MONITORING THERMAL COMFORT IN THE BUILT ENVIRONMENT USING A WIRED SENSOR NETWORK

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

2016

By
Luke Anthony Pitt
School of Electrical and Electronic Engineering

Contents

List of Tables

List of Figures

Abstract

This thesis documents a sensor networking project with an interest in internal environment monitoring in relation to thermal comfort. As part of this project sensor nodes were designed, built and deployed. Data was collected from the nodes via a wired Ethernet network and was stored in a database. The network remains operational several years after its initial deployment. The collected data was analyzed in conjunction with data from a local meteorological station and the building's smart fiscal energy meters. The analysis suggests the possibility of automated thermal comfort classification using data from a sensor network.

University Name: The University of Manchester
Author Name: Luke Anthony Pitt
Degree Title: Doctor of Philosophy
Thesis Title: Monitoring Thermal Comfort in the Built Environment Using a Wired Sensor Network
Date: 29th April 2016

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

I would like to thank my supervisors Peter R Green and Barry Lennox for the help, support and feedback they have provided when undertaking this project and writing this thesis. Other staff at the University of Manchester's School of Electrical and Electronic Engineering particularly deserving of thanks are those of the mechanical workshop who were of great assistance in producing the cases for the Ethernet nodes; and Mr Derrick Bradshaw of the PCB production facility for producing all the blank prototype PCBs, and allowing me to experiment with applying the PCB production process to new uses. Thanks too to those who participated in my trial by interacting with the nodes and providing their comfort votes, without which much of my data analysis would have been impossible. I would also like to thank the Engineering and Physical Sciences Research Council (EPSRC) for funding my project. Finally I'd like to thank my parents for their support, particularly towards the end of writing my thesis.

Chapter 1

Introduction

1.1 What this Project is About

This project investigates the use of a sensor network for monitoring the internal environment of a building, in relation to maintaining occupant thermal comfort while reducing energy use. As part of this project a sensor node design was created, 20 nodes were fabricated, and 17 were deployed. Chapter 4 describes the hardware used as part of this project. These sensor nodes transmit their measurements to a central server, which stores the measurements in a database. Chapter 5 describes the software which is running on the sensor nodes and the server, and the communication protocol used for communication between the nodes and server.

The sensor network has been operating since September 2012. Some analysis has been performed on the data collected over a period greater than a year. This analysis and its results are presented in Chapter 7 and the mathematical methods used in that chapter are described in Chapter 6.

Chapter 2 is a literature review which provides background information related to this project. A more in depth overview of the project is given in Chapter 3.

1.2 Motivation for Project

Climate change is a very real problem which requires action to counter [1]. Methods of countering climate change are referred to as 'mitigation' methods [2]. In 2010 buildings accounted for nearly one third of all global energy use. Space heating accounted for approximately one third of the energy used by buildings [3]. The primary motivation for this research is that of reducing energy use. Building monitoring systems installed in naturally ventilated passively cooled buildings have previously been used to achieve significant energy use reductions with improved ventilation scheduling [4].

1.3 Project Aims and Objectives

The project aims to investigate the use of sensor networks to monitor thermal comfort, and whether they can be used to reduce energy usage. A related aim is to investigate whether a sensor network can be used to measure variables sufficient for predicting occupant thermal comfort; and whether such a predictive model coupled with real-time sensor data can be used to control heating systems to maintain occupant comfort while reducing energy usage. To work towards achieving these aims the project worked on the following objectives:

- Design a sensor node which
  - is low cost
  - produces data that informs design of future sensor nodes — relatively high sampling rate, large number of sensors
  - is reliable
  - requires zero maintenance
  - operates indefinitely
- Produce multiple copies of the sensor node and deploy them to form a sensor network
- Operate the sensor network for a period greater than one year
- Produce a dataset for use in future research
- Conduct analysis on the gathered dataset

1.4 Contributions to Subject

This project has made a number of contributions, the most important of which is the dataset collected from the sensor nodes. This dataset is unlike datasets collected by other sensor network research projects because it has:

- a higher temporal sampling frequency (especially compared to wireless implementations)
- a larger number of nodes at a higher spatial frequency

- more parameters measured on each node than some networks with similar objectives
- operated for a longer duration than other comparable networks
- occupant comfort votes associated with the data which many projects do not have

This isn't to say that there aren't projects which are superior in some of these aspects, but there aren't projects which are superior in all of these combined. That is to say that while other projects may exceed this in some aspects, this dataset is above average in all aspects, and is intended to be adequate for most analysis one would wish to perform on data from a sensor network.

A benefit of the above average number of sensors on the nodes is that future node designers can test their data analysis algorithms on this data set. This allows them to determine which parameters are actually valuable for their node to measure at design time, rather than making educated guesses at what they believe will be relevant.

The analysis performed on the data suggests the possibility of being able to measure and automatically classify comfort using a sensor network using parameters that are easier to measure than those required by the industry standard Predicted Mean Vote comfort equation.

The sensor network itself serves as an example for sensor network researchers at a time where most research is focused on wireless sensor networks. It demonstrates the viability, reliability and longevity of a wired sensor network. Higher installation costs of wired networks is often cited as a reason for choosing wireless nodes. However by utilising the building's existing Ethernet network, this project demonstrated that the installation cost of wired sensor nodes isn't necessarily much higher than wireless versions, and is partially offset by the lack of need for radio hardware or certification tests.

Chapter 2

Literature Review

2.1 Introduction

Climate change is a major problem facing mankind [1]. As part of the effort to combat it the UK is committed to reducing its greenhouse gas emissions by 20% from emission levels in 1990 by 2020, and by 80% by 2050 [5, 6]. In 2013 30.65% of all energy usage in the UK was attributed to heating systems; excluding transport uses, that figure rises to 47.84% of the energy used [7]. With such a large proportion of the UK's energy used for heating it is essential to research ways to reduce heating consumption in order to hit the emissions targets.

This chapter reviews literature related to thermal comfort and sensor networks for monitoring the environment within buildings in relation to occupant thermal comfort.

2.2 Thermal Comfort

Thermal comfort is defined as:

> "That condition of mind which expresses satisfaction with the thermal environment and is assessed by subjective evaluation."
>
> [8, ASHRAE Standard 55]

Being a qualitative and subjective definition makes it difficult to optimize for. Surveys of occupant thermal comfort typically use the 7 point ASHRAE thermal sensation scale [8, 9]. This scale ranges from -3 (cold) to $+3$ (hot) with 0 being neutral. Votes with a magnitude of 2 or more indicate that the occupant is dissatisfied with the environment.

In thermal comfort literature there are two main schools of thought, the static and the adaptive. The Predicted Mean Vote (PMV) model is at the core of the static school of thought. It is used to predict the average comfort vote of a large group of occupants based on a number of measured parameters. The PMV and the standards that are based upon it are referred to as the static model, they are discussed in Section 2.2.1. Competing with the static model is the newer school of thought referred to as the 'adaptive' model [10]. The adaptive approach is discussed in Section 2.2.2.

It is important to maintain occupants thermal comfort as thermal discomfort results in lower productivity and by definition, occupant discomfort [11]. Being too hot has been shown to reduce productivity in office environments by approximatly 2% per °C. Being too cold has a direct physiological affect on productivity of manual labourers as the body directs blood away from the hands, reducing control [11].

For large buildings such as most commercial and public buildings, modern Heating Ventilation and Air Conditioning (HVAC) systems are built to the ISO 7730 standard in Europe and the ASHRAE 55 standard in the USA [8, 12, 13]. These standards specify limits on cyclic variations and 'drifts and ramps', and provide methods of calculating the "optimum" internal temperature for a building at a given time of year. Drifts and ramps are changes from one temperature to another over time. The standards place limits on the maximum temperature change in a given time period. A drift is different from a ramp in that a temperature drift is something that occurs naturally where as a ramp is driven artificially by the HVAC system. Cyclic variations are temperature changes in which the temperature variation isn't monotonic. What differentiates a cyclic variation from a series of drifts and ramps in alternating directions is that cyclic variations happen in a shorter period of time. The ASHRAE standard limits cyclic variations to a peak-to-peak amplitude of 1.1°C in a period of 15 minutes. HVAC systems are classified by the standards based on how closely they maintain the temperature to a given set point [9].

The standards, while primarily based on the static model, have started to incorporate ideas from the adaptive school of thought in limited ways [9]. The ASHRAE 55 standard specifies the 'Adaptive Comfort Standard' (ACS). The ACS applies only to buildings with no mechanical cooling system. It specifies a target temperature based on the monthly mean outdoor temperature and the amount of deviation from the temperature that's acceptable. ISO 7730 does not have any adaptive specifications, but EN 15251 specifies an adaptive temperature model for free running buildings, similar to the adaptive model in ASHRAE 55 [14]. The EN 15251 model has advantages over the ASHRAE 55 model as it uses an exponentially weighted running mean of recent outdoor air temperatures, rather than historical monthly mean temperatures [9].

2.2.1 The Static Model

The dominant quantitative measurement pertaining to thermal comfort is the Predicted Mean Vote developed by P.O. Fanger [15]. The PMV uses an equation to convert a number of measured parameters into a number on to the ASHRAE 7 point scale [8]. The number represents the average expected result of a survey of a large population of healthy adults exposed to the measured conditions voting on the 7 point ASHRAE scale.

The PMV model is based on the physics of heat transfer from the body to an unchanging environment. The PMV models is referred to as the static model as it assumes a steady state environment, in which the occupant is a passive receiver. The relationship between the input parameters for the PMV equation and occupant comfort was established experimentally using many trials with subjects in climate chambers. The environmental parameters of the chambers were varied individually in a controlled manner and the occupant's comfort votes were recorded. The most important factors affecting thermal comfort are [15]:

- activity level (how much heat the body is producing its self)
- how insulating the clothing being worn is
- air temperature
- mean radiant temperature
- air velocity
- ambient water vapour pressure (humidity)

Related to the PMV is the Predicted Percentage Dissatisfied (PPD). This is a number that can be calculated which predicts the average percentage of people who will be dissatisfied with the thermal environment for a given PMV. The goal of traditional heating systems is to minimize the PPD. Heating systems are rated by how well they do this. Some parameters of the PMV equation such as those relating to occupant activity level, occupant body surface area and clothing level are difficult to measure in practise. Heating engineers estimate these parameters based on expected room usage when installing heating systems on a commercial scale. This can lead to incorrect heating, especially if what the room is used for is changed from when the heating system was setup.

The model has been found to be free from substantive biases overall, but some biases have been found in the input parameters [16]. The PMV model was originally created with the intention that it only be used for artificial environments regulated by a HVAC system [17]. In buildings with both heating and cooling the model works well. Indeed occupants come to expect a very well regulated environment in such buildings and become dissatisfied with the environment more quickly than in naturally ventilated buildings [18], unlike free-running buildings where occupants are more accepting of wider variation of temperatures [19]. Comfort temperature varies little in a HVAC building, where as in a free-running building comfort temperature is linearly related to the outdoor temperature [19] and is affected by the average outdoor temperature of the previous few days [20].

The PMV has been applied to all people in any type of building, but doing this has been questioned. One flaw in particular is that the majority of climate chamber studies were performed with university aged subjects in North America and northern Europe [17]. Additionally it has been argued by environmental psychologists and human ecologists that such laboratory studies simplify the relationship between occupant and environment too much. Such methods ignore completely the differences between the mental representations of the different occupants in the same environment as well as their backgrounds and prior experiences [18]. The 'one temperature suits all' idea of the static model has also been criticised. The idea was supported by climate chamber studies with a small group of travellers — who may have become acclimatized to the local conditions where the study was taking place. Studies carried out in a variety of locations around the world have revealed a range of preferred temperatures from 17 to 30°C [18].

2.2.2 The Adaptive Model

In contrast to the static model, which views occupants as passive recipients of the environment, the adaptive approach recognises that the occupant can play an active part in maintaining their thermal comfort via multiple feedback loops [21]. The fundamental assumption the adaptive approach is based on is known as the adaptive principle:

"If change occurs such as to produce discomfort, people react in ways which tend to restore their comfort." [19]

Since occupants will make changes to achieve thermal comfort, the adaptive principal suggests that the ideal environment isn't one with a specific temperature (as per the static model), but one in which the occupants can most readily adapt to and/or alter in order to achieve thermal comfort [22]. Compared to the static model, the adaptive approach suggests the possibility of reducing energy usage while maintaining occupant comfort. This is because the static model specifies a narrow band of acceptable temperatures which requires a large amount of energy to artificially force the environment to conform to; where as the adaptive model permits a wider range of temperatures provided the occupants have sufficient means to adapt the environment or adapt to the environment. This is especially relevant for people's homes, where people have the most control over their environment. This is important as housing consumed the most energy by sector in 2012 [23].

There are three main ways that occupants can make changes to achieve thermal comfort: behavioural changes; physiological changes; and psychological changes [10].

Behavioural changes refer to some physical activity that an occupant might undertake to adjust their thermal comfort. Examples would include changing clothing level; drinking a hot or cold beverage; moving into or out of an area of sun, possibly by manipulating blinds; opening or closing a window; changing the settings on any HVAC equipment they can control; scheduling activities to be at a time of day with a more appropriate thermal environment.

Physiological changes refer to changes an occupants body makes to adapt to become comfortable with a thermal environment. The most common form of this is acclimatization, where discomfort with a new environment diminishes over a period of days or weeks. This may be experienced by travelling to a region which is much hotter or colder than one's normal environment and staying for a sufficient length of time. One will find that initially the temperature seems too hot or too cold, but over time the discomfort diminishes until the temperature of the environment becomes 'normal' and expected. Changes typically take a few days to a couple of weeks to manifest. Examples of changes to a hot climate include a lower pulse rate, maintenance of a lower body temperature and sweating more readily [24]. Physiological changes also include genetic adaption, wherein a group of people living in an environment develop ways of coping better with that environment. An example of this is the higher basal metabolic rate in indigenous Siberian populations [25].

Psychological changes relate to changing ones expectations for the environment. It includes cultural and cognitive factors which affect ones perception and reaction to the environment [21]. An example is the mind interpreting a stimulus as being less intense after repeated or long term exposure to it. There is no mechanism in the

PMV to handle this type of adaption. Thermal comfort that one perceives is ultimately an psychological condition, in that it only exists as an electro-chemical state in the brain. As such there isn't a fixed temperature or temperature region that one finds comfortable. Occupants can change their expectations for the environment and thus become comfortable.

2.3 Improving HVAC Systems

This section describes some of the problems with existing Heating Ventilation and Air Conditioning (HVAC) systems, and possible improvements. Topics discussed are the poor utilization of existing HVAC controls, improved control strategies, the use of occupancy monitoring to improve efficiency and reduce energy usage, and the recent advent of the smart thermostat.

2.3.1 Poor Utilization of HVAC Control

Home heating systems often operate on a timer in combination with a thermostat. Offices and other places of work can have similar systems installed. Larger buildings often have a system designed and configured by Building Service Designers, Mechanical Engineers or Building Service Engineers, which may or may not be integrated with a Building Management System (BMS). Occupants typically don't have access to controls for such systems.

When occupants have access to heating controls in their place of work, those controls typically are under utilized for a number of reasons. The occupants may not know where the controls are, how they work, or if they have permission to operate them. Additionally as there is a large time delay between changing a setting on the control panel and a noticeable change in the environment, it often isn't apparent to the occupant using the control panel whether their interaction has had any affect. This can lead them to think that the panel doesn't work or the system isn't switched on. Additionally it often isn't apparent based on an examination of the control panel what affect a change of settings will have [26]. Occupants often have a better understanding of the heating systems and temperature controls in their own homes than they of their work places [27]. Individual occupants may have conflicting thermal preferences, such conflicting preferences can prevent occupants from adjusting controls to satisfy their own preferences.

2.3.2 Improved Control Strategies

It has been suggested that occupant comfort could be better maintained if HVAC control systems regulated a value analogous to thermal comfort instead of regulating temperature. Research has been done examining using PMV as the controlled variable, and fitting the sensors required to calculate the PMV in the room being regulated by the HVAC system [28].

One problem with this is that the PMV is computationally expensive to calculate as it requires iteratively solving a nonlinear equation using numerical methods. Doing so can take too long for embedded processors as part of a real-time system. This is particularly the case if the initial estimate values are far away from the true values as this greatly increases the number of iterations required before convergence is achieved [28]. The standard method for dealing with this is to use look-up tables of precalculated results [13, 15]. One approach to solving this problem used a Neural Network to calculate PMV approximations in a constant amount of time [28]. This provided a good approximation of the PMV for the values the network was trained on.

A second problem is that several of the parameters for calculating the PMV are difficult to measure automatically and in real time. In particular the problem of automatically determining occupant clothing insulation and activity level remains unsolved. PMV based control systems typically require user input for these values or assume a constant value which is set when installing the system. This is a problem as these two variables are two of the most significant methods occupants have of active thermal self regulation.

The Adaptive Control Algorithm (ACA) was created as part of a large European project to turn adaptive thermal comfort theory into a HVAC control specification [20]. As part of the project thermal comfort surveys were carried out in five countries. The project proposed some equations to calculate preferred temperatures, and the constants for these equations were found by analysing the results of the thermal comfort surveys. The most significant difference between the ACA and traditional setpoint temperature is that the ACA determines the target temperature based on an exponentially weighted running mean of the recent history of the outside temperature. The ACA was deployed to two buildings, one in the UK and one in Sweden. Both buildings had the same model of HVAC control system and the ACA software was deployed as a new controller for the existing system.

Thermal comfort surveys were also carried out while the ACA was deployed. The results showed no significant changes in comfort level and no increase of discomfort.

The ACA resulted in a higher internal temperature as compared to the setpoint temperature that the building's control system was normally set to. As the ACA was deployed during a time of year when air conditioning is in use the higher target temperature resulted in lower energy use.

2.3.3 Occupancy Monitoring

A method for reducing heating system energy usage is to attempt to eliminate unnecessary heating by using occupancy monitoring. Energy usage is reduced by not heating rooms which aren't occupied. This method of energy use reduction has been studied for lighting systems and results in between 17–60% energy savings [29]. Occupancy monitoring has been implemented commercially for heating systems, with the company claims reduces energy usage by 35–40% [30]. Passive Infrared (PIR) sensors are used in both this system and the study on lighting. PIR sensors are commonly used in burglar alarm systems and outdoor security lighting. They are relatively cheap, simple to use and commonly available. Other occupancy technologies are available such as ultrasonic occupancy detectors, some of which can detect smaller movements than PIR sensors. More elaborate occupancy detection methods have been investigated such as using networked computers which are already present in a room [31].

2.3.4 Smart Thermostats

Learning thermostats are the latest commercial development for home thermal control. The canonical example is the Nest Learning Thermostat. It keeps a record of when occupants change the target temperature and to what temperature it was set. In future it automatically makes the same change without user interaction. The Nest unit recognises similar patterns happening on different days and then when a schedule change is made on one day, it makes the same change on the other days it has recognised as being similar to that day [32]. The Nest Thermostat has reported energy savings between 10 and 25% [33].

2.4 Similar Systems

This section describes several sensor networks, the majority of which are the products of university research projects. This section has been divides into two parts. The first describes systems which are designed for building monitoring. These systems collect data for it to be interpreted by a building manager or engineer. The second part describes systems designed with thermal comfort in mind. These systems may survey occupants about their thermal environment, may interface with the building's HVAC system to effect changes in the thermal environment, or they may do both. These systems use sensor networks to better monitor or regulate the thermal environment.

2.4.1 Sensor Networks for Building Monitoring

This section describes three sensor networks intended for building monitoring. These systems are all wireless sensor networks which record data about the indoor environment and other aspects of building performance. The data from these networks is not used by any control systems or other automated systems, it is simply recorded for future examination by a human. The first two systems are located in office building used by researchers, the third is based in a residence.

The first system described here is a Wireless Sensor Network (WSN) deployed in the Environmental Research Institute building at the University of Cork in Ireland [34]. It was produced to research the use of wireless building performance monitoring (BPM). The belief being investigated was whether a WSN could provide more information about internal conditions and thus enable building managers to make better decisions. The nodes are modular, consisting of a series of square boards stacked on top of each other. The nodes are small, with the square boards having a side length of 25mm. The modularity of the nodes allows the use case of each node to be modified. This allows some nodes to control actuators in addition to or instead of mounting sensors. A total of 60 nodes were deployed. Nodes communicate with IPv6 over low power WPAN (6LoWPAN). The nodes send measurement data to a server via a bridge node which links the WSN to a computer. The data sent to a server is then presented in a GUI.

The nodes detect occupancy, the opening and closing of doors and windows, the flow and temperature of water in water pipes, humidity, temperature and light levels. The nodes use PIR sensors to detect occupancy. The opening and closing of doors and windows is detected by the accelerometer sensor of nodes mounted on those features. To save power the node remains in a low power sleep state most of the time, and is woken from a low power state by a vibration sensor when the door or window is moved. When woken in this way it uses the accelerometer to determine whether the door or window is being opened or closed (direction of movement), and by how much (angle of change). A surface mount temperature sensor and a ultrasonic flow meter allow a node to measure temperature and flow of water in water pipes. The sensor module board also features a SHT11 sensor to measure temperature and humidity, and a light level sensor.

Compared to other projects, this project features a large number of sensor nodes, deployed with a high spatial density with an average number of sensors. The paper does not specify how long the network operated for, the battery life of a node or the rate the sensors weer sampled at. Examination of the graphs in the paper suggest a sample rate of 1 reading every 6 minutes, with window or door sensors sending additional updates whenever they detect movement, and the occupancy detectors updating once every half hour or whenever they detect occupancy. The data collected by the nodes is stored, but only used to plot graphs.

The second system described in this section is a network of 7 wireless sensor nodes deployed in the Building and Fire Research Laboratory at the National Institute of Standards and Technology in Maryland USA [35]. The nodes used were commercially available off-the-shelf products which cost $100 in 2007. The nodes consisted of two stacked PCBs attached to a battery holder which holds two AA batteries. The nodes do not have any form of case. One PCB had a microcontroller and radio onboard, the other was the sensor board. The sensor board featured temperature, light, acceleration and magnetic sensors. The batteries of the node would last 61 days with average use, with a maximum lifetime of a year if the node remained in low power mode throughout. The nodes formed a mesh network using the ZigBee protocol. One of the nodes acted as a gateway and relayed the data from the other nodes via serial link to a PC running a program written in Java. The Java program received the data and stored it in a MySQL database. A website was produced to display the data in the database.

In comparison to other projects, this project used a relatively small number of nodes with very few sensors. The nodes sampled their sensors more frequently than other projects with samples being taken once every 8 seconds. This higher sample rate will have contributed to the lower lifespan of the nodes' batteries. The use of ZigBee and a gateway node connecting the network to a computer is typical of WSN projects. The use of a database is superior for long term operation compared to projects which simply store the received data in a text file. The data collected was only used to populate tables of measurements on a website, as the project was only intended to demonstrate the steps needed to implement a WSN.

The third project discussed is a wireless sensor network installed in a three-story townhouse [36]. The aim of the project was to discover the challenges and obstacles that prevent building operators and researchers from deploying WSNs for building monitoring. The project used commercially available mesh network development kit boards for the sensor nodes. These boards were mains powered. Appropriate sensors were connected to the boards' input connections to turn them into sensor nodes. One board was connected to a computer via serial link to record the data from the nodes in a text file. The development kit came with software for establishing a mesh network, but the application layer to take the sensor readings and send them through the network was custom written, as was the software on the computer to record the data.

The network was deployed in several stages. Initially one node was used to monitor an air handling unit, with sensors monitoring air temperature, humidity and fan energy usage. Another node was used to monitor a heat pump. Separate nodes were used to monitor the indoor and outdoor temperature and humidity. After three months a node was added to monitor a water heater. This node had temperature sensors on the inlet and outlet of the water heater, a flow rate sensor, a temperature sensor inside the heater's insulation jacket, an ambient temperature sensor outside the insulation jacket, and an electrical power meter on the heater's power input. Six months into the project four new nodes were added to the network. These nodes monitored temperature and humidity inside an exterior facing wall. These nodes were spaced to determine if there was a moisture gradient inside the wall.

The project's conclusion was that while the installation of the nodes and setup of the mesh network was simple, adding sensors to the nodes and writing low level code to configure the nodes are tasks outside the comfort zone of many heating and building engineers. The authors suggested that wider use of sensor networks would require commercially available sensor nodes with pre-installed sensors appropriate for building monitoring (and provided a list of suggested parameters to sense); and that the engineers installing them should only need worry about high level configuration for their particular project. They also noted the unreliability of the wireless network. There were problems with both the availability of nodes decreasing over time, and errors occurring in the transmitted messages resulting in corrupted data. They suggested that work should be done to produce a common industry standard for sensor nodes to communicate data, and that consideration should be given for securing the communications of the nodes. It was noted that the wireless nature of the nodes was advantageous as it reduced the installation time and meant that data wires didn't need to be installed. This limited disruption for the occupant of the house. It was suggested that the use of battery powered nodes would further improve the situation as it would remove the need for power lines, which may prove inconvenient if there are many nodes installed.

2.4.2 Sensor Networks for Thermal Comfort and Control

This section describes four sensor network studies which may monitor thermal comfort or which may control the HVAC systems of a building, or both. A patent describing a system which performs these functions is also discussed. The first project describes two wireless systems deployed in different operating conditions, and compares them to the traditional wired alternative. The systems control and monitor a HVAC system. The second project discussed uses a sensor network to monitor conditions, survey software to record occupant comfort preferences, and an algorithm to control the building's HVAC system. The third project uses a sensor network to monitor and control more environmental parameters than a traditional HVAC system. It regulates temperature, light level and CO_2 level. The fourth project investigates using a sensor network to measure PMV in real time. The final item in this section is a patent which describes a an algorithm for thermal regulation using a sensor network which monitors the environment and provides a physical interface for occupants to express their thermal preferences.

The first study discussed here was conducted by the U.S. Department of Energy Office of Building Technology. It featured two wireless sensor networks deployed at the Pacific Northwest National Laboratory [37]. The purpose of this study was to investigate the costs of wireless sensor networks as compared to traditional wired sensor networks in two different usage scenarios.

The first system in the study had 30 nodes deployed in a heavy steel and concrete building. There were 10 nodes deployed per floor with an additional repeater on each floor. The repeater forwarded the signals to a gateway node connected to a building management network. The project considered 6 different wireless technologies and found one to be marginally cheaper than the estimated wired installation costs initially. After two cycles of battery replacements the wireless system becomes more expensive overall. The second battery replacement would take place 10 years after deployment. The batteries were made to last an estimated 5 years by having the nodes sample their sensors very infrequently (once every 10 minutes). This sample rate was much lower than that offered by the traditional wired system. This made the wireless sensors unsuitable for real-time air handler closed loop control, but the update rate was considered suitable for zone control (for which it was typical for the temperature to change over 30 minute periods). The wireless cost savings would have been greater had the repeaters not been required due to the building's structure, but conversely had more repeaters been required the wireless costs would have exceeded those of a wired system.

The second system in the study had nodes connected to three rooftop HVAC units. The nodes communicated their readings to a central receiver unit connected to a computer, which would then forward them to a LAN, WAN or the Internet. Unlike the first system these nodes could be powered from the mains power that was already connected to the rooftop HVAC units, eliminating the need for batteries. Not using batteries allowed higher sample rates and eliminated ongoing battery replacement costs. These nodes also had clear lines of sight between their antennas, meaning that repeaters were not required as there was nothing to attenuate the radio signals. The lack of repeaters further lowered the cost of the rooftop wireless system as compared to the inside wireless system. These factors made the rooftop wireless system cheaper than a traditional wired system, but only if the HVAC units were over a certain distance from the receiver unit. Over short distances the cost of the wireless transmitters exceeded the cost of using wires to connect the units.

The study was conducted in 2002 and the costs of wireless hardware have fallen since. The study does show however that the suitability of a wireless system depends on the situation. Sensor nodes can benefit from not being totally wireless by utilising wired mains power if it is conveniently available. Similarly if only short runs of cable are required to connect a node to a wired network then the node can benefit from the more reliable and higher data rate communication channel provided by the wired network, while still being cost competitive as compared to a wireless system.

The second project is described as a Human-Building Interaction for Thermal Comfort (HBI-TC) framework [38]. It features a per occupant thermal comfort profile which is informed by comfort votes from an survey application on an individual's smartphone or tablet, or alternatively a web application on a computer. There is a BMS controller module which is linked to the building's BMS and HVAC system which provides control of the HVAC temperature set point. The BMS controller receives data from a wireless sensor network and uses the occupant thermal profiles and recent comfort votes to determine the HVAC set point.

The first stage of the project was producing the thermal comfort profiler. The profiler produced thermal models for each individual occupant using the system. The profiler was fed thermal preference votes by each user from a survey application. The survey application allowed occupants to vote on a slider between the arbitrary values of -50 to $+50$ to rate their thermal preference relative to the current conditions. Positive values indicated a desire for a warmer environment and negative values indicated a preference for a cooler environment. The magnitude of the values is analogous to the strength of preference or level of dissatisfaction with the current environment.

To measure the ambient conditions at time of comfort vote, and when the system is being used to regulate conditions, a wireless sensor network was used. The networked nodes have 8 sensors in total. The environmental parameters of temperature, relative humidity, CO_2 level, light level and sound are measured. Occupancy is detected with a PIR sensor. There is door switch sensor to detect whether a door is open or closed. There is also a motion sensor but it is not specified what motion it detects. The nodes send the data once a minute to a database using a WiFi network. The nodes are based on the Arduino Black Widow board and housed in a box.

The thermal comfort profiler model was tested over a period of 3 weeks. The test was conducted in the autumn season in an office building in California. The building's HVAC set point was varied between 18–29°C over the three week period. Four occupants rated their comfort using the smartphone app. The feedback was used to determine the changes to the temperature set point to encourage participation in the test. This data was used to train a model of five fuzzy sets representing the conditions very warm, warm, neutral, cold, and very cold. Sensitivity analysis showed that between 40 to 50 pieces of evenly distributed feedback were required per occupant to produce an accurate comfort model for said occupant.

As part of the personalised comfort profiling the complete system operated for two months in four zones of an office building. Initially the system operated in training mode in which it adjusted the temperature of the controlled zones based on immediate feedback from the survey application. Once sufficient survey votes had been collected for profiles to be reliable, the system switched to regulating the temperature set point to minimize a calculated error value based on current sensor node readings and occupant comfort preferences. The system still responded to comfort votes in this second mode.

The performance of the BMS control was assessed by using the system to control two zones. Zone A in the north side of the building had three rooms, and zone B in the south side had two. All the rooms were of similar size and configuration of furniture and windows. Each room was fitted with a wireless sensor node close to the door. Each room also had the pre-existing BMS temperature sensor connected to the HVAC system. The BMS temperature sensor continued to be used by the HVAC system as normal, but the HVAC system's set point was controlled by the thermal model running on a computer which received data from the wireless sensor nodes. The system proved functional and responded to occupant comfort requests appropriately.

The third project discussed is called the Integral Control system of Indoor Environment (ICsIE) [39]. It is an experimental control system installed in a room in an office building in the Faculty of Civil and Geodetic Engineering, University of Ljubljana. It monitors both indoor and outdoor parameters. Internally, humidity, temperature, light level, CO_2 and the energy used for both heating and cooling are measured. Externally, wind speed and direction, temperature, light level, humidity, precipitation type and direct and reflected solar radiation are measured. The internal sensors are distributed around the room. There is one light level sensor beside the windows and one in the middle of the room. They measure the illumination of two distinct workplaces within the room. On the wall opposite the windows are the air temperature, the relative humidity sensor and the CO_2 sensor.

The ICsIE regulates three distinct parts of the environment: the visual environment, the thermal environment, and the olfactory environment. The visual environment is controlled by adjusting the illumination level. This is done by controlling blinds and artificial lighting. The thermal environment is regulated by operating windows, blinds and active heating and cooling systems. The olfactory environment is measured by proxy of the CO_2 level. The system maintains the level of CO_2 below a threshold value by actuating the windows. It uses a Proportional-Integral(PI) controller and a fuzzy logic system running on a Programmable Logic Controller (PLC). Measured data is sent from the PLC to a PC to be recorded. The system prioritised occupant comfort over energy efficiency, and was believed to have performed well as the occupants very rarely overrode the settings the system chose.

The fourth project is a system which attempts to measure PMV in real time using a sensor network. Measuring PMV in real time is an area of interest for researchers. The real time measurement of PMV would allow a control system to regulate the PMV to minimize occupant dissatisfaction with the environment. An example system was deployed in an air-conditioned office building by researchers in Hong Kong [40]. The system used a network of sensors to produce a single PMV sensor. The network used sensor nodes to measure the temperature of the surfaces surrounding the occupant to determine their radiant temperature. The network node which served as the PMV sensor would theoretically measure airspeed, relative humidity, and air temperature at its location. The PMV sensor node in the implementation lacked appropriate sensors and used adjustable preset values. The paper suggested using an application-specific integrated circuit (ASIC) to measure these values [41]. The system also required the the measurement of the occupant's thermal radiation level, and suggested using a thermopile aimed at the occupant. A rotary encoder was used in place of a thermopile to input the occupant thermal radiation level from a list of preset values when testing the system. The system obtained the last two required input parameters for the PMV equation (occupant clothing insulation level and occupant metabolism level) from an application running on the occupant's computer. The occupant is required to select a value appropriate to their current condition from a list of values presented to them, and update their selection as their situation changes. While many of the inputs to the system were simulated and thus prevented the system from actually measuring the PMV in real time; it did demonstrate the use of a lookup table combined with interpolation as a practical method for low powered devices to determine an approximate value (to within 0.5 of a PMV) for the PMV for a limited range of input values.

The final item in this section is a patent assigned to Siemens Energy & Automation Inc. It describes a system in which an optimisation program determines a temperature set point based on data from many sensors which are part of a sensor network [42]. The sensor nodes monitor temperature and humidity, and provide an interface for user interaction. The user interaction is in the form of two buttons. One button is used to indicate that the current environment is too hot, the other that it is too cold. The data from the sensor nodes is stored in a database, and a program generates one or more occupant preference profiles from the data. A controller uses the profiles to generate a temperature set point using an optimisation algorithm.

2.5 Conclusion

In order to reduce the effects of global warming, mankind needs to reduce its energy use. A very large proportion of energy in the western world is used for heating and air-conditioning. Besides increasing people's comfort, HVAC systems improve productivity and in some climates, seasons or sections of the population they are essential for remaining healthy. Since people won't stop using HVAC systems, improving HVAC efficiency is therefore essential.

To improve their efficiency one needs to reduce the amount of energy used to achieve the same level of perceived occupant satisfaction. Rigorous scientific research has been conducted into what conditions people find satisfactory. An important product of this research is the PMV model. While the PMV works well in laboratory conditions, there are some problems with its application in the real world. In particular some important inputs to the equation (e.g. clothing level, metabolic level) are difficult to measure. The traditional solution is to use fixed values based on expected room usage for these parameters when configuring a HVAC system. The PMV model assumes occupants are passive receivers of their environment, which is not true in the real world. In addition to actively changing PMV parameters the HVAC system can't measure such as clothing level, occupants will actively modify their environment (e.g. by operating windows), change their activity schedule based on expected thermal conditions, or change their perceptions and expectations of what conditions they consider to be comfortable conditions.

To address these faults of the PMV model the adaptive model of thermal comfort was proposed by researchers. The adaptive model seeks to provide occupants with an environment that they can most readily adapt to or alter to achieve satisfaction. It considers not just the immediate environment the occupant finds themselves in, but the recent history of outdoor conditions, as these have been shown to affect people's expectations for their environment. Modern standards for HVAC systems are still based on the PMV model and encourage regulating building temperatures to be within a narrow range, which is very energy intensive. Recently some aspects of the adaptive model have been adopted by the standards for some types of buildings.

An active area of research for increasing HVAC system efficiency is building monitoring sensor networks. There are several ways these are expected to improve HVAC efficiency. One way is detecting occupancy patterns and only heating rooms when they are, or are expected to be occupied. Another is using sensor networks to monitor many environmental parameters and using them to predict or measure occupant comfort, and then using the heating system the minimum amount required to achieve occupant comfort. In addition to thermal comfort these sensor networks often measure light level, potentially allowing reduced lighting energy usage and/or improved visual environment satisfaction. CO_2 level is also often measured by these systems as it is a good indicator of indoor air quality. It works as an indicator because humans generate CO_2 at a similar rate to the rate they generate metabolism byproducts and bioeffluents, which are typically the main pollutants of indoor environments [43]. The CO_2 level affects occupant satisfaction with their environment and also their health. It is sensible for sensor networks intended to reduce HVAC energy usage to measure CO_2 level as the ventilation of a building is related to its heating and air conditioning.

So far research into sensor networks for thermal regulation has demonstrated that they can produce environments occupants are satisfied with and have shown the possibility of energy use reduction. Prediction and classification of occupant thermal comfort based on measured environmental parameters hasn't been convincingly shown and requires further research.

Chapter 3

Project Overview

To investigate the objectives described in Chapter 1.3, a sensor network was designed and implemented. This chapter provides an overview of what was built and why — also covered here is why certain methods of data analysis were used on the collected data. Later chapters will provide the technical details describing exactly what was done and how.

The sensor network that was deployed consists of 17 sensor nodes — an overview of which is provided in Section 3.1 with specific details in Chapter 4. These nodes communicate with a central server which stores their readings in a database. This server also receives data from a local meteorological station and the building's smart utility meters. An overview of the software is provided in Section 3.2, more details are provided in Chapter 5. A number of data analysis methods have been used on the data in the database. The motivation for these is explained in Section 3.3 with details of the results given in Chapter 7. The analytical methods used for the data analysis are described in Chapter 6.

There are several aspects which combine to differentiate this project from others investigating sensor networks for internal environmental monitoring. Firstly, this project uses a wired communication channel, rather than wireless which more research focuses on. The wired channel removed the ultra low power requirement placed on wireless sensor networks. This enabled the sensor nodes used in this project to mount more sensors, take measurements more frequently and operate for a long duration without maintenance. Secondly, this project deployed a large number of sensor nodes in a relatively high spatial density. Thirdly, this project combines the data from the sensor nodes with data from a local meteorological station and the building's smart fiscal energy meters.

The reason for collecting weather and smart meter data in addition of the sensor node data is because of the principle of ‘Boosting’ classifiers. This is where a number of weak classifiers are combined into an arbitrarily strong classifier [44]. This works because data which is uncorrelated with a parameter of interest neither improves nor degrades your knowledge about the parameter, but information that is correlated — even weakly — improves your knowledge of a parameter. This means that monotonicity applies to your knowledge of a parameter as you add more information — whether it’s correlated or not. Neither the smart meter nor the weather data can be used alone for classification, but they both contain some information which is related to comfort. So having that information available allows for production of stronger classifiers.

One of the aims of this project is to inform future sensor network research and node design. This led to the design of the nodes and specifications for data collection being deliberately over engineered. Firstly the sensor nodes and weather data are sampled every 30 seconds. This sample period was chosen as it was assumed to be much shorter than the rate that the environment changes significantly. This over sampling enables analysis to be performed with simulated slower sample rates to allow optimum sampling rates for a given task to be determined. Secondly the nodes had a number of sensors measuring duplicate parameters. This allows assessment of the real world performance of the sensors over a long period of time, to inform future node design decisions. The duplicate light sensors also enable investigation of whether it is valuable to sense the directionality of light when attempting to measure occupant comfort level.

Figure 3.1 provides an overview of the system. The arrows indicate the direction of travel of data. It shows the sensor nodes which send their data to the data receiver software on the server. That software also sends the current time and time to take the next sample back to the nodes. The server also receives the data from the meteorological station and fetches the smart meter data. All the data the server receives or collects is stored in a database. The database can be accessed remotely to allow the stored data to be analyzed. There is also software running on the server to monitor the status of the system and perform automated error recovery.

3.1 Sensor Nodes

The sensor nodes are the core piece of work in this project. They are the tools produced which were used to generate the most significant contribution to the subject — the

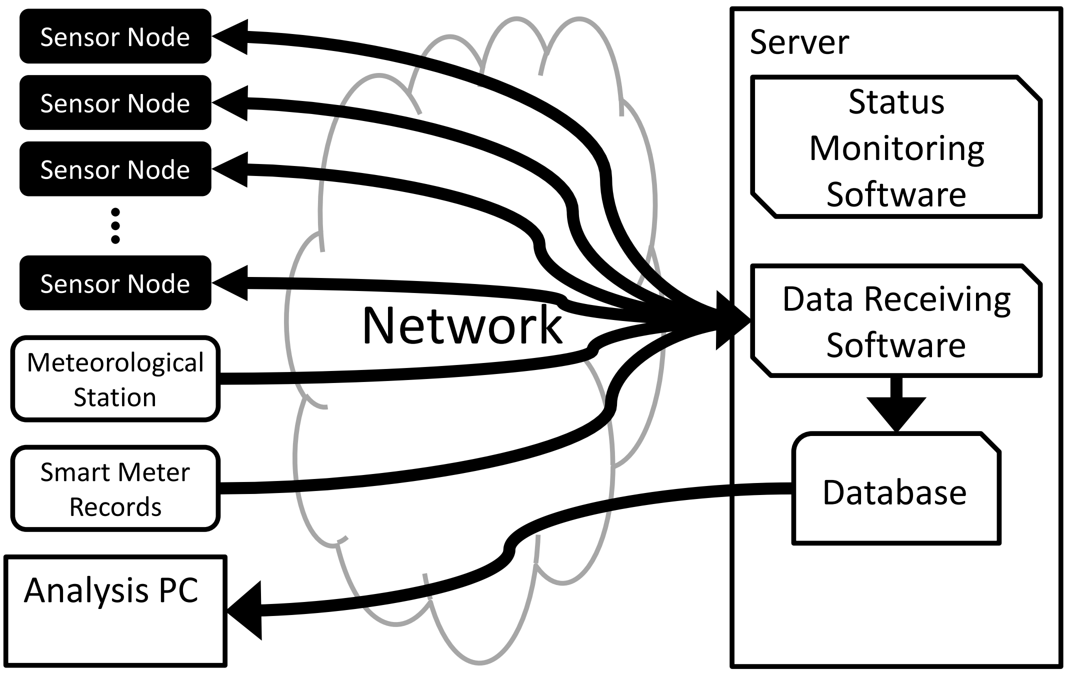


FIGURE 3.1: System overview diagram

novel and unique dataset of sensor node data. The sensor nodes are essentially small boxes which house a variety of sensors. These sensors measure local environmental parameters and transmit the measurements to a server to be stored in a database.

An important feature of the nodes produced is that, in addition to measuring local environmental parameters, they also allow occupants to indicate their current comfort level. The user interface (or UI) was based on the PMV 7 point voting scale, but simplified so that it had just three categories — ‘Too Hot’, ‘Just Fine’ (aka ‘Ok’ or ’Neutral’) and ‘Too Cold’. The change from the 7 point scale to 3 points was done for reasons including simplifying the UI so it can be used without instruction and the impracticality of fitting 7 buttons on the nodes.

3.2 Server Software

The primary job of the server software was that of receiving and storing the data collected by the sensor nodes, meteorological station and building smart fiscal meters. In addition to directly performing that task, great focus was given to making the system reliable and capable of recovering from errors. This included the writing of several pieces of software to monitor, perform error recovery, and provide automated status reports on the software that directly fulfilled the primary job of the server software. These pieces of secondary software ensured that the system would be able to perform its task in the long term, and reduced the chances of silent unrecoverable errors leading to lost data.

3.3 Data Analysis

3.3.1 Purpose

One of the aims of this project is to investigate the use of sensor networks to produce heating control systems that maintain occupant comfort while reducing energy usage. Before this can be achieved the data needs to be analyzed to find out what information can actually be obtained by a sensor node and if it is of any use.

3.3.2 Motivation for Classification

A significant amount of the data analysis focused on seeing if it's possible to create a classifier, which given current sensor readings will output whether an occupant would be comfortable. The reason for this is such a model would then transform the sensor node from something that measures environmental parameters, to something that measures comfort. A comfort sensor which requires no user interaction can then be used to provide feedback for a heating control system which regulates the environment to maintain a level of comfort, rather than a traditional level of temperature or humidity.

The PMV model is based on parameters which aren't practical to measure such as occupant clothing level, wet bulb temperature and airflow. It was of interest whether the sensors on the sensor node provided data which is indicative of thermal comfort. If they did, then it would be possible to be able to build a model that determines in real time what the thermal comfort level of occupants is. The model could then be used to control a heating system to regulate comfort level in a closed loop manner. The model could also be used to determine which of the controllable parameters requires the least energy to change the environment from a given uncomfortable environment to a comfortable one.

Chapter 4

System Hardware

In order to investigate the utility of sensor networks in relation to reducing energy usage while maintaining thermal comfort, a sensor network was designed and deployed. 20 bespoke sensor nodes called Ethernet Nodes were produced, 17 of which were deployed as a sensor network. This chapter describes the physical aspects of their design, development and deployment. A node is shown in Figure 4.1.

4.1 Ethernet Nodes

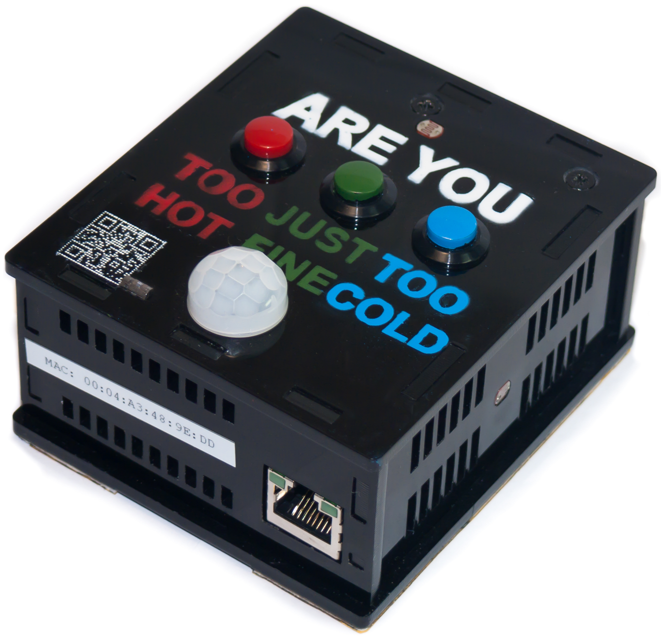


FIGURE 4.1: A fully assembled Ethernet Node

4.1.1 Design Objectives

There were a number of design objectives guiding the design of the nodes. These were:

1. Low Cost
2. Long Lifespan with zero maintenance
3. Provide reference for future work by over-engineering the nodes to gather more data than is necessary — both in terms of sampling frequency and measurement of the same parameter with multiple types of sensor.

The first objective that the nodes be low cost is important since the number of nodes which can be deployed with a given budget is inversely proportional to the cost of a node. Being able to deploy a large number of nodes was important for this project as the main purpose of this project is to investigate the utility of sensor networks, and using a network with only a small number of nodes for the investigation would yield less representative results. The target when designing the node was that each node should cost around £25. This was a target to aim for and guide design decisions rather than something that could actually be achieved with the small scale of production involved in this project.

The second objective was that the nodes should operate for a long time (longer than several years) without requiring any maintenance. This objective effectively ruled out the use of batteries for powering the nodes, as batteries would eventually need to be replaced. It also set a standard for the level of build quality for the nodes — they had to be built to last.

The third objective involved equipping the nodes with multiple sensors for most measured parameters. This was to allow real world comparisons of more expensive sensors and cheaper alternatives to be performed. To a certain extent this conflicts with the first objective as it would be cheaper to have a single sensor per measured environmental parameter; however, so long as the additional sensors were also low cost and their addition didn't reduce the number of nodes that could be produced, then their inclusion doesn't clash with the low cost objective. Furthermore the cost of a node without their inclusion can still be calculated and potentially reduced should the cheaper sensors prove to be adequate replacements.

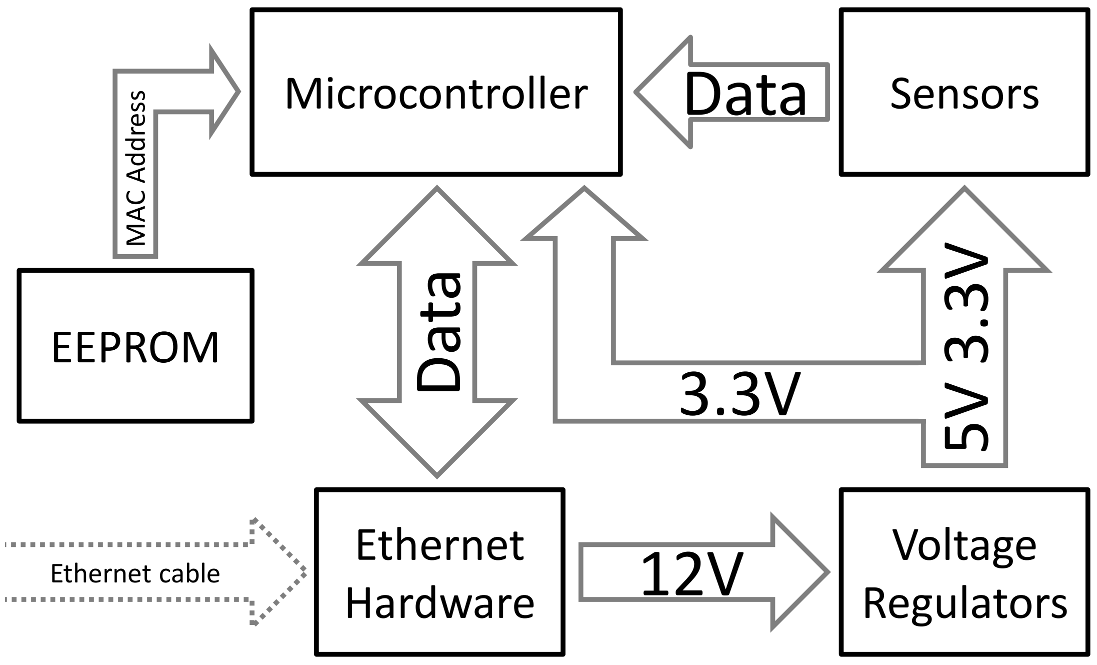


FIGURE 4.2: Block diagram of an Ethernet Node

4.1.2 Overview

Figure 4.2 gives an overview of the Ethernet Node design in block diagram form. The Ethernet cable provides the node with power and also its network connection. The node's Ethernet hardware separates the power from the network connection and links the network connection to the microcontroller. The power from the Ethernet cable is sent to voltage regulators to be converted to the 5V and 3.3V used by the rest of the node's components. The microcontroller receives data from the sensors and transmits it via the Ethernet hardware to a networked server (not shown). The microcontroller gets its unique MAC address from a pre-programmed EEPROM chip. The EEPROM chip also receives power from the voltage regulators, but this is not shown on the diagram.

Figure 4.3 shows the inside of the node. Bottom left is the Ethernet socket. To the right of it are four resistors and two capacitors which are part of the Ethernet communication hardware. To the right of those is the power supply hardware. This includes two voltage regulator components (black boxes, far right); two large electrolytic capacitors; an inductor; a connector for alternative power input; and a switch to select between input power from the Ethernet socket or the alternative power input. On the far right are two hexagonal posts which hold the board down and in place. The ends

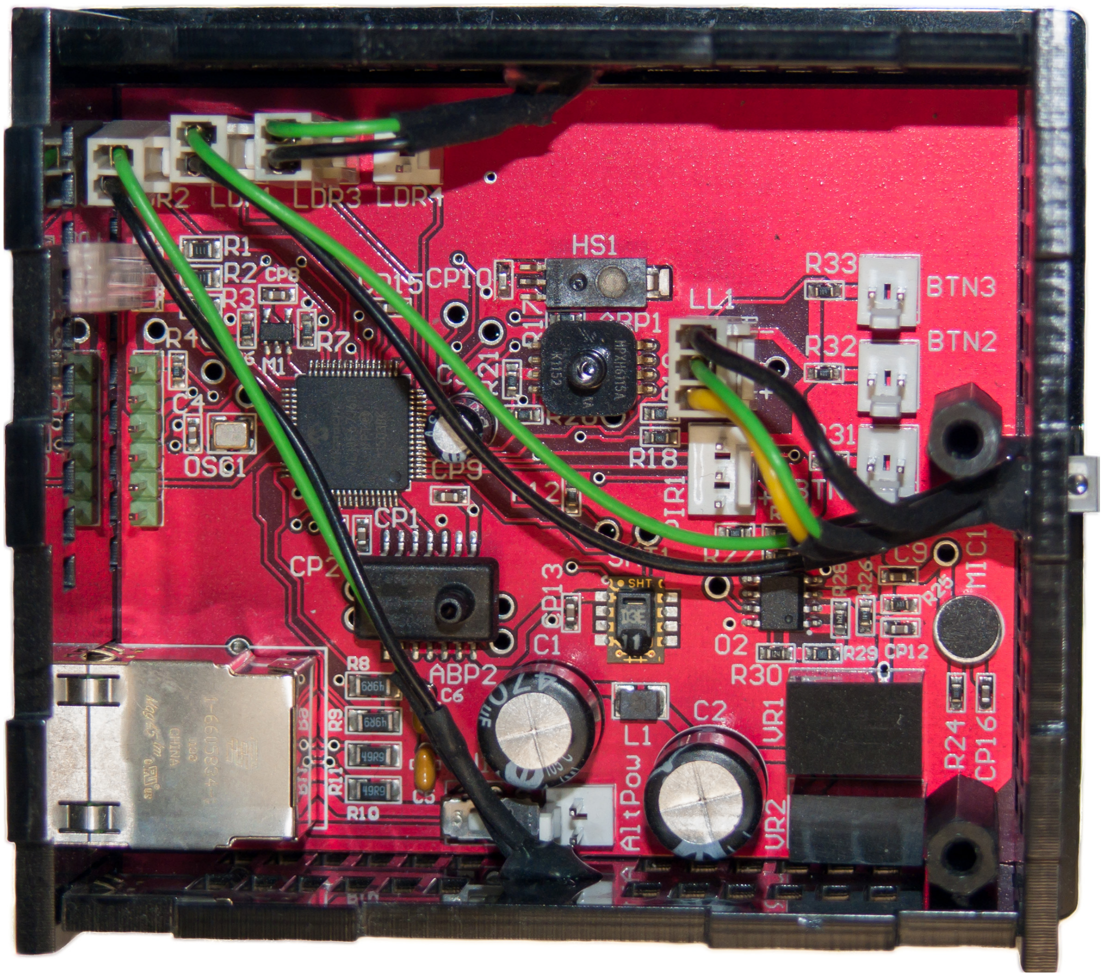


FIGURE 4.3: Photograph of the inside of an Ethernet Node. The front face has been removed and the front face sensors are disconnected.

of the posts are threaded to allow the front face to be bolted into them. The board is supported underneath by two shorter hexagonal posts of the same size directly below them. Bolts screw in through the bottom face, through the bottom posts, through holes in the PCB and into the top two posts.

Above the Ethernet socket in Figure 4.3 is the green programming connector for programming the microcontroller. To the right of the programming connector is the 25 MHz clock source. To the right of that is the 64 pin microcontroller. By the top left corner of the microcontroller is the EEPROM containing the node's MAC address. To the left of that are three clear light guides under which are three LEDs. Around these components are various resistors required for their operation and decoupling capacitors. All other components not discussed so far which aren't capacitors are sensor related.

4.1.3 Microcontroller

An 8-bit PIC18F66J60 [45] was selected for use in the sensor nodes. An 8-bit microcontroller was chosen as the sensor nodes did not require significant processing power. The 18F66J60 specifically was selected because it had Ethernet physical layer hardware built in. Having a portion of the Ethernet hardware built in was important because it reduced component count which both reduced component cost, saved space on the PCB and simplified the circuit design — thus saving time debugging.

A side effect of using the built in Ethernet hardware is that the microcontroller needed to be clocked at 25MHz, which is much slower than the 41.667MHz maximum clock rate [45]. The lower processing speed turned out not to be a problem as the sensor node does not have much processing to do. The lower clock rate is actually beneficial as it reduces the node's power consumption. Although the node is powered by mains (either directly or via PoE) and isn't required to minimise energy usage like a battery powered node, low power consumption is still important for the sensor node to prevent self heating from distorting sensor measurements.

4.1.4 Ethernet Hardware

Although the selected microcontroller had an Ethernet physical layer modem built in, some supporting hardware was required to make it functional. Ethernet signals are magnetically isolated from the communicating devices. The required magnetics are not included as part of the microcontroller, so were required externally. The MAG-45 [46] PCB mounted Ethernet socket integrates the required magnetics into an Ethernet socket, reducing component count and saving board space. A specialized Power over Ethernet (PoE) version of the MAG-45 was selected. This version includes bridge rectifiers to rectify a differential voltage between the Ethernet pairs and output it on two power pins. These pins were connected to the node's voltage regulation circuit. In addition to the magnetics the microcontroller's Ethernet module required some external bias resistors and capacitors.

4.1.5 Sensors

The nodes were designed with a large number of sensors, several of which measured the same parameter. This is because project is intended to provide a reference for future node designs and research, and investigates new methods of predicting occupant comfort. Some parameters such as the pressure and humidity had both lower and higher cost sensors. The reason for this is to provide data to guide future design decisions on whether the higher accuracy and resolution of the more expensive sensors is worth their added cost. Other sensors such as the Light Dependant Resistors (LDRs) had duplicates positioned in different physical orientations to measure light from different directions. This was to allow investigation as to whether the direction the light in the room came from contains any useful information.

Temperature

There are two temperature sensors on the Ethernet nodes. The most precise is on the SHT11 humidity and temperature sensor module. This sensor uses a two wire serial data interface to communicate 14-bit temperature readings with a resolution of 0.01°C. The SHT11 datasheet claims an accuracy of 0.3°C [47]. The other temperature sensor on board is the SP100-7T combined pressure and temperature sensor. This sensor reports temperature to integer precision via SPI. Its datasheet claims accuracy to be within 2.4 to 4.8 degrees of the actual value [48].

The power usage of the nodes is low enough that self heating of the node is not believed to be an issue that will affect the temperature sensors. Additionally the SHT11 datasheet recommends taking samples at a frequency of 1Hz or less to keep self heating of the sensor below 0.1°C. The $1/30$Hz sampling rate used by the nodes satisfies this condition.

Humidity

There are two humidity sensors on the Ethernet nodes. One is the more expensive SHT11 sensor which uses serial communication, the other is the cheaper Honeywell HIH-5031 analogue humidity sensor. Their datasheets claim they both have the same 3 % RH accuracy [47, 49]. However the SHT11 has its reading digitized by an in package 12-bit ADC, whereas the HIH-5031 is read by the microcontroller's 10-bit ADC via a connecting PCB trace which is more susceptible to external interference.

Pressure

There are two pressure sensors on the sensor node. There is the more expensive SP100-7T sensor which uses SPI to communicate its readings which were taken using an internal ADC. The second pressure sensor is the MPXA6115A which outputs its reading as an analogue voltage. The MPXA6115A is internally temperature compensated [50]; whereas the SP100-7T requires a calculation be performed using a temperature reading (such as one from its internal temperature sensor), using a formula provided in the relevant datasheet [48]. The analogue output voltage is read by the microcontroller's 10-bit ADC. The datasheet for the SP100-7T does not specify the resolution of its ADC as a number of bits or otherwise, however it state that it has a resolution of 1.25 kPa per LSB — which is approximately an order of magnitude less precise than the MPXA6115A.

Light

There are two types of light sensor on the sensor nodes. The first is a TAOS TSL252R-LF photodiode which outputs a voltage that is linearly related to light level [51]. The second type of light sensor is the cheaper Light Dependant Resistor (LDR). The two types of sensor are sensitive to different regions of the electromagnetic spectrum. It wasn't known at design time whether there was any advantage to using the more expensive photodiode over the cheaper LDR, so both were included on the final node design to gather data to inform future decisions.

There is one linear photo diode mounted to the top of the node which faces forward. There are four LDRs mounted on the node's case, one facing forward, one facing left, one facing right, and one facing upward. The reason for multiple LDR sensors which face in different directions was to permit investigation as to whether detecting the direction as well as the intensity of light provides any information which can be used to predict thermal comfort. The reason for duplicate LDRs instead of duplicate linear photodiodes was due to the lower cost of the LDR relative to the photodiode. The forward facing LDR is placed in close physical proximity to the linear photodiode so that they receive approximately the same level of illumination. Though the design of the case and the photodiode result in the photodiode being shielded from illumination coming from a light source at a lower height than the node.

A comparison of the sensors has been performed and an equation was derived from the results to convert linear photodiode voltage readings into the equivalent values that the LDR would produce given the same illumination level. The derived equation is shown in Equation 4.1. The test exposed both types of sensors to the ambient indoor light which consisted of both artificial light and indirect sunlight. The light level was varied artificially before being allowed to vary naturally. The artificial variation was to record the sensors response to light intensities that were higher than would typically be expected in the normal use of the nodes. The test lasted for slightly less than 24 hours and sampled the output of the sensors once per second. The data was collected by connecting the sensors to a National Instruments PCI-6120 and using LabVIEW to record the values. Figure 4.4 shows a plot of the LDR output voltage values vs the linear photodoide output voltage for a range of light levels.

$$V_{ldr}(V_{ll}) = \begin{cases} 0.017e^{1.4347V_{ll}} & \text{if } V_{ll} < 2.1 \\ 5.9037V_{ll}^4 - 53.421V_{ll}^3 + 181.86V_{ll}^2 \\ \qquad -275.17V_{ll} + 156.13 & \text{if } 2.1 \leq V_{ll} < 2.75 \\ 0.0000353909e^{3.84219144432V_{ll}} & \text{if } 2.75 \leq V_{ll} \end{cases} \tag{4.1}$$

It was found that the LDR responded to a greater range of light levels. Due to the non-linearity of the LDR it provided a higher sensitivity when reading low light levels. At high light levels, while the photodiode provided greater sensitivity, its output clipped once a certain intensity was reached, where as the LDR's output remained analogous to the light level.

Occupancy

The node features a Passive Infrared (PIR) occupancy sensor. This sensor outputs a logic high voltage when it detects a sudden change in the distribution of infrared sources in its field of view — such as a person moving. Having a measure of room occupancy is very important for this project as it permits analysis related to heating of unoccupied rooms and the impact of room occupancy on the environmental conditions. The PIR sensor only provides a boolean value of whether the room is occupied or not. A count of the number of occupants would be preferred, but determining the number of occupants is a difficult problem to solve and would have required more expensive hardware. The PIR sensor used on the sensors nodes is a Panasonic AMN34112 [52].

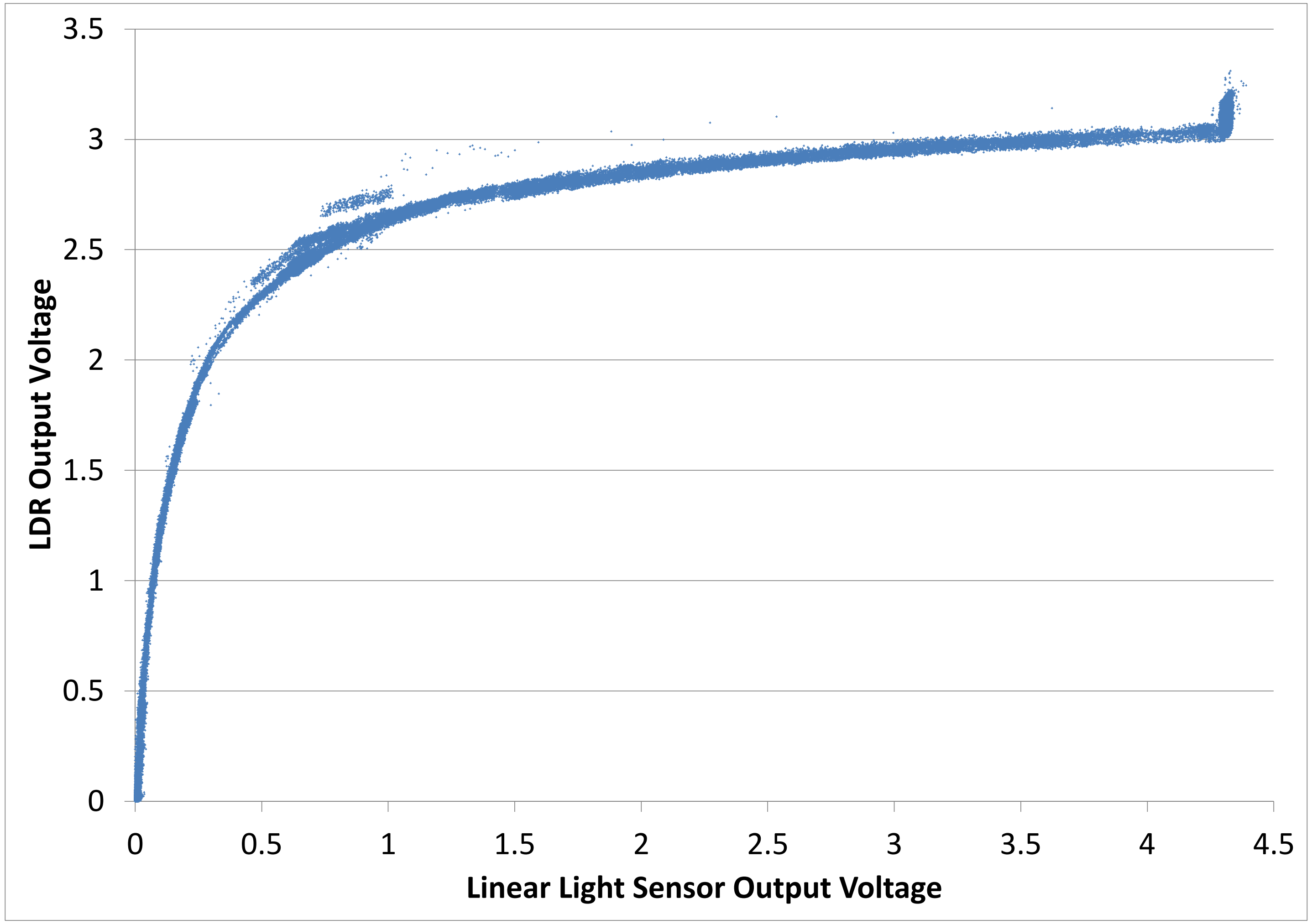


FIGURE 4.4: Scatter-plot of the output voltage of the LDR against the output voltage of the linear photodiode. There are 85542 data points plotted.

Sound

The nodes have an electret microphone on board. The microphone is used to measure the average and peak amplitude of the sound around the node. It was thought that the sound level could potentially be indicative of occupancy. If this is the case then the cheaper microphone could be used in place of the more expensive PIR occupancy sensor in future systems which require low cost occupancy detection. Including a microphone alongside an occupancy sensor allows analysis to be performed to determine this. It also allows for investigation as to whether the ambient sound contains information which could be used to predict occupant comfort.

The microphone is connected to the microcontroller via two opamps. The signal travels from the microphone to the first amplifier via an AC coupling capacitor and resistive biasing network [53]. The first opamp implements a non-inverting amplifier with a gain of 13. A non-inverting amplifier is used to avoid the amplifier affecting the signal from the biasing circuit. The second opamp is used in an inverting configuration with a gain of 12. A potential divider provides a reference voltage to the amplifiers halfway between 3.3V and 0V. The circuit diagram of the microphone circuit is provided in Appendix B.10.

System Voltage

The SP100-7T digital pressure sensor also measures the actual voltage of the system's +3.3V supply rail. Using this measured value instead of a constant 3.3 when calculating the values read by the microcontroller's ADC was considered. However it was decided against doing this as it would require additional testing to see if it improved results, or if it would only introduce more noise.

Sensors Not Used

It was desired to measure CO_2 and air movement but sensors to measure these parameters were not included on the nodes. The low cost CO_2 sensors examined typically operated on 12V or more and made use of a heated element. The nodes do not have a regulated 12V supply and it was undesirable to add a third positive voltage rail to the nodes. The warm-up time of the heating element would have necessitated running the heater constantly. There was concern that the small size of the node would have meant that the heater would have affected the temperature sensors. The final and most significant issue preventing the use of a CO_2 sensor was that an appropriate supplier could not be found at the time of design and testing.

Air velocity is one of the main factors affecting thermal comfort and thus measuring it is relevant for a sensor network studying thermal comfort. Unfortunately air velocity meters designed for use in the open air (as opposed to measuring flow in a pipe) are larger than the node and are costly. Locating an air speed sensor was also problematic. It was undesirable to have sensors mounted so they stuck out of the node, or were connected via a cable to it and mounted near by. It was also suspected that the walls of the node's case would restrict the flow too much for measurement of the air speed to be performed inside the node's case. Using a thermistor to produce a value analogous to airspeed was considered. The principal of operation is that a thermistor is kept at a temperature above ambient by a control loop that alternately measures its resistance and heats it by driving current through it. Airspeed affects the cooling rate of the thermistor, thus when airspeed is higher, the control loop has to heat the thermistor more to keep it at the target temperature. Other design priorities prevented the investigation and development of this sensor.

4.1.6 Sensor Calibration

Calibration of a number of the Ethernet node's sensors was performed at the Centre for Atmospheric Science which is part of the University of Manchester's School of Atmospheric and Environmental Science. The sensors tested were the pressure, humidity and temperature sensors. The tests were performed to investigate whether there was much variation between measurements of the same value made by different nodes, the accuracy of the values, and to determine correction factors to map measured values to true values.

Humidity and Temperature

The humidity and temperature sensors were tested at the same time as relative humidity is a function of temperature. The reference measurement devices were a Retronix Unicap S3 and a Retronix Hygroclip SC05. Two Ethernet Nodes were sealed in a bell jar connected to a dew point generator. The Ethernet Nodes transmitted their measurements to a laptop running the Data Receiver Server software and a PostgreSQL database. The measurements from the reference instruments were logged to a file on another computer. Figure 4.5 shows the experimental setup measuring the ambient temperature and humidity prior to placing the nodes in the bell jar.

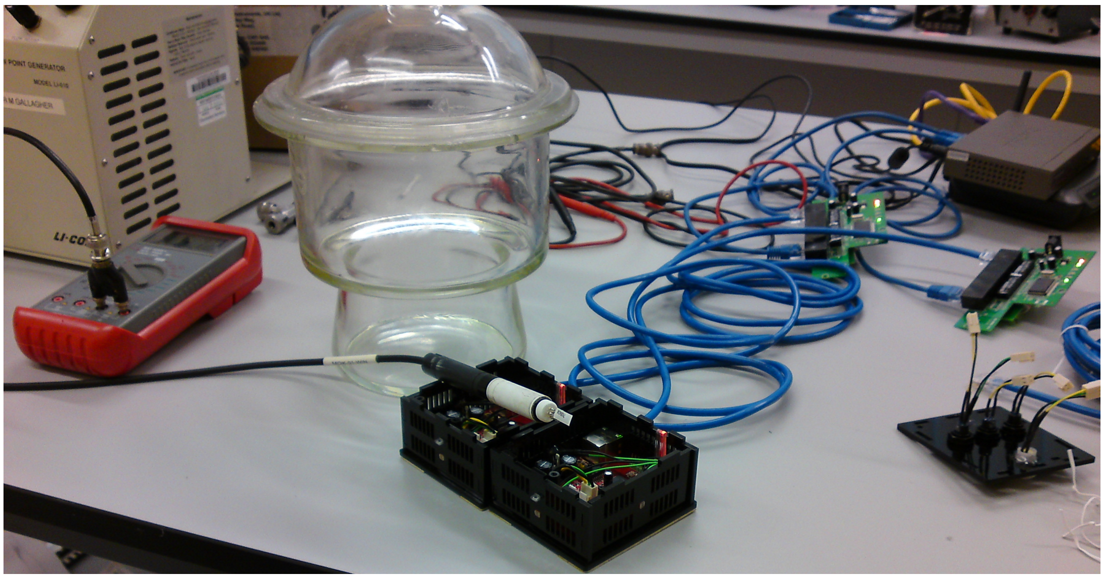


FIGURE 4.5: Image from left to right: The beige box is the dew point generator; a red and grey multimeter; the bell jar used in the experiments; the multimeter is connected to a reference humidity sensor on top of two sensor nodes which have their front panels removed; blue Ethernet cables connecting the nodes to modified Ethernet switches (green PCBs), which provide power to the nodes and connect their Ethernet data connection to the Power over Ethernet supplying Ethernet switch (dark beige box top right) that is on top of a router and connected to it with a purple Ethernet cable; a router which provides DHCP functionality and is connected via a yellow Ethernet cable to a laptop (not shown) which receives data from the nodes.

The dew point was varied using the dew point generator. First the dew point was swept from 2.7°C to 17.2°C over a period of 48 minutes. Next the temperature was increased to 32°C over 20 minutes using a hot air source applied to the outside of the bell jar. The temperature in the jar was decreased to –3°C by placing the jar in a freezer. It was noted that the low temperature caused one node to stop working below 4°C and the other at 0°C. Both nodes began functioning — intermittently at first, then normally — once restored to a warmer temperature.

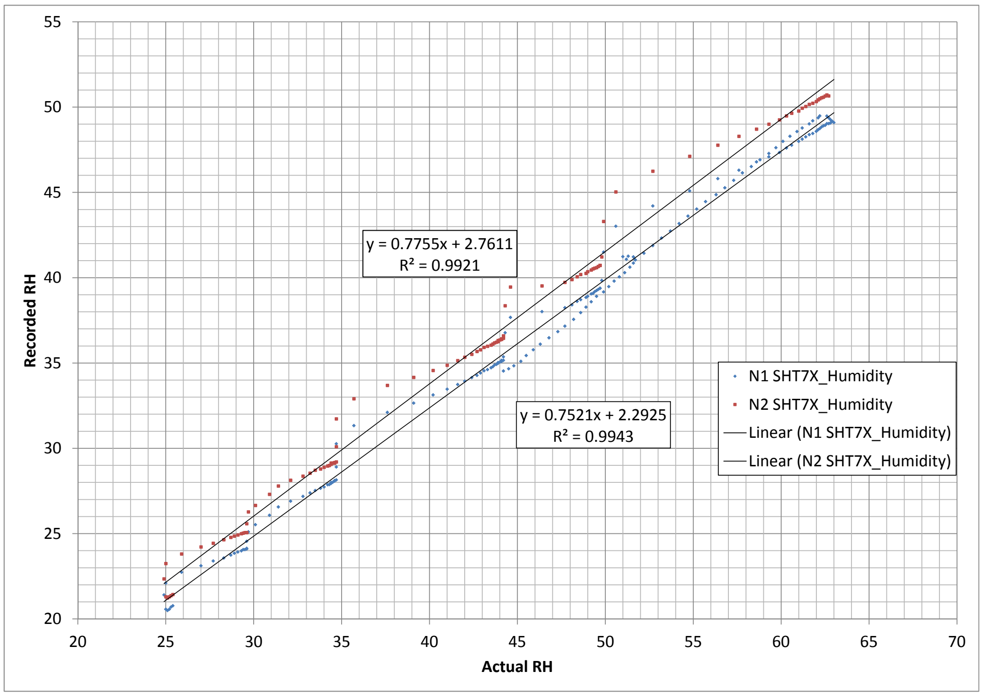


FIGURE 4.6: A plot of the relative humidity measured by the two nodes and against the relative humidity measured by the reference sensor. Linear trend lines are also shown, as is the equation and R squared value for one of the trend lines. There are 447 points in the plot.

A program was written to read the results from the sensor nodes and the reference sensors and output values with aligned timestamps to a file. Analysis of the data mainly focused on the node's SHT11 sensor as this is the most accurate and highest resolution temperature and humidity sensor on the node. Analysis showed that both the humidity and temperature measurements did not match closely to the reference values, but were linearly related to the actual values, so are simple to correct. The results for the humidity measurements shown in Figure 4.6 show that the humidity is incorrect by a constant factor of about 1.29. The results above 80% RH were ignored since they were recoded while the nodes were not operating correctly due to being frozen.

The results for temperature are shown in Figure 4.7. They show that the temperature that the sensors report is offset from the real the temperature by a constant 5°C. Aside from the constant offset, the temperature reported by the SHT71 sensors had an excellent linear relationship with the real temperature. The curve between 2°C and 12.5°C should be ignored for the purposes of calibration as it was recorded while the node was warming up after being frozen — a state that had a detrimental affect on the node's functionality.

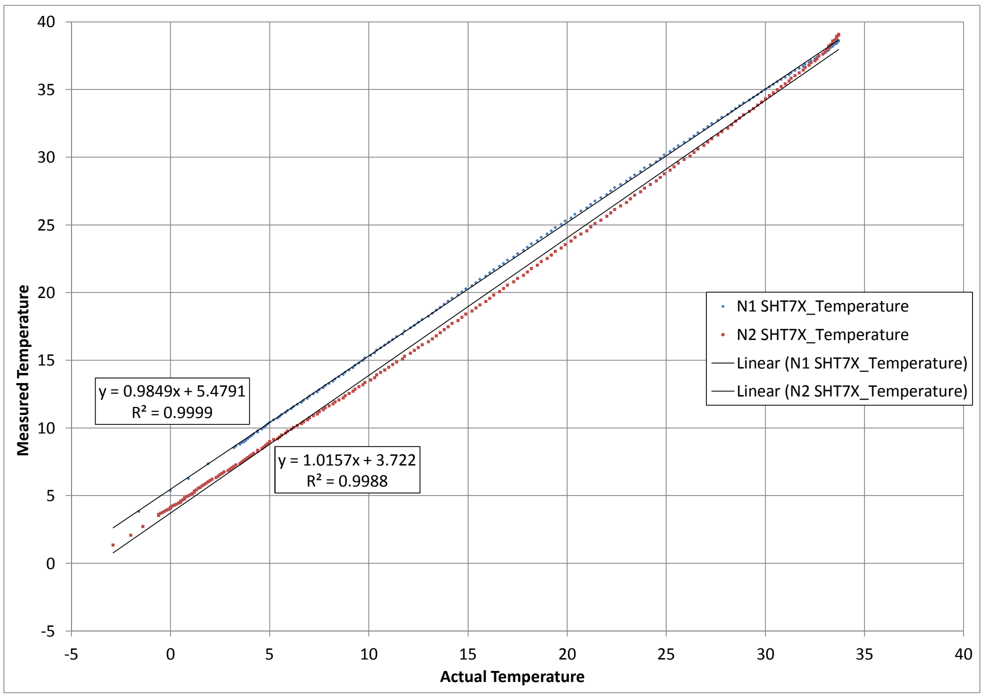


FIGURE 4.7: A plot of the temperature measured by the two nodes and against the temperature measured by the reference Retronix HygroClip SC05 sensor. Linear trend lines are also shown, as are their equations and R squared values.

Pressure

The Pressure test was carried out after removing the nodes from the freezer for the temperature test. This caused the nodes to function intermittently. It was possible to complete the pressure test, since the nodes would send one reading each time they were reset before they stopping working again. As the nodes warmed up they began to function normally. The pressure test was performed by placing a reference pressure probe in the bell jar with the nodes and connecting the sealed jar to an air pump. The reference sensor was a Baritron 750brdpcd2ga. Like the temperature and humidity tests, the node's readings were stored in the database. Unlike the tests the reference pressure readings were recorded by hand on computer with timestamps taken from the same clock that the sensor nodes used as a reference.

For the first part of the test the pressure in the jar was reduced to 13.5PSI. The pressure was then returned to normal atmospheric pressure. The pressure was then increased to 15.275PSI. The range of pressures the nodes were subjected to was chosen to exceed the range of pressures they could be reasonably expected to experience under normal atmospheric conditions.

The tests showed that the uncalibrated values of the analogue pressure sensors were below the actual pressure by approximately 20 kPa, but that they did vary with changing pressure, and that the errors could be corrected for to produce accurate values if required. Figure 4.8 shows a scatter plot of the pressure calibration results. There are fewer points than for other plots because the reference samples were recorded manually, and because the nodes were operating intermittently due to having been frozen in the earlier temperature calibration experiments.

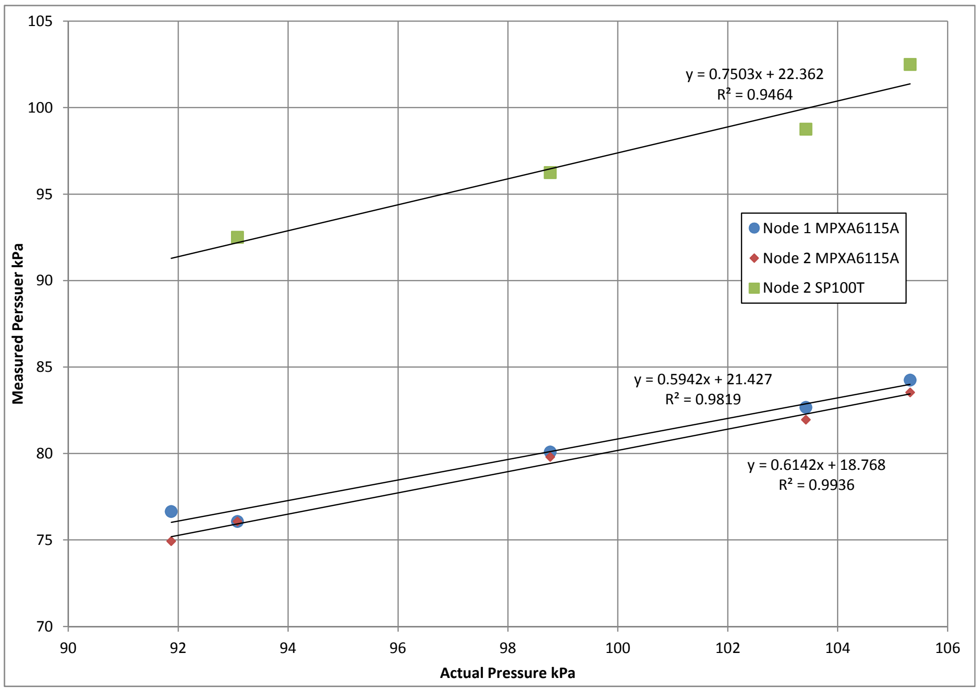


FIGURE 4.8: A plot of the pressure measured by the two nodes and against the pressure measured by the reference Baritron 750brdpcd2ga sensor. Linear trend lines are also shown, as are their equations and R squared values.

4.1.7 Case

The project required the sensor nodes to be in operation for an extended time period and be subject to user interaction while operating. This resulted in the requirements that the sensor node case be robust, be of a build quality above the typical lab prototype, and not be unsightly. There were additional requirements that the case be cheap and simple to manufacture; that it not inhibit airflow to the internal sensors; that it provide a non-destructive method of wall mounting; and that it convey instructions for its user interactivity.

The final design for the case — shown in Figure 4.1 — features 6 interlocking pieces of acrylic. The pieces are primarily held together by friction, but four screws connected to two internal pillars are used to provide a compressive force for added security. The internal pillars also served to secure the internal circuit board in position. The case was designed at the same time as the PCB layout was being finalised. This was so that components could be moved to provide space for the case's PCB supports to reach the PCB without interfering with components, and so the case could be designed to accommodate the Ethernet socket.

The acrylic pieces of the case were cut from a sheet of acrylic using a CNC laser cutter. The laser cutting process was found to be very flexible and was used to achieve some more sophisticated objectives. One example is the QR code which appears on the front of the nodes. The QR code was produced by applying masking tape to the approximate region of the front panel where the QR code should appear. Then using the laser cutter on a low power setting, burning away the masking tape and a small amount of the acrylic in the regions of the QR code which were to be painted white. When painted with a spray paint the remaining masking tape protected the regions which were to remain black, but filled the burnt depressions which were intended to be white. The tape could then be removed once the paint was dry so that paint only remained in the depressions forming the QR code.

The laser cutting process was also modified to achieve the objective of affixing the node to walls in a non-destructive manner. This was done by applying double sided foam tape to the approximate area from which the back panel of the node case was to be cut from. When the back panel was cut the tape was also cut to perfectly align with the edge of the case. The end result when assembled is that the wall-facing face of the case has a layer of double sided adhesive foam pre-applied which perfectly follows the edges of the node. It was decided to use foam tape as the method of installation since it doesn't require any drilling; and it is compressible meaning that it will conform to an uneven surface while holding the node steady. It also makes the nodes very simple to install as one simply needs to peel off the backing paper covering the foam tape, and then press the node's adhesive side against the wall one wishes to install it to (after first ensuring the location is dust free).

Clear acrylic light guides were cut using the laser cutting process. These fitted into slots cut in two of the six pieces of the black acrylic case during their laser cutting process. They guided light from three surface mount LEDs on the node's PCB to the front of the node.Having light guides rather than case mounted LEDs reduces the cost as SMD LEDs are cheaper and require less labour to install. Subjectively the light guides give the nodes an appearance of being higher quality than case mounted LEDs would.

Psychological Considerations

While the nodes were being designed it was thought that the colour of the nodes might influence the comfort votes — a blue coloured node might subconsciously suggest cold to an occupant about to vote, or a red node might suggest hot to them. In order to avoid introducing a bias to the occupant comfort votes careful consideration was given to the visual appearance of the case design. The colour of the case was chosen to be black as black doesn't have a temperature commonly associated with it. Black was also chosen to give the nodes a more stylish appearance. The question text 'ARE YOU' and QR code were coloured white as white also doesn't have a temperature commonly associated with it and it contrasts well with black.

The buttons and their associated labels 'TOO HOT', 'JUST FINE' and 'TOO COLD' were given colours different to the white text to give visual cues as to which button was for which option. The 'TOO HOT' button and text were coloured red as red is considered a warm colour. The 'TOO COLD' text and button were coloured blue as blue is considered a cool colour. The 'JUST FINE' text and button were coloured green. This was because green isn't thought to suggest a temperature but is a different colour than the other temperature neutral colours already used on the node (black and white).

Care was taken to make the total area of each colour used for the buttons and their labels approximately equal. This was done to prevent one colour or option dominating the appearance of the node which might appear to suggest a preferred option. This was done by selecting words for each option with an approximately equal number of letters so that the text could be arranged in a similar manner for each option while taking up a similar sized area. This is why term ‘JUST FINE’ was used rather than ‘Neutral’, ‘Comfortable’ or ‘OK’. ‘OK’ also has the problem of being commonly used on buttons which confirm actions, which might result in people thinking they need to press it after selecting one of the other options.

4.1.8 Power

The requirements that the nodes operate indefinitely with zero maintenance ruled out the use of batteries as a sole means of powering the nodes. The large number of sensors on the node results in a power usage which is too great to make energy scavenging approaches practical. This left low voltage DC adapters powered by mains power as the only viable long term option. A downside of using mains is that it requires running a wire to the node. This limits the potential installable locations of the nodes to places near power points — unless long cables are used. Long power cables add to installation costs, and some consider them unattractive.

Since a sensor node will already have a cable running to it for communications, it was investigated as to whether that cable could serve a dual purpose and supply power to the node in addition to being a communication medium. The IEEE has a standard for such a use case known as IEEE802.3af Power Over Ethernet (PoE) [54]. Ethernet cables consist of four pairs of wires. PoE provides power by applying a differential voltage between two or more pairs of wires. This does not affect the Ethernet communication as signal pairs are magnetically isolated; and the voltage applied to the pair is a common mode voltage, which doesn’t affect the differential signal voltage.

Power over Ethernet requires some additional hardware in addition to voltage regulators in order to perform a handshake protocol. This protocol signals to the Ethernet switch that the device supports Power over Ethernet and how much power the device requires. This additional hardware adds cost to the node, but is offset by the fact that many powered devices can share a single power-supply instead of requiring one each.

There was a requirement that the nodes could be installed without removing the occupant’s access to the room’s Ethernet port. To meet this requirement it was decided to use an Ethernet switch to share the port between the node and the occupant. Since the node required power it was decided to put the PoE receiving hardware on the Ethernet switch. These switches are discussed further in Section 4.2. Adding PoE receiving hardware to the node, and both PoE receiving and supplying hardware to the switch would have increased the cost of the system and required further hardware design work to build the PoE supply hardware. Instead it was decided that the switch would receive power using IEEE802.11af PoE, but would simply supply 12V to the spare pair contacts on one of its ports.

When it was time to deploy the Ethernet nodes there were delays in acquiring the Power over Ethernet supply Ethernet switches. An alternative means of powering the nodes and switches was used. A number of Ethernet cables were modified by connecting a female DC power receptacle to the unused spare pairs of the Ethernet cable. The connection was made in such a way that only the spare pair pins on the unmodified end of the cable would be connected to the DC power. The unmodified ends of these cables were connected to the Ethernet nodes. The modified ends of the cables were connected to the Ethernet socket providing a connection to the network. A 12V DC wall adapter was then connected to the power receptacle on the modified cable to power the node. In the case that there were no free Ethernet sockets in the installation area, an unmodified 5 port Ethernet switch was used to share the Ethernet socket with the existing users and the node. Both the switch and the node were powered with a wall adapter which was modified to have two plugs on the end of its DC output cable. One of the plugs powered the Ethernet switch, the other powered the node via the modified cable.

A high efficiency switching voltage regulator was used for the main 3.3V supply on the node. This was to minimize the effect of component power consumption heating the node and affecting sensor measurements. A cheaper linear regulator was used for the 5V rail. This was acceptable as there are only two sensors powered by the 5V rail. These draw very little current, resulting in little power dissipation in the voltage regulator module.

4.1.9 Node Cost

The nodes were designed to be low cost; but with the exception that there would be duplicate sensors - which would inform future design decisions and possibly lead to cheaper future nodes. The total cost of all the components on the Ethernet Node's PCB is £57.34. The cost of PCB manufacture and assembly was £51.70 per board, with an additional setup and tooling cost of £255 for the batch of 20. The cost of the a 2030mm x 1500mm x 3mm acrylic sheet the cases for the node were cut from was £177.30. Each node required 30290 mm^2 costing £1.76. Each node required 200 mm of adhesive tape which came from a 25m roll costing £15.07 — total cost of tape per node was £0.12. There were an additional £13.91 of sensors and buttons mounted on the case. This makes the total cost per node £137.58 excluding labour costs of painting the nodes, assembling the case and fabricated PCB.

The cost of the nodes can be reduced by removing the duplicate sensors. The selection of which sensors to remove can be informed by the data collected by the installed nodes. Removing some of the sensors directly reduces the cost; but also has a compound effect on the cost as fewer sensors results in a smaller PCB — which is cheaper to fabricate — and a smaller case — which requires less material. The cost would be further reduced if the +5V rail was removed — by replacing the sensors that operate on 5V with ones that would run on the 3.3V rail, or if they were removed entirely — as this would eliminate the 5V voltage regulation hardware, again saving on both component cost and board space. Manufacturing a larger number of nodes would also reduce costs as components could be bought in larger quantities for a lower cost per component, and a lower manufacturing price per PCB could be negotiated.

4.2 Power over Ethernet Switches

It was required that the installation of an Ethernet Node should not result in the disconnection of an existing device from its Ethernet connection. To achieve this in locations where all available Ethernet sockets are already in use (or will be used in future) the use of an Ethernet switch was required. It was originally planned to integrate a 3 port Ethernet switch into the Ethernet node, however there was insufficient time to redesign the node and debug both the switch software and hardware. Instead it was decided to use low cost commercial Ethernet switches. As the use of Power over Ethernet had already been decided upon for the project it was decided that the Ethernet switch should be powered using Power over Ethernet to avoid consuming a mains socket. As PoE powered Ethernet switches are targeted at businesses, there are no low cost consumer versions available commercially. It was decided to modify low cost non-PoE powered Ethernet switches to be powered by PoE.

A sample 5 port Ethernet switch was purchased, disassembled and assessed visually and electrically. It was found to be designed in such a way which made it readily modifiable for the project's needs. The switch was modified by removing the Ethernet transformer from one of the Ethernet ports and attaching a custom designed PCB in its place. The custom PCB performs the same role as the removed transformer — magnetically decoupling the Ethernet data signals — but also implements IEEE802.11af Power over Ethernet. The custom PCB converts the received power into 12V DC which it supplies to the voltage regulator input on switch's PCB. The 12V is also supplied to the spare pair connections of another of the switch's Ethernet sockets to provide power to the Ethernet Node. The modified Ethernet switch with custom PCB is shown in Figure 4.9.

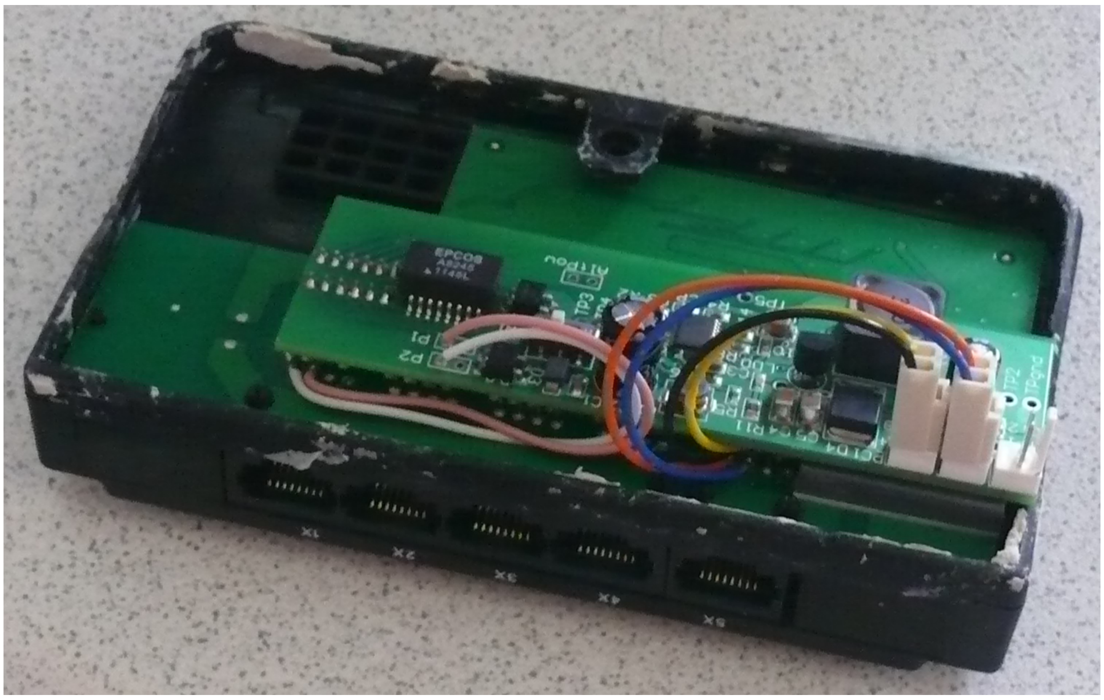


FIGURE 4.9: The Power over Ethernet Switch circuit boards. The upper smaller board is the PoE receiving board. The lower larger board is the Ethernet switch. The upper board receives power using IEEE802.11af PoE from the left most Ethernet socket and supplies 12V to the right most socket and the lower PCB.

4.3 Deployment

The main deployment of Ethernet nodes occurred early in September 2012 in which 15 nodes were deployed. Fourteen of the nodes were deployed across two floors of office space in the Sackville Street Building, which is a grade 2 listed building owned by the University of Manchester. The lower floor consisted of traditional offices, whereas the upper floor is an open plan work place. The locations of the deployed nodes are shown in Figure 4.10. This part of the building has two exterior walls and no floors above it. The fifteenth node was deployed to an office located in the corner of the building, but on the same floor as the lower floor to which the majority of the nodes were deployed to.

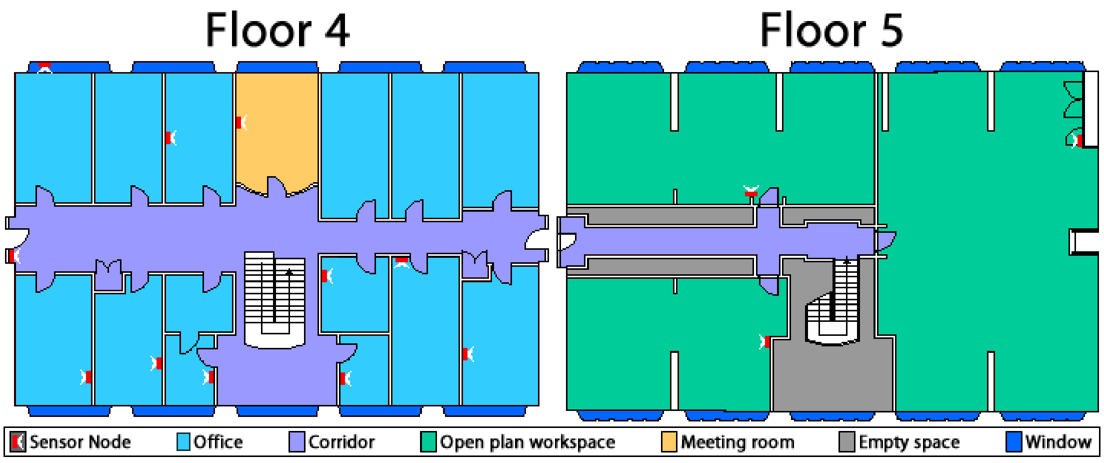


FIGURE 4.10: A map depicting the locations of nodes in the main deployment area

Two more nodes were deployed in October and December. One was located in the office next door to the office the fifteenth node was deployed to. The other was deployed to a student work space distant from the main deployment area.

4.4 Footstep Detector

While investigating possible sensors for the sensor node a footstep detector was conceived of, designed, built and tested. The motivation behind the footstep detector was that footsteps are indicative of occupancy — making the footstep detector a motion sensitive occupancy detector which is not limited by line of sight. The design featured a laser which was bounced between two mirrors which were suspended on thin plastic cables and onto a light dependant resistor. The output voltage of the LDR was then fed to a software implemented correlator. The correlator was created using LabVIEW and matched the received signal against a reference signal taken of the output after a footstep. When the output of the correlator exceeded a threshold it asserted a footstep had been detected, otherwise it indicated that no footstep had been detected. The prototype detector is shown in Figure 4.11. The photo was taken inside a box filled with artificial smoke to allow the path of the laser to be photographed.

The detector's method of operation is hereby explained. Footsteps cause vibrations in the surrounding area, the suspended mirrors when subjected to these vibrations would themselves then vibrate. The mirror's vibration would alter the path of the laser. Since the laser bounced multiple times between the two mirrors, the affect of the vibration would be multiplied since the laser's path is adjusted in multiple places.

Sensitivity may be adjusted by varying the number of reflections of the laser, though this is limited by the fact that each reflection reduces the laser's intensity. The deflection of the laser from its quiescent path is detected by the LDR — which under quiescent conditions is covered by half the laser's light circle — as an increasing and decreasing amount of light exposure. The correlator distinguishes the detected vibration pattern created by footsteps from other vibration patterns which may occur in the environment.

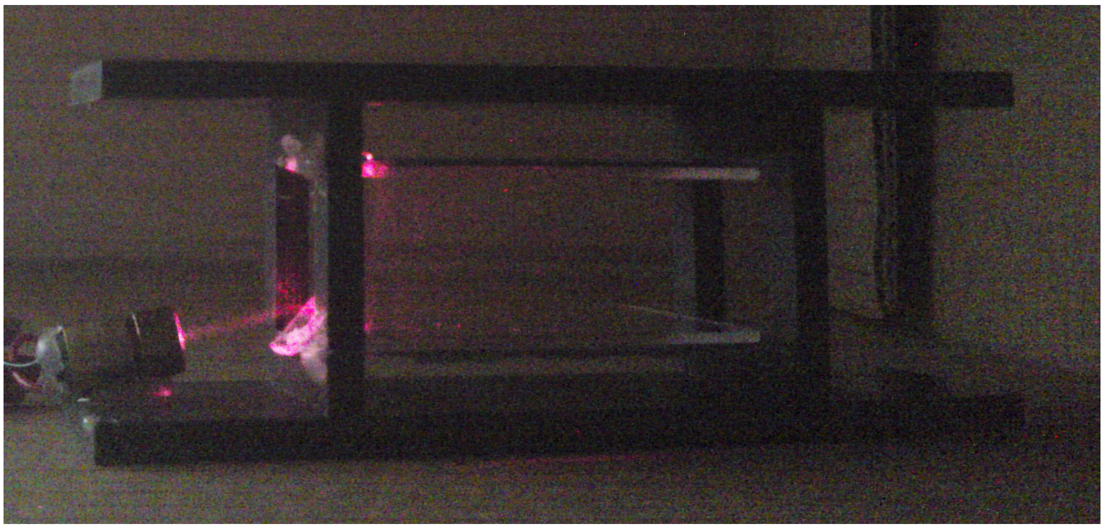


FIGURE 4.11: The footstep detector prototype

The footstep detector was found to work well, however it was not used in the sensor nodes for a number of reasons. Its size was similar to the sensor node's, which would have made it difficult to integrate. While the effort required to build it was not significant, producing 20 duplicates would have required too large an investment of time. Additionally the correlator was implemented on a PC, and it was unclear whether the 8-bit microcontroller on the sensor nodes could perform the correlation without interfering with its other operations. It was decided not to use the foot step detector before any quantitative analysis of its performance was carried out. It is described here due to its novelty and because the initial qualitative assessment of its performance suggested that it is suitable for further development.

Chapter 5

System Software

This chapter describes the software written for this project both for the sensor nodes and the server. Two of the most significant items are the software on the sensor nodes, and a piece of software running on the project's server called the Data Receiver Server (DRS). The software on the node is responsible for initiating communication with the DRS and for taking measurements at the times the DRS tells it to. This is discussed in detail in Section 5.3. The DRS is responsible for coordinating the nodes and interfacing with a database to store the sensor readings from the sensor nodes. The DRS's functionality is discussed in Section 5.2.

5.1 Aspects common to both Server and Sensor Node

There is communication between software running on the server and the software running on the sensor nodes. As a result of this there are some aspects common to both pieces of software. This section introduces those shared aspects.

5.1.1 Communication Protocol

The communication between the sensor nodes and the Data Receiver Server is bidirectional over an Ethernet IP based network. This project uses User Datagram Protocol (UDP) [55] for transmitting messages across the network. UDP was chosen as its simplicity allows for lower latency in communication – which is important for synchronizing the nodes times with the server. Figure 5.1 shows where the communication protocol used sits on the network stack. Communication is always initiated by the sensor nodes as the server has no way of knowing the addresses of the sensor nodes or whether they are online. The nodes know the server's address as it is preprogrammed into them. The server always sends a reply to valid messages from sensor nodes, and the reply always contains the current system time and the next time the node should take measurement samples.

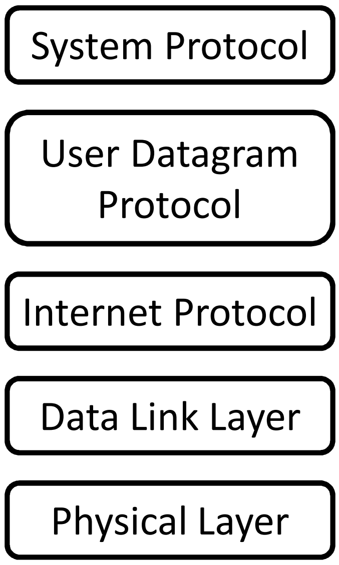


FIGURE 5.1: Network stack diagram.

There are two types of message a node can send, and two types of message the Data Receiver Server can send. The sensor node can send `Introduction` messages and `Sensor Readings` messages. The Data Receiver Server sends `Who Are You` messages and `OK` messages. The message formats are described in detail in Appendix A.

The `Introduction` message describes how many sensors the node has and of what type. It is sent when a sensor node starts up, or in response to a `Who Are You` message. The `Who Are You` message is sent as a reply by the server if it receives a `Sensor Readings` message from a node for which an `Introduction` message has not been received. Figure 5.2 shows the sequence of communication which occurs when an `Introduction` message is sent.

The `Sensor Readings` message contains the sensor readings for the sensors described in the node's `Introduction` message, and the timestamp of when they were taken. The server responds to `Sensor Readings` messages with either the `OK` message — if the node has previously introduced its self — or with the `Who Are You` message. The `OK` message acknowledges the reception of the `Sensor Readings` message and maintains the sensor node's clock and updates the time the next readings should be taken. If the node doesn't receive an `OK` message within a time limit after sending a `Sensor Readings` message then the `Sensor Readings` message is resent. This sequence of communication caused by a `Sensor Readings` message is depicted in Figure 5.3.

The acknowledgement function of the `OK` messages and the resending of the `Sensor Readings` message is an important feature of the communication protocol. This is because the UDP protocol used for communication does not guarantee message delivery. Since the communication medium is wired Ethernet, very little packet loss was expected. So it was assumed that only one retry was necessary — if both failed then there is likely to be something wrong with the network and sending more packets would not help.

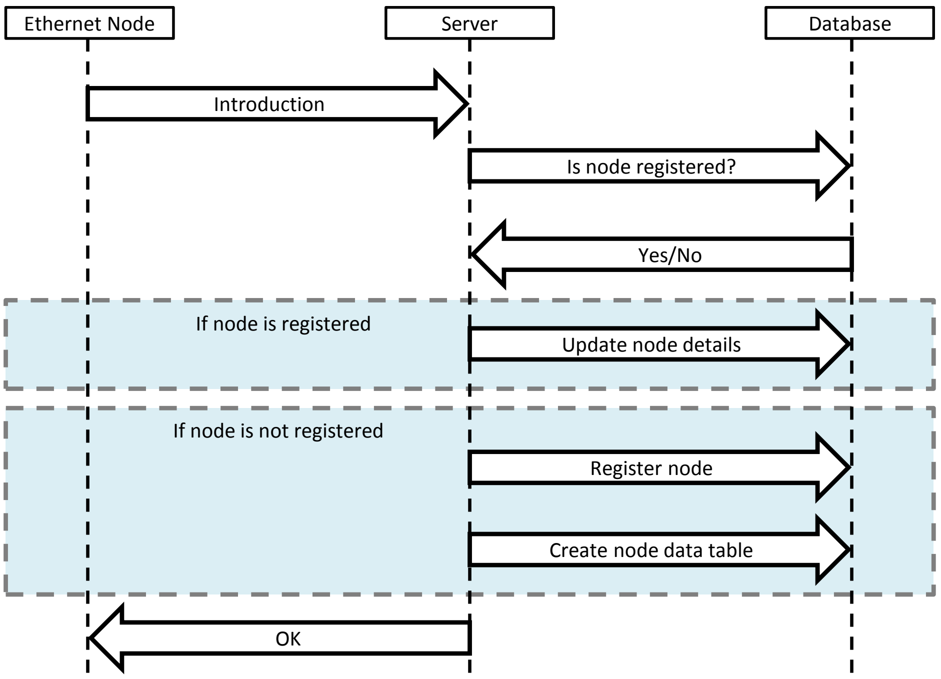


FIGURE 5.2: The sequence of communication triggered by an `Introduction` message.

5.1.2 Time

The subject of representing absolute time over long periods is very complex and nuanced. In day to day life, time is represented as a year, month, day, hours, minutes, and seconds; which seems simple enough until you consider factors such as leap years, leap seconds and daylight saving time (DST), as well as political changes as to when or whether DST applies. Writing software which represents time in this way and takes into account all the exceptions to the general rules and the exceptions to the exceptions is a very difficult task to do correctly. Indeed it's impossible to create a stand-alone clock which represents the current date and time in this form as this form is subject to change — such as the day daylight saving time takes effect changing for political reasons. Since representing time in this manner would have required writing a large amount of code for the sensor nodes, and would most probably fail to handle all the exceptions to the rules and thus become inaccurate at some point, an alternative was sought.

"Unix time" is the way time is represented in Unix based computers. It represents

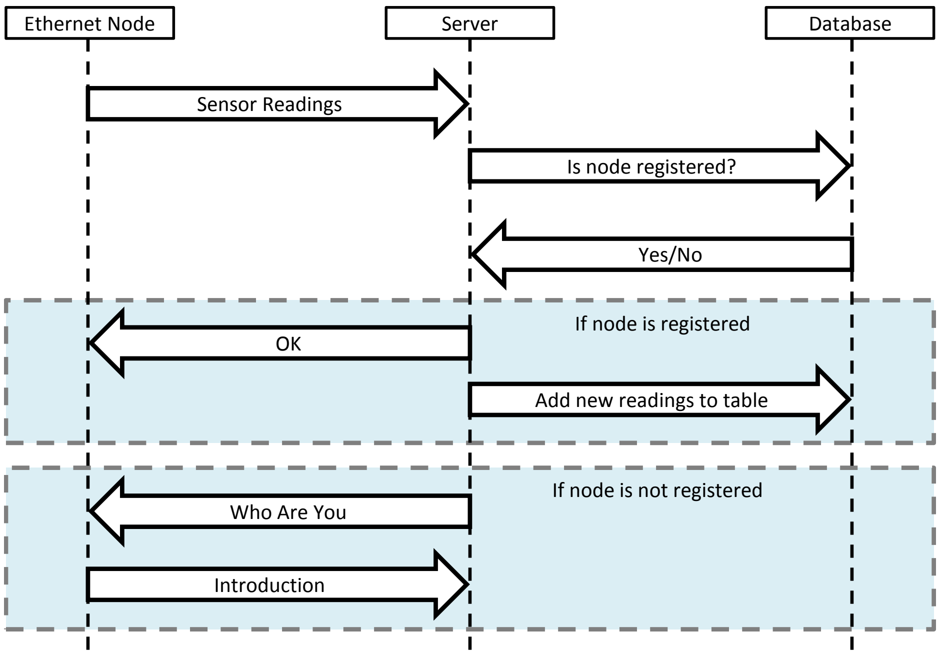


FIGURE 5.3: The sequence of communication triggered by a `Sensor Readings` message. The `Introduction` message sent in response to the `Who Are You` message will trigger the sequence shown in Figure 5.2.

time as a 64-bit integer number of seconds since midnight on January 1 1970 [56] — that date is referred to as the Unix 'epoch'. The number of milliseconds past the second is stored separately. This is a much simpler system of representing time that sidesteps all the complexities and political issues of the previously discussed system. It was decided that this system was also unsuitable for the sensor nodes. This was because working with 64-bit numbers on low power microcontrollers is computationally expensive and sub-integer precision was desired, but not to millisecond accuracy. Instead a custom time format was created which was inspired by the Unix time format.

In the time format used by this project time is represented by an epoch and the number of ticks since that epoch. A tick is a period of 50 milliseconds. The epoch is an 8-bit number used to identify a day. It is used to prevent confusion as to which day measurements belong to — without it there could be ambiguity as to which day measurements taken around midnight belong to should they get delayed. The epoch is advanced one value at midnight each day. Zero is not a valid epoch value. There may appear to be a problem as this system can only uniquely represent 255 days and the sensor nodes are intended to collect data for years. The small number of uniquely identifiable days is not a problem as in this time system the ticks only represent absolute time since the current epoch; and the epoch is used to identify on which of the previous 255 days a message was sent relative to the current day. This means that as long as a message takes less than 256 days to arrive then the server can always determine at what time its measurements were taken. The server converts the custom time format to the standard time format used by the PostgreSQL database when storing readings. The server's time is defined as the correct time, the nodes attempt to minimize the difference between their local clock's time and the server's time.

5.2 Data Receiver Server

The Data Receiver Server (DRS) performs several important functions. The DRS is the software that the sensor nodes communicate with. It maintains the system clock on which the sensor sampling is scheduled. The DRS tells the sensor nodes the current time and when they should next take samples, and the sensor nodes send the samples to the DRS. The DRS manages a PostgreSQL database [57]. It registers new nodes in the database, creating tables and the functions to input data into them as necessary. When samples are received from nodes it stores the readings in the database. Figure 5.4 shows the relationship of the DRS, nodes and database. Physically both the DRS and the database are running on the same machine, and are connected using the loopback adepter rather than physical Ethernet cable.

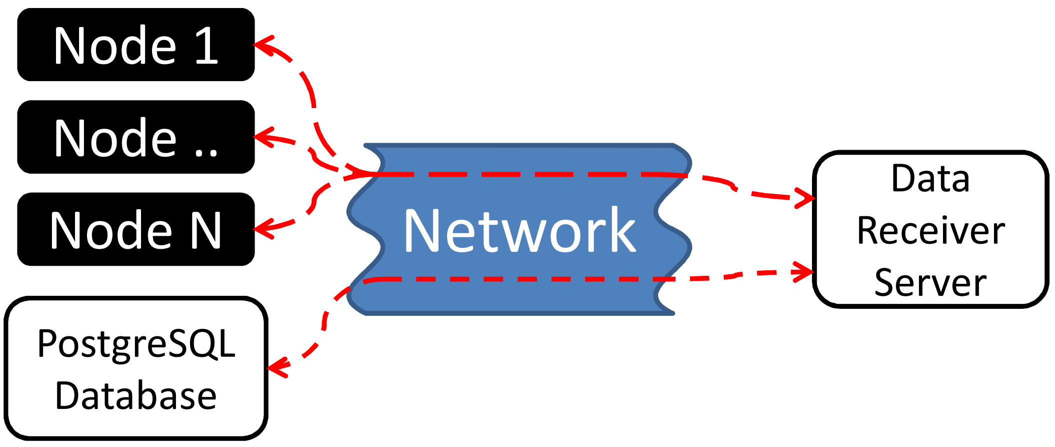


FIGURE 5.4: A diagram showing the relationship of the nodes, the DRS and the database.

5.2.1 DRS Operation

When the DRS starts it initialises the UDPServer module, and if that is successful it sends an email [58] to the system administrator that the DRS has started. The UDPServer module encapsulates the core communication functionality of the DRS. When started the UDPServer opens a UDP port to receive messages from the nodes and creates a new thread to listen to that port and handle any received packets.

When a packet is received the UDPServer checks if the packet starts with a magic number [59, 60] indicating it is a message from a node. If the magic number is incorrect the packet is discarded. If it is correct the server attempts to interpret the packet as a message. If the message has successfully been decoded and has a valid epoch the UDPServer thread passes the message to a thread from the system thread pool, and then returns to listening to listening for more packets. If the packet was valid but it had an invalid epoch then the UDPServer thread replies to the packet's source address with an `OK` message to tell the node the correct time. Passing the received message to the system thread pool for processing is done to allow parallel processing of messages which are received in close temporal proximity.

When the system thread pool is given a message to process it assigns an existing available thread to process the message. The processing of the message begins with obtaining a connection to the PostgreSQL database. What happens next depends on the type of message that is being processed. If the message is an `Introduction` message then an `OK` message is sent in reply, and then the sensor node is registered in the database (if it hasn't already been registered). If necessary, appropriate tables for storing its sensor readings and stored procedures for inserting them — as well as performing other operations — are created. If the node has already been registered then no changes to the database are made. If the number of sensors has changed then the database entries for the node's sensors are changed. Currently the server does not check if the sensor data type has changed, and does not modify the sensor readings table for the node to reflect changed sensors. As a result, nodes that change their sensors are incompatible with the current version of the DRS.

If the message being processed is a `Sensor Readings` message then the database is queried to see if it is from a node which has been registered. If it isn't then a `Who Are You` message is sent in reply to ask the node to register itself. If the node is registered then an `OK` message is sent in reply. After the `OK` reply has been sent, the sensor readings in the message are stored in the database. The time it took from receiving the UDP packet to adding the sensor readings to the database is then recorded to a performance log file. The DRSMonitor program is then sent a message indicating that sensor readings for a particular node have been successfully received and added to the database.

5.2.2 System Clock Implementation

Creating a clock which has precise updates of short time periods and which maintains accuracy over long periods of time on a multitasking operating system (OS) is not a simple matter. The timer libraries provided with most programming languages are typically intended for use in tasks which do not require long-term accuracy. Additionally they do not guarantee that the time between the events they raise will be exactly what was requested as they are subject to the OS's scheduler. The OS does provide timers with guarantees about the regularity of events in the short term, but these are intended for media playback applications and are subject to drift in the long term.

Because of these problems this project took an approach to updating the system clock which didn't involve using timers that generate events at regular intervals. Since the time format used by the project measures time in 50ms ticks past midnight UTC, one can determine in absolute wall clock time when the next tick is. When the system clock is started an event is scheduled to call the time update method at the time of the next tick. When the time update method is triggered by an event, the time is updated. After the time is updated the current wall clock time is obtained from the OS and the number of milliseconds until the next tick is calculated. A timer is then created to generate an event in that many milliseconds time to update the system clock.

Using this method the system clock maintains accuracy over long periods of time, but is still subject to the jitter [61] problems of using the timer libraries [62]. The jitter however is not considered to be a major problem since it is much less than the 50ms of a clock tick. There is however a problem with this method in that it relies on the accuracy of the OS's time. If the OS time is incorrect or drifts then so will the project's time. Indeed it was found that when the OS performed a network time update it was causing sensor reading data to be lost as new data had the same timestamps as already existing data. To address this, steps were taken to force the OS to synchronize its time with an external server more frequently to avoid its clock becoming too inaccurate.

5.3 Sensor Nodes

The software on the sensor nodes is responsible for establishing communications with the central server, taking measurements at the time specified by the central server and sending the measurements to the central server. The software works as a finite state machine. Figure 5.5 shows the states of the sensor node's Finite State Machine (FSM) and transitions relating them.

5.3.1 Communications

The node communicates using wired Ethernet. A modified version of the Microchip TCP/IP library [63] is used to control the node's Ethernet hardware. The library was modified by removing unused parts to save space and patching a bug. The bug was located in the MACPut function of the ETH97J60 file in the 2010-10-19 version of the Microchip TCP/IP library. The function was supposed to write a byte of data to the EDATA register to be sent by the Ethernet hardware. The bug was in a line of inline assembly where a `movff` instruction was reading from a C variable rather than the hardware register which the C variable had been assigned to in the line above the erroneous line. The result of this bug was that no data was written to the register, so no messages sent contained any data.

When the node starts up its first priority is to establish Ethernet communications. First the Ethernet library is configured with default settings and the node's MAC address which is read from an external preprogrammed EEPROM chip. The Ethernet library is then used to activate the Ethernet hardware. All this takes place in the _reset state. The reset state also initilizes the different modules; performing actions such as configuring the LEDs for output, setting the clock to the default invalid time, and resetting the sensor statistics. The node then waits in the _macUnlinked state until the Ethernet hardware reports that there is a valid connection at the hardware level to another device. When this occurs, the node transitions to the _invalidIP state. In this state the node attempts to use DHCP [64] to obtain an IP address for itself, or use a default pre-programmed one if it has been configured not to use DHCP.

After obtaining an IP address the node moves to the _beginDNSLookup state. In this state the node attempts to initiate a DNS lookup to obtain the server's IP address using the url for the server which was preprogrammed into the node. The node will make multiple attempts to contact a DNS server. Should it fail repeatedly it will move to the _unknownServerIP state. In the _unknownServerIP state the node

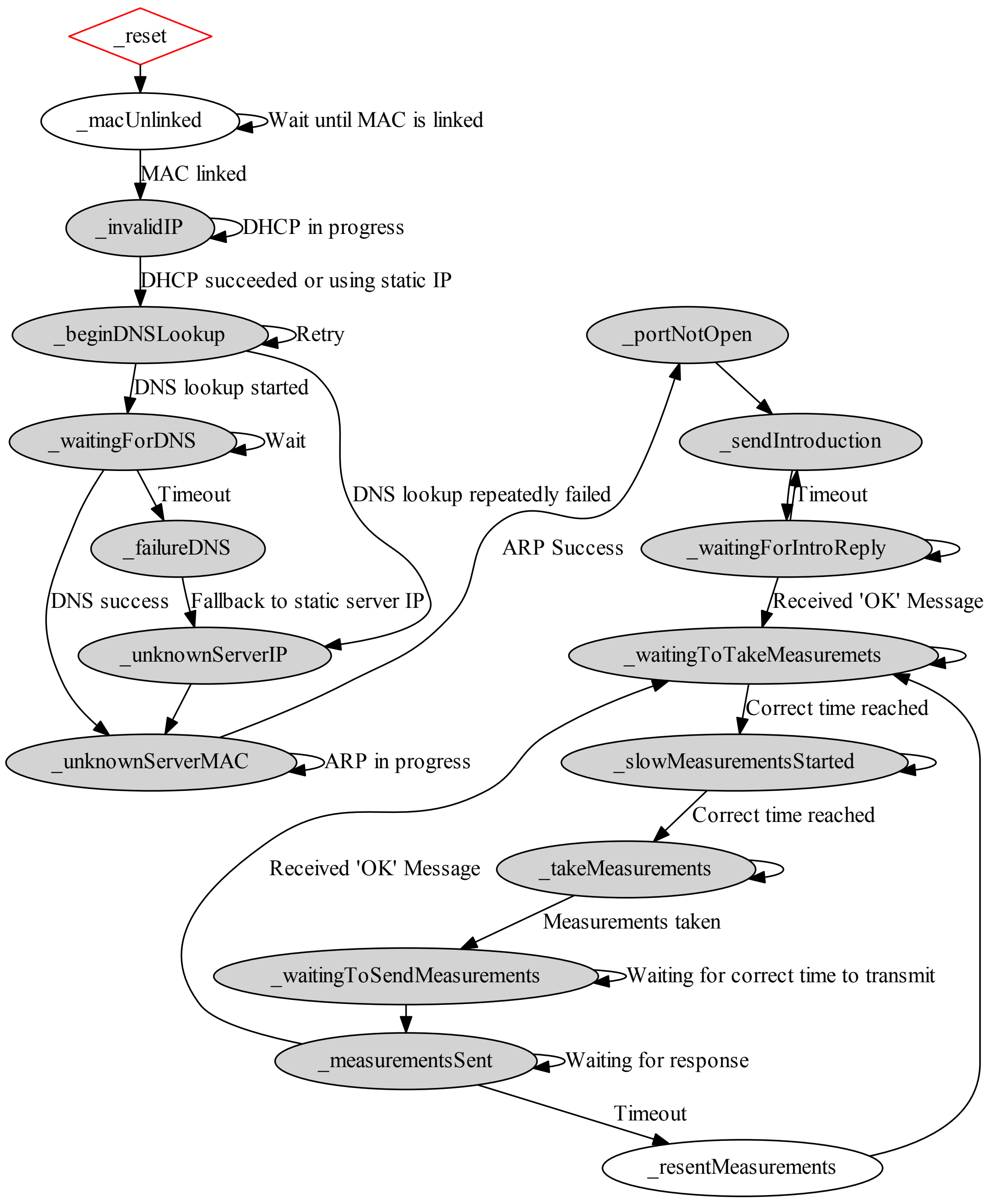


FIGURE 5.5: The finite state machine of the Sensor Node. Not shown are the transitions from all states with grey backgrounds to _reset state which occur when the Ethernet MAC layer is not connected. Also all states transition to _reset 5 minutes after the last message received from the server or last reset.

selects the preprogrammed IP address of the server to be used and then moves to the _unknownServerMAC state. If it succeeds in contacting a DNS server in the _beginDNSLookup state, it moves to the _waitingForDNS state to wait for the response to the DNS lookup request. The node waits a timeout period for a successful DNS response. Should it not arrive within that time it moves to the _failureDNS state. The _failureDNS state is a placeholder state for additional error handling. It does nothing except move the node to the _unknownServerIP state. If there is a response identifying the server's IP address the node stores the server's IP address and moves to the _unknownServerMAC state.

In the _unknownServerMAC the node repeatedly sends an ARP request and waits for the response. When it receives a response it stores the server's MAC address and moves to the _portNotOpen state. In the _portNotOpen state the node opens the local UDP port it will use to communicate with the server.

At this point the low-level initialisation is complete. The node then moves to the _sendIntroduction state to begin the application layer initilization by sending the `Introduction` message to the server using UDP. The node will then transition to the _waitingForIntroReply state to wait for the server's reply. If the timeout of 200 ticks (10 seconds) on the server's reply expires, it moves back to the _sendIntroduction state to try sending the introduction again. If the node does receive a response from the server in the _waitingForIntroReply, state then the initilization is complete and it transitions to the _waitingToTakeMeasurements state.

5.3.2 Data Collection

When in the _waitingToTakeMeasurements state several tasks are performed. The node checks for and handles any messages received from the server. Next the microphone is sampled and the audio statistical measurements (average and peak sound level) are updated with the new sample value. The current time is then checked to see if it is time to begin taking measurements. If it is, then a function call is made to start the SHT11 sampling its temperature sensor, as it takes several hundred milliseconds to sample. The node then moves to the state _slowMeasurementsStarted. If it isn't time to start taking measurements then the node remains in the _waitingToTakeMeasurements state.

The _slowMeasurementsStarted state continues to sample the microphone like the _waitingToTakeMeasurements state, but does not check for messages from the server as handling these may delay sampling of the sensors. The time is checked to see if it is appropriate to initiate the next phase of measurement collection. If it is, a function is called to record the result from the SHT11's temperature sensor and start it sampling the humidity. The node then moves to the _takeMeasurements state. If it is not the correct time the node remains in the _slowMeasurementsStarted state.

In the _takeMeasurements state the node samples the microphone and updates the audio statistics. It then checks the time to see if it is time to sample the remaining sensors. If it is, then the node obtains the humidity reading from the SHT11 sensor and samples all the other sensors. Where possible sensor sampling is done in parallel to minimize the time between samples being taken. The current time is then recorded to be used in the `Sensor Readings` message that will be sent to the server. The node then moves to the _waitingToSendMeasurements state. If it isn't time to sample the sensors then the node remains in the _takeMeasurements state.

The _waitingToSendMeasurements state exists to prevent all nodes from sending their data samples to the server at exactly the same time. In this state the node continues to sample the microphone and update the audio statistics. It also checks the time to see if it is time to send the measurements. The time to send measurements is after a fixed delay after the sampling time. The delay is different for each node as it is calculated using a 24 bit Fowler–Noll–Vo (FNV) hash [65] of the Network Interface Controller identifier part of the node's MAC address [66]. The delay ranges between 0 and approximately 12.8 seconds. If the time is correct then the node sends the sampled sensor values in a `Sensor Readings` message to the Data Receiver Server and moves to the _measurementsSent state. Otherwise it remains in the _waitingToSendMeasurements state.

In the _measurementsSent state the node samples the microphone, updates the audio statistics and checks for a response from the Data Receiver Server for the previously sent `Sensor Readings` message. If there is a message and it's an `OK` message then the node moves to the _waitingToTakeMeasurements state. If the message is a `Who Are You` message, the node sends an `Introduction` message but remains in the _measurementsSent state. If no `OK` message is received within a timeout period of 200 ticks (10 seconds), the node resends the previously sent `Sensor Readings` message and moves to the _resentMeasurements state.

The _resentMeasurements is a placeholder state where additional error handling can be added to check for a response to the re-sent `Sensor Readings` message. At present the state provides no functionality other than to transition the node to the _waitingToTakeMeasurements state.

5.3.3 Time

It is important for the sensor node to maintain an accurate track of time in order to properly synchronize taking measurements and desynchronize sending them. When reset, the node sets the epoch value to zero to indicate it doesn't know what the correct epoch is. The node uses the 16-bit "Timer1" to generate high priority interrupts every 50ms to increment the tick count. During these interrupts a software watchdog timer is decremented. This timeout is used to reset the node should it reach zero. Its initial value is 6000 which when decremented every 50 milliseconds gives 5 minutes. The value is reset to 6000 every time the node receives a message from the server.

When the node receives a reply from the server, the timestamp included in the message is used to update the node's time. If the node's epoch is different to the epoch in the message from the server, the node assumes that its time is invalid and changes its time to match the timestamp in the message from the server. This guarantees that the nodes time will be set to an incorrect value as the timestamp in the message from the server is of a time in the past. This is not a significant problem because the time difference between that timestamp and the true time is only the amount of time the message took to be sent, travel across the network and be received. If that time is less than 1 tick (50 milliseconds) then the clock will be changed to the correct value. If it's greater then the clock will be corrected when the node next communicates with the server. If the epoch in the message from the server matches the node's epoch, then the number of ticks is updated to the time the server said it sent the message plus half the round trip time (the time between sending the message to the server and receiving the reply).

If the round trip time was over 2000 ticks (100 seconds), it is assumed something went wrong and that the node's time is invalid. The node's time is then changed to be the same as the time the server said it sent the message. While this guarantees the node's current time will be behind the server's current time, it also guarantees that the previous inaccurate state will be significantly improved, and that the node's time should be properly synchronized by the next communication with the server.

5.4 Weather Data

Weather data is sent to the central server from the Whitworth meteorological station in a UDP packet every 30 seconds. Software running on the central server called 'WeatherReceiver' receives the weather reading UDP packets, parses them and stores them in the PostgreSQL database. Any errors the program encounters are written to a log file and also sent in UDP packets to a port on the loopback address which is listened to by another program called ‘WeatherMonitor’. The WeatherReceiver also sends an ‘OK’ message to the WeatherMonitor program every time it successfully stores received weather readings in the database. The WeatherMonitor program is described in Section 5.5.3.

5.5 Status Assurance

Since the system is required to collect data for an indefinite period greater than a year it is important that steps be taken to maintain its operation. The philosophy used to maintain system uptime is that failures shouldn’t be silent — that any unusual events be reported to the system administrator so that they can be investigated or remedied as needed. In order to do this a number of programs were written to monitor the state of the programs collecting data. The job of these programs is to perform automated recovery in the event of errors, and to alert the system administrator of any problems or unusual events. The reason for separate programs monitoring the state of the data collection programs is that if a fatal error occurred in the data collection program then that program is no longer running and thus can’t send the error message or start a new instance of itself.

5.5.1 SendAlert

The ‘SendAlert’ program is a simple command-line program that sends an email to the system administrator. When invoked, the command-line parameters are used for the contents of the email. If the first parameter begins with “-StartService” then the program also attempts to start a service running whose name is specified in the first parameter after a space after the “-StartService” flag.

The SendAlert program is invoked by the OS in the event that any of the service programs written for this project fail. It is used to automatically re-start the failed service and to inform system administrators that the service failed. If the DRSMonitor and the WeatherMonitor guard the data collection programs, then SendAlert is the program which guards the guards. Trust is placed with the OS to invoke SendAlert when needed. There are no higher levels of meta-monitoring programs since if the OS can’t be trusted, then any additional programs running on that OS can’t either.

5.5.2 DRSMonitor

The DRSMonitor program is a program which monitors the status of the DataReceiverServer program. It is run automatically by the OS as a service. The DRSMonitor checks how many instances of the DataReceiverServer program are running and starts or stops them as necessary, to ensure that there is always exactly one instance running. It logs to disk any errors or important events such as starting a DRS instance. It also sends notifications of these events by email to the system administrator.

5.5.3 WeatherMonitor

The WeatherMonitor program serves to monitor the status of the WeatherReceiver program and to alert the system administrator about any problems. It is started automatically by the OS as a service. If the WeatherMonitor detects the WeatherReceiver is not running, it starts a new instance of it. Whenever the WeatherMonitor starts the WeatherReceiver, it sends an email indicating this. If the WeatherMonitor detects that there it more than one instance of the WeatherReceiver, it terminates them all and starts a new instance. If the WeatherMonitor doesn't receive any 'OK' messages from the WeatherReceiver for over 5 minutes, it starts to send emails indicating that there is a problem, and for how long it has been since the last reading was received. The period between each email is doubled each time one is sent. The error emails stop being sent when an 'OK' message is received. This resets the period to wait before sending an error email, and triggers an email to be sent indicating readings are being received again.

5.6 Smart Meters

The university has a system of 'smart meters' which are networked fiscal energy meters. These meters record the usage of parameters such as electrical power (both active and reactive), gas, water and steam. These recorded readings are stored in a database at 30 minute intervals. A program was written which interfaces with this database each night and retrieves the previous day's readings for selected buildings and stores them in the project's PostgreSQL database.

There are several reasons for copying the data from the university's server to the project's server. The interface to the university's database does not allow complex queries to be run unlike the project's database. It reduces the load on the university's database as data is retrieved only once rather repeatedly as analysis is re-run. It means that only one type of database interface is required for the analysis software. It also enables the provision of limited access to selected meters wouldn't be allowed or possible with the university's database.

5.7 Database

A database was used to store the sensor data for this project. The database chosen for use in this project is the PosgreSQL database [57]. This is a well proven open source SQL database.

5.7.1 Database Structure

Each node has a table in the database to store the data received from that node. There are four other tables which store metadata about the nodes and their database tables. This structure was chosen as it minimized duplication of data, and simplified searching for and retrieval of data programmatically.

tblNodes

`tblNodes` is the primary metadata table. It stores the list of registered nodes. There is one row for each node. The SQL statement for creating this table is shown below:

```
CREATE TABLE "tblNodes"
(
  "NodeID" numeric(20,0) NOT NULL,
  "PublicID" bigserial NOT NULL,
  "SocketAddress" text,
  "NumberOfSensors" integer,
  "LastSeenTime" timestamp with time zone,
  "IsAlive" boolean,
  PRIMARY KEY ("NodeID"),
  UNIQUE ("PublicID")
)
```

`NodeID` is the unique ID used to identify each node. It is based on the node's MAC address which is burned into flash memory on each node. The numeric data type can store numbers to arbitrary precision. It is configured here to store 20 significant figures with no fractional parts. The reason for using the numeric type is because the NodeIDs are 64 bit unsigned numbers (often with their most significant bit set to 1), whereas the largest standard integer type that PostgreSQL supports is signed 64 bit integer. While the unsigned 64 bit NodeIDs could be stored as signed values, it was decided that they should be stored in a manner that keeps their sign and apparent value correct. This was to avoid any potential bugs where a signed to unsigned conversion was missed, particularly when converting the number to a hexadecimal notation.

`PublicID` is an ID number assigned by the database. Each node gets assigned a PublicID when it gets an entry added to this table. The purpose of this ID is to allow the node sensor data tables to be distributed anonymously, while maintaining a consistent manner to refer to the tables. Internally the tables and nodes are referred to by their NodeID; but externally they can be renamed using the PublicID to Node1, Node2 etc. This enables people to consistently discuss the tables without knowing which table is from the node in which room. `SocketAddress` is the network address (IP and port number combination) that the last communication from the node was received from. It is stored as text so as to be convieniently human readable. `NumberOfSensors` is the number of sensors on the node. `LastSeenTime` is the time of the last transmission from the node. `IsAlive` indicates in a convenient manner whether the DRS timeout for the node has expired or not.

tblSensorTypes

`tblSensorTypes` stores the different types of sensors the nodes use. This table is filled out by hand prior to any nodes being installed. It is used to relate the `SensorType` numbers the nodes identify their sensors by to human readable identifiers. The SQL to create the table is shown below:

```
CREATE TABLE "tblSensorTypes"
(
  "SensorType" integer NOT NULL,
  "SensorDescription" text,
  "Units" text,
  PRIMARY KEY ("SensorType")
)
```

`SensorDescription` is a human readable identifier of the sensor, typically identifying what the sensor measures and possibly the part number of the sensor. `Units` contains a text string identifying the units the sensor readings are in such as `°C` or `kPa`.

tblSensors

`tblSensors` stores which sensors are present on which nodes. There is one row per sensor on each node. It is also a link table linking `tblNodes` to `tblSensors`. The SQL used to create it is shown below:

```
CREATE TABLE "tblSensors"
(
  "NodeID" numeric(20,0) NOT NULL,
  "SensorID" integer NOT NULL,
  "SensorType" integer,
  "DataType" integer,
  PRIMARY KEY ("NodeID", "SensorID"),
  FOREIGN KEY ("NodeID") REFERENCES "tblNodes" ("NodeID") MATCH
      SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

`NodeID` is the node identifier. It references the `NodeID` in `tblNodes`. `SensorID` is the sensor identifier. Each sensor on a node has a unique ID number; this field stores that number. The primary key is made up of both the `NodeID` and `SensorID`. This means that there will be only one row for a particular sensor on a particular node. `SensorType` stores the type of sensor. It references `SensorType` in `tblSensorTypes`. This referencing isn't enforced by the database engine using constraints. This is to allow nodes to register with new types of sensors before those sensor types have been added to `tblSensorTypes`. `DataType` stores the data type of the data from this type of sensor (signed or unsigned integer or floating point) and its size in bytes. This field is described more in Appendix A.

tblNodeLocations

`tblNodeLocations` is an extra table that is not required for system operation but was created for convenience and maintenance purposes. It stores human readable text names or descriptions of the locations of each node. This enables one to find out where a node is located in the real world from its `NodeID`. The SQL for creating this table is shown below:

```
CREATE TABLE "tblNodeLocations"
(
  "NodeID" numeric(20,0) NOT NULL,
  "LocationText" text,
  "Position_X" double precision,
  "Position_Y" double precision,
  "Position_Z" double precision,
  CONSTRAINT PRIMARY KEY ("NodeID")
)
```

`NodeID` is the node identifier. It references the `NodeID` in `tblNodes`. `LocationText` is a text string naming or describing the location of the node. It is typically a room number or name. `Position_[X|Y|Z]` are intended to store the nodes position in the building as 3D coordinates. They were not used in practise.

The `NodeID` is intended to match the `NodeID` in `tblNodes`. There is no constraint to ensure that the node described exists in the system. Such a constraint would avoid entering invalid NodeIDs; but would prevent assigning a location for a node in the database before the node has been activated in the real world and performed its first registration. The

Node sensor data tables

Each node has its own table for its sensor data. The tables are named `tblNode[NodeID]` where `[NodeID]` is the node's NodeID expressed in hexadecimal as an uppercase string of 16 characters. The tables for each node can contain different fields as nodes may have different numbers and types of sensors. The only field common to all tables of this type is the `TimeStamp` field. This field is the primary key as there is one row per data sample, where a data sample from a node contains the data from all the sensors on that node at a particular time. The remainder of the fields in the table are named `Sensor[SensorID]` where `[SensorID]` is the ID of a particular sensor on that node. The sensor fields have a data type appropriate to the sensor they store data for. SQL to create a node sensor data table would resemble the SQL below:

```
CREATE TABLE "tblNodeC47345FEFFA30400"
(
  "TimeStamp" timestamp with time zone NOT NULL,
  "Sensor0" real,
  "Sensor1" real,
  "Sensor2" smallint,
  "Sensor3" integer,
...
  "Sensor15" smallint,
  "Sensor16" smallint,
  PRIMARY KEY ("TimeStamp")
)
```

5.8 Publication of Data

The project's server is configured to run a script every Sunday. The script saves a copy of all the Ethernet node sensor data to a file. The file is processed to replace the node MAC addresses with anonymizing numerical identifiers and compressed. The anonymizing numbers used are the `PublicID` numbers assigned to each node described in Section 5.7.1. The compressed file is copied to a folder which is publicly accessible via URLs listed in Appendix C. The smart meter data and the weather data is not made available, as the rights to distribute those datasets are owned by other parties.

5.9 Conclusions

The system software has functioned as intended for over a one year period, and continues to do so. The only interruptions to operation have been due to external factors. The system automatically recovers once the external factors have been resolved, and correctly sends notifications of the start and end of interruptions as designed. The conclusion drawn from this track record is that the system software is fit for purpose and works well.

Chapter 6

Analytical Methods Used

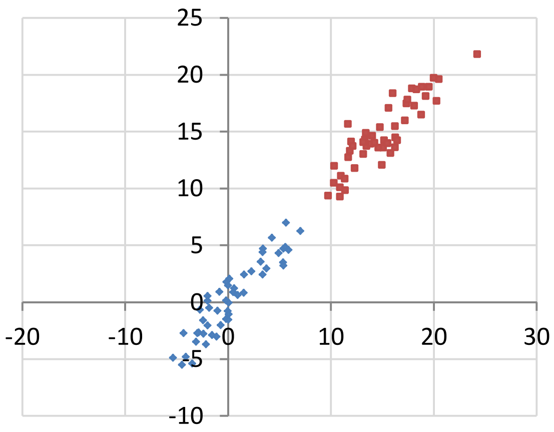
This chapter describes a number of techniques used in the analysis of the data collected from the nodes. The use of these techniques to perform the analysis of the data collected from the nodes is described in Chapter 7. This chapter describes what these techniques do and how they work. The result of these techniques on some example data is presented to illustrate and aid understanding of their effects.

6.1 Example Dataset

To aid the description of some of the methods that are described in this chapter, two example datasets have been produced. Dataset 1 consists of two groups of 50 points randomly generated from Gaussian distributions with standard deviations of 1 and 5. They have been rotated by 45 degrees anticlockwise. One group has been translated by 15 units in both positive x and y directions. Dataset 2 consists of two groups of 50 points randomly generated from Gaussian distributions with standard deviations of 1 and 10. They have been rotated by 45 degrees clockwise. One group has been translated by 5 units in both negative x and y directions and the other by 5 units in the positive x and y directions. These have been plotted in Figure 6.1.

6.2 Principle Component Analysis

Principal Component Analysis (PCA) is a method for creating a coordinate transform matrix W which transforms data from data space to PCA space. The PCA space will have as many axes as there were parameters in data space, and all the axes are orthogonal to each other. The matrix transforms data so that it is aligned to the axes of PCA


(A) Dataset 1

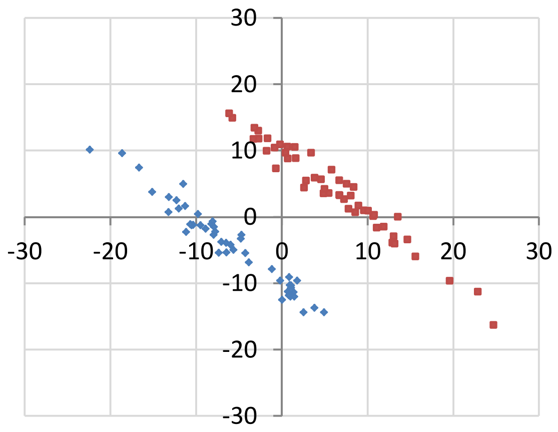
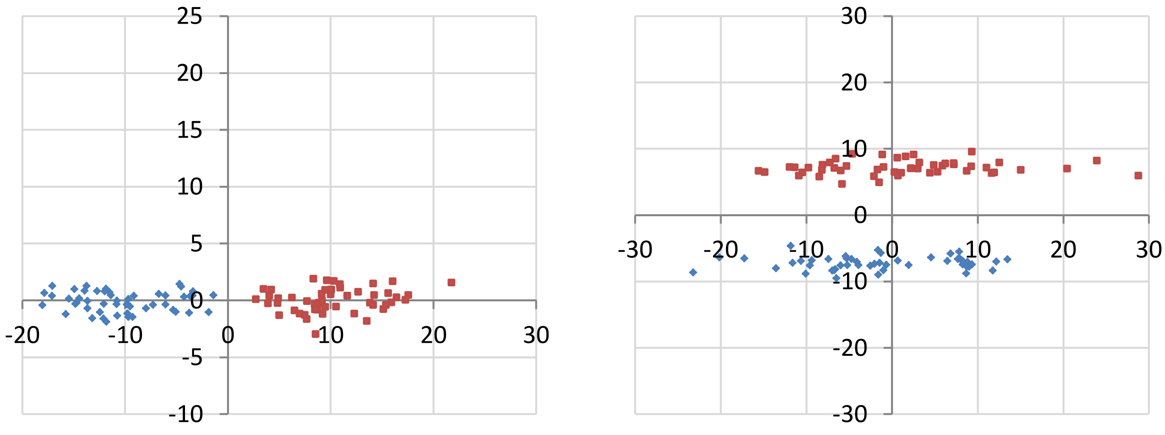

(B) Dataset 2

FIGURE 6.1: Example Datasets

space according to its variance. The transformed data is aligned so that the direction of maximum variation runs along along the first PCA axis. For each subsequent axis, the data is transformed (without changing the alignment of previous axes) such that the direction of maximum variation (which is also orthogonal to all the previously found directions of maximum variation) is aligned to the axis being considered. This means that in PCA space the first axis is the direction of most variation, the second axis the direction of second most variation, all the way through to the last axis which is the direction of least variation. The transformed data values are called scores and are stored in a matrix T. The scores are calculated by multiplying the input data in matrix X by the W matrix so $T = XW$. When applied to physical systems the columns of the X matrix represent different measured parameters. The rows represent sample values taken at a common instant of time. In abstract systems the rows represent coordinates in an N dimensional space, and the N columns represent the axes of that space.

PCA is commonly used for dimensionality reduction. This is achieved by removing some of the axes with least variation. This is because the axes with most variation contain the majority of the information describing the data, and the axes with least variation only contain minor details. Indeed an axis with no variation contains no information at all. Figure 6.2a shows the results of PCA on dataset 1. The two variables were both strongly correlated with each other in dataset 1. The result of PCA was that two correlated variables were mapped onto a single axis which describes the majority of the variation of the data, and a second axis with only minor variation. It is standard procedure with PCA to mean centre the data and normalize it. In this example the normalization step hasn't been performed to make visual comparison with the untransformed data easier. PCA makes no distinction between different classes within a dataset, as such it was applied to the example datasets as if the data from both groups were a single group. As such the colour and marker type of the points only serve to illustrate how the example data was transformed.



(A) Dataset 1 (B) Dataset 2

FIGURE 6.2: PCA Results

The PCA transform isn't just a sequence of rotations of the data (although it could be performed as such). Implementations of the PCA transform typically determine the direction of maximum variation and put appropriate numbers in the W matrix to align that direction to the PCA axis. For example a 2D PCA transform could be performed by using Least Squares fitting (discussed in Section 6.4) to find a line of best fit. One then needs to create a matrix to transform the data so that the line runs along the x-axis. Which direction along the line of best fit becomes the positive direction of the axis is subject to implementation details. One could use the slope of the line found to create a rotation matrix to rotate the data to make the line run along the x-axis. A different method of doing PCA may have the transformed data mirrored in either or both axes. If one examines the example datasets in Figure 6.1 and the transformed versions in Figure 6.2, one can see that the transform of dataset 2 could be achieved with just rotations, but the transform of dataset 1 requires a reflection.

For datasets with more than two variables one needs to find multiple directions of maximum variation which are all orthogonal to each other. With N variables one needs to find $N-1$ directions of maximum variation. This can be done by first finding the overall direction of maximum variation as in the 2D case discussed in the previous paragraph and storing the resulting transform for the first PCA axis. Then to find the transform for each subsequent PCA axis one can subtract all the already found principal components from the un-transformed input data and then find the new direction of maximum variation. The subtraction of the already found principal components eliminates the variation in the directions they are aligned. The result of this is that when the new direction of maximum variation is found it will be orthogonal to them (as directions not orthogonal to them have just had their variance reduced); and it will also be the direction with the next most variation, as the directions are found in order of size.

There are many methods of performing PCA with various advantages and disadvantages. The non-linear iterative partial least squares (NIPALS) algorithm works in a similar manner to what was described in the previous paragraph. It is used on high dimensional data when one only wants the first few principal components. This is because it calculates the principal components one at a time so only the components that are needed get calculated. The disadvantage is that with each principal component calculated numerical error accumulates due to the finite precision of computer numerical representations. This results in an increasing loss of orthogonality between the principal components. The PCA transform matrix can also be found via calculating the covariance matrix or the correlation matrix. These take more calculation to find than NIPALS but guarantee orthogonality and calculate all principal components at once. In practise singular value decomposition (SVD) is normally used instead. The specific details of these methods are beyond the scope of this chapter.

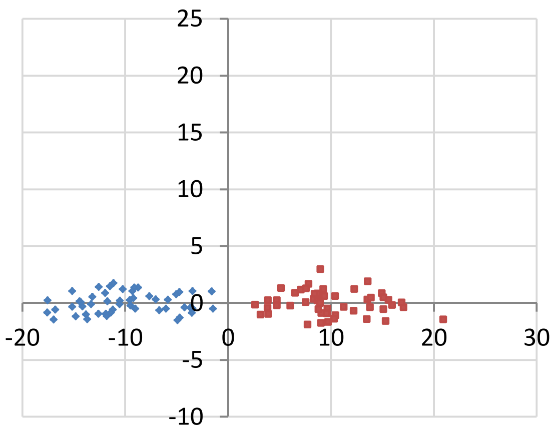
For further information and more mathematically rigorous descriptions of PCA see *Multi- and Megavariate Data Analysis* [67] and the *PLS Manual* [68]. *Multivariate Data Analysis* by Cooley and Lohnes [69] provides some example data and some transformed result values (although one value has an error).

6.3 Linear Discriminant Analysis

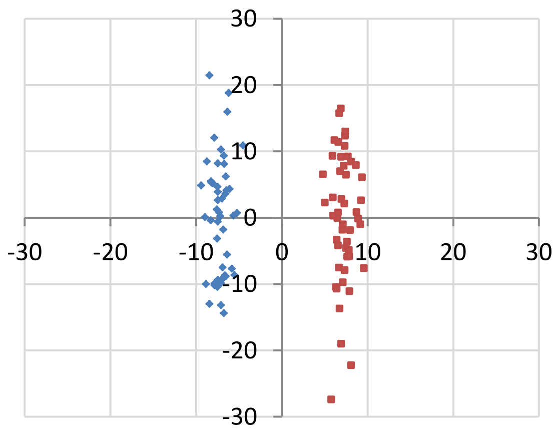
Linear Discriminant Analysis (LDA) is similar to PCA in that it is a method where you input data in a matrix X and get back a coordinate transform matrix W which maps the input data into a new coordinate space using matrix multiplication. The equation $S = XW$ describes the transformation of the input data into the LDA coordinate space, where S is the matrix of LDA scores. Unlike PCA which operates on the data X

alone, LDA requires labelled data. The labels specify which class each row of the X matrix belongs to. LDA is different to PCA in that PCA attempts to maximize the variance in each axis, but LDA attempts to maximize the variance between classes while simultaneously minimizing the variance within a class. The result of this is that along an axis the data points of different classes get separated (if possible) and the points of the same class tend to occupy the same region.

Like PCA the axes are ordered in how well they achieve the objective of the method, so the first LDA axes have the most separation between classes. This makes LDA particularly suitable for classification problems as having good separation between different classes is beneficial for many types of classifier, or it may enable the use of a simpler classifier. For example the input data in Figure 6.1 could be classified by drawing a straight line between the two classes, but in the LDA transformed data in Figure 6.3 one only needs to check if the x axis value is positive or negative to assign a class. Contrast this with the PCA results in Figure 6.2 where with dataset 1 the results of PCA and LDA are the same (apart for being mirrored in the x axis); but for dataset 2 however the PCA and LDA results have their axes swapped. This illustrates the difference as PCA maximized the variance of all the data along the x axis where as LDA maximized the separation between classes along the x axis.



(A) Dataset 1



(B) Dataset 2

FIGURE 6.3: LDA Results

6.4 Least Squares Fitting

Least Squares is a method for producing an equation whose output optimally fits some example data, were optimal means that the sum of the squared differences between the equation's output and the example data is minimized. The difference between the equation's output and the value of the data point is called the residual. It is a standard technique in regression analysis. Least Squares can be used when one knows the form of equation one wishes to fit to the data, but doesn't know the values of the coefficients of the terms of the equation. For Least Squares to work the system needs to be over determined — there need to be more data points than there are unknowns in the equation.

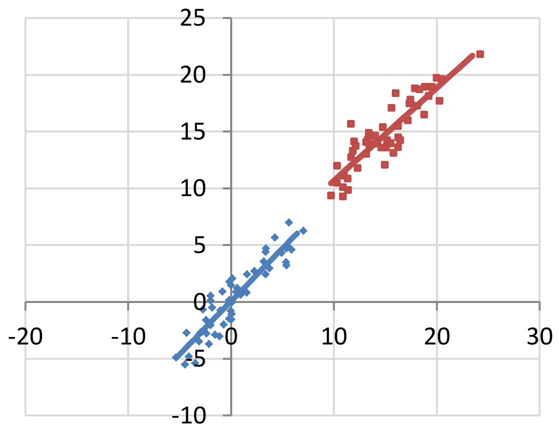
There are a number of methods for performing Least Squares. Assume that the input example values are in matrix X and the corresponding output example values are in Y. The method used in this project was to perform QR decomposition on the input data X, resulting in $QR = X$. A linear set of equations are then solved for a given set of values Y such that $Xa = Y$. The effect of this is that you get a vector a which contains the coefficients one needs to multiply the values in X by to best fit the values in Y.

As an example consider one of the groups of data from the example datasets. It contains two dimensional data. First split the coordinates of the points in two, and store the x values in a vector X and store the y values in vector Y. We will attempt to fit a straight line to the data. The equation for a line is $y = mx + c$. In that equation, we know the values of y — they are stored in vector Y — and we know the values of x — they are stored in X. This leaves two unknown constants m and c. This means that the vector a will contain values for m and c. Solving the equation $Xa = Y$ can be thought of as a way to work out the answer to the question "What values in a do I need to multiply X by to get Y".

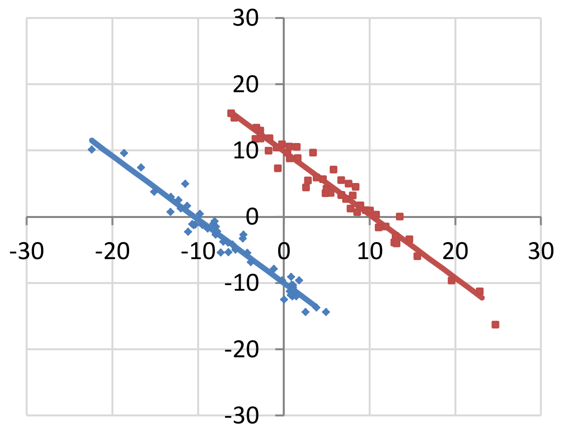
Before X and a can be multiplied they need to have appropriate dimensions. Currently X only has one column of values (the x values) and a has two values (the coefficient of x, m and the constant c). An extra column filled with 1's can be added to X. This means that when X is multiplied by a that the x values in X get multiplied by the m values in a, and the 1's in X get multiplied by c in a, and the results of this multiplication get added together to produce the output y value in Y. This implements the equation $y = mx + 1c$ which is the same as $y = mx + c$. The extra column doesn't have to be filled with 1's, it can be any non-zero value, as long as the value is the same in each row it doesn't matter. If the value was 2 instead of 1 then the calculated value of c in a would be half the size as it would be if 1 was used, as using 2 would mean you are finding constants for the equation $y = mx + 2c$.

Now that X is the correct size, the QR decomposition can be applied to it, and then using the QR decomposition in place of X the equation $Xa = Y$ can be solved using the example values stored in Y to get the values of a. Once one has the values of a one can make up the x values of X and see what the predicted y values are by multiplying X by a.

The results of Least Squares applied to the groups in the example datasets are shown in Figure 6.4. Least squares was applied to each of the groups of data points individually. It returned values which described lines of best fit. The lines were drawn on the plots by generating some artificial x values, and using the least squares results to produce y values. A line linking the points representing the artificial x values and their corresponding calculated y values was plotted on top of the original data points. It is a property of the lines of best fit produced by this method that they run along the direction of maximum variation of the data points they were generated from.



(A) Dataset 1



(B) Dataset 2

FIGURE 6.4: Least Squares Results

6.5 Leave One Out Cross Validation

It is standard practise in machine learning applications to divide a dataset and use one part to train the model, and another part to test the model's performance. This is done because one wants to produce models which represent the general principals behind the data. If the models were tested with the data used to train them, then the best performing models would be ones which simply remembered the training data — which does not require them to generalize and be able to perform well on data they haven't seen. In general the more data an algorithm has to work on, the better it performs [70, 71], indeed if there is insufficient data then it may not be possible to produce a model at all. In cases where the available data for producing a model is limited the reduction in the training set size resulting from the standard division of the data can be severely detrimental to the model's performance. If the size of the testing set is reduced to provide a bigger training set, then the accuracy and the confidence of the performance statistics is reduced.

Cross validation is a method for using all the data for both training and testing. It avoids the problem of testing the model with data it was trained on by producing multiple models with different partitions of traning and test data, and then averaging the results. It also allows a greater proportion of the data to be assigned to the training set. It avoids the problem of the weaker statistical properties of small testing sets by averaging the results of many of the small sized testing sets. There are a number of methods of performing cross validation which differ in how the dataset is split these include methods which split the data into fixed size sets in an orderly manner and ones which assign data to randomly sized sets in a random manner. This section will discuss the Leave-p-Out Cross Validation (LpO CV) technique.

Leave-p-Out cross validation is a method of cross validation where p values from the dataset are assigned to the test set, and the rest of the data is used for the training set. The production of the model using the training set, and the subsequent testing using the test set is repeated with a different combination of points for the test set until all possible test sets have been used. The average of the testing results is used as the result of the leave-p-out method. p is often a small number as it allows a larger training set, additionally the number of repetitions is equal to C_p^n where n is the total number of points in the dataset, and C is the binomial coefficient function. As p increases C_p^n increases rapidly and it soon becomes computationally infeasible to train and test the model as many times as would be required. If larger p values are desired then there are methods which select the p points at random, and are then run for as many iterations are practical or until a desired level of statistical strength in the result has been achieved.

The Leave One Out Cross Validation (LOOCV) is a special case of the LpO CV algorithm where $p = 1$. It requires as many iterations as there are data points as there is one left out (i.e. selected to be used as test data) each time. While the bias in the estimation of performance is small for all cross validation techniques, LOOCV has the least bias. The bias is towards underestimating the performance of the classifier. This is because a classifier built with only part of the available data will not perform as well as one built with all the available data. Since cross validation techniques train the classifiers on part of the data and use part for testing, the classifiers produced are weaker than a classifier built with all the data. Thus the result of cross validation is measuring the average performance of a number of classifiers which are weaker than a classifier produced with all the data. The LOOCV method reduces the performance penalty to a minimum as it only removes one data point from the training set. This minimizes the difference in performance between the classifiers produced using LOOCV and a classifier produced using all the data.

6.6 Gaussian Fitting

The Gaussian or normal distribution is a distribution that appears in many natural systems. It is often the default distribution chosen to model real systems when there is little to no prior information about the actual distribution of the system. It can be used to describe data of any number of dimensions, with each dimension being described with a mean and a standard deviation (SD). The mean and SD are used in an equation to describe a bell shaped curve called a bell curve. It is unimodal with the highest point at the mean. The SD determines the height and width, a low SD makes a tall thin curve, a high SD makes a wide low curve. The curve has a total area of exactly 1 and it never reaches a value of 0 at either side. For most purposes the curve is assumed to be approximately 0 after 3 standard deviations away from the mean.

The normal distribution is a probability density function. This means that the area under the curve gives the probability of values within that area. That is, if you wish to know the probability of a specific range of values occurring in the modelled system, one can find the area (or volume or hyper volume for more dimensions) under the curve from the lowest value in the range to the highest value. The probability of a specific value is always 0 as the area under a point is 0. This is not intuitive at first as it is common in every day life to ask questions of the form “What is the probability that 3mm of rain will fall today?”. The unintuitiveness arises from the unspoken assumed approximation of the value in the question, where as if one were to answer that question using a probability density function, the question one would be really asking is “What is the probability of exactly 3mm of rain falling, not even a plank length more or less”, to which it is more apparent that the answer is 0. A more mathematically strong argument for this is that one is asking for the probability of choosing one particular value from an uncountable infinity of possible values, and 1 out of uncountable infinity is 0.

A normal distribution can be fitted to some example one dimensional data by taking its mean and standard deviation. As is typical of statistical methods, the more data points used the closer to the true distribution the fitted model will be — particularly if the data actually does come from a normally distributed system. In higher dimensions the mathematics becomes more tricky. Assume that the data to be fitted is stored in a matrix X whose columns represent different variables, and the rows represent the samples of those variables. First the means of the columns are calculated. The means are used in calculating the covariance matrix for X. The covariance matrix is a matrix which describes how each column varies in relation to every other column. It’s calculated by calculating the sum of the products of the difference between each value in a column and the column mean (the residual), and the residual of every other column. When columns vary together — both have positive or negative differences from their respective means — then the sum of those products tends to be large, when they aren’t correlated the sum of products tends to be small. The covariance matrix is a well defined mathematical concept with a number of properties which don’t need to be discussed further here. After calculating the covariance matrix its inverse is found, and so is its determinant. At this point the necessary values and matrices needed to represent the multivariate Gaussian function have been calculated.

To calculate the height of the hyper curve at point p using the previously found values one first subtracts the previously found column means from p to get d. Then one multiplies d by the inverse of the covariance matrix, and then multiplies this result by the transpose of d. The first element of this result (the element whose index is $[0,0]$) is multiplied by -0.5. The final value is given by multiplying a factor by e raised to the power of this result. The factor is given by the inverse of $\sqrt{(2\pi)^{[nd]}[dc]}$ where $[nd]$ is the number of dimensions, and $[dc]$ is the determinate of covariance.

6.7 k-Nearest Neighbour

k-Nearest Neighbour (kNN) is a classification method used to decide a class for a data point using labelled training data. The class of the data point is determined by examining the classes of the k training data points nearest it and choosing the class which is the most common from those points. The value of k controls a tradeoff between how accurately the classifier describes the classes and sensitivity to noise in the training data. A small value of k results in a more complex border between classes that captures small scale details. The larger the value of k the smoother the border between classes is, the less detail is captured, and the less susceptible to noise the classifier is.

To choose a value for k one typically produces a range of classifiers and measuring their performance on test data. Typically one starts from a small value of k and works upwards. One then chooses the value which performed best. A k value of 1 simply classifies a point as being of the same class as the closest point in the training dataset. The case of $k = 1$ is known as the nearest neighbour classifier. For the kNN the value of 3 is often chosen as the lowest value of k. If there are only two classes then k is chosen to be odd to prevent draws. There is no standard method of resolving draws. As such an implementation may report that a draw occurred, or use some custom method of deciding the draw such as reducing the value of k until there is no longer a draw. Domain knowledge may be used for deciding draws and may lead to decision rules such as preferring the class with the fewest examples over the more frequently occurring class.

Chapter 7

Data Analysis

This chapter describes the analysis performed on the data collected by the sensor nodes. There were three main stages to the analysis. The first stage was to get a rough conception of the nature of the data such as how it was distributed and how it varied with time. Next how the data related to thermal comfort was examined. Finally a number of classification methods were investigated to produce classifiers to map measured data to comfort classes.

7.1 Overview of Methods Used

This section describes the methods used to analyze the data in the order which they were used. For the initial analysis, parameters, which intuition suggested would be most important, were plotted on multidimensional scatter plots against time. This was done to provide a rough overview of how conditions in the rooms being monitored changed throughout the day, and how this varied between days. The comfort votes were used to divide the data, and plots were produced for the different measured parameters showing the average value for each vote class and the spread. This was to see if the recorded values for the vote classes matched initial expectations, and if any parameters could simply be thresholded to allow classification. The plots showed that comfort votes could not be trivially classified based on measuring one parameter, and that the relationship between measurements and comfort was more complex than initially expected. The univariate analysis methods and results are described in Section 7.2.

Since the classification problem could not be solved using univariate methods, multivariate analysis was turned to. The first multivariate method used was Principal Component Analysis (PCA) which is described in Section 6.2. It was hoped that PCA could transform the high dimensional input data into an equivalent dataset, in which the majority of the variation occurred in a small number of dimensions. Multidimensional scatter plots of the dimensions with the most variance were produced, as were plots of how much variation was represented in each of the transformed dimensions. Lists of which parameters contributed to the transformed dimensions and by how much they contributed were also produced to identify the variables which have the biggest influence on the whole dataset. These results are discussed in Section 7.3.2. A number of methods were used to attempt to classify the transformed data. These include Hotelling's T^2, Least Squares Fitting and distance functions. More details of the methods used and why they failed to work in Section 7.3.3.

With PCA not providing satisfactory results, the related method of Linear Discriminant Analysis (LDA) was used. A description of LDA can be found in Section 6.3. As with PCA the initial LDA results were plotted on multidimensional scatter plots. These showed promise as clear clusters of comfort votes were visible. These are shown in Section 7.4.2. With the method showing promising results, attempts were made to produce classifiers which could predict occupant comfort votes based on the measured variables. These classification methods and their results are discussed in Section 7.5.

7.2 Initial Visualisation

To begin analyzing the data, the first action taken was to plot the raw data. This was done to provide an overview of how the data varied over time, and to check that it was of a form fitting reasonable expectations for the parameters measured. The data for times when comfort votes were available was plotted on an interactive 3D scatter plot with the markers coloured based on the comfort votes. The assignment of colours to markers based on comfort vote would make it possible to observe any clusters of similar votes indicating regions in the parameter space indicative of comfort or discomfort. The scatter plot was displayed using software previously written by the author for the AASN4IP project. Since only three variables can be assigned axes in three dimensional space, the variables that literature indicated most significantly affect thermal comfort were assigned axes. These were temperature, humidity and light level [15]. Light level was chosen as a proxy for radiant solar thermal energy as the nodes had no sensors to measure the mean radiant temperature of their surroundings.

Figure 7.1 is a sample screenshot of the 3D plot output. The data is displayed inside a grey textured cube. The data is plotted with the mean at the centre of the cube. The walls represent a distance three standard deviations away from the mean in three axes. The variables of the plot have been assigned the four most significant principal components produced by the analysis described in Section 7.3. The most significant component is plotted on the x-axis (left-right); the next is plotted on the y-axis (down-up); the third is plotted on the z-axis (towards-away). The fourth is plotted as the data point's colour. The colour data is normalized such that 0 is the mean and the maximum absolute value of any data point is 1. The colours are assigned such that black is 0, pure red represents a value of +1, and blue represents a value of -1. The colour fades between these colours proportionately according to the data point's value. The data represents a contiguous block of samples taken at 30 second intervals over 27 days, 18 hours and 26 minutes. There are 79960 points in total. In the figure the viewpoint has been elevated and is looking down on the data at an angle.

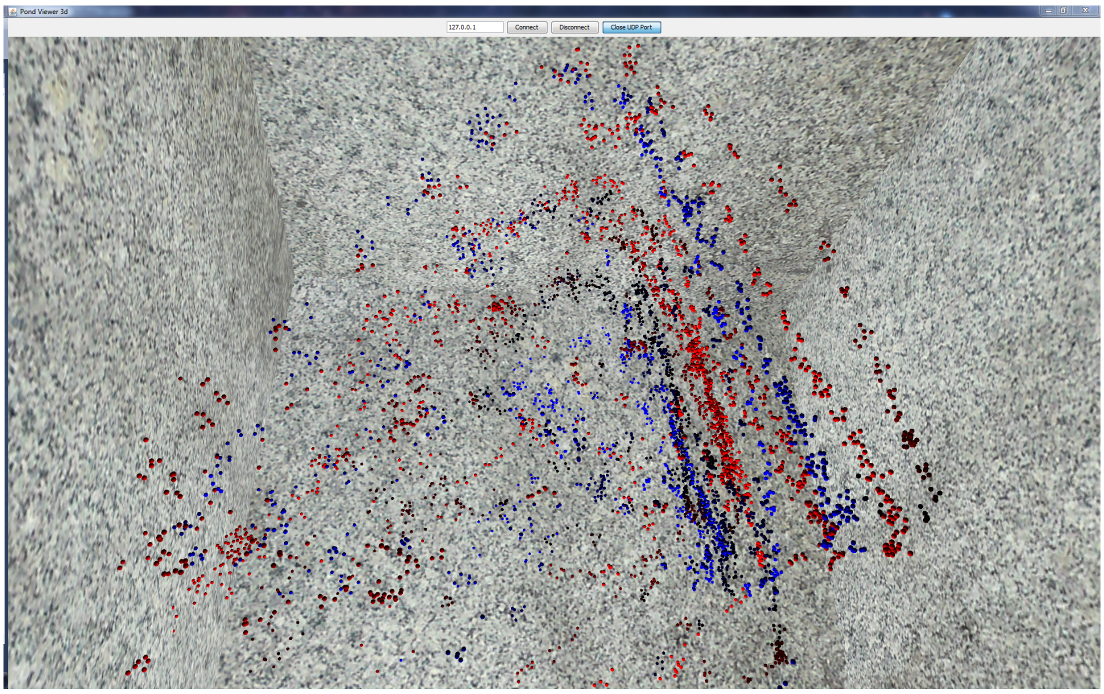


FIGURE 7.1: A screenshot of the 3dPondViewer rendering the output of the 3dScatterPlot application.

There were no clear clusters of like classes present in the scatter plots for what were believed to be the most important parameters. Given the large number of parameters it was decided to turn to multivariate analysis to achieve a dimensionality reduction. This was done with the intention to make practical the observation of regions in the parameter space which the comfort votes indicate are of a particular comfort rating.

7.3 Principal Component Analysis

Principal Component Analysis (PCA) was the first multivariate method used in this project. Principal Component Analysis is a method of generating a co-ordinate transform matrix — the W matrix. The matrix maps the N-dimensional input data into an N-dimensional space (PC space) with orthogonal axes PC_0 – $PC_{N\text{-}1}$. The data is transformed in such a way that the greatest variation is in the direction of PC_0, then the next greatest is in the direction of PC_1, and so on, until $PC_{N\text{-}1}$ which is the axis upon which there is the least variation. The values of the transformed input data in PC space are called 'scores'. The values in the transformation matrix are called 'weights' or 'loading vectors'. There are N weights (one for each output PC dimension) for each of the N input parameters. The mathematics of PCA are described in Chapter 6.2.

PCA can be used to achieve a dimensionality reduction as in many cases most of the variation in the data is described by the first few PC space dimensions. The reason for this is that correlated input variables are mapped to the same dimension in PC space. This means that one can throw away the higher PC dimensions with minimal information loss. PCA can also tell you which parameters are contributing the most variance to the data — these are typically the most important ones. To do this one looks at the weights which map the input data to for the first few PC dimensions. The input parameters with the highest magnitude weights are those which contribute the most to the transformed axes which represent the most variation, and thus are the parameters responsible for most of the variation in the dataset.

7.3.1 Use in Project

Much of the PCA analysis was performed on a combined dataset of all the sensors of all the sensor nodes, the data from the meterological station and the data from the building's fiscal smart meters measuring utility usage. This combined dataset had a total of 256 parameters.

There were several reasons for choosing Principal Component Analysis. The high dimensionality of the data made gaining an understanding of how the comfort votes related to the data via visual inspection difficult. Furthermore the initial analysis of the data showed that the data showed that univariate analysis would be ineffective for the problem at hand, so using multivariate methods such as PCA was essential.

PCA was used to gain a better understanding of the dataset. It was also investigated to be used as a step in classification of the data. The first use of PCA as with the initial analysis, was to simply plot the results. This was to see if anything of value could be observed from inspection of the processed data.

The most significant principal component weights were analyzed to find out what contributed the most to the variation in the dataset. This produced a list of parameters which may be more important than others. It also indicated that the data from the smart meters did not contribute significantly to the variation observed.

PCA was used as a pre-processing method for building classifiers to attempt to classify the environment in rooms as too hot, ok or too cold based on sensor measurements. This work is described in Section 7.3.3.

7.3.2 PCA Results

Initially the first two principal components weer plotted on a scatter plot. However no obvious features could be discerned. This was partly due to the great number of data points, but also there was nothing in the transformed PC space one could use as a reference to be able to relate to what was plotted in it. The next step was to colour the points of the plot based on their timestamps. This is shown in Figure 7.2. From this plot it was clear that the time of day heavily influenced the two most significant principal components. Given the large number of light sensors used by the project, the heating effect of the sun and the fact that a university workday is somewhat synchronized with the day-night cycle this should have been expected. The most significant principal component spends most of the day to the right of the graph, but travels to the left and back during morning and midday. The second principal component spends most of the day decreasing slowly between mid afternoon, through the night until mid morning, before increasing when the first travels to the left and back.

As the first two principal components described changes with the frequency of one cycle per day additional components were plotted. The result of plotting the first 3 principal components along the positional axes of 3D space, and using the fourth principal component to determine the plotted point's colour is shown in Figure 7.1. The third and fourth principal components changed much more slowly than the first two. Instead of being linked to the day-night cycle, cycling through a similar pattern each day, they'd change slowly over the course of several days. It was assumed that parameters indicative of thermal comfort would have a greater rate of change than this. Plotting data points coloured by their comfort votes in 3D space according to the first 3 principal components, revealed no apparent clusters of comfort votes.

As the initial plotting didn't show what was hoped, the contents of the principal

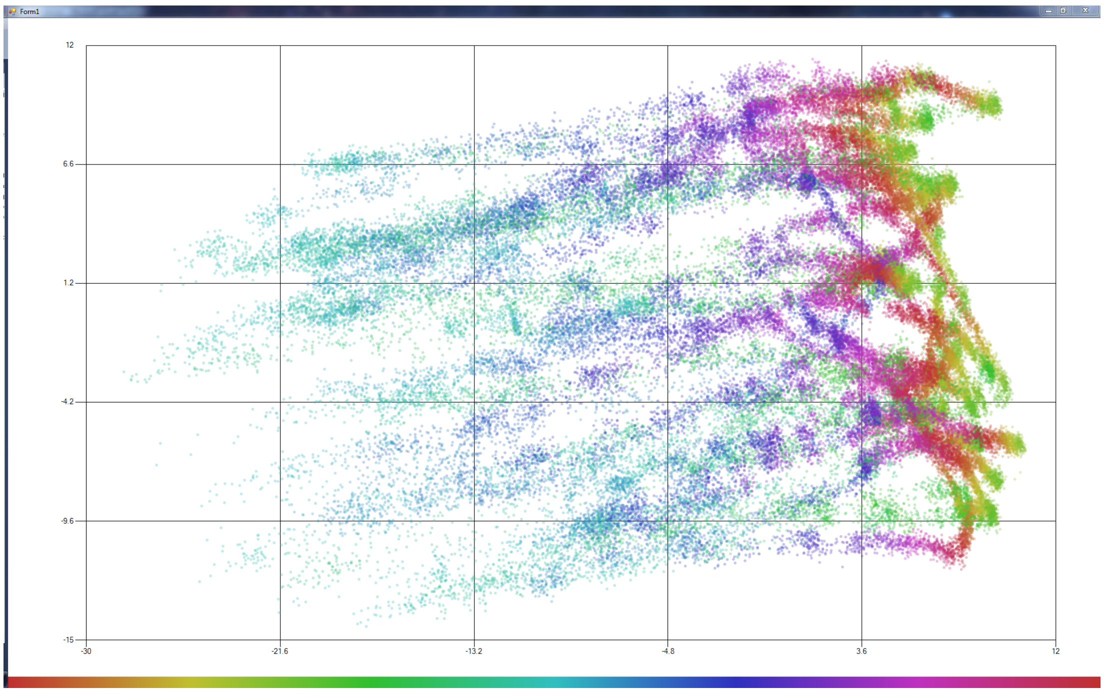


FIGURE 7.2: Plot of the two most significant principal components coloured by timestamp. PC_0 is plotted against the x-axis and PC_1 is plotted against the Y-axis. The colour strip along the bottom depicts the colours used for the time of day. The leftmost colour is used for points at midnight, the middle for midday, and the rightmost for the instant before midnight.

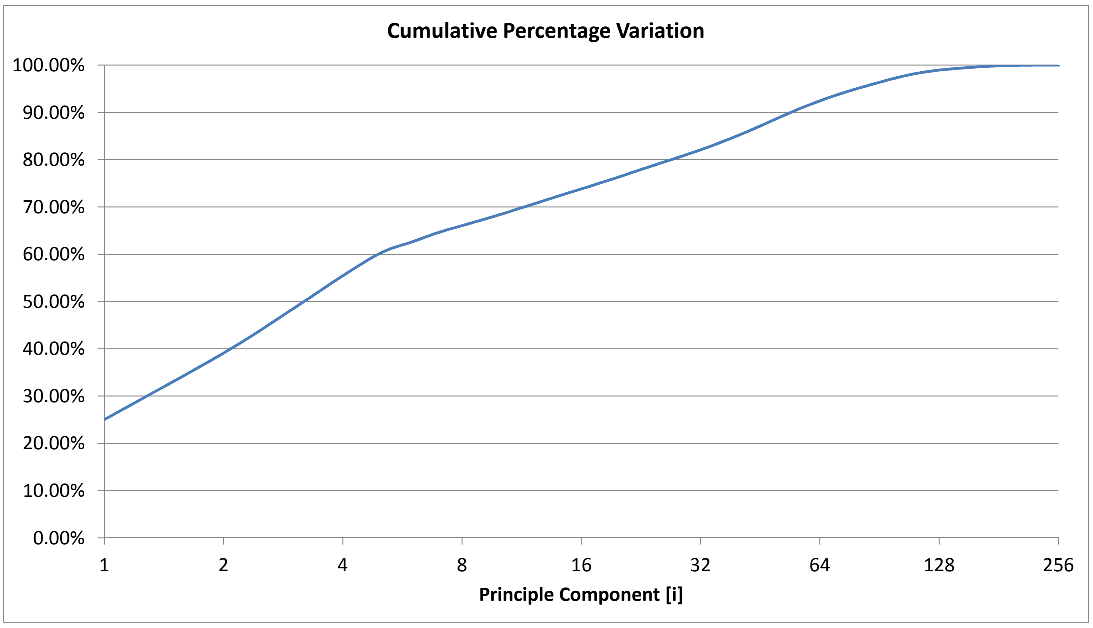


FIGURE 7.3: Plot of the cumulative percentage of variation represented by each principal component

components were examined. It was expected that the majority of the variation would be described by a small number (less than 10) of principal components. It turned out that 64 of 256 were needed to represent approximately 93% of the data, and 128 were needed to represent 98%. As such PCA did not yield the desired dimensionality reduction. Figure 7.3 shows the cumulative percentage variation represented by the principal components.

The parameters with the largest weights, which combined contributed seven eighths of the total weighting of each principle component, were examined. This was to gain a better understanding of what parameters the principal components represented, and what high or low values of a component meant in the real world. This helped the understanding of what some of the principal components represented, but didn't help with determining if any of the principal components were useful for classification.

7.3.3 Classification

An initial assessment of the viability of producing classifiers from PCA processed data was carried out. Two methods of processing the PCA transformed data to allow for classification via thresholding were investigated. Classifiers were not produced with them, as inspection of the transformed data showed that separation between classes had not been achieved. A classifier was produced using Least Squares on the PCA transformed data, however its performance proved to be unsatisfactory.

Method

Both the distance method and the T^2 method had the same preparation steps. The same analysis method was performed for each node. First, for each of the three comfort classes [Hot, Ok, Cold], the database was queried for times when only the comfort class being queried for had votes cast. Next, for each time for each comfort vote class previously retrieved, the database was queried for data for the chosen measurement parameters, to produce, for each vote class, a set of groups of data points clustered temporally. Partial classifiers for each vote class were created – each partial classifier would classify data shown to it as being of the class it was trained on or not. Each partial classifier was trained on all the data available for that class and tested against the data for all classes.

The 'partial classifiers' are not complete classifiers. Whereas a complete classifier would perform processing on the PCA transformed data to produce a scalar result, and then, depending if the result was above or below a threshold, classify the result as being of a class or not; the partial classifiers performed all the same processing except they did not perform the final thresholding to determine the class — as such they don't actually classify data. The reason for not building complete classifiers was because the thing being investigated was whether the scalar result produced by the processing was suitable for thresholding. Should it be found suitable, then a complete classifier could be produced using the given processing method.

The data was not split into training, testing and validation sets, since the objective was to assess if the classification method being tested had the potential to work at all – should it be found to do so, then more rigorous assessment would be carried out.

Hot, Ok and Cold partial classifiers were created for each individual node with sufficient comfort votes. The data used to train the classifier was the particular node's sensors and the weather station data. The smart meter data was not used as previous analysis had suggested it did not contribute significantly to the variation in the data. It was planned to investigate supplementing a single node's data with data from additional nodes to potentially improve results, if classification using a single node's data demonstrated potential.

Distance Method

With the training data for each class selected and prepared with PCA as described in Section 7.3.3, the 'Distance Method' of processing prepared the data for classification for each class's classifier as follows. First, the data points to be classified are transformed into PCA space by using the previously calculated PCA transformation for the particular class the classifier classifies the data as being in or not. The classifier calculates a scalar value for each data point to be classified in the following manner: each principal component score for that data point is first squared; each squared value is then normalized by dividing it by the standard deviation of the transformed training data for the principal component in question; the final value for the data point is the sum of its normalized squared values.

The principle of operation of this classifier is that data similar to that which it was trained on, would be close to the origin of the PC space and thus have a low sum of normalized squares, whereas data dissimilar to what it was trained on would be further away from the origin and would have a higher value. So, to classify some data as being of the same class as the classifier was trained on, one would simply need to threshold the calculated value.

This was not the case as the spread of values for test data from the other classes had significant overlap with the spread of values for the test data of the same class that the classifier was also trained on. The results of the T^2 classifier shown in Figure 7.4 are similar enough to be illustrative of the results of this classifier. The data points from the same class the classifier was trained on did have the lowest values as expected, but they also have a large proportion of higher values which overlap significantly with the results from the other classes. The lowest value results from the classes the classifier was trained on, are much lower than was expected. It was expected that they would be significantly higher than the majority of the results from the class the classifier was trained on, whereas they were only slightly higher than the lowest results.

T^2 Method

The T^2 based classifier is very similar to the distance method. The singular difference is that, prior to squaring the PC score for the data being classified, it has the mean value of the PC scores of the training data for that particular component subtracted. Thus the distance is measured from the mean value of the training data rather than the origin of the PC space.

Like the distance method, the T^2 method classifiers also failed to produce the separation between the scores of the different classes required for classification. An example of the results of the T^2 method are shown in Figure 7.4. In the plot the classifier's output value for a particular point determines its y-axis position. The left three vertical groups of data points are the results of the classifier trained on the 'Hot' vote data points. The middle group of three sets of data points are the results given by a classifier trained on 'OK' labelled samples. The three right most groups of points are the results from a classifier trained only on 'Cold' voted samples. For visibility the results of each class of data have been separated in the x-axis direction; the results within a class have had their x-axis position varied to reduce the effect of data points hiding others by being plotted on top of them. The points are coloured according to what vote class they originally came from.

Least Squares Classifier

Least Squares is a mathematical method of producing function which describes an optimal line of best fit for a dataset. The mathematics of Least Squares fitting are described in Section 6.4. Unlike the Difference and T^2 methods a complete classifier was produced using the Least Squares method. As with the other methods, each class

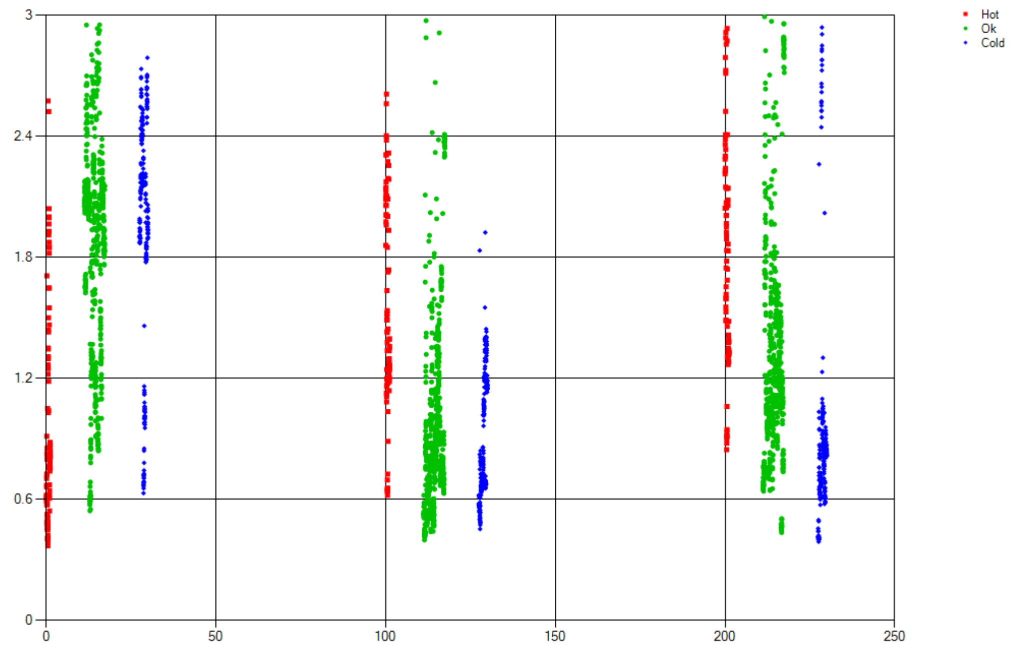


FIGURE 7.4: A plot showing the scores of the test data of each comfort vote class for each T^2 classifier trained. The axes are described in the text referencing this figure.

had a classifier produced which would classify a data point as being of that class or not. The classifiers produced were tested using Leave One Out Cross Validation (LOOCV). In the LOOCV method a classifier is trained on all the data except for one piece of data which is used to test it. The training and testing is repeated using a different piece of data for testing each time until all data has been used for testing. When the testing results are all combined they are then representative of the performance of the classifier. Classifiers were produced for each node. Classifiers for each node were produced using just the data from the node alone, and also from the node combined with the weather data. Classifiers were only produced for nodes which had sufficient data to produce a classifier.

Given a matrix X containing rows of data, and a column vector Y containing scalar values; Least Squares produces a vector F that describes a linear line of best fit mapping the rows of X to their corresponding Y value. When X is multiplied by F it produces values Y', which are the values approximating Y for the given X values according to the line of best fit. For each node the database was queried for data for times when any of that node's comfort vote buttons were pressed. The database was then queried for the node's sensor readings for those times (and the weather readings too if they were being used). This data was stored in the X matrix for that node. The database was also queried for the comfort vote counts for the node at those times; the Hot vote counts were stored in a vector Y_h; the Ok vote counts in Y_o; and the Cold counts in Y_c. Note here that X contains a row of data for any time interval when at least one button was pressed at least once, and the Y_h, Y_o, Y_c vectors contain the number of times their respective button was pressed in that time interval — which may be zero if another button was pressed, or more than one if the button was pressed multiple times. Least Squares was then applied to the Y_h, Y_o, Y_c vectors to produce F_h, F_o, F_c vectors. The result of this is that, each classifier was trained on data which had Y values greater or equal to 1 for X data labelled as being from the class of the classifier, and 0 for data which isn't from the classifier's class.

The principle behind the classifier's operation was that the Least Squares process would fit a line which had a larger (closer to 1 or above) value in the region of space where the votes for the classifier's class were located, and a smaller (closer to 0 or below) value away from that region [where the votes from the other classes were]. Classification would then be achieved by thresholding the value of the line for a given data sample, if the value is above the threshold then the data is classified as the same class as the classifier, otherwise it isn't. To gauge the best case performance of the classifiers, the threshold value was set to a numerical approximation of the optimal value as determined with a binary search. The binary search was used to maximize the R^2 value by varying the threshold used to classify the training data. The binary search was limited to 5 iterations for performance reasons.

The results of the classifiers trained on the raw results of a node are shown in Table 7.1; the results of classifiers trained on the node data which had been processed with PCA are shown in Table 7.2; and the results of the classifiers trained on combined, PCA processed, node and weather data are shown in Table 7.3. As can be seen from the tables none of the classifiers had high accuracy in both true negatives and true positives. Often one had a very high value while the other had a very low value. This was because many of the classifiers were simply classifying all data points as being from the most common class (or as not being from the less common classes). The classifiers may have had greater success had they been trained on data processed such that it had an equal number of data points in each class. However, at the time the analysis was performed, if the data points for training were selected such that each class had the same number of points as the class with the least, then there would have been very few data points to train the classifiers on. Unfortunately, the distribution of comfort votes across the classes can't readily be controlled to produce a more even distribution; it would require manipulating the building's heating schedule to produce various internal environments which the occupants may find objectionable.

The way the Least Squares classifier was designed to work was flawed. The classifier attempts to fit a hyper-plane to the data. This is analagous to fitting a straight line to two dimensional data. The first problem is that the hyper-plane is linear. Each classifier had high values for points of their own class and low values for points of other classes. The problem becomes aparrent if one considers the results of trying to fit a line to two dimensional data, where as one moves alone the x-axis the data points have low y values, then high values then low values again. Any line which runs through one of the outer groups of low values and through the middle group of high values, would have high y values over the region where the other group of low y valued data points are. To fit the data correctly one would need to fit a function which has low values, then high values then low values again — such as a bell curve or a quadratic function. This mode of failure is applicable to this classifier as each class's classifier was designed to represent the region of that class in the data-space as having high values, and that region would be surrounded by low values representing all other conditions not of the class of the classifier.

Examination of the failure of PCA for classification

PCA proved ineffective as a preprocessing method for classification. The main reason for this is that PCA finds orthogonal axes in order of their variation. In this dataset the greatest variation was caused by the day and night cycle. This meant that the classifiers were mainly using data which was related to the time of day. As thermal comfort isn't directly a function of time of day (it's possible to be in any comfort state at any time of the day) the PCA processed data did not contain good information on thermal comfort for the classifiers to use.

Since the classification methods tried so far had failed, it was decided that an examination of the dataset was required in order to get a better understanding of it and how its contents related to thermal comfort. To this end a program was written which plotted Gaussian curves of the measured and calculated parameters. Multiple curves were plotted on a chart. Each curve was generated by segregating the parameter data based on a number of aspects, such as comfort votes or room occupancy. The results of this program showed that, while there were some differences of the mean values of parameters grouped by comfort vote, the variance was so large that the classes would

Node	Classifier	False Positives	False Negatives	True Positives	True Negatives	Total Error	Total Correct	Error Percent	Correct Percent	False Positive Percent	False Negative Percent	True Positive Percent	True Negative Percent
E1B	Hot	16.00	13.00	15.00	17.00	29.00	32.00	47.54	52.46	26.23	21.31	24.59	27.87
	Ok	7.00	21.00	1.00	32.00	28.00	33.00	45.90	54.10	11.48	34.43	1.64	52.46
	Cold	9.00	7.00	6.00	39.00	16.00	45.00	26.23	73.77	14.75	11.48	9.84	63.93
E1D	Hot	0.00	0.00	0.00	378.00	0.00	378.00	0.00	100.00	0.00	0.00	0.00	100.00
	Ok	0.00	1.00	377.00	0.00	1.00	377.00	0.26	99.74	0.00	0.26	99.74	0.00
	Cold	0.00	1.00	0.00	377.00	1.00	377.00	0.26	99.74	0.00	0.26	0.00	99.74
E1H Barry Lennox	Hot	0.00	4.00	1.00	41.00	4.00	42.00	8.70	91.30	0.00	8.70	2.17	89.13
	Ok	7.00	9.00	21.00	9.00	16.00	30.00	34.78	65.22	15.22	19.57	45.65	19.57
	Cold	6.00	7.00	4.00	29.00	13.00	33.00	28.26	71.74	13.04	15.22	8.70	63.04
E1M Meeting Room	Hot	7.00	6.00	2.00	27.00	13.00	29.00	30.95	69.05	16.67	14.29	4.76	64.29
	Ok	11.00	10.00	16.00	5.00	21.00	21.00	50.00	50.00	26.19	23.81	38.10	11.90
	Cold	8.00	8.00	3.00	23.00	16.00	26.00	38.10	61.90	19.05	19.05	7.14	54.76
E8 Igor Labs	Hot	51.00	32.00	30.00	41.00	83.00	71.00	53.90	46.10	33.12	20.78	19.48	26.62
	Ok	51.00	7.00	93.00	3.00	58.00	96.00	37.66	62.34	33.12	4.55	60.39	1.95
	Cold	32.00	42.00	14.00	66.00	74.00	80.00	48.05	51.95	20.78	27.27	9.09	42.86
F1 East Side	Hot	5.00	9.00	1.00	91.00	14.00	92.00	13.21	86.79	4.72	8.49	0.94	85.85
	Ok	14.00	18.00	37.00	37.00	32.00	74.00	30.19	69.81	13.21	16.98	34.91	34.91
	Cold	13.00	15.00	26.00	52.00	28.00	78.00	26.42	73.58	12.26	14.15	24.53	49.06
F1A South Block	Hot	10.00	13.00	28.00	49.00	23.00	77.00	23.00	77.00	10.00	13.00	28.00	49.00
	Ok	4.00	6.00	0.00	90.00	10.00	90.00	10.00	90.00	4.00	6.00	0.00	90.00
	Cold	17.00	10.00	69.00	4.00	27.00	73.00	27.00	73.00	17.00	10.00	69.00	4.00
F1B North Block	Hot	8.00	5.00	4.00	25.00	13.00	29.00	30.95	69.05	19.05	11.90	9.52	59.52
	Ok	7.00	8.00	20.00	7.00	15.00	27.00	35.71	64.29	16.67	19.05	47.62	16.67
	Cold	5.00	5.00	8.00	24.00	10.00	32.00	23.81	76.19	11.90	11.90	19.05	57.14
Low Corridor	Hot	83.00	31.00	19.00	148.00	114.00	167.00	40.57	59.43	29.54	11.03	6.76	52.67
	Ok	56.00	21.00	181.00	23.00	77.00	204.00	27.40	72.60	19.93	7.47	64.41	8.19
	Cold	21.00	46.00	9.00	205.00	67.00	214.00	23.84	76.16	7.47	16.37	3.20	72.95

TABLE 7.1: Performance results for Least Squares based classifier on node data not processed with PCA.

Node	Classifier	False Positives	False Negatives	True Positives	True Negatives	Total Error	Total Correct	Error Percent	Correct Percent	False Positive Percent	False Negative Percent	True Positive Percent	True Negative Percent
E1B	Hot	16.00	16.00	12.00	17.00	32.00	29.00	52.46	47.54	26.23	26.23	19.67	27.87
	Ok	1.00	21.00	1.00	38.00	22.00	39.00	36.07	63.93	1.64	34.43	1.64	62.30
	Cold	3.00	10.00	3.00	45.00	13.00	48.00	21.31	78.69	4.92	16.39	4.92	73.77
E1D	Hot	0.00	0.00	0.00	378.00	0.00	378.00	0.00	100.00	0.00	0.00	0.00	100.00
	Ok	0.00	231.00	147.00	0.00	231.00	147.00	61.11	38.89	0.00	61.11	38.89	0.00
	Cold	0.00	1.00	0.00	377.00	1.00	377.00	0.26	99.74	0.00	0.26	0.00	99.74
E1H Barry Lennox	Hot	0.00	5.00	0.00	41.00	5.00	41.00	10.87	89.13	0.00	10.87	0.00	89.13
	Ok	4.00	20.00	10.00	12.00	24.00	22.00	52.17	47.83	8.70	43.48	21.74	26.09
	Cold	3.00	10.00	1.00	32.00	13.00	33.00	28.26	71.74	6.52	21.74	2.17	69.57
E1M Meeting Room	Hot	1.00	6.00	2.00	33.00	7.00	35.00	16.67	83.33	2.38	14.29	4.76	78.57
	Ok	4.00	17.00	9.00	12.00	21.00	21.00	50.00	50.00	9.52	40.48	21.43	28.57
	Cold	3.00	11.00	0.00	28.00	14.00	28.00	33.33	66.67	7.14	26.19	0.00	66.67
E8 Igor Labs	Hot	50.00	43.00	19.00	42.00	93.00	61.00	60.39	39.61	32.47	27.92	12.34	27.27
	Ok	23.00	62.00	38.00	31.00	85.00	69.00	55.19	44.81	14.94	40.26	24.68	20.13
	Cold	28.00	41.00	15.00	70.00	69.00	85.00	44.81	55.19	18.18	26.62	9.74	45.45
F1 East Side	Hot	0.00	10.00	0.00	96.00	10.00	96.00	9.43	90.57	0.00	9.43	0.00	90.57
	Ok	9.00	29.00	26.00	42.00	38.00	68.00	35.85	64.15	8.49	27.36	24.53	39.62
	Cold	19.00	13.00	28.00	46.00	32.00	74.00	30.19	69.81	17.92	12.26	26.42	43.40
F1A South Block	Hot	8.00	12.00	29.00	51.00	20.00	80.00	20.00	80.00	8.00	12.00	29.00	51.00
	Ok	1.00	6.00	0.00	93.00	7.00	93.00	7.00	93.00	1.00	6.00	0.00	93.00
	Cold	3.00	45.00	34.00	18.00	48.00	52.00	48.00	52.00	3.00	45.00	34.00	18.00
F1B North Block	Hot	7.00	9.00	0.00	26.00	16.00	26.00	38.10	61.90	16.67	21.43	0.00	61.90
	Ok	2.00	16.00	12.00	12.00	18.00	24.00	42.86	57.14	4.76	38.10	28.57	28.57
	Cold	5.00	8.00	5.00	24.00	13.00	29.00	30.95	69.05	11.90	19.05	11.90	57.14
Low Corridor	Hot	16.00	48.00	2.00	215.00	64.00	217.00	22.78	77.22	5.69	17.08	0.71	76.51
	Ok	30.00	95.00	107.00	49.00	125.00	156.00	44.48	55.52	10.68	33.81	38.08	17.44
	Cold	8.00	54.00	1.00	218.00	62.00	219.00	22.06	77.94	2.85	19.22	0.36	77.58

TABLE 7.2: Performance results for Least Squares based classifier on node data processed with PCA.

Node	Classifier	False Positives	False Negatives	True Positives	True Negatives	Total Error	Total Correct	Error Percent	Correct Percent	False Positive Percent	False Negative Percent	True Positive Percent	True Negative Percent
E1B	Hot	1.00	22.00	6.00	32.00	23.00	38.00	37.70	62.30	1.64	36.07	9.84	52.46
	Ok	5.00	19.00	3.00	34.00	24.00	37.00	39.34	60.66	8.20	31.15	4.92	55.74
	Cold	2.00	7.00	6.00	46.00	9.00	52.00	14.75	85.25	3.28	11.48	9.84	75.41
E1D	Hot	0.00	0.00	0.00	378.00	0.00	378.00	0.00	100.00	0.00	0.00	0.00	100.00
	Ok	0.00	297.00	81.00	0.00	297.00	81.00	78.57	21.43	0.00	78.57	21.43	0.00
	Cold	0.00	1.00	0.00	377.00	1.00	377.00	0.26	99.74	0.00	0.26	0.00	99.74
E1H Barry Lennox	Hot	0.00	5.00	0.00	41.00	5.00	41.00	10.87	89.13	0.00	10.87	0.00	89.13
	Ok	0.00	28.00	2.00	16.00	28.00	18.00	60.87	39.13	0.00	60.87	4.35	34.78
	Cold	2.00	11.00	0.00	33.00	13.00	33.00	28.26	71.74	4.35	23.91	0.00	71.74
E1M Meeting Room	Hot	4.00	7.00	1.00	30.00	11.00	31.00	26.19	73.81	9.52	16.67	2.38	71.43
	Ok	1.00	23.00	3.00	15.00	24.00	18.00	57.14	42.86	2.38	54.76	7.14	35.71
	Cold	3.00	10.00	1.00	28.00	13.00	29.00	30.95	69.05	7.14	23.81	2.38	66.67
E8 Igor Labs	Hot	29.00	43.00	19.00	63.00	72.00	82.00	46.75	53.25	18.83	27.92	12.34	40.91
	Ok	14.00	74.00	26.00	40.00	88.00	66.00	57.14	42.86	9.09	48.05	16.88	25.97
	Cold	19.00	37.00	19.00	79.00	56.00	98.00	36.36	63.64	12.34	24.03	12.34	51.30
F1 East Side	Hot	3.00	9.00	1.00	93.00	12.00	94.00	11.32	88.68	2.83	8.49	0.94	87.74
	Ok	3.00	30.00	25.00	48.00	33.00	73.00	31.13	68.87	2.83	28.30	23.58	45.28
	Cold	10.00	19.00	22.00	55.00	29.00	77.00	27.36	72.64	9.43	17.92	20.75	51.89
F1A South Block	Hot	2.00	21.00	20.00	57.00	23.00	77.00	23.00	77.00	2.00	21.00	20.00	57.00
	Ok	2.00	6.00	0.00	92.00	8.00	92.00	8.00	92.00	2.00	6.00	0.00	92.00
	Cold	7.00	53.00	26.00	14.00	60.00	40.00	60.00	40.00	7.00	53.00	26.00	14.00
F1B North Block	Hot	0.00	5.00	4.00	33.00	5.00	37.00	11.90	88.10	0.00	11.90	9.52	78.57
	Ok	0.00	28.00	0.00	14.00	28.00	14.00	66.67	33.33	0.00	66.67	0.00	33.33
	Cold	1.00	10.00	3.00	28.00	11.00	31.00	26.19	73.81	2.38	23.81	7.14	66.67
Low Corridor	Hot	58.00	34.00	16.00	173.00	92.00	189.00	32.74	67.26	20.64	12.10	5.69	61.57
	Ok	18.00	123.00	79.00	61.00	141.00	140.00	50.18	49.82	6.41	43.77	28.11	21.71
	Cold	20.00	51.00	4.00	206.00	71.00	210.00	25.27	74.73	7.12	18.15	1.42	73.31

TABLE 7.3: Performance results for Least Squares based classifier on node and weather data processed with PCA.

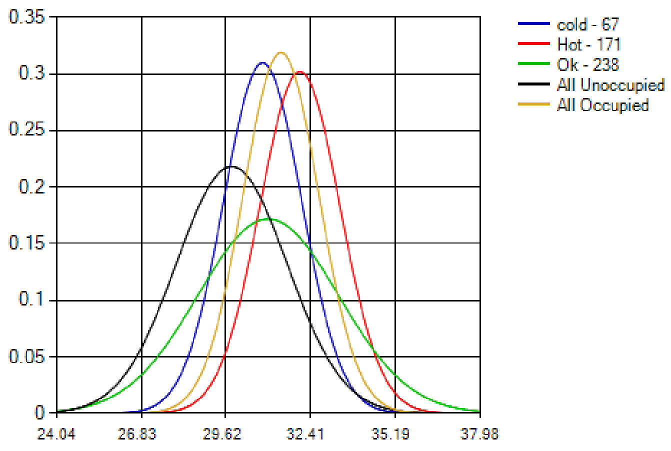
be indistinguishable from any single parameter.

The Low Corridor node is located in a corridor. As many people pass it each day (unlike nodes located in offices with a single or small number of occupants) it received the highest number of comfort vote button presses. As a result of this the comfort votes from this node are most representative of a large population. Figure 7.5a and Figure 7.5b show the Gaussian curves for the Low Corridor Ethernet Sensor Node's temperature sensors. The mean temperatures of the samples grouped by comfort votes agree with expectations — that is, cold votes had the lowest mean temperature, and hot votes had the highest — and thermal comfort research. The curves have a large overlap meaning that, knowing just the temperature, one can't say with certainty for which class an occupant would vote.

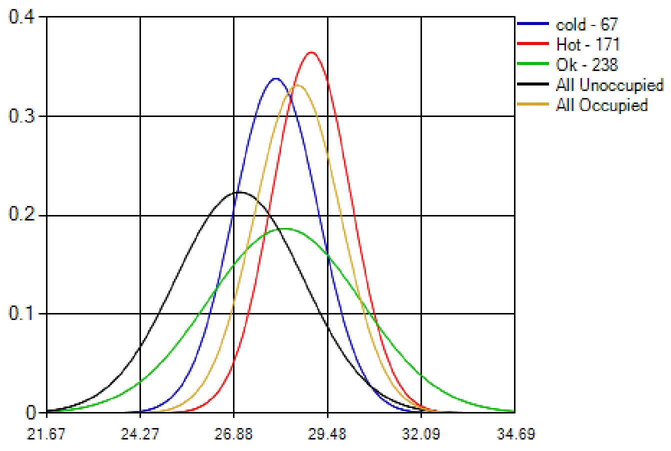
The graphs of data grouped by the comfort votes from the Low Corridor node for the indirect solar radiation (Figure 7.5c) and temperature (Figure 7.5d) from the meteorological station, and the LDR readings from some other nodes, were similar to the graphs for the node's temperature sensors. These were some of the most reassuring results in that they conformed to expectations and had some separation between the average values of the classes.

However the data from the rest of the nodes was not as promising. Many had the comfort votes for temperature in the 'wrong' order — i.e. not cold, then ok, then hot as temperature ascends — though this was attributed to some of the classes having very few data points. Insufficient votes wasn't the only problem. The E8 node is an example of a node in a small electronics laboratory with a large number of votes for each class. Comparing the results grouped by the E8 node's data (Figure 7.5e and Figure 7.5f) to the results grouped by the low corridor node (Figure 7.5b and Figure 7.5d), one can see that the results for a small population are quite different from those of a larger population. The results for the E8 node show very little difference in the mean values for both internal and external temperature.

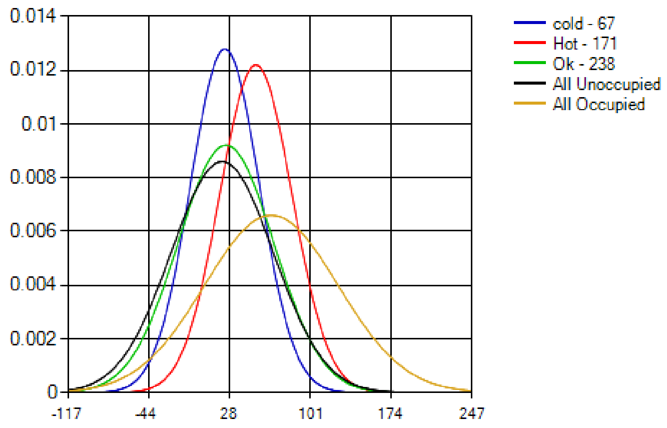
The majority of the graphs for the principal components had the curves overlapping greatly. Figure 7.6 shows the graphs for the first two principal components with the data points grouped by the Low Corridor node's comfort votes. These graphs are representative of the other principal components and of the graphs which used comfort votes from the other nodes to group the data points.



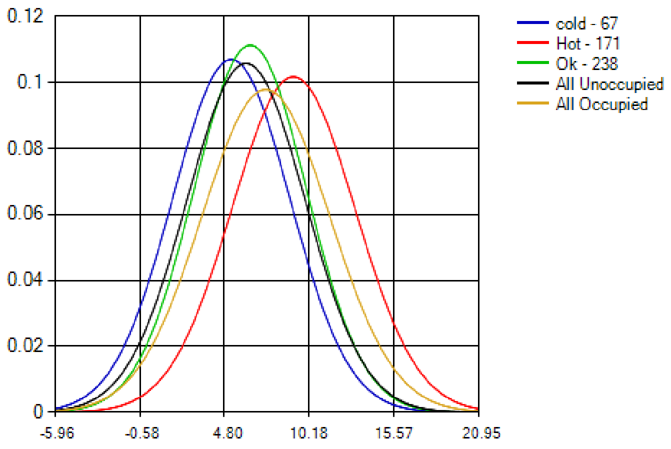
(A) Low Corridor SP100T Temperature grouped by Low Corridor node votes



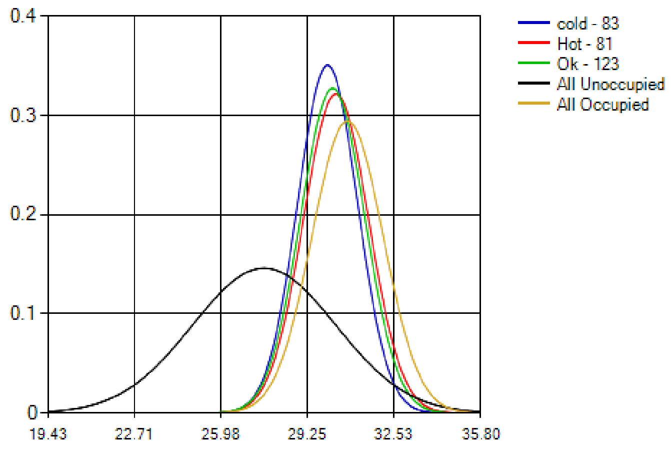
(B) Low Corridor SHT71 Temperature grouped by Low Corridor node votes



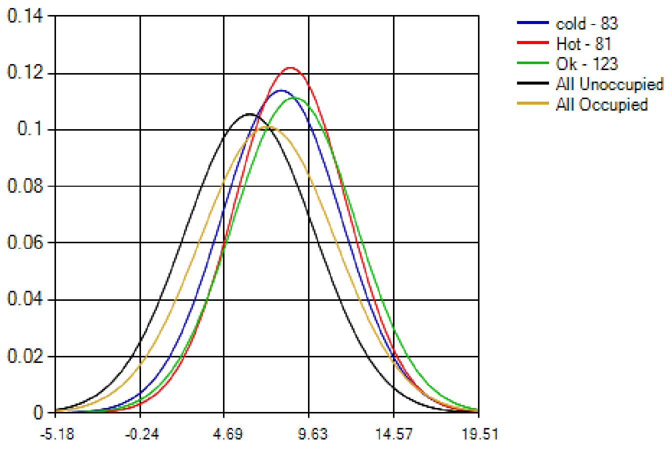
(C) Weather station indirectDownwardSolarRadiation grouped by Low Corridor node votes



(D) Weather station temperature grouped by Low Corridor node votes

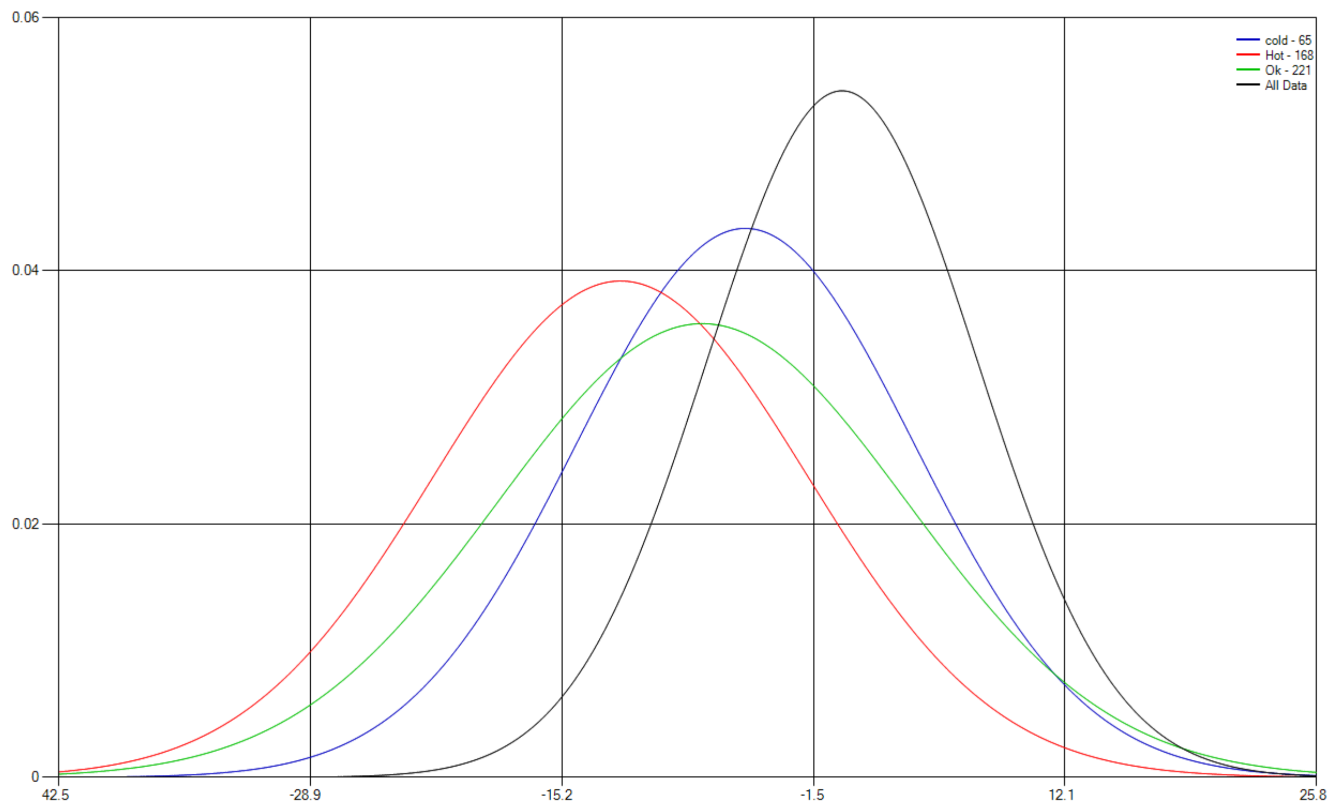


(E) E8 node SHT7X Temperature grouped by E8 node votes

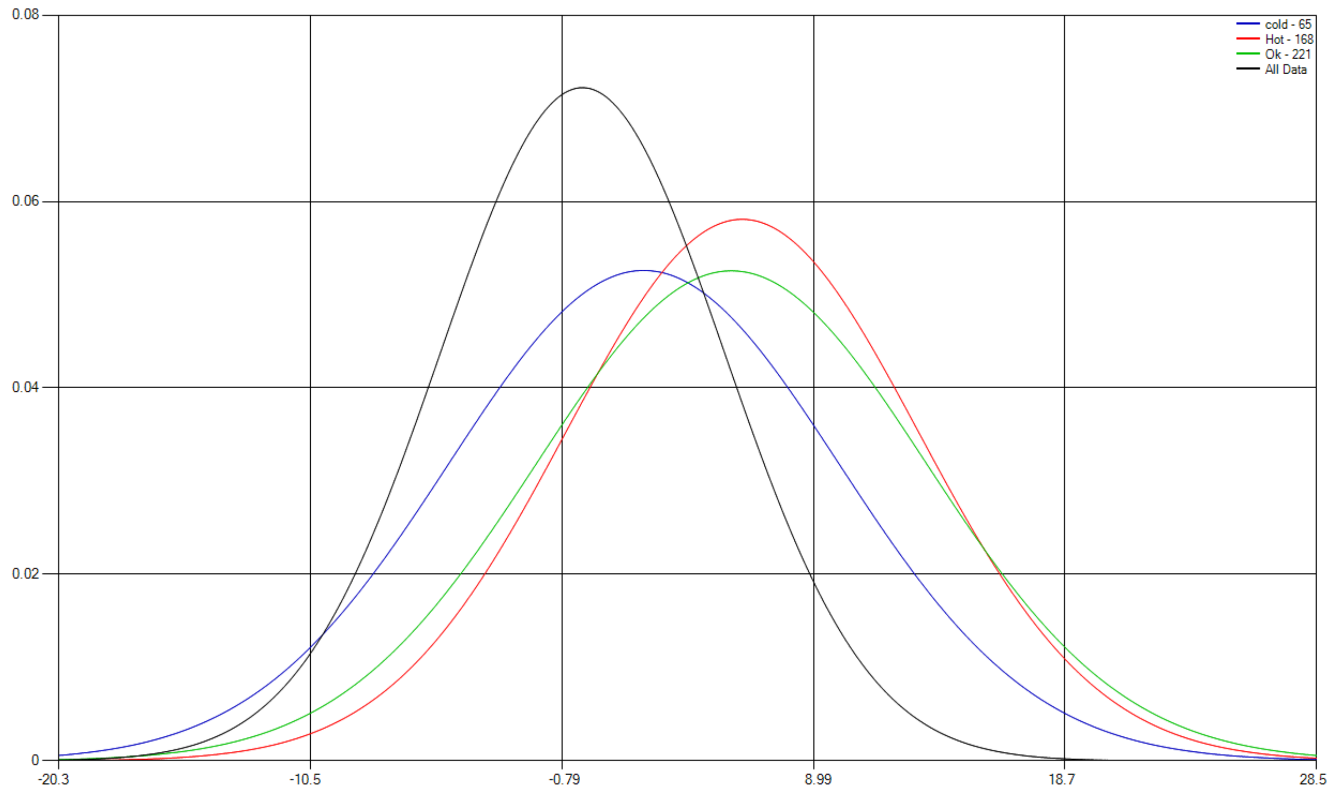


(F) Weather station temperature grouped by E8 node votes

FIGURE 7.5: Gaussian curves fitted to the samples collected by the two temperature sensors on the Low Corridor Ethernet Sensor Node. The samples are divided into groups based on the PIR occupancy sensor and the comfort vote button presses of the Low Corridor Ethernet Sensor Node (figures A–D) and the E8 Ethernet Sensor Node (figures E and F). The legends show the count of data points that the curves were fitted to. The x-axis values are the raw uncalibrated values.



(A) Principal Component 0



(B) Principal Component 1

FIGURE 7.6: Gaussian curves fitted to the two most significant principal components of all the data collected. The samples are divided into groups based on the comfort vote button presses of the Low Corridor Ethernet Sensor Node. The legends show the count of data points that the curves were fitted to. The x-axis values are the raw uncalibrated values.

7.4 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a multivariate method similar to PCA. Like PCA it analyzes a dataset and produces a transformation matrix to transform data into a new parameter space based on its variation. Unlike PCA the data it analyzes is labelled into classes prior to being processed. LDA transforms the data into a parameter space such that the variation between classes is maximized, while the variation within a class is minimized. As a result of this, it tends to separate the classes if possible. The dimensions of the transformed parameter space are ordered such that the first ones have the most variation between classes, and the last ones have the least.

7.4.1 Use in Project

LDA was used in much the same way as PCA. In some cases it was essentially used as a direct replacement for PCA. Since LDA was more successful at separating the classes than PCA, there was more analysis and classification work done using LDA for pre-processing than with PCA.

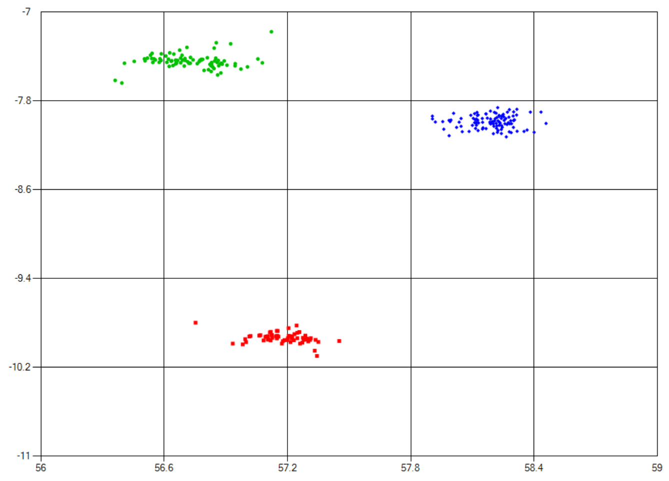
A range of classification methods were used to attempt to classify the LDA output, they include the Cosine Windowing Function classifier; the Gaussian Likeliness classifier; the Gaussian Probability classifier; and the K-Nearest Neighbour classifier. These are described in Section 7.5.

7.4.2 Results

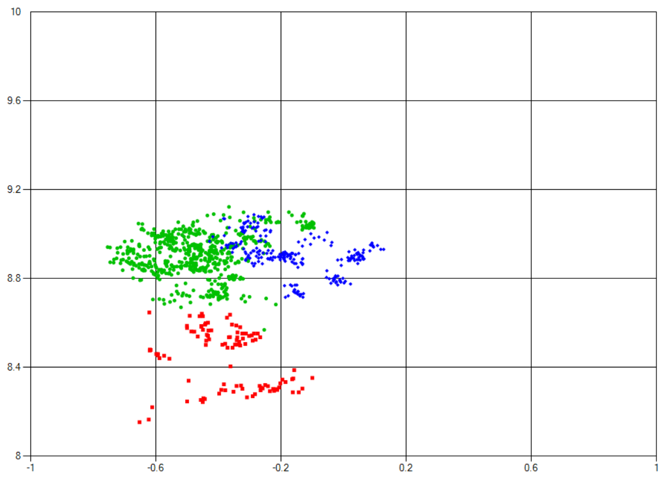
Like with PCA, the first thing done with LDA was to plot its most significant output components on a 2D scatter plot. Unlike with PCA, the LDA model is built using labelled data. As a result, instead of using all the data to build the model, only data within 5 minutes of a comfort vote button press was used to build the model. The data was labelled as being of the same class as the button press it was within 5 minutes of. If there were multiple buttons pressed in that time span then there were multiple instances of that data value included with labels appropriate for each of the button presses. The reason for the time range was that statistical methods work better with more data, and most nodes did not have a large number of button presses. The time range increased the number of data points included for each button press from 1 to 21. It also meant that the data points for the button press effectively describe the average conditions at the time the button was pressed, rather than the conditions at a specific instance. The time of 5 minutes was chosen as it was assumed that the thermal comfort conditions were unlikely to change significantly during that time. When building the models, the only data used is the weather data and data from the the node whose comfort votes are being used to select the data. This is a total of 33 variables; 19 from the meteorological station, and 14 from the node.

The results of this were promising, as the plots clearly showed the comfort votes of like classes were grouped together. Example plots from 4 rooms are shown in Figure 7.7. Figure 7.7a uses data from a node in a single occupant office. This particular node only had 12 comfort votes, yielding 252 data points in its plot. As can be seen the clusters are well separated. The clear class distinctions are likely due to the low number of comfort votes as compared to the dimensionality of the space. Figure 7.7b is a single occupant office similar to the previously mentioned plot, but it has a greater number of comfort votes — 43 votes yielding 903 points plotted. While their classes aren't separate any more, three clouds of points (one for each class) can be clearly recognised. Figure 7.7c shows the results of a multi-occupant electronics lab. While the data from the previous two plots can be assumed to primarily representative of a single individual, the data from this node represents multiple individuals. As a result of this, the classes are less distinct, as different people have different comfort zones, which are overlapping and not differentiated when plotted here. A second result of this is that there are more comfort votes; over 100 yielding over 2200 points plotted. This also contributed to the overlapping of the classes. Figure 7.7d is from a corridor leading to the offices in the main deployment area. This node received the largest number of comfort votes (around 270) due to the number of people walking past it. Again as the number of users and comfort votes increase, the separation between the clouds decreases, though they do remain contiguous.

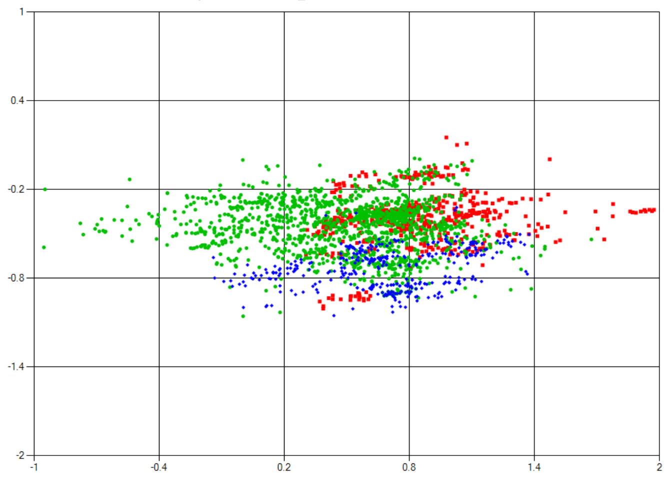
Since the 2D scatter plots for the two most significant axes of the LDA process showed promise for a basis of classification, it was decided to investigate whether the next most significant axis would further separate the classes. A program was written to perform the LDA process and feed the results into the 3dPondViewer application. Some sample results are shown in Figure 7.8. The three most significant LDA axes are plotted in order of significance against the Y, X and Z axes of the 3D space. The data is plotted inside a cube that is aligned to the axes of the 3D space which acts as a visual reference. The data is scaled such that the maximum or minimum data point on any axis is plotted at the position of the cube face that intersects that axis. The plots on the left are viewed from the front, resulting in the Y axis running up and down the



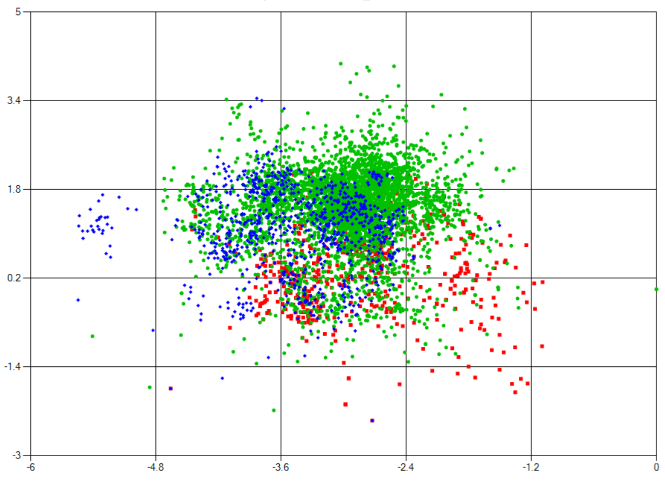
(A) Single Occupant Office (few votes)



(B) Single Occupant Office

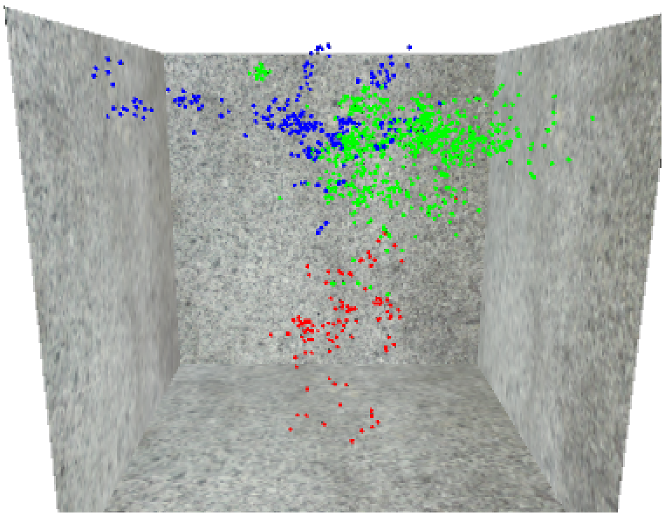


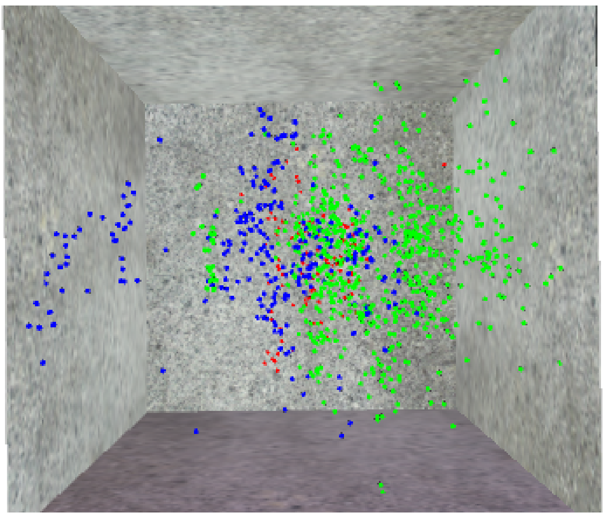
(C) E8 — multi-occupant lab

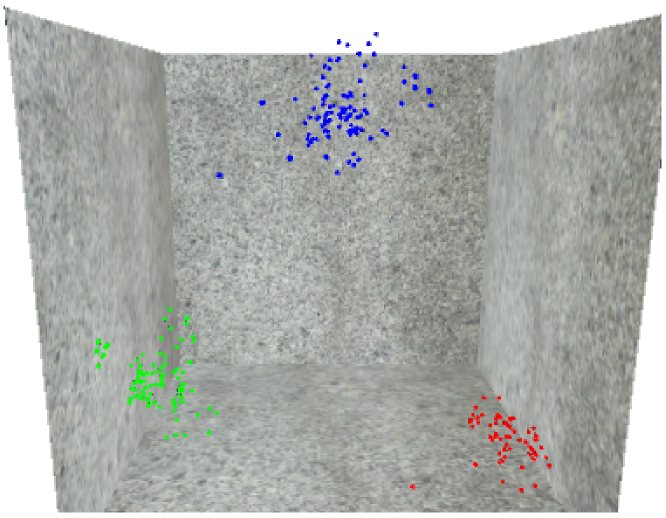


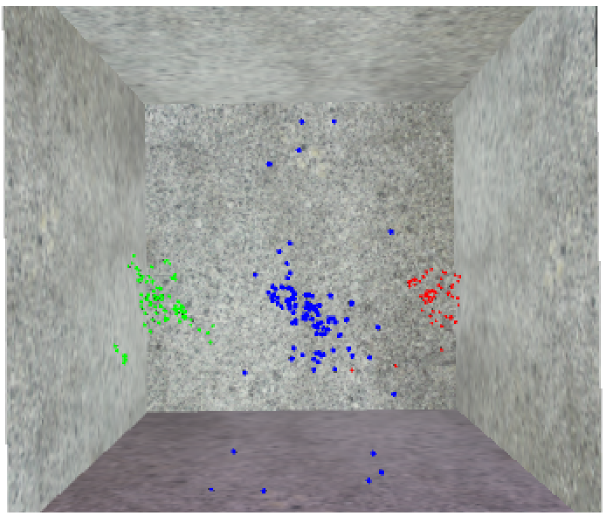
(D) Low Corridoor

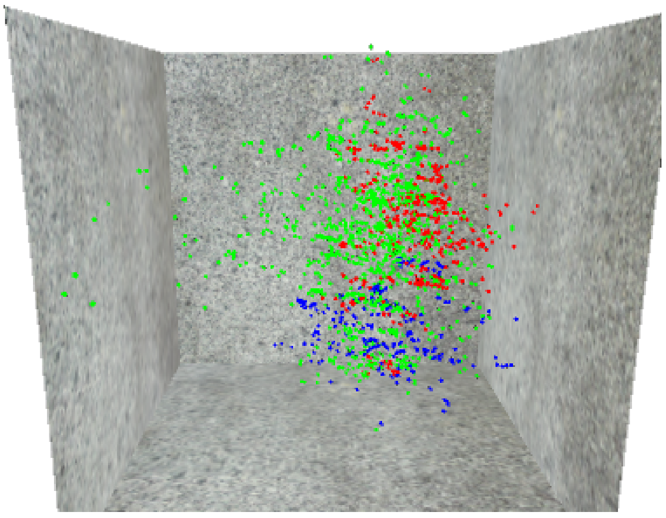
FIGURE 7.7: A selection of plots of 2D LDA results. The results come from four different locations. There are two single occupant offices, one office where the occupant voted often and on where the occupant did not. There is a small multi occupant lab and there is a corridoor, both of which have many votes. Point colours correspond to the following training sets — Red: Too Hot, Green: Just Fine and Blue: Too Cold.


(A) Single occupant office front view


(B) Single occupant office top view


(C) Single occupant office (few votes) front view


(D) Single occupant office (few votes) top view


(E) Multi-occupant electronics lab front view

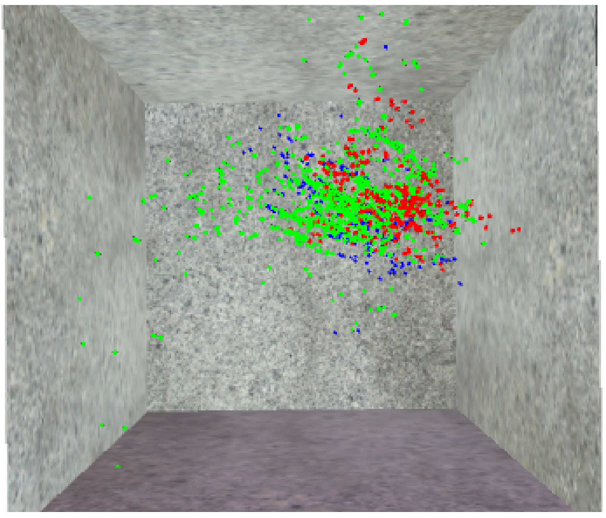

(F) Multi-occupant electronics lab top view

FIGURE 7.8: 3D plots of the first 3 most significant LDA axes for three different nodes.

page, the X axis running across the page, and the Z axis into the page. These plots are comparable to the two dimensional plots in Figure 7.7. The plots on the right are viewed from above, resulting in the Z axis running up and down the page, the X axis across the page, and the Y axis out of the page. This perspective allows the distribution of points along the third LDA axis to be clearly viewed.

The plots showed that moving from two to three dimensions didn't result in further separation of the classes. As a result all further processing based on LDA was performed only on the two most significant LDA axes.

7.4.3 Histograms

Prior to classification histogram plots of the LDA processed data were produced. These were produced to gain a better understanding of the results of the LDA and how their structure and distribution would affect classifiers. This enabled via inspection to check the validity of the assumptions some classifiers are based of — such as that the data follows a Gaussian distribution. The plots were produced for all the data divisions that would be used when producing classifiers. The division technique used is described in Section 7.5.2.

7.5 Classification

There are a number of methods for producing classifiers to predict the comfort classification (Too Hot, Ok, Too Cold) given a set of measured environmental parameters. All the classifiers discussed here are based on two dimensional data that has been produced by processing the measured environmental parameters using Linear Discriminant Analysis.

Table 7.4 provides a summary of the accuracy of the classifiers produced. There is a subsection discussing each classifier in the table. Section 7.5.4 discusses the Cosine window function based classifier. Two Gaussian based classifiers are discussed in Section 7.5.5 and Section 7.5.6. Section 7.5.7 discusses the k-Nearest Neighbour (k-NN) classifier. Finally 8.2.1 discusses areas of interest for future work relating to building classifiers for this dataset.

Classifier	Average accuracy of all classifiers	Average Hot classifier accuracy	Average Ok classifier accuracy	Average Cold classifier accuracy
Cosine Window	49.42	44.52	52.68	49.23
Gaussian Likeliness	49.81	47.61	49.65	50.45
Gaussian Probability	49.67	47.04	49.65	50.82
k-NN (K=5)	44.95	39.33	58.73	34.93
k-NN (K=7)	45.23	38.65	59.67	35.51
k-NN (K=11)	45.36	38.15	60.23	35.82
k-NN (K=15)	45.37	37.97	60.78	35.48
k-NN (K=19)	45.49	37.65	61.39	35.54
k-NN (K=25)	45.51	37.48	61.41	35.81

TABLE 7.4: The overall average accuracy of all classifiers, and the average accuracy of each classifier across all rooms

Interpreting Accuracy Results

When interpreting the class accuracy results one needs to compare the accuracy value to the value one would expect from random chance. In most instances this is 33.3%, some nodes only have data for two classes so one should compare their values to 50

Values lower than random chance indicate that the classifier is sacrificing performance of that class to artificially inflate its performance in another class. An extreme example of this would be a two class classifier with 100

7.5.1 Dataset

The dataset used for creating classifiers for each node consists of data from the node the classifier is being made for and the data from the Whitworth meteorological station. The data for each class for a node is selected using the comfort votes from that node. Node and weather data between 5 minutes before and 5 minutes after the time of each comfort vote, is added as a group of points to the set of data to be used for creating the classifier. It is labelled as being of the same class as the comfort vote which was used to select it. With 30 seconds separating readings and a 5 minute selection range, a comfort vote can select up to 21 rows of data (forming 21 data points) to go into a group. Only rows of data where both node and weather data are available are used, so a group may contain less than 21 data points.

There are several reasons for selecting up to 21 data points to represent each comfort vote. Firstly, instead of training the classifiers on the specific environmental conditions at the time of the vote, including samples from the surrounding time trains the classifier to recognise similar conditions as being from the same class too, which aids in the generalization of the classifiers. Secondly, LDA is a statistical method, and statistical methods work better with more data. Thirdly, some nodes had recorded fewer comfort votes than there were measured parameters. LDA cannot be performed if there are less measurements than parameters. Increasing the number of measurements, by including samples from around the time of the comfort vote, enabled LDA to be performed for these nodes, and thus allow classifiers to be produced.

There was significant variation in the number of votes per node. Multi-occupant work areas received more votes than those in single occupant offices, but not as many as those in well trafficked areas. Table 7.5 shows how many comfort votes each node received and the resulting data points used for classification, in the time frame used for testing the classifiers.

Name	Total Votes	Hot Votes	Ok Votes	Cold Votes	Total Points	Hot Points	Ok Points	Cold Points
E1H Barry Lennox	67	7	46	14	1261	133	862	266
E1B	78	31	28	19	1461	589	511	361
E1M Meeting Room	140	29	88	23	2635	551	1647	437
E1G	20	7	8	5	370	123	152	95
E1A	27	4	19	4	513	76	361	76
C34 Student Electronics Workshop	159	44	56	59	2854	814	956	1084
E1L	21	10	8	3	399	190	152	57
F1 East Side	380	69	242	69	7206	1311	4591	1304
F1A South Block	240	130	26	84	4510	2450	481	1579
E8 Igor Labs	157	43	84	30	2963	812	1584	567
E7 Peters Room	21	8	13	0	365	140	225	0
F1B North Block	142	65	55	22	2665	1228	1025	412
Low Corridor	668	77	518	73	12671	1463	9821	1387
Total	2120	524	1191	405	39873	9880	22368	7625
Average	163.08	40.31	91.62	31.15	3067.15	760.00	1720.62	586.54

TABLE 7.5: Data about the dataset used for training and testing classifiers

7.5.2 Leave One Out Cross Validation

Due to the small number of button presses recorded on some nodes, it was found that the models wouldn't form correctly if the training set's size was reduced by removing multiple data points to use as test data to assess the models. The points would be intermixed and the regions of space dominated by points of a particular class wouldn't be apparent. In order to assess the models without causing them to fall apart a method was needed that required very few data points be removed from the training set in order to test the model. Leave One Out Cross Validation (LOOCV) is a method of cross validation which requires only one data point to be removed from the training set in order to test the model.

If you have N data points, LOOCV works by building the model N times, and each time using $N-1$ points to train the model, and the 1 point left out is used to test it. A different point is chosen to be used to test the model. The mean accuracy is then calculated based on the results of all the tests. The mean accuracy is representative for models built by the method being used and trained on $N-1$ data points. The variance associated with the accuracy measure is higher using this method than other validation methods. More details on LOOCV can be found in Section 6.5.

The LOOCV method was modified slightly when used in this project. As was described in Section 7.5.1, each button press was transformed into a group of up to 21 data points. Instead of leaving each individual data point out once, individual groups of data points were left out instead. The idea was that this would be equivalent to leaving out an individual button press. This was important because the points all described similar conditions having only a short time between them. If a single point of the group of up to 21 points was left out, there would be a guarantee of up to 20 points which were very similar to it that would be in the training data. This would be almost equivalent to not removing the data point at all, and testing the classifier using the training data.

7.5.3 Accuracy

The accuracy values presented in this section were calculated using the following method. When a classifier is trained and tested on data from a node, the results of the classifier are stored in a file. The file stores the results of each iteration of the LOOCV on a separate row. Each row contains the number of points being classified in that iteration — the data points that were left out that iteration; the actual class of those data points; the number of true positives and false negatives; and the number of data points classified into each of the three classes.

Given a file of classifier results for a node, the accuracy for each class is calculated by summing up the total number of points for the class whose accuracy is being determined giving the '`totalClassPoints`'. The number of points whose actual class is the same as the class whose accuracy is being calculated, which were correctly classified as being from that class, is totalled giving the '`totalClassified`'. The accuracy of the classifier at classifying points of the class whose accuracy is being calculated (the '`classAccuracy`'), is the result of dividing `totalClassified` by `totalClassPoints`.

The overall '`Average Accuracy`' of the classifier is calculated as the mean of the three class's `classAccuracy` values. If a `classAccuracy` value was not present (perhaps due to the class not receiving any comfort votes), then the `Average Accuracy` was calculated as the mean of the classes which were present.

Table 7.4 displays the average accuracy across all the nodes of each classifier. The accuracies displayed for each class in that table are calculated as the mean of the `Average Accuracy`s for each node for a given classifier. The average accuracy column in this table is calculated as the mean of those values.

7.5.4 Cosine Local Neighbourhood Windowing Function

The first method used to attempt to classify the LDA processed data is the Cosine Local Neighbourhood Windowing Function. The classifier is prepared by running a set of labelled training data through the LDA process. This produces a set of labelled data points in LDA space, and a transformation matrix which maps measured parameters to LDA space.

When classifying a data point, the data point's position in the LDA space is first determined by multiplying the data point's readings by the transformation matrix produced when processing the training data. Next all the LDA transformed training data points within a set range r_a of the position of the data point to be classified are found. The range has a subscript $_a$ as it can take on a different value for each axis in the multidimensional dataset. Next, a value is calculated for each of the three possible classes. The value for a given class is the weighted sum of all the data points of that class from the data points that were previously selected, due to being within the range of the point to be classified. The data point is classified as being of the class which has the highest total.

Name	Average Accuracy	Hot Accuracy	Ok Accuracy	Cold Accuracy
E1H Barry Lennox	58.58	51.13	64.85	59.77
E1B	53.73	59.08	44.23	57.89
E1M Meeting Room	48.92	35.93	53.86	56.98
E1G	20.43	30.89	28.29	02.11
E1A	56.69	43.42	75.35	51.32
C34 Student Electronics Workshop	42.05	28.50	35.46	62.18
E1L	24.61	36.32	37.50	00.00
F1 East Side	59.94	41.80	62.32	75.69
F1A South Block	58.19	62.78	42.83	68.97
E8 Igor Labs	37.62	22.54	45.52	44.80
E7 Peters Room	72.98	79.29	66.67	-
F1B North Block	65.12	63.60	75.22	56.55
Low Corridor	43.56	23.44	52.72	54.51
Average	49.42	44.52	52.68	49.23

TABLE 7.6: Results of the Cosine Local Neighbourhood Windowing Function classifier

The weighting of each training data point is calculated according to Equation 7.1. The variable d_a represents the distance between the training data point and the data point to be classified along the axis a. A is the total number of dimensions the classifier is being used in.

$$\frac{\sum_{a=1}^{A} \cos(d_a/r_a)}{A} \tag{7.1}$$

In the classifier trained on the LDA processed data, A was chosen to be 2. This meant that d_a took on the values of the distance between the training data point and the data point to be classified along the x-axis and the y-axis. The range for a given axis r_a is calculated as half the average of each class's standard deviations of the LDA processed training data for that axis.

This windowing function makes data points, which are closer to the point to be classified, contribute more than points which are further away. It was inspired by the k-NN classifier in that it looks in the local area of the point, but it considers the values of points in a wider area, and is affected by the density of points in an area. This has a smoothing effect on its output value and prevents it from being affected by a small number of tightly clustered points from one class in an area dominated by points of another class.

Table 7.6 contains the accuracy results for the classifiers produced for each node. Overall the classifier performed well for a first attempt. It had poor performance for nodes with few points. Nodes which had multiple people interacting with them had lower performance than those which only had a single person voting; though they would typically have average performance at classifying one or two classes.

7.5.5 Gaussian Likeliness

The Gaussian Likeliness classifier fits three Gaussian surfaces to the LDA processed data. One surface is fitted to the training points of each class. To classify a point, its coordinates in LDA space are determined; then the value of each of the Gaussian surfaces at that point is calculated. The point is classified as being of the class whose Gaussian surface was highest at that point.

Unlike the Cosine Windowing and the k-NN classifiers the Gaussian Likeliness classifier does not classify a position in LDA space as being of a particular class based on the points in the local area. Those other classifiers may have islands of one class mixed in with another if there is a sufficient density of points of that class in a local area. The Gaussian Likeliness classifier is different in that it divides up space into three distinct regions (one region for each class) with well defined boundaries.

Unlike the Cosine Windowing classifier and the k-NN classifiers, once the classifier is trained the training data is no longer, needed as it is only used to produce the model and not used in the actual classification process. This means that the Gaussian Likeliness classifier will classify points faster if there is a large training set. This is because the other classifiers need to compare a point to all the other points when classifying a point, whereas this classifier needs only calculate the height at a point on some pre-computed Gaussian surfaces. In Big-O notation, this classifier is $O(1)$ where as classification in other classifiers is $O(N)$ where N is the number of points in the training dataset. Only requiring the curves to be stored also means that a trained classifier requires much less memory to store and use.

The accuracy results of the Gaussian Likeliness classifier are shown in Table 7.7. The results are fairly similar to the Cosine Windowing classifier with most nodes performing better than random chance. Similarly, the classifier performed poorly on two of the nodes which only had 20 comfort votes in total across all three classes, but performed well on the node with 21 votes spread between just two of the classes. Again the classifier performs less well on some nodes (the Electronics Lab, Electronics Workshop, and the corridor node), where there are multiple people voting. It is noted that the performance of this classifier for the Hot class is slightly improved for the Student Electronic Workshop node as compared to the Cosine Windowing classifier; though

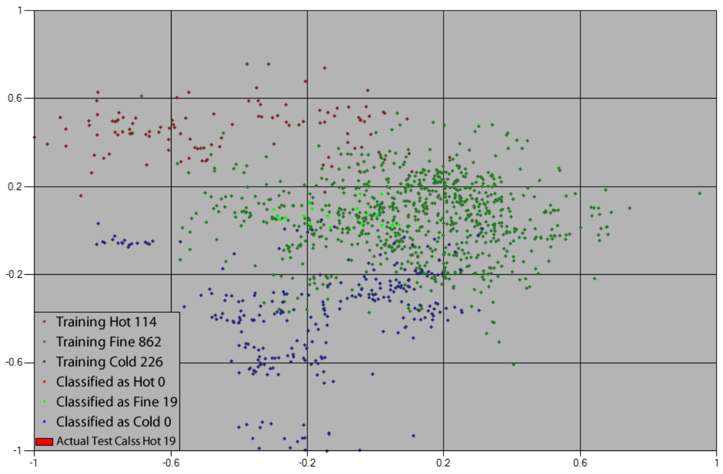
Name	Average Accuracy	Hot Accuracy	Ok Accuracy	Cold Accuracy
E1H Barry Lennox	58.16	51.13	60.56	62.78
E1B	54.50	57.56	43.05	62.88
E1M Meeting Room	49.72	41.20	50.76	57.21
E1G	18.98	21.95	32.89	02.11
E1A	58.01	38.16	70.08	65.79
C34 Student Electronics Workshop	43.67	38.21	35.15	57.66
E1L	28.07	55.26	28.95	00.00
F1 East Side	60.92	50.34	58.03	74.39
F1A South Block	57.77	62.82	44.70	65.80
E8 Igor Labs	36.92	26.72	42.23	41.80
E7 Peters Room	72.75	79.29	66.22	-
F1B North Block	64.14	66.53	62.05	63.83
Low Corridor	43.88	29.73	50.80	51.12
Average	49.81	47.61	49.65	50.45

TABLE 7.7: Results of the Gaussian Likeliness classifier

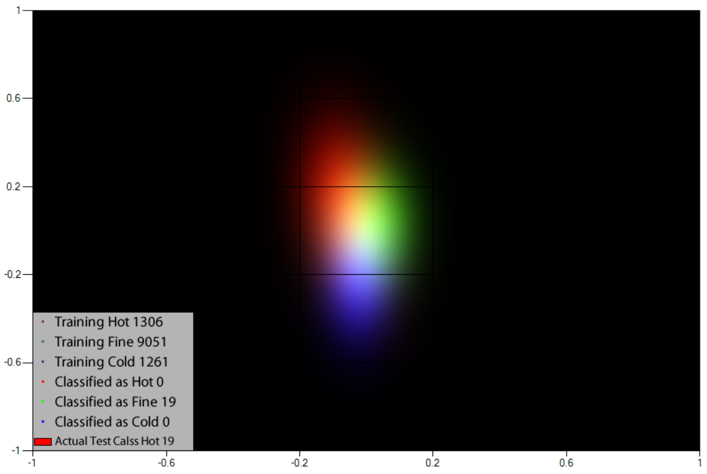
performance is now only slightly better than random chance. Again performance is reasonable for the multi-user nodes in the open plan work areas (the East Side, North and South Block nodes), and the Meeting Area node. Overall the results for each node are very similar to the Cosine Windowing classifier, with performance largely governed by the dataset. The classifier is marginally better than the Cosine Windowing classifier on average.

Figure 7.9 shows some example plots of the results of the Gaussian Likeliness classifier. There are three figures each for a single occupant office and the corridor node. The office node plots are on the left and the corridor plots are on the right. Figure 7.9a shows the data points used for training the classifier along with a group of test data points in a lighter colour. This plot is for reference to compare to the visualizations of the classifier. Figure 7.9c visualizes the 2D Gaussian surfaces by changing the RGB colour values of the background. The value of the Hot class Gaussian curve at a point determines the red channel value at that point; the value of the Ok class's curve controls the green channel; and the value of the cold curve sets the blue channel. The values are mapped such that a 0 value of the curve is 0 of the related colour, and the value at the highest point of the surface is 255, the maximum colour value. In Figure 7.9e the background colour is set to be that of the class whose curve is highest at each particular point.

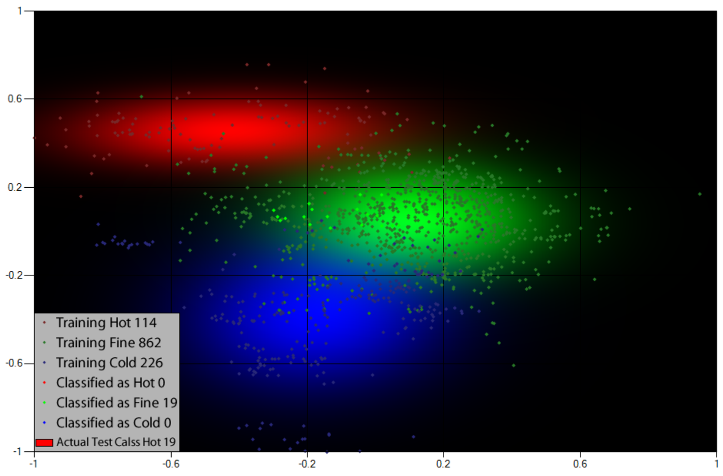
Figure 7.9c provides a visual representation of the 2D Gaussian surface which was



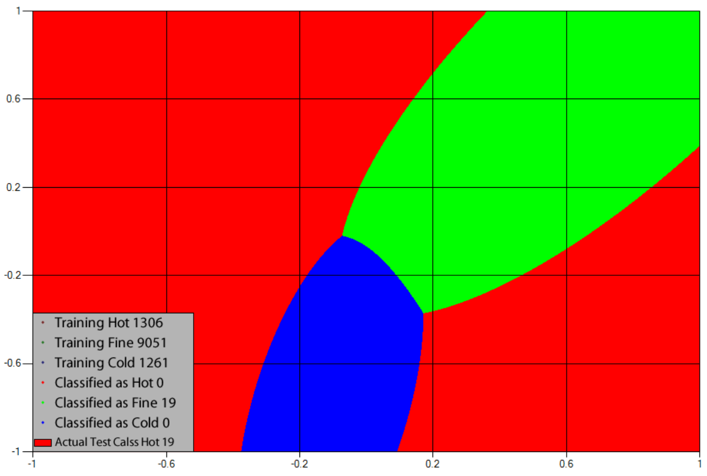
(A) Data points only



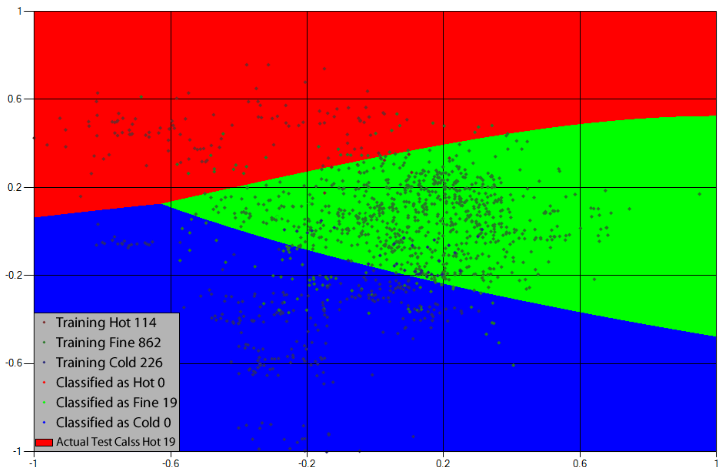
(B) Background coloured by Gaussian Curve values No data points



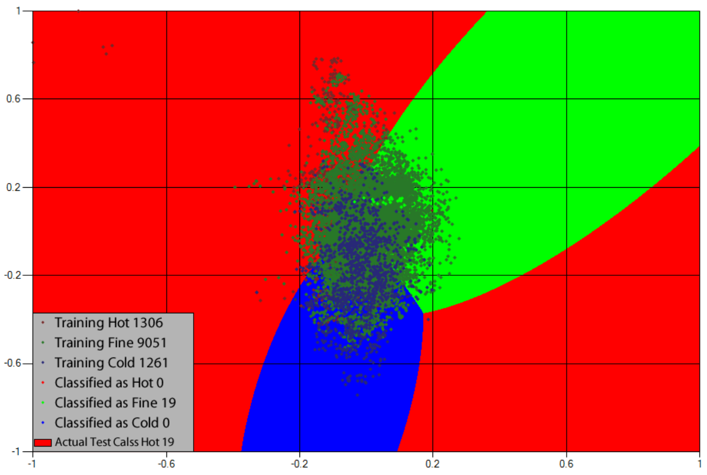
(C) Background coloured by Gaussian Curve values



(D) Background set to colour of class with highest curve at that point, no data points



(E) Background set to colour of class with highest curve at that point



(F) Background set to colour of class with highest curve at that point

FIGURE 7.9: Various plots showing results of the Gaussian Likeliness classifier. Plots on the left are from a single occupant office. Plots on the right are from a frequently used corridor. Key indicates how many points of each class were used for training and testing, what class the test points were and what they were classified as.

fitted to the data points of each class. Visual inspection suggests that the curves fit the data from this node quite well. Figure 7.9e shows the classification regions that the curves define. It can be seen that the boundaries between the regions are positioned such that they divide the data points in a way that places the majority of the points of a class in the correct region.

Figure 7.9b is the same type of plot as Figure 7.9c, but for the corridor node. It doesn't show the data points as including them on the plot would obscure the background and prevent one from getting an idea of what the Gaussian surfaces of the classifiers look like. Compared to Figure 7.9c the curves are much closer together and overlap more. This is a result of the greater overlap of the clouds of data points of each class. This is believed to be due to the corridor node being used by many people as opposed to just one; so its data is made up of many people's comfort zones, which are all slightly different.

Figure 7.9d is a plot for the corridor node which shows the regions where each classifier's curve is the highest of all the curves, and thus what data points in that region would be classified as. Again the data points are not plotted as they would prevent one from seeing where the boundaries are. Figure 7.9f includes the data points. One can compare Figure 7.9b and Figure 7.9d to Figure 7.9f and relate their background to the point cloud to understand how they relate.

The corridor node plots illustrate why the classifiers were typically correctly classifying the test points at a rate of 50–60%. There is such a large overlap of points of different classes that if you define a region as being of a certain class then one is certain to mis-classify a large number of points. A solution to this would be to move to a higher dimensional space for classification. Increasing the dimensions could increase the spacing between the majority of the points of each class. However, the reason this analysis was done on 2D data was that prior investigation of LDA showed that the third LDA dimension didn't contribute any significant class separation. To make additional dimensions work, one would require something to further differentiate the different classes; either additional environmental parameters (additional sensors on the nodes or using data from multiple nodes or other sources) or some other processing besides LDA (perhaps include non-linearly transformed versions of measured variables).

One may see the lower right region of Figure 7.9d being classified as hot as a problem with the classifier, since it's on the opposite side of the data point cloud to where the majority of the hot class training data points are. The reason this area is classified as being hot is because the hot training data has a higher standard deviation than the other two classes. This results in the hot classifier's Gaussian surface having less height than the others, but having a higher value further away from the mean of its class; whereas the other class's curves are higher but drop close to 0 after a shorter distance travelled from their respective means. If one were using this type of classifier in a production system, one should consider not just which curve is highest at a particular point like this plot does, but what the height value of the curve is at that point. When one looks at the lower left region in Figure 7.9d, one can see that all the curves have values close to 0 in that region. The height of the curve can be used as a proxy for confidence in classifying a point as being of that curve's class. It might be more reasonable to give an error, or return no class for regions which have very low curve heights. Alternatively, one could classify points in regions, where all curves are below a certain height, as being of the class whose mean is closest to that point.

7.5.6 Gaussian Probability

The Gaussian Probability classifier is a modified version of the Gaussian Likeliness classifier. The Gaussian Likeliness classifier uses the height of the Gaussian surfaces as a proxy for how likely each particular class is at that point. It doesn't give a measure of probability, as the probability at a particular point on the surface is given by the volume under that point; and since the area under a point is 0, the probability of any class at any point is 0 regardless of the height of the surface. The Gaussian Probability classifier calculates a probability of a class at a particular point by calculating the volume under a small area around the point. It then classifies that point as the class which has the highest probability.

The probability calculated is a numeric approximation to the true probability. The area around the sample point is divided into a number of squares. Only those squares whose centres are within a certain radius of the sample point are used in calculating the area. The length of a squares side is $r/7$, where r is the radius. Each square within the radius of the sample point has the height of the Gaussian surface in its centre point calculated. The probability is calculated as the sum of these heights multiplied by the area of each square.

The true probability could be calculated if the N-dimensional Gaussian library written for this project was extended to provide an N-dimensional error function. There was insufficient time to do this, so the numerical approximation approach was used instead.

Name	Average Accuracy	Hot Accuracy	Ok Accuracy	Cold Accuracy
E1H Barry Lennox	58.19	51.13	60.67	62.78
E1B	54.50	57.56	43.05	62.88
E1M Meeting Room	49.72	41.20	50.76	57.21
E1G	18.76	21.95	32.24	02.11
E1A	58.98	36.84	70.36	69.74
C34 Student Electronics Workshop	43.55	37.84	35.15	57.66
E1L	27.89	54.74	28.95	00.00
F1 East Side	61.03	50.27	58.20	74.62
F1A South Block	57.79	62.73	44.70	65.93
E8 Igor Labs	36.96	26.72	42.17	41.98
E7 Peters Room	70.25	74.29	66.22	-
F1B North Block	64.15	66.61	62.24	63.59
Low Corridor	43.88	29.60	50.72	51.33
Average	49.67	47.04	49.65	50.82

TABLE 7.8: Results of the Gaussian Probability classifier

Table 7.8 shows the results of the Gaussian Probability classifier. As one would expect, they are similar to the Gaussian Likeliness classifier's results. They are however very slightly worse.

7.5.7 k-Nearest Neighbour

The k-Nearest Neighbour (k-NN) is a classic and well known classifier used in many machine learning applications [72]. To classify a point, the k-NN algorithm examines the class of the k nearest data points to the point to be classified in the labelled training set. The point is classified as being of the class which is most frequent in those k data points. Using a smaller value of k results in greater sensitivity to fine details of the training set, but also to noise present in the training data. A larger k results in the classifier representing the larger structure of the classes, with a less detailed boundary region.

As the best value of k was not known, classifiers with a range of values of k were produced. The values of k which were used were 5, 7, 11, 15, 19 and 25. The average results of these classifiers are presented in Table 7.4. To save space this subsection presents more detailed results only for the classifiers with k-values of 5, 15 and 25.

The results for the k-NN classifier with a k value of 5 are shown in Table 7.9. This classifier used the lowest value of k. The results are very poor. Some classes for some

Name	Average Accuracy	Hot Accuracy	Ok Accuracy	Cold Accuracy
E1H Barry Lennox	44.06	16.54	75.41	40.23
E1B	54.96	57.56	41.68	65.65
E1M Meeting Room	44.57	27.59	72.50	33.64
E1G	17.89	21.95	29.61	02.11
E1A	53.00	34.21	74.79	50.00
C34 Student Electronics Workshop	44.26	41.28	50.73	40.77
E1L	29.39	44.74	43.42	00.00
F1 East Side	49.51	31.27	76.76	40.49
F1A South Block	51.31	79.80	14.55	59.59
E8 Igor Labs	35.13	29.68	59.47	16.23
E7 Peters Room	58.98	42.86	75.11	-
F1B North Block	64.50	73.62	59.22	60.68
Low Corridor	36.76	10.25	90.31	09.73
Average	44.95	39.33	58.73	34.93

TABLE 7.9: Results of the kNN5 classifier

of the nodes have good scores, but the other classes on those same nodes have very poor scores. This means that for those nodes the classifier was simply classifying the majority of points as being from the class which had the best score. Typically in these cases the class which had the best score was also the one with the most data points. Indeed, in 7 out of 13 of the nodes, the order of the classes sorted by their classification accuracy from highest to lowest, was same as the order of the classes when sorted by the number of training data points each class had.

Table 7.10 shows the results for the k-NN classifier with a k of 15. 15 is in the middle of the range of k values tested. The results of this larger k value are similar to those of the smaller $k = 5$ value. Some results have shown a slight improvement in all classes, but in others only the class with the most votes gained accuracy at the expense of the other classes.

The results of the highest k value of 25 are shown in Table 7.11. As with the other k-NN classifiers, the classifier performed reasonably well on nodes which had a similar numbers of data points for each class, and performed badly on nodes where one class's data points significantly outnumbered the other classes. The higher value of k did improve performance slightly overall but decreased it slightly in some cases. This k-NN classifier did perform better than the Gaussian Likliness classifier in the F1B North Block case but less well in all others.

The performance of the k-NN classifiers is quite sensitive to the training set. Given

Name	Average Accuracy	Hot Accuracy	Ok Accuracy	Cold Accuracy
E1H Barry Lennox	44.93	16.54	75.41	42.86
E1B	55.59	58.74	42.66	65.37
E1M Meeting Room	45.04	26.86	75.53	32.72
E1G	17.62	21.14	29.61	02.11
E1A	52.75	32.89	75.35	50.00
C34 Student Electronics Workshop	44.99	35.87	51.78	47.32
E1L	30.18	43.16	47.37	00.00
F1 East Side	50.43	28.76	81.73	40.80
F1A South Block	51.67	82.29	12.89	59.85
E8 Igor Labs	34.16	25.25	63.83	13.40
E7 Peters Room	60.01	43.57	76.44	-
F1B North Block	66.69	73.94	62.54	63.59
Low Corridor	35.80	04.65	94.98	07.79
Average	45.37	37.97	60.78	35.48

TABLE 7.10: Results of the kNN15 classifier

Name	Average Accuracy	Hot Accuracy	Ok Accuracy	Cold Accuracy
E1H Barry Lennox	45.82	18.05	76.57	42.86
E1B	55.49	59.42	41.68	65.37
E1M Meeting Room	44.45	23.59	76.56	33.18
E1G	17.62	21.14	29.61	02.11
E1A	52.84	32.89	75.62	50.00
C34 Student Electronics Workshop	45.81	33.54	53.24	50.65
E1L	30.39	43.16	48.03	00.00
F1 East Side	50.70	28.30	83.08	40.72
F1A South Block	52.02	82.53	13.10	60.42
E8 Igor Labs	33.60	21.55	66.73	12.52
E7 Peters Room	59.79	43.57	76.00	-
F1B North Block	67.59	75.24	62.24	65.29
Low Corridor	35.57	04.24	95.85	06.63
Average	45.51	37.48	61.41	35.81

TABLE 7.11: Results of the kNN25 classifier

a “nice” training set where each class has a similar (and large) number of data points, and without too much overlap of the points in different classes, the k-NN classifier performs quite well. In most cases however, the data from the nodes is not “nice”. In most cases one class occurs far more frequently than the others, in one case a class isn’t represented at all as the room never gets cold. There is also significant overlap in the classes. Since k-NN is based on counting a number of points and choosing the most frequent class in that count, it can get affected detrimentally if in the training set one class occurs more frequently — as there are simply more of that class’s points about to count.

Chapter 8

Conclusion

This chapter discusses what this project intended to achieve, how well the project met its aims and objectives, and the conclusions that can be drawn from the project and suggests possible avenues of future research.

The main motivation for the work is the need to reduce energy usage to combat climate change. Heating and air-conditioning systems use a lot of energy, so finding a way of reducing their energy use while maintaining satisfactory conditions for occupants is desirable. This thesis examines the use of a sensor network to classify occupant comfort based on both the local environment around the sensor nodes and the external environment. Having a model that can predict occupant comfort based on real time data of environmental parameters would allow the development of an indoor environment control system that regulates comfort in a closed loop manner. The same model could be used to predict comfort with, or hypothetical values of, parameters. Optimisation algorithms could then be applied to the said control system to minimize its power usage while maintaining occupant comfort.

The specific aims and objectives of the project are described in Chapter 1.3. Overall the project has met its aims and objectives.

8.1 The System

The sensor network as deployed consists of 17 sensor nodes connected to an Ethernet network. These nodes send data to a database server. The original plan called for multiple sensor networks to be developed. There was to be an initial small scale deployment, followed by a larger deployment of nodes. The nodes in the second deployment would be of an improved design based on lessons leaned and data from the initial deployment. Control of the building's heating system was also planned. The plans for both the larger deployment and the heating control had to be dropped when development of the initial batch of nodes took longer than planned.

Overall the system has performed well. It has operated since September 2012 without any sensor network hardware or software failures, and continues to do so at time of publication.

8.1.1 Hardware

A total of 20 nodes were produced, of which 17 were deployed and 3 were kept for calibration and demonstration purposes. These nodes cost more to produce than the original target price for a node, but are still of lower cost than comparative units which are available on the market. The main reason for the higher cost was that they were deliberately outfitted with more sensors than necessary. In particular they had multiple sensors measuring the same environmental parameters. This was done to enable designers of future nodes to analyze the data from these nodes and decide if they need the cheaper or more expensive version of a sensor, or if having directional light intensity measurement is important. Analysis of this nature was going to be performed when designing the second version of the nodes for the second larger deployment, but time limitations prevented this as the development of the first generation of nodes took longer than expected.

Some analysis of the sensors was performed, but it was mainly limited to calibration to assess the relationship between measured values and true values of the parameters. The analysis of whether the lower quality sensors were useful, or whether the directional light sensors were of benefit was not performed. The data is available as a resource to aid future sensor network designers.

Overall all aspects of the system have performed well. The hardware met its objectives and fulfilled its purpose. The calibration performed provided factors to correct measured values, though in the end it wasn't necessary for the analysis performed as the data normalisation step would remove any calibration adjustments.

8.1.2 Software

Custom software was written for both the nodes and the server they send their data to. The primary functions of the server software were to synchronize the data sampling of the nodes and to receive and store data transmitted to the server. The secondary functions related to monitoring the status of the system, sending out alert emails based on the system status and providing access to the received data. The primary functions of the software on the node were to establish contact with the server, to sample its sensors at the correct time and to transmit the samples to the server. The secondary functions were to communicate a description of the node's sensors to the server if requested, and to provide user feedback via the LEDs.

Due to the requirement for the system to operate for a long duration the software was designed to both send reports about the system status and to detect and automatically recover from errors. Different error recovery strategies were employed on the nodes and the server. The nodes had a simple strategy of reverting to a known good state by resetting themselves should any deviation from normal operation be detected. This strategy was chosen since there are two types of problem the nodes can encounter: firstly problems that can be fixed from software and secondly, problems which can't be fixed from software. If the problem is one which can't be fixed from software, then no recovery procedure would be effective, and resetting the node has no effect. If the problem is one that can be fixed from software, then traditionally it would be necessary to identify the error and then take steps to recover. This strategy requires anticipating all recoverable error types. The simpler strategy of resetting the node is guaranteed to put the node into a known good state. This may cause the loss of some measurements, but it means that all measurements sent and successfully received by the server are valid measurements.

The main piece of server software called the DataReceiverServer (DRS) performed the primary functions of the server software. It maintained the system's representation of time, synchronized the node's sampling, stored received data in the database, and handled node registration. The database used for the project was the open source PostgreSQL database. These two pieces of software worked well to collect and store the data from the sensor nodes. The status of the DRS and the nodes was monitored by another program. This program received two types of heartbeat messages from the DRS. One type was a simple timed heartbeat message to indicate the DRS program was running. The other was sent whenever a node's data packet was received. This type included the node's ID to allow tracking of which nodes were active without querying the database. The program that received these heartbeat messages fulfilled the secondary functions of the server, that of monitoring the system status and sending out alert emails should it be too long since a heartbeat message was received from the DRS.

The storage of weather readings was handled similarly. There is a piece of software that receives the weather readings and stored them in the database, and there is a piece of software that monitors it and sends alerts should it be too long since the last reading was received. The software to retrieve the building's fiscal smart meter readings did not have monitoring software as it retrieved it's data periodically from an external source rather than receive data sent to it.

The monitoring software would continue to send emails so long as a monitored component had failed. The time between emails being sent was doubled each time one was sent to avoid generating too many emails for a long duration outage.

Overall the software developed worked very well. The monitoring software generated prompt notifications of any errors. These errors were usually the power to a lab being turned off at the end of the day, or an occupant unplugging a node from an Ethernet wall socket to plug in a computer rather than use the provided Ethernet switch. This allowed the errors to be corrected with actions such as plugging the node back in, or routing power from outside the lab to the node. Using the PostgreSQL database to store the measurements was hugely beneficial for analyzing the data. Storing the data in a text file as other research sensor networks have done in the past would have made analysis impractical give the size of the dataset resulting from the duration, frequency and number of nodes used in this project.

8.1.3 Network protocol

The nodes and the server used a custom protocol to communicate. The protocol served multiple purposes. It allowed the server to set the node's clocks to the correct time and to communicate the next data sampling time to the nodes. It allowed the nodes to send their data to the server as well as resend data the server did not acknowledge receiving. It also allowed the node to transmit a description of itself and its sensors to the server and also for the server to request the description to be sent.

A custom protocol was used since it allowed the desired functionality to be implemented with a minimum of overhead. Since the nodes operated on an IP based network the two main protocols to build the custom protocol on top of were TCP and UDP. This protocol was built on top of the UDP protocol because it was a lighter weight protocol than TCP and because TCP was unsuitable for the time synchronization functionality of the protocol. Using a custom network protocol has worked well for this project. It solved the problems it was designed to and it did so with a minimum of overhead.

8.1.4 Sensor Nodes V2

Some design work was done on a second version of the sensors nodes. This design was intended to integrate the Ethernet switch into the sensor node so only one device needed to be installed in each location. The new design featured a 32-bit PIC microcontroller and a 3 port Ethernet switch integrated circuit one PCB. The change to the 32-bit PIC was due to it supporting the [R]MII interface required to use the third port of the Ethernet switch IC.

The 32-bit microcontroller also offered several advantages. It had a unique MAC address preprogrammed in it and thus eliminated the need for the external MAC address containing EEPROM chip the first design used. The greater processing power afforded by the 32-bit microcontroller permitted use of more CPU intensive sensor data processing — in particular Fourier transforms of the microphone data could be taken to produce average and peak levels of different frequencies during the sampling period. It was also of interest to investigate whether an occupancy detector could be developed by connecting a inexpensive webcam to the microcontroller's USB port and implementing a basic face detection algorithm to attempt to count the number of occupants visible from the node.

Development of the redesigned sensor nodes stopped when it became apparent that it would take too long to develop them, and that the focus of the project needed to change to analysing the collected data rather than collecting more.

8.2 Analysis

The aim of the analysis performed was to produce a classifier that can predict occupant comfort from the data produced by the sensor network as well as the data from the building's smart fiscal meters and data from a nearby meteorological station. Before attempting to produce classifiers, time was spent analyzing the data to gain a better understanding of the dataset as well as its relation to the comfort votes.

The initial analysis was quite disheartening. There was much overlap in parameter values for different vote types in all measured parameters. Worse still the average value of the parameters of different vote types would sometimes be in an order one wouldn't expect. For example if the parameter was temperature one would expect the cold class to have the lowest average value, the hot class to have the highest average value and the fine class to be in the middle. However Figure 7.5f in Chapter 7 provides an example where cold class has the lowest average value, hot class is next highest, and fine class has the highest mean value. At face value such results would suggest that if people think they are too hot, you need to increase the temperature to make them satisfied. This is clearly wrong. The explanation for results like this is that thermal comfort is a multivariate problem, so there were differences in other parameters that were making people happy at the higher temperature and feel too hot at the lower temperature. The initial analysis clearly showed that a univariate approach would not work for comfort classification.

Initial attempts at multivariate analysis did not go well. PCA was used both to reduce the number of parameters used in the analysis and to try and transform the data parameter space into something with more separation between the classes. When run on 256 parameters in the database it showed that about half of them contained the vast majority of information and the other half were essentially duplicating this information but that still left 128 parameters to work with. An examination of the spread of the different vote classes along the axes in PCA space produced similar results to the initial univariate analysis. There was much overlap in the values of each vote class making them unsuitable for producing classifiers. Animating the plots of the most significant principal components so that points were added and removed based on the time they were collected suggested the reason PCA was producing unsatisfactory results. There was a clear 24 hour cycle in the data points. PCA arranges axes based on which has the most variation. Parameters which vary together contribute to the same axis. The sun greatly affects many parameters, even indirectly, and it affects them at the same time. When the sun rises it increases the light level, the temperature as well as other parameters, but it also indirectly affects other parameters. The building being monitored is primarily used in the day time. This means that when the sun rises; the values of the fiscal smart meters rise; the occupancy detector values rise; the sound level rises; and the building temperature rises because the heating was set to turn on ready for the people whose arrival is related to the position of the sun. The PCA charts were essentially descriptions of the time of day rather than descriptions of comfort.

After the failure of PCA a different but related multivariate method called LDA was tried. The use of LDA was possible due to having comfort classes labelled by comfort votes. LDA attempts to find a coordinate transform that maximises the variation between classes while minimizing the variation within a class. Since it wasn't trying to maximize the overall variation it wasn't 'distracted' by the sun and it produced results with reasonable separation between classes. The LDA results had the majority of the separation of the classes contained within the first two axes. The separation was not perfect however, there was still some spread between classes. This is a result of the nature of thermal comfort however. Comfort is a sliding scale rather than a group of distinct classes. It also deals with averages of people, and people have different values they consider acceptable or uncomfortable. The regions of overlapping comfort classes are the regions where one transitions between regions where the majority of people are comfortable and where the majority of people are uncomfortable.

The classifiers based on LDA worked as well a can be expected given that there is no clear delineation between classes. The boundaries between classes produced by the best classifiers appeared to be positioned as well as they could be given the dataset. There were no clear directions parts of them could be moved that would result in improved accuracy in all classes. Time constraints brought a close to the analysis of the data. Section 8.2.1 describes some methods of analysis that seem to be good avenues to explore based on what was learned from the analysis performed so far.

8.2.1 Potential Methods of Analysis

This section describes some analysis methods which were thought to have potential, but were not explored due to time constraints.

Changing Classification Type

One of the issues affecting the accuracy of the classification is the fact that there isn't actually a clear distinct separation between the comfort classes in reality. There is an overlap in the region between where most people choose one class and where they choose another. This means that in this transition region if you draw a hard boundary between the classes and definitively classify a point as being of a specific class then you are guaranteed to make incorrect classifications as there are points from more than one class in that region.

A possible solution to this is to stop trying to classify a point as being of a specific class. Instead a numerical scale could be used. One possibility could be mapping the data to a scale similar to the standard 7 point comfort scale, although this could be problematic as the comfort votes from the nodes don't indicate the magnitude of discomfort, only the direction (hot vs cold) of discomfort or if there was no discomfort. The level of discomfort might be estimable based on a measure of the distance between the point being classified and an averaged value of the '`OK`' class. This however does not seem like a robust approach. A better method might be to assign values to the regions of data space based on the proportion or number of comfort votes in that region. This would result in assigning regions where there are many cold votes negative values and regions with hot votes positive values and neutral regions with values close to zero. The magnitude of the value would indicate the confidence that that region was of a particular type of discomfort, rather than how much discomfort that region caused.

Another possibility is a two dimensional approach. The analysis showed that for this dataset two dimensions were generally required to represent the data as one dimension didn't result in good separation of the classes and three dimensions didn't add to the separation. It is possible that attempting to classify comfort on a one dimensional scale is a fundamentally flawed approach and is based upon the mind's simplified and warped perception of reality. One possible two dimensional representation would be one axis representing the proportion of cold labelled points in the region the point is in vs the number of neutral points, and the other axis representing the proportion of hot labelled points vs neutral points. These two axes could be converted from Cartesian coordinates to polar coordinates to give a magnitude of discomfort and a direction indicating the type of discomfort. Unlike the one dimensional scale, this representation would allow the case where occupants are both strongly too hot and too cold. It's not necessarily clear what this would mean for a control system regulating temperature, but the occupant comfort votes have indicated that such regions do exist, so a comfort model that can represent these regions might be more accurate than one that can't.

Bayesian Analysis

The large variation in the number of comfort votes, and thus data points between the different classes for each node, proved problematic for some of the classification methods. One solution to this problem would be to remove points from the classes with a greater number of points until they had the same number of points as the class with the fewest points. This is not ideal as it literally is throwing away information about how the measured parameters relate to comfort. It also removes some information that isn't directly measured by the data points; the relative frequency of different comfort states. It is apparent from the data that the different comfort states are experienced to different extents. This means that, based on past data, some states are more likely than others. This is information that could be used when producing a classifier. Bayes Theorem provides a method to include this information into the classifier.

Bayes Theorem is an equation that provides a method of calculating the probability of a comfort class given a data sample, based on: the probability of that sample if it is indeed of that class; the probability of that class; and the probability of the sample regardless of what class it is [73]. The mathematical form of Bayes Theorem is shown in Equation 8.1. $P(C|S)$ is the probability of a class given a sample, $P(S|C)$ is the probability of the sample given the class, $P(C)$ is the probability of the class, and $P(S)$ is the probability of the sample.

$$P(C|S) = \frac{P(S|C)P(C)}{P(S)} \tag{8.1}$$

The probability of a sample given a class ($P(S|C)$) is calculated as part of the Gaussian Probability classifier discussed in Section 7.5.6. The probability of the class $P(C)$ can be estimated based on the number of data points the class has as compared to the number of data points in all the classes. The probability of the sample $P(S)$ can be found by using the law of total probability to expand $P(S)$ into the sum of probabilities of the sample given each class multiplied by the probability of that class. The expanded equation is shown in Equation 8.2. In the equation C_h refers to the Hot class, C_o refers to the Ok class, C_c refers to the Cold class, and C_x is a variable representing which class you are calculating the probability for. C_x would get substituted with each of C_h, C_o and C_c to calculate the probability of each class given the sample, and then the sample would be classified as the class with the highest probability.

$$P(C_x|S) = \frac{P(S|C_x)P(C_x)}{P(S|C_h)P(C_h) + P(S|C_o)P(C_o) + P(S|C_c)P(C_c)} \tag{8.2}$$

Bayes Theorem can also be used with likeliness rather than probability, which would let the likeliness values calculated as part of the Gaussian Likeliness classifier in Section 7.5.5 be used. To use the likeliness value one would need to substitute $L(S|C)P(S)$ for $P(S|C)$. Performing this substitution in Equation 8.1 allows cancellation of $P(S)$ from numerator and denominator to give the simpler equation Equation 8.3. This is a superior solution as it does not require a decision to be made as to what size area around a point on the Gaussian Likeliness curve to integrate to produce a probability, and simply uses the likeliness at that point.

$$P(C|S) = L(S|C)P(C) \tag{8.3}$$

Frequency Analysis

The main reason for the failure of PCA was that PCA finds orthogonal axes in order of the variation along them, and the most significant variation in the data set was caused by the day and night cycle. Thus the data trained on the first few most significant axes produced by PCA was essentially being trained on data representative of the time of day, which isn't directly indicative of any comfort state. The large scale changes in sensor readings brought about by the day-night cycle dominated the results to the exclusion of the smaller changes which would be indicative of thermal comfort.

A possible solution to this problem would use Fourier analysis. The problematic effects of the day-night cycle on sensor readings can readily be removed using frequency based techniques. While their effect on the sensor readings is very large, their frequency (around 1 cycle per day) is very low, much lower than the frequency of anything which would be relevant for changes to thermal comfort throughout the day. By processing the data with a high pass filter set to remove these very low frequencies one could eliminate the day-night effect from the data. This might improve classification based on PCA results as the PCA wouldn't "get distracted" by the large scale variations brought about by the day-night cycle.

One problem is that the frequency range of effects that would be relevant to classification of thermal comfort is not obvious. Clearly 1 cycle per day is much too slow, and 1Hz would be much too fast, but where in between these the useful data is would be a topic for further research. The data set produced by this project is sampled at 1 sample per 30 seconds, meaning the upper frequency of effects accurately represented in the data is 1 cycle per minute.

8.3 Discussion

While the project has been successful, there were some unexpected problems. One factor that had not been consciously considered was the variability of people. The main problem this caused was increasing the overlap between class regions for nodes in places where multiple occupants used them. It would be of benefit to future thermal comfort monitoring schemes to keep track of which occupant voted, in addition to when and where. The modern solution to this would be to use an app on a smart phone, but there are some problems doing that. One of the main reasons for having the user interface on the front of the node is it minimizes the effort an occupant needs to expend to interact with the system and takes advantage of the fact that people like pressing buttons. There is essentially zero gain to interacting with the system for an occupant, so the cost (in terms of effort or otherwise) needs to be comparable. A smartphone app requires an occupant to have a smartphone, download and install an app, remember to use the app repeatedly and possibly set up a user account. A physical box on the wall with blinking lights serves as a reminder to occupants to vote, but only at a time when it's most convenient for them to and when it costs them nothing to do so i.e. when they're standing right next to it. Finding a suitable user interface is an essential step in producing a monitoring system that can produce data for optimal thermal comfort classification. This situation does change somewhat if the data collection is actively used in real time by a control system regulating the environment, but only if the interaction has some noticeable effect soon after the interaction occurred. This is because there is now a noticeable benefit to the occupant to interacting with the system, so therefore some small cost is acceptable too.

Another problem was the building used for the deployment area. It is an old building with thick walls and so has a very high thermal mass as compared to modern buildings. This meant that the conditions inside did not vary hugely. Additionally it has a heating system. These two facts meant that the occupants weren't exposed to much variety of conditions while being surveyed, and they didn't experience many cold conditions because the heating system was working as it was supposed to. The plan this project used was to simply leave the sensor nodes in the test area and record the votes as a variety of conditions occurred over time. The limited variation of conditions in the building worked against this plan. Indeed it works against any system that intends to learn an occupant's comfort preferences by monitoring them in an environment that's already regulated or that naturally doesn't vary much. For a system to learn what conditions an occupant doesn't like, it needs those conditions to occur. This may mean that a system that learns occupant preferences would have a training period where it subjects occupants to a variety of conditions or it would be pre-programmed with some average results of some test subjects, which it then adapts based on the occupant's feedback.

The issue of high thermal mass would have been resolved had the second deployment of nodes in other buildings gone ahead, though conditions in those buildings would still depend on how well their HVAC systems regulated their environment. While monitoring an old building made producing classifiers challenging, trying to avoid the problem by monitoring newer buildings isn't the solution. The reason for this is that retrofitting advanced heating control systems in old buildings is one of the plans for reducing energy usage to combat climate change. Solving the classification problem in the more challenging environment of an old building is valuable because it's the harder problem to solve, and because it's the case that needs to be solved. Having sensor nodes in newer buildings is still beneficial however. It would provide a reference for comparison, and would allow the production of classifiers which can be shown to be able to work independent of building type.

It was originally planned that a control system would be produced that regulated the heating based on occupant feedback. This control system was not produced due to time constraints and concerns that localised control based on adjusting radiator valves may result in a catastrophic over-pressure in the building's very old boiler. An ironic result of this is that while the project's main motivation was to reduce energy usage, it has lead to an increase in energy usage due to operating a server and the sensor nodes for over three and half years, but made no other changes to the operation of the building.

While the system worked reliably, there were periods where data was not recorded. These were due to power cuts, network failures, or third party interference with the nodes. There is a small amount of non-volatile storage on the nodes both in the microcontroller and in the external flash memory module that contains the node's MAC address. There are a couple of reasons this was not used to store samples for later transmission during network outages. Firstly, the size of the memory available and the size of a measurement sample mean that not many samples can be stored. It was expected that the network would be reliable, and that if it were to fail then the failure would be of a serious nature and the time it would take to fix would be longer than the number of samples the node could store. In such an event storing a few more samples wasn't expected to impact the study much.

The second more important reason is related to confidence in the sampling timing. While effort was put into making the node's internal time keeping be accurate; it was felt that trust shouldn't be placed in it as there may be some unforeseen bug that causes an error in the node's time if it was disconnected from the server for a long duration. In such an event there isn't any way to check that the samples a node sends when it reestablishes contact with the server are actually from the time the node claims they are from. By not storing samples for later sending it is ensured that when the node sends a sample the longest it has been since it updated its clock from the server is 5 minutes — the time out before a node resets due to no contact with the server. There was sufficient confidence in the node's time keeping over that short period of time that there would be no samples with erroneous timestamps inserted into the database. The preference was to lose some samples instead of allowing potential invalid samples be collected.

Thirdly continuing to operate for a long period without contact with the server violates the node's error recovery principle of resetting whenever any error condition is detected. The concept of storing data for future transmission is not without merit however. For a future data collection network it is worth investigating if there is sufficient nonvolatile storage and a trusted time keeping module in the node. A possible solution to allow the system to reset on error conditions but not lose track of time in doing so would be to put the time keeping part of the system in a module external to the main microcontroller.

The system monitoring software worked well overall, but there was one incident where it caused problems. This was a case where some erroneous condition occurred on the server that caused the weather monitoring software to crash. The software that monitors the weather monitoring software detected the crash and started a new instance of the weather monitoring software, but it did so before the previous crashed instance had released its resources. This caused the new instance to crash because it couldn't access these resources. The code didn't handle this error as it was expected that there would only ever be one instance of the weather monitor running. While the new instance crashed, it did successfully send an error email before exiting. The software monitoring the weather monitor detected the new instance had failed so started a new instance, which suffered the same problem as the previous one. Unfortunately the email rate limiting was done by an in memory variable within the weather monitoring process and didn't involve any persistent storage of when the last email was sent. This normally wasn't a problem as the weather monitor usually only started when the server started, and stopped when it shut down. In this case however it was a problem, as each instance of the weather monitor that was started initialised its rate limit variable to allow it to immediately send an email. Each instance that was started would send an email before crashing due to the way the previous instance crashed, and then crash in such a way that it would cause the next instance started to crash.

The problem was identified and fixed, but not before several thousand emails were sent to everyone who was to be informed of a failure of weather data. Worse still, the emails indicated that the weather data was not being sent, when it was in fact being sent correctly and only the monitoring software was at fault. The lessons learned from this were that important rate limiting components should use persistent storage, and that ideally they should be separated from the things that use them. A better version of the system would have had a separate program that was responsible for sending email on behalf of all the other programs. This program would only handle sending emails and rate limiting them using persistent storage of when previous emails were sent. The limited scope of the program would reduce the chance for bugs to be made.

References

[1] US Government. Overview. In National Climate Assessment and Development Advisory Committee (NCADAC), editors, *National Climate Assesment 3*, volume 3 of *National Climate Assesment*. U.S. Global Change Research Program, 2014. URL `http://nca2014.globalchange.gov/downloads`.

[2] David Victor, Dadi Zhou, Essam Hassan, Mohamed Ahmed, Pradeep Kumar Dadhich, Jos Olivier, H-Holger Rogner, Kamel Sheikho, Mitsutsune Yamaguchi, Giovanni Baiocchi, Yacob Mulugetta, Linda Wong, Arnulf Grbler, and Alick Muvundika. Introductory chapter. In *Fifth Assessment Report (AR5) Climate Change 2014: Mitigation of Climate Change*, Fifth Assessment Report, intergovernmental report 1. Intergovernmental Panel on Climate Change (IPCC), 2014. URL `https://www.ipcc.ch/report/ar5/wg3/`.

[3] Oswaldo Lucon, Diana rge Vorsatz, Azni Zain Ahmed, Hashem Akbari, Paolo Bertoldi, Luisa F. Cabeza, Nicholas Eyre, Ashok Gadgil, L. D. Danny Harvey, Yi Jiang, Enoch Liphoto, Sevastianos Mirasgedis, Shuzo Murakami, Jyoti Parikh, Christopher Pyke, Maria Virginia Vilario, Peter Graham, Ksenia Petrichenko, Jiyong Eom, Agnes Kelemen, Volker Krey, Marilyn Brown, Tams Plvlgyi, Fonbeyin HenryAbanda, and Katarina Korytarova. Buildings. In *Fifth Assessment Report (AR5) Climate Change 2014: Mitigation of Climate Change*, Fifth Assessment Report, intergovernmental report 9. Intergovernmental Panel on Climate Change (IPCC), 2014. URL `https://www.ipcc.ch/report/ar5/wg3/`.

[4] Andreas Wagner, Michael Klebe, and Christopher Parker. Monitoring results of a naturally ventilated and passively cooled. *International Journal of Ventilation*, 6 (1):3–20, June 2007. ISSN 1473-3315. URL `http://www.ijovent.org/doi/abs/10.5555/ijov.2007.6.1.3`.

[5] COMMISSION OF THE EUROPEAN COMMUNITIES. Adapting to

climate change: Towards a european framework for action. White paper, COMMISSION OF THE EUROPEAN COMMUNITIES, 2009. URL http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2009:0147:FIN:EN:PDF.

[6] UK Government. The climate change act 2008. Online, November 2008. URL http://www.legislation.gov.uk/ukpga/2008/27/contents.

[7] Department of Energy & Climate Change. Energy consumption in the uk - chapter 1: Overall data tables. Online, Sepyember 2014. URL https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/358232/overall.xls.

[8] ASHRAE. Standard 55 - 2004.

[9] Joost van Hoof, Mitja Mazej, and Jan L. M. Hensen. Thermal comfort: research and practice. *FRONTIERS IN BIOSCIENCE-LANDMARK*, 15:765–788, JAN 2010. ISSN 1093-9946. doi: 10.2741/3645.

[10] Richard De Dear and Gail Schiller Brager. Developing an adaptive model of thermal comfort and preference, final report. Technical report, American Society of Heating, Refrigerating and Air Conditioning Engineers, 1998. URL http://www.cbe.berkeley.edu/research/other-papers/de%20Dear%20-%20Brager%201998%20Developing%20an%20adaptive%20model%20of%20thermal%20comfort%20and%20preference.pdf.

[11] O Seppanen, WJ Fisk, and D Faulkner. Control of temperature for health and productivity in offices. In M Geshwiler, editor, *ASHRAE Transactions 2005, Vol 111, Pt 2*, volume 111 of *ASHRAE TRANSACTIONS*, pages 680–686, 1791 TULLIE CIRCLE NE, ATLANTA, GA 30329 USA, 2005. Amer Soc Heating Refrigerating & Air Conditioning Engineers, AMER SOC HEATING, REFRIGERATING AND AIR-CONDITIONING ENGS. Annual Meeting of the American-Society-of-Heating-Refrigerating-and-Air-Conditioning-Engineers (ASHRAE), Denver, CO, 2005.

[12] *Temperature variation in centrally controlled HVAC buildings and its impact on comfort and performance*, July 2008. BBA Indoor Environmental Consultancy. URL http://nceub.commoncense.info/uploads/W2008_57Boerstra.pdf.

[13] Iso 7730:2005.

[14] Technical Committee CEN/TC 156 “Ventilation for Buildings”. En 15251 indoor environmental input parameters for design and assessment of energy performance of buildings addressing indoor air quality, thermal environment, lighting and acoustics, 2007.

[15] P. O. Fanger. *Thermal Comfort : analysis and applications in environmental engineering*. Danish Technical Press, 1970. ISBN 9788757103410.

[16] Michael A Humphreys and J Fergus Nicol. The validity of iso-pmv for predicting comfort votes in every-day thermal environments. *Energy and Buildings*, 34(6):667 – 684, 2002. ISSN 0378-7788. doi: 10.1016/S0378-7788(02)00018-X. URL `http://www.sciencedirect.com/science/article/pii/S037877880200018X`. ¡ce:title¿Special Issue on Thermal Comfort Standards¡/ce:title¿.

[17] J. Van Hoof. Forty years of fanger’s model of thermal comfort: comfort for all? *Indoor Air*, 18(3):182–201, 2008. ISSN 1600-0668. doi: 10.1111/j.1600-0668.2007.00516.x. URL `http://dx.doi.org/10.1111/j.1600-0668.2007.00516.x`.

[18] Ardeshir Mahdavi and Satish Kumar. Implications of indoor climate control for comfort, energy and environment. *Energy and Buildings*, 24(3):167 – 177, 1996. ISSN 0378-7788. doi: http://dx.doi.org/10.1016/S0378-7788(96)00975-9. URL `http://www.sciencedirect.com/science/article/pii/S0378778896009759`.

[19] JF Nicol and MA Humphreys. Adaptive thermal comfort and sustainable thermal standards for buildings. *ENERGY AND BUILDINGS*, 34(6):563–572, JUL 2002. ISSN 0378-7788. doi: 10.1016/S0378-7788(02)00006-3. Conference on Moving Thermal Comfort Standards into the 21st Century, WINDSOR, ENGLAND, APR 05-08, 2001.

[20] KJ McCartney and JF Nicol. Developing an adaptive control algorithm for europe. *ENERGY AND BUILDINGS*, 34(6):623–635, JUL 2002. ISSN 0378-7788. doi: 10.1016/S0378-7788(02)00013-0. Conference on Moving Thermal Comfort Standards into the 21st Century, WINDSOR, ENGLAND, APR 05-08, 2001.

[21] Richard J De Dear and Gail Schiller Brager. Thermal adaptation in the built environment: a literature review. *Energy and Buildings*, 27(1):83–96, FEB 1998. ISSN 0378-7788. doi: 10.1016/S0378-7788(97)00053-4. URL `http://www.sciencedirect.com/science/article/pii/S0378778897000534`.

[22] J. F. Nicol and M. A. Humphreys. Thermal comfort as part of a self-regulating system. *Building Research and Practice*, 1(3):174–179, 1973. doi: 10.1080/09613217308550237. URL `http://www.tandfonline.com/doi/abs/10.1080/09613217308550237`.

[23] Department of Energy & Climate Change. United kingdom housing energy fact file: 2013. In *Domestic energy fact file and housing surveys*. Department of Energy & Climate Change, 2013. URL `https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/345141/uk_housing_fact_file_2013.pdf`.

[24] R.H. FOX. Heat acclimatisation and the sweating response. In J.L. MONTEITHL.E. MOUNT, editor, *Heat Loss from Animals and Man*, chapter 13, pages 277 – 303. Butterworth-Heinemann, 1974. ISBN 978-0-408-70652-0. doi: http://dx.doi.org/10.1016/B978-0-408-70652-0.50019-6. URL `http://www.sciencedirect.com/science/article/pii/B9780408706520500196`.

[25] Alexia Cardona, Luca Pagani, Tiago Antao, Daniel J. Lawson, Christina A. Eichstaedt, Bryndis Yngvadottir, Ma Than Than Shwe, Joseph Wee, Irene Gallego Romero, Srilakshmi Raj, Mait Metspalu, Richard Villems, Eske Willerslev, Chris Tyler-Smith, Boris A. Malyarchuk, Miroslava V. Derenko, and Toomas Kivisild. Genome-wide analysis of cold adaptation in indigenous siberian populations. *PLoS ONE*, 9(5):e98076, 05 2014. doi: 10.1371/journal.pone.0098076. URL `http://dx.doi.org/10.1371%2Fjournal.pone.0098076`.

[26] Sami Karjalainen and Avi Koistinen. User problems with individual temperature control in offices. *BUILDING AND ENVIRONMENT*, 42(8):2880–2887, AUG 2007. ISSN 0360-1323. doi: 10.1016/j.buildenv.2006.10.031. URL `http://www.sciencedirect.com/science/article/pii/S0360132306003349`.

[27] Sami Karjalainen. Thermal comfort and use of thermostats in finnish homes and offices. *Building and Environment*, 44(6):1237 – 1245, 2009. ISSN 0360-1323. doi: http://dx.doi.org/10.1016/j.buildenv.2008.09.002. URL `http://www.sciencedirect.com/science/article/pii/S0360132308002266`.

[28] S. Atthajariyakul and T. Leephakpreeda. Neural computing thermal comfort index for {HVAC} systems. *Energy Conversion and Management*, 46 (1516):2553 – 2565, 2005. ISSN 0196-8904. doi: http://dx.doi.org/10.1016/j.enconman.2004.12.007. URL `http://www.sciencedirect.com/science/article/pii/S0196890405000166`.

[29] B Von Neida, D Maniccia, and A Tweed. An analysis of the energy and cost savings potential of occupancy sensors for commercial lighting systems. *JOURNAL OF THE ILLUMINATING ENGINEERING SOCIETY*, 30(2):111+, SUM 2001. ISSN 0099-4480.

[30] Energy efficiency through occupancy detection. URL `http://www.floridaenergy.ufl.edu/wp-content/uploads/S3-Energy-Efficiency-Mortellaro-Energy_Efficiency_through_Occupancy_Detection.pdf`.

[31] R. Melfi, B. Rosenblum, B. Nordman, and K. Christensen. Measuring building occupancy using existing network infrastructure. In *Proceedings of the 2011 International Green Computing Conference and Workshops*, IGCC '11, pages 1–8, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-1-4577-1222-7. doi: 10.1109/IGCC.2011.6008560. URL `http://dx.doi.org/10.1109/IGCC.2011.6008560`.

[32] Nest Labs. What is auto-schedule and how does it learn? Online, 3 2015. URL `https://nest.com/support/article/How-does-Auto-Schedule-learn`.

[33] Kris Sangani. The heat is on. *Engineering & Technology*, 9(7):49–51, August 2014.

[34] Essa Jafer, Rostislav Spinar, Paul Stack, Cian OMathuna, and Dirk Pesch. Wireless sensor network deployment for building environmental monitoring and control. *Ambient Media and Systems*, 70:20–27, 2011.

[35] Won-Suk Jang, William M. Healy, and Mirosaw J. Skibniewski. Wireless sensor networks as part of a web-based building environmental monitoring system. *Automation in Construction*, 17:729 – 736, 2008. Issue 6.

[36] William M Healy. Lessons learned in wireless monitoring. *ASHRAE journal*, 47 (10):54, 2005.

[37] M Kintner-Meyer, Michael R Brambley, Teresa A Carlon, and Nathan N Bauman. Wireless sensors: technology and cost-savings for commercial buildings. *Teaming for Efficiency: Proceedings, 2002 ACEEE Summer Study on Energy Efficiency in Buildings*, 7:7–121, 2002.

[38] Farrokh Jazizadeh, Ali Ghahramani, Burcin Becerik-Gerber A.M. ASCE, Tatiana Kichkaylo, and Michael Orosz. A human-building interaction framework for personalized thermal comfort driven systems in office buildings. *Journal of Computing in Civil Engineering*, 2013.

[39] Mitja Kosir, Ales Krainer, and Ziva Kristl. Integral control system of indoor environment in continuously occupied spaces. *AUTOMATION IN CONSTRUCTION*, 21:199–209, JAN 2012. ISSN 0926-5805. doi: 10.1016/j.autcon.2011.06.004.

[40] Wai Leung Tse and Wai Lok Chan. A distributed sensor network for measurement of human thermal comfort feelings. *SENSORS AND ACTUATORS A-PHYSICAL*, 144(2):394–402, JUN 15 2008. ISSN 0924-4247. doi: 10.1016/j.sna.2008.02.004.

[41] Jeongho Kang, Yongduk Kim, Hyungpyo Kim, Junwhan Jeong, and Sekwang Park. Comfort sensing system for indoor environment. In *Solid State Sensors and Actuators, 1997. TRANSDUCERS '97 Chicago., 1997 International Conference on*, volume 1, pages 311–314 vol.1, Jun 1997. doi: 10.1109/SENSOR.1997.613646.

[42] System and method for climate control set-point optimization based on individual comfort, October 2010.

[43] Povl Ole Fanger and Technical Committee CEN/TC 156. Cen cr 1752 - ventilation for buildings - european design criteria for the indoor environment. Technical report, European Committee for Standardization, 1998.

[44] RobertE. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990. ISSN 0885-6125. doi: 10.1007/BF00116037. URL `http://dx.doi.org/10.1007/BF00116037`.

[45] Microchip. Pic18f97j60 family data sheet, July 2011. URL `http://ww1.microchip.com/downloads/en/DeviceDoc/39762f.pdf`.

[46] TE Connectivity. Mag45 modular jacks with integrated magnetics. Online, June 2009. URL www.te.com/commerce/DocumentDelivery/DDEController?Action=showdoc&DocId=Catalog+Section%7F82066_Mod_Jacks_and_MRJ21_Prod_-_MAG45%7F1107%7Fpdf%7FEnglish%7FENG_CS_82066_Mod_Jacks_and_MRJ21_Prod_-_MAG45_1107.pdf%7F1-6605834-1.

[47] Sensirion. Datasheet sht1x, December 2011. URL http://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/Humidity/Sensirion_Humidity_SHT1x_Datasheet_V5.pdf.

[48] Sensonor. Sp100 series compensated pressure sensors datasheet, June 2009. URL http://www.sensonor.com/media/30318/sp100-7(t),%20sp100-7a(t),%20sp100-12a%20datasheet.pdf.

[49] Honeywell. Hih-5030/5031 series low voltage humidity sensors datasheet, March 2010. URL http://sensing.honeywell.com/honeywell-sensing-hih5030-5031%20series-product-sheet-009050-2-en.pdf.

[50] Freescale Semiconductor. Mpxa6115a series datasheet, October 2012. URL http://www.freescale.com/files/sensors/doc/data_sheet/MPXA6115A.pdf.

[51] TAOS. Tsl252 datasheet, September 2007. URL http://www.ams.com/eng/content/download/250424/976445/file/TSL252_Datasheet_EN_v1.pdf.

[52] Panasonic. Panasonic amn34112. Online. URL http://www3.panasonic.biz/ac/ae/search_num/index.jsp?c=detail&part_no=AMN34112.

[53] Paul Horowitz and Winfield Hill. *The Art of Electronics*. Cambridge University Press, 2nd edition, 1989. ISBN:0-521-37095-7.

[54] IEEE. Ieee std 802.3af-2003, amendment to ieee std 802.3-2002, data terminal equipment (dte) power via media dependent interface (mdi), June 2003. URL http://standards.ieee.org/getieee802/download/802.3af-2003.pdf.

[55] J. Postel. User datagram protocol. RFC 768 (INTERNET STANDARD), August 1980. URL http://www.ietf.org/rfc/rfc768.txt.

[56] *The Unix Programming Manual*. Bell Telephone Laboratories, Bell Telephone Laboratories, Incorporated Murray Hill, New Jersey, 7 edition, January 1970. URL `http://plan9.bell-labs.com/7thEdMan/v7vol1.pdf`.

[57] Postgresql website. Online. URL `http://www.postgresql.org/`.

[58] Encyclopaedia Britannica Online. e-mail. Online, April 2014. URL `http://www.britannica.com/EBchecked/topic/183816/e-mail`.

[59] magic number. Online, April 2014. URL `http://www.catb.org/jargon/html/M/magic-number.html`.

[60] W.J. Croft and J. Gilmore. Bootstrap protocol. RFC 951 (Draft Standard), September 1985. URL `http://www.ietf.org/rfc/rfc951.txt`. Updated by RFCs 1395, 1497, 1532, 1542, 5494.

[61] THE INTERNATIONAL TELEGRAPH and TELEPHONE CONSULTATIVE COMMITTEE (CCITT). G.810 : Definitions and terminology for synchronization networks. Published in the Blue Book, Fascicle III.5 (1989), November 1988. URL `https://www.itu.int/rec/T-REC-G.810/en`.

[62] *Real-time operating system timing jitter and its impact on motor control*, volume 4563, National Institute of Standards and Technology Building 220, Room B124 Gaithersburg, MD 20899-0001, October 2001. National Institute of Standards and Technology (NIST), SPIE. doi: 10.1117/12.452653. URL `http://dx.doi.org/10.1117/12.452653`.

[63] Nilesh Rajbhart. The microchip tcp/ip stack. Aplication Note AN833, Microchip, 2002. URL `http://ww1.microchip.com/downloads/en/AppNotes/00833b.pdf`.

[64] Ralph Droms. Dynamic host configuration protocol. RFC 2131 (INTERNET STANDARD), March 1997. URL `http://tools.ietf.org/html/rfc2131`.

[65] Glenn Fowler, Landon Curt Noll, , and Phong Vo. Fowler / noll / vo (fnv) hash home page. URL `http://isthe.com/chongo/tech/comp/fnv/`.

[66] IEEE. Eee standard for local and metropolitan area networks: Overview and architecture, February 2002. URL `http://standards.ieee.org/getieee802/download/802-2001.pdf`.

[67] L. Eriksson, T. Byrne, E. Johansson, J. Trygg, and C. Wikstrm. *Multi- and Megavariate Data Analysis*. MKS Umetrics, third revised edition edition, 2013. ISBN -10: 91-973730-5-2.

[68] Barry M. Wise, Jeremy M. Shaver, Neal B. Gallagher, Willem Windig, Rasmus Bro, and R. Scott Koch. *PLS_Toolbox 3.5 Manual*. Eigenvector Research, Inc., Eigenvector Research, Inc., 830 Wapato Lake Road, Manson, WA 98831 USA, 2004. URL `www.eigenvector.com`. ISBN: 0-9761184-0-8.

[69] William W. Cooley and Paul R. Lohnes. *Multivariate data analysis*. John Wiley & Sons Inc, 1971.

[70] Michele Banko and Eric Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics (ACL 2001)*, pages 26–33. Association for Computational Linguistics, 2001. URL `http://nlp.cs.swarthmore.edu/~richardw/papers/P01-1005.pdf`.

[71] A. Halevy, P. Norvig, and F. Pereira. The unreasonable effectiveness of data. *Intelligent Systems, IEEE*, 24(2):8–12, March 2009. ISSN 1541-1672. doi: 10.1109/MIS.2009.36.

[72] B. R. Kowalski and C. F. Bender. K-nearest neighbor classification rule (pattern recognition) applied to nuclear magnetic resonance spectral interpretation. *Analytical Chemistry*, 44(8):1405–1411, 1972. doi: 10.1021/ac60316a008. URL `http://dx.doi.org/10.1021/ac60316a008`.

[73] Computational statistics. Online. URL `http://www.eng.utah.edu/~cs5961/Resources/bayes.pdf`.

Appendix A

Communication Protocol Message Specification

OK

The ‘OK’ message described in Table A.1 is a message the server sends in response to messages from nodes. The ‘OK’ message serves several purposes. Firstly it serves as an acknowledgement of reception of a message. The second purpose of the message is maintaining timing and synchronisation of the nodes. The ‘OK’ message includes the time it was sent according the server as well as the next time measurements should be taken.

Bit offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Message version = 0x01								Magic value = 0x6D								Magic value = 0x61								Magic value = 0x67							
32	Magic value = 0x69								Magic value = 0x63								Message Type OK = 0x01								Current clock tick - Low byte							
64	Current clock tick - Middle byte								Current clock tick - High byte								Current clock epoch								Next measurement time tick - Low byte							
96	Next measurement time tick - Middle byte								Next measurement time tick - High byte								Next measurement time epoch															

TABLE A.1: OK message specification

Who are you?

The ‘Who are You?’ message contains essentially the same information as the ‘OK’ message — as can be seen comparing Table A.1 and Table A.2. It is different in the respect that it requires a node to reply to it upon reception with an ‘Introduction’ message.

Bit offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Message version = 0x01								Magic value = 0x6D								Magic value = 0x61								Magic value = 0x67							
32	Magic value = 0x69								Magic value = 0x63								Message Type Who Are You? = 0x02								Current clock tick - Low byte							
64	Current clock tick - Middle byte								Current clock tick - High byte								Current clock epoch								Next measurement time tick - Low byte							
96	Next measurement time tick - Middle byte								Next measurement time tick - High byte								Next measurement time epoch															

TABLE A.2: Who Are You? message specification

Introduction

The ‘Introduction’ message is sent by a node to describe its sensors to the server. The message consists of a header described in Table A.3 and then a description of the sensors on the node.

Bit offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Magic value = 0x6D								Magic value = 0x61								Magic value = 0x67								Magic value = 0x69							
32	Magic value = 0x63								Message version = 0x01								Message version = 0x00								Message Type Introduction = 0x02							
64	NodeID = MAC address of the node extended to EUI-64 bit address																															
96	(Low bytes first, high bytes last)																															
128	Number of sensors - Low Byte								Number of sensors - High byte																							

TABLE A.3: Introduction message header specification

The sensor descriptions follow the header and are formatted as follows:
For each sensor the following is appended to the message:

- The ‘SensorID’ an integer identifying the sensor on the node that sent this message— stored as a 16-bit unsigned value.
- The ‘Sensor Type’ which identifies the type of sensor — stored in a 16-bit wide field as an unsigned integer.
- The size in bytes of the data type used for this sensors readings — stored in a 8-bit field.
- The ‘Data Type’ — stored in an 8-bit wide field.

The values for ‘Sensor Type’ and ‘Data Type’ are described in Table A.4a and Table A.4b respectively.

The last two 8-bit wide fields — the size in bytes of the sensor data and the ‘Data Type’ — can be thought of as a single 16-bit field describing the data type. Thus an IEEE754 single precision floating point number would be described as 0x0402 as it has four bytes and the floating point type is assigned the value of ‘2’ in the table of types Table A.4b.

Value	Sensor Name
1	SHT7X_Temperature
2	SHT7X_Humidity
3	PIR
4	Mic
5	LDR
6	ThermoCouple
7	AnHumidity
8	AnPressure
9	LinearLight
10	SP100T_Temperature
11	SP100T_Pressure

(A) Table of current 'Sensor Type's

Value	Data Type
0	Unsigned Integer
1	Signed Integer
2	Floating Point

(B) Table of current 'Data Type's

TABLE A.4: Tables relating 'Sensor Type's and 'Data Type's to values representing them in messages

The size in bytes field is used in creation of the database table to store that particular sensor's readings. The system currently does not use any variable length data types. If a variable length data type were to be used then the value specified here should be the maximum length of that data type. If the database being used does not require the maximum size of variable length data be specified then this field may contain any value. Though it would be sensible to either set it to the maximum anticipated value, or set it to 0 to indicate that it should be ignored.

Sensor readings

The 'Sensor Readings' message contains the measured values of a node's sensors. These are preceded by a header described in Table A.5.

Bit offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Magic value = 0x6D								Magic value = 0x61								Magic value = 0x67								Magic value = 0x69							
32	Magic value = 0x63								Message version = 0x01								Message version = 0x00								Message Type Sensor Readings = 0x01							
64	NodeID = MAC address of the node extended to EUI-64 bit address (Low bytes first, high bytes last)																															
96																																
128	Measurement tick - Low byte								Measurement tick - Middle byte								Measurement tick - High byte								Measurement epoch							
160	Number of readings - Low Byte								Number of readings - High byte																							

TABLE A.5: Sensor Readings message header specification

The header specified in Table A.5 is followed by a number of sensor readings. The number of readings is specified in the header. The format of the readings is as follows: For each sensor the following is appended to the message:

- The ‘SensorID’ — A 16-bit unsigned integer identifying the sensor.
- The ‘Reading Length’ — A 16-bit unsigned integer which specifies the length of the value reading in bytes.
- The sensor reading — This is of length [Reading Length] and in a format specified in the introduction message.

Appendix B

Circuit Diagrams

Not shown are the 13 100nF decoupling capacitors between VCC and GND: CP1-CP3, CP5-CP10, CP12, CP13, CP15 and CP16.

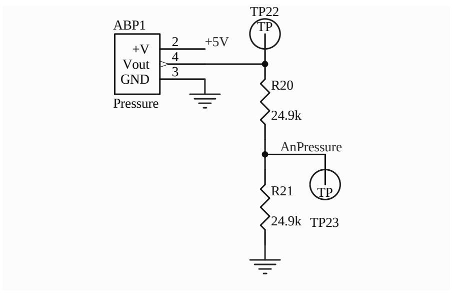


FIGURE B.1: Analogue Pressure Sensor Circuit Diagram

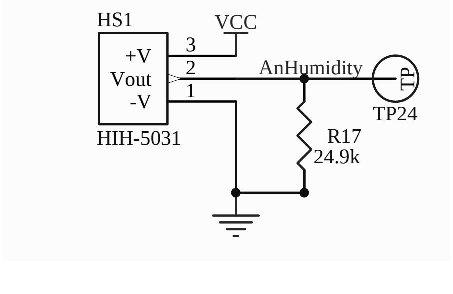


FIGURE B.2: Analogue Humidity Sensor Circuit Diagram

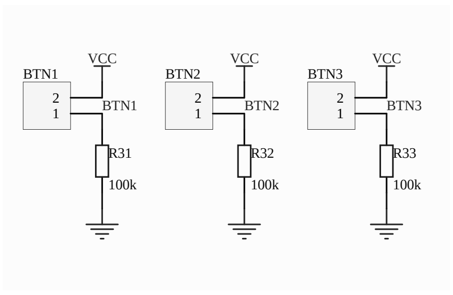


FIGURE B.3: Button Circuit Diagram

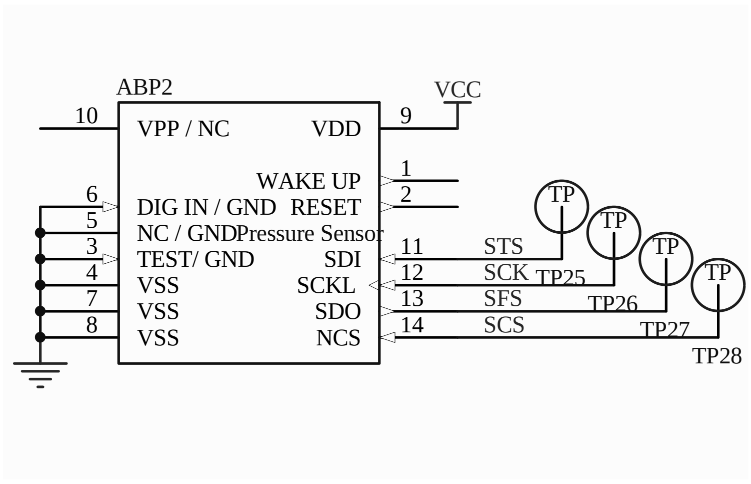


FIGURE B.4: Digital Pressure Sensor Circuit Diagram

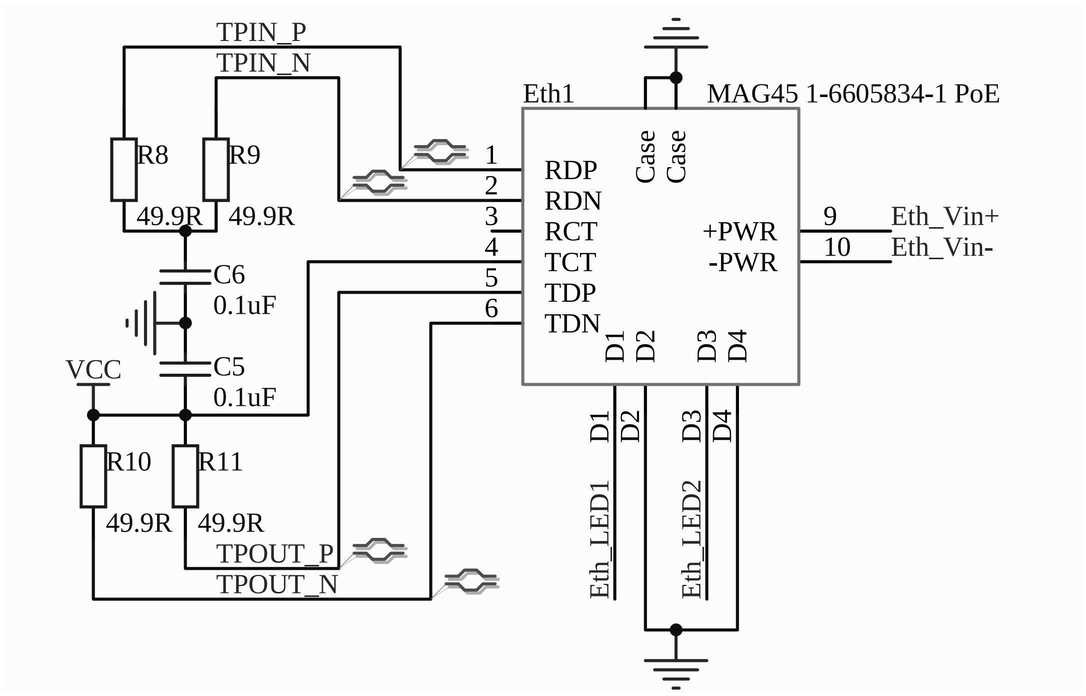


FIGURE B.5: Ethernet Circuit Diagram

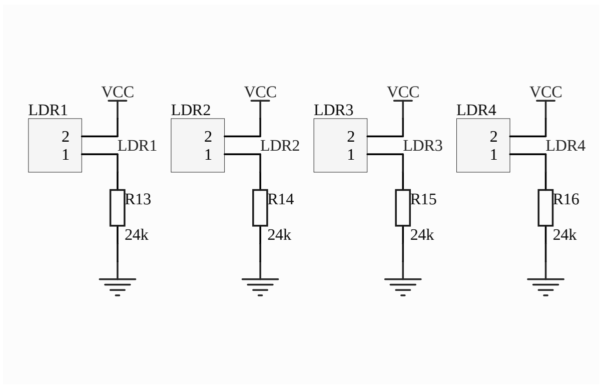


FIGURE B.6: LDR Sensor Circuit Diagram

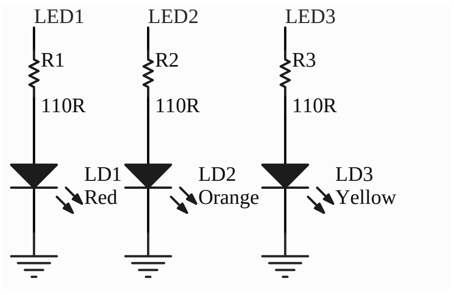


FIGURE B.7: LED Circuit Diagram

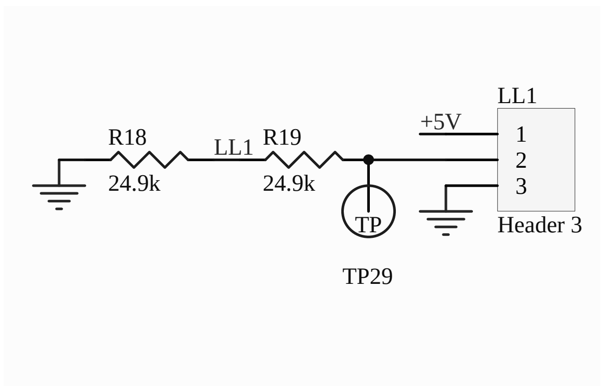


FIGURE B.8: Linear Light Sensor Circuit Diagram

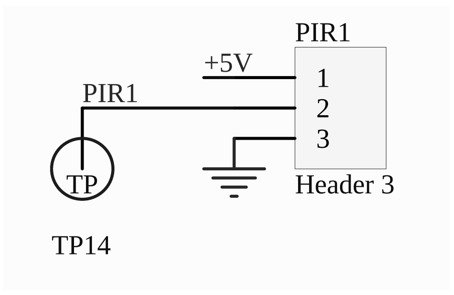


FIGURE B.9: PIR Sensor Circuit Diagram

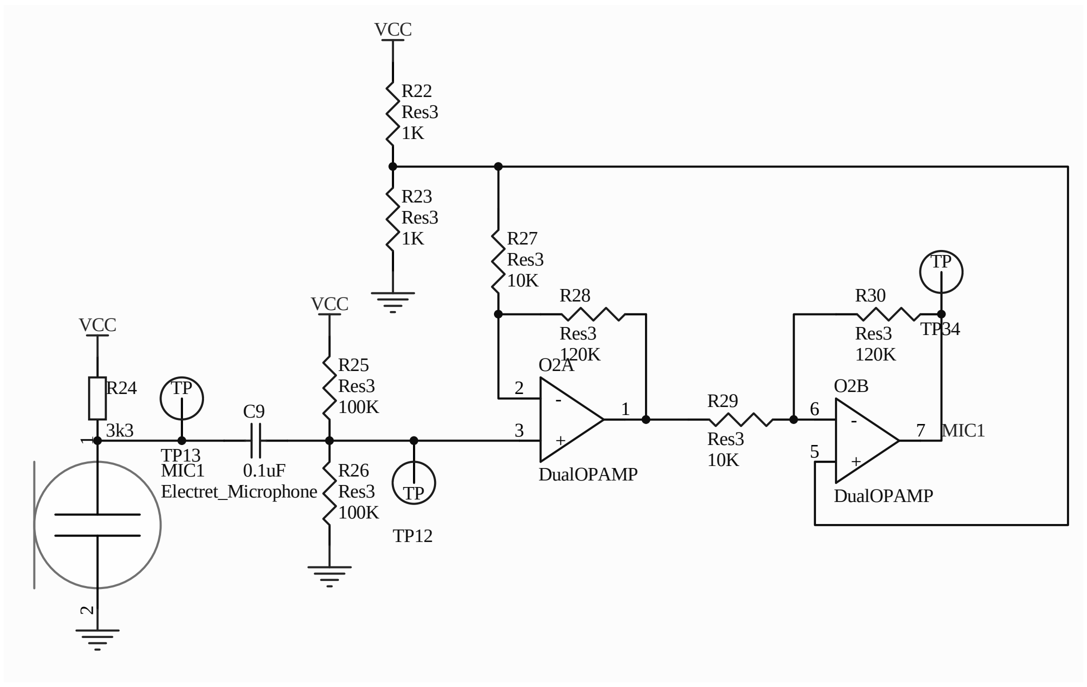


FIGURE B.10: Microphone Circuit Diagram

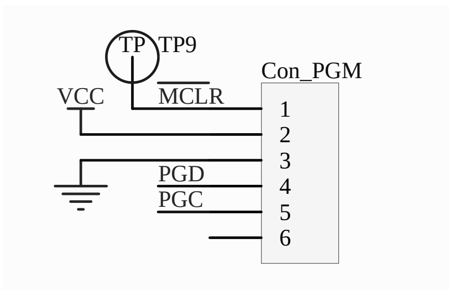


FIGURE B.11: Programming Connection Circuit Diagram

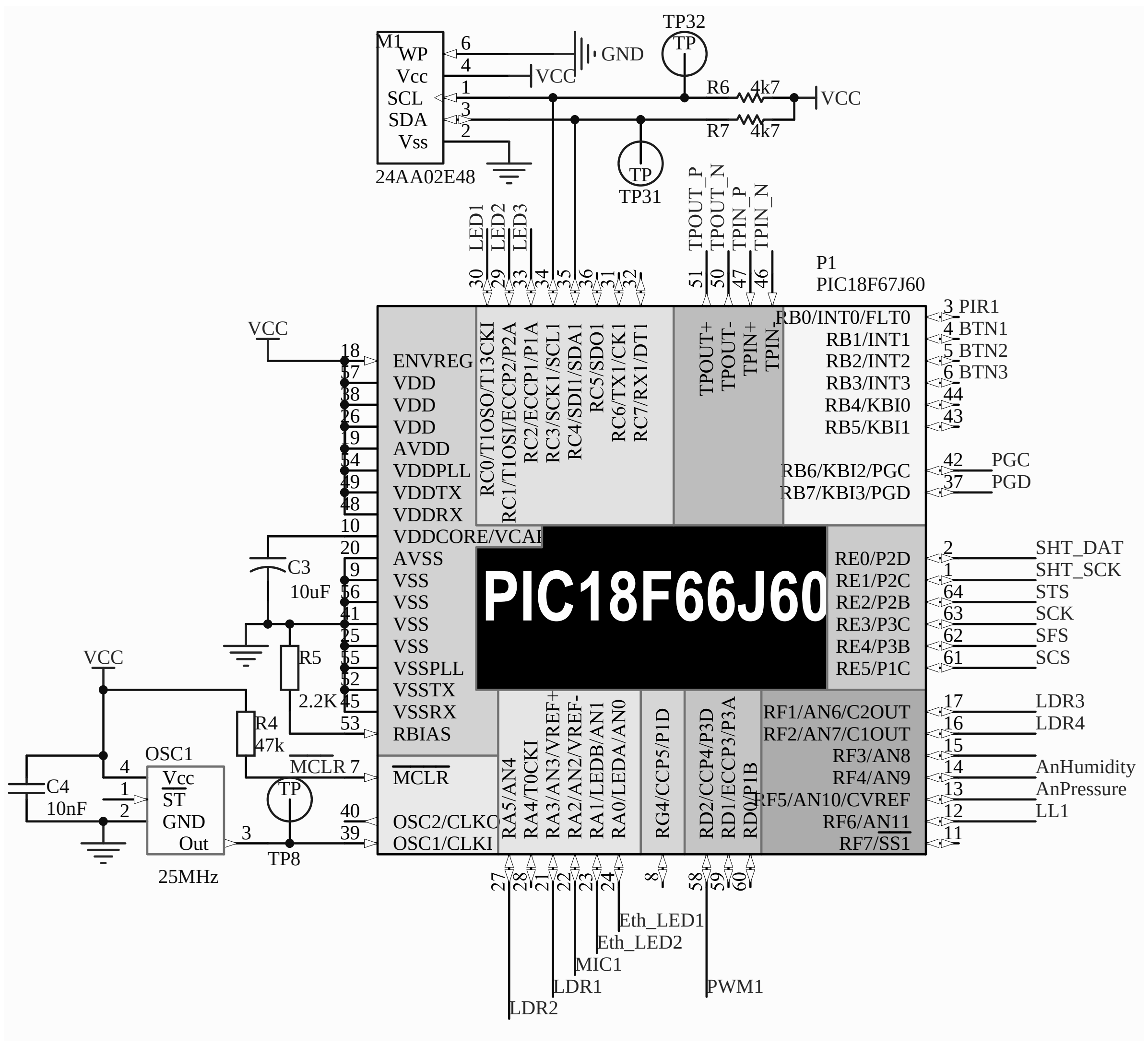


FIGURE B.12: Microcontroller Circuit Diagram

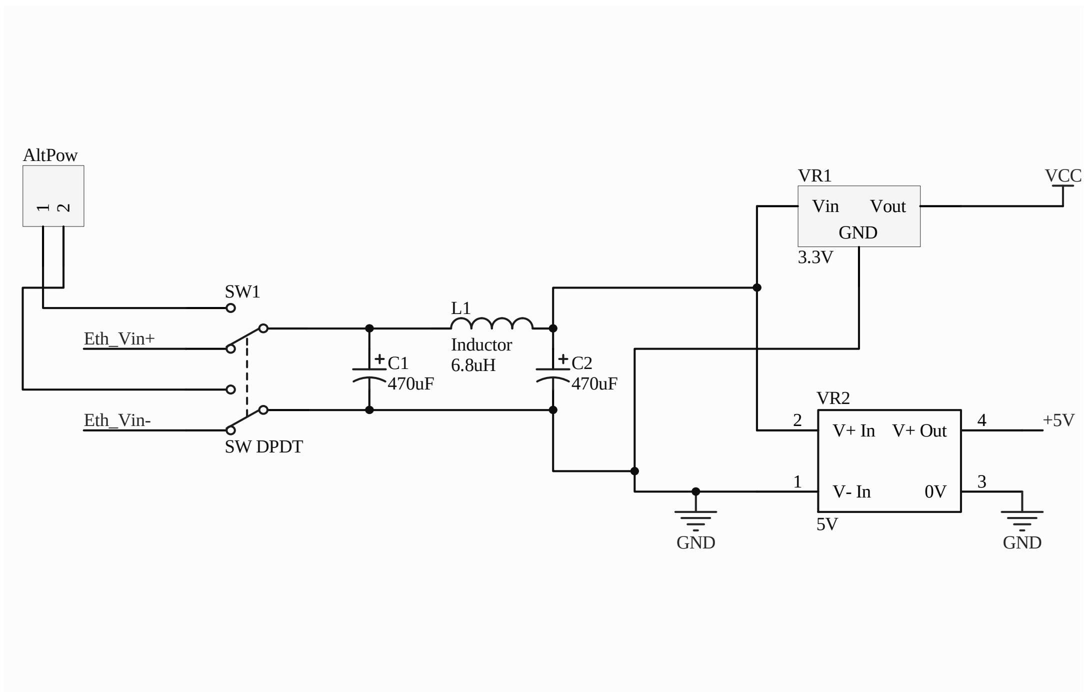


FIGURE B.13: Power Input and Voltage Regulator Circuit Diagram

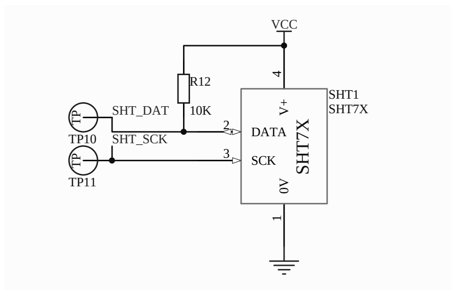


FIGURE B.14: SHT7x Digital Humidity and Temperature Sensor Circuit Diagram

Appendix C

Digital Data

Anonymized versions of the data collected by the sensor nodes are exported from the database every Sunday. These archives are available at the following URLs at time of print:
`https://www.dropbox.com/sh/fd9htrr0ch30dvi/AAAGnrnPxf8Q1SjTO2bTQiDka`
`https://drive.google.com/folderview?id=0B-yrHbj51xfgRzhjbkFDb1p2eFk`

The URL for the project website is:
`http://www.eee.manchester.ac.uk/envmon`

Included with the physical version of this thesis is a DVD containing source code, hardware designs and sensor data.

The meteorological data used in this project was collected by the Whitworth Meteorological Observatory. This data may be obtained from:
`http://www.cas.manchester.ac.uk/restools/whitworth/`.

Appendix D

List of Variables and Acronyms

D.1 Variables

ID	Name	Location
0	SHT7X_Temperature	PCB
1	SHT7X_Humidity	PCB
2	PIR1	Front Panel, Facing Front
3	Mic1Avg	PCB
4	Mic1Peak	PCB
5	LDR1	Top Face, Facing UP
6	LDR2	Right Face, Facing Right
7	LDR3	Left Face, Facing Left
8	LDR4	Front Panel, Facing Front
9	AnHumidity	PCB
10	AnPressure	PCB
11	LinearLight1	Top Face, Facing Front
12	SP100TTemperature	PCB
13	SP100TPressure	PCB
14	SW1 (Too Hot votes)	Front Panel
15	SW2 (Just Fine votes)	Front Panel
16	SW3 (Too Cold votes)	Front Panel

TABLE D.1: Table of variables the Ethernet Nodes measure and send send to the server, the sensor ID numbers assigned to them and a description of the location of the physical sensor they are derived from. The location describes where on the node the sensor is and the direction the sensor is facing if relevant. The description of which face of the case a physical sensor is mounted on assumes one is looking at the front face of the Ethernet Node and the face with the Ethernet port is the bottom face.

SW1, SW2 and SW3 contain a count of how many times their linked buttons have been pressed since the last transmission of measurements. Note on the SW1 is linked to to BTN3 on the PCB which is connected to the ‘Too Hot’ button; SW2 is linked to

BTN2 which is connected to the ‘Just Fine’ button; and SW3 is linked to BTN1 which is connected to the ‘Too Cold’ Button.

D.2 Acronyms

6LoWPAN	IPv6 over low power WPAN
AC	Alternating Current
ACS	Adaptive Comfort Standard
ADC	Analog-to-Digital Converter
ARP	Address Resolution Protocol
ASHRAE	American Society of Heating, Refrigerating, and Air-Conditioning Engineers
ASIC	Application-Specific Integrated Circuit
BMS	Building Management System
BPM	Building Performance Monitoring
CNC	Computer Numerical Control
CPU	Central Processing Unit
CV	Cross Validation
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DRS	Data Receiver Server
DST	Daylight Saving Time
EEPROM	Electrically Erasable Programmable Read-Only Memory
EPSRC	Engineering and Physical Sciences Research Council
FNV	Fowler-Noll-Vo
FSM	Finite State Machine
GUI	Graphical User Interface
HBI-TC	Human-Building Interaction for Thermal Comfort
HVAC	Heating Ventilation and Air Conditioning
IC	Integrated Circuit
ICsIE	Integral Control system of Indoor Environment
ID	Identity
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
ISO	International Organization for Standardization
kNN	k-Nearest Neighbour
LAN	Local Area Network

LDA	Linear Discriminant Analysis
LDR	Light Dependant Resistor
LED	Light Emitting Diode
LOOCV	Leave One Out CV
LpO CV	Leave-p-Out CV
MAC	Media Access Control
NIPALS	Non-linear Iterative Partial Least Squares
OS	Operating System
PC	Personal Computer
PC	Principal Component
PCA	Principal Component Analysis
PCB	Printed Circuit Board
PI	Proportional-Integral
PIR	Passive Infrared
PLC	Programmable Logic Controller
PMV	Predicted Mean Vote
PoE	Power over Ethernet
PPD	Predicted Percentage Dissatisfied
PSI	Pounds per Square Inch
QR (code)	Quick Response (code)
RGB	Red Green Blue
RH	Relative Humidity
(R)MII	(Reduced) Media-Independent Interface
SD	Standard Deviation
SMD	Surface-Mounted Device
SQL	Structured Query Language
SVD	Singular Value Decomposition
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator
USB	Universal Serial Bus
UTC	Coordinated Universal Time
WAN	Wide Area Network
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network