The SPOT* System for Flexible Personal Heating and Cooling

Alimohammad Rabbani and S. Keshav
School of Computer Science, University of Waterloo
Waterloo, Ontario, Canada
Technical Report CS-2015-02

ABSTRACT

Building on our prior work on the SPOT and SPOT+ **S**mart **P**ersonalized **O**ffice **T**hermal control systems, this paper presents SPOT*, a cost-effective, flexible system for personalized heating and cooling. Like its predecessors, SPOT* senses occupancy and worker comfort to reactively control office temperature. Specifically, it uses the Predicted Mean Vote equation to determine worker comfort and actuates a fan or a heater to adjust the comfort level so that it lies between -0.5 and +0.5 in the ASHRAE comfort scale. Unlike the two prior systems, however, which use many sensors and a per-user compute engine, SPOT* greatly reduces costs by using the fewest possible sensors and a lightweight compute engine that can optionally even be located in the cloud. Moreover, SPOT* provides both cooling and heating using a speed-controlled desktop fan, rather than only controlling heating using a radiant heater. Finally, SPOT* is less intrusive in that it does not use a camera. We find that the per-user cost for SPOT* is about $185 compared to $1000 for SPOT. Moreover, in a preliminary deployment, SPOT* is able to improve user comfort by 78%.

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 their overall energy use [**?**, **?**]. Reducing the energy consumption of HVAC systems is, therefore, an important and necessary step in reducing their carbon footprint [**?**, **?**].

We present SPOT*, a low-cost, flexible, legacy-compatible *personal thermal comfort system*. SPOT* augments an existing central HVAC system by bridging the gap between the comfort provided by the central HVAC and individual worker preferences. By allowing the central HVAC to use higher temperature set points in summer and lower ones in winter, we believe that SPOT* can reduce energy usage. Our overall goal is to reduce building energy use yet ensure that workers are always comfortable.

This work extends prior work on the SPOT and SPOT+ personal thermal control systems [**?**, **?**]. Both systems use the Predicted Personal Vote (PPV) model (described below) to automatically adjust room heating to maintain a desired comfort level. SPOT is *reactive*, in that it only heats the room when the worker is actually present, and SPOT+ is *pro-active*, pre-heating the workspace before the arrival of the worker, or turn off heating in anticipation of the worker's departure. SPOT* differs from both systems in five significant ways. First, it controls *both* heating and cooling, so can be used both in winter and in summer. Second, it is about an order of magnitude *less expensive*. Third, it implements an innovative software architecture that allows *flexible tradeoffs* between cost, privacy, and data durability. Specifically, it allows most software components to execute either on the thermal controller, in the Internet cloud, or on the worker's smartphone. Fourth, the use of a fan instead of a radiant heater makes it possible to *rapidly react* to worker discomfort. Finally, it is far less intrusive than our two prior systems, because it does not use a camera.

SPOT* is composed from five components: sensors, actuators, control logic, data store, and user interface. Sensors detect worker comfort. Actuators turn on a fan or a fan as well as a heater to cool or heat the worker, respectively. Logic computes the PPV from sensed data and decides on actuation levels. The data store stores historical sensor data and system events. Finally, the user interface allows the user to correct for errors in the PPV model and to override control decisions. By flexible composition and location of these elements, SPOT* allows tradeoffs between cost, privacy, and data durability. For instance, if all the elements are in the workspace, the system is expensive, but private. If software elements are in the cloud, instead, the cost is reduced but privacy can be compromised.

Our work makes the following contributions:

- We have designed SPOT*, a low-cost, flexible personalized workspace thermal control system.
- We have built four SPOT* devices in two different configurations in a real testbed.
- We find that SPOT* improves average user comfort by 78% in our deployment.

The rest of this paper is laid out as follows. Section 2 presents a background on quantitative comfort modelling and an overview of SPOT systems. In Section 3 we explain our design goals and SPOT*'s architecture. Implementation details and different hardware and software components of SPOT* are presented in Section 4. Section 5 evaluates accuracy, cost, effectiveness, and efficiency in SPOT*. Finally, in Section 6, we discuss different configurations of SPOT*, and how it can be integrated with the central HVAC system to optimize energy consumption and conclude this paper.

2. BACKGROUND

We first review the Predicted Personal Vote (PPV) comfort metric introduced in [**?**], which is used to evaluate personal thermal comfort in indoor environments. We then describe SPOT and SPOT+ systems and explain their shortcomings which motivate SPOT*. Finally, we present the occupancy detection method that we use in this paper.

2.1 Predicted Personal Vote (PPV) Model

The PPV model is a generalization of the well-known Predicted Mean Vote (PMV) model [**?**, **?**]. The PMV model estimates an average worker's comfort level on the 7-point ASHRAE scale[1] using a function $f_{pmv}(\cdot)$:

$$pmv = f_{pmv}(\mathbf{x}) = f_{pmv}(t_a, \bar{t_r}, v_{ar}, p_a, M, I_{cl}) \qquad (1)$$

where pmv is the predicted mean vote and $\mathbf{x}$ denotes the following environmental and personal variables:

- 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

For reasons of space, we refer the reader to [**?**] for details of the function f_{pmv}.

To evaluate comfort in workspaces occupied by a *single* worker, SPOT* computes a Predicted Personal Vote (PPV) as an affine transform of pmv:

$$ppv = f_{ppv}(pmv) \qquad (2)$$

This function is learnt using least squares linear regression during a training phase, with the worker providing ground truth on comfort level.

2.2 SPOT and SPOT+

SPOT and SPOT+[**?**, **?**] , the predecessors to SPOT*, control comfort in a personal workspace. SPOT reactively controls a space heater to increase temperature gradually, and maintain comfort (i.e. PPV) in winters. SPOT+ takes a step further and uses occupancy patterns for prediction to implement pre-heating and pre-cooling. SPOT+ also takes advantage of an optimal control strategy for heating to minimize energy consumption.

[1]Cold (-3), Cool (-2), Slightly Cool (-1), Neutral (0), Slightly Warm (+1), Warm (+2), and Hot (+3).

Shortcomings. While these two systems prove to be effective and efficient, they have some important shortcomings that we address in SPOT*:

- **Cost.** The use of multiple fine-grained sensors in SPOT and SPOT+ makes them expensive. For instance, both systems use a Kinect camera and a PC to process the video feed from the Kinect. Although this high level of sensing makes them accurate, the $1000 per-office cost is a strong barrier to adoption.
- **Heating only.** In both systems, a space heater is used to increase temperature and bring comfort level above a threshold in winters. They are inadequate for situations where the user feels warm or hot, for example, during summers; or with mis-configured HVAC systems, even in winter.
- **Standalone deployment..** Because they process video images in real time, and the need for worker privacy, the SPOT and SPOT+ systems are not network-enabled. This prevents making smarter central HVAC control decisions based on inputs from multiple SPOT instances.
- **Intrusiveness.** A camera points to the user at all times in both systems, and a moving infrared sensor tracks movements of the user. Some SPOT/SPOT+ users found this both intrusive and unnerving.

We address these shortcomings in SPOT*.

2.3 Occupancy Detection

Happily, occupancy detection is, by now, a solved problem. Hailemariam et al [**?**], propose a way to infer occupancy based on motion sensor data with 97.9% accuracy [**?**]. They use the AMN23111 motion sensor [**?**], and record motion values with a 2Hz frequency. After two minutes, they compute the standard deviation of these values and declare the station occupied if it is above a threshold. We detect occupancy using a similar approach.

3. DESIGN

We now discuss the design goals, and the overall system architecture of the SPOT* system. For ease of writing, from now on, when we refer to SPOT, we mean both the SPOT and SPOT+ systems.

3.1 Design Goals

We have four main design goals, that we denote D1-D4.

D1: Reduce Cost. Our primary design goal is to reduce the price of SPOT*, compared to SPOT. The main reason our two prior systems were expensive was because they monitored workers using a Kinect camera that is directly connected to a powerful processing unit (i.e. a PC with Microsoft Windows) which accounts for about 50% of the overall system cost.

D2: Allow both Cooling and Heating. A secondary design goal is for SPOT* to provide comfort both in summer and in winter, unlike SPOT, which only addressed heating.

D3: Keep User Data Private. User occupancy data is inherently private information that should not be exposed to third parties. We would like to guarantee this despite allowing SPOT* to be on the Internet.

D4: Improve User Experience. SPOT's infrared camera, used for clothing level detection, closely tracks user movements. Some users found this unnerving, and therefore, one of our goals in building SPOT* is to not require observation of users with a web or infrared camera.

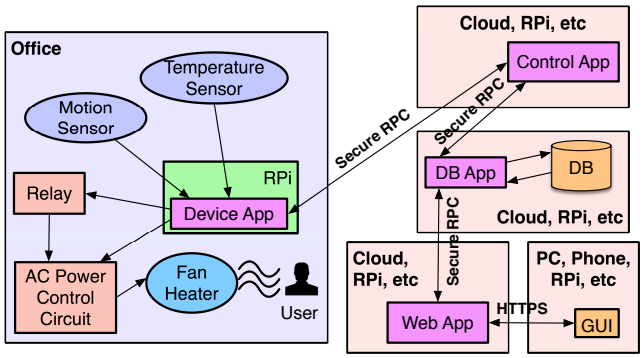

Figure 1: SPOT* has 5 main components. Actuation and sensing (left box), control application, data storage, web application, and graphical user interface. All software components communicate through RPC to allow easy deployment of different configurations.

3.2 System Architecture

Our overall system architecture is shown in Figure 1. Each of the software components shown on the right hand side can be executed on a per-office embedded compute platform, in the cloud, or even on a worker's smartphone. We now discuss how this architecture is motivated by our four design goals, D1-D4.

Goal **D1** results in a complete redesign of our hardware platform. We replace the expensive Kinect sensor and the per-office PC-based processing unit with a simple motion sensor and a Raspberry Pi B+ (RPi) [**?**]. This has the following repercussions.

First, without the Kinect, we are unable to automatically detect the worker's clothing level. Instead, we assume that the clothing insulation factor is $0.6clo$, and provide a simple web-based user interface for workers to tune their comfort setpoint to reflect a different clothing insulation factor. For instance, if they are wearing lighter clothes, they would tune their setpoint to a higher ASHRAE comfort level. The Kinect also was used for occupancy detection. In SPOT*, this is done using an inexpensive passive infrared motion sensor.

Second, even if we don't need a PC to process video data from the Kinect, we still need an in-office compute platform for sensing and actuation. We selected the RPi as the replacement. The RPi, which we have extended to allow actuation in addition to sensing (see Section 4), costs only about $40 compared to $500-PC used in SPOT. The cost of the in-office compute platform can be reduced even further if compute logic and storage occur primarily to the cloud. This allows further cost reductions by replacing the RPi with a tiny embedded processor (see Section 6). Moreover, moving processing and storage to the cloud makes the system much easier to deploy, albeit with reduced worker privacy.

We further reduced costs by reducing the number of sensors: SPOT* measures air temperature using a temperature sensor and assumes that this temperature is identical to the background radiant temperature. It also computes air velocity as a function of the fan speed (which is known), and given that SPOT* is deployed in an HVAC-controlled office space, it assumes that the humidity is controlled to 50%, and that the worker's metabolic rate is $1.2met$.

We address **D2** by replacing the space heater with a desktop fan/heater that we modify so that its speed can be controlled by the RPi (Section 4.1.3). Varying the fan speed allows us to respond to user comfort. This change in hardware has the added benefit that SPOT* can react nearly immediately to a change in user comfort.

SPOT* has five software components: actuation and sensing, control logic, data storage, web application, and user interface. Software components, except actuation and sensing, can execute either on the RPi, in the cloud, or on the user's smartphone, using python's RPyC library in SSL mode [**?**] for communications. This software architecture allows movement of software components to meet the user's privacy requirements, to meet **D3**. For example, in the most-privacy-preserving configuration, SPOT* operates standalone without transmitting data outside the office. In other configurations, occupancy data will securely leave the office and must be securely stored in the cloud.

Replacing the Kinect with a passive infrared occupancy sensor also partly addresses **D4**. In addition, as described in Section 4.5, a manual override option in the GUI allows the user to tune the system operation.

Thus, our system meets all four design goals.

We now sketch our system architecture (see Figure 1). Details of the implementation can be found in Section 4.

3.2.1 Actuation, Sensing, and Device App

This part of the system consists of a heating/cooling device, sensors, actuators, and a 'device app' executing on the RPi. The device app communicates with the control app (see Section 3.2.2) to send updates, and receive and execute commands. It requires minimal computation, and therefore can even be executed on an even lighter-weight platform, such as a mote. In fact, before we decided on using an RPi, we built a prototype system using a Zolertia Z1 mote [**?**]. However, the Z1 turned out to be much more expensive than the RPi!

3.2.2 Control Application

Based on occupancy and comfort, the control app decides to turn the fan/heater on or off, determines the fan's speed, and communicates with the device app to carry out the command. The control app can run on the RPi or the cloud. The

control app also communicates with the DB app to log all events–such as changes in temperature and occupancy and changes in device status–and to receive user feedback.

3.2.3 Data Storage

SPOT* keeps a log of measurements, events, and user feedback. The data can be stored on the RPi for better privacy, or can be moved to the cloud to provide better reliability and availability, and enable coordination with a centralized building HVAC system in the future (see Section 6). We use the data for:

- System administration and debugging,
- Making control decisions based on user input,
- Providing feedback to the user in the form of historical charts (see Figure 5).

The data storage component consists of a database, and a DB application that provides an interface for other software components. For better security, this component limits database access to certain functions, such as inserting and querying occupancy and temperature, querying users, and inserting device states.

3.2.4 Web Application

The web application both provides information to SPOT* users and administrators, and receives feedback from them. The web application can run on the RPi (to provide access only to the user), or can run in the cloud (to enable universal access). During the training phase, the web application collects votes from the user to determine coefficients for the PPV equation. Specifically, it computes the anticipated comfort from the PMV equation and collects user votes on comfort level. It then computes a linear regression to translate from the PMV value to the PPV value. It also provides a manual override for users to deal with exceptional conditions. In addition, it visualizes measurements and events for users to monitor their comfort and occupancy.

3.2.5 Graphical User Interface

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 SPOT* box. The user can access the GUI directly from the box (see Figure 6).

4. IMPLEMENTATION DETAILS

In this section, we discuss the details of our implementation. The source code for SPOT* is available online on GitHub[2].

4.1 Actuation and Sensing

Actuation and sensing consists of a desktop fan/heater (Figure 2) 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.

[2]`https://github.com/AlimoRabbani/SPOTstar`

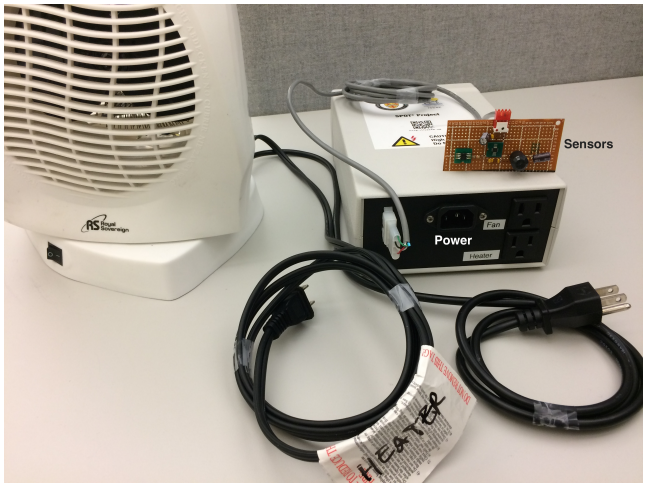

Figure 2: Hardware components of actuation and sensing in SPOT*. The fan/heater is shown on the right side, and the actuation box is on the left side. In this picture, the two independent power cords of the modified fan/heater are connected to the box. The modular design of the box, allows usage of different types of fans and heaters.

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, and its modular design allows connecting different types of heaters and fans, as discussed in Section 4.1.1. 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

Unlike SPOT, which uses a space heater for room heating, SPOT* uses a fan/heater to provide comfort in both winters and summers. We modified the Royal Sovereign HFN-20 [**?**], to control its heating coil and cooling fan independently. The two power cords in the modified version are connected to the power outlets on the actuation box (See Figure 2). The relays inside the actuation box determine the state of the fan and the heating coil, and the AC power control circuit sets the speed of the fan with a maximum air velocity of $2.1ms^{-1}$.

4.1.2 Sensing Box

We use only two sensors in SPOT*: a temperature sensor and a motion sensor. Data from these sensors is used to compute the PPV value as discussed in Section 3.2. 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 [**?**]. 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. The AMN22111 passive infrared human detection sensor is more sensitive to slight motions, and has a lower 2m detection range compared to other Panasonic AMN series sensors (which have ranges up to 10m) [**?**]. 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 generate one value close to 1000 and another close to 0. The closer these values are to 1000 and 0, the greater the intensity of movement. Therefore, over a 30-second window, a standard deviation near 0 means almost no movement while higher standard deviations correspond to more movements (See Figure 3). Note that unlike the Kinect system used in SPOT, which could instantly detect occupancy, there is a 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 [**?**] 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 Box

The actuation box physically controls the fan/heater using relays, a custom made AC power control circuit, and an RPi. It has two power outlets, one for the fan and one for the heater (Figure 4).

Relays. Two RPi GPIO output pins are connected to two electromechanical relays[3] to close and open the AC circuit of the fan and the heater independently. Upon receiving a command from the control app (see Section 4.1.4), 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 modified and built a standard AC-control circuit [**?**] 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. Figure 4 contains the circuit as populated on a manufactured PCB.

Raspberry Pi. The RPi runs Raspbian [**?**], and is connected to and powered through our custom-built circuit with a 40-pin ribbon cable. It executes commands and controls status lights on the box by toggling output signals on GPIO pins. In a networked setup, we use an Edimax EW-7811Un USB dongle [**?**] to connect the RPi to the building's wireless network.

4.1.4 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.

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

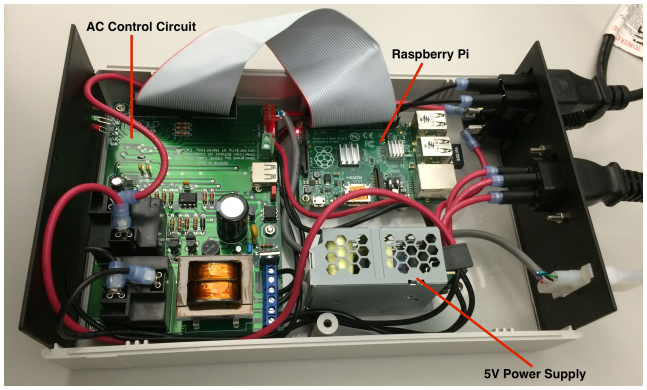

Figure 4: Inside of SPOT*'s actuation box. The AC control circuit on the left side communicates with the RPi to turn the fan/heater on or off, and set the fan's speed. The RPi, can connect to the building's network using a WiFi USB dongle, and a 5V 3A stable power supply keeps the box running.

The device app collects motion data twice every second, computes the standard deviation over a 30-second period, 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 [**?**] data every two minutes [**?**]. However, we found that the more sensitive AMN22111 motion sensor [**?**] allows us to lower the occupancy detection interval from 2 minutes to 30 seconds. To reduce the amount of inter-process and network communications, the device app collects motion data twice a second, computes its standard deviation, and sends only this value to the control app every 30 seconds. It also reads and transmits 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 spins with its maximum speed while the heating coil is powered.

4.2 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 logged on the storage. It also makes a control decision based on occupancy and PPV, and invokes the appropriate procedures on the device app.

4.2.1 Occupancy Inference

As discussed in Section 4.1.4, the control app receives a standard deviation value O that represents movement intensity in every 30-second period. We determine if the station is occupied if this value exceeds a threshold $T_o = 17.25$ [**?**]. 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:

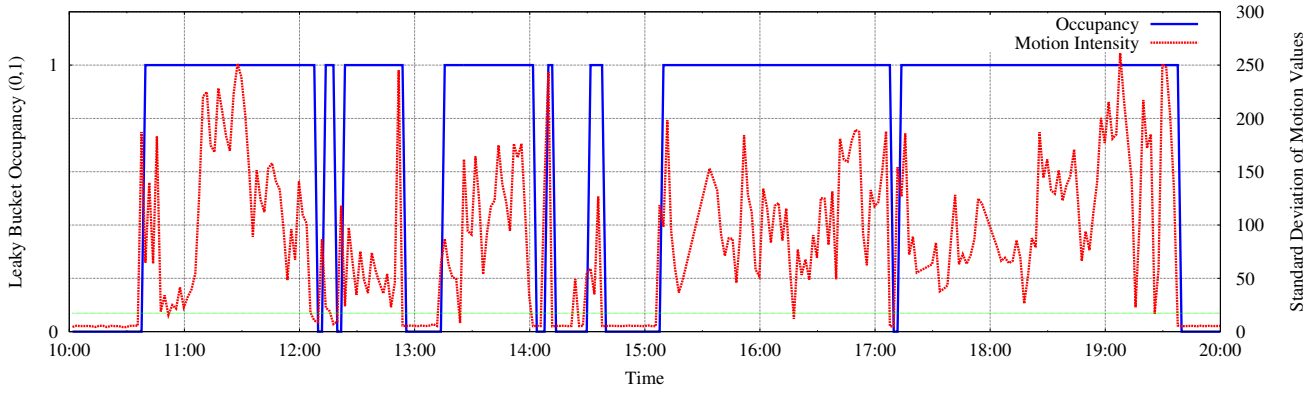

Figure 3: Occupancy inference, based on the standard deviation of motion values (i.e. motion intensity) during 30-second time windows over a 10-hour period. A threshold of $T_o = 17.25$ determines occupancy in each time interval. The results show 96% accuracy in detecting occupancy in our deployment.

- 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).

4.2.2 Decision Making

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, the PPV is re-calculated for each increment of 0.1 ms^{-1} in the fan speed, starting from zero, until the PPV (at that air speed) is within the comfort range or we reach the maximum possible fan velocity of $2.1\frac{m}{s}$. Thus, the decision uses the PPV equation to derive the necessary fan speed to achieve a desired personal comfort level.

Algorithm 1 Control app's MakeDecision procedure

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

4.3 Data Store and DB Application

We use MongoDB [**?**] to store the data, and 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 to provide user preferences to it. It also communicates with the web app to store user preferences and provide data to the web app for visualization. Because of SPOT*'s flexible architecture, we can run the DB app and the MongoDB storage locally on the RPi, or in the cloud.

4.4 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 webserver[4]. To ensure privacy and security, we use HTTPS [**?**] and require users to login with their credentials.

4.4.1 Training Period

Recall that SPOT* requires training to estimate the affine translation between PMV and PPV values. Users can start training periods at will. 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.

[4] `http://blizzard.cs.uwaterloo.ca/spotstar/`

Figure 5: 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.

4.4.2 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:

$$PPV = a * PMV + b - B_o \quad (3)$$

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

4.5 Graphical User Interface

Internet 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 SPOT* 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 [**?**]. 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 6).

Figure 6: For isolated installations of SPOT*, 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.

5. EVALUATION

We built four SPOT* devices and deployed them in offices at the University of Waterloo. At the time of writing this paper, we have data from this deployment for about 25 days, with about a 12-day break in the middle.

Here, we compare the cost of SPOT* with SPOT, measure its accuracy in detecting occupancy, evaluate users' comfort when using SPOT* with both subjective and objective measures, and explore its energy consumption.

5.1 Cost

Table 1 shows the hardware components used in SPOT* with approximate prices we paid, and estimated cost with mass production. Note that the per-user cost of software and cloud servers is negligible for large deployments. Therefore, we do not include it in Table 1. Even for a single prototype, the overall system cost is about 82% lower than SPOT, which costs approximately $1000 per user. In addition, if mass-produced, we estimate that it would about one order of magnitude cheaper than SPOT.

5.2 Occupancy Detection

To achieve goals D1 and D4, we replaced the Kinect with a motion sensor and modified the approach proposed in [**?**] to detect occupancy. Assuming perfect placement, the Kinect was 100% accurate in detecting occupancy. By eliminating the Kinect, we reduce the cost, but the accuracy also declines. To estimate the accuracy of occupancy detection, we measured occupancy using both a video camera (with human tagging of occupied periods) and the passive infrared sensor used in SPOT*. Over a 3-day period, SPOT* had a 96% accuracy, which is excellent. We note that, in addition, our occupancy detection approach causes a 30s delay in detecting occupancy. Figure 3 shows how standard deviations of motion data translate into occupancy during a 10-hour period.

Item	Prototype Price	Est. Volume Price
Raspberry Pi	$40	$15
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 1: Cost breakdown of hardware elements used in SPOT*. The table shows our approximate prototype cost, and the estimated mass-production price for each element.

5.3 Comfort

We measured the effectiveness of SPOT* in maintaining user comfort in two ways. First, we compute the average absolute discomfort[**?**] in the presence and absence of the SPOT* during the 25-day 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

The **average absolute discomfort** objectively measures how uncomfortable a user feels. Let $d(t)$ be the absolute discomfort at time t defined as

$$d(t) = max(|ppv(t)| - T_c, 0) \quad (4)$$

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 SPOT+, we set T_c to 0.5 in our evaluations. Then, $\hat{d}$ or average absolute discomfort, is defined to be the average of $d(t)$ only at times when the user is present. Specifically, let $m(t) = 0$ when the workspace is not occupied and be equal to 1 when occupancy is detected ($L = C$ in the leaky bucket). Then,

$$\hat{d} = \frac{\Sigma_{t=0}^{T} d(t)m(t)}{\Sigma_{t=0}^{T} m(t)} \quad (5)$$

To measure the performance of SPOT*, we calculate the average absolute discomfort of users with and without SPOT*. Instead of measuring this value after turning off control actions, we observe that SPOT* performs no control actions when the workspace is unoccupied. So, the average absolute discomfort when the workspace is unoccupied is identical to its expected value in the *absence* of SPOT*. Thus, to measure average discomfort in absence of SPOT*, we simply define:

$$\dot{d} = \frac{\Sigma_{t=0}^{T} d(t)m'(t)}{\Sigma_{t=0}^{T} m'(t)} \quad (6)$$

where $m'(t)$ is 0 when the user is present and 1 when the the workspace is not occupied (i.e., $m' = 1 - m$).

In our experiments, The average $\hat{d}$ for all four users is 0.16 compared to 0.73 for $\dot{d}$. Therefore, SPOT* improves user comfort by 78% in this admittedly limited trial. Figure 7 shows a comparison between average discomfort when SPOT* is in use, and when it is not in use, for each individual user.

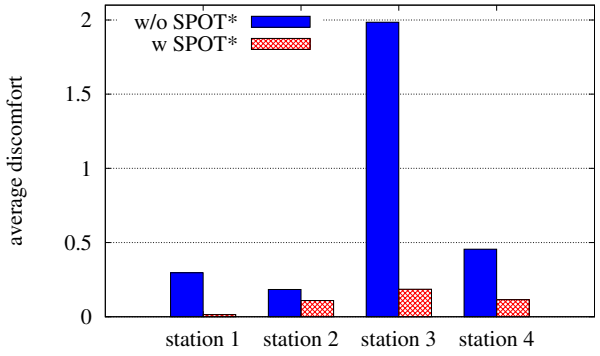

Figure 7: Average discomfort of users when SPOT* is being used, and when it is not being used. For each user, the average discomfort decreases significantly when SPOT* is maintaining comfort.

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:

$$E_s = \frac{\text{number of slider events}}{\text{total occupied hours}} \quad (7)$$

The experiments show that the average E_s for all users is 0.38, about two events per day. We expect this value to decrease over time as users get used to the system. Figure 8 shows E_s for the four deployed SPOT* systems.

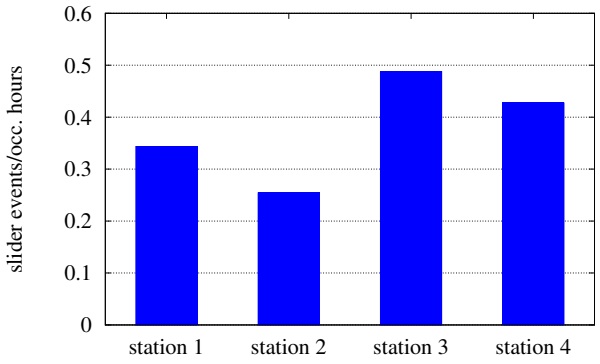

Figure 8: The number of offset slider events per occupied hour for each station. We use this indicator (E_s) as the subjective measure for user discomfort. The average of 0.38 for E_s indicates approximately two events per day.

5.4 Energy Consumption

SPOT* 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 1300 watts when turned on.

We estimate the total energy consumption of each deployed SPOT* instance by summing up the energy consumption of the three sources above. We find that the average energy consumption of SPOT* during the 25-day period is approximately 8 kWh. Figure 9 shows total energy consumptions for each deployed SPOT* device. Unfortunately, we are unable to compare this usage with that of a central HVAC system, but suspect that it is at least an order of magnitude lower.

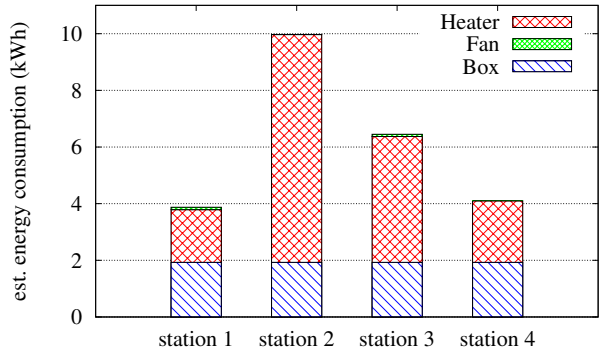

Figure 9: Total estimated energy consumption for each SPOT* device during 25 days in kWh. All devices, consume the same amount of energy to keep running. A significant portion of consumption belongs to the heating coil, and a small amount is consumed to power the fan.

6. DISCUSSION AND CONCLUSION

Motivated by the shortcomings of the SPOT and SPOT+ systems, we have presented the design and evaluation of the SPOT* system. SPOT* meets all of its design goals and, because of its much lower cost, will soon be widely deployed: we are currently preparing 70 systems for deployment in an entire wing of a campus building.

We gained several insights from our work. First, the primary determinant of user comfort is workspace temperature. With SPOT, we were extreme in sensing every component of the PMV equation. In SPOT*, we achieved nearly the same level of comfort as SPOT, but with only temperature measurements and judicious choice of parameters for the PMV equation, using manual overrides to deal with errors in this choice. This has turned out to be a good design choice. Our test users were especially happy with the fact that, unlike SPOT, SPOT* responds nearly immediately to manual control.

The design choice of using flexible, re-locatable software components has also proven to be a good one. With nearly no effort, we can configure a system to be standalone, which makes privacy-sensitive users happy, or be networked, which opens up the possibility of coordinating SPOT* actions with that of a central HVAC, something we would like to pursue in future work. We also hope that heater/fan manufacturers will, some day, build in a mote-like embedded compute platform into their devices, allowing us to deploy SPOT* 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 a straightforward one. We initially considered the Arduino, a smartphone running Android, and a Zolertia Z1 mote. Indeed, our first deployment was using the Z1, which has a MSP430 controller, no 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.

Finally, we would be remiss if we were not to mention a major limitation of our work. Although we were motivated by the need to reduce building energy use, we were unable to persuade our building managers to actually let us turn the heating setpoint to a lower level, since we did not have enough SPOT* devices to manage an entire building zone. Thus, we are unable to conclude that SPOT* will, indeed, reduce building energy consumption. However, as mentioned above, we are in the process of putting together a deployment, and hope to report on the results from this work in the future.

7. ACKNOWLEDGEMENTS

This work would not have been possible without a strong team to back us up. Costin Ograda-Bratu helped with building boards, hacking heaters, and a multitude of hardware issues. Milad Khaki was responsible for hardware design for both the actuation and sensing boards. The CSCF and IST staff at the University of Waterloo allowed us access to campus wireless networks for our deployments. Our brave volunteers who tested SPOT* were Peter Forsyth and Jim Summers. Our thanks to them all.