# Realistic and Repeatable Emulation of Energy Harvesting Environments

JOSIAH HESTER, LANNY SITANAYAH, TIMOTHY SCOTT, and JACOB SORBER, Clemson University

Harvesting energy from the environment makes it possible to deploy tiny sensors for long periods of time, with little or no required maintenance; however, this free energy makes testing and experimentation difficult. Environmental energy sources vary widely and are often difficult both to predict and to reproduce in the lab during testing. These variations are also behavior dependent—a factor that leaves application engineers unable to make even simple comparisons between algorithms or hardware configurations, using traditional testing approaches.

In this paper, we describe the design and evaluation of Ekho, an emulator capable of recording energy harvesting conditions and accurately recreating those conditions in the lab. This makes it possible to conduct realistic and repeatable experiments involving energy harvesting devices. Ekho is a general-purpose, mobile tool that supports a wide range of harvesting technologies. We demonstrate, using a working prototype, that Ekho is capable of reproducing solar, RF, and kinetic energy harvesting environments accurately and consistently. Our results show that Ekho can recreate harvesting-dependent program behaviors by emulating energy harvesting conditions accurately to within 77.4 µA for solar and 15.0 µA for kinetic environments, and can emulate RF energy harvesting conditions consistently.

CCS Concepts: • **Computer systems organization** **Sensor networks**; *Embedded hardware*; • **Hardware** *Power estimation and optimization*;

Additional Key Words and Phrases: Energy Harvesting, Emulation, I–V curves, RFID

**ACM Reference format:**
Josiah Hester, Lanny Sitanayah, Timothy Scott, and Jacob Sorber. 2017. Realistic and Repeatable Emulation of Energy Harvesting Environments. *ACM Trans. Sensor Netw.* 9, 4, Article 39 (March 2017), 33 pages.
DOI: 0000001.0000001

## 1 INTRODUCTION

Energy is the greatest single limiting factor for many mobile sensing applications, and recent advances in energy harvesting are making it possible to deploy smaller sensing devices for long periods of time without the need for regular maintenance (to replace or recharge batteries). A wide range of environmental sources are readily available, and whether a device harvests solar [20, 25], radio frequency (RF) [3, 32], kinetic [26] or even energy from other devices [21], harvested energy sources are often highly variable, scarce, and difficult to predict.

Battery technology has been able to store surplus energy harvested during the high energy periods to be used during the low energy periods. However, rechargeable batteries wear out, charge slowly,

This research is supported by the National Science Foundation (NSF) under award number ACI-1212680.
Author's addresses: J. Hester and L. Sitanayah and T. Scott and J. Sorber, School of Computing, Clemson University.
 1550-4859/2017/3-ART39 $15.00
DOI: 0000001.0000001

require special protection and charging circuitry, pose environmental risks, and fundamentally limit the lifetime and deployability of today's mobile computing devices. These have inspired a range of capacitor-based energy storage solutions for sensor devices [13, 14, 34, 39–41] that harvest energy, charge quickly, and can store only enough energy for short bursts of operation. Using capacitors as energy storage, smaller sensing devices can be deployed in remote or otherwise challenging environments for much longer periods of time. These batteryless devices can operate in a truly untethered fashion, allowing perpetual data collection while requiring little or no maintenance.

However, hardware and software solutions for energy harvesting sensing devices with limited energy storage are difficult to design, debug, and especially to evaluate. Harvested energy varies and energy storage constraints continue to tighten in order to accommodate smaller mobile form factors. The consequence is that, in addition to the more traditional challenges faced by mobile devices (like uncertain network connectivity), it is often difficult for system designers to predict how their devices will behave at runtime. Reliably comparing different algorithms, approaches, or configurations is often impractical or extremely labor-intensive. These challenges are primarily the result of two key characteristics of energy harvesting systems: 1) *energy harvesting is erratic and unpredictable*, and 2) *the amount of energy harvested depends not only on environmental conditions, but also on the device's behavior at runtime.*

The combination of a behavior-dependent energy supply and a high degree of runtime volatility makes repeatable experimentation impractical, using traditional testing strategies. Two test runs with *the same hardware and software* may result in dramatically different results, due to differences in energy harvesting conditions. Two runs with *different software or hardware configurations* may produce dramatically different results under the same harvesting conditions, assuming that harvesting conditions can be replicated. Runtime conditions are often vastly different from in-lab conditions, and may be difficult to replicate during testing. In order to compare different algorithms or different hardware configurations, a system designer must currently either run a large number (hundreds or thousands) of tests under realistic runtime conditions and compare results stochastically (a labor-intensive and imprecise approach), or control energy harvesting conditions in simulation.

Simulators have been developed that predict the power consumption [5, 10, 30, 32, 36] and even the energy harvesting [14, 32] behaviors of sensor devices. Unfortunately, most ignore the impact of device behavior on energy harvesting. Indeed, simulators must depend entirely on an accurate model of the test device's hardware characteristics. As device hardware evolves, or when a designer wants to try out a different hardware component (e.g., a new sensor, actuator, or processor), the simulation software must be updated—often involving a significant amount of in-lab measurement and testing.

This paper explores a third option, *emulation*. Instead of depending on software models of energy harvesting and consumption, an energy harvesting emulator records energy harvesting conditions and then accurately reproduces the recorded conditions (in the form of physical "harvested" power) to a real test device running in the lab. This approach provides system designers with a realistic and repeatable evaluation technique, without sacrificing flexibility—modifying the hardware and software on the test device does not require any changes to the emulator.

In this paper, we describe the design, implementation, and evaluation of Ekho, a tool that records and emulates energy harvesting conditions for capacitor powered energy harvesting devices. Ekho is generally applicable to a wide range of harvesting technologies. Ekho uses a novel method to explore and record an energy harvesting environment by modulating the load using a precisely controlled digital potentiometer. This energy harvesting environment (solar, RF, kinetic) is processed and stored to be later replayed through a custom analog front-end which serves as a current source. To evaluate Ekho, we have developed two prototypes; a desktop version, and a mobile recording version. Using these prototypes, we evaluate Ekho's ability to replicate energy harvesting conditions both accurately

and consistently. In our evaluation we found Ekho is consistent within 68.7 µA[1] from test run to test run, emulating recorded solar harvesting environments to mote-class devices running a variety of test programs. Ekho reproduces a recorded solar trace with a mean error of less than 77.4 µA from the recorded surface. We also found that Ekho was able to record RF energy harvesting environments and replay them with high fidelity, and low error rates for most transmit powers. Finally, we found that Ekho can reproduce kinetic energy environments with a mean error of 15.0 µA from the recorded surface. It should be noted that a faster processor was used for the kinetic results which significantly contributed to the accuracy gains over solar. Parts of this work were presented in [15].

## 2 HARVESTING ENERGY...AGAIN

Ambient energy, harvested from the environment, is key to the success of any sensing and pervasive computing application that requires small devices to operate maintenance-free over long periods of time. Energy in its many forms (solar, RF, mechanical, thermal, etc) can be converted into electrical energy that can be stored in batteries or capacitors and used to power the device's processor, sensors, and other components. Rechargeable batteries usually wear out after a few years due to a limited number of charge cycles, charge slowly and pose disposal and safety hazard. Capacitors, however, can last for decades of useful operation, charge quickly and are more environmentally friendly. **The work in this paper focuses on battery-less energy harvesting devices, however, the techniques are generally applicable to battery powerede sensing systems, just not as useful.**

Unfortunately, designing devices that effectively use this never-ending supply of free energy is challenging. Unlike traditional battery powered sensors (which duty cycle to prolong lifetime), energy harvesting devices must work opportunistically. Having too much or too little energy is equally inefficient and wasteful—energy leakage is high when the devices reach their full capacity. Greedily using energy can restrict functionality, while under utilization of harvested energy is wasteful in terms of computation that could have been performed.

Additionally, nearly all environmental energy sources vary widely and unpredictably at runtime, and as new applications require smaller form factors and lower energy storage capacities [4], power supply volatility increasingly influences and defines device behavior. Devices, like *computational RFIDs* (CRFID) [3, 33, 42], that replace batteries with small capacitors and store only enough energy for, at most, a few seconds of operation are especially susceptible, and may see their supply voltage increase threefold or fall to zero in seconds. Power supply fluctuations affect a device's runtime behavior in ways that are often difficult to predict or reproduce in the lab during testing.

Matters are complicated further by the fact that the energy harvested by each device depends not only on environmental conditions, but also on the device's supply voltage at runtime. The relationship between supply voltage and charge current can be characterized by an I–V curve, a function that describes how harvesting current (I) changes, with respect to the device's supply voltage (V). Different programs (loads) will occupy different areas of the I–V curve as shown by Figure 1.

Figure 2 shows six (6) example I–V curves, two produced by a solar panel under high and low light conditions, two produced by a Peltier generator—which converts thermal differentials into electrical current—under 5 °C and 10 °C thermal differentials, and two produced by RF energy from a reader at +32.5 dBm and +27.75 dBm. In all three cases, environmental changes alter the harvester's I–V curve. In addition, each harvester produces its own distinct "family" of curves, with a common characteristic shape. These curves change over time, resulting in I–V surfaces,sequences of I–V curves belonging to a certain family.

[1]depending on capacitance

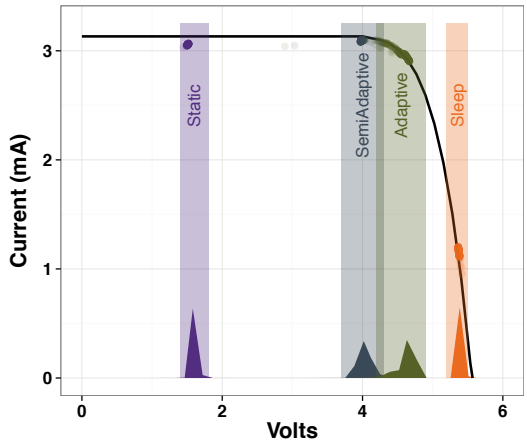



Fig. 1. This figure shows how program behaviors influence energy harvesting performance. Four (4) programs' harvested currents, measured over a 20 s period of time, are shown with respect to their supply voltage, while connected to a programmable solar environment (see Section 4) generating a single I–V curve (shown in black). Points represent an average of many samples (many points are not contained in shaded regions), and the histogram's along the bottom show the sample density at each point. Due to differences in behavior (power consumption), each example program occupies a different section of the I–V curve. These differences result in significant variations in harvested power.

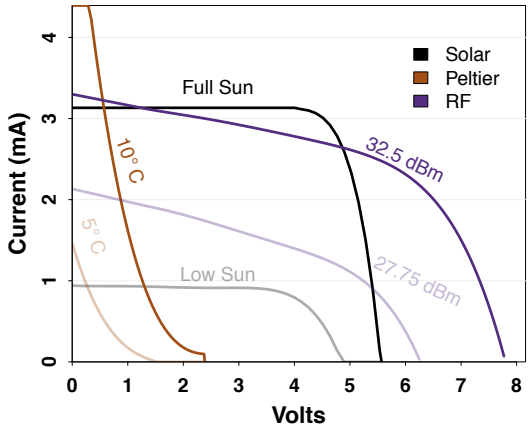



Fig. 2. Six I–V curves are shown, produced by three different energy harvesters—a solar panel, a Peltier generator, and an RF Reader—each under two (2) different energy harvesting conditions. Each harvester produces its own "family" of curves, with a common characteristic shape. Each of the above I–V curves were captured with the recording feature of Ekho. Note that the Peltier curve has been scaled $17x$ for purposes of illustration.

At runtime, an energy harvester's I–V curves impact program behaviors and experimental outcomes. For example, two algorithms that draw different amounts of current will deplete their capacitors at differing rates, resulting in different supply voltages, and, consequently, different amounts of harvested power ($P = IV$).

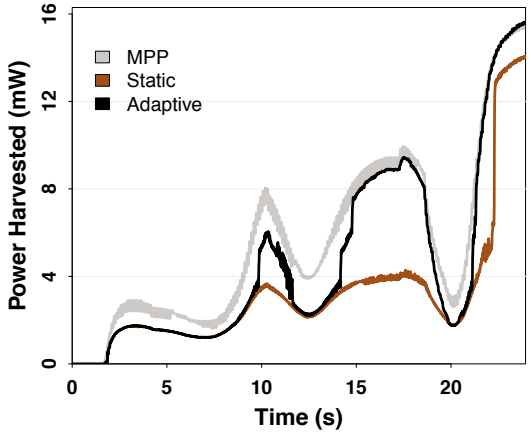



Fig. 3. Harvested power is shown for two TI EZ430-RF2500 target boards running two different programs that both write to flash in different ways—the Static program writes as fast as possible if there is any power available, the Adaptive program adapts so it can write even when harvestable energy is scarce—under the same solar energy harvesting conditions. Differences in power consumption result in different amounts of harvested power.

Figure 3 illustrates this scenario by showing the amount of power harvested by two TI EZ430-RF2500 devices running two different programs under the same solar harvesting conditions. Both periodically read data from an on-board temperature sensor, however the Adaptive program modulates its wait time depending on the voltage so it can sense when energy is scarce, while the Static program senses and writes whenever it is able. Under the test conditions, Adaptive stayed near the high energy knee of the I–V curve (see Figure 1), maximizing on available energy by watching its supply voltage, while the Static program harvested significantly less energy by being greedy. The *maximum power point (MPP)* is also shown to demonstrate the amount of power that could potentially have been harvested by a device with the right supply voltage.

This is illustrated further by Figure 4 and Figure 5. These I–V surfaces were recorded by Ekho from solar and kinetic energy environments, respectively. Different execution, behavior, or energy consumption causes different paths across the surface with different parts of the surface (those closest to the MPP) being more efficient than others.

Consequently, any attempt to predict how a low-power energy harvesting device will behave in the wild, must take into account the harvester's I–V characteristics and the resulting program variation. There are two common methods to predicting device behavior; (1) replaying a harvested power trace gathered from a device, and (2) using a programmable energy environment such as a light-box.

**Replaying Power:** One approach to making energy harvesting reproducible is to measure the harvested power as the device executes, and then replay the collected power trace. This approach has been used in other harvester-powered mobile systems [37], and our early efforts focused on replaying power traces.

Replaying a power trace is an attractive technique to predict device behavior under certain energy conditions since designing the hardware is simple and straightforward, and provides a reasonably accurate solution for devices with a stable supply voltage—like those with large batteries, which typically vary by less than half a volt when between 15% and 85% of a full charge. When the battery

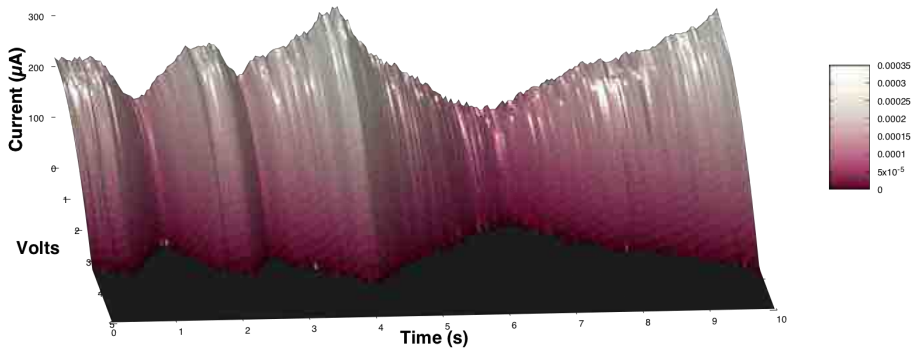



Fig. 4. Shown is a solar I–V surface recorded by Ekho. A solar panel was attached to the harvester input of Ekho and a light was shown on the panel from different distances. The large hills are when the light was closest to the panel, with the valleys being when the light was farthest away.

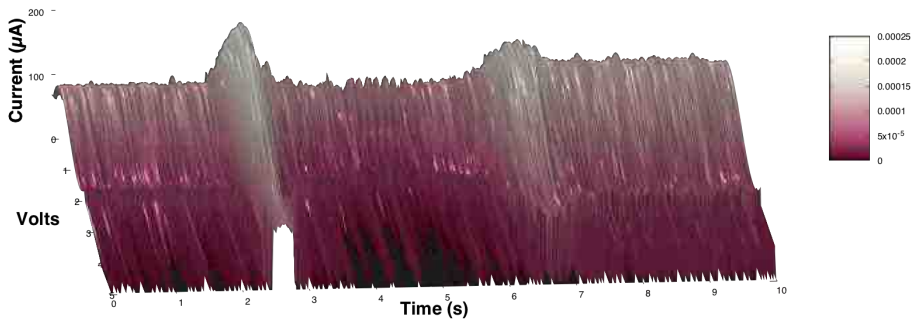



Fig. 5. Shown is a kinetic I–V surface recorded by Ekho. This surface was generated by attaching a weighted MIDE V21BL piezo ceramic to a surface transducer generating a sine wave that ramped from 160 Hz to 390 Hz, then back to 160 Hz. The large dips are the result of the surface transducer generating the resonant frequency of the piezo, causing a burst of extra harvestable energy. Ekho allows system designers to capture electrical characteristics dependant on the energy environment (like resonant frequency) in a straightforward way.

is nearly full or empty, simply replaying a power trace to simulate this will over or underestimate the energy that would be harvested in an actual deployment scenario.

While replaying power will work most of the time for devices with large batteries, devices that store their energy primarily in small batteries or capacitors have much less stable supply voltages that explore much more of the energy harvester's I–V curve. Figure 6 illustrates the I–V characteristics that are produced by recording the power harvested at a single point and replaying that power during experiments. Replaying constant power results in an effective I–V curve, defined by $I = \frac{P}{V}$, where $P$ is the power being replayed. The figure shows three such I–V curves that could be inferred from the same solar I–V curve. In all cases, the "constant power" curves approximate the real energy harvesting characteristics in only a small part of their range. In a later section, we compare the results of emulating power with different training sets to Ekho, and the light-box mentioned below.

**Programmable Energy Environments:**

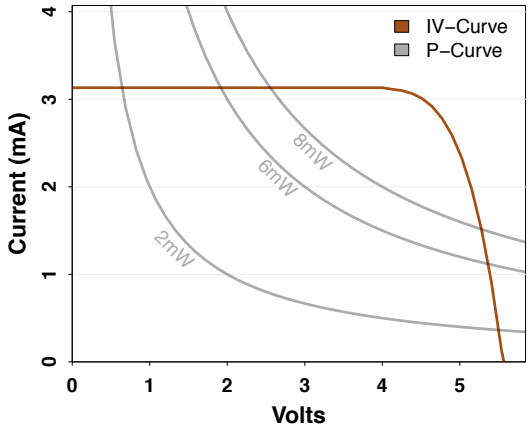



Fig. 6. Shown is a single I–V curve, and the consequences of choosing constant power to represent it. Depending on the load, the generated P-curves from the power trace can cause unrealistic changes in the programs actual harvested energy. As shown, emulating constant power is a poor replacement for emulating the actual I–V curve.

Programmable energy environments offer a somewhat more comprehensive effort at reproducing energy conditions than simple power replay. These environments make an effort to isolate an energy source, such as solar, RF, or vibrations, and create a repeatable environment to provide energy to a system [2]. Unfortunately, these devices are often difficult to construct and require careful calibration in order to provide accurate and reproducible results. For example, a reproducible vibrational energy environment requires special care to reduce the effect of ambient vibrations and noise events; experiments can be interrupted or affected by environmental events such as students walking down a hall, a chair moving, or even loud noises. These devices also tend to possess many points of failure or errata introduction. However, these analog solutions to programmable energy environments are much better than naive approaches that simplistically replay power.

In developing Ekho, we made extensive use of three such environments, dubbed the "light-box", the "RF-box", and the "shake-table". The light-box consists of a vehicle headlamp whose output is controlled via microcontroller and offers the ability to provide a controlled amount of light directly to a solar panel with minimal influence from outside sources. The energy produced by the light-box can then be used to power a low energy system and produce reasonably repeatable results, however it is not perfect as Figure 7 shows. As the light-box heats up, it changes the energy conditions inside the box, causing program behaviors to change from one run to the next. The RF-box is lined with copper mesh, effectively creating a Faraday cage, that isolates the interior from wireless interference, which can cause variation in harvesting current. Inside the RF-box is a programmatically controlled antenna that can power small CRFID tags such as the Umich Moo [42]. By modulating the transmit power different harvesting conditions can be created. The shake-table uses a signal generator connected to a surface transducer to excite a piezoelectric ceramic, which provides power. The amplitude and frequency of vibration can be adjusted to create different energy harvesting conditions. Constructing other programmable energy environments requires similar steps to isolate the energy source.

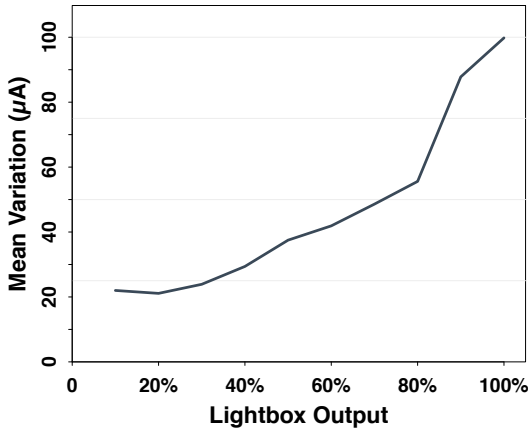



Fig. 7. The light-box mean variation between runs increases with light intensity for static loads. The light-box, unlike Ekho is susceptible to environmental changes and care must be taken to control those. Temperature changes after long use are one such factor that affects repeatability.

## 3 EKHO

The Ekho emulator is designed to capture the physical characteristics of an energy harvesting environment and recreate those environmental conditions in order to enable repeatable and realistic in-lab testing. Ekho does not emulate program behaviors, but captures features of the energy environment that allow testing of different program behaviors in a realistic way. Rather than focus on supporting a specific harvesting technology, our design of Ekho is focused on providing a generally applicable tool that supports a wide range of energy sources, while providing users with flexibility, mobility, accuracy, and consistency.

**Generality:** In Ekho, energy harvesting conditions are represented as I–V curves—an abstraction that, as discussed in Section 2, can be used to characterize any common energy harvesting technology. Changes in harvesting conditions over time are represented by combining multiple I–V curves into I–V surfaces. This generality frees the experimenter from designing expensive custom hardware such as a light-box or Faraday cage[2] to test devices before deployment. Ekho uses a novel method to explore and record these I–V surfaces by quickly modulating the load using a precisely controlled digital potentiometer. This allows Ekho to rapidly explore any I–V surface, including RF, with minimal changes in experimental setup.

**Flexibility:** A key focus of our design is to allow application designers to effortlessly compare different software and hardware options. Ekho achieves this by mimicking the physics of an energy-harvester, providing realistic and repeatable power to real test devices. Using this approach, trying out a new sensor, energy harvester, scheduling algorithm, or even a new processor, requires no changes to the emulator, no profiling or modeling. The user makes the desired change to the experimental setup and continues using Ekho with no Ekho-specific configurations or alterations.

**Mobility:** Since Ekho is a supporting tool for tiny energy harvesting devices deployed in many different conditions, a mobile form-factor is essential for the recording function. This feature allows

[2]A Faraday cage is a container made of conducting material that prevents the entry or escape of an electromagnetic field.

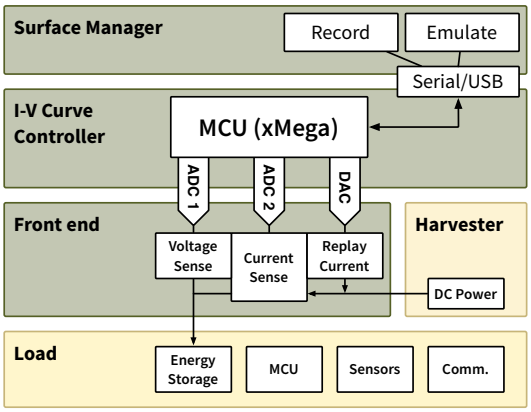



Fig. 8. Ekho consists of three interdependent modules: a surface manager that stores I–V surfaces and manages the high-level recording and emulation logic for the system; a low-latency controller that sequentially emulates the I–V curves that correspond to each single point in time during an emulated surface; and a front-end module that facilitates controllable current emulation and provides signal conditioning that is needed for taking accurate current and voltage measurements.

Ekho to be deployed with existing systems, or pre-deployed to characterize the energy environment beforehand. We have designed a deployable version to demonstrate the mobility of Ekho.

**Accuracy:** An energy harvesting emulator is only as useful as it is able to accurately recreate energy harvesting conditions. At runtime, devices often experience a wide range of rapidly-changing harvesting conditions. Ekho is designed to accurately estimate I–V surfaces of varying shapes and magnitudes. This allows Ekho to recreate recorded conditions with sufficient accuracy to mimic energy fluctuations and patterns that devices confront in the myriad conditions of real-world deployment.

**Consistency:** Perhaps the most important goal for Ekho is consistency. No two recorded traces of energy harvesting conditions will be identical, and test engineers may often be willing to tolerate emulations that are similar, but not identical, to those recorded in the wild. In contrast, experiments that aim at comparing different algorithms or hardware choices require that test runs be consistent. Inconsistent emulation yields results that are not reproducible and difficult to interpret. Ekho offers favorable accuracy behaviorally and physically compared to other controlled energy harvesting environments but excels in reproducing energy conditions consistently.

### 3.1 System Architecture

In order to achieve these goals, we have designed a system architecture, shown in Figure 8, which consists of three interdependent modules: a *surface manager* that stores I–V surfaces and manages the high-level recording and emulation logic for the system; a low-latency *I–V curve controller* that sequentially emulates the I–V curves that correspond to each single point in time during an emulated surface; and an analog *front-end* module that facilitates controllable current emulation and provides signal conditioning that is needed for taking accurate current and voltage measurements, especially during periods when harvested energy is scarce.

Ekho's surface manager controls both the recording and emulation of energy harvesting conditions. This includes receiving current and voltage measurements from the I–V Controller during recording, estimating I–V curves from the received measurements, storing I–V surfaces in a digital format,

and sending I–V curves one-by-one to the I–V curve controller during emulation. The storage and computational requirements for these activities fit comfortably within the capabilities of the current generation of laptop and desktop computers. The mobile version separates the recording surface management from the emulation, but otherwise performs the same function.

In order to accurately emulate I–V curves received from the surface manager, the I–V curve controller must be able to quickly gather current and voltage measurements and respond to those changes appropriately (within a few µs). This requirement is most easily satisfied by a processor with integrated analog-to-digital (ADC) and digital-to-analog (DAC) capabilities, a combination that is rarely found in today's high-speed processors, but which are provided by some higher-speed microcontrollers, like Atmel's AVR XMEGA line of controllers [6], and some ARM controllers, which we use in our prototypes, described in Section 4.

The I–V curve controller relies on the third module, an analog front-end, to provide the amplification and other signal conditioning needed for accurate I–V curve emulation and measurement. When capturing energy harvesting conditions, this circuit is placed between the harvester and test load. During emulation, the front-end takes on the role of energy harvester, providing the device under test with a current supply that mimics the energy source being emulated.

The following sections describe how these modules work together, in two different operating modes, to record and emulate harvesting conditions.

### 3.2 Recording I–V Surfaces

Ekho captures the energy harvesting conditions by measuring them directly. Electrical current is measured by the front-end as it flows from the energy harvester into the test device's storage capacitor. Current is measured by observing the amplified voltage drop across a low-tolerance sense resistor. The test device's supply voltage is also measured. These current-voltage (I–V) measurements are converted from analog voltages to digital values by the I–V curve controller as rapidly as possible and passed along to the surface manager for post-processing, where I–V curves are generated from the I–V pair point cloud.

This series of recorded I–V pairs represent a single path across the three-dimensional surface that represents the harvesting conditions during the trace; the surface manager's challenge is to estimate the entire surface from this single path. Each recorded I–V pair captures one point on the I–V curve that represents harvesting conditions at the time it was captured. When considered alone, each point could have been produced by an infinite number of different I–V curves; however, a series of I–V measurements can be used to infer the current I–V curve's shape, assuming 1) that the measurements are gathered quickly before the state of the I–V curve changes measurably, and 2) that the measurements adequately span the I–V curve's voltage range. Taking measurements rapidly (>1 million samples/second) is straightforward. Inducing enough supply voltage volatility to quickly and fully characterize the I–V curve at each point in time requires more care. A key contribution of Ekho is its novel method to induce supply voltage volatility.

*3.2.1 Inducing Supply Voltage Volatility.* At runtime, the power consumption of a typical test device (or test load), like a CRFID or mote-class sensor, does not often change rapidly enough or significantly enough to explore the entire I–V curve. This is illustrated in Figure 9, which shows two sets of 6,000 I–V pairs collected by Ekho over a period of 30 ms, under similar solar harvesting conditions, while using two different test loads: an off-the-shelf TI EZ430-RF2500 mote [17], and a custom *smart* test load that we have designed specifically for inducing voltage changes in order to assist with Ekho's recording mode.

The custom "smart" load is a digital potentiometer controlled by a microcontroller which rapidly alters its power consumption in order to induce large fluctuations in supply voltage for more accurate

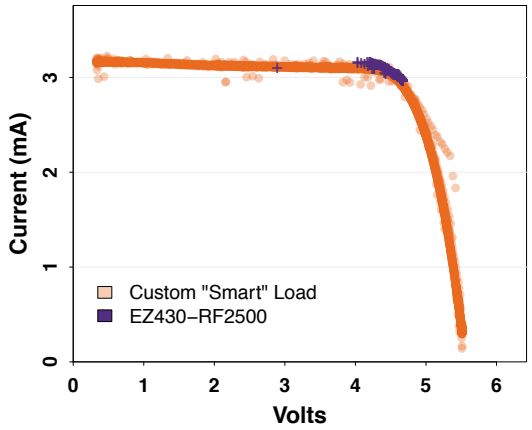



Fig. 9. This figure shows recorded I–V measurements, as produced by both the Ekho smart load device and a typical mote-class sensor device. By intentionally increasing the power supply volatility, the smart load provides much better coverage of the I–V curve being recorded, which improves Ekho's recording accuracy.

recording. The Arduino controls the potentiometer and makes it cycle through a predetermined number of resistance settings for a given time delay. These changes produce a wide range of load currents that explore different parts of the I–V curve. As long as the cycle frequency is high enough, and the upper and lower bound of the potentiometer's resistance settings can exercise the extreme ends of the curve, the shape of any instantaneous I–V curve can be gathered. In our experiments we have found that a 100 kΩ potentiometer provides a large enough range. For custom "smart" load "cycle frequency" (the number of times a second the smart load cycles through all its resistance settings), we found that 100 Hz can capture solar I–V curves, and 1000 Hz and above is sufficient to approximate an RF I–V surface

As shown in Figure 9, when using the custom load, the measurements are spread evenly across the I–V curve, the *smart* load effectively explores the entire I–V curve, while the mote measures only a small part of the curve.

*3.2.2 Surface Construction.* Once these measurements have been captured, the surface manager uses a curve-fitting algorithm to estimate the shape of the I–V curve that most closely fits each window of data, and the series of inferred I–V curves make up the I–V surface that is stored for later use during testing. A variety of curve-fitting algorithms exist, which could be used. For the sake of simplicity, we use the polynomial fitting algorithm provided by the GNU Scientific Library (GSL) [12], and have found it to work well in practice, both in terms of accuracy and efficiency. The size of each window is configurable (and is closely related to the custom load cycle frequency), and represents a tradeoff between temporal accuracy and I–V curve accuracy. If the window is too small, containing too few points with poor coverage, the estimated I–V curves may be inaccurate. If the window is too large, then short-term changes in the I–V surface could be effectively filtered out of the captured representation, decreasing the temporal accuracy of the surface; however this is harvester dependent. Trading temporal accuracy for a larger window (and therefore I–V curve accuracy) will not influence the final behaviors of most programs running on slow changing solar surfaces where curves switch at less than 100 Hz. However, for RF surfaces this can pose a significant problem as curves can change upwards of one thousand times a second.

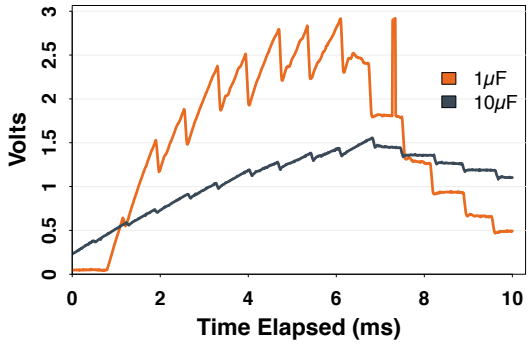



Fig. 10. This figure shows the effect of capacitance while recording an RF I–V surface. As the capacitance increases the output is averaged, and important features are lost. Each peak is the custom "smart" load changing its resistance setting, these peaks are absorbed by the larger capacitance, which means that voltage volatility is lost. Because of this the I–V surface is not as fully explored. For energy environments, like solar, that evolve slowly, this may be acceptable, for volatile RF or kinetic harvesting environments, important surface information will be lost.

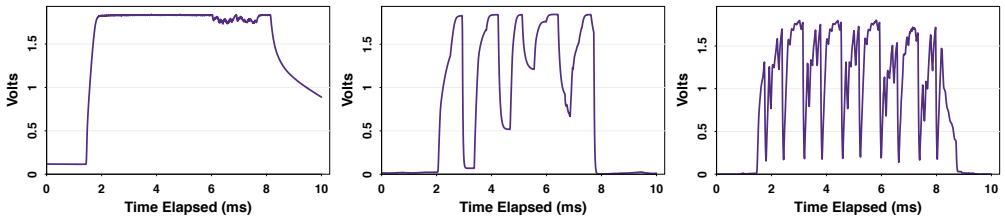
*3.2.3 Complicating Factors.* Care must be taken when recording an energy environment. The sensitivity of the capacitor powered energy harvesting device under consideration and the accuracy required for emulating will influence decisions made when recording. Choosing the capacitance and cycle frequency of the custom "smart" load is critical to an accurate I–V surface recording. Capacitance while recording has the effect of averaging out the surface over some time period as shown in Figure 10. This smoothing effect is desirable for I–V surfaces that change slowly (such as solar), as it reduces noise, and results in a cleaner representation of each individual I–V curve. However, applying extra capacitance to fast changing surfaces (such as RF) will average out peaks and valleys in the recorded surface. This averaging will change the final harvested power, and therefore the final program behavior. This smoothing capacitance is only necessary if I–V surfaces that are being recorded with Ekho are noisy, in most cases, a large smoothing capacitance is unnecessary.

The cycle frequency of the smart load also plays a factor in determining the accuracy of the final constructed surface. If cycle frequency is set too low for a particular harvester type, the final surface will be missing important features. Alternatively, if it is set too high, Ekho may not be able to emulate it fast enough. Figure 11 shows differences in curve coverage and how they can affect the final surface. Ekho is configured to support a wide range of harvester types. Each of these harvester could require different combinations of cycle frequency and capacitance. In our experiments, we have found that a capacitance of 10 µF and a cycle frequency of 100 Hz is adequate for recording solar surfaces, while capacitances less than 0.1 µF and cycle frequencies of at least 1 kHz is required for recording RF and kinetic surfaces accurately.

### 3.3 Emulating I–V Surfaces

Ekho emulates stored I–V surfaces in three phases. First, the Surface Manager preprocesses each I–V curve in the surface for efficient transmission and emulation. Second, the curves are communicated at the appropriate time to the I–V Curve Controller. Third, the I–V curve is emulated by using the signal conditioning capabilities provided by the front-end.

In order for Ekho to emulate energy harvesting efficiently, each I–V curve needs to be represented compactly, in a form that reduces the computational workload of the I–V curve controller. To this




(a) Trace with a constant resistive load. Using just this to generate an I–V curve will miss important parts of the curve.

(b) Trace with smart load cycling at 1 kHz. The rapid voltage fluctuations explore more of the the I–V curve. However, the cycle frequency is not quite high enough to capture every part of the curve.

(c) Trace with smart load cycling at 10 kHz. The cycle frequency is high enough to capture details of the curve such that we can approximate the entire surface reasonably well.

Fig. 11. RF energy harvesting voltage trace over 10ms, with three different custom "smart" load cycle frequencies.

end, each curve is discretized down to $2^n + 1$ points. A power of 2 is used for efficiency in looking up currents based on ADC-provided voltage measurements. The choice of $n$ represents a tradeoff between smaller I–V curves which can be communicated more quickly, and larger curves which may represent the original curve most accurately. By default, Ekho uses 65-point curves ($n = 6$), which provides good results for most types of energy environments. Additionally, in order to reduce the computational load further, the surface manager precomputes the DAC value that is required to produce the desired current.

After the surface is preprocessed, the surface manager begins emulation, sending each I–V curve to the I–V curve controller at the time it is to be emulated. The new curve replaces the old curve in the microcontrollers' RAM, point-by-point, as it is received. The rate at which new curves need to be sent depends on the harvester being emulated (some harvesters' curves change faster than others). For especially fast surfaces, like RF, the I–V curve controller can store the entire surface in RAM to facilitate greater than 1 kHz curve updates. For the current prototype, this limits RF surface length to under two seconds, this limitation can be overcome using external memory to allow much longer RF traces to be emulated.

Throughout this process, the I–V curve controller emulates each curve by measuring the test device's supply voltage and playing the appropriate voltage to the front-end using its DAC, repeatedly. Finding the right DAC value requires two I–V curve lookups to find the two closest points on the curve and a linear interpolation between the two found DAC values. The voltages output by the DAC are amplified by the front-end (increasing the range up to nearly 8 V), and the amplified output is connected, through a low-tolerance 400 Ω resistor followed by the 10 Ω sense resistor (used for current sensing) to the test device's capacitor. This produces a predictable harvesting current ($I = \frac{V}{410\Omega}$).

Note that the feedback loop executed by the I–V curve controller must be extremely fast. The action of emulating a current, in addition to the current draw of the test device, causes the supply voltage to increase or decrease, which necessitates a change in current. If the feedback loop is too slow, then a larger storage capacitor could be used on the device under test, if design constraints allow. Using a larger capacitor to store the harvested energy will cause the supply to change more slowly, giving Ekho more time to respond. However, in our evaluation with the most recent version

(a) Analog Front-end (b) Micro Ekho (c) All-in-one

Fig. 12. **Shown are our three reference Ekho implementations.** Figure 12(a) shows the desktop analog front-end; this can be used for emulation and recording, and with any MCU and smart load. Figure 12(b) shows our mobile Ekho recording prototype. The mobile version is 9x smaller than the desktop version, and does not need any supporting hardware (such as the dedicated PC for the desktop version) to record I–V surfaces. With a small button battery, it can record for weeks. Figure 12(c) shows an all-in-one recording and emulation Ekho device as a Teensy 3.6 ARM daughter board.

of Ekho, we found that our emulation speed was more than sufficent to emulate with high accuracy, without changnging the size of the storage capacitor of the device under test.

## 4 IMPLEMENTATION

In order to evaluate the efficacy and usefulness of our approach, we implemented four different prototypes, and the software, tools, and programmable energy environments to evaluate them, . This section gives implementation details of each.

### 4.1 Hardware

In order to evaluate the efficacy and usefulness of our approach, we have implemented four different prototypes; an analog front-end (shown in Figure 12(a)), a deployable, MSP430 based I–V surface recorded (sown in Figure 12(b)), a larger, more accurate ARM based I–V surface recorder (not shown), and an emulation only Teensy ARM breakout board (shown in Figure 12(c)). We use the evaluation setup shown in Figure 13 to conduct all our experiments. This desktop evaluation unit consists of a surface manager, an I–V curve controller, and a custom analog front-end. The mobile unit contains only the components necessary to record I–V surfaces; the analog front-end, the smart load, and a micro-controller as hybrid I–V curve controller.

*4.1.1 Desktop Evaluation Unit.* The desktop system employs a variety of different hardware components. The surface manager is implemented using a Windows 7 (64-bit) desktop. The analog front-end is implemented with a custom printed circuit board (PCB) that provides filtering and amplification for accurately measuring low-amplitude current and voltage signals. The analog front-end is powered by a 9V DC source. While Ekho is designed for low current harvesting scenarios, our current implementation can accept harvester input voltages up to 8 V and input current up to 0.5 A. This allows a broad range that can handle most low power devices. Our prototype uses two different devices to implement the I–V curve controller functionality—an Atmel ATXmega256A3B microcontroller [6] when in emulation mode, and an NI USB-6356 data acquisition device (DAQ) [16] when in record mode. The DAQ provides the needed high-speed data collection capabilities needed for recording, while the ATXmega provides a 32 MHz processor with integrated ADC and DAC

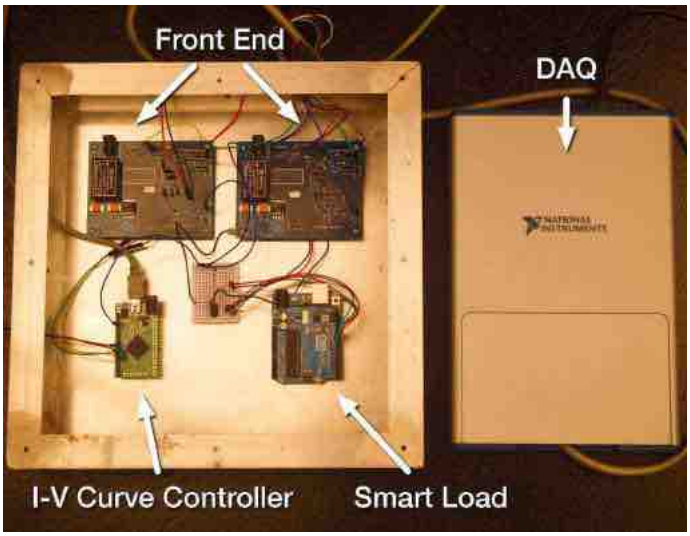



Fig. 13. Our prototype desktop Ekho evaluation setup, including two custom analog front-end boards, the ATXmega256A3B-based I–V curve controller, and the "smart" load used to explore I–V surfaces during recording. Note that while only a single front-end board is needed for Ekho to function, we include two so that we can easily switch between experimental configurations. A shielded enclosure and shielded cabling are used to reduce induced measurement noise. An external NI USB-6356 data acquisition device (DAQ) (shown on the right) is used in our experiments to confirm Ekho's measurements. The DAQ can also be used to provide recording speeds that exceed the capabilities of the I–V curve controller when needed. For kinetic evaluation, the ATXmega256A3B was replaced with a "Teensy" ARM Cortex-M4 breakout board.

for low-latency emulation. Total cost of the system, including ATXmega, Arduino, custom circuit boards, and parts (excluding the DAQ) is less than $700. The low-amplitude signals that Ekho must measure are highly susceptible to noise, induced from ambient electromagnetic radiation (from AC power lines and RF transmitters). A shielded enclosure and shielded cables are used throughout, in order to mitigate this problem. In later versions of desktop Ekho; specifically the version used for gathering the kinetic results, the I–V curve controller was implemented on a "Teensy" Cortex-M4 breakout board. The "Teensy" more memory and greater computational power when compared with the original ATXmega I–V controller. We also use the NI USB-6356 to collect voltage and current measurements during our experimental evaluation, as is described in the following section.

*4.1.2 Smart Load.* In order to support more accurate recording (as described in Section 3.2), we have developed a custom "smart" test load for our evaluation unit, that rapidly modulates its power consumption. This induces large fluctuations in supply voltage enabling more accurate recording. We have implemented this *smart load* using an Arduino Uno to control a digital potentiometer. The potentiometer [28] acts as a resistive load, with 128 settings ranging from 134 Ω to 100 kΩ of resistance. During the record phase the Arduino cycles through a predetermined number of these resistance settings randomly for a given time delay, producing a wide range of different load currents that explore different parts of the current I–V curve. Both prototypes use the smart load; however, the mobile version uses a 256 tap digital potentiometer with a wider resistance range of 120 Ω to 2 MΩ.

*4.1.3 Desktop Emulation Unit.* We anticipate that many designers will only be interested in the emulation function of Ekho, and will use previously recorded I–V surfaces to test with. Additionally,

many of those interested in using the recording features, will do so with a separate unit, such as the Micro Ekho described in the next section. Therefore, a dedicated emulation module could be needed. We implemented an emulation only version of Ekho that functions as a Teensy 3.2 [31] daughter board, shown in Figure 12(c). This unit can be powered over USB, but supports emulating I–V curves of up to 11.5V using the Analog Devices ADP1613 and AD623. In batches of 1000, including all components (except the Teensy 3.2), PCB fabrication, and assembly, we estimate the cost of the emulation unit to be only $11.47.

*4.1.4 Mobile Recording.* A key requirement of Ekho is that of mobility; it must be deployable to energy environments where sensors will be deployed. System designers can build in resiliency in their systems by recording I–V traces in future deployment environments, and then emulating those environments in the lab with Ekho on systems in development. To make the capture easier, we designed a mobile prototype that can be deployed and left in a future deployment area, powered by batteries, collecting I–V information to a microSD card. Once retrieved, the collected traces can be converted to I–V surfaces and emulated on hardware in development. Like the desktop version, the mobile version was designed to have as large an input voltage range as possible to accommodate the widest array of energy harvesters. We have developed two mobile recording prototypes to satisfy a variety of deployment scenarios. Figure 12(b) shows the smallest version; which is centered around an MSP430FR5728 as I–V curve controller. This mobile prototype, termed “Micro Ekho”, is meant for wearable, and wildlife tracking applications. In addition to the MSP430 microcontroller, Micro Ekho has a rectifier for AC signals, a CC1101 radio for communicating I–V information back to a basestation, and simplified “smart load” circuitry. The simplified “smart load” uses four N-channel MOSFETs with different resistance settings instead of a digital potentiometer. The MOSFETs are switched one at a time in rapid succession, exercising the attached energy harvester in the same way as the digital potentiometer, albeit with fewer stops. MOSFETs are a much cheaper and more power efficient solution, that allows recording higher voltage energy harvesters with a low voltage reference, suitable for deployment. In batches of 1000, including all components (except an enclosure and battery), PCB fabrication, and assembly, we estimate the cost of the Micro Ekho unit to be only $14.14.

Our second mobile prototype, termed “Mini Ekho” (not picture) records I-V characteristics in the same way as the larger prototype. It is built around an off-the-shelf 96 MHz Cortex-M4 breakout board. The Cortex-M4 functions as the I–V curve controller; using it’s built in ADCs to sense voltage and current off the analog front-end. Mobile Ekho is mounted onto the processor breakout as a daughter board composed of the amplifiers and sense resistor that make up the analog front-end, the smart load (a 2 MΩ AD5242 256-tap digital potentiometer), the regulator, and a slot for a microSD card to store the raw surface data. Mobile Ekho can sense voltages up to 16 V and currents up to 0.5 A The entire system is powered by a Li-Po battery, regulated to 3.3V. The total cost of the mobile version of Ekho, including "Teensy", custom circuit boards, and parts is less than $45.

## 4.2 Software

In addition to the Ekho apparatus itself, we have also implemented the software necessary for recording, processing, and emulating energy environments for both the desktop, evaluation, and mobile versions. For recording on the desktop evaluation unit, we interfaced with the NI USB-6536 to record a physical energy environment. The NI USB-6536 is capable of sampling rates up to 1 MHz and we use usually use sampling rates between 200 kHz and 500 kHz in our experiments. The code used for processing recorded data, was implemented using a combination of python, C, and C++ to process and gather relevant data and generate I–V curves. We use the GNU scientific library [12] for polynomial fitting of I–V curves and surfaces from recorded I–V traces. We also used Numpy [1] and

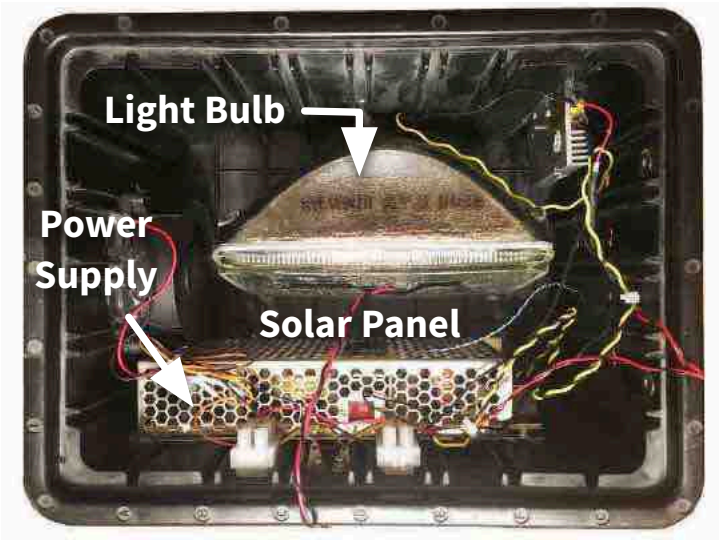



Fig. 14. Our prototype light-box implementation provides a reproducible solar harvesting environment. We use it in our experiments to provide reproducible "ground truth" harvesting conditions. The implementation consists of an automotive headlight, a solar panel, a power supply, and an Arduino Due (not pictured) which serves as programmable dimmer-switch.

Scipy [19] to verify the quality of fit. We use R and Gnuplot for I–V surface visualization and data verification.

For recording on the mobile version, we implemented software that manages recording and storing of I–V point cloud data, that can then be converted into I–V surfaces using the above tools at a later time.

For emulating I–V environments, we used custom software, written in C, to handle timing on the PC that is responsible for appropriately timing traces and relaying data to the I–V controller as necessary. The microcontroller code is written in C and stores a curve in memory. It then constantly alternates polling an ADC for new voltage readings and a USART for new curve data. As voltage readings and curve data become available, it alters its DAC output and stored curve data appropriately. For emulating curves that change very fast, but are short in duration, the entire surface is kept in the memory of the I–V controller.

### 4.3 Programmable Energy Environments

The light-box used for much of the energy recording and emulating experiments is shown in Figure 14. Our light-box consists of a light source (an automotive headlight), which can provide different intensity settings depending on a PWM input. A solar panel is mounted inside the chassis which provides shielding from outside light sources. An Arduino Duemilanove-328 uses pulse-width modulation to drive a dimmer switch inside the light-box to control light intensity. This provides a relatively repeatable energy environment for comparison with Ekho.

To facilitate a repeatable and noise-free RF energy environment, we built a small Faraday cage out of brass screen and copper mesh seals, fully enclosed in a wooden box as shown in Figure 15. This RF-box effectively isolates the interior of the box from external radio, wifi and other types

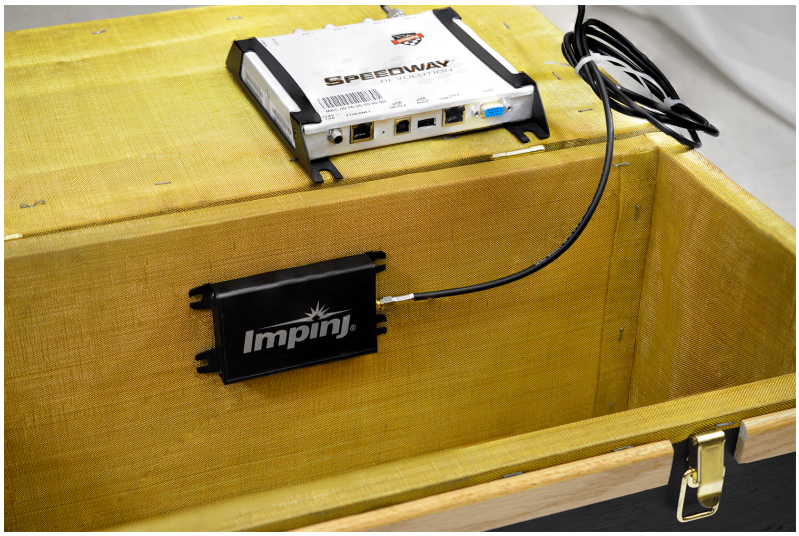



Fig. 15. Our prototype RF-box provides a reproducible RF harvesting environment used in our experiments to provide "ground truth" harvesting conditions. The RF-box is composed of a wooden shell layered with brass screen and copper mesh seals. The copper mesh creates a Faraday cage, isolating the inside of the box from common RF interference. Inside the box is an antenna, driven by a programmatically controlled reader.

of wireless interference. We mounted a programmable antenna connected to an Impinj Speedway Revolution UHF RFID Reader on the bottom of the cage to act as an energy source for RFID scale motes. By changing the transmit power of the antenna, many different I–V surfaces can be created. However, since each transmit power can generate thousands of different I–V curves, this is not always necessary.

We designed a repeatable kinetic energy harvesting environment we termed the "shake-table". It is composed of a programmable signal generator with amplifier, and surface transducer, which excites a MIDE V21BL piezoelectric ceramic. As the ceramic is excited, it produces an AC voltage which is then rectified by an LTC3588 into a DC voltage, suitable for powering low power devices. We used the MIDE V21BL and LTC5388, but any combination of harvester and ceramic could be used. By modulating frequency and amplitude of vibration, different I–V surfaces can be generated.

### 4.4 Tools

When first attempting to profile and record I–V surfaces with Ekho, it is often necessary to view graphical representations of the I–V surface and I–V point clouds that inform the surface generation. We have developed graphical tools with C/C++ and OpenGL that render I–V point cloud data in real time to aid the designer in tuning Ekho to fully capture I–V surfaces. The ability to view the raw I–V point cloud, and the I–V surface it generates in real time, allow the designer to quickly make decisions on fit, scale, and usefulness of the surface under consideration. Besides providing 2D and 3D graphical representations of the I–V data, these tools also output helpful metrics (such as curves per second) that allow the designer to estimate space I–V curve controller memory requirements for recording and emulation.

Table 1. Replaying power to emulate Solar energy environment

| | Replaying power with different training programs (flash writes) | | | | | |
|---|---|---|---|---|---|---|
| | Static training | | SemiAdaptive training | | Adaptive training | |
| **Program** | *mean* | *stddev* | *mean* | *stddev* | *mean* | *stddev* |
| Static | 330.0 | 3 | 558.0 | 10 | 566.0 | 2 |
| SemiAdaptive | 214.0 | 10 | 414.0 | 5 | 469.2 | 24 |
| Adaptive | 29.6 | 4 | 16.0 | 3 | 5.1 | 1 |

All software drivers, tools, rendering programs, and hardware designs have been made available via our website[3].

## 5 EVALUATION

In this section, we evaluate Ekho's ability to accurately capture energy harvesting conditions and consistently reproduce them in order to provide energy harvesting system designers with tighter experimental control, during testing. Specifically, we evaluate the consistency and accuracy of Ekho with respect to three programmable physical environments; the light-box, the RF-box and the shake-table. As a comparison, we also evaluate the previously discussed naive approach of replaying a recorded power trace (always replaying the same power, regardless of voltage). This comparison is conducted for a variety of different harvesting traces and loads (i.e. test programs). We also provide a more focused evaluation of Ekho's individual components (record and emulate) in order to explore the current limitations of Ekho and our prototype implementation.

In our experiments, Ekho was able to emulate solar I–V surfaces more consistently than our light-box, in terms of reproducing program behavior; in physical terms, Ekho is able to consistently produce solar I–V characteristics that vary by less than 68.7 µA (depending on capacitance) from test run to test run, emulating recorded solar I–V surfaces to mote-class devices running a variety of test programs. Ekho reproduces the solar I–V trace with a mean error of less than 77.4 µA from the recorded surface. Ekho reproduced kinetic surfaces with a mean error of 15.0 µA for medium impedance static loads, and slightly higher error for low, and high impedance static loads. Demonstrating the generality of Ekho; Ekho was able to emulate RF I–V surfaces significantly more consistently than the RF-box, for three (3) different transmit powers. In our experiments Ekho was able to reproduce RF energy harvesting conditions effectively such that program behaviors were accurate in comparison to the RF-box. In contrast, we also show how the naive approach of emulating constant power produces behavioral results that are inconsistent with the light-box, for battery-less, energy harvesting devices, and inadequate for predicting the performance of these small devices in deployment.

### 5.1 Methodology

Our evaluation involves emulating a total of 10,647 solar I–V curves, generated from 27 different randomly generated light-box traces (ranging from 6 seconds to 5 minutes in length), for a total of 1,029,000 solar I–V curves tested. We also record or emulate a total of 599 kinetic I–V curves, generated from four different randomly generated shake-table traces (ranging from 500 milliseconds to 10 seconds in length), for a total of 2995 kinetic I–V curves tested. In our evaluation comparing the accuracy of emulating constant power versus emulating I–V, we emulate a total of 3408 constant

[3]http://persist.cs.clemson.edu

power curves, generated from three program's harvested power traces. We emulate a total of 320 RF I–V curves, generated from three different recorded transmit power levels, for a total of 6400 RF I–V curves tested.

**Emulation Platforms:** Two computational platforms were used for emulating I–V surfaces during the evaluation of Ekho. The first, an XMega microcontroller, was used to emulate Solar and RF surfaces; the second, an off-the-shelf Teensy ARM Cortex-M4 breakout board, was used to emulate kinetic I–V surfaces. The ARM Cortex-M4 was overclocked to 96 MHz and used native USB for curve updates, allowing for much faster emulation (and better error results as shown in Table 6). The XMega microcontroller ran at 32 MHz and was limited by the speed of the onboard UART for curve updates. The XMega microcontroller was used for all solar and RF I–V surface emulation, while the Teensy ARM Cortex-M4 was used for all kinetic I–V surface emulation.

**Test Devices:** For test devices in our solar and constant power experiments, we use the EZ430-RF2500, a mote-class device produced by Texas Instruments, that consists of a MSP430F2274 ultra-low power microcontroller and a low power, 2.4 Ghz CC2500 radio; a 10 µF capacitor is used to store energy. For our RF experiments, we use the Umich Moo [42], an ultra-low power CRFID platform built around a MSP430F2618 microcontroller, and RF harvesting hardware. No batteries were used as power sources in any experiment, each mote device is powered exclusively from energy harvested and held in small capacitors. The static loads for kinetic testing were implemented using a digital potentiometer.

**Programs:** For solar experimentation; the EZ430-RF2500 devices run three different programs—Static, SemiAdaptive, and Adaptive—that provide different power consumption profiles and represent behaviors commonly seen in sensing applications. All three periodically read from the MSP430's internal temperature sensor and store the value to the sensor's internal flash memory. Between readings, all three programs put the the processor to sleep to conserve energy. They differ in how they manage energy. Static maintains a steady sampling rate regardless of energy availability. SemiAdaptive reduces its sampling rate when its voltage drops below a set threshold (2.3 V), in order to spend more time asleep and hopefully avoid a power failure. In addition to reducing its sampling rate during low energy conditions, Adaptive also increases its sampling rate when the its capacitor voltage exceeds a predetermined threshold (2.7 V), using its excess energy to collect more data. For RF experimentation, the Umich Moo devices run one program—Sense-and-CRC—that senses the internal temperature of the MSP430 using an on-board ADC five times, averages the readings, then performs a cyclic redundancy check (CRC) on the resulting data.

**Harvesting Traces:** We use the light-box, described previously, to provide a reproducible physical environment to serve as the ground truth for our solar experiments. We generate light-box traces, by randomly choosing a small number of light intensity settings distributed over a short amount of time, and interpolating those points using cubic splines, with exact boundary conditions. This is done multiple times to produce sets of different light-box traces. To test responsiveness of Ekho each of the solar traces changes much more rapidly than what would be seen in an outdoor deployment, with variations every 60ms. RF harvesting traces are generated for us by the inherent volatility of an RF reader, for our evaluation, we only modulate the transmit power. Despite this, RF traces are naturally more frantic and change more rapidly than any solar traces generated. Kinetic harvesting traces were generated by the shake-table by modulating the frequency and amplitude of the vibrations. Kinetic harvesting traces were especially sensitive to ambient vibrations and noise and were very difficult to reproduce consistently.

**I–V surfaces:** From the randomly generated light-box traces, the transmit power traces gathered in the RF-box, and the kinetic shake-table traces, I–V surfaces are generated using the previously mentioned *smart*-load. Parameters such as cycle frequency (number of times a second the smart load goes through all its resistance settings), and capacitance are chosen so as to give the best results for each surface type. Choosing different capacitance or period values can have a significant effect on the final granular accuracy of the recorded surface, as discussed in Section 3.2.3. Drawing on those observations, surfaces generated for RF emulation were gathered with smoothing capacitance $<0.1\,\mu F$ and very high cycle frequency, while the solar surfaces had smoothing capacitance $10\,\mu F$ and much lower cycle frequency. When capturing kinetic surfaces, no smoothing capacitance was added, as the computational platform used for emulating kinetic I–V surfaces was significantly faster than the one used for solar and RF.

**Constant power surfaces:** Using the light-box traces mentioned above, we generate constant power surfaces from recorded power traces captured as different programs execute. Each power surface is generated from a single recorded trace chosen arbitrarily from a set of device runs. We create constant power surfaces for each of the three EZ430-RF2500 programs mentioned while running on a light-box trace.

**Distance metrics:** To evaluate the physical accuracy of Ekho, a metric was needed to compare two I–V curves. This is difficult for two reasons. First, an I–V curve relates two incommensurable units (Volts and Amperes), this renders as meaningless any euclidian distance from the curve. Second, there is not a one-to-one mapping between an observed (I,V) pair and an emulated (I,V) pair. The observed point could correspond to any number of points on the curve being emulated. In our development of Ekho, we have explored two metrics, current error (assuming the observed voltage is correct and measuring the difference in current) and voltage error (assuming the observed current is correct and measuring the difference in voltage). The current error is amplified (even for points very near the surface, as shown in Figure 16) when the voltage is high and the I-V curve is steep. Voltage errors are similarly amplified when the voltage is low, and current is high.

Using these emulation platforms, test devices, test programs, programmable environments, harvesting traces, surfaces, and metrics, we evaluate Ekho's ability to record and recreate energy harvesting traces produced by the light-box, RF-box, and shake-table. We measure the accuracy and consistency of Ekho, explore the the experimental characteristics that affect the system's performance, and demonstrate the generality of Ekho. We attempt to answer these questions:

(1) *How consistent/repeatable is Ekho?*
(2) *How accurately, in terms of behavior and physical conditions, can Ekho emulate energy environments?*
(3) *Is Ekho able to record and emulate multiple types of energy harvesting environments effectively?*

Before we can explore these questions, we first need to understand exactly why simulating constant power is not sufficient for accurate testing of small, capacitor powered energy harvesting devices, we show our results in Section 5.2. We then explore solar harvesting consistency results of Ekho in Section 5.3, and show solar recording and emulation accuracy of Ekho in Section 5.4. We show consistency and behavioral accuracy results of Ekho with regards to RF energy harvesting in Section 5.5. We then study the accuracy of recording and emulation of kinetic energy sources using Ekho in Section 5.6. Lastly, we compared the recording performance of the desktop version of Ekho and the mobile version in Section 5.7.

Table 2. Program behaviors are shown for three test programs, when harvesting energy from the light-box and from the Ekho emulator.

| | Program behavior (flash writes) | | | |
|---|---|---|---|---|
| | Ekho | | Light-box | |
| **Program** | *mean* | *stddev* | *mean* | *stddev* |
| Static | 326.0 | 4 | 291.6 | 8 |
| SemiAdaptive | 441.2 | 14 | 418.0 | 35 |
| Adaptive | 11.6 | 9 | 11.2 | 11 |

### 5.2 Emulating P vs. I–V

To evaluate our claim that emulating power is not sufficient to simulate an actual energy harvesting environment in a deployment, we created constant power traces gathered from executing our three sample programs (Static, SemiAdaptive, and Adaptive). Each power trace was generated from one run of a program. We then compared the behavior (flash writes) by running each program on the light-box five (5) times, then on the Ekho generated I–V surface five (5) times, then on all three of the generated constant power-surfaces five (5) times each. We measure differences in program behavior by recording the number of successful writes to flash memory that were performed by each test run. Table 1 shows the results of emulating constant power using power traces recorded at program execution. By using constant power to emulate what is actually an I–V curve, behavior (here shown as flash writes) is dramatically different than deployment behavior as compared to the light-box behavioral results in Table 2. Table 1 shows that constant power-surfaces generally underestimated or overestimated available energy for all programs except (as would be expected) the program that the constant power-surface was generated from. In some cases the error was very apparent; when using Static as a training set, running SemiAdaptive gave half the amount of expected flash writes; when using SemiAdaptive as a training set for a constant power surface, and running Static on that surface, the flash writes were severely overestimated.

The choice of which run to use to generate the power trace can have a large impact on the emulated behavior. For example, if an outlier run (where lower or higher than average flash writes occurred) was arbitrarily chosen as a training set to generate a constant power surface, programs ran on the generated surface could significantly differ from the average run. This is apparent in the Adaptive column and row of Table 1, where runs on the Adaptive Training surface produced significantly less flash writes than the average Adaptive run on the light-box shown in Table 2. These results confirm what was previously shown in Figure 6, emulating power is not effective for devices with a volatile supply voltage.

### 5.3 Reproducing Program Behavior

The primary objective of Ekho is to make device behaviors consistently repeatable in spite of variations due to the energy harvesting. Our first experiment examines Ekho against this goal.

In this experiment, we recorded a randomly generated light-box trace using Ekho. We then use Ekho to emulate the recorded surface fifteen (15) times, five (5) times using each of our three test programs as the test device. As a point of comparison, we also run each of our three test programs five (5) times powered by the light-box directly, using the same randomly generated trace. We measure differences in program behavior by recording the number of successful writes to flash memory that were performed by each test run. As we noted in Figure 7, the lightbox variation would increase as

Table 3. Physical consistency over multiple runs is shown with the three test programs. The same light-box traces were recorded, and then emulated by Ekho with the same load to compare the physical consistency and repeatability of experimentation. Ekho performs with nearly the same physical consistency as the light-box.

| | **Physical error by program** | |
|---|---|---|
| | **Ekho** | **Light-box** |
| **Trace** | *mean* | *mean* |
| Static | 66.3 µA | 50.6 µA |
| SemiAdaptive | 72.6 µA | 56.2 µA |
| Adaptive | 67.3 µA | 53.1 µA |

Table 4. Emulation error—the distance of an emulated point from the intended I–V surface—is shown for the three test programs, while emulating a Ekho-recorded randomly-generated light-box trace. Ekho has a slightly higher error rate on the high voltage, low current area of the I–V curve, this is because slight changes in voltage are accompanied by large changes in current; however, this area of the curve is generally avoided as it denotes an inefficient use of harvested energy.

| | **Emulation error** | |
|---|---|---|
| **Program** | *mean* | *stddev* |
| Static | 87.2 µA | 46.7 µA |
| SemiAdaptive | 86.9 µA | 46.2 µA |
| Adaptive | 88.9 µA | 72.5 µA |

temperature increased after long and consistent use. To account for this, between each lightbox run, we waited one minutes to allow the lightbox to achive thermal equilibrium. Ekho does not suffer from this inconsistency problem, and is not suscpetible to tempeature variations.

Table 2 shows the results of this experiment. For each program the average number of flash writes per test run, and the standard deviations are shown for both the Ekho and the light-box test runs. This table shows the result for a single light-box trace; however, we have found these results to be consistent across all of the randomly generated traces, we have tested. For all three programs, the behavior of the devices under Ekho emulation closely approximates the ground-truth behaviors. Behaviorally, Ekho was more consistent and had a smaller standard deviation in flash writes for each test program. In each case, Ekho emulation was comparable in physical consistency with the light-box output as shown in Table 3.

### 5.4 Emulating Solar Energy Sources

Ekho is designed to make program behaviors deterministic, by accurately and consistently reproducing the physical energy harvesting environments that determine program behaviors. In order to determine how well Ekho reproduces the physical energy harvesting environment, we also measure the characteristics of the emulated energy harvesting conditions.

Table 4 shows the measured current error between the emulated surface and the intended I–V surface. For each program the mean emulation error (the distance in µA from the emulated surface to the intended surface for a voltage) is shown, as well as the standard deviation of this error. The choice of current error is arbitrary, as voltage error could also be used to the same result; both values are derived through the mechanisms discussed in Section 5.1. These results are for a single,

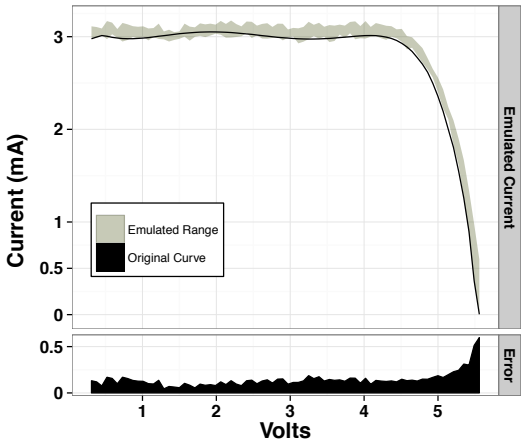



Fig. 16. Shown in the top part of this figure are the target curve Ekho is emulating, and the actual range emulated for a number of test runs on a program. The bottom part of the figure shows the error amount for different parts of the emulated curve. As the slope increases, the current error increases past the knee of the I–V curve.

representative light-box trace; other traces tested produced comparable results. For all programs, Ekho was able to reproduce I–V surfaces accurately enough that behavior remained consistent.

Emulation error is also influenced by the natural shape of the I–V curve as shown in Figure 16; on the high voltage part of the curve, past the knee the error rate increases as the slope of the curve increases, since minimal changes in voltage come with large current changes. While Ekho was not as consistent in emulation error in this area of the curve, this is not as important, as energy harvesting sensors start wasting energy (that could power useful computation) when they enter the steep high-voltage end of the curve that denotes a full capacitor with minimal processing/current draw. For example, a deep sleep program (refer to Figure 1). This is a departure from a traditional sensors application paradigm; which uses duty cycling to prolong battery life, and therefore extend the lifetime of the sensor.

### 5.5 Emulating RF Energy Sources

In this experiment we profiled the program behavior of devices powered through the RF-box (detailed in Section 4). We then compared those results to the same device powered by Ekho instead. To profile RF-box behaviors, we placed the detached RF energy harvester from a Moo inside the RF-box over the reader antenna. We then took another Moo and connected it to the output of the harvester, outside of the Faraday cage so that the RF signals would not interfere with each other. This provided a repeatable, programmable RF energy environment that served as a ground truth for all our RF experiments with Ekho. We ran similar consistency experiments as were run on the light-box; the Moo under test was programmed to sense its internal ADC for temperature five times, average that data, then perform CRC on the data. It did this as many times as it could before brownout. In this experiment, we did not write to flash as in other programs due to the power and voltage requirements, which are difficult to achieve in ultra-low power CRFID platforms. Using the DAQ and Ekho, we monitored the number of CRC calculations performed and when they occurred, and the times that the Moo lost power. We conducted these experiments multiple times and found that the RF-box was reasonably consistent (as shown in Table 5) in regards to program behavior. However, we found that

Table 5. Program behaviors are shown for the Sense-and-CRC test program running on the Umich Moo, when harvesting energy from the reader inside the Faraday cage and from the Ekho emulator (emulating the recorded RF I–V surface). The number of successful Sense-and-CRC readings were counted and compared. Additionally, the total harvested energy is shown for each transmit power. For all transmit powers, the Ekho-powered devices closely approximate the ground truth behaviors. Ekho reproduces these behaviors with significantly better consistency than the RF-box, especially for lower transmit power. Note that since the entire surface is stored in RAM of the I–V curve controller, the error rates are much lower than with solar emulation.

| | | **RF program behavior (CRC)** | | | | | |
|---|---|---|---|---|---|---|---|
| | | **Ekho** | | | **RF-box** | | |
| **Transmit power** | **Harvested energy** | *mean* | *stddev* | *error* | *mean* | *stddev* | *error* |
| +21.25 dBm | 0.55 mJ | 23.6 | 0.6 | 2.3% | 21.0 | 8.3 | 39.4% |
| +27.75 dBm | 2.57 mJ | 208.7 | 0.7 | 0.3% | 189.2 | 39.1 | 20.7% |
| +32.5 dBm | 3.88 mJ | 237.3 | 1.3 | 0.5% | 266.2 | 12.5 | 4.7% |

Table 6. Emulation accuracy for kinetic I–V surfaces.

| | | **Emulation error** | |
|---|---|---|---|
| **Load** | *energy* | *mean* | *stddev* |
| Low Impedance Load | 709.7 µJ | 23.7 µA | 3.5 µA |
| Medium Impedance Load | 713.8 µJ | 15.0 µA | 1.8 µA |
| High Impedance Load | 685.1 µJ | 38.7 µA | 2.3 µA |

at low transmit power, the consistency of the RF-box deteriorated dramatically. We attribute this to timing issues with synchronizing the RF-box, the DAQ, and the host computer. Using a dedicated (but expensive) signal generator could provide a more repeatable RF energy environment at low transmit power.

To evaluate Ekho's behavioral performance with volatile RF energy, we recorded and then constructed an I–V surface generated from each of three different transmit power levels using Ekho. To vary the I–V surface, we at first varied the transmit power over time; this proved unnecessary as an RF surface changes upwards of 1000 times a second for an arbitrary transmit power. We then used Ekho to emulate each recorded surface nine (9) times, for the Sense-and-CRC program running on the Umich Moo. We measure differences in program behavior by recording the number of CRC calculations that were performed by each test run.

Table 5 shows the results of this experiment. For each program the average number of CRC calculations per test run, the standard deviations, and the error rates are shown for both the Ekho and the RF-box test runs. This table shows the results for RF traces that were 120 ms in length. While this may seem a short timespan, because of the volatility of RF energy, eighty (80) different I–V curves were emulated in this window. In all cases, the behavior of the devices under Ekho emulation closely approximates the ground-truth behaviors of the RF-box. Behaviorally, Ekho was significantly more consistent and had a smaller standard deviation and error rate for each transmit power, but especially for the lower transmit powers.

### 5.6 Emulating Kinetic Energy Sources

In this experiment we captured a kinetic I–V surface using the shake-table (detailed in Section 4) and Ekho. Using the emulation feature of Ekho we ran three different constant loads over the surface, and recorded the emulation error (difference between the expected and emulated current) for each, similar to the experiment in Section 5.4 and Table 4. The results of this experiment are shown in Table 6. The total energy, mean distance from the expected curve, and standard deviation from the expected curve are shown. As shown, Ekho is able to reproduce kinetic I–V surfaces on medium impedance loads to within 15.0 µA. For high and low impedance loads for this kinetic I–V surface, emulation error is slightly higher. Physical emulation results for kinetic I–V surfaces are notably better in terms of accuracy than solar. This is not because of some inherent difference in the harvesters, but is owed to the updated, faster hardware used in emulating kinetic surfaces using the ARM Cortex M-4 as opposed to the XMega. If previous solar experimentation were replicated using the newer hardware, we expect emulation error would be reduced.

### 5.7 Mobile Ekho

For our evaluation, we compare the recording performance of the desktop version and the two mobile versions in terms of size, applicability, and energy requirements. Both mobile versions of Ekho were designed with three goals in mind, namely: 1) to be small enough to be deployable, 2) to be an all-in-one solution for recording different I–V surfaces, and 3) to have a low enough power budget to be deployed for a week or more.

**Mini Ekho:** Mini Ekho is nearly ten times smaller than the desktop version; the desktop version is 97.79 mm x 75.87 mm, while Mini Ekho is 21.92 mm x 36.32 mm. This small form factor enables easy deployment and testing.

The mini version satisfies the all-in-one requirement by replacing the PC and DAQ with a dedicated ARM microcontroller. The desktop unit requires extra hardware support (a DAQ for I–V measurement and a PC for surface management) that is built in to the ARM microcontroller. While some accuracy is sacrificed per sample as the DAQ uses a 16-bit ADC and the mini version has a 12-bit ADC, this is overcome by averaging multiple samples to approximate an I–V surface. While recording speed on the mini version is lower than the desktop version by a factor of two, this is a non-issue, as recording speed of the mini version is still over 1000 curves per second (depending on the smart load duty cycle); well beyond the maximum emulation speed of Ekho. Mobile Ekho also has a greater I–V surface range because of better component choice for the digital potentiometer; while desktop Ekho can sense currents as low as 10 ÂţA, mini Ekho can sense as low as 0.05 ÂţA, allowing a wider evaluation of potential energy environments.

The mini and desktop units are very different in terms of energy usage. Desktop Ekho must be plugged in and draws a minimum of 100 mA at 9 V. Mobile Ekho can be powered by a Li-Po battery, and typically draws 38 mA when recording and 138mA when writing a page to the SD card. Mobile Ekho can last for over two weeks in deployment in a solar environment powered by a 1750 mA h battery at 3.6 V, recording a single I–V curve a minute during the daylight hours.

**Micro Ekho:** Micro Ekho, is 34 mm x 34 mm, not including the coin cell battery and enclosure. Just as with Mini Ekho, Micro Ekho is an all-in-one solution, but tuned for ultra low power operation, allowing longer, connected deployments. Micro Ekho has reduced accuracy and precision, as it only has a 10-bit ADC, and can only acquire five different points on the I–V curve at a time. However, the long lifetime and long range radio allow it to be placed semi-permanently.

### 5.8 Complicating Factors

Different decisions when recording, and emulating I–V surfaces influence the final accuracy and consistency of Ekho. Because of the hardware limitations of the I–V controller (specifically the serial communication speed), Ekho using the XMega microcontroller emulation platform is only able to emulate I–V surfaces that switch curves at a maximum rate of 135 Hz. That speed can be increased by using the ARM as the emulation platform. Since RF surfaces can switch curves upwards of 1000 times a second, when emulating RF, those surfaces must be held entirely in RAM on the XMega microcontroller or ARM I–V controller. This allows a curve switching speed beyond 1 kHz (for 65 point I–V curve representations), but limits surface length to sub-second levels on the XMega microcontroller, and 1-2 seconds on the ARM Cortex M4. To simulate longer surfaces, either a larger RAM must be used, or other types of fast access external memory be made available (FRAM, or an SD card), or a faster BUS implemented in hardware. Another factor is response time—the I–V curve controller on the XMega microcontroller is able to respond to current fluctuations in 4 µs. Added capacitance in the circuit increases accuracy as it allows more time for the controller to respond to changes in current, but this also decreases the maximum effective curve switching rate. This can cause Ekho to skip curves when emulating, and thereby reduce accuracy of the whole surface. In emulation, the speed at which the controller can deliver curves limit the range of surfaces that can be emulated. Also, the number of points used to represent a curve in memory influences this. Choosing fewer points (for example 33 instead of the current 65) can increase emulation speed. Tradeoffs between emulation speed, and accuracy can be made in multiple places throughout the recording/emulation chain. Many of the same tradeoffs detailed in Section 3.2.3 apply to emulation as well.

## 6 RELATED WORK

We proposed the conceptual framework for Ekho at HotPower 2011 [43] and introduced the idea of I–V curve-based emulation of energy harvesting conditions, but presented only a cursory evaluation of a prototype that lacked the ability to record and estimate I–V curves and could only emulate single static I–V curves with error rates as high as $170\mu A$. Our work builds on the ideas proposed in HotPower 2011 [43], and demonstrates that I–V surfaces can be accurately recorded and emulated, in order to provide both realistic and repeatable experimentation.

**Emulation Techniques:** Another closely related project, SunaPlayer [2] provided an analog hardware solution, specifically designed for larger solar harvesting applications, which used a high gain Darlington transistor to approximate the shape of a solar I–V curve. They also used a model of solar output based on temperature, humidity, and ambient light conditions to capture solar harvesting conditions. While this approach does allow for capture and replay of energy harvesting conditions, this approach is limited to a single energy harvesting technology (solar), while Ekho can record and emulate Solar and RF. Additionally, the latency of converging to a specific point on the I–V curve for emulation can reach as high as ten seconds, which is prohibitive for some applications. Ekho converges on an emulation point in a matter of microseconds.

Analog battery simulator B# [5, 30] is tangentially related to Ekho, in that it measures a current load, then computes a voltage from a battery simulator and mimics that voltage on a regulator. However, B# is only applicable to specific battery chemistries, and does not support recording of an actual I–V surface (it uses a battery simulator as a voltage lookup), nor does it have the ability to emulate volatile energy sources like RF and kinetic, or even solar. Ekho provides a more generalized approach that works with surface mount capacitor powered devices and does not rely on

computationally expensive simulation models or profiling specific battery chemistries. Ekho is not a battery simulator, nor is it meant for devices that use batteries.

S# extends B# by emulating large wattage solar panels[24], with the intention of saving designers time by testing in the lab, before deployment, much like Ekho. As opposed to Ekho, S# ignores I–V characteristics and instead uses a simplified approach to emulation, by taking the input current and the amount of sunlight to determine the final power generated at the load. This approach works well because the devices S# is meant for have stable supply voltages. Ekho is specifically engineered to work with unstable supply voltages. Additionally, like B#, S# only works for solar cells, and a very specific range of solar cells (1-24V). Ekho can work with input currents and voltages much lower, fitting in the vision of enabling extremely power constrained, capacitor based, energy harvesting sensors. This also allows Ekho to emulate all the energy harvesting sources we highlighted in this paper.

EmPro [29] is an wireless sensor emulator that takes into account environmental variations, as well as sensor inputs. EmPro emulates power sources (only batteries, and solar panels), radio attenuation, and plays back sensor inputs. EmPro does not however, take into account how the load of a sensor can change the voltage, and therefor the amount fo energy harvested. While EmPro provides a more holistic approach by attempting to emulate most of the concern of a wireless sensor node, EmPro is constrained to only two power sources, high wattage solar panels, and batteries. Ekho generalizes the energy harvesting problem that EmPro ignores, allowing for different hardware and software sensing solutions to be tested against multiple energy sources without changing designs.

**Model Driven Simulation:** Other related work include simulation tools for low-power sensors [10, 27, 36], some of which support RFID-scale sensing by considering I–V relationships when simulating harvesting conditions [14]. As described previously, these techniques are able to provide many of the same benefits as Ekho, but at a higher maintenance cost (models need to be updated to support new hardware). When emulating using a model, a user must record every feature of the physical world that is relevant to the model. This often makes models very specialized, making them useless when new harvesting paradigms are developed. Ekho records the electrical properties of the harvester in question. It doesn't need different sensor readings for each harvester. Harvester models are complementary to Ekho, as the I–V surfaces Ekho (since they are in a digital format) could be used as input to other simulators like the ones mentioned. Even instruction level simulators like [10] could benefit from using Ekho I–V surfaces to inform their energy model, we discuss this further in Section 7.

**Profiling and Efficiency Tools:** Finally, a number of tools make it possible to measure and characterize the energy consumption of embedded devices [18, 38]. In addition to enabling energy aware software systems, we envision these technologies being extended in order to allow sensor devices to profile their own harvesting conditions in order to better predict their future energy budgets and operate more closely to their power harvesting potential.

## 7 DISCUSSION & FUTURE WORK

Ekho is designed to be the multipurpose tool for recording and emulating a wide range of energy harvesting environments, provided in a single integrated package. Based on the results described in the previous section, Ekho promises to make it possible to experiment with a wide range of low-power, energy harvesting devices, to an extent that has, to date, been infeasible. In spite of this promise, our current implementation is limited in a number of important ways. This section discusses these limitations and our efforts in the coming months on future work.

**Ease of Use:** Ekho is meant to simplify design, testing, and experimentation with tiny, batteryless sensors. This is achieved with a simple recording and emulation interface, and a suite of software tools. To record with Ekho, the developer only has to plug in the leads of the energy harvester of interest, into the receiving inputs of Ekho, and then either deploy the Ekho (most common for solar), or actuate the harvester in the lab (for example with an RFID reader). The generated I–V surfaces are then recorded and digitized to a desktop computer, or microSD card for later use. To emulate with Ekho, all the designer has to do is replace the energy harvester on the device that will be tested with Ekho. After connection, the developer can use the developer tools (detailed in Section 4) that allow visualization of recorded I–V surfaces as well as the ability to watch emulation in progress.

**Hardware Constraints:** Many of the limitations of our Ekho prototype stem from current hardware restrictions that can be overcome by readily available parts. The 7.4 ms curve update limitation when emulating energy environments is the result of speed constraints imposed by the USB-serial port on the XMega microcontroller. We have shown that we can overcome this limit using on-board RAM to store surfaces, which was necessary for RF emulation. We have also shown that managing the emulation loop with the Teensy ARM Cortex-M4 microcontroller, improves the error results, as curve updates happen sub-milli-second. Moving our emulation platform to be exclusively on the ARM Cortex-M4 will allow Ekho to emulate I–V surfaces that require more frequent I–V curve updates, and will also allow longer RF I–V surfaces. Replacing our current 12-bit ADC and DAC with faster 16-bit models will also improve accuracy and measurement ability.

**Smart Load:** The synthetic "smart load" used to explore the I–V surface could potentially be invasive or detrimental to the harvester under test. Some harvester sources (such as solar) can generate a significant amount of energy that if they have an excess of energy not used by the load. If this energy is not discharged after some amount of time, it will burn off in the form of heat, potentially damaging the harvester or changing the I–V surface[4]. The "smart load" may have the potential to cause this type of situation. In our experiments, we have not characterized or observed this effect for the small, low energy harvesters that Ekho is designed for, possibly because the "smart load" explores the surface very quickly, not allowing excess energy to be stored for long in the harvester. We expect this could become a problem when using Ekho to record much larger wattage systems (specifically solar panels). To address this for larger energy harvesting systems, designers could tune the load range of the "smart load" using information about the harvester under test. By constraining the range and avoiding the edges of the harvester's I–V curve, this heat effect could be avoided, using information that would already be in hand. However, we consider this an open question, and are interested in exploring, and characterizing this effect in small harvesters that are used in batteryless sensors.

**Environmental Factors:** Ekho does not necessarily take into account source-centric concerns that affect the performance of the energy harvester. For example, with solar energy, the irradiance and the temperature are not taken into account. For kinetic energy the accelerations (and possibly angular velocity) are ignored. For RF harvesting, the source attenuation and humidity of the air are not considered. Ekho records the electrical properties of the harvester in question. It doesn't need different sensor readings for each harvester. This makes Ekho more general than other approaches. However, since Ekho generates a digital representation of an energy harevsting surface, it is not far fetched to imagine that these other environmental conditions can be tagged as meta-data to any surface that is recorded. For example: an solar I–V surface could be recorded alongside a temperature and luminance sensor. This serves the purpose of mapping the environmental conditions to the I–V surface of a particular harvester. This metadata could be kept together along with location, time of

[4]However, this would likely not change the device behavior as this would only happen when the device has more than enough energy to perform its tasks.

day, and other information, gibing designers more information on how different energy harvesters work in different conditions. We envision creating a globally available repository of I–V surfaces tagged with this metadata, that will allow researchers to coordinate and share environments and results in a common format.

**Automation:** We also plan to explore ways to automatically tune some of Ekho's system parameters, like the window size used to infer I–V curves, the smoothing capacitance, and the custom "smart" load cycle frequency during record. Automatically detecting the window size by actively detecting the level of movement across an I–V curve and adjusting the window size to use the smallest possible complete data set, will make Ekho easier to use and improve recording precision. Automatically adjusting the period and capacitance of the smart load will also make the recording process more straightforward. We anticipate that simple switching circuitry could change the capacitance, per test run, however, as demonstrated with our newest version of Ekho, often this smoothing capacitance is unnecessary if the recording speed is fast enough. We anticipate not using any smoothing capacitance in future versions of Ekho.

**Integration:** Simulation approaches are often preferred when developing new protocols and applications for sensor networks. This allows testing, debugging, analysis, and system scalability investigations when time is limited and cost is a factor. Well-known sensor network simulators such as Cooja [11] for Contiki [8] and TOSSIM [22] for TinyOS [23] only have a power consumption model, i.e. Energest [9] and PowerTOSSIM [35], respectively, for better understanding the energy consumption of sensor network applications. The lack of support for emulating energy harvesting conditions limits the development of energy harvesting-based applications in these simulators. A tool that utilizes environmental data to compute the harvested power in Contiki is available [7]. However, unlike Ekho, this tool estimates the amount of energy harvested using traces of light values with some strong assumptions for harvesting efficiency, charging efficiency and no battery leakage. With Ekho, it is possible to feed the simulators with traces from gathered I–V surfaces to simulate sensor networks with more realistic energy harvesting setting.

We find this integration most interesting for instruction level device simulators, especially MSP-Sim [10]. One of the main limitations with Ekho (and emulation in general) is that the emulation can not be sped up, so it becomes difficult to exhaustively test many different harvesting conditions. By combining the speed of simulation with the accuracy of Ekho I–V traces, hundreds or thousands of energy harvesting conditions could be explored for a particular device or application, without spending a significant amount of time. We envision using I–V surfaces pre-recorded with Ekho to generate a realistic energy harvesting model for MSPSim. We leave this integration for future work.

## 8 CONCLUSIONS

In this paper, we have described the design and evaluation of Ekho, an emulator that makes reproducible experimentation with capacitor powered energy harvesting devices possible, without the need for hardware and harvester models (required by simulators). Ekho is able to record energy harvesting conditions and accurately recreate those conditions in a laboratory setting consistently. We have also described the design and use of a mobile version of Ekho; allowing profiling of energy environments at locations where sensors are already in place, or will be in place.

Ekho is a general-purpose tool that supports a wide range of harvesting technologies. We have demonstrated, using a working prototype, that Ekho is capable of reproducing harvesting-dependent program behaviors by emulating solar energy harvesting conditions accurately to within 77.4 μA, and more consistently than our light-box, a programmable solar harvesting environment. We also found that Ekho can reproduce kinetic energy environments with a mean error of 15.0 μA from the

recorded surface. Demonstrating the generality of Ekho; we have shown that Ekho is able to emulate RF I–V surfaces significantly more consistently than our RF-box, for a variety of loads and I–V surfaces. Ekho was able to reproduce RF energy harvesting conditions effectively such that program behaviors were accurate in comparison to the RF-box.

As embedded sensing devices continue to become smaller, with tighter energy constraints, energy harvesting will continue to become more important, and tools like Ekho will make possible the realistic and thorough testing that will be needed to deploy those devices with confidence.

## ACKNOWLEDGMENTS

The authors would like to thank Kevin Fu, Hong Zhang, Ryan Archer, and Negin Salajegheh, for early contributions to Ekho. We thank Charlie McDonald of Clemson Electrical Engineering and Instrumentation for help with building the RF-box. We thank Maren Sorber and Joel Christensen for contributing experimental code.

This research is supported by the National Science Foundation (NSF) under award number ACI-1212680. The views and conclusions contained in this paper are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of NSF.

## REFERENCES

[1] David Ascher, Paul F. Dubois, Konrad Hinsen, Jim Hugunin, Travis Oliphant, and others. 2001. Numerical python. (October 2001).

[2] Stanislav Bobovych, Nilanjan Banerjee, Ryan Robucci, James P. Parkerson, Jackson Schmandt, and Chintan Patel. 2015. SunaPlayer: High-accuracy Emulation of Solar Cells. In *Proceedings of the 14th International Conference on Information Processing in Sensor Networks (IPSN '15)*. ACM, New York, NY, USA, 59–70. DOI: http://dx.doi.org/10.1145/2737095.2737110

[3] Michael Buettner, Ben Greenstein, and David Wetherall. 2011. Dewdrop: An Energy-Aware Task Scheduler for Computational RFID. In *Proceedings of the 8th USENIX Symposium on Networked Systems Design and Implementation*.

[4] Gregory Chen, Matthew Fojtik, Daeyeon Kim, David Fick, Junsun Park, Mingoo Seok, Mao-Ter Chen, Zhiyoong Foo, Dennis Sylvester, and David Blaauw. 2010. Millimeter-Scale Nearly Perpetual Sensor System with Stacked Battery and Solar Cells. In *IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. DOI: http://dx.doi.org/10.1109/ISSCC.2010.5433921

[5] Pai H. Chou, Chulsung Park, Jae Park, Kien Pham, and Jinfeng Liu. 2003. B#: A Battery Emulator and Power Profiling Instrument. In *Proceedings of the 2003 International Symposium on Low Power Electronics and Design*. ACM, 288–293.

[6] Atmel Corporation. 2013. AVR XMEGA Microcontrollers. http://www.atmel.com/products/microcontrollers/avr/avr_xmega.aspx. (October 2013). [Online; accessed 11 October, 2013].

[7] Riccardo Dall'Ora, Usman Raza, Davide Brunelli, and Gian P. Picco. 2014. SensEH: From Simulation to Deployment of Energy Harvesting Wireless Sensor Networks. In *Proceedings of the 9th IEEE Workshop on Practical Issues in Building Sensor Network Applications (SenseApp'14)*. IEEE, Edmonton, Canada, 566–573.

[8] Adam Dunkels, Björn Grönvall, and Thiemo Voigt. 2004. Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors. In *Proceedings of the 1st IEEE Workshop on Embedded Networked Sensors (EmNets-I)*. IEEE Computer Society, Tampa, FL, USA.

[9] Adam Dunkels, Fredrik Österlind, Nicolas Tsiftes, and Zhitao He. 2007. Software-based On-line Energy Estimation for Sensor Nodes. In *Proceedings of the 4th Workshop on Embedded Networked Sensors (EmNets'07)*. ACM, Cork, Ireland, 28–32.

[10] Joakim Eriksson, Adam Dunkels, Niclas Finne, Fredrik Österlind, and Thiemo Voigt. 2007. MSPsim—an Extensible Simulator for MSP430-equipped Sensor Boards. In *Proceedings of the 4th European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session*. Springer.

[11] Joakim Eriksson, Fredrik Österlind, Niclas Finne, Nicolas Tsiftes, Adam Dunkels, Thiemo Voigt, Robert Sauter, and Pedro J. Marrón. 2009. COOJA/MSPSim: Interoperability Testing for Wireless Sensor Networks. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques (SIMUTools'09)*. ACM, Rome, Italy.

[12] Mark Galassi, Jim Davies, James Theiler, Brian Gough, Gerard Jungman, Michael Booth, and Fabrice Rossi. 2003. *GNU Scientific Library: Reference Manual*. Network Theory Ltd. http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0954161734

[13] Maria Gorlatova, Robert Margolies, John Sarik, Gerald Stanje, Jianxun Zhu, Baradwaj Vigraham, Marcin Szczodrak, Luca Carloni, Peter Kinget, Ioannis Kymissis, and Gil Zussman. 2013. Prototyping Energy Harvesting Active Networked Tags (EnHANTs). In *Proceedings of the 32nd IEEE International Conference on Computer Communications (INFOCOM'13)*. IEEE, Turin, Italy, 585–589.

[14] Jeremy Gummeson, Shane S. Clark, Kevin Fu, and Deepak Ganesan. 2010. On the Limits of Effective Hybrid Micro-Energy Harvesting on Mobile CRFID Sensors. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys)*. ACM, 14.

[15] Josiah Hester, Timothy Scott, and Jacob Sorber. 2014. Ekho: Realistic and Repeatable Experimentation for Tiny Energy-harvesting Sensors. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems (SenSys'14)*. ACM, New York, NY, USA, 1–15. DOI:http://dx.doi.org/10.1145/2668332.2668336

[16] National Instruments. 2012. NI X Series Multifunction Data Acquisition. http://sine.ni.com/ds/app/doc/p/id/ds-163/lang/en. (October 2012). [Online; accessed 11 October, 2013].

[17] Texas Instruments. 2013. http://www.ti.com/tool/ez430-rf2500. (October 2013). [Online; accessed 11 October, 2013].

[18] Xiaofan Jiang, Prabal Dutta, David Culler, and Ion Stoica. 2007. Micro Power Meter for Energy Monitoring of Wireless Sensor Networks at Scale. *Proceedings of the 6th International Conference on Information Processing in Sensor Networks* (2007). http://portal.acm.org/citation.cfm?id=1236386

[19] Eric Jones, Travis Oliphant, Pearu Peterson, and others. 2001–. SciPy: Open Source Scientific Tools for Python. (October 2001–). http://www.scipy.org/

[20] Aman Kansal, Jason Hsu, Sadaf Zahedi, and Mani B. Srivastava. 2006. Power Management in Energy Harvesting Sensor Networks. *ACM Transactions on Embedded Computing Systems* (2006).

[21] Ye-Sheng Kuo, Sonal Verma, Thomas Schmid, and Prabal Dutta. 2010. Hijacking Power and Bandwidth from the Mobile Phone's Audio Interface. In *Proceedings of the 1st Annual Symposium on Computing for Development (DEV)*.

[22] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. 2003. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys'03)*. ACM, Los Angeles, CA, USA, 126–137.

[23] Philip Levis, Sam Madden, Joseph Polastre, Robert Szewczyk, Kamin Whitehouse, Alec Woo, David Gay, Jason Hill, Matt Welsh, Eric Brewer, and David Culler. 2005. TinyOS: An Operating System for Wireless Sensor Networks. In *Ambient Intelligence*, Werner Weber, Jan M. Rabaey, and Emile Aarts (Eds.), Vol. II. Springer Berlin Heidelberg, 115–148.

[24] Di-Jie Li and Pai H Chou. 2004. Maximizing efficiency of solar-powered systems by load matching. In *Low Power Electronics and Design, 2004. ISLPED'04. Proceedings of the 2004 International Symposium on*. IEEE, 162–167.

[25] Kris Lin, Jason Hsu, Sadaf Zahedi, David C Lee, Jonathan Friedman, Aman Kansal, Vijay Raghunathan, and Mani B. Srivastava. 2005. Heliomote: Enabling Long-Lived Sensor Networks Through Solar Energy Harvesting. In *Proceedings of ACM Conference on Embedded Networked Sensor Systems (SenSys)*.

[26] Scott Meninger, Jose O. Mur-Miranda, Rajeevan Amirtharajah, Anantha Chandrakasan, and Jeffrey H. Lang. 2001. Vibration-to-Electric Energy Conversion. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 9, 1 (2001).

[27] Geoff V. Merrett, Neil M. White, Nick R. Harris, and Bashir M. Al-Hashimi. 2009. Energy-Aware Simulation for Wireless Sensor Networks. In *Proceedings of the 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'09)*. IEEE, 1–8.

[28] Microchip. 2013. 7/8-Bit Single/Dual SPI Digital POT with Volatile Memory. (October 2013). [Online; accessed 11 October, 2013].

[29] Chulsung Park and Pai H Chou. 2006. EmPro: an environment/energy emulation and profiling platform for wireless sensor networks. In *Sensor and Ad Hoc Communications and Networks, 2006. SECON'06. 2006 3rd Annual IEEE Communications Society on*, Vol. 1. IEEE, 158–167.

[30] Chulsung Park, Jinfeng Liu, and Pai H. Chou. 2005. B#: A Battery Emulator and Power-Profiling Instrument. *IEEE Design & Test of Computers* 22, 2 (2005).

[31] PJRC. 2016. PJRC Teensy 3.2 Microcontrollers. https://www.pjrc.com/teensy/teensy31.html. (April 2016). [Online; accessed 10 April, 2016].

[32] Benjamin Ransford, Jacob Sorber, and Kevin Fu. 2011. Mementos: System Support for Long-Running Computation on RFID-Scale Devices.. In *Proceedings of the 16th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*.

[33] Mastooreh Salajegheh, Yue Wang, Kevin Fu, Anxiao (Andrew) Jiang, and Erik Learned-Miller. 2011. Exploiting Half-Wits: Smarter Storage for Low-Power Devices. In *Proceedings of the 9th USENIX Conference on File and Storage Technologies (FAST)*.

[34] Alanson P. Sample, Daniel J. Yeager, Pauline S. Powledge, Alexander V. Mamishev, and Joshua R. Smith. 2008. Design of an RFID-Based Battery-Free Programmable Sensing Platform. *IEEE Transactions on Instrumentation and Measurement* 57, 11 (Nov. 2008), 2608–2615.

[35] Victor Shnayder, Mark Hempstead, Bor-rong Chen, Geoff Werner Allen, and Matt Welsh. 2004. Simulating the Power Consumption of Large-Scale Sensor Network Applications. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys'04)*.

[36] Victor Shnayder, Mark Hempstead, Bor-Rong Chen, and Matt Welsh. 2004. PowerTOSSIM: Efficient Power Simulation for TinyOS Applications. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*. http://www.eecs.harvard.edu/~shnayder/ptossim/

[37] Jacob Sorber, Alexander Kostadinov, Matthew Garber, Matthew Brennan, Mark D. Corner, and Emery D. Berger. 2007. Eon: A Language and Runtime System for Perpetual Systems. In *Proceedings of ACM Conference on Embedded Networked Sensor Systems (SenSys)*.

[38] Thanos Stathopoulos, Dustin Mclntire, and William J. Kaiser. 2008. The Energy Endoscope: Real-Time Detailed Energy Accounting for Wireless Sensor Nodes. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN'08)*. IEEE, 383–394.

[39] Stewart Thomas, Jochen Teizer, and Matthew Reynolds. 2010. Electromagnetic Energy Harvesting for Sensing, Communication, and Actuation. In *Proceedings of the 27th International Symposium on Automation and Robotics in Construction (ISARC'10)*. IAARC, Bratislava, Slovakia.

[40] Yong Yang, Lili Wang, Dong K. Noh, Hieu K. Le, and Tarek F. Abdelzaher. 2009. SolarStore: Enhancing Data Reliability in Solar-Powered Storage-Centric Sensor Networks. In *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services (MobiSys'09)*. ACM, Krakow, Poland, 333–346.

[41] Daniel Yeager, Fan Zhang, Azin Zarrasvand, Nicole T. George, Thomas Daniel, and Brian P. Otis. 2010. A 9 $\mu$A, Addressable Gen2 Sensor Tag for Biosignal Acquisition. *IEEE Journal of Solid-State Circuits* 45, 10 (Oct. 2010), 2198–2209.

[42] Hong Zhang, Jeremy Gummeson, Benjamin Ransford, and Kevin Fu. 2011. *Moo: A Batteryless Computational RFID and Sensing Platform*. Technical Report UM-CS-2011-020. UMass Amherst Department of Computer Science.

[43] Hong Zhang, Mastooreh Salajegheh, Kevin Fu, and Jacob Sorber. 2011. Ekho: Bridging the Gap Between Simulation and Reality in Tiny Energy-Harvesting Sensors. In *Proceedings of the 4th Workshop on Power-Aware Computing and Systems (HotPower'11)*. ACM, New York, NY, USA, Article 9, 5 pages. DOI: http://dx.doi.org/10.1145/2039252.2039261

Received February 2015; revised March 2016; accepted March 2017