Ekho: Realistic and Repeatable Experimentation for Tiny Energy-Harvesting Sensors

Josiah Hester, Timothy Scott, Jacob Sorber
School of Computing
Clemson University
{jhester, tscott2, jsorber}@clemson.edu

Abstract

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 tool that supports a wide range of harvesting technologies. We demonstrate, using a working prototype, that Ekho is capable of reproducing both solar and RF 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 environments, and can emulate RF energy harvesting conditions significantly more consistently than a programmable RF harvesting environment.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: Microprocessor/microcomputer applications

General Terms

Measurement, Experimentation, Performance, Instrument

Keywords

Energy Harvesting, Emulation, I–V curves, RFID

SenSys'14, November 3–5, 2014, Memphis, TN, USA.

ACM 978-1-4503-3143-2/14/11 ...$15.00
http://dx.doi.org/10.1145/2668332.2668336

1 Introduction

Harvested energy is vital to the success of many mobile sensing applications. No longer limited primarily by battery capacities, smaller sensing devices can be deployed in remote or otherwise challenging environments for much longer periods of time, by gathering solar [14, 16], kinetic [17], or RF energy [3, 21]—devices can even harvest energy from other devices [15]. Regardless of the source, this freely-available energy allows low power sensing devices to operate in a truly untethered fashion, collecting valuable data perpetually, 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, 7, 20, 21, 23] and even the energy harvesting [9, 21] behaviors of sensor devices. Unfortunately, most ignore the impact of device behavior on energy harvesting, and 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, and 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) is processed and stored to be later replayed through a custom analog front-end which serves as a current source. 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.

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 for decades of useful operation.

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; too much or too little energy is equally inefficient and wasteful. Greedily using energy can restrict functionality, while under utilization of harvested energy is wasteful in terms of computation that could have been performed.

[1] depending on capacitance

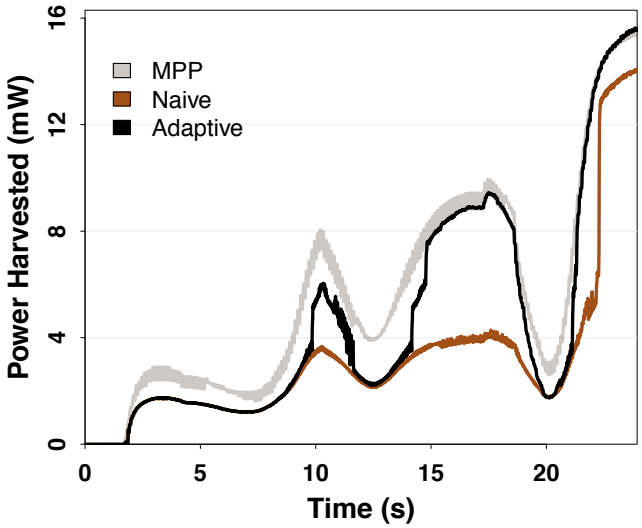


Figure 1. Harvested power is shown for two TI EZ430-RF2500 target boards running different programs that both write to flash in different ways—one writes as fast as possible if there is any power available, the other adapts so it can write even when harvestable energy is scarce—under the same solar energy harvesting conditions. Differences in power consumption result in different amounts of harvested power.

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, 22, 26], 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 3.

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.

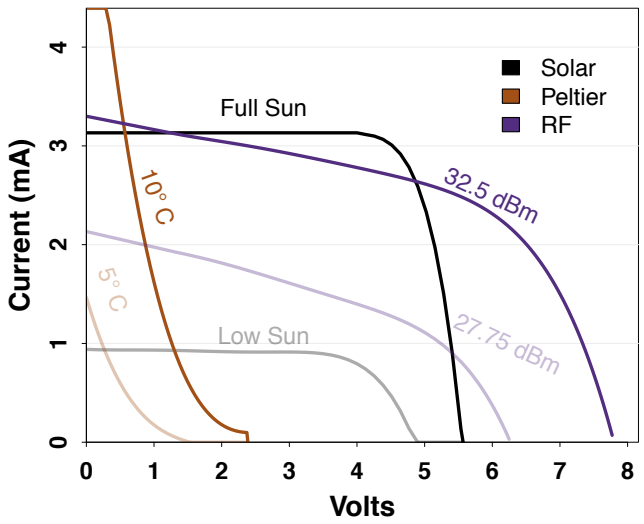


Figure 2. Six I–V curves are shown, produced by three different energy harvesters—a solar panel, a Peltier generator, and an RF Reader—each under two (2) different energy harvesting conditions. Each harvester produces its own "family" of curves, with a common characteristic shape. Each of the above I–V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

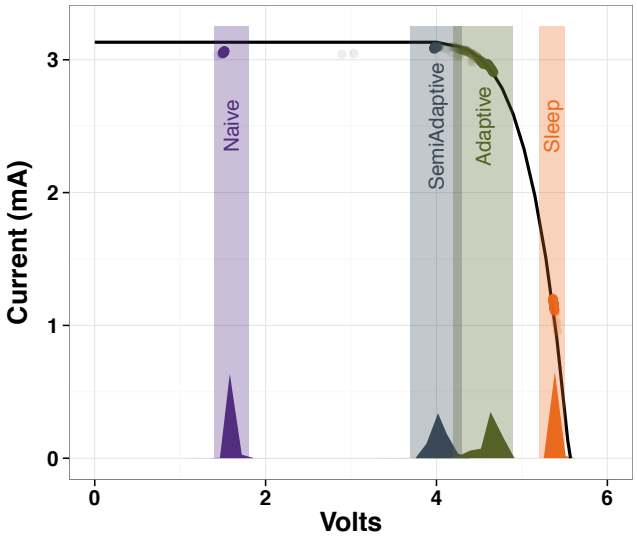


Figure 3. This figure shows how program behaviors influence energy harvesting performance. Four (4) programs' harvested currents, measured over a 20 s period of time, are shown with respect to their supply voltage, while connected to a programmable solar environment (see Section 4) generating a single 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 signifiant variations in harvested power.

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 ($P = IV$).

Figure 1 illustrates this scenario by showing the amount of power harvested by two TI EZ430-RF2500 devices running 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 3), 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.

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 doing this; (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 [24], and our early efforts focused on replaying power traces.

Replaying a power trace is attractive as a predictive technique since designing the hardware is simple and straightforward, and provides a reasonably accurate solution for devices with a constant 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 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, at runtime. Figure 4 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 har-

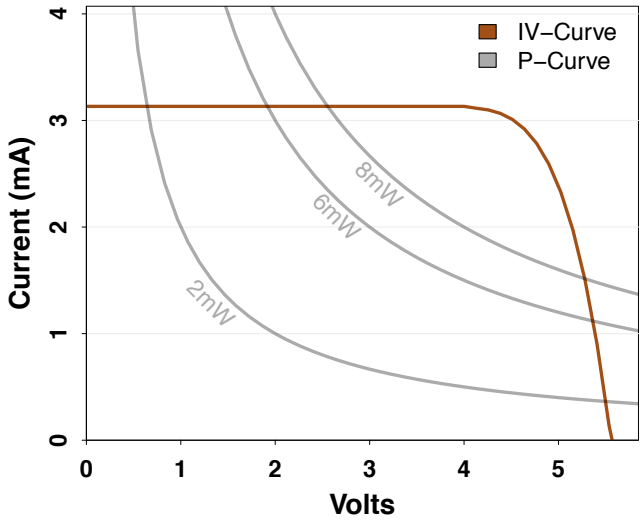


Figure 4. Shown is a single I–V-curve, and the consequences of choosing constant power to represent it. Depending on the load, the generated P-curves from the power trace can cause unrealistic changes in the programs actual harvested energy. As shown, emulating constant power is a poor replacement for emulating the actual I–V-curve.

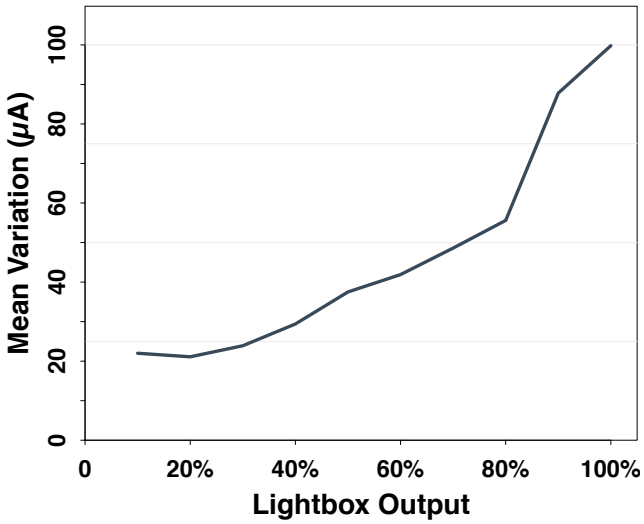


Figure 5. The light-box mean variation between runs increases with light intensity for static loads. The light-box, unlike Ekho is susceptible to environmental changes and care must be taken to control those. Temperature changes after long use are one such factor that affects repeatability.

vesting 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:

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, heat, or vibrations, and create a repeatable environment to provide energy to a system [2]. The shortfalls of these devices come in several key areas. Often, construction of these devices requires significant time and expertise to create an accurate replay environment. These devices also tend to possess many points of failure or errata introduction. These analog solutions to programmable energy environments are much better than many naive approaches that solely simulate power replay.

In developing Ekho, we made extensive use of two such environments, dubbed the "light-box" and the "RF-box". 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 5 shows. The RF-box is constructed so as to isolate the interior from wireless interference; which can cause variation in harvesting current. Inside the RF-box is a programmatically controlled antennae that can power small CRFID tags such as the Umich Moo. By modulating the transmit power different harvesting conditions can be created. Other programmable energy environments take similar steps to isolate sources such as Kinetic energy harvesting with controllable shake tables or heat energy harvesting via Peltier generators between controlled temperature plates.

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, 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 to test devices before deployment. Ekho uses a novel method to explore and record these I–V-surfaces by quickly, and randomly 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.

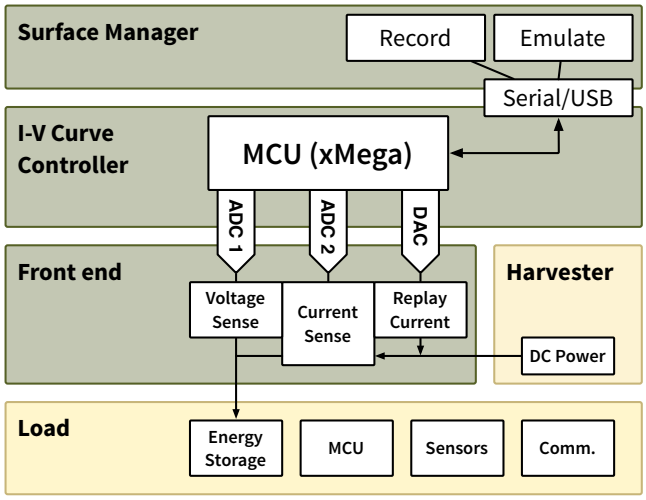


Figure 6. Ekho consists of three interdependent modules: a surface manager that stores I–V surfaces and manages the high-level recording and emulation logic for the system; a low-latency controller that sequentially emulates the I–V curves that correspond to each single point in time during an emulated 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.

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 simply makes the desired change and continues testing.

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

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 6, 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, 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.

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 feature 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], which we use in our prototype, 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 (a standard technique). 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.

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

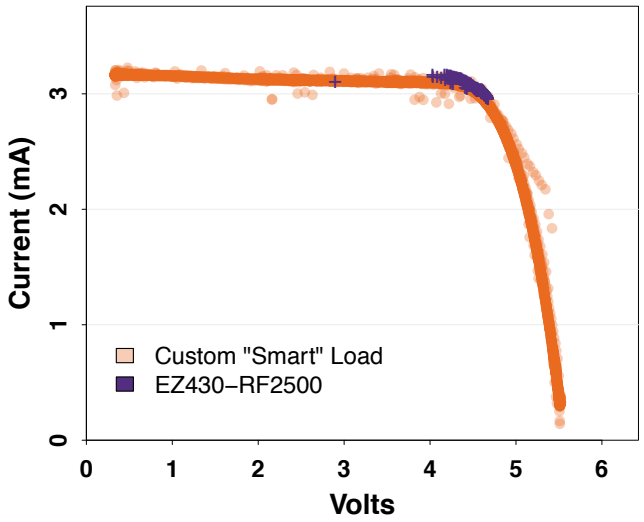


Figure 7. This figure shows recorded I–V measurements, as produced by both the Ekho smart load device and a typical mote-class sensor device. By intentionally increasing the power supply volatility, the smart load provides much better coverage of the I–V curve being recorded, which improves Ekho's recording accuracy.

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 I–V curve changes too much, 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 it's 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 7, 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 [11], 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 100 kΩ digital potentiometer controlled by an Arduino which rapidly alters its power consumption, in order to induce large fluctuations in supply voltage for more accurate 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 different 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 100Hz can capture solar I–V curves, and 1000Hz and above is sufficient to approximate an RF I–V surface

As shown in Figure 7, 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.

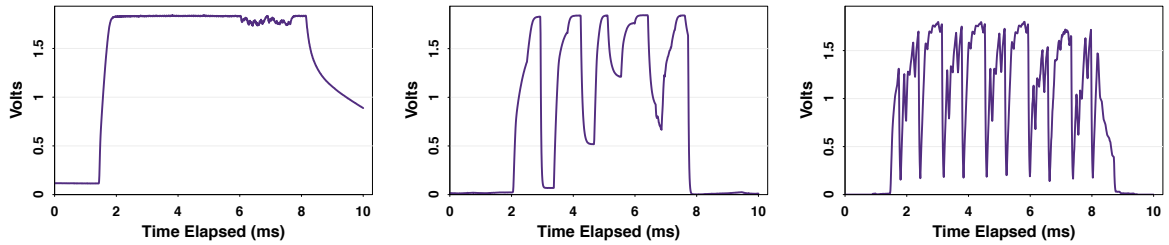
3.2.2 Surface Construction

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. We use the polynomial fitting algorithm provided by the GNU Scientific Library (GSL), 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 trade-off 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 much larger problem as curves can change upwards of one thousand times a second.

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 9, while this is desirable for slow changing solar surfaces, as it reduces noise, and makes for cleaner capture of each individual I–V curve, for fast changing RF surfaces a low capacitance value must be chosen. Because RF surfaces are so volatile, averaging out peaks and valleys in the recorded surface can change the final harvested power, and therefore the final program behavior.

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



(a) Trace with a constant resistive load. Using just this to generate a 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.

Figure 8. RF energy harvesting voltage trace over 10ms, with three different custom "smart" load cycle frequencies.

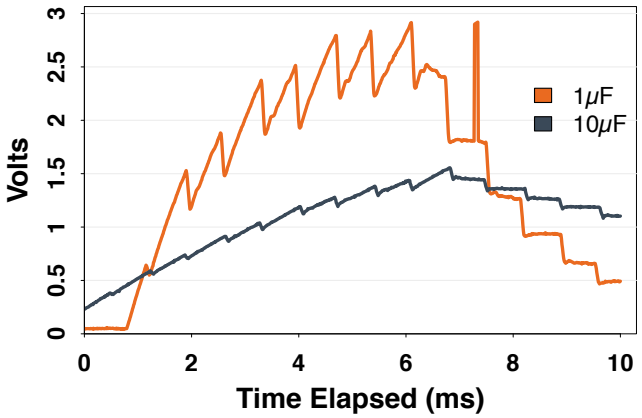


Figure 9. This figure shows the effect of capacitance while recording an RF I–V surface. As the capacitance increases the output is averaged, and important features are lost. Each peak is the custom "smart" load changing it's resistance setting, these peaks are absorbed by the larger capacitance, which means that voltage volatility is lost. Because of this the I–V surface is not as fully explored. For solar, this may be acceptable, for volatile RF harvesting environments, important surface information will be lost.

type, the final surface will be missing potentially important features, alternatively, if it is set too high Ekho may not be able to emulate it fast enough, Figure 8 shows differences in curve coverage and how they can affect the final surface. Ekho is configured to support a wide range harvester types, and therefore can handle many 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 a capacitance less than 0.1 µF and a cycle frequency of at least 1 kHz is required for recording RF 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 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 xMega's 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 stores the entire surface in RAM to facilitate 1 kHz curve updates. For the current prototype, this limits RF surface length to under one second, in future implementations, external memory (FRAM, SDCARD) will allow much longer RF traces to be emulated.

Throughout this process, the I–V curve controller emulates each curve by simply 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 closes 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

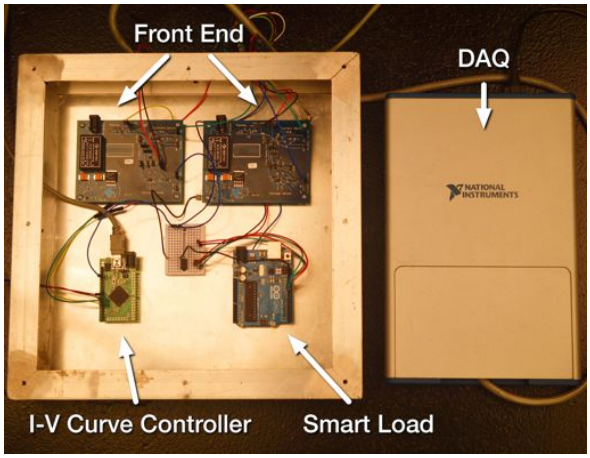


Figure 10. The core of our prototype Ekho implementation, including two custom analog front-end boards, the ATXmega256A3B-based I–V curve controller, and the "smart" load used to explore I–V surfaces during recording. Note that while only a single front-end board is needed for Ekho to function, we include 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.

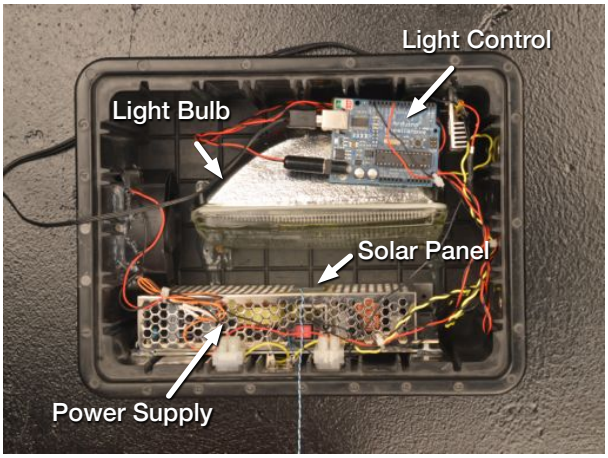


Figure 11. Our prototype light-box implementation provides a reproducible solar harvesting environment that we use in our experiments to provide reproducible "ground truth" harvesting conditions. The implementation consists of a automotive headlight, a solar panel, an Arduino which serves as programmable dimmer-switch, and necessary power supply.

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. Using a larger capacitor to store the harvested energy will cause the supply to change more slowly, giving Ekho more time to respond.

4 Implementation

In order to evaluate the efficacy and usefulness of our approach, we have implemented a prototype Ekho emulator, shown in Figure 10, that is able to record and replay energy harvesting conditions. This prototype consists of a surface controller, an I–V controller, and a custom analog front-end.

The 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) [10] 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 for low-latency emulation. Total cost of the system, including ATXmega, Arduino, custom circuit boards, and parts (excluding the DAQ) is less than $700. In future versions of Ekho, we hope to combine the I–V controller functionality into a single computing device. 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.

We also use the NI USB-6356 to collect voltage and current measurements during our experimental evaluation, as is described in the following section.

In addition to the Ekho apparatus, itself, we have also implemented the software necessary for recording, processing, and emulating energy environments. For recording, we interface with the NI USB-6536 to record a physical energy environment. The NI USB-6536 offered sampling rates up to 1 MHz and usually operates between 200 kHz and 500 kHz. During processing, we used a combination of python, C, and C++ to process and gather relevant data and generate I–V curves. The GNU scientific library [8], as well as Numpy [1] and Scipy [13], provide polyfit and data processing power that generate I–V curves and surfaces from recorded I–V traces. R provided some surface visualization and data verification functionality as well. For emulating I–V environments, we use custom software written in C to handle timing on the PC that is responsible for appropriately timing traces and relaying data to the XMega as necessary. The XMega 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

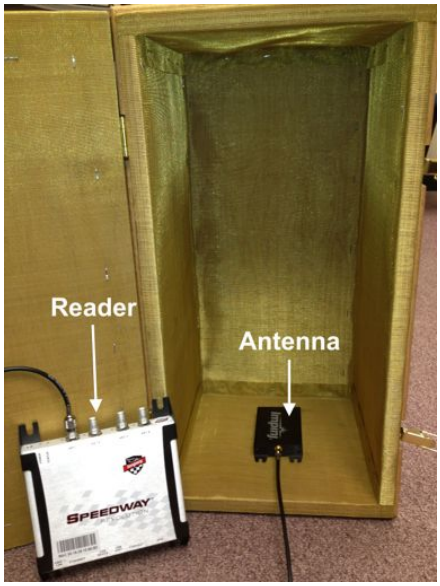


Figure 12. Our prototype RF-box provides a reproducible RF harvesting environment used in our experiments to provide "ground truth" harvesting conditions. The RF-box is composed of a wooden shell layered with brass screen copper mesh seals. Inside the box is an antennae, driven by a programmatically controlled reader.

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 XMega. All code and hardware designs will be made available via our website at publication time.

In order to support more accurate recording (as described in Section 3.2), we have also developed a custom "smart" test load, which rapidly alters its power consumption, in order to induce large fluctuations in supply voltage for more accurate recording. We have implemented this *smart load* using an Arduino Uno to control a digital potentiometer. The potentiometer [19] acts as a resistive load, with 128 settings ranging from 0 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.

The light-box used for much of the energy recording and emulating experiments is shown in Figure 11. As implemented, our lightbox consists of a light source (an automotive headlight), which can provide 256 different intensity settings. A solar panel is mounted inside the chassis which provides shielding from outside light sources. An Arduino Duemilanove-328 uses pulse-width modulation to driver 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 12. This RF-box effectively isolates the interior of the box from radio, wifi and other types of wireless interference. We mounted a programmable antennae 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 antennae, many different I–V surfaces can be created. However, since each transmit power can generate thousands of different I–V curve, this is not always necessary.

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 two programmable physical environments; the light-box and the RF-box. 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 I–V characteristics that vary by less than 68.7 μA[2] 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. 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 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. In our evaluation comparing the accuracy of emulating constant power versus emulating I–V, we emulate a total of 3408 constant 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.

[2] depending on capacitance

	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

Table 1. This figure 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.

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, 2.4Ghz CC2500 radio; a 10 µF capacitor is used to store energy. For our RF experiments, we use the Umass Moo [26], 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.

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 Umass Moo devices run one program—Sense and CRC—that senses the internal temperature of the MSP430 using an onboard ADC five times, averages the readings, then CRC's 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 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 much more frantic and interesting than any solar traces generated.

I–V surfaces: From the randomly generated light-box traces, and the transmit power traces gathered in the RF-box, 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 affect 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 µF and very high cycle frequency, while the solar surfaces had smoothing capacitance 10 µF and much lower cycle frequency.

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 1-to-1 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 13) 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 devices, test programs, programmable environments, harvesting traces, surfaces, and metrics we evaluate Ekho's ability to record and recreate energy harvesting traces produced by both the light-box and the RF-box. 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. Lastly, we show consistency and behavioral accuracy results of Ekho with regards to RF energy harvesting in section 5.5.

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. 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 that the constant power-surfaces generally underestimated or overestimated 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 Lightbox shown in Table 2. These results confirm what was previously shown in Figure 4, 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.

	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

Table 2. Program behaviors (in the form of flash writes) are shown for three test programs, when harvesting energy from the light-box and from the Ekho emulator (emulating the recorded light-box trace). For all applications, the Ekho-powered devices closely approximate the ground truth behaviors. Ekho reproduces these behaviors with better consistency than the light-box.

	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

Table 3. 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 an inefficient use of harvested energy.

Table 2 shows the results of this experiment. For each program the average number of 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 4.

5.4 Emulation Accuracy

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 mea-

	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. 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.

sure the characteristics of the emulated energy harvesting conditions.

Table 3 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, as discussed in Section 5.1. These results are for a single, 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 13; 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 3). 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 antennae. 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 CRC the data. It did this as many times as it could before brownout. Using the DAQ and

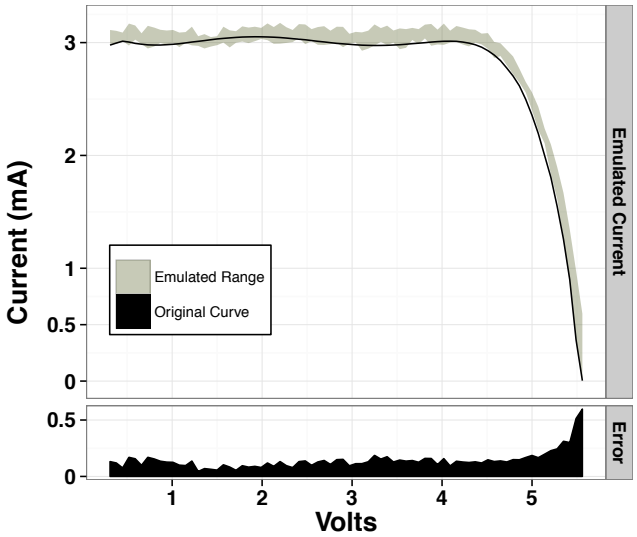


Figure 13. Shown in the top part of this figure are the target curve Ekho is emulating, and the actual range emulated for a number of test runs on a program. The bottom part of the figure shows the error amount for different parts of the emulated curve. As the slope increases, the current error increases past the knee of the I–V-curve.

Ekho, we monitored the number of CRC's 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 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 Umass Moo. We measure differences in program behavior by recording the number of CRC's that were performed by each test run.

Table 5 shows the results of this experiment. For each program the average number of CRC's 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

		RF Program Behavior (CRC's)					
		Ekho			RF-Box		
Transmit Power	**Harvested Energy**	*mean*	*stddev*	*error*	*mean*	*stddev*	*error*
$+21.25dBm$	0.55 mJ	23.6	0.6	2.3%	21.0	8.3	39.4%
$+27.75dBm$	2.57 mJ	208.7	0.7	0.3%	189.2	39.1	20.7%
$+32.5dBm$	3.88 mJ	237.3	1.3	0.5%	266.2	12.5	4.7%

Table 5. Program behaviors are shown for the Sense and CRC test program running on the Umass Moo, when harvesting energy from the reader inside the Faraday Cage and from the Ekho emulator (emulating the recorded RF trace). The number of successful sensed and CRC'd 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.

and had a smaller standard deviation and error rate for each transmit power, but especially for the lower transmit powers.

5.6 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 XMega microcontroller I–V controller (specifically the serial communication speed), Ekho running regularly is only able to emulate I–V surfaces that switch curves at a maximum rate of 135 Hz. 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 I–V controller. This allows a curve switching speed beyond 1000 Hz (for 65 point I–V curve representations), but limits its surface length to sub-second levels. To simulate longer surfaces, either a larger RAM must be used, or other types of fast access external memory be made available (FRAM, or SDCARD), or a faster BUS implemented in hardware. We plan to address this in the future. Another factor is response time—the XMega microcontroller is able to respond to current fluctuations in 4 µs, added capacitance in the circuit increases accuracy somewhat as it allows more time for the XMega microcontroller to respond to changes in current, but this also decreases the maximum effective curve switching rate, causing Ekho to skip curves when emulating, and thereby reducing accuracy of the whole surface. In emulation, the speed at which the XMega microcontroller can deliver curves influences the maximum surface speed. 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 [27] 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 [27], and demonstrates that I–V surfaces can be accurately recorded and emulated, in order to provide both realistic and repeatable experimentation.

Another closely related project provided an analog hardware solution, specifically designed for larger solar harvesting applications [2], 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. It is also difficult to know how well this approach compares to Ekho, since accuracy and consistency results are not available.

Other related work include simulation tools for low-power sensors [18, 23], some of which support RFID-scale sensing by considering I–V relationships when simulating harvesting conditions [9]. 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).

Analog battery simulator B# [5] [20] 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.

Finally, a number of tools make it possible to measure and characterize the energy consumption of embedded devices [12, 25]. 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.

Many of the limitations of our Ekho prototype stem from current hardware restrictions that can be overcome by readily available parts. The current 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 onboard RAM to store surfaces, which was necessary for RF emulation. In the future, improving this update speed by adding more RAM or implementing a faster BUS 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.

Ekho has no inherent constraints that preclude mobile deployment; however, we designed a larger prototype to simplify testing, debugging, and evaluation. Future work on Ekho will involve developing a much smaller mobile version that will enable I-V surface recording in remote field locations. This version will not require the use of a dedicated DAQ.

In order to evaluate other harvesting environments that change more rapidly (like kinetic / vibration), we will also need to develop new programmable energy environments for testing. These will complement our existing light-box and RF-box. We are in the process of developing programmable vibration energy and thermal energy harvesting environments. These will present a new challenge to emulate energy environments that present different I–V patterns and eccentricities. Future work on Ekho will include performance verification using different energy harvesting devices and environments.

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.

8 Conclusions

In this paper, we have described the design and evaluation of Ekho, an emulator that makes reproducible experimentation with 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.

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. 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.

9 Acknowledgements

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. We also thank our shepherd Bodhi Priyantha, and our anonymous reviewers, for their helpful comments.

10 References

[1] D. Ascher, P. F. Dubois, K. Hinsen, J. Hugunin, T. Oliphant, et al. Numerical python, 2001.

[2] S. Bobovych, J. P. Parkerson, and N. Banerjee. Evaluating solar panel-driven systems in the laboratory. In *Proceedings of the seventh ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*, WiNTECH '12, pages 97–98, New York, NY, USA, 2012. ACM.

[3] M. Buettner, B. Greenstein, and D. Wetherall. Dewdrop: An energy-aware task scheduler for computational RFID. In *Proceedings of the 8th USENIX Symposium on Networked Systems Design and Implementation*, 2011.

[4] G. Chen, M. Fojtik, D. Kim, D. Fick, J. Park, M. Seok, M.-T. Chen, Z. Foo, D. Sylvester, and D. Blaauw. Millimeter-scale nearly perpetual sensor system with stacked battery and solar cells. In *IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2010.

[5] P. H. Chou, C. Park, J. Park, K. Pham, and J. Liu. B#: a battery emulator and power profiling instrument. In *Proceedings of the 2003 international symposium on Low power electronics and design*, pages 288–293. ACM, 2003.

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

[7] J. Eriksson, A. Dunkels, N. Finne, F. Österlind, and T. Voigt. 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, 2007.

[8] M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, M. Booth, and F. Rossi. *GNU Scientific Library: Reference Manual*. Network Theory Ltd., Feb. 2003.

[9] J. Gummeson, S. S. Clark, K. Fu, and D. Ganesan. 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, 2010.

[10] N. Instruments. Ni x series multifunction data acquisition. `http://sine.ni.com/ds/app/doc/p/id/ds-163/lang/en`. [Online; accessed 11 October, 2013].

[11] T. Instruments. `http://www.ti.com/tool/ez430-rf2500`. [Online; accessed 11 October, 2013].

[12] X. Jiang, P. Dutta, D. Culler, and I. Stoica. 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.

[13] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–.

[14] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava. Power management in energy harvesting sensor networks. *ACM Transactions on Embedded Computing Systems*, 2006.

[15] Y.-S. Kuo, S. Verma, T. Schmid, and P. Dutta. Hijacking power and bandwidth from the mobile phone's audio interface. In *Proceedings of the 1st Annual Symposium on Computing for Development (DEV)*, 2010.

[16] K. Lin, J. Hsu, S. Zahedi, D. C. Lee, J. Friedman, A. Kansal, V. Raghunathan, and M. B. Srivastava. Heliomote: Enabling long-lived sensor networks through solar energy harvesting. In *Proceedings of ACM Sensys*, 2005.

[17] S. Meninger, J. O. Mur-Miranda, R. Amirtharajah, A. Chandrakasan, and J. H. Lang. Vibration-to-electric energy conversion. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 9(1), 2001.

[18] G. V. Merrett, N. M. White, N. R. Harris, and B. M. Al-Hashimi. Energy-aware simulation for wireless sensor networks. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON'09. 6th Annual IEEE Communications Society Conference on*, pages 1–8. IEEE, 2009.

[19] Microchip. 7/8-bit single/dual spi digital pot with volatile memory. `http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Components/General%20IC/22060b.pdf`. [Online; accessed 11 October, 2013].

[20] C. Park, J. Liu, and P. H. Chou. B#: a battery emulator and power-profiling instrument. *IEEE design & test of computers*, 22(2), 2005.

[21] B. Ransford, J. Sorber, and K. Fu. Mementos: System support for long-running computation on RFID-scale devices. In *Proceedings of the 16th Intl. Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2011.

[22] M. Salajegheh, Y. Wang, K. Fu, A. A. Jiang, and E. Learned-Miller. Exploiting half-wits: Smarter storage for low-power devices. In *Proc. 9th USENIX Conference on File and Storage Technologies (FAST)*, 2011.

[23] V. Shnayder, M. Hempstead, B.-R. Chen, and M. Welsh. PowerTOSSIM: Efficient power simulation for tinyos applications. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[24] J. Sorber, A. Kostadinov, M. Garber, M. Brennan, M. D. Corner, and E. D. Berger. Eon: A Language and Runtime System for Perpetual Systems. In *Proc. ACM SenSys*, 2007.

[25] T. Stathopoulos, D. Mclntire, and W. J. Kaiser. The energy endoscope: Real-time detailed energy accounting for wireless sensor nodes. In *Information Processing in Sensor Networks, 2008. IPSN'08. International Conference on*, pages 383–394. IEEE, 2008.

[26] H. Zhang, J. Gummeson, B. Ransford, and K. Fu. Moo: A batteryless computational RFID and sensing platform. Technical Report UM-CS-2011-020, UMass Amherst Department of Computer Science, June 2011.

[27] H. Zhang, M. Salajegheh, K. Fu, and J. Sorber. 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, pages 9:1–9:5, New York, NY, USA, 2011. ACM.