Enabling Self-Powered Internet of Things with Integrated Cyber-Physical Modeling and Deployment Tools

A

Dissertation

Presented to
the faculty of the School of Engineering and Applied Science
University of Virginia

in partial fulfillment
of the requirements for the degree

Doctor of Philosophy

by

Victor Ariel Leal Sobral

August 2024

APPROVAL SHEET

This

Dissertation

is submitted in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

Author: Victor Ariel Leal Sobral [illegible]

This Dissertation has been read and approved by the examing committee:

Advisor:	Dr. Bradford Campbell
Advisor:	Dr. Jonathan Goodall
Committee Member:	Dr. Steven Bowers
Committee Member:	Dr. Benton Calhoun
Committee Member:	Dr. John Stankovic
Committee Member:	Dr. John Lach
Committee Member:	
Committee Member:	

Accepted for the School of Engineering and Applied Science:

[illegible]

Jennifer L. West, School of Engineering and Applied Science

August 2024

Abstract

As the internet of things (IoT) technology evolves and matures, billions of new devices are expected to be deployed and provide value for applications in many different areas such as building automation, healthcare, industry, city management and farming. While some IoT devices can rely on mains power, other devices might be deployed in locations without access to any cabled infrastructure and require either battery power or energy harvesting to operate. As the number of IoT devices increases or if the device's location is hard to reach, the burden of periodical battery replacement becomes important. Therefore, self-powered IoT (SPIoT) devices capable of generating the energy they need to operate is a promising solution to enable sustainable and scalable IoT infrastructure. However, enabling SPIoT applications is challenging due to the coupled nature between energy generation, device's hardware operation, and application requirements, and due to the technological complexity of integrating and deploying end-to-end IoT applications. For instance, if a SPIoT adopter wants to replace a battery-powered device for a self-powered one, estimating if this device would work as well as the old one depends on: (1) how much energy this device can capture in the space; (2) how often does the application requires the device to become active; (3) how efficiently does the device uses energy; (4) how much energy the device can store; and (5) how easy is to integrate the device to the old device's infrastructure. Answering these questions requires an integrated cyber physical perspective on SPIoT applications, combining models of environments, devices and application requirements to provide a framework that supports the design, evaluation, and deployment of SPIoT applications.

To address these challenges, we introduce an integrated cyber physical SPIoT modelling framework and design tools that: (1) Enables evaluation of SPIoT applications by modeling energy generation, storage, and consumption; (2) Informs energy harvesting data collection that supports SPIoT hardware designers; (3) Allows SPIoT designers to select harvesters and storage components that meet their application requirements; and (4) Support SPIoT deployments by introducing the energy harvesting score, a 100-0 value that reflects how much useful energy is available in an environment, and, if used as a rating, indicating minimal energy requirements to support SPIoT operation. We also introduce a cloud-based data storage and visualization tool, easing end-to-end IoT application development. We use two practical examples to motivate the design and evaluation of SPIoT, one being a water quality station powered by solar and water flow kinectic energy, and the other example

being a water leak sensor powered by a thermal energy source. With this modeling framework and deployment tools, we will enable ubiquitous end-to-end SPIoT applications by supporting the design and deployment of SPIoT devices comparable with battery-powered devices without battery's limited lifetime constraint.

Acknowledgments

I'm grateful for all the love and support I received from my family: to my parents Martha and Sobral, my brother Andre, my aunt Monica, and my uncle Christoph.

I could not have made it so far without all the guidance, patience, and support that I received from previous and current mentors that I had the privilege to meet along my academic journey: Robson, Ana Isabela, Luciana, Chantal, Pierre, Bruno, Valeria, John Lach, Jon Goodall, and Brad Campbell. I'm also grateful for the support, insight and discussion with my dissertation committee members professors Jack Stankovic, Ben Calhoun and Steve Bowers.

I would like to thank all amazing friends that I had the opportunity to meet during my years at Charlottesville for all the support and joy they bring to my life, the friends I have met as lab mates, within the Brazilian community, and through the University of Virginia activities. Finally, I would like to thank all my friends in Brazil and France that believed and supported me all along.

Contents

List of Tables

List of Figures

Chapter 1: Introduction

Motivated by a remarkable progress on information and communication technology (ICT), researchers envisioned the internet of things (IoT), the idea of connecting physical objects to the internet, allowing them to exchange information and to operate in coordination [83]. Since the IoT idea's conception, many applications have been demonstrated in different areas such as in agriculture [41], building automation [82], smart factories [139], and stormwater system management [24]. As a reference for the impact of IoT in modern society, analysts estimated that $1.6 trillion economic value was generated from IoT solutions in 2020 [37] and the number of device connections was expected to reach 14.7 billion by 2023 [35].

Although IoT end devices can be designed to use wired communication solutions, adopting wireless communications is desirable among many applications since it allows for better scalability by eliminating the cost and maintenance issues associated with cabled network infrastructure as well as allowing increased IoT device mobility. In a similar way, relying on cabled power infrastructure can either limit viable IoT deployment locations to nearby existing AC wiring and outlets or result in increased installation costs. If an application requires deploying IoT devices in locations free from any cabled infrastructure, the IoT device cannot use mains power and must instead rely on one of two main alternatives: either use battery power or generate itself the energy required to operate. As the number of IoT devices increases or if the device's location is hard to reach, the burden of periodical battery replacement becomes important. Therefore, self-powered IoT (SPIoT) devices capable of generating the energy they need to operate is a promising solution to enable sustainable and scalable IoT infrastructure.

A major challenge to design SPIoT is the uncertainty associated with energy generation, also known as energy harvesting [1]. Harvestable energy depends on the dynamic characteristic of the energy source, for example, a solar panel placed in a conference room with no windows will have its energy generation correlated with the room's lamp operation, which is in turn correlated with room occupancy, which can vary across the day. The same solar panel placed outdoors will have its energy generation correlated with sunlight radiation. While some energy sources are highly predictable as it is the case for outdoor solar, other sources can be challenging to predict, notably if they are associated with human behavior as it is the case for many indoor solar applications. Characterizing and modeling energy harvesting across the range of energy sources and deployment locations is an important step to understand the constraints that SPIoT devices need to satisfy.

Due to energy harvesting uncertainty, SPIoT designers face two main challenges: (1) to properly size energy harvesters and energy storage; and (2) to dynamically adapt the device's operation to available energy, also known as dynamic power management (DPM). While energy harvester size impacts the overall energy availability, DPM goal is to deliver optimal quality of service (QoS) with the available energy. For instance, a solar-powered camera used in a security application could increase its frame rate or image resolution at the cost of extra energy consumption to improve QoS on sunny days and do the opposite to save power and avoid power outage on cloudy days. However, if users are only interested in a fixed QoS (a fixed frame rate for the solar-powered camera example), the SPIoT will only deliver the intended QoS if the energy harvesting conditions meet the device's energy consumption requirements. If a SPIoT device meets the intended QoS during all deployment lifetime, it means that the SPIoT can replace a battery-powered IoT counterpart without any drawback.

However, deploying SPIoT devices that can replace battery-powered IoT is not trivial. From the SPIoT designer's perspective, they don't have enough information about energy harvesting characteristics on deployment locations. From the SPIoT adopter's perspective, they don't have enough information on the device operation and use of energy. This information mismatch often translates in subpar deployments, either resulting in oversized energy harvesters or unreliable and intermittent SPIoT operation.

While IoT device-level reliability is required to achieve successful deployments, data management is also critical to useful IoT applications. Data acquired by sensors are usually transmitted to gateways, and then to servers, that store, and provide access to the data. Deploying end-to-end IoT applications requires designers to navigate through a challenging and ever changing array of solutions, from wireless communication protocols to cloud services. Lowering the barrier-to-entry to deploy end-to-end IoT applications can foster the adoption of IoT solutions, including SPIoT.

In our work, we introduce an integrated cyber-physical modeling approach and deployment tools to enable self-powered internet of things. The proposed modeling approach leverages the coupled nature of energy harvesting and SPIoT device models to: (1) Evaluate and design end-to-end applications, estimating energy harvesting using harvester models and energy consumption requirements from application requirements and off-the-shelf IoT hardware components; (2) Collect and share energy harvesting datasets that supports SPIoT hardware designers; and (3) Support SPIoT deployments by introducing the energy harvesting score and rating metrics, a 100-0 value that reflects how much useful energy is available in an environment and express SPIoT minimal energy requirements to support operation. Finally we introduce a cyber infrastructure tool to support IoT deployments by offering cost-effective, flexible, and easy to maintain cloud infrastructure to store, and visualize sensor data.

We represent the scope of this thesis in Figure 1.1, with the deployment tools being the energy harvesting score and the cloud cyber infrastructure. Our cyber-physical framework for designing and deploying end-to-end SPIoT is represented in Figure 1.2, with all analysis, designs and deployment tools in this work being related to blocks in this flow chart.

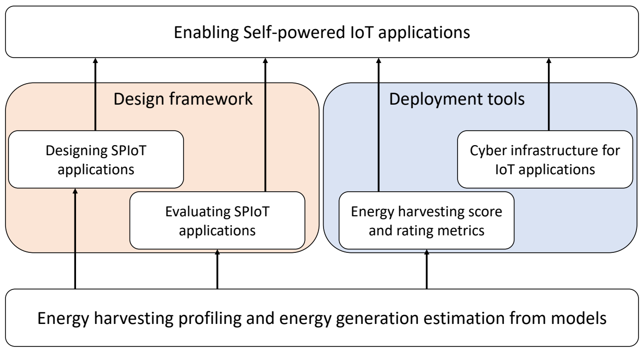
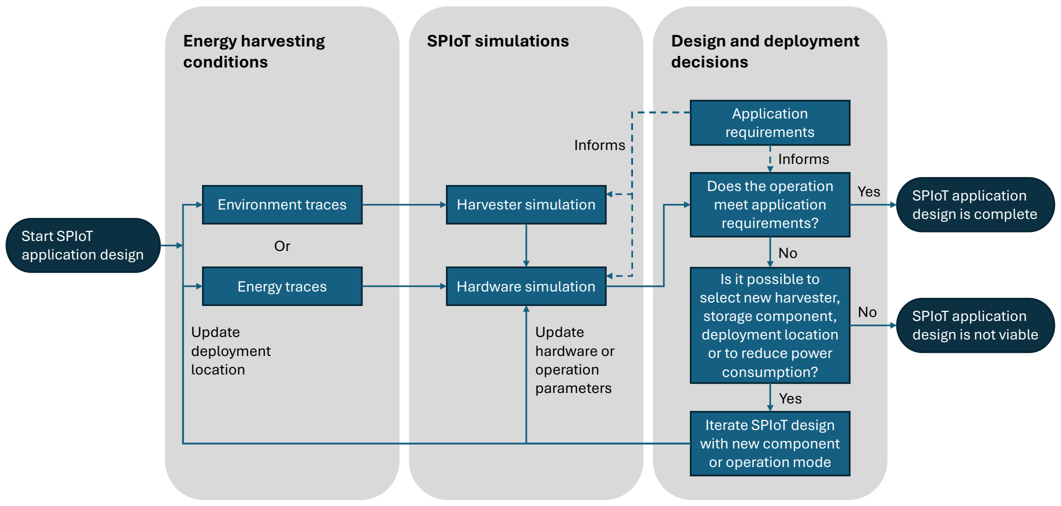

Enabling Self-powered IoT applications
Design framework
Deployment tools
Designing SPIoT applications
Evaluating SPIoT applications
Cyber infrastructure for IoT applications
Energy harvesting score and rating metrics
Energy harvesting profiling and energy generation estimation from models

Figure 1.1: **Enabling Self-Powered Internet of Things with Integrated Cyber-Physical Modeling and Deployment Tools.** Energy harvesting profiling, and models for energy generation are integrated with SPIoT models to provide a framework to design and evaluate SPIoT applications. To support deployments, energy harvesting information is translated to a score metric that can be used as a reference to characterize spaces, and define ratings representing SPIoT minimum energy requirements.

1.1 Thesis Statement

Designing and deploying end-to-end self-powered Internet of Things (SPIoT) is challenging due to the intersection of challenges between unpredictable energy generation, increased hardware complexity, application requirements, and data management. A cyber-physical modeling framework that integrates energy harvesting estimations and device energy-level simulations, along with a generic cloud-based data management system, can allow SPIoT designers to select a harvester, deployment location, device operation rate, and data backend that enable 100% availability at assumed harvesting conditions and a SPIoT design within size and total cost constraints. This modeling framework and deployment tools will enable ubiquitous end-to-end SPIoT applications by supporting the design and deployment of SPIoT



Energy harvesting conditions
SPIoT simulations
Design and deployment decisions
Start SPIoT application design
Environment traces
Or
Energy traces
Update deployment location
Harvester simulation
Hardware simulation
Informs
Update hardware or operation parameters
Application requirements
Informs
Does the operation meet application requirements?
Yes
No
SPIoT application design is complete
Is it possible to select new harvester, storage component, deployment location or to reduce power consumption?
No
SPIoT application design is not viable
Yes
Iterate SPIoT design with new component or operation mode

Figure 1.2: **Flow chart representing the cyber-physical framework for designing and deploying end-to-end self-powered IoT.** We demonstrate in each chapter the use of different sets of components in this chart to support or perform a SPIoT application design.

devices comparable with battery-powered devices without battery's limited lifetime constraint.

1.2 Contributions

We summarize our main contributions as follows.

1. Our data collection study resulted in the first publicly available multiple-week dataset on thermal energy harvesting and also the first available dataset of its length using I-V curve profiling technique, complementing existing datasets in the literature for solar [115, 54] and kinetic [53] energy harvesting. We found that we can characterize thermal energy harvesting output from a commercially available thermoelectric generator, using a custom interface printed circuit board and off-the-shelf components such as a raspberry pi and sensor breakout boards;

2. We introduce a new energy harvesting score metric as a tool to support SPIoT deployment by realistically representing usable harvested energy by a SPIoT

device in a single number 0-100, and using a SPIoT model that expands on available models [70, 31] by considering three energy conversion paths with respective efficiencies. We found that we can use the simulated availability of a set of reference SPIoT devices to score energy harvesting conditions;

3. We introduced and performed a cost analysis for an open source cloud-based IoT backend tool that uses flexible modern cloud services such as serverless functions and API gateway to enable end-to-end IoT applications, as opposed to more common end-to-end IoT applications using dedicated application servers [92, 24]. We found that cloud services initially developed for web applications can be used together with The Things Network as building blocks of IoT end-to-end applications at low design and maintenance effort;

4. We are the first work to use a large streamflow velocity dataset to analyze the viability of using mini hydro turbines to power a water quality station IoT device, complementing other works using estimated or measured solar energy [70, 31]. We found that by using large available datasets with the selected mini turbine and solar panel harvester models, we could estimate harvesting conditions without the effort of doing field collection on all theses sites. We combined the harvesting conditions with SPIoT device models to simulate how a SPIoT application would perform in a wide range of deployment locations and we evaluated the feasibility of a water quality application with different harvesting modalities; and

5. The water leakage sensor design is one of the first works demonstrating how to use a thermal energy harvesting dataset to inform SPIoT design, while other state-of-the-art works adopt solar power [70, 31] and use datasets to calibrate and validate their models[31]. We found that with our collected thermal dataset, an energy harvesting front-end model and energy consumption data from a commercially available water leak sensor, we are able to simulate different SPIoT design options and choose harvesters and capacitor components as to reach 100% availability under recorded energy harvesting conditions.

Chapter 2: Related work

A considerable amount of research on ICT and IoT applications has been published in recent years[4, 67, 86, 94, 118], contributing to the maturity of the technology by identifying and proposing solutions to numerous challenges [32, 112, 123]. To narrow down relevant literature for this work, we select three main areas to explore: (1) Self-powered IoT models, consisting of works that introduced models for IoT devices, applications and energy generation; (2) Energy harvesting profiling, consisting of works that proposed techniques and conducted energy harvesting profiling experiments; and (3) End-to-end IoT applications, consisting of practical IoT system designs from sensor to gateways, cloud infrastructure, and user interface tools.

2.1 Self-Powered IoT Models

An example of a typical SPIoT application is depicted in Figure 2.1, capturing the energy flow from the solar radiation source to solar panel harvester, and to the power conditioning circuitry of the IoT device where it will be stored and consumed by controller, communication and sensing tasks to deliver a certain service to the user through a network infrastructure. This system can be abstracted as a set of models as depicted in Figure 2.2.

In their work [70], Kansal et al. introduced many model components of SPIoT applications and proposed a dynamic power management (DPM) algorithm to predict energy harvesting and adapt the IoT device operation based on energy availability and application utility. Kansal et al. also introduced the concept of energy neutral operation (ENO), an SPIoT operation mode where energy consumption matches energy generation at the end of a fixed time window. For the case study in [70], a sensor operating in an outdoor environment harvests energy from solar radiation and adapts its duty cycle based on actual and predicted energy generation in a way that application utility is maximized and the energy stored in the beginning of a day is approximately the same as in the end of that day.

For the power conversion model, Kansal et al. [70] assumes a constant average efficiency term that represents the energy losses to charge and discharge a battery

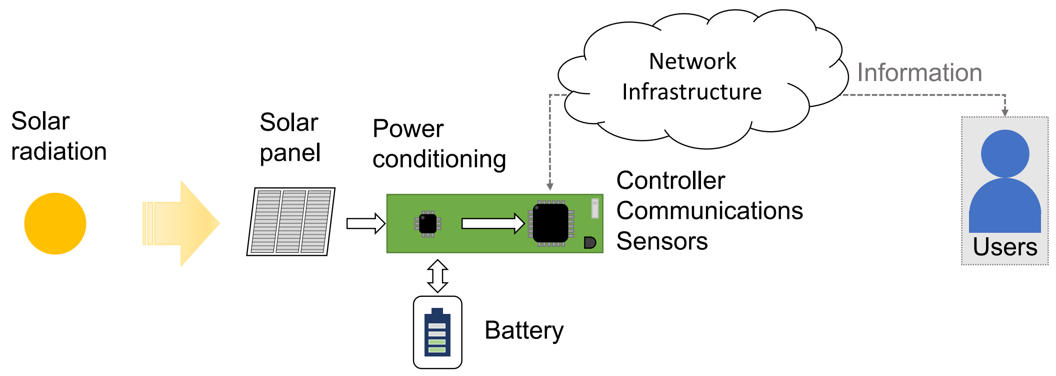


Figure 2.1: **Example of SPIoT application domains from energy generation to device operation to end user.** Solar energy is captured by solar panel and then stored in a battery or consumed by the SPIoT device. IoT tasks require energy consumption by controller, sensors or communication to deliver a service to the application's users through a network infrastructure.

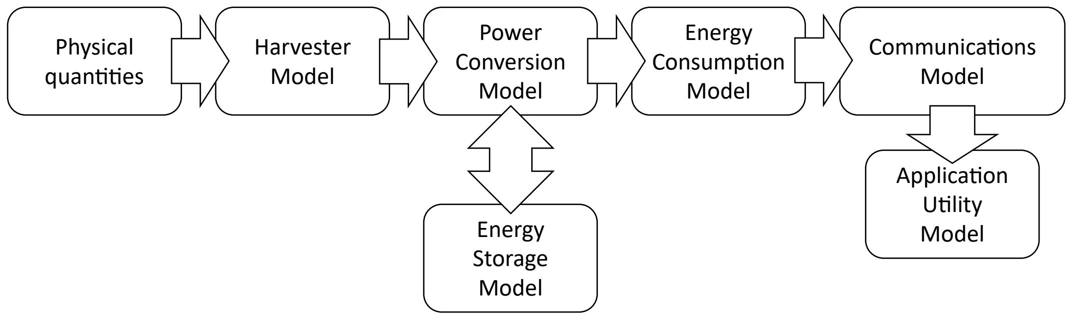


Figure 2.2: **Block diagram of a model for a SPIoT application.** Physical quantities are models related with the energy source, as for instance the solar radiation. Harvester, power conversion, energy storage and energy consumption models represent the power flow in an SPIoT device hardware. Communication model represents non-idealities on the information path from device to application processing. Application utility model represents how much weight a service tasks carries to the end goal of the application.

component. As for the energy storage model, only a constant energy leakage rate term is considered.

To represent uncertainty on energy generation and consumption, Kansal et al. [70] assumes a known, or measured profile of harvested and consumed energy. This model represents the total energy time series inside a time interval by three parameters as depicted in Figure 2.3, where ρ is the average power, σ_1 captures the maximum energy burst and σ_2 captures the maximum energy drought.

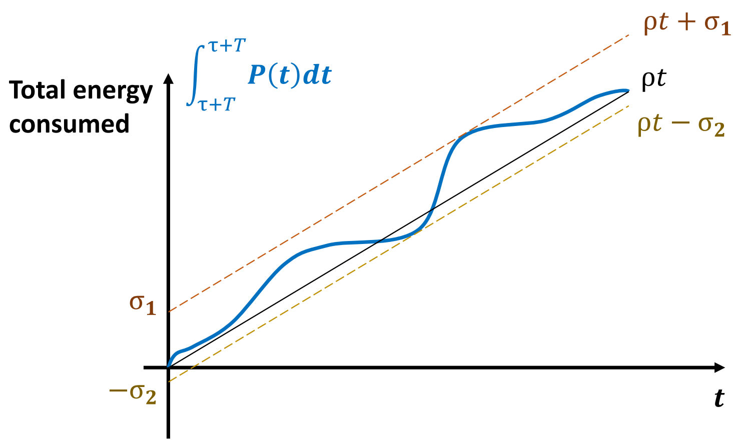


Figure 2.3: **Energy generation or consumption uncertainty 3-parameter model.** Energy bursts are represented by the two σ parameters and average power is captured by the parameter ρ. If for instance the blue line represents generated energy and the black line represents energy consumption, uninterrupted operation of the SPIoT device requires σ_2 as the initially stored energy in the battery and $\sigma_1 + \sigma_2$ as the maximum battery capacity.

The model introduced by Kansal et al. takes into consideration a communication model for the case of a network of sensors, representing the relationship between an SPIoT's receiver duty-cycle and the reception of a wake-up signal.

The power management algorithm introduced by Kansal uses a simplified utility model consisting of 3 regions, the first for which there is no value in duty cycle operations bellow certain frequency, the second with linear returns for increased duty-cycle frequencies and a third region for which the utility is constant for duty cycle frequencies above certain threshold. This utility function and the simple efficiency model for energy storage results in a power management algorithm that prioritizes the use of energy while energy generation is above the expected average and keeps energy consumption to the minimum value for energy generation bellow the expected average, then the algorithm spreads exceeding energy evenly until the maximum utility return is achieved.

Following the introduction of the general model by Kansal et al., many works focused on improved dynamic power management (DPM) algorithms with more complex prediction models so devices can better adapt their operation based on energy availability. Spies et al. [122] provide a summary of the SPIoT models available for designers, as well as energy harvesting considerations, including an estimate range and order of magnitude of energy harvesting for different energy sources. The authors of [122] also make a compilation and expand on the models introduced by [70].Finally, the authors of [122] mention design considerations about power management unit (PMU) as minimum voltage required to operate, cold start and capacitors as intermediate storage elements but they don't further explore these considerations in the modeling effort.

The work by Buchli et al. [31] uses similar models to [70], with the goal of designing a SPIoT device that also harvests energy from outdoor solar source. Instead of using daily energy generation measurements to predict future energy generation as in [70], Buchli et al. use a physics-based solar radiation model to derive the output power of a solar panel for multiple years of operation. Buchli et al. use this multiple-year model to extend the time window considered for the ENO from a scale of days in the case presented by [70] to a scale of years, allowing for instance the system to store exceeding energy during the summer months, and then use it during the winter months. Authors of [31] also consider an empirical attenuation factor to account for average cloud coverage in their solar radiation based model, so they can set SPIoT operation accordingly. Their coupled SPIoT and energy generation models allows them to make design time decisions about energy storage capacity, as to account to the expected needed capacity for the long term ENO. In a following work, Buchli et al. [30] employ a more realistic utility function for the dynamic power management operation, considering the error in the SPIoT application outcome when the measurement frequency varies. Instead of the three region utility function used in [70], their utility function has two regions, one where utility is zero and other where utility assumes an exponential shape with diminishing returns as the SPIoT duty cycle increases.

To achieve a more realistic SPIoT device model, Masoudinejad et al.[89] combined subsystem models obtained from measured data over different domains, integrating solar harvester models [88], power management unit model [91] and battery model [90]. As their harvester model, Masoudinejad et al. characterized solar panels in [88], with a benchtop equipment setup capable of reproducing a range of light radiation intensities, temperatures, and light source types. For the power management unit model, Masoudinejad et al. measured the average efficiency of power management circuits in their work [91], varying input voltages and currents. For their energy storage model, Masoudinejad et al. derived a measurement-based open-loop battery model in [90]. The final integrated model was demonstrated in [89] as an useful tool for identifying bottlenecks in system design, but without including energy harvesting dynamics models, neither application utility models.

More recently, Brunner et al. introduced an integrated simulation environment for SPIoT devices [29] that allows designers to explore the complex design space of SPIoT devices and applications. In their simulator, users can perform experiments combining multiple SPIoT hardware components and energy harvesting models. In their work [29], Brunner et al. also demonstrated how their simulator can be used to optimize SPIoT component and operation parameter selections to achieve higher throughput in intermittent operation applications. While Masoudinejad et al. introduced the integration of SPIoT subcomponents to characterize their own platform [89], the simulator introduced by Brunner et al. [29] provides a generic framework to describe and simulate integrated SPIoT models that are able to support SPIoT application analysis, and design time decisions.

An important challenge for different applications is how to define the value of the acquired sensor data. In [70], this value is defined as the utility function, a measure of how much better an application performs with increased duty-cycle frequency. Although the utility function in [70] was only superficially motivated, other works have demonstrated how to use classification metrics [50] and application metric error as utility functions [30]. Other aspects of information quality or utility were discussed in [21] and concepts can be translated for each application. In [51], the authors consider time varying utility, acknowledging that for some applications, data value depends on when it was acquired.

Insights and takeaways on SPIoT modeling related works

While Kansal et al. [70] introduced some of the fundamental ideas in SPIoT modeling, other authors contributed extensively with practical SPIoT design and analysis considerations, including integration with energy generation models, SPIoT component selection, and determining utility functions. We summarize the main SPIoT modeling approaches discussed here in Table 2.1.

Our takeaway from this SPIoT modeling literature review is that estimating real-world SPIoT operation behavior requires modeling of interdependent characteristics of energy generation, conversion, storage, consumption, and application utility. We also realize that SPIoT modeling is often used to support the design of SPIoT solutions with the most efficient use of energy, as for instance informing dynamic power management (DPM) strategies [75, 70, 30, 2, 50]. However, not all applications can benefit from DPM, since this approach assumes that generated energy surplus can always be used to increase the utility of applications, for instance increasing the sampling rate of SPIoT devices. For applications where there is little to no benefit of using DPM, the challenge is estimating if energy harvesting conditions can support an required energy consumption level for most or all of the deployment time. From this observation, we identify the following opportunities to support SPIoT design and

Table 2.1: SPIoT modeling prior work

Related work	**Energy generation considerations**	**SPIoT device considerations**	**Utility model**
Kansal et al. [70]	average energy, bursts and droughts	efficiencies, max storage capacity and leakage	arbitrary, linear
Buchli et al. [31]	physics based astronomical model	efficiencies, max storage capacity and leakage	application based, exponential
Spies et al. [122]	average energy, bursts and droughts	efficiencies, max storage capacity and leakage	Not considered
Masoudinejad et al. [88]	Not considered	empirical models for harvester, energy conversion and battery	Not considered
Brunner et al. [29]	Energy harvesting traces supported	empirical models supported	Not considered

increase its adoption by applying integrated SPIoT models to perform: (1) Energy harvesting data acquisition that is useful for SPIoT hardware designers, as we show in Chapter 4; (2) Analysis of potential SPIoT applications with different harvesting modalities, as we show in Chapter 7; and (3) Informed SPIoT deployments, as we show in Chapter 5 and Chapter 7.

2.2 Energy Harvesting Profiling

Although harvesters have been thoroughly studied and modeled in laboratory bench top setups [88, 56, 42, 40, 78, 101], estimating how much energy can be harvested in real deployment conditions is challenging since the harvesters can be exposed to a wide range of changing and hard-to-predict environments, depending on the energy source of choice. One common approach to characterize energy harvesting is collecting profiles in relevant environments and using them as a proxy to the expected energy generation dynamics for a given SPIoT application.

As an example of early energy harvesting profiling work, Gorlatova et al. collected energy irradiance data from indoor and outdoor light sources [54, 55], and kinetic energy sources [53, 38]. To estimate energy available to an EH IoT device using these datasets, we need to model the harvester's transfer function from raw energy measurements to estimated converted electric power. Other examples of datasets that can be used to derive raw available energy are The National Solar Radiation Data Base (NSRDB) [111] and SolarAnywhere [36] for outdoor solar energy harvesting, and National Water Information System (NWIS) [124] for hydro power. We demonstrate the use of solar and hydro kinetic datasets to estimate harvested energy in Chapter 7, and in its associated published work[85].

In a following energy harvesting profiling effort, Sigrist et al. [115, 116] collected energy harvesting data from indoor light in office spaces. They measured environment parameters, as well as voltage and current traces of their energy harvesting front end, containing an indoor solar panel connected to a power management circuit that supplies a virtual battery. Their dataset provides a valuable insight on realistic energy conversion efficiencies from their used solar panel coupled with a maximum power point tracking power management circuit. Their dataset can be used to estimate how much power is available to an EH IoT device, assuming the same energy harvesting front-end of their data acquisition platform. We use their dataset as a reference for our work in Chapter 5 to estimate typical harvesting conditions and support our analysis.

Fan et al. demonstrated a recording platform in their work [49] with similar capabilities to [115] while also being able to simulate super capacitor storage components and a controlled load. Fan et al. only recorded short energy harvesting traces in the scale of hours, while Sigrist et al. made available multiple years of profiled energy harvesting data.

Another approach to collect energy harvesting traces was proposed by [58], where the I-V curve of harvesters are measured in real deployment conditions. I-V curves of harvesters are achieved by sweeping the impedance attached to the harvester output and measuring the loaded harvester voltages and currents. The I-V curve information can be useful to circuit designers to simulate power management solutions as these curves are sufficient to emulate harvesters in a range of loading scenarios. For instance, we show in Chapter 4, and in our published work [121, 76], how we collected a dataset of thermal energy harvesting conditions from residential settings by estimating the current-voltage curves of a thermoelectric generator (TEG). This information can be used to simulate the TEG model adopted in our data acquisition system, allowing designers to simulate power management circuits of EH IoT devices. A disadvantage of the I-V energy harvesting profiling approach is the large amount of data that needs to be collected to reproduce the I-V curve in fast changing harvesting conditions.

Insights and takeaways on energy harvesting profiling related works

While all previously discussed works provide valuable information for designers, there is a trade-off between generality and accuracy when using their collected data to estimate the actual harvested energy by a SPIoT device. For instance, measuring raw available energy is an upper bound on the available energy to be harvested, but it requires a model for the harvester efficiency. If more information is collected about the conditions, or environment parameters where the harvester is deployed (temperature, orientation, etc.), it is possible to adopt a more precise harvester model. Both these approaches are general, in the sense that they are not biased to any particular harvester component. On the other hand, measuring parameters from a specific harvester, power management circuit, or complete SPIoT design is a better representation of the operation of these measured devices, but it becomes more challenging to translate the collected data from the hardware components used to collect data to other alternative parts and circuits. We summarize the main energy harvesting profiling approaches discussed here in Table 2.2.

Table 2.2: Energy harvesting profiling prior work

Measurement approach	**Mathematical representation**	**Usefulness**
Raw available energy [54]	$\widehat{P_{harv}} = \eta P_{raw}$, e.g. irradiance ($W/m^2$)	Upper bond on P_{harv}
Environment parameters [53]	$\widehat{P_{harv}} = f(p_1, p_2, \ldots)$, e.g. acceleration	Harvester design
Harvester output [58]	$(V_{harv}, I_{harv}) = f(load)$, e.g. $\widehat{P_{harv}}@R_{load}$	Hardware design
PMU output[116]	$\widehat{P_{harv}} = P_{PMU}$	More realistic P_{harv}
Stored energy [49]	$E_{str} = E_{str(last)} + E_{PMU} - E_{cons}$, e.g. V_{cap}	Operation evaluation

Our takeaway from this energy harvesting profiling literature review is that all profiling approaches are valuable to different stages of the SPIoT design, and hardware-biased approaches are more suited to be integrated with SPIoT application models. We select the harvester output profiling approach used by Hester et al.[58] to profile thermal energy harvesting in Chapter 4, demonstrating a practical way of capturing realistic energy generation data with an off-the-shelf thermoelectric generator component. We later use the profiled data to inform the design of the energy harvesting front end of a water leakage sensing application in Chapter 7. We select the environment parameters approach to estimate the energy output of harvesters where harvester output measurements are not available. We estimate power output of a solar panel from illuminance measurements in Chapter 5, to support the characterization of harvesting conditions in indoor office spaces. We also estimate energy generation of a miniature hydro kinetic turbine and a solar panel in Chapter 7, to evaluate if the generated energy would meet the consumption requirements of a water quality sensor station.

2.3 End-to-End IoT Applications

To understand the real-world constraints and usefulness of IoT applications, we investigate the general architecture of IoT systems, from sensor to gateway, and application server. While in the previous sections we discussed device-level operation and energy harvesting conditions, in this section we will focus on practical deployment and application server design challenges to build useful and cost effective IoT solutions. As an example end-to-end IoT application, we design and deploy a smart city flood warning tool consisting of commercial IoT sensors, gateways and cloud based tools for data storage and visualization. The developed IoT cyberinfrastructure is discussed in Chapter 6 and in our published work [77].

As information and communication technologies (ICT) diversifies and becomes more complex, selecting, developing and integrating hardware and software solutions for IoT smart city and building automation projects ([67], [126], [92]), requires a broad range of technical knowledge, frequently resulting in an important adoption barrier to many communities. To implement a complete IoT solution, the designer must develop or select components as sensor hardware and respective communication technology, gateway infrastructure, and application software, which nowadays often includes cloud computing resources. To address this challenge and to make IoT solutions more accessible, The Things Industry (TTI) [66] created and maintained The Things Network (TTN) [98], a set of open-source tools and APIs to provide the basic software infrastructure to deploy IoT sensors based on LoRaWAN [93, 114], a low-power and wide-area network (LPWAN) communication protocol. This open-source project enables contributors around the globe to publicly share TTN compatible gateways that can connect LoRaWAN sensors to a cloud backend service maintained by TTI. Using TTN for smart city projects has been successfully demonstrated in the literature for different applications (e.g. [24, 43]) while also benefiting communities by creating an open LoRaWAN communication infrastructure that can be leveraged by other projects.

To illustrate some of the possible IoT architecture solutions to smart city projects, we selected two works, the first one targeting Radon gas concentration monitoring [92] and the second one a smart stormwater system using LoRaWAN and TTN [24]. The Radon gas concentration monitoring work was selected to represent a typical IoT project, where the study made use of available components and tools to build their own remote sensing solution. The smart stormwater system work was selected as an example of a similar application goal using LoRaWAN and TTN, but adopting alternative design components to our system.

Radon Gas Monitoring Application.

To monitor concentrations of Radon gas at indoor locations, the authors of [92] presented an IoT system that collects and transmits data to a remote server where values are stored. We summarize the IoT system architecture used for the Radon gas application in Figure 2.4.

As sensing devices, the authors of [92] adopted a commercially available Radon Scout gas sensor connected to a Raspberry Pi 3 device used as a controller and connected to the internet. The Raspberry Pi was programmed to read and transmit sensor data to their remote server through a Message Queuing Telemetry Transport Protocol (MQTT) [95] communication interface. The server receives sensor data through a MQTT broker that publishes received messages to a subscribed MQTT client managed by a Node-RED [102] application responsible for parsing and storing the data in a MySQL database [103]. Finally, a web server interface was created to read the database and display a table of stored sensor readings to users.

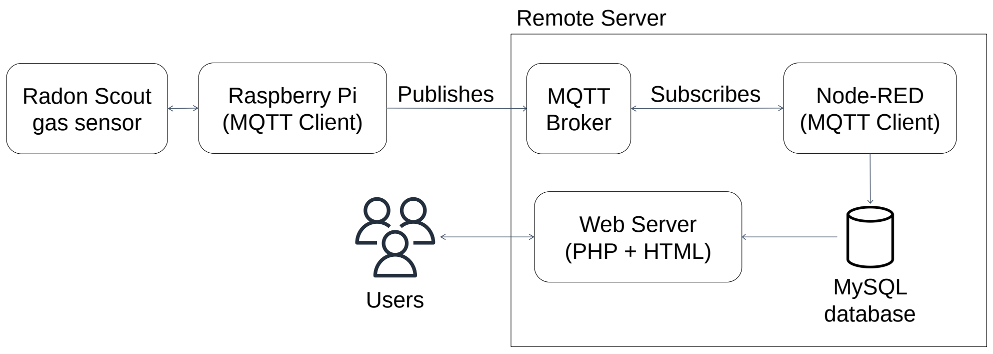


Figure 2.4: **IoT system architecture diagram for the Radon gas monitoring application.**

For our flood warning use case, we adopted commercially available LoRaWAN gateway and sensors. While our LoRaWAN gateway requires internet connectivity similarly to the Raspberry Pi controller used in this related work [92], the LoRaWAN sensors can be deployed hundreds of meters away from the gateway, which allowed us to reach our desired deployment locations. We used TTN as our network server to register and manage devices, reducing development time and enabling better scalability as new sensors only need to be registered to our TTN application. For this related work [92], authors needed to individually configure the MQTT clients in each one of their Raspberry Pi devices to publish sensor measurements to their server's MQTT broker, as well as individually manage any security key. Instead of using Node-RED to parse and ingest data as adopted in [92], we used a python script to manage the data ingestion and parsing system that periodically receives data from our TTN application through a MQTT client. As our data storage solution, we also adopted a MySQL database, similarly as presented in [92], but we also decided to create a dedicated long-term cloud-based storage solution using AWS S3 as a backup to the MySQL database. For this long-term data storage backup, we used AWS Lambda service to create a serverless and independent data ingestion solution to periodically request data from TTN storage integration and store it in AWS S3. Instead of displaying sensor data through a website server, we created a dashboard on a Grafana application [74] to plot relevant sensor information such as measurements and battery voltage level. Although the authors of [92] were targeting an indoor Radon gas monitoring application, some of their system components could be adopted by other IoT applications such as collecting and displaying data from LoRaWAN sensors connected to TTN. For instance, TTN offers a MQTT Broker service that can publish received LoRaWAN messages to subscribed clients, making it possible to re-use the server infrastructure described in [92] by updating the broker address, client credentials, parsing function, database configuration and sensor measurement variables.

Smart Stormwater System Application

For the stormwater monitoring system introduced in [24], the authors deployed a set of sensors around the Illawarra-Shoalhaven region in Australia. Their sensors relied on either LoRaWAN or 4G cellular network to communicate, depending on each sensor's required data rate. Sensing devices included water-level sensors, tipping bucket rain gauges sensors, pressure and humidity sensors, lagoon monitoring devices, and culvert blockage monitoring system. To receive data collected by the LoRaWAN based sensors, the authors deployed a network of TTN gateways in the study region. This gateway infrastructure was also seen by the authors as an investment to support other future applications including education related projects. We summarize the IoT system architecture used for the smart stormwater system application in Figure 2.5.

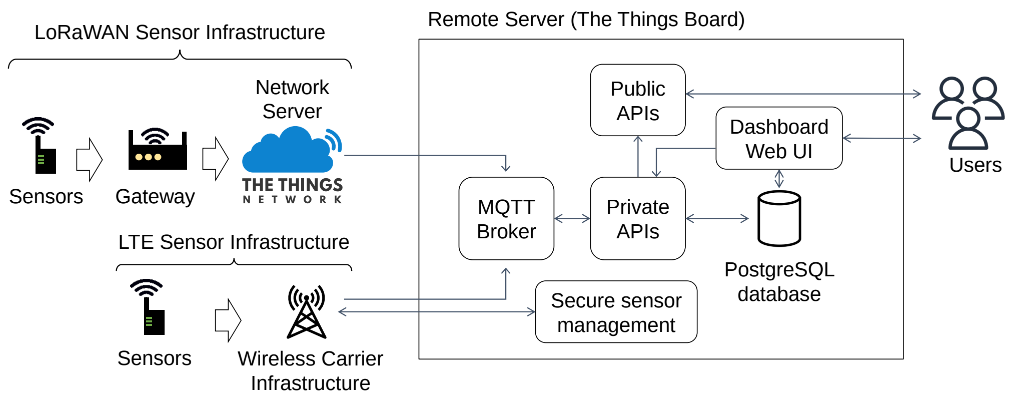


Figure 2.5: **IoT system architecture diagram for the related work storm water monitoring application.**

To store and display the collected data, authors of [24] adopted the open-source solution provided by the ThingsBoard [22], using MQTT protocol to receive LoRaWAN sensor data from TTN and store it in a PostgreSQL database. ThingsBoard also provides alerting and graphical interface tools to generate custom dashboards to display sensor data in real time and send automated alert messages. Authors have not specified if the server solution was hosted in a computer owned by them or a cloud solution, however ThingsBoard offers a platform as a service solution where they host their system in the cloud with pricing currently ranging from \$10/month to \$749/month.

Despite offering data storage, API access and visualization tools, Thingsboard is a turnkey software solution that requires users to have an always-running server to ingest, store and visualize data. On the other hand, our solution leverages cloud services to break down data ingestion from other on-demand uses, namely: (1) a serverless data ingestion and storage cloud application using AWS Lambda [17] and AWS S3 [9]; (2) a virtual machine instance with a MySQL database to provide responsive data access; and (3) a second virtual machine hosting a Grafana server [74] to provide data visualization. Our serverless data ingestion solution requires only a few lines of code to query sensor data from TTN, parse, and store data as a csv file in AWS S3, thus reducing complexity to manage bugs and update the system when compared to full servers such as ThingsBoard[22]. Using dedicated virtual machines for a database and visualization allows for tailored resource allocation based on the application needs, the flexibility to provide only the needed service, and code isolation to facilitate upgrading, adding, or switching services (e.g. replacing MySQL with PostgreSQL).

Insights and takeaways on end-to-end IoT application related works

We realized that end-to-end SPIoT applications can be penalized by over complex, costly and hard to manage cyberinfrastructure, hindering IoT adoption, and lowering IoT application cost-benefit. We also identify data storage and visualization as fundamental services required for most IoT applications. To provide these services, developers usually deploy their own servers at high cost, both in terms of money and time. This task is specially demanding when developers have backgrounds other than computer science, such as in embedded system design or instrumentation for example. On the other hand, more cloud services are recently becoming available that can reduce the burden on application server development, system maintenance and cost, such as serverless functions, and low-cost data storage. Another advantage of cloud services is the flexibility to tailor the system to the IoT application needs, allocating computational resources and storage when needed.

Our takeaway from this end-to-end IoT applications literature review is that navigating through IoT application technology can be challenging to developers without background on cloud services, and sharing experiences of application servers using cloud based solutions can be very valuable to them. We discuss the development of a cloud based data storage and visualization tool in Chapter 6, with serverless data ingestion capabilities, and on-demand resource allocation for data visualization.

Chapter 3: Motivating Applications of IoT Deployment Tools and SPIoT Modeling

To motivate our integrated self-powered IoT analysis framework, we selected two applications that illustrate practical challenges to design SPIoT devices: (1) water leakage detection; and (2) water quality monitoring. As to motivate our IoT deployment frameworks, we selected two example use cases to highlight our approaches: (1) Informing deployment of indoor solar SPIoT for smart buildings; and (2) Deploying IoT cyber infrastructure for smart city stormwater applications. While the context is based on these example use cases and applications, the fundamental concepts we discuss for each of them are translatable to many other smart city and building infrastructure use cases and applications. In this chapter we will discuss the highlighted flow chart elements depicted in Figure 3.1.

3.1 Water Leakage Detection

In an indoor air quality survey performed in 100 randomly selected buildings, the Environment Protection Agency (EPA) listed water damage as one of the main sources of pollution [131] as it can lead to mold growth, which in turn can cause serious health problems to people [20]. In this same study, the EPA found that 85% of the surveyed buildings experienced water damaged in the past and 45% were experiencing ongoing problems with leaks causing water damage, with some of the most common water leakage locations being occupied spaces, basement, roof and mechanical rooms. To prevent increased property damage, mold growth and associated health risks to building occupants, IoT sensors can perform early detection of leakages by being placed close to locations with high risk of experiencing water damage.

A requirement for these leakage detection devices is to be able to timely communicate to some alarm system when a leakage event is detected. An IoT designer can also set the device to periodically send a simple communication event known as heartbeat to inform the application that the sensor is still operational and build up confidence that the leakage detection system is working. To make efficient use of energy and increase the leakage detector sensor availability, designers need to choose a low power

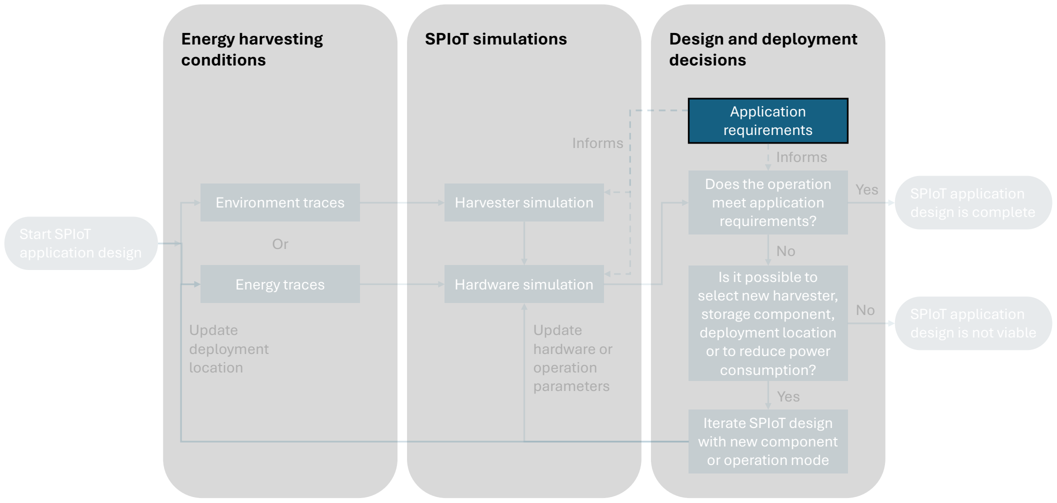

Energy harvesting conditions
SPIoT simulations
Design and deployment decisions
Application requirements
Informs
Informs
Start SPIoT application design
Environment traces
Or
Energy traces
Harvester simulation
Hardware simulation
Does the operation meet application requirements?
Yes
SPIoT application design is complete
No
Is it possible to select new harvester, storage component, deployment location or to reduce power consumption?
No
SPIoT application design is not viable
Yes
Iterate SPIoT design with new component or operation mode
Update deployment location
Update hardware or operation parameters

Figure 3.1: **Flow chart components related with application requirements.** Application requirements informs possible harvesting modalities, SPIoT hardware capabilities, and operation modes.

sensor and wireless communication strategy, as well as an adequate heartbeat reporting frequency.

A current challenge with such leakage detection applications is that plumbing infrastructure is usually located in hard-to-reach environments where might not be any AC wiring and outlets available and the cost for periodical IoT battery replacement can be significant. Furthermore, these are usually low-light locations, as inside wall gaps or in machinery rooms and therefore using solar powered sensors might not be practical. An alternative solution to power these water leakage sensors is using the thermal energy from pipes carrying warm water. The temperature difference from the pipe surface and the surrounding air can be used by a thermoelectric generator to power a water leakage sensor. Since the pipe temperature depends on water usage behavior, energy harvesting will also be correlated with warm water usage and therefore complex to model.

3.2 Water Quality Monitoring

Monitoring water resources with in-situ sensing systems is an important tool to address many hydrological and environmental problems such as flooding, runoff pollution, and aquatic ecosystem degradation [105, 25, 106]. For instance, government regulators or citizen scientists can monitor a stream's water quality by sampling parameters such as water's PH, conductivity, temperature, dissolved oxygen and oxidation reduction potential to identify runoff pollution and assess the potential risk to both the aquatic life and people making recreational use of the stream. In typical water quality monitoring applications, designers select one or a set of the previously mentioned sensors and set up a monitoring station to periodically acquire and report sampled measurements. Designers can set the sampling rate to be dynamically adjusted to capture more data during particular events like rainfalls or to provide better time domain resolution when power is abundant. Applications can require the data to be locally stored or transmitted through available wireless communication infrastructure like cellphone network.

While some water monitoring stations might be deployed in locations with access to mains power, many locations of interest do not have access to any cabled infrastructure and therefore they must rely on battery power or energy harvesting. Although researchers have proposed different energy harvesting sources to power water monitoring stations[69, 34], many research applications and commercial products adopt solar energy harvesting as it is a cost effective and reliable solution[99, 65, 84, 100, 5]. However, for some low-light locations as inside canyons or under dense canopy, relying on solar energy to power water monitoring stations might not be practical. An alternative source of energy that is currently under explored is the Kinetic power of flowing water. Extracting energy from flowing water in a small scale has been demonstrated by commercial mini turbines of up to 180mm of diameter and capable of delivering up to 15 Watt of power[64]. Self-powered water sensing stations powered by water flow's kinetic energy has the potential to increase the number of viable deployment locations, however also introducing energy generation uncertainty, since, for instance, water flow velocity can drastically change based on precipitation, which is usually more complex to model than solar radiation.

3.3 Informing deployment of indoor solar SPIoT for smart buildings

About of 40% of the energy use in the United States and the European Union is consumed in homes and buildings, according to the United States Department of Energy [129] and the European Commission [47]. In the United States, this energy consumption costs over $400 billion each year, and 75% of the produced electricity [129]. To make buildings more efficient, save energy, and comply with energy use regulations [72, 135], researchers developed complex automation systems with IoT sensors, capturing detailed information about the building state and energy use [87]. In this context, energy harvesting is a promising solution to enable large-scale IoT systems to gather building data, since battery replacement and waste management becomes impractical as the number of IoT devices increase.

However, designing and deploying energy harvesting Internet of Things devices is challenging due to the spatial-temporal variability of harvestable energy. For instance, a device harvesting energy from indoor lights is dependent on how often its deployment space is illuminated. This device might operate as intended in a high-occupancy and frequently illuminated space, yet underperform in less used spaces and when occupancy rates drop. Contrast this with a battery-powered device, which will operate as intended regardless of human activity (until the battery runs out). This inherent uncertainty suppresses the adoption of energy harvesting devices.

We propose a framework to support informed energy harvesting IoT deployment based on a new construct called the energy harvesting score. This score is a number from 0-100 that characterizes the usable energy in an environment available to an energy harvesting device. This same score serves as a rating for devices, characterizing the energy requirements of a device. If the score is above the rating, that device will function as expected in the environment. This framework can be used to characterize energy generation at potential deployment locations and guide the installation of SPIoT devices in smart buildings. We discuss our energy harvesting score and rating deployment tool in Chapter 5.

3.4 Deploying IoT cyber infrastructure for smart stormwater applications

With aging infrastructure and climate change, existing stormwater infrastructure has been failing to respond to atypical precipitation events in many cities. To address this challenge and avoid flooding events, some cities have invested in expensive stormwater infrastructure expansion and complex control systems. However, a more cost effective solution was demonstrated by [25], where IoT sensors collect real-time data as precipitation and water level in different parts of the city's stormwater infrastructure, supporting management decisions and making optimal use of existing infrastructure. This approach also allows stormwater infrastructure managers to make informed decisions on infrastructure expansion investments to make their cities more resilient.

Building an useful smart stormwater IoT solution requires not only deploying the physical sensors and gateway devices, but also cyber infrastructure to store and give users access to data. It is important that the cyber infrastructure is flexible, cost effective, easy to maintain, and integrate with other solutions.

Chapter 4: Energy Harvesting Profiling

Energy harvesting characterization and modeling is an important step to address energy harvesting uncertainty and support the motivating SPIoT applications discussed in Chapter 3. As each deployment location and energy source have their own dynamic characteristics, data collection in relevant environments is a critical to obtain a realistic representation of SPIoT device's energy harvesting constraints. In this chapter we will discuss the highlighted flow chart elements depicted in Figure 4.1. To drive the discussion on energy harvesting profiling, we will use the block diagram on Figure 4.2 as a reference for a typical energy harvesting device. Each energy harvesting profiling approach consists in measuring parameters from the different stages of the energy conversion and consumption path.

4.1 Energy harvesting profiling approaches

The most general approach for energy harvesting profiling is measuring P_{raw} as the upper bound of the total available energy at the profiled deployment location. For instance, in the solar energy harvesting profiling dataset colected by [54], irradiance measurements in W/cm^2 can be readily translated to raw available energy P_{raw} by multiplying irradiance by the harvester area in cm^2. While this approach provides useful information to SPIoT designers, it is most valuable for energy harvester designers so they can optimize energy harvesting efficiency for the typical operating conditions. Another factor that can play a role in energy conversion efficiency are environment parameters such as temperature, humidity or air flow velocity. If circuit designers want to estimate the power output of an energy harvester, they need to create a model that converts P_{raw} and relevant environment conditions to harvested output power P_{harv}. This process can be complex depending on the energy harvesting modality and adopting simplified models might result in inaccurate harvester output power estimation. For example, in thermoelectric energy generator (TEG) applications, while temperature difference is a good indicator of output power, energy generation is driven by heat transfer and other more complex factors as airflow and heatsink design can have a significant impact on realistic power output estimation.

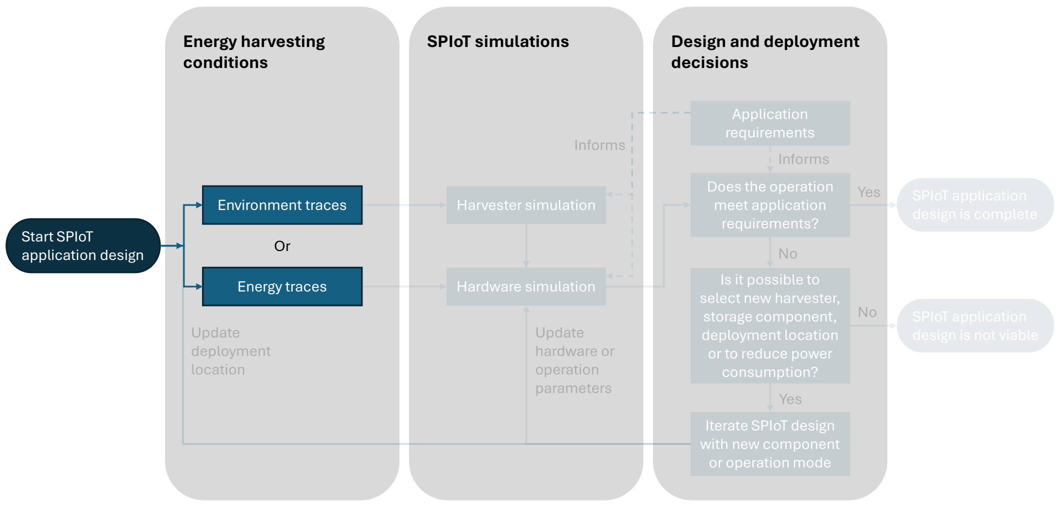

Energy harvesting conditions
SPIoT simulations
Design and deployment decisions
Start SPIoT application design
Environment traces
Or
Energy traces
Update deployment location
Harvester simulation
Hardware simulation
Informs
Update hardware or operation parameters
Application requirements
Informs
Does the operation meet application requirements?
Yes
SPIoT application design is complete
No
Is it possible to select new harvester, storage component, deployment location or to reduce power consumption?
No
SPIoT application design is not viable
Yes
Iterate SPIoT design with new component or operation mode

Figure 4.1: **Flow chart components related with energy harvesting characterization.** Energy harvesting characterization is required to understand the design and deployment constraints of a SPIoT application.

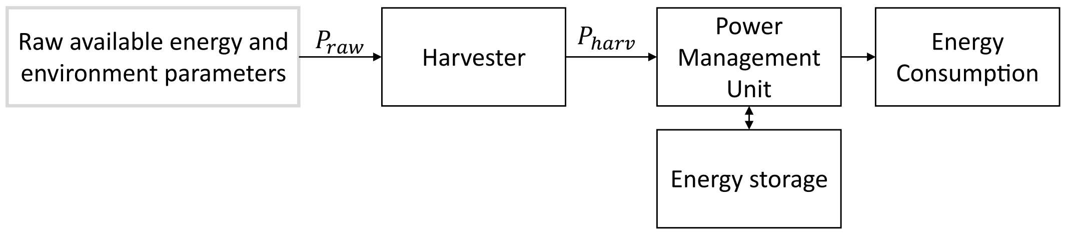


Figure 4.2: **Block diagram of the power system of a typical energy harvesting device.** P_{raw} is the maximum available power that an ideal harvester could convert to electricity. P_{harv} is the actual power output of a given harvester. The energy storage block represents capacitors or battery elements used to store energy. The energy consumption block represents all power consuming circuitry of the SPIoT device including, for instance, microcontroller, radio, and sensors.

Another possible energy harvesting profiling approach is deploying a particular energy harvester device and measuring its output voltages and currents under a range of loading conditions. This approach was presented by [58], where they periodically sweep the resistance of a load connected to the harvester output, recording the harvester's resulting output currents and voltages during each sweep. By assuming stable harvesting conditions across each load sweep, the set of measured voltages and currents during a sweep is an I-V curve that models the harvester's output behavior for this harvesting conditions. This I-V curve can be used by SPIoT circuit designers to calculate the maximum power that the harvester can deliver at each moment in time (also known as maximum power point) and to simulate the efficiency of power management circuits that convert voltage levels provided by the harvester to voltage levels suitable to remaining SPIoT device components. While this approach is less general than measuring the raw available power, it is more easily translatable to parameters useful to circuit designers and it allows them to make a more realistic estimation of available energy.

By deploying an energy harvesting front-end consisting of harvester, power management unit circuit and a constant-voltage current sink as a virtual battery, the authors of [115] collected voltages and current measurements of both the harvester output and the power management circuit output. This approach is still less general than the harvest output characterization, but it provides realistic estimates of available power to SPIoT circuit designers as long as they use the same energy harvesting front-end. One of the challenges of adopting this approach is to accurately capture current output from power management units using pulse frequency modulation because of the high sampling frequency required to properly characterize these fast current pulses. Another limitation is that the measured P_{harv} depends on the internal power management unit state, since the integrated circuit component that they adopted in their study (BQ25505) performs maximum power point tracking around every 16 seconds. This means that the voltages and currents measured for the power management unit are usually close, but they could not be exactly at the maximum power point of the harvester.

Finally, another alternative indirect energy harvesting profiling approach would be to actually deploy a complete SPIoT solution and monitor its energy storage state of charge and the device's operation behavior. This approach results in the lowest flexibility to translate energy harvesting measurements to other applications, but, as a field test, it can be very informative to a SPIoT device designer. Other major drawbacks of this approach is the information loss when the energy storage voltage saturates and when the devices suffers a power outage.

As a good trade-off between realistic energy harvesting representation and flexibility to apply the profiles in circuit level simulations, we selected the harvester I-V curve monitoring approach described by [58] as our energy profiling methodology. We assume that collecting a time-series of I-V curves can provide enough information for SPIoT circuit designers to optimize their power management unit circuits and enables realistic SPIoT generated energy simulation.

4.2 Energy harvesting profiling platform

To collect I-V curves from harvesters, we need a controllable load and circuits to measure current and voltage. If the controlled load is resistive, currents can be calculated by dividing the measured voltage values by the known resistance values. Then a controller is required to timely sweep the controlled load values and collect voltage measurements. The general architecture of the designed energy harvesting profiler platform is depicted in Figure 4.3, where the harvester output is connected to a controlled load and voltage measurements are amplified by a programmable gain amplifier (PGA) and taken by an analog-to-digital converter(ADC). A controller is responsible to read ADC values and control the resitive load values through an inter-integrated circuit (I^2C) interface. Measurements are then stored locally and transmitted to a database hosted in the cloud.

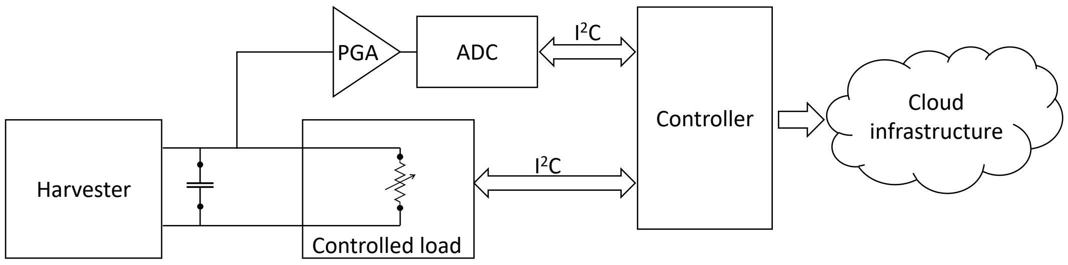


Figure 4.3: **Block diagram of an energy harvesting profiler using the harvester I-V curve method.** Harvester output is connected to a small filtering capacitor and a controlled resistive load. Harvester's output voltage is amplified by a PGA and read by an ADC. A controller is responsible to set the controlled load resistance values, collect voltage measurements and locally store or transmit collected data to a cloud-hosted database.

To collect energy harvesting profiles of thermal energy sources, we selected a commercial thermoelectric generator (TEG) module that was compact and easy to use, the EHA-PA1AN1-R03 which is manufactured by II-VI Marlow Industries. Since the TEG device manufacturer indicated that the optimal load resistance for this TEG is around 1.5 ohms, we designed a controlled load circuit to have configurable resistances around this value and capture points evenly spread in the range of the TEG I-V curve. We selected the resistance values of 0.1, 0.47, 1.5, 4.7 ohms and open circuit as the possible controlled resistance values to characterize the TEG I-V curve. To calibrate the platform measurements, We measured the resistance of each active channel (0.2251, 0.5990, 1.6245, 4.8070 ohms respectively), as well as the short-circuited probe's resistance (0.1080 ohms). The ADC used was an ADS1015, configured to provide 12-bit voltage readings and a gain of 8, resulting in a voltage range of [-0.512V, +0.512V] and resolution of 0.25mV. The interface between the TEG and the controlled load and measurement setup is depicted in Figure 4.4 and a typical TEG I-V curve is depicted in Figure 4.5.

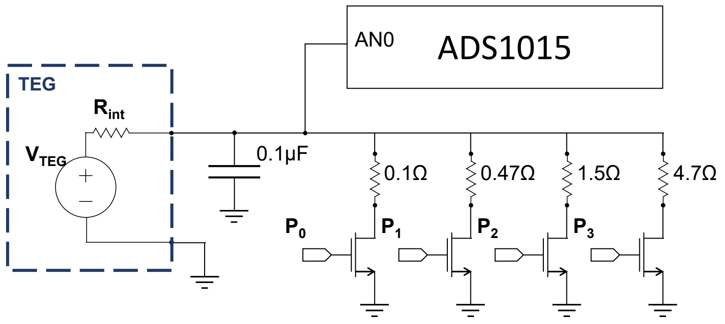


Figure 4.4: **TEG I-V curve profiler interface.** We consider the TEG model as an internal resistance in series with a voltage source and we connect its output to a filtering ceramic capacitor, a controlled load and an analog to digital converter. Our controlled load circuit interface uses four resistance paths controlled by low on-resistance MOSFETs operating as switches. We use an ADS1015 integrated circuit to perform the PGA and ADC functions.

To capture some information about the environment conditions for which the TEG I-V curves were recorded, we added a thermocouple type K temperature sensor and respective off-the-shelf conditioning circuit to the energy harvesting profiler recording platform. The temperature sensor is attached to the same thermal energy source surface as the TEG device and it records the hot temperature surface and the ambient temperature. We selected a Raspberry pi as the controller of the energy harvesting profiler recording platform and programed it to set the controlled load resistance path, collect ADC and temperature sensor readings and manage data record and cloud database streaming. The block diagram for the complete TEG energy harvesting recording platform is depicted in Figure 4.6. The thermal energy harvesting recording platform uses the components listed in Table 4.1 and the deployed platform is depicted in Figure 4.7.

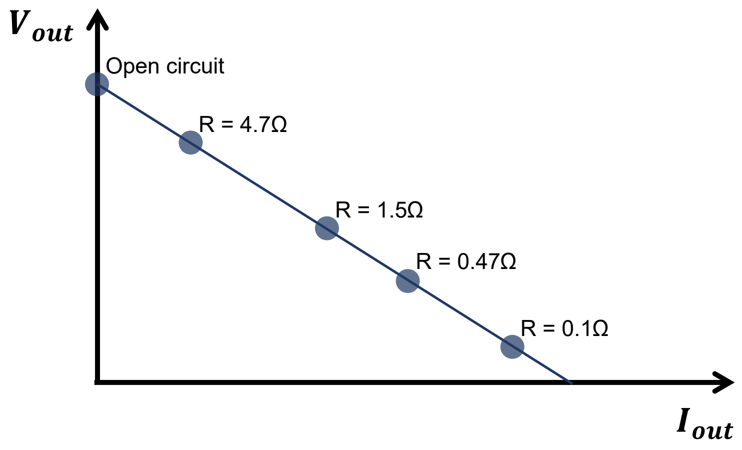


Figure 4.5: **TEG I-V curve measurements example.** By switching the load resistance, we can sample different points in the TEG I-V curve.If, for instance, the TEG internal resistance is 1.5 ohms, connecting a load resistance of this same value will result in a voltage output of half the magnitude of the open circuit voltage.

Table 4.1: Thermal energy harvesting recording platform components

Component list specification
Raspberry pi 3 model A+ with a 32GB microSDHC UHS-1 A1 card (Sandisk) and a 2.5A microUSB power supply (Pro-elec).
Qwiic pHat board (Sparkfun) and 2 qwiic cables to connect modules over I2C interface.
Qwiic Thermocouple Amplifier - MCP9600 (Sparkfun) with a K-type thermocouple (Pimoroni).
Custom PCB for TEG I-V curve profiling based on PCA9536 and ADS1015 integrated circuits.
Mini-Harvester Thermal Energy Generator (Marlow EHA-PA1AN1-R03) with 1.5 ohms optimal load and 20x20mm surface area.

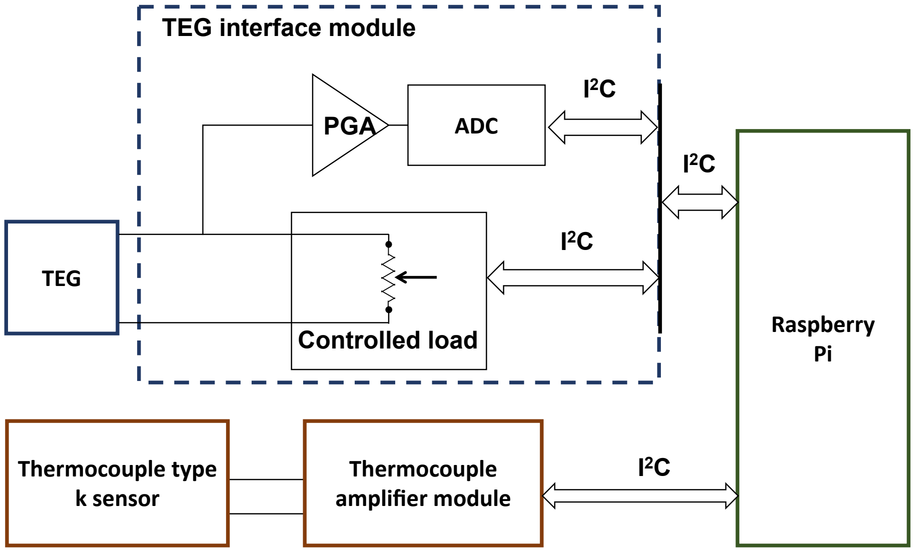


Figure 4.6: **TEG energy harvesting recording platform.** The designed platform records the TEG voltage outputs for five different conditions: open-circuit and load resistance set to 0.1, 0.47, 1.5 and 4.7 ohms. We record both the energy source surface temperature where the TEG is attached and ambient temperature. The whole system is controlled by a Raspberry pi, also responsible to manage the collected data.

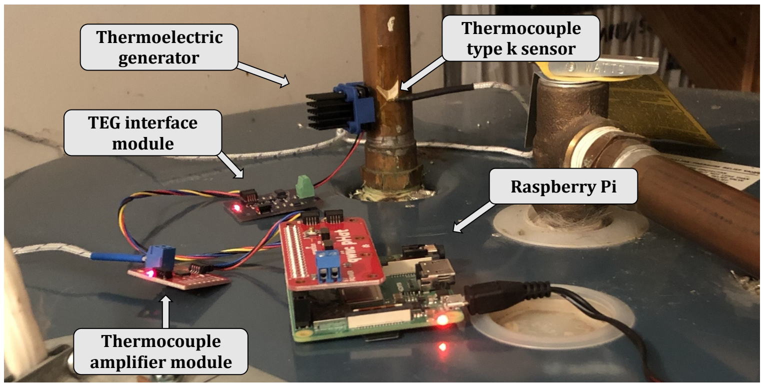


Figure 4.7: **Thermal energy harvesting recording platform deployed at a water boiler location.** (dataset: TEG001_env1.h5).

4.3 Thermal Energy Source Profiles in Residential Settings

As a demonstration of the thermal energy harvesting profiling platform, we conducted a study to characterize thermal energy sources in residential settings[121]. We submitted this study to the University of Virginia's Social and Behavioral Sciences Institutional Review Board (IRB-SBS) and we obtained a waiver (UVA IRB-SBS # 4406) to deploy the energy harvesting profiler boards at the residence of volunteers. Since this study was conducted during the COVID pandemic, we kept social distancing by asking the volunteers to install the device themselves, following up with a zoom call to guide their steps.

In this study we deployed the thermal energy harvesting profilers in 16 different locations including pipes carrying warm water, electronic devices, a refrigerator compressor, and an HVAC vent. The data collection period varied for each location, the shortest deployment lasting 19 days and the longest 53 days. In total, the dataset contains 544 days worth of thermal energy recordings. Using the collected measurements, we estimated the TEG's internal resistance, then computed the TEG's maximum power point (MPP) and simulated the output current of a LTC3108 boost converter circuit. To the best of our knowledge this was the first study to spatially and temporally characterize thermal energy sources in residential settings for multiple weeks. This study generated a dataset available in [76] through LibraData, the University of Virginia's data repository (`https://doi.org/10.18130/V3/M9CP9C`). The thermal energy harvesting trace collections represents the highlighted block in the SPIoT design framework flow chart of Figure 4.8.

4.3.1 The dataset

The thermal energy harvesting recording platform was configured to perform one measurement every 0.5 seconds consisting of: the TEG hot surface and ambient temperature; and the TEG output voltage under open-circuit condition as well as connected to each load resistor (0.1, 0.47, 1.5 and 4.7 ohms). Measured data points are saved as CSV files in the Raspberry Pi SD card. The sampling period of 0.5 seconds was chosen as a compromise between data set size and time domain resolution to monitor typical thermal sources present in residential settings.

Volunteers installed the energy harvesting recording platform in 16 residential locations following an approved IRB protocol and they provided a picture of each installation setup, shared as part of the dataset. Seven of the selected locations were over warm water conducting pipes close to water boilers. Five were under sinks, either over warm water conducting pipes or over wastewater conducting pipes. The following locations only had one deployment: over a WiFi router, over a NAS data storage system, attached to a heater system, and over a refrigerator's compressor. Due to practical deployment challenges of the profiling devices, accidents during the deployment resulted in disconnected TEG devices or invalid temperature readings.

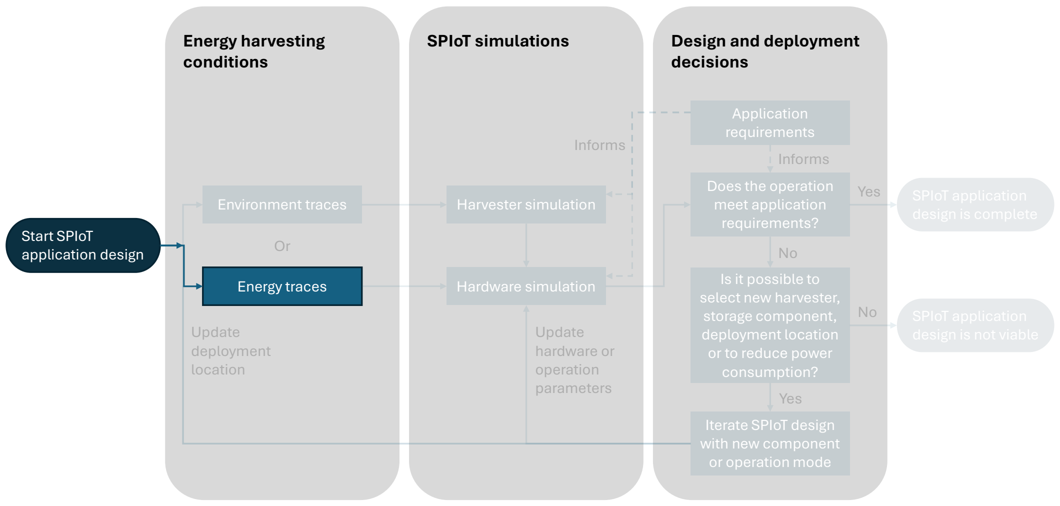

Energy harvesting conditions
SPIoT simulations
Design and deployment decisions
Start SPIoT application design
Environment traces
Or
Energy traces
Update deployment location
Harvester simulation
Hardware simulation
Update hardware or operation parameters
Informs
Application requirements
Informs
Does the operation meet application requirements?
Yes
SPIoT application design is complete
No
Is it possible to select new harvester, storage component, deployment location or to reduce power consumption?
No
SPIoT application design is not viable
Yes
Iterate SPIoT design with new component or operation mode

Figure 4.8: **Flow chart components related with thermal energy source profiles.** Collected I-V curves can be used directly with the SPIoT hardware model to simulate the device operation.

To help filtering data, boolean flag columns were added to the dataset to identify data points related to poor deployment conditions.

To complement the dataset, we estimated the TEG internal resistance from our measurements by performing a least squares fit of the I-V curve data points at each timestamp value. For the used TEG device under typical operating conditions, the estimated internal resistance is approximated to either 1.2 or 1.3 ohms. Using the estimated internal resistance, we calculated the maximum power point (MPP) of the TEG device and the maximum power density of this device by dividing the MPP to the TEG surface area. Finally, we interpolated the LTC3108 curves provided in the manufacturer data sheet [19] to derive the relationship between TEG open-circuit voltage and output charging current for the estimated TEG internal resistance of either 1.2 ohms or 1.3 ohms. This open-circuit voltage to charge current curve represents a LTC3108 based boost converter circuit with a 1:100 ratio transformer as specified in page five of [19]. All the dataset measured and calculated parameters are summarized in Table 4.2. The dataset was formatted using python Pandas module as dataframes and saved as Hierarchical Data Format (HDF) files using the function "`pandas.to_hdf()`" with default settings.

4.3.2 Dataset usecase examples

A straight forward dataset use case would be evaluating how much energy can be generated at different locations as a first estimate of application feasibility. For instance, the calculated TEG MPP output represents the best case scenario for the used TEG device in the recorded conditions. Figure 4.10 depicts power generation distribution at MPP for two boiler locations deployments as in Figure 4.7. An IoT designer could use these distributions to evaluate what fraction of the time the power available could support an specific application or it could also be used to determine if an application outcome is expected to be similar for these locations. As a reference, the harvesting levels of hundreds of micro-watts recorded at the boiler environments is comparable to the energy harvested by small solar cells in low-level indoor light conditions [115].

While the TEG MPP represents an upper bound on the available energy to an IoT device, the boost converter output current provides a more realistic estimation of the IoT's net usable energy by taking into account practical circuit efficiency parameters. Using the TEG internal resistance and the LTC3108 boost converter circuit model curves previously mentioned, it is possible to estimate the output current that is available to charge a storage device, for instance a super capacitor. Figure 4.9 depicts the estimated charging current output profile of the LTC3108 boost converter if it was connected to the TEG in the recorded environments. This profile can inform how long it would take to a capacitor to fully charge in these conditions and what is the expected maximum average current consumption of a energy harvesting IoT device for this environment. Furthemore, energy harvesting prediction algorithms to adapt IoT operation given energy generation fluctuations can also be evaluated using this dataset as performed by [115].

Table 4.2: Thermal energy harvesting profiler dataset parameters

Parameter	Unit	Description
`timestamp`	milliseconds	Unix timestamp in UTC
`voltage_open_circuit`	volts	TEG open-circuit output voltage
`voltage_R_p1`	volts	TEG output voltage with 0.1 ohm load
`voltage_R_p47`	volts	TEG output voltage with 0.47 ohm load
`voltage_R_1p5`	volts	TEG output voltage with 1.5 ohm load
`voltage_R_4p7`	volts	TEG output voltage with 4.7 ohm load
`temperature_ambient`	degree Celsius	Ambient temperature
`temperature_surface`	degree Celsius	TEG hot surface temperature
`teg_internal_resistance`	ohms	Internal resistance of the TEG model
`teg_mpp_uw`	microwatts	TEG maximum power point (MPP) output
`teg_mpp_density_uw_per_cm2`	microwatts / centimeter squared	TEG MPP density
`boost_voc_mv`	millivolts	TEG open circuit voltage input for LTC3108 model
`boost_ichg_ua`	microamperes	Charging current output of LTC3108 model
`flag_thermocouple_invalid`	-	Boolean flag for invalid temperature measurements
`flag_teg_disconnected`	-	Boolean flag for invalid TEG measurements

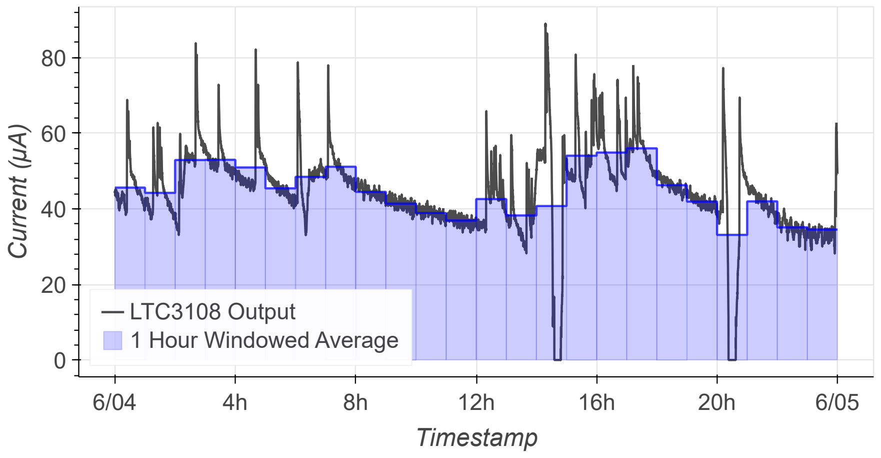


Figure 4.9: **LTC3108 output current.**Using the dataset with a LTC3108 circuit model, we simulated the net current available to an IoT device. For the shown period of dataset TEG001_env1, an IoT designer can select an energy storage device to buffer energy through one hour windows and set an energy-adaptive device to operate with average current consumption modes between 32 and 56 μA..

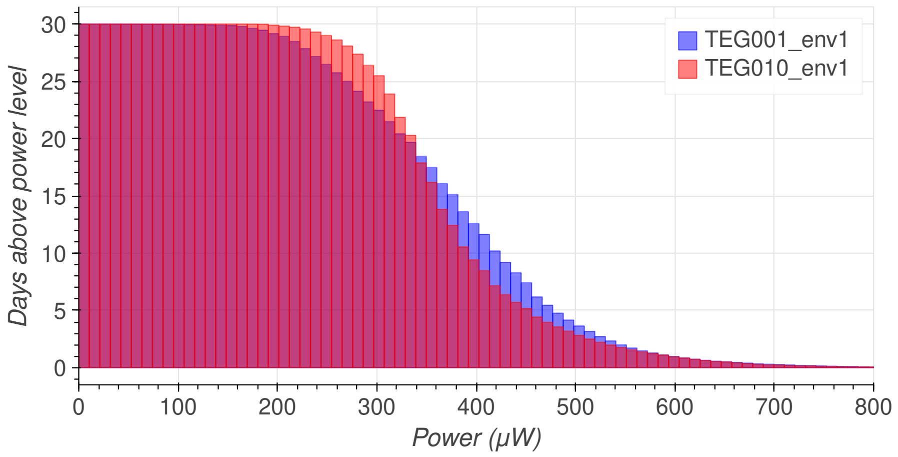


Figure 4.10: **Comparison of energy harvesting at two boiler locations.** Both recorded environments near boilers have potential to generate 330 μW or more for two thirds of the time, however environment TEG010_env1 can generate at least 260 μW more consistently than TEG001_env1.

4.4 Contributions and Outcomes

From the best of our knowledge, our data collection study resulted in the first publicly available multiple-week dataset on thermal energy harvesting and also the first available dataset of its length using I-V curve profiling technique. We show that we can characterize thermal energy harvesting output from a commercially available thermoelectric generator, using a custom interface printed circuit board and off-the-shelf components such as a raspberry pi and sensor breakout boards. Our dataset complements state-of-the-art energy harvesting profiling studies [116, 115], and earlier literature [54, 53] by providing a dataset for a new harvesting modality in thermal energy, while also using the data collection approach introduced by [57].This thermal energy harvesting dataset can be used by SPIoT designers to simulate, compare and evaluate device design solutions powered by thermal energy sources as we previously demonstrated. This dataset and data acquisition platform will facilitate SPIoT designs adopting thermal energy harvesting modality.

Chapter 5: Energy Harvesting Score

Energy harvesting (EH) is a promising solution to power Internet of Things (IoT) devices without burdensome and wasteful battery replacement, enabling manageable large-scale IoT systems to be deployed. However, after two decades of research [81, 138, 80, 59, 60, 39] and compelling predictions [60], widespread adoption of small, indoor, energy efficient energy harvesting devices remains elusive.

Why has adoption remained limited, with only a few commercially available examples [46, 48]? We posit the *inherent uncertainty* of devices powered by energy harvesting hinders their marketplace acceptance. A battery powered device will work out-of-the-box every time as the device designer can specify and provide a suitable battery. With energy harvesting, that guarantee is much more tenuous; the device designer cannot control, or even know, the energy availability where the device is deployed. Likewise, the device user also does not know what level of available energy the device designer expected for the particular IoT device.

Existing approaches for this uncertainty aim to allow devices to operate even with minimal intermittent harvestable energy. Approaches including checkpointing [107, 119, 3], dynamic power management[70, 30, 75, 2], and energy-neutral computing [134, 117, 128] all equip devices to operate even with variable energy availability. Yet, these approaches still operate within expected bounds: too little energy and the device will fail to meet user expectations (e.g., not sampling data often enough) or not operate at all; too much energy and the device may be overprovisioned for the deployment (e.g., too large or expensive). Ultimately, while these techniques do address some of the challenges with computing on harvested energy, they do not resolve the uncertainty about how a device will perform in a given environment.

We claim a new technique is needed to bridge the gap between the expectations of the device manufacturer and the realities of the user's deployment. To this end, we propose a new rating system for both energy-generating environments and energy harvesting devices. We introduce the energy harvesting score, a rating from 0-100 that holistically captures not just the *amount* of available energy in an environment but also its utility for real-world devices. A score of 0 indicates that there is virtually no harvestable energy, and a score of 100 indicates that there is abundant available energy suitable to power a wide range of energy harvesting devices. In addition to

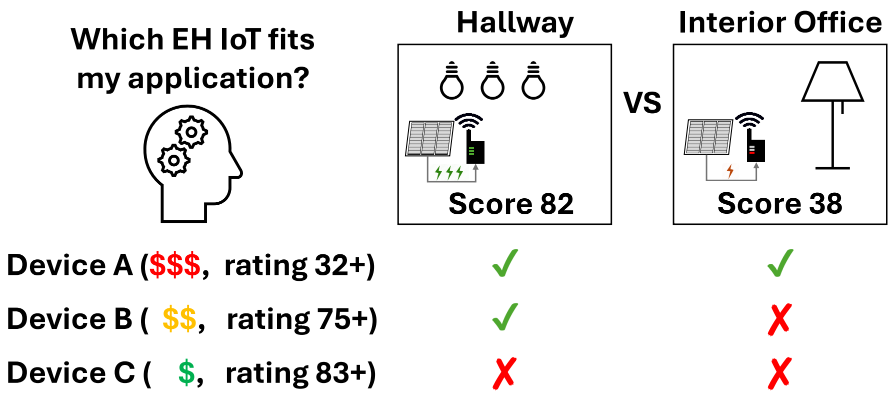

Which EH IoT fits my application?
Hallway
Score 82
VS
Interior Office
Score 38
Device A ($$$, rating 32+)
✓
✓
Device B ($$, rating 75+)
✓
✗
Device C ($, rating 83+)
✗
✗

Figure 5.1: Energy harvesting score framework. IoT users can determine if an energy harvesting device is suitable for their deployment by comparing the device's rating to the environment's energy harvesting score.

scoring how much energy is available, our approach also serves as a rating for energy harvesting devices. Manufacturers can rate their energy harvesting device to indicate what energy harvesting score this device needs to operate as expected. If the device is deployed in an environment with a score of at least its rating, the device will have the energy available to operate as the designer expects. This effectively eliminates the uncertainty with using energy harvesting devices.

This approach abstracts the complexity of reasoning about how a device's energy requirements match with what energy is available in a particular space. Today, to estimate how well an energy harvesting device will operate, users need to not only have a deep knowledge of the device's energy consumption behavior, but also need to estimate how much energy this device will be able to harvest over its lifetime wherever it is deployed. We believe this uncertainty leads to risk when using energy harvesting devices, serving as a major adoption barrier for energy harvesting IoT solutions as users usually cannot quantify the risk of failing to meet application goals.

Other approaches for characterizing available energy provide extensive, detailed information about what energy is available. Sigrist et al. performed a long-term EH profiling experiment [116], characterizing indoor solar energy harvesting in office spaces over multiple years. Such traces enable researchers to simulate energy harvesting devices in such environments to determine device requirements to achieve levels of availability comparable with battery powered devices. Additionally, tools like Ekho [57] capture and replay full I-V curves to simulate devices under precise conditions.

What is missing from these techniques is that they assume the specific energy-harvesting device is known to test under the captured conditions. This is beneficial for device manufacturers, but doesn't provide a generic way to reason about how devices will operate in arbitrary environments. This mismatch occurs due to two reasons. First, analyzing a harvesting trace requires knowing more than just the available energy, it requires considering the combination of a device's energy harvester, energy storage, and energy consumption. Depending on exactly how much energy is available and when, how much energy the device already has stored and how much it can store, and what the device's instantaneous energy needs are determines how the device will actually operate. As such, capturing a trace and performing just a statistical summary (e.g., an average) is insufficient for scoring an environment. For example, a "spiky" harvesting trace with high, short peaks of harvestable power might have a reasonably high average, but a device with a small energy storage element will be unable to effectively capture the available energy. Second, the device manufacturer is likely not the device installer, and while the manufacturer can test the device under various conditions, the manufacturer does not know *which* conditions are representative of a particular deployment. Further, without laborious in-situ profiling, it is difficult for the manufacturer to learn that information. What is missing is a way to characterize each side (the manufactured device and the deployment environment) independently, and then compare when making purchasing and installation decisions.

To accomplish this, our approach is able to characterize an environment by simulating how a representative energy harvesting device would perform in that environment. Simulating the device's operation is key to understand failures due to long energy droughts or lost energy due to a full energy reservoir. If the average device performs exactly as expected we assign a score of 50. If the device performs above its expected utility (e.g., samples more frequently), we proportionally increase the score. If the device must be enhanced to operate as expected (e.g., with a larger harvester or more energy storage), we decrease the score. While conceptually simple, enabling this property in a way that is useful for rating devices as well requires a particular mechanism for mapping increased or decreased performance to the score. Also, this scoring can be done independently of knowing the types of devices that are available or which devices the user might be considering. With the scoring capability, we can then assign scores to the traces (for example traces generated by Ekho [57]) manufacturers use when testing their devices. The score of the minimal trace where their device operates as expected becomes the device's rating. The device will then operate successfully in environments with a score at or above the device's rating. Now, comparing potential devices with a particular deployment no longer requires evaluating traces, but instead is just comparing two numbers.

We define the energy harvesting score algorithm and create a prototype to calculate the score for both rooms in energy harvesting datasets as well as from our own building. We then show how various spaces compare, and how spaces with similar average harvestable power traces can actually have diverging utility for real devices. We also prototype a realistic energy-harvesting device, assign it a rating, and then compare how it functions in spaces with scores above and below that rating. The prototype achieves 100 % availability in the higher score space and 55 % availability in the lower score space.

With this design, we contribute a new approach for helping realize the vision of batteryless devices by mitigating the risk adopters encounter when sacrificing the predictability of batteries for the uncertainties of energy harvesting. We provide a new algorithm for characterizing the utility of an energy-harvesting trace for actual devices that is not only representative but also prescriptive as it facilitates matching devices with environments. Additionally, we provide realistic use cases for this approach, and implement an end-to-end prototype to demonstrate how this approach works in representative environments. The energy harvesting score introduced in this chapter uses all the highlighted flow chart components of SPIoT design and deployment framework shown in Figure 5.2.

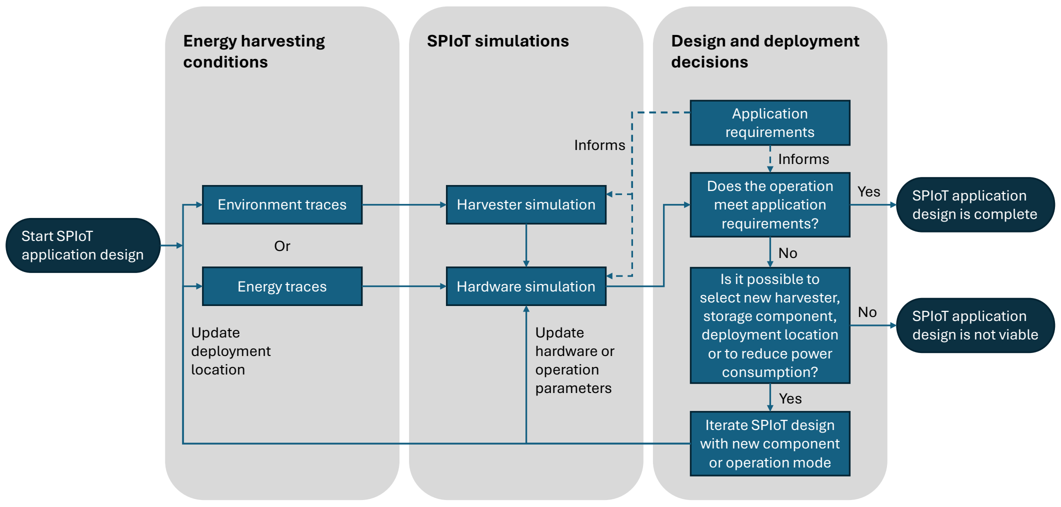

Energy harvesting conditions
SPIoT simulations
Design and deployment decisions
Application requirements
Informs
Informs
Start SPIoT application design
Environment traces
Or
Energy traces
Harvester simulation
Hardware simulation
Does the operation meet application requirements?
Yes
SPIoT application design is complete
No
Is it possible to select new harvester, storage component, deployment location or to reduce power consumption?
No
SPIoT application design is not viable
Yes
Iterate SPIoT design with new component or operation mode
Update deployment location
Update hardware or operation parameters

Figure 5.2: **Flow chart components related with the energy harvesting score.** To obtain the energy harvesting scores, we simulate a reference SPIoT device with the energy harvesting dataset collected by Sigrist et al. [115] and with the energy harvesting conditions estimated from our office spaces illuminance dataset.

5.1 Energy Harvesting and IoT Modeling Complexity

A complete model of energy-harvesting operation should determine how much energy an EH IoT device has at any location at any point in time. However, creating an accurate model is challenging on multiple fronts. First, energy availability often depends on physical phenomena and human behavior, which can be difficult to model. Second, energy consumption by the device requires models for energy conversion, storage and consumption for each device design. And third, the first two facets are coupled, meaning they cannot be modeled entirely individually. In this section we describe these various modeling challenges to highlight what our proposed energy-harvesting score design must be able to address.

For simplicity, we use a photovoltaic-powered IoT device to exemplify the modeling challenges. However, these challenges exist with other harvesting modalities as well.

5.1.1 Energy Harvesting Challenges

Stochastic events: Consider a solar-powered IoT device in a conference room illuminated with artificial light: energy generation is correlated to the state of the room's light. Modeling this energy source would require us to model human behavior related with this room's occupancy, what is often challenging.
Multiple sources: If a solar-powered sensor is exposed to a combination of natural and artificial lights, the resulting model would be required to capture both behaviors, what increases the complexity of the model.
Local deployment variation: Deploying a solar-powered IoT device in different locations inside the same room can drastically impact the energy harvesting availability for that device, due to light intensity variations, and the angle between panel and light source. A scaling factor can be used to model the effect of these deployment variations, but like the harvesting parameters it is not straightforward to estimate.

5.1.2 IoT Device Challenges

Harvester parameters: The efficiency of solar panels varies with the angle, intensity and spectrum of the light source. The manufacturing model of a solar panel and its size can be represented as a scaling factor for the generated energy by an EH IoT, but this factor is not straightforward to estimate.
Power management circuit parameters: Energy conversion circuits can improve harvesting efficiency by employing maximum power point tracking, increase the energy storage capacity by charging capacitors at high voltages, and provide regulated voltage sources to the efficient operation of the remaining IoT components. Modeling

power management circuits is non trivial because energy conversion efficiencies can depend on the coupled state of harvester, storage and load.
Energy storage parameters: An ideal energy storage device would be capable of buffering an unlimited quantity of energy, filtering any harvesting trace and providing a constant energy output. However, realistic energy storage devices are complex components with leakage, charging and discharging efficiency and capacity limitations, while also being subject to aging and temperature effects.
Operation driven by application: Energy requirements of an EH IoT device can be complex or depend on human behavior. For example, a door open/close sensor activates when a door is opened or closed, and understanding the workload requires modeling human behavior. For sensors with more complicated analytics (e.g. on-device machine learning algorithms) the compute requirements can vary significantly.

5.1.3 Energy Harvesting-Consumption Coupling Challenges

Energy Generation Spikes: Large energy generation events (energy spikes) can surpass the buffering capabilities of the EH IoT energy storage component and result in energy being wasted as there is no on-board capacity to store it. Due to this EH IoT device limitation, a model for energy harvesting conditions should not over value energy generation spikes.
EH drought periods: On the other hand, severe low harvesting events can cause the EH IoT device to run out of energy even if the energy storage was at full capacity before the event. A useful model for energy harvesting conditions should consider droughts as a primary factor to determine if an EH IoT device is suitable to an energy harvesting environment.

5.2 Vision and Use Cases

To contextualize our design, we first articulate our vision for an energy harvesting score which attempts to reduce the uncertainty in using energy-harvesting devices in real-world use cases. Then, we illustrate some example scenarios of how various stakeholders might use the energy harvesting score.

5.2.1 Energy Harvesting Score

We envision the energy harvesting score (EH score) as a number from 0-100 that captures how much energy is available to a realistic EH IoT device in a particular location. The higher the score, the more usable energy is available. Of particular importance is addressing the challenges from Section 5.1.3. The score must not simply be a statistical reduction of an energy trace as that fails to incorporate the limitations of real-world devices. Instead, the score must capture the variability of available energy in that location *and* the ability of a typical EH IoT device to make use of that energy.

The same scale also serves as a rating for devices. Manufacturers of EH IoT devices can assign their devices an energy harvesting rating (EH rating) that denotes the minimal energy harvesting score needed for the device to function as expected. Lower ratings represent more resilient and adaptable EH IoT devices that are able to operate in a wider range of energy harvesting conditions. However, it is infeasible for manufacturers to test in all conditions, so the score must be able to simplify rating a device.

With these two properties, the EH score can reduce the uncertainty of using EH IoT devices. Users can characterize their spaces and assign a score, and manufacturers can rate their devices on the same scale. If a space has a score higher than the device's rating, the device should work as expected. The EH score improves the communication between designers and adopters to align expectations, support EH IoT deployments, and drive EH IoT adoption.

5.2.2 Example Use Cases for Users

We illustrate potential use cases for users of EH IoT devices.

Informing EH IoT deployments

To decide if an EH IoT is a viable option to their application, adopters can estimate the EH score of their spaces and compare with the rating of their EH IoT device options. For example, consider a facility that wants to install solar-powered temperature and humidity sensors in an office space. First they would calculate the EH score of their rooms by measuring the illuminance level when lights are on and the duty cycle of those lights. Then, they can compare that EH score to the ratings of available EH IoT devices. If there are no EH IoT devices available with rating below the EH score value, the facility can choose between accepting reduced data availability with intermittent operation, or using battery powered devices.

Adapting EH IoT deployments

EH IoT adopters can use EH score calculations to understand how changes to their spaces can impact available energy to EH IoT devices. For example, a solar-powered air quality sensor can have plenty of natural light coming from a nearby window, then experience energy shortages if the user decides to install sun-blocking curtains. Other example situations affecting energy generation can be: changing lamp models from diffuse to spot illumination; and reallocating furniture and causing a shaded area over the IoT device. EH IoT adopters can recalculate the EH score of the new spaces and decide if they will need to replace the EH IoT for another one with lower rating, move the EH IoT deployment location, or use primary battery IoT options.

5.2.3 Example Use Cases for Manufacturers

We illustrate potential use cases for manufactures of EH IoT devices.

Commercializing EH IoT devices

EH score rating can communicate the value of an engineering investment to create a more efficient EH IoT device. This device will have a lower rating (i.e., it can operate effectively with less available energy) and signal that the device can be used more broadly. This can also justify an increased cost for the device. For example, an IoT manufacturer can have two versions of a solar-powered outdoor camera, a more expensive product containing a complex power management circuitry, and another cheaper product with a simpler diode-based solution. A lower EH IoT rating can be used to justify to a consumer a higher price tag, while also helping EH IoT adopters to avoid frustration with devices incapable of consistent operation in the energy harvesting conditions of the deployment location.

Standardization and communication

To efficiently communicate the EH IoT requirements, manufacturers can include EH rating information in their device's product manual and data sheet. Manufacturers could also provide the respective EH rating values for power-saving operating modes of EH IoT, enabling the EH IoT adopters to understand the energy requirements for each device use case. Extensive investigation of EH scores in typical IoT deployment spaces can inform EH IoT manufacturers what range of EH conditions they are expected to support.

5.3 Design

To introduce our rationale behind designing the energy harvesting scoring system, we gradually increase complexity of potential score metrics until we meet the vision and use cases discussed in Section 5.2. Then we discuss a methodology derive the energy harvesting rate of an EH IoT device. Finally we discuss the steps to calculate the EH score of a real world energy harvesting trace.

5.3.1 Energy Harvesting Score

The score must holistically capture the available harvestable energy in an environment. The score is generic to any harvester modality. We start with a simple approach, explain why that is inadequate, and build to our proposed solution.

Calculating mean energy as an EH score

Mean available energy is a simple and intuitive way of representing energy harvesting environments. An ideal EH IoT device is able to convert any energy generation trace into a constant energy source with amplitude equal to the mean energy value, given enough time. Therefore, mean energy is a good upper bound on how much energy can be harvested by an EH IoT device in a given location.

However, energy buffering capabilities are limited in real-world EH IoT devices. Energy storage components are limited to a maximum capacity and they can't store arbitrarily large bursts of energy. Therefore, simply averaging available energy is ineffective when energy generation is very inconsistent, such as in large bursts followed by long droughts.

Incorporating harvesting consistency in an EH score

To better quantify inconsistent and bursty energy, one way is to consider how average energy varies with time in a rolling fixed-length time window. For illustration, assume we calculate the average energy generation for the first 24 hours of a week long EH time series. Then we shift this 24 hour window by one hour, and we repeat this calculation. We continue until the end of the week. The resulting points represent the distribution of average generated energy expected within every one-day long time window. The lowest value is the 24 hour period with lowest average energy generation, and therefore characterizes the energy generation drought that an EH IoT device must be able to tolerate.

While averaging energy generation within a fixed time-window provides valuable insight on the effects of limited energy storage capacity, understanding how much storage is needed requires knowing the power draw of the device in its operation.

Including energy consumption

To create an EH score that takes into account application-level consumption (i.e., for tasks like compute and communication), we need a reference EH IoT device model with power consumption requirements. We can then simulate this reference design using the available energy trace and estimate what proportion of time the device will have enough available energy to meet its energy consumption requirements. However, we must define what a reasonable reference design is.

To select a reasonable energy consumption profile, we analyze real-world traces and assume that for a reasonable consumption profile: (1) energy generation and consumption are reasonably balanced in typical harvesting environments; (2) there exists surplus generation in high harvesting environments; and (3) there exists insufficient generation to meet consumption demands in poor harvesting environments. These assumptions are based on assuming that devices will be developed to match their general deployment scenarios. If this assumption does not hold, then either devices are significantly overprovisioned and will work everywhere, or are very constrained and will operate in nearly no scenarios. Either extreme largely invalidates the utility of the score (as the operation of the device is already known), so we consider these to a reasonable method for determining a reference consumption level. Then, to determine the typical harvesting conditions, we select median average energy traces from real-world energy harvesting measurements. From there we simulate a reference EH IoT device using those traces, varying the energy consumption, and select the consumption level that results in balanced generation-consumption. This then informs how much energy storage is required. However, running this simulation requires a reference IoT device so we can input realistic energy conversion efficiency parameters and energy storage leakage and capacity parameters.

Defining reference devices

To create a reasonable reference model for an IoT device we start with existing models such as the ones available in the literature [70, 31]. To obtain reasonable reference EH IoT device parameters for those models, we search examples of achievable performance from available off-the-shelf components documentation and EH IoT designs in the literature. For instance, commonly used power management circuits for solar energy harvesting claim energy conversion efficiencies of 70% to 90%. As energy storage solutions, EH IoT designs in the literature use super capacitor components with capacitance values around 1 F. As for reference harvesters, energy harvesting profiling datasets have shown reasonable energy generation from off-the shelf solar cells and thermoelectric generators. With this, we have a device model we can use to evaluate harvesting traces under realistic conditions. But, we still require some evaluation criteria.

Defining successful operation

To analyze how well our reference design operates in a given environment, we record the percentage of time the device is available. We consider that the device is available when the energy available to the device (both current harvesting and stored energy) is higher than the minimum required level for the instantaneous intended operation. Essentially, this implies the device operates as it would with a new battery: when the device is sleeping or active is not constrained by the power supply but rather by the device's code. This also has the desirable side effect that the score is compatible with existing energy harvesting programming techniques. Platforms such as checkpointing schemes and energy prediction algorithms can still be very effective at ensuring successful operation. We are only interested in observing when the device is prevented from operating due to lack of energy.

Availability is then the fraction of the deployment or simulation time that the device is available. This enables matching the reference consumption to the available energy within the bounds of realistic hardware to understand the suitability of the environment. However, using availability as the score directly would only allow the score to meaningfully represent a small window of environments (specifically those environments which are similar to the median average traces mentioned in Section 5.2), as for many traces the availability will always be 100%. We require a way to characterize a broader set of environments.

Calculating over- and under-supply

With our setup from Section 5.2, we can calculate the reference device's availability on the harvesting trace from the target location. However, many harvesting traces will likely lead to 100% availability, yet are not necessarily the same. We need a method to differentiate these harvesting traces. To characterize more points, we linearly scale the harvesting trace to increase and decrease the available harvestable energy. This effectively models the differences when using larger or more efficient harvesters (or vice-versa). With this, we can better characterize the harvesting trace.

To effectively match the trace to our reference device, we attempt to find the lowest scaling factor where the device is *just* able to reach 100% availability. This is the point where any less incoming energy means the device suffers an outage. This scaling factor can be above or below one depending on the the environment. If none of the scaling factors in our considered range is suitable we discontinue the search and assign a score of 0 to the environment. With this scaling factor, we derive a second device model with different harvesting parameters.

With this second device model, we simulate both this device and the reference device across the range of scaled harvesting traces to determine each device's availability at each scaling point. An example is shown in Figure 5.3. This then captures both how the reference design functions, but also how well the minimal device that works just as expected in the environment functions. Importantly, we can use this to determine how much better (or worse) the reference design is than the minimal design.

To calculate this difference, we calculate the area between the two availability curves (e.g., area in between the purple and green lines in Figure 5.3). We then normalize this value (i.e., the area) by dividing it by the area above (or below) the availability curve for the reference device. In the case where the reference design outperforms the minimal device, we use the area above the reference availability curve. This effectively computes a ratio of how much of the extra available energy is not required to support the same workload in this environment. If this ratio is close to 1, then the minimal device needs almost none of the excess energy provided by the environment, meaning the score will increase significantly. If the ratio is close to

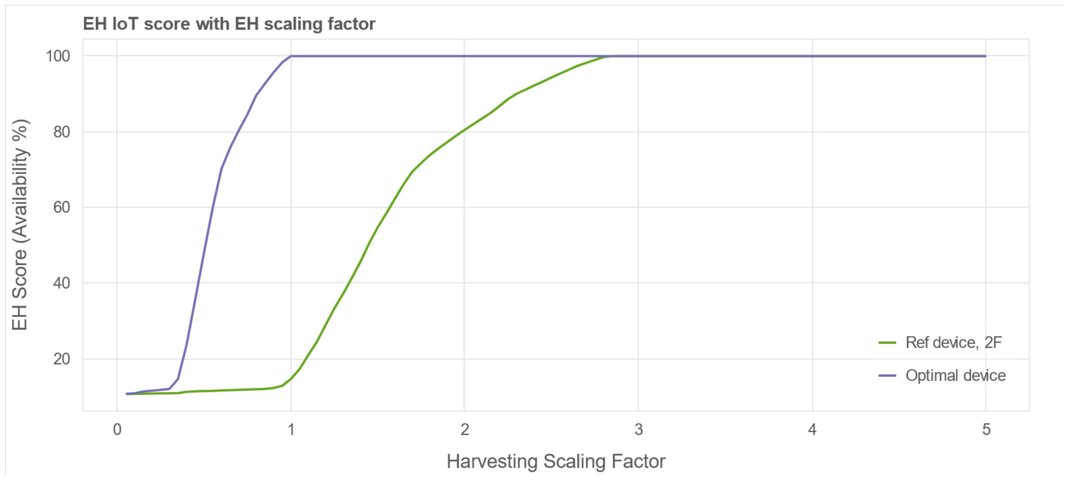


Figure 5.3: Resulting availability from scaled energy generation traces.

0, the minimal device is very similar to the reference device and the score will only increase by a small amount. In the case where the minimal device outperforms the reference device, we use the area below the reference availability curve. This computes the ratio of how much of the hypothetical extra energy is required to have the device work satisfactorily in the given environment. A low ratio means the score drops only a little, and a high ratio means much more energy is required and the score drops more significantly.

Incorporating varying capacitance

With the technique from Section 5.2, the score can reflect the interplay between harvesting capability and a particular space. However, energy harvesting devices can also use varied energy storage elements for size, cost, longevity, leakage, and other reasons. The score must be able to reflect this dimension as well. That is, if an environment has more consistently available energy, and therefore requires a smaller storage element, the score should be higher. This signals that the environment can support more constrained energy harvesting devices.

We incorporate this by repeating the algorithm in Section 5.2 with differing sized capacitors. We record the smallest capacitor that still enables the scaled device to reach 100% availability. We then use that capacitor value to assign the numeric score.

Assigning a numeric score

Finally, we are able to assign a score from 0-100. We first divide the range into sequential bins based on the number of capacitance values used in Section 5.2. For example, if we used four values then we consider bins 1-25, 25-50, 51-75, and 76-100. The harvesting environment is then assigned a bin based on the smallest storage size that was successful. If the smallest capacitor worked, then the trace is assigned the highest bin (i.e., 76-100). If only the largest capacitor worked, then the bin is the lowest (i.e., 1-25).

Within the bin, we use the output of Section 5.2 to determine the specific score. If the ratio was 0, meaning the reference device just worked for the harvesting trace, then the score is set at the middle of the bin. If the reference device was overprovisioned, then we use the ratio to increase the score within the upper half of the bin. If the reference device underperformed the minimal device, we use the ratio to decrease the score within the lower half of the bin.

EH score summary

The EH score reflects how much usable energy can be generated by an EH IoT device at a given location. The score is calculated by estimating how well a realistic reference EH IoT device achieves full operation (i.e., stored energy is always sufficient for intended operation) under a given location's harvesting conditions. If the exact median reference device works exactly as intended (i.e., has no excess energy and 100% availability) the score is 50. If the reference device works better (i.e., has surplus energy) or worse (i.e., does not reach 100% availability), we alter the harvesting capability and storage capacity to determine a new optimal device design point. By comparing that design point to the reference device we can adjust the score accordingly.

Critically, using realistic device simulations and adjusting both storage and harvesting capabilities ensures the score accurately reflects the interplay between the device characteristics and the environmental energy availability characteristics. This makes the score not just a representation of the energy profile, but a realistic summary that is useful for characterizing devices. This, then, makes the score suitable for rating devices and enabling the matching we propose to help reduce the uncertainty when using energy harvesting devices.

5.3.2 EH IoT Device Rating

The energy harvesting rating is the minimum EH score required by the EH IoT device to operate at 100% availability. To obtain a device's rating, the device creator runs the device using a collection of synthetic energy harvesting traces that cover a range of EH scores. Each trace is assigned a score using our scoring algorithm. Then we find the minimum scores of the traces that were able to fully support the device and use those scores to assign a rating to the device.

5.3.3 Calculating the EH Score in Practice

To calculate the EH score of a deployment location, we need to obtain an EH time series representing that space. This EH time series can be obtained by (1) directly measuring the reference device harvester output at the deployment location; (2) measuring variables that can be used to estimate the energy harvesting output, such as temperature for TEGs, or radiance and illuminance for solar cells; (3) modeling the energy source behavior over time, for example modeling the sunlight hours that an outdoor solar harvester will be exposed to. Note that each method to obtain this EH time series has its own uncertainties and assumptions associated with it.

Along with the EH time series, we also need to define the reference EH IoT parameters. We select the desired harvesting modality and typical device parameters such as reference harvester, energy conversion efficiencies and energy storage component characteristics. Then we assume what are typical harvesting conditions to estimate the respective maximum energy consumption that supports nearly 100% availability under these typical conditions. We define a harvester scaling range, and a list of storage capacity values to evaluate. Finally, we start the EH score calculation by simulating the reference EH IoT with lowest capacity under a range of scaled versions of the EH time series previously obtained. If no simulation results in 100% availability, the EH score is zero, otherwise we find the optimal scaling factor by taking the lowest factor that results in 100% availability for the EH IoT device simulation. We assume an optimal EH IoT model with the optimally scaled harvester, and we repeat the simulations with the same harvesting scale factors. Finally we compare the area in between the availability curves to find the EH score for the lowest capacity storage value. If the score is zero, we move on to the next larger capacity parameter and we repeat the previous steps until finding capacity and scaling factors that are adequate for the harvesting conditions. If EH score is zero for all energy storage capacities, the combined EH score is also zero.

5.4 Modeling and Simulations

To derive the EH scoring framework, we introduce the model and parameters used to simulate the reference EH IoT device.

5.4.1 Simulation design

We decided to base this EH score metric analysis on indoor solar modality of energy harvesting, so we could use the EH dataset collected by Sigrist et al. [116] as a reference for harvesters, efficiency parameters, and typical harvesting environments. We decided to perform all simulations to calculate the EH score using python, due to its flexibility and popularity. Our simulations calculate the stored and consumed energy for an EH IoT device.

5.4.2 EH IoT model

We base our EH IoT model on the analysis of energy harvesting devices by Kansal et al. [70], assuming three possible energy paths: from harvester to storage; from storage to load; and from harvester to load. We consider that each path has an efficiency parameter, and the energy storage device has maximum capacity and leakage parameters. The adopted EH IoT model is depicted in Figure 5.5. We consider that at each simulation step the model checks if available energy meets target energy consumption requirements, and we consider that the device is turned off when the stored energy is bellow a certain minimum energy parameter. The complete discrete time simulation model equations are listed in Figure 5.5.

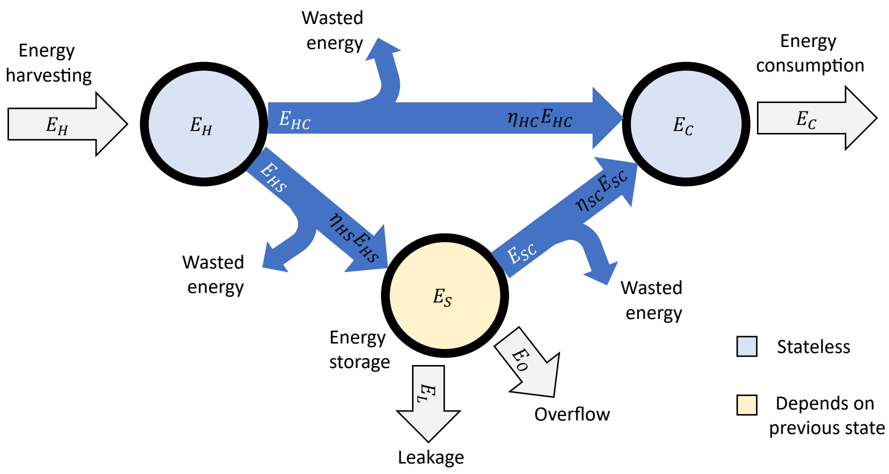


Figure 5.4: **Energy flow diagram for the EH IoT model.** This flow energy model is a high level representation of how an EH IoT device generates, stores and consumes energy. The model assumes losses due to non-ideal energy conversion efficiency, energy storage leakage, and energy storage capacity saturation.

$$\begin{cases} E_{HC}(t) = min(E_H(t), E_C^*(t)/\eta_{HC}) \\ E_{HS}(t) = E_H(t) - E_{HC}(t) \\ E_L(t) = L_R E_S(t-1) \\ E_{SC}(t) = max(min((1/\eta_{SC}) * (E_C^*(t) - \eta_{HC} E_{HC}(t)), E_S(t-1) - E_L(t) - B_{min}), 0) \\ E_C(t) = \eta_{SC} E_{SC}(t) + \eta_{HC} E_{HC}(t) \\ E_O(t) = max(E_S(t-1) + \eta_{HS} E_{HS}(t) - E_{SC}(t) - E_L(t) - B_{capacity}, 0) \\ E_S(t) = max(min(E_S(t-1) + \eta_{HS} E_{HS}(t) - E_{SC}(t) - E_L(t), B_{capacity}), 0) \end{cases}$$

Figure 5.5: **Set of equations for the EH IoT energy flow model.** These equations describe the behavior of the model, capturing efficiency parameters, energy storage leakage and capacity saturation.

5.4.3 Reference device

To derive the parameters of the reference EH IoT device, we first selected power management solutions and energy storage components that are reasonable for indoor solar EH devices. To obtain the energy conversion efficiency parameters, we selected the bq25505 harvesting circuitry used in the indoor solar EH dataset. We select a super capacitor as the energy storage element with 1F capacitance and leakage of 6 μA at 5V. We assume that the required minimum capacitor voltage to operate the EH IoT is 2.1V and the maximum allowed capacitor voltage is 4.5V. We assume average efficiency of 80 % for the energy conversion from harvester to storage, also 80 % efficiency from storage to consumption, and 95 % efficiency for direct harvester to load conversion. We consider a simulation step of one minute, and the EH IoT operation as being one minute of active energy consumption every 10 minutes.

Then we select the typical energy harvesting conditions from the EH dataset [116] by selecting profiles with one-month window length, and calculating average energy generation within each month worth of data. We select the median values of the monthly average energy generation distribution, between the 49 and 51 percentiles. We found that the location referred as "pos06" of the dataset contains most of the 30 day periods with median monthly average energy generation. We use these windowed profiles as a reference of typical energy harvesting conditions for indoor solar applications. We simulate the reference EH IoT device with varying average energy consumption, and we find that consuming 0.54 Joules/day will result in 99 % EH IoT availability in average for the selected energy harvesting profiles. Figure 5.6 depicts the simulations results used to obtain the reference device average energy consumption values.

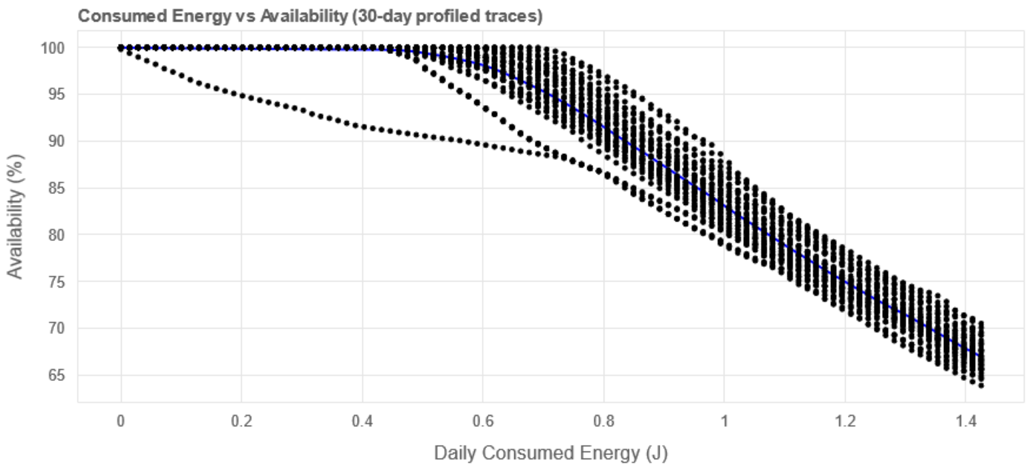


Figure 5.6: Simulating the reference EH IoT device availability for a range of average daily energy consumption values. We find that 0.54 Joules/day is the maximum power consumption value for which most of the simulated typical harvesting conditions achieve 99 % availability.

In summary, the adopted reference EH IoT model parameters are: Harvesting to consumption efficiency of 95 %; Harvesting to storage efficiency of 80 %; Storage to consumption efficiency of 80 %; Maximum stored energy of 10.125 J (1 F, 4.5 V); Minimum operational stored energy of 2.205 J (1 F, 2.1 V); Energy storage leakage rate of 1.44E-4, which is the fraction of the stored energy lost as leakage in one minute (6 μA at 5 V); and Initial stored energy of 6.165 J (half of the usable range).

To cover a wider range of possible EH IoT reference devices, we select a harvesting scaling range, and also a set of energy storage components. We choose the harvesting scale between 0.05 and 5, as to consider harvesters generating between 20 times less energy to five times more energy than the reference IoT device. We consider capacitor values of 2 F, 1 F, 500 mF, 250 mF, and 125 mF as typical choices for EH IoT devices.

5.4.4 EH score calculation

To calculate the EH score, first we take a 30-day period trace window of energy harvesting data as an arbitrary long trace to capture daily energy generation variations. Then we start by assuming the EH IoT model with the lowest capacitor of the reference range (125mF), and running EH IoT simulations for each scaled harvesting trace from 0.05x to 5x in 0.05 steps. If 100 % availability was reached in any of these simulations, we assume the optimal harvester scale as being the lowest scaling factor resulting in 100 % availability. If 100% availability is not reached in any of the simulations, we change the capacitance parameter to the next larger value (500mF) and repeat the previous steps. With the optimal harvester scale, we run again EH IoT simulations for each scaled harvesting trace, now multiplied by the optimal harvester scale. We calculate the normalized area between the optimal device availability and the original reference device availability to obtain an area score. Finally this area score is mapped to the respective region 0-100 range of the EH score (for instance, if the normalized area score was 0.8 with the 125mF storage parameter, the score range is between 80 and 100, the resultant EH score then being 96). If we go through all reference capacitance values without finding simulations resulting in 100 % availability, we assume the EH score is zero. We summarize this procedure in algorithm 1, with the function simRefEHIoT(*cap*,*scale*) returning the availability of the reference EH IoT device with storage element capacitance *cap* and energy harvesting input trace scaled by *scale*.

5.5 Evaluation

To evaluate the energy scoring framework, we perform EH score calculations for energy generation profiles available in the previously adopted dataset [116]. Also using the dataset, we show that it is possible to estimate the energy generation and consequently the EH score from lux data. We then use this lux to power estimation to calculate the EH score of another set of office building spaces for which illuminance data is available to us.

We perform controlled experiments with an energy harvesting front-end connected to a commercial Bluetooth sensor device, showing its energy storage element charging and discharging behavior to estimate its rating. We then deploy two versions of this setup, one in a consistent but low-intensity harvesting location and the other in a consistent and high-intensity harvesting location. We observe the EH IoT operation behavior in these real-world deployment locations to show either intermittent operation, or energy storage charge build up. We also evaluate a third higher consumption EH-IoT device using a similar energy harvesting front-end and a commercial LoRaWAN IoT device. We deploy this device in a location with consistent energy generation but intermediate harvesting intensity when compared to the previous two deployments. We record energy storage level and device operation, showing that the

Algorithm 1: EH score calculation algorithm

```
Scaling factor list = 0.05 until 5, with 0.05 step;
Capacitance list = [125, 250, 500, 1000, 2000];
Function Calculate EH score
    forall cap in Capacitance list do
        forall scale in Scale factor list do
            $avList$ = simRefEHIoT($cap$,$scale$);
        end
        if 100% in avList then
            $scaleList100av$ = find $scale$ where $avList$ is 100%;
            $scale_{opt}$ = min($scaleList100av$);
            forall scale in Scale factor list do
                $avList_{opt}$ = simRefEHIoT($cap$,$scale * scale_{opt}$);
            end
            $areaScore$ = Find and normalize area between $avList$ and
              $avList_{opt}$;
            $EHscore$ = $areaScore$ mapped to respective 0-100% region of
              respective $cap$ value;
            return $EHscore$
        end
    end
    return zero
end
```

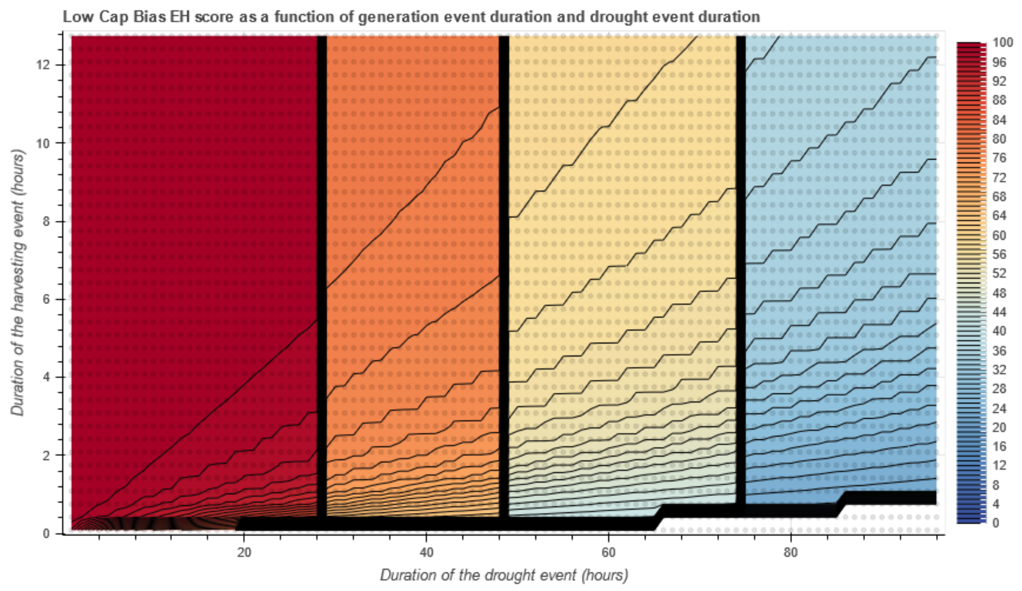


Figure 5.7: Simulating the reference EH Score for synthetic harvesting traces. We assume simple traces of energy generation events of 750 lux intensity followed for droughts of zero energy generation. The vertical axis represents how long energy generation bursts last, and the horizontal axis how long the droughts last. The dots represents the points of the grid that were simulated.As we move from left to right, we can see clear boundaries where the storage element couldn't survive the droughts.

higher energy consumption results in intermittent behavior and requires additional harvesting capabilities to operate.

5.5.1 Calculating scores for the EH dataset

To calculate the EH score from the long term energy harvesting dataset, we randomly select a 30-day sample of each available dataset location and we perform the steps described in Section 5.4.4 to obtain the EH score from measured generated power traces. We find EH scores of 64.3, 70, 99.7, 98.1, 92.7, and zero for the respective locations pos06, pos13, pos14, pos16, pos17, and pos18. We find that locations pos14, pos16, and pos17 have the best harvesting conditions.

To calculate EH score in a wider range spaces, we investigate if lux measurements are a good proxy for indoor solar energy generation traces. We take as reference the location pos06 of the long term indoor solar energy harvesting dataset [116], and we calculate a simple second order polynomial fit as a model to estimate from average lux data what would be the energy generation during a time step of one minute. Our model coefficients are (2.66e-08 ,2.44e-05, -4.71e-05), and the mean absolute error of the model is 44 μJ. We find that the model is adequate for locations ”pos13” and ”pos14”, in addition to location ”pos06”, with EH score estimation error lower than 1.

5.5.2 Calculating scores for a smart building using lux data

We use the lux to power model obtained previously to estimate the EH score for other real-world indoor locations in an office space. We obtain the lux dataset for this space from commercially available Enocean [46] and Awair [23] smart building sensors. Figure 5.10 depicts the office space floor plan and the calculated EH score values for each space.

5.5.3 EH IoT controlled experiments

To estimate the rating of an energy harvesting device, we place it in a light-proof box, with a controllable smart LED as a light source. We expose the device to a constant lux source of 186 lux until the main storage element is completely charged,

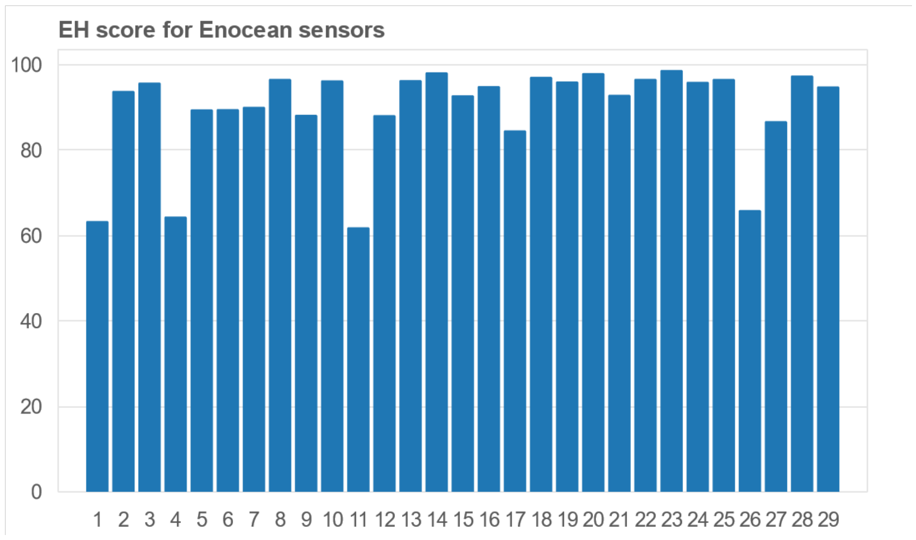


Figure 5.8: EH scores estimated from Enocean devices lux measurements at office space settings.

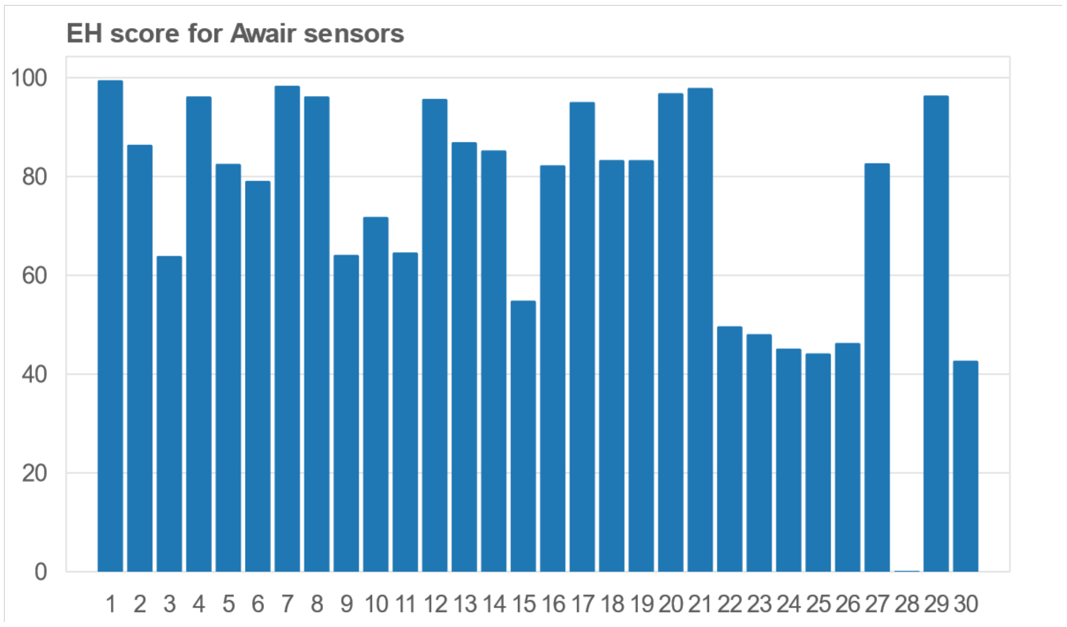


Figure 5.9: EH scores estimated from Awair devices lux measurements at office space settings.

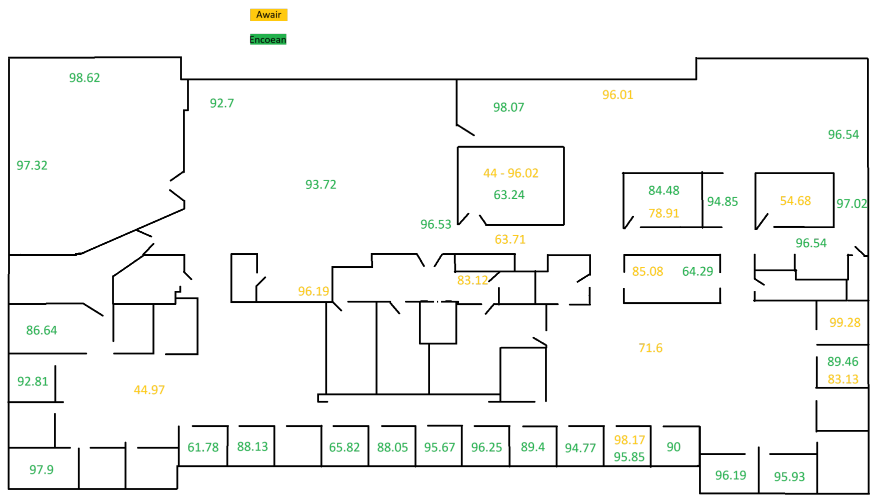


Figure 5.10: EH Scores for an office space. We use historical lux measurements from Enocean and Awair sensors to estimate energy generation.

then we turn the light source off and we record the device operation behavior until shutdown. This experiment allow us to roughly estimate the rating of each device by comparing it to an reference IoT device model with similar capabilities (rating > 80+, 186 lux is enough to support operation). More precise EH rating is possible with extensive testing, or developing precise device operation models, but we considered it out of the scope of this work.

5.5.4 EH IoT deployments

To evaluate the EH score in real-world deployments we select two locations to deploy the energy harvesting devices, one with score 96.2 and the other with score 83.1 (Figure 5.11).

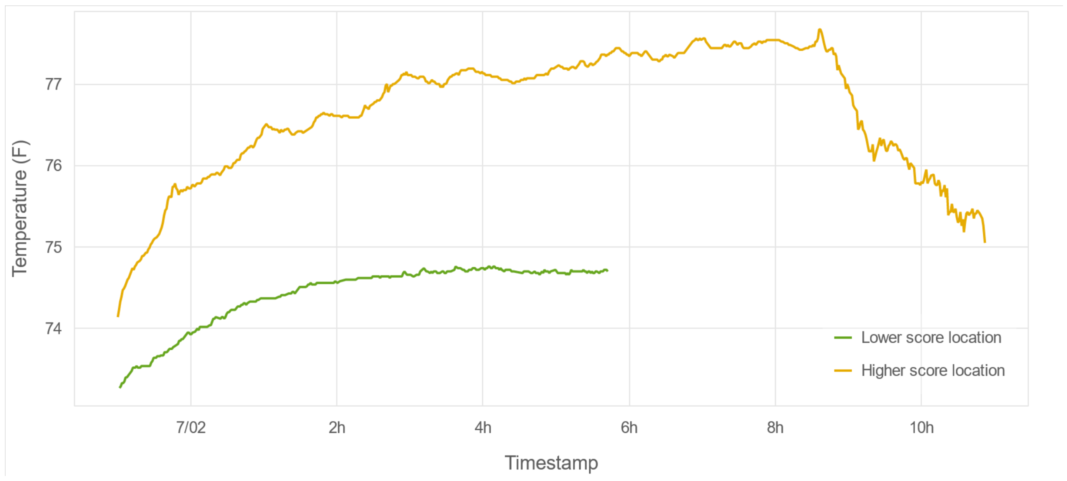


Figure 5.11: Temperature sensor deployment showing intermittency on lower score environment with availability around 55%.

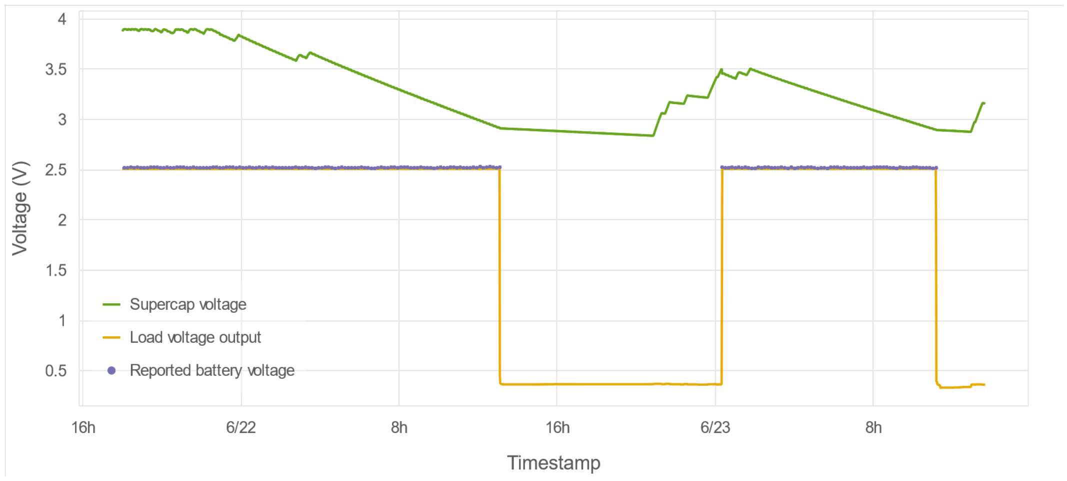


Figure 5.12: Deployment high power consumption, intermittency disrupts operation.

5.6 Limitations

Calculating EH scores and device ratings is challenging in real-world scenarios due to a myriad of factors related with data availability, deployment variability and EH IoT hardware/software complexities.

EH data availability: To understand the viability of deployments of EH IoT devices, we need to collect more EH data. Not only long-term data acquisition is important to understand complex energy sources characteristics over time, but also collecting metadata about factors driving energy sources can be helpful to group and classify harvesting characteristics. An extensive and coordinate effort to collect and share EH data can be very beneficial to accelerate EH IoT technology development and adoption.

Deployment variability: Energy generation can be significantly affected by deployment variability, for instance if the angle of between a solar panel and an energy source changes, or if a shadow is cast on the solar panel. Some of this variability can be expressed as a scaling factor in the energy generation trace, but other might require more complex models. It is important to explore what are real-world factors contributing to deployment variability and investigate their impact in EH IoT deployments.

EH IoT hardware/software and application complexity: Estimating an EH IoT device rating based on its components specification is challenging due to the large design space available when combining available harvesters, power management circuits and energy storage components. Manufacturers are the ones best positioned to provide an energy model model of their systems, but they rarely make detailed information available.For some applications, energy consumption is triggered by hard to predict events, what increases the complexity of device rating. Design decisions on buffering sensor data, re-transmitting information and performing local processing can also make EH IoT modeling more complex.

5.7 Conclusion

The energy harvesting score is a tool for concisely (with just one number from 0-100) communicating both the harvestable energy available in a given environment and the energy requirements of an energy harvesting device. This brings the type of predictable pairing that has been long used in other computing systems (e.g., specifying minimum RAM requirements for software, the required version of a software dependency, or the physical interface plug of a hardware module) to energy harvesting devices. Designing this score requires carefully balancing characterizing available energy with the real-world limitations of embedded devices while also supporting a diverse range of environments on the same 0-100 scale.

We propose this mechanism to help accelerate the deployment of more sustainable energy harvesting devices by quantifying and minimizing the risk of using intermittently powered devices instead of comparatively safe, but hard to maintain and environmentally hazardous, battery powered devices. With this score, device manufactures have a standard mechanism for communicating what conditions their devices require to work as expected. Further, we envision the calculation of the device rating becoming standard fare in future datasheets to communicate exactly how a device operates under intermittent energy. Likewise, future Internet of Things deployments have a tool for quickly assessing how well various options for self powered, batteryless devices will work in their scenarios. This approach helps mitigate the downsides of harvesting uncertainty to help make energy harvesting ubiquitous.

5.8 Contributions and Outcomes

From the best of our knowledge, our energy harvesting score is the first energy harvesting metric to represent the realistic available energy to a SPIoT device, by assuming real-world hardware limitations expressed as energy storage capacity, minimal stored energy to support operation, stored energy leakage and energy conversion efficiencies. We found that we can use the simulated availability of reference SPIoT devices to characterize harvesting conditions, with a higher score being assigned to more consistent and plentiful energy harvesting deployment locations. The energy harvesting score allows SPIoT adopters to match deployment locations with device energy harvesting requirements by selecting a deployment location with a score supported by the SPIoT device rating. To calculate the score, we modify the SPIoT model by Kansal et al. [70] to account for energy storage discharge efficiency and the efficiency of direct power consumption from harvester as to provide a more general framework to describe real-world SPIoT hardware. Compared with the model adopted by Buchli et al. [31], our model takes into account the direct path between energy generation and consumption and energy storage leakage. Our introduced SPIoT model is given by a set of equations representing a discrete-time system, what facilitates its implementation into simulators.

Chapter 6: IoT Cyber Infrastructure for Smart Cities

Recent advances in information and communication technologies (ICT) are enabling internet of things (IoT) smart city projects to collect and analyze vast amounts of data as an effort to support more environmentally and economically sustainable communities [113, 127]. For instance, smart stormwater projects have shown successful IoT based infrastructure monitoring applications to address communities' operation and planning challenges [24, 104, 44]. As IoT devices become more pervasive, collected data is expected to play an increasingly central role to inform communities' decisions and, therefore, it is critical to develop and maintain cyber infrastructure to collect, store and visualize sensor data.

However, as a growing number of new ICT technologies become available, the task of developing and integrating hardware and software solutions for IoT smart city projects can demand extensive specialized knowledge in different ICT domains [126, 67], which can be challenging to IoT system designers. To reduce IoT systems design effort and to make IoT solutions more accessible, The Things Industry (TTI) [66] created and sponsored The Things Network (TTN) [98], a set of open-source tools to provide the basic software infrastructure to deploy IoT sensors based on LoRaWAN [93, 114], a low-power and wide-area network (LPWAN) wireless communication protocol. This open-source project enables contributors around the globe to publicly share TTN compatible gateways that can connect LoRaWAN sensors to a network server known as The Things Stack, which is maintained by TTI. Using TTN for smart city projects has been successfully demonstrated in the literature for different applications (e.g. [24, 43]) while also benefiting communities by creating an open LoRaWAN communication infrastructure that can be leveraged by other IoT projects such as air quality monitoring [26].

Although deploying an IoT system is greatly simplified by using TTN tools, their goal is to provide only the network server infrastructure and leave the application server to be developed by users. For instance, long-term data storage, graphical user interfaces (e.g., plotting tools) and the capacity to send alarm notifications are functionalities not supported by TTN's network server. To achieve such functionalities, users need to develop their own application server or adopt third-party service providers such as Ubidots [130] and myDevices [96]. Another possible solution is to develop a custom server using TTN open-source networking solution and modify it to include application layer functionalities; however, this solution implies an increased server workload and code maintenance requirements when compared to only developing and hosting application layer functions. While third-party application servers might provide great value to many applications, users might still decide to develop their own application server solution to achieve more control over their data, to create customized application solutions, or to reduce recurring costs. However, developing an application server implies selecting, developing and integrating software modules to achieve application's goals, which can be challenging due to the large diversity of architecture options and software solutions currently available as commercial products and open-source modules. In this context, IoT application case studies can offer users a valuable insight into developing and integrating software systems to meet application goals. To help guide users on the path of creating integrated IoT smart city applications, we introduce our use case of a flood warning system for a suburban watershed in Virginia, USA. Our system uses a pressure sensor and two ultrasonic sensors to monitor water levels at three locations on the stream network, and a weather station to monitor precipitation rates. All our monitoring devices use LoRAWAN to communicate to TTN's network server. We developed and integrated a scalable set of cloud-based application tools to perform long-term data storage, data visualization, and automated alarm notification functionality. We discuss implementation challenges and insights for our system, as well as a cost analysis using Amazon Web Services (AWS). To support user's planning and decision-making, we included a cost analysis section where we evaluate how costs currently evolve with time, number of sensors, and data storage requirements.

The main contributions of this end-to-end IoT application development can be summarized as: (1) our work provides practical insights on the development of cloud-based tools for IoT applications, an emerging area that is frequently overlooked on empirical IoT research; (2) we propose a general cloud back-end system architecture that can guide IoT developers to quickly prototype smart city applications by using our demonstrated tools such as serverless data ingestion for IoT historical backup data storage, on-demand MySQL database and Grafana servers, and a RESTful API for programmatic data access; and (3) we perform a cost analysis for the first few years of using AWS cloud services in an IoT application, highlighting the cost-effectiveness of our proposed solution, and providing to IoT developers a cost estimate of these cloud services under varying number of sensors and data rate. In this chapter we discuss the highlighted block of the SPIoT design and deployment framework flow chart in Figure 6.1.

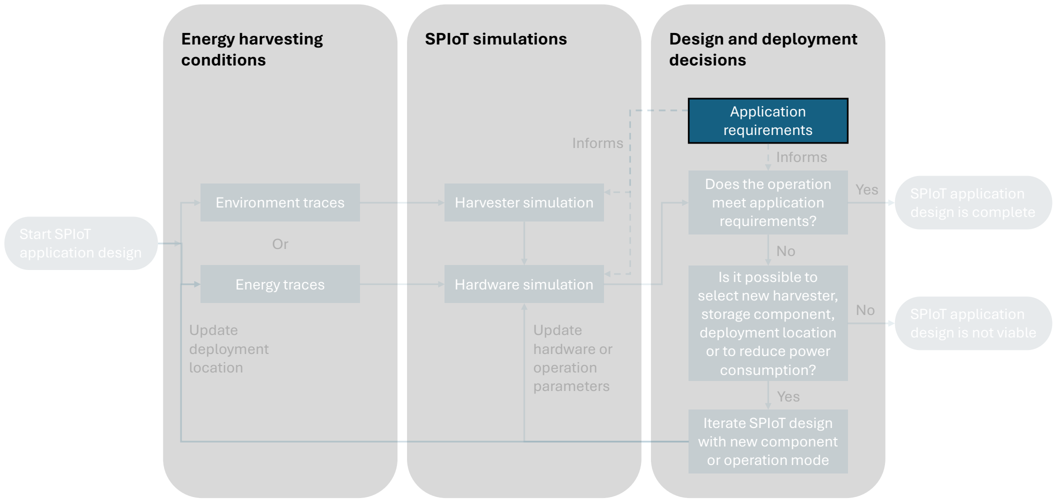

Energy harvesting conditions
SPIoT simulations
Design and deployment decisions
Application requirements
Informs
Informs
Start SPIoT application design
Environment traces
Or
Energy traces
Harvester simulation
Hardware simulation
Does the operation meet application requirements?
Yes
SPIoT application design is complete
No
Is it possible to select new harvester, storage component, deployment location or to reduce power consumption?
No
SPIoT application design is not viable
Yes
Iterate SPIoT design with new component or operation mode
Update deployment location
Update hardware or operation parameters

Figure 6.1: **Flow chart components related with the cloud-based cyber infrastructure.** Designing an end-to-end IoT applications requires making system-level decisions that impact SPIoT hardware component selection, for instance defining a communication protocol such as LoRa.

6.1 Example Application Motivation and Objectives

With the increase of weather variability and flooding [61], it is vital that communities launch flood mitigation initiatives for the safety and quality of life of their residents. To create a sensing and alert system, we need to collect real time sensor data from various locations around a city, parse, store, provide responsive visualization, and transmit alert messages. For preemptive flood management strategies, we also need to collect data about existing infrastructure and land features to model storm-water flow and forecast future flood conditions.

This example application's main goal was to demonstrate cloud-based application solutions to support monitoring and alerting of flooding events. Basic features of our application system include data collection, storage, visualization and alert creation as well as a RESTful API to provide data access to data-driven environmental forecasting, and physics-based stormwater flow simulation. Although this use case is focused on flood warning, we describe each component and lessons learned in a general way, so it can be easily translated to other smart city use cases.

6.2 Methodology

6.2.1 System Architecture Overview

Data flows from sensors to our cloud-based software solution as depicted in the system architecture diagram in Figure 6.2. We built our cloud-based system using Amazon Web Services (AWS) to take advantage of their latest resources and capabilities such as serverless functions (AWS Lambda [17]), data storage (Amazon S3 [9]), API gateway interfaces (Amazon API Gateway [6]), and computing instances (Amazon EC2 [16]).

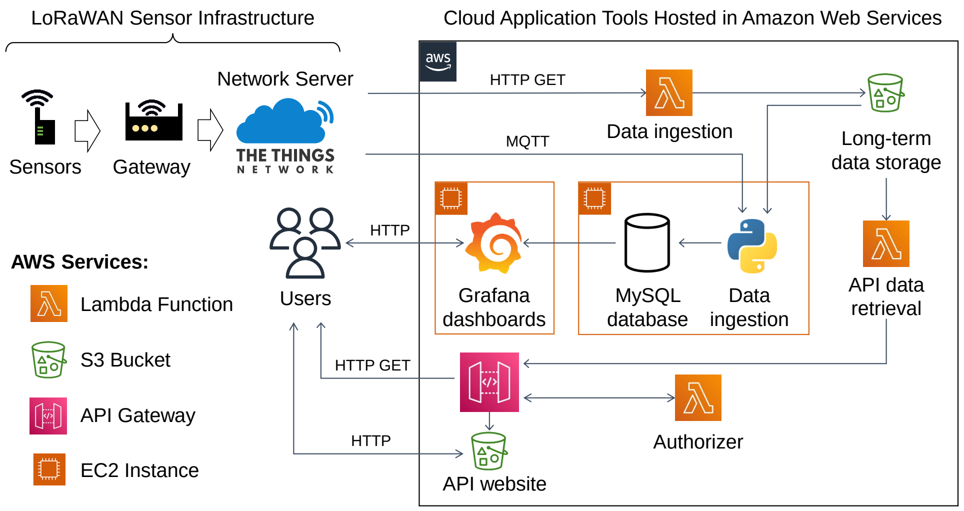


Figure 6.2: **Our system architecture diagram using Amazon Web Services and The Things Network.**

Using AWS Lambda [17], sensor measurements are queried from our TTN application, transformed, and uploaded as csv files to our long-term cloud data storage solution in an AWS S3 bucket [9]. We adopted a MySQL database server to provide responsive data access to our application. The MySQL database is hosted in an AWS EC2 instance [16] alongside a python script that ingests historical sensor data when the virtual machine starts up, and another script that connects to our application's TTN MQTT broker to receive and ingest real time sensor data. The data is then queried for visualization, monitoring, and alerts through a graphical user interface

(GUI) tool, Grafana [74]. Both MySQL and Grafana EC2 instances are only started under demand if users need fast access to structured data or a monitoring dashboard respectively. Sensor data can also be programmatically downloaded using our RESTful API, as for instance in scripts to perform data analysis tasks in Jupyter notebooks [68], or to perform modeling tasks with Storm Water Management Model (SWMM) software [132]. We also hosted a static website to document the API interface and offer users direct access to data download using Swagger UI [125]. While not explicitly shown, we assume simulation and modeling tasks will be performed by users in their own servers that could either be hosted by EC2 instances in AWS, by other cloud providers, or also hosted on their own computer machines.

6.2.2 Design Requirements

Our cloud-based system requires several different components to work in conjunction to meet application requirements. First, the deployed IoT sensors must successfully relay messages to the TTN network server to deliver real time data. The data must then be received, processed and stored in our MySQL database and the S3 long-term data storage for backup purposes. To make the system simpler to develop and manage, we adopted a single cloud service provider to develop our application's services and tools. In this system's case, it was hosted by provisioning services through Amazon Web Services (AWS). Next, for this system to be sustainable and meet different users' cost constraints, it must operate at minimum cost and have efficient resource consumption. The system must also be intuitive and straightforward to deploy, use, maintain, and modify.

6.2.3 System Components

Sensors, TTN and Ingestion to Cloud Platform

As proof of concept, we deployed three water level monitoring sensors and one weather station in a flood prone watershed in Charlottesville, VA. All four devices were connected to The Things Network (TTN) through a LoRaWAN gateway installed in the same neighborhood region as the devices. We utilized commercial sensors from Decentlab [52] to focus efforts on data gathering, storage, and analysis systems rather than sensor's hardware and software. Another motivation behind this decision was to make our solution more general and easily translatable to other smart city projects based on sensor hardware compatible with The Things Network (TTN). We have left sensor deployment details out of the scope of this work, since our main goal is to advance the software back-end infrastructure of IoT systems.

Sensors communicate using LoRaWAN [93, 114] with TTN compatible gateways that interface with TTN network server through an internet connection. The sensors were connected to TTN to enable cost effective interfacing and management, and to utilize the platform's available single-day storage via TTN's data storage integration service. To query data from TTN and upload it to the Amazon Web Service (AWS) stack, we wrote a python function to perform a HTTP Get request to retrieve data for a particular application. This data querying python function runs as an Amazon Lambda service that is periodically executed, set initially to run in one-hour intervals. To ingest real-time data to our MySQL database, we used MQTT clients connected to our TTN applications' MQTT brokers. TTN network server MQTT broker publishes new sensor data to our MQTT clients as soon as it is available in their server, providing our application with timely access to information.

Cloud Platform and Used Services

We decided to develop our application using AWS tools, but the same application architecture can be reproduced using equivalent services from other cloud providers. For regions impacted by flooding, high availability of the computing backend is imperative due to the need for quick analysis of incoming weather and real time water level data. AWS offers high availability, which includes regional failovers in case a data center is taken offline. Deploying and redeploying resources on AWS can also be quickly automated using AWS CloudFormation [13], a tool used to provision specified resources (such as Lambda, EC2, RDS, etc.) through a provided script. The code written for the backend of the cloud-based system can be found at [133].

Amazon Lambda. AWS provides a serverless computing platform known as Amazon Lambda [17], which allow users to run their custom functions on demand. The underlying infrastructure of Lambda is maintained by AWS, which means the system developer must only worry about choosing the correct runtime environment to deploy their code. Using Lambda, the sensors are queried for uplink data at specified intervals. The uplink is then parsed, and the data is transformed to only include information pertinent to the application. The sensors' uplink data is uploaded to S3 for long-term storage and becomes available to be queried into the MySQL database when needed. After the Lambda function finishes uploading the transformed data, it automatically shuts off, allowing the user to pay only for the computing time and memory resources used rather than provisioning a continuously running machine (e.g. EC2). Lambda was chosen for our solution due to ease of scalability with future added devices, monitoring, high availability, and resource efficiency. For instance, if a new TTN application is added to the system, the existing Lambda function can be promptly updated to query sensor data. Should multiple applications need to report data in overlapping intervals, the same Lambda function can run in parallel of up to 1,000 instances if needed.

The Lambda functions for this use case requires modification from the default settings. We used the AWS SDK Pandas Lambda Layer [14] to query from TTN, parse data and to store or read data from a S3 bucket. Python's Pandas module is used to quickly transform and manipulate data. The urllib3 module is used to send HTTP requests to The Things Network storage integration and retrieve sensor data. Other configurations for the Lambda function include setting the allocated memory to 192 MB (determined by AWS Compute Optimizer [8]), timeout limit of 1 minute, and being triggered to run once every hour. The lambda function triggering period can be adjusted based on application needs, where shorter periods translate to lower latency between data being available on TTN and stored in the S3 bucket but also resulting in higher costs for the lambda function computing service.

Another use we make of AWS Lambda is to return stored sensor data requested by our RESTful API and manage user authentication. When receiving a query from Amazon Gateway API, a lambda function is initially executed to check an authorization token provided in the API request and authorize or deny the API request. If authorization is granted, a second lambda function reads, and parse data stored in the long-term data storage solution in the S3 bucket to return the required data to the API gateway. This lambda function to query data from S3 and return to the API gateway is configured to allocate 512 MB of data as a compromise between cost and performance to serve the API functionality and timeout limit of 1 minute. The authorizer function uses default settings of 128 MB memory allocation and 3 second timeout due to the simplicity of our currently adopted solution that only checks if the authorization key input matches a hard coded string value.

Amazon S3 Data Storage. Amazon Simple Storage Solution (S3) is a cost-effective way to store data for ex-tended periods. Data collected by sensors are uploaded in S3 for long-term storage as a read-only resource of the raw data feed. These readings can be used to repopulate the database in case of a database failure or migration and can be done using the python library created for this system. AWS also maintains a python module (BOTO3 [18]) that allows users to download a copy of the readings from S3 to a local machine. All readings in S3 are currently stored as the AWS Standard tier for regular access for this application example.

Another use for the S3 storage is hosting static websites. We used a S3 bucket to store our RESTful API documentation using the Swagger UI interface [125]. Our website is based on the Swagger UI demonstration provided in their github page, adapted to read an OpenAPI 3.0 description of our API service. The static website contains the API server address, a description of the required header, all accepted parameters and the possibility to perform an API GET request trial with parameters provided by the user.

Amazon Elastic Cloud Compute. To host MySQL and Grafana, two Amazon Elastic Compute Cloud (Amazon EC2) instances were provisioned. Amazon EC2 allows for a continuous computing platform on the cloud, which allows access to the database and Grafana when needed. The developed system uses t3.micro instances with 10 GB storage, which fits the needs of this example application by minimizing costs while still maintaining reliable performance for the relatively low number of sensors currently in the system. A more capable instance could be used to serve a larger number of users or for a use cases requiring quicker response times. For this study, MySQL and Grafana were hosted on two separate EC2 instances for simpler management and increased flexibility, allowing for example easy replacement of visualization software or on-demand use of MySQL database to allow fast data access to applications. It may also be worthwhile to adopt AWS Relational Database Service (RDS) [11] instead of an EC2 instance running MySQL as the system database solution and then scale the RDS database based on the application's requirements for maintainability and access speed. This was considered, but not implemented in this study because RDS comes at a higher cost. However, RDS has the advantage that it provides built-in scalability as data volumes and users grows. The Results section includes a cost comparison between these alternatives for hosting the database and a discussion of pros and cons of each alternative.

Relational Database Design and Implementation

As our relational database, we selected MySQL as a simple solution with wide community support. We deployed MySQL on the cloud through Bitnami [28], which provides a preconfigured virtual machine image which is ready to be loaded to an Amazon EC2 instance. We created an entity relationship diagram (ERD) to normalize the sensor readings as shown in Figure 6.3. The ERD is centered around the Measurements entity, which stores the value of individual data points along with the time of data collection (Received_at). The Devices entity stores the device's unique identifier (Device_ID), device's model (Device_model), the last received battery reading of the device (Last_battery) and the last activity timestamp (Last_activity). Similarly, the Locations entity contains data on the latitude, longitude and altitude for each location that data is collected from, along with a unique identifier for each location. For each value in the values table, the Variables entity stores the data points' unique display name and the unit of the variable. The Measurements entity has a one-to-many relationship with the three other entities, meaning that each value data point can only have one device, variable, and location, while the remaining entities can have many values for each data point in their tables. This ERD was developed by advancing an approach by previous related research [33]. This design of the database allows for easy further advancement and change as additional devices and variables can be more easily incorporated.

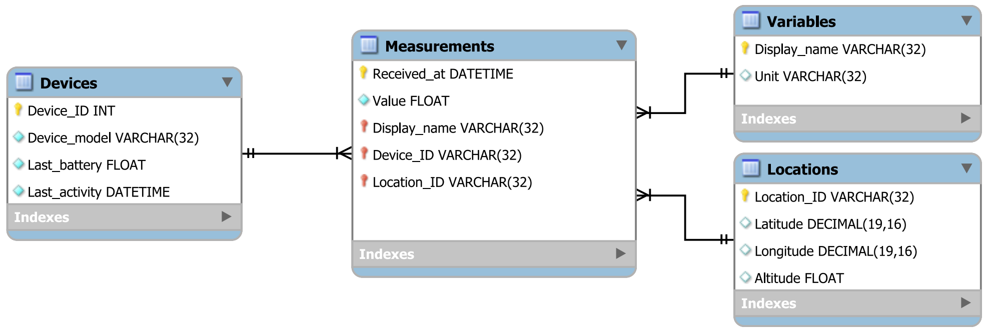


Figure 6.3: **Entity relationship diagram for database design.**

Graphical User Interface

This system allows users to visualize and monitor data through Grafana, an open-source analytics platform for querying, visualizing, and alerting on data metrics. Grafana was selected as the software solution to visualize incoming data due to its dynamic dashboards, built-in alerting capabilities, and its specialization in time series data. Grafana was deployed on the cloud through Bitnami [28], which provides a system image of a pre-configured Grafana stack on AWS. A connection was then made to the MySQL database in Grafana to access the data for visualization. Dashboards of each monitoring station were created to display relevant information for users. In Figure 6.4 we show an example of the dashboard for the water depth monitoring station. This dashboard includes a graph of the water depth over time, statistics on the water depth values for the set time range, a water level gauge of the current depth, a map of the sensor station location, and a gauge of the sensor battery level. The water depth graph and gauge allow users to view the current and past water levels in relation to a threshold of 0.4m to signify flooding. Grafana's built-in alert system can send alert notifications if the incoming data triggers a set alert rule. As an example, the water depth dashboard has alert rules set to send a notification through the messaging application Slack [120] if the 0.4m threshold is met, although sending alerts to other systems or via email is also possible.

RESTful API

Our RESTful API serves as a programmatic interface for users to quickly download data from sensors. We created the API using Amazon API Gateway service [6] and lambda functions both to manage API access and to read, parse and return data

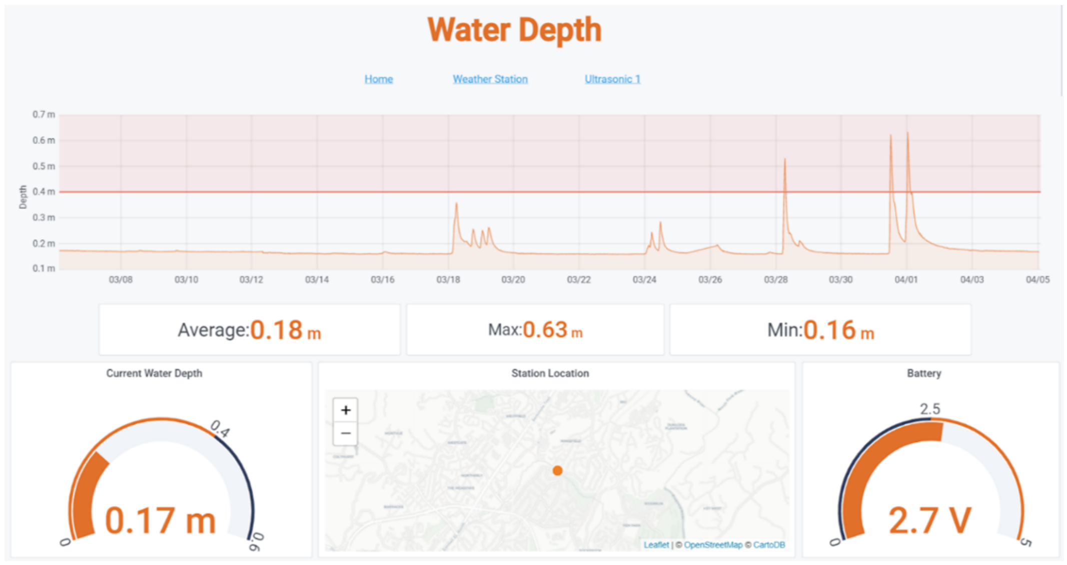
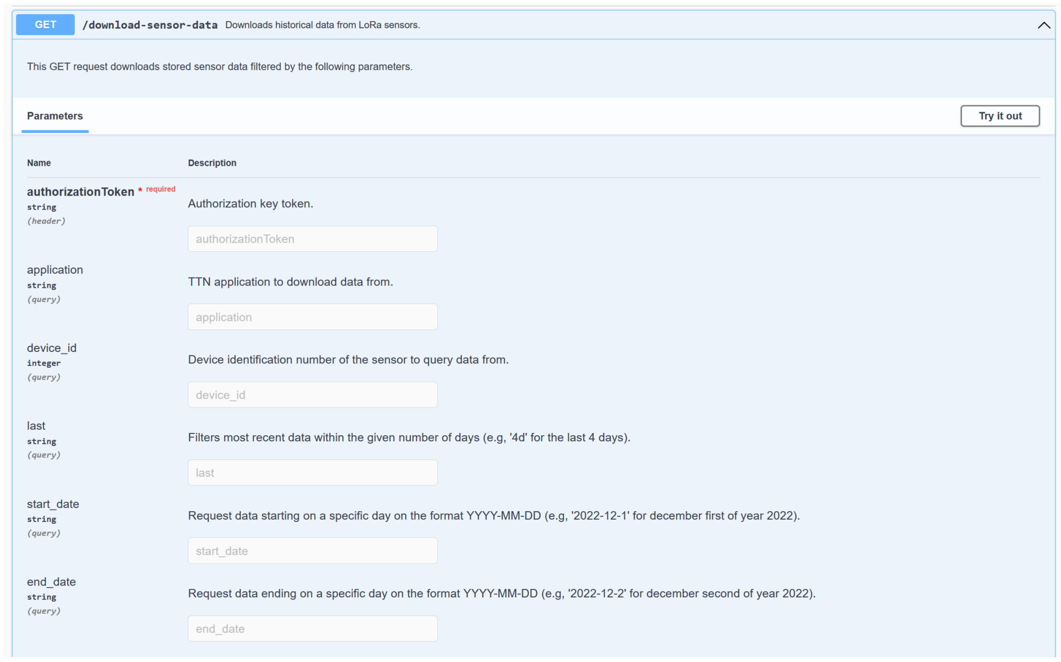


Figure 6.4: **Grafana decision support dashboard of a water depth monitoring sensor.**

from our long-term data storage solution in AWS S3. To document our RESTful API and provide easy access to sensor data, we created a static website using Swagger UI and it is currently hosted using AWS S3 buckets. We also enabled CORS in our API Gateway service, and we added a custom header with an authorization token for access control. We described our API following the Open API 3.0 framework and stored it as a json file loaded to a specification variable in the javascript code for our documentation website. To download data using the API, the user will be required to input a valid authorization token to be granted access. Although we are currently using a simple custom lambda function to grant access, other more comprehensive user access management tool can be used in future versions, such as Amazon Cognito [10]. Other available parameters to customize the sensor data request are: "application", which selects which TTN application do download data from; "device_id" which selects de-vices using a unique identifier; and "last" or "start_date" and "end_date" which al-lows selection of periods of time to download data. Using the "last" parameter, users can retrieve data collected by the sensors from the time of querying to the day specified. Using the "start_date" parameter, users can specify the beginning of the time range of the dataset to download. By default, if only one of "start_date" or "end_date" parameters are provided, data from the single specified day will be returned. Using the API, the user can request datasets for any of the available sensors. In Figure 6.5, we illustrate a typical use of the API to request data from a pressure sensor by using the Swagger graphical user interface. In Figure 6.6 we show a typical API call with parameters and the response.



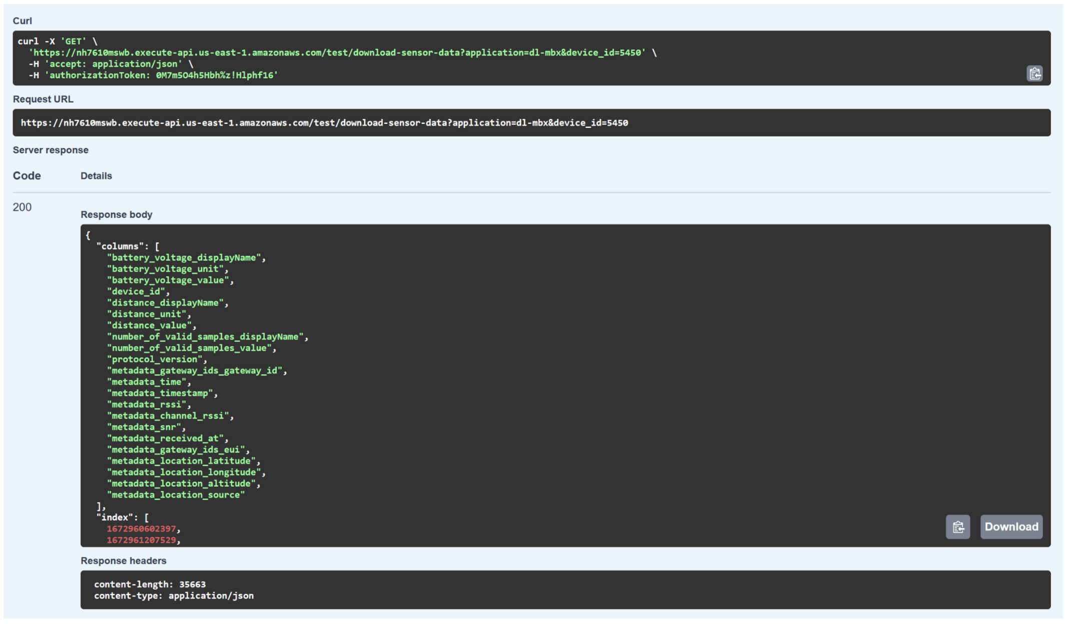
GET `/download-sensor-data` Downloads historical data from LoRa sensors.

This GET request downloads stored sensor data filtered by the following parameters.

Parameters | Try it out

Name	Description
authorizationToken * required string *(header)*	Authorization key token. authorizationToken
application string *(query)*	TTN application to download data from. application
device_id integer *(query)*	Device identification number of the sensor to query data from. device_id
last string *(query)*	Filters most recent data within the given number of days (e.g, '4d' for the last 4 days). last
start_date string *(query)*	Request data starting on a specific day on the format YYYY-MM-DD (e.g, '2022-12-1' for december first of year 2022). start_date
end_date string *(query)*	Request data ending on a specific day on the format YYYY-MM-DD (e.g, '2022-12-2' for december second of year 2022). end_date

Figure 6.5: **Example of parameters for the sensor data download API.**



Curl

```
curl -X 'GET' \
  'https://nh7610mswb.execute-api.us-east-1.amazonaws.com/test/download-sensor-data?application=dl-mbx&device_id=5450' \
  -H 'accept: application/json' \
  -H 'authorizationToken: 0M7m5O4h5Hbh%z!Hlphf16'
```

Request URL

```
https://nh7610mswb.execute-api.us-east-1.amazonaws.com/test/download-sensor-data?application=dl-mbx&device_id=5450
```

Server response

Code | Details

200

Response body

```
{
  "columns": [
    "battery_voltage_displayName",
    "battery_voltage_unit",
    "battery_voltage_value",
    "device_id",
    "distance_displayName",
    "distance_unit",
    "distance_value",
    "number_of_valid_samples_displayName",
    "number_of_valid_samples_value",
    "protocol_version",
    "metadata_gateway_ids_gateway_id",
    "metadata_time",
    "metadata_timestamp",
    "metadata_rssi",
    "metadata_channel_rssi",
    "metadata_snr",
    "metadata_received_at",
    "metadata_gateway_ids_eui",
    "metadata_location_latitude",
    "metadata_location_longitude",
    "metadata_location_altitude",
    "metadata_location_source"
  ],
  "index": [
    1672960602397,
    1672961207529,
```

Download

Response headers

```
content-length: 35663
content-type: application/json
```

Figure 6.6: **Response from API using example parameters.**

6.3 Results and Discussion

6.3.1 Discussion of Alternative System Components and Potential System Enhancements

Cost Analysis of Cloud Services

The first two versions of the databases created for this application example were hosted in a MySQL database using Amazon Aurora [12] and then Amazon RDS [11]. For the application example needs, Aurora and RDS costs presented a constraint, which is the reason we chose two EC2 instances to host MySQL and Grafana that meets user requirements at a lower average cost. The current virtual machine cyber infra-structure costs between \$24 and \$210/year, depending on how long the EC2 instances will be required to be available. However, the database hosted this way may require maintenance such as updating software, or service to fix bugs, along with providing no regional failover. In the event an AWS region experiences an outage, regional failover allows a copy of the database hosted in a separate region to quickly take over operations. Since in our use case we might not need Grafana and the MySQL database to be always available, the EC2 instances can be shut down and only started under demand, for example when users expect an incoming storm. Turning off the EC2 instances reduces the recurring costs to only the instance's storage units, which costs around \$12/year for each instance using currently 10GB of memory space or around \$24/year for both EC2 instances. Should the application require seamless regional failover and high database performance, one alternative solution is to provision two redundant in-stances running Amazon RDS for MySQL with multi availability zone support. This configuration estimated costs are \$623.28/year, considering on-demand instance base costs and 10 GB of SSD storage. Memory storage calculations and associated costs with S3 and the database configurations are provided in the tables (2-7). Our calculations for tables (2-7) were done based on the current sensor device configuration of the system (Figure 6.7), pricing rates at the development time (January 2023) and a projected 5-year use. The default measurement frequency for the system is 1 measurement every 10 minutes, averaging 4380 readings per month. To account for temporary measurement frequency increases during storm events, calculations instead used a figure of 4800 readings per month. One csv file is uploaded every hour to S3 for each registered TTN application, with each write request to S3 costing \$0.000005. Sensor devices currently in use are 1 eleven parameter weather station (DL-ATM41), 1 pressure/liquid level and temperature sensor (DL-PR26), and 2 ultrasonic distance/level sensors (DL-MBX), with one TTN application for each sensor model type, resulting in a total of 3 TTN applications. The average payload size for these 4 sensors is 343 bytes after parsing and transforming and the csv file header average size is 822 bytes. Since the weather station contains more measurements per reading than the other 2 sensor type, its sampling frequency has the most significant impact in the used data storage space. It is important to note that, when data is stored in the MySQL database, the weather station requires almost 5 times as much storage capacity as either of the other 2 sensor devices. Since the current system is based on these 4 sensor devices, AWS storage configurations may need to be readjusted based on the chosen sensors for the application's system.

Device Type	Model	Measured Variables	Readings/Month
Atmospheric	DL-ATM-41	18	4800
Pressure	DL-PR-26	3	4800
Ultrasonic (unit 1)	DL-MBX	3	4800
Ultrasonic (unit 2)	DL-MBX	3	4800

Figure 6.7: **Adopted Sensors for the Proof-of-Concept IoT System.**

Device	1 Month	1 Year	5 Years
Atmospheric	7.13	85.58	427.92
Pressure	1.50	18.02	90.098
Ultrasonic	1.50	18.02	90.09
Current Config (CC)	11.63	139.64	698.19
Average (CC)	2.91	34.91	174.54

Figure 6.8: **MySQL database storage (MB) requirements over time per device type (4800 readings/month).**

Overall yearly system costs can also be lowered by configuring S3 and EC2 instance provisioning and by using built-in AWS cost optimization tools. For S3, if the backup data will not be frequently accessed, it is recommended to change the access tiers of the data. For this application example, the data is stored under Standard tier, which costs \$0.023 per GB. In future iterations of the system, it is recommended to use Intelligent-Tiering: Standard-Infrequent Access (\$0.0125 per GB), One Zone-Infrequent Access (\$0.01 per GB), or even Glacier tiers (\$0.004 per GB). For the Infrequently Ac-cessed and Glacier tiers, there is a retrieval fee for every gigabyte retrieved. Infrequently Accessed will allow for millisecond latency to the user when requesting data, whereas with Glacier it can take minutes or hours. Deleting data from non-standard S3 tiers before their minimum storage duration will charge the user

Number of Devices		Q1	Q2	Q3	Q4	Total
1 [1]	Storage (GB)	0.006	0.012	0.018	0.025	-
	Cost (USD)	0.03	0.03	0.03	0.03	0.12
4	Storage (GB)	0.023	0.046	0.070	0.093	-
	Cost (USD)	0.03	0.03	0.04	0.04	0.14
50	Storage (GB)	0.23	0.47	0.70	0.93	-
	Cost (USD)	0.04	0.06	0.07	0.09	0.26
100	Storage (GB)	0.46	0.47	0.70	0.93	-
	Cost (USD)	0.05	0.08	0.11	0.14	0.38

[1] Only one TTN application was considered for this case.

Figure 6.9: **S3 storage costs calculations for generic sensor devices in the first year (343 Bytes/ sensor payload, 828 Bytes/csv header, 4800 sensor payloads/month and 3 TTN applications).**

Number of Devices	1 Month	1 Year	5 Years
1	0.01	0.05	0.27
5	0.02	0.27	1.37
25	0.11	1.37	6.87
50	0.23	2.75	13.73
100	0.46	5.49	27.47

Figure 6.10: **MySQL database storage (GB) requirements over time based on number of generic IoT devices (1kB/reading and 4800 readings/month).**

Number of Devices	Storage Requirement (GB)	Cost/Month (USD)	Cost/Year (USD)
1	5	8.09	97.10
4	5	8.09	97.10
25	10	8.59	103.10
50	15	9.09	109.10
100	30	10.59	127.10
Every 5 new devices	+2	+0.20	+2.40

Figure 6.11: **Database cost on single EC2 instance (t3.micro) assuming a single instance, storage requirements for 5 years, and generic IoT devices (1kB/reading and 4800 readings/month).**

Number of Devices	Storage Requirement (GB)	Cost/Month (USD)	Cost/Year (USD)
1	5	51.94	623.28
4	5	51.94	623.28
25	10	54.24	650.88
50	15	56.54	678.48
100	30	63.44	761.28
Every 5 devices	+2	+0.40	+2.40

Figure 6.12: **Database cost on 2 separate RDS EC2 instance (db.t3.micro) with multi availability zone deployment and assuming generic IoT devices (1kB/reading and 4800 readings/month).**

Number of Devices	Storage Requirement (GB)	Cost/Month (USD)	Cost/Year (USD)
1	5	61.39	736.68
4	5	61.39	736.68
25	10	62.39	748.68
50	15	64.44	773.28
100	30	67.44	809.28
Every 5 devices	+2	+0.30	+3.60

[1] For 50 devices and more, we estimated higher IOPS to handle the average measurement writing load. The cost also includes running 2 EC2 instances by default for regional failover.

Figure 6.13: **Database cost on Aurora (t3.small)1 and assuming generic IoT devices (1kB/reading and 4800 readings/month).**

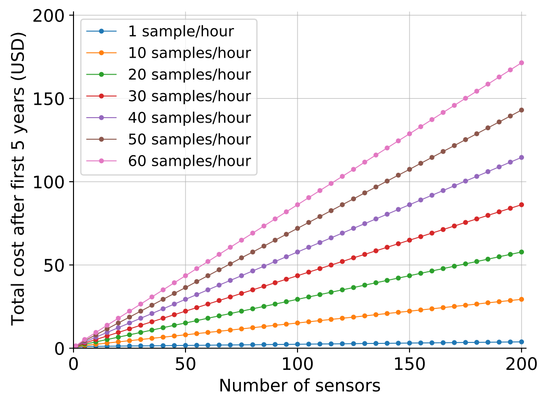
for the respective minimum storage duration. Infrequently Accessed and Glacier tiers also have a minimum capacity charge per object, so it is recommended to combine individual readings into larger datasets (i.e. monthly readings per sensor) to store as one file in these tiers. To reduce costs of EC2 instance provisioning (including for RDS and Aurora), AWS allows for reserving instances in 1 and 3 year increments instead of using on-demand instances, bringing costs down by up to 38%. The costs calculated in this paper are using the current configuration of the system which uses on-demand EC2 instances and considering they will remain always on.

As shown in Figure 6.8, our current configuration of four sensors reporting on average between six and seven samples per hour (4800 samples/month) results in a MySQL database of less than 140 MB of data at the end of the first year of operation. In Figure 6.9, we show that storing this amount of data in AWS S3 service would cost $0.14 for the first year and even scaling to 100 sensors with the same average data rate would result in $0.38 storage costs. This indicates that many small to medium scale applications could benefit from this data storage service to backup sensor data at low costs. In Figure 6.10, we estimate the size of a MySQL database for the first five years, assuming generic sensor samples of 1kB size being uploaded at the rate of 4800 samples/month as we adopted in our example application. The estimated MySQL database size is then used to inform the storage requirement of the virtual machines hosting the respective MySQL databases as shown in Figure 6.11. Our system with 4 sensors would cost about \$97.10/year with each one GB increase in storage space resulting in an additional cost of \$1.20/year. This analysis shows that up time of EC2 servers has the greatest impact on the overall system cost and turning them off while not being required can result in substantial savings. To reduce costs even further, MySQL server disk images can be saved in the S3 data storage service, eliminating EC2 server costs while they are shut down for long periods. As a brief exploration on alternative robust database services offered by AWS, we assume in Figure 6.12 two Amazon RDS EC2 instances with multi availability zone deployment, and, in Figure 6.13, Amazon Aurora managed database on a more powerful EC2 instance. Both solutions result in total costs over \$600/year, representing six times the cost of running a database in a single EC2 MySQL server. Therefore, we recommend using our proposed EC2 MySQL server solution when a failover system is not critical to the application due to the substantial cost savings. In figure 8 we estimate how the cost of S3 data storage varies with sampling rate, operation total duration, number of sensors, and sampling rate. For these calculations we used a simplified estimation model considering only a fee of \$0.023 per GB stored, and \$0.000005 fee of per write request. As in the tables previously introduced, we assume up to 3 TTN applications and one data request and ingestion operation per hour.

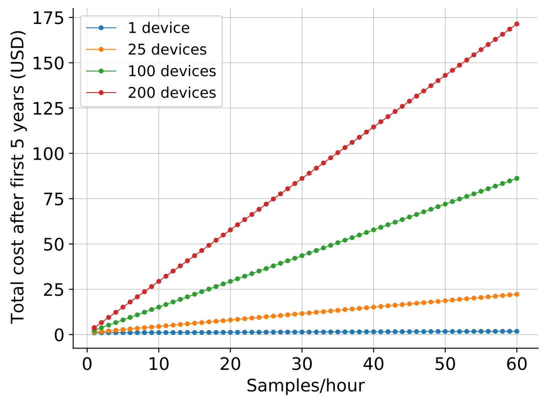
With the S3 storage costs curves depicted in Figure 6.14, IoT application developers can estimate how the number of sensors and data rate parameters influence the total S3 storage costs, as well as how these costs accumulate with time. For instance, in Figure 6.14d, we can verify that the cost of S3 data storage of an application with 50 sensors for the first ten years is comparable to an application with 200 sensors for the first five years.

Discussion About the RESTful API Limitations and Data Access

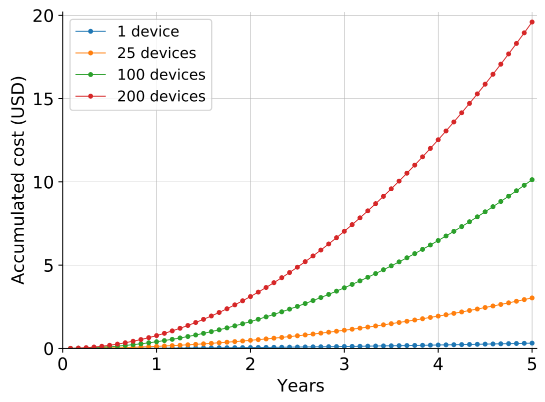
We identified some limitations when testing the sensor data download Application Programming Interface (API). Through an endpoint provided by the Amazon API Gateway, a user request gets passed to the Lambda function to retrieve datasets from the S3 storage, which needs to be parsed before being returned to the user. The first limitation of the solution adopted in this example application is the maximum 30 second timeout on API Gateway requests when large datasets are requested. Even after the retrieval code was optimized to run faster, there was a second limitation through Lambda, which is a payload limit size of 6 MB. For large datasets (e.g. 1 month of data from the weather sensor), the Lambda is not able to send to the user their requested dataset. Therefore, we recommend only using the RESTful API to download data for a few days at each GET request. An alternative and faster solution



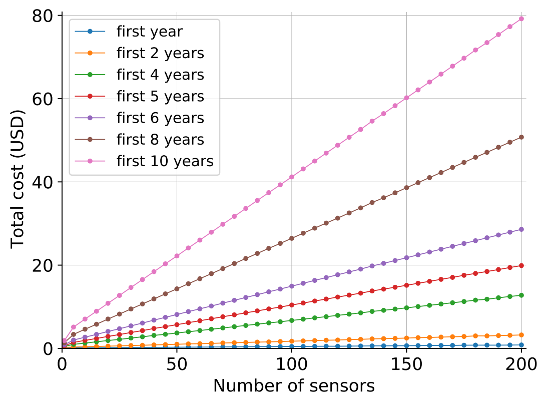
(a) Total S3 cost after 5 years as a function of number of sensors.



(b) Total S3 cost after 5 years as a function of sampling rate.



(c) Total S3 cost evolution with time, considering a fixed sampling rate of 4800 samples per month.



(d) Total S3 cost evolution with number of sensors, considering a fixed sampling rate of 4800 samples per month.

Figure 6.14: **S3 storage costs with varying parameters.**

to download a large amount of data is using the AWS provided BOTO3 python library [18] and downloading the raw csv files directly. We recommend downloading the raw csv files when data is needed in order of a few months of sensor data. Another available alternative solution to perform more responsive data exchange in larger sizes is to query data directly from the MySQL database running in the EC2 instance. We recommend using the MySQL database when data in order of a few weeks is needed for applications such as dynamic websites requiring fast responses or time sensitive simulations.

Security Considerations

Cloud service providers such as AWS acknowledge that security is a major concern for users and provide management tools to support the creation of secure applications. For instance, when deploying a cloud-based system, it is recommended to create an AWS organization with trusted users to manage AWS Identity and Access Management (IAM) roles and policies. Although in hindsight we agree that creating an AWS Organization from the beginning would be best, our research team initially used separate AWS accounts to create and manage Lambda, S3, and EC2 instances based on who was working in each part of the system, resulting in a poor managing practice. Therefore, we recommend access privileges to AWS services to be tailored to developers and systems administrators that oversee each subsystem. We utilized secure shell (SSH) with key pair generated by AWS to access the EC2 instances, using SSH port forwarding to access the Grafana user interface. Although this approach limits the number of EC2 instance ports accessible through the web, it also results in a worse user experience due to the increased number of required steps to access the Grafana dashboards. For future versions of the system, we recommend creating a user access webpage using AWS Cognito service and reverse proxy to serve the Grafana application, without having the need to use SSH tunnels and still avoiding directly exposing ports of the EC2 instance to the web.

Alternatives for Graphical User Interface

Providing users with easily understandable information in a clear and efficient manner is paramount when working with large amounts of time series data. In this application example, were compared three data visualization platforms to find the best tool to effectively communicate information, namely Grafana, AWS QuickSight [7], and AWS SageMaker [15]. QuickSight was initially determined as the platform that best met cost, visualization, analysis, and alerting capabilities requirements. However, after creating a QuickSight account and working with the platform, we found that it does not support embedding visualizations in websites without assigning each user with permissions to view. We then determined that QuickSight was not a suitable tool as it did not meet some of our envisioned uses for the application. After conducting more research on data visualization platforms, we decided that Grafana would be the best tool for this application due to its ability to easily share and embed visualizations. Grafana allows for the creation of snapshots of dashboards which can then be used to share interactive dashboards publicly through snapshot links. Additionally, Grafana is designed for time series data and allows for alerts to be sent out through many alert notifiers such as text message, email, and Slack.

Opportunities for Forecasting and Advanced Analytics

The long-term data gathered by this monitoring system can support the generation of accurate forecasting real time models in the areas of interest. Developing such models with longer observation periods would better assess seasonality effects and, therefore, could reduce the uncertainty arising from precipitation effects, creating more accurate forecasts. Users can feed sensor data from our RESTful API to simulate the generated models and provide real time forecasts on demand. Another potential study that could benefit the creation of forecasting models would be evaluating optimized sampling intervals for each location, as wide variation of water depth between collection intervals can hide patterns and result in less accurate statistical analysis.

6.3.2 Discussion of the System Performance

To analyze the system performance, we can break down the proposed system to a few main data paths, namely: (1) serverless data ingestion, receiving data from TTN and saving to the S3 bucket; (2) MySQL server startup and historical data ingestion from the S3 bucket; (3) MySQL live data ingestion through MQTT; (4) Grafana data query from MySQL; and (5) RESTful API data query. The serverless data ingestion operates independently of the other system com-ponents and its latency is dominated by the lambda function execution time, which takes up to 4.8 seconds. The MySQL server startup includes the EC2 instance boot up, queries to historical data from the S3 bucket, and most recent data from TTN storage integration, leading to a startup latency of up to 10 minutes in the current version of the system. This startup time can be improved, but we assume that the MySQL server can typically be turned on hours before an event of interest (in our example application, triggered by a storm forecast). For the live data ingestion, data is received from TTN through MQTT and a python script ingests data to the database within milliseconds. More in depth study is still required to analyze the impact of high sensor data rates, but EC2 instance computational power can be upgraded to avoid possible bottlenecks. For the Grafana query from the MySQL database, the co-location of EC2 servers in the same availability zone results in overall good performance. Again, more in depth study is required to analyze performance degradation when scaling the number of users logged to the Grafana server. Finally, as previously discussed in the subsection 5.1.2, the

RESTful API has some significant limitations, and its use should be restricted to accessing small batches of data. RESTful API latency can be improved by optimizing the S3 querying lambda function and creating larger S3 objects aggregating a larger number of measurements.

6.3.3 Broader Impacts of this Study

In this work we introduced a cloud-based data storage and visualization tool for smart city IoT projects that can be leveraged by researchers in academia and industry to quickly prototype applications, allowing them to promptly evaluate the impact of their solutions in the real world. The low cost and maintenance requirements of cloud solutions can enable a higher range of experimentation and collaboration between smart city projects, combining IoT data accessibility with computational resources for modeling and simulation. Furthermore, lowering the barrier-to-enter of cloud systems can foster the development of new smart city solutions, supporting more environmentally and economically sustainable communities.

6.4 Conclusion

While data collected by IoT smart city applications are a central asset in supporting management and planning decisions for many communities, designing, and deploying IoT solutions is still challenging due to system integration complexity, reliability limitations and cost. We presented a cloud data storage and visualization system for smart cities, leveraging reliable existing technology to integrate a complete IoT monitoring solution hosted in AWS and costing under \$26/year for long-term data storage and \$0.0204/hour of use for MySQL database and Grafana servers. By using this cloud-based solution together with TTN infrastructure and commercial LoRaWAN sensors, users can collect, store, and visualize datasets to address their needs and integrate their own services. We demonstrated the use of the system for a flood warning system application example with river and weather LoRaWAN sensors. The cloud-based system design uses serverless data ingestion to provide a simple and cost-effective data storage solution that is independent of other services such as data visualization. An on-demand database and visualization servers offer flexibility to adapt to application needs while saving costs and simplifying maintenance operations. Furthermore, we explored the different AWS tiers and their respective reliability/cost trade-off so users can make informed decisions when tailoring our system to their own application. As opposed to focusing mainly on the example application as commonly done in the literature, we highlight common tasks that are required by IoT project and share our insights in leveraging modern cloud services to simplify IoT back-end system design and optimize costs.

6.5 Contributions and Outcomes

From the best of our knowledge, the proposed IoT cyber infrastructure is one of the first public works to demonstrate the use of modern serverless functions and API gateway cloud services to support an end-to-end IoT application, while also performing a cost analysis of the solution and making the code open source. We found that cloud services initially developed for web applications can be used together with The Things Network as building blocks of IoT end-to-end applications at low design and maintenance effort. While existing works introducing end-to-end IoT applications in the literature [92, 24] focus on a dedicated application server, our cloud-based solution relies on cloud services to provide cost-effective resources that can be allocated on demand with the high reliability offered by cloud providers.

Chapter 7: Case Studies

To demonstrate how SPIoT modeling approaches can support applications, we selected two case studies introduced in Chapter 3: water quality monitoring; and water leakage detection. The first case study on water quality monitoring demonstrates how we can use existing datasets and models to evaluate a self-powered water quality station design and assess the potential of a small water turbine energy harvester for this application when compared to solar energy harvesting. The second case study on water leakage detection demonstrates how our collected thermal energy harvesting dataset and our SPIoT models can help to inform the design of a self-powered water leakage detection sensor for use in building automation applications.

7.1 Water Quality Monitoring

As introduced in Chapter 3, exploring different energy harvesting solutions for water quality stations can increase the range of possible deployment locations and allow government regulators and citizen scientists to collect important water quality information on streams. The goal of this case study is to investigate what is the energy harvesting potential of small commercial hydro turbine harvesters when compared to solar panels to support water quality stations. To achieve this goal, we modeled a water quality station based on the Open Storm platform [25], considering typical water quality sensing modalities: turbidity, PH, specific electrical conductivity, temperature, dissolved oxygen and oxidation reduction potential. To simulate energy generation, we used harvester models and available datasets of solar irradiance and water flow velocity. In our published work [85], we also derived a model to represent the average energy loss in solar panels due to shades from canopy cover [85]. We assumed that the application is concerned with the up time of the water quality station and its sampling rate (or sampling interval) metrics. Finally, we evaluated the energy wasted as overflow when the energy storage device is full and no more energy can be stored. This self-powered water quality station analysis uses the highlighted blocks in Figure 7.1.

The overall modeling framework used in this case study is illustrated in Figure 7.2. Streamflow velocity time series collected by the United States Geological

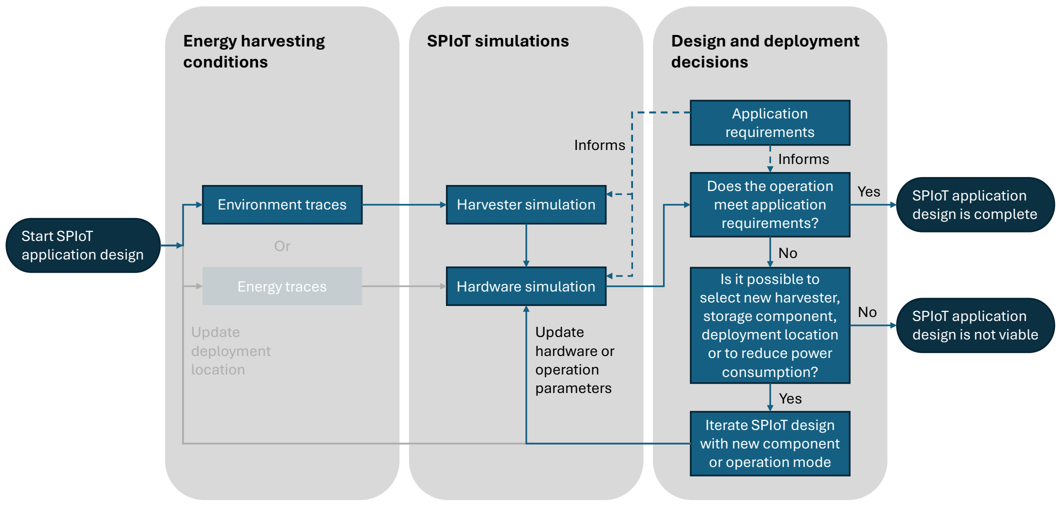

Energy harvesting conditions
SPIoT simulations
Design and deployment decisions
Start SPIoT application design
Environment traces
Or
Energy traces
Update deployment location
Harvester simulation
Hardware simulation
Informs
Update hardware or operation parameters
Application requirements
Informs
Does the operation meet application requirements?
Yes
No
SPIoT application design is complete
Is it possible to select new harvester, storage component, deployment location or to reduce power consumption?
No
Yes
SPIoT application design is not viable
Iterate SPIoT design with new component or operation mode

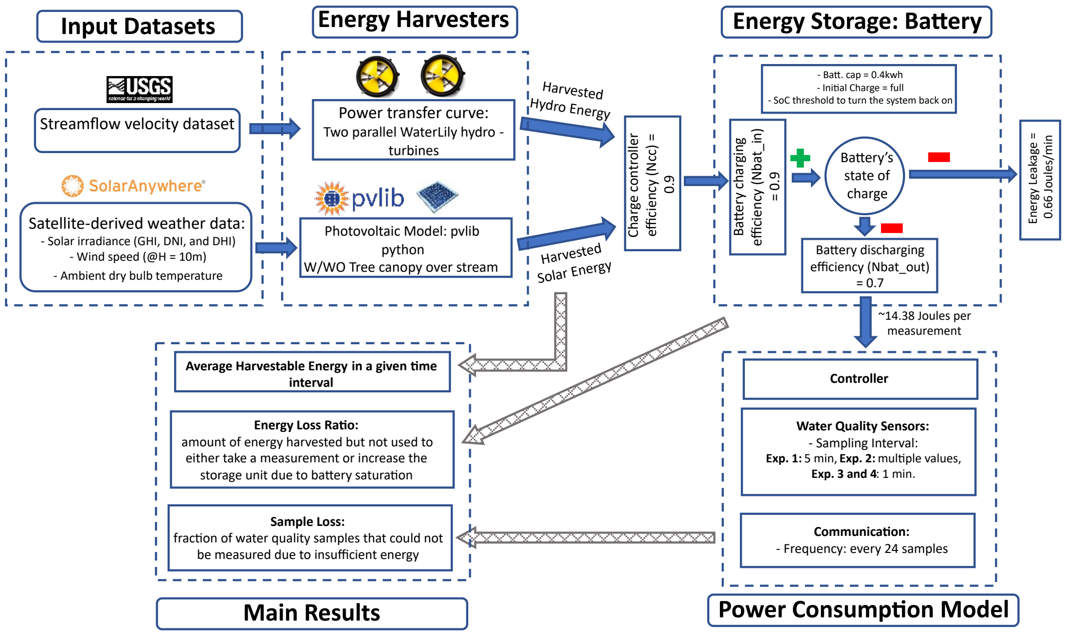
Figure 7.1: **Flow chart components related with the self-powered water quality station design.** We use streamflow velocity and solar radiation parameters as environment traces, then we simulate a SPIoT device. We adjust the sampling interval to evaluate potential SPIoT applications.

Survey (USGS) is used with a hydro turbine model to simulate the energy generation from the water kinect energy, while solar irradiance, wind speed and ambient temperature are used with a photovoltaic panel model to simulate solar energy generation. The water station design follows a harvest-store-use model as in [30], with charge controller and battery charging efficiencies of 90% and battery discharging efficiency of 70%. The battery model was based on a commercially available device with capacity and energy leakage of 0.4kWh and 0.66J/min respectively.

7.1.1 Datasets

Hydro energy harvesting dataset

The United States Geological Survey (USGS) has recorded and shared streamflow velocity for about 400 locations out of 14,481 (active and inactive/discontinued) stream locations across the United States. These 400 sites are located on natural streams or man-made channels [124]. Velocity measurements are recorded using acoustic Doppler velocity meters. This data is often used to estimate the flow discharge in waterbodies [79]. The available velocity time series for each USGS site



Input Datasets
USGS
Streamflow velocity dataset
SolarAnywhere
Satellite-derived weather data:
- Solar irradiance (GHI, DNI, and DHI)
- Wind speed (@H = 10m)
- Ambient dry bulb temperature
Energy Harvesters
Power transfer curve:
Two parallel WaterLily hydro - turbines
pvlib
Photovoltaic Model: pvlib python
W/WO Tree canopy over stream
Harvested Hydro Energy
Harvested Solar Energy
Charge controller efficiency (Ncc) = 0.9
Energy Storage: Battery
- Batt. cap = 0.4kwh
- Initial Charge = full
- SoC threshold to turn the system back on
Battery charging efficiency (Nbat_in) = 0.9
Battery's state of charge
Battery discharging efficiency (Nbat_out) = 0.7
Energy Leakage = 0.66 Joules/min
~14.38 Joules per measurement
Controller
Water Quality Sensors:
- Sampling Interval:
Exp. 1: 5 min, Exp. 2: multiple values, Exp. 3 and 4: 1 min.
Communication:
- Frequency: every 24 samples
Power Consumption Model
Average Harvestable Energy in a given time interval
Energy Loss Ratio:
amount of energy harvested but not used to either take a measurement or increase the storage unit due to battery saturation
Sample Loss:
fraction of water quality samples that could not be measured due to insufficient energy
Main Results

Figure 7.2: **Simulation model diagram for self-powered water quality stations.** We used available solar and streamflow datasets to simulate an energy harvesting water quality station model and assess harvestable energy, wasted energy due to battery overflow and sample loss due to power outages.

is either a reading from the sensor directly (identifiable using the USGS parameter code 72254), mean streamflow velocity over the cross-sectional area of the waterbody (identifiable using the USGS parameter code 72255), or both. Because the streamflow velocity time series is either measured at a point or represents the mean streamflow velocity at a cross section, it does not capture the variation of streamflow velocity in depth, width, and along the river. Additionally, there were a few negative values reported at many of the USGS sites, which can happen due to backwater flow conditions that can occur at stream confluences, streams flowing into lakes or reservoirs, tide-affected streams, regulated stream flows (dams or control structures), strong prevailing winds, or where structures (e.g., bridges and culverts) restrict flow.

The available streamflow velocity record duration varies across the 400 sites from about four months to about 12 years. Most often, streamflow velocity measurements are recorded on a 15-minute interval. We chose to use a five-year study period of 2010 to 2014 because it had the maximum number of sites with recordings during this period. We identified 42 sites with a low missing data fraction during this period. In total, 2.1% of the dataset was missing due to data gaps. The largest gap within the data was 23.47 days and the median gap across all sites was 1.1 hours. Across all sites, 80% of the data gaps were less than 4.2 hours. Because the missing data made up a small fraction of the overall dataset, and because the gaps tended to be small relative to the overall period of analysis, we used a simple linear interpolation as a standard gap filling technique applied consistently across all data gaps. The resulting gap-filled streamflow velocity time-series for the 42 sites were used to estimate the harvestable energy using a flow-velocity to power transfer curve from a small hydro-turbine, which was provided by the manufacturer.

Solar energy harvesting dataset

To estimate the harvestable solar energy, historical satellite-derived estimated weather data provided by Clean Power Research's SolarAnywhere for the continental United States was used [36]. The dataset includes solar irradiance (global horizontal irradiance, direct normal irradiance, and diffuse horizontal irradiance), wind speed at a height of 10 meters, and ambient dry bulb temperature. The spatial resolution of the data is 10 km and the data is available on an hourly time step. The weather dataset was linearly interpolated for every minute to be consistent with the streamflow velocity dataset. The interpolated dataset was used as input for a photovoltaic (PV) model to estimate the harvestable solar energy.

7.1.2 Sensor stations specifications

The water monitoring sensor station simulation model has four main factors: (1) energy from the external environment, (2) one or more harvesters, (3) an energy storage component, and (4) an electrical load from sensors and other electronic components. The energy from the external environment (e.g., water kinetic energy or solar radiation) is collected and converted to a usable energy form (i.e., electrical energy) by the harvester to meet the power demand of the load. The energy storage component is necessary to store energy during periods when the harvested energy is greater than the consumed energy and then provide the stored energy during periods when the energy consumption by the unit exceeds the harvested energy. The external environment was explored in the previous section. Here we provide specifications of the harvesters, energy storage component, and the power consuming unit (load) that were used in the study.

Energy harvesters

To estimate the harvestable hydro power, a small commercial portable hydro-turbine called "WaterLily" was considered [64], one of the very few commercially available options of its kind. Depending on the stream hydrology and channel morphology, customized hydro-turbines to adopt with the conditions can be considered. Figure 7.3 depicts the WaterLily hydro-turbine streamflow velocity to power transfer curve provided by the manufacturer. The minimum streamflow velocity value required to generate energy is 0.5 m/s and the peak power of about 14.3 W is achieved at approximately 3.2 m/s. We assumed the hydro-turbine could harvest energy from both positive and negative flows, thus the negative velocities described earlier were treated the same as positive velocities. Throughout all the hydro simulations, we used a parallel combination of two WaterLily hydro-turbines. Therefore, at a given flow velocity, the harvested power from Figure 7.3 is doubled. Using two hydro-turbines was necessary to provide enough energy for the water monitoring station.

To estimate the harvestable solar power, we used pvlib python, an open-source community supported tool [63]. In this study, we used pvlib python version 0.6.4 [62]. For all the harvesting locations, the simulations assumed a 20 Watt 12 Volt solar panel [73] operating at the maximum power point. The surface azimuth of the solar panel was set to 180 degrees for all energy harvesting sites during the entire simulation period while the solar panel surface tilt was considered to be equal to the latitude of the energy harvesting location. The weather dataset described previously in the section "solar energy harvesting dataset" was used as input to the solar energy harvester system simulations.

Power consumption model for the water quality station

The sensor station power consumption model assumed in this work includes three main modules: controller, sensor, and communication. The whole station was assumed to have two operating modes: active, where all modules are consuming energy,

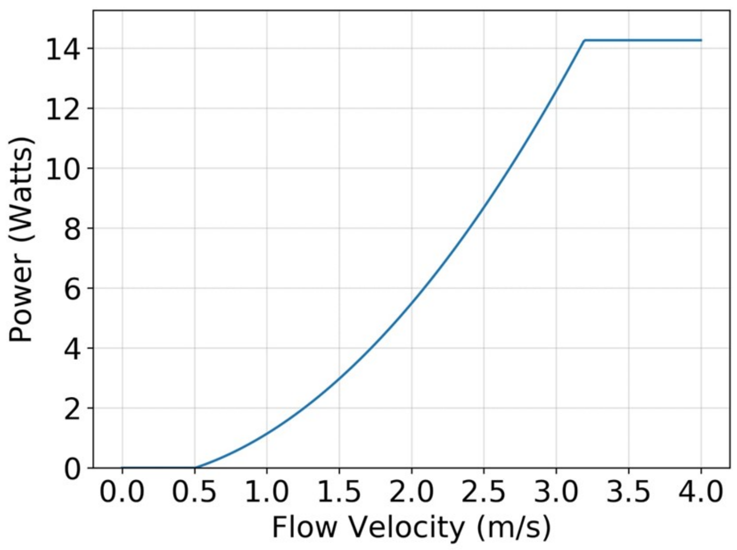


Figure 7.3: **Water Lilly turbine streamflow to power transfer curve.** This curve provided by the manufacturer indicates how much power we can expect to generate at different streamflow velocities.

and idle (or sleep), where all modules are inactive. The sensor station alternates between active and idle modes, acquiring sensor measurements and periodically transferring data over the cellular network in active mode, while saving energy in between measurements by alternating to idle mode. The station's average energy consumption is then used as input to the simulator. For the purposes of the analysis performed in this work, the sensor station hardware platform from the Open Storm project [25] is used as a reference of a possible implementation of such system.

The controller module is a programmable device capable of interfacing and turning sensor and communication modules on and off. For example, the Open Storm project adopts a custom printed circuit board with a Cypress CY8C5888LTI-LP097 Programmable System-on-Chip (SoC). The controller module together with the lowest power consuming sensing modality was reported by the Open Storm project team to consume about 10 mA at 3.7V voltage source on active mode. For the purposes of the analysis performed in this work, the controller power consumption is assumed to be 37mW and it should be viewed as a target power consumption budget that the controller module must adhere to. The idle or sleep mode was reported by the Open Storm project team to consume up to 60µA at 3.7V and this worst case will be considered on the sensor station model.

The sensor modules selected for this analysis and supported by the Open Storm project comprises of a collection of typical water sensors used in a sensor station

(Table 7.1). The active power consumption for each module together with the recommended inline voltage isolator (part number BE-IVI) is provided by the manufacturer [110]. The Open Storm project team reported that it typically performs one measurement every 10 minutes and it takes 5 seconds to perform a measurement.

Table 7.1: Water quality sensors and their respective power consumption at 3.3V power supply. Total energy consumption was calculated based on a 5-second sampling duration on-time.

Sensor	**Part Number**	**Power (W)**	**Energy per meas. (J)**
Temperature	EZO-RTD	0.287	1.435
Dissolved Oxygen	EZO-DO	0.277	1.386
Elec. Conductivity	EZO-EC	0.320	1.600
PH	EZO-PH	0.287	1.435
Oxid. Red. Pot.	EZO-ORP	0.277	1.386
Sum	-	1.448	7.242

One additional parameter typically measured on water quality sensor stations is turbidity. Since Atlas Scientific currently does not commercialize turbidity sensors, we considered the turbidity sensor manufactured by Global Water [137] in this work. The Global Water turbidity sensor WQ730 consumes up to 60mA at 12 V power supply, resulting in a power consumption of 720mW. The WQ730 sensor requires a recommended warm-up time of 8 seconds before start measuring and for the purposes of this analysis, we considered a one-second window to actually acquire the measurement, resulting on a total of 9 seconds of active time and consuming a total of 6.48 Joules per measurement.

The communication module is an electronic device capable of transmitting and receiving information through a cellphone network. The Open Storm project [25] sensor node uses the module Telit CC864-DUAL to perform the communication task and an average of 25 mA current consumption at 3.7 V voltage source was reported by the Open Storm project team. Each communication event was reported to happen every hour and take up to 60 seconds. In this work, the worst case of 60 seconds for a communication event (total energy consumption of 5.55J per event) is assumed, while no energy is consumed while in idle or sleep mode. It is also assumed that the controller will remain active for the whole transmission event of 60 seconds, consuming a total of 2.22J per transmission event. Each communication event is assumed to happen every 24 sampling events, what would result in one communication event every two hours for a five-minute sampling interval. The average energy consumption per measurement spent on the communication task is therefore estimated to be 0.324J per measurement event.

To calculate the total energy consumption per measurement of the complete station using all sensors (All five Atlas Scientific sensors and the Global Water turbidity sensor), we add the energy consumption per measurement of all sensors (13.722J), the energy consumption of the controller device during sensor measurements (0.333J for 9 seconds operation due to the turbidity sensor) and the average energy consumption per measurement of the communications module (0.324J), resulting in a total of approximately 14.38J. The average energy consumption per simulation step (E_{load}) is calculated by Equation (7.1).

$$\overline{E_{load}} = \frac{SimStep * E_{active}}{SI} + SimStep * P_{idle} * (1 - \frac{9}{SI} - \frac{60}{TI}) \tag{7.1}$$

Where $SimStep$ is the simulation step in seconds, SI is the sampling interval in seconds, E_{active} is the total energy consumption per measurement in Joules, P_{idle} is the sensor station power consumption during idle mode in Watts (assumed to be 222 μW) and TI is the transmission interval in seconds, the time between successive communication events. For a simulation step of one minute, a sampling interval of five minutes and transition interval of two hours, (E_{load}) is calculated to be approximately 2.89 Joules per minute.

Since it is usually more than enough to measure the water quality parameters at waterways at sub-hourly intervals, which successfully captures the dynamics of the change in parameters, the default sampling interval for all the water quality sensors was fixed to five minutes unless another sampling interval is noted.

Energy storage unit and charge controller

Power storage is necessary to manage the energy harvesting fluctuations over the deployment period of the energy harvesting system to store energy during high energy harvesting periods and use the stored energy during energy harvesting droughts. Also, on many occasions, the instantaneous harvested power is not enough to directly power the electronics which again emphasizes the need for an energy storage medium.

The battery model used in this analysis considers charging and discharging efficiencies as well as a constant energy leakage parameter as used in [30]. The following parameters were considered for this analysis: maximum capacity of 0.4 kWh; constant leakage of 0.66 Joules per minute (2% self-discharge in one month); charging and discharging efficiencies of 0.9 and 0.7 respectively are assumed. The battery is considered to be at full charge in the beginning of each simulation. One example of a commercial battery with 0.4 kWh capacity is the Victron Energy 12V 34 Ah battery [45].

A charge controller is considered to control battery's charging and discharging operation as assumed in the work done by [30]. The charge controller protects the battery by disconnecting the load when stored energy reaches a minimum level (complete depletion in our case) and only reconnecting it after the battery is recharged to threshold percentage of the nominal capacity (assumed as 30% in this work). The charge controller efficiency is assumed to be 90% as used in [30].

7.1.3 Energy generation, store and consumption model

The simulation model used in this work follows the work presented in [30] by considering a harvest-store-use energy model and calculating the battery state of charge after every simulation step. Equations (7.2) to (7.4) summarize the adopted model.

$$Ebat_{out}[k] = \frac{\overline{E_{load}}}{Nbat_{out}} + E_{leak} \tag{7.2}$$

$$Ebat_{in}[k] = min(Nbat_{in} * N_{CC} * E_h[k], max(0, Nbat_{out} * B_{nom} - B[k-1] - Ebat_{out}[k])) \tag{7.3}$$

$$B[k] = max(0, min(Nbat_{out} * B_{nom}, B[k-1] + Ebat_{in}[k] - Ebat_{out}[k])) \tag{7.4}$$

Where: $Ebat_{out}[k]$ is the resulting energy decrease on the battery's charge ($B[k]$) at the simulation step k by supplying the load (E_{load}) with discharging efficiency $Nbat_{out}$ and energy leakage E_{leak}. $Ebat_{in}[k]$ is the resulting energy increase on the battery's charge due to the harvested energy ($E_h[k]$) with charging efficiency $Nbat_{in}$ and charge controller efficiency N_{cc}. These equations also limit the battery's charge between zero and the battery's maximum capacity ($Nbat_{out} * B_{nom}$), therefore resulting in energy lost as overflow when the battery capacity is not large enough to accommodate the potential incoming charging energy. More details about the adopted model and its parameters can be found in [30].

7.1.4 Energy harvesting simulations

To perform the energy harvesting simulations, we used the pvlib python library for simulating solar energy harvesting from historical hourly satellite-derived estimated weather data described earlier and WaterLily power transfer curve to harvest energy from gap-filled streamflow velocity explained earlier using two hydro-turbines. Then, a self-powered discrete-time water monitoring station was simulated adopting the energy generation, store and consumption model from [30] described in the previous subsection. Integrating all of these, we were able to determine the state of the self-powered sensing system components (e.g., instantaneous harvestable power based on the input energy or the battery charge level) for any given simulation time period for a certain monitoring system with known parameters (e.g., battery capacity or initial charge). Additionally, our simulations provide two main outputs: sample loss and energy loss. Sample loss is the fraction of samples that could not be measured due to insufficient energy. This occurs when the sensor station load exceeds the harvestable energy and available energy from the storage unit. Energy loss is the amount of energy that could be potentially harvested but cannot be used to either take a measurement or increase the storage unit (because it is already full). These two parameters help to understand the two limits of energy for the system: too little energy preventing sampling on the one end, and too much energy exceeding storage limits on the other end.

There are several simplifying assumptions and limitations that are important to understand when translating these simulation results to real-world practice. First, when deploying a hydro-turbine in a river to harvest power for a real-world application, since the particular harvester considered has rotating parts, it is prone to stop functioning when debris becomes lodged in its blade. In this study, we assumed that the hydro-turbine is not affected by this practical reality because we assume there is a way to protect the blade from debris using a protective filter or screen that would not significantly alter the streamflow velocity. Also, the WaterLily hydro-turbine was used throughout all the hydro harvesting simulations as it was one of the very few commercially available small-scale hydro-turbines. Depending on the stream hydrology and channel morphology, customized hydro-turbines other than WaterLily hydro-turbine can be considered to better adopt with local stream and channel conditions at a harvesting site. For the solar energy harvesting simulations, the reduction of solar radiation caused by topography (i.e., hills and valleys.) is not considered in this study; only solar radiation reductions due to tree canopy is considered in the published work version[85]. Ignoring the topography should not introduce significant errors into the solar energy harvesting estimation because the 42 river locations considered in the study are located in regions with low-relief topography. Lastly, for both the hydro and solar energy harvesting, we assumed that a cellphone network coverage is always available at all the locations to transmit and receive information which in reality might be unavailable in some locations or some periods of time.

7.1.5 Implications of model limitations and estimation errors

As it was described earlier in this section, the water quality sensor station model is primarily built upon validated results of a wide set of models ranging from environmental parameters to electronics and sensor operation behavior. Although each model component has its own limitations and estimation errors and the complete system was not prototyped, the overall goal of this modelling effort was to make reasonable assumptions that enables a practical and coarse assessment of energy harvesting sensor stations at a scale that would be otherwise prohibitive in terms of time and financial resources. Furthermore, the current model is expected to be adapted for particular cases of interest with prototype and measurements of both energy generation under relevant environments and energy consumption of specific electronics and sensors. As an example of such an adaptation, for locations with significant topographic relief (e.g., for rivers in steep canyons), the negative effect of topography on solar energy harvesting cannot be ignored. However, the locations studied in this research were all located in low-relief areas, which justifies ignoring the topography when estimating solar energy harvesting.

7.1.6 Results and discussion

Figure 7.4 shows the average harvestable hydro and solar energy in five-minute intervals over the 2010-2014 period at each of the 42 USGS sites. Figure 7.5 presents the same information in map form to aid better understanding of the spatial patterns of energy harvesting for the energy harvesting sites. These results provide insight into how much hydro and solar power can be harvested at each site and how the sites compare to each other in terms of the potential for hydro and solar energy harvesting. The average harvestable hydro energy in five minutes ranged from 0 to 778 Joules with a median over the 42 sites of 19.1 and an average of 87.5 Joules. On the other hand, the average harvestable solar energy in five minutes ranged from 994.3 to 1460.8 Joules with a median of 1285.1 and an average of 1242.6 Joules over the 42 sites. This assumes no reductions in solar energy harvesting due to tree canopy, an assumption that is explored in [85]. Under this assumption, and not surprisingly, solar energy harvesting almost always significantly exceeded the potential for hydro energy harvesting, by an average factor of 14.2x across all stations.

A number of stations had hydro energy harvesting potential comparable with solar energy harvesting potential. A few of these stations with consistently high streamflow velocities are at high latitudes and in regions with significant cloud cover. Less cloud cover is largely responsible for stations in the west recording higher average harvestable solar energy (between 1279 to 1460 Joules per five minutes) compared to those in the central and eastern US (between 994 to 1193 Joules per five minutes). As shown in Figure 7.4 and Figure 7.5, one site located in Michigan (04159130), an area with significant cloud cover, had the greatest average harvestable hydro energy (778 Joules per five-minute intervals) and the greatest average harvestable hydro energy over the average harvestable solar ratio (i.e., 78%) among all the 42 USGS sites. This site is also at a high latitude receiving less solar radiation compared to sites in lower latitudes. The location also has consistently high streamflow velocities throughout the simulation period, always exceeding the minimum streamflow velocity value required to generate energy (i.e., 0.5 m/s). Another site (04165710) also located at higher latitudes with rather high streamflow velocities, had comparable hydro power potential to that of the solar power potential with an average harvestable hydro energy over average harvestable solar energy ratio of near 30%. However, two other sites (07374525 and 09527597), which had significant streamflow velocities leading to a hydro energy harvesting potential about 50% of the solar energy harvesting potential, were not located at high latitudes.

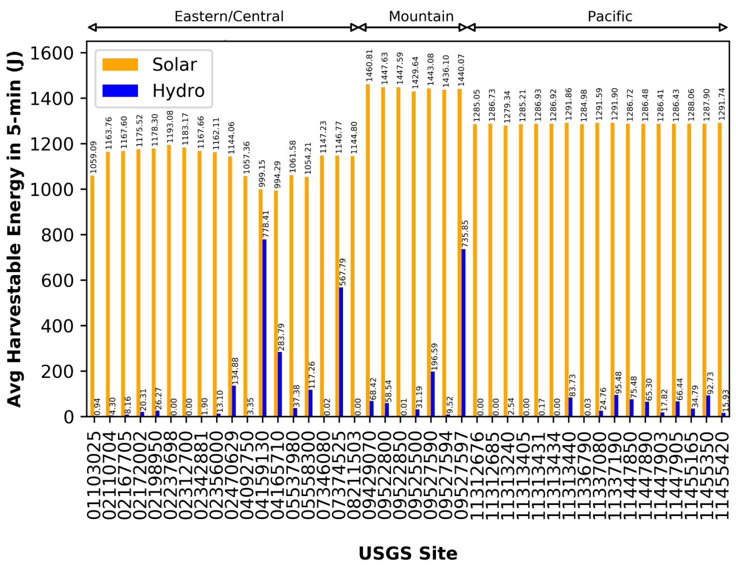


Figure 7.4: **Estimated harvestable solar and hydro power.** Average harvestable solar (in absence of tree canopy) and hydro energy (Joules) in five-minute intervals over the 2010-2014 simulation period at each of the 42 USGS sites.

After estimating the average energy generation accross sites with solar and hydro power, we then considered the complete water quality station operation with a 5-minute sampling interval. Figure 7.6 (a) shows the percentage of sample loss at each station due to power outages when using only the two hydro energy harvesters and sampling every five minutes. When using only the two hydro energy harvesters, the average sample loss for the 42 sites was 45.3%, while 10 sites had no sample loss for a fixed sampling interval of five minutes. Figure 7.6 (b) shows the energy loss due to battery saturation over total harvestable energy for the same sites and over the same time period using the two hydro energy harvesters. To leverage this extra energy and reduce the energy loss, sampling can be done more frequently. The median and average energy loss over total harvestable energy help show the variability of such extra energy across the stations which were 12.4% and 32.1% respectively across all stations.

Table 7.2 shows classification of the sites based on the scenarios of sample loss and/or energy loss when using only the hydro energy harvester. For the 10 of the sites where no sample loss was expected while energy loss was experienced, these sites can either have fixed sampling rates smaller than five minutes with no sample loss, or can have dynamic sampling intervals over the sensors' deployment period. For these sites, sampling intervals can be decreased some, but not much based on the availability of harvestable energy. For 14 other sites, both sample loss and energy loss

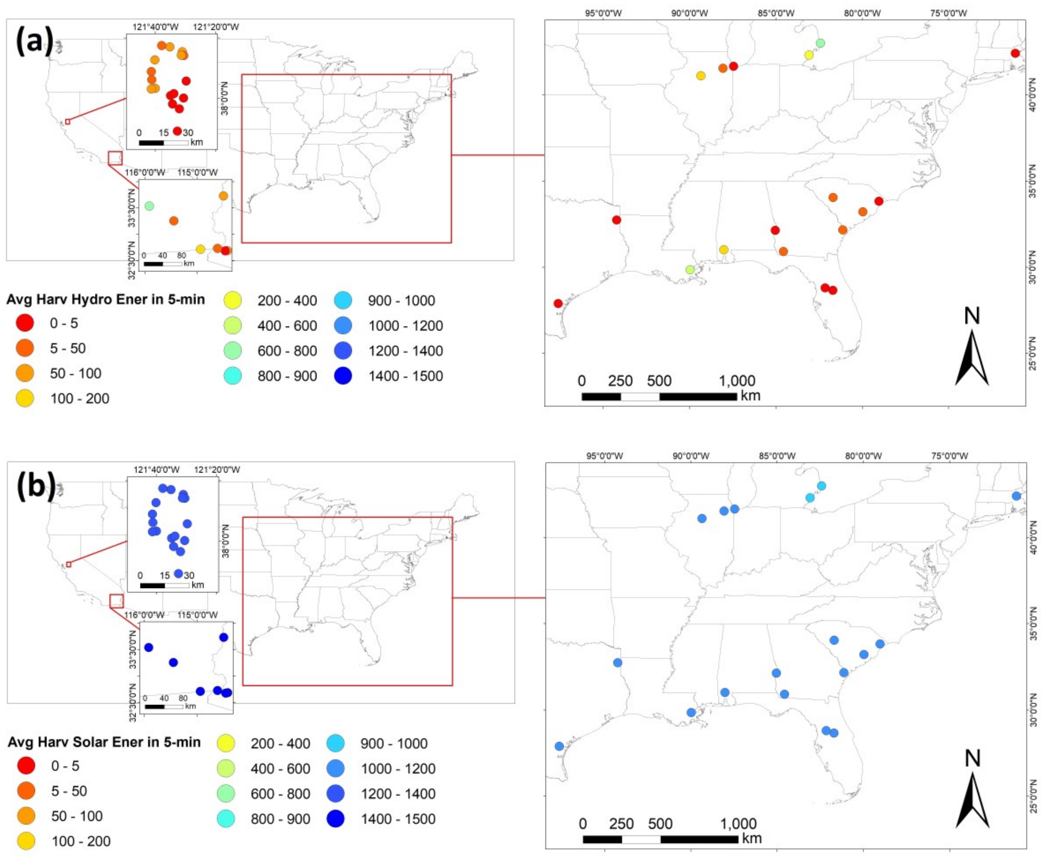


Figure 7.5: **Spatial distribution of estimated harvestable solar and hydro power.** Average harvestable energy (Joules) in five-minute intervals for all the 42 USGS locations over the 2010-2014 simulation period at each of the 42 USGS sites (a) hydro energy harvesting, (b) solar energy harvesting (in absence of tree canopy). The results are shown in three different frames based on the locations of the sites for both (a) and (b) (right frames: US Eastern, top left frames: US Pacific time, and bottom left frames: US Mountain time zone).

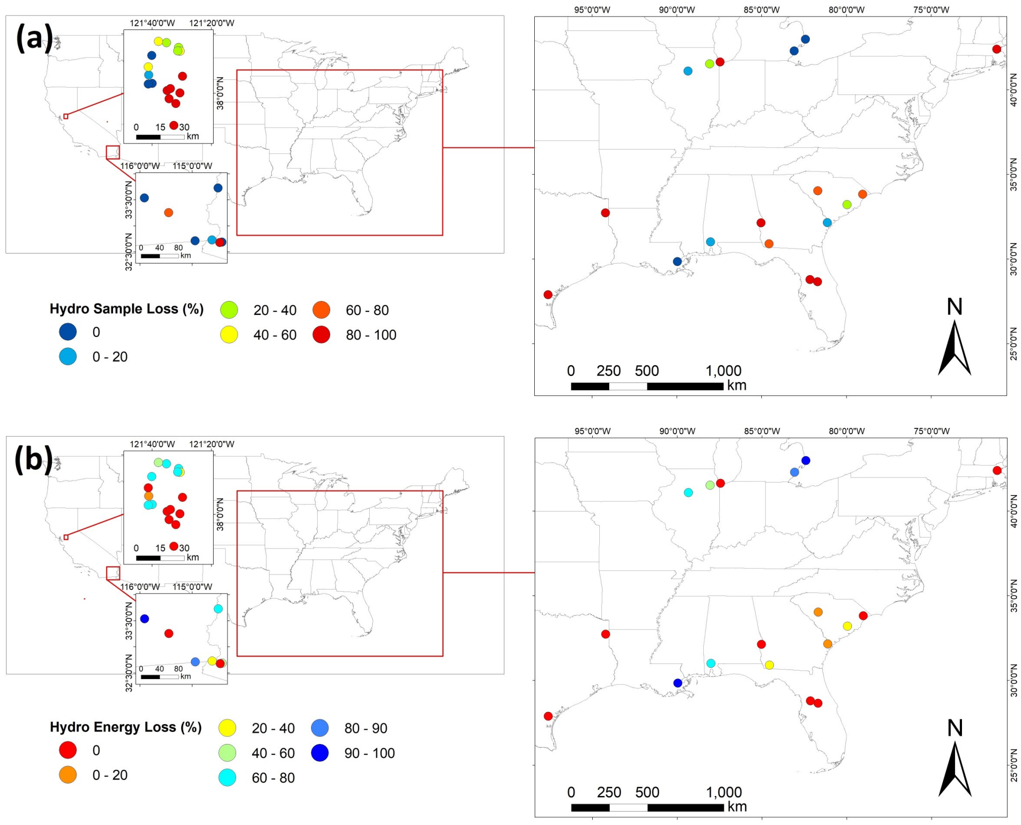


Figure 7.6: **Water Quality station simulations.** (a) The percentage of five-minute interval sample loss due to power outage (b) energy loss due to battery saturation over total harvestable energy (%) for hydro energy harvesting. The results are shown in three different frames based on the locations of the sites for both (a) and (b) (right frames: US Eastern, top left frames: US Pacific time, and bottom left frames: US Mountain time zone).

were experienced, indicating that there are long periods where little or no power is harvested, despite short periods of very high streamflow velocity that exceeded battery capacity. For 18 of the sites, sample loss ranging from 42.9% to 88.6% occurred with no energy loss. These sites harvest less power compared to other sites and depend more heavily on the initial battery charge; the more the initial charge is, the less sample loss they experience.

Table 7.2: USGS sites classification based on sample/energy loss for five-minute sampling interval and given design for battery (capacity =0.4kwh, initial charge=0.4kwh).

Class number	**Number of sites**	**Energy Loss happening**	**Sample Loss happening**
1	10	YES	NO
2	0	NO	NO
3	14	YES	YES
4	18	NO	YES

When using only the solar energy harvester, the results showed no sample loss while the energy loss over total harvestable solar energy had values ranging from 97.0 to 98.0% for all the 42 locations. Therefore, for the given energy consumption budget from the water monitoring system, the sampling interval can be significantly lowered to more finely measure water parameters using solar energy harvesting alone, assuming cloud cover is the only factor limiting solar energy harvesting at the site. This again emphasizes the high potential of the solar compared to hydro energy harvesting under this open sky assumption.

To better understand the hydro power harvesting potential for the 42 sites, we first explored the smallest sampling interval leading to no sample loss for each site Figure 7.7. 12 sites were able to sample at every five minutes or smaller sampling intervals while they experienced no sample loss. These sites have higher potential for hydro power harvesting compared to rest of the sites and had average harvestable energy ranging from 58.5 to 778.4 Joules per five minutes. Most interestingly, three of such sites (04159130, 04165710, and 09527597) were able to sample as frequent as every one minute without experiencing any sample loss. Another important observation among such 12 sites with high hydro power potential is that there are five sites with average harvestable energy from 58.5 to 90 Joules per five minutes supporting sampling intervals as low as two- or three-minutes with no sample loss while for site 07374525 with an average harvestable energy of 567.8 Joules per five-minutes, the lowest supported sampling interval with no sample loss is higher (five-minutes) due to presence of extreme low and high streamflow velocity periods at this site. Seven sites had more typical hydro power potential supporting minimum sampling interval with no sample loss in the range of six to 15 minutes. The average harvestable energy for such sites ranged from 15.9 to 117.3 Joules per five minutes. Eight sites with lower hydro power potential could support sampling intervals between 16 and 66 minutes with average harvestable energy from 3.3-65.3 Joules per five minutes while two other sites were able to support much less frequent samplings with no sample loss (i.e., sampling intervals of 152 and 235 minutes). Finally, for 13 of the sites with extremely low or no hydro power potential, regardless of how large the sampling interval was selected, sampling loss was experienced (no value for minimum sampling interval with no sample loss). This was expected, as we assumed 2% self-discharge in one month, meaning that after at most 50 months, a battery with full initial charge would be completely depleted (for a case where no sensing nor communication is performed) while the entire simulation period for the simulations was 5 years (60 month). This equates to no samples being taken for the 10 last months of the simulations for such sites.

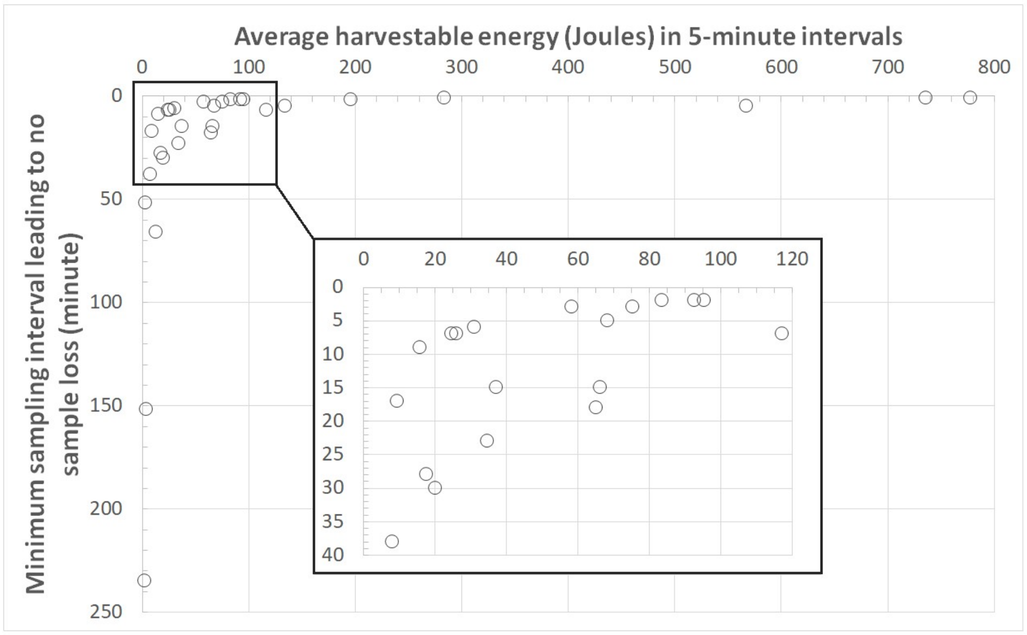


Figure 7.7: **Average harvestable energy vs minimum sampling interval resulting in no sample loss.** Minimum sampling interval leading to no sample loss against the average harvestable energy for 29 out of the 42 USGS locations over the 2010-2014 simulation period at each of the USGS sites using only hydro harvesting. For 13 of the sites with extremely low or no hydro harvesting potential irrespective of how large the sampling interval was chosen, sample loss was always experienced (no value for minimum sampling interval with no sample loss for such sites exist).

We further explored the effect of sampling interval on hydro-powered sensing system, for three chosen sites with different hydro power potentials. Figure 7.8 shows the effect of sampling interval on sample loss and harvestable energy loss ratio for three different harvesting sites. These three sites were selected based on their potential harvestable hydro energy representing (a) a low-potential hydro harvesting site (USGS 04092750), (b) a typical-potential hydro harvesting site (USGS 05537980), and (c) a high-potential hydro harvesting site (USGS 04165710). The low-potential hydro harvesting site is in class 4 of Table 7.2, meaning no energy loss occurred but sample loss did occur when the sampling interval was five-minute. The average harvestable energy in five minutes for this site was 3.3 Joules. For a five-minute sampling interval, the sample loss was 86.8% while no energy loss was expected. For sampling intervals greater than or equal to 52 minutes, no sample loss was expected (which made this site as a site with low hydro power potential). For any sampling interval, from one to 60 minutes, no energy loss was experienced, suggesting the battery may be oversized for this scenario.

The typical-potential harvesting site, 05537980, is in class 3 of Table 7.2, meaning both energy loss and sample loss occurred when the sampling interval is five-minute. The average harvestable energy in five minutes for this site was 37.4 Joules. For a five-minute sampling interval, the sample loss was 31.4% while the energy loss ratio was 47.4%. However, for sampling intervals greater than or equal to 15 minutes, no sample loss was experienced (which made this site as a site with typical hydro power potential) but the energy loss ratio became even higher. The high-harvesting site is in class 1 of Table 7.2, meaning energy loss occurred but no sample loss when the sampling interval was five-minute. The average harvestable energy in five minutes for this site was 283.8 Joules. For a five-minute water quality measurement sampling interval, the sample loss was zero while the energy loss was 89.6%. Even by decreasing the sampling interval to one-minute, no sample loss was observed (which made this site as a site with high hydro power potential) while the energy loss ratio was still considerable. This suggests that for sites that have high potential for hydro energy harvesting, it is possible to set a fixed sampling interval as low as one-minute to have a better temporal picture of the water quality dynamics within a system.

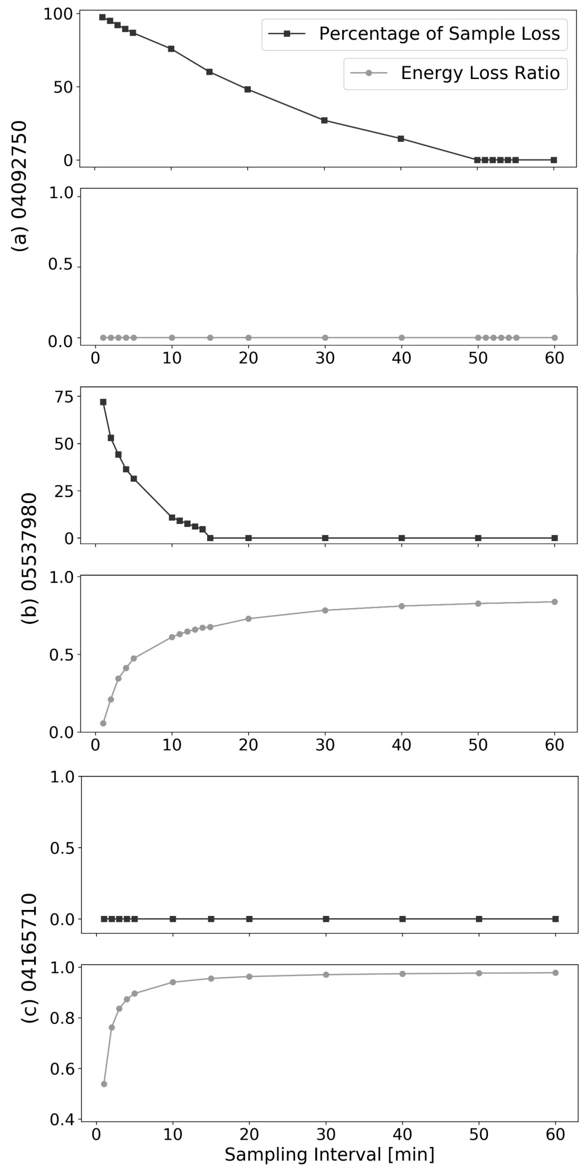


Figure 7.8: **Average harvestable energy vs minimum sampling interval resulting in no sample loss.** Minimum sampling interval leading to no sample loss against the average harvestable energy for 29 out of the 42 USGS locations over the 2010-2014 simulation period at each of the USGS sites using only hydro harvesting. For 13 of the sites with extremely low or no hydro harvesting potential irrespective of how large the sampling interval was chosen, sample loss was always experienced (no value for minimum sampling interval with no sample loss for such sites exist).

7.2 Water Leakage Detection

The goal of this use case is to design a water leakage detection SPIoT sensor powered by thermal energy. We adopt as our baseline harvester the same mini-harvester thermal energy generator (Marlow EHA-PA1AN1-R03) used in Chapter 4. By selecting this harvester, we can use the dataset collected on the thermal energy study to simulate how different hardware and operation choices affect the water leakage detection application as shown in Section 4.3.2. For this water leakage sensor, we assume the use of the LTC3108 integrated circuit in our energy harvesting front-end, with a circuit schematics (Figure 7.10) following the LTC3108 documentation. In this design we use the highlighted blocks of the SPIoT design and deployment framework in Figure 7.9.

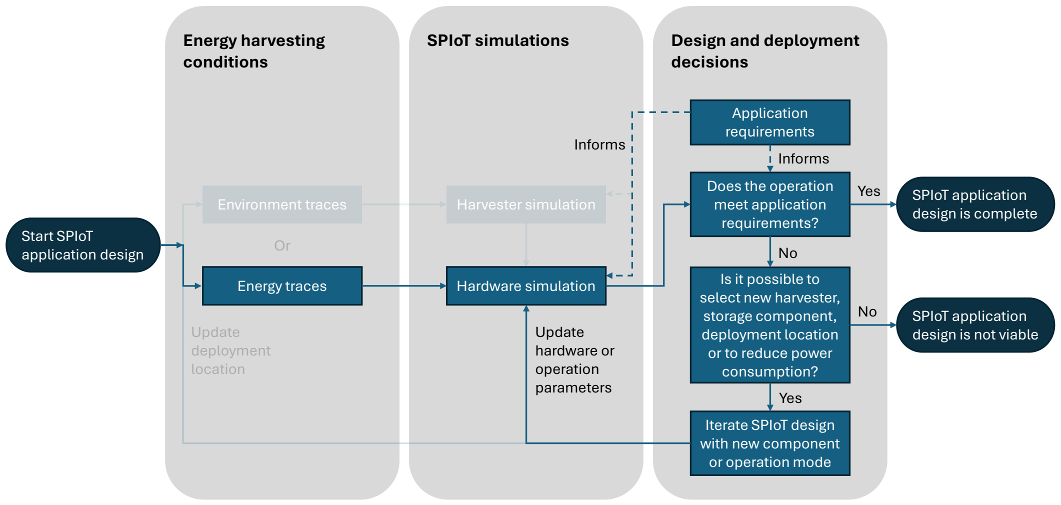

Energy harvesting conditions
SPIoT simulations
Design and deployment decisions
Application requirements
Informs
Informs
Environment traces
Harvester simulation
Does the operation meet application requirements?
Yes
SPIoT application design is complete
Start SPIoT application design
Or
No
Energy traces
Hardware simulation
Is it possible to select new harvester, storage component, deployment location or to reduce power consumption?
No
SPIoT application design is not viable
Update deployment location
Update hardware or operation parameters
Yes
Iterate SPIoT design with new component or operation mode

Figure 7.9: **Flow chart components related with the self-powered water leakage sensor design.** We use the energy traces collected in the thermal energy harvesting dataset to simulate and select components for the SPIoT device.

To demonstrate the use of integrated SPIoT models in the device design phase, we assumed our water leakage application requires to communicate using LoRaWAN, with a keep alive interval of 10 minutes and an alarm interval of 5 minutes when water is detected. We also assume that the alarm state will be turned off in up to 24 hours. As a reference, Dragino's LWL02, a commercially available water leakage

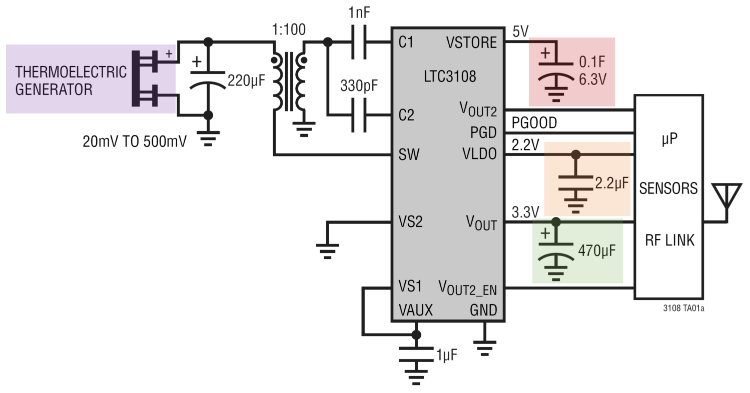


Figure 7.10: **LTC3108 reference circuit schematics.** System and figure were derived from the LTC3108 datasheet. We later modify this circuit by adding TEG devices in parallel and replacing capacitors connected to Vstore and Vout, as to meet the water leakage application energy consumption demands.

sensor IoT device powered by a primary battery, adopts a 12-hour keep alive interval as its default configuration to save battery. Therefore the self-powered water leakage sensor design should be able to provide much more fine-grained observability on the sensor operation status.

To summarize, the design requirements for this study case are: (1) Use thermal energy harvesting to support the sensor operation; (2) Use LoRaWAN as its communication protocol; (3) Transmit keep alive messages every 10 minutes; and (4) When water is detected, transmit every 5 minutes for at least 24 hours.

To demonstrate this SPIoT device design task, we retrofit a water leakage sensor LWL02 to use thermal energy harvesting, and we set the device to operate with a 10 minute keep alive interval. As the energy harvesting front end, we use an energy harvesting demonstration board, the LTC3108EDE. We adopt super capacitors as the energy storage components, and multiple units of the TEG model Marlow EHA-PA1AN1-R03 as harvesters. With the load and harvesting circuitry set, the design decisions are reduced to selecting appropriate number of harvesters and super capacitor components.

To understand the energy consumption requirements, we perform experiments with off-the shelf super capacitors to verify what devices can support the high current consumption of the LoRaWAN radio without dropping their voltage significantly. We decide that 1F is an adequate storage capacity to support the LWL02 device for a few communication cycles. We use the LWL02 manufacturer documentation to derive the device's power consumption over a range of keep alive interval configurations as we show in Figure 7.11. We perform an energy consumption experiment by attaching a 1F super capacitor to the LWL02, and verifying the super capacitor voltage drop with each communication event, confirming the energy consumption information from the manufacturer (Figure 7.12).

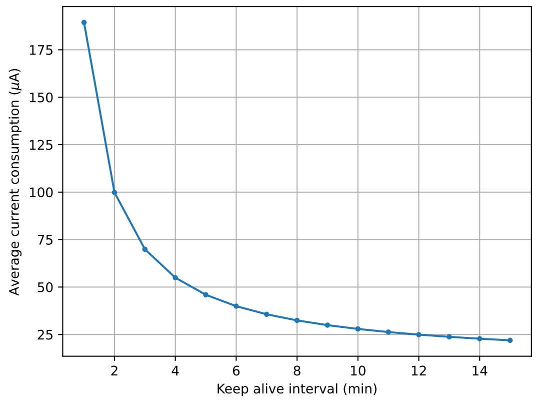


Figure 7.11: **Water leakage LWL02 energy consumption vs keep alive interval configurations.** Estimated average current is 27.9 μA for a 10-min keep alive interval and 45.9 μA for a 5-min interval.

With the expected sensor's energy consumption, we create an SPIoT model as used in Chapter 4, modeling the LTC3108 charging and discharging behavior as described in the manufacturer documentation. As shown in the experiment of Figure 7.12, an 1F super capacitor can provide up to 8 hours of sensor operation when full, and the device's operation remains stable along these 8 hours, meaning that voltage drop due to the device's peak current consumption does not cause disruptions(for instance device's reset). We then select 1F as a good compromise energy storage component to be attached to the Vout node of the circuit. We select a smaller 220 mF capacitor for the Vstore node, as a compromise between capacity and leakage characteristics.

We run SPIoT simulations using the recorded thermal energy harvesting profiles for location TP001 of Chapter 4, and we sweep the choice for number of TEG components to be used in series as a scaling factor for energy harvesting as done with the energy harvesting score in Chapter 5. In Figure 7.13 we show the energy harvesting conditions simulated, and in Figure 7.14 we show the voltage level of the super capacitors used in the water leakage sensor design.

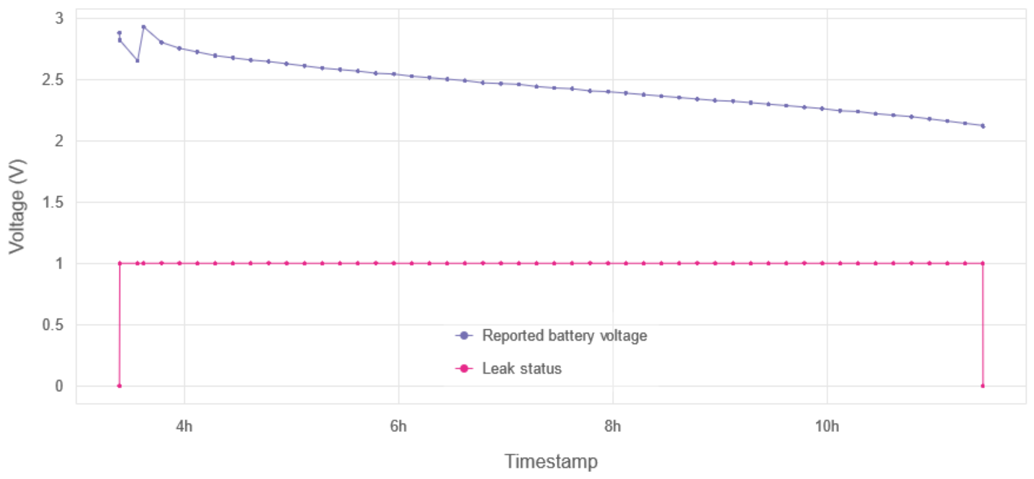


Figure 7.12: **Water leakage retrofit super cap discharge experiment.** The purple line represents the battery voltage readings from the LWL02 device. The pink line is the leak status output for reference. Discharge rate matches the manufacturer's information with an average current consumption of 27.9 μA.

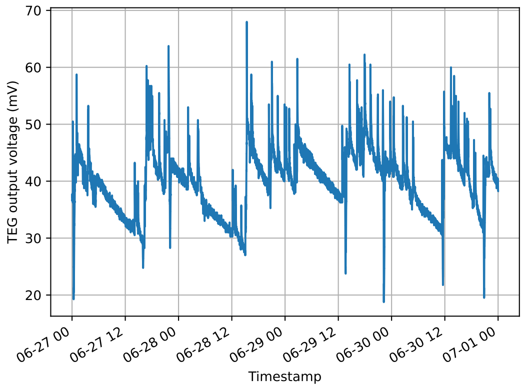
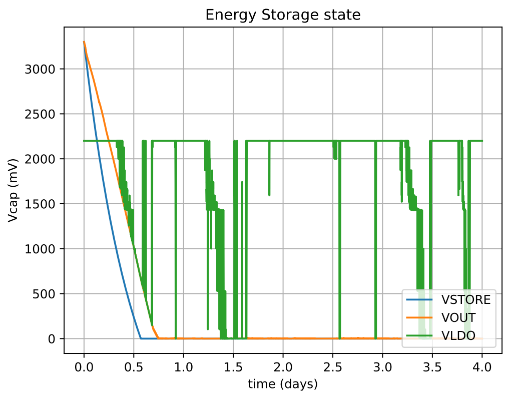
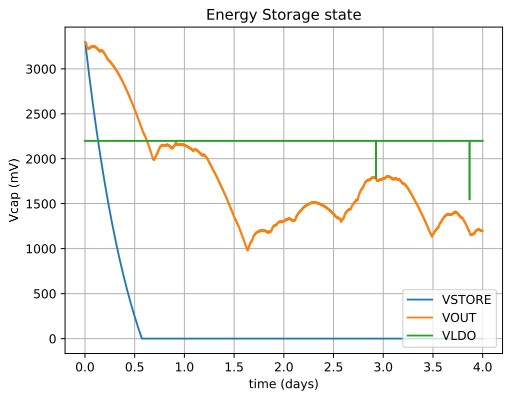


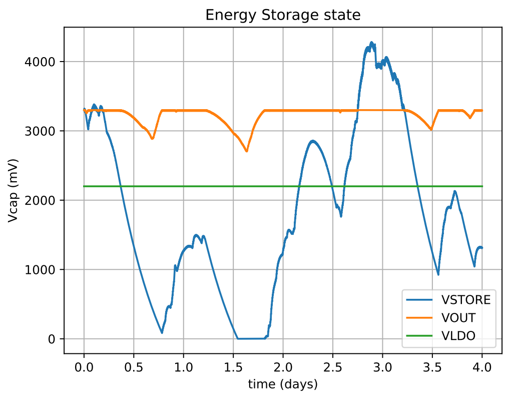
Figure 7.13: **Open circuit voltage of a single TEG in deployment TP001.**



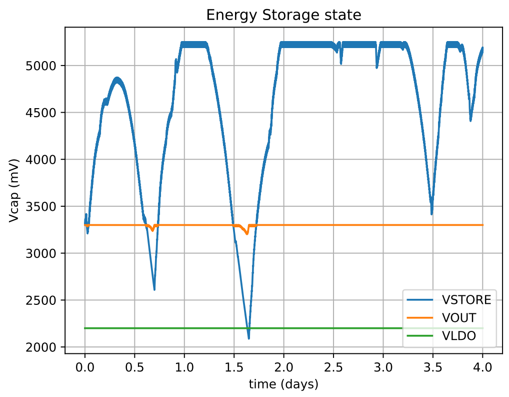
(a) Simulation of the water leakage sensor with one TEG.



(b) Simulation of the water leakage sensor with two TEGs.



(c) Simulation of the water leakage sensor with three TEGs.



(d) Simulation of the water leakage sensor with four TEGs.

Figure 7.14: **Water leakage sensor simulations.** The device turns off when the Vout drops bellow 2.1V. Simulations with one and two TEGs show intermittent operation. Simulations with three and four TEG components results in continuous operation of the water leakage sensor.

From the SPIoT simulations, we find that with 3 TEG modules, the energy harvesting front end is capable of supporting LWL02 normal operation indefinitely and the alarm operation for more than 24 hours. We then build and deploy a prototype to evaluate its operation behavior as we show in Figure 7.15.

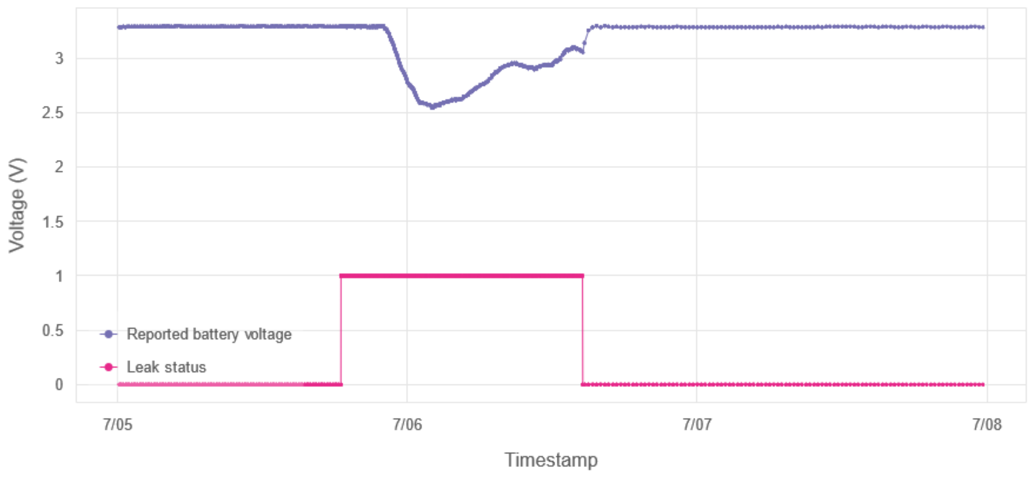


Figure 7.15: **Water leakage LWL02 retrofit operation on deployment site.** The device keep operating through a water leakage event, as intended in the design phase.

7.3 Contributions and Outcomes

From the best of our knowledge, the water quality station design was the first work to use a large streamflow velocity dataset to analyze the viability of using mini hydro turbines to power an IoT device. We found that by using large available datasets with the selected mini turbine and solar panel harvester models, we could estimate harvesting conditions without the effort of doing field collection on all theses sites. We combined the harvesting conditions with SPIoT device models to simulate how a SPIoT application would perform in a wide range of deployment locations and we evaluated the feasibility of a water quality application with different harvesting modalities. While Buchli et al. [31] used their own astronomical radiation model and solar panel conversion model to estimate harvested energy, we used satellite-derived solar radiation parameters from SolarAnywhere [36] and pvlib [63], a community managed python module to simulate solar panels with more complex models.

From the best of our knowledge, the water leakage sensor design use case is one of the first studies to show how to use an energy harvesting dataset to inform SPIoT design. We found that with our collected thermal dataset, an energy harvesting front-end model and energy consumption data from a commercially available water leak sensor, we are able to simulate different SPIoT design options and choose harvesters and capacitor components as to reach 100% availability under recorded energy harvesting conditions. The work from Buchli et al.[31] also demonstrates the use of an energy harvesting trace to design an SPIoT device, but instead of using profiled traces, they use a model to generate a synthetic energy harvesting trace to support the SPIoT design, and then a solar radiation dataset to calibrate and validate their system. Our study case is also one of the first studies to demonstrate a practical SPIoT design using thermal energy harvesting modality as opposed to the most common case using outdoor solar[70, 31].

Chapter 8: Closing Remarks

We demonstrated in this thesis the use of integrated SPIoT models to profile energy harvesting data, and we create a scoring metric that reflects how much useful energy is available to be harvested by SPIoT devices in a given location. We show a SPIoT design methodology using application requirements, SPIoT models and energy harvesting profiled traces. We show how to use SPIoT models to evaluate harvesting modalities from datasets containing physical quantities related to energy generation. Finally we complement the device-level design with cyber infrastructure needed to achieve end-to-end applications, by designing a cloud-based data storage and visualization system.

8.1 Open Problems

We select a few important open problems for each contribution stated in Section 1.2: (1) How should we choose which environments are more important to characterize? When do we decide that we have captured enough information about the energy harvesting conditions? How significant are local variations on energy harvesting deployments?; (2) How do we perform quantitative evaluations for the energy harvesting score? How to derive SPIoT device ratings from real-world devices? How to estimate energy harvesting scores for an even larger number of spaces? How do we drive the energy harvesting score adoption by the general public?; (3) How to design more robust and cost effective IoT backend systems that fits user requirements? How do we lower the barrier-to-entry of end-to-end IoT applications?; (4) How to use other existing datasets to support SPIoT design with new modalities? How do we plan data collection for future applications?; and (5) How can we extract more information from collected energy harvesting profiles to support SPIoT design? How to quantify the impact of selected SPIoT components on the robustness of the solution or on its energy harvesting score rating?

8.2 Publications

My individual work and collaborations during my time at the University of Virginia resulted in nine published papers. My first-author published works[85, 77, 121] covered in this thesis are presented respectively in Chapter 7, Chapter 6, and Chapter 4. The contents of Chapter 5 were submitted to SenSys 24' conference, and are currently under review.

8.2.1 Completed Works

1. Exploring the complementary relationship between solar and hydro energy harvesting for self-powered water monitoring in low-light conditions [85]
2. Thermal Energy Harvesting Profiles in Residential Settings [121]
3. A cloud-based data storage and visualization tool for smart city IoT: flood warning as an example application [77]
4. RetroIoT: retrofitting internet of things deployments by hiding data in battery readings [108]
5. Low power but high energy: The looming costs of billions of smart devices [136]
6. NexusEdge: Leveraging IoT Gateways for a Decentralized Edge Computing Platform [97]
7. SolarWalk: smart home occupant identification using unobtrusive indoor photovoltaic harvesters [27]
8. SolarWalk Dataset: Occupant Identification Using Indoor Photovoltaic Harvester Output Voltage [109]
9. Hydrologic Modeling and System Optimization for IoT Flood Management [71]

8.2.2 Submitted and Planned Works

1. Enabling Successful IoT Deployments with the Energy Harvesting Score (submitted to SenSys '24)
2. fReeLoaders: A Reinforcement Learning based Task Scheduler for IoT Ecosystems (planned, initial draft completed)

3. Software Defined Solar Sensors (planned, initial draft completed)
4. Sensor-free Weather Monitoring using LoRa RSSI (planned, initial draft completed)
5. Estimating Energy Harvesting Conditions from Building Datasets (planned, conception stage)
6. A Cloud-based Platform to Support Stormwater Infrastructure Management, and Decision Making with Simulated, Crowd-sourced and Real-Time IoT data (planned, conception stage)

Bibliography

[1] Kofi Sarpong Adu-Manu, Nadir Adam, Cristiano Tapparello, Hoda Ayatollahi, and Wendi Heinzelman. Energy-Harvesting Wireless Sensor Networks (EH-WSNs): A Review. *ACM Transactions on Sensor Networks*, 14(2):10:1–10:50, April 2018.

[2] Rehan Ahmed, Bernhard Buchli, Stefan Draskovic, Lukas Sigrist, Pratyush Kumar, and Lothar Thiele. Optimal Power Management with Guaranteed Minimum Energy Utilization for Solar Energy Harvesting Systems. *ACM Transactions on Embedded Computing Systems*, 18(4):30:1–30:26, June 2019.

[3] Saad Ahmed, Saad Ahmed, Naveed Anwar Bhatti, Naveed Anwar Bhatti, Naveed Anwar Bhatti, Muhammad Hamad Alizai, Muhammad Hamad Alizai, Junaid Haroon Siddiqui, Junaid Haroon Siddiqui, Luca Mottola, and Luca Mottola. Fast and energy-efficient state checkpointing for intermittent computing. *ACM Transactions in Embedded Computing Systems*, 2020.

[4] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys Tutorials*, 17(4):2347–2376, 2015. Conference Name: IEEE Communications Surveys Tutorials.

[5] Md. Eshrat E.E Alahi, Najid Pereira-Ishak, Subhas Chandra Mukhopadhyay, and Lucy Burkitt. An Internet-of-Things Enabled Smart Sensing System for Nitrate Monitoring. *IEEE Internet of Things Journal*, 5(6):4409–4417, December 2018. Conference Name: IEEE Internet of Things Journal.

[6] Amazon Web Services. Amazon API Gateway | API Management | Amazon Web Services. `https://aws.amazon.com/api-gateway/`. Accessed: 2023-01-06.

[7] Amazon Web Services. Amazon QuickSight - Business Intelligence Service. `https://aws.amazon.com/quicksight/`. Accessed: 2022-09-14.

[8] Amazon Web Services. AWS Compute Optimizer. https://aws.amazon.com/compute-optimizer/. Accessed: 2022-09-14.

[9] Amazon Web Services. Cloud Object Storage – Amazon S3. https://aws.amazon.com/s3/. Accessed: 2022-10-21.

[10] Amazon Web Services. Customer Identity and Access Management – Amazon Cognito. https://aws.amazon.com/cognito/. Accessed: 2023-01-09.

[11] Amazon Web Services. Fully Managed Relational Database - Amazon RDS. https://aws.amazon.com/rds/. Accessed: 2022-09-14.

[12] Amazon Web Services. Fully MySQL and PostgreSQL Compatible Managed Database Service | Amazon Aurora | AWS. https://aws.amazon.com/rds/aurora/. Accessed: 2022-09-14.

[13] Amazon Web Services. Provision Infrastructure as Code - AWS CloudFormation - AWS. https://aws.amazon.com/cloudformation/. Accessed: 2023-01-23.

[14] Amazon Web Services. Quick Start — AWS SDK for pandas 2.18.0 documentation. https://aws-sdk-pandas.readthedocs.io/en/stable/. Accessed: 2023-01-06.

[15] Amazon Web Services. Sagemaker machine Learning. https://aws.amazon.com/sagemaker/. Accessed: 2022-09-14.

[16] Amazon Web Services. Secure and resizable cloud compute – Amazon EC2. https://aws.amazon.com/ec2/. Accessed: 2022-10-21.

[17] Amazon Web Services. Serverless Computing - AWS Lambda. https://aws.amazon.com/lambda/. Accessed: 2022-09-14.

[18] Amazon Web Services. Boto3 - The AWS SDK for Python. https://github.com/boto/boto3, October 2022. Accessed: 2022-10-21.

[19] Analog Devices. *LTC3108 DC/DC converter IC (Rev D.)*, 3 2019. https://www.analog.com/en/products/ltc3108.html.

[20] Birgitte Andersen, Jens C. Frisvad, Ib Søndergaard, Ib S. Rasmussen, and Lisbeth S. Larsen. Associations between fungal species and water - damaged building materials. *Applied and Environmental Microbiology*, 77(12):4180–4188, June 2011.

[21] Italo Armenti, Philip Asare, Juliana Su, and John Lach. A methodology for developing quality of information metrics for body sensor design. In *Proceedings of the conference on Wireless Health*, WH '12, pages 1–8, New York, NY, USA, October 2012. Association for Computing Machinery.

[22] The ThingsBoard Authors. ThingsBoard - Open-source IoT Platform. `https://thingsboard.io/`. Accessed: 2021-08-13.

[23] Awair. Awair - Air Quality Monitor. `https://www.getawair.com/`.

[24] Johan Barthelemy, Mehrdad Amirghasemi, Bilal Arshad, Cormac Fay, Hugh Forehead, Nathanael Hutchison, Umair Iqbal, Yan Li, Yan Qian, and Pascal Perez. Problem-Driven and Technology-Enabled Solutions for Safer Communities. In Juan Carlos Augusto, editor, *Handbook of Smart Cities*, pages 1–28. Springer International Publishing, Cham, 2020.

[25] Matthew Bartos, Brandon Wong, and Branko Kerkez. Open storm: a complete framework for sensing and control of urban watersheds. *Environmental Science: Water Research & Technology*, 4(3):346–358, 2018. Publisher: Royal Society of Chemistry.

[26] Philip J. Basford, Florentin M. J. Bulot, Mihaela Apetroaie-Cristea, Simon J. Cox, and Steven J. Ossont. LoRaWAN for Smart City IoT Deployments: A Long Term Evaluation. *Sensors*, 20(3):648, January 2020. Number: 3 Publisher: Multidisciplinary Digital Publishing Institute.

[27] Md Fazlay Rabbi Masum Billah, Nurani Saoda, Victor Ariel Leal Sobral, Tushar Routh, Wenpeng Wang, and Bradford Campbell. Solarwalk: smart home occupant identification using unobtrusive indoor photovoltaic harvesters. In *Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, BuildSys '22, page 178–187, New York, NY, USA, 2022. Association for Computing Machinery.

[28] Broadcom. Bitnami. `https://bitnami.com/`. Accessed: 2022-09-14.

[29] Hannah Brunner, Jasper de Winkel, Carlo Alberto Boano, Przemysław Pawełczak, and Kay Uwe Römer. Simba: A unified framework to explore and facilitate the design of battery-free systems. In *Proceedings of the 23rd International Conference on Information Processing in Sensor Networks (IPSN)*, Proceedings - 23rd ACM/IEEE International Conference on Information Processing in Sensor Networks, IPSN 2024, pages 138–150, May 2024. 23rd ACM/IEEE International Conference on Information Processing in Sensor Networks: IPSN 2024 ; Conference date: 13-05-2024 Through 16-05-2024.

[30] Bernhard Buchli, Felix Sutton, Jan Beutel, and Lothar Thiele. Dynamic power management for long-term energy neutral operation of solar energy harvesting systems. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, SenSys '14, page 31–45, New York, NY, USA, 2014. Association for Computing Machinery.

[31] Bernhard Buchli, Felix Sutton, Jan Beutel, and Lothar Thiele. Towards Enabling Uninterrupted Long-Term Operation of Solar Energy Harvesting Embedded Systems. In Bhaskar Krishnamachari, Amy L. Murphy, and Niki Trigoni, editors, *Wireless Sensor Networks*, Lecture Notes in Computer Science, pages 66–83, Cham, 2014. Springer International Publishing.

[32] Ismail Butun, editor. *Industrial IoT: Challenges, Design Principles, Applications, and Security*. Springer International Publishing, Cham, 2020.

[33] Katie Carlson, Ashif Chowdhury, Andrew Kepley, Ethan Somerville, Kevin Warshaw, and Jonathan Goodall. Smart Cities Solutions for More Flood Resilient Communities. In *2019 Systems and Information Engineering Design Symposium (SIEDS)*, pages 1–6, April 2019.

[34] Qi Chen, Ye Liu, Guangchi Liu, Qing Yang, Xianming Shi, Hongwei Gao, Lu Su, and Quanlong Li. Harvest Energy from the Water: A Self-Sustained Wireless Water Quality Sensing System. *ACM Transactions on Embedded Computing Systems*, 17(1):3:1–3:24, September 2017.

[35] Inc. Cisco Systems. Cisco Annual Internet Report (2018–2023) White Paper. `https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html`. Accessed: 2022-01-19.

[36] L.L.C. Clean Power Research. Solaranywhere. `https://data.solaranywhere.com`. Accessed: 2019-10-09.

[37] McKinsey & Company. Where and how to capture accelerating IoT value. `https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/iot-value-set-to-accelerate-through-2030-where-and-how-to-capture-it`. Accessed: 2022-01-19.

[38] Mina Cong, Kanghwan Kim, Maria Gorlatova, John Sarik, John Kymissis, and Gil Zussman. CRAWDAD data set columbia/kinetic (v. 2014-05-13). Downloaded from http://crawdad.org/columbia/kinetic/, May 2014.

[39] Minhao Cui, Qing Wang, and Jie Xiong. Bracelet+: Harvesting the leaked rf energy in vlc with wearable bracelet antenna. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, SenSys '22, page 250–262, New York, NY, USA, 2023. Association for Computing Machinery.

[40] Simone Dalola, Marco Ferrari, Vittorio Ferrari, Michele Guizzetti, Daniele Marioli, and Andrea Taroni. Characterization of thermoelectric modules for powering autonomous sensors. *IEEE Transactions on Instrumentation and Measurement*, 58(1):99–107, 2009.

[41] Danco Davcev, Kosta Mitreski, Stefan Trajkovic, Viktor Nikolovski, and Nikola Koteli. IoT agriculture system based on LoRaWAN. In *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, pages 1–4, June 2018.

[42] Fang Deng, Xianghu Yue, Xinyu Fan, Shengpan Guan, Yue Xu, and Jie Chen. Multisource energy harvesting system for a wireless sensor network node in the field environment. *IEEE Internet of Things Journal*, 6(1):918–927, 2019.

[43] Adam Drenoyanis, Raad Raad, Ivan Wady, and Carmel Krogh. Implementation of an IoT Based Radar Sensor Network for Wastewater Management. *Sensors*, 19(2):254, January 2019. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

[44] Christian Ebi, Fabian Schaltegger, Andreas Rüst, and Frank Blumensaat. Synchronous LoRa Mesh Network to Monitor Processes in Underground Infrastructure. *IEEE Access*, 7:57663–57677, 2019. Conference Name: IEEE Access.

[45] Victron Energy. Victron energy - batteries. `https://www.victronenergy.com/batteries`. Accessed: 2022-02-22.

[46] EnOcean. EnOcean – Sustainable IoT Solutions. `https://www.enocean.com/en/`.

[47] European Commission, Directorate-General for Energy. Energy Performance of Buildings Directive. `https://energy.ec.europa.eu/topics/energy-efficiency/energy-efficient-buildings/energy-performance-buildings-directive_en`. Accessed: 2024-07-10.

[48] Everactive. Self-Powered Industrial IoT. `https://everactive.com/`.

[49] Dawei Fan, Luis Lopez Ruiz, Jiaqi Gong, and John Lach. EHDC: An Energy Harvesting Modeling and Profiling Platform for Body Sensor Networks. *IEEE Journal of Biomedical and Health Informatics*, 22(1):33–39, January 2018. Conference Name: IEEE Journal of Biomedical and Health Informatics.

[50] Dawei Fan, Luis Lopez Ruiz, and John Lach. Application-driven dynamic power management for self-powered vigilant monitoring. In *2018 IEEE 15th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, pages 210–213, March 2018. ISSN: 2376-8894.

[51] Kai Geissdoerfer, Raja Jurdak, Brano Kusy, and Marco Zimmerling. Getting more out of energy-harvesting systems: energy management under time-varying utility with preact. In *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*, IPSN '19, pages 109–120, New York, NY, USA, April 2019. Association for Computing Machinery.

[52] Decentlab GmbH. Decentlab. `https://www.decentlab.com`. Accessed: 2021-08-26.

[53] Maria Gorlatova, John Sarik, Guy Grebla, Mina Cong, Ioannis Kymissis, and Gil Zussman. Movers and shakers: kinetic energy harvesting for the internet of things. *ACM SIGMETRICS Performance Evaluation Review*, 42(1):407–419, June 2014.

[54] Maria Gorlatova, Aya Wallwater, and Gil Zussman. Networking low-power energy harvesting devices: Measurements and algorithms. In *2011 Proceedings IEEE INFOCOM*, pages 1602–1610, April 2011. ISSN: 0743-166X.

[55] Maria Gorlatova, Michael Zapas, Enlin Xu, Matthias Bahlke, Ioannis (John) Kymissis, and Gil Zussman. CRAWDAD data set columbia/enhants (v. 2011-04-07). Downloaded from http://crawdad.cs.dartmouth.edu/columbia/enhants, April 2011.

[56] Behrang H. Hamadani and Mark B. Campanelli. Photovoltaic characterization under artificial low irradiance conditions using reference solar cells. *IEEE Journal of Photovoltaics*, 10(4):1119–1125, 2020.

[57] Josiah Hester, Timothy Scott, and Jacob Sorber. Ekho: realistic and repeatable experimentation for tiny energy-harvesting sensors. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, SenSys '14, page 1–15, New York, NY, USA, 2014. Association for Computing Machinery.

[58] Josiah Hester, Lanny Sitanayah, Timothy Scott, and Jacob Sorber. Realistic and Repeatable Emulation of Energy Harvesting Environments. *ACM Transactions on Sensor Networks*, 13(2):16:1–16:33, April 2017.

[59] Josiah Hester and Jacob Sorber. Flicker: Rapid prototyping for the batteryless internet-of-things. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, SenSys '17, New York, NY, USA, 2017. Association for Computing Machinery.

[60] Josiah Hester and Jacob Sorber. The Future of Sensing is Batteryless, Intermittent, and Awesome. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, SenSys '17, pages 1–6, New York, NY, USA, November 2017. Association for Computing Machinery.

[61] Glenn A. Hodgkins, Paul H. Whitfield, Donald H. Burn, Jamie Hannaford, Benjamin Renard, Kerstin Stahl, Anne K. Fleig, Henrik Madsen, Luis Mediero, Johanna Korhonen, Conor Murphy, and Donna Wilson. Climate-driven variability in the occurrence of major floods across North America and Europe. *Journal of Hydrology*, 552:704–717, September 2017.

[62] Will Holmgren, Calama-Consulting, Tony Lorenzo, Uwe Krien, bmu, Mark Mikofski, Cliff Hansen, DaCoEx, konstant_t, Anton Driesse, mayudong, Leland Boeman, Ed Miller, Heliolytics, Marc A. Anoma, jforbess, pyElena21, Volker Beutner, Todd Hendricks, MLEEFS, Johannes Dollinger, Cedric Leroy, Cameron Stark, Kevin Anderson, Jonathan Gaffiot, Johannes Oos, Giuseppe Peronato, Birgit Schachler, Alan Mathew, and Alaina Kafkes. pvlib/pvlib-python v0.6.4, June 2019.

[63] William F. Holmgren, Clifford W. Hansen, and Mark A. Mikofski. pvlib python: a python package for modeling solar energy systems. *Journal of Open Source Software*, 3(29):884, September 2018.

[64] Seaformatics Systems Inc. WaterLily Turbine. `https://shop.waterlilyturbine.com/products/waterlily-turbine`. Accessed: 2022-02-11.

[65] YSI Inc. / Xylem Inc. Ysi water Quality Monitoring Buoys, Profilers, Real Time Data Systems. `https://www.ysi.com/products/monitoring-buoys-and-platforms?Page=1`. Accessed: 2022-02-11.

[66] The Things Industries. The Things Industries. `https://www.thethingsindustries.com/`. Accessed: 2021-08-11.

[67] Amna Iqbal and Stephan Olariu. A Survey of Enabling Technologies for Smart Communities. *Smart Cities*, 4(1):54–77, March 2021. Number: 1 Publisher: Multidisciplinary Digital Publishing Institute.

[68] Project Jupyter. Project Jupyter. `https://jupyter.org`. Accessed: 2022-09-14.

[69] Ervin Kamenar, Saša Zelenika, David Blažević, Senka Maćešić, Goran Gregov, Kristina Marković, and Vladimir Glažar. Harvesting of river flow energy for wireless sensor network technology. *Microsystem Technologies*, 22(7):1557–1574, July 2016.

[70] Aman Kansal, Jason Hsu, Sadaf Zahedi, and Mani B. Srivastava. Power management in energy harvesting sensor networks. *ACM Transactions on Embedded Computing Systems*, 6(4):32–es, September 2007.

[71] Nicolas Khattar, Taja M Washington, Arnold Mai, Lili Malinowski, Andrew N. Bowman, Khwanjira Phumphid, Victor A Leal Sobral, and Jonathan L. Goodall. Hydrologic modeling and system optimization for iot flood management. In *2023 Systems and Information Engineering Design Symposium (SIEDS)*, pages 137–142, 2023.

[72] Vasiliki Kourgiozou, Daniel Godoy Shimizu, Mark Dowson, Andrew Commin, Rui Tang, Dimitrios Rovas, and Dejan Mumovic. A new method for estimating the smart readiness of building stock data using display energy Certificate data. *Energy and Buildings*, 301:113673, December 2023.

[73] Kyocera. Kyocera KS20, Solar Panel, 20 Watt, 12 Volt. `https://www.ecodirect.com/product-p/kyocera-ks20.htm`. Accessed: 2019-09-09.

[74] Grafana Labs. Grafana: The open observability platform. `https://grafana.com/`. Accessed: 2022-09-14.

[75] Trong Nhan Le, Alain Pegatoquet, Olivier Berder, and Olivier Sentieys. A power manager with balanced quality of service for energy-harvesting wireless sensor nodes. In *Proceedings of the 2nd International Workshop on Energy Neutral Sensing Systems*, ENSsys '14, pages 19–24, New York, NY, USA, November 2014. Association for Computing Machinery.

[76] Victor Ariel Leal Sobral, John Lach, Jonathan L. Goodall, and Bradford Campbell. Thermal Energy Harvesting Profiles in Residential Settings, 2021.

[77] Victor Ariel Leal Sobral, Jacob Nelson, Loza Asmare, Abdullah Mahmood, Glen Mitchell, Kwadwo Tenkorang, Conor Todd, Bradford Campbell, and Jonathan L. Goodall. A cloud-based data storage and visualization tool for smart city iot: Flood warning as an example application. *Smart Cities*, 6(3):1416–1434, 2023.

[78] Chiara Lenz, Sergei Vostrikov, Philipp Mayer, and Michele Magno. From heat to power: Assessing thermoelectric energy harvesting for self-sustainable sensors. In *2023 IEEE International Workshop on Technologies for Defense and Security (TechDefense)*, pages 384–389, 2023.

[79] V.A. Levesque and K.A. Oberg. Techniques and Methods 3–A23: Computing Discharge Using the Index Velocity Method. `https://pubs.usgs.gov/tm/3a23/`. Accessed: 2019-08-25.

[80] Feng Li, Tianyu Xiang, Zicheng Chi, Jun Luo, Lihua Tang, Liya Zhao, and Yaowen Yang. Powering indoor sensing with airflows: a trinity of energy harvesting, synchronous duty-cycling, and sensing. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, New York, NY, USA, 2013. Association for Computing Machinery.

[81] Kris Lin, Jennifer Yu, Jason Hsu, Sadaf Zahedi, David Lee, Jonathan Friedman, Aman Kansal, Vijay Raghunathan, and Mani Srivastava. Heliomote: enabling long-lived sensor networks through solar energy harvesting. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, SenSys

'05, page 309, New York, NY, USA, 2005. Association for Computing Machinery.

[82] Qiliang Lin, Yi-Chung Chen, Fangliang Chen, Tejav DeGanyar, and Huiming Yin. Design and experiments of a thermoelectric-powered wireless sensor network platform for smart building envelope. *Applied Energy*, 305:117791, January 2022.

[83] Marten Lohstroh, Hokeun Kim, John C. Eidson, Chadlia Jerad, Beth Osyk, and Edward A. Lee. On Enabling Technologies for the Internet of Important Things. *IEEE Access*, 7:27244–27256, 2019. Conference Name: IEEE Access.

[84] EHP Environment Ltd. Ehp-environmental buoy monitoring solutions. `https://www.ehpenvironment.com/ca/monitoring-solutions-ca/environmental-buoy-2/`. Accessed: 2022-02-11.

[85] Iman Maghami, Victor A. L. Sobral, Mohamed M. Morsy, John C. Lach, and Jonathan L. Goodall. Exploring the complementary relationship between solar and hydro energy harvesting for self-powered water monitoring in low-light conditions. *Environmental Modelling & Software*, 140:105032, June 2021.

[86] Luca Mainetti, Luigi Patrono, and Antonio Vilei. Evolution of wireless sensor networks towards the Internet of Things: A survey. In *SoftCOM 2011, 19th International Conference on Software, Telecommunications and Computer Networks*, pages 1–6, September 2011.

[87] Ignacio Martínez, Belén Zalba, Raquel Trillo-Lado, Teresa Blanco, David Cambra, and Roberto Casas. Internet of Things (IoT) as Sustainable Development Goals (SDG) Enabling Technology towards Smart Readiness Indicators (SRI) for University Buildings. *Sustainability*, 13(14):7647, January 2021. Number: 14 Publisher: Multidisciplinary Digital Publishing Institute.

[88] Mojtaba Masoudinejad. Data-Sets for Indoor Photovoltaic Behavior in Low Lighting Conditions. *Data*, 5(2):32, June 2020. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

[89] Mojtaba Masoudinejad. *Modeling energy supply unit of ultra-low power devices with indoor photovoltaic harvesting*. PhD thesis, TU Dortmund University, 2020. Accepted: 2020-10-07T09:38:48Z.

[90] Mojtaba Masoudinejad. Open-Loop Dynamic Modeling of Low-Budget Batteries with Low-Power Loads. *Batteries*, 6(4):50, December 2020. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute.

[91] Mojtaba Masoudinejad, Michele Magno, Luca Benini, and Michael ten Hompel. Average Modelling of State-of-the-Art Ultra-low Power Energy Harvesting Converter IC. In *2018 International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM)*, pages 99–104, June 2018.

[92] Alexandra Medina-Pérez, David Sánchez-Rodríguez, and Itziar Alonso-González. An Internet of Thing Architecture Based on Message Queuing Telemetry Transport Protocol and Node-RED: A Case Study for Monitoring Radon Gas. *Smart Cities*, 4(2):803–818, June 2021. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

[93] Kais Mekki, Eddy Bajic, Frederic Chaxel, and Fernand Meyer. A comparative study of LPWAN technologies for large-scale IoT deployment. *ICT Express*, 5(1):1–7, March 2019.

[94] Azana Hafizah Mohd Aman, Elaheh Yadegaridehkordi, Zainab Senan Attarbashi, Rosilah Hassan, and Yong-Jin Park. A Survey on Trend and Classification of Internet of Things Reviews. *IEEE Access*, 8:111763–111782, 2020. Conference Name: IEEE Access.

[95] MQTT. MQTT - The Standard for IoT Messaging. `https://mqtt.org/`. Accessed: 2022-09-20.

[96] Inc. myDevices. myDevices Documentation. `https://docs.mydevices.com/`. Accessed: 2024-07-23.

[97] Nabeel Nasir, Victor Ariel Leal Sobral, Li-Pang Huang, and Bradford Campbell. Nexusedge: Leveraging iot gateways for a decentralized edge computing platform. In *2022 IEEE/ACM 7th Symposium on Edge Computing (SEC)*, pages 82–95, 2022.

[98] The Things Network. The Things Network. `https://www.thethingsnetwork.org/`. Accessed: 2021-08-11.

[99] Inc. NexSens Technology. Nexsens data Buoys. `https://www.nexsens.com/products/data-buoys`. Accessed: 2022-02-11.

[100] Bassirou Ngom, Moussa Diallo, Bamba Gueye, and Nicolas Marilleau. LoRa-based Measurement Station for Water Quality Monitoring: Case of Botanical Garden Pool. In *2019 IEEE Sensors Applications Symposium (SAS)*, pages 1–4, March 2019.

[101] Prusayon Nintanavongsa, Ufuk Muncuk, David Richard Lewis, and Kaushik Roy Chowdhury. Design optimization and implementation for rf energy harvesting circuits. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2(1):24–33, 2012.

[102] OpenJS Foundation. Node-RED. `https://nodered.org/`. Accessed: 2022-09-20.

[103] Oracle Corporation. MySQL. `https://www.mysql.com/`. Accessed: 2022-09-21.

[104] Vishwas Powar, Christopher Post, Elena Mikhailova, Chuck Cook, Mohammad Mayyan, Akshay Bapat, and Clifford Harmstad. Sensor networks for hydrometric monitoring of urban watercourses. In *2019 IEEE 16th International Conference on Smart Cities: Improving Quality of Life Using ICT IoT and AI (HONET-ICT)*, pages 085–089, October 2019. ISSN: 1949-4106.

[105] Michael Rode, Andrew J. Wade, Matthew J. Cohen, Robert T. Hensley, Michael J. Bowes, James W. Kirchner, George B. Arhonditsis, Phil Jordan, Brian Kronvang, Sarah J. Halliday, Richard A. Skeffington, Joachim C. Rozemeijer, Alice H. Aubert, Karsten Rinke, and Seifeddine Jomaa. Sensors in the Stream: The High-Frequency Wave of the Present. *Environmental Science & Technology*, 50(19):10297–10307, October 2016. Publisher: American Chemical Society.

[106] Sydney S. Ruhala and Jay P. Zarnetske. Using in-situ optical sensors to study dissolved organic carbon dynamics of streams and watersheds: A review. *Science of The Total Environment*, 575:713–723, January 2017.

[107] Prakhar Sah and Matthew Hicks. Hitchhiker's guide to secure checkpointing on energy-harvesting systems. In *Proceedings of the 11th International Workshop on Energy Harvesting & Energy-Neutral Sensing Systems*, ENSsys '23, page 8–15, New York, NY, USA, 2023. Association for Computing Machinery.

[108] Nurani Saoda, Md Fazlay Rabbi Masum Billah, Victor Ariel Leal Sobral, and Bradford Campbell. Solarwalk dataset: Occupant identification using indoor photovoltaic harvester output voltage. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, SenSys '22, page 1031–1034, New York, NY, USA, 2023. Association for Computing Machinery.

[109] Nurani Saoda, Md Fazlay Rabbi Masum Billah, Victor Ariel Leal Sobral, and Bradford Campbell. Solarwalk dataset: Occupant identification using indoor photovoltaic harvester output voltage. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, SenSys '22, page 1031–1034, New York, NY, USA, 2023. Association for Computing Machinery.

[110] Atlas Scientific. Environmental Robotics. `https://atlas-scientific.com/`. Accessed: 2020-10-11.

[111] Manajit Sengupta, Yu Xie, Anthony Lopez, Aron Habte, Galen Maclaurin, and James Shelby. The National Solar Radiation Data Base (NSRDB). *Renewable and Sustainable Energy Reviews*, 89:51–60, 2018.

[112] Sajjad Hussain Shah and Ilyas Yaqoob. A survey: Internet of Things (IOT) technologies, applications and challenges. In *2016 IEEE Smart Energy Grid Engineering (SEGE)*, pages 381–385, August 2016.

[113] Ahmed M. Shahat Osman and Ahmed Elragal. Smart Cities and Big Data Analytics: A Data-Driven Decision-Making Use Case. *Smart Cities*, 4(1):286–313, March 2021. Number: 1 Publisher: Multidisciplinary Digital Publishing Institute.

[114] Jothi Prasanna Shanmuga Sundaram, Wan Du, and Zhiwei Zhao. A Survey on LoRa Networking: Research Problems, Current Solutions, and Open Issues. *IEEE Communications Surveys Tutorials*, 22(1):371–388, 2020. Conference Name: IEEE Communications Surveys Tutorials.

[115] Lukas Sigrist, Andres Gomez, and Lothar Thiele. Dataset: Tracing Indoor Solar Harvesting. In *Proceedings of the 2nd Workshop on Data Acquisition To Analysis*, DATA'19, pages 47–50, New York, NY, USA, November 2019. Association for Computing Machinery.

[116] Lukas Sigrist, Andres Gomez, and Lothar Thiele. Long-Term Tracing of Indoor Solar Harvesting, August 2019.

[117] Marco Silva, André Riker, Joel Torrado, José Santos, and Marilia Curado. Extending energy neutral operation in internet of things. *IEEE Internet of Things Journal*, 9(10):7510–7524, 2022.

[118] Jaspreet Singh, Ranjit Kaur, and Damanpreet Singh. Energy harvesting in wireless sensor networks: A taxonomic survey. *International Journal of Energy Research*, 45(1):118–140, 2021.

[119] Priyanka Singla, Priyanka Singla, Shubhankar Suman Singh, Shubhankar Suman Singh, Smruti R. Sarangi, and Smruti R. Sarangi. Flexicheck: An adaptive checkpointing architecture for energy harvesting devices. *Design, Automation and Test in Europe*, 2019.

[120] Slack Technologies. Slack is your digital HQ | Slack. `https://slack.com/`. Accessed: 2022-09-14.

[121] Victor Ariel Leal Sobral, John Lach, Jonathan L. Goodall, and Bradford Campbell. Thermal Energy Harvesting Profiles in Residential Settings. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, SenSys

’21, pages 520–523, New York, NY, USA, November 2021. Association for Computing Machinery.

[122] Peter Spies, Markus Pollak, and Loreto Mateu, editors. *Handbook of Energy Harvesting Power Supplies and Applications.* Jenny Stanford Publishing, New York, June 2015.

[123] John A Stankovic and Jack Davidson. Raising Awareness of Security Challenges for the Internet of Trillions of Things. *The Bridge on Cybersecurity*, page 6, 2019.

[124] United States Geological Survey. USGS Water Data for the Nation. `https://waterdata.usgs.gov/nwis`. Accessed: 2019-08-25.

[125] Swagger UI. swagger-ui/dist at master · swagger-api/swagger-ui. `https://github.com/swagger-api/swagger-ui`. Accessed: 2023-01-06.

[126] Abbas Shah Syed, Daniel Sierra-Sosa, Anup Kumar, and Adel Elmaghraby. IoT in Smart Cities: A Survey of Technologies, Practices and Challenges. *Smart Cities*, 4(2):429–475, June 2021. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

[127] Nikolay Tcholtchev and Ina Schieferdecker. Sustainable and Reliable Information and Communication Technology for Resilient Smart Cities. *Smart Cities*, 4(1):156–176, March 2021. Number: 1 Publisher: Multidisciplinary Digital Publishing Institute.

[128] Yigit Tuncel, Ganapati Bhat, Jaehyun Park, and Umit Y. Ogras. Eco: Enabling energy-neutral iot devices through runtime allocation of harvested energy. *IEEE Internet of Things Journal*, 9(7):4833–4848, 2022.

[129] U. S. Department of Energy, Office of Energy Efficiency & Renewable Energy. About the Building Technologies Office. `https://www.energy.gov/eere/buildings/about-building-technologies-office`. Accessed: 2024-07-10.

[130] Ubidots. IoT platform | Internet of Things | Ubidots. `https://ubidots.com/`. Accessed: 2022-09-14.

[131] OAR US EPA. Summarized Data of the Building Assessment Survey and Evaluation Study. `https://www.epa.gov/indoor-air-quality-iaq/summarized-data-building-assessment-survey-and-evaluation-study`, September 2014. Accessed: 2022-02-20.

[132] ORD US EPA. Storm Water Management Model (SWMM). `https://www.epa.gov/water-research/storm-water-management-model-swmm`, May 2014. Accessed: 2022-09-14.

[133] UVA Hydroinformatics. uva-hydroinformatics/iot-cloud-platform: Cloud IoT Platform. `https://github.com/uva-hydroinformatics/iot-cloud-platform`. Accessed: 2023-01-23.

[134] Peter Wägemann, Tobias Distler, Heiko Janker, Phillip Raffeck, Volkmar Sieh, and Wolfgang SchröDer-Preikschat. Operating energy-neutral real-time systems. *ACM Trans. Embed. Comput. Syst.*, 17(1), aug 2017.

[135] Gabriela Walczyk and Andrzej Ożadowicz. Building Information Modeling and Digital Twins for Functional and Technical Design of Smart Buildings with Distributed IoT Networks—Review and New Challenges Discussion. *Future Internet*, 16(7):225, July 2024. Number: 7 Publisher: Multidisciplinary Digital Publishing Institute.

[136] Wenpeng Wang, Victor A. Leal Sobral, Md Fazlay Rabbi Masum Billah, Nurani Saoda, Nabeel Nasir, and Bradford Campbell. Low power but high energy: The looming costs of billions of smart devices. *SIGENERGY Energy Inform. Rev.*, 3(3):10–14, oct 2023.

[137] Global Water. Water instrumentation for environmental monitoring. `https://www.ysi.com/products/global-water`. Accessed: 2022-02-22.

[138] Chengjie Zhang, Affan Syed, Young Cho, and John Heidemann. Steam-powered sensing. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, SenSys '11, page 204–217, New York, NY, USA, 2011. Association for Computing Machinery.

[139] Long Zhao, Igor Brandao Machado Matsuo, Yuhao Zhou, and Wei-Jen Lee. Design of an Industrial IoT-Based Monitoring System for Power Substations. *IEEE Transactions on Industry Applications*, 55(6):5666–5674, November 2019. Conference Name: IEEE Transactions on Industry Applications.