Mites: Design and Deployment of a General-Purpose Sensing Infrastructure for Buildings

SUDERSHAN BOOVARAGHAVAN, Carnegie Mellon University, United States
CHEN CHEN[†], University of California San Diego, United States
ANURAG MARAVI, Carnegie Mellon University, United States
MIKE CZAPIK, Carnegie Mellon University, United States
YANG ZHANG[†], University of California Los Angeles, United States
CHRIS HARRISON, Carnegie Mellon University, United States
YUVRAJ AGARWAL, Carnegie Mellon University, United States

There is increasing interest in deploying building-scale, general-purpose, and high-fidelity sensing to drive emerging smart building applications. However, the real-world deployment of such systems is challenging due to the lack of system and architectural support. Most existing sensing systems are purpose-built, consisting of hardware that senses a limited set of environmental facets, typically at low fidelity and for short-term deployment. Furthermore, prior systems with high-fidelity sensing and machine learning fail to scale effectively and have fewer primitives, if any, for privacy and security. For these reasons, IoT deployments in buildings are generally short-lived or done as a proof of concept. We present the design of Mites, a scalable end-to-end hardware-software system for supporting and managing distributed general-purpose sensors in buildings. Our design includes robust primitives for privacy and security, essential features for scalable data management, as well as machine learning to support diverse applications in buildings. We deployed our Mites system and 314 Mites devices in Tata Consultancy Services (TCS) Hall at Carnegie Mellon University (CMU), a fully occupied, five-story university building. We present a set of comprehensive evaluations of our system using a series of microbenchmarks and end-to-end evaluations to show how we achieved our stated design goals. We include five proof-of-concept applications to demonstrate the extensibility of the Mites system to support compelling IoT applications. Finally, we discuss the real-world challenges we faced and the lessons we learned over the five-year journey of our stack's iterative design, development, and deployment.

CCS Concepts: • **Computer systems organization** → **Sensor networks**; **Sensors and actuators**.

Additional Key Words and Phrases: Sensing and Sensor Technologies, Distributed Sensor Network, Real–World Deployment

ACM Reference Format:
Sudershan Boovaraghavan, Chen Chen, Anurag Maravi, Mike Czapik, Yang Zhang, Chris Harrison, and Yuvraj Agarwal. 2023. Mites: Design and Deployment of a General-Purpose Sensing Infrastructure for Buildings. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 7, 1, Article 2 (March 2023), 32 pages. https://doi.org/10.1145/3580865

[†]Authors contributed to the project while they were at Carnegie Mellon University.

Authors' addresses: Sudershan Boovaraghavan, sudershan@cmu.edu, Carnegie Mellon University, Pittsburgh, United States; Chen Chen, chenchen@ucsd.edu, University of California San Diego, La Jolla, United States; Anurag Maravi, anuragmaravi12@gmail.com, Carnegie Mellon University, Pittsburgh, PA, United States; Mike Czapik, mike_czapik@yahoo.com, Carnegie Mellon University, Pittsburgh, United States; Yang Zhang, yangzhang@ucla.edu, University of California Los Angeles, Los Angeles, United States; Chris Harrison, chris.harrison@cs.cmu.edu, Carnegie Mellon University, Pittsburgh, United States; Yuvraj Agarwal, yuvraj@cs.cmu.edu, Carnegie Mellon University, Pittsburgh, United States.

2474-9567/2023/3-ART2
https://doi.org/10.1145/3580865

1 INTRODUCTION

The vision of building-scale, rich-sensing infrastructure that powers a wide variety of sensing applications has been a long-standing research goal. Application domains span from infrastructure utilization (e.g., conference room availability, coffee machine utilization, parking garage capacity, lounge occupancy, trash receptacle overflows) [10, 25, 73] to occupant wellness and productivity (e.g., room temperature, noise level, artificial lighting intensity, color temperature, air circulation, and HVAC performance) [54, 61, 103, 108]. Such data is also vital in making buildings more sustainable and maintainable.

Supporting these diverse applications and stakeholder needs, as well as robustly and flexibly operating in the complex and dynamic physical context of a "living" building, requires a multifaceted technical solution that has so far been elusive [18, 27, 96]. While existing research has proposed many building-scale distributed sensing approaches [3, 6, 14, 16, 19, 24, 42, 49, 50, 52, 59, 87, 88, 97, 106], comprehensive and real-world deployments of general-purpose IoT sensors continue to be relatively rare and challenging [11, 41, 44, 98]. A major limitation is that today's building-wide sensing approaches are not built with the necessary primitives for long-term and stable deployment at scale or with the richness in sensing necessary to support a diverse range of applications. For example, several research efforts have proposed heterogeneous sensing systems in different application domains, such as energy analysis [87, 106], occupancy modeling [6, 19, 42, 50], thermal control [5, 14], building modeling [66], safety [82] and maintenance applications [21]. These systems depend on *purpose-built* hardware instead of *general-purpose* designs that only sense a limited set of parameters, such as presence (using PIR or motion sensors) or energy usage. Fidelity is typically low, and systems lack accompanying tools, interfaces, and features to support impactful long-term deployment. Even state-of-the-art commercial building management systems such as those offered by Johnson Controls [51, 92] sense a limited set of parameters (e.g., presence, temperature, humidity) consisting of vertically integrated stacks that are most often single-point, vendor-controlled solutions with little to no extensibility to enable other applications. While numerous cloud-based sensor kits offer rapid prototyping capabilities (e.g., Adafruit FeatherBoards [2], Arduino/RaspberryPi + miscellaneous sensor shields [60, 84]), none of these approaches are integrated with necessary components such as storage, security, and machine learning components that are critical for practical deployment.

Several systems [3, 49, 58, 88, 97] have attempted to overcome at least some of these limitations with cloud-connected, general-purpose sensing infrastructure. Most often, these efforts deploy sensor suites that consist of multiple sensor modules, either as a single device or a constellation of devices. Although these approaches have shown the promise of high-fidelity sensing and machine learning, these systems have only been deployed at a small scale, only for shorter durations, have limited functionality for end-user's data access, and have limited primitives for privacy and security [11, 44]. In addition, these systems only support a subset of building stakeholders, such as building managers or administrators, are not easily scaled to operate in real-world IoT environments, and often do not support closely integrated machine learning (ML) tools.

In this work, we move beyond these prior efforts to create a unified, high-fidelity, and general-purpose sensing system, Mites, for smart buildings. During the design of the Mites system, we had to address numerous key technical challenges (elaborated further in Section 2). First, we present the design of a custom Mites hardware device to provide the most comprehensive suite of sensors to enable general-purpose, high-fidelity sensing. Importantly, the Mites device firmware performs on-board featurization of the sensor data to obscure Personally Identifiable Information (PII). This 'edge' featurization balances data privacy while providing useful data that can support a diverse range of applications. We then present the design of our custom Mites software backend to handle data streams from hundreds of Mites devices efficiently and securely while supporting various primitives for data privacy and granular data access control by occupants. We also needed to design Mites to support the management and maintenance of a building-wide IoT network. To adapt to real-world environments and to enable reliable data collection, we developed two key system-level features: (a) an adaptive packet rate mechanism

where the Mites devices dynamically adapt their data rates based on network conditions; and (b) an opportunistic data sending approach that adds edge intelligence on the Mites device to decide when to send data (or not). To enable diverse applications, we integrated a closely-coupled ML layer with features to support the end-to-end ML lifecycle (data labeling, model training and optimizations, and efficient model serving). To showcase the design features of Mites, we developed five proof-of-concept applications across three application domains (building maintenance, occupancy modeling applications, and activity modeling applications), targeting different stakeholders. In summary, we make the following contributions:

- We present the design and development of Mites[1], a general-purpose sensing system that consists of a new physical sensor board and a backend architecture that supports fine-grained sensing in large-scale building deployments with a series of architectural optimization for scalable data processing, privacy, security, and machine learning.
- We evaluate our iterative design decisions through a series of microbenchmarks and end-to-end evaluations of the Mites system to handle sensor streams at scale. We show that our system adapts to real-world network conditions and achieves high data delivery rates (94% of packets). We also show that the Mites system effectively achieves our stated design goals in a real-world deployment.
- We created five proof-of-concept applications in three application domains to demonstrate the flexibility and extensibility of the Mites system to address challenges in smart buildings.
- Finally, we describe our experience, including the real-world challenges we faced during our five-year journey of iterative design and deployment of 314 Mites devices in Tata Consultancy Services (TCS) Hall at Carnegie Mellon University (CMU) — a 90,000 sqft, five-story mixed-use office building with real-world occupants.

2 CHALLENGES & DESIGN GOALS

Our overarching goal in designing the Mites system was to be general-purpose to support both existing building use cases (e.g., climate control and occupant comfort, fault diagnosis, energy efficiency) as well as emerging ML-supported smart building applications (e.g., context sensing towards occupant wellness, resource management, fault detection, and diagnosis). In addition, we aim for the Mites system to support future use cases without requiring the underlying Mites sensor device or the software stack to be changed. This section identifies key challenges in enabling a general-purpose sensing infrastructure for buildings (Section 2.1) and use them to motivate six core design goals and requirements that underpin different aspects of the Mites system (Section 2.2).

2.1 Challenges in Enabling a General-Purpose Sensing Infrastructure for Buildings

2.1.1 Diverse Application Requirements. Smart building applications have diverse requirements in terms of sensor data, privacy, security, support for machine learning, scale, and reliability. For example, environmental modeling [10, 25], management and maintenance [35, 73], energy apportionment [8] only require data at a coarse granularity or at a low temporal fidelity (e.g., average hourly temperature across floors, etc.). In addition, these applications may require less privacy-sensitive data (such as temperature or humidity) and may not need support for machine learning. On the other hand, applications such as occupancy-based HVAC control [5, 17] and sophisticated fault detection algorithms [25, 67] need highly fidelity data from multiple sensors, including those denoting occupancy. More importantly, emerging use cases such as detecting human activities using ML [57, 58], or detecting semantically meaningful contexts for occupant productivity or wellness [54, 61, 108], require rich sensor data from multiple modalities at finer granularities and fidelity to be not only effective but also often need data from more privacy-invasive sensors such as audio or presence/movement. Ultimately, supporting these current and emerging applications needs support for rich multi-modal sensing, data annotation tools, built-in ML support, and extensive support for privacy controls.

[1]For more information on this project, please visit https://mites.io/

2.1.2 Diverse Stakeholder Requirements. Buildings consist of diverse stakeholders, including the occupants, building/facility managers, and administrators, each with their own requirements for privacy, security, and potential uses of IoT sensor data. These requirements have come to the forefront given high-profile compromises of consumer IoT devices [23, 38, 72, 110], general safety and security issues with them [107], and consumer concerns about their privacy implications [36]. These requirements are, however, different across stakeholders. Building managers require the sensing hardware and infrastructure to be secure and can access the sensor data from Building Management Systems (BMSs) to monitor and manage the overall building. In addition, they require the sensing infrastructure to support several use cases (e.g., understanding general building occupancy trends to optimize lighting schedules and maintaining the IoT network) with significantly less granular data (e.g., averages across the entire floor). On the other hand, building occupants have traditionally not had access to any of the sensor data from the BMS, including even temperature/humidity trends from their own offices. Furthermore, given the increasing privacy concerns with IoT [36], occupants will naturally require expressive mechanisms for notice (e.g., what sensor data is being captured, what apps are using this data) and choice (e.g., turn on/off sensors in their own spaces, data sharing controls). Most importantly, we must ensure that any collected sensor data cannot personally identify an individual (no-PII) and that any indirect association risk of an occupant and the data from their office is also mitigated.

2.1.3 Lack of System Primitives to Support Large-Scale and Long-Term Deployments. Supporting existing and emerging smart building applications requires high-fidelity sensing, as mentioned above. However, the data generated from such IoT devices cannot be transmitted over low-power, low-bandwidth networks, and cost/practicality necessitates leveraging existing WiFi infrastructure in buildings. Using the WiFi infrastructure is also challenging since it must be shared with other traditional consumer devices, implying that IoT devices must be efficient in their network use and adapt their data rates as the bandwidth changes during the day. Furthermore, with a scale of hundreds to thousands of these sensors in a building, the backend design to gather data must dynamically adapt to efficiently manage all these data streams simultaneously based on the available compute resources [18, 27]. Finally, to support long-term deployments lasting many years, the system design must be resilient and fault-tolerant to handle common failures that are either intermittent (network drops) or more catastrophic (e.g., machines crashing or power outages) and be able to recover. Last but not least, these features need to be supported in a way that makes it easy to manage a network of this scale.

2.1.4 Need for a Closely Coupled End-to-End ML Service. Emerging applications such as human activity detection [57, 58], or facilitating occupant wellness and productivity [54, 61, 108] require building ML models to analyze sensor data. In addition, different stakeholders may customize their applications to create ML-based 'virtual sensors' [7] to detect activities (e.g., talking on the phone, in a meeting) or inferences (e.g., door open/closed, door knock) from IoT sensor data [57, 58]. In addition, for privacy, different stakeholders may require access control to keep their training data and models private and share it only selectively. Ultimately, supporting these different use cases requires closely integrated ML services to support the entire IoT ML lifecycle, including interfaces to provide training data, training ML models, deploying models, and keeping them up to date to maintain accuracy. In addition, at the scale of hundreds of these sensors in a typical building, these ML tasks need to be managed efficiently to scale to the available computing resources.

2.2 Design Goals

We briefly describe below six key goals that guided the design of the Mites system to address the challenges outlined above.

- **D1: Rich Sensing to Enable Diverse IoT Applications:** To support existing and emerging smart building applications with diverse sensing needs, the underlying sensing infrastructure should support capturing a wide

variety of physical phenomenon (*D1.1: Breadth*). Furthermore, sensor data must be synchronized and sampled at sufficient temporal fidelity to enable robust capture of subtle (e.g., writing on the whiteboard) and transient signals (e.g., door closed) in order to be truly general purpose. Additionally, the machine learning output must be accurate and stable (*D1.2: Accuracy*).

- **D2: Built-in Support for Privacy and Security:** Most IoT platforms and deployments do not provide sufficient security primitives or privacy notice and choice mechanisms [105]. Since these sensors are present in private offices and shared spaces, captured sensor data should prevent the identification of an individual (*D2.1: No PII*), including mitigating indirect association using metadata about occupants and the sensor location (*D2.2: Mitigate Association Risk*). In addition, security primitives are needed to prevent interception of and tampering with sensor data (*D2.3: Data Security*) and mitigate potential adversarial attacks (*D2.4: Tamper Resistant*). Finally, the occupants should have expressive mechanisms for notice and choice, i.e., to know and control what data is being collected in their space, how it is used, and who can access it (*D2.5: Privacy Controls*).
- **D3: Scalable, Reliable and Resilient System for Long-Term Deployments:** Building-scale IoT systems need to handle the computational requirements of several thousands of sensors efficiently (*D3.1: Efficient Compute*) by leveraging existing network WiFi infrastructure (*D3.2: Efficient Network Use*), as well as dynamically adapt to available compute and network resources (*D3.3: Adaptable*). Finally, the system design should provide reliable data collection from a dense sensor deployment and recover from common types of failures seamlessly (*D3.4: Reliable and Resilient*).
- **D4: Comprehensive Support for System Management and Maintenance:** IoT deployments in buildings, particularly research efforts, are often short-lived and expensive to maintain beyond a few weeks [40, 48] since they lack features to monitor and manage devices at scale. Hence, for multi-year operation, the system needs mechanisms for monitoring the health of all components (*D4.1: Comprehensive Monitoring*), as well as the ability to remotely manage individual devices (*D4.2: Device Management*).
- **D5: Integrated Machine Learning (ML) Capabilities:** The system design should support the entire lifecycle of ML inferencing on multi-modal IoT sensor data. This includes user interfaces to allow different building stakeholders to view live sensor data and to provide training labels for inferences and events they are interested in (*D5.1: Data Annotation and Management*). Furthermore, the system must efficiently handle training and inference requests from hundreds of sensor streams and applications based on available compute resources (*D5.2: Efficient ML*).
- **D6: Designed for Extensibility:** Current IoT-based sensing solutions for buildings tend to be vertically integrated systems designed for a limited set of purposes, often just one (e.g., occupancy, motion-triggered lighting), and are not readily extended to support other use cases [51]. Furthermore, any updates to include new features over time tend to require extensive ground-up redesign. Thus, the system design should use extensible APIs that enable adding new components and applications (*D6.1: Extensible Architecture*) supporting different stakeholders (*D6.2: Diverse End-Users*).

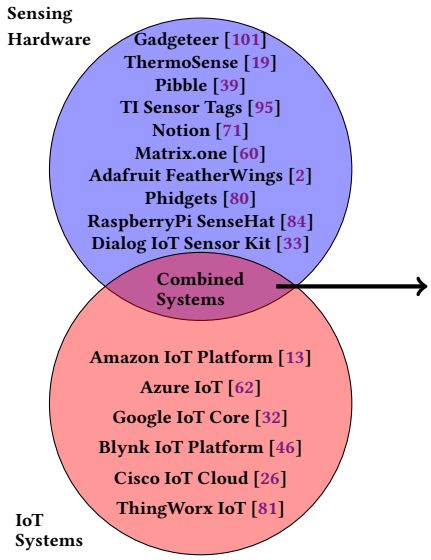
3 RELATED WORK

We categorize the prior work as: IoT sensors (Section 3.1), IoT cloud platforms (Section 3.2), and systems with hardware and software backends (Section 3.3). We compare Mites with the most relevant ones in Table 1.

3.1 IoT Sensors

In recent years, both researchers and industry have made strides towards providing hardware sensing solutions that capture different environmental facets [19, 39, 60, 71, 95, 101]. These fall broadly into three categories: (1) *Prototyping platforms*, (2) *Special-Purpose*, and (3) *General-Purpose*. *Prototyping platforms* often include developer kits that have sensors and actuators for communities wanting to build quick prototypes and proof-of-concepts

Table 1. Comparison of Mites with prior work on large-scale, hardware-software sensing systems. The Mites system addresses all the stated design goals to provide a full-stack, general-purpose sensing solution.



	Related Work	Design Goals: Rich Sensing	Privacy & Security	Scalability	Reliability & Resiliency	Management & Maintenance	Integrated ML	Extensibility
Academic	SmartSantander [89]	✓	◆	✓	◆	✗	✗	✗
	MIT PlaceLab [45]	✓	✗	✗	◆	✓	✗	✗
	CASAS project [30]	✓	✗	✗	✗	✓	✗	✗
	SPHERE project [109]	✓	✗	✗	✗	◆	✗	✗
	AASLC [56]	✗	✗	✗	✗	✗	◆	✗
	Synthetic Sensors [58]	✓	◆	✗	✗	✗	◆	◆
Commercial	Philips HomeLab [31]	✗	✗	✗	◆	✗	✗	✗
	Sense.Mother [90]	✓	✗	✗	✗	✗	◆	✓
	MakeSense IoTEgg [49]	✓	✓	✗	✗	✗	◆	✓
	Particle IoT Apps [76]	✗	◆	✓	◆	◆	✗	✗
	Building Management Systems [51, 92, 99]	✗	✗	✓	◆	◆	✗	✗
	Mites	✓	✓	✓	✓	✓	✓	✓

✓: Available ✗: Not Available ◆: Partial features available

(e.g., Phidgets [80], Gadgeteer [101], Arduino or Adafruit Featherboards [2]). Other *Special-Purpose* sensing approaches target a single environmental facet using single or multiple sensors, towards specific special-purpose applications. For example, Risojevic et al. [86] designed a low-power sensor node with a microphone to detect ambient sound levels, while ThermoSense [19] used a low-power thermal sensor array and a PIR sensor to estimate room occupancy. In contrast, *general-purpose* sensing advocates the use of several sensing modalities to detect multiple events and environmental facets, at the same time. For example, TI's Sensor Tag [95] integrates numerous sensors on a small battery-powered device with an accompanying smartphone app to gather data over the BLE radio. While promising, such approaches generally have small-scale deployments (e.g., homes) for short durations, focused on sensing technology and specific applications (e.g., activity recognition). For long-term, large-scale building deployments, these approaches fall short in several design goals such as security/privacy, scalability, reliability, and integrated ML (see overview in Table 1).

3.2 IoT Cloud Platforms

Over the years, several IoT cloud-based platforms (such as Microsoft IoT Core [62], AWS IoT [13], Arduino Cloud [12], and ThingWorx [81]) have emerged to support the developers of IoT devices. Although these platforms provide several functionalities relevant to IoT sensor deployments (e.g., data storage, scalability, reliability), they lack the features and customizability to enable useful sensing in buildings. These platforms are hardware agnostic and general-purpose in nature but lack the end-to-end design to support several design requirements of a high-fidelity sensing infrastructure. For example, while cloud platforms can scale arbitrarily, it comes at a significant cost since all the data would need to be sent to the cloud (our test deployment would send 0.5TB/day from 300 sensors), in addition to adding significant latency. Furthermore, the security and privacy support are limited given these are closed vendor platforms, with the lack of expressive and granular privacy primitives needed for sensor data (e.g., controlling individual sensor streams) or even where the data is stored and processed (often no on-premise support). Although these platforms support the use of the other cloud-based ML systems (e.g., [1, 65]), they are generic in nature and are not meant to be tailored towards the smart building use cases.

3.3 Combined Systems (IoT Sensors + IoT Cloud Platforms)

The most relevant related works are research efforts and commercial systems that consist of IoT sensing hardware and an accompanying software backend. We compare and contrast Mites with several of these efforts in Table 1, highlighting whether they can meet our design requirements.

In the context of *Smart City* testbeds, researchers have deployed thousands of sensors on an urban scale to enable applications such as environmental monitoring and transportation (e.g., SmartSantander [89]) and low-level wireless protocols (e.g., FIT IoT-LAB [3], Signpost [4]). In addition, other solutions that combine prototyping hardware and cloud solutions to build applications in air pollution monitoring and precision farming [76] have been deployed commercially at the city scale. While these deployments are city-scale, they have limited sensing hardware (e.g., environmental sensors) specific to the application. Moreover, they do not offer comprehensive management capabilities and have limited support for security and privacy controls or ML.

Within a *Living Lab* context, Philips HomeLab [31], PlaceLab [45] and Georgia Tech's Aware Home [55] instrumented spaces with cameras and microphones and other simpler sensors for activity recognition, behavioral monitoring, etc. Recent research approaches towards creating a *Smart Home* has focused on instrumenting real-world homes with sensors for applications such as activity recognition or elder care. For example, Klakegg et al. [56] proposed a non-intrusive sensing platform by leveraging proximity and contact switches to recognize activities of the elderly living alone. Synthetic Sensors [58] proposed general-purpose sensing using a device with nine different ambient sensors and combines it with ML to recognize activities and events (e.g., water faucet running) at different levels of abstraction (e.g., microwave beep vs. cooking in the kitchen). We built upon this work and, in fact, inspired some of our own sensor choices in our Mites device [58]. However, these approaches are geared towards smaller-scale home setups, and their focus is either on the sensing technology itself or building an application toward activity recognition. In addition, they provide limited, if any, support for the building requirements for high-fidelity sensing, scale, security, privacy, reliability, and management.

Researchers have also proposed smart building deployments for different applications such as occupancy detection [5, 39], energy management [59], etc. Most of these sensor deployments were prototypes that lasted for a short duration of the study. Projects such as SmartCampus [68] and Sensor Andrew [88] deployed IoT devices within buildings with sensors to measure temperature, humidity, etc., in real-world office environments. Similarly, MakeSense [49] deployed several custom-built sensing devices called IoTEgg to capture occupant activities using environmental data obtained from the device. These deployments again are not geared towards supporting high-fidelity, general-purpose sensing, nor do they address many of the requirements for long-running deployments to support diverse applications for different stakeholders (Table 1).

4 MITES SYSTEM ARCHITECTURE

We describe the overall architecture of our Mites system for general-purpose high-fidelity sensing in buildings, highlighting how we achieve our key design goals from Section 2.2. We describe each of the underlying components (as shown in Figure 1) in further detail, namely, the Mites hardware sensor package (Section 4.1) and the firmware (Section 4.2), our Mites software backend (Section 4.3, and a set of User Interfaces (UIs) for managers and building occupants (Section 4.3.7). Finally, in Section 4.1, we describe the implementation details of our Mites system.

4.1 Mites Sensor Board

The lowest foundation layer of our stack comprises our Mites hardware package. Our goal is to provide high-fidelity sensing of various ambient environmental facets in physical spaces in a building, ultimately enabling a diverse set of IoT applications. These applications include existing ones such as energy apportionment [9], managing HVAC systems [5, 15, 17], improving occupant comfort [14, 19, 28], occupancy analytics [28, 99], automatic fault diagnosis [67]. In addition, numerous new applications can be built based on the rich set of

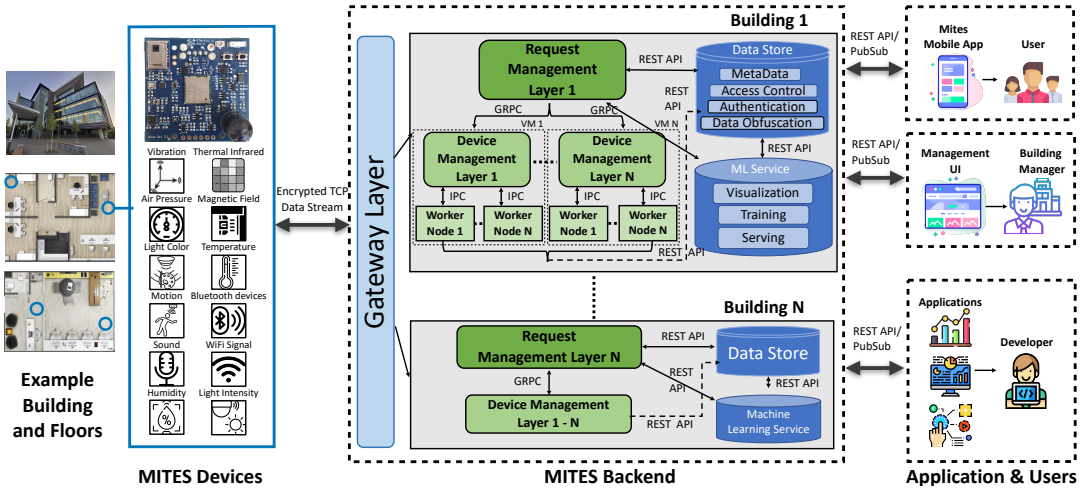


Fig. 1. Overview of the Mites system and deployment. Each room has a Mites device on the wall and in the ceiling, with larger rooms and shared areas having multiple devices in the ceiling. Each device sends an encrypted stream of featurized data for 12 sensor dimensions to our Mites software backend, which provides several key features supporting large-scale data collection and APIs for application development.

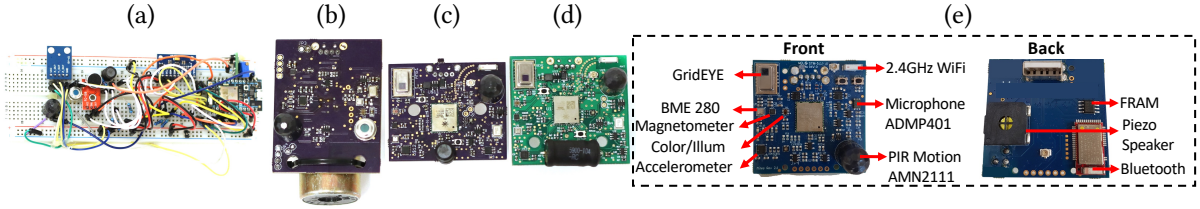


Fig. 2. Design of Mites sensor board (a–e) shows the multiple design iterations of our Mites device. Our final design (e) consists of nine discrete sensors with twelve unique sensor dimensions (vibration, thermal infrared, air pressure, magnetic field, light color, temperature, motion, Bluetooth devices, sound, WiFi signal strength, humidity, and light intensity).

ML-powered inferences, such as detecting the context of office occupants, improving occupant productivity, detecting resource waste, and improving facility management. We built our Mites device around the Particle Photon P0 [77]. We also added an additional 1 MB FRAM chip to support future memory requirements. For ease of deployment of the Mites devices in buildings, we designed two versions, both of which are powered by a standard USB-A male plug. For greenfield deployments, we designed a separate Power-over-Ethernet (PoE) to 5V USB PCB using an 802.3.af compatible AG9900M PoE [93] module. Since our Mites deployment was in a brand new university building, we had the opportunity to work closely with the architects in the building design phase itself. We had contractors install PoE switches in network closets and run Ethernet cables to all Mites locations in walls and ceilings. However, in existing buildings, Mites devices will have to be incrementally deployed, and hence we also have a second version of our device which uses a standard wall-powered 5V USB adapter. In addition, we decided to use WiFi (2.4 GHz) for data transfer in both versions, as WiFi is ubiquitous in almost all buildings and does not require additional PoE switches and infrastructure.

4.1.1 Designing the Mites Device to Capture a Diverse Set of Physical Attributes. While there are several multi-modal sensors [49, 58, 95], our goal was to create a device with the union of numerous sensor modalities to capture a wide variety of environmental facets (*D1.1: Breadth*). In addition, the multi-modal sensors used in prior

approaches were custom prototypes for small-scale studies, and their hardware or software was not available for us to modify. More importantly, building our own hardware device allows us to choose specific sensors and control all aspects of the firmware, including signal processing and computation performed on edge. Figure 2 illustrates the multi-year iterative hardware design process [69] which was crucial in identifying hardware problems early, improving sensing fidelity, helping with sensor selection, and managing hardware revision costs. The first set of revisions, shown in Figure 2(a-d), and also denoted as I1 in the project timeline (Figure 12, from April 2016 - February 2018 [58]) was important for us to explore different sensors, accuracy, range, fidelity, space/volume, and cost tradeoffs. We explored alternatives, for example, using a geophone [63] vs. various 6-axis IMUs for measuring ambient vibrations. In our latest design, shown in Figure 2(e), we settled on the TDK InvenSense MPU-6500 [94] as it provided a good trade-off with respect to its footprint and sensitivity. We also incorporated a Bluetooth Low Energy (BLE) module, the MDBT40 [85], which enables each Mites device to advertise its presence and services using our custom implementation of the beaconing mode (iBeacon or Eddystone protocols). Our Mites mobile app can then search for Mites devices in close proximity, allowing the user to discover and claim a device in their office (Section 4.3.7 provides details). Incorporating BLE also enables additional services, for example, searching for nearby devices for context-driven services and counting the number of occupants. Altogether, we believe our design offers the most comprehensive set of sensors in a single package to date (*D1.1 - Breadth*).

4.1.2 Robust Signal Processing for High Signal-to-Noise Ratio. Just breadth of sensing is not enough by itself since sensor data needs to be captured at sufficient fidelity for feature extraction and accurate inferences. In our latest design, we made several enhancements towards this goal. For example, we found that the microphone and the thermal sensor (GridEYE [75]), while promising [58], were susceptible to low signal-to-noise ratio (SNR). Hence, we explored different off-the-shelf omnidirectional microphones, both top and backported, to measure their sensitivity and their range. In our latest design, we chose a top-ported MEMS microphone that includes a low noise input buffer and an output amplifier. We designed the amplifier to interface with an analog front end whose gain could be controlled programmatically based on SNR, which enhances the microphone signal as needed. Overall, to reduce signal noise, we use standard mixed-signal PCB design techniques, such as separating the power rails for analog and digital sensors, as well as isolating the analog/digital ground planes. We also placed all digital sensors on the left half of our PCB and the analog ones on the right and kept the power traces as short as possible. Finally, we selected amplifiers with proper bandwidth and shunted the input bias currents with external resistors. While these techniques are known to circuit designers, we found that without careful hardware design, the subsequent featurized data obtained after signal processing from our early Mites prototypes did not provide accurate inferences (*D1.2: Accuracy*). We show in Section 5.2.1 that with our iterative design process of a careful selection of sensors and hardware design and rigorous efforts in signal processing, our current Mites hardware iteration (Figure 2(e)) improves the accuracy in detecting activities that happen at a longer range by two-fold in comparison to the previous version (Figure 2(d)) of the devices, thus, showing an improvement in accuracy.

4.2 Mites Firmware

4.2.1 Edge Featurization to Denature Sensor Data. We featurize and denature the raw sensor data on-board to ensure that raw privacy-sensitive data does not leave our Mites device. For example, the microphone data is converted into a low-fidelity spectral representation alongside basic statistical features (min, max, median, variance, etc.) to prevent reconstruction of the source audio or to decipher speech or detect the speaker's identity. We have two high-frequency sensors: A microphone (16 kHz) and an accelerometer (X, Y, and Z at 4 kHz). Furthermore, we have nine low-frequency sensors (sampled at 10 Hz): temperature, humidity, air pressure, light color, light intensity, WiFi RSSI, motion, magnetometer (X, Y, and Z), and the GridEYE sensor [75].

For our high-frequency sensors, such as the microphone, we maintain a rolling 256-point buffer for each sensor channel's time-series data. We then calculate a Fast Fourier Transform (FFT) for this time-series data on the

device. Specifically, every 100 ms, we take the first 256 values out of 1600 values (as data is sampled at 16 kHz), discarding the rest of the data (13440 of 16000 samples per second). We then calculate a low-resolution FFT that produces 128 frequency bins to further denature the data. We only keep the magnitude information from the FFTs and discard phase information, further preventing reconstruction of the original signal. In addition, note that the send rate of the sensor can be further reduced by building managers or end users from 10 Hz to 2 Hz (2 FFTs), 1 Hz (1 FFT), etc. For our low-frequency sensors, we keep a rolling ten-sample buffer that stores approximately one second of data. Before data transmission, we compute the same seven statistical features used above: min, max, range, average, sum, standard deviation, and centroid. This level of featurization ensures that the raw sensor data is denatured before being sent out, unable to be reconstructed, and substantially safeguards against PII from being collected (*D2.1: No PII*).

We selected the features (FFT and statistical values) based on prior literature to ensure that the featurized data from these sensors can detect multiple environmental facets with varying temporal characteristics. For instance, activity recognition applications [20, 25, 29, 57] that need to identify subsecond- to seconds- scale events (e.g., *door knock* to *brewing coffee*) have been supported by FFT data from high-frequency sensors (e.g., Accelerometer, Microphone). Similarly, applications [10, 35] that need to track hour-, day- to week-long events such as *day light patterns, lighting usages* can be characterized by the statistical data such as average, standard deviation from the low-frequency and slow varying sensors (e.g., temperature, humidity, color, and illumination). Thus, we use a combination of FFT and statistical features (min, max, standard deviation, etc.) for our high and low sample sensors to enable a good tradeoff between privacy and generalizability to different applications.

4.2.2 Time Synchronization of Sensor Data. Even minor drifts in timestamps for data from sensors on each Mites device and *across* different Mites devices can prevent correlating different data streams and reduce accuracy in event detection (*D1.2: Accuracy*). For example, a door-closed event (measurable by a microphone) is associated with a structural vibration (measurable by the accelerometer closer to the door). This co-occurrence of signals enables robust ML-based inferences of real-time events. Ultimately, this boils down to accurate time synchronization across all the Mites devices and their data streams. To do this, we implemented a time-sync protocol in the Mites firmware that synchronizes the wall clock time with our backend, on power up and then periodically (currently an hour). We determined empirically that a 1 hour time sync limits the drift to below 100 ms, sufficient for our send rates of 1 - 10 Hz. Accurate time sync addresses any interleaving due to our asynchronous sampling of the sensors and mitigates processing/transmission delays before the packet is processed by our backend.

4.2.3 Watch Dog Timers. In our real-world deployment, we observed instances where some Mites devices would go into a state where they could not connect to our backend despite retrying. To recover from such hard failures (e.g., when the SOC we use crashes, WiFi network issues), we implemented a hardware watchdog to reboot the Mites device if the application or the system firmware blocks for more than a minute. After a hard reset, the Mites device typically immediately comes back online, re-initializing all sensors and resuming operation (*D3.4: Reliable and Resilient*). We also implemented a separate software watchdog to monitor the status of all low-level sensors on each Mites device. If any sensor fails to read for 60 sec, the system firmware is automatically restarted. This software watchdog only helps if the application logic can still execute correctly. Examples include blocking on I/O, I2C errors, or blocking on memory allocation (*D3.4: Reliable and Resilient*).

4.2.4 Sensor Control. Given the wide variety of physical sensors (particularly a microphone) on the Mites device, and its ubiquitous deployment across public/shared spaces and private offices, a key requirement is that authenticated users, or designated administrators, can enable/disable one or more specific sensor streams on each Mites device. This feature is essential for privacy (*D2.5: Privacy Controls*), as it gives users visibility and control of what is being sensed in their private offices/spaces.

4.2.5 Secure Data Communication. To set up an end-to-end secure channel between each Mites device and our backend, the system firmware on our devices combines standard asymmetric key cryptography to bootstrap a symmetric session key that is unique per connected device. Each Mites device creates an asymmetric key pair, and during initial commissioning, its public key is sent to the Mites gateway, and the gateway's hostname is added to each Mites device's flash storage. Thus, each Mites device and the Mites backend can mutually authenticate each other. After the initial handshake, the backend creates an AES 128-bit symmetric session key to be used for each session with a particular Mites device, rotated periodically. With these safeguards, we prevent the possibility that an adversary takes over a Mites device remotely by changing the hostname-to-IP-address mapping, as the Mites devices will not be able to complete a successful handshake and will not send any sensor data. In addition, a tampered device that does not communicate with our backend will raise a flag and generate a notification to administrators (*D2.3: Data Security & D2.4: Tamper Resistant*).

4.3 Mites Software Backend: Architecture and Design

We designed a custom backend with capabilities to efficiently handle all encrypted data streams from hundreds of Mites devices in a typical building deployment (Sections 4.3.1, 4.3.2 and 4.3.3). We also introduce a novel mechanism for privacy aware data collection (Section 4.3.4), integrate a scalable ML service to create intelligent inferences from featurized sensor data (Section 4.3.6), and expose a consistent API for application developers to create compelling apps in the future (Section 6.4). We now review these components in greater detail.

4.3.1 Load Balancing and Scalable Data Processing. Each Mites device serializes the featurized sensor data into our custom packet format, encrypts it, and sends it to the Device Management Layer (DML). The DML decrypts and deserializes these data to extract time-stamped values for each sensor on each Mites device. Our featurization step is critical to reducing data dimensionality and for data transmission efficiency, resulting in a 2 KB packet if all sensors are enabled on the Mites device (20 KB/s @10 Hz). However, at the scale of thousands of Mites streams, efficient handling is challenging. Our first iteration of the DML used a single-threaded design, rendering it unable to handle more than 20 Mites streams before introducing significant processing delays due to buffering. After several iterations, our current scale-out design incorporates a dedicated load balancer (the DML, as shown in Figure 1). Our load balancer spins up multiple identical stateless DML worker processes, each on its own processor context, for parallel data stream processing. Each DML spawns multiple workers depending on the compute capability of the machine they run on, enabling scalable stream processing with the ability of the DMLs to add more workers as necessary to scale up (*D3.1: Efficient Compute*).

4.3.2 Adapting the Device Packet Rate based on Real-World Network Conditions. At first glance, the maximum data rate of 20 KB/s per Mites device does not seem high even with hundreds of devices, and we assumed that enterprise WiFi networks would easily handle it. For example, our building deployment with 314 deployed Mites devices, translate into a mere total bandwidth of 6.5 MB/s. Our real-world empirical measurements, however, showed that our devices were unable to maintain this 10 Hz send rate reliably due to device reboots and packet loss (as shown in Figure 6(a)). There are numerous reasons for this, including only three non-overlapping channels in 2.4 GHz leading to co-channel interference, channel contention caused due to relatively small packets [100] sent by hundreds of Mites devices, as well as other WiFi devices on the same network.

To address this challenge, we designed a simple adaptive packet rate scheme where each Mites device adapts its send rate to the underlying network conditions (*D3.1: Efficient Compute & D3.2: Efficient Network Use*). Our scheme increases the packet rate of a device whenever they send reliable data within each hour (additive increase) and approximately halves the send rate in case of observed packet drops (multiplicative decrease) to discrete levels 1 Hz, 5 Hz, or 10 Hz. Our scheme leads to individual Mites devices with different send rates based on their observed network conditions, and our evaluations show that it significantly improves the overall packet rates as

Table 2. Opportunitic Data Sending: comparing different algorithms. For the Mites data, a basic Euclidean distance metric is used to calculate the difference between the sensor readings from the "Background" (i.e., no activity happening) and the current sensor readings provide the highest accuracy. Note that using PIR motion data to detect when to send data, while being more efficient to compute (400 μs), leads to a significantly higher false positive rate (49.5%) and missed activities.

Models	Accuracy	Latency (seconds)	Model Size (MB)
Using on PIR data	50%	0.0004	-
Euclidean	89%	0.035	-
Dynamic Time Wrapping (DTW)	90%	0.232	-
Dynamic Time Wrapping (DTW) - windowed	92%	0.312	-
Linear Outlier Factor	100%	1.7	3.3
KNN	99%	2.8	1.5
Logistic Regression	100%	0.01	0.6

compared to a statically configured send rate. A key lesson based on our experience is that while WiFi networks are pervasive and attractive for IoT deployments, using them for building-scale, high-fidelity general-purpose sensing requires comprehensive mechanisms to adapt to existing network conditions dynamically.

4.3.3 Opportunistic Data Sending. The total amount of data sent by each Mites device over a day translates to 1.65 GB at 10 Hz. However, only a fraction of these sensor data is actually useful and likely to reveal interesting events given long periods of inoccupancy and no human activity, including nights and weekends. Ultimately, this leads to redundant WiFi data transmission, computational costs to process the data, storage for the backend databases, and unnecessary computation from our ML service. Instead, in Mites, we propose an opportunistic data sending approach by adding edge intelligence on the Mites device. We implemented an anomaly detection algorithm using Euclidean distance as a metric in the Mites firmware with two classes: (a) the "Ambient Background"; and (b) all the other classes of interest. (*D3.3: Adaptable*) As a heuristic, we capture this ambient background profile for each Mites device late at night when we observe no movement. Then, we use this adaptive background model for each sensor channel (rolling mean and standard deviation). We store the "Ambient Background" values on each Mites device itself and calculate a normalized Euclidean distance metric with actual featurized data.

As part of a microbenchmark, we collected data for 12 different real events (e.g., activities such as talking, and knocking) for a Mites device in an example office. We experimented with a number of different non-ML-based methods such as sending data when motion is detected using the PIR sensor, or distance-based anomaly detection algorithms such as Euclidean, Dynamic Time Warping (DTW) and DTW-windowed. In addition, also explored several ML-based methods (Linear Outlier, KNN, LR) for anomaly detection and measured their accuracy, missed events (false negatives), latency and model size. The comparisons are shown in Table 2. We ruled out using ML-based methods due to the model sizes and/or their computational complexity. Ultimately, we found that the Euclidean distance metric worked best. In the firmware of each Mites device, we calculate the Euclidean distance between two arrays: the current featurized data and the data from its own "background" profile. Each device only sends the data if it is greater than a threshold (set conservatively). In addition, each device still sends *periodic keep-alive* packets at a configurable interval (set at ~5 mins) regardless to our backend. We show the efficacy of this approach in terms of reducing the data sent in our end-to-end evaluations (Section 5).

4.3.4 Architectural Design for Privacy-Aware Data Collection. Given that some of the Mites devices are located in offices with a single or a limited set of shared occupants (2 - 4 people) there is a risk of indirectly associating (with some probability) the sensor data in those spaces with the behavior of one or more of those individuals. For example, while the PIR data in an office indicates *someone's* presence, it is *more likely* to be due to the actual office occupant(s).

In our threat model, we assume that researchers and building managers managing the system are trusted. We consider outside attackers who may try to gain access to sensor data and associate it with individuals who may

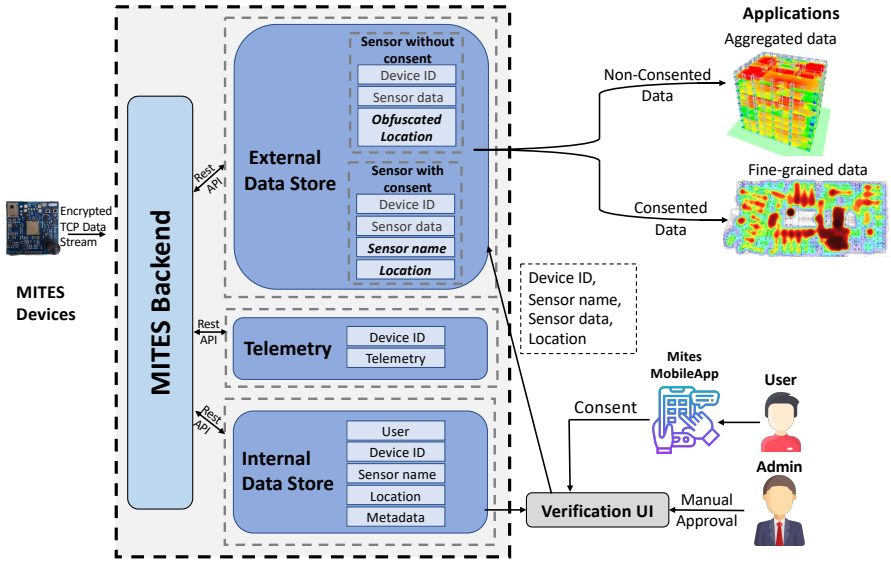


Fig. 3. Overview of our privacy-preserving data collection architecture. We show the three distinct DataStores, with the metadata information they have. "Internal Data Store" keeps the entire mapping of actual locations and users and is kept separate. The actual location and the real owners are only added to the database with sensor data, "External Data Store", after the user's consent and are manually verified by a trusted Admin. The "Telemetry Data Store" stores the telemetry data for Mites devices to monitor the status of devices.

not have yet consented to data collection. In addition, we assume that developers of apps (e.g., administrators and community members) may try to access more data than they need (over-privileged apps). We also assume that authenticated occupants may be honest but curious and may try and access data they don't have access to (e.g., from someone's office) or turn on/off sensors in their own spaces without the knowledge of their co-occupants.

Given this potential association risk, we explored a novel design to enable privacy-aware data collection from individual/shared offices by removing the precise location of the Mites device by default (*D2.2: Mitigate Association Risk*). Our approach works as follows. A Mites device in an office is initially tagged with an obfuscated location ID that corresponds to a group of offices on a floor instead of its actual location/room. For example, a Mites device will have location metadata as `Corridor:300; Cardinality: East; Room #Random_Hash` instead of its actual location in `Room:301, Floor 3`, indicating that it could be in any one of the several offices in that corridor (e.g., there are 10 offices in corridor 300, on the East facing side of our building).

Notably, we choose these obfuscated location IDs to ensure the grouping of a minimum set of offices (e.g., 7), thus breaking the association of a Mites device and any data from it with an exact location (i.e., an office). Subsequently, if an occupant wants to interact with the Mites device in their office, we collect their informed consent to associate the actual location of the Mites device with its data, since for most applications (e.g., HVAC control, viewing sensor data in your office) the actual location is necessary. For shared offices, we require consent from all occupants before we can associate the Mites device data with its actual location. However, there are corner cases where our approach needed further refinement. For example, consider the case when after a number of office occupants have consented (i.e., real locations associated with their Mites devices), only a few offices in a corridor remain unconsented (i.e., the Mites devices using obfuscated location IDs). In this scenario, it is feasible for an adversary to indirectly infer the real location of an obfuscated location ID by observing which locations are missing from a corridor (i.e., only Rooms 301 and 302 are missing from Corridor:300). To address

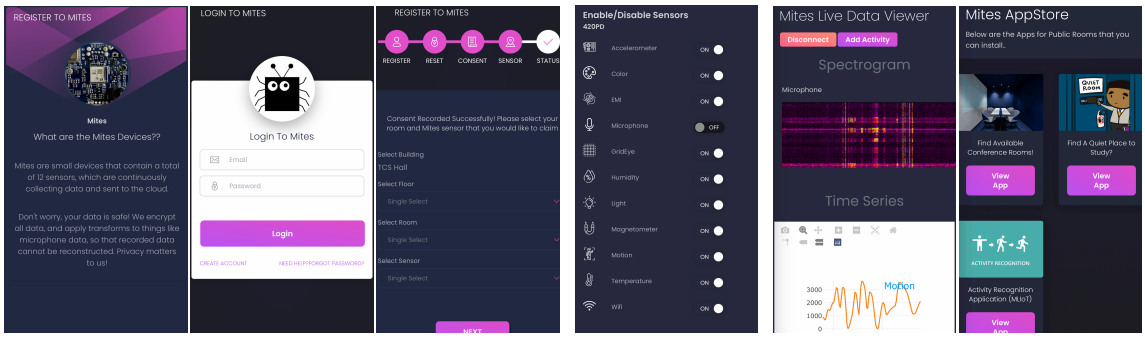
this, when the number of non-consented offices in a particular corridor and cardinality (e.g., C300 East) drops below a threshold (7 offices), we automatically obfuscate the remaining anonymous location IDs to be completely random e.g., "Corridor:Unknown;Cardinality:Unknown; Room#Random_Hash". Furthermore, to ensure at least seven random IDs, we designate a random set of Mites devices location IDs to also have anonymous location IDs.

Our privacy-aware data collection architecture is shown in Figure 3, which includes three separate Data Store instances. The "Internal Data Store" stores all metadata, such as device name, device identifiers, and the actual locations (e.g., Floor, Room, Building) of the Mites devices along with occupant information. This information is kept separate, with limited access, and is not used during regular operation. "Telemetry Data Store" stores the telemetry data for Mites devices such as reboots, packet rates, and overall uptime, as well as logs for backend status monitoring. The "External Data Store" stores all sensor data from the Mites devices, using obfuscated locations for sensor data for spaces with unconsented occupants. The actual location and the owners are only added to the "External Data Store" after the user's opt-in consent and are manually verified by a trusted administrator. Note that sensor data with obfuscated locations are still useful for certain applications (see Section 6) and aggregate statistics, e.g., the average temperature for rooms in a corridor or coarse occupancy patterns at the floor level, without any privacy risks. This approach prevents the indirect association of the non-consented individuals with the Mites sensor data collected in their offices.

4.3.5 Data Model to Create Views of Sensor Data for Privacy. Applications may need data from one or more sensors on a Mites device (e.g. occupancy detection may need PIR and thermal GridEYE data). Similarly, occupants may want to share data from a subset of their sensors with other users. Thus, having fine-grained mechanisms to enable/disable access to specific sensor(s) from a Mites device, as well as specifying the level of access (read, write, change permissions) is necessary to prevent overprivileged apps. Our data model uses a notion of a 'parent' sensor to which we post all the data from a Mites device and then implement 'data views' which can be created on demand to grant/revoke read or write access to each sensor as needed. While the solution to grant permissions to each of the 13 sensors individually may seem ideal, it leads to a management burden for the user and results in system inefficiency to post data to our backend (e.g., 130 POST REST API calls, rather than just 10 per second). This design prevents overprivileged apps or shared sensor views from violating the occupant's privacy (*D2.5: Privacy Controls*), while also being efficient in terms of data ingestion, storage, and its use for ML-based inferences.

4.3.6 Scalable Tools to Support IoT Machine Learning (ML) Lifecycles. Most modern IoT applications, ranging from personal assistants to energy analysis, rely on robust ML tools to have better insights from raw sensor data and thus help make critical operational decisions. Several ML platforms provide the necessary prototyping tools for ML (e.g., AWS IoT [13], Google Vertex AI [32]). However, such systems need to be heavily customized to incorporate the features and requirements of large-scale high-fidelity sensor deployments. We integrate our Mites system with one such extensible ML platform, called MLIoT [22], since it supports several of the necessary primitives to enable ML at scale, such as policy-driven selection of which ML models to use, deployment on heterogeneous hardware, and local deployment. We build upon MLIoT to incorporate several new features necessary for our large-scale IoT deployment use case, namely: (1) Adding support to visualize, record, and label high-fidelity data, and (2) enhancements to scale to a large number of requests for simultaneous model training and serving. Notably, MLIoT [22] was tested on roughly ten ML instances, while our use cases require hundreds to even thousands of concurrent ML instances for a medium-sized building such as TCS Hall.

Mites Data Visualization: Mites system provides a data visualization tool to support a typical ML lifecycle of getting inferences from featurized IoT sensor data (see Figure 4(c)). Essentially, it allows users to visualize different featurized data streams, provide training labels for events of interest using a programming-by-demonstration approach (PbD), train ML models, and deploy them to serve inferences – all without the user having to write any code. Low-frequency sensors are displayed as a time-series data plot while the FFT representation from the high-frequency sensors is shown as a spectrogram in our tool, allowing users to visually see whether an



(a) App onboarding, login & device claiming process (b) Privacy controls (c) Data viewer & ML (d) Applet installation

Fig. 4. Screenshots of the Mites mobile application. Our app provides occupants with controls for Mites devices in their office and the ability to view sensor data from the devices accessible to them. (a) The onboarding process for the Mites device, which includes a login to our system and claiming a Mites device to their account, and gathering user consent. (b) Various privacy controls. (c) Live data viewer allows a user to view sensor data and annotate different events for training ML models. (d) Different "Applets" that they can install, which use the sensor data from their Mites devices, some of which can have their own consent screens.

event or activity manifests across different sensor channels (*D5.1: Data Annotation and Management*). Our Mites visualization interface allows users to save a window of training data, and then use the saved data to label different types of events which are temporal (e.g., music playing) and instantaneous (e.g., door knock). The labels can then be used by any ML service to train ML models.

Optimizations to Handle Large Number of Requests ML Tasks: To handle training and inference requests on a scale from data from hundreds of Mites devices, the Mites system must efficiently serve prediction requests based on available resources (*D5.2: Efficient ML*). The underlying MLIoT system [22], which we extended for our needs, only supported a request-response pattern (e.g., using GRPC or REST APIs), which were blocking and unable to handle bursty loads with lots of POST requests from the Mites worker processes. We added support for PubSub interfaces primarily due to their non-blocking and asynchronous nature. Furthermore, we add the ability for each training serving instance pair to directly subscribe to featurized data using another PubSub stream from our Data Store to mitigate the need to poll for new data to make inferences upon, to make it event-driven.

4.3.7 Mites Mobile Application for User Control. Ultimately, we designed the Mites system to support various stakeholders, including the occupants, the building managers, administrators, and researchers. This usability is important to ensure longer-term community participation. We considered a number of requirements and key functionalities when designing the Mites mobile app, which is the main interface used by these stakeholders. First, many users will be interacting with the Mites multimodal sensing device for the first time and will need to be onboarded with its purpose and capabilities and give their informed consent. Second, the app needs to authenticate users and allow them to search for Mites devices by name/location or by proximity. Once the occupant can identify their Mites device, e.g., in their office, they need to "claim" the device and go through the onboarding process as shown in Figure 4(a). Note that all claiming requests must be manually validated by a trusted admin who checks the building directory for office occupants with the person trying to claim a device. For visibility on the data collection and to enable/disable any sensors in their spaces and respond to any permission requests to share their sensor data, we provide users with granular controls as shown in Figure 4(b). Users can also view the sensor data from the sensors they own and have claimed, including annotating different events and specifying labels, which are then used to train ML models to provide inferences, as shown in Figure 4(c). In

the future, users can also install "applets" that would use the sensor data on their Mites devices after they grant explicit permission to the granularity of the individual sensor, as illustrated in Figure 4(d).

5 EVALUATION

5.1 Experimental Setup

We deployed Mites devices in TCS Hall on the Carnegie Mellon campus, a medium-sized university building (90,000 square feet, five floors) comprising offices, shared spaces, research labs, and classrooms. While our deployment consists of 334 Mites devices, several occupants requested the devices in their offices to be turned off using our privacy controls. As a result, we evaluate and report data for 314 Mites that have been in operation for the duration of our experiment. During the deployment, while several devices required manually updating the firmware of the devices to enable bug fixes, *none* of the deployed final version of Mites devices experienced a hardware failure that required replacement, which is a testament to our iterative hardware design process. On the contrary, a significant portion of our earlier Mites prototypes experienced hardware failures. For our evaluation, our Mites backend is configured for a single-building deployment on our campus using the following setup. We have one Gateway machine (2 cores & 4 GB RAM), one RML (8 cores & 16GB RAM) and three DML machines (each 4 cores & 8 GB RAM). In addition, we set up three data store instances (each 8 cores & 16 GB RAM) and three instances of ML services (each 8 cores & 16 GB RAM). All these Virtual Machines(VMs) are provisioned on two physical machines in our current setup, each costing close to $13000. As shown in Figure 1, all Mites devices eventually stream data to the DML machines. All sensor data from the Mites DML device are stored in the external data store, and the device metrics (e.g., packet rates, uptime) are stored in the telemetry data store.

Our Mites firmware is based on Particle's open-source implementation Particle Device OS [78]. We built on this firmware to enable secure communication with the Mites backend and modified the system firmware with our custom protocol for packetization. We then built a custom application firmware that runs on top of the base system firmware to implement the features mentioned in Section 4.2. We implemented our Mites backend in Node.js [70], which includes the Gateway, RML, and DML, and is designed to be cross-platform and follow a stateless design. We also built upon Particle's spark protocol barebones reference protocol [79] to enable communication between Mites devices and our custom backend. We built our initial version of the Mites Mobile App using Google's Flutter framework [43]. We eventually pivoted to using the Vue.js web framework [102] to build a cross-platform Mites web application instead since it was easier to develop and maintain.

5.2 System Microbenchmarks

We first evaluate the different components of the Mites system that support our design goals, namely, features that enable rich sensing (D1) and make our system scalable, reliable, and resilient (D3). The features to enable privacy and security (D2) and system monitoring (D4) goals have been elaborated upon in their own sections, and hence we do not evaluate them here further.

5.2.1 Robust Signal Processing of Mites Device. To illustrate the benefits of our iterative design process of the hardware (Section 4.1.1) and rigorous signal processing to provide a high signal-to-noise ratio (Section 4.1.2), we compare the accuracy of inferring a set of activities between the current iteration (Figure 2(e) of our device and an older version (Figure 2(d)). For a direct comparison, we capture the signals from both iterations simultaneously while performing common office activities at 1 m, 3 m, and 5 m distances from the devices. In addition, we collect the same set of features and train the same machine learning algorithm (Random Forest) for activity prediction using data collected at a 1 m distance. Table 3 shows that our current iteration (Figure 2(e)) has higher accuracies (5% more at 1 m, 12% more at 3 m and 17% more at 5 m range) for activities (such as *Phone Ring*, *Vacuum Cleaning*, etc.) over a range of distances.

Table 3. Comparing the accuracy of inferring different activities based on sensor data captured at various distances (1 m, 3 m, and 5 m) for different iterations of the Mites device (Figure 2(d) vs. Figure 2(e)). The accuracy of the Mites device is much higher than that of the previous version in various activities over a range of distances. The Mites device (Figure 2(e)) improves inference accuracy compared to the older version (Figure 2(d)) by 5%, 12%, 17% at 1 m, 3 m, and 5 m, respectively.

Activities	Sensor Activation	Baseline Activity (1m)		Activity (at 3m)		Activity (at 5m)	
		D	E	D	E	D	E
Coffee Machine	GridEYE	100%	100%	NA	NA	NA	NA
Jumping	Accel	100%	100%	100%	100%	25%	43%
Lights ON	Color,Illum	100%	100%	0	0	0	0
Movement	Motion	55%	74%	0	0	0	0
Phone Ring	Mic	83%	100%	57%	92%	39%	86%
Talking	Mic	87%	84%	33%	84%	16%	62%
Vacuum	Mic, Accel	100%	100%	77%	100%	0	79%
Overall Accuracy		86%	91%	45%	57%	23%	40%

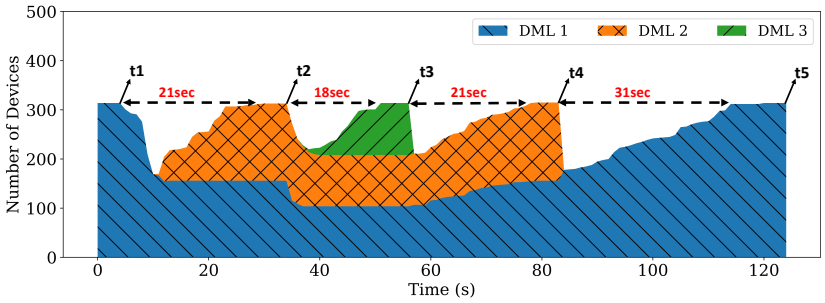
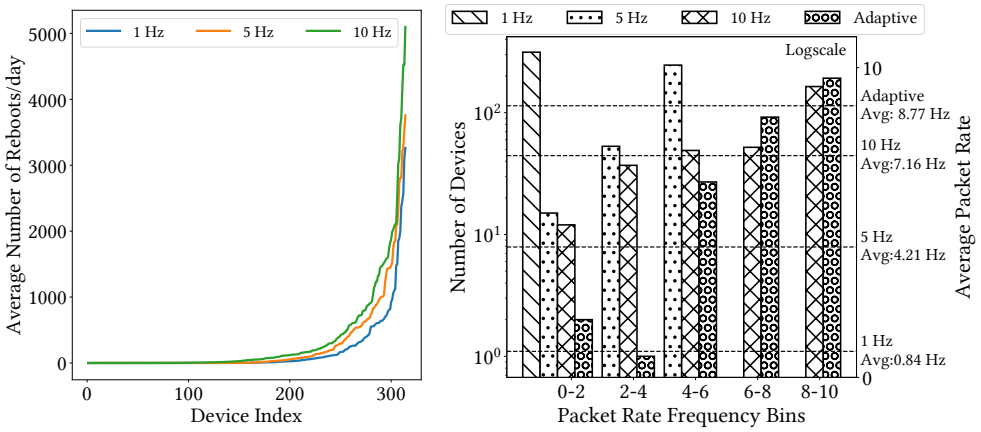


Fig. 5. Benchmarking Load balancing and Fault tolerance of Mites system with three Device Management Layer (DML) nodes (DML-1, DML-2, DML-3) that handle a total of 314 Mites devices. The area plot shows that all devices stream data to DML-1 initially. As DML-2 and DML-3 are instantiated at time instances t1 and t2, we see the Mites devices reboot and are *load-balanced* across the available DML nodes 2 and 3. Similarly, when the DML nodes go offline (At t4, DML3 is offline, and at t5, DML2 is offline), the remaining devices reboot and connect to the available DML nodes showing fault tolerance and recovery. We also see that the time taken for all devices to recover is very short, ranging from 18 – 31 seconds.

5.2.2 System Scalability and Reliability. Figure 5 illustrates a timeline trace of our load balancing and fault tolerance design across a three DML node deployment (*D3.4: Reliable and Resilience*). In this experiment, we start by handing all 314 Mites on a single DML node (DML-1). We then incrementally add a second node (DML-2) and a third node (DML-3). As the timeline shows, the sensors are load balanced across all three nodes (at t=60 seconds / t3). To illustrate our fault tolerance design, we see that when DML3 is offline (at t=57 seconds / t3) and when DML2 is offline (at t=83 seconds / t4), all the Mites devices fail over to the only remaining node, DML1, in a short period (15 seconds).

5.2.3 System Resiliency – Adapting the Device Packet Rate based on Real-World Network Conditions. We compare the efficacy of our adaptive send rate technique vs. statically configured packet rates (1 Hz, 5 Hz, and 10 Hz) in Figures 6(a) and Figure 6(b). Figure 6(a) shows that at higher send rates (10 Hz or 5 Hz), there are many more devices with a significantly higher number of reboots than at 1 Hz. Reboots are caused by the hardware watchdog resetting the device when a Mites device cannot send data for a period of time (60 seconds).

In contrast, we showcase the efficacy of our adaptive packet rate scheme in Figure 6(b). For this figure, we collected data for 12 days for different packet rate configurations. We collected data for each static configuration at packet send rates (1 Hz, 5 Hz, and 10 Hz) for 3 days each and another 3 days by enabling our adaptive packet rate scheme. Figure 6(b) shows a histogram of the number of devices in different bins of packet send rates for each



(a) Variability in device reboots

(b) Adaptive versus static packet rate configurations

Fig. 6. Efficacy of our adaptive packet rate scheme. On the left graph, we see that the number of reboots is much lower for lower send rates (1 Hz or 5 Hz) compared to sending data at the full 10 Hz. On the right graph, we show a histogram of the number of sensors binned into different packet rates. As seen in the histogram, our adaptive scheme has a much higher fraction of sensors in the 6 - 8 Hz and 8 - 10 Hz bins as compared to the 10 Hz static configuration while also observing significantly fewer reboots overall (not shown in the figure). The overall send rate for our adaptive scheme is 8.77 Hz (on average) compared to 7.16 Hz and 4.21 Hz for static 10 Hz and 5 Hz settings.

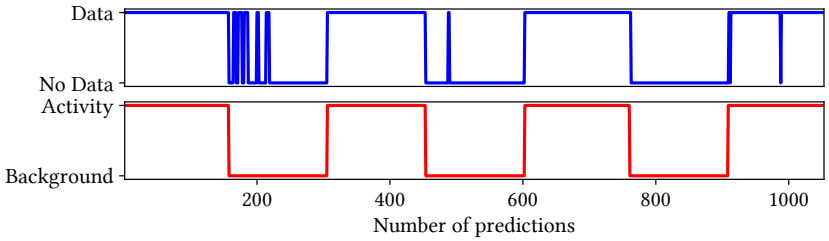


Fig. 7. Evaluation of the opportunistic data sending algorithm. The red line represents the ground truth when an activity happens, and the blue line represents when the Mites device sends the data. The false positive (incorrectly identifying background when activity is happening) and false negative rate (incorrectly identifying activity when activity is not happening) are 0.4% and 6%, respectively, implying that we do not miss activities.

configuration. We can see that our adaptive packet rate scheme performs the best with the highest fraction of sensors in the 6 - 8 Hz and 8 - 10 Hz bins and overall has the highest average packet send rate of 8.8 Hz compared to various static send rates demonstrating its effectiveness.

5.2.4 System Resiliency – Opportunistic Data Sending. In Figure 7, we illustrate the working of the opportunistic data sending algorithm (Section 4.3.3) where the Mites device sends data whenever an activity is performed with few incorrect predictions (<3). We benchmarked our approaches using different activity patterns and background noise. We identify the false positive (incorrectly identifying background when an activity is happening) and false negative rates (incorrectly identifying an activity is happening when it is only the background) as 0.4% and 6%, respectively. We conservatively optimize for lower false negatives rather than false positives since we want to ensure that we do not miss actual activities at the cost of potentially sending unnecessary data.

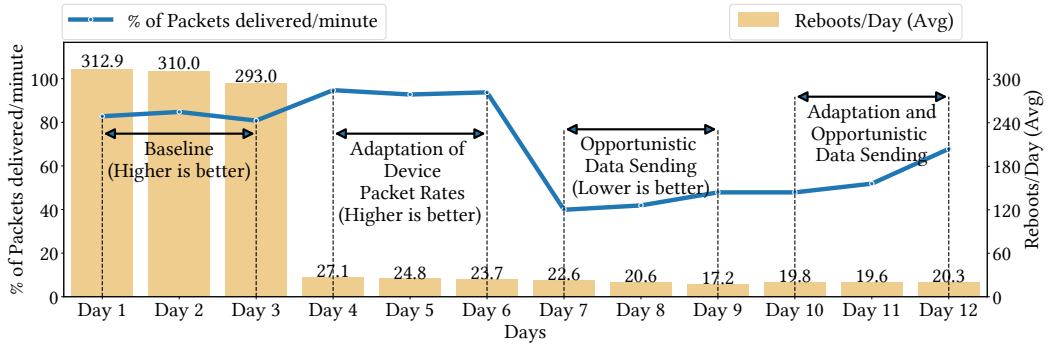


Fig. 8. Comparison of Mites system performance with various system optimizations enabled as shown by the average % of packets delivered/minute and average reboots per day across all 314 Mites devices in our deployment, gathered over 12 days. We observe that the adaptive packet rate optimization (Days 4 - 6) results in more packets delivered compared (and lower reboots ~25 across the deployment) to a statically configured rate of 10 Hz for all devices (baseline, Days 1 - 3). When we enable the opportunistic data send optimization (Days 7 - 9), we see that both the number of packets delivered and average reboots are lower since data is only sent from the Mites devices during periods of significant change from the 'Ambient Background.' Finally, when we enable both optimizations (Days 10 - 12), the average % packets delivered (reboots comparable) is generally higher and depends on the activities happening.

5.3 Overall System Evaluation

We evaluate the performance of our end-to-end system in comparison with various system optimization mentioned in Section 4.3.1 are enabled. We evaluate our system performance based on two metrics: (a) percent packets delivered/minute and (b) overall resources (CPU % and Memory) used by the Mites system.

Figure 8 plots the percentage of packets delivered/minute and the average reboots/day from the 314 Mites devices to show the efficacy of our adaptive packet rate scheme (Section 4.3.2) and opportunistic data sending (Section 4.3.3) optimizations. We performed this evaluation for 12 days, ensuring no other external factors (network reboot, user requests, etc.) affected the data collection. From Day 1 to Day 3, we configured all devices to send packets at 10 Hz (baseline). Given this is a real-world deployment, we limited our data collection to 12 days primarily to prevent any service interruptions to real-world users. From Day 4 to Day 6, we enabled the adaptive packet rate scheme. From Day 7 to Day 9, we enable opportunistic data sending. Finally, from Day 10 to Day 12, both adaptive packet rate and opportunistic data sending are enabled. We observe that enabling adaptive packet rate increases the percentage of packets delivered per minute (mean: 94% packets/min, standard deviation (SD): 1) while reducing the average reboots/day (~27) compared to the baseline of sending data at 10 Hz (mean: 83% packets/min, SD: 2) and 300 average reboots/day. Since the Mites system adaptively switched packet rates to maximize packet delivery rates, we can achieve higher packet rates. Similarly, only enabling opportunistic data send results in a lower percentage of packets (mean: 43% packets/min, SD: 4.16). This is because devices send data (at 10 Hz) only when a likely event is detected by comparing the sensor data to the ambient background. Finally, when we enable both optimizations, we see a higher packet delivery rate (mean: 56% packets/min, SD: 10.58) whenever an event occurs. This shows that devices send data without packet loss whenever an activity happens.

We also evaluate the resources consumed by the end-to-end Mites system and assess the effect of the Mites optimizations (adaptive packet rate and opportunistic data send) compared to the baseline (data sent at 10 Hz). Figure 9 shows the % CPU and % Memory usage consumed by Mites system itself ('Mites-Backend') and by the external services ('Data Store' or 'ML Service') in the scenarios mentioned: (a) baseline system, with no system optimizations, enabled and with all devices sending packets at 10 Hz (baseline), and (b) both system optimization is enabled – adaptive packet rate and opportunistic data sending. We measure and report these metrics for all scenarios (mean and standard deviation). We observe that the Mites system consumes the highest

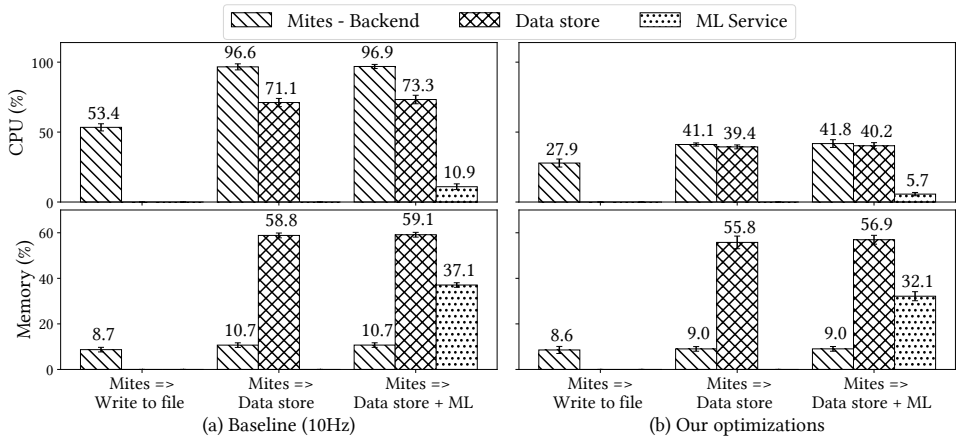


Fig. 9. System resources utilization comparison of the Mites system when extended to send the data from the Mites devices, to store it to a file or to send to the Data Store and ML Service in two scenarios: (a) baseline system (statically configured device data send at the rate of 10 Hz) and (b) our system with various optimizations enabled. Overall, the Mites system extends to different services, consuming resources efficiently when our system optimizations are enabled.

CPU and Memory resources in the baseline scenario with no system optimization (96.9% CPU and 10.7% memory when sending to the Data Store+ML) as shown in Figure 9 (a). In addition, we observe that the external services themselves also consume significant resources, with the Data Store using 73.3% CPU and ML Service using 10.9% CPU in the baseline case. In comparison to when we enable both optimizations, as shown in Figure 9 (b), CPU usage drops considerably for the Mites backend (41.8%), the Data Store (40.2%) and ML Service (5.7%) for the Mites => Data Store + ML scenario. This is because fewer data is being sent from the Mites themselves, saving computation resources across the board. Memory usage remains largely similar, as expected. Overall, our results demonstrate the efficacy of different design optimizations (adaptive packet rate and opportunistic data send) to obtain featurized sensor data at a high rate reliably from our entire network of Mites devices while ensuring that our end-to-end system consumes fewer resources.

6 APPLICATIONS

The Mites system offers several unique primitives to enable researchers and developers to build a wide variety of compelling IoT applications for smart buildings. This section presents five exemplary applications in different domains: building management and maintenance (Section 6.1), occupancy modeling (Section 6.2) and activity modeling (Section 6.3). Our goal is to demonstrate how the Mites system can be used to rapidly prototype new applications. In addition, these applications are geared towards different stakeholders of the building (occupants, building managers, etc.) and leverage different design primitives of our system.

6.1 Management and Maintenance

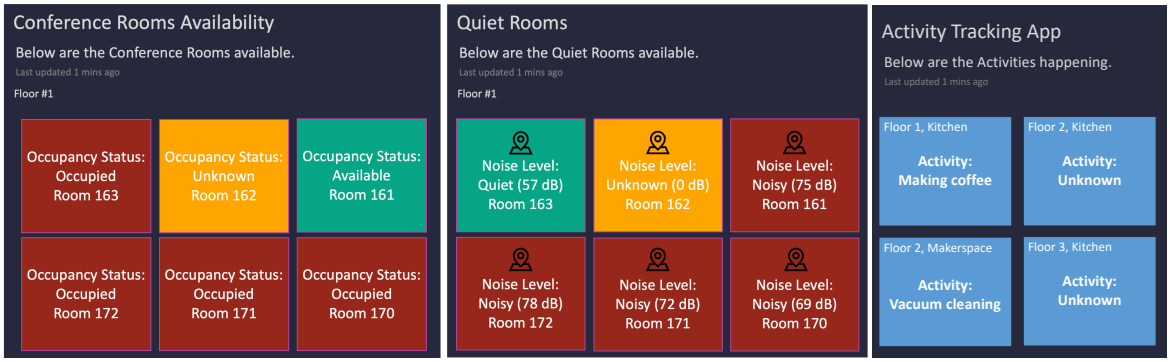
Prior efforts have prototyped applications for building managers such as building environmental modeling [10, 25], fault diagnosis and detection (FDD) [47, 67, 83], and management and maintenance [35, 73]. Such applications can help with obtaining building rating certifications around Well Building standards [103] that assess the sustainability of the building and the health and well-being of the building occupants. These applications benefit from the Mites system as it provides necessary primitives such as rich multi-modal sensing at scale while ensuring the essential privacy features needed for the building occupants (e.g., providing data only at the granularity the applications need). We present two applications we built on top of the Mites system for monitoring the building environment and the health of the sensor deployment.

(a) Spatio-Temporal Sensor Data Viewer (b) Spatial Device Health Viewer

Fig. 10. Management and Maintenance application built on the Mites system. (a) Spatio-temporal view of the temperature data of each office on a specific floor, captured from the Mites. The color-scaled temperature values that indicate the variation of warmth in rooms are less accurate for location-obfuscated data. (b) The location of individual Mites devices anchored on the floor plan. The color for each circle is filled with a corresponding scaled RGB value based on the number of reboots of that device – an important metric to assess the devices' health spatially.

6.1.1 Spatial Environment Monitoring Application. To provide a comprehensive view of the environment within a building (e.g., hot spots in the building), we implemented a spatio-temporal viewer application that stakeholders such as building managers and occupants can use to dynamically visualize current and historical sensor data captured from Mites devices. Figure 10(a) illustrates the temperature data from Mites devices in different physical spaces of the building as a heatmap (red=warm, yellow=less warm, blue=colder). We see that certain rooms and corridors in the building are warmer than others, indicating higher setpoints or potentially higher occupancy. Importantly, to ensure the privacy of the occupants of location-obfuscated rooms, the Mites system ensures that sensor data from these locations are random and grouped together into a set of rooms (Section 4.3.4). Building Managers can use this application to detect HVAC system faults and diagnose whether they are localized or systemic. In addition, when other environmental sensor data is overlayed on the same floorplan, it can be used for environmental modeling applications [25]. Building managers can also see how daylight varies across spaces (using the color and the illumination sensor) or isolate noise issues when occupants complain about adding additional insulation to walls.

6.1.2 Device Health Monitoring Application. The Mites system gathers numerous health metrics from each Mites device, e.g., the status (online/offline), the number of reboots, and signal strength. While building managers can merely view these data as a time series plot, it is often challenging to diagnose faults and find the primary cause without contexts such as the physical location or how devices in close proximity are performing (i.e., "is WiFi poor in a particular area?"). Figure 10(b) shows our spatio-temporal interface showing the reboots experienced by the Mites devices on one floor of our building. Individual circles show the location of the Mites device, and the color gradients denote the number of reboots (dark green indicates low reboots per day, while light green and yellow denote higher reboots). Our interface allows the selection of the desired time range (e.g., January 1st to 31st, 2022) and the replay of the data during that period. This interface has been an invaluable debugging tool during our deployment of 314 Mites devices on our building testbed to find transient and/or persistent issues. Our campus network team worked closely with us throughout the process and has repeatedly mentioned that the Mites deployment in the building and this specific interface were incredibly useful to them in finding and fixing problems with the WiFi network. Specifically, the Mites devices that were having trouble connecting to



(a) Conference Room Availability App (b) Find a Quiet Space App (c) Activity Tracking App

Fig. 11. Occupancy and activity modeling applications built on top of Mites. (a) Example occupancy application that detects the availability of conference rooms using PIR sensor data. (b) An application that identifies quiet spaces in the building using acoustic features from multiple Mites devices (wall and ceiling) in the same location. (c) Activity recognition application that can detect activities in a kitchenette such as *making coffee*, *heating food* utilizing several sensor channels.

WiFi, and their locations, formed a distributed WiFi client 'observatory' of sorts and helped identify WiFi dead spots, incorrect WiFi AP TX power configurations, and an intermittent issue with authentication credentials not being cached. In particular, this was the case, even though we have an enterprise-grade WiFi infrastructure.

6.2 Occupancy Modeling

Applications in the domain of occupancy modeling, such as occupancy detection [42, 91], occupant identification [37, 74], and occupancy-based control [5, 17, 19, 53] benefit from sensor deployment such as Mites. These applications primarily model the occupancy of a space or enable occupancy-based equipment control (e.g., HVAC). Such applications are valuable for building managers for energy-efficient building operations and for occupants who want comfortable workspaces. Supporting such applications requires several features of the Mites system such as rich sensing, privacy support, scale, and ML. We present two such applications that we have built for the availability of conference rooms and quiet spaces to work in the building.

6.2.1 Availability of Conference Rooms. Building managers and occupants often struggle to find meeting rooms and other common spaces. For example, conference rooms that are previously booked for meetings remain unused if the meetings are canceled or finished earlier. To address this, we built an app on top of the Mites system to identify rooms in the building as shown in Figure 11(a). This application connects to the Mites platform, identifies public resources such as conference rooms, accesses the sensor data (PIR) from the Mites devices located on the ceiling, and predicts the occupancy (green-available, red-occupied) of the space. The application also uses the ML features (Section 4.3.6) in our system to train an ML-based binary classifier model to distinguish between occupancy and non-occupancy. Building managers can use this coarse usage data and correlate it with environmental parameters like the temperature, humidity, daylight, glare, and noise from surrounding spaces to determine and ultimately fix issues that lead to some conference rooms being used less over others.

6.2.2 Find a Quiet Space. Similar to the previous application, identifying quiet or noisy locations in the building is crucial. Sound within an enclosed space from HVAC equipment, appliances, and other people, have shown to hinder productivity, focus, and memory retention in students and office employees [103]. We built a *Find Quiet space in the building* app that utilizes sensor data (microphone) from multiple Mites devices in an office/shared space (ceiling and wall) to accurately obtain the noise level in the space. Specifically, for larger rooms or halls,

combining audio data from different locations is important to (a) accurately predict noise levels and (b) localize the area with higher noise levels in the room. The application leverages the Mites system's scalable stream processing capability (Section 4.3.1) to obtain real-time sensor data from multiple devices in the room. The application then converts the obtained FFT features from the audio sensors to individual decibel values [34]. We then use the WELL building standards [104] to calculate the Sound Pressure Level (SPL) and compare it with the threshold provided to identify quiet spaces over noisy ones.

6.3 Human Activity Modeling Applications

Applications in the domain of human activity modeling will also benefit from the Mites system [29, 57, 58]. The occupants of the building can use this information to understand their activities in their personal spaces to assess their wellness (e.g., productivity or stress). These applications are geared towards detecting activities and their patterns and require several capabilities offered by Mites, namely, rich sensing modalities, data annotations tools for in-situ training of activity labels, scalable ML, and support for privacy controls. We built an activity recognition application to identify common activities performed in public locations, such as kitchenettes and our maker space. Figure 11(c) shows the activity recognition application for kitchenettes predicting activities such as *making coffee, heating food, using the sink, or toasting bread* using the sensor data from the Mites devices located in the space (particularly the accelerometer and microphone). It also leverages our scalable ML tools to annotate activities of interest, train an ensemble of models, and deploy them for prediction.

6.4 Extensible Design for Rapid Prototyping of IoT Applications

The Mites system exposes a set of REST APIs and PubSub interfaces to its backend to access the sensor data from the Mites devices or the metrics data that allow the developers to prototype IoT applications rapidly. Our REST APIs support individual and batch queries of time-series data for configurable time ranges. To support real-time low latency streaming of both sensor data as well as inferred events, we incorporated a real-time PubSub broker service (RabbitMQ) that uses the Advanced Message Queuing Protocol (AMQP) protocol. This service enables event-based asynchronous communication to deliver real-time data to different subscribers (applications) at scale, allowing many-to-many communication patterns. Notably, both our REST API and the PubSub interface use the same underlying permission mechanisms for enforcing access control to specific sensors from each Mites device for security and privacy. Using these interfaces, new applications and services can be easily integrated with the underlying Mites system, with less than 40 lines of code.

7 DISCUSSION

We have been working towards our vision of a general-purpose ubiquitous sensing infrastructure to enable smart building applications for more than five years. Figure 12 illustrates a timeline view of various key events in our iterative process of design, implementation, deployment, and refinement with real-world stakeholders, including building architects, the facility design group, contractors, and building occupants in the loop. While IoT devices are being deployed worldwide, including buildings, requirements such as security, user privacy, maintenance, and ML integration are often not first-class design constraints (Section 3). We believe that our insights into the design, development, and deployment of our building-scale sensing infrastructure based on our stated set of goals will be useful to researchers and practitioners doing similar deployments in the future, helping them achieve systems that are likely to succeed in the real world. We have organized our takeaways into five categories:

7.1 Iterative Hardware Design: Don't Reinvent the Wheel, but You May Have to Build Your Own

Before embarking on building our own multi-modal sensor package, we attempted to leverage existing designs such as the TI Sensor tag or Rasberry PIs with sensor "HATs". Ultimately, they were all inadequate for several

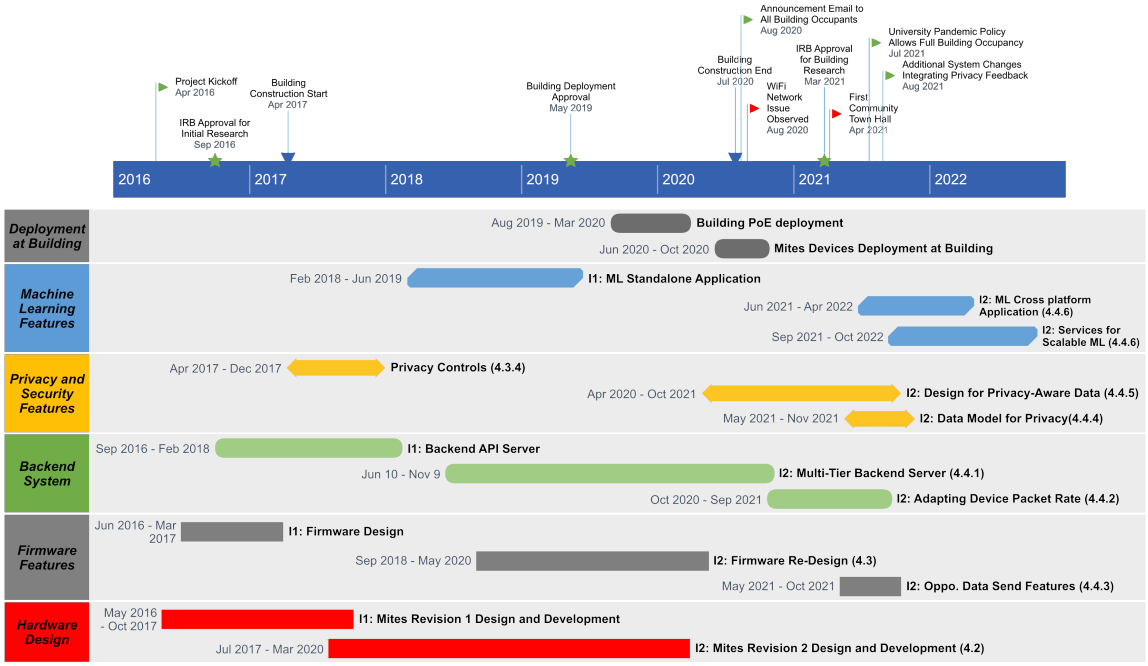


Fig. 12. Overall timeline of our Mites project. We highlight key events at the top and the features we worked on for different parts of our stack. The timeline shows our iterative design process, including multiple revisions to the Mites device, our backend, and integration with services such as ML. We also highlight the features that were directly based on our experiences and stakeholder feedback, such as the opportunistic data send feature or new privacy features, such as obfuscated locations.

reasons, such as lack of sensing capabilities, inextensible design, cost, poor programming support, lack of flexibility, etc. Ultimately, we decided to build our own prototype and leverage the sensors chosen by other researchers. Therefore, our first prototype was a 'Lo-Fi' breadboard version with ten different sensors (Figure 2(a)). After multiple iterations, we finally came up with a custom PCB design optimized for size, cost, and layout (Figure 2(b-c)). Notably, testing these prototypes in the real world was crucial. Although our first design worked (as shown in Figure 12 as I1: Mites Rev1 Design), we decided to eliminate some sensors (e.g., non-contact infrared temperature sensor) since they were superseded by others (e.g., PIR - Passive Infrared) or were capturing similar information at higher costs (e.g., a geophone is expensive compared to a MEMS IMU) [58]. Moreover, during the initial design of the Mites hardware, after much exploration, we decided to build our Mites device around the Particle's P0 module [77]. It provided us with the base prototyping functionalities in their system firmware (open source DeviceOS) and provides complete flexibility for building and customizing our own application firmware. This saved us significant time in the development process but also created a dependency on Particle's P0 hardware module, their DeviceOS, and any limitations (limited memory, older microcontroller, limited support, etc.).

7.2 Known Unknowns and Unknown Unknowns for Real-World Sensor Deployments

During our initial deployment of devices in the real world, we encountered some unexpected hardware and software problems. For example, we implemented hardware and software watchdog timers (Section 4.2) to force devices to reboot when they go into different error states, believing that it would clear all expected faults (e.g., known unknowns). Indeed, these recovery mechanisms worked for a small scale of sensors (10 - 50 sensors) for a few months, but when we deployed all 314 sensors in our building, multiple devices would periodically stop

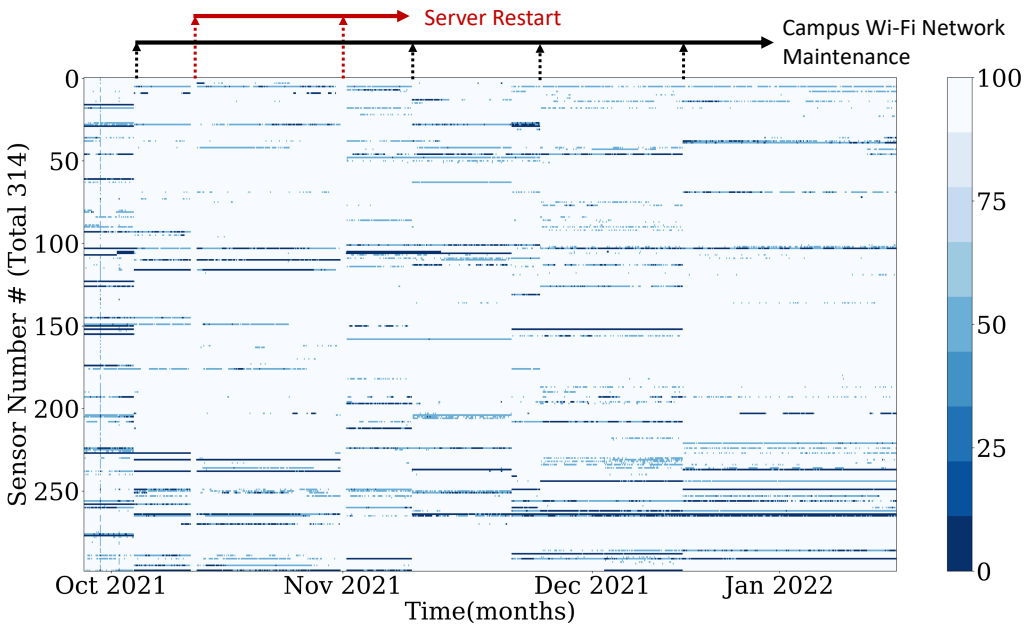


Fig. 13. Overall packet rate performance (0 - 100%) of the each Mites device over a four-month period. From Oct 2021 - Jan 2022, we see the trend that the packet rate of devices changes from worse (dark blue) to better (light blue), while some of them continue to remain worse due to WiFi receptivity issues after several upgrades to the campus WiFi network maintenance (e.g., software updates to APs) over time.

sending data, and the watchdog timers did not help recovery (e.g., unknown unknowns). After several weeks of debugging our hardware (August 2021, as shown in Figure 13), our backend, and numerous discussions with the campus networking team, we found that the enterprise WiFi controllers installed in the building randomly moved some of the Mites devices to an unregistered mode, allowing WiFi association but dismissing connection to our backend and hence not triggering the watchdog. This issue was resolved by manually clearing the state of the Aruba WiFi controllers, and a bug was filed with the vendor.

7.3 Real-world Conditions are Chaotic and Unpredictable

Each Mites device sends 2 KB packets at a maximum rate of 10 Hz (that is, 20 KB/s) over traditional enterprise 2.4 GHz WiFi networks. During our initial deployment at different locations on campus, we did not identify any problems with packet rate drop or connectivity issues of the devices. Additionally, we assumed that sending data at 20 KB/s from 300+ devices would be easily handled by enterprise WiFi networks. However, as mentioned in Section 4.3.2, achieving reliable packet delivery at 10 Hz was nontrivial even for enterprise WiFi networks (Figure 6(a)). The potential reasons for this include limited non-overlapping channels in 2.4 GHz, co-channel interference, and contention between Mites devices sending packets. To overcome this, we developed the adaptive packet rate mechanism based on real-world network conditions (Section 4.3.2). A key lesson based on our experience building-scale, high-fidelity general-purpose sensing requires comprehensive mechanisms to dynamically adapt to existing network conditions.

7.4 Deployability is the Key to Reducing Costs

We optimized the design of the Mites system for deployability, which helped us reduce the overall cost of device deployment and labor costs to install them. For example, our Mites devices, running 110 V AC drop, is

prohibitively expensive in buildings due to code requirements. Hence, to overcome this, we chose low-voltage Power-over-Ethernet (PoE) and plenum-rated Ethernet cables, which are substantially cheaper to run. In addition, to simplify this deployment process, especially for contractors who deployed the Mites devices, we programmed the Mites devices to have a "SoftAP" mode, which we extended to the application firmware. Therefore, during the deployment, to connect the Mites to the WiFi, the contractor would pull a Mites device from a box, connect it to the POE cable inside the 1-gang work box, and use SoftAP mode to connect it to the designated WiFi network. After the device successfully connected to our backend, denoted by its LEDs, using a simple interface, the contractor marked the device (indicated by its ID) with its physical location. End-to-end, our contractors reported that they took between 15 - 20 minutes per device installation at a labor cost of $100 / hour. This led to an estimated cost of $25 - 35 per device. Note that this did not include the cost of running the Ethernet cables to all locations, adding 1-gang wall boxes, and connecting the cables to network switches in the closets. Overall, to run the additional POE cables to the network equipment closet and single-gang workboxes, for around 350 total PoE Ethernet drops, our contractors charged us around $120*K* USD, including the installation of the actual sensors (resulting in approximately $340 per Mites device installation). Without these features, retrofitting the devices in an existing building would lead to higher costs. Our key insight here is that POE drops should be opportunistically added to convenient locations when a new building is being built to future-proof it for making it smart by simply adding Mites-like sensors. We designed two versions, both of which are powered by a standard USB-A male plug. Furthermore, to support ad-hoc deployments of Mites sensor nodes, especially when 110 V AC wall power is available, we design our device to be powered by a standard USB power adapter.

7.5 Understanding the Community Sense of Privacy is the Key to Building Trust

We designed and implemented numerous privacy mechanisms on the Mites device itself (e.g., edge featurization and denaturing of sensitive sensor data), as well as expressive transparency primitives and controls enabled by our Mites system and application. We built these privacy controls in anticipation of building occupant privacy requests. All TCS Hall occupants were informed of the Mites deployment (August 2020), explaining the sensor boards, our plan to submit a study design to the IRB and begin collecting sensor data, and ways to contact the research team. Notably, this was before occupants moved in since COVID restrictions were still in place. Then, after our IRB protocol was approved (March 2021) and occupants had started to occupy the new building, we held various town halls (the first of which was in April 2021, with the last held in July 2021). In these meetings, we discussed the Mites deployment and answered any questions or concerns. Throughout this community feedback period, we also provided mechanisms for community members to provide anonymous feedback. However, as with any new technology, there was skepticism and pushback from some members of our TCS Hall community. This feedback from our community members and occupants of the building was extremely helpful and led us to redesign some aspects of our data model (Sections 4.3.4), such as reducing the indirect association risk. Importantly, this improvement period happened before any sensor data was collected. We also allowed building occupants to request unplugging the Mites device in addition to the POE-based power-off mechanisms we had designed from the start of the project. We also point readers to our 20-page Mites FAQ document [64], describing our security primitives as well as the various notice and choice mechanisms we implemented for occupant privacy. Although this feedback cycle took time, it was critical for us to incorporate feedback, improve the system, and gain community trust. As of the time of writing – after more than two years of deployment – nine offices have opted out of the deployment (out of approximately 110 offices).

8 LIMITATIONS AND FUTURE WORK

While our design and implementation achieves our stated goals and has been deployed in a real-world building for almost two years, we acknowledge several limitations we encountered and opportunities for future work.

For instance, while we did incorporate a BLE module into our Mites device and implemented a proof-of-concept beaconing mode and BLE scanning mode, the full potential of this functionality has not been fully exploited to provide location-based services. In addition, BLE could also be leveraged to provide an alternate mesh network for communication, although BLE supports much lower date rates than WiFi. Additionally, BLE improves the extensibility of Mites devices by allowing them to receive sensory feeds streamed from other BLE-enabled sensors. We plan to investigate these features in our future work.

Although our adaptive and opportunistic data packet streaming strategies improved network performance, we still observed some reboots and packet losses. Furthermore, even featurized data from sensors might make some occupants uncomfortable. Both of these limitations could benefit from offloading more computation to edge devices, leaving fewer data to be streamed out. In response, we plan to explore methods to push computation further to the sensor edge, including edge-ML, under the constraints of computation and memory.

We have not performed a usability study on either our Mites mobile app or the management UI built using Mites APIs since our focus in this paper was on the system design aspects of Mites. In addition, we built the five applications and did not focus on designing a comprehensive set of smart building applications for occupants. A comprehensive usability study of Mites, especially done longitudinally, would merit its own investigation in the future and could reveal many interesting insights about how users utilize features of our system, what applications are enabled by the Mites they desire, and how they interact with applications built on top. We intend to work on this aspect in future work.

9 CONCLUSION

Real-world deployment of a large-scale sensing system is challenging due to the lack of design and architectural support for high-fidelity sensing. In addition, existing sensing systems are limited as they are geared toward specific, vertically integrated applications. Such methods are unreliable, have limited functionality for users, and have few, if any, primitives for privacy and security. We have presented Mites, a scalable end–to–end hardware-software stack for supporting and managing high-fidelity distributed general-purpose sensing in buildings. Specifically, the goals of the Mites system are to support ubiquitous large-scale management and operation of infrastructure in a way that is extensible, easy to use, and provides security while maintaining occupant privacy. We deploy and evaluate our Mites system in TCS Hall, a five-floor mixed-use office building on the Carnegie Mellon University campus. We share our key insights and sincerely believe they will impact future researchers and practitioners attempting to design and deploy a similar general-purpose sensing system for buildings.

ACKNOWLEDGMENTS

This work was partially supported by NSF Award SaTC-1801472 and the CMU's CyLab Security and Privacy Institute. We gratefully acknowledge a gift by JP Morgan Chase to support research on smart buildings at Carnegie Mellon. We acknowledge Mike Kelley from the CDFD at CMU as well as our building architect Greg LaForest from BCJ, without whose guidance and support this deployment would not have happened. We would like to thank Anind Dey, Abhijit Hota, Gierad Laput, and Robert Xiao for their help with the project's early development. We also thank Lorrie Cranor, Mayank Goel, James Herbsleb, and Jason Hong for their support and guidance throughout the TCS deployment. We would also like to acknowledge SynergyLab members Haojian Jin, Han Zhang, Shan Wang, and Shreyas Nagare as well as Andrew Kuznetsov, Jayanth Krishna Mogali, and Neeha Dev Arun for their invaluable feedback on the early revisions of the paper and our stack. Finally, we would like to sincerely thank the members of our community and occupants of TCS Hall for their invaluable feedback in enabling this multi-faceted project. Their support and critiques helped us iteratively develop an infrastructure taking into account a plurality of stakeholders. We also thank our anonymous reviewers for their constructive feedback on our paper.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation* (Savannah, GA, USA) *(OSDI'16)*. USENIX Association, USA, 265–283.

[2] Adafruit. 2023. Adafruit feather platform. https://learn.adafruit.com/adafruit-feather.

[3] Cedric Adjih, Emmanuel Baccelli, Eric Fleury, Gaetan Harter, Nathalie Mitton, Thomas Noel, Roger Pissard-Gibollet, Frederic Saint-Marcel, Guillaume Schreiner, Julien Vandaele, and Thomas Watteyne. 2015. FIT IoT-LAB: A Large Scale Open Experimental IoT Testbed. In *Proceedings of the 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT) (WF-IOT '15)*. IEEE Computer Society, USA, 459–464. https://doi.org/10.1109/WF-IoT.2015.7389098

[4] Joshua Adkins, Branden Ghena, Neal Jackson, Pat Pannuto, Samuel Rohrer, Bradford Campbell, and Prabal Dutta. 2018. The Signpost Platform for City-Scale Sensing. In *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. NA, New York, NY, USA, 188–199. https://doi.org/10.1109/IPSN.2018.00047

[5] Yuvraj Agarwal, Bharathan Balaji, Seemanta Dutta, Rajesh K. Gupta, and Thomas Weng. 2011. Duty-cycling buildings aggressively: The next frontier in HVAC control. In *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks*. IEEE, New York, NY, USA, 246–257.

[6] Yuvraj Agarwal, Bharathan Balaji, Rajesh Gupta, Jacob Lyles, Michael Wei, and Thomas Weng. 2010. Occupancy-Driven Energy Management for Smart Building Automation. In *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building* (Zurich, Switzerland) *(BuildSys '10)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/1878431.1878433

[7] Yuvraj Agarwal, Christopher Harrison, Gierad Laput, Sudershan Boovaraghavan, Chen Chen, Abhijit Hota, Bo Robert Xiao, and Yang Zhang. 2019. Virtual Sensor System. US Patent 10,436,615.

[8] Yuvraj Agarwal, Thomas Weng, and Rajesh K. Gupta. 2009. The Energy Dashboard: Improving the Visibility of Energy Consumption at a Campus-Wide Scale. In *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings* (Berkeley, California) *(BuildSys '09)*. Association for Computing Machinery, New York, NY, USA, 55–60. https://doi.org/10.1145/1810279.1810292

[9] Yuvraj Agarwal, Thomas Weng, and Rajesh K. Gupta. 2009. The Energy Dashboard: Improving the Visibility of Energy Consumption at a Campus-Wide Scale. In *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings* (Berkeley, California) *(BuildSys '09)*. Association for Computing Machinery, New York, NY, USA, 55–60. https://doi.org/10.1145/1810279.1810292

[10] Akram Syed Ali, Christopher Coté, Mohammad Heidarinejad, and Brent Stephens. 2019. Elemental: An open-source wireless hardware and software platform for building energy and indoor environmental monitoring and control. *Sensors* 19, 18 (2019), 4017.

[11] Uchenna P Daniel Ani, Jeremy M Watson, Benjamin Green, Barnaby Craggs, and Jason RC Nurse. 2021. Design considerations for building credible security testbeds: Perspectives from industrial control system use cases. *Journal of Cyber Security Technology* 5, 2 (2021), 71–119.

[12] Arduino. 2023. Arduino Cloud. https://cloud.arduino.cc/.

[13] Amazon AWS. 2023. AWS IoT| Industrial, Consumer, Commercial, Automotive | Amazon Web Services. https://aws.amazon.com/iot/.

[14] Bharathan Balaji, Jason Koh, Nadir Weibel, and Yuvraj Agarwal. 2016. Genie: A Longitudinal Study Comparing Physical and Software Thermostats in Office Buildings. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (Heidelberg, Germany) *(UbiComp '16)*. Association for Computing Machinery, New York, NY, USA, 1200–1211. https://doi.org/10.1145/2971648.2971719

[15] Bharathan Balaji, Hidetoshi Teraoka, Rajesh Gupta, and Yuvraj Agarwal. 2013. ZonePAC: Zonal Power Estimation and Control via HVAC Metering and Occupant Feedback. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings* (Roma, Italy) *(BuildSys'13)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/2528282.2528304

[16] Bharathan Balaji, Chetan Verma, Balakrishnan Narayanaswamy, and Yuvraj Agarwal. 2015. Zodiac: Organizing Large Deployment of Sensors to Create Reusable Applications for Buildings. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments* (Seoul, South Korea) *(BuildSys '15)*. Association for Computing Machinery, New York, NY, USA, 13–22. https://doi.org/10.1145/2821650.2821674

[17] Bharathan Balaji, Jian Xu, Anthony Nwokafor, Rajesh Gupta, and Yuvraj Agarwal. 2013. Sentinel: Occupancy Based HVAC Actuation Using Existing WiFi Infrastructure within Commercial Buildings. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems* (Roma, Italy) *(SenSys '13)*. Association for Computing Machinery, New York, NY, USA, Article 17, 14 pages. https://doi.org/10.1145/2517351.2517370

[18] Beecham. 2023. Why IoT Projects Fail, a Beecham Research report. https://www.idglat.com/afiliacion/whitepapers/2020-5-ar-why-iot-projects-fail-en.pdf.

[19] Alex Beltran, Varick L. Erickson, and Alberto E. Cerpa. 2013. ThermoSense: Occupancy Thermal Based Sensing for HVAC Control. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings* (Roma, Italy) *(BuildSys'13)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/2528282.2528301

[20] Sejal Bhalla, Mayank Goel, and Rushil Khurana. 2022. IMU2Doppler: Cross-Modal Domain Adaptation for Doppler-Based Activity Recognition Using IMU Data. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 4, Article 145 (dec 2022), 20 pages. https://doi.org/10.1145/3494994

[21] Arka Bhattacharya. 2016. *Enabling scalable smart-building analytics*. University of California, Berkeley, University of California, Berkeley.

[22] Sudershan Boovaraghavan, Anurag Maravi, Prahaladha Mallela, and Yuvraj Agarwal. 2021. MLIoT: An End-to-End Machine Learning System for the Internet-of-Things. In *Proceedings of the International Conference on Internet-of-Things Design and Implementation* (Charlottesvle, VA, USA) *(IoTDI '21)*. Association for Computing Machinery, New York, NY, USA, 169–181. https://doi.org/10.1145/3450268.3453522

[23] Z. Berkay Celik, Leonardo Babun, Amit Kumar Sikder, Hidayet Aksu, Gang Tan, Patrick McDaniel, and A. Selcuk Uluagac. 2018. Sensitive Information Tracking in Commodity IoT. In *27th USENIX Security Symposium (USENIX Security 18)*. USENIX Association, Baltimore, MD, 1687–1704. https://www.usenix.org/conference/usenixsecurity18/presentation/celik

[24] Chen Chen, Ke Sun, and Xinyu Zhang. 2020. CapTag: Toward Printable Ubiquitous Internet of Things: Poster Abstract. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems* (Virtual Event, Japan) *(SenSys '20)*. Association for Computing Machinery, New York, NY, USA, 669–670. https://doi.org/10.1145/3384419.3430410

[25] Bhawana Chhaglani, Camellia Zakaria, Adam Lechowicz, Jeremy Gummeson, and Prashant Shenoy. 2022. FlowSense: Monitoring Airflow in Building Ventilation Systems Using Audio Sensing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 1, Article 5 (mar 2022), 26 pages. https://doi.org/10.1145/3517258

[26] Cisco. 2023. Cisco Edge Intelligence - Edge to Multi-Cloud IoT Data Flow - Cisco. https://www.cisco.com/c/en/us/solutions/internet-of-things/edge-intelligence.html.

[27] Cisco. 2023. Cisco Survey Reveals Close to Three-Fourths of IoT Projects Are Failing. https://newsroom.cisco.com/press-release-content?articleId=1847422.

[28] Comfy. 2023. Home - Comfy. https://comfyapp.com/.

[29] Marios Constantinides, Sanja Šćepanović, Daniele Quercia, Hongwei Li, Ugo Sassi, and Michael Eggleston. 2020. ComFeel: Productivity is a Matter of the Senses Too. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 4, Article 123 (dec 2020), 21 pages. https://doi.org/10.1145/3432234

[30] Diane J. Cook, Aaron S. Crandall, Brian L. Thomas, and Narayanan C. Krishnan. 2013. CASAS: A Smart Home in a Box. *Computer* 46, 7 (July 2013), 62–69. https://doi.org/10.1109/MC.2012.328

[31] B De Ruyter, E Aarts, P Markopoulos, and W Ijsselsteijn. 2005. Ambient intelligence research in homelab: Engineering the user experience. In *Ambient Intelligence*. Springer, New York, NY, USA, 49–61.

[32] Google Cloud Developers. 2023. Cloud IoT Core | Google Cloud. https://cloud.google.com/iot-core.

[33] Dialog IOT. 2023. DA14583 IoT Sensor Development Kit | Dialog. https://www.dialog-semiconductor.com/iotsensor.

[34] DSP Related. 2023. Decibels | Mathematics of the DFT. https://www.dsprelated.com/freebooks/mdft/Decibels.html.

[35] Rizanne Elbakly, Moustafa Elhamshary, and Moustafa Youssef. 2018. HyRise: A Robust and Ubiquitous Multi-Sensor Fusion-Based Floor Localization System. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 3, Article 104 (sep 2018), 23 pages. https://doi.org/10.1145/3264914

[36] Pardis Emami Naeini, Janarth Dheenadhayalan, Yuvraj Agarwal, and Lorrie Faith Cranor. 2021. Which Privacy and Security Attributes Most Impact Consumers' Risk Perception and Willingness to Purchase IoT Devices ?. In *2021 IEEE Symposium on Security and Privacy(SP)*. IEEE, New York, NY, USA, 519 – 536. https://doi.org/10.1109/SP40001.2021.00112

[37] Jonathon Fagert, Mostafa Mirshekari, Pei Zhang, and Hae Young Noh. 2022. Recursive Sparse Representation for Identifying Multiple Concurrent Occupants Using Floor Vibration Sensing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 1, Article 10 (mar 2022), 33 pages. https://doi.org/10.1145/3517229

[38] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. 2016. Security Analysis of Emerging Smart Home Applications. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, San Jose, California, 636–654. https://doi.org/10.1109/SP.2016.44

[39] Francesco Fraternali, Bharathan Balaji, Yuvraj Agarwal, Luca Benini, and Rajesh Gupta. 2018. Pible: Battery-Free Mote for Perpetual Indoor BLE Applications: Demo Abstract. In *Proceedings of the 5th Conference on Systems for Built Environments* (Shenzen, China) *(BuildSys '18)*. Association for Computing Machinery, New York, NY, USA, 184–185. https://doi.org/10.1145/3276774.3282823

[40] Francesco Fraternali, Bharathan Balaji, Dezhi Hong, Yuvraj Agarwal, and Rajesh K. Gupta. 2021. Marble: Collaborative Scheduling of Batteryless Sensors with Meta Reinforcement Learning. In *Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation* (Coimbra, Portugal) *(BuildSys '21)*. Association for Computing Machinery, New York, NY, USA, 140–149. https://doi.org/10.1145/3486611.3486670

[41] Alexander Gluhak, Srdjan Krco, Michele Nati, Dennis Pfisterer, Nathalie Mitton, and Tahiry Razafindralambo. 2011. A survey on facilities for experimental internet of things research. *IEEE Communications Magazine* 49, 11 (2011), 58–67. https://doi.org/10.1109/MCOM.2011.6069710

[42] Shadan Golestan, Sepehr Kazemian, and Omid Ardakanian. 2018. Data-Driven Models for Building Occupancy Estimation. In *Proceedings of the Ninth International Conference on Future Energy Systems* (Karlsruhe, Germany) *(e-Energy '18)*. Association for Computing Machinery, New York, NY, USA, 277–281. https://doi.org/10.1145/3208903.3208940

[43] Google. 2023. Flutter | Flutter makes it easy and fast to build beautiful apps for mobile and beyond. https://github.com/flutter/flutter.

[44] Sasirekha GVK, Adhisaya T, Aswini P, Jyotsna Bapat, and Debabrata Das. 2019. Challenges in the Design of an IoT Testbed. In *2019 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT)*. IEEE, Jaipur, India, 14–19. https://doi.org/10.1109/ICCT46177.2019.8969009

[45] Stephen S Intille, Kent Larson, Emmanuel Munguia Tapia, Jennifer S Beaudin, Pallavi Kaushik, Jason Nawyn, and Randy Rockinson. 2006. Using a live-in laboratory for ubiquitous computing research. In *International Conference on Pervasive Computing*. Springer, Springer, New York, NY, USA, 349–365.

[46] Blynk IoT. 2023. Blynk IoT platform: for businesses and developers. https://blynk.io/.

[47] Milan Jain, Mridula Gupta, Amarjeet Singh, and Vikas Chandan. 2019. Beyond Control: Enabling Smart Thermostats for Leakage Detection. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 1, Article 14 (mar 2019), 21 pages. https://doi.org/10.1145/3314401

[48] Cisco Jasper. 2023. The hidden costs of delivering iiot services: Industrial monitoring & heavy equipment.

[49] Jie Jiang, Riccardo Pozza, Nigel Gilbert, and Klaus Moessner. 2020. MakeSense: An IoT Testbed for Social Research of Indoor Activities. *ACM Trans. Internet Things* 1, 3, Article 17 (June 2020), 25 pages. https://doi.org/10.1145/3381914

[50] Ming Jin, Ruoxi Jia, Zhaoyi Kang, Ioannis C. Konstantakopoulos, and Costas J. Spanos. 2014. PresenceSense: Zero-Training Algorithm for Individual Presence Detection Based on Power Monitoring. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings* (Memphis, Tennessee) *(BuildSys '14)*. Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/2674061.2674073

[51] Johnson Controls Developers. 2023. Johnson Controls. https://www.johnsoncontrols.com/.

[52] Areg Karapetyan, Sid Chi-Kin Chau, Khaled Elbassioni, Majid Khonji, and Emad Dababseh. 2018. Smart Lighting Control Using Oblivious Mobile Sensors. In *Proceedings of the 5th Conference on Systems for Built Environments* (Shenzen, China) *(BuildSys '18)*. Association for Computing Machinery, New York, NY, USA, 158–167. https://doi.org/10.1145/3276774.3276788

[53] Areg Karapetyan, Sid Chi-Kin Chau, Khaled Elbassioni, Majid Khonji, and Emad Dababseh. 2018. Smart Lighting Control Using Oblivious Mobile Sensors. In *Proceedings of the 5th Conference on Systems for Built Environments* (Shenzen, China) *(BuildSys '18)*. Association for Computing Machinery, New York, NY, USA, 158–167. https://doi.org/10.1145/3276774.3276788

[54] Sunder Ali Khowaja, Aria Ghora Prabono, Feri Setiawan, Bernardo Nugroho Yahya, and Seok-Lyong Lee. 2018. Contextual activity based Healthcare Internet of Things, Services, and People (HIoTSP): An architectural framework for healthcare monitoring using wearable sensors. *Computer Networks* 145 (2018), 190–206.

[55] Julie A. Kientz, Shwetak N. Patel, Brian Jones, Ed Price, Elizabeth D. Mynatt, and Gregory D. Abowd. 2008. The Georgia Tech Aware Home. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems* (Florence, Italy) *(CHI EA '08)*. Association for Computing Machinery, New York, NY, USA, 3675–3680. https://doi.org/10.1145/1358628.1358911

[56] Simon Klakegg, Kennedy Opoku Asare, Niels van Berkel, Aku Visuri, Eija Ferreira, Simo Hosio, Jorge Goncalves, Hanna-Leena Huttunen, and Denzil Ferreira. 2022. CARE: Context-awareness for elderly care. *Health and Technology* 11, 1 (2022), 211–226.

[57] Gierad Laput and Chris Harrison. 2019. Exploring the Efficacy of Sparse, General-Purpose Sensor Constellations for Wide-Area Activity Sensing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 2, Article 55 (jun 2019), 19 pages. https://doi.org/10.1145/3328926

[58] Gierad Laput, Yang Zhang, and Chris Harrison. 2017. Synthetic Sensors: Towards General-Purpose Sensing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 3986–3999. https://doi.org/10.1145/3025453.3025773

[59] Du Li, Bharathan Balaji, Yifei Jiang, and Kshitiz Singh. 2012. A Wi-Fi Based Occupancy Sensing Approach to Smart Energy in Commercial Office Buildings. In *Proceedings of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings* (Toronto, Ontario, Canada) *(BuildSys '12)*. Association for Computing Machinery, New York, NY, USA, 197–198. https://doi.org/10.1145/2422531.2422568

[60] Matrix Labs. 2023. MATRIX is the World's Most Popular App Ecosystem for the Convergence of AI and Internet of Things. https://matrix.one/.

[61] Georgios Meditskos and Ioannis Kompatsiaris. 2017. iKnow: Ontology-driven situational awareness for the recognition of activities of daily living. *Pervasive and Mobile Computing* 40 (2017), 17–41.

[62] Microsoft. 2023. Azure IoT – Internet of Things Platform | Microsoft Azure. https://azure.microsoft.com/en-us/overview/iot/.

[63] Mostafa Mirshekari, Shijia Pan, Adeola Bannis, Yan Pui Mike Lam, Pei Zhang, and Hae Young Noh. 2015. Step-Level Person Localization through Sparse Sensing of Structural Vibration. In *Proceedings of the 14th International Conference on Information Processing in Sensor Networks* (Seattle, Washington) *(IPSN '15)*. Association for Computing Machinery, New York, NY, USA, 376–377. https:

//doi.org/10.1145/2737095.2742924
[64] Mites.io developers. 2023. Mites.io - TCS Hall Deployment FAQs. https://mites.io/TCSDeployment/Mites-FAQs.pdf.
[65] MLFlow. 2023. MLflow - A platform for the machine learning lifecycle | MLflow. https://mlflow.org/.
[66] Srinarayana Nagarathinam, Arunchandar Vasan, Venkatesh Sarangan, Rajesh Jayaprakash, and Anand Sivasubramaniam. 2018. Good Set-Points Make Good Neighbors: User Seating and Temperature Control in Uberized Workspaces. In *Proceedings of the 5th Conference on Systems for Built Environments* (Shenzen, China) *(BuildSys '18)*. Association for Computing Machinery, New York, NY, USA, 144–147. https://doi.org/10.1145/3276774.3276781
[67] Balakrishnan Narayanaswamy, Bharathan Balaji, Rajesh Gupta, and Yuvraj Agarwal. 2014. Data Driven Investigation of Faults in HVAC Systems with Model, Cluster and Compare (MCC). In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings* (Memphis, Tennessee) *(BuildSys '14)*. Association for Computing Machinery, New York, NY, USA, 50–59. https://doi.org/10.1145/2674061.2674067
[68] Michele Nati, Alexander Gluhak, Hamidreza Abangar, and William Headley. 2013. SmartCampus: A user-centric testbed for Internet of Things experimentation. In *2013 16th International Symposium on Wireless Personal Multimedia Communications (WPMC)*. WPMC, New York, NY, USA, 1–6.
[69] Carman Neustaedter and Phoebe Sengers. 2012. Autobiographical Design in HCI Research: Designing and Learning through Use-It-Yourself. In *Proceedings of the Designing Interactive Systems Conference* (Newcastle Upon Tyne, United Kingdom) *(DIS '12)*. Association for Computing Machinery, New York, NY, USA, 514–523. https://doi.org/10.1145/2317956.2318034
[70] Node web developers. 2023. Node.js. https://github.com/nodejs/node.
[71] Notion. 2023. Notion DIY Smart Monitoring System. https://getnotion.com/.
[72] Temitope Oluwafemi, Tadayoshi Kohno, Sidhant Gupta, and Shwetak Patel. 2013. Experimental Security Analyses of Non-Networked Compact Fluorescent Lamps: A Case Study of Home Automation Security. In *LASER 2013 (LASER 2013)*. USENIX Association, Arlington, VA, 13–24. https://www.usenix.org/laser2013/program/oluwafemi
[73] Kumar Padmanabh, Adi Malikarjuna, Sougata Sen, Siva Prasad Katru, Amrit Kumar, Sai Pawankumar C, Sunil Kumar Vuppala, and Sanjoy Paul. 2009. ISense: A Wireless Sensor Network Based Conference Room Management System. In *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings* (Berkeley, California) *(BuildSys '09)*. Association for Computing Machinery, New York, NY, USA, 37–42. https://doi.org/10.1145/1810279.1810288
[74] Shijia Pan, Tong Yu, Mostafa Mirshekari, Jonathon Fagert, Amelie Bonde, Ole J. Mengshoel, Hae Young Noh, and Pei Zhang. 2017. FootprintID: Indoor Pedestrian Identification through Ambient Structural Vibration Sensing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 3, Article 89 (sep 2017), 31 pages. https://doi.org/10.1145/3130954
[75] Panasonic. 2023. AMG8853 - Infrared Array Sensor GridEYE - Built-in Sensors - Industrial Devices & Solutions - Panasonic. https://industrial.panasonic.com/ww/products/pt/grid-eye/models/AMG8853.
[76] Particle. 2023. IoT Environmental Monitoring Solutions | Particle. https://www.particle.io/solutions/iot-environmental-monitoring/.
[77] Particle. 2023. Particle - Welcome to real IoT. https://particle.io/.
[78] Particle. 2023. Particle Firmware. https://github.com/particle-iot/device-os.
[79] Particle. 2023. Particle Spark Protocol. https://github.com/particle-iot/spark-protocol.
[80] Phidgets Inc. 2023. Phidgets Inc. - Products for USB Sensing and Control. https://www.phidgets.com.
[81] PTC. 2023. ThingWorx: Industrial IoT Software | IIoT Platform | PTC. https://www.ptc.com/en/products/thingworx.
[82] Haroon Rashid, Nipun Batra, and Pushpendra Singh. 2018. Rimor: Towards Identifying Anomalous Appliances in Buildings. In *Proceedings of the 5th Conference on Systems for Built Environments* (Shenzen, China) *(BuildSys '18)*. Association for Computing Machinery, New York, NY, USA, 33–42. https://doi.org/10.1145/3276774.3276797
[83] Haroon Rashid, Nipun Batra, and Pushpendra Singh. 2018. Rimor: Towards Identifying Anomalous Appliances in Buildings. In *Proceedings of the 5th Conference on Systems for Built Environments* (Shenzen, China) *(BuildSys '18)*. Association for Computing Machinery, New York, NY, USA, 33–42. https://doi.org/10.1145/3276774.3276797
[84] Raspberry Pi. 2023. Buy a Sense HAT – Raspberry Pi. https://www.raspberrypi.com/products/sense-hat/.
[85] Raytac. 2023. Raytac MDBT40-P256V3 Nordic nRF51822 Bluetooth Module. https://www.raytac.com/product/ins.php?index_id=66.
[86] Vladimir Risojević, Robert Rozman, Ratko Pilipović, Rok Češnovar, and Patricio Bulić. 2018. Accurate Indoor Sound Level Measurement on a Low-Power and Low-Cost Wireless Sensor Node. *Sensors* 18, 7 (2018), 123–132. https://doi.org/10.3390/s18072351
[87] Alex Rogers, Siddhartha Ghosh, Reuben Wilcock, and Nicholas R. Jennings. 2013. A Scalable Low-Cost Solution to Provide Personalised Home Heating Advice to Households. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings* (Roma, Italy) *(BuildSys'13)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/2528282.2528284
[88] A. Rowe, M. E. Berges, G. Bhatia, E. Goldman, R. Rajkumar, J. H. Garrett, J. M. F. Moura, and L. Soibelman. 2011. Sensor Andrew: Large-scale campus-wide sensing and actuation. *IBM Journal of Research and Development* 55, 1.2 (2011), 6:1–6:14. https://doi.org/10.1147/JRD.2010.2089662
[89] Luis Sanchez, Luis Muñoz, Jose Antonio Galache, Pablo Sotres, Juan R Santana, Veronica Gutierrez, Rajiv Ramdhany, Alex Gluhak, Srdjan Krco, Evangelos Theodoridis, et al. 2014. SmartSantander: IoT experimentation over a smart city testbed. *Computer Networks* 61

(2014), 217–238.
[90] Sense.se. 2023. Sen.se Mother. The Universal Monitoring Solution. https://www.pcmag.com/reviews/sense-mother.
[91] Oliver Shih, Patrick Lazik, and Anthony Rowe. 2016. AURES: A Wide-Band Ultrasonic Occupancy Sensing Platform. In *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments* (Palo Alto, CA, USA) *(BuildSys '16)*. Association for Computing Machinery, New York, NY, USA, 157–166. https://doi.org/10.1145/2993422.2993580
[92] Siemens Developers. 2023. Siemens. https://www.siemens.com/global/en.html.
[93] Silvertel. 2023. Ag9900 – Silvertel | Power Over Ethernet Modules | Telecom Modules. https://silvertel.com/9900-2/.
[94] TDK InvenSense. 2023. MPU-6500. https://www.digikey.com/en/products/detail/tdk-invensense/MPU-6500/4385413.
[95] Texas Instruments. 2023. TIDC- SENSORTAG. https://www.ti.com/tool/TIDC-CC2650STK-SENSORTAG.
[96] The Huntington News. 2023. NU administration removes occupancy sensors in ISEC in response to privacy, ethical concerns. https://huntnewsnu.com/69260/campus/nu-administration-removes-occupancy-sensors-in-isec-in-response-to-privacy-ethical-concerns/.
[97] Tom Torfs, Tom Sterken, Steven Brebels, Juan Santana, Richard van den Hoven, Vincent Spiering, Nicolas Bertsch, Davide Trapani, and Daniele Zonta. 2012. Low power wireless sensor network for building monitoring. *IEEE Sensors Journal* 13, 3 (2012), 909–915.
[98] Ikram Ud Din, Mohsen Guizani, Suhaidi Hassan, Byung-Seo Kim, Muhammad Khurram Khan, Mohammed Atiquzzaman, and Syed Hassan Ahmed. 2019. The Internet of Things: A Review of Enabled Technologies and Future Challenges. *IEEE Access* 7 (2019), 7606–7640. https://doi.org/10.1109/ACCESS.2018.2886601
[99] VergeSense Developers. 2023. Home | VergeSense Great decisions require great data. https://vergesense.com/.
[100] Patrick Verkaik, Yuvraj Agarwal, Rajesh Gupta, and Alex C. Snoeren. 2009. Softspeak: Making VoIP Play Well in Existing 802.11 Deployments. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation* (Boston, Massachusetts) *(NSDI'09)*. USENIX Association, USA, 409–422.
[101] Nicolas Villar, James Scott, Steve Hodges, Kerry Hammil, and Colin Miller. 2012. . net gadgeteer: A platform for custom devices. In *International Conference on Pervasive Computing*. Springer, Springer, Berlin, Heidelberg, 216–233.
[102] Vue.js developers. 2023. Vue.js - The Progressive JavaScript Framework | Vue.js. https://vuejs.org/guide/introduction.html.
[103] WELL. 2023. WELL | IWBI. https://www.wellcertified.com/.
[104] WELL. 2023. WELL PERFORMANCE RATING | Acoustic Performance. https://v2.wellcertified.com/en/performance-rating/acoustic%20performance/feature/1.
[105] Teng Xu, James B. Wendt, and Miodrag Potkonjak. 2014. Security of IoT systems: Design challenges and opportunities. In *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, San Jose, California, 417–423. https://doi.org/10.1109/ICCAD.2014.7001385
[106] Liang Yang, Zimu Zheng, Jingting Sun, Dan Wang, and Xin Li. 2018. A Domain-Assisted Data Driven Model for Thermal Comfort Prediction in Buildings. In *Proceedings of the Ninth International Conference on Future Energy Systems* (Karlsruhe, Germany) *(e-Energy '18)*. Association for Computing Machinery, New York, NY, USA, 271–276. https://doi.org/10.1145/3208903.3208914
[107] Tianlong Yu, Vyas Sekar, Srinivasan Seshan, Yuvraj Agarwal, and Chenren Xu. 2015. Handling a Trillion (Unfixable) Flaws on a Billion Devices: Rethinking Network Security for the Internet-of-Things. In *Proceedings of the 14th ACM Workshop on Hot Topics in Networks* (Philadelphia, PA, USA) *(HotNets-XIV)*. Association for Computing Machinery, New York, NY, USA, Article 5, 7 pages. https://doi.org/10.1145/2834050.2834095
[108] Shaila Zaman, Amanveer Wesley, Dennis Rodrigo Da Cunha Silva, Pradeep Buddharaju, Fatema Akbar, Ge Gao, Gloria Mark, Ricardo Gutierrez-Osuna, and Ioannis Pavlidis. 2019. Stress and productivity patterns of interrupted, synergistic, and antagonistic office activities. *Scientific data* 6, 1 (2019), 1–18.
[109] Ni Zhu, Tom Diethe, Massimo Camplani, Lili Tao, Alison Burrows, Niall Twomey, Dritan Kaleshi, Majid Mirmehdi, Peter Flach, and Ian Craddock. 2015. Bridging e-Health and the Internet of Things: The SPHERE Project. *IEEE Intelligent Systems* 30, 4 (July 2015), 39–46. https://doi.org/10.1109/MIS.2015.57
[110] Jan Henrik Ziegeldorf, Oscar Garcia Morchon, and Klaus Wehrle. 2014. Privacy in the Internet of Things: threats and challenges. *Security and Communication Networks* 7, 12 (2014), 2728–2742.