Context Aware System Design
Christine S Chan, Michael H Ostertag, Alper Sinan Akyürek, and Tajana Šimunić Rosing
UC San Diego, 9500 Gilman Drive, La Jolla, CA, USA

ABSTRACT
The Internet of Things envisions a web-connected infrastructure of billions of sensors and actuation devices. However, the current state-of-the-art presents another reality: monolithic end-to-end applications tightly coupled to a limited set of sensors and actuators. Growing such applications with new devices or behaviors, or extending the existing infrastructure with new applications, involves redesign and redeployment. We instead propose a modular approach to these applications, breaking them into an equivalent set of functional units (context engines) whose input/output transformations are driven by general-purpose machine learning, demonstrating an improvement in compute redundancy and computational complexity with minimal impact on accuracy. In conjunction with formal data specifications, or ontologies, we can replace application-specific implementations with a composition of context engines that use common statistical learning to generate output, thus improving context reuse. We implement interconnected context-aware applications using our approach, extracting user context from sensors in both healthcare and grid applications. We compare our infrastructure to single-stage monolithic implementations with single-point communications between sensor nodes and the cloud servers, demonstrating a reduction in combined system energy by 22-45%, and multiplying the battery lifetime of power-constrained devices by at least 22x, with easy deployment across different architectures and devices.

Keywords: Context-aware, Internet of Things, Connected health, User modeling, Scalable applications

1. INTRODUCTION
What happens if the Internet of Things (IoT) is fully realized? Market researchers speculate that there will be anywhere between 25 billion\(^1\) to 50 billion devices\(^2\) globally by 2020. That translates to about 2.3 ZB of fresh data produced annually\(^2\)! As the number of devices and volume of data explode, resources that we take for granted today will become highly contended - radio spectrum, communication bandwidth, even the processing capabilities of the cloud.\(^3\) The current strategies that aggregate all data to the cloud for processing will not scale for the future IoT. The pervasive sensing and scope of data promised by the IoT promises pervasive sensing of our daily life and an unprecedented scope of data, but raw health and environmental data collected is often difficult for an average citizen to interpret.

Our overarching goal is to enable both citizens and researchers to collect sensing data and draw actionable insights. To that end, we design and utilize a *context engine* framework\(^4\) and show its application in environmental and smart grid applications. This work presents the versatile deployment of a modular context extraction framework (*context engines*) on interconnected devices of various processing capabilities - from microcontrollers to edge servers. Context engines are provably scalable in terms of reducing the volume of training data required for large datasets\(^5\) - we will also show the energy impact of delegating the same tasks to different machines, and trade offs between execution speed and battery lifetimes.

To fully leverage the opportunity that ubiquitous data, cheap computation devices and vast Internet connectivity provides us, we must develop new ways for devices and software to interact and share valuable insights. Some environmental sensing procedures are still very labor intensive - for example, the daily pollen count report involves a human’s visual identification of particles on a collected data sample.\(^6\) It also takes time and resources to collect larger datasets about historical, population-level statistics. How does our vast collection of personal

---

Further author information:
Tajana S Rosing. E-mail: tajana@ucsd.edu.
data inform population health knowledge and vice versa? How can we use machine learning beyond physician expertise? How can new software infrastructure run efficiently on the diverse range of devices in the IoT? Since we hope to reduce redundant efforts and share data between applications, we take advantage of publicly available data sources to learn about demographics, city planning data, citizen habits, and environmental data, and how that implicates respiratory health. For example, to investigate asthma risks, ER admissions and hospitalization rates are used to model asthma complications.

Popular mobile consumer-level products that enable crowd-sourced air pollution monitoring and smart meters for homes provide an unprecedented level of sensing granularity. Specifically, we will discuss how our system performs machine learning techniques combining air quality data and other personal context to produce real-time feedback on a user’s local environment and personalized health impacts. The sample applications we present can be fully fleshed out to enable citizens and researchers to track environmental context and obtain actionable data. In smart grids, our work incorporating knowledge of residential user behavior into the actuation of smart appliances has shown a potential 12% in annual grid energy costs. In smart health, we will discuss how the strategic deployment of a hierarchical sensing and health modeling application can deliver personalized health information to individual users with flexibility and energy efficiency. With the diverse budgetary and practical deployment constraints to consider, we can show a reduction in combined system energy by 22-45%, and multiplying the battery lifetime of power-constrained devices by at least 22x. Then, citizens can receive real-time feedback on environmental health impact, doctors easily and quickly test hypotheses for suspected correlations between environment and symptoms, while technology providers can gain insights from huge datasets collected to target application deployment and maintenance of diverse hardware.

2. RELATED WORK

The IoT was initially associated with simply providing value-add to user convenience products and commercial advertising. The public sector was one of the first big industries to adopt IoT technologies, widely deploying systems for the smart grid and environmental monitoring to name a few. We will summarize both aspects in the following sections. But with increasing adoption comes increasing loads and responsibilities to the user base, such that the underlying system infrastructure must be engineered to meet performance, reliability and safety constraints. Publications exist on various aspects of energy management in a power grid such as building demand response and generation control. For residential control, previous studies include appliance automation, lighting, appliances, renewables, and energy storage. Neighborhood level control with energy storage or renewable energy sources attempts to minimize energy costs.

Many researchers have utilized machine learning and data mining techniques to draw new insight from large studies of geographical models. Some have proven hypotheses about the effects of urbanization on asthma hospitalization rates or the link between air pollutants vs. asthma incidence. The DELPHI project is a large-scale integration of electronic health records (EHR), environmental data, medication usage and physical activity monitors. They present a specific personalized dashboard that summarizes this data and displays relevant data any specific patient. This is a quickly developing area. As a field, we have some understanding of correlations between environmental/geographical factors for widespread conditions like asthma, but are notably missing automated system for individual patients to incorporate this data in daily life.

As sensor networks and context-aware computing become increasingly flexible and personalized, much effort has been made towards empowering citizens to monitor and analyze their own environment. Healthcare or fitness applications in the IoT require hardware and software systems for crowd sourced air quality monitoring leading to more comprehensive studies of populations and geographic factors in public health. Wearable pollutant monitors have seen success in the market, whether for personal or industrial use - the degree of connectivity varies. And to effectively integrate sensed data from various sources, many communication protocols have been researched for maximum usability and efficiency, such as the pub/sub model for crowd sourcing, and aggregation inside the network.

There are many ideas and prototypes for crowd sensing, but precious little validation, so we are motivated to find the connection to higher-level actionable health information. Electrochemical sensors are a cost efficient solution for gas measurement, with price tags on the order of hundreds of US dollars as compared to those
used at regulatory stations, which cost tens of thousands of dollars. Sensor readings are calibrated and verified against trusted regulatory station readings as a golden standard. The EPA includes measurements of multiple air pollutants in the definition of the air quality index (AQI). The first stage of this work produced a mobile modular sensor platform that detects air pollution indicators including nitrogen dioxide, ozone and carbon monoxide at a parts-per-million (ppm) granularity, reporting either batched or real-time results directly to users via a mobile application and aggregated all data to the Internet. Our proposed platform has increased processing power such that it can execute some machine learning algorithms, and can be paired with an IoT gateway or directly upload data via Wi-Fi to the cloud. In the interest of maintaining scalable data volumes and communication bandwidth, we will implement and study the opportunities for intelligent processing natively on the sensor node, instead of naively aggregating all data in the cloud.

3. CONTEXT MIDDLEWARE: THE CONTEXT ENGINE

We envision a world in which intelligence is present at every layer of device hierarchy. Smart sensors will not only sense the physical world around them, but also provide data management and analysis capabilities (e.g. detecting anomalies). IoT gateway devices will gather the data from relevant sensors in their surroundings and derive context for ever changing applications. In contrast to today’s systems, data analysis will occur at each layer in our system, and therefore demands novel and scalable software infrastructure, machine learning algorithms that can leverage data across distributed computing nodes, privacy strategies that are sensitive to individual privacy needs while providing sufficient utility across the hierarchy of devices, and communication methods that can effectively aggregate data to handle inevitable congestion while ensuring application data timeliness needs are met.

Under the old paradigm, silo-ed, standalone applications collect and process all the data they rely on, resulting in redundant computations across independent applications. Considering the expansive growth of data and wide range of computation nodes available, we propose a more scalable way to develop applications across the ecosystem. In our new paradigm (Figure 1), we have identified an improved approach: multiple general-purpose functional units (context engines) that each drive data processing for a single output context variable, recomposed to be functionally equivalent to conventional monolithic application. Each low-level sensor is only sampled from minimal times, and once their raw data is processed into higher-level information (e.g. from raw GPS or radio signals to semantic “locations”), that information can be shared instead of redundantly computed.

As Figure 1 shows, the system operates as a hierarchy of multiple-input-single-output (MISO) context engine units to improve machine-learning reasoning while reducing data redundancy and accomplishing the same functionality as the corresponding state-of-the-art multi-input-multi-output (MIMO) applications. Smaller hierarchical functional units represent simpler data translation at the tradeoff of more functional units. This promotes the use of general data transformation in each context engine using machine learning to generate outputs in place of application-specific code. We showed the theoretical arguments for the scalability of this composition in terms of training data required and machine learning execution time in previous papers. Others have tested and measured the feasibility of executing traditional machine learning algorithms on resource constrained devices. The energy and delay of both linear-and non-linear algorithms can be automatically determined, based largely on network bandwidth and data size.

Once the context has been generated, we aggregate data prior to communicating it to the next layer in our system hierarchy by using our novel data aggregation scheme which ensures an optimal tradeoff between meeting application timeliness needs while minimizing communication costs. Thus, a context-aware application can be created by simply specifying the inputs and output of each functional unit alone, and allowing hierarchical machine learning to generate and train a model based on input and output observations. Exposing intermediate data reduces the complexity and redundancy of applications in the larger infrastructure, and enables easy data sharing among other engines. Creating such a hierarchy of context engines is very beneficial for distributed data processing, as it makes it very easy to decompose a large application into sets of context engines that can each run on various devices present in the system. Because context is derived and generated at each layer, the total amount of data sent is dramatically reduced as compared to today’s state of the art, thus making such large-scale deployments not only possible but also much more efficient.
If they participate in the context engine ecosystem, each device should share their processed data, which may serve as further input data into other applications. For example, air quality sensor data has a multitude of uses. It can be used simply for local display, or to inform the actuation of building ventilation, or for regional environmental studies. In traditional application design, each programmer would have to set up data sampling and processing of the analog electrochemical sensor voltage levels into human readable gas concentration values. In a context engine ecosystem, only the first programmer would have to put in that effort - consequent applications can simply share that higher level context.

Since we have access to small, low power programmable devices with some (albeit limited) processing power, context engines are one opportunity to execute generalized machine learning at the edge of the cloud. They can be used to provide timely feedback to users in the field, without the need to send all raw data to the cloud, and wait for processing, and the return delivery of useful information. These techniques can also enable these low power devices to manage their own sensing behavior, such as down sampling intelligently without missing critical data, thus saving battery power and restricting power-hungry data transmissions to the cloud.

4. CONTEXT-AWARE SMART GRID APPLICATION AND RESULTS

Our earlier work exploited the fact that electricity delivery systems are being equipped with smart devices at all levels as part of the IoT (e.g. sensors and actuators in junction boxes, individual homes, etc). In order to evaluate the efficacy of our strategies in different geographies and demographics, to provide a starting point for control in homes which do not yet have a context-extraction capability, and to enhance capabilities in homes which only have partial context available (e.g. smart meter data and only one users context out of a family of four), we also enhance our context engine with a capability to distill context from more general data sources, e.g. American Time of Use Survey (ATUS), Residential Energy Consumption Survey (RECS), California Household Travel Survey (CHTS), or similar surveys from around the world.

Using such data sources, we design a system capable of modeling daily activities of different individuals with activity graphs, from which we derive the corresponding energy usage with flexibility regions, which are provided as inputs into user-context-aware control policies. It can operate on both real-time data and from...
statistical data inputs, such as survey data, and send signals to actuate both stationary and mobile smart loads. Our system divides the total tasks into individual working nodes and a single aggregator to process sensed data and control signals for a residential neighborhood, showing that modeling user context brings over 14% improvement in energy flexibility prediction accuracy and 12% reduction in annual grid energy cost. In this way, the context engine can be used to gather data from heterogeneous sources (physical and virtual) not only to display information, but to provide control back to the grid.

5. CONTEXT-AWARE SMART HEALTH APPLICATION

Similarly, in healthcare settings, data is collected from and aggregated on various devices in a hierarchy, including wearable monitors, environmental sensors, in-home care equipment, and larger electronic health records (EHR). Heightened privacy concerns in healthcare settings further motivate the need for allocating tasks to nodes close to the edge of the cloud, limiting the communication and exposure of sensitive medical data.

In fact, the context engine framework is general enough that the same coding infrastructure used for our previous smart grid work can be easily instantiated for a healthcare application. For example, a context engine running on an air quality sensor nodes works to detect anomalies in continuous air quality data. The anomalies can then be collected by the patients cell phone, running an intermediate content engine, that will correlate air quality anomalies with physical activity and respiratory metrics to learn when asthma related problems occur. At larger scale, a context engine running in the cloud may use data from many individuals context engine outputs to gather correlations between asthma, geographic location, air quality and physical activity. Meanwhile, the intermediate data generated can be shared for other uses, reducing the computational load for those applications.

5.1 Population health

At a population level, hospitalization rates serve as an indicator for health, and the primary diagnosis for any particular emergency room admission or hospitalization represents the overall risk that a population has towards that condition. We have preliminary results based on annual county-level data from the EPA and California Department of Public Health. The EPA data includes daily pollutant averages and maximum hourly counts, as well as temperature and humidity data, for each county in California. The demographic data includes hospital and emergency room asthma admission rates per 10,000 persons, for each county annually, as well as the percentage of residents who live within 0.5 miles of a public park, and their commute habits. The pollutant count and demographic data show linear correlations for the concentration of pollutants such as CO and Ozone and the likelihood that more citizens will be admitted to the ER for asthma complications. On the other hand, the percentage of citizens who live within 0.5 mile of a park is negatively correlated with the county’s rate of hospital admissions for asthma. The model goes beyond just linear relationships - it includes up to 2nd or 3rd order cross-dependencies between different input variables (depending on the amount of public data available).
Based on this model, for any given environmental sample of gas pollutants and related cross-dependencies (temperature, humidity), we predict the related asthma risk. We divide the dataset into 10 subsets for cross-validation of the regression model. Currently, using various subsets of the environmental data, we can predict a county’s hospital and emergency room admission rate for asthma with 25-35% error. This is useful to test hypotheses on the general population by leveraging existing historical data, without instrumentation. Once correlations are identified between variables of interest, then we can refine the top-down model with personal details as they are made incrementally available. Respiratory sensitivity varies widely among citizens even asthma patients can have different triggers depending on their vulnerabilities. There is value in helping each individual track and learn how their condition responds to acute triggers (e.g. known allergens) as well as their vulnerability according to daily air exposure. Going forward, we will also move into monthly and seasonal data. We also plan to include data from more census and city planning sources, such as what fraction of the population lives or works in proximity to highways, industrial areas, bus depots, or trains.

5.2 Personal health

With the population health model, citizens can now have a idea of what environmental factors generally have effects on respiratory health. However, they cannot apply this knowledge to their daily lives until they have 1) up-to-date data about their immediate environment (e.g. carbon monoxide concentration in the surrounding city block is 2 parts per million), 2) a translation from that current, local data into a quantifiable health-related terms (e.g. CO 2 ppm is a safe level of air quality, no action needed), and possibly 3) personalized factors such as allergies, medical history, or level of aerobic activity. Such a personal application should give insights on whether the user is going through typical exposure, or may need to be alerted to sudden and unexpected changes in their environment.

Since each person’s health and activities vary, it is important for us to automatically sample and extract their behavioral context. For such an application, we can split up the tasks in order of:

1. Environmental raw sampling: Although the Alphasense electrochemical gas sensors we use have some temperature adjustment, calculating the cross dependencies with other changing environmental factors, including temperature, humidity, barometric pressure, and other gases) is non-trivial, requiring at least 2nd order level polynomials with high-precision coefficients.

2. Sensor data summarization and anomaly detection: To reduce the amount of transmissions from sensor nodes to collector nodes in the network, we can use anomaly detection techniques to filter out “typical” samples from “anomalous” ones. We use a SMCTC-based\textsuperscript{45} particle filter to predict the expected value according to an irregular time series stream of air quality values. If an incoming sensing sample deviates from the expectation, it is labeled as an anomaly. The results can be used to reduce sensor power consumption and communication congestion by down sampling whenever samples are “typical” instead of “anomalous”.

3. Nonlinear modeling of health risk for patient feedback:

Finally, we build and update Taylor-expansion based nonlinear regression models (TESLA)\textsuperscript{46} to learn which environmental factors have the highest correlation with asthma incidents at a population level, as described in the previous section.

In the remaining sections, we will describe the experimental setup for evaluating several variations of this application implementation on a hierarchy of devices. we will describe a specific sensor node design that is capable of collecting multiple environmental samples and processing some data.

6. SMART HEALTH EXPERIMENTAL SETUP

We roughly categorize the available system into three different types of compute nodes, in order of increasing resources: sensor nodes, gateways and edge servers. In our testbed, they are represented by the air quality sensor platform (Particle Photon system-on-chip with ARM Cortex M3), Raspberry Pi 3 Model 3 (Broadcom system-on-chip with ARM Cortex A53)\textsuperscript{47} and an edge server (similar to Intel Xeon D Processor\textsuperscript{48} or Dell Edge...
Gateway 5000\textsuperscript{[49]). The three application subtasks will be delegated to different devices. The sampling task is uniquely executed on the air quality sensor. The modeling task is an aggregation function and must run on an IoT gateway or server.

6.1 Edge server and gateways
The Raspberry Pi 3 (RPi) has a Broadcom BCM2837 SoC, where the processor runs at 1.2 GHz by default, and the typical power consumption we have observed is between 1.2W - 2.5W depending on utilization, and up to 3.5W when communicating with Wi-Fi. The edge server we used runs at 2.6 GHz and consumes between 12W - 45W depending on the utilization, and is constantly connected to the Internet via Wi-Fi. The air quality board communicates with gateways via Bluetooth, or directly with an edge server using Wi-Fi. The RPi has built in Bluetooth and Wi-Fi capabilities, we assign it to communicate with the edge server via Wi-Fi. The RPi consumes 2.9-3.5W when using Wi-Fi\textsuperscript{[37]}, while the AQ board consumes 100mW when transmitting on BLE, and 150mW when transmitting on Wi-Fi.

6.2 Sensor Node: Modular Air Quality Sensing
As a test platform, we developed an air quality sensing board, which served as a context-generating sensor where context is derived from local environmental conditions (e.g. pollutants used as air quality indicators, temperature, humidity). The platform was designed with the primary goal of accurately sensing pollutants while maintaining modularity and extensibility, allowing end users and researchers to reconfigure the board and replace modules as required to tailor the functionality to their specific end needs. The result is a platform that can interface with various sensing modalities using standard communication protocols (I2C, SPI, analog, UART, USB, and BLE) and process the data with an ARM Cortex M3 microprocessor.

![Sensor Node Diagram](image)

Figure 4. Left: the AQ board with default Alphasense sensors, supporting SHT11 humidity and MS5440C pressure sensors built-in, and additional extension headers. Right: AQ board with additional sensors, optional Bluetooth module and alternative processor.

6.2.1 Sensors
The air quality sensing platform can interface with any 3.3 or 5.0 V sensor that communicates using I2C, SPI, analog, or UART. The interface options permit a variety of sensing modalities to be used. Our configuration utilized electrochemical sensors for traditional air quality indicators ($\text{NO}_2$, $\text{CO}$, $\text{O}_3$), nondispersive infrared sensors for CO$_2$, photoionization detectors for volatile organic compounds (VOCs), and a variety of environmental sensors (temperature, humidity, barometric pressure). A list of tested sensors is shown in Table 1.
Table 1. Sensing elements of the air quality sensing platform, including built-in and optional extensions.

<table>
<thead>
<tr>
<th>Sensor Name</th>
<th>Measured element</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensiron SHT11</td>
<td>Humidity</td>
<td>Proprietary 2-wire</td>
</tr>
<tr>
<td>Sensiron SHT11</td>
<td>Temperature</td>
<td>Proprietary 2-wire</td>
</tr>
<tr>
<td>TE Connectivity MS5540C</td>
<td>Barometric pressure</td>
<td>SPI</td>
</tr>
<tr>
<td>TE Connectivity MS5540C</td>
<td>Temperature</td>
<td>SPI</td>
</tr>
<tr>
<td>Alphasense NO2-A43F</td>
<td>Nitrogen dioxide (NO$_2$)</td>
<td>Analog</td>
</tr>
<tr>
<td>Alphasense O3-A431</td>
<td>Ozone (O$_3$)</td>
<td>Analog</td>
</tr>
<tr>
<td>Alphasense CO-A4</td>
<td>Carbon Monoxide (CO)</td>
<td>Analog</td>
</tr>
<tr>
<td>Alphasense PT1000</td>
<td>Temperature</td>
<td>Analog</td>
</tr>
<tr>
<td>Telaire T6713</td>
<td>Carbon Dioxide (CO$_2$)</td>
<td>I2C</td>
</tr>
<tr>
<td>Mocon pID-TECH eVx</td>
<td>Volatile Organic Compounds</td>
<td>Analog</td>
</tr>
<tr>
<td>ams iAQ-Core</td>
<td>Volatile Organic Compounds</td>
<td>I2C</td>
</tr>
<tr>
<td>STMicroelectronics STC3105</td>
<td>Power Consumption</td>
<td>I2C</td>
</tr>
</tbody>
</table>

The base configuration of the platform uses three electrochemical sensors for monitoring the traditional air quality indicators, which include NO$_2$ (Alphasense NO2-A43F), O$_3$ (Alphasense O3-A431), and CO (Alphasense CO-A4). These chemicals are defined in the US National Ambient Air Quality Standards as part of the Clean Air Act. While these three sensors were selected for our application, Alphasense makes a variety of other pin compatible sensing elements for hydrogen sulfide (H$_2$O) from sewage treatment plants, nitric oxide (NO) from automobile emissions, and sulfur dioxide (SO$_2$) from coal power plants. The sensors are mounted to a companion analog front end (AFE) from Alphasense, which assists with voltage regulation and signal amplification. Electrochemical sensors offer a high level of accuracy at a low current consumption. Each sensing element has two electrodes which give analog outputs for the working and auxiliary electrodes. The difference in signals is approximately linear with respect to the gas concentration detected but have dependencies with temperature, humidity, barometric pressure, and cross-sensitivities with other gases. By accurately measuring the environmental conditions, the effect of these dependencies on the measured signal can be reduced. Due to the kinetics of the sensing mechanism, the gas sensors have a long required warm up period that prevents systems from power cycling for energy savings. In our experience, a warmup period of 30 minutes is sufficient to reach valid values. After which, the sensing elements will respond to a change in environmental conditions with a 60, 45, and 20 s response time for NO$_2$, O$_3$, and CO, respectively.

The environmental sensors (MS5540C and SHT11) are important for correcting the environmentally related offset in electrochemical sensor readings and for accurately measuring temperature, humidity, and pressure, which can affect medical conditions. The sensors self-calibrate and are able to deliver accurate readings after a short delay at startup. The TE Connectivity MS5540C is a barometric pressure sensor capable of measuring across a 10 to 1100 mbar range with 0.1 mbar resolution. Across 0°C to 50°C, the sensor is accurate to within ±1 mbar and has a typical drift of −1 mbar per year. The Sensiron SHT11 is a relative humidity sensor capable of measuring across the full range of relative humidity (0 to 100% RH) with a 0.05% RH resolution. Both sensors come equipped with temperature sensors with ±0.8°C and ±0.4°C accuracy, respectively. The sensors stabilize to environmental changes in under 30 seconds, which is sufficiently fast to capture changes in the local environment.

All sensing elements rely on a passive diffusion mechanism for pollutants to reach the sensing elements. While an active air flow rate can result in faster response times and potentially more accurate readings, an exhaust fan has prohibitively high power consumption for a mobile, embedded platform. To study the effect of air flow and enclosure design, we are in the process of producing multiple 3D printed enclosures that promote different levels and directions of air flow for comparative testing.
The electrochemical sensors generate an analog output, which is connected through a header to a pair of ADCs (TI ADS6115). The 16-bit ADC has a programmable gain amplifier before the conversion circuitry, allowing the effective full-scale range between differential inputs to vary between ±0.256 V and ±4.096 V. The ADC has a maximum full-scale range of ±6.144 V, but it cannot be fully utilized with the 5.0 V supply rail of the board. The adjustable gain is a useful feature that permits dynamic sampling resolution for capturing small variations of typically low signals and the high signals near pollution sources. The ADC is operated in a single-shot conversion mode where all channels are sampled sequentially in accordance with our desired sampling rate. The ADC has the ability to perform continuous conversions, but it is not practical to sample at 8 Hz when the required sampling rate is 1/5 to 1/30 Hz. By not continuously sampling, the average ADC power consumption drops from 150 µA to < 2µA.

The platform has headers for connecting additional sensors. In our configuration, total VOCs (TVOCs) and carbon dioxide were important aspects for quantifying indoor air pollution. VOCs are a major indicator of air quality in indoor environment and are outgassed from a variety of building materials, cleaning agents, paints, and microbial agents. Carbon dioxide concentrations have been linked to decreases in concentration and productivity and can be used to estimate occupancy. Both of these sensors (shown in Figure 4 connect to headers on the topside of the platform. These sensors were added to the air quality sensing platform with no design changes, showcasing the modularity of the design. Different sensors can be connected as required for end user applications.

Initial data was collected by co-locating a single sensor platform with a reference station at the Colorado Department of Health and Environment monitoring site. Using 2 weeks of minute-by-minute data, data analysis and verification of the gas chemistry model was performed. This calibration model was used in the conversions for all remaining boards. Depending on the mode of deployment and availability of other nearby sensors, we expect to adjust the model computation for each individual board based on reference data from either a regulatory air quality monitoring station, or a calibrated mobile sensor node. The mean standard deviation with direct emissions across 6 sensors over 2.75 hours: 27.8 ppb. Mean standard deviation in ambient environment without direct emissions in a 4.5 hr period: 8.3 ppb. We can adjust the ADC gains dynamically online, and update the model parameters through the processor.

### 6.2.2 Processing and communication

The air quality sensing platform is compatible with the Particle Photon and the Particle Electron. The Particle Photon runs on an ARM Cortex-M3 32-bit core at 24 MHz with a real-time clock (RTC), 1 MB flash, and 128 KB SRAM (ST Micro STM32F205). Application code is written in C++. It has a Wi-Fi module (Broadcom BCM43362) through which it can upload data to the internet and have its firmware updated wirelessly. The Broadcom Wi-Fi module has a single antenna connection and can support IEEE 802.11 b/g/n. The Particle Electron has the same ARM processor but is installed with a 3G cellular module instead of the Wi-Fi module. The U-bloc SARA-U260 and G350 chipsets provide 3G and 2G cellular connectivity, respectively. Although developed for direct interface with the Particle devices, Arduino boards could be substituted with minor additional wiring. Each of the Particle modules offer a secondary power connection to the RTC and SRAM, allowing the board to retain a correct timestamp and critical configuration values in low-power situations that may cause the microprocessor to reset.

The microprocessor collects readings directly from digital sensors and digitized values from the ADCs. It arbitrates the various buses and slaves and then compiles the data into a JSON-formatted message. Each sample record currently includes a full real-time timestamp, uptime of the board, and values from all sensors on board, but it can be updated to include power management statuses, calibrations, or contextual output from the on-board Context Engine. Messages are sent in ASCII over UART to the Bluetooth Low Energy (BLE) 4.0 module (HM-11), which wirelessly relays the message to a paired personal device. BLE is well suited to the air quality sensing application because it offers a secure, low-power communication channel. The low bandwidth of BLE is not a problem with the low sampling rate of the system. For stationary deployments, Wi-Fi and 3G protocols can be used to communicate to a backend server. The networks are prevalent within densely and moderately populated areas. While BLE is provided by default for communication, any communication module that communicates serially can be added to the communication extension header, such as the HC-06 Bluetooth 2.3 module for backwards compatibility.
In addition to sending data, the platform has a two-way interface with a mobile application on Android phones (currently available for Android 4.4+). JSON-formatted commands allow the user to remotely toggle whether data is stored locally on the SD card, change the sampling interval, adjust configuration parameters for the analog sensors, and update the conversion model on the fly.

### 6.2.3 Power

The air quality sensing platform can be operated in a stationary or mobile mode, drawing power from a rechargeable battery or wired USB connection, respectively. We equipped a 1800 mAh lithium-ion battery, but for a higher energy density, a primary cell option could be used if it connects to a standard 2-pin (2 mm spacing), keyed power connector. When connected to a powered USB line, the battery is recharged through an on-board battery charging IC. Topside LEDs serve as power and charging indicators.

In addition to a time-based sleep mode, the Photon can enter deep sleep mode and be woken up by hardware interrupts. On our platform, interrupts can be generated by the BLE module to signal when a nearby device has been connected or by programmable alert pins on the ADCs, so that the board may conserve energy by going into a low power mode until pollutant levels reach a desired threshold without manual polling.

Air quality measurements are low-frequency signals, varying on the timeframe of minutes and hours. Combining a low sampling rate with deep sleep mode on the microprocessor enables significant power savings. For reference across the operating temperature range of $-20^\circ\text{C}$ to $+60^\circ\text{C}$, the Particle Photon consumes $30\text{ mA}$ in normal operational mode with the Wi-Fi off, $1\text{ mA}$ in sleep mode, and $99\mu\text{A}$ in deep sleep mode.

The sampling rate can be adjusted for the required application (e.g., mobile sensing may require a longer battery life than a stationary deployment). On the scale of environmental sampling rates ($<1\text{ Hz}$), the $17\mu\text{s}$ wake up time does not affect performance. The smallest sampling period is defined by component hold times and bus delays, taking $290\text{ ms}$ to update readings from all sensors on board. The sampling rate is appropriate for monitoring ambient conditions, but further shortening the awake time can improve the operational lifetime of the device.

Our base configuration equipped with the Particle Photon has two primary methods of communication: BLE and Wi-Fi. The BLE module is based on the TI CC2541 chipset, which has current consumption of $15\text{ mA}$ and $8.5\text{ mA}$ when transmitting and receiving, respectively, but only $600\mu\text{A}$ in sleep mode. The Wi-Fi module has even higher power consumption at $80\text{ mA}$ average. Due to the low average power that is inherent to the BLE protocol, BLE can be in regular communication with nearby devices, but the high power cost of Wi-Fi prevents it from being regularly deployed in mobile settings. If it is needed, the Wi-Fi can be cycled on for short periods for burst communication to a backend database. If real-time measurements are not required in an application, data can be stored to a local SD card for later analysis, removing the need for higher power communication.

If VOC or CO$_2$ measurements are required for an application, the board should be configured into a stationary, plugged position due to the high current draws of the sensors. The VOC sensors pull $39.7\text{ mA}$ and the CO$_2$ sensor draws an additional $30.2\text{ mA}$ baseline with large spikes in current ($500$-$600\text{ mA}$ for $560\text{ ms}$) when the infrared lamp turns on. While the system can provide these current draws, the battery would not last long enough for sustained mobile deployment. A summary of the current consumption in different operational modes can be seen in Table 2.

Our platform currently consumes $192.4\text{ mW}$ during active sampling and processing, and $52.2\text{ mW}$ in deep sleep mode. With a $1.8\text{ Ahh}$ battery, a sensor node lasts $5.0$ days at a $5\text{ s}$ sampling interval and $5.6$ days at a $30\text{ s}$ sampling interval. This is a conservative characterization. Power efficiency improves by optimizing code use and aggressively power gating various modules on the board whenever possible. For reference, other commercially available air quality sensors, such as the AirBeam$^{29}$ and CairClip,$^{30}$ can only measure up to 10 and $36\text{ hrs}$ on a single charge, respectively.
Table 2. Power consumption in various operational modes using a 4.0 V source. The “Env Only” sensor configuration refers to the basic environmental sensors (humidity, temperature, and pressure) and the “AQI Sensors” refer to the air quality sensing platform with three electrochemical sensors (NO₂, O₃, and CO) and the environmental sensors.

<table>
<thead>
<tr>
<th>Microcontroller</th>
<th>Bluetooth</th>
<th>WiFi</th>
<th>Sensors</th>
<th>Power (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep Sleep</td>
<td>Off</td>
<td>Off</td>
<td>None</td>
<td>0.42</td>
</tr>
<tr>
<td></td>
<td>Deep Sleep</td>
<td>Off</td>
<td>Env Only</td>
<td>10.15</td>
</tr>
<tr>
<td></td>
<td>Deep Sleep</td>
<td>Off</td>
<td>AQI Sensors</td>
<td>52.19</td>
</tr>
<tr>
<td>Idling</td>
<td>Off</td>
<td>Off</td>
<td>None</td>
<td>144.20</td>
</tr>
<tr>
<td></td>
<td>Deep Sleep</td>
<td>Off</td>
<td>Env Only</td>
<td>153.26</td>
</tr>
<tr>
<td></td>
<td>Deep Sleep</td>
<td>Off</td>
<td>AQI Sensors</td>
<td>192.40</td>
</tr>
<tr>
<td>Active (24 MHz)</td>
<td>Connected</td>
<td>Off</td>
<td>Env Only</td>
<td>231.62</td>
</tr>
<tr>
<td></td>
<td>Connected</td>
<td>Off</td>
<td>AQI Sensors</td>
<td>249.92</td>
</tr>
<tr>
<td></td>
<td>Deep Sleep</td>
<td>Connected</td>
<td>AQI Sensors</td>
<td>519.94</td>
</tr>
</tbody>
</table>

7. SYSTEM EVALUATION AND RESULTS

We first evaluate the capabilities of each node for running the three subtasks in our health system (listed in Section 5.2. For the machine learning tasks, there are various configurations to choose from for accuracy and speed of execution. Figure 5 show the costs of executing tasks on various devices. In anomaly detection, we use a particle filter with \( l \) represents the number of iterations and \( p \) represents the number of particles. We observe that execution is always slowest on the AQ board and fastest on a server, but energy efficiency varies more.

In a real hierarchical system where context engines are delegated to various connected nodes, each sensor nodes, gateways and cloud servers would be running different workloads with different data sizes, requiring different communications.

![Figure 5](http://proceedings.spiedigitallibrary.org/)

Figure 5. For different configurations/data sizes: Total execution time for particle filtering (anomaly detection) and Taylor-expansion (health risk model) on different nodes. Note that y-axis is in log scale.

Next, we’ll consider this heterogeneous system, where sensor nodes are running smaller workloads than servers, but we also include data transmission costs. For anomaly detection on the limited AQ board, we choose the 100/100 configuration (100 particles and running 100 iterations for time series prediction) - that is a reasonable processor and memory usage to still allow for timely sampling and communication activities. For the particle filtering model running on the larger Raspberry Pi and server, we choose the 1000/100 configuration, accounting for the fact that they will be processing more diverse, larger sets of data. A third-order Taylor series is used in the non-linear model. Table 3 shows some examples from the spectrum of distributed computation vs. cloud aggregation. System 1 and 2 spread tasks between the AQ board and the Raspberry Pi only. System 3 and 5 use only the AQ board and the edge server. System 4 utilizes all three nodes.
In a simple two-node system, the AQ board collects all environmental samples and transmits them to the cloud (either through the gateway RPi over Bluetooth or directly to an edge server via WiFi), one of which will calculate the risk model. The Taylor-expansion based model requires second by second data, and is updated each time a new sample arrives. If the AQ Board does not perform anomaly detection, it must send out a new data sample every second (86400 samples per day) to support timely risk model updates. But if instead of transmitting all its data immediately, it is able to run anomaly detection locally, it may be able to batch its mundane data until something “interesting” shows up on the sensors. Each invocation of anomaly detection at the edge of the cloud may perform slower in an isolated speed test, but it is very useful for cutting down the amount of meaningless/redundant information that gets transmitted. In the case with anomaly detection available, we begin by setting a default sampling interval of 30 seconds (2880 samples per day). If an air quality sample deviates from its expected value, we set a higher sampling rate of 1 Hz. Thus, the risk model gets occasional “confirmation” sample readings at 30s intervals when data is not changing, and gets updated, detailed samples every second if new data arises that might change the model computation.

Table 3. Task distributions to context engines residing on each node

<table>
<thead>
<tr>
<th>Node:</th>
<th>Sensor board</th>
<th>Gateway</th>
<th>Edge server</th>
</tr>
</thead>
<tbody>
<tr>
<td>System 1</td>
<td>Sampling</td>
<td>Anomaly detection</td>
<td>Risk model</td>
</tr>
<tr>
<td>System 2</td>
<td>Sampling</td>
<td>Anomaly Detection</td>
<td>Risk model</td>
</tr>
<tr>
<td>System 3</td>
<td>Sampling</td>
<td>Anomaly detection</td>
<td>Risk model</td>
</tr>
<tr>
<td>System 4</td>
<td>Sampling</td>
<td>Anomaly detection</td>
<td>Risk model</td>
</tr>
<tr>
<td>System 5</td>
<td>Sampling</td>
<td>Anomaly Detection</td>
<td>Risk model</td>
</tr>
</tbody>
</table>

We estimate the total system energy across nodes, for different system configurations and different real-life sensing situations. Figure 6 shows the difference in energy consumption for these various task allocations, given different amounts of anomalies detected in a day. For example, “1 min daily” indicates that roughly a total 1 minute of air samples per day are anomalous and require higher sampling and transmission rates to the risk model. The more anomalies there are in a day, the more energy all systems must expend to process them - but it also widens the gap between systems that distributed workloads to smaller sensor nodes compared to those that aggregate in the cloud. To compare fairly among systems that still include the most powerful edge server (Systems 3-5), we see an energy difference of 21.5-44.35%.

Figure 6. The total daily energy consumption of nodes in the ecosystem, for different task allocations across nodes, and different levels of variation in actual air quality sampled
While the total energy consumption of an ecosystem certainly affects the bottom line for the application vendor, there are additional critical constraints to be met for devices at the “edge” of the cloud - such as battery life. In the far right columns of Figure 5, it seems that there is little difference between the energy consumption of Systems 4 and 5. But Figure 7 shows that the lifetime of a battery-fun sensor node in those systems can be dramatically different. The AQ board Photon processor and the Raspberry Pi may both be realistically run on portable batteries, and are often sold as such. Figure 7 shows a potential lifetime of each node for a full battery charge of 1800mAh and 10,000mAh respectively. As we observed before in Figure 5, the AQ board is not the most energy efficient at running machine learning code in isolation, but when considering the larger coordinated system, it greatly benefits from automatically down-sampling and going into sleep mode when inactive. Where the AQ board transmits via Wi-Fi (to edge server), 69-76% of battery life is consumed for communication. When it transmits data via Bluetooth (to Raspberry Pi 3), it consumes 24.1% by default, and 18.6% when down-sampling after anomaly detection. Comparing System 5 (sending all 1 Hz samples to the edge server) and System 1/3 (sending 30 Hz typical samples and 1 Hz anomalous samples to the Raspberry Pi/edge server), we can multiply the battery lifetime by 22-72x. The lower end of that battery lifetime increase can be achieved if the node eventually detects many anomalies (2 hours total daily), leading to fewer opportunities to down sample, but the higher end of the lifetime increase can be achieved in cases where most data collected is typical, such that only occasional samples have to be delivered to the cloud.

Additionally, using Wi-Fi to communicate from the sensor node to the cloud is more power-expensive than communicating via Bluetooth to a nearby gateway. By arranging the tasks such that the sensor node only has to communicate via Bluetooth to the Raspberry Pi, we can extend the battery lifetime of the AQ board by 96.2%, 73.2%, 94.2% and 87.5% for each case of total anomalies detected daily. The mode of communication may dramatically affect the deployment and operations budget. While it may seem more convenient to aggregate data directly from a sensor node to the Internet (a selling point of the Particle Photon we chose), battery limitations may motivate designers to provide connectivity via closer Bluetooth-enabled gateways instead.

It comes as no surprise that low-power sensing devices are far slower at performing machine learning functions than servers. However, delegating part of automation tasks to them can yield great benefits. In this section, we have studied the energy effects of delegating the same tasks to different machines and with different communication methods. We found that there is a benefit to running an anomaly detection algorithm on a sensor node or a gateway, yielding far higher battery lifetimes for resource-constrained nodes (up to 72x in specific cases) and lower energy costs in the larger system (22-45%).

8. CONCLUSION

The rise of Internet of Things and availability of personal connected sensors enables citizens to have finer knowledge of their immediate surroundings, more so than just relying on coarse grain regulatory reports in
the past. To fully leverage the opportunity that ubiquitous data, cheap computation devices and vast Internet connectivity, we must develop new ways for devices and software to interact and share valuable insights.

In this work, we demonstrate the versatility of a previously proposed context engine framework to be deployed across a hierarchy of devices for a smart health application. To integrate personal context with large population-size datasets, we also present a custom sensor platform that connects individual users to the world of data through either an IoT gateway or an edge server (or both). The air quality board we describe in this paper is a low cost, embedded component of the IoT that enables both experts and lay-users to measure the air quality of their immediate surroundings. Notably, the hardware and software design are intended to be reconfigurable and extendable with minimal effort. Because IoT devices provide both computation and communication of various levels, they give us many different ways to implement the data analysis that is required to process raw data into high level, human-readable feedback. We propose several examples of how subtasks in a health application might be distributed to various nodes in the system, and compare those distributions with respect to energy consumption and device lifetimes. By judiciously allocating machine learning tasks to sensor nodes at the edge of the cloud, we can greatly reduce the volume of data transmission between devices, thus reducing the overall system energy by 22-45%, and in particular multiplying the battery lifetime of power-constrained devices on the order of 22-72x.

ACKNOWLEDGMENTS

This work is sponsored in part by TerraSwarm, a funded center of STARnet, a Semiconductor Research Corporation (SRC) program sponsored by MARCO and DARPA, as well as NSF Grant No. 1446912. The authors would like to thank Prof. William Griswold at UC San Diego, Prof. Michael Hannigan at UC Boulder and their research teams for lending their expert advice.

REFERENCES


Zaouak, Olivier, e. a., “High performance cost effective miniature sensor for continuous network monitoring of H2S.”


“California Department of Transportation - Division of Transportation Planning.”

“Centre for Time Use Research - information gateway.” http://timeuse-2009.nsms.ox.ac.uk/information/studies/.


