to do over the Easter break. I attribute this to the structure of the MComp course I am on, since unlike in third year, where dissertations were decided upon at the start of the academic year (indeed the structure of the module mandated 10 credits worth of progress in semester 1), this dissertation project was only introduced after semester 1 had ended. Whilst I understand having to work on a research project at the same time as challenging masters modules would be a lot of work, in hindsight I wished I had began working on this project much earlier. I also attribute my uneven workload to my own inability to choose a topic for this project and requiring a list of possible topics to choose from. To revisit my earlier point, if I had decided on a topic of my own accord then I would have been able to start work earlier.

However, if I did choose my own topic without assistance, I doubt I would have chosen occupancy detection, since it was an area of research I was completely unaware of, despite working in a smart building for almost 2 years. Over the course of this project, I have gained a great appreciation for this field of study and have a real interest in collecting and using data, particularly with regards to the security of collecting data, an aspect that I wasn’t able to feature in this project as much as I wanted to.

Another issue that once again arose in completing this project was my tendency to spend too long working on difficult tasks, or indeed tasks I perceived to be difficult, only to find out they had simple solutions and I had wasted time that could have been spent working on other aspects of the project. Solving problems like this requires more active time management on my part, as well as better utilisation of my dissertation supervisor and other members of staff. Not having mandated meetings with supervisors this year lead to me meeting with my supervisor only twice, which in hindsight I should have done more frequently to make my development time more efficient.

5.3 Further Work

Whilst the work I have submitted fulfils the aim and objectives of this project, there is a lot more work that can be done.

Simple cosmetic changes that can be made to the program include more user functionality- while currently the user must input both the desired metric and room to pull data from, there is no reason why the user could not supply timescales to access historic data, as well as being able to specify what zone they wanted to assess for bigger rooms with multiple sensor clusters. With so many variables introduced, user input could be moved from a command line interface to a friendlier GUI, showing a user the available rooms and metrics that can be accessed. Further visual enhancements could be made to the graph produced, showing multiple environmental data series on the same plot, or further inspect individual points by utilising pop-up menus when hovered over.

More complex additions can be made to the datasets themselves. As explored in the evaluation earlier, some problems in the result collecting arose due to the binary nature of the occupancy sensor. Therefore work still needs to be done on collecting data on the number of people in a room, such as a solution proposed using RFID tags [19]. Correlating this data with environmental metrics using the methodology contained in this report would produce a better result. Furthermore, using multiple environmental metrics would produce a more accurate result, as shown in the 'hybrid sensor' section of the background research and further in experiments such as [20].

Once accurate correlation data is collected, then the values can be used to train machine learning models such as neural networks and hidden Markov models to predict occupancy patterns. For example, if the occupant of an office typically comes in to work at a certain time, then the heating systems can achieve a desired temperature in time for when they arrive, and the lighting systems can be timed accordingly.

Collecting data in this way can lead to security concerns, and further work can be done to determine the minimum amount of data required to accurately predict a room's occupancy patterns, and 'fake' data can be included in the dataset to obfuscate data to potential attackers. A substantial extension to this project would be to determine the balance between accurate prediction of occupancy and the privacy of a building's user.

References

[1] www.kmccontrols.com/products/Understanding_Building_Automation_and_Control_Systems.aspx, last accessed 12/05/19

[2] Morvaj, B., Lugaric, L., Krajkar, S.: Demonstrating Smart Buildings and Smart Grid features in a Smart Energy City. 3rd International Youth Conference on Energetics 2011. July 2011.

[3] Conte, G., De Marchi, M., Nacci, A.A., Rana, V., Sciuto, D.: BlueSentinel: a first approach using iBeacon for an energy efficient occupancy detection system. Proceedings of 1st ACM Conf. Embed. Syst. Energy-Efficient Build. November 2014

[4] www.learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/how-pirs-work,last accessed 12/05/19

[5] Calis G, Deora S, Li N, Becerik-Gerber B, Krishnamachari B.: Assessment of WSN and RFID technologies for real-time occupancy information. In Proceedings of the 28th International Symposium on Automation and Robotics in Construction (ISARC 2011), June 2011

[6] Ramoser, H.,Schlogl,T.,Beleznai,C.,Winter,M.,Bischof,H.:Shape-based detection of humans for video surveillance. , Proceedings of IEEE Int. Conf. on Image Processing, Pages 1013–1016,2003

[7] Palidwar,J. Optical Filters open up new uses for MWIR, LWIR Systems, photonics.com, last accessed 13/05/19

[8] www.littelfuse.com/technical-resources/technical-centers/temperature-sensors/thermistor-info/what-is-a-thermistor, last accessed 13/05/19

[9] Babin,S. Water Vapor Myths: A brief tutorial, www.atmos.umd.edu, last accessed 13/05/19

[10] www.ncl.ac.uk/computing/about/usb/, last accessed 13/05/19

[11] api.usb.urbanobservatory.ac.uk/, last accessed 13/05/19

[12] 2.python-requests.org//en/master/, last accessed 13/05/19

[13] docs.python.org/2/library/datetime, last accessed 14/05/19

[14] scikit-learn.org/stable/index., last accessed 14/05/19

[15] scipy.org, last accessed 14/05/19

[16] numpy.org, last accessed 14/05/19

[17] matplotlib.org, last accessed 14/05/19

[18] 3d.usb.urbanobservatory.ac.uk, last accessed 16/05/19

[19] Li, N., Carlis, G., Becerik-Gerber, B.: Measuring and monitoring occupancy with a RFID based system for demand-driven HVAC operations. Automation in Construction, Pages 89-99,2012

[20] Lam, K.P., Höynck, M., Dong, B., Andrews, B., Chiou, Y., Zhang,R., Benitez,D., Choi, J.: Occupancy Detection through an Extensive Environmental Sensor Network in an Open Plan Office Building, Proceedings of the 11th International IBPSA Conference, Pages 1452-1459, July 2009

Appendix A: Full experiment results

1. CO2

	G.069	1.017	2.015	3.032	4.036
6/5	56.4	50.7	-45.9	-72.8	8.9
7/5	28.9	31.2	39.2	21	24.3
8/5	41.5	-49.6	51.8	-5.7	51.7
9/5	-6	-44.7	18.9	39.9	53.3
10/5	28.6	56.6	60.7	51.6	47.8
11/5	21.6	-8.9	15.5	40.7	-15.4
12/5	3.5		25.3	-21.1	3.9
AVG	24.93	6.05	23.93	8.09	25.51

Occupancy data unavailable for room 1.017 on 12/5

2. Temperature

	G.069	3.032	4.036
6/5	69.4	-67.9	69.4
7/5	59.7	-25.9	59.7
8/5	61	-53.1	61
9/5	40.5	-57.1	40.5
10/5	45.4	-54.3	45.4
11/5	13.7	-42.6	13.7
12/5	1	48.1	1
AVG	41.53	-36.11	41.53

Temperature data unavailable for rooms 1.017 and 2.015

3. Humidity

	G.069	1.017	2.015	3.032	4.036
6/5	21.6		72	42.1	41.3
7/5	13.5	-5.8	-28.8	16.4	19.2
8/5	-12.9	-14.7	61.5	27.3	6.9
9/5	6.7	-19.5	5.4	0.5	-11.8
10/5	-23.8	22	9.9	5.6	31.2
11/5	11.5	14.5	-37.5	-26.4	-26.5
12/5	-3		-34.9	10.2	-7.6
AVG	1.94	-0.7	6.8	7.9	7.53

Occupancy data unavailable for room 1.017 on 12/5

Humidity data unavailable for room 1.017 on 6/5