Towards a Low-Cost, Non-Invasive System for Occupancy Detection using a Thermal Detector Array

Ash Tyndall

This report is submitted as partial fulfilment
of the requirements for the Honours Programme of the
School of Computer Science and Software Engineering,
The University of Western Australia,
2015

Abstract

With the increasing inter-networked and inexpensive nature of embedded systems and systems, occupancy sensing, the detection of the presence and number of people in a given space, is becoming a cost-effective area of research. Knowing the number of occupants in a space can help reduce energy consumption and greenhouse gas emissions in both small homes and in large office buildings through more efficient climate control. The goal of this project was to develop an occupant sensing system that is low-cost, non-invasive, reliable and energy efficient.

After examining the available sensing options, we concluded that a low-resolution Thermal Detector Array (lrTDA) would be the most appropriate sensor for our project, as it has non-invasiveness advantages. The key paper in that area, termed Thermosense, used an lrTDA in combination with image subtraction, feature extraction and machine learning classification algorithms to make occupancy predictions with a Root Mean Squared Error (RMSE) of only 0.35 occupants, at an estimated cost of $170. However, due to component availability issues, ThermoSense could not be directly replicated in the Australian market.

We designed our own sensing system for $185 based using an Arduino, a Raspberry Pi, and a different lrTDA (the MLX90620), which had a narrower, rectangular field of view. This system consumed approximately 51 mA of power when active. We also developed the Thermal Array Library (TArL), a software library that implemented ThermoSense's occupant detection algorithm. We then investigated ThermoSense's classification methods, which used numeric Multi-Layer Perceptron, k-Nearest Neighbors and Linear Regression algorithms and found our results differed from theirs significantly in both RMSE and correlation, with both lower. These results suggest that classifiers used may be very sensitive to the specific sensor's properties.

After further experimentation with our own suite of nominal classification algorithms, we found an approach (K*) that achieved an RMSE of 0.304 and a precision of 82%, which improves upon ThermoSense's results. We reflected upon our four criteria, and with our choice of sensor, falling component costs, our accuracy results and power consumption statistics, we concluded that our sensing system prototype met our defined criteria.

Keywords: Energy Saving, Internet of Things, Low-Cost Sensing, Non-Invasive Sensing, Occupancy Sensing, Smart Homes, Thermal Sensing
CR Categories: J.7, C.3

Acknowledgements

Writing this dissertation has been my first substantial foray into academic research, and I could not have done it without the assistance and support of so many people.

Firstly, my supervisors Professor Rachel Cardell-Oliver and Professor Adrian Keating have been fundamental to this project's success. Rachel's ability to steer me away from completely redefining the project's parameters at every setback has been of critical importance, as has been her encouragement and computer science advice. Similarly, without Adrian's enviable knowledge of electronics, his skill in constructing prototypes, and his valuable perspective on thermal sensing, this project could not have happened.

Secondly, I wish to acknowledge the generous support of the University, the University Alumni, and the Hackett Foundation through the Hackett Foundation Alumni Honours Scholarship, which I was awarded at the beginning of my project. The financial support this scholarship provided has allowed me to commit more of my time to my research over working, and I can safely say that without it, my research would be of a much poorer quality.

Thirdly, to my friends and family, who have provided support and encouragement throughout this project, thank you. I cannot imagine where I would be right now without them.

Additionally, I would like to acknowledge Alex Beltran, Varick L. Erickson and Alberto E. Cerpa, the authors of ThermoSense [7]. Without their groundwork on thermal occupancy sensing, nothing in this dissertation would be possible.

Finally, I would like to acknowledge Linda Salzman Sagan, "Tompw" and "Holek", the authors of `http://commons.wikimedia.org/wiki/File:Human_outline.svg`, which I have adapted for Figure 4.6.

Contents

Abstract **ii**

Acknowledgements **iv**

1 Introduction **1**

2 Literature Review **5**

2.1 Intrinsic traits 6

2.1.1 Static traits 6

2.1.2 Dynamic traits 8

2.2 Extrinsic traits 8

2.2.1 Instrumented traits 8

2.2.2 Correlative traits 9

2.3 Analysis 10

2.4 Research Gap 12

3 Design and Implementation **13**

3.1 Hardware 13

3.1.1 Sensing 14

3.1.2 Pre-Processing 14

3.1.3 Analysis / Classification 16

3.1.4 Component Costs 16

3.2 Software 20

3.2.1 ThermoSense Implementation 22

3.2.2 Sensing 23

3.2.3 Pre-Processing 24

3.2.4 Analysis / Classification 25

3.3 Summary 28

4 Evaluation **30**

4.1 Sensor Properties 30

4.1.1 Bias 30

4.1.2 Noise 32

4.1.3 Sensitivity 34

4.2 Classification 34

4.2.1 Data gathering 36

4.2.2 Data labeling 36

4.2.3 Feature extraction and data conversion 36

4.2.4 Running Weka Tests 38

4.2.5 Classifier Experiment Set 40

4.3 Results 44

4.3.1 Classification 44

4.3.2 Energy Efficiency 46

4.4 Discussion 47

4.4.1 Classification 47

4.4.2 Energy Efficiency 51

5 Conclusions **53**

5.1 Evaluation of Criteria 53

5.1.1 Low Cost 53

5.1.2 Non-Invasive 54

5.1.3 Reliable 54

5.1.4 Energy Efficient 55

5.2 Future Work 55

5.2.1 Broader Data Collection 56

5.2.2 Different Feature Vectors 56

5.2.3 Different Classification Algorithms 56

5.2.4 Sub-pixel localization 56

5.2.5 Improving Robustness . . . 57
5.2.6 Field-of-view modifications . . . 57
5.2.7 New Sensors . . . 57
5.3 Summary . . . 58

Bibliography **58**

A Classification Algorithms **62**
A.1 Neural Networks . . . 62
A.2 k-nearest Neighbors . . . 62
A.3 Linear Regression . . . 63
A.4 Naive Bayes . . . 63
A.5 Support Vector Machines . . . 63
A.6 Decision Trees . . . 64
A.7 0-R . . . 64

B Knowledge Flows **65**

C Original Honours Proposal **67**
C.1 Background . . . 67
C.2 Aim . . . 68
C.3 Method . . . 69
C.3.1 Hardware . . . 69
C.3.2 Classification . . . 69
C.3.3 Robustness / API . . . 70
C.4 Timeline . . . 70
C.5 Software and Hardware Requirements . . . 71
C.6 Proposal References . . . 72

List of Tables

2.1 Comparison of information provided by different sensors types discussed with reference to the project's requirements 11

3.1 Three-tier structure of prototype hardware with corresponding components used 14

3.2 Breakdown of component costs for minimum viable implementation 17

3.3 Summary of code written and used within the Thermal Array Library 20

4.1 Weka parameters used for different classifications algorithms . . . 42

4.2 Breakdown of Classification Experiment Set data by true number of occupants 42

4.3 Classifier Experiment Set Results 45

4.4 Energy Consumption Results 46

List of Figures

2.1 Occupancy sensor taxonomy proposed by Teixeira, Dublon and Savvides [23] 6

3.1 MLX90620, Passive Infrared Sensor, and Arduino integration circuit diagram 15
3.2 Component breakdown of sensing system prototype 18
3.3 Sensing system prototype mounted on roof 19
3.4 Architecture of prototype sensor with tiers, software, communication protocols and information flow 21
3.5 MLX90620 block diagram (adapted from datasheet [19]) 24
3.6 Annotated Arduino initialization sequence and thermal packet serial output 25

4.1 Mean values of 4 minute night sky thermal capture plotted over sensor's 16×4 grid 31
4.2 Standard deviation of 4 minute night sky thermal capture plotted over sensor's 16×4 grid 32
4.3 Plot of occupant and background sensor noise at sampling speeds 0.5 Hz – 8 Hz 33
4.4 Temperature plot of five of the MLX90620's pixels as a hot object moves across them at a constant velocity 35
4.5 Process flow diagram for turning raw sensor input into occupancy estimates 37
4.6 Classifier Experiment Set Setup (measurements approximate) . . 43
4.7 Plot of three features against each other with occupancy truth values 49

B.1 Numeric knowledge flow 65
B.2 Nominal knowledge flow 66

List of Listings

3.1 Annotated and abbreviated image subtraction and feature extraction code from the Thermal Array Library 29
4.1 C4.5 Decision tree generated by Weka's J48 implementation from the Classification Experiment Set data 50

CHAPTER 1

Introduction

The proportion of elderly and mobility-impaired people in the Australian population is predicted to grow dramatically over the next century, leaving a large proportion of the population unable to live independently [8]. Human care for these groups is frequently performed on a volunteer basis, with nearly 40% of such carers committing 40 hours or more per week on caring, and having significantly lower labor force participation as a result [5]. Given the proportion of carers in the population is only projected to increase as the population ages, serious investments need to be made into technologies that can provide the support the elderly and mobility-imparted need to live independent of human assistance.

Additionally, the emergence of carbon pricing in many countries to combat anthropogenic climate change, as well as underinvestment in Australian energy infrastructure, are causing rising energy prices [22]. These prices have a particularly large effect on the elderly and disabled, as these demographics typically have below-average incomes.

Coinciding with these issues is the booming embedded systems and sensor industries, which are creating increasingly smaller computer and sensing systems. Every iteration, these systems become more powerful, more affordable, and more networked. This phenomena, termed the Internet of Things (IoT), has produced sub-$50 devices such as the Arduino and Raspberry Pi which unlock enormous potential to create computing systems to help combat these and other issues. One can envision a future 'smart home for the disabled' which leverages the IoT to offer a variety of services to help reduce financial and physical burdens alike.

Sensing techniques to determine occupancy, the detection of the presence and number of people in an area, are of particular use to the above demographics. Detection can be used to inform various devices to change state depending on the presence or absence of occupants, enabling a variety of useful automations. In particular, such a system could better regulate energy hungry devices to help reduce financial burden and greenhouse gas emissions. Household climate control, which in some regions of Australia accounts for up to 40% of energy usage [4]

is one area in which occupancy detection can reduce costs. Several papers have found efficiency can be significantly increased, with some some approaches providing annual energy savings of up to 25% [7].

The field of sensors capable of predicting occupancy is quite broad. However, in many cases they suffer from problems of installation logistics, difficult assembly, assumptions on user's technology ownership and/or component cost. In the smart home envisioned, accuracy is important, but accessibility is paramount.

In this research project, we construct an IoT-style occupancy detection sensor system that forms part of a theoretical 'smart home for the disabled.' This system must meet the following qualitative accessibility criteria;

- *Low Cost*: The set of components required should aim to minimise cost, as these devices are intended to be deployed in situations where the serviced user may be financially restricted.

  We consider a prototype that costs less than $300, with an anticipated reduction in cost as technology improves to be sufficiently low cost.

- *Non-Invasive*: The sensors used in the system should gather as little information as necessary to achieve the detection goal; there are privacy concerns and adoption issues with sensors that are perceived to be invasive.

  We consider a sensor that obfuscates the identities and activities of those sensed to be sufficiently non-invasive.

- *Reliable*: The system should be able to operate without user intervention or frequent maintenance, and should be able to perform its occupancy detection goal with a high degree of accuracy.

  We consider a system with an accuracy above 75% to be sufficiently reliable for a prototype system.

- *Energy Efficient*: The system may be placed in a location where there is no access to mains power, or where the retrofitting of appropriate power interfaces can be expensive (such as a residential roof); the ability to survive for long periods on only battery power is important.

  In the prototype stage, we consider a device that can run for one week on a battery that can be reasonably mounted on a roof to be sufficiently energy efficient.

Ultimately, this project attempts to answer the following research question: How can one create a complete sensing system to detect occupancy in a low cost, non-invasive, reliable and energy efficient way?

We take a traditional experimental approach to this research question by dividing the question into a series of sub-questions, resulting experiments and discussion. This occurs over three core chapters:

1. *Literature Review*

   An extensive literature review will be performed of the field of occupancy detection to determine which sensors types are most apparently appropriate for occupancy detection given the above criteria.

   The chapter identifies that existing systems do not meet our current needs and that low-resolution thermal sensing appears to offer the best maximization of our criteria.

2. *Design*

   Once the most appropriate sensor type is determined, a complete sensing system will be developed to provide a platform for experimentation and evaluation of the sensor, as well as to capture, store, visualize and replay sensor data for those purposes.

   Firstly, we describe the development of an extensible hardware architecture based upon an Arduino, Raspberry Pi, a Passive Infrared Sensor (PIR) and the Melexis MLX90620 (*Melexis*) sensor.

   Secondly, we describe the implementation of a custom and reusable software library, the the Thermal Array Library (TArL), which consists of both low-level code running on the Arduino embedded platform, and high-level code for image analysis and occupant prediction.

3. *Evaluation*

   Once a prototype exists, a methodology will be developed to evaluate the properties of the sensing system, and experiments will be designed and conducted to test different algorithms effectivenesses in using the system's data for occupancy detection.

   Firstly, we investigate properties of the sensor that may influence the sensing system's ability to accurately detect occupants by performing a series of experiments and analyzing the interesting properties of the resulting data.

   Secondly, we detail the methodology by which thermal data and ground truth is captured, as well as the details of the software pipeline required to generate occupancy predictions. This includes how we approach machine learning, which algorithms we choose, and what parameters they use.

   Thirdly, we perform a set of thermal captures, annotate them with ground truth information, then apply our chosen machine learning algorithms and

occupancy detection methodology to the data to generate a series of results. We also measure the energy consumption of the prototype while capturing thermal data.

Finally, we compare our accuracy and energy efficiency data against the identified state of the art in the field, hypothesize as to why different machine learning approaches have achieved different results, and provide an in-depth analysis on the power consumption of the prototype and methods by which the energy efficiency could be further improved.

CHAPTER 2

Literature Review

Within this chapter we consider the broad variety of sensing systems available, and how well different types of sensors meet our sensing criteria. It can be difficult to approach the broad variety of sensor types in the field, so a structure must be developed through which to evaluate them. Teixeira, Dublon and Savvides [23] propose a 5-element human-sensing criteria which provides a structure through which we may define the broad quantitative requirements of different sensors.

These quantitative requirements can be used to exclude sensing options that clearly cannot meet the requirements before the more specific qualitative accessibility criteria will be considered for those remaining sensors.

The quantitative criteria elements are;

1. *Presence*: Is there any occupant present in the sensed area?
2. *Count*: How many occupants are there in the sensed area?
3. *Location*: Where are the occupants in the sensed area?
4. *Track*: Where do the occupants move in the sensed area? (local identification)
5. *Identity*: Who are the occupants in the sensed area? (global identification)

At a fundamental level, this research project requires a sensor system that provides both Presence and Count information. To assist with the reduction of privacy concerns, excluding systems that permit Identity information will generally result in a less invasive system also. The presence of Location or Track information are irrelevant to our project's goals, but overall, minimizing these elements should in most cases help to maximize the energy efficiency of the system also.

Teixeira, Dublon and Savvides [23] also propose an occupancy sensor taxonomy (see Figure 2.1), which categorizes different sensing systems in terms of what

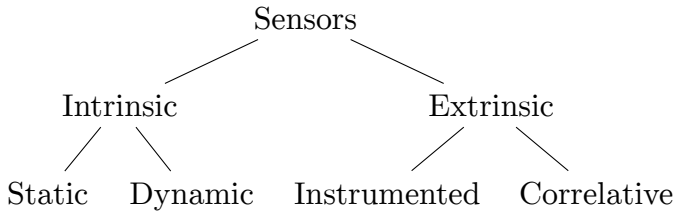


Figure 2.1: Occupancy sensor taxonomy proposed by Teixeira, Dublon and Savvides [23]

information they use as a proxy for human-sensing. We use this taxonomy here as a structure through which we group and discuss different sensor types.

2.1 Intrinsic traits

Intrinsic traits are those which can be sensed that are a direct property of being a human occupant. Intrinsic traits are particularly useful, as in many instances they are guaranteed to be present if an occupant is present. However, they do have varying degrees of detectability and differentiation between occupants. Two main subcategories of these sensor types are static and dynamic traits.

2.1.1 Static traits

Static traits are physiologically derived, and are present with most (living) occupants. One key static trait that can be used for occupant sensing is that of thermal emissions. All human occupants emit distinctive thermal radiation in both resting and active states. The heat signatures of these emissions could potentially be measured with some apparatus, counted, and used to provide Presence and Count information to a sensor system, without providing Identity information.

Beltran, Erickson and Cerpa [7] propose ThermoSense, a system that uses a type of thermal sensor known as a Thermal Detector Array (TDA). This sensor is much like a camera, in that it has a field of view which is divided into "pixels"; in this case an 8×8 grid of detected temperatures. This sensor is mounted on an embedded device on the ceiling, along with a Passive Infrared Sensor (PIR) for basic motion detection, and uses machine learning algorithms to detect human heat signatures within the raw thermal and motion data it collects. ThermoSense measures accuracy with Root Mean Squared Error (RMSE), an average of the absolute value that their prediction deviated from the true result.

They achieve an RMSE of 0.35 occupants, which they indicate is sufficient for accurate occupancy detection.

Another static trait is that of CO_2 emissions, which, like thermal emissions, are emitted by human occupants in both resting and active states. By measuring the buildup of CO_2 within a given area, one can use a variety of mathematical models of human CO_2 production to determine the likely number of occupants present. Hailemariam et al. [14] trialled this as part of a sensor fusion within the context of an office environment, achieving a $\approx 94\%$ accuracy. Such a sensing system could provide both the Presence and Count information, and exclude the Identity information as required. However, CO_2 based detection methods have serious drawbacks, discussed by Fisk, Faulkner and Sullivan [12]: The CO_2 feedback mechanism is very slow, taking hours of continuous occupancy to correctly identify the presence of people. In a residential environment, occupants are more likely to be moving between rooms than an office, so the system may have a more difficult time detecting in that situation. Similarly, such systems can be interfered with by other elements that control the CO_2 buildup in a space, like air conditioners and open windows. This is also much more of a concern in a residential environment compared to the studied office space, as the average residence can have numerous such confounding factors that cannot easily be controlled for.

Occupant identification can also be achieved through the use of video or still-image cameras and advanced image processing algorithms. Video can be used in occupancy detection in several different ways, achieving different levels of accuracy and requiring different configurations. The first use of video, POEM, proposed by Erickson, Achleitner and Cerpa [11] is the use of video as a "optical turnstile": The video system detects potential occupants and the direction they are moving in at each entrance and exit to an area, and uses that information to extrapolate the number of occupants within the turnstiled area. They report the system achieves up to a 94% accuracy. However, the main issue with such a system applied to a residential environment is the system assumes that there will be wide enough "turnstile areas", corridors of a fairly large area that connect different sections of a building, to use as detection zones. While such corridors exist in office environments, they are less likely to exist in residential ones.

Another video-based sensor system is proposed by Serrano-Cuerda et al. [20], that uses ceiling-based cameras and advanced image processing algorithms to count the number of people in the captured area. They measure accuracy using an F-score, which takes into account both false-positive and false-negative results, and is reported as 0.967, which is highly accurate. Such a system could be successfully applied to the residential environment, as both it and the "optical turnstile" model provide Presence and Count information. However, these

systems also allow Identity to be determined, and thus are perceived as privacy-invasive.

2.1.2 Dynamic traits

Dynamic traits are usually products of human occupant activity, and thus can generally only be detected when a human occupant is physically active or in motion.

Ultrasonic systems, such as Doorjamb proposed by Hnat et al. [15], use clusters of such sensors above doorframes to detect the height and direction of potential occupants travelling between rooms. This acts as a turnstile based system, much like POEM [11], but augments this with an understanding of the model of the building to correct for invalid and impossible movements brought about from sensing errors. This system provides an overall room-level tracking accuracy of 90%, however to achieve this accuracy, potential occupants are intended to be tracked using their heights, which has privacy implications. The system can also suffer from problems with error propagation, as there are possibilities of "phantom" occupants entering a room due to sensing errors.

Solely PIR based systems, like those used by Hailemariam et al. [14], involve the motion of the sensor being averaged over several different time intervals, and fed into a decision tree classifier. This PIR system alone produced a $\approx 98\%$ accuracy. However, such a system, due to only motion detection capabilities, can only provide Presence information, and is unable to provide Count information, or detect motionless occupants.

2.2 Extrinsic traits

Extrinsic traits are those which are actually other environmental changes that are caused by or correlated with human occupant presence. These traits generally present a less accurate picture, or require the sensed occupants to be in some way "tagged", but they are generally also easier to sense themselves. The sensors in this category have been divided into two subcategories.

2.2.1 Instrumented traits

One extrinsic trait category is instrumented approaches; these require that detectable occupants carry with them some device that is detected as a proxy for the occupant themselves.

The most obvious of these approaches is a specially designed device. Li et al. [18] used RFID tags and several antennas as a triangulation and tracking mechanism to pinpoint tag-carrying occupants to a specific HVAC thermal zone. For stationary occupants, there was a detection accuracy of $\approx 88\%$, and for occupants who were mobile, the accuracy was $\approx 62\%$. Such a system could be re-purposed for the residence, however it requires occupants to be constantly carrying their tags, which is less likely in such an environment. Additionally, the accuracy for this system is not necessarily high enough for a residential environment, where much smaller rooms are used.

To make extrinsic detection more reliable, Li, Calis and Becerik-Gerber [16] leverage a common consumer device; WiFi enabled smart phones. They propose the *homeset* algorithm, which uses the phones to scan the visible WiFi networks, and from that information estimate if the occupants are present or absent in their home by "triangulating" their position from the visible WiFi networks. This solution does not provide the fine-grained Presence data that we need, as it is only able to triangulate the phone's position very roughly with the wireless network detection information.

Balaji et al. [6] also leverage smart phones to determine occupancy, but in a more broad enterprise environment: Wireless device association logs are analysed to determine which access points in a building a given occupant is connected to. If this access point falls within the radio range of their designated "personal space", they are considered to be occupying that personal space. This technique cannot be applied to a residential environment, as there are usually not multiple wireless hotspots present.

Finally, Gupta, Intille and Larson [13] use specifically the GPS functions of the smartphone to perform optimisation on heating and cooling systems by calculating the "travel-to-home" time of occupants at all times and ensuring at every distance the house is minimally heated such that if the potential occupant were to travel home, the house would be at the correct temperature when they arrived. While this system does achieve similar potential air-conditioning energy savings, it is not room-level modular, and also presupposes an occupant whose primary energy costs are from incorrect heating when away from home, which isn't necessarily the case for the elderly or disabled demographics considered in this dissertation.

2.2.2 Correlative traits

The second of our discussed subcategories are correlative approaches. These approaches analyse data that is correlated with human occupant activity, but

does not require a specific device to be present on each occupant that is tracked with the system.

The primary approach in this area is work done by Kleiminger et al. [17], which attempts to measure electricity consumption and use such data to determine Presence. Electricity data was measured at two different levels of granularity; the whole house level with a smart meter, and the consumption of specific appliances through smart plugs. This data was then processed by a variety of classifiers to achieve a classification accuracy of more than 80%. Such a system presents a low-cost solution to occupancy, however it is not sufficiently granular in either the detection of multiple occupants, or the detection of occupants in a specific room. Additionally, it may be too invasive, due to the number of sensors and sensor plugs involved.

2.3 Analysis

From these various sensor options, there are a few candidates that provide the necessary quantitative criteria (Presence and Count); these are thermal, CO_2, Video, Ultrasonic, RFID, WiFi association and WiFi triangulation based methods. All sensing options are compared in Table 2.1.

In the context of our four qualitative accessibility criteria, CO_2 sensing has several reliability drawbacks, the predominant ones being a large lag time to receive accurate occupancy information and interference from a variety of air conditioning sources which can modify the CO_2 concentration in the room in unexpected ways.

Video-based sensing methods suffer from invasiveness concerns, as they by design must have a constant video feed of all detected areas.

Ultrasonic methods suffer from reliability concerns when a user falls outside the prescribed height bounds of average humans. The detection accuracy of wheelchair bound occupants, a potential demographic of our proposed sensing system, are not discussed in the Doorjamb paper. The paper indicates various complications such as hats or carrying items may affect the detection, which suggests that a wheelchair may do the same. Ultrasonic methods also provide weak Identity information through height detection.

RFID sensing also has several drawbacks; it is a difficult value proposition to get residential occupants to carry RFID tags with them continuously. Another drawback is that the triangulation methods discussed are too unreliable to place occupants in specific rooms in many cases, and may suffer from read range issues depending on the specific RFID technologies used.

	Requires		Excludes	Irrelevant	
	Presence	Count	Identity	Location	Track
Intrinsic					
Static					
Thermal	✓	✓	✓	✓	
CO_2	✓	✓	✓		
Video	✓	✓	✗	✓	✓
Dynamic					
Ultrasonic	✓	✓	✗		✓
PIR	✓	✗	✓		
Extrinsic					
Instrumented					
RFID	✓[1]	✓	✓	✓	
WiFi assoc.[2]	✓[1]	✓	✗	✓	
WiFi triang.[2]	✓[1]	✓	✗		
GPS[2]	✓[1]	✗	✓	✓	
Correlative					
Electricity	✓[1]	✗	✓		

[1]Doesn't provide data at required level of accuracy for home use.

[2]Uses smartphone as detector.

Table 2.1: Comparison of information provided by different sensors types discussed with reference to the project's requirements

WiFi association is not granular enough for residential use, as the original enterprise use case presupposes a much larger area, as well as multiple wireless access points, neither of which a typical residential environment has.

WiFi triangulation is a good candidate for residential use, as there are most likely neighbouring wireless networks that can be used as virtual landmarks. However, it suffers from the same granularity problems as WiFi association, as these signals are not specific enough to pinpoint an occupant to a specific room.

For approaches presupposing smartphones being present on each occupant, it similarly difficult to ensure that occupants are carrying their smartphones with them at all times in a residential environment, as with RFID tags. Another issue with smart phones is that they can represent an expense that the target markets of the elderly and the disabled may not be able to afford, as opposed to a dumb mobile or landline phone.

Finally, we have thermal sensing. It provides both Presence and Count information, as it uses occupants' thermal signatures to determine the presence of people in a room. It does not however provide Identity information, as thermal signatures are not sufficiently unique with the technologies used to distinguished between occupants. Such a sensor system is presented as low-cost and energy efficient within ThermoSense [7]. The system non-invasive by design and can reliably detect occupants with a low Root Mean Squared Error. For our specific accessibility criteria, thermal sensing appears to be the most suitable option.

2.4 Research Gap

It is clear that ThermoSense's use of the Grid-EYE sensor provides a system that meets our goals of low cost, non-invasiveness, reliability and energy efficiency. However, there is room for improvement in these goals. The TMote Sky, the embedded controller for the ThermoSense design is expensive (estimated to be $100+), outdated (released in 2006) and does not appear to be easily acquirable within Australia (manufacturer's website is no longer available). Additionally, research has indicated that the Grid-EYE sensor is not available to purchase within Australia, or to order into Australia from other countries.

Because of the difficulty in acquiring ThermoSense's exact components, we believe there is a clear gap within the Australian market for an occupancy sensor that meets these goals, particularly with reference to the newer technologies available. Additionally, as the Grid-EYE cannot be used, an investigation into how well a substitute sensor can meet replicate ThermoSense's results will be necessary.

CHAPTER 3

Design and Implementation

To investigate thermal sensing's potential, a software and hardware prototype (the "sensing system") must now be constructed to provide a platform for experimentation and evaluation of the sensor chosen, as well as to capture, store, visualize and replay sensor data for those purposes. We will first discuss the hardware foundations of the project, then the architecture of the software developed to run on those foundations.

3.1 Hardware

As reliability and future extensibility are core concerns of the project, a three-tiered system is employed with regards to the hardware involved in the system (Table 3.1). At the bottom, the Sensing Tier, we have the sensors themselves. Connected to the sensors via those respective protocols is the Preprocessing Tier, hosted on an embedded system. The embedded device polls the data from these sensors, performs necessary calculations to turn the raw sensor information into actionable data, and communicates this via Serial over USB to the third tier. The third tier, the Analysis Tier, is run on a fully fledged computer. In our prototype, it captures and stores temperature and motion data it receives over Serial over USB, as well as visual data for ground truth purposes.

In the current prototype, the Analysis Tier merely stores captured data for offline analysis, in future prototypes this analysis can be done live and served to interested parties over a RESTful API. In the current prototype, the Analysis and Sensing Tiers are connected by Serial over USB, in future prototypes, this can be replaced by a wireless mesh network, with many Preprocessing/Sensing Tier nodes communicating with one Analysis Tier node.

Analysis Tier	Raspberry Pi B+
Preprocessing Tier	Arduino Uno R3
Sensing Tier	Melexis MLX90620 & PIR

Table 3.1: Three-tier structure of prototype hardware with corresponding components used

3.1.1 Sensing

As discussed in the Literature Review, using a Thermal Detector Array (TDA) appear to be the most viable way to achieve the high-level goals of this project. ThermoSense [7], the primary occupancy sensor in the TDA space, used the low-cost Panasonic Grid-EYE sensor for this task. This sensor, costing around $50, was suggested by ThermoSense to be effective in occupancy detection. However, while still available for sale in the United States, we were unable to order the sensor for shipping to Australia due to supplier-vendor contract agreements outside of our control. Using a sensor with such restrictions in place goes against an implicit criteria of the parts used in the project being relatively easy to acquire.

This forced us to search for alternative sensors in the space that fulfill similar criteria but were available in Australia. The sensor we chose was the Melexis MLX90620 (*Melexis*) [19], a TDA with similar overall qualities that differed in several important ways; it provides a 16×4 grid of thermal information, it has an overall narrower field of view and it sells for approximately $80. Like the Grid-EYE, the *Melexis* communicates over the 2-wire I^2C bus, a low-level bi-directional communication bus widely used and supported in embedded systems.

We envision a further advanced prototype to have wireless networking in a smaller form factor, much like ThermoSense. However, due to time and resource constraints, the scope of this project has been limited to a minimum viable implementation. This prototype architecture has been designed such that a clear path to an idea system architecture involving each Pre-Processing Tier and Analysis Tier being connected by a wireless mesh network to enable easy installation in households.

3.1.2 Pre-Processing

Due to low cost, broad support and ease of development, the Arduino platform was selected as the host for the Preprocessing Tier, and thus will handle the I^2C communication with the *Melexis*. Initially, this presented some challenges, as the *Melexis* recommends a power and communication voltage of 2.6V, while the

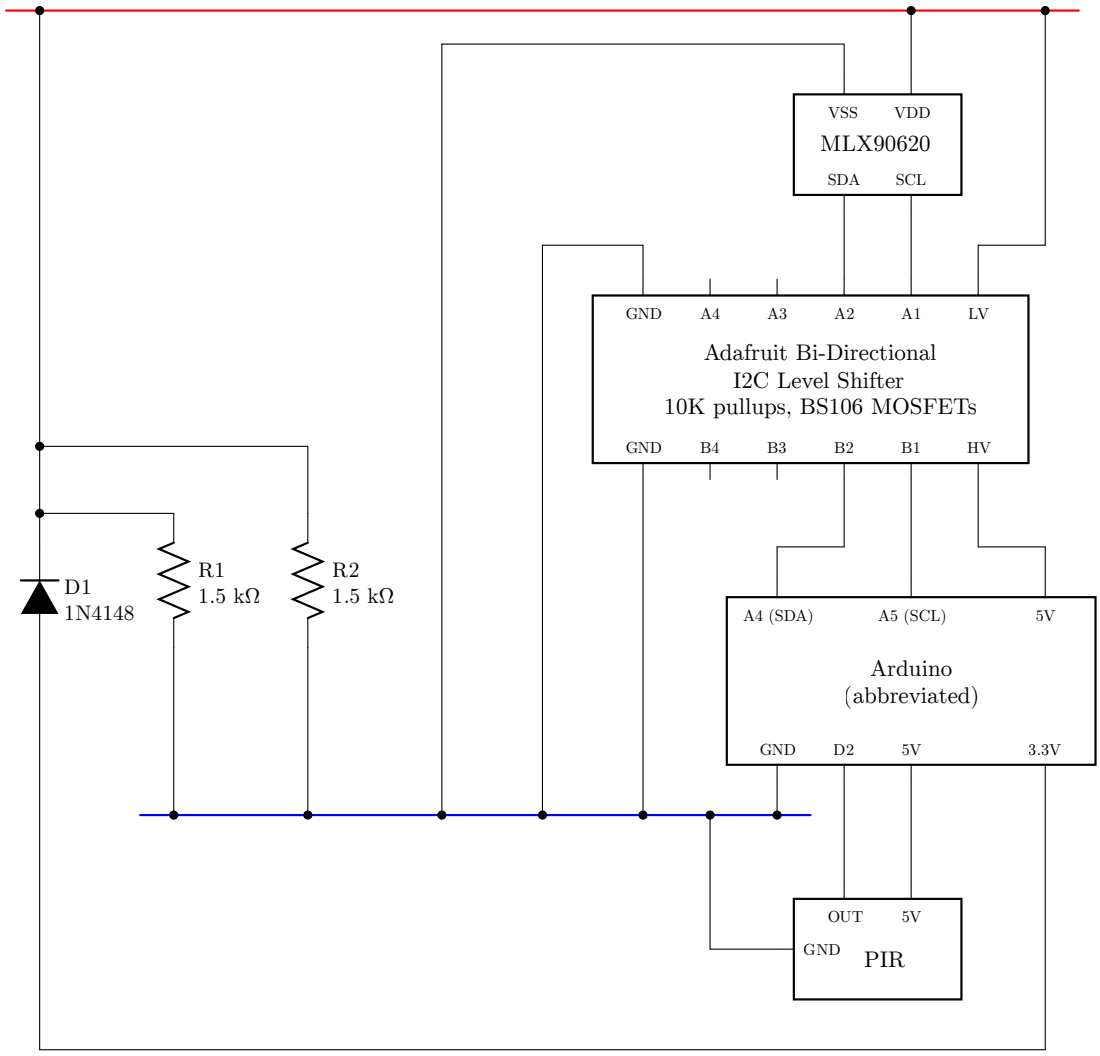


Figure 3.1: MLX90620, Passive Infrared Sensor, and Arduino integration circuit diagram

Arduino is only able to output 3.3V and 5V as power, and 5V as communication. Due to this, it was not possible to directly connect the Arduino to the *Melexis*, and similarly due to the two-way nature of the I^2C 2-wire communication protocol, it was also not possible to simply lower the Arduino voltage using simple electrical techniques, as such techniques would interfere with two-way communication.

A solution was found in the form of a I^2C level-shifter, the Adafruit "4-channel I2C-safe Bi-directional Logic Level Converter" [1], which provided a cheap method to bi-directionally communicate between the two devices at their own preferred voltages. The layout of the circuit necessary to link the Arduino and the *Melexis*, including the use of this converter, can be seen in Figure 3.1.

Additionally, as used in the ThermoSense paper, a Passive Infrared Sensor (PIR) motion detector [2] was also connected to the Arduino. This sensor, operating at 5V natively, did not require any complex circuitry to interface with the Arduino. It is connected to digital pin 2 on the Arduino, where it provides a rising signal (a "trigger") in the event that motion is detected, which can be configured to cause an interrupt on the Arduino. In the configuration used in this project, the sensor's sensitivity was set to the highest value and the timeout for re-triggering (the trigger reoccurring) was set to the lowest value (approximately 2.5 seconds). Additionally, the continuous re-triggering feature (whereby the sensor produces continuous rising and falling signals for the duration of motion) was disabled using the provided jumpers.

3.1.3 Analysis / Classification

For the Analysis Tier, the Raspberry Pi B+ was chosen, as it is a powerful and inexpensive computer capable of running Linux. The Arduino is connected to the Raspberry Pi over USB, which provides it both power and the capacity to transfer data. In turn, the Raspberry Pi is connected to a simple micro-USB rechargeable battery pack, which provides it with power, and subsequently the Arduino and sensors.

3.1.4 Component Costs

As being low-cost is one of the project's goals, we have summarized the cost of each of the components of the prototype in Table 3.2a. We believe that for a prototype, this cost is sufficiently low. In the envisioned system, there would only be one Raspberry Pi in the system, and it would not require a camera, lowering the cost to around $\$50 + \$135n$ where n is the number of sensors. Similarly, as

technology improves, sensor technology expected to continue to fall in price, causing the most expensive component, the infrared sensor, to become increasingly cost-effective.

When we compare this to the estimated cost of the ThermoSense system (Table 3.2b), we believe that it achieves a suitably comparable cost for a prototype. When removing the aspects of the prototype that would be unnecessary in the final version, the difference is only $15.

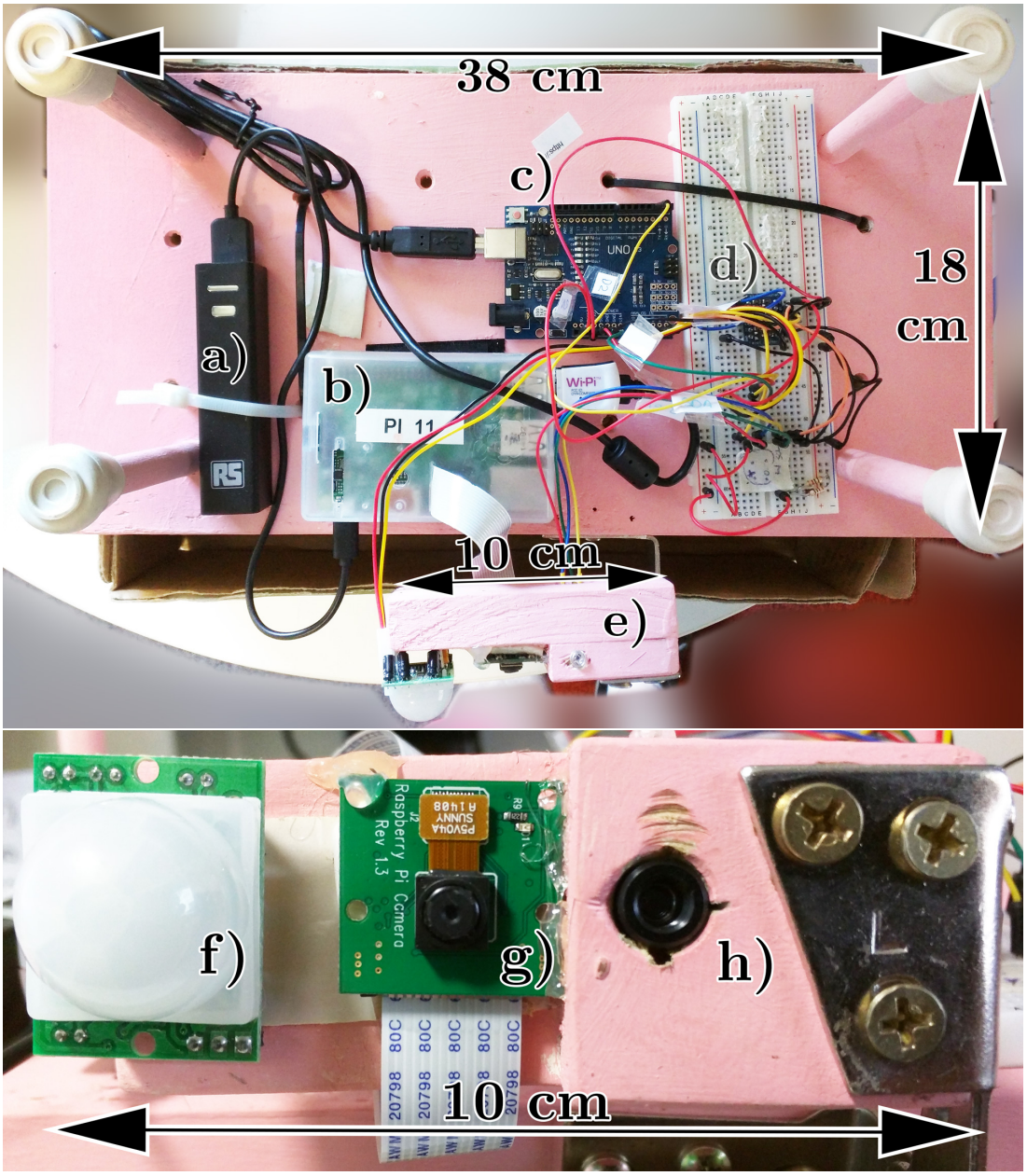
Part	**Cost**
MLX90620	$80
Raspberry Pi B+	$50
Arduino Uno R3	$40
Passive Infrared Sensor	$10
I^2C level shifter	$5
TOTAL	**$185**

(a) Our project

Part	**Cost**
TMote Sky	$110
Grid-EYE	$50
Passive Infrared Sensor	$10
TOTAL	**$170**

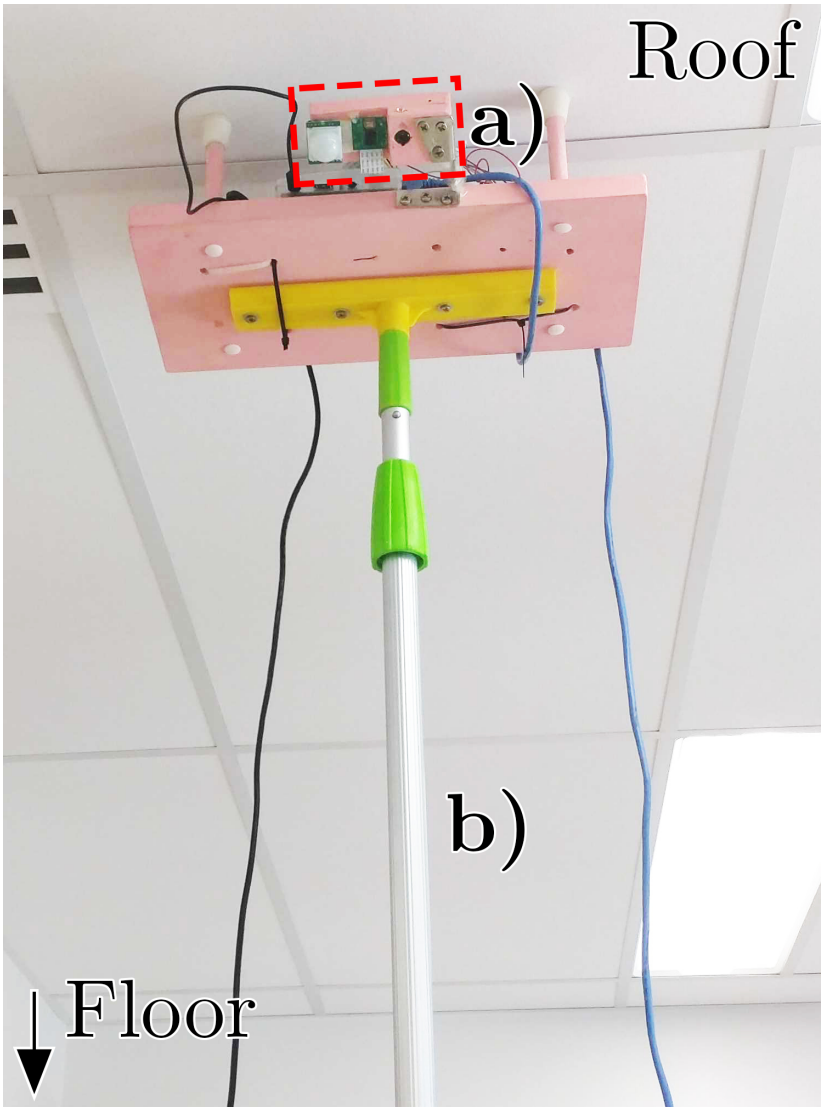
(b) ThermoSense (estimated)

Table 3.2: Breakdown of component costs for minimum viable implementation



a) Battery pack
b) Raspberry Pi
c) Arduino
d) Level-shifting circuitry
e) Movable sensor mount
f) PIR
g) Camera
h) MLX90620

Figure 3.2: Component breakdown of sensing system prototype



a) Sensors

b) Mounting pole

Figure 3.3: Sensing system prototype mounted on roof

Category	SLOC
TArL Python	674
`cam`	425
`features`	191
`pxdisplay`	58
TArL Arduino	492
`mlx90620_driver`	492
Analysis Scripts	147
Capture Scripts	234
Total	*1,624*

(a) Source Lines Of Code written

Library	Version
Arduino	
SDK	1.6.4
`SimpleTimer`	1.0
Python	
`networkx`	1.9.1
`numpy`	1.8.0
`matplotlib`	1.3.1
`picamera`	1.10
`Pillow`	2.8.1

(b) Libraries used

Table 3.3: Summary of code written and used within the Thermal Array Library

3.2 Software

At each layer of the described three-tier software architecture (pictured in greater detail in Figure 3.4), software exists to govern the operation of that tier's functionality. For the sensing tier, the Melexis MLX90620 (*Melexis*) and Passive Infrared Sensor (PIR)'s own software is used, while for the other tiers, a bilingual software library, the Thermal Array Library (TArL), was developed to provide a suite of functions to enable the easy data collection and analysis of information from the hardware prototype. TArL is split into two parts:

At the Preprocessing Tier, the Arduino, the TArL *Melexis* driver is found, which is written in the default Arduino C++ derivative language. The used of a low-level language is important at this tier as careful management of memory usage and algorithmic complexity is required in such a resource-constrained environment.

At Analysis Tier, a general purpose computer is used, and that is where the bulk of TArL can be found. As the processing environment has become less constrained, the choice of language becomes a possibility. In this instance, Python was chosen as TArL's language on the Analysis Tier. Python was chosen as it is a high-level language with excellent library support for the functions required of the Analysis Tier, including serial interface, the use of the Raspberry Pi's built in camera, and image analysis. The 2.x branch of Python was chosen over the 3.x branch, despite its age, due a greater maturity in support for several key graphical interface libraries. Much of the Analysis tier's code is based upon the ThermoSense implementation, which we describe here.

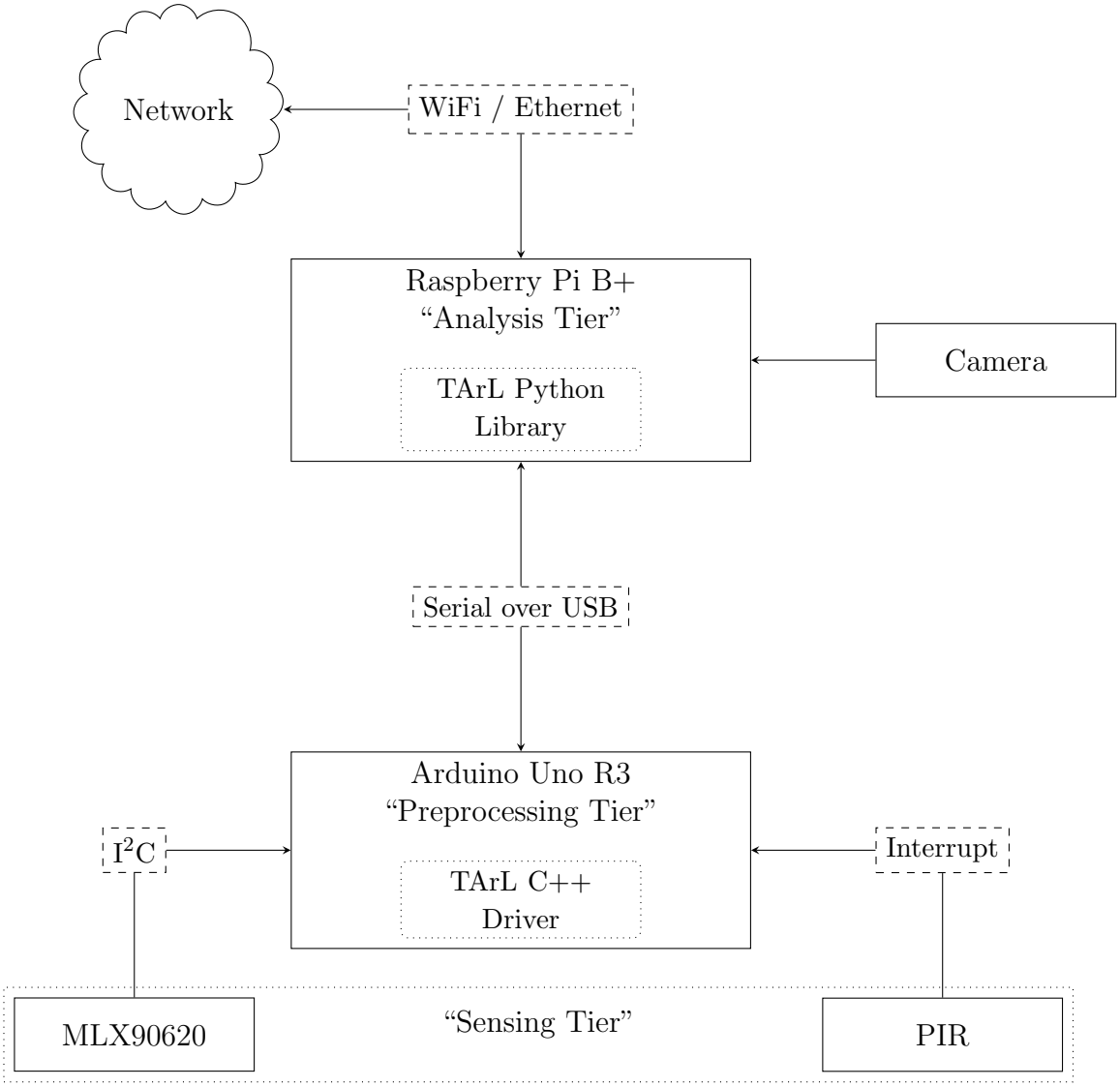


Figure 3.4: Architecture of prototype sensor with tiers, software, communication protocols and information flow

3.2.1 ThermoSense Implementation

The ThermoSense approach combines a PIR, which detects motion, and a Thermal Detector Array (TDA) which creates a thermal image, to determine occupancy in a sensor fusion. These sensors are fused by leveraging the PIR to determining the $0/>0$ case and the TDA to determine specific occupancy numbers. The specific TDA used subdivides the visible area into an 8×8 grid of sections from which temperatures can be derived. This sensor system is attached to the roof on a small embedded controller which is responsible for collecting the thermal data and transmitting it back to another computer for analysis via a low powered wireless networking protocol.

Machine learning classification, the use of algorithms and training data to generate models that can make predictions on previously unseen data, is a large part of the ThermoSense paper. ThermoSense uses supervised learning algorithms, which require a set of examples with the correct answer.

Supervised classification techniques can be split into two different classes of techniques; numeric and nominal. Numeric techniques provide predictions that are numerical in nature, that is, they return results on a continuous number line. Nominal techniques provide predictions whereby each new data point is predicted to belong to one of a set of predetermined classes, for example, colors of the rainbow.

The training data classification algorithms use is a set of examples that have the corresponding correct answer attached. The set of values that describe each example are known as feature vectors.

Occupants are separated from background radiation through the use of an image subtraction algorithm maintaining per-pixel mean and standard deviation values to update a thermal background map. If no motion is detected, this map is updated using a slow-moving Exponential Weighted Moving Average (EMWA) over a 15 minute time window. If the room remains occupied for a long period, a more complex scaling algorithm is used which considers the coldest points in the room empty, and averages them against the new background, then performs an EMWA with a lower weighting.

This per-pixel average and standard deviation information updated every frame is used to determine several characteristics to be used as feature vectors. The determination of the feature vectors was subject to experimentation, since the differences at each grid element are too susceptible to individual room conditions to be used as feature vectors. Instead, a set of three different features was designed;

1. **Number of active pixels**: The total number of pixels that are considered "active" in a given frame

2. **Number of connected components**: If each active pixel is joined with its immediate active neighbors, how many "islands" of active pixels (termed connected components in graph theory) exist.

3. **Size of largest connected component**: The number of active pixels contained within the largest connected component

These feature vectors were compared against three classification approaches; K-Nearest Neighbors, Linear Regression and an Artificial Neural Network. This final classification is subject to a final averaging process over a four minute window, which was determined by experimentation to accurately remove the presence of independent errors from the raw classification data.

In the ThermoSense system, there exists a PIR whose purpose is to determine if there is currently motion in the detected area. This motion detection is averaged over a time window and is used by the ThermoSense system to provide Presence information to the system. Because of this, it is not necessarily a requirement that cases with zero people are provided to the classification algorithms above, as the PIR alone can determine this information. ThermoSense performed experimentation to determine if the classification was more accurate when instances of empty rooms were provided to the classification algorithm vs. not. They found that generally not providing the empty case to the classification algorithm improved accuracy.

3.2.2 Sensing

The *Melexis* itself is its own computer (see Figure 3.5), containing EEPROM storage, RAM and unspecified code to perform "digital filtering" on the 16×4 array of digital active thermopiles. We are able to communicate with the *Melexis* through the provided I^2C interface, which offers commands to read both the EEPROM, and the sensor's RAM directly.

The sensor's EEPROM contains configuration values that the interfacing device is required to input into the device's RAM as part of a multi-step initialization sequence, and also contains constants used as part of the raw data to °C conversion process. The sensor's RAM contains the partially-filtered raw data, which is updated with reference to a clock frequency set between 0.5 Hz to 512 Hz in the initialization process.

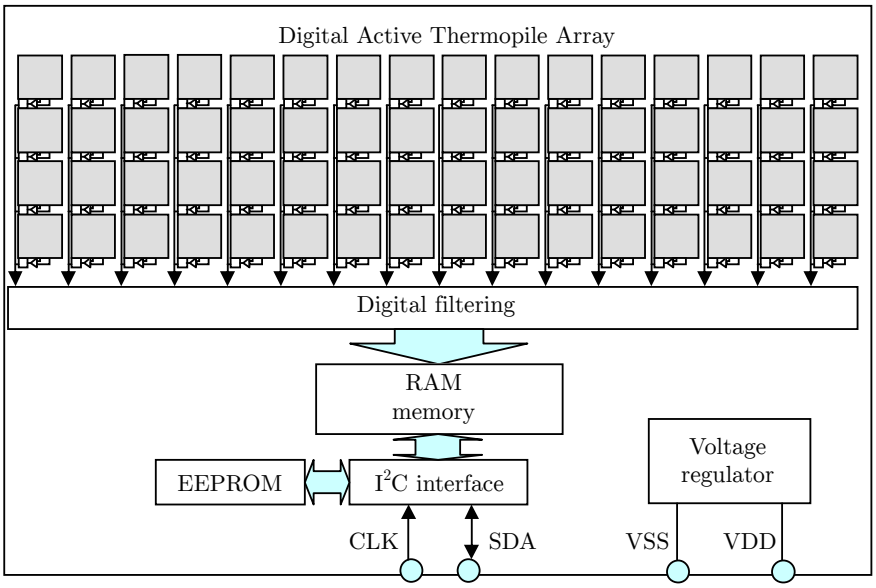


Figure 3.5: MLX90620 block diagram (adapted from datasheet [19])

The sensor's documentation offers no information regarding reconfiguration of the sensor's internal programming code, nor what code exists on the sensor when purchased. As such, we refer to the sensor's dataset [19] and use only the documented commands to interface with the sensor.

3.2.3 Pre-Processing

On the Arduino, the TArL C++ Driver is written as one monolithic Arduino program, termed `mlx90620_driver.ino`. This program's purpose is to take simple commands over serial to configure the *Melexis* and to report back the current temperature values and PIR motion information at either a pre-set interval, or when requested.

To calculate the final temperature values that the *Melexis* offers, a complex initialization and computational process must be followed, which is specified in the sensor's datasheet [19]. This process involves initializing the sensor with values attained from the on-board EEPROM, then retrieving a variety of normalization and adjustment values, along with the raw sensor data, to compute the final temperature result.

The basic algorithm to perform this normalization was based upon the provided datasheet [19], as well as code by users "maxbot", "IIBaboomba", "nseidle" and others on the Arduino Forums [3] and was modified to operate with the newer

```
INIT 0                      # Initialization sequence begins at time=0 milliseconds
INFO START                  # Information section starts
DRIVER MLX90620             # MLX90620 driver is being used
BUILD Feb  1 2015 00:00:00  # Driver was compiled at specified date
IRHZ 1                      # Infrared data is being sampled at 1Hz
INFO STOP                   # Information section ends
ACTIVE 33                   # Sensor is active at ready to send data at time=33 milliseconds

START 34                    # Thermal packet begins at time=34 milliseconds
MOVEMENT 0                  # No movement in this frame
1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0
1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0
1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0
1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0
STOP 97                     # Thermal packet ends at time=97 milliseconds
```

Figure 3.6: Annotated Arduino initialization sequence and thermal packet serial output

Arduino "Wire" I^2C libraries released since the authors' original posts. To ensure the driver can be adapted and extended in the future, the code was also restructured and rewritten to be both more readable, to introduce a set of features to make the management of the sensor data easier for the user, and for the information to be more human readable.

The first of the features introduced was the human-readable format for serial transmission. This allows easy debugging of the sensor with only a rudimentary serial console. When the Arduino is powered up, an initialization sequence is output (Figure 3.6). This specifies several things that are useful to the user; the attached sensor ("DRIVER"), the build of the software ("BUILD") and the refresh rate of the sensor ("IRHZ"). Several different headers, such as "ACTIVE" and "INIT" specify the current millisecond time of the processor, thus indicating how long the execution of the initialization process took (33 milliseconds).

Once booted, the user is able to send several one-character commands to the sensor to configure operation. Depending on the input sensor configuration, IR data may be periodically output, or otherwise manually triggered. This IR data is produced in the packet format described in Figure 3.6.

3.2.4 Analysis / Classification

On the Analysis Tier, TArL's set of Python libraries and accompanying capture and analysis scripts were developed to interface with the Arduino, parse and interpret its data, and to provide data logging and visualization capabilities. TArL's Python portion provides 4 main feature sets across 3 files; the `Manager`

series of classes, the `Visualizer` class, the `Features` class and the `pxdisplay` module.

`Manager` classes

The Manager series of classes are the direct interface between TArL's C++ Arduino driver and TArL's Python components. They implement a multi-threaded serial data collection and parsing system which converts the raw serial output of the connected Arduino into a series of Python data structures that represent the collected temperature and motion data of each captured frame. Several different versions of the `Manager` class exist to perform slightly different functions. When initializing these classes the sample rate of the *Melexis* can be configured, and it will be sent through to the Arduino for updating.

`BaseManager` is responsible for the implementation of the core serial parsing functions. It also provides a threaded interface through which the *Melexis*'s continuous stream of data can be subscribed to by other threads. The primary API, the `subscribe_` series of functions, return a thread-safe queue structure, through which thermal packets can be received by various other threads when they become available.

`Manager`, the primary class, provides access the *Melexis*'s data at configurable intervals. When initializing this class, you may specific 0.5, 1, 2, 4 or 8 Hz, and the class will configure the Arduino to both set the *Melexis* to this sample rate, and to automatically write this data to the serial buffer at the same rate. This serial interface is multi-threaded, as at higher serial baud rates if data is not polled continuously the internal serial buffer fills and data is discarded. By ensuring this process cannot be blocked by other parts of the running program this problem is mostly eliminated.

`OnDemandManager` operates in a similar way to `Manager`, however it uses a polling model instead of the periodic model of the other classes. Scripts may request thermal/motion data from the class at any interval, and `OnDemandManager` will poll the Arduino for information and block until this information is parsed and returned.

Finally, `ManagerPlaybackEmulator` is a simple class which can take a previously created thermal recording from a file, and emulate the `Manager` class by providing access to thread-safe queues which return this data at the specified Hz rate. This class can be used as a means to playback thermal recordings with the same visualization functions.

`pxdisplay` functions

The `pxdisplay` module is a set of functions that utilize the `pygame` library to create a simple live-updating window containing a thermal map representation of the thermal data. One can generate any number of `pxdisplay` objects, which leverage the `multithreading` library and `multithreading.Queue` to allow thermal data to be sent to the display.

The class also provides a set of functions to set a "hotest" and "coldest" temperature and have RGB colors assigned from red to green to blue for each temperature value that falls between those two extremes.

`Visualizer` class

The `Visualizer` class is the natural compliment to the `Manager` series of classes. The functions contained within can be provided with a Queue object (generated by a `Manager` class) and can perform a variety of visualization and storage functions.

From the recording side, the `Visualizer` class can "record" a thermal capture by saving the motion and thermal information to a simple `.tcap` file, which stores the sample rate, timings, thermal and motion data from a capture in a simple, plain-text format. The class can also read these files back into the data structures `Visualizer` uses internally to store data. If `Visualizer` is running on a Raspberry Pi, it can also leverage the `picamera` library and the `OnDemandManager` class to synchronously capture both visual and thermal data for the purposes of ground truth verification.

From the visualization side, `Visualizer` can leverage the `pxdisplay` module to create thermal maps that can update in real-time based on the thermal data provided by a `Manager` class. The class can also generate both images and movie files from thermal recordings using the `PIL` and `ffmpeg` libraries.

`Features` class

As discussed in Subsection 3.2.1, ThermoSense [7] demonstrated the separation of "background" information from "active" pixels, and from that information, the extraction of the features necessary for a classifier to correctly determine the number of people in an 8×8 thermal image.

In accordance with the pseudo-code outlined in the ThermoSense paper and our description of the implementation, the algorithm described in Listing 3.1 was created to extract the three features identified by ThermoSense; number

of active pixels, number of connected components and size of largest connected components.

Given the scope restriction to a minimum viable implementation, the portion of the ThermoSense code dealing with scaling the thermal background for rooms without motion was not implemented. For connected component determination, we leveraged the `networkx` graph library.

The output of feature information is the extent to which the `Features` class is involved in machine learning classification. The code used to perform the actual classification step is discussed in Chapter 4.

3.3 Summary

We believe that the hardware and software architecture presented here lays a solid foundation on which experimental data can be collected. The hardware architecture, as discussed, as been specifically selected to ensure that there is a transition path from the current USB Serial Pre-Processing/Analysis connection to one which does this wirelessly. The software library, TArL, has been written to be robust and general, so that its functionality is both useful in the current situation, and also for future experiments with this and other prototypes.

```
# INITILISATION: Import libs, set up variables
import math, itertools, networkx

w, h       = 16, 4       # Get thermal image dimensions
wgt        = 0.01        # Weighting for exp. weighted moving avg.
fst_frame  = get_frame() # 1st thermal frame, set elsewhere (2D array)
back       = fst_frame   # Thermal background b (2D array)
means      = fst_frame   # Per pixel x̄ (2D array)
pstds      = [[0]*w]*h   # Per pixel intermediate σ (2D array of 0)
stds       = [[0]*w]*h   # Per pixel complete σ (2D array of 0)
n          = 1           # Processed frames counter

# f: New frame received from sensor, starting at the 2nd frame (2D array)
# is_motion: If there has been motion detected over given time window.
def get_features(f, is_motion):
  n      += 1                # Increment frame counter
  active  = []               # Init empty active list
  g       = networkx.Graph() # Init graph structure

  # BACKGROUND UPDATE: Iterate over every pixel and update if no motion
  for i, j in itertools.product( range(w), range(h) ):
    # If no motion update b_{i,j}, x̄_{i,j}, & σ_{i,j}  with f_{i,j}
    if not is_motion:
      back[i][j]   = wgt * f[i][j] + (1 - wgt) * back[i][j]      # b_{i,j}
      means[i][j]  = means[i][j] + (f[i][j] - means[i][j]) / n  # x̄_{i,j}
      pstds[i][j]  = pstds[i][j] + (new[i][j] - means[i][j])
                       * (c - means[i][j])
      stds[i][j]   = math.sqrt(pstds[i][j] / (n-1))              # σ_{i,j}

    # GRAPH GENERATION: If (f_{i,j} - b_{i,j}) > 3σ_{i,j} add pixel to active & graph
    if (f[i][j] - back[i][j]) > (3 * stds[i][j]):
      active.append((i,j))

      # Link all adjacent active pixels in graph structure
      for ix, jx in [(-1, -1), (-1, 0), (-1, 1), (0, -1)]:
        g.add_edge((i,j), (i+ix,j+jx)) if (i+ix, j+jx) in active

  # CONNECTED COMPONENTS: Get connected comps. from graph & gen features
  cons           = list( networkx.connected_components(g) )
  num_active     = len(active)
  num_connected  = len(cons)
  size_connected = max(len(c) for c in cons) if len(cons) > 0 else None

  return (num_active, num_connected, size_connected)
```

Listing 3.1: Annotated and abbreviated image subtraction and feature extraction code from the Thermal Array Library

CHAPTER 4

Evaluation

In this chapter we devise a set of experiments to test the sensor system's properties and come to conclusions as to their effect on our ability to detect occupants. We then outline a process for taking raw sensor data and performing occupancy predictions with it. Using that process, we then devise a set of occupancy scenarios, and experiment with different machine learning algorithms abilities to determining occupancy from that data.

4.1 Sensor Properties

In order to best utilize the Melexis MLX90620 (*Melexis*), we must first understand the properties it exhibits, and their potential effects on our ability to perform occupancy measurements. These properties can be broadly separated into three different categories; bias, noise and sensitivity.

4.1.1 Bias

When detecting no infrared radiation (IR), the sensor should indicate a near-zero temperature, as the sensor's method of determining temperature involves measuring IR. If in such conditions the temperatures indicated are non-uniform, that suggests that the sensor has some level of bias in its measurements. We attempted to investigate the possibility of such bias by performing thermal captures of the night sky. While this does not completely eliminate the IR, it does remove a significant proportion of it.

To test this, the thermal sensor was exposed to the night sky at a capture rate of 1 Hz for 4 minutes, with the sensing results combined to create a set of means and standard deviations for the pixels at "rest". The average temperature detected was 11.78 °C, with the standard deviation remaining less than 0.51 °C over the entire exposure period. The resultant mean thermal map (Figure 4.1)

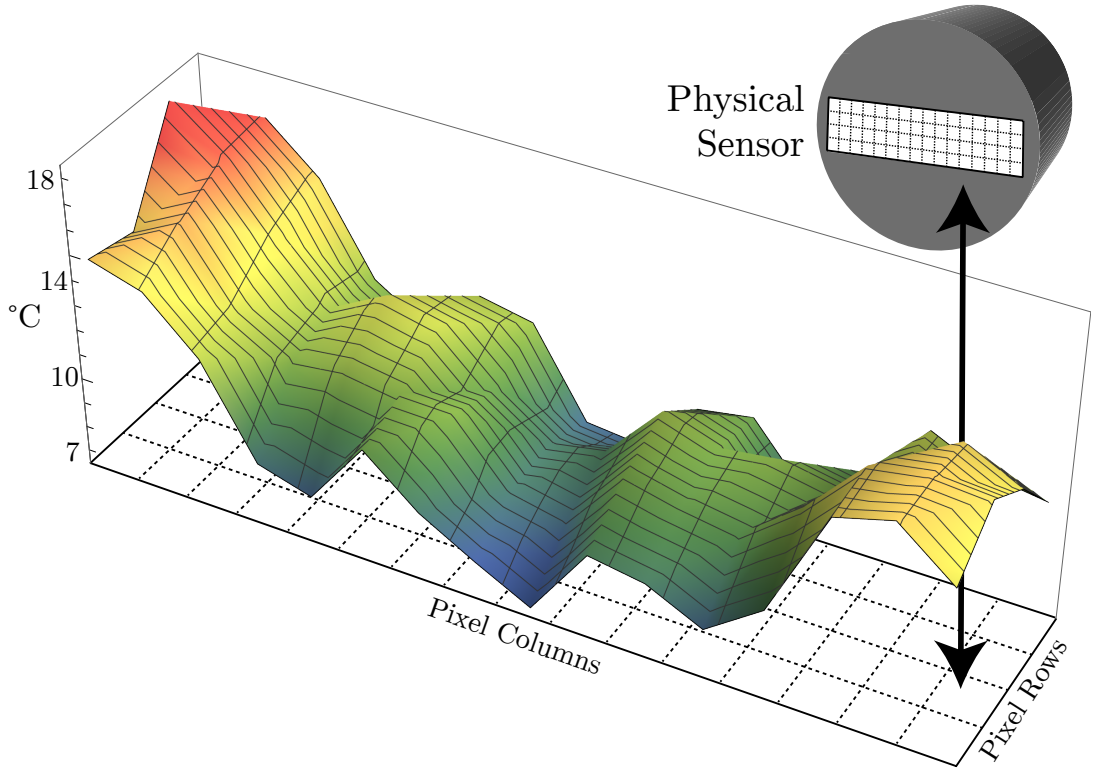


Figure 4.1: Mean values of 4 minute night sky thermal capture plotted over sensor's 16×4 grid

shows that the four center pixels maintain a similar temperature around 9 °C, with temperatures beginning to deviate as they became further from that point.

The most likely cause of bias is related to the physical structure of the sensor. The *Melexis* is a rectangular sensor which has been placed inside a circular tube. Due to this physical arrangement, the sides of this rectangular sensor will be significantly closer to these edges than the center. If the sensor's casing is at an ambient temperature higher than the measurement data (as they likely were in this case) thermal radiation from the sensor package itself could provide significant enough to cause the edges to appear warmer than the observed area of the sky. This effect could be controlled for by cooling the sensor package to below that of the ambient temperature being measured, however, this is not a concern in this project, as the method of calculating a thermal background will compensate for any such bias provided it remains relatively constant, which if our hypothesis is correct, it should.

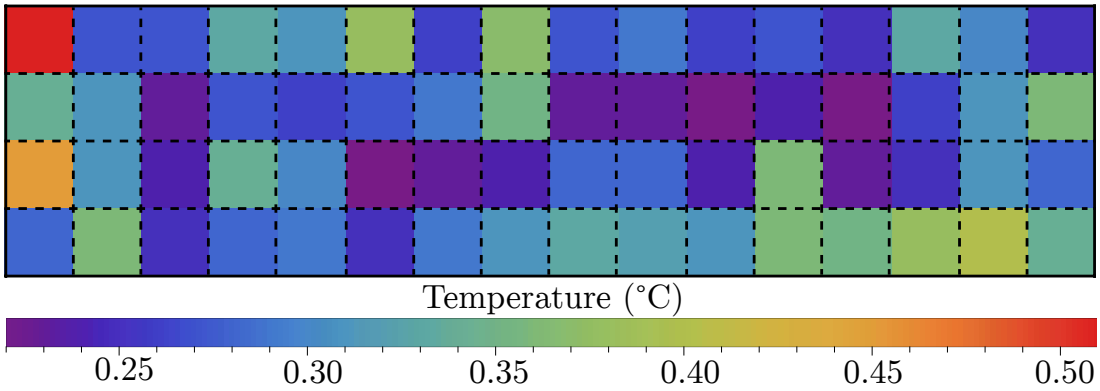


Figure 4.2: Standard deviation of 4 minute night sky thermal capture plotted over sensor's 16×4 grid

4.1.2 Noise

One of the primary features of the *Melexis* is the ability to sample the thermal data at a variety of sample rates between 0.5 Hz and 512 Hz. It was noted in preliminary experimentation that a higher sample rate appeared to result in noisier temperature readings. As our experiments focus on separating objects of interest from a thermal background, it is important to determine if this noise would post an issue for our use case.

Figure 4.3 plots one of the central pixels of the sensor in a scenario where it is detecting a background, and when it is viewing a person, at the five different sample rates achievable with the current hardware configuration. We can see in these plots that the data becomes significantly more noisy as the sample rate increases, and we can also conclude that the sensor uses a form of data smoothing at lower sample rates, as the variance in data increases with sample rate. If the sample rate were to increase beyond a certain threshold, it is likely that the ability for the sensing system to disambiguate between objects of interest and the background would diminish.

In the 0.5 Hz case, the third standard deviation above background ($3\sigma_b$) is 6.4 °C below the minimum occupant value ($\min(o)$) detected. As the noise increases, this gap slowly decreases, with 5.75 °C for 1 Hz, 5.53 °C for 2 Hz, 4.48 °C for 4 Hz, and finally 3.15 °C for 8 Hz. In none of these cases is $3\sigma_b \geq \min(o)$, which would completely rule that sample rate out.

Based on the data, noise will not pose any issue as the slowest sampling rate of 0.5 Hz is sufficient for the system's needs, and shows a sufficiently large gap between occupant and background temperatures.

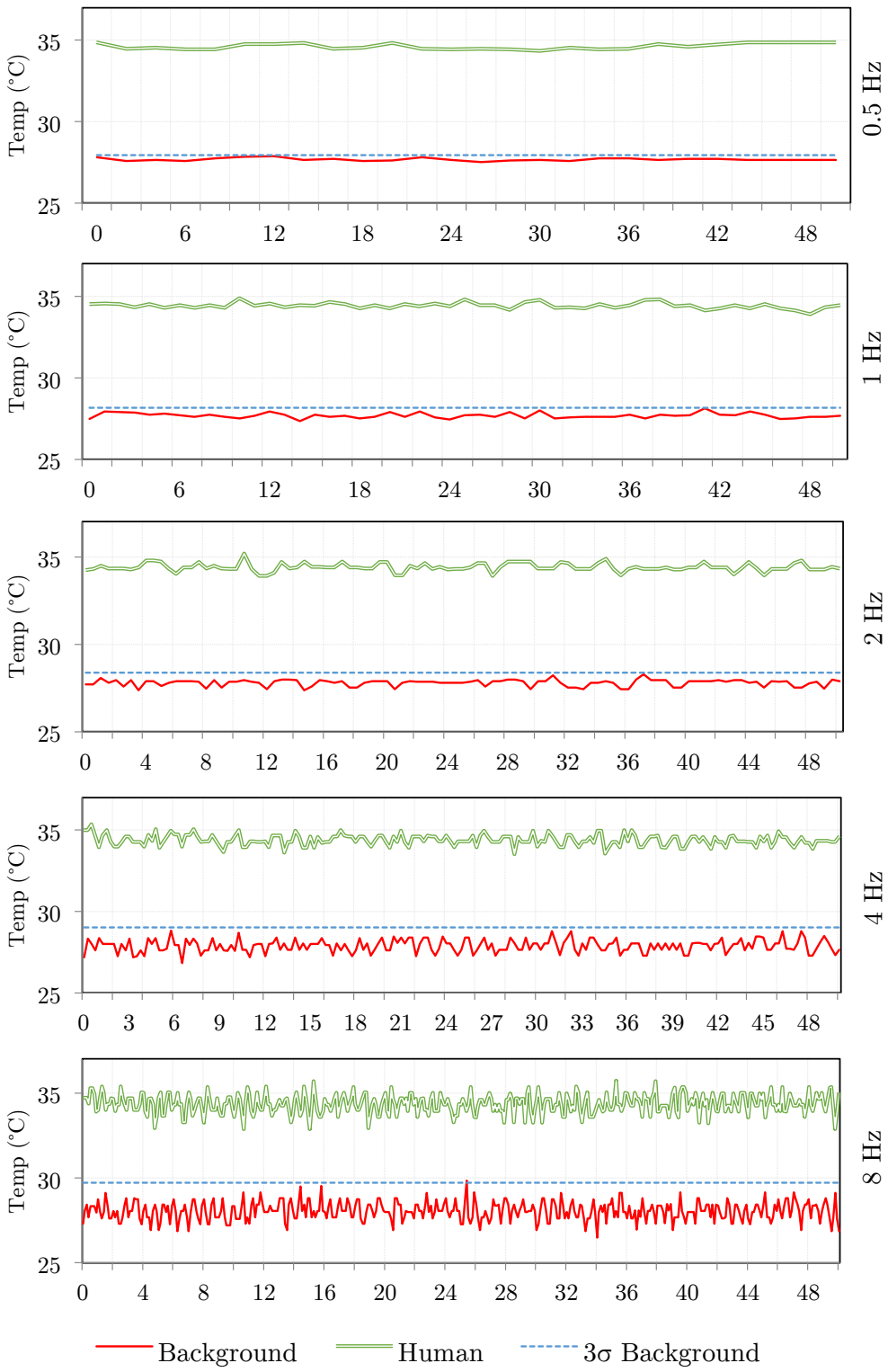


Figure 4.3: Plot of occupant and background sensor noise at sampling speeds 0.5 Hz – 8 Hz

4.1.3 Sensitivity

The *Melexis* is a sensor composed of 64 independent non-contact digital thermopiles, which measure infrared radiation to determine the temperature of objects. While they are bundled in one package, the sensor's block diagram discussed previously (Figure 3.5) shows that they are in fact wholly independent sensors placed in a grid structure. This has important effects on the properties of the data that the *Melexis* produces.

Figure 4.4 shows a smoothed temperatures graph of six of the sensor's central pixels as a hot object is moved from left to right at an approximately constant speed. One of the most interesting phenomena in this graph is the variability of the object's detected temperature as it moves "between" two different pixels; there is a noticeable drop in the objects detected temperature. Each pixel appears to exhibit a bell-curve like line, with the detected temperature increasing from the baseline and peaking as the object enters the center of the pixel, and the detected temperature similarly decreasing as the object leaves the center.

This phenomenon has several possible causes. One likely explanation is that each individual pixel detects objects radiating at larger angles of incidence to be colder than they actually are: As the object enters a pixel's effective field of view, it will radiate into the pixel at an angle that is at the edge of the pixel's ability to sense, with this angle slowing decreasing until the hot object is directly radiating into the pixel's sensor, causing a peak in the temperature reading. As the object leaves the individual element's field of view, the same happens in reverse.

This phenomena is not anticipated to impact our intended use case, as it only strongly affects objects at pixel or sub-pixel size. In our experimental conditions the sensor will not be sufficiently distant that humans could be detected as such sizes.

4.2 Classification

With a greater understanding of the prototype's caveats, it is now possible to gather both thermal and visual data in a synchronized format. This data can be collected and used to determine the effectiveness of the machine learning classification algorithms used. Due to the prototype's technical similarly to ThermoSense [7], a similar set of experimental conditions will be used, with a comparison against ThermoSense being used as a benchmark. To this end, several experiments were devised, each of which had its data gathered and processed in accordance with the same general process, outlined in Figure 4.5 and discussed

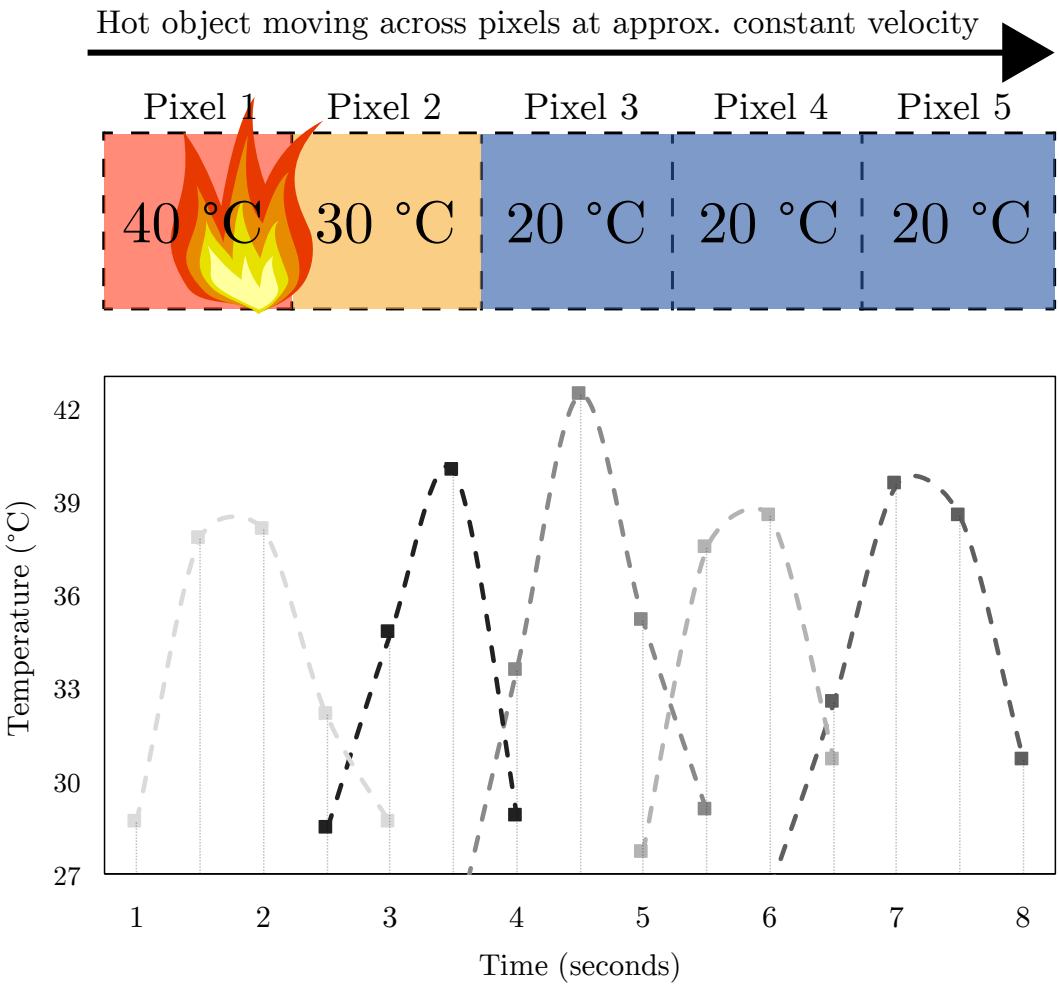


Figure 4.4: Temperature plot of five of the MLX90620's pixels as a hot object moves across them at a constant velocity

in more detail in this section.

4.2.1 Data gathering

As the camera and the Arduino are directly plugged into the Raspberry Pi, all data capture is performed on-board through SSH, with the data being then copied off the Pi for later processing. To perform this capture, the main script used is `cap_pi_synced.py`.

`cap_pi_synced.py` takes two parameters on the command line; the name of the capture output, and the number of seconds to capture. The script initializes the `picamera` library, then passes a reference to it to the `capture_synced` function within the `Visualizer` class. The class will then handle sending commands to the Arduino to capture data in concert with taking still frames with the Raspberry Pi's camera.

When the script runs, it creates a folder with the name specified, storing both the thermal capture inside a file named `output_thermal.hcap`, and a sequence of files with the format `video-%09d.jpg` corresponding to each visual capture frame.

4.2.2 Data labeling

Once this data capture is complete, the data is copied to a computer with GUI support for labeling. The utility `tagging.py` is used for this stage. This script is passed the path to the capture directory, and the number of frames at the beginning of the capture that are guaranteed to contain no motion. This utility will display frame by frame each visual and thermal capture together, as well as the computed feature vectors (based on a background map created from the first n frames without motion).

The user is then required to press one of the number keys on their keyboard to indicate the number of people present in this frame. This number will be recorded in a file called `truth` in the capture's directory. The next frame will then be displayed, and the process continues. This utility enables the quick input of the ground truth of each capture, which is necessary for the classification stage.

4.2.3 Feature extraction and data conversion

Once the ground truth data is available, it is now possible to utilize the data to perform various classification tests. For this, we use version 3.7.12 of the

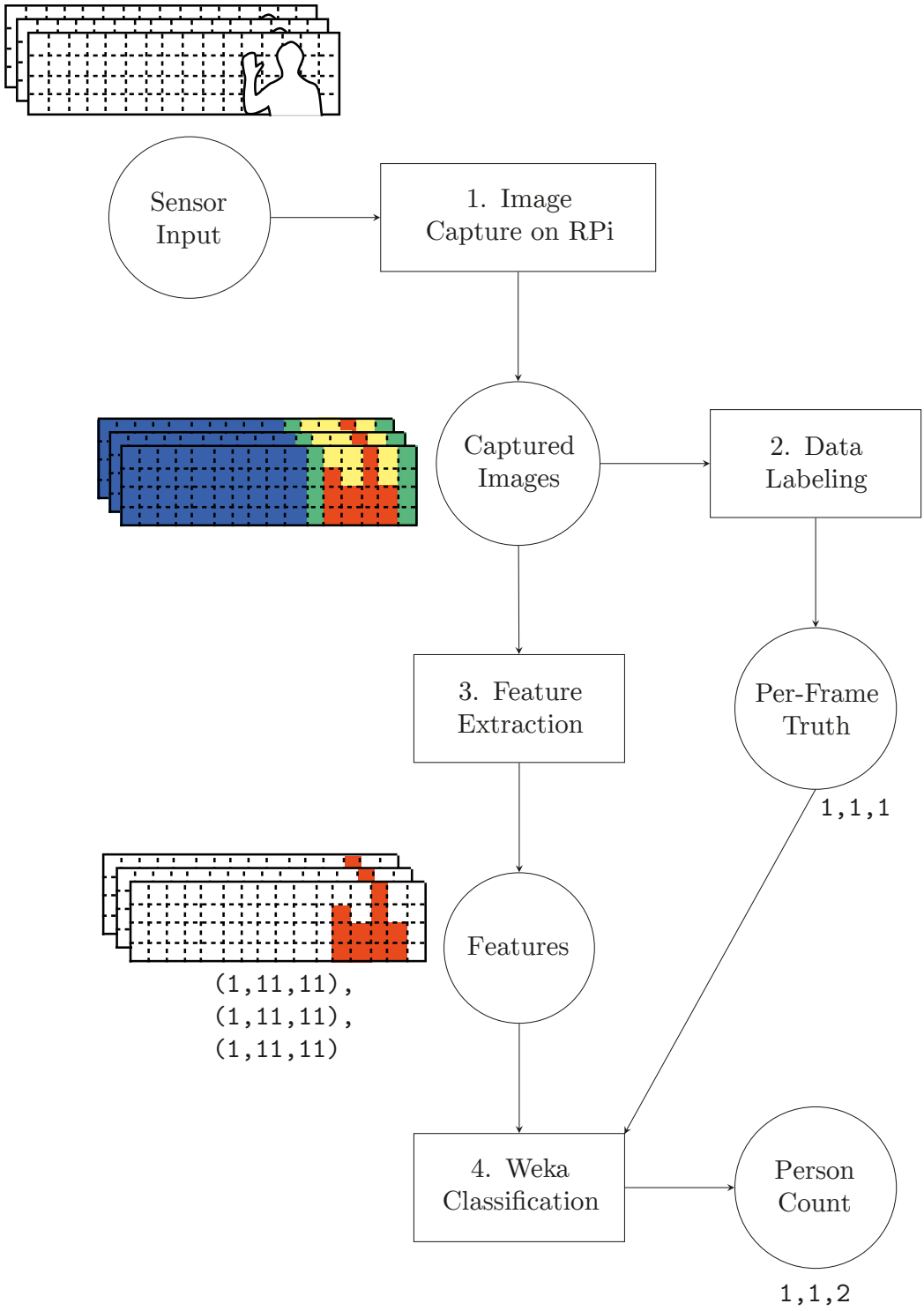


Figure 4.5: Process flow diagram for turning raw sensor input into occupancy estimates

open-source Weka toolkit [25], which provides easy access to a variety of machine learning algorithms and the tools necessary to analyze their effectiveness.

We collate and export the ground truth and extracted features from multiple captures to a Comma Seperated Value (CSV) file for processing with Weka. `weka_export.py` takes two parameters, a comma-separated list of different experiment directories to pull ground truth and feature data from, and the number of frames at the beginning of each capture that can be considered as "motionless." With this information, a CSV-file file is generated.

4.2.4 Running Weka Tests

Once the CSV file is generated, it is then possible process this file through Weka. Weka provides a variety of machine learning classification algorithms to choose from. To investigate ThermoSense's results, we replicate their algorithmic implementations in Weka. We discuss the operation of each of these algorithms generally in Chapter A. In total, Thermosense uses three machine learning classifiers:

Firstly, they use a neural network with a hidden layer of five neurons, with a sigmoid activation function for the hidden layer and a linear activation function for the output layer. They test only the one, two and three person cases, relying on their Passive Infrared Sensor (PIR) to detect the zero person case. They use 70% of their data for training the neural net, 15% for testing the net and the final 15% for validating their results. ThermoSense conducts tests interpreting the number of people as a numeric attribute.

We use Weka's "MultilayerPerceptron" neural network to recreate this, which creates a hidden layer of $(attributes + classes)/2$ (three) by default, however we manually reconfigure this to be one hidden layer of five neurons, like ThermoSense. It uses a sigmoid activation function for all neurons, except in the case that a numerical answer is to be predicted, in which case like ThermoSense, it uses a linear activation function for the output layer.

Secondly, they use a k-Nearest Neighbors (where $k = 5$) with the distance between features being determined by their Euclidean distance. For determining the class label, higher weightings are given to training points inversely to their distance from the point being classified.

Weka's "iBk" function is used to perform a KNN calculation, configuring `distanceWeighting` to be "Weight by 1-distance" and `KNN` to be 5, to make the classification as similar in function to the ThermoSense approach as is possible. ThermoSense does not specify what validation technique they used, so we elected

to use a 10-fold cross-validation.

Thirdly, they use a Linear Regression model of $y = \beta_A A + \beta_S S + \beta$, whereby A is the number of active pixels, S is the size of the largest connected component, and the β values represent the corresponding coefficients. They opt to exclude the third feature, the number of connected components, as their testing indicates that excluding it minimizes the Root Mean Squared Error (RMSE) further.

We use Weka's "LinearRegression" function, and exclude the `numconnected` attribute from the feature vector list, as ThermoSense does, to attempt to match this approach.

In addition to the above three techniques, we also use nominal versions, in addition to our own techniques (detailed in). Our techniques are predominantly well known, and as with the ThermoSense techniques above, are described in more detail in Chapter A. The one algorithm that isn't particularly well known that we have chosen is the K* algorithm, which we describe in further detail here:

The KStar (K*) algorithm, developed by Cleary and Trigg [9] presents a different approach to k-nearest Neighbors type algorithms. With K* the distance used to compare similar points is not the Euclidean distance, but rather an entropic distance, a measure of how much effort is required to convert one example into another. This has several positive effects; it makes the algorithm more robust to missing values, and also it enables the classifier to output either a numeric or nominal result.

We have decided to use K* as one of our classification algorithms as it presents an interesting and different approach to the more well known algorithms above, and also allows the investigation of KNN-like techniques in the numeric area. K* is present in Weka as "KStar," and we will opt to use it in its default state.

To help maximize the efficiency of the classification tasks, we use the Weka's Knowledge Flow interface, which provides a drag-and-drop method of creating complex input and output schemes involving multiple different classification algorithms. We generate an encompassing flow that accepts an input CSV file of the raw data, and performs all numeric and nominal classification at once, returning a text file with the results of each of the different classification techniques run. The Knowledge Flow's structure can be seen in Chapter B. To further automate this, a Jython script, `run_flow.py` is used to call the flow through Weka's Java API. After this is complete, the script then runs a series of regular expressions on the output text data to generate a summary CSV file with the relevant statistical values.

For those tests that are "nominal," the `npeople` attribute was interpreted as nominal using the "NumericToNominal" filter, which creates a class for each

value detected in `npeople`'s columns. For those tests that are "numeric," `npeople` is left unchanged, as by default all CSV import attributes are interpreted as numeric. For all tests where not specifically stated otherwise, we use 10-fold cross-validation to validate our results.

As the data we are using is based on real experiments, the number of frames which are classified as each class may be unbalanced, which could in turn cause a bias in the classification results. We found that in most cases, the data that most unbalanced the set was that of the zero case, as it was the case most present in the data. As ThermoSense previously demonstrated, the use of the PIR alone allows for determining the zero/not-zero case effectively without classification algorithms. Due to this, we attempt to rebalance our dataset by excluding all zero cases from the data Weka receives.

4.2.5 Classifier Experiment Set

In our first set of experiments, a scene was devised in accordance with Figure 4.6 that attempted to sense people from above, as did ThermoSense. The prototype was set up on the ceiling, pointing down at a slight angle to both prevent the interference of the mounting pole (see Figure 3.3) and to increase thermal mass detected. For ease of use, the prototype was powered by mains power, and was networked with a laptop for command input and data collection via Ethernet. This set of experiments involved between one and three people being present in the scene, moving in and out in various ways in accordance with the following scripts.

The first four sub-experiments involved people standing;

- Sub-exp 1: One person walks in, stands in center, walks out of frame.
- Sub-exp 2: One person walks in, joined by another person, both stand there, one leaves, then another leaves.
- Sub-exp 3: One person walks in, joined by one, joined by another, all three stand there, one leaves, then another, then another.
- Sub-exp 4: Two people walk in simultaneously, both stand there, both leave simultaneously.

The latter five sub-experiments involved people sitting;

- Sub-exp 5: One person walks in, sits in center, moves to right, walks out of frame.

- Sub-exp 6: One person walks in, joined by another person, both sit there, they stand and switch chairs, one leaves, then another leaves.
- Sub-exp 7, 8: One person walks in, joined by one, joined by another, they all sit there, one leaves, one shuffles position, then another leaves, then another. (x2)
- Sub-exp 9: Two people walk in, both sit there simultaneously, both leave simultaneously.

In these experiments people moved slowly and deliberately, making sure there were large pauses between changes of action. The people involved were of average height, wearing various clothing. The room was cooled to 18 degrees for these experiments.

Each experiment was recorded with a thermal-visual synchronization at 1 Hz over approximately 60-120 second intervals. Each experiment had 10-15 frames at the beginning where nothing was within the view of the sensor to allow the thermal background to be calculated. Each frame generated from these experiments was manually tagged with the ground truth value of its number of occupants using the tagging script discussed in Subsection 4.2.2. A breakdown of the number of frames collected for each occupancy type can be found in Table 4.2.

The resulting features and ground truth were combined and exported to CSV allowing Weka to analyze them. This data was analyzed with the feature vectors always being considered numeric data and with the ground truth considered both numeric and nominal.

Type	**Attribute**	**Weka Class & Parameters**
Neural Network (ANN)	Nominal, Numeric	`weka.classifiers.functions .MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 15 -S 0 -E 20 -H 5`
k-nearest Neighbors (KNN)	Nominal, Numeric	`weka.classifiers.lazy.IBk -K 5 -W 0 -F -A "weka.core.neighboursearch .LinearNNSearch -A \"weka.core .EuclideanDistance -R first-last\""`
Naive Bayes	Nominal	`weka.classifiers.bayes.NaiveBayes`
Support Vector Machine (SVM)	Nominal	`weka.classifiers.functions.SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions .supportVector.PolyKernel -C 250007 -E 1.0"`
C4.5 Decision Tree	Nominal	`weka.classifiers.trees.J48 -C 0.25 -M 2`
K*	Nominal, Numeric	`weka.classifiers.lazy.KStar -B 20 -M a`
Linear Regression	Numeric	`weka.classifiers.functions .LinearRegression -S 0 -R 1.0E-8`
0-R	Nominal, Numeric	`weka.classifiers.rules.ZeroR`

Table 4.1: Weka parameters used for different classifications algorithms

Occupants	**Instances**	**% (incl. 0)**	**% (excl. 0)**
0	444	42.2	
1	252	24.0	41.5
2	291	27.7	47.9
3	64	6.1	10.5

Table 4.2: Breakdown of Classification Experiment Set data by true number of occupants

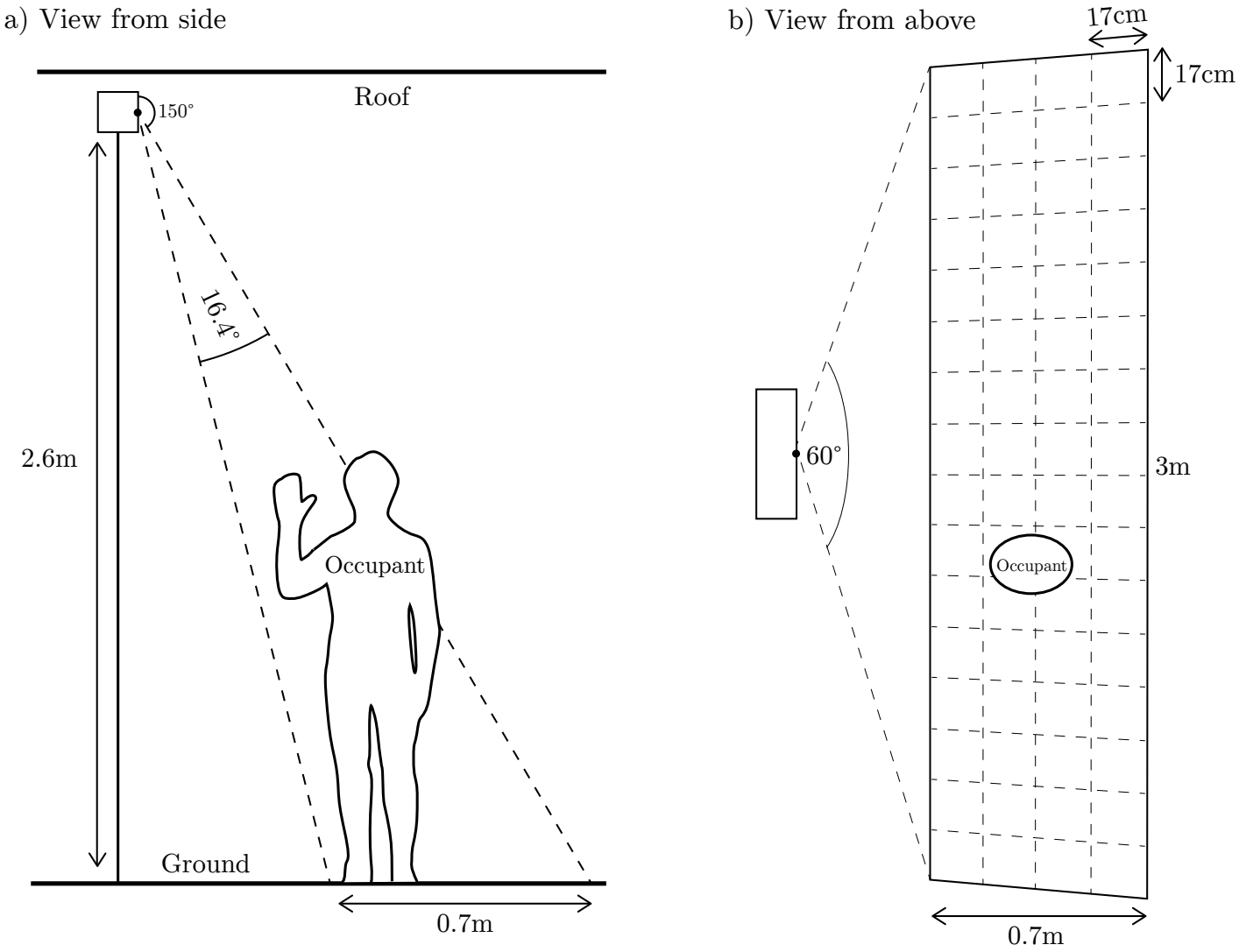


Figure 4.6: Classifier Experiment Set Setup (measurements approximate)

4.3 Results

4.3.1 Classification

Significant care was taken to ensure that the same classification parameters were used between our experiments and those performed in ThermoSense to provide as accurate as possible a comparison between our results. However, there were some ambiguities with the ThermoSense results that have made it more difficult to determine which parameters to choose. In particular, with reference to the k-Nearest Neighbours tests (KNN), it was ambiguous within the ThermoSense paper as to if they had elected to use a nominal classification or a numeric classification for this data.

Because of this, four tests were performed overall to replicate the ThermoSense results as closely as possible; KNN tests for both numeric and nominal representations of data, a Multi-Layer Perceptron numeric test (MLP) and a Linear Regression numeric test (Lin Reg). With these tests we found that our prototype did not achieve comparable results. ThermoSense reported correlation coefficients (r) of around 0.9 for their MLP and Lin Reg tests, however we could not replicate these results, with our best being 0.69 and 0.59 respectively. We were also unable to achieve the low Root Mean Squared Errors (RMSEs) reported by ThermoSense, with their RMSEs for KNN, MLP and Lin Reg being 0.346, 0.385 and 0.409 respectively, while ours were 0.364 (KNN Nominal Case), 1.123 (KNN Numeric Case), 0.592 (MLP) and 0.525 (Lin Reg). Our numeric KNN test performed worse than the 0-R benchmark for numeric tests, indicating a very poor result, with it achieving an RMSE of 1.123 vs. the 0-R's 0.651.

For our own proposed nominal classification algorithms, our accuracies were significantly improved, and in some cases exceeded the RMSEs reported by ThermoSense. Within our dataset, the K* and C4.5 algorithms were most accurate, with accuracies of 82.56% and 82.39% respectively. They both achieved RMSEs lower than the best achieved by ThermoSense, with their 0.304 and 0.314 a significant improvement on ThermoSense's KNN RMSE of 0.346.

Following down the ranking, our nominal MLP performed next best, with an accuracy of 77.14%, and an RMSE of 0.362, which is slightly higher than ThermoSense's best result. Following, the Support Vector Machine (SVM) implementation achieved a relatively poor accuracy of 67.18% with an RMSE of 0.398, and finally the Naive Bayes (N. Bayes) approach, achieved the worst accuracy of 63.59% with an RMSE of 0.405. None of these techniques however achieved an RMSE or accuracy worse than our 0-R benchmark, which achieved an RMSE of 0.442 and an accuracy of 49.74%.

Classifier	**RMSE**	**Precision (%)**	**Correlation (r)**
ThermoSense Actual			
KNN[1]	0.346		
Lin Reg	0.385		0.926
MLP	0.409		0.945
ThermoSense Replication			
KNN[1]	0.364	65.65	
MLP	0.592		0.687
Lin Reg	0.525		0.589
KNN[1]	1.123		0.377
Numeric			
K*	0.423		0.760
0-R	0.651		-0.118
Nominal			
K*	0.304	82.56	
C4.5	0.314	82.39	
MLP	0.362	77.14	
SVM	0.398	67.18	
N. Bayes	0.405	63.59	
0-R	0.442	49.74	

[1]: Included zero in training data
%: Precision, for a nominal test
r: Correlation coefficient, for a numeric test

Table 4.3: Classifier Experiment Set Results

In our sole numeric choice of K*, we found that it achieved a better correlation than any ThermoSense technique, with $r = 0.760$. Additionally, its RMSE of 0.423 was also superior.

Connected		Power	
MLX	**PIR**	**mA**	**V**
✓	✓	52	4.92
✓	✗	52	4.92
✗	✓	46	4.92
✗	✗	46	4.92
Power Saving[1]		34	4.92

[1]: `SLEEP_MODE_PWR_DOWN` AVR mode on Arduino
MLX: Melexis MLX90620 (*Melexis*)
PIR: Passive Infrared Sensor (PIR)

Table 4.4: Energy Consumption Results

4.3.2 Energy Efficiency

A YZXStudio USB 3.0 Power Monitor was used to measure power consumed by the Pre-Processing and Sensing tier together while experimenting, as in a more advanced prototype, they would be envisioned to be run on battery power. This was done by connecting the Arduino's USB cable to the Monitor, and the Monitor to a computer. It was calculated (see Table 4.4) that the average power consumption was 52 mA at 4.92 volts with a sample rate of 0.5 Hz, while continuously outputting data over USB Serial. This power consumption did not vary measurably between sample rates.

To determine the draw from the PIR and Thermal Detector Array (TDA), we disconnected all sensors from the Arduino, and ran the power measurement again. The same code was run on the Arduino. This time we received a result of 46 mA for all sample rates. We found that this power consumption appeared to be from the *Melexis*, as adding/removing the PIR had negligible effects on power consumption. Power consumption also did not appear to vary depending on the temperature of the objects being detected, or the motion properties of the scene. We can then conclude that the sensors themselves draw around 6 mA of power.

We also ran the Arduino in the most aggressive power saving mode available on the hardware, and measured how much power was consumed with and without sensors attached. It appears that this power saving mode disables all sensor power, as in both cases a draw of 34 mA was measured. This represents a power saving of 12 mA from the base Arduino power consumption.

4.4 Discussion

4.4.1 Classification

As discussed in Subsection 4.3.1 our prototype achieves positive results. However, our results indicate that there is a fundamental difference between our set of experiments and those performed by ThermoSense. None of our attempts to replicate their results succeeded, with every replicated result being significantly worse than that of ThermoSense. The most likely reason for this is that the differences in the field of view of the Melexis MLX90620 (*Melexis*) when compared to the Grid-EYE is significant enough to affect the suitability of the algorithms used. In particular the *Melexis* created far more instances of partial people within the sensed region. This presents a key caveat for any future researchers attempting to reapply ThermoSense's methodology to a different sensor.

With the specific classification techniques that we choose to test, we found quite a variation in their success. Our best techniques, K* and C4.5, were very similar in result and there is quite a gap to the third-best technique, the Multi-Layer Perceptron. We can only speculate as to why this is the case, but it is notable that both K* and C4.5 use entropy measures to make decisions,.

Examining the plots of the Euclidean distance for the features found in Figure 4.7, it is apparent there is no obvious separation between the different classes, with the three sets of values overlapping significantly. This explains k-Nearest Neighbors (KNN) and Support Vector Machine's (SVM) poor performance with our data.

In the case of KNN, a selection of the nearest neighbors are not in many cases going to be indicative of the feature vector's class membership. A similar idea applies n the case of SVM, the aforementioned plots suggest that there may be no clear hyperplane through which the different classes of data can be separated.

The success of entropy-based methods suggests that the separation of classes is less of an issue when distance is measured with entropy, as in the case of K*, or when entropy is used to divide the dataset, as in the case of C4.5.

Due to the white box nature of decision trees, it is possible for us to further example the tree generated by the C4.5 algorithm to determine how sensible the branches are. We can see from Listing 4.1 that C4.5 has chosen to use the number of active pixels as the primary split within the tree, with the size of the largest connected component as the secondary split, and finally the number of connected components as the tertiary split.

Our worst selected technique, unsurprisingly, was Naive Bayes. It is unsur-

prising that it performed so poorly, as the "Naive" part of the technique is an assumption of independence between the different features input, which is clearly false with our features. All three of our features relate to the same underlying graph and are most definitely correlated with each other.

By using the K* or C4.5 machine learning algorithm, we are confident that the prototype could achieve appropriate levels of accuracy for its occupancy goals.

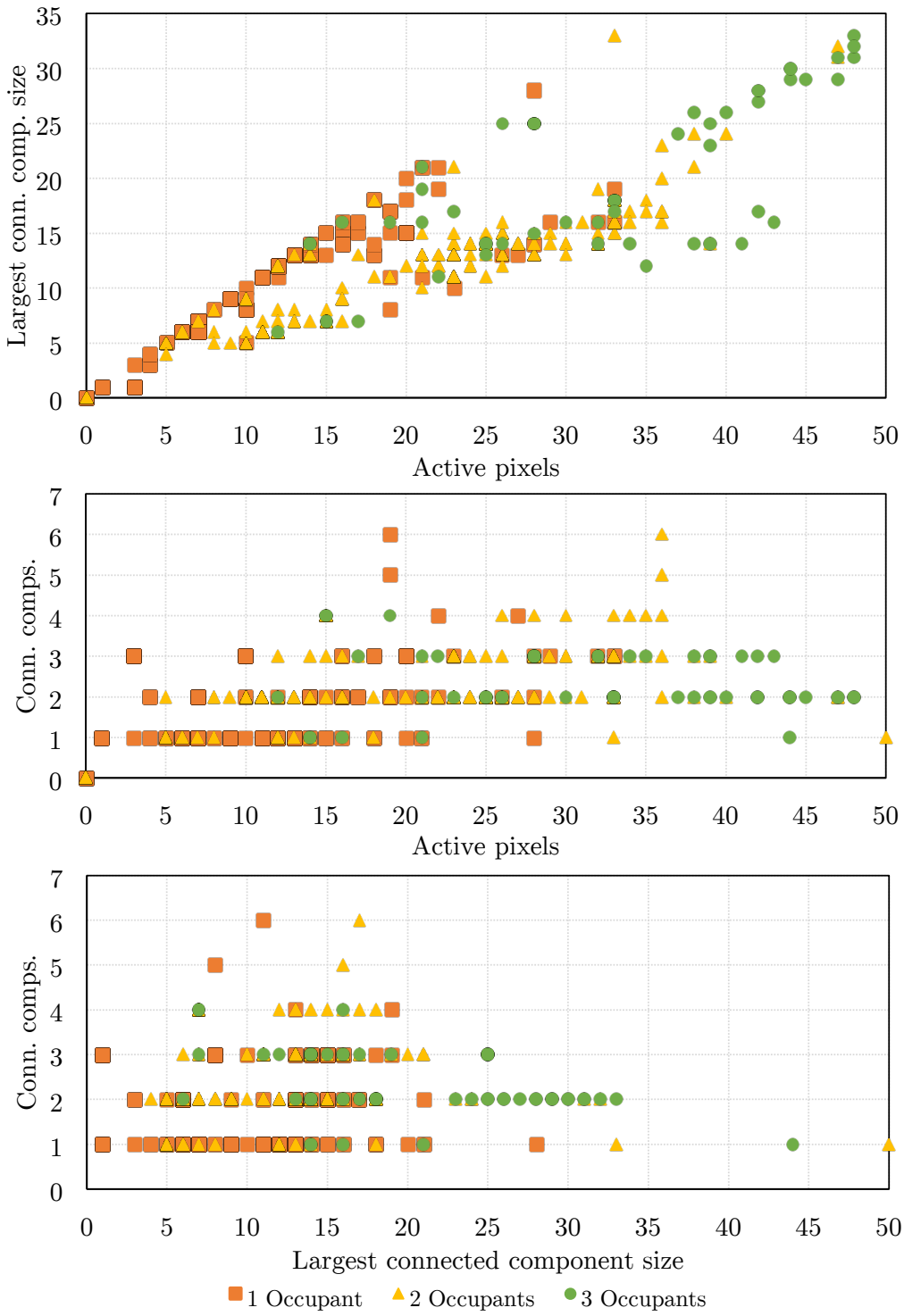


Figure 4.7: Plot of three features against each other with occupancy truth values

```
numactive <= 20
|   sizeconnected <= 10
|   |   numactive <= 10
|   |   |   numconnected <= 2
|   |   |   |   numactive <= 9
|   |   |   |   |   numactive <= 5
|   |   |   |   |   |   numactive <= 4: 1 (22.96/5.6)
|   |   |   |   |   |   numactive > 4: 2 (16.0/3.0)
|   |   |   |   |   numactive > 5
|   |   |   |   |   |   sizeconnected <= 5: 2 (2.0)
|   |   |   |   |   |   sizeconnected > 5: 1 (67.0/6.0)
|   |   |   |   numactive > 9: 2 (22.0/3.0)
|   |   |   numconnected > 2: 1 (20.0)
|   |   numactive > 10: 2 (68.0/6.0)
|   sizeconnected > 10: 1 (132.04/22.4)
numactive > 20
|   sizeconnected <= 24
|   |   numactive <= 36
|   |   |   sizeconnected <= 15: 2 (144.0/17.0)
|   |   |   sizeconnected > 15
|   |   |   |   numactive <= 22
|   |   |   |   |   numactive <= 21
|   |   |   |   |   |   numconnected <= 1: 1 (3.0/1.0)
|   |   |   |   |   |   numconnected > 1: 3 (2.0)
|   |   |   |   |   numactive > 21: 1 (2.0)
|   |   |   |   numactive > 22: 2 (44.0/14.0)
|   |   numactive > 36: 3 (12.0/4.0)
|   sizeconnected > 24
|   |   numconnected <= 1: 2 (4.0/2.0)
|   |   numconnected > 1: 3 (24.0/2.0)

# numactive: Number of active pixels in frame
# sizeconnected: Number of active pixels in largest connected component
# numconnected: Number of connected components in frame
```

Listing 4.1: C4.5 Decision tree generated by Weka's J48 implementation from the Classification Experiment Set data

4.4.2 Energy Efficiency

As discussed in subsection 4.3.2, the prototype is comparatively power hungry, as the ThermoSense system achieved a power consumption of around 5mA, approximately 10 times less. This can be attributed to the TMote Sky being a very energy efficient platform. However, within the scope of this project we restricted our adherence to low-power consumption to an very basic level, rather than committing significant time to reducing this consumption through software or advanced hardware means.

ThermoSense indicated that for each of their nodes, which consumed approximately 5mA each, there was a combined 6000 mAh of battery capacity available. They indicated that at a sample rate of 0.2 Hz (once every five seconds) their prototype was able to run for approximately three weeks.

Our prototype can boot and take one measure in approximately 100 ms, indicating it would need to be awake for 2% of each five second window to perform a one sample. This calculates to an average power consumption of approximately 35 mA, assuming our current Arduino power draw. With a similar battery capacity, we calculate that our prototype would last approximately one week.

However, we contend that in a real world prototype, a sample rate of 0.01 Hz (once per 100 seconds) would be sufficient, with even slower sample rates being a possibility.Similarly, the Arduino used was not targeted towards the extreme energy efficiency of the TMote Sky. There exist Arduino prototypes that use the same processor (and thereby are fully compatible) that can be constructed easily and reduce base power consumption to 7 mA, without sleep. In sleep mode, the power consumption drops to an impressive 43 µA.

At such a sample rate, the sensor would need to be awake for 0.1% of the sample window, with a calculated consumption rate of 13 mA when active (7 mA base power + 6 mA for components) and 43 µA when sleeping. The average power consumption of the prototype in this scenario is approximately 60 µA. On a 6000 mAh battery pack, such a prototype is estimated to have more than a decade of power, a longer life than the self-discharge rate of the battery itself.

However, this estimation does not account for the radio that would be included in latter versions of a prototype. A commonly used radio used in such devices is the ZigBee. Dementyev, Hodges and Taylor et al. [10] suggest that a ZigBee radio would consume approximately 4.2 µA when sleeping and 9.3 mA when awake, with a minimum wake time of around 270 ms to connect to the other nodes on the network. With a minimum data transfer rate of 20 kb/s and approximately 200 bytes of data to transmit each interval, we can set the upper limit of the radio wake time to be 300 ms.

This brings the prototype's power consumption while awake to approximately 23 mA and 65 µA when asleep. Assuming the entire prototype would need to be awake for the radio transmission period, and the wake interval remains 0.01 Hz, the wake period now accounts for 0.3% of the total period, producing an average power consumption of around 140 µA. With this rate of consumption, we still measure the prototype's battery life in several years.

CHAPTER 5

Conclusions

The smart-home economy continues to grow, with automation being one of the main areas driving growth. The ability to detect occupants present within a space is an important smart-automation feature, with the implications for climate control energy efficiency alone being highly significant.

This project has created an prototype occupancy detection system for such a smart home environment that meets four criteria; Low Cost, Non-Invasive, Energy Efficient and Reliable. This prototype was based upon the ceiling-mounted thermal imaging approach of ThermoSense [7], which after extensive analysis proved to be the best option given our criteria.

5.1 Evaluation of Criteria

5.1.1 Low Cost

One of our primary goals was to create a system that was inexpensive enough that it would be suitable for both office environments with hundreds of rooms, as well as in smart homes for the disabled and elderly, both of which are areas where per-unit cost is an issue.

As discussed in the Design chapter (Chapter 3, subsection 3.1.4), the cost of our proposed sensing system is around $185, on par with the ThermoSense system. Compared with most thermal sensing systems, this is very inexpensive, as devices incorporating thermal imaging can cost in the hundreds, thousands or even tens of thousands of dollars. We admit that $185 is still quite expensive for such a sensing system, when taking into consideration that many would be needed for one home. However, while this is the cost of such components today, this is by no means the cost of them tomorrow. Prices for all of the components involved in this design are falling rapidly, in particular that of the thermal sensor: In the future work section we discuss a sensor that takes the price per pixel from

the $1.25 for the Melexis MLX90620 (*Melexis*) to a mere $0.07.

Right now we are at the stage where this technology is economical for researchers to investigate, but a future where it becomes economical for consumers is approaching fast. We believe by selecting the components that we have at the current price point, we have met the project's goal of low cost.

5.1.2 Non-Invasive

To ensure wide adoption, minimizing privacy concerns is a must. We viewed creating a system with little means by which to surveil occupants as the best way to minimize such concerns.

As discussed in the Literature Review (Section 2.3), we concluded that the *Melexis* provides the best trade-off between accuracy and non-invasiveness of those sensing systems studied. It provides this tradeoff from two different angles; the infrared aspect and the low-resolution aspect.

By sensing in the infrared spectrum, many elements of automatic and manual person identification become more difficult, as many such methods rely on using color information to make such decisions. Similarly, by having the sensor constrained to such a low resolution, it is also quite difficult to perform person or action identification, due to the very little information available.

Through our architectural decisions, we believe that the project's goal of producing a non-invasive sensing system has been achieved.

5.1.3 Reliable

Creating a system that is wholly automated and can detect occupants with a high level of accuracy is important to ensure that climate control and other occupant-driven tasks are reliably executed.

As discussed in Subsection 4.3.1, we were unable to replicate ThermoSense's RMSEs of 0.346, 0.409 and 0.385 with either k-nearest Neighbors, Linear Regression, or Multi-Layer Perceptron respectively. This suggests that the classifiers ThermoSense used were highly sensitive to their sensor's specific properties.

However among our own selected machine learning algorithms, K* and C4.5 achieved accuracies in the 80% range. These algorithms also improved upon ThermoSense's best RMSE with RMSEs of 0.304 and 0.314 respectively. Both of these algorithms leverage entropy measures as a way of partitioning data, suggesting that entropy-based approaches may be particularly suited to our dataset.

By using the K* or C4.5 machine learning algorithm, we are confident that the prototype could achieve appropriate levels of accuracy for its occupancy goals, and believe that the reliability requirements of our project have been met.

5.1.4 Energy Efficient

Finally, as the system would hopefully be suitable for use in existing buildings, we aimed to create a system that could operate efficiently on battery power, as retrofitting power on a roof location would further add to the cost of the sensing system.

As discussed in the Energy Efficiency results (Subsection 4.3.2), with the same sample rate and battery size and while sleeping between samples, our prototype is estimated to last one week compared to ThermoSense's three.

However, we consider the sample rate of once every five seconds used by ThermoSense to be unnecessarily fast, and instead advocate for a sample rate of once every 100 seconds. With such a sample rate, and using a proposed alternative Arduino system with a significantly reduced power draw, we estimate a power draw lower than that of ThermoSense, with the system being to able to last years without charge.

Accordingly, while the current prototype is not as energy efficient as the ThermoSense prototype, the Arduino used is not the most energy efficient available, and the software running on it makes no attempt to sleep the hardware while no processing is occurring. We believe that with such measures, our prototype could achieve similar or better energy efficiency than ThermoSense, and have ensured that our architectural decisions will not restrict such energy saving modifications.

We recommend that in future work, further energy saving measures, such as sleeping between sampling, and using lower power draw Arduinos are investigated to determine how much power it is possible to save.

5.2 Future Work

This project has attempted to explore the area of thermal sensing and occupancy with some depth, and with the developed software and hardware prototype, has laid the foundation for many more projects that build upon this project's work. Some areas of future research are discussed here;

5.2.1 Broader Data Collection

Data collection was constrained to one set of ten experiments. Each of these experiments had the sensing system recording at the same height and the same angle. While this data was varied somewhat through half of the experiments involving sitting occupants (thereby increasing their thermal signature from the sensor's perspective), an exploration of how the results differed based on the sensor's viewing angle, as well as the sensor's distance from the ground would have served to improve the result set.

5.2.2 Different Feature Vectors

Exploring how different subsets of the three current features, or possibly new features derived from the thermal capture, affect the accuracy of the machine learning algorithms may demonstrate interesting results.

5.2.3 Different Classification Algorithms

While the set of explored classification algorithms was significant, there is always room for improvement in that regard. Exploring how different parameters of these algorithms affected the results, and how different algorithms altogether faired would have added significantly to the experimental data. Emerging work in the area of classification algorithm selection and parameter optimization [24] could assist in determining which algorithm suited the Melexis MLX90620 (*Melexis*) the most automatically.

5.2.4 Sub-pixel localization

The characteristics of the *Melexis*'s individual thermopiles, discussed in Subsection 4.1.3, make potentially possible an algorithm for the calculation of the position and size of an object with a sub-pixel accuracy. By exploiting the fact that an object's detected temperature changes based on its sub-pixel position, either in the center or on the boundary of two pixels, it may be possible to further refine the edges of thermal objects detected, and increase the effective resolution and accuracy of the sensor system.

5.2.5 Improving Robustness

One of the main areas of the project that was deemed out of scope was the introduction of a wireless mesh networking architecture to the project. Future prototypes would consist of an many-to-one relationship between the Sensing/Preprocessing tier and the Analysis tier. Exploring the best way to mesh network these components while maintaining all the pre-existing criteria of the project would be involved and useful.

Similarly, the current prototype uses a breadboard design that increases the size of the prototype significantly, as well as reduces the physical robustness of the prototype in the long-term. Converting the *Melexis* and PIR circuit into a printed circuit board that fits onto the Arduino as a shield would both reduce the size of the prototype, as well as improve reliability for the future.

5.2.6 Field-of-view modifications

Several different techniques could be used to improve upon the field-of-view limitations of the *Melexis*, and exploring them and their cost/complexity implications would be useful. The first of these is applying a lens to the sensor, effectively expanding the field-of-view, but at the cost of distorting the image. Compensating for this distortion while maintaining accuracy presents an intriguing problem.

In another direction, using a motor with the *Melexis* to "sweep" a room, and thereby constructing a larger image of the space could also resolve the field-of-view issues. However, this approach also presents problems in stitching the images together in a sensible way, the distortion caused by rotating the sensor, as well as handing cases in which a fast-moving object is represented multiple times in the stitched image.

5.2.7 New Sensors

During this project, an updated version of our sensor, the MLX90621, was released. This version (at a similar price point) doubles the field-of-view in both the horizontal and vertical directions, addressing many of the problems encountered with the size of detection area in low-ceiling rooms. This version offers nearly complete backwards compatibility with the older version. Updating the project code-base to support it and re-running the experiments with the increased field-of-view to determine how much of an improvement it is would be interesting.

In addition to this, significantly higher resolution sensors are beginning to

come to the market. The FLiR Lepton [21], which sells in a development kit for $350, offers an 80×60 pixel sensor with a comparable field-of-view to the Grid-EYE. Exploring the increases in accuracy achievable though such significant increases in resolution would be useful.

5.3 Summary

Fundamentally, in this project we set out to create a sensing system that met our requirements of Low-Cost, Non-Invasiveness, Reliability and Energy Efficiency, and we have indeed created such a sensing system. The work required to achieve this project's goals was extensive, from reviewing the state of occupancy sensing, to developing a hardware prototype and software library, to performing experimentation on the system and its properties, and to validate the reliability and energy efficiency of the prototype with a series of experiments.

This dissertation and accompanying sensing system prototype both validates the methods and results of the ThermoSense paper, discovers key caveats surrounding the ThermoSense approach, and also creates a software and hardware base on which future research into the area of occupancy in thermal imaging can be explored.

Bibliography

[1] Adafruit. 4-channel I2C-safe bi-directional logic level converter - BSS138 (product ID 757). http://www.adafruit.com/product/757. Retrieved January 7, 2015.

[2] Adafruit. PIR (motion) sensor (product ID 189). http://www.adafruit.com/product/189. Retrieved February 8, 2015.

[3] Arduino Forums. Arduino and MLX90620 16X4 pixel IR thermal array. http://forum.arduino.cc/index.php/topic,126244.0.html, 2012. Retrieved January 7, 2015.

[4] Australian Bureau of Statistics. Household water and energy use, Victoria: Heating and cooling. Tech. Rep. 4602.2, 2011. Retrieved October 6, 2014 from http://www.abs.gov.au/ausstats/abs@.nsf/0/85424ADCCF6E5AE9CA257A670013AF89.

[5] Australian Bureau of Statistics. Disability, ageing and carers, Australia: Summary of findings: Carers - key findings. Tech. Rep. 4430.0, 2012. Retrieved April 10, 2015 from http://www.abs.gov.au/ausstats/abs@.nsf/Lookup/D9BD84DBA2528FC9CA257C21000E4FC5.

[6] Balaji, B., Xu, J., Nwokafor, A., Gupta, R., and Agarwal, Y. Sentinel: occupancy based HVAC actuation using existing WiFi infrastructure within commercial buildings. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems* (2013), ACM, p. 17.

[7] Beltran, A., Erickson, V. L., and Cerpa, A. E. ThermoSense: Occupancy thermal based sensing for HVAC control. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings* (2013), ACM, pp. 1–8.

[8] Chan, M., Campo, E., Estève, D., and Fourniols, J.-Y. Smart homes - current features and future perspectives. *Maturitas 64*, 2 (2009), 90–97.

[9] Cleary, J. G., Trigg, L. E., et al. K*: An instance-based learner using an entropic distance measure. In *Proceedings of the 12th International Conference on Machine learning* (1995), vol. 5, pp. 108–114.

[10] Dementyev, A., Hodges, S., Taylor, S., and Smith, J. Power consumption analysis of bluetooth low energy, zigbee and ant sensor nodes in a cyclic sleep scenario. In *Wireless Symposium (IWS), 2013 IEEE International* (2013), IEEE, pp. 1–4.

[11] Erickson, V. L., Achleitner, S., and Cerpa, A. E. POEM: Power-efficient occupancy-based energy management system. In *Proceedings of the 12th international conference on Information processing in sensor networks* (2013), ACM, pp. 203–216.

[12] Fisk, W. J., Faulkner, D., and Sullivan, D. P. Accuracy of CO2 sensors in commercial buildings: a pilot study. Tech. Rep. LBNL-61862, Lawrence Berkeley National Laboratory, 2006. Retrieved October 6, 2014 from `http://eaei.lbl.gov/sites/all/files/LBNL-61862_0.pdf`.

[13] Gupta, M., Intille, S. S., and Larson, K. Adding gps-control to traditional thermostats: An exploration of potential energy savings and design challenges. In *Pervasive Computing*. Springer, 2009, pp. 95–114.

[14] Hailemariam, E., Goldstein, R., Attar, R., and Khan, A. Real-time occupancy detection using decision trees with multiple sensor types. In *Proceedings of the 2011 Symposium on Simulation for Architecture and Urban Design* (2011), Society for Computer Simulation International, pp. 141–148.

[15] Hnat, T. W., Griffiths, E., Dawson, R., and Whitehouse, K. Doorjamb: unobtrusive room-level tracking of people in homes using doorway sensors. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems* (2012), ACM, pp. 309–322.

[16] Kleiminger, W., Beckel, C., Dey, A., and Santini, S. Inferring household occupancy patterns from unlabelled sensor data. Tech. Rep. 795, ETH Zurich, 2013. Retrieved October 6, 2014 from `http://eaei.lbl.gov/sites/all/files/LBNL-61862_0.pdf`.

[17] Kleiminger, W., Beckel, C., Staake, T., and Santini, S. Occupancy detection from electricity consumption data. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings* (2013), ACM, pp. 1–8.

[18] Li, N., Calis, G., and Becerik-Gerber, B. Measuring and monitoring occupancy with an RFID based system for demand-driven HVAC operations. *Automation in construction 24* (2012), 89–99.

[19] Melexis. Datasheet IR thermometer 16X4 sensor array MLX90620. http://www.melexis.com/Infrared-Thermometer-Sensors/Infrared-Thermometer-Sensors/MLX90620-776.aspx, 2012. Retrieved January 7, 2015.

[20] Serrano-Cuerda, J., Castillo, J. C., Sokolova, M. V., and Fernández-Caballero, A. Efficient people counting from indoor overhead video camera. In *Trends in Practical Applications of Agents and Multiagent Systems*. Springer, 2013, pp. 129–137.

[21] SparkFun. FLiR Dev Kit. https://www.sparkfun.com/products/13233. Retrieved April 8, 2015.

[22] Swoboda, K. Energy prices–the story behind rising costs. In *Parliamentary Library Briefing Book - 44th Parliament*. Australian Parliament House Parliamentary Library, 2013. Retrieved February 3, 2015 from http://www.aph.gov.au/About_Parliament/Parliamentary_Departments/Parliamentary_Library/pubs/BriefingBook44p/EnergyPrices.

[23] Teixeira, T., Dublon, G., and Savvides, A. A survey of human-sensing: Methods for detecting presence, count, location, track, and identity. Tech. rep., Embedded Networks and Applications Lab (ENALAB), Yale University, 2010. Retrieved October 6, 2014 from http://www.eng.yale.edu/enalab/publications/human_sensing_enalabWIP.pdf.

[24] Thornton, C., Hutter, F., Hoos, H. H., and Leyton-Brown, K. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (2013), ACM, pp. 847–855.

[25] University of Waikato. Weka. http://www.cs.waikato.ac.nz/ml/weka/. Retrieved March 10, 2015.

APPENDIX A

Classification Algorithms

Here we describe the basic operation of the well-known classification algorithms used to classify our occupancy information. For those algorithms not discussed in detail in Chapter 4, we also provide information on how to implement them in Weka.

A.1 Neural Networks

An Artificial Neural Network (ANN) uses neurons as a model for machine learning. A number of input neurons connected to the feature vectors is fed into another network of neurons (the "hidden layer"), each of which has an activation function which determines what set of inputs will make it fire. This network then connects to a number of output neurons which can be examined to determine the network's predicted result. In the nominal result case, there is one neuron for each possible class (with each neuron being a binary yes/no), and in the numeric result case, there is one neuron without an activation function that outputs a raw numerical estimate. Neural networks can approximate functions of nearly any complexity with sufficient neurons in the correct topology, and are a commonly used classification technique.

A.2 k-nearest Neighbors

A k-nearest Neighbors (KNN) approach uses the topology of the training data as a means to classify future data. For each data point that requires classification, a majority vote of its k nearest neighbors in the training data determines which class it belongs to. KNN is one of the simplest machine learning algorithms, and due to its classification technique, is highly sensitive to classes that overlap.

A.3 Linear Regression

A Linear Regression approach attempts to construct a linear equation to describe the relationship between a dependent variable (in this case, the number of people in the space), and a number of other indicator variables (in this case, the three feature vectors). Generally, the equation takes the form $y = m_1x_1 + ... + m_nx_n + c$, where each of the feature vectors (x_n) is multiplied by a weight (m_n), and then a final constant (c) is added to provide the final prediction.

A.4 Naive Bayes

A Naive Bayes approaches uses a simple application of Bayes' probability theorem to construct a probability of a given value belonging to a given class taking into account what is already known about the distribution of each of the classes in the data set, and the classification of those points that surround the point needing classification. One of the disadvantages of the Naive Bayes approach (and the source of its naivety) is that it assumes independence between each of the variables used for classification.

In our data, the assumption of independence of variables is not correct, as each of the features are different representations of the same underlying data. However, due to Naive Bayes' ubiquity and simplicity, it can be illuminating to see how well a very common but poorly suited classifier fares with our data set. Within Weka, we use the "NaiveBayes" function, which has little by way of configuration, thus is left in its default state.

A.5 Support Vector Machines

Support Vector Machines (SVM) attempt to classify data by trying to find a plane that best separates two classes in a higher dimensional space. They do this by determining "support vectors," which are those data points that lie on the "edge" of the separation between classes, and then finding the plane that maximizes the margin between the two classes being tested. SVM is another common classification technique that we elect to investigate.

For our purposes, we use Weka's "SMO" function, which implements the Sequential Minimal Optimization algorithm, an efficient and recent method for training SVMs. For datasets with more than two classes (such as ours), the "one vs. one" method is used, whereby an SVM is created for each pair of classes,

and then a method of majority voting is used to determine which class is the ultimately correct one.

A.6 Decision Trees

Decision Tree based approaches use a flow-chart of logical conditions which when met cause a data point to be classified as a specific class. Decision Tree classifiers generally use a partitioning approach whereby they split the data using a specific metric to maximize the tree's effectiveness. The advantages of Decision Trees are that they are considered to be "white boxes," meaning that the result that they generate is human readable. This is useful, as in addition to the classifier providing its prediction of which class suits the data best, the tree can also be inspected to determine if the decisions it has extrapolated appear to be sensible, and even tweaked by humans if necessary.

One quite common algorithm for generating decision trees is C4.5, which is implemented by the "J48" function in Weka. C4.5 uses a measure of information gain, a concept rooted in information theory and entropy, to determine when to create splits in the tree. There are few configurable parameters for this approach, and for those we use the Weka defaults.

A.7 0-R

0-R is our final classification algorithm. 0-R is a simple classifier that on nominal prediction will classify all new data as belonging to the category that was most common in the training data, and on numeric prediction will classify all new data as being the mean of all test data. A 0-R classifier, clearly, is not a serious classification technique, however it is useful in establishing a baseline from which to compare all other classification results.

In Weka, the 0-R classifier is known as "ZeroR" and accepts no parameters.

APPENDIX B

Knowledge Flows

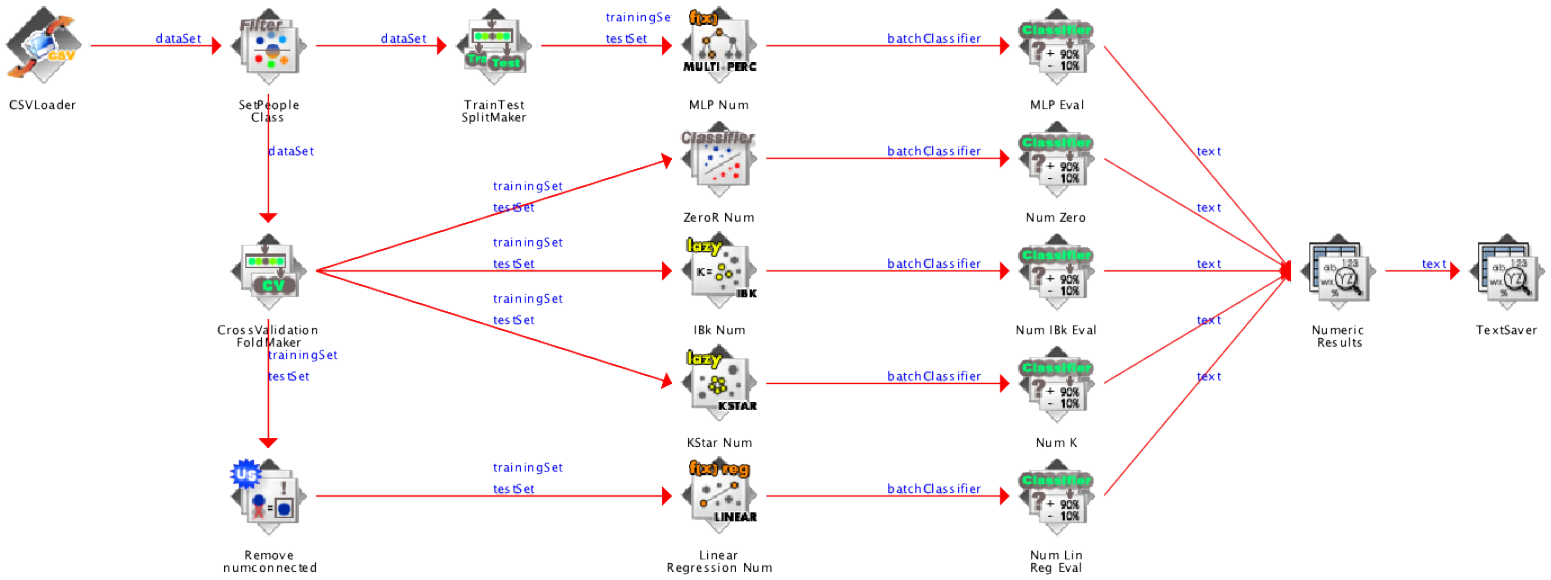


Figure B.1: Numeric knowledge flow

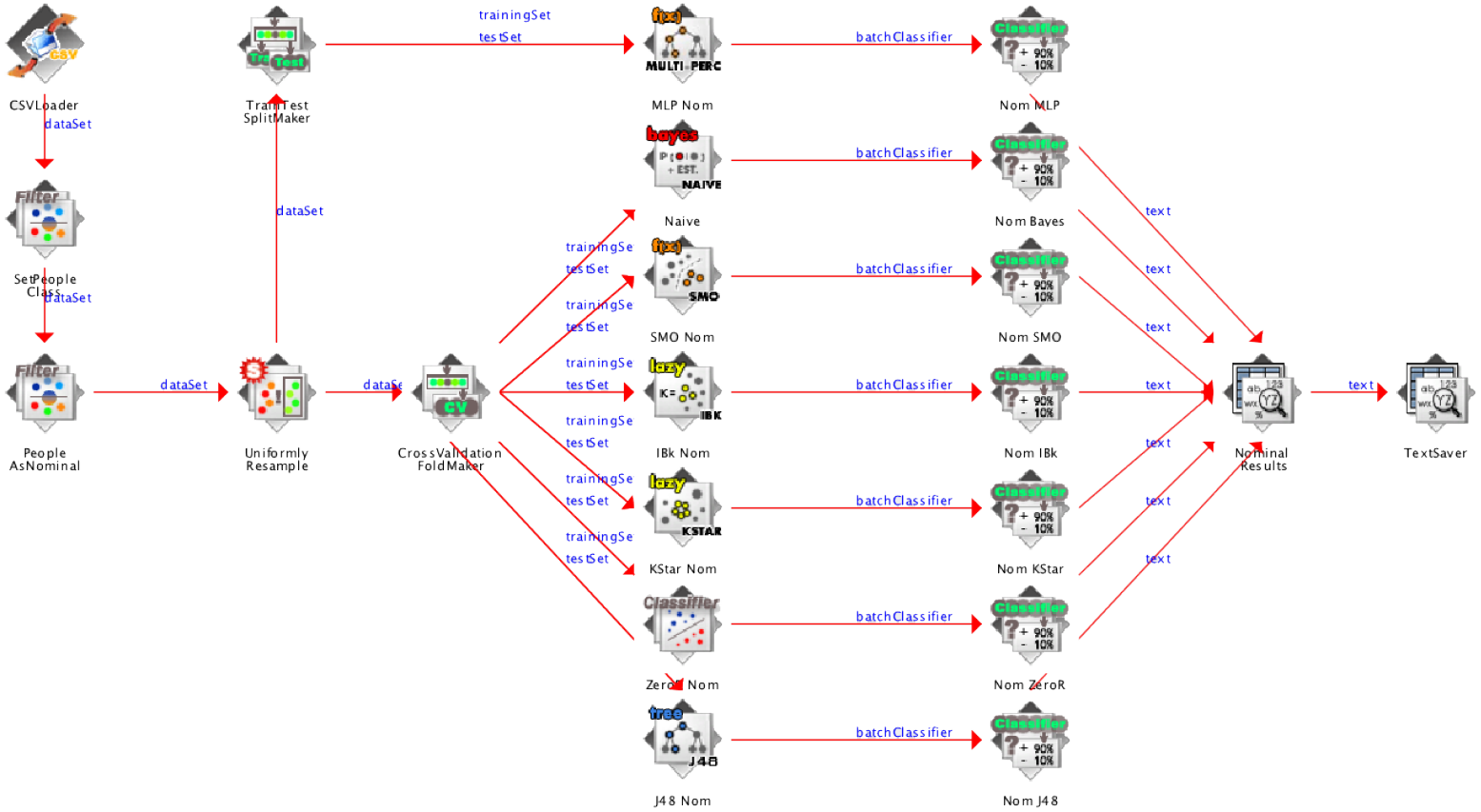


Figure B.2: Nominal knowledge flow

In Weka, Knowledge Flows can be defined, which provide an easy way to replicate a series of Weka functions. We provide a unified knowledge flow at the `run_flow.py` script to execute it on a given data set. However, we also replicate the numeric and nominal flows here (separated due to size) for those interested.

APPENDIX C

Original Honours Proposal

Title:	Developing a robust system for occupancy detection in the household
Author:	Ash Tyndall
Supervisor:	Professor Rachel Cardell-Oliver
Degree:	BCompSci (24 point project)
Date:	October 8, 2014

C.1 Background

The proportion of elderly and mobility-impaired people is predicted to grow dramatically over the next century, leaving a large proportion of the population unable to care for themselves, and consequently less people able care for these groups. [5] With this issue looming, investments are being made into a variety of technologies that can provide the support these groups need to live independent of human assistance.

With recent advancements in low cost embedded computing, such as the Arduino [1] and Raspberry Pi, [14] the ability to provide a set of interconnected sensors, actuators and interfaces to enable a low-cost 'smart home for the disabled' is becoming increasingly achievable.

Sensing techniques to determine occupancy, the detection of the presence and number of people in an area, are of particular use to the elderly and disabled. Detection can be used to inform various devices that change state depending on the user's location, including the better regulation energy hungry devices to help reduce financial burden. Household climate control, which in some regions of Australia accounts for up to 40% of energy usage [2] is one particular area

in which occupancy detection can reduce costs, as efficiency can be increased dramatically with annual energy savings of up to 25% found in some cases. [7]

Significant research has been performed into the occupancy field, with a focus on improving the energy efficiency of both office buildings and households. This is achieved through a variety of sensing means, including thermal arrays, [4] ultrasonic sensors, [10] smart phone tracking, [11][3] electricity consumption, [12] network traffic analysis, [15] sound, [9] CO2, [9] passive infrared, [9] video cameras, [6] and various fusions of the above. [16][15]

C.2 Aim

While many of the above solutions achieve excellent accuracies, in many cases they suffer from problems of installation logistics, difficult assembly, assumptions on user's technology ownership and component cost. In a smart home for the disabled, accuracy is important, but accessibility is paramount.

The goal of this research project is to devise an occupancy detection system that forms part of a larger 'smart home for the disabled' that meets the following accessibility criteria;

- *Low Cost*: The set of components required should aim to minimise cost, as these devices are intended to be deployed in situations where the serviced user may be financially restricted.
- *Non-Invasive*: The sensors used in the system should gather as little information as necessary to achieve the detection goal; there are privacy concerns with the use of high-definition sensors.
- *Energy Efficient*: The system may be placed in a location where there is no access to mains power (i.e. roof), and the retrofitting of appropriate power can be difficult; the ability to survive for long periods on only battery power is advantageous.
- *Reliable*: The system should be able to operate without user intervention or frequent maintenance, and should be able to perform its occupancy detection goal with a high degree of accuracy.

Success in this project would involve both

1. Devising a bill of materials that can be purchased off-the-shelf, assembled without difficulty, on which a software platform can be installed that performs analysis of the sensor data and provides a simple answer to the occupancy question, and

2. Using those materials and softwares to create a final demonstration prototype whose success can be tested in controlled and real-world conditions.

This system would be extensible, based on open standards such as REST or CoAP, [8][13] and could easily fit into a larger 'smart home for the disabled' or internet-of-things system.

C.3 Method

Achieving these aims involves performing research and development in several discrete phases.

C.3.1 Hardware

A list of possible sensor candidates will be developed, and these candidates will be ranked according to their adherence to the four accessibility criteria outlined above. Primarily the sensor ranking will consider the cost, invasiveness and reliability of detection, as the sensors themselves do not form a large part of the power requirement.

Similarly, a list of possible embedded boards to act as the sensor's host and data analysis platform will be created. Primarily, they will be ranked on cost, energy efficiency and reliability of programming/system stability.

Low-powered wireless protocols will also be investigated, to determine which is most suitable for the device; providing enough range at low power consumption to allow easy and reliable communication with the hardware.

Once promising candidates have been identified, components will be purchased and analysed to determine how well they can integrate.

C.3.2 Classification

Depending on the final sensor choice, relevant experiments will be performed to determine the classification algorithm with the best occupancy determina-

tion accuracy. This will involve the deployment of a prototype to perform data gathering, as well as another device/person to assess ground truth.

C.3.3 Robustness / API

Once the classification algorithm and hardware are finalised, an easy to use API will be developed to allow the data the device collects to be integrated into a broader system.

The finalised product will be architected into a easy-to-install software solution that will allow someone without domain knowledge to use the software and corresponding hardware in their own environment.

C.4 Timeline

Date	Task
Fri 15 August	*Project proposal and project summary due to Coordinator*
August	Hardware shortlisting / testing
25–29 August	*Project proposal talk presented to research group*
September	Literature review
Fri 19 September	*Draft literature review due to supervisor(s)*
October - November	Core Hardware / Software development
Fri 24 October	*Literature Review and Revised Project Proposal due to Coordinator*
November - February	*End of year break*
February	Write dissertation
Thu 16 April	*Draft dissertation due to supervisor*
April - May	Improve robustness and API
Thu 30 April	*Draft dissertation available for collection from supervisor*
Fri 8 May	*Seminar title and abstract due to Coordinator*
Mon 25 May	*Final dissertation due to Coordinator*
25–29 May	*Seminar Presented to Seminar Marking Panel*
Thu 28 May	*Poster Due*
Mon 22 June	*Corrected Dissertation Due to Coordinator*

C.5 Software and Hardware Requirements

A large part of this research project is determining the specific hardware and software that best fit the accessibility criteria. Because of this, an exhaustive list of software and hardware requirements are not given in this proposal.

A budget of up to $300 has been allocated by my supervisor for project purchases. Some technologies with promise that will be investigated include;

Raspberry Pi Model B+ Small form-factor Linux computer
Available from `http://arduino.cc/en/Guide/Introduction`; $38

Arduino Uno Small form-factor microcontroller
Available from `http://arduino.cc/en/Main/arduinoBoardUno`; $36

Panasonic Grid-EYE Infrared Array Sensor
Available from `http://www3.panasonic.biz/ac/e/control/sensor/infrared/grid-eye/index.jsp`; approx. $33

Passive Infrared Sensor
Available from various places; $10–$20

C.6 Proposal References

[1] Ardunio. http://arduino.cc/en/Guide/Introduction. Accessed: 2014-08-09.

[2] Australian Bureau of Statistics. Household water and energy use, Victoria: Heating and cooling. Tech. Rep. 4602.2, October 2011. Retrieved October 6, 2014 from http://www.abs.gov.au/ausstats/abs@.nsf/0/85424ADCCF6E5AE9CA257A670013AF89.

[3] Balaji, B., Xu, J., Nwokafor, A., Gupta, R., and Agarwal, Y. Sentinel: occupancy based HVAC actuation using existing WiFi infrastructure within commercial buildings. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems* (2013), ACM, p. 17.

[4] Beltran, A., Erickson, V. L., and Cerpa, A. E. ThermoSense: Occupancy thermal based sensing for HVAC control. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings* (2013), ACM, pp. 1–8.

[5] Chan, M., Campo, E., Estève, D., and Fourniols, J.-Y. Smart homes - current features and future perspectives. *Maturitas 64*, 2 (2009), 90–97.

[6] Erickson, V. L., Achleitner, S., and Cerpa, A. E. POEM: Power-efficient occupancy-based energy management system. In *Proceedings of the 12th international conference on Information processing in sensor networks* (2013), ACM, pp. 203–216.

[7] Erickson, V. L., Beltran, A., Winkler, D. A., Esfahani, N. P., Lusby, J. R., and Cerpa, A. E. Demo abstract: ThermoSense: thermal array sensor networks in building management. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems* (2013), ACM, p. 87.

[8] Guinard, D., Ion, I., and Mayer, S. In search of an internet of things service architecture: REST or WS-*? a developers perspective. In *Mobile and Ubiquitous Systems: Computing, Networking, and Services*. Springer, 2012, pp. 326–337.

[9] Hailemariam, E., Goldstein, R., Attar, R., and Khan, A. Real-time occupancy detection using decision trees with multiple sensor types. In *Proceedings of the 2011 Symposium on Simulation for Architecture and Urban Design* (2011), Society for Computer Simulation International, pp. 141–148.

[10] Hnat, T. W., Griffiths, E., Dawson, R., and Whitehouse, K. Doorjamb: unobtrusive room-level tracking of people in homes using doorway sensors. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems* (2012), ACM, pp. 309–322.

[11] Kleiminger, W., Beckel, C., Dey, A., and Santini, S. Poster abstract: Using unlabeled Wi-Fi scan data to discover occupancy patterns of private households. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems* (2013), ACM, p. 47.

[12] Kleiminger, W., Beckel, C., Staake, T., and Santini, S. Occupancy detection from electricity consumption data. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings* (2013), ACM, pp. 1–8.

[13] Kovatsch, M. CoAP for the web of things: from tiny resource-constrained devices to the web browser. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication* (2013), ACM, pp. 1495–1504.

[14] Raspberry pi. `http://www.raspberrypi.org/`. Accessed: 2014-08-09.

[15] Ting, K., Yu, R., and Srivastava, M. Poster Abstract: Occupancy inferencing from non-intrusive data sources. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings* (2013), ACM, pp. 1–2.

[16] Yang, Z., Li, N., Becerik-Gerber, B., and Orosz, M. A multi-sensor based occupancy estimation model for supporting demand driven HVAC operations. In *Proceedings of the 2012 Symposium on Simulation for Architecture and Urban Design* (2012), Society for Computer Simulation International, p. 2.