Practical Systems for Personal Thermal Comfort

by

Ali Mohammad Rabbani Esfahani

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2016

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Conventional centralized HVAC systems cannot provide office workers with personalized thermal comfort because workers in a single zone share a common air handling unit and thus a single air temperature. Moreover, they heat or cool an entire zone even if a single worker is present, which can waste energy. Both drawbacks are addressed by Personal Environmental Control (PEC) systems that modify the thermal envelope around a worker's body to provide personalized comfort. However, most PEC systems are both expensive and difficult to deploy, making them unsuitable for large-scale deployment. In contrast, we present two novel PEC systems: SPOTlight and its successor OpenTherm. These systems are carefully designed for practical, rapid, and scalable deployment. Intuitive web-based interfaces for user controls allow OpenTherm to be installed in only about 15 minutes, including user training. It is also low-cost (as low as US$80, in volume) because it uses the fewest possible sensors and a lightweight compute engine that can optionally be located in the cloud. In this thesis, we present the detailed design of SPOTlight and OpenTherm systems, and results from a cumulative 81 months of OpenTherm's operation in 15 offices. In our objective evaluation, we find that OpenTherm improved user comfort by ~67%. This is in comparison to when only the central HVAC system with no knowledge of occupancy and inability to control offices individually is used.

Acknowledgements

I would first like to extend my sincere gratitude to my adviser Prof. Keshav of the David Cheriton School of Computer Science at the University of Waterloo. Prof. Keshav kept the door to his office always open, and I had the luxury of discussing my problems with him in detail whenever I ran into one. He simplified hard problems, and gave me hope in harder times. He allowed this thesis to be the result of my own work, while steering me in the right direction when I needed his guidance.

Costin Ograda-Bratu and Milad Khaki were closely involved in my project. With their help, I was able to build a system prototype, and deploy OpenTherm devices on campus. CSCF and IST staff worked with me to find a solution for my project's network requirements, and implement it on the campus-wide wireless network. The 15 participants in the OpenTherm deployment dedicated their time, and enabled us to test, troubleshoot, and evaluate our system. I would like to thank them all.

I would also like to acknowledge Prof. Rosenberg of the Department of Electrical and Computer Engineering, and Prof. Wong of the School of Computer Science at the University of Waterloo as the readers of this thesis, and I am gratefully indebted to them for their valuable comments on this thesis.

Finally, I must express my very profound gratitude to my parents for providing me with unconditional support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Dedication

To my mom and dad, who have supported me unconditionally throughout my life. I would have never been able to accomplish much without them.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction and Background

1.1 Introduction

Heating, Ventilation and Air-Conditioning (HVAC) in residential and commercial buildings consumes significant amounts of energy in most developed countries, accounting for 15%-20% of overall energy use in these countries [3, 4]. Reducing the energy consumption of HVAC systems is, therefore, an important and necessary step in reducing their carbon footprint [5, 6].

Conventional centralized HVAC systems[1] have long struggled to provide thermal comfort to individual office workers[2]. This is for two intrinsic reasons. First, workers typically cannot express their individual comfort preferences to the HVAC system. Instead, HVAC parameters such as temperature setpoints are chosen by building managers to provide comfort to the 'average' worker. Second, a single Air Handling Unit (AHU) is usually shared by multiple workers who may feel comfortable at different temperatures. Therefore, no matter what temperature the AHU is set to, one or more workers may be uncomfortable[3] [7]. Moreover, they heat or cool an entire zone even if a single worker is present, which can be wasteful.

These drawbacks motivate the design of Personal Environmental Control (PEC) systems [2, 7] that provide workers in buildings with conventional centralized HVAC, with a

[1] Throughout this thesis, "conventional HVAC systems" refers to systems that do not have occupancy information, and cannot control the temperature of each room individually.

[2] We use the terms "worker" and "user" interchangeably in this thesis.

[3] Per-office re-heaters or Variable Refrigerant Flow systems do allow individualized control, but are costly, so they are rare in North America.

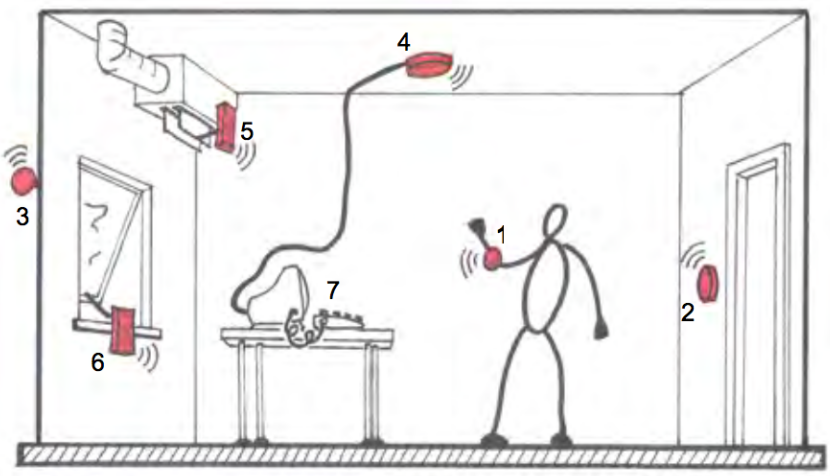

Figure 1.1: The PEC system presented in [1] controls windows and vents, has many sensors, and requires users to wear a special device on their wrists.

personalized system for thermal comfort that can reduce energy consumption as well. Several PEC systems have been studied over the past two decades [1, 2, 8, 9, 10, 11]. However, each of these systems has some drawbacks. Some systems are onerous to use, for example requiring workers to wear devices on each wrist [1] (Figure 1.1) or special clothing [11]; some are intrusive, using video cameras to watch workers [9, 10]; some can only be used for heating, not cooling [9]; others modify a worker's desk to add vents [2, 8], which is disruptive. All these systems are quite expensive, as well. Thus, none of them are suitable for immediate, large-scale, practical deployment.

In this thesis, we present SPOTlight and OpenTherm, individual thermal comfort systems that can be rapidly and cost-effectively deployed. To achieve this primary goal, our sub-goals are:

- **Low per-unit cost**. We estimate each OpenTherm unit costs about $80 in volume production.
- **Plug-and-play deployment**. OpenTherm reduces the cost and time to deploy significantly. Setting up a single OpenTherm unit takes about 5 minutes.
- **Legacy compatibility**. There is no need to rebuild or modify existing central HVAC systems to use OpenTherm.

- **Ease of use**. Users do not have to interact with OpenTherm to benefit from it. They also don't need to wear any devices or special clothing.
- **Ease of user training**. It takes less than 10 minutes to train and prepare users for using OpenTherm.

We designed and implemented SPOTlight and OpenTherm to achieve the goals above. SPOTlight was our first iteration on building the system, which uses a Zolertia Z1 wireless mote as a compute/communication module. Our second iteration, OpenTherm, is built around a Raspberry Pi with a lower cost and more capabilities than the Z1 mote. Both the hardware and the software are Free and Open Source[4]. OpenTherm is an improved, more complete version of the system. Therefore, our primary focus in this thesis is on OpenTherm.

After building several prototypes, we designed a better-looking, more sophisticated version of OpenTherm for end-users and built 65 of them. At the time of writing this thesis, we have 15 of them deployed on campus that are operating reliably, and we have collected data for a total of about 58,000 hours.

The rest of this thesis is laid out as follows. Section 1.2 presents a background on quantitative comfort modelling and an overview of prior work. In Chapter 2, we show the architectures of our two versions of the system. Implementation details of our first iteration, SPOTlight, are presented in Chapter 3. Chapter 4 presents the details of hardware and software components in OpenTherm, and how it is different from SPOTlight. Chapter 5 evaluates OpenTherm's performance. Finally, in Chapter 6, we conclude the thesis.

1.2 Background

1.2.1 Personal Environmental Control Systems

Personal Environmental Control (PEC) systems seek to simultaneously meet two goals. First, it is impossible for conventional centrally controlled HVAC systems to have 100% user satisfaction in buildings where multiple people must share the same temperature set-point [7]. A PEC system modifies the thermal envelope around users to increase comfort. Second, buildings' HVAC energy consumption accounts for a significant portion of CO_2 emissions in the world [7, 12, 13], with about half of the emissions coming from commercial

[4]https://blizzard.cs.uwaterloo.ca/spotstar/about/

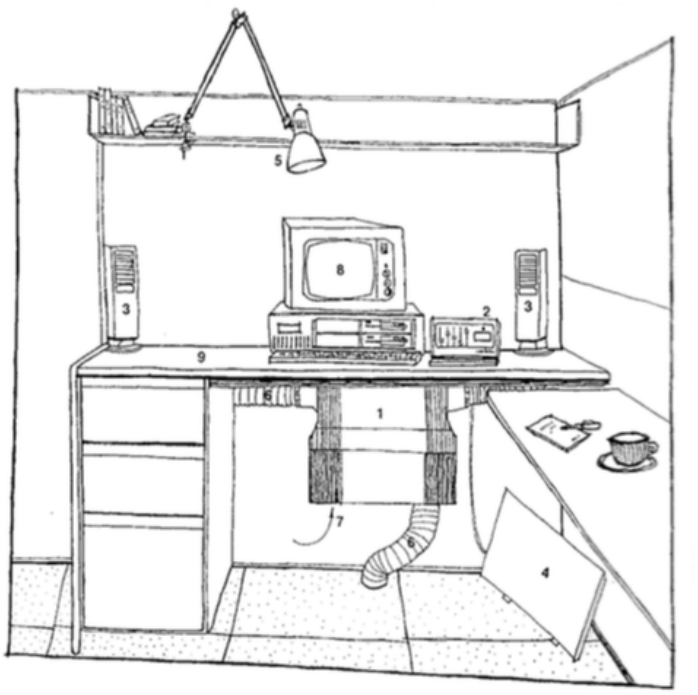

Figure 1.2: Bauman et al. modified user workstations with fans and heating panels, and let them control their systems individually [2].

buildings [3, 14]. With climate change being an area of intense focus, researchers are exploring alternatives to reduce HVAC energy consumption, while maintaining user comfort. For example, Erickson et al. focus on occupancy-based control [5, 6] of HVAC systems, which allows buildings to **violate** comfort regimes when unoccupied. A PEC system meets the same goal by allowing common areas in the building to be heated less in winter and cooled less in summer, while keeping users comfortable in their workspaces.

The seminal work on PEC system design is by Bauman et al. [2] who perform a field study and provide a group of users with a manually controlled desktop task/ambient conditioning (TAC) system. They show significant comfort improvements when users are equipped with the TAC. However, deploying TAC requires extensive modifications to user workspaces, shown in Figure 1.2, including drilling holes into their desk, which is both expensive and disruptive.

Zhang et al. build a system that locally heats or cools crucial parts of body (i.e. hands, feet, face), to keep users comfortable [8]. While their system reduces energy consumption, and has a fine control over how users feel, it requires extensive user engagement. Moreover,

although the paper does not present the cost of the system, it appears that it can be quite expensive, due to the use of individual heaters for hands and feet, and two separate face-level pedestal fans.

Recently (December 2014), the US ARPA-E has funded several research teams to develop Local Thermal Management Systems as part of the DELTA program [15, 11]. The primary goal is to allow buildings to be operated across a wider range of setpoints, reducing overall energy consumption. However, the projects have yet to reveal the details of their operation. We observe that most of the efforts appear to be directed towards the development of smart clothing that can perform heating or cooling functions. Forcing workers to wear special clothing appears to be both onerous and impractical.

The work closest to ours is SPOT and SPOT+ [9, 10]. SPOT reactively controls a space heater to control room temperature in winter. In contrast, SPOT+ predicts occupancy to allow pre-heating. SPOT+ also uses an optimal control strategy to minimize energy consumption. Although interesting as prototypes, the use of multiple fine-grained sensors in SPOT and SPOT+, including a Microsoft Kinect, make them expensive ($1,000 per office). Neither system provides cooling and both are intrusive, because they need to process video images in real time.

1.2.2 Comfort Models

The best known model for human comfort in buildings is the Predicted Mean Vote (PMV) model [16]. The PMV model estimates the average worker's comfort level on the 7-point ASHRAE scale (see Table A.1) using a function $f_{pmv}(\cdot)$:

$$pmv = f_{pmv}(t_a, \bar{t}_r, v_{ar}, p_a, M, I_{cl}) \tag{1.1}$$

where pmv is the predicted mean vote and :

- t_a is the air temperature
- $\bar{t_r}$ is the mean background radiant temperature
- v_{ar} is the air velocity
- p_a is the humidity level
- M is the metabolic rate of a worker
- I_{cl} is the worker's clothing insulation factor

Gao et al. [9] have generalized this to the Predicted Personal Vote (PPV), which is the model we use in OpenTherm. Appendix A explains *pmv* in detail, but we defer the computation of PPV from PMV to [9].

A more recent alternative to the PMV model is the adaptive model for thermal comfort [17] that is commonly used in HVAC research for naturally ventilated buildings, and centrally air-conditioned buildings where occupants have adaptive controls (such as operable windows) [7]. OpenTherm is targeted at legacy environments that lack such controls. Therefore, we use PPV noting that, in the absence of training by users, OpenTherm defaults to using PMV instead of PPV.

1.2.3 Comfort and Air Movement

Many studies show reduced energy consumption and improved comfort in the presence of air movement [18, 19, 20, 21]. In [2], Bauman et al. show significant improvements in comfort by providing users with a controllable desktop fan. Drawing upon these results, OpenTherm uses the cooling effect of air movement in warm temperatures to improve user comfort. It also uses hot air movement in colder temperatures to immediately react to changes in comfort or occupancy.

Chapter 2

Design

We now discuss the design of the SPOTlight and OpenTherm systems. Both systems serve similar goals discussed in Section 1.1, but have different approaches towards solving the problem. SPOTlight tackles the problem by placing a wireless device with minimal capabilities on the user side, and moving most of the computation to the cloud. OpenTherm on the other hand, uses a more capable Raspberry Pi, and provides more flexibility and reliability.

2.1 SPOTlight Architecture

SPOTlight's overall system architecture is shown in Figure 2.1. For SPOTlight, we use a wireless mote whose computing functionality is moved to a cloud-based Virtual Execution Environment (VEE) [22]. These motes form a wireless mesh, and connect to the cloud using a gateway. In addition to low cost, moving processing to the cloud makes the system easier to deploy.

Thus, SPOTlight's system architecture can be decomposed into three parts: a client side, a gateway, and a VEE-based server side. We put the least possible amount of sensing and logic on the client side to reduce costs. The gateway connects clients to VEEs over secure communication channels, and VEEs have all the logic and make decisions to control clients. We explain the details of each part in Chapter 3. A more detailed version of this architecture is shown in Figure 3.1.

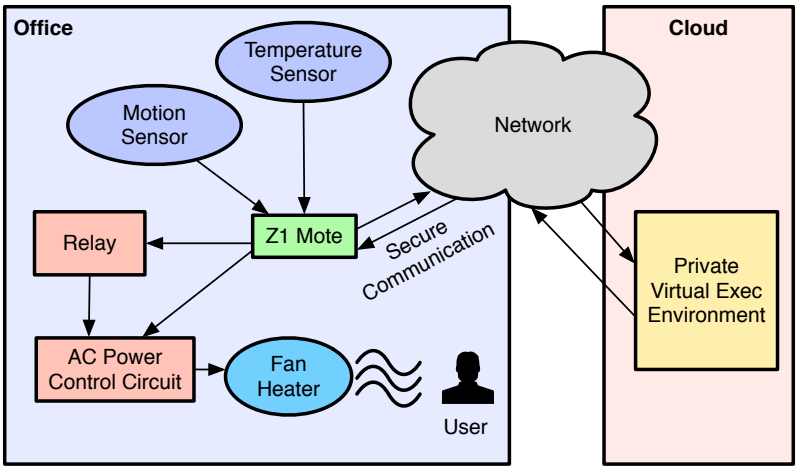

Figure 2.1: Architecture of SPOTlight. The embedded Z1 platform collects temperature and motion data and sends them to a private Virtual Execution Environment over the network. The VEE makes a decision to control the heater and the speed of the fan connected to its corresponding Z1 mote and sends a command to it. Finally, the Z1 mote executes that command and sets the fan speed using a solid state relay and a custom built circuit.

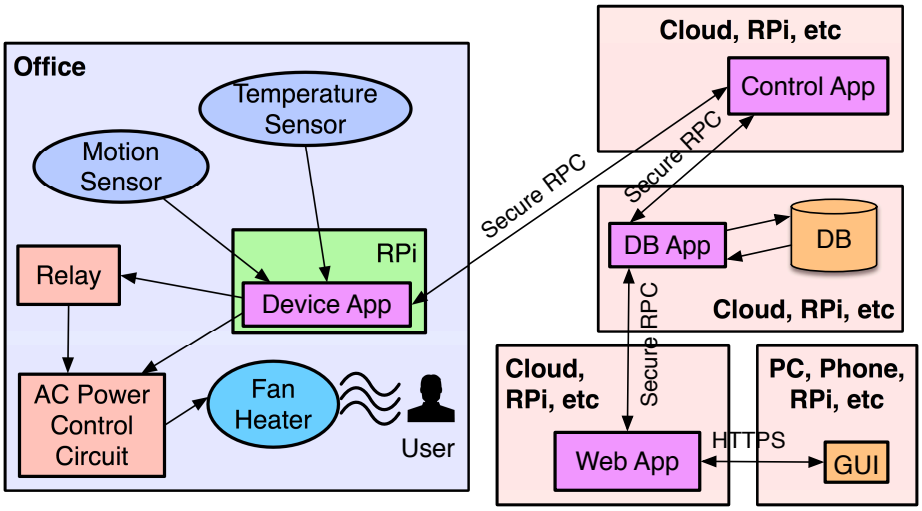

Figure 2.2: Architecture of OpenTherm, which has 5 main components. Actuation and sensing (left hand side of the figure), control application, data storage, web application, and graphical user interface. All software components communicate through RPC to allow the system to easily assume different configurations. OpenTherm is more reliable and flexible.

2.2 OpenTherm Architecture

OpenTherm includes both software and hardware components, and is fairly complex. We therefore first present an outline of the system architecture (Figure 2.2), then make a second pass over the design, presenting implementation details in Chapter 4.

We start with a description of the hardware placed in each office (the left hand side of Figure 2.2). This part of the system consists of a heating/cooling device (essentially a fan with a resistive heating coil), temperature and occupancy sensors, and solenoid actuators to turn the fan and heater on or off. The software entity that interfaces with the sensors and actuators is the *device manager application* (labeled 'device app' in the figure) executing on a Raspberry Pi (RPi). The device manager application reads sensor values from sensors and writes commands to the actuators. It sends the read sensor values to a *control app* and receives commands from it, that it carries out. The control app receives sensor measurements from the device app and uses this to decide to turn the fan/heater on or off and to select the fan's speed.

The past history of occupancy, temperature measurements, actuation events, and device training data are stored in a database (DB) by a *DB app*. This is useful for system administration and debugging, making control decisions based on user preferences, and providing feedback to the user in the form of historical charts (see Figure 2.3). The DB can reside on the RPi for privacy, where it physically does not leave the office, or on the cloud to provide better reliability and availability.

User interface with OpenTherm is by means of a *Web app* that provides status information to OpenTherm users and also allows them to communicate their comfort preferences to the control app. Finally, the *GUI* is the only software component of the system visible to users. In a networked setup, the GUI can be invoked on any device with an internet browser (e.g. PC, smart phone, RPi). For standalone installations, we have added a 7-inch touch screen to the RPi and installed it on the OpenTherm box. The user can access the GUI directly from the box (see Figure 4.4).

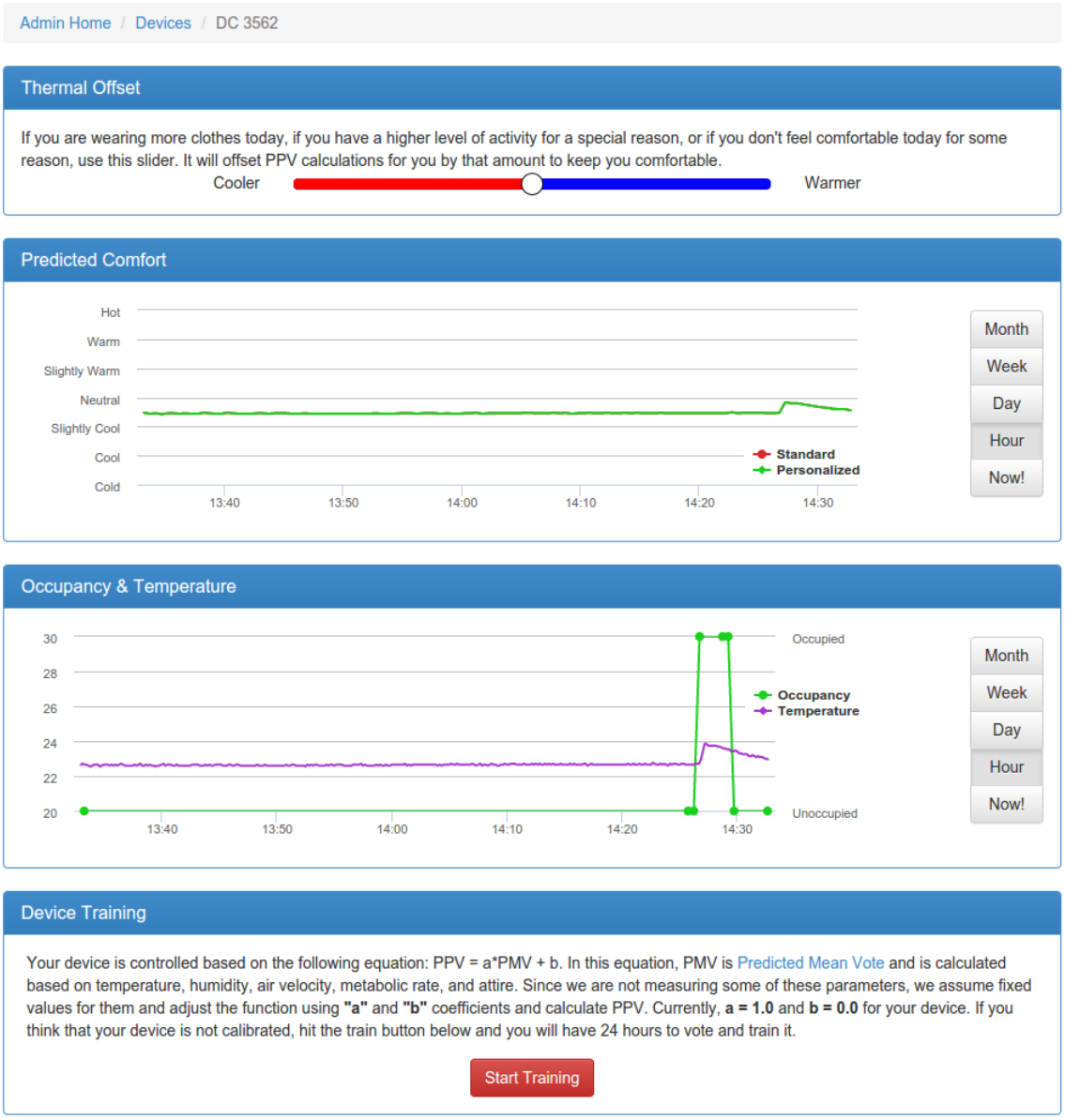

Figure 2.3: The graphical user interface, viewable with standard internet browsers. The top section allows users to manually fine-tune the system for temporary changes. The graphs in the middle visualize comfort (PMV and PPV), occupancy, and temperature over time. Users use the bottom section to start training their devices. During the training, users vote periodically on how they feel based on the 7-point ASHRAE scale (See Appendix A).

Chapter 3

Implementation 1: SPOTlight

SPOTlight was our first attempt in building a system to meet the goals described in Section 1.1. We built SPOTlight based on the Zolertia Z1 wireless mote. Later on, we found out that we could achieve our goals more easily, and with a more robust architecture, using a Raspberry Pi. In this chapter, we discuss building blocks of SPOTlight, which eventually led us to create OpenTherm, which is discussed in Chapter 4.

The **Zolertia Z1** mote is an integral part of SPOTlight. These are embedded devices that have an MSP430 processor and a ZigBee wireless module. The mote has a number of input and output pins connected to the processor that we use for data collection and actuation. Phidgets ports on the mote allow us to connect Phidgets sensors to it. We run the Contiki operating system which can run multiple processes to control the mote.

We now discuss the details of each component of SPOTlight: **client side**, **gateway**, and **server side**.

3.1 Client Side

The client-side consists of four major components: a compute platform, a desktop fan, sensors, and an actuator. Little computation needs to be performed at the client. Therefore, we use an inexpensive Zolertia Z1 mote, which also controls the actuator, as the compute platform. The motes form a wireless mesh with other motes to communicate with the gateway. We describe each component now.

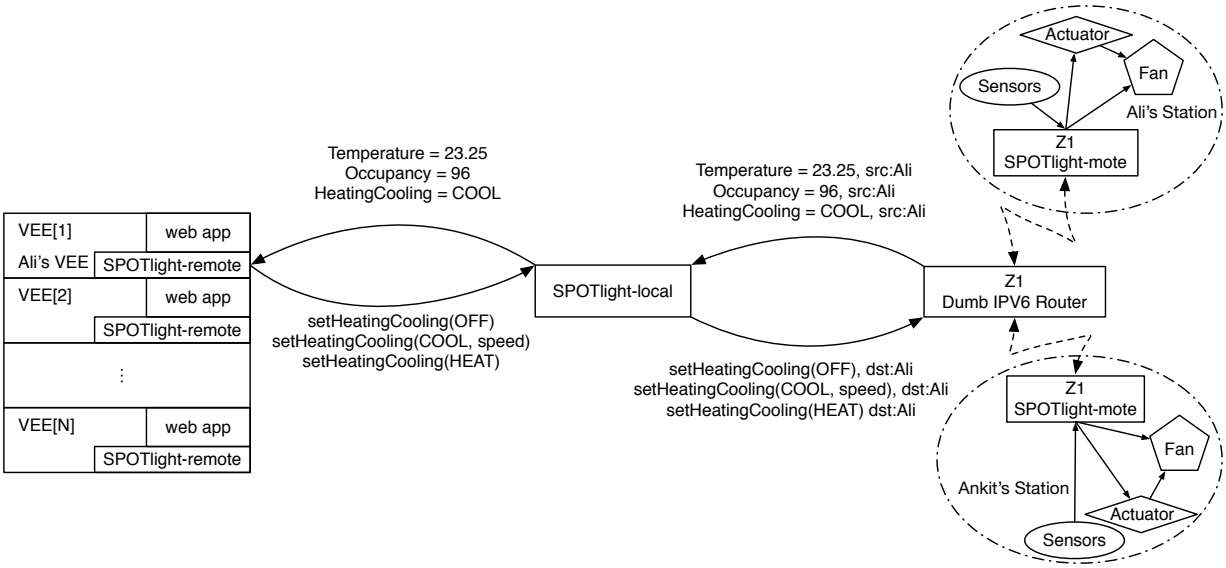

Figure 3.1: On the client side, There is a Z1 mote, which has an application called SPOTlight-mote running on it. SPOTlight-mote collects data from sensors and sends them over to the server via a middle box. When SPOTlight-mote receives a command from its server, it executes that command using an actuator. The communication between the motes and the middle box is wireless, and represented by dashed lines in the figure. On the server side, SPOTlight-remote receives information from its corresponding Z1 mote, makes a decision and issues a command to the mote if necessary. SPOTlight-remote has a TCP connection with SPOTlight-local which is a forwarding application running on a PC in the building. SPOTlight-local acts as a router between the local mesh network of Z1 motes and remote VEEs.

3.1.1 Compute platform

We use Z1 motes running the Contiki operating system as our compute platform. It has two main tasks: It collects data from sensors and transmits them over a wireless mesh to a gateway, and from there to a remote VEE. It also receives commands from the VEE and executes them by controlling an actuator.

A mote runs four processes, one master process and three I/O processes, each of which is responsible for a specific event: read from temperature sensor, read from the motion sensor, process packet arrival. Figure 3.2 shows a Z1 mote and how its pins are used to collect sensor data and execute commands. We now discuss the operation of the mote in greater detail.

1. **Sensor data collection.** To collect data from sensors, the Contiki operating system provides a library called *z1-phidgets* which can read raw data from Phidgets ports. We use this library to collect motion data from a sensor connected to a 5V Phidgets port. We also collect air temperature data using the on-chip temperature sensor (see Section 3.1.2).

2. **Communication.** Z1 motes form a wireless mesh using the ZigBee protocol. Motes use this network to send sensor data to VEEs and receive commands from them. A Z1 mote that acts as a border router connects a PC to this mesh (see Section 3.2).

3. **Command execution.** Z1 motes receive commands from a server-side VEE. The mote translates commands into actions by changing the values of the MSP430 microprocessor output pins that are connected to the actuator. Specifically, we use two control modes. In the first mode, we use Contiki's *relay-phidget* library to toggle a 3V DC output pin on or off. By connecting this output to a solid state relay, we turn the fan on and off. This mode, however, is not sufficient to control fan speed, where we need finer-grained control. In the second control mode, we select one of 100 fan speeds using the MSP430 processor's 12-bit digital to analog converter (DAC12_0) to create a controlled analog output control voltage supplied to the fan speed control circuit described in Section 3.1.3.

3.1.2 Sensors

We use only two sensors in SPOTlight, a temperature sensor and a motion sensor. Data from these sensors are used to compute PPV.

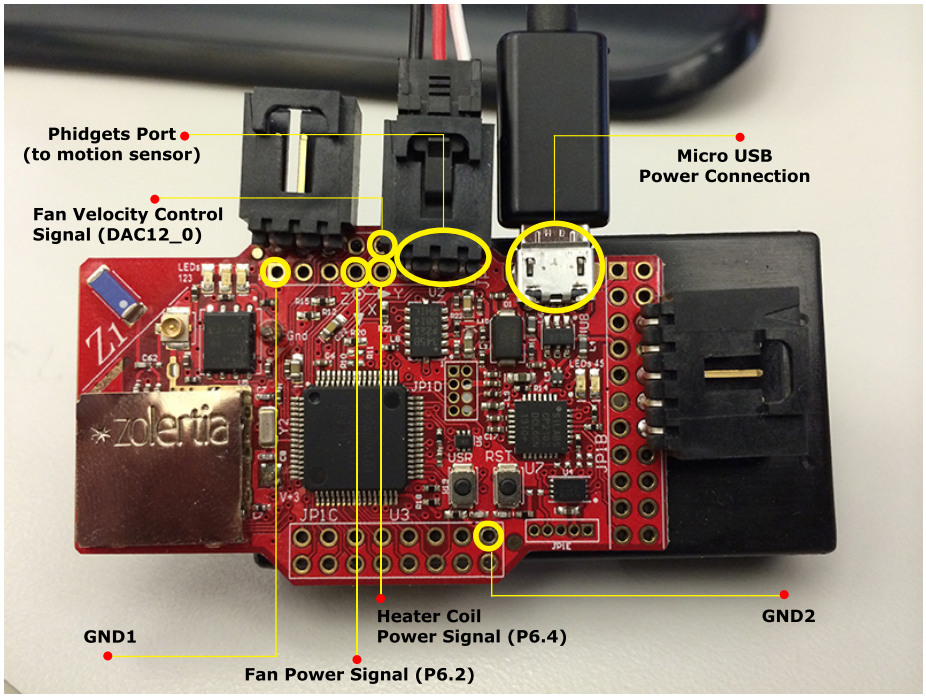

Figure 3.2: Zolertia Z1. We use different pins on the mote to read sensor values, and control the fan.

To obtain **temperature** readings, we use Z1's on-board TMP102 temperature sensor with 0.06 °C resolution. We use Contiki's *tmp102* library to read values from this sensor every 10 seconds and send them to the VEE.

We reliably detect **occupancy** using an approach presented in [23]. In this approach, the standard deviation of motion sensor data over a 2-minute window, determines occupancy for that timeframe. To reduce the amount of network communications, the mote collects motion data twice every second, computes its standard deviation, and sends only this value to the VEE every 2 minutes. To collect motion data, we use Phidgets 1111 motion sensor. This board has a Panasonic AMN23111 passive infrared human detection sensor, and is connected to the Z1 mote using a Phidgets port. As OpenTherm uses a similar mechanism to detect occupancy, details of reading motion values are discussed in Section 4.1.2.

3.1.3 Actuator

Since we are using an AC powered fan and the Z1 mote is a DC device, we cannot directly control the fan with it. The 0V to 3V DC output pins of the mote are the only outputs we can use to turn the fan on or off and control its speed. Therefore, we use solid state relays to close and open the fan's AC circuit, and a custom circuit to control its speed.

Solid State Relays Solid state relays are devices that can open or close a circuit based on an input signal. We choose a relay with 3V DC input and 120V AC output. Therefore, we can connect a DC output pin of Z1 (P6.2) to the relay and turn the fan on or off by setting the pin value to 1 or 0.

AC Power Control Circuit In addition to turning the fan on and off, we must control its speed to keep PPV around 0. Therefore, we built a circuit capable of controlling the fan speed with a DC analog input (See Section 4.1.3 for details). Z1's DAC12_0 is connected to this circuit and controls the fan speed by changing the voltage. Figure 3.3 includes the prototype circuit built for SPOTlight on a breadboard.

3.1.4 Heating and Cooling Device

To provide a worker with both heating and cooling, we modified the Royal Sovereign HFN-20 [24] personal fan/heater to control its heating coil and cooling fan independently. Two relays turn the fan and the heating coil on or off, and the AC power control circuit controls the speed of the fan, up to a maximum air velocity of $2.1ms^{-1}$.

3.2 Gateway

The gateway terminates the wireless mesh communications and provides connectivity between the Internet and the wireless nodes. It connects motes with IPv6 addresses to VEEs using IPv4 addresses in the cloud. The gateway has two components: a border router, and a local PC.

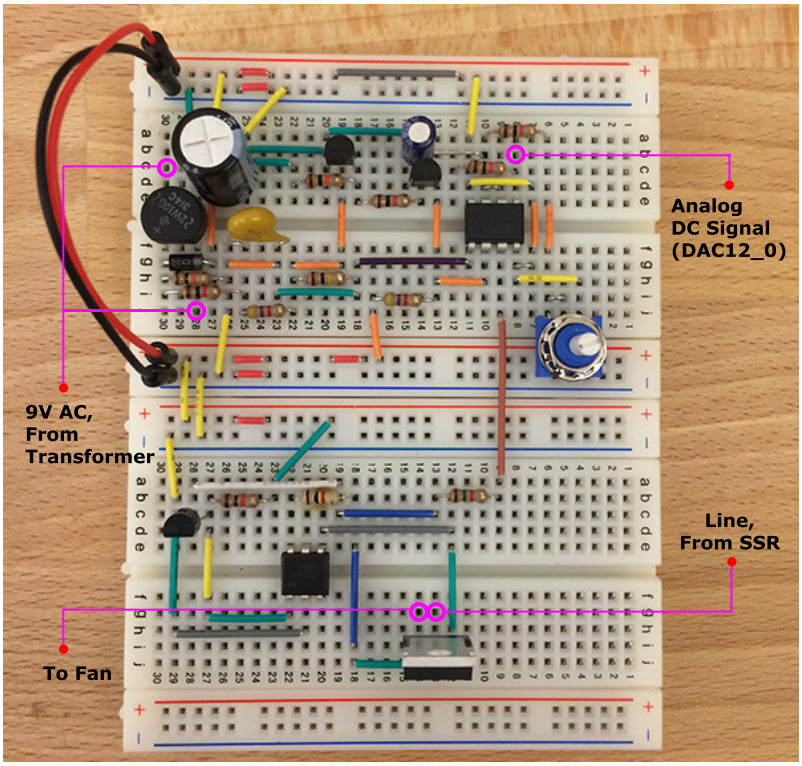

Figure 3.3: AC power control circuit on a breadboard. This prototype was built for SPOTlight to control the speed of the fan.

Mote Address	VEE Address
aaaa::c30c::0::0::9e	172.19.16.2
aaaa::c30c::0::0::48	172.19.16.3

Table 3.1: Sample mapping between VEEs and Z1 motes. The gateway uses this one-to-one mapping to forward packets.

3.2.1 Border Router

We choose one mote as the border router and connect it to a local PC with a USB port. We create a virtual TUN [25] interface on the PC that talks to the border router using the *SLIP* protocol [26]. The TUN interface takes an IPv6 address, becomes a part of the mesh network, and receives packets that are destined to it through the border router.

3.2.2 Local PC

SPOTlight-local runs on the local server and acts as a forwarder between VEEs and the motes. It receives occupancy and temperature data from motes, finds their corresponding VEE and forwards the data to it. It also receives commands from VEEs, finds their paired mote and forwards the commands to it. In *SPOTlight-local* we use a table to map motes with IPv6 addresses to VEEs with IPv4 addresses. The one-to-one mapping allows *SPOTlight-local* to be a simple forwarder that does not know about the content of packets. The end-to-end encryption protocol is therefore very straightforward to implement. Table 3.1 shows a sample mapping between VEEs and Z1 motes.

3.3 Server Side

Each user owns a private VEE that is paired with a corresponding Z1 mote. There are two major components in the VEE, as shown in Figure 3.1: *SPOTlight-remote* and a web application. *SPOTlight-remote* calculates PPVs, makes decisions and sends commands to the mote to control the state and speed of the fan. The web application collects votes from the user and provides them to *SPOTlight-remote*. We now discuss these components in more detail and talk about the steps we take to ensure privacy of user data.

3.3.1 Server Application

SPOTlight-remote listens for TCP connections from *SPOTlight-local*. With each connection, a temperature or occupancy update is sent. We update application variables with each update to reflect them in the next decision.

1. **Occupancy Inference** As discussed in Section 3.1.2, *SPOTlight-remote* receives a standard deviation value O that shows movement intensity in every 2-minute period. SPOTlight-remote infers occupancy using a leaky bucket method described in Section 4.3.1.

2. **Decision Making** *SPOTlight-remote* makes decisions in a reactive manner using the leaky bucket and temperature data. Every time *SPOTlight-remote* receives an occupancy update. it looks at the leaky bucket, calculates PPV, makes a decision, and if necessary, sends a command to the mote. Algorithm 1, which is also used in OpenTherm, shows how a decision is made.

3. **Air Velocity Calculation** After *SPOTlight-remote* determines that cooling is needed, it must calculate the air velocity V that the fan should provide. Since the maximum V our fan can provide is $2.1ms^{-1}$ we divide the $[0, 2.1]$ interval to 0.1 steps. If $PPV > T_P$ we increase V from 0 with 0.1 steps until PPV goes below T_P or $V = 2.1$. We convert this speed into an integer between 0 and 99 and send it along with the cooling command to the mote.

3.3.2 Web Application

We build a web application to collect votes used in PPV calculations from users. The application allows users to privately vote their comfort level at any given time. *SPOTlight-remote* later fetches the vote and matches it to the PMV value at that point in time. This creates $(PMV, Vote)$ pairs that are used in learning user preferences and calculating forthcoming PPV values. Figure 3.4 shows how users specify their comfort in SPOTlight's web application. OpenTherm further extends this web app and creates a more sophisticated environment for users to interact with the system.

3.3.3 End-to-End Encryption

To protect privacy of users we use an end-to-end encryption protocol. When programming the Z1 mote, we create a shared key between the mote and it's paired VEE on a secure

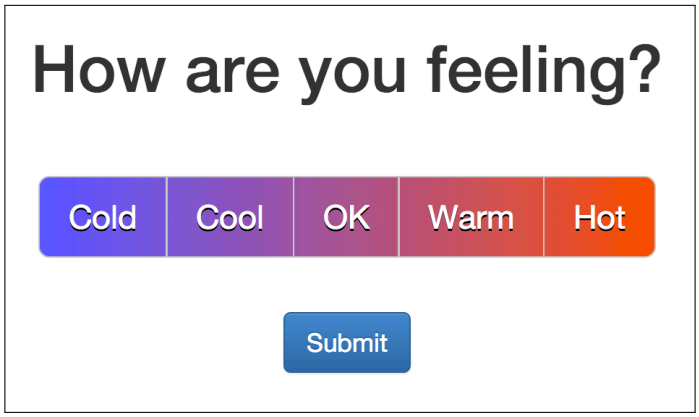

Figure 3.4: SPOTlight's web app. The app simply collects votes from users about their comfort to determine the coefficients for PPV calculation. In OpenTherm, the web app is more sophisticated and has added capabilities.

channel. With this shared key, communications are encrypted using AES between the VEE and the mote. Since we don't have many different commands and we only send them when the user was recently present, one can infer occupancy even from encrypted commands. To prevent occupancy inference, we add random meaningless characters to the end of commands. We also send meaningless commands at random time intervals to the mote.

Chapter 4

Implementation 2: OpenTherm

OpenTherm is the second iteration of the SPOTlight system described in Chapter 3. In this chapter, we discuss the details of OpenTherm[1]. We note that this implementation has been approved for electrical safety by the appropriate regulatory agency and is therefore eligible for deployment outside of our laboratory.

4.1 Actuation and Sensing

Actuation and sensing consists of a desktop fan/heater (Figure 4.1) to maintain user comfort, sensors to measure air temperature and occupancy, actuators to turn the fan/heater on or off and control its speed, an RPi that acts as both a network and a compute node, and a device application that runs on the RPi to communicate with other software elements of the system.

This component consists of two hardware devices that we designed and implemented: an actuation box, and a sensing box. The **actuation box** contains the RPi and actuators. The sensors are placed in a separate **sensing box** that is closer to the user. Here, we describe how each of these components are implemented and how they work together.

4.1.1 Heating and Cooling Device

OpenTherm uses the same heating and cooling device as the one discussed in Section 3.1.4. The device has two separate power cords; one for the heating coil, and one for the fan.

[1]The source code for OpenTherm is available at https://github.com/AlimoRabbani/SPOTstar/

Figure 4.1: Hardware components of actuation and sensing in OpenTherm. The fan/heater is shown on the left side, and the actuation box is on the right side. In this picture, the two independent power cords of the modified fan/heater are connected to the box. The modular design of the box allows usage of different types of fans and heaters.

Figure 4.1 shows this device, with its two power cords beside the actuation box.

4.1.2 Sensing

To reduce costs, OpenTherm uses default values for humidity and clothing level in the PPV equation. Specifically, given that OpenTherm is meant to be deployed in an HVAC-controlled office space, it assumes that the humidity is controlled to 50%. Also, it assumes that an office worker's metabolic rate is a constant $1.2met$ and wearing $0.6clo$. Thus, it requires only two sensors: a temperature sensor and a motion sensor. Data from these sensors are used to compute the PPV value as discussed in Section 1.2.2.

The temperature and motion sensors are thermally separated and are placed in the sensing box along with an analog-to-digital converter (ADC) so that only digital values travel on the sensing communication link, reducing the effect of noise.

Temperature Sensor. To obtain temperature readings, we use the AD22100 surface-mount temperature sensor with 0.1 °C resolution [27]. The temperature sensor has an analog output and is connected to the RPi through the ADC. This sensor's temperature values are later used in PPV calculations.

Occupancy Detection. We use the AMN22111 passive infrared human detection sensor [28]. It outputs analog values that are converted to values between 0 and 1000 on the RPi. When there is no movement, the sensor output values are approximately 500. Each movement causes the sensor to first generate one value close to 1000 and then another close to 0. The closer these values are to 1000 and 0, the greater the intensity of movement. Over a 30-second window, a standard deviation close to 0 indicates almost no movement, and thus no occupancy, while higher standard deviations correspond to more movement (See Figure 4.2). Note that there is an approximately 30s delay in detecting occupancy with this approach.

Analog-to-Digital Converter. Both the temperature and occupancy sensors generate analog outputs, and, since the RPi does not have analog inputs, we connect a MAX11612 analog-to-digital converter [29] to the RPi through the I2C serial pins. The ADC converts sensor outputs to 12bit digital signals and sends them to the RPi upon request.

4.1.3 Actuation

The actuation box physically controls the fan/heater using relays, a custom made AC power control circuit, and an RPi. It has two power controllers, one for the fan and one

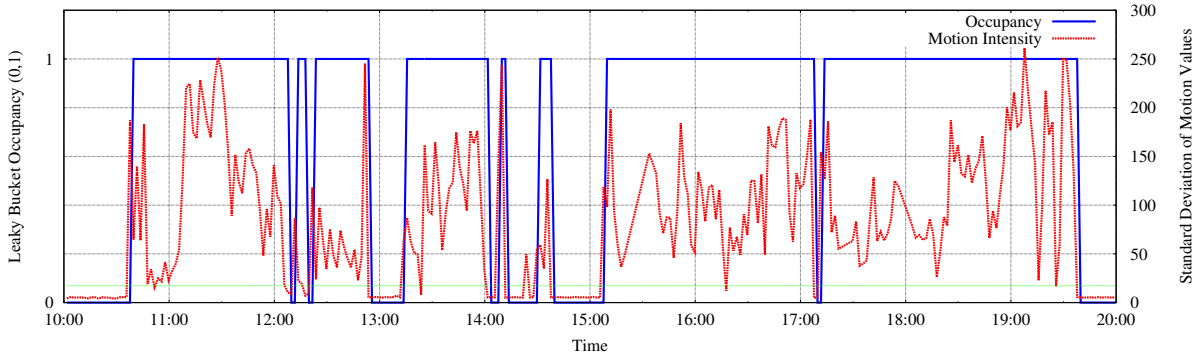

Figure 4.2: An example of occupancy inference, based on the standard deviation of motion values (i.e. motion intensity) during 30-second time windows over a 10-hour period. Ground truth occupancy is shown in blue and the motion threshold is shown in green. A threshold of $T_o = 17.25$ determines occupancy in each time interval. The results show 96% accuracy in detecting occupancy in our deployment.

for the heater (Figure 4.3).

Relays. Two RPi GPIO output pins are connected to two electromechanical relays[2] to close and open the AC circuit of the fan and the heater independently. Upon receiving a command from the control app (see Section 4.2), the device app on the RPi sets the two GPIO outputs to 0V or 3V respectively to execute the command.

AC Power Control Circuit. We implemented a standard AC-control circuit [30] to control the fan speed. It limits the current going through the fan using a TRIAC that modulates the current based on a control signal from a 12-bit MAX5805 digital-to-analog converter (DAC). The DAC's output voltage is controlled by the RPi using the I2C serial protocol. The left hand side of Figure 4.3 shows the PCB that we manufactured.

Raspberry Pi. The RPi runs the Raspbian operating system, and is connected to and powered through our custom-built AC control circuit with a 40-pin ribbon cable. It runs the device app and controls status lights on the box by toggling output signals on GPIO pins. In a networked configuration, we use an Edimax EW-7811Un USB dongle to connect the RPi to the building's WiFi network.

[2]We use electromechanical relays, rather than solid state relays, to reduce cost.

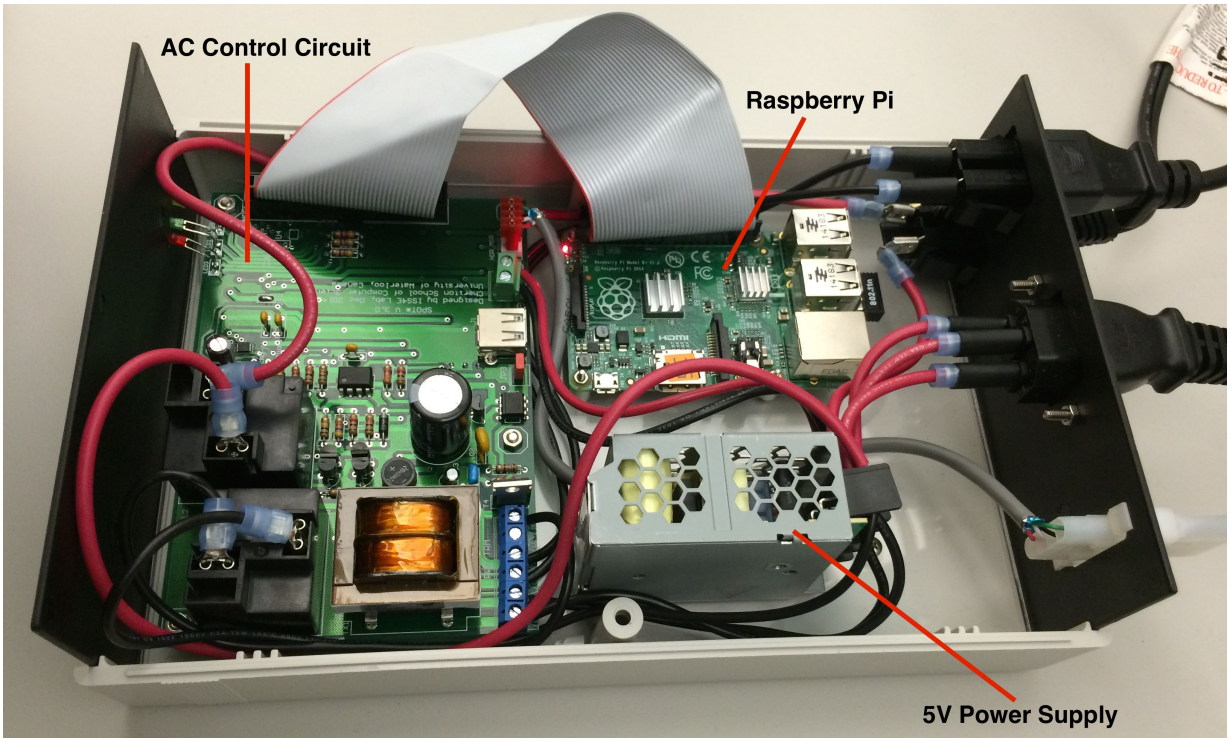

Figure 4.3: Inside of OpenTherm's actuation box. The AC control circuit on the left side communicates with the RPi to turn the fan/heater on or off, and set the fan's speed. The RPi, can connect to the building's WiFi network using a WiFi USB dongle, and a 5V 3A stable power supply keeps the box running.

4.1.4 Failure Recovery

As OpenTherm is designed to work with minimal user interaction, it is important to ensure that the system has a low failure rate, and that it can recover from them. We identified two important types of failures: overheating, and WiFi disconnections.

Overheating. By design, OpenTherm does not overheat the room. However, a software glitch, or a hardware failure can rarely cause the heater to be turned on for an extended period of time. We have several mechanisms in place to make sure this cannot happen.

1. The RPi checks for CPU temperature every two minutes, and shuts down everything if it goes above 80° Celsius.
2. A mechanical thermostat, which is set to its maximum temperature, does not let the heater turn on if the air temperature is high. This thermostat is a built-in component of our fan/heater.
3. The fan/heater has a tip sensor that prevents the fan or the heater from turning on if it is tipped over.
4. We cannot turn the heating coil on without the fan spinning, as it has a plastic enclosure that cannot tolerate high temperatures. Since we separated the wiring for these two, it is possible that a failure causes only the heating coil to turn on. A thermal fuse shuts the power off, if only the heating coil is turned on.

WiFi disconnections. A USB WiFi dongle is connected to the RPi. Sometimes, the electrical current drawn by the dongle is higher than what the RPi can provide, which causes it to disconnect from the network. The RPi does not automatically recover from this failure, causing OpenTherm systems to go offline. On the RPi, we check the assigned IP address on the wlan interface every 10 minutes, and restart networking if we cannot find an IP.

4.2 Device App

The device app has several tasks. It collects data from sensors and transmits them to the control app locally or over the network. It also executes commands received from

the control app. Using the I2C protocol, it reads sensor measurements from the ADC connected to the RPi.

The device app collects motion data twice every second, and sends it to the control app. Hailemariam et al. report that occupancy can be reliably detected by finding the standard deviation of the AMN23111 motion sensor [28] data every two minutes [23]. However, we found that the more sensitive AMN22111 motion sensor [28] allows us to lower the occupancy detection interval from 2 minutes to 30 seconds.

To reduce the amount of inter-process and network communication, the device app computes the standard deviation of raw motion values every 30 seconds, and sends only this value to the control app. It also reads and transmits the measured temperature every 10 seconds.

Upon receiving a command from the control app, the device app toggles GPIO outputs connected to relays to execute the command. In addition, it communicates with the DAC using I2C protocol to alter its output and set the speed of the fan. Due to the design of our selected fan/heater, to guarantee safe operation, we must make sure that the fan always spins with its maximum speed when the heating coil is powered.

4.3 Control Application

The control app listens for RPC connections from the device app. Each call from the device app updates either the temperature or the standard deviation of motion values. The control app passes these values to the DB app to be stored. It also makes a control decision based on occupancy and PPV, and invokes the appropriate procedures on the device app, as discussed next.

4.3.1 Occupancy Inference

The control app receives a standard deviation value O that represents movement intensity in every 30-second period from the device app. It determines if the station is occupied if this value exceeds a threshold $T_o = 17.25$ [23]. Note that in a shared office, background movements may cause false positives. To avoid this situation, we employ a low-pass filter in the form of a leaky bucket with water level L and capacity C as follows:

- L is 0 when the application starts.

- On update, if $O \geq T_o : L = min(L + 1, C)$
- On update, if $O < T_o : L = max(L - 1, 0)$

The control app infers that the station is occupied if $L = C$ and unoccupied if $L = 0$. Intermediate values for L imply that the user has recently left, recently arrived, or there was background movement close to the station. Therefore, it makes decisions oblivious to user presence when $0 < L < C$ (see Algorithm 1, described below).

4.3.2 Control Decisions

The control app makes reactive control decisions using the inferred occupancy and measured temperature data as shown in Algorithm 1. On receiving a motion sensor update (i.e. every 30 seconds), it updates the occupancy leaky bucket, then calculates the PPV value assuming a fan speed of zero. If the station is definitely occupied and the PPV value exceeds the comfort range T_c (i.e., the worker is too hot), the achievable PPV is re-calculated for each fan speed value in increments of $0.1ms^{-1}$, starting from zero, until the PPV (at that air speed) is within the comfort range or we reach the maximum possible fan velocity of $2.1ms^{-1}$. This determines the minimum fan speed needed to achieve a desired personal comfort level. On the other hand, if the station is occupied and the PPV value lies below the comfort range (i.e., the worker is too cold), then the heating coil is turned on, and, for safety, the fan speed is set to its maximum level.

4.4 Data Store and DB Application

We use MongoDB to store data. We also implement a DB app that restricts database access to limited functions (e.g. inserting and querying occupancy and temperature, querying users, and inserting device state) for better security. The DB app communicates with the control app to log events and updates, and provide it with worker preferences. It also communicates with the web app to store worker preferences and provide data to the web app for visualization. Because of OpenTherm's flexible architecture, we can run the DB app and the MongoDB storage locally on the RPi, or in the cloud. The data model used in OpenTherm is presented in Appendix B.

The data is primarily used to show historical data to users, get feedback from them, and enable central administration of all devices in our deployment. In addition, we can

Algorithm 1 Control app's MakeDecision procedure

1: **if** $L = 0$ **and** $Heat = true$ **then**
2: $\quad StopHeating()$
3: **else if** $L = 0$ **and** $Cool = true$ **then**
4: $\quad StopCooling()$
5: **else if** $L > 0$ **and** $Heat = true$ **and** $PPV > 0 - T_c$ **then**
6: $\quad StopHeating()$
7: **else if** $L > 0$ **and** $Cool = true$ **and** $PPV < T_c$ **then**
8: $\quad StopCooling()$
9: **else if** $L = C$ **and** $PPV > T_c$ **then**
10: $\quad S \leftarrow CalculateSpeed()$
11: $\quad StartCooling(S)$
12: **else if** $L = C$ **and** $Heat = false$ **and** $PPV < 0 - T_c$ **then**
13: $\quad StartHeating()$

further use this data to mine occupancy patterns and use them to predict occupancy, which is useful in an integrated system discussed in 6.3. Strange temperature patterns can also be extracted from our data, to provide insight to building managers. For example, in our current deployment, some offices are heated up to 28° Celsius over winter nights.

4.5 Web Application

We design and implement the web application using the Flask microframework to:

- Collect votes from the user during training periods.
- Provide a manual override to the user.
- Visualize temperature, occupancy, and comfort data for users and administrators.
- Debug, monitor, and administer.

The web application runs on a WSGI Apache instance, which is proxied through a publicly available Apache web server[3]. To ensure privacy and security, we use HTTPS [31] and require users to login with their credentials.

[3]https://blizzard.cs.uwaterloo.ca/spotstar/

4.5.1 Training Period

OpenTherm has an optional device training to estimate the affine translation between PMV and PPV values [10]. Users can start device training periods at will, and PMV is used by default if they never train the system. We collect votes from the users based on the 7-point ASHRAE scale and match them with the PMV value at the time of voting. Once the user ends the training we run a least squares linear regression on the collected points to determine $f_{ppv}(pmv)$. After the training, the new PPV equation is used to predict user comfort. The training typically takes about a day, and we advise users to train their devices once in the beginning of summer and winter to keep them calibrated.

4.5.2 Comfort Offset

In addition to infrequent training, we provide a manual offset override to allow users to adjust their comfort level as needed, such as when they are unwell, or when they are wearing more or fewer clothes than usual. The offset B_o is 0 by default and adjusts the PPV equation in the following way:

$$PPV = a * PMV + b - B_o \tag{4.1}$$

When $B_o < 0$, the user prefers cooler conditions, and when $B_o > 0$ warmer conditions are preferred.

4.6 Graphical User Interface

Standard browser-viewable content based on jQuery and Bootstrap is generated by the web app. In a networked setup, this GUI is accessible on users' desktop computers and smart phones.

In an isolated local OpenTherm setup, we equip the actuation box with a 7-inch resistive touch screen LCD. The LCD is connected to the RPi using an HDMI cable through Adafruit's touch screen controller board [32]. In this setup, the data remains physically on-site and the user can access the GUI only on the box from a web browser user interface (see Figure 4.4).

Figure 4.4: For isolated installations of OpenTherm, a 7-inch touch screen LCD is mounted on the actuation box. This LCD is connected to the RPi inside the box and provides a local GUI to the user. In this configuration, which has the best privacy protection, occupancy data remains physically in the office.

Chapter 5

Evaluation

We received approval from the Office of Research Ethics at the University of Waterloo to install and evaluate OpenTherm in offices on-campus. To recruit, we sent an invitation[1] to all building residents; participants self-selected primarily because of their dissatisfaction with the existing HVAC systems comfort level.

At the time of writing this thesis, we have deployed 15 OpenTherm devices in both offices and cubicles in our building. On average, each device has been working for about 5.5 months, a total of ~58,000 hours of operation. We have collected data from our earliest deployed device for about 11 consecutive months, and for almost a month from our latest deployment. Over the last 4 months, only two failures have happened, and re-plugging the device to the power outlet fixed the problem in both occurrences[2].

In this section, we compute the cost of OpenTherm, measure its accuracy in detecting occupancy, evaluate users' comfort when using OpenTherm with both subjective and objective measures, and explore its energy consumption.

5.1 Hardware Cost

Table 5.1 shows the hardware components used in OpenTherm with approximate prices we paid, and estimated cost in mass production. Table 5.2 shows the cost for SPOTlight.

[1] Appendix D shows the recruitment letter we sent to faculty, staff, and students on-campus who had offices. We used the name SPOT* instead of OpenTherm at that time.

[2] Users are provided with a 1-page user manual to troubleshoot the device on their own. See Appendix C.

Item	Prototype Price	Est. Volume Price
Raspberry Pi	$40	$5
WiFi dongle	$10	$5
sensors	$20	$10
AC circuit components	$50	$20
fan/heater	$25	$20
PCB manufacturing	$20	$10
enclosures	$10	$5
wires, connectors, etc	$10	$5
Total	**$185**	**$80**

Table 5.1: Cost breakdown of hardware elements used in **OpenTherm**. The table shows our approximate prototype cost, and the estimated mass-production price for each element. A $17 touchscreen panel is used for the standalone version of OpenTherm. However, our deployment currently consists only of network-enabled devices, which do not include the touch screen panel.

We observe that our second iteration, OpenTherm, has a significantly lower cost. Note that the per-user cost of software and cloud servers is negligible for large deployments. Therefore, we do not include it in these two tables. Even for a single prototype, the overall system cost for OpenTherm is only $185, dropping to $80 in volume production.

5.2 Occupancy Detection

Figure 4.2 shows how standard deviations of motion data translate into occupancy inferences during a typical 10-hour period. To estimate the accuracy of occupancy detection, we measured occupancy using both a video camera (with manual tagging of occupied periods) and the passive infrared sensor used in OpenTherm. Over a 3-day period, OpenTherm had a 96% accuracy, which is comparable to numbers reported in [23].

5.3 Effectiveness In Maintaining Worker Comfort

We measured the effectiveness of OpenTherm in maintaining user comfort in two ways. First, we compute the average absolute discomfort in the presence and absence of the

Item	Prototype Price	Est. Volume Price
Zolertia Z1	$150	$100
Fan Heater	$35	$25
Motion Sensor	$35	$20
Solid State Relay	$30	$25
9V AC Transformer	$25	$15
Control Circuit	$20	$10
Total	**$295**	**$195**

Table 5.2: Cost breakdown of hardware elements used in **SPOTlight**. The table shows our approximate prototype cost, and the estimated mass-production price for each element.

OpenTherm during the deployment period. Second, we measure how frequently users needed to manually override the system, as an indicator of how many times the users felt uncomfortable.

5.3.1 Average Absolute Discomfort

Gao et al. introduced the **average absolute discomfort** metric to quantify how uncomfortable a user feels [10]. Let $d(t)$ be the absolute discomfort at time t defined as

$$d(t) = max(|ppv(t)| - T_c, 0) \tag{5.1}$$

where threshold T_c determines a PPV range in which the user is comfortable. A T_c of 0.5 means the user is comfortable at time t if $-0.5 < ppv(t) < 0.5$. To be consistent with this work, we set T_c to 0.5 in our evaluation. Then, $\bar{d}$ or average absolute discomfort, is defined to be the average of $d(t)$ over time conditional on the user being present. Specifically, let $m(t) = 0$ when the workspace is not occupied and be equal to 1 when occupancy is detected. Then,

$$\bar{d} = \frac{\Sigma_t d(t)m(t)}{\Sigma_t m(t)} \tag{5.2}$$

To quantify OpenTherm's performance, we need to calculate the average absolute discomfort of users with and without OpenTherm. Observe that OpenTherm performs no control actions when the workspace is unoccupied. So, the average absolute discomfort in the *absence* of OpenTherm is simply the average of $d(t)$ over time, when the workspace is

unoccupied. Thus, to measure average discomfort in absence of OpenTherm, we define:

$$\bar{d^*} = \frac{\Sigma_t d(t) m'(t)}{\Sigma_t m'(t)} \tag{5.3}$$

where $m'(t)$ is 0 when the user is present and 1 when the the workspace is not occupied (i.e., $m' = 1 - m$). To be conservative, we further set $m'(t) = 0$ between 7pm and 7am regardless of occupancy. Therefore, we measure average discomfort in absence of OpenTherm, only for the normal working hours, when our subjects are likely to be at their workstations.

In our deployment, we found that the average absolute discomfort for all fifteen users in the presence of OpenTherm is 0.07 compared to 0.21 in its absence. Thus, OpenTherm improves user comfort by 67% in this trial. Moreover, this value is substantially lower than the comparable value for SPOT [9], a similar reactive system, whose reported average absolute discomfort is 0.20, and roughly comparable to that of SPOT+ [10], a more complex predictive system, that reduces average absolute discomfort to 0.02. As we will see below, this is despite a nearly one order of magnitude reduction in energy consumption.

Figure 5.1 shows a comparison between average discomfort when OpenTherm is in use and when it is not in use for each individual user. We find that all users experienced increase comfort with OpenTherm, with some users (at stations 5, 7, and 8) experiencing significantly greater comfort, which we corroborated by direct interviews. Although illustrative, we plan to report on a detailed study of user experiences with OpenTherm in future work.

5.3.2 Manual overrides

The offset slider in our GUI is an indicator of how many times the users felt uncomfortable during the course of our deployment. Therefore we define E_s as the measure of user discomfort as:

$$E_s = \frac{\text{number of slider events}}{\text{total occupied hours}} \tag{5.4}$$

The average E_s for all users is 0.08, which considering weekly occupied hours, means that the average user changed the offset about three times every two weeks. Of course, this could be due to excellent thermal regulation by the building HVAC system. However, participants were self-selected as those who were most dissatisfied with their existing comfort. Thus, we draw the conclusion that OpenTherm was able to correctly set the participants' thermal envelope.

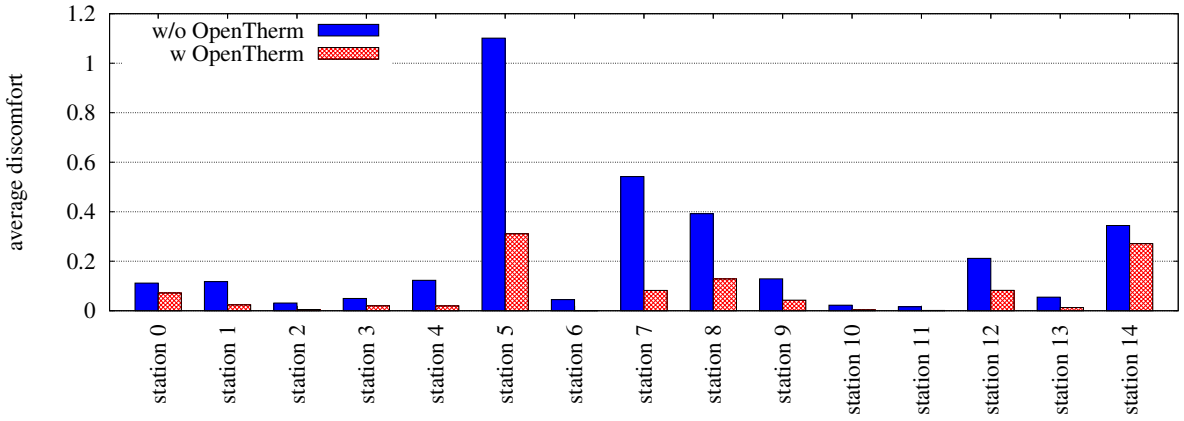

Figure 5.1: Average discomfort of users when OpenTherm is being used, and when it is not being used. For each user, the average discomfort decreases significantly when OpenTherm is maintaining comfort.

5.4 OpenTherm Energy Cost

OpenTherm has three sources of energy consumption:

- The actuation box, including the RPi and the AC control circuit (constant 3.5 watts),
- The fan, consuming linearly between 8 and 15 watts depending on speed,
- The heater coil, consuming 700 watts when turned on.

We estimate the total energy consumption of each deployed OpenTherm instance by summing up the energy consumption of the three sources above. We find that, despite being provisioned with a 700W heating coil, the average monthly energy consumption of each OpenTherm device during the deployment is approximately 10 kWh, and the median consumption is only 4.7 kWh. This is roughly equal to a single 40W light bulb left on for eight hours each day. Compared to SPOT and SPOT+, which consume an average of roughly 150kWh and 230kWh per month, respectively [9, 10], OpenTherm reduces energy consumption by an order of magnitude, while providing better or comparable user comfort. Figure 5.2 shows average monthly energy consumption for each deployed OpenTherm device.

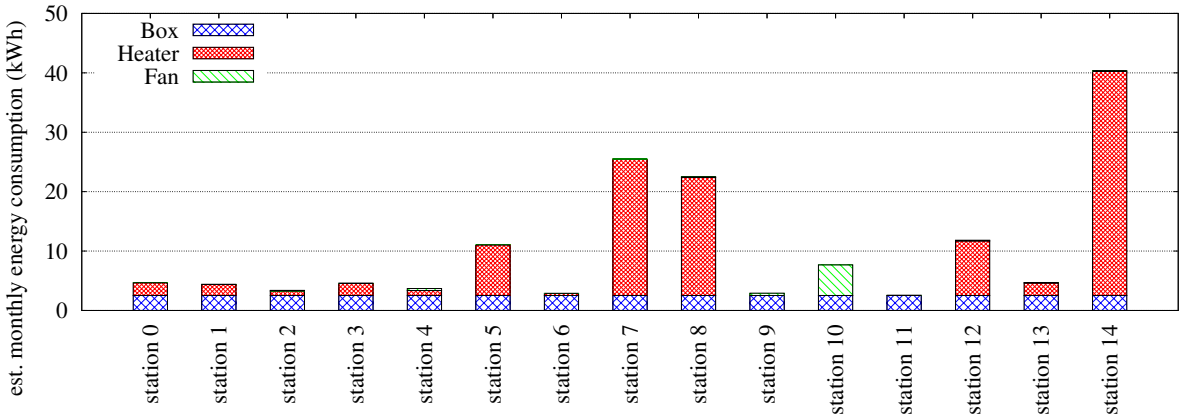

Figure 5.2: Estimated monthly energy consumption for each OpenTherm device in kWh. A significant portion of consumption belongs to the heating coil and operating the system, and a small amount is consumed to power the fan. Station 10 is surprisingly warm, even in winter, hence the significant fan consumption.

Chapter 6

Conclusions

Motivated by the need for a practical system for personal thermal comfort, we presented the design and evaluation of the OpenTherm system.

6.1 Meeting Design Goals

We now discuss how we met our goals of:

- **G1** Low per-unit cost
- **G2** Plug-and-play deployment
- **G3** Legacy compatibility
- **G4** Ease of use
- **G5** Ease of training

To meet goal **G1** (reducing system cost) OpenTherm is built around the Raspberry Pi B+ single-board computer and a commodity heater/fan that we purchased from a hardware store. The RPi costs about $40 (newer versions cost $5!) and the heater/fan costs about $35. The remainder of the costs, for sensors and actuators, adds to this base cost, but we have tried our best to keep costs low. With mass production, we estimate that the unit cost would be around $80, which is within striking range of the ARPA-E goal of $60/user [15].

Given that the closest related work is the two SPOT systems [9, 10], we now discuss how we achieve a similar goal as them, but at lower cost.

- SPOT uses a $200 Kinect for occupancy sensing. Instead, OpenTherm uses a $20 passive IR motion sensor.
- SPOT also uses a Kinect to sense a worker's clothing level. Instead, we assume that the default clothing insulation factor is $0.6clo$ and provide a simple web-based user interface for workers to indicate that their clothing level is lower or higher than this default.
- OpenTherm measures air temperature using an inexpensive temperature sensor and assumes that this temperature is identical to the background radiant temperature, so does not measure background temperature separately.

OpenTherm is designed to be rapidly deployed (goal **G2**): all that needs to be done during a deployment is for the system to be placed on the user's desk and a sensor box placed near the user. A single power cord is plugged in and a laptop is used to enter configuration parameters in a central database using a GUI. The entire process takes only a few minutes.

Legacy compatibility (goal **G3**) is achieved trivially: if the central system provides adequate user comfort, then OpenTherm does nothing, becoming active only when the personal thermal envelope of the user needs modification.

Ease of use and training (goals **G4** and **G5**) are met by means of the web-based GUI for training and comfort setpoint selection. There are no controls on the desktop system itself, only a flashing yellow light indicating normal operation. If the light is not flashing, the user simply unplugs and re-plugs the device and normal operation continues. Thus, the user manual is a single page of text.

6.2 Insights

We gained several insights from our work. First, in an air-conditioned building, most comfort parameters except temperature are kept stable enough not to affect the PMV equation significantly. Therefore it is not necessary to sense every component of the PMV equation. OpenTherm estimates worker comfort with only temperature measurements and judicious choice of default parameters for the PMV equation. Errors in this estimate are corrected by a user interface that allows manual overrides. This also returns a certain degree of control over their own comfort to users, which they appreciate.

The design choice of using flexible, relocatable software components has also proven to be a good one. With little effort, we can configure a system to be standalone, suitable for privacy-sensitive workers, or to be networked, which opens up the possibility of coordinating OpenTherm actions with that of a central HVAC, something we would like to pursue in future work. We hope that heater/fan manufacturers will, some day, build in a mote-like lightweight embedded compute platform into their devices, allowing us to deploy OpenTherm on them, by moving the control logic, storage, and web app to the Internet cloud. This would be a fascinating use case for the Internet of Things.

Our choice of using a Raspberry Pi as the compute platform was not straightforward. We initially considered the Arduino, a smartphone running Android, and a Zolertia Z1 mote as alternatives. Indeed, our first deployment, described in Chapter 3, was using the Z1 mote, which has an MSP430 controller, a minimal operating system, and a few 10s of KB of RAM. After struggling for some months with this platform, we were pleasantly surprised to find that the Raspberry Pi was not only much more powerful than the Z1, but was two-and-a-half times cheaper. We did not really need the smartphone screen, and the Arduino has no operating system and is not much cheaper than the RPi, hence our final choice. Through GUI-enabled interaction and information access, users can potentially train and personalize equipment such as office lighting. Thus, OpenTherm can serve as the basis for a per-office Internet-of-Things deployment.

6.3 Limitations and Future Work

Our work suffers from one primary limitation. We were motivated, in part, by the goal of reducing building energy use by choosing a higher setpoint in summer and a lower setpoint in winter compared to standard operating procedure. Unfortunately, we were unable to persuade our building managers to actually let us modify the building temperature setpoint. Thus, we are unable to determine whether OpenTherm will, indeed, reduce building energy consumption. We hope to address this in future work.

An additional goal in future work is to do a more detailed user study to estimate the gain in productivity and user comfort from the use of OpenTherm. This survey is prepared and reviewed by the Office of Research Ethics, and is ready to be conducted. We are also keen to explore the joint optimal control of centralized HVAC systems when augmented with desktop devices.

References

[1] Mark Feldmeier and Joseph A Paradiso. Personalized hvac control system. In *Internet of Things (IOT), 2010*, pages 1–8. IEEE, 2010.

[2] Fred S Bauman, Thomas G Carter, Anne V Baughman, and Edward A Arens. Field study of the impact of a desktop task/ambient conditioning system in office buildings. *ASHRAE Transactions*, 104:1153, 1998.

[3] Luis Pérez-Lombard, José Ortiz, and Christine Pout. A review on buildings energy consumption information. *Energy and buildings*, 40(3):394–398, 2008.

[4] Lawrence Berkeley National Laboratory. Closing in on zero-energy buildings. http://eetd.lbl.gov/newsletter/nl29/, 2009. Accessed: Jan 2015.

[5] Varick L Erickson, Yiqing Lin, Ankur Kamthe, Rohini Brahme, Amit Surana, Alberto E Cerpa, Michael D Sohn, and Satish Narayanan. Energy efficient building environment control strategies using real-time occupancy measurements. In *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, pages 19–24. ACM, 2009.

[6] Varick L Erickson, Miguel Á Carreira-Perpiñán, and Alberto E Cerpa. OBSERVE: Occupancy-based system for efficient reduction of HVAC energy. In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, pages 258–269. IEEE, 2011.

[7] RJ Dear, T Akimoto, EA Arens, G Brager, C Candido, KWD Cheong, B Li, N Nishihara, SC Sekhar, S Tanabe, et al. Progress in thermal comfort research over the last twenty years. *Indoor air*, 23(6):442–461, 2013.

[8] Hui Zhang, Edward Arens, DongEun Kim, Elena Buchberger, Fred Bauman, and Charlie Huizenga. Comfort, perceived air quality, and work performance in a low-

power task–ambient conditioning system. *Building and Environment*, 45(1):29–39, 2010.

[9] Peter Xiang Gao and S. Keshav. SPOT: a smart personalized office thermal control system. In *Proceedings of the fourth international conference on Future energy systems*, pages 237–246. ACM, 2013.

[10] Peter Xiang Gao and S Keshav. Optimal Personal Comfort Management Using SPOT+. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, pages 1–8. ACM, 2013.

[11] Hal Hodson. Smart clothes adapt so you are always the right temperature. *New Scientist*, January 30 2016.

[12] Diana Ürge-Vorsatz, LD Danny Harvey, Sevastianos Mirasgedis, and Mark D Levine. Mitigating co2 emissions from energy use in the world's buildings. *Building Research & Information*, 35(4):379–398, 2007.

[13] Mark Levine, Diana Ürge-Vorsatz, Kornelis Blok, Luis Geng, Danny Harvey, Siwei Lang, Geoffrey Levermore, A Mongameli Mehlwana, Sevastian Mirasgedis, Aleksandra Novikova, et al. Residential and commercial buildings. *Climate change*, 20:17, 2007.

[14] A Chen. Closing in on zero-energy buildings. *EETD Fall Newsletter*, 8(2), 2009.

[15] ARPA-E. Arpa-e delta program: Delivering efficient local thermal amenities. http://arpa-e.energy.gov/?q=arpa-e-site-page/view-programs. Accessed: Jan 2016.

[16] Poul O Fanger et al. Thermal comfort. analysis and applications in environmental engineering. *Thermal comfort. Analysis and applications in environmental engineering.*, 1970.

[17] Richard De Dear and Gail Schiller Brager. Developing an adaptive model of thermal comfort and preference. *ASHRAE Transactions*, 104(1):145–167, 1998.

[18] Hui Zhang, Edward Arens, Sahar Abbaszadeh Fard, Charlie Huizenga, Gwelen Paliaga, Gail Brager, and Leah Zagreus. Air movement preferences observed in office buildings. *International Journal of Biometeorology*, 51(5):349–360, 2007.

[19] Jørn Toftum. Air movement–good or bad? *Indoor Air*, 14(s7):40–45, 2004.

[20] Tyler Hoyt, Hui Zhang, and Edward Arens. Draft or breeze? preferences for air movement in office buildings and schools from the ashrae database. 2009.

[21] Tyler Hoyt, Kwang Ho Lee, Hui Zhang, Edward Arens, and Tom Webster. Energy savings from extended air temperature setpoints and reductions in room air mixing. In *International Conference on Environmental Ergonomics 2009*, 2009.

[22] Rayman Preet Singh, Srinivasan Keshav, and Tim Brecht. A cloud-based consumer-centric architecture for energy data analytics. In *Proceedings of the fourth international conference on Future energy systems*, pages 63–74. ACM, 2013.

[23] Ebenezer Hailemariam, Rhys Goldstein, Ramtin Attar, and Azam Khan. Real-time occupancy detection using decision trees with multiple sensor types. In *Proceedings of the 2011 Symposium on Simulation for Architecture and Urban Design*, pages 141–148. Society for Computer Simulation International, 2011.

[24] Inc Royal Sovereign International. Oscillating Fan Heater - HFN-20. http://www.royalsovereign.ca/appliances/heaters/heater-hfn-20-785.html. Accessed: Jan 2015.

[25] Inc. Linux Kernel Organization. Universal tun/tap device driver. https://www.kernel.org/doc/Documentation/networking/tuntap.txt. Accessed: Mar 2016.

[26] The Internet Engineering Task Force (IETF). A nonstandard for transmission of ip datagrams over serial lines: Slip. https://tools.ietf.org/html/rfc1055. Accessed: Mar 2016.

[27] Analog Devices. Voltage Output Temperature Sensor with Signal Conditioning. http://www.analog.com/static/imported-files/data_sheets/AD22100.pdf. Accessed: Jan 2015.

[28] Panasonic. MP Motion Sensor NaPiOn. http://www3.panasonic.biz/ac/e/control/sensor/human/napion/index.jsp. Accessed: Jan 2015.

[29] Maxim Integrated. Low-Power, 4-/8-/12-Channel, I2C, 12-Bit ADCs in Ultra-Small Packages. http://www.maximintegrated.com/en/products/analog/data-converters/analog-to-digital-converters/MAX11612.html. Accessed: Jan 2015.

[30] Giorgos Lazaridis. Voltage Controlled AC Light Dimmer. http://pcbheaven.com/circuitpages/Voltage_Controlled_AC_Light_Dimmer/. Accessed: Jan 2015.

[31] The Internet Engineering Task Force. HTTP Over TLS. http://tools.ietf.org/html/rfc2818, May 2000. Accessed: Jan 2015.

[32] Adafruit. Tfp401 hdmi/dvi decoder to 40-pin ttl breakout - with touch. http://www.adafruit.com/products/2219. Accessed: Jan 2015.

[33] ISO ISO. 7730: Ergonomics of the thermal environment–analytical determination and interpretation of thermal comfort using calculation of the pmv and ppd indices and local thermal comfort criteria. *International Organization for Standardization*, 2005.

[34] U.S. Energy Information Administration. Commercial buildings energy consumption survey - office buildings. http://www.eia.gov/consumption/commercial/reports.cfm, September 2010. Accessed: Jan 2015.

[35] U.S. Energy Information Administration. Overview of commercial buildings, 2003. http://www.eia.gov/consumption/commercial/reports.cfm, December 2008. Accessed: Jan 2015.

[36] Natural Resources Canada. Energy use data handbook, 1990 to 2010. http://oee.nrcan.gc.ca/corporate/statistics/neud/dpa/home.cfm. Accessed: Jan 2015.

[37] Zolertia. Z1 platform. http://zolertia.com/products/z1. Accessed: Jan 2015.

[38] American society of heating refrigerating and air conditioning engineers. *ASHRAE STANDARD: An American Standard: Thermal Environmental Conditions for Human Occupancy*. American Society of Heating refrigerationg and air conditioning engineers, 1992.

APPENDICES

Appendix A

Details of the PMV Model

The PMV model [16] predicts the average person's comfort as a function of four environmental variables (i.e. air temperature, radiant temperature, air velocity, humidity) and two personal variables (i.e. clothing and physical activity). The output of the PMV equation is a numerical value (i.e. vote) that is interpreted using the 7-point ASHRAE thermal sensation scale [38], shown in Table A.1.

PMV is computed as:

$$\begin{aligned} pmv =& (0.303 \cdot exp(0.036 \cdot M) + 0.028) \cdot \\ & \{(M - W) - 3.05 \cdot 10^{-3} \cdot (5733 - 6.99 \cdot (M - W) - p_a) \\ & - 0.42 \cdot ((M - W) - 58.15) - 1.7 \cdot 10^{-5} \cdot M \cdot (5867 - p_a) \\ & - 0.0014 \cdot M \cdot (34 - ta) - 3.96 \cdot 10^{-8} \cdot f_{cl} \cdot ((t_{cl} + 273)^4 \\ & - (\bar{t}_r + 273)^4) - f_{cl} \cdot h_c \cdot (t_{cl} - t_a)\} \end{aligned} \tag{A.1}$$

where t_{cl} is the clothing surface temperature, and W is the effective mechanical power which is 0 for most indoor activities. Variable t_{cl} can be evaluated by:

$$\begin{aligned} t_{cl} =& 35.7 - 0.028 \cdot (M - W) - I_{cl} \cdot \\ & (3.96 \cdot 10^{-8} \cdot f_{cl} \cdot ((t_{cl} + 273)^4 - (\bar{t}_r + 273)^4) + f_{cl} \cdot h_c \cdot (t_{cl} - t_a)) \end{aligned} \tag{A.2}$$

Variable h_c is the convective heat transfer coefficient, which is derived as:

Vote	Comfort Level
+3	Hot
+2	Warm
+1	Slightly Warm
0	Neutral
-1	Slightly Cool
-2	Cool
-3	Cold

Table A.1: 7-point ASHRAE scale in PMV model

Activity	**Metabolic Rate**	
	W/m^2	met
Reclining	46	0.8
Seated, relaxed	58	1.0
Sedentary activity	70	1.2
Standing, medium activity	93	1.6

Table A.2: Typical metabolic rates

$$h_c = \begin{cases} 2.38 \cdot |t_{cl} - t_a|^{0.25} & \text{if } 2.38 \cdot |t_{cl} - t_a|^{0.25} > 12.1 \cdot \sqrt{v_{ar}} \\ 12.1 \cdot \sqrt{v_{ar}} & \text{if } 2.38 \cdot |t_{cl} - t_a|^{0.25} < 12.1 \cdot \sqrt{v_{ar}} \end{cases} \tag{A.3}$$

Variable f_{cl} is the clothing surface area factor, which is derived as:

$$f_{cl} = \begin{cases} 1.00 + 1.290 I_{cl} & \text{if } I_{cl} \leq 0.078 m^2 \cdot K/W \\ 1.05 + 0.645 I_{cl} & \text{if } I_{cl} > 0.078 m^2 \cdot K/W \end{cases} \tag{A.4}$$

In practice, the metabolic rate and the clothing insulation are first estimated by Table A.2 and Table A.3. Given the clothing insulation I_{cl}, we calculate the clothing surface temperature t_{cl} and the convective heat transfer coefficient h_c by applying Equation A.2 and A.3 in an iterative fashion. Finally, by using Equation A.1 and A.4, we estimate the Predicted Mean Vote.

Clothing	**Clothing Insulation (I_{cl})**	
	clo	$m^2 \cdot K/W$
Panties, T-shirt, shorts, light socks, sandals	0.30	0.050
Underpants, shirt with short sleeves, light trousers, light socks, shoes	0.50	0.080
Panties, petticoat, stockings, dress, shoes	0.70	0.105
Underwear, shirt, trousers, socks, shoes	0.70	0.110
Panties, shirt, trousers, jacket, socks, shoes	1.00	0.155
Panties, stockings, blouse, long skirt, jacket, shoes	1.10	0.170
Underwear with long sleeves and legs, shirt, trousers, V- neck sweater, jacket, socks, shoes	1.30	0.200
Underwear with short sleeves and legs, shirt, trousers, vest, jacket, coat, socks, shoes	1.50	0.230

Table A.3: Thermal insulation for different clothing levels

Appendix B

OpenTherm's Data Model

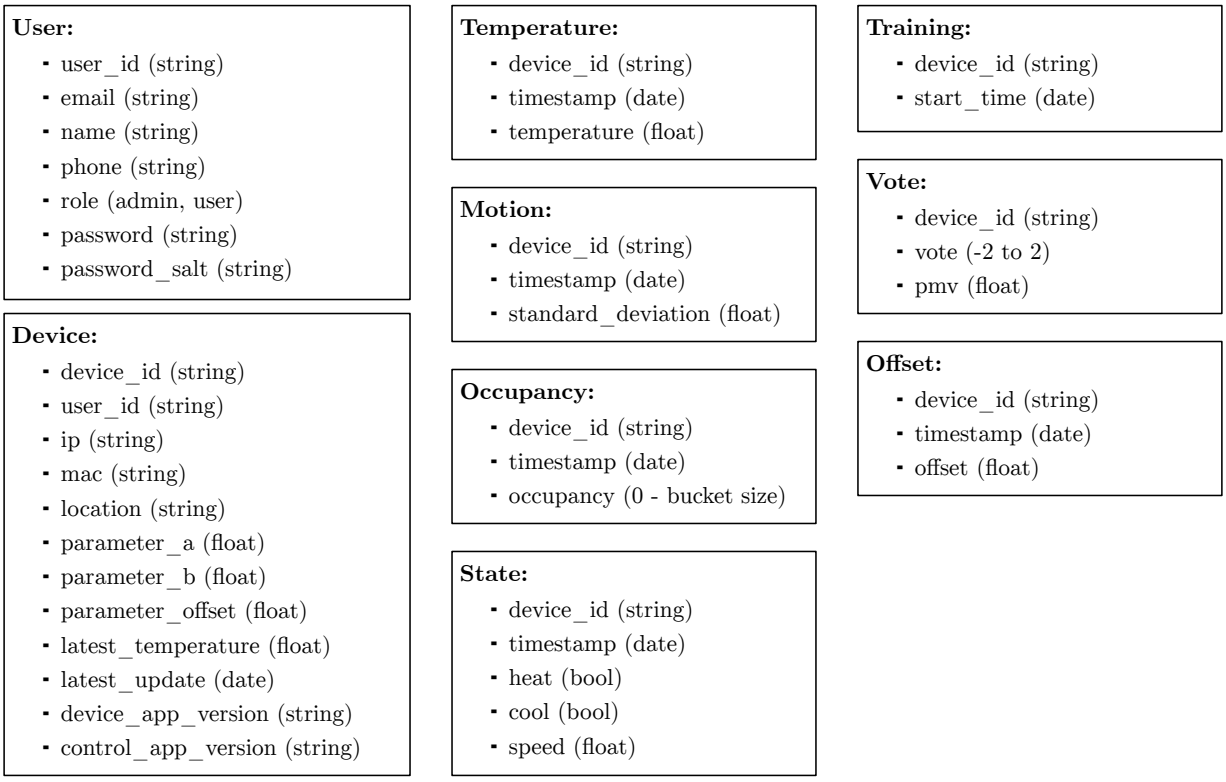

Figure B.1: The data model used in in OpenTherm. The data from the deployment is stored in a MongoDB database on a back-end server.

Appendix C

OpenTherm's Instruction Manual

OpenTherm User Manual

Setup Instructions

You have been given three devices: a fan/heater, a larger controller box, and a smaller sensor box.

Fan/Heater

- Place the fan/heater on your desk about 1.5 meters (4-5 feet) away from you.
- We recommend that you set the fan/heater to **oscillating mode** (I setting on the black I/O switch)
- As suggested by Plant Ops, we recommend that you keep the heater on mode I and not II.
- The thermostat needs to be on maximum all the time.

Sensor

- Place the sensor box on your desk, or mount it on a wall, closer than 2 meters (about 6 feet) away.
- The sensor box has a 90° viewing angle and should directly point at where you usually sit.
- The sensor box must be placed somewhere between you and the fan/heater (it needs to perceive temperature changes caused by the heater).
- Connect the sensor box to the controller box with the 4-pin connector.

Controller

- The location of the controller is not important. It is convenient to simply put the fan/heater on top.
- Connect one end of the power cord to the box.
- Connect the other end of the power cord to an outlet which is not overloaded.
- To avoid the need for resetting the circuit breaker in case of connecting too many devices, we recommend that a power bar with a resettable fuse be used to connect this device.

It takes 1-2 minutes for the device to boot up, and it should start operating normally as soon as the yellow light starts blinking. After the device starts working, you can login at https://blizzard.cs.uwaterloo.ca/spotstar/ with your UW email and selected password to customize the device or view its operation. You need not turn the device off when you leave for home: it will turn off automatically.

Troubleshooting

Light Color	State	Meaning	Action Required
Yellow	blinking	normal operation	none
	solid ON/OFF	interrupted operation	please re-plug the device
Red	ON	heater is on	contact administrator if: 1. red is on, and green is off 2. light state does not match real operation
	OFF	heater is off	
Green	ON	fan is spinning	
	OFF	fan is not spinning	

Contact

If you have any questions, please contact the project administrator at cogradab@uwaterloo.ca.

Appendix D

Recruitment Letter

OpenTherm Recruitment Letter

Faculty, staff, and students in Computer Science who have an office or a workstation at the Davis Center are invited to participate in a research study conducted by Alimohammad Rabbani, under the supervision of Prof. S. Keshav, in the School of Computer Science. As a participant, you will be given an experimental OpenTherm system, built by researchers at the Information Systems and Science (ISS4E) lab for personal thermal comfort, consisting of a standard heater/fan unit and custom-built controller. This system will monitor your comfort and heat or cool you to maintain *your* thermal comfort at all times without the need for your interaction. More information about the device and how it operates is available on our website (http://blizzard.cs.uwaterloo.ca/spotstar/about/).

The objectives of the study are to understand the performance of OpenTherm in daily use. Specifically, we will analyze the occupancy and temperature data that your system collects to learn how well OpenTherm works. We are also interested in examining the possibility of integrating OpenTherm with the building's central Heating, Ventilation and Air Conditioning (HVAC) system to create a more efficient hybrid centralized/decentralized HVAC system in the future.

If you are willing to participate, please register by filling out this form (https://blizzard.cs.uwaterloo.ca/spotstar/participate/) to allow us to obtain your name, UW email address, and room number. This information will only be used to register you as a participant in the study. If you do not wish to use an online form, please send your full name and DC room number by email to amrabban@uwaterloo.ca. Please be informed that only the first 65 respondents will be invited to participate in the study.

If you are selected for the study, we will ask you to meet with us at the ISS4E lab (DC 2554) to configure your OpenTherm device, see a demonstration of the system, and pick up a device for your own use. You can then take the device to your office and set it up. The device will initially use default comfort values for its operation. However, you can customize its behaviour with your feedback using a web-based user interface. You will receive an instruction sheet that will show you how. This sheet includes the steps that you should to take if the device malfunctions.

During the study, occupancy and temperature data will be collected by the device and sent to a secure server on campus. You will have read-only access to the data through our web application. We will use this data to analyze how well your system is working. We are planning to run the study for 3 years, but you can request to withdraw your participation at any time, and we will remove the device from your office, and erase all the stored data that we have collected from your office.

From time to time, we may request your participation in a web survey or in-person interview so that we can learn how well you like using the system. Participation in such a survey or interview is completely voluntary.

We have adopted several software and hardware prevention mechanisms to avoid overheating of the heater/fan unit:

- The fan is equipped with a tip sensor that mechanically shuts down the fan/heater if tipped over.
- The heater has a thermal fuse that shuts the power to the heating coil in case of overheating.
- The control box is flame-retardant, preventing the spread of fire if there is internal fire in the box.
- The control box constantly monitors its internal temperature and turns itself off if it overheats.

We have obtained Electrical Safety Authority (ESA), Plant Operations, and Safety Office approvals for this device.

It is important for you to know that any information that you provide, and the occupancy and temperature data that we collect from your office, will be confidential. All of the data will be summarized and no individual could be identified from these summarized results. The data, collected from this study will be maintained on a password-protected computer database in a restricted access area of the university. As well, the data will be electronically archived after completion of the study and maintained for three years and then erased. Should you have any questions about the study, please contact Alimohammad Rabbani at amrabban@uwaterloo.ca, or Prof. Keshav at keshav@uwaterloo.ca or (519) 8884567, Ext.34456.

Be assured that this study has been reviewed and received ethics clearance through a University of Waterloo Research Ethics Committee. However, the final decision about participation is yours. If you have any comments or concerns resulting from your participation in this study, please feel free to contact Dr. Maureen Nummelin in the Office of Research Ethics at 1-519-888-4567, Ext. 36005 or maureen.nummelin@uwaterloo.ca.

Thank you for considering participation in this study.