Hot or Not: Leveraging Mobile Devices for Ubiquitous Temperature Sensing

Joseph Breda, Amee Trivedi, Chulabhaya Wijesundara, Phuthipong Bovornkeeratiroj, David Irwin, Prashant Shenoy, and Jay Taneja
{jbreda,cwijesundara,jtaneja}@umass.edu,{amee,phuthipong,shenoy}@cs.umass.edu,irwin@ecs.umass.edu
University of Massachusetts, Amherst

ABSTRACT

This paper introduces a novel technique to measure indoor ambient air temperature using the battery temperature sensor found on typical smartphones. We develop physics-based models to predict ambient air temperature that consider the many warming and cooling scenarios faced by phones and account for the excess heat generated by smartphone components such as the CPU, screen, network, and charging hardware. To accommodate never-before-seen devices, we also develop a domain adaptation technique that leverages previously derived models, substantially reducing the overhead of learning accurate models for a new phone. We evaluate our models for a range of devices, operating scenarios, ambient temperatures, and phone cases, with mean errors generally less than 1.5% of ambient temperature. We also present a case study to demonstrate the utility of our approach for spatial and temporal monitoring of ambient temperature variations in an office building; while indoor conditions vary by as much as 13°F, mean error in measurement by our models is 1.4%.

CCS CONCEPTS

• **Human-centered computing** → *Ubiquitous and mobile computing*.

KEYWORDS

Smartphone, Temperature sensing, Temperature control, Secondary sensing

1 INTRODUCTION

Smartphones have become ubiquitous in recent years, with nearly 3 billion smartphone users worldwide in 2019. Modern smartphones are rich in sensing capabilities and equipped with a plethora of sensors such as GPS, accelerometers, gyroscopes, microphones, and cameras, among others. This has enabled mobile sensing to emerge as a viable sensing paradigm, with smartphones being used to monitor our environment in new and interesting ways. In recent years, researchers have used smartphone sensing and crowdsourcing of data to monitor water quality [8], state of the electric grid [3], health parameters such as sleep [16], and social context [19]. Smartphones can also be employed to sense the ambient environment inside buildings, which is a target application of our work.

Buildings account for nearly 70% of the total electricity and 40% of the total energy consumption in most countries [2], with heating and cooling (HVAC) accounting for nearly half of a typical building's energy footprint. A building's HVAC systems are designed to provide optimal comfort to its occupants while being energy efficient. In practice, however, HVAC systems are far from perfect and can be sub-optimal in terms of providing user comfort, energy efficiency, or both. Office and commercial buildings, in particular, use zone-based control for heating and cooling, where each zone, which comprises part or all of a building floor, can be independently controlled. Zone-based HVAC systems are often controlled using a small number of thermostats that are deployed over a larger area, often resulting in subtle temperature differences across a zone, which in turn impacts user comfort. In some cases, a single thermostat may control a large area such as a hallway, atrium, or a suite of rooms. Residential smart thermostats such as the Nest and Ecobee have attempted to address this issue by coupling thermostats with additional temperature sensors deployed in various rooms to track these temperature differences [1], which enables them to maintain a more uniform temperature across a zone. In commercial buildings, deployment of such additional temperature sensors over larger areas are expensive, labor intensive to install, produce massive amounts of e-waste as new hardware is rolled out, and may not improve HVAC operation. These reasons motivate a software defined alternative that leverages the ubiquity of mobile devices of building occupants as proxy sensors providing the same metrics with no installation or additional hardware (Also applicable in developing regions where installation is unfeasible).

Our paper presents such an alternative approach where we utilize smartphone sensing to estimate ambient temperature at different locations inside a building. Our hypothesis is that the on-board thermistor-based battery temperature sensor, designed for monitoring battery temperature and maintaining battery safety, can serve as a proxy sensor for the phone's surrounding environment. Our approach comprises modeling the difference between the phone's battery temperature and ambient air temperature as a function of screen activity, battery, CPU, and network processor state as well as a layer of activity recognition. Our approach, in effect, turns every smartphone, regardless of its ongoing activity, into a digital thermometer for ambient temperature sensing, which can then be used to monitor, or crowdsource, temperatures at different locations inside a building providing more granular data for building control systems.

BuildSys '19, November 13–14, 2019, New York, NY, USA

ACM ISBN 978-1-4503-7005-9/19/11…$15.00
https://doi.org/10.1145/3360322.3360856

In designing our models and our approach, our paper makes the following contributions. First, we develop a modeling approach that combines physics-based models, using Newton's Law of Cooling, with statistical state identification methods to infer ambient temperature. Second, since our models are specific to each phone type and deriving them from scratch is laborious and time-consuming, we develop a model generalization approach that leverages prior models learned for other phones with a small amount of sampling to quickly derive a custom model for a new phone. Our experiments validate our models using several phones with different hardware characteristics and Android OS versions and show that: (i) our models are able to predict ambient air temperature across a range of phone states and phone models with mean errors of 2-5%, (ii) we are able to learn prediction models for a new phone using only a fewsamples in conjunction with prior models, significantly reducing the model derivation overhead for new phones, and (iii) our models are effective even when the phones are equipped with various types of phone cases. We also conduct a case study throughout one floor of an academic research building to show how our models can track ambient temperature differences over time and spatial differences in temperature across a floor.

2 BACKGROUND AND RELATED WORK

In this section, we present background and related work on mobile sensing as well as building heating and cooling, and discuss challenges to ambient temperature sensing.

Mobile Sensing: As sensor-rich smartphones have become ubiquitous, mobile sensing has emerged as a feasible paradigm for gathering environmental data. Our work aims to extend the capabilities of a commodity smartphone [1] to measure a new modality: indoor ambient air temperature. To our knowledge, no current phones possess an onboard ambient air temperature sensor.[2] However, nearly all phones possess a battery temperature sensor. Our hypothesis is that the battery temperature sensor can be used to indirectly sense the ambient air temperature.

The feasibility of using the battery temperature sensor for ambient temperature sensing has previously been explored [5, 11]. These efforts accurately estimated the daily average outdoor temperature across entire cities using crowd-sourced battery temperature data retrieved by a distributed network of mobile phones gathering measurements. This approach was designed for outdoor temperature sensing and required data for a large number of phones to obtain a coarse non-localized temperature estimate. There was no correction for heat-generating activities on the phone itself. In our case, we wish to use a single phone as a "thermometer" and use its battery temperature sensor to predict the ambient air temperature inside a building without relying on a large number of phones. Of course, our system can leverage multiple phones, each of which acts as an independent thermometer, to map out the spatial and temporal variations in ambient air temperature across a building floor.

Building Sensing: HVAC systems in buildings are responsible for maintaining user comfort and meeting safety standards, while also being as energy efficient as possible. Commercial and office buildings typically use a system of zones and controllable air vents to deliver an appropriate amount of heating or cooling to each portion of the building floor. Although a zone may be configured for a specific temperature set-point, differences in airflow, heat gain or loss through windows, and other thermodynamic factors cause subtle but noticeable differences in temperature across a floor or even across a part of a single large room. These temperature differences can impact user comfort since parts of the room or floor can be warmer (or cooler) than others. As an example, Figure 1 shows a heatmap of a single floor of a building with temperatures varying by as much as 13°F. If the building floor is highly instrumented with temperature sensors, these spatial differences can be detected and corrected using individually controllable HVAC vents, thereby improving overall user comfort. However, buildings, especially older ones, rarely have this degree of instrumentation and controlThis paper explores the use of smartphone-based mobile sensing to fill this gap – by sensing ambient air temperature at different locations on a building floor (*i.e.*, across a single room or among multiple rooms) to measure spatial and temporal variations.

Figure 1: Variability of temperature across one floor of an office building. Ambient air temperature varies by 13°F.

Many researchers have worked to improve building HVAC performance, including efforts to improve control by better managing zone temperatures [4, 14], providing personalized thermostat recommendations [13], designing entire personalized HVAC control systems at the individual scale [12], or even assigning people to desks by comfort preferences [10]. Others have focused on alternative measurement methodologies, including using WiFi [15] and video [7]. Our work aims to be another channel of granular and accurate measurement to allow thermostat control to improve comfort of building occupants, enabling better coverage in more common zone-based thermostat/HVAC control loops, or providing the basis for future personalized HVAC alternatives.

Challenges in ambient temperature sensing: The use of smartphones for ambient air sensing involves many challenges. The first challenge is the lack of an onboard sensor to directly sense ambient air temperature, which needs to be inferred through data from other sensors. Here, we work from the principle that the battery temperature at any instant is a function of the ambient temperature and the heat dissipated by the battery and other hardware due to ongoing phone activities . As a result, the battery temperature will always be higher than the ambient air temperature, and we need to develop techniques that infer the ambient air temperature by accounting for the effects of the phone's activities on the battery temperature.

A second challenge is to determine the state and context of the phone to correctly predict ambient air temperature. Our work assumes that the software and hardware state (what the phone

[1] We also note that our approach easily extends to other commodity mobile devices such as smart watches and tablets.

[2] The 2013 Samsung Galaxy S4 and Motorola Moto X possessed such a sensor, but in both cases, the ambient temperature sensor was removed in later model iterations due to poor accuracy and inefficient use of on-board real estate.

is doing) and context (the environment and surroundings of the phone) has a direct impact on its heat dissipation—for example, screen activity has a different temperature impact than CPU load, and likewise resting on a table or in a pocket alters these impacts as well. Determining context and state of the phone is a precursor for accurate ambient temperature inference.

A final challenge is that phones differ significantly across vendors and models in terms of their battery capacities, screen size, CPU capabilities, and OS versions. As a result, an ambient temperature sensing model that is designed for a particular phone will not directly carry over to a different phone (even from the same vendor) due to differences in the screen, CPU, and battery characteristics that change the heat dissipation behavior of the phone. At the same time, deriving a custom model for each phone is time-consuming and onerous, requiring that our approach be generalizable or easily boot-strapped to a new phone using models learned from current phones.

Problem Statement: Given this background and the above challenges, our paper seeks to develop (i) models for inferring the *indoor* ambient air temperature inside a building given the phone's battery temperature, (ii) models to determine the phone's physical context and state to enable accurate temperature estimates for the current context and state, and (iii) efficient and automated methods to learn ("generalize") these models for a new phone.

3 AMBIENT TEMPERATURE SENSING USING SMARTPHONES

In this section, we present the key intuition behind our approach, our physics-based models, and our data-driven methods for inferring phone context and state.

3.1 Physics-based Modeling

As noted earlier, every phone has a sensor for continuously monitoring battery temperature. The primary intuition behind our approach is that the battery temperature is a function of ambient temperature and the heat dissipated via the battery and phone components due to user- and OS-driven phone activity.

When the phone is idle for a sustained period, it follows that the battery temperature will approach the ambient temperature. However, when the phone is in active use, the screen, CPU, network, or charging activities will all dissipate heat causing the battery temperature to rise far beyond the ambient air temperature. To determine the ambient air temperature in this case, we must somehow determine (or model) the increases in temperature due to these activities and "subtract" their effects from the measured battery temperature. The residual temperature is then an estimate of the ambient air temperature. Similarly, when the phone becomes idle after a period of sustained use, the phone and battery enter a cooling period. In this phase – typically lasting anywhere from 30 to 55 minutes depending on the device and ambient temperature – the battery temperature converges towards the ambient temperature. Once again, we must model the residual heat effects from past activities during this cooling phase and subtract these effects on the measured battery temperature to obtain the residual (*i.e.*, ambient) temperature.

We model these thermodynamic effects using Newton's Law of Cooling [6], which states that the rate of heat loss is directly proportional to the difference in the temperatures between an object and its surroundings. Thus, the higher the battery temperature, the greater the rate of heat loss from the phone to its surroundings. Of course, when the phone is in active use, two effects come into play. The phone activity consumes energy and results in heat dissipation that causes a rise in temperature. At the same time, as the difference between onboard and ambient temperature grows, this heat is dissipated at an ever increasing rate, resulting in a cooling effect. Initially, the heating effect dominates the cooling effect, causing a net increase in temperature. Since the rate of cooling rises with increasing temperature as per Newton's law, the rate of heat loss rises until an equilibrium point where the heat generated by the phone is equal to the heat loss to the surroundings, and the battery temperature stabilizes at this point.

When the user stops using their phone, the phone enters a cooling phase, where the heat loss dominates, causing the battery temperature to drop; the rate of drop diminishes as the battery temperature approaches the ambient temperature as per Newton's law.

We use an exponential model to model the warming phase when the phone is in active use:

$$T_B = T_A + C \cdot e^{kt} + b \tag{1}$$

where T_B denotes the battery temperature, T_A denotes the ambient air temperature, C and k are parameters that capture the effect of the material of the device (and possible phone case) on the rate of warming, t is the time elapsed since the phone became active from idle state, and b is a parameter that captures the steady state behavior of the device.

Figure 2 depicts the behavior of the battery temperature of a Google Pixel phone using active screen use and our exponential model based on Newton's Law of Cooling to capture this behavior. Interestingly, the same exponential model, but with inverse parameters, can capture the behavior during the cooling phase, as depicted by the following equation:

$$T_B = T_A + C \cdot e^{-kt} + b \tag{2}$$

Again, the decreasing portion of the curve in Figure 2 depicts this behavior.

The idle as well as the steady equilibrium behavior is modeled using a linear function.

$$T_B = T_A + C \cdot t + b \tag{3}$$

While the C parameter is useful for capturing gradual drift in T_B during transition periods between idle and cooling or warming and steady, long periods of idle state are better modeled using $C = 0$ yielding:

$$T_B = T_A + b \tag{4}$$

Although direct modeling of power consumption on a phone is known to be challenging [9], our physics-based approach models its temperature impact on the battery, which we hypothesize is easier to model.

3.2 Modeling Phone States

While the above functions derived from Newton's Law of Cooling describe the overall thermodynamic behavior of the phone, the

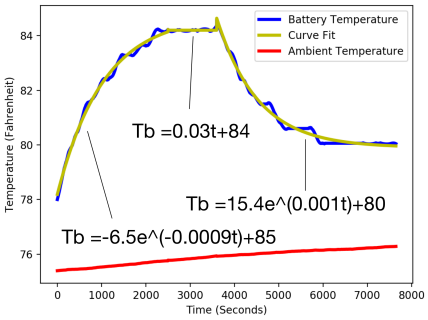
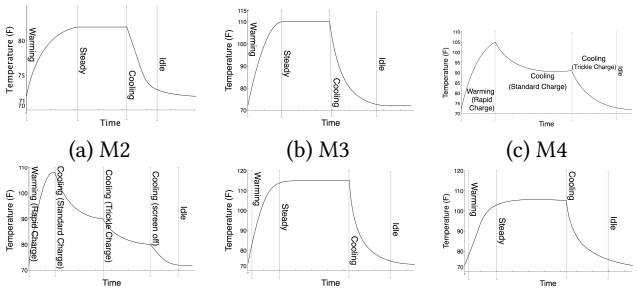


Figure 2: Exponential warming, linear steady state, and exponential cooling fit to battery temperature as it rises and falls in our M2 (screen use) experiment.

actual behavior at a particular instant depends on the phone state, duration in each state, and frequency of state transitions. Phone state is defined as the different types, and combinations, of activities that draw varying amounts of power and therefore result in different rates of warming and cooling and different peak steady-state temperatures. Each such behavior can be modeled using a different set of C, k and b parameters.

We model seven distinct phone states to capture different phone behavior under different usage scenarios. These seven states do not characterize all the possible computational states of a device, but they do capture the four primary activities that largely govern the phone's energy usage and heat dissipation from the battery: (i) screen usage, (ii) CPU usage, (iii) network usage, and (iv) battery charging. As a general rule of thumb when combining multiple phone activities, the heat dissipation from CPU activities is the greatest, followed by battery charging, network activity, and screen usage in that order – as observed from our experiments with different phone models. We model the effects of these activities individually and in various combinations as follows.



(a) M2 (b) M3 (c) M4

(d) M5 (e) M6 (f) M7

Figure 3: Modeling the impact of various combinations of screen, CPU, network and charging activities on the phone battery temperature.

State M1: Our first model captures the phone idle state or an equilibrium steady state where the heat generation equals heat loss, yielding a fixed difference between battery and ambient temperature. We use Equation 3, or its simpler variant Equation 4, to model state M1.

State M2: Next, we model the phone behavior during active screen usage when the accompanying CPU usage is low. This phone exhibits three distinct phases (referred to as sub-states) with an initial warming phase when the screen is in active use, an equilibrium steady state with continued screen activity, and a final cooling phase when the screen activity stops and the battery temperature starts cooling down. Figure 3(a) depicts this behavior modeled by Equations 1, 4, and 2, respectively.

State M3: This state models phone behavior exhibited during screen usage accompanied by high CPU usage, and models the combined effects. Like in state M2, the phone exhibits three distinct phases – warming, steady, and cooling – only with a higher peak temperature and steeper exponential curves in warming and cooling. Figure 3(b) depicts this behavior.

State M4: This state models battery charging while idle. Charging dissipates heat and thus battery temperature changes characteristically throughout charging. Charging from a low battery level includes an initial charge phase where the battery is charged rapidly to a certain charge level, followed by a steady charge to a high charge level, and a final trickle charge until the battery is full. Consequently, as shown in Figure 3(c), the charging behavior includes a high heat warming phase during rapid charging to reflect the high current, a first cooling phase where the battery is charged with a lower steady current causing it to dissipate less heat (yielding a *slow* cooldown), and a second cooling phase during the trickle charge phase where temperature converges to idle temperature.

State M5: This state models active screen usage accompanied by low CPU usage while the phone is charging. The heat dissipation of the phone due to charging is greater than that due to screen usage. Consequently, the overall phone behavior in this state resembles state M4, rather than state M2, but with a greater rate of temperature increase during warmup due to added screen usage. Similarly, the first cooling phase shows a slower decay than that in state M4 as screen activity raises the equilibrium temperature to which this cooling phase converges. Figure 3(d) depicts this behavior.

State M6: This state models high CPU and screen usage when the phone is charging and thus captures the heat dissipation from all three activities. Overall, heat generated from high CPU usage is greater than that from battery charging, which is itself greater than that from screen usage. As shown in Figure 3(e), the warmup and cooldown phases are most rapid due to the highest peak temperature reached of any state.

State M7: This state models the combined effects of high screen and network usage (and therefore associated CPU usage). The combined effects of screen and network activity makes the phone behave similar to state M3, especially since the energy consumption of the screen is greater than that of the network chips on the phone, causing the former effect to dominate – see Figure 3(f).

Note that we have also modeled, but omitted from here, scenarios with active CPU and/or network use with an inactive screen. In these cases, since the screen is inactive, signifying no foreground use, CPU and network usage are from background activities, which are subject to aggressive power management by the OS in order to minimize their battery power draw.

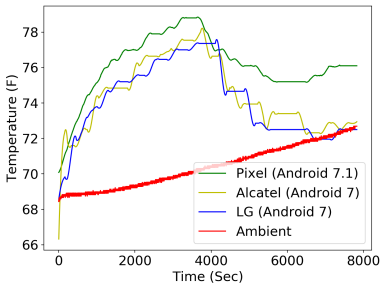


Figure 4: Battery temperature behavior varies by phone type as illustrated by the behavior of three phones for state M2 (screen use).

3.3 Model Derivation

Our seven state models capture the phone's battery temperature behavior across a wide range of different scenarios. We have developed an automated method to derive these state models for any Android phone empirically. Our approach to automating model derivation involves a mobile app that runs on the phone and can emulate the various usage scenarios (*e.g.*, screen on or off, different CPU loads, *etc.*) to put the phone in any desired state (*e.g.*, M1, M2, *etc.*). To derive a specific model, the app emulates the usage scenario corresponding to that model and records a time-series of battery temperature as well as several other OS metrics. It also gathers ground truth ambient temperature using a HOBO temperature logger (model MX2301) that enables ambient temperature recording on the phone via Bluetooth. The result is a labeled time-series dataset of battery temperature, ambient temperature, and OS metric at each time for each scenario.

Once empirical time-series data for a state have been gathered, we subtract ground truth ambient air temperature from battery temperature, leaving the temperature difference, ΔT. Our system fits a curve to this ΔT time-series using Equations 1–4 for each sub-state to derive model parameters C, k and b for each. Python's Scipy curve fit is used to fit appropriate functions on the raw data after smoothing it using an off-the-shelf Savitzky-Golay filter.

Since battery temperature behavior under states M1-M7 varies by phone type due to hardware and software differences, as depicted in Figure 4, our automated model derivation mobile app will need to derive a separate set of M1-M7 model parameters for each phone. This is a time-consuming process and takes over 24 hours to gather a full data set for M1-M7 states. In Section 4, we describe domain adaptation techniques to speed-up model derivation by leveraging past models that have been learned for other phones to bootstrap model derivation and significantly reduce these overheads.

3.4 Context and State Identification

This section presents models for determining the phone's physical context as well as its current state and sub-state.

Context Identification: The phone's physical context indicates whether the phone is outdoors or indoors, stationary or in motion, in a pocket, in a hand, on a table, in a bag, *etc.*, as well as how long it has been in the current context. Context and phone activity recognition is well-studied in the literature [17, 18]. Many prior approaches are based on black-box activity recognition models using decision trees or k-Nearest Neighbors (kNN) that leverage onboard sensors such as gyroscopes and accelerometers as well as other OS measurements for identifying activities. While any of these past approaches can be adapted for our work, we use a simple approach inspired by these prior efforts that uses a random forest (decision tree) classifier to determine whether the phone is stationary or in motion, and whether it is on a table, in a pocket, or in a user's bag/backpack. Using only these 3 classes, trained on data from each of these three contexts, we were able to design a classifier with 91.5% accuracy in 5-fold cross validation using only accelerometer, gyroscope, proximity sensor, and light sensor data as features.

To distinguish between indoor and outdoor contexts, we determine the strength of WiFi and GPS signals and predict that the phone is outdoors if there is a strong GPS signal or when the GPS signal is stronger than WiFi, and indoors when the WiFi signal is stronger than GPS. Finally, our context detection also determines whether the phone is charging by sensing the value and direction of battery current.

The ideal conditions for ambient temperature prediction occur when the phone is stationary and directly exposed to ambient air (*e.g.*, resting on a table). Other conditions such as stationary in a bag or hand are more challenging but feasible. Our approach is currently conservative and does not make predictions when the phone is in motion (*e.g.*, when the user is walking) since we require a settling time of a few minutes at a location, at minimum, before making temperature predictions. [3]

State and Sub-state identification Once the phone's context has been determined, we must then determine its current state and sub-state – doing so enables our approach to choose the correct model (*e.g*, M2-cooling) and the corresponding model parameters to predict ambient temperature. We model the state and sub-state identification problem as a decision tree with nodes of the tree testing the values of features determining the state and sub-state and the leaf of the tree as the set of parameters (C, k, b) to predict the temperature using Newton's Law of Cooling for the identified state and sub-state.

State identification: As noted earlier, battery temperature is impacted by four primary phone activities: screen use, CPU usage, network usage, and battery charging. The combination of these four attributes results in various states; to identify the state of a phone at a given instant in time we need a state identification mechanism. Our state identification approach uses a rule-based system that considers the CPU utilization, network utilization, screen state, and battery charging as Boolean features, with CPU/network utilization as "high" if resource utilization exceeds 80% and "low" otherwise, screen state as either "on" or "off", and battery charging state as "true" if the phone is being charged and "false" otherwise. Models M1-M7 consider 7 different combinations of these four Boolean features; to identify the current state, we use a rule-based approach that checks the current value of these four Boolean features and maps the current combination to the corresponding state. It should be noted that while new privacy restrictions were added in Android

[3]Temperature sensors, such as the ones on the phone, require a settling period to "acclimate" to a new location before they can accurately measure temperature.

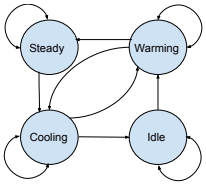


Figure 5: State transition diagram among model sub-states.

8 that prevent applications from accessing CPU load directly, CPU load is naturally aggregated across both time and cores and is a function of metrics readily available to unrooted applications such as battery current.

Sub-state identification: Once we have identified the phone state, the next step is sub-state identification. Each state of the phone has sub-states associated with it, such as warming, cooling, steady, and idle. Some states such as M4 and M5 have multiple cooling sub-states and lack a steady state as seen in Figure 3.

Our approach first samples the battery temperature over a time window; since the raw data can be noisy and is non-differentiable, we first smooth this data using a moving average smoothing filter and calculate the smooth signal's gradient. A new time-series state signal is formed by labeling data points with a positive gradient as in warming state and points with a negative gradient as in cooling state. If the gradient is within some margin of zero, we label this state as steady or idle and differentiate between the two based on the previous state. As seen in Figure 5, steady state can only be reached after warming and idle can only be reached after cooling. We then smooth and bin this state signal to sporadic errors.

3.5 Ambient Temperature Estimation: Putting it all together

Having discussed modeling of phone battery temperature as well as context, state, and sub-state identification, we now discuss how to sequence these mechanisms to predict ambient air temperature.

Ambient air temperature prediction involves three steps. First, our approach invokes context identification to determine whether the phone is indoors and if it has remained in the present location for a period that is greater than a minimum settling period. If the phone is outdoors or in motion, we terminate the rest of estimation process and retry after a certain duration. Next, we invoke the state and sub-state identification techniques to determine the current state and sub-state (which indicates whether the phone is warming, steady, cooling, or idle). This reveals the correct C, k, and b parameters to input into which Equation 1-4 to model ΔT, the difference between the battery and ambient air temperatures: $\Delta T = T_B - T_A$. Figure 6 depicts an M2 curve-fit for modeling ΔT. Finally, we sample the battery temperature T_B and use the appropriate sub-state model function from Section 3.1 to obtain ΔT and subtract it from T_B to obtain T_A.

Since all of our sub-state models are functions of time, we need to know the time since entering the current sub-state t_s. If this time (*i.e.*, model parameter t) is known – for instance by using significant changes in the battery current as an indicator of state change and maintaining a counter of t_s since that change – then we can simply substitute the value of t_s into the corresponding equation to obtain ΔT at that point. However, this requires our approach to run in the

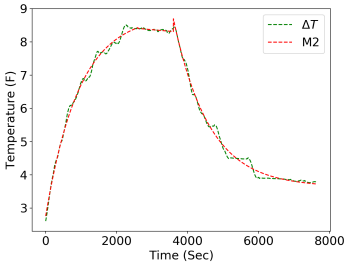


Figure 6: An example of an M2 (screen use) model and the empirical $T_B - T_A$ M2 curve.

background to monitor state and sub-state changes and demands highly accurate sub-state detection. A different approach involves sampling the battery temperatures at a minimum of two points t_1 and t_2. In this case, t_1 represents an unknown duration t since the phone entered the current sub-state, while t_2 represents a known offset Δt from t_1. Since the rise or fall in battery temperatures ΔT_B between these two instants is known and since the warming and cooling curves are exponential (non-linear), there should be only exactly one point in the curve where battery temperature changes by ΔT_B over a time duration Δt. Put another way, the samples at t_1 and t_2 yield two equations with two unknowns, t and T_A, which can trivially be determined.

A third method leverages the gradient calculation used to derive the sub-state to make predictions using a single test point. Given a sample at some absolute time t_1 with measured battery temperature T_B, we would like to determine time since entering the current sub-state t_s. We have previously calculated an instantaneous gradient T'_B of test data for this point used to detect sub-state. We can then apply the same gradient calculation to our modeling function $\Delta T(t_s)$ to produce its gradient function $\Delta T'(t_s)$. We can then taken the *argmin* of $\Delta T'(t_s) - T'_B$ to find the t_s along our model with the gradient closest to our sample's gradient. For an exponential model, we should see a unique gradient at each point along the curve.

Both of these latter methods solve two key problems:

(1) They make our system robust to sub-state detection error as t_s will be found as time since the actual sub-state transition and not since the detection of sub-state change (which can be erroneous).
(2) They allow our system to smoothly transition from one sub-state to another without formally completing the first sub-state. That is, if the cooling sub-state is entered halfway through the warming stage, the system can intelligently estimate t_s to be roughly halfway between 0 and the full duration of the sub-state.

4 DOMAIN ADAPTATION FOR NEW PHONES

In this section, we present our domain adaptation approach that enables accelerated bootstrapping of our models on a new phone. Our approach involves using an existing model for a previously learned phone (herein, a "seed") and optimizes its parameters over time to better characterize the new device. To do so, we use Gradient Descent (GD), which requires a ground truth temperature to perform optimization. We assume that the new model is learned without a

ground truth temperature sensor (ideal for learning models *in situ* for a new phone). To estimate ground truth ambient temperature, recall that battery temperature in the idle state reveals, or closely resembles, the actual ambient air temperature. Hence, T_B measured in an idle state just before a transition is assumed to be the ground truth ambient temperature T_A at that instant, and we assume that ambient air temperature does not change over a scale of minutes.

We collect the measurements of the first n samples (at a sampling rate of 1 sample/minute) and create a minibatch of data points x_i, y_i. Now, using the (C, k, b) parameters of a seed model, we make predictions of the ambient temperature $h_\theta x_i$. Next, we design a cost function, which is least sum of squares error of h and is defined as: $\ell(\theta) = \frac{1}{n}\sum_{t=1}^{n}((h_\theta x_i) - y_i)^2$

We run GD with the above cost function for m epochs and compute the new parameter values. Next, we compute the ratio of the optimized parameters to seed parameters and adjust the cooling curve parameters proportionately.

Further, once the parameters of the warming and cooling states have been optimized, the estimated T_A values at both the transition point from warming to steady and at the transition point of steady to cooling can be used as ground truth for optimizing our steady state model. Alternatively, we can linearly interpolate the T_B at the transition point from idle to warming and from cooling to idle to generate estimated ground truth for any model, assuming that Δt between these transition points is short enough. Interestingly, this method can be applied periodically along a device's lifespan to characterize changes in its thermodynamic behavior due to age or a change in case on the device.

5 EXPERIMENTAL RESULTS

Experimental Methodology. The experimental setup for evaluating our models and our approach involves using five different phones that vary by vendors, hardware characteristics such as screen size and battery capacity, and OS version (Android 5 through 7). Table 1 lists the phones used in our experiments. We also conducted experiments with phones running Android 8 and 9, but new privacy restrictions on Android 8 and 9 limit the type of OS statistics that can be monitored by user-level processes and the time resolution at which some sensor data is exposed is limited; when a phone is rooted, which effectively removes these restrictions, our techniques produce similar results to those shown below for Android 8 and 9, but we omit these results for brevity and visual clarity. These restrictions do not prevent unrooted devices from making temperature estimates although they reduce the sampling rate, limiting the frequency of temperature estimates.

Unless specified otherwise, we use the Google/HTC Pixel phone as the default phone for our experiments; further, while we experiment with a range of model states, unless specified otherwise, we report results for the default M2 model, since it is a common case for phone usage representing screen activity by the user with CPU and network utilization of less than 80%. To conduct our experiments, we first use our automated model derivation app to gather data and derive M1-M7 models for each of our five phones. We then subject these phones to various scenarios, as discussed below, and compare the efficacy of our model predictions with ground truth data from the HOBO temperature loggers.

Make	Model	OS	Screen Size (in.)
Google/HTC	Pixel	Android 7.1	5.0
LG	Stylo3	Android 7	5.7
Alcatel	5044R	Android 7	5.0
Samsung	sm-g550t	Android 6	5.0
Motorola	Moto X	Android 5	4.7

Table 1: Characteristics of phones used in our evaluation.

5.1 Model Derivation

We begin by deriving models M1-M7 for various phones using our automated model derivation app. Figure 7 depicts the empirical behavior of battery temperature of the Google/HTC Pixel phone running Android 7.1 as well as the model curves, obtained using curve fits, for different states and sub-states. As can be seen, the phone behavior and the derived models closely resemble the ideal models depicted in Figure 3.

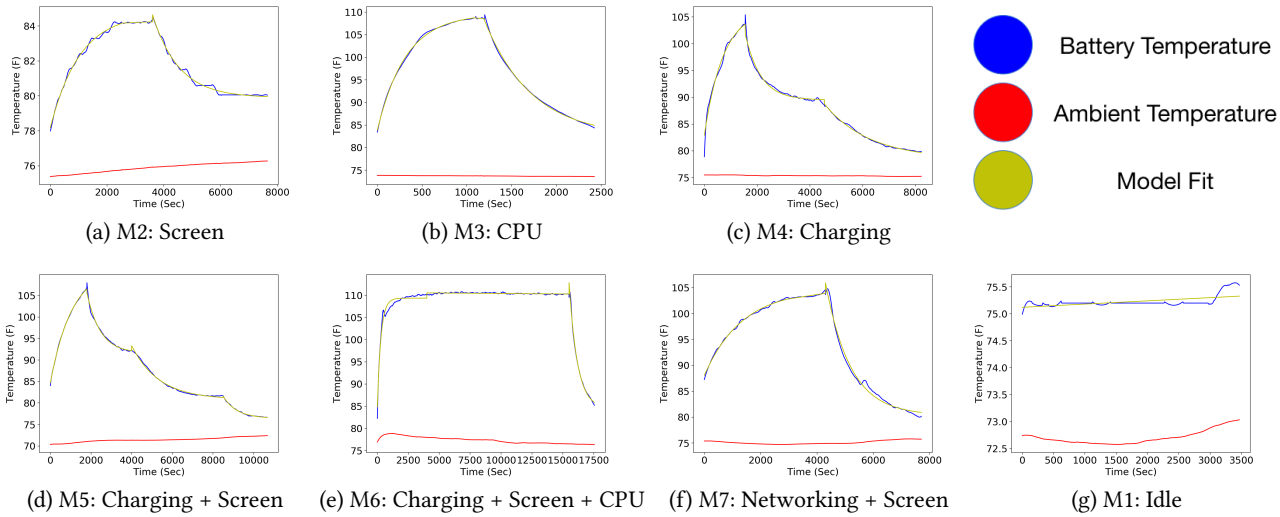
Next, we conduct an experiment to validate the accuracy of each of these models in predicting ambient air temperature. We put each phone into different states and sub-states, sample a few data points for each, and use our models to predict the ambient air temperature. As an example, Figure 8 depicts the predicted and ground truth temperatures for one such state, namely M2 on the Pixel phone; as can be seen, the predicted temperature tracks the ground truth ambient temperature closely across all sub-states of M2, although we see more errors in the initial phase of each sub-state.

Figure 11 depicts the percentage error, shown as box plots, for temperature predictions seen across various states and sub-states for the Pixel phone. The figure shows that models which capture the effects of charging (M4, M5, M6) experience more error than those that do not. But more importantly, models which capture high CPU load (M3, M6, M7) experience higher error. Most models yield mean errors of 1-2%, while high CPU load based models experience as high as 4-7% mean error in some sub-states. Note that since the majority of a device's lifetime is spent in idle state (M1) or with screen activity (M2), our models will typically exhibit low errors.

Next, we repeat this experiment for all our phone models. Figure 12(a) depicts the accuracy of ambient temperature predictions for various phones for the M2 state. As shown, the error in ambient temperature predictions is less than 2% for all phone models, and is less than 1% for several phones. This result shows that our approach works across a range of phones with varying hardware characteristics and Android OS versions.

Finally, we repeat our experiments under different ambient temperature conditions. While our models are derived for ambient temperature close to 70°F (21°C), this experiment evaluates the effectiveness of these models in predicting a range of indoor temperatures – from 70°to 85°F (21°to 29°C). Figure 12(b) shows the percentage error across different ambient air temperatures. The result shows that the temperature predictions are accurate, with mean error of 0.4-2% over a range of indoor temperatures, and our models maintain their accuracy even for ambient temperatures that are as much as 15°F higher than that used for deriving the models, likely due to our use of physics-based models. [4]

[4]Though we note that by the definition of Newton's laws of thermodynamics, variation in ambient temperature should have a slight effect on warming and cooling and



(a) M2: Screen (b) M3: CPU (c) M4: Charging

(d) M5: Charging + Screen (e) M6: Charging + Screen + CPU (f) M7: Networking + Screen (g) M1: Idle

Figure 7: Phone battery temperature behavior for a Google/HTC Pixel phone during various phone states.

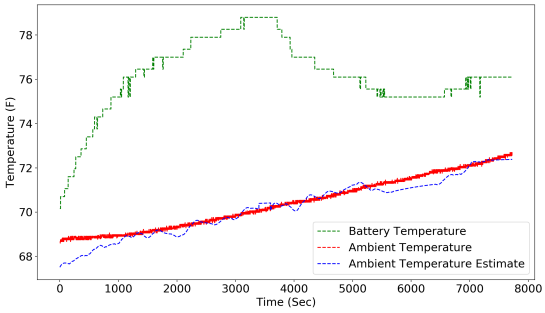


Figure 8: An example of using the M2 (screen use) model to predict ambient air temperature.

5.2 State and Sub-state Identification

As discussed in Section 3.4, context and activity recognition is a well-studied problem with state-of-the-art approaches providing high accuracy; our simple random forest classifier, inspired by these approaches, yields 91.5% accuracy in 5-fold cross validation using only accelerometer, gyroscope, proximity sensor, and light sensor data. Consequently, we omit further evaluation of context identification and focus on state and sub-state identification instead.

To evaluate the efficacy of the decision tree and rule-based model for state and sub-state identification, we used a time series of sensor readings from the phone and periodically involved our model to determine the current state and sub-state. Figure 9(a) depicts the smoothed battery temperature, while Figure 9(b) depicts the sub-states identified by our approach, while Figure 9(c) shows the accuracy of these predictions. As can be seen, there are occasional errors immediately after a sub-state transition, but our approach yields good accuracy for most predictions.

therefore steady equilibrium temperature. As the range of ambient air temperatures of commercial buildings is small, this effect is negligible for our scenarios.

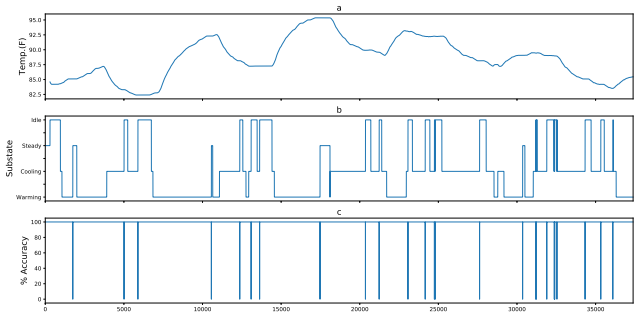


Figure 9: Sub-state identification: (a) Smoothed battery temperature (b) Instantaneous sub-state labels (c) Accuracy of sub-state labels

Model	M1	M2	M3	M4	M5	M6	M7
4 sub-states	100.0	94.1	84.2	89.3	86.2	81.5	85.7
2 sub-states	100.0	98.6	99.3	98.6	98.4	96.3	98.3

Table 2: Accuracy (in %) of sub-state detection for each model treated as (row 1) a 4 sub-state system and (row 2) with steady and warming simply as the asymptotic portions of the exponential curves.

Table 2 shows the accuracy of our state/sub-state identification across different models M1-M7. We find that model accuracy lies between 81.5% - 100%. M6 experiences the lowest accuracy (followed by M3) due to noise caused by the CPU's inherently bursty nature. However, these accuracies can be significantly increased to 96.3% - 100% by treating the steady state and idle state as asymptotic portions of the exponential warming and cooling curves, respectively. We found the warming model could be applied to steady sub-state temperatures and even produce higher accuracy in some cases. This is also true for cooling applied to warming, though due to the lengthy nature of most idle sub-states (many hours). it is preferred to be treated as its own sub-state.

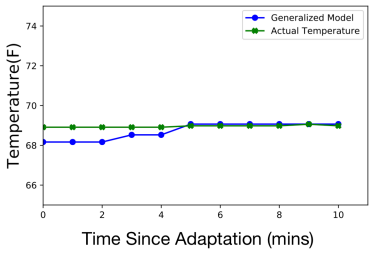


Figure 10: Ambient temperature prediction for a warming curve beyond the training phase after adjusting the parameters using domain adaptation.

5.3 Model Adaptation

To evaluate the efficacy of our domain adaptation method, we assume that we have prior models learned for two phones (Pixel running Android 7.1 and LG Style3 running Android 7). To learn a model for a new Alcatel phone, we first use a greedy heuristic to pick the previously learned model with closest hardware characteristics – in this case, the heuristic chooses the Pixel phone to serve as the initial model for Alcatel. we then optimize the selected model using a few sample data points to customize the model to the new device. Figure 10 depicts that with just ten samples and iterative updates of model parameters, the predictions converge to the actual temperature with an accuracy of 99.8% for the warming curve. This result demonstrates that new models can be "bootstrapped" using previously learned ones quickly without having to undertake a laborious full data collection phase for a new phone.

5.4 Impact of Phone Cases

Since phone cases are common for smartphones, it is important to consider the efficacy of our predictions method on phones with cases. For example, based on the material and surface area cover of the phone, the phone case may alter the rate of the phone's heat dissipation, which may also interfere with our temperature prediction. We conducted an experiment where we equipped the Pixel phone with six cases of varying thickness and materials (ranging from a slim case to a drop-proof thick Otterbox case).

In each case, we used our learned model for the phone to predict the ambient temperature and compared it to ground truth. Our results are shown in Figure 13. As shown, the mean prediction errors range from 0.3% to 4%, while the mean error without a phone case was 1%. Of the 6 cases, the Google brand case and the Anccer case had the lowest error. These cases were made of a thin and firm polymer plastic and left the most surface area of the phone exposed to the air (only covering the back and sides of the device). The Spigen and FDesign cases were both made of a thicker rubber-like material and covered encased all but the screen. Finally, the Otterbox and Asuwish cases experienced the highest error. These cases were multilayered first with a rubber casing which covered all sides of the device, and then a hard plastic outer shell placed over the rubber. The Otterbox also incorporated a clear thin plastic screen protector. This shows that as the surface area coverage and mass of the phone case increases, so does prediction error.

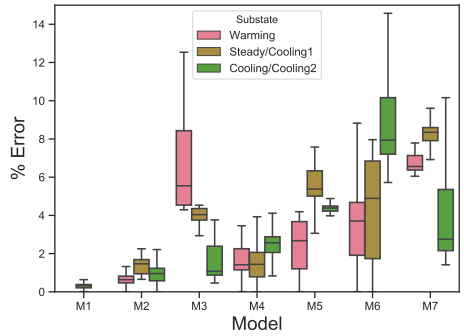


Figure 11: Errors (in %) of temperature predictions across model states and sub-states.

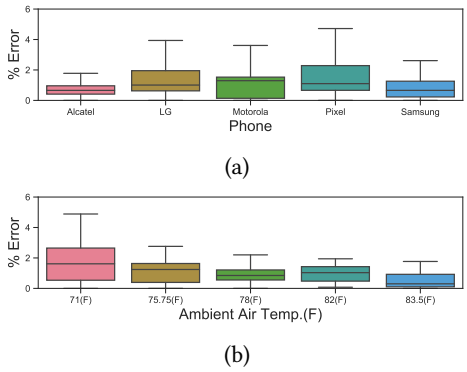


Figure 12: Errors (in %) of M2 (screen use) temperature predictions for a range of (a) devices and (b) ambient temperatures.

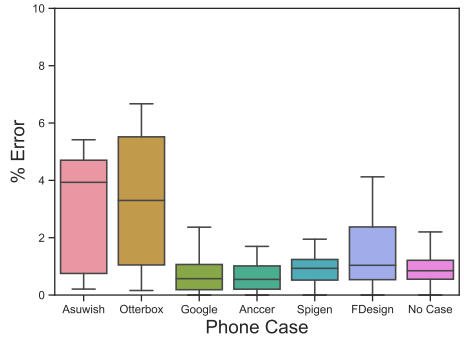
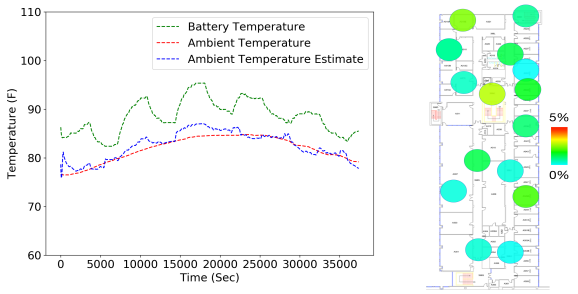


Figure 13: Errors in temperature prediction for the Google Pixel M2 (screen use) state with 6 different phone cases.

Overall these results show that our prediction methods are just as effective for phones with cases as without, which enhances their utility for crowdsourcing predictions from a broad set of phones.



(a) (b)

Figure 14: Temporal (a) and spatial (b) case studies showing 1.3% error in temperature prediction over 12 hours of varied phone behavior and ambient temperature and 1.4% error in temperature prediction over the 15 distinct locations with varying ambient temperatures and environments.

5.5 Case Study: Ambient Temperature Estimation in Buildings

In this section we discuss two case studies performed using our M2 model: one temporal shown in Figure 14(a) and one spatial shown in Figure 14(b) and discuss their performance.

‘

In our temporal case study shown in Figure 14(a), we left the phone alternating between M1 and M2 state with all represented sub-states for a 12-hour duration. This test saw an aggregate error of 1.3% across this duration while the ambient temperature varied by 6°F. Similarly Figure 14(b) shows a test of our M2 experiment with all its sub-states in 15 locations of an office building with ambient temperatures differing as much as 13°F from room to room. This test saw a mean error of 1.4% across all 15 locations, highlighting the accuracy of our techniques in a range of conditions.

6 CONCLUSIONS

In this paper, we demonstrated the feasibility of using smartphones for ambient temperature sensing inside buildings for subsequent HVAC optimization. Our approach is based on modeling battery temperature behavior using a physics-based model that uses Newton's Law of Cooling. We also presented a domain adaptation method that leveraged previously derived models to significantly reduce the overhead of learning a model for a new phone. Our results show that our approach can provide high sensing accuracy with errors of 1-7% across a range of operating conditions, with error dropping to 1% for the most common operating conditions. Our case study demonstrated the utility of our approach for spatial and temporal monitoring of temperature variations in office buildings, with an error of 1.4% across a building floor whose ambient temperature varies by 13°F.

Our future work will address many enhancements to our models. First, we aim to study if model error can be further reduced if the model parameters C, k, and b are themselves treated as functions of temperature, CPU loads, and context. This can be done through a combination of physics-based and black-box models that may yield further improvements. Generalizing our models to make predictions while the phone is in motion or subjected to other types of activities (*e.g.*, in-hand, pocket, bag, etc.) is ongoing work. Additionally, incorporating an indoor localization system can broaden the utility of this work by producing temporal and spatial temperature mapping across entire buildings automatically.

Acknowledgements: This research is supported by NSF grants 1763834, 1802523, and 1836752. We thank Maha Awaisi and Sivan Nachum for their help with data collection. We also thank our shepherd Polly Huang for guidance and the anonymous reviewers for their insightful comments in preparing our final draft.

REFERENCES

[1] [n. d.]. The Ecobee Smart Thermostat. https://www.ecobee.com.
[2] [n. d.]. Energy Information Association FAQ. https://www.eia.gov/tools/faqs/faq.php?id=86&t=1.
[3] J. Breda and J. Taneja. 2018. Fancy That: Measuring Electricity Grid Voltage Using a Phone and a Fan. In *Proceedings of ACM COMPASS '18.*
[4] Comfy Inc. 2019. Comfy App. www.comfyapp.com.
[5] A. Grush. [n. d.]. AndroidAuthority, WeatherSignal turns your phone into a personal weather station. https://www.androidauthority.com/weathersignal-app-206232/.
[6] F. Incropera, T. Bergman, D. DeWitt, and A. Lavine. 2007. *Fundamentals of Heat and Mass Transfer.* John Wiley and Sons.
[7] F. Jazizadeh and S. Pradeep. 2016. Can computers visually quantify human thermal comfort?. In *3rd ACM Int'l Conference on Systems for Energy-Efficient Built Environments (BuildSys '16).*
[8] S. Levin, S. Krishnan, S. Rajkumar, N. Halery, and P. Balkunde. 2016. Monitoring of fluoride in water samples using a smartphone. *Science of The Total Environment* 551-552 (2016), 101 – 107.
[9] J.C. McCullough, Y. Agarwal, J. Chandrashekar, S. Kuppuswamy, A. Snoeren, and R. Gupta. 2011. Evaluating the Effectiveness of Model-Based Power Characterization. *In Proceedings of the USENIX Annual Technical Conference* (June 2011).
[10] S. Nagarathinam, A. Vasan, V. Sarangan, R. Jayaprakash, and A. Sivasubramaniam. 2018. Good set-points make good neighbors - User seating and temperature control in uberized workspaces. In *5th ACM Int'l Conference on Systems for Energy-Efficient Built Environments (BuildSys '18).*
[11] A Overeem, James Robinson, H Leijnse, Gert-Jan Steeneveld, Berthold Horn, and Remko Uijlenhoet. 2013. Crowdsourcing urban air temperatures from smartphone battery temperatures. *Geophysical Research Letters* (08 2013).
[12] W. Pasut, H. Zhang, E. Arens, and Y. Zhai. 2015. Energy-efficient comfort with a heated/cooled chair: results from human subject tests.. In *Building and Environment, Vol. 84, 10-21.*
[13] D. Pisharoty, R. Yang, M.W. Newman, and K. Whitehouse. 2015. ThermoCoach: Reducing Home Energy Consumption with Personalized Thermostat Recommendations. In *2nd ACM Int'l Conference on Systems for Energy-Efficient Built Environments (BuildSys '15).*
[14] K. Saurav, M. jain, and S. Bandhyopahyay. 2018. Reducing Energy Consumption for Space Heating by Changing Zone Temperature: Pilot Trial in Lulea, Sweden. In *9th ACM Int'l Conference on Future Energy Systems (e-Energy '18).*
[15] A. Trivedi, J. Gummeson, D. Irwin, D. Ganesan, and P. Shenoy. 2017. iSchedule: Campus-scale HVAC Scheduling via Mobile WiFi Monitoring. In *8th ACM Int'l Conference on Future Energy Systems (e-Energy '17).*
[16] O. J. Walch, A. Cochran, and D. B. Forger. 2016. A global quantification of "normal" sleep schedules using smartphone data. *Science Advances* 2 (May 2016), e1501705–e1501705. https://doi.org/10.1126/sciadv.1501705
[17] J Wiese, S Saponas, and AJ Brush. 2013. Phoneprioception: enabling mobile phones to infer where they are kept. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems.*
[18] J. Yang, E. Munguia-Tapia, and S. Gibbs. 2013. Efficient In-Pocket Detection with Mobile Phones. In *UbiComp'13.*
[19] O. Yurur, C. Liu, Z. Sheng, V. Leung, W. Moreno, and K. Leung. 2016. Context-Awareness for Mobile Sensing: A Survey and Future Directions. *IEEE Communications Surveys and Tutorials* 18, 1 (2016).