Managing Energy-Harvesting Environmental Monitoring Systems

by

Asher G. Watts

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Software Engineering and Intelligent Systems

Department of Electrical and Computer Engineering
University of Alberta

Abstract

Data collection is difficult in remote locations due to limited access and scarce resources. To power environmental monitoring equipment, and allow it to operate independently of maintenance and infrastructure, energy can be harvested from the environment; however, this makes the system dependent upon its environment and requires energy management to maintain performance. The objective of energy management in environmental monitoring is to produce the highest quality data set possible. These devices require simple, robust control, so a fuzzy inference system is used to produce a controller that maps device states to actions. To ensure that the fuzzy controller selects the best actions for each state, it is tuned by expert knowledge and genetic optimization. The simulation results from the Arctic and Boreal regions both show that these controllers allow energy consumption to be matched to the local energy profile, which improves performance over operating at a fixed level of energy consumption. By reducing the rate of data capture when energy is scarce, the monitor prevents failure and conserves energy. When energy is plentiful, the rate of data capture was increased to acquire high quality data. Managing energy and data storage allows the control system to delay energy-intensive operations, like wireless transmission, until environmental conditions became favourable. By combining energy management with suitable storage technologies, the vulnerability of the monitoring system is reduced to the point where storage can compensate for environmental energy scarcity and a suitable level of performance can be guaranteed for a deployment period of a few years.

Acknowledgements

I thank my supervisors, Dr. Musilek and Mr. Wyard-Scott, for their support and guidance throughout this degree; you made this time in academia possible.

I am grateful to Dr. Musilek and the CEOS group for providing opportunities to travel abroad. These experiences were unlike anything I had done before.

Dr. Khabbazian, Dr. Joseph, and Mr. Wyard-Scott all provided positions for marking and teaching here at the university. Thank you all for the chance to see university from this perspective, and the support these opportunities provided.

The fine folks I met in the FACIA lab, and those I encountered during teaching, have enriched my time at the university.

Finally, I thank my family, friends, and roommates for putting up with me during graduate school.

This research was supported by the Queen Elizabeth II Graduate Scholarship, in addition to research and teaching assistantships.

Contents

List of Tables

List of Figures

Symbols and Abbreviations

MAX_{OP}	Maximum Non-Redundant Operational Level
MIN_{OP}	Minimum Acceptable Operational Level
ACIS	Agro-Climatic Information Service
AGDM	Alberta Agriculture and Rural Development
AWS	Arctic Weather Station
DFS	Dynamic Frequency Scaling
DVFS	Dynamic Voltage and Frequency Scaling
DVS	Dynamic Voltage Scaling
EC	Environment Canada
EHS	Energy Harvesting System
EM	Energy Management
EMEND	Ecosystem Management Emulating Natural Disturbance
FIS	Fuzzy Inference System
FRBS	Fuzzy Rule-Based System

HAPM	Harvester-Aware Power Management
IFS	Inuvik Fuzzy Strategy
MCU	Microcontrol Unit
MPPT	Maximum Power Point Tracking
PAR	Photosynthetically Active Radiation
PM	Power Management
RBS	Rule Based System
SBE	Simulator Back End
SD	Secure Digital
SFE	Simulator Front End
SOC	State of Charge
UOD	Universe of Discourse
WSN	Wireless Sensor Networks

Chapter 1

Introduction

1.1 Motivation

Many fields of science require data to analyse the environment. This requirement for data motivates environmental monitoring, which is concerned with observing the world, its creatures and systems, to conclude how they are, how they change, and how they interact. This broad range of data collection is critical to the public, government, industry, and future generations. Monitoring informs resource management, habitat conservation, and natural disaster mitigation. Technology has enabled humanity to impact the environment as never before; however, modification and pollution of the environment will create serious problems that cannot be approached without the data from monitoring. To determine the human impact on the environment and to separate this from naturally occurring changes, a significant amount of data must be collected, processed, and disseminated in an efficient and organized fashion.

Environmental issues that affect the whole planet, like changing climates and rising oceans, require consistently accurate, long-term data collection distributed over vast areas [52, 92]. Monitoring with the temporal and spatial scope required to investigate global issues was not possible until recently. Short-term, spatially-localized environmental studies, which have succeeded in the past, are not useful at this level [55]. Global environmental trends have always been important, but there had never been a way to examine them before now. With this in mind, how can data of this volume and scope be collected and interpreted?

It is a feat of data capture to collect high quality data at the spatial and temporal scope required, with appropriate resolution, for detailed analysis of a whole environment [8, 39]. Historically, environmental monitoring has relied upon manual data collection. The process was subject to the human limitations of trained personnel. Humans are slow, expensive, non-durable, and collect data of widely varying quality [58]. An individual collecting data throughout a region must take sequential measurements and must travel between sites; spatial distribution imposes travel time, which seriously limits sampling frequency and data rate. Extreme or hazardous conditions prevent human access, which interrupts or completely eliminates data collection. In addition to the expense of deploying and coordinating personnel, the number of samples and sampling locations at the human scale may not provide the level of temporal or spatial density required to make acceptable conclusions from the data [20, 58]. Human resources for this style of monitoring are limited; not even the “army of grad students” approach [68], deploying vast amounts of cheap labour, will solve this problem. With limited budgets and human resources, even a well-designed study may not be able to produce data of sufficient quality [39]. Alternately, manual data collection is versatile and robust: sensors and equipment can be calibrated or repaired in the field and qualitative data is naturally collected.

Automated remote environmental monitoring is growing in popularity as electronic devices and communication networks become more sophisticated [58]. It eliminates many problems associated with manual techniques but gives up the versatility and flexibility provided by a human. Automated monitoring stations are limited to modern sensor technology and can only perform specific measurement and communication tasks within their design. That said, they can be employed to do this limited scope of work exceptionally efficiently. Modern electronics and communication technologies permit low power solutions to be deployed for a reasonable cost, in large numbers, over a sizeable area, all while remaining networked and wirelessly accessible [57]. Stations are expendable, at least in comparison to human life, and can be deployed for long periods of time in hostile or dangerous environments. Once deployed, manual collection cannot compete with the spatial and temporal data resolution provided by modern sensing technology [39, 57, 68]. Automated monitoring is capable of a far greater scope than manual methods: weather stations can stand for years in the same area, while a satellite can see huge sections of land at once [44,53]. As long as there is a specific set of quantitative environmental variables to be recorded, automation is generally exceptional.

Automated remote environmental monitoring systems have many limitations. Deployment and maintenance both require human intervention, so automation does not necessitate autonomy. These problems cannot, currently, be solved or managed by the monitoring equipment. Alternately, the monitor's energy and data resources are partially under its control. Because these systems are dependent upon an energy source, energy management has become an important consideration, particularly when energy harvesting technologies are included. The presence of a limited environmental power source changes the focus of the monitoring device from minimizing energy consumption and maximizing device longevity, to optimizing performance and allocating resources. Low power electronics and advances in microcomputing have provided an excellent platform for investigating the field of energy management for remote environmental monitoring [35, 57, 68, 78].

1.2 Objective

The objective of this thesis is to investigate and address some of the challenges present in remote environmental monitoring systems. The two main difficulties considered are energy scarcity and limited access. The intent is to show that energy management and appropriate system design can guarantee a high level of performance between the extended maintenance periods of the device. This facilitates the ultimate goal of environmental monitoring: the collection of high quality data.

The two main tools used in this approach are software engineering and intelligent systems. Simulations permit energy management to be investigated in detail without the substantial cost and delays associated with deployment, while computational intelligence is involved with the design of control systems for these devices.

This objective is completed in three parts:

- A thorough summary of background information on environmental monitoring and energy harvesting is provided. This includes the objectives and challenges of monitoring along with the fundamentals of energy management theory. This meets the objective by supporting the rest of the investigation with materials and references for further research.

- The software simulation tools used in this thesis have proven effective for modelling hardware and control system performance. Additionally, the simulation allows the fuzzy controllers used here to be efficiently tested. These tools assist with the study and design of environmental monitoring systems.

- The results of simulations based on environmental data from different regions provides an opportunity for analysis of energy management techniques and system design. These results confirm their effectiveness and show how important simple design choices are at eliminating failure and guaranteeing performance, while addressing the challenges of environmental monitoring.

Ultimately, the objective of remote environmental monitoring is the production of high quality data [72, 93, 95].

1.3 Organization

This thesis is organized into eight sections, including this one. The first three chapters provide background information on energy management for environmental monitoring. The central two chapters describe the simulations used to investigate energy management and the guiding principles of their development. The final chapters explain applied energy management strategies and discuss their results.

Chapter 2 provides a general discussion of background content for the thesis. It is a collection of supporting information for the rest of the document.

This chapter includes an introduction to environmental monitoring, a glance at energy storage and harvesting hardware, and a brief overview of two computational intelligence techniques. An introduction to monitoring techniques provides context for automated remote environmental monitoring. Typical hardware involved in energy management is summarized. Finally, fuzzy controls and genetic optimization are outlined so that they can be freely discussed later.

Chapter 3 provides a detailed look at energy management for wireless remote monitoring. The chapter outlines monitoring limitations along with common management strategies. This includes a discussion of current theory and how it relates to system design, management, and implementation.

Chapter 4 begins with a description of the physical hardware modelled later in the chapter. The simulations for both the Arctic Weather Station and the SolarNode are broken down by module and explained in detail. This chapter looks at how these platforms are modelled in software and how they implement energy management strategies. The SolarNode simulation, presented in detail, is in line with common control and intelligent systems methodologies.

Chapter 5 begins with a discussion of issues involving the exchange of energy for data. This includes the collection, storage, and value, of energy and data. These ideas are framed within the context of the simulation. This is followed by an explanation of exchange rate and its management.

Chapter 6 explains how computational techniques are applied to the problem of energy management. First, the fuzzy control framework for a remote monitor is described, then multiple applied examples are discussed. Finally, the genetic optimization technique for shaping the fuzzy controller into a useful and adaptable energy management strategy is explained.

Chapter 7 uses data from remote regions around the globe to test the simulations. This is intended to demonstrate the versatility of energy management and show that these strategies are adaptable. The first set of simulations show the importance of environmental adaptation in energy management. The second set of simulations examine the effect of component sizing and resource storage. Finally, system improvements and fault prevention are demonstrated by additional simulations. The results of these simulations are discussed with respect to the computational methods used to construct them, as well as the energy management concepts they highlight.

The final, closing segment of the thesis is Chapter 8. This summarizes the work of the thesis and states how it fits within the field. An analysis of how the work here could be built upon is added for future consideration.

Various appendices support the document. This includes code to build a sample fuzzy control surface and a set of example computations. Contact information for the research group is provided so that all code and simulations can be requested for further research or scrutiny. Finally, a brief description of some performance metrics is included to supplement Chapter 7.

Chapter 2

Background

2.1 Environmental Monitoring

2.1.1 Environmental Data Collection

Environmental data collection is a diverse and constantly changing field; this thesis only deals with a small part of it. To distinguish the content of this thesis, environmental monitoring is divided into several pairs of groups. There are exceptions and grey areas, but strict binary separation highlights specific experimental properties, as shown in Figure 2.1.

Remote and *accessible* environmental monitors are differentiated by proximity to infrastructure. An *accessible monitor* is able to utilize external power and data connections or is conveniently located for device maintenance. A *remote monitor* does not have this support and requires more reliability, robustness, and autonomy, which makes data collection challenging.

The data collection process can be separated based on its level of automation. An *automated process* employs electronics and sensors to capture quantitative data; this permits data collection at high frequencies and a high degree

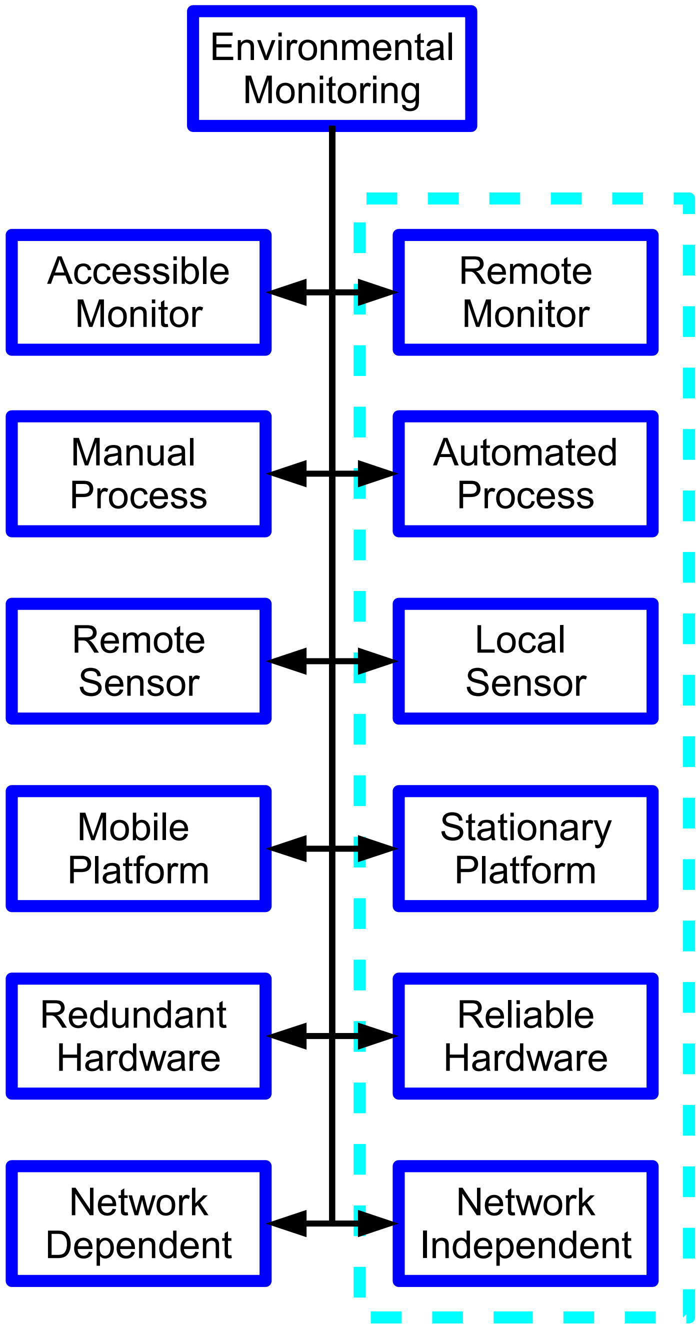


Figure 2.1: This diagram divides various environmental monitoring techniques into two sets. This shows that the monitoring methods discussed in this document are a small part of a much larger field. The outlined box holds the categories which are relevant to this thesis.

of repeatability. Automating data collection reduces versatility: an *automated process* must be reconfigured, if not entirely redesigned, to modify its function. A *manual process* for data collection is limited by human speed and subject to human error. Trained personnel can adapt to field conditions and modify their measurement techniques; they have a natural advantage when it comes to qualitative assessment.

Sensing can be classified based upon where the sensor is in relation to the phenomena it is capturing. In-situ sensing, using a *local sensor*, occurs when the sensor is near or in the measurable phenomena. A thermistor, for example, is in-situ, because it directly captures changes in temperature. Ex-situ sensing, commonly called remote sensing, involves investigating the measurable phenomena at a distance with a *remote sensor*. Satellite images are an example of this type of monitoring; the capture device is located in space, and is isolated from the environment it monitors. Remote sensing can have exceptional spatial scope to its data, particularly when satellites are involved. In-situ sensing measures a local phenomena with limited spatial scope.

The monitoring platform may be *mobile* or *stationary*. A *stationary platform* provides a consistent location for data capture, while a *mobile platform* can change its location. The division between these two areas is not determined by whether the sensor or sensor platform is moving, but rather if it moves through, or with respect to, the phenomena it is attempting to monitor. If a *stationary platform* is attached to some mobile data source, it will be considered stationary in this work as long as the mobile entity is being monitored rather than its surroundings. Under this condition, the monitor and the subject are not in motion with respect to each other.

With advances in low power electronics and wireless technologies, automated sensing can be networked with relative ease. Manual sensing can also be networked, in a sense, by coordinating groups of personnel, but that is not considered. Networking can be local, where sensing units, personnel or platforms, communicate within their environment; this is common in *Wireless Sensor Networks* (WSN), a field currently bustling with research activity [3]. Some platforms are so *network dependent*, they are individually useless. Networking may also connect an ex-situ observer to an in-situ sensor. The level of network dependency is not determined by how active the monitoring platform is within the network, but by how functional it will be if network access is limited.

The cost and expendability of sensing choices is also important. Expensive sensing technologies are difficult to spread out and network based on cost alone. Increasing the number of sensing platforms may increase the redundancy of each unit in the network. This can create a redundancy-reliability trade-off: many cheap sensors may be able to compensate for their limitations by producing many data points. *Redundant hardware* can tolerate losses, while *reliable hardware* invests in avoiding failure all together. Cheap sensors are easily replaced and focus on redundancy, while expensive devices focus on reliability [3, 69].

Figure 2.1 shows how these pairs of groups are combined to limit the scope of this work. While other areas are not directly considered, this work is still applicable, to some degree. This work specifically focuses on automated, in-situ, stationary sensing platforms. The platforms are reliable enough that they are intended to provide complete data sets for each deployment location, but are not so expensive that loss or failure is disastrous. It is assumed that they have some limited network access but do not require it to function. They may share data with an ex-situ observer, and between themselves at some level, if necessary, but are fully independent.

2.1.2 Monitoring Applications

Opportunities for environmental monitoring systems are rapidly expanding. Sensing platforms become more viable technologies as they become smaller, cheaper, and easier to network. Recent developments have been so successful that an amusing new term has appeared to express their unprecedented data production: the "data deluge" [25, 69].

The diversity of available sensing technologies has enabled a wide variety of applications. Automated sensing can monitor things that are too dangerous, too subtle, or too boring for a person to accurately measure. Applications in harsh environments, like the high Arctic or active volcanoes, benefit from the robustness and expendability of automated monitoring systems [13]. Precision sensing can capture gas fluxes at rates and precisions impossible for a person to match [25]. Alternately, weather stations can capture climatic data at precise intervals continuously for years [13]. The number of environmental variables that can be measured by sensing devices is continuously expanding. Many examples are provided in [5], [13], [25], and [86] which will not be repeated or discussed here.

Remote monitoring projects with applications similar to those mentioned in this thesis are the Enviro-Net project, cryosphere climate monitoring, and Arctic weather stations [8, 52, 57, 78]. The Enviro-Net project uses wireless sensor networks to collect a variety of environmental variables such as leaf temperature, *photosynthetically active radiation* (PAR), soil temperature, and humidity [57, 78]. Cryosphere climate monitoring analyses the impact of the Arctic on the rest of the planet and how it will evolve due to climate change. They share monitoring resources to capture a broad spectrum of environmental variables over the region [8, 52]. Additionally, routine monitoring of weather patterns throughout the Arctic requires high-reliability weather stations to provide a large spatial perspective [64]. The harsh conditions of these remote environments encouraged the adoption of automated sensing techniques. Narrowing the scope of applications down to this small subset is intended to set the stage for this document.

2.2 Energy Considerations

Energy management in environmental monitoring is critical to success, so basic concepts in both energy harvesting and energy storage are discussed. These ideas are introduced here so they may be used without explanation later.

2.2.1 Energy Harvesting

Energy harvesting is the only way for a remote device to replenish its energy without receiving maintenance. Novel technologies are constantly being developed and vary in collection technique, efficiency, and power output. From the perspective of this document, the monitor's collection methods are less important than the total power generated by all attached harvesting technologies. These technologies are briefly outlined.

Solar Energy

Many environmental systems are driven by sunlight, so solar energy is a natural choice to power the devices that monitor them [72]. Solar energy harvesting is accessible, successfully commercialized, and provides good power density for microelectronic devices. Other sources of energy harvesting are useful but typically less popular.

The availability of solar energy is dependent upon a location's position relative to the sun. A solar panel perpendicular to the sun's incoming energy collects the most direct radiation; other angles collect less depending upon how far they are from this alignment. The angle of the sun with respect the earth's surface changes with the day, the season, and the distance from the equator. Diurnal solar cycles are related to the earth's rotation, while seasonal changes are due to the earth's angular tilt as it orbits the sun. Seasonal changes act like an envelope signal to the daily cycles. The farther a point is from the equator, the larger this seasonal effect is. Environments close to the equator experience less variation in their seasonal solar energy cycles than places near the poles [76]. Above the Arctic Circle, the sun will not appear for a few months in the winter while providing continuous daylight throughout a part of the summer: these are Polar Night and Polar Day, respectively. With a time, date, and position, the energy harvest of a solar panel can be predicted [11,63]. A stationary panel has no control over this harvest, but its output power signal can be used to learn future solar resource availability [35].

Photovoltaic solar panels use semiconductor technology to convert radiant energy from the sun to electrical energy. Photons transfer their energy to charge carriers as they are absorbed by the material of the solar cell [38]. This produces something similar to a voltage-limited current source, where

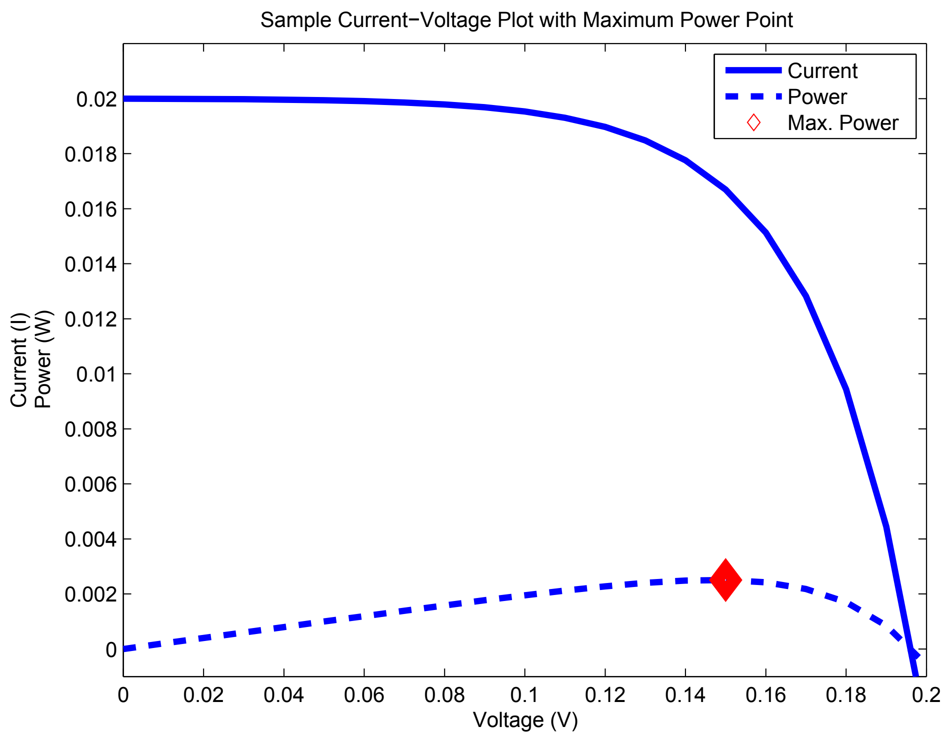


Figure 2.2: This is an example plot of the current-voltage values of a solar cell, with the maximum power point indicated by a diamond. The photocell produces no power in the short-circuit current or open-circuit voltage conditions. Maximum power is produced at the "knee" of the plot where $V * I$ is maximized. The assumed values which were used to make this plot are available in Table 2.1 and Equation 2.1.

the excited charge carriers are driven into the load with some voltage potential. Because environmental conditions and the load's operating level change regularly, electronics using solar panels should regulate current consumption and voltage levels to ensure than the maximum power is drawn from the solar panel [75]: this is called *Maximum Power Point Tracking* (MPPT). Figure 2.2 shows how current, voltage, and power output are related. Equation 2.1 was used to produce this figure using data from Table 2.1.

Table 2.1: Example variables used to generate Figure 2.2 based on Equation 2.1 taken from [38]

Variable:	Fixed Value	Typical Units	Description
I	N/A	Ampere	Total solar cell current
V	N/A	Volt	Voltage (Light dependent)
I_{ph}	20 mA	Ampere	Photocurrent
I_o	20 µA	Ampere	Reverse saturation current
$e/k_B * T$	25.85 mV	Volt	Thermal voltage at 300K
n	1	Constant	Material ideality factor
Current at Max. Power	16.7 mA	Ampere	Current of Max. Power Point
Voltage at Max. Power	0.15 V	Voltage	Voltage of Max. Power Point
Max. Power	2.5 mW	Watt	Max. Power Point

$$I = -I_{ph} + I_o \left[exp\left(\frac{eV}{nk_BT} \right) - 1 \right] \tag{2.1}$$

There are two main constants used to characterize a solar panel: the short circuit current, I_{sc}, and the open circuit voltage, V_{oc} [38]. The plot in Figure 2.2, shows that the current supplied by an illuminated cell is nearly constant as the operating voltage, V, increases towards the open circuit voltage. The current then quickly drops off and reaches zero at V_{oc}. Because power is $P = IV$, the point on the curve very near the maximum voltage, just before the operating current drops off, is where the maximum amount of power can be harvested. The harvested current is driven by illumination, while the operating voltage is set by the load, so the ideal operating point is constantly changing; solar cell charge controllers are typically equipped with some form of MPPT. A more detailed analysis of solar cell operation can be found in [38].

The inherent efficiency of a solar cell is dependent upon manufacturing technique, semiconductor composition, and surface structure [38]. For a current look at solar cell performance, see [49] and [50]. Many different crystalline semiconductors can be used to capture photons for electricity, and each combination has different properties [38]; however, from the perspective of energy harvesting, this is not directly significant. Environmental monitoring considerations include how much power the solar cell can produce, what conditions it can tolerate, and how expensive it is.

The convenience of this technology has lead to widespread adoption of energy harvesting for remote monitoring systems. Solar panels become more prevalent as they improve with their commercialization: costs fall and efficiencies improve, while they remain convenient and durable. Solar panels have no moving parts to be inhibited by debris and can be mounted securely in different positions to avoid being dislodged by wind or covered by precipitation [64]. Solar panels are an excellent choice for the power requirements of microelectronics.

Some examples of environmental and ecological monitoring systems which have successfully implemented solar energy harvesting are [86]: Prometheus [33], Ambimax [54], Heliomote [75], Everlast [82], Sunflower [85], and Hydrowatch [88]. These represent a wide variety of energy harvesting and storage configurations, as well as a range of power requirements. These applications are not explored here, but are useful references for further research in the future. A wireless environmental monitoring device that harvests solar energy is discussed in [71], [72], and [95], and outlined here in Section 4.1.2.

Other Energy Sources

Energy harvesting technologies are constantly diversifying [86]. While radiant energy harvesting is dominant, mechanical and thermoelectric technologies are becoming more common [75, 86]. Mechanical energy is usually harvested by rotating or oscillating a power transducer, but moving parts can be a disadvantage. Energy from the wind can be captured using miniaturized turbines or vibrating membranes [54, 67]. Even the kinetic energy of raindrops can be collected using vibrations [24]. No moving parts are required to exploit thermoelectric gradients to capture energy [75,86]. However, the unifying problem of all these options at this power level, is their poor energy density and underdeveloped commercialization. Compared to solar energy, their disadvantages make them secondary choices unless a particular niche application finds them exceptionally convenient.

These technologies have one key advantage: diversity. While they may not be the first choice for many applications, selecting multiple harvesting technologies distributes energy collection between different mediums and helps improve system reliability. When the sun is down, the wind or temperature gradients may still be exploited. This provides redundant energy sources that prevent a system from depending entirely upon storage when solar energy is not available.

The energy production of novel environmental sources may be more difficult to predict than solar energy. This may be due to limited research and modelling, or a characteristic of the source itself. Some locations have better wind resources than others, but this does not mean that the wind blows more predictably, only that on average it produces more energy. Novel harvesting methods are a suitable supplement for, but inadequate replacement of, solar energy. By increasing the number of harvesting elements, the system has more potential points of failure than before, particularly given the developmental advantage solar energy has over other sources; the other sources are useful, but must be added carefully. Also, each of these new technologies has different characteristics than solar energy, and thus may require additional hardware to ensure they operate at their maximum power point [54]. MPPT circuitry for solar energy is well developed, but other technologies need time to catch up [67]. The new source of energy must be worth the increase in system complexity, management overhead, and risk of failure.

2.2.2 Energy Storage

Energy storage is integral to energy management. While some environmental monitoring systems can operate independent of energy storage, they are missing a key strategic opportunity [86]. Namely, the monitoring platform does not need to consume less energy than the harvester can provide at any time, but rather less energy *on average* than the harvester can provide within the bounds of the storage device [35]. This important difference is critical to Chapter 3. For now, only the storage medium and methodology are addressed.

Primary and Secondary Storage

For this document, energy storage technologies will be loosely divided into two categories: primary storage and secondary storage. Primary storage cannot be replenished, at least without maintenance, while secondary storage can be recharged. Storage technologies often do not fit directly into these two categories. For example, if a device uses rechargeable batteries as secondary storage, but has no ability to recharge them, are they secondary or primary storage? Maintenance allows them to be recharged, but from the perspective of the device, they are no different than primary batteries [71]. Within this application, they are divided into two groups based upon their implementation from the perspective of the monitoring system. Later in this document, primary energy storage will be referred to as the *energy reserve*, while secondary energy storage will be referred to as the *energy buffer*. This avoids confusion with battery-specific terminology and is important to Section 4.1, which discusses hardware platforms.

Primary energy storage is simple and disposable. Because it cannot be recharged, this medium must start with the maximum energy it will require for the application. For electrical generators, fuel is also a form of primary energy storage because refuelling is a form of maintenance. Primary storage makes an excellent energy supplement when other resources are unavailable; however, the finite supply in primary storage guarantees the device will eventually fail [35]. If the time until failure is longer than the regular maintenance cycle of the monitoring platform, primary storage is a viable solution [95]. If the power consumption of the device is small relative to the energy reserve of primary storage, the simplicity and convenience of this medium is difficult to compete with. This is ideal for short-term, low-power monitoring applications.

Energy can pass in and out of secondary energy storage without serious restriction. Unlike primary storage, which must hold the total energy required before deployment, secondary storage can hold energy it receives during operation. However, secondary storage may require different recharging techniques that increase the complexity and possible failure points of the device. Variations in environmental energy can be buffered to a degree as long as the storage device is sized appropriately. This allows an attached device to operate at a consistent level of performance [93]. Secondary storage has an averaging and filtering effect on incoming power and allows energy to be passed between the storage, harvest, and load modules of the monitoring system [93–95]. This can be a complex process that makes energy management necessary to ensuring good performance.

Batteries and Ultracapacitors

Environmental monitoring devices have two options for storing electrical energy: batteries and capacitors [75]. While batteries have been common in electronics for a long time, ultracapacitors have only recently developed enough to compete. Batteries are a form of chemical storage, while capacitors hold charge as an electric field in their layered construction [47, 48].

Primary and secondary battery technologies are divided by whether their cell chemistry allows them to be safely recharged. Primary batteries are rendered unusable after consumption and should be recycled. Secondary batteries are the focus here, with two cell chemistries, lead-acid and lithium, of particular interest. Lead acid batteries are a cheap, durable, established technology and this has made them ubiquitous [7, 51]. Unfortunately, they are heavy and vulnerable to environmental conditions [64]. In comparison, relatively new lithium batteries have much better energy density and are more tolerant to environmental conditions. Unfortunately lithium batteries are more expensive and complicated to work with than their lead-based counter parts. Nickel metal hydride batteries are also used in environmental sensing, as seen in [75] and [34], but seem to be losing ground as lithium batteries continue to develop. When it comes to battery selection, it is important to consider cost, energy capacity, energy density, charging complexity, and environment tolerance.

High performance capacitors, interchangeably called supercapacitors or ultracapcitors, have become more popular as their technology improves. They have a massive surface area for their volume and as such have excellent power density: far better than that of a battery [46]. Their internal construction does not undergo chemical changes like a battery, so they have a much longer lifespan. Though these devices have a problem with leakage, it is less serious than in earlier devices. Electrical charge needs time to distribute itself through the volume of the device's large surface area, which results in a voltage drop early after charging [9, 96]. This is a problem for applications which need to recharge quickly, but does not affect those with periodic low-intensity charge cycles like energy harvesting in environmental monitoring. Ultracapacitors have advanced to the point where the have an advantage in durability, lifetime, and predictability [9]. Some monitoring platforms are discarding batteries entirely on the grounds of their poor cycle life [82].

Ultracapacitors are more commonly used as an energy supplement to eliminate short-term cycling of the battery [86]. This simultaneously improves the lifetime of the battery, while keeping its sizeable energy reserve available for shortfalls in energy harvest. Such a device is outlined in Section 4.1.2.

2.2.3 Hardware Topologies

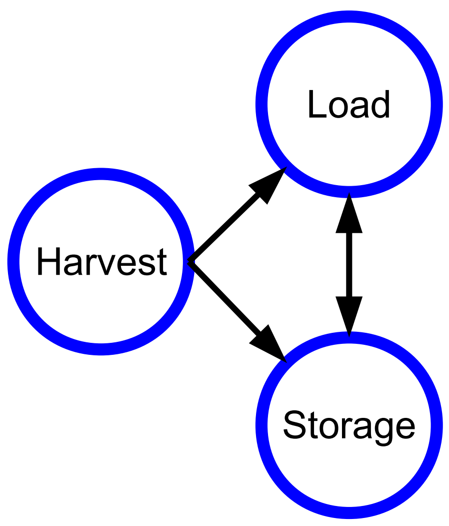
The energy system topology is fixed by the initial design process and determines what management techniques are available to the platform; it dictates how energy passes from the source to the load. Energy storage must also be designed well in advance of deployment, but is more flexible in terms of capacity and configuration.

Energy System

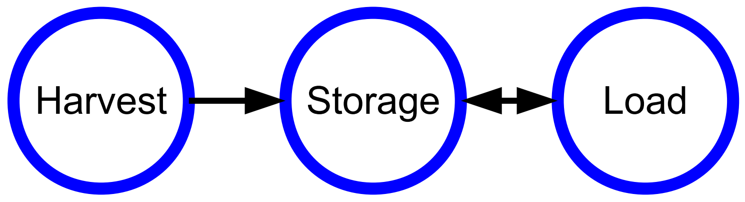
The energy system of a device manages the collection, storage, and internal distribution of power for consumption. From an energy perspective, most platforms can be broken down into processes: energy collection, energy consumption, and energy storage. The harvest module collects energy, the load module consumes energy, and the energy storage module stockpiles and redistributes this energy as necessary. Some platforms avoid storage completely and directly connect the harvest and load modules together. This is not covered here, as it does not encourage energy management.

The topology shown in Figure 2.3a has the three modules connected in a triangular, or ring, formation. The storage and load modules are in parallel from the perspective of the harvest module. Energy from the harvest module is directly accessible to the load, and surplus energy is passed to storage. If less energy is harvested than the load requires to operate, storage supplements the load's demand. Efficiency losses from power conversion and storage leakage are not imposed on all harvested energy, which may provide an efficiency advantage. However, this configuration enforces design constraints on what harvesting technologies can be used based on load voltage level requirements. Coordinating power switching and energy management between the modules also increases design complexity. This configuration is implemented by the *Arctic Weather Station* (AWS) discussed in Section 4.2.1 [93].

The configuration shown in Figure 2.3b is linear. The energy storage and load modules are connected in series and may exchange energy. All energy consumed by the load must first pass through storage. Because there is only



(a) Harvested energy is shared between modules.



(b) Harvested energy is passed directly to storage.

Figure 2.3: These two energy system topologies show how the energy harvest, energy storage, and load modules, share energy. Figure 2.3a has the three modules connected in a triangle so that the load and energy storage share power between each other. Harvested energy is provided to the load and surplus energy is stored. This removes some efficiency losses but increases system complexity. Figure 2.3b, where all three modules are connected sequentially, requires all harvested energy to pass through the storage module. This consistently provides the filtering and averaging effect of storage but imposes losses due to leakage and power conversion on all energy consumption.

one path for energy through the system, it is simple to design and has fewer points of failure. All of the modules may operate at different voltages because level conversion hardware will need to be placed between them to ensure that the harvester supplies energy at its maximum power point and the load can operate at a consistent level. This topology enforces losses on the system: energy storage leakage and power converter losses are the consequences of this simple design. This configuration is partially implemented by the hardware platform discussed in Section 4.1.2 [71, 72, 95].

Tiered Storage

The two different classes of energy storage from Section 2.2.2, the buffer and the reserve, can be arranged into layers. A single layer is the simplest to implement but puts the entire burden for a reliable energy supply on one storage technology. A single energy reserve is guaranteed to fail once exhausted, while a single buffer may be unreliable. Increasing the number of layers, or *tiers*, of energy storage increases the reliability of the platform by increasing redundancy in the energy system.

Tiered storage is more complex than a single layer, but allows energy buffers and energy reserves to work together. Different levels of energy buffers may be paired to compensate for each others weaknesses. For example, rechargeable lithium batteries are a cost-effective solution for holding a sizeable amount of energy for a decent price; however, batteries wear out after repeated cycling. Supercapacitors have excellent cycling properties but are ineffective at holding charge for a long time. Pairing these two technologies allows the supercapacitor to absorb short-term energy changes, while the batteries handle long-term energy surpluses and shortfalls [86]. An example of this configuration is the Prometheus mote from [33]. Chapter 7 shows this is a formidable advantage over single-layer systems. The complexity of having two different storage technologies, each with their own charging requirements, may not be worth the cost or design difficulty; instead, a simple reserve can be installed which provides the required additional energy security, but it must be replaced at regular intervals within its lifespan [71].

2.3 Computational Techniques

The energy management strategies discussed later in this document require background information in two areas of computational intelligence: fuzzy systems and genetic algorithms. Fuzzy systems were used to construct the control scheme for operation in real time simulation. The genetic algorithm was used to evolve a suitable set of parameters for the fuzzy controller based upon simulated performance.

2.3.1 Fuzzy Sets, Rules, and Operations

Fuzzy systems began with the work of Zadeh in 1966 [97]. The field was further explored by the work of Mamdani [40, 41], Takagi, and Sugeno [87]. Where binary systems assign the properties of *true* or *false* to variables, fuzzy systems instead assign variables some degree of truth. An item associated with a binary set is either entirely included or entirely excluded, with no possible ambiguity. For fuzzy sets, items have some degree of *membership* within a fuzzy set, and are not divided between only two membership conditions [45]. In fact, binary logic is a subset of fuzzy logic.

Fuzzy sets are usually used to describe some property: a linguistic variable. For example, consider a battery. The battery may have a “LOW”, “MEDIUM”, or “HIGH” level of charge. For convenience, it will be represented in percent. These three descriptors are used to build three fuzzy sets, while the range of all possible levels of charge is said to be the *universe of discourse* (UOD). All points within the UOD are assigned membership in the fuzzy sets based upon how well they are described by each of the three linguistic variables.

These fuzzy sets can interact with each other in a manner similar to binary sets. The four basic operations are negation (or complement), inclusion (or containment), union, and intersection. These are established in [97] and described in [40], with further details are available in [61] and [62]. Fuzzy relations “capture the associations” among fuzzy sets and can use information from multiple UODs [61]. Fuzzy implication is used to extend knowledge onto a fuzzy set.

Fuzzy inference has two parts, an antecedent and a consequent, that make up the premise and consequence, respectively [81, 87]. All fuzzy sets used as variables in the antecedent are associated using fuzzy relations and then the implication process is used to infer the consequent. Usually, the implication is an intersection operation, or triangular norm, between the related antecedents and the consequent [31,62]. The knowledge from the relation is implicated onto the set, or relation, which makes up the consequent. The inference process is often formalized into linguistic fuzzy rules which are then used for making decisions or controlling processes [40].

2.3.2 Fuzzy Control

The fuzzy control methods used in Chapter 6, employ "fuzzy rules" as discussed in Section 2.3.1. The *fuzzy inference system* (FIS) and *fuzzy rule-based system* (FRBS) are two names for this technique [31, 40, 87]. Rules are generally presented in the "*IF* x is A *THEN* y is B" form, where "x is A" is the antecedent of the rule and "y is B" is the consequent of the rule, designated by IF and $THEN$ respectively. Fuzzy implication is used to apply the knowledge of the antecedent to the consequent. The x and y variables represent state values, while the A and B variables represent fuzzy sets, in which x and y, respectively, have some degree of membership. The system is split into three stages: *fuzzification*, inference, and *defuzzification* [31].

The *fuzzification* stage takes numerical state values and assigns them some membership in the fuzzy sets involved in the inference stage. The control system is described by a number of different state dimensions; each of these are covered by overlapping fuzzy sets representing linguistic descriptions of the dimension's properties. To return to the example from earlier, the battery charge could be a state variable, because it describes part of the battery system. Suppose it is currently three quarters full. This value would be "fuzzified" by comparing it to the three fuzzy sets mentioned earlier: "LOW", "MEDIUM", and "HIGH". The current state of the battery would be excluded from the "LOW" set, a membership of zero, but would have some membership in both the "MEDIUM" and "HIGH" sets. The "fuzzification" process has shifted the knowledge in this state dimension from a rigid numerical description to the three fuzzy sets. All state information input into the fuzzy controller undergoes this process so that it can be used in fuzzy inference.

The fuzzy inference stage computes the consequent of the rule using fuzzy relations and stored knowledge. [31] describes three parts of the internal computational structure: two are based on knowledge and the other is based on relations. Information about membership of the fuzzy sets and the organization of the fuzzy rules make up the "*knowledge base*", while the relations and inference of the rules is done by the "*decision-making unit*". In addition to the sets included in each rule, the pairing of antecedents and consequents is an important part of the *knowledge base*. This section of the fuzzy controller is the most open to optimization and adjustment. The shape and number of fuzzy sets is easily tuned to improve performance, while selecting an appropriate combination of rules depends upon the demands of the application.

The *defuzzification* stage takes the information from the inference stage, which exists as fuzzy relations built from implications, and aggregates it back into values useful for the control process. The new knowledge from the rule based system is still fuzzy and cannot be sent to an external system. There are many techniques for the aggregation process with different advantages in complexity and accuracy. The *center-of-gravity* technique is common; this takes the sum of the results of the implication process and divides it by the sum of the antecedents [23]. This is like taking the area of an object and dividing it by its height to find an average length. Depending upon the how the consequents are presented, either as individual values or as whole functions, this may become an involved process.

In practice, each state dimension will have some UOD: a range of possible and valid values. The result of the controller for every possible combination of input values for all states can be computed offline. This forms some mapping between the input state information and the output action command. State information is the basis for decision-making and provides the fuzzy controller with some idea of what is happening to the platform. The output of the controller shall be called an action; some control systems consider this output yet another state, but here this approach will not be used. In the same sense that the term *state* refers to both the actual condition of the hardware and the numerical data passed to the controller, so also is *action* used to refer to both the numerical output of the controller and operations that the platform executes under the controller's recommendation. Examples of this type of controller in action are available in [65], [93], [94], and [95]. Additional information on fuzzy modelling and control is available in [16], [59], and [60].

2.3.3 Genetic Algorithms

Intelligent agents run into a trade-off between *exploring* for new knowledge and *exploiting* present knowledge. Exploiting takes advantage of present information to better the state of the agent, while exploring risks short term failure for possible long term advantage. Genetic algorithms are a computational intelligence technique that balances these two strategies. The biological origins of genetic algorithms started with Fraser, Reed, and Bremermann [19]. This was further developed by Holland and his students which brought them near to their present state [22]. The basic concepts of this technique are discussed here.

As suggested by the name, genetic algorithms are based on natural selection and evolution from biology. First of all, genetic algorithms use a *population* of solutions to solve problems. The population consists of a variety of valid, but not necessarily useful, solutions which are described by some set of properties, typically referred to as *genes* or *chromosomes*. This naming convention reflects the algorithm's biological inspiration. The initial population is usually made up of random individual candidate solutions covering the problem space. The population can then be evaluated with respect to the problem criteria. Once again, with a reference to natural selection, the performance of the candidate solutions is referred to as their *fitness* [22]. So far this is the same as randomly solving the problem in parallel several times and comparing the results. At this point, the algorithm then uses its population's fitness to determine which solutions are best at solving the problem, and uses this knowledge to help traverse the space of all possible solutions. The algorithm assumes that candidate solutions are somehow linked: combining properties of two good solutions may produce another, possibly better, solution. This is similar to biological evolution, the result of natural selection [17].

To navigate the problem space, the algorithm relies on two mechanisms: *mutation* and *crossover*. These operations are applied to members of the population in a similar way to how they work in nature. The *mutation* operation explores along a dimension of an individual solution. It involves modifying part of the individual's properties to see how the problem space varies with respect to that point. A small amount of mutation allows local exploration, while a large amount of mutation will cause the individual to jump about the problem space. This can be useful for preventing the individual from getting trapped in a local, suboptimal solution. To improve the fitness of an individual, a small amount of mutation may help good individuals move to better locations in the problem space. The *crossover* operation exploits the fitness of two or more individuals in the population by recombining them. Two or more individuals may swap properties through *genetic recombination*. This creates a new individual with characteristics between the originals in the problem space. This technique allows good individuals to combine their differences to explore for a more desirable solution. This is generally not a local form of exploration, and helps the search move through the problem space. The crossover operation slowly pulls the population together; mutation is required to keep individuals spread out [17].

With these evolutionary mechanisms creating new individuals, the population must be managed to keep the algorithm tractable. This is where the *selection* operation comes in. *Selection* uses the fitness of each solution to determine which are desirable and which should be improved. If the population needs to remain a constant size, then solutions which are not fit are replaced in the search for better ones. The selection process is often randomized with the probability of remaining in the population related to the fitness of the individual. The selection stage divides the population into generations; the algorithm is iterative, so after the exploration and selection phases, where less fit solutions are replaced, the population is in a new generation [22]. If it is imperative that the most fit individual survive to ensure that the best solution is always present, it may be designated an *elite* individual and be exempt from the selection process [17]. In a way, this anchors the population about these *elite* individuals.

Genetic algorithms have been used to optimize parameters in environmental monitoring before. In [64] a genetic algorithm was successfully used to size energy harvesting and storage devices for a simulated *Arctic Weather Station* (AWS). Later, in [93], a genetic algorithm was used to optimize a simple fuzzy rule-based system for controlling an AWS. Unfortunately the document does not go into significant detail on the topic. The application of this technique will be discussed in Chapter 6 and Chapter 7.

Chapter 3

Energy Management

3.1 Importance of Energy Management

Environmental remote monitoring is driven by the demand for data. It is convenient to monitor phenomena around infrastructure: access is simple and energy is readily available. Although data from these locations is easy to obtain, it is of limited use for environmental analysis. To collect accurate environmental data, sensing on location must be taken away from human convenience and that makes monitoring challenging. There are three issues, aside from project cost, which immediately appear: access to monitoring equipment, limited equipment capacity, and required routine maintenance. Energy management and harvesting systems can mitigate these problems, but they have limitations of their own. All of these improvements must be keep at a reasonable cost for projects to remain feasible.

3.1.1 Monitoring Limitations

Environmental monitoring devices must overcome cost, capacity, and reliability limitations. Automating environmental monitoring eliminates the cost and inconsistency of deploying human resources for the task, but transfers the responsibility of monitoring to electronics and energy storage. A system based upon storage has its maximum energy at the beginning of deployment because there is no way for energy to enter the system during operation. The sensor platform's capabilities are limited to those of its processor and the sensor array. These components must also be reliable enough to execute the application without failure.

Storing all the energy required for a long-term monitoring application limits the length of operation to the storage capacity. The capacity, divided by the rate of energy consumption, roughly determines how long the hardware can function. This means extended deployments require massive, reliable storage. This is a problem for both cost and access, because storage is bulky, heavy, and expensive, which complicates transportation to remote areas [65, 66]. If multiple locations need to be monitored, then storage limitations are enough to restrict the scope of the project. Batteries dominate energy storage for low power electronics, but their short lifetimes incur cost and access problems when they are replaced [82]. They are also notorious for low energy density and poor durability [79,82]. Including additional storage or other technologies to provide a margin of safety in the energy system aggravates these issues further [95].

Access to remote locations, like the Canadian Arctic or tropical forests, is difficult and expensive. This limits when and how long human resources have to setup monitoring stations and perform maintenance in the field. This is complicated by the difficulty of transporting materials, due to bulk and weight, and the cost of access, if multiple sites and visits are required [57, 66]. Simply deploying a sensor platform can cost more than the devices themselves [75]. Increasing reliability to reduce device maintenance and failure helps solve the access problem. However, legacy systems that are already installed need solutions that are compatible with their present equipment [66].

These issues are interdependent and there is no way to completely solve the problem. Data which is hard to collect will remain valuable for science and demand innovative monitoring solutions. In Section 3.1.2 there are two advancements that help eliminate the present cost, access, and reliability issues for this application: energy harvesting and energy management.

3.1.2 Monitoring Advancements

This document looks at combating the limitations of environmental monitoring in two ways: energy harvesting and energy management. These solutions increase the complexity of the monitoring platform and may be expensive initially, but legitimize their inclusion by reducing failures and maintenance while improving data quality. This is facilitated by general technological progress in the field of microelectronics, wireless communications, and electrical engineering. The data side of sensing is improved by cheaper microcontrollers, able to execute more processes with less power, and affordable low power wireless communications, which provide access to remote data. Improvements to the energy density of storage and the efficiency of harvesting hardware support the energy systems of modern sensing platforms.

Energy harvesting is an important addition to the field of environmental monitoring. Without harvesting, the energy stored in the monitoring system when it is initially sent to the field limits how long it can operate. For long deployment periods or energy intensive activities, this becomes bulky, heavy, and expensive [65, 66]. Harvesting provides a medium for environmental energy to enter the system, so stored energy does not need to be transported. Equipment cost and storage capacity are a problem when the monitor depends upon storage, but when energy is collected from the environment the basis of the problem changes from capacity to rate [35]. This removes the energy-based time constraints on long-term deployment and reduces the size requirements of storage [35,74,86]. Ideally the upfront cost and complexity of adding energy harvesting to the systems will pay for itself by reducing the cost of storage and maintenance access, while improving the quality of data from the monitor by allowing longer deployment periods with fewer failures.

With all these constraints and complications the remote monitor requires some method of coordinating its actions to invest its limited energy resources as effectively as possible. Energy management organizes resource consumption to improve the efficiency of the monitoring platform. Even without harvesting this reduces the bulk required to store energy for long applications by limiting the amount of energy leaving the system due to waste and inefficient consumption. Energy management also assists the platform's hardware. A poorly managed battery will have a significantly shorter lifespan and much worse performance than a well managed one [7], so energy management also reduces trips for maintenance, like refuelling or battery replacement, that are not feasible from a financial perspective. The addition of harvesting binds the energy system of the monitor to its environment: a variant energy source which must be effectively managed. Finally, the availability of wireless technologies now allows sensor platforms to coordinate, share data, and transmit information to an external user, further eliminating the cost and necessity of field access for data recovery [26,57,64,78]. The advantage of wireless radio comes at the cost of a huge energy investment, which must also be managed. Wireless transmission is so energy intensive it is sole subject of many research projects [10,26]. Considering the increasing complexity of environmental monitoring, due to new technologies like energy harvesting and wireless communication, the importance of management to the monitoring system is clear.

These additions to the environmental monitoring platform increase hardware overhead, either to manage new capabilities or handle the computational requirements of management software. Any technique used to invest resources in real time must be easily solved by the limited faculties of a simple embedded platform [29]. Modern microprocessor-based devices allow more advanced power management to occur [65]; however, processors generally increase in cost and power consumption as they become more capable. A platform's processing resources constrain the complexity of management strategies. Powerful processors can implement more complex software management, but to maintain compatibility with legacy systems, which do not have the benefit of modern low-power microcontrollers, computational simplicity is a necessity [65].

3.1.3 Environmental Power Constraints

The energy system of the monitor is affected by its environment because energy available for harvest is constantly changing and ambient conditions affect hardware operation. Each harvesting system can absorb energy in only a few ways, so if that form of ambient energy is not present, other forms of energy do not have a medium to enter the monitor's energy system. When hospitable, the effect of ambient conditions on hardware components is negligible, but the capabilities and efficiencies of these devices deviate from the norm as conditions become extreme.

Energy management assumes the amount of energy in an environment at a given instant in time is limited, but the total amount of energy is effectively unlimited [35]. This is the difference between environmental power and environmental energy. The power a harvester can capture in any environment is constrained by the energy availability at that time instant and the properties of the harvesting hardware. This is a case of unlimited supply, but limited rate [36]; the total energy harvest from the environment is constrained by how long the harvester is operational in the field. Power will be available even after the device fails to collect it. This increases the maximum length of time the monitor can operate without maintenance from that provided by energy storage, to the critical hardware component with the shortest lifespan [9, 82].

The "inexhaustible supply" [29] of environment energy "ubiquitous to the operating space" [74] of environmental monitoring devices makes energy harvesting a natural solution to their energy problem. It is important to harvest "ambient" energy sources like the heat gradients, wind, or sun light to extend the operational period of the device [86]. While environmental conditions are constantly changing, energy can always be exploited. These energy sources are beyond the control of monitoring devices, but can be predicted or estimated with some degree of accuracy [35,64]. Only energy harvesting techniques based on predictable sources are considered here.

The environment affects the operation of monitoring hardware. Batteries are once again a culprit for varying with their environment, particularly lead-acid batteries, which have become ubiquitous due to their low cost and simple operation. Ambient temperature affects the cell chemistry of batteries, influencing energy storage and device longevity, while also influencing the operation of electronic components, such as the conversion efficiency of solar cells [7,38]. Sensors can be disabled by “high humidity and temperatures” when outside of their typical operating parameters [78]. The durability and ruggedness of the platform itself must withstand interactions with wildlife, precipitation, and debris. The environment sometimes completely disables device operation, like when moving parts are filled with blowing ice, solar panels are covered in snow or foliage, and cables or cases are damaged by wildlife [57,65,78]. Some of these factors cannot be dealt with regardless of real-time management complexity and must be accounted for or eliminated during the design phase. Provided these factors are extreme enough, as in the Arctic, the environment may completely restrict access to personnel as well.

That said, not all environmental factors should be accounted for in simulations; devices must be reasonably robust to survive deployment, but unpredictable accidents should not be modelled. Considering the energy implications of a solar panel covered in snow year round in the Arctic, or a power cable destroyed by local wildlife, will not produce interesting management decisions: if device operation is completely inhibited by random environmental factors design is ineffective and simulation is useless. The only consideration to be given in this circumstance is mitigation. This is the same reason why uncontrollable and unpredictable energy sources are not considered: if there is no way to know about it before hand it is impossible to design for it. The results of modelling and simulation must remain interesting and relevant to the application: the production and management of data.

3.2 Energy Reduction and Management Techniques

A high-efficiency low-power hardware platform is the foundation of a successful remote monitoring system. If the device itself does not effectively invest its energy, it undermines all future energy management. Regardless of how brilliantly software allocates energy resources, the system will always suffer if there have been poor design choices at the hardware level. [74] states, "perhaps the most crucial aspect of sensor network energy optimization is the design of the sensor-node platform itself. Ultimately, it is the sensor-node hardware that consumes the energy, so if the platform itself is not energy-efficient, no amount of higher layer optimization will yield desired results." All attempts at improving energy efficiency are subject to the idea of a "break-even cycle" [5]: the increase in overhead incurred by adding energy management capabilities must result in a decrease in energy consumption of at least the same amount. This should also take into account the increase in system complexity as well; if a new configuration results in no net increase in efficiency but increases the complexity of the system, the change is not advantageous.

Energy conservation techniques and management strategies help meet the demands of the monitoring application while coping with hardware limitations and environmental conditions. They can be roughly divided into two groups: hardware solutions and software solutions.

Hardware solutions are generally concerned with eliminating energy waste. Since the energy consumption of the monitoring platform is effectively set by its hardware, low-power design is a fundamental concept of energy management [13, 74]. Hardware components consume energy in two ways: statically and dynamically. Static consumption occurs any time when electronic components are exposed to a voltage potential. This comes from currents leaking through transistors, resistances, or internal pathways of storage technologies [28, 32, 64, 96]. Even if a device enters a low-power idle or sleep mode, energy is still lost in this fashion. These losses are combated by using lower voltages, which use less energy, or *power gating* hardware modules to ensure that energy is not supplied to unused hardware components [28].

Dynamic losses come from switching digital hardware and other changes in device state. Dynamic power consumption is proportional to the frequency at which devices are switched and to the square of the voltage they are exposed to [32]. This is because transistors hold and dump charge through a small capacitance when they are cycled [28]. Dynamic energy consumption can be reduced by lowering the supply voltage used by the monitoring platform, so less energy is held by capacitances, and by reducing the number of times components are cycled. On the transistor level, low voltage devices cannot cycle as fast as high voltage devices, so there is an incentive to reduce computational speed to reduce dynamic power consumption [28]. Reducing dynamic losses by preventing modules from cycling is called "*clock gating*". Dynamically changing the supply voltage or clock frequency of a processor has resulted in various dynamic voltage and frequency scaling schemes: DVS, DFS, or DVFS, for short [13, 28, 83, 84]. The techniques used to deal with these two sources of power loss can be broadly classified as *waste reduction* or *performance scaling*.

Software solutions are concerned with how to use the tools provided by hardware power reduction techniques. They generally fall into the *performance* *scaling* category. The software level does this in two ways, called *continuous performance scaling* and *discrete performance scaling* for convenience. *Dynamic Voltage and Frequency Scaling* (DVFS) is a continuous technique. In a continuous performance scaling scenario, a monitoring system changes the voltage and frequency of the active hardware to reduce energy consumption *during* operation; this is generally done by software, though it may be done with additional processor-based hardware. The functionality of the device may be incrementally reduced, but not halted; the device may idle if the workload is completed early, but it never completely ceases operation.

The discrete performance scaling techniques operate at a fixed speed for some time, then completely turn off: all operations are halted. This process is commonly called duty cycling. It is simple to implement and requires minimal overhead. To keep power consumption low, tasks are finished quickly, then the platform switches to a low power mode to wait for new tasks. As devices become faster, tasks are executed so fast that low power modes of operation dominate the energy budget of the device and the device's minimum power consumption becomes very important. The minimum energy level of the device is very influential in device longevity. A poor efficiency, low-power state, or "sleep mode", will harm the efficiency of a device as the length of the sleep mode increases. During this time, the static losses, discussed earlier, dominate the power consumption of the device.

Both of these software strategies require some sort of scheduling to distribute the tasks of the monitoring device over its self-imposed computational restrictions. As the functionality of the monitoring device increases, so does the complexity of scheduling. An individual device is easier to manage than a network of devices and network scheduling for low power wireless sensor networks is a field of active research unto itself [10, 26, 84].

Introducing the complexity of energy management creates a problem. The hardware and software required to implement these management techniques must consume significantly less energy than is saved by implementing them [5, 74]. This can be a problem for DVFS solutions, which require more overhead than duty cycling. Devices that have a continuous task, like data processing or managing a network in real time, benefit from DVFS [83], but devices with very simple tasks, where little processing or scheduling is required, do not benefit from the required increase in complexity or cost. Duty Cycling has its own problems as well: transitioning between operational modes may save energy, but requires an energy investment. The device must "break-even" for duty cycling to be worthwhile [5]. Essentially, if the device must switch between its operational and sleeping states too often, or if its sleeping state consumes too much static power, then duty cycling is not viable.

Improving energy efficiency through sensor management is an entire field to itself and mentioned here at a cursory level. To quote [74], "the overall goal of various sensor power-management schemes is energy-aware sensing so that a sensor acquires a measurement sample only if needed, when needed, where needed, and with the right level of fidelity." To manage this from the data acquisition perspective, [74] mentions a choice between sensing with a "scalable fidelity" or selectively disabling power intensive sensors to save energy; these data management techniques reflect the energy management techniques of dynamic voltage and frequency scaling, online performance scaling, and duty cycling, offline performance scaling. These ideas are also mentioned in [5] as "adaptive sensing", which dynamically changes the operation of sensors with respect to the environmental data signal. This is analogous to the energy management technique of dynamic duty cycling a hardware platform with respect to the harvested energy signal to match energy production and consumption; the two techniques are often used simultaneously. [5] mentions "harvest-aware adaptive sampling", which is similar to the "*harvester-aware power management*" (HAPM) mentioned in [64]. It is important to note that this suggests that not only are the relative data properties captured by a monitoring system different between applications, but also vary with time. This creates a difficult dynamic optimization problem when determining how much energy to invest in data collection, in real time [27].

3.3 Energy Management Theory

The work collected in [35] covers the content of several previous publications, by Kansal et. al., that are the cornerstone of power, or energy, management theory: [29], [34], [36], [37], and [75]. The results of these works are common enough that they deserve a summary. One equation in particular, Equation 3.1, appears in several subsequent works, such as [5], [64], and [86]. These mention the concepts collected in the work of Kansal et. al. as the theoretical basis for energy management for sensor platforms dependent on environmental energy harvesting. This section provides a description of this model and its set of variables for describing energy management.

The framework of Kansal et. al. breaks the energy system of the monitoring device into three parts: consumption, production, and storage. The load, the monitoring hardware, consumes power. This takes energy from two sources: an energy source and energy storage. The former role is fulfilled by harvesting

Table 3.1: A summary of variables used in energy management theory [35]

Variable:	Typical Units	Description
t	Second	Time
T	Second	Period; Time duration of interest
η	Percent	Conversion or Storage Efficiency
B	Joule	Energy Storage (Battery) Capacity
B_0	Joule	Initial Storage (Battery) Energy
$P_s(t)$	Watt	Source Power; Instantaneous energy production
$P_c(t)$	Watt	Power Consumption; Instantaneous energy consumption
$P_{leak}(t)$	Watt	Power Leakage; Instantaneous internal energy loss

hardware, while the latter role is filled by an energy buffer or battery. The model also considers other effects such as inefficiency and parasitic loss. This simple model is presented as Equation 3.1 below.

The variables from Table 3.1 are useful for concretely describing energy management. To begin, the pervasive equation of power management theory, presented in [35] and developed throughout earlier works, is:

$$0 \leq B_0 + \underbrace{\eta \int_0^T [P_s(t) - P_c(t)]^+ dt}_{\text{Excess Production}} - \overbrace{\int_0^T [P_c(t) - P_s(t)]^+ dt}^{\text{Consumption Deficit}} - \underbrace{\int_0^T P_{leak}(t) dt}_{\text{Internal Energy Loss}} \leq B \qquad \forall T \in [0, \infty) \tag{3.1}$$

where the $[x]^+$ operation is defined as [35, 36]:

$$[x]^+ = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{3.2}$$

This has reduced the operation of the system to its simplest elements. $P_s(t)$ is subject to the intermittence and variability of the environment. Depending upon how controllable the hardware of monitoring platform is, $P_c(t)$ is variable, but in a manner that is simpler to model. While these variables are power based, both these values are immediately integrated, so energy is utilized for computations. This means that both B_0 and B are energy values as well. $P_{leak}(t)$, as described in the original text, is considered the, "leakage power for the energy buffer" [35], but it can be generalized to include all consistent losses parasitic to the hardware platform, as they are conceptually similar. Additionally, η is defined as the "charging efficiency" of the "energy buffer" [35], but this too can be generalized to capture all percentage energy losses on the pathway from harvest to consumption. These extensions are in line with the equations presented by [35] and are necessary to model the system as complexity increases.

When characterizing sources and loads, the power signals are broken up into two parts: the average power production as time approaches infinity and short-term energy deviations from this average [35, 36]. The first part is the steady state average of each signal, shown in Table 3.2 as ρ_1, ρ_2, and ρ_{leak}. These values are used to match power consumption with the power supply as described in Chapter 5. Improperly matching these power signals results in wasted energy or guaranteed device failure, depending upon whether consumption and leakage are larger or smaller than production. The second part of the source and load characterization is the cumulative deviation of the instantaneous power signals from their average; this is measured in energy. This is called the signal's "burstiness" according to [35] and [36], and it includes steady state components where production and consumption are offset from one another, requiring storage to make their averages equal, in addition to random environmental variations. σ_1 and σ_2 capture this "burstiness" for the source, while σ_3 and σ_4 capture it for the load, in Equations 3.3 through 3.6, using variables from Table 3.2. σ_4 is usually zero, because loads typically do not produce energy, so it has been left out of Equation 3.6. All σ variables are used to describe some total energy offset over the entire deployment period, T; they can be used to characterize what amount of energy storage is required ensure that energy neutral operation can be achieved for a given source and load pairing.

$$\int_{\tau}^{\tau+T} P_s(t)dt \leq \rho_1 T + \sigma_1 \tag{3.3}$$

$$\int_{\tau}^{\tau+T} P_s(t)dt \geq \rho_1 T - \sigma_2 \tag{3.4}$$

$$\int_{T} P_c(t)dt \leq \rho_2 T + \sigma_3 \tag{3.5}$$

$$\int_{T} P_c(t)dt \geq 0 \tag{3.6}$$

The idea of "energy neutral" operation is derived from the definitions of

Table 3.2: A summary of characterization variables for energy harvesting theory [35, 36]

Variable:	Typical Units	Description
$P_s(t)$	Watt	Harvester Power Production
$P_c(t)$	Watt	Load Power Consumption
τ	Second	Start Time; May be any Point in Time.
T	Second	Operational Period
ρ_1	Watt	Average Instantaneous Energy Harvest
ρ_2	Watt	Average Instantaneous Energy Consumption
ρ_{leak}	Watt	Average Instantaneous Internal Energy Loss
σ_1	Joule	Energy Production Surplus
σ_2	Joule	Energy Production Deficit
σ_3	Joule	Energy Consumption Surplus
σ_4	Joule	Energy Consumption Deficit; Generally zero.
η	Percent	Conversion or Storage Efficiency

the sources and loads in Equations 3.3 through 3.6, and the model in Equation 3.1. Energy neutral operation requires three conditions:

1. The average power harvested from the environment is greater than or equal to the average power consumed by the load and lost to leakage.

2. There is sufficient storage to buffer all "burstiness", or temporary energy offsets, between these average power signals.

3. The initial energy in the system is sufficient to buffer all energy shortfalls until a steady state is reached where the other two conditions are true.

These conditions are shown as three equations in "Theorem 2.2 Energy Neutral Operation" of [35]. This "energy neutral" idealization assumes that all harvested energy is completely consumed by the load or various leakages and all "bursts" of energy are within storage capacity. No infinitesimal amount of energy may be wasted (parasitic losses are not considered waste but an unavoidable side effect of operation) or the model breaks. It also assumes that the harvesting hardware is sized such that as the operational period of this ideal device approaches infinity, the average power harvest equals or exceeds the energy lost or consumed (not wasted) by the device. An energy storage device that can completely store all initial energy required to handle production shortfalls, both periodic and transient, while keeping enough initial capacity available to hold all future peaks in production, will be able to completely eliminate device failure, provided that the average power production of the harvesting unit is greater than the average consumption of the load. Under these conditions, when energy consumption is matched with energy production, the device should be able to operate until its hardware fails. All energy related failures are eliminated because there is no point in time when the monitoring system does not have the energy required to operate. An *ideal* device would be able to operate infinitely, but these do not exist, so perpetual operation is constrained to the lifetime of the sensing platform itself.

Chapter 4

Environmental Monitoring Platforms

4.1 Hardware Background

Two environmental monitoring platforms are considered in this research: the *Arctic Weather Station* (AWS) and the *SolarNode*. This chapter briefly outlines their hardware characteristics, and then describes how they are modelled in software.

The AWS is the focus of energy management research by Pimentel in [64], [65], and [66]. This work was continued in [93] and [94]. The complete documentation for the platform simulator is presented in Pimentel's thesis, [64].

The SolarNode prototype is a new environmental monitoring platform designed by M. Prauzek and briefly described in [71], [72], and [95]. It is a low-power, modular sensing solution which integrates modern energy harvesting and storage technologies.

4.1.1 Arctic Weather Stations

AWSs are deployed for long-term automated climatological monitoring [21,80]. These are legacy systems with established commercial technologies: what they lack in versatility, they make up for in convenience and reliability. They provide an extended data set for a single location. The AWS comes as several connected modules rather than a single, enclosed monitoring solution. The support structure generally consists of a tripod and single upper mast with a crossbar. This holds sensors away from the vertical mast where the main hardware modules are attached. Aside from energy storage, the station is stable, light, and easy to deploy, making it ideal for remote monitoring. Typically AWSs use sealed lead-acid batteries for energy storage, but their bulk and weight complicate the operation and deployment of the weather station. A photograph of a deployed AWS is presented in Figure 1 of [65] and Figure 1 of [66].

The structure of the AWS provides convenient mounting options for energy harvesting modules; the mast allows solar panels to be placed vertically, high above the ground, to avoid snow cover. The combination of solar panels for energy harvesting and sealed lead-acid batteries for energy storage is common; the convenience of these two technologies is hard to surpass. The only overhead this pairing requires is a solar charge controller, to prevent over or undercharging the battery and provide MPPT for the solar panel. The simplicity of data logging and processing hardware restricts the energy management strategies available to the platform to those with simple computations [64–66].

To generalize, a fully functional AWS will have the following components:

- An Energy Source: to power the monitor

- A Sensor Array: to capture environmental data
- A Data Logger: to manage measurements and interface with sensors
- A Transmitter: to provide remote access to measurements
- Energy Storage: to buffer the energy source from energy consumption
- Data Storage: to protect measurements before they reach the user

The AWS is only one of many possible configurations for automated weather stations. It is assumed that only these six modules are available for manipulation. Any energy management strategy must consider these resources, as must any simulation for a similar purpose. The original simulator for the AWS, created by Pimentel, is available in [64]. A modified version, used in [93] and [94], is presented in Section 4.2.

4.1.2 SolarNode Prototype

The SolarNode prototype is a small, securely enclosed, wireless sensing platform for environmental monitoring. This platform is modular and configurable, which makes it ideal for testing energy management strategies in the field, and avoids the problems associated with proprietary devices. It is still in the testing and prototyping stage, so a simulation was created to help predict how it will perform. This simulation is described in Section 4.3.

The SolarNode has been designed for remote environmental sensing: it is solar powered, energy concious, and wireless. As mentioned in Section 3.2, inefficiencies designed into the system are difficult to optimize back out later, so this platform focuses on energy efficiency at the lowest level. The hardware uses components to minimize power consumption and provide a strong foundation for energy management strategies. An internal solar energy harvesting module is included to supplement long term operation.

Six small solar panels provide a harvester area of 6.48cm^2; each panel has 3 small square cells with side lengths of 6mm [30]. These are mounted on a slated printed circuit board inside the clear, weather-proof case [95]. All harvested solar energy is directed to a solar charge controller that manages the harvesting process [90]. Energy harvesting is intended to minimize the SolarNode's dependence on stored energy.

The SolarNode uses tiered energy storage. The first level of energy storage is an energy buffer made from two 60F supercapacitors attached to the solar charge controller [12], while the second level is an energy reserve of 2 AA batteries. While these batteries may be rechargeable, the hardware does not have that faculty, so they are an energy reserve regardless of their chemical properties. Currently, the batteries are expected to be lithium-ion [14, 15]. Both layers of energy storage pass through identical switching regulators, effectively multiplexing the platform's energy connection [89]. The energy reserve is only engaged when the energy buffer falls below the limit set by the solar charge controller.

The two levels of energy storage are paralleled by the device's two layers of data storage. The node is equipped with a data buffer: a small, non-volatile, internal memory chip [42]. To provide long term, removable data storage, a *Secure Digital* (SD) card is included. This is significantly larger than the data buffer and effectively acts as a data reserve: analogous to the energy reserve. Data on the platform is accessible in two ways: manually, by removing the SD card, and remotely, by wireless radio connection. The wireless connection is only effective for short range communication, so a supporting base station is required for remote access to the data. The different sensor types attached to the platform determine the required energy investment and data return for measurement operations.

The sensing platform is equipped with a modern, low-power MCU to manage its operation [6]. This central processor is directly accessible for programming and interfaces with all attached sensors. Any energy management strategy implemented on this platform is executed by this chip. While there is plenty of processing power available in comparison to legacy devices, such as those in Section 4.2.1, the MCU is not a computer and requires most energy management strategies to be prepared before deployment.

4.2 Previous Simulations

The work presented here grew out of the simulator in [65], [66], and published in its final form in [64]. The evolution of the simulator between these three documents suggests that it was initially intended to estimate environmental energy harvesting and environmental effects on energy storage. As energy management became a more prevalent topic, the simulation grew to encompass more of the load so that control opportunities would appear. The complete code for this simulator, including the code required to prepare input data, is presented in the Appendices of [64].

4.2.1 Arctic Weather Stations

Subsequent work on the Environment Canada (EC) AWS simulator has focused almost exclusively on modularizing components and improving load model detail. This work and its results are presented in [93] and [94].

The removal of the energy harvesting systems from Figure 5.2 in [64] is the most notable difference between it and the simulator in Figure 4.1. The energy harvesting modules presented in [64] are detailed; however, their involvement with energy management is mostly overhead from a computational perspective because the data they process is invariant. These models will produce the same energy harvest for a certain harvester configuration and environmental data set every time. The environmental data was recorded once, processed once, and will not change again. The energy harvesting module of the simulation serves to shift information from the environmental data domain to the energy signal domain. Initially, the focus of this simulator was optimizing this shift in domain through detailed modelling, but here the focus is on effectively investing energy resources.

The only variable that changes the energy signal between simulator runs is the “random failure” variable. According to Appendix A3.2.2 of [64], it turns off an energy harvester for up to two weeks. Essentially, it is intended to mask and randomize the harvested energy signal, demonstrating robustness of control, rather than actually modelling device failure. Since this is the case, the “random failure” signal can be removed from the input, and if still necessary, moved to the output of the energy harvesting module. Because this was the only randomizing factor in the energy harvesting subsystems, they are now completely invariant between runs. They can be divided off into an independent simulation: this is called the *Simulator Front End* (SFE). The remaining

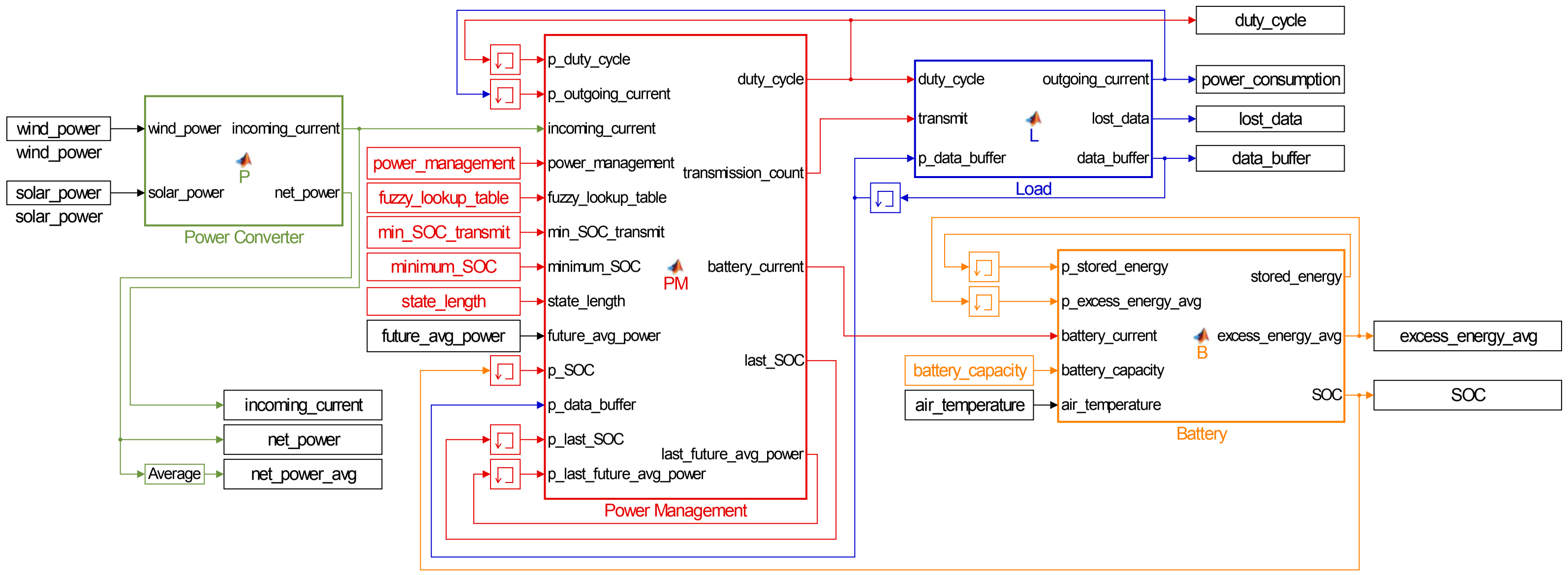


Figure 4.1: The Mathworks Simulink diagram shown above is the modified AWS simulator. The modules in the simulator are similar to those in [64]; however, all work related to energy harvesting has been removed to run independently. The connections between function blocks have been changed to provide more appropriate state information to the controller.

modules, called the *Simulator Back End* (SBE), require two sets of data from the SFE to operate: the computed energy signal from the harvester models and and environmental data that directly affects hardware performance. For example, ambient temperature directly affects the storage capacity of sealed, lead-acid batteries, so the SFE must pass it to the SBE with the energy signal.

Separating the SFE from the SBE has many advantages. First, the computational load of the SFE is completely abstracted from the SBE. The majority of computational effort in the previous simulator was concentrated in the energy harvesting subsystems; now that they are removed, the new simulator is fast and lightweight. The genetic optimization process, discuss in Section 6.4, requires the simulator to run a large number of times. This is simple now that the computations required to redundantly recompute environmental energy availability are not carried out. Before, the energy signal from the weather data had to be recomputed hundreds, if not thousands, of times by the harvester module, with the only changes due to the "random failure" variable. Now the precomputed energy signal from the SFE is all the SBE needs. Additionally, the SFE can easily be changed without affecting the SBE.

From the perspective of energy management, the AWS has some interesting components. These are the modules which are observable, their state can be determined, or controllable, their state can be modified. Section 4.2.1 lists 7 items which are relevant to the simulation of the weather station: the energy source, the sensor array, the data logger, the transmitter, energy storage, and data storage. In this model, the activity level of the sensor array and satellite transmitter can be controlled, while the state of energy and data storage can be observed. The operation of the data logger itself is captured in the model and neither observed or controlled. The energy source, the harvester, is not

Table 4.1: Simulator settings for the AWS

Module	Current per 2160 Hours at 12 Volts	Current per Hour
Data Logger	84	0.0389
Sensor Array	112	0.0519
Transmitter	80	0.0370

observed directly; instead, its effect on energy storage is observed. For the sake of control, it is assumed that some information on the internal memory of the data logger is available. While the AWS is a simple device, the internal memory cannot be a complete mystery. Additionally, if the state of the data buffer is not observable, then the system is only capable of feed-forward control from a data perspective, which decouples the energy for data exchange. In that case, the controller could not determine when additional measurements lose value by overwriting earlier ones. Because the performance of the platform is determined by the quality of data it produces, the state of the data buffer must be made available.

The parameters of the simulation are provided in three documents: Table 2.1 of [64], Table 3 of [65], and Table 1 of [66]. These values are converted to some hourly consumption of amps, at 12V, by taking the total current consumption over 90 days, and dividing that by the number of hours in 90 days. These values are presented in Appendix A2.7 of [64]. The results of this computation are provided in Table 4.1.

It is assumed that the *data logger* and *sensor array* can be duty cycled. The minimum operational interval is 15 minutes, meaning 1 hour timesteps have duty cycles between 25% and 100%, in 25% increments. The simulator does not account for the energy consumption overhead of duty cycling. If the device turns off it has no way of fixing this to some repetitive period, and so cannot intentionally operate less than once a timestep. Thus, any timestep where a measurement is not taken, may be considered a device failure. This is remedied in the simulation of Section 4.3. The *transmitter* consumes energy, some current at 12V, based on the number of transmissions it makes in a timestep. Transmission also does not consider the overhead associated with establishing a satellite link, error checking, handshaking, or retransmitting lost packets. While this is acceptable for a model, is does not allow an adaptive control system to learn interesting energy-saving behaviour. The simulator never intentionally groups transmissions to eliminate overhead and there is no opportunity for feedback within the transmission process [94].

Energy harvester and storage sizing are directly modified by genetic optimization in [64]. In fact, the results of that work are taken for granted here, and no attempt at optimizing harvester or storage sizing has been made. The ambient temperature signal from the SFE is fed directly into the storage module, while the energy signal is still passed through the power converter. The power converter is assumed to have 90% efficiency and perfectly convert the energy signal to a current value at 12V for the SBE. Because of this assumption there is no advantage to working in current, so this is abandoned in favour of working with energy in Section 4.3. That said, the energy storage and energy harvesting modules are essentially unmodified from their original form. Harvester sizing is abstracted out of the SFE so that it can be easily changed between simulations without recomputing the entire energy harvest data set.

For this simulator, *energy storage* is a rechargeable energy buffer; here it has been fixed at 300Ah [93]. This is very large, but two of the three Arctic test locations require this amount of storage. The wind energy harvesting module proved to be unreliable, so it was removed from the simulator. The solar panel is assumed to be 0.07m^2 with 90° inclination and 10° azimuth for all locations. Standardizing the simulation for testing between years and locations may not be optimal, but it was very convenient: small differences in setting between location no longer confound comparison, making analysis and control optimization straight forward.

The load and power management modules were changed the most. Control system modifications are discussed in Section 6.3. The *power management* module now uses the previous timestep's state information to filter noise in the control signal [94]; this is discussed in Section 6.2.1. Additionally, the data buffer's state information is fed back into the *power management* module to provide closed loop control over the data side of the simulation. Given that data collection is the ultimate goal of remote monitoring, it is important to include it in the management module. The energy for data exchange that the controller is attempting to optimize is dependent upon some level of data-related feedback [93]. In the previous simulation, the *transmission* action was dependent on the real time *incoming current* value; however this has been removed. The *incoming current* from the solar cells would be difficult to measure and the state of the energy buffer must already be observed; the energy state of the system is based upon the energy buffer level. Hourly current from the solar panel is very small in comparison to the size of the buffer, so using the energy buffer as a state value filters the harvester energy signal. This made the actions, measure and transmit, controllable and dependent upon an observable state: the energy buffer level.

The *load* was modified to provide feedback to the *power management* module and use the energy consumption values provided in Table 4.1. An additional output, *lost data*, was added to the module to simplify data related performance evaluation. This signal shows when data is lost, due device failure or being overwritten while in the data buffer.

Working with this simulator exposed some fundamental control problems; a more modular system was required to manage its energy for data exchange. The transition to a new system is detailed in Section 4.3. This new simulation is intended to split energy management into three parts: energy production, energy consumption, and control. Energy production has already been split off into the SFE, while the SBE must be divided again. The SBE is divided between the *hardware* and *control* modules: the former is platform specific and can be reconfigured to work with different remote monitoring devices, as seen later, while the latter abstracts control from the monitoring platform completely. The *hardware* module should work with physical values, while the *controller* operates with relative state information.

4.3 Modular Node Model

As mentioned earlier in Section 4.2.1, the simulator needed to change focus from storage and harvester sizing in [64], to a more modular, abstracted simulation for energy management. This relegates restrictive, hardware-specific code to a single module that can be changed for different applications, and allows the other module to focus entirely on control. At the present, the simulation is called the “Modular Node Model”. It is written in the Mathworks Simulink environment, for convenience and compatibility with the existing AWS simulator. This provides a convenient visual workspace to manipulate modules and signals, as well as a powerful environment to process data and generate

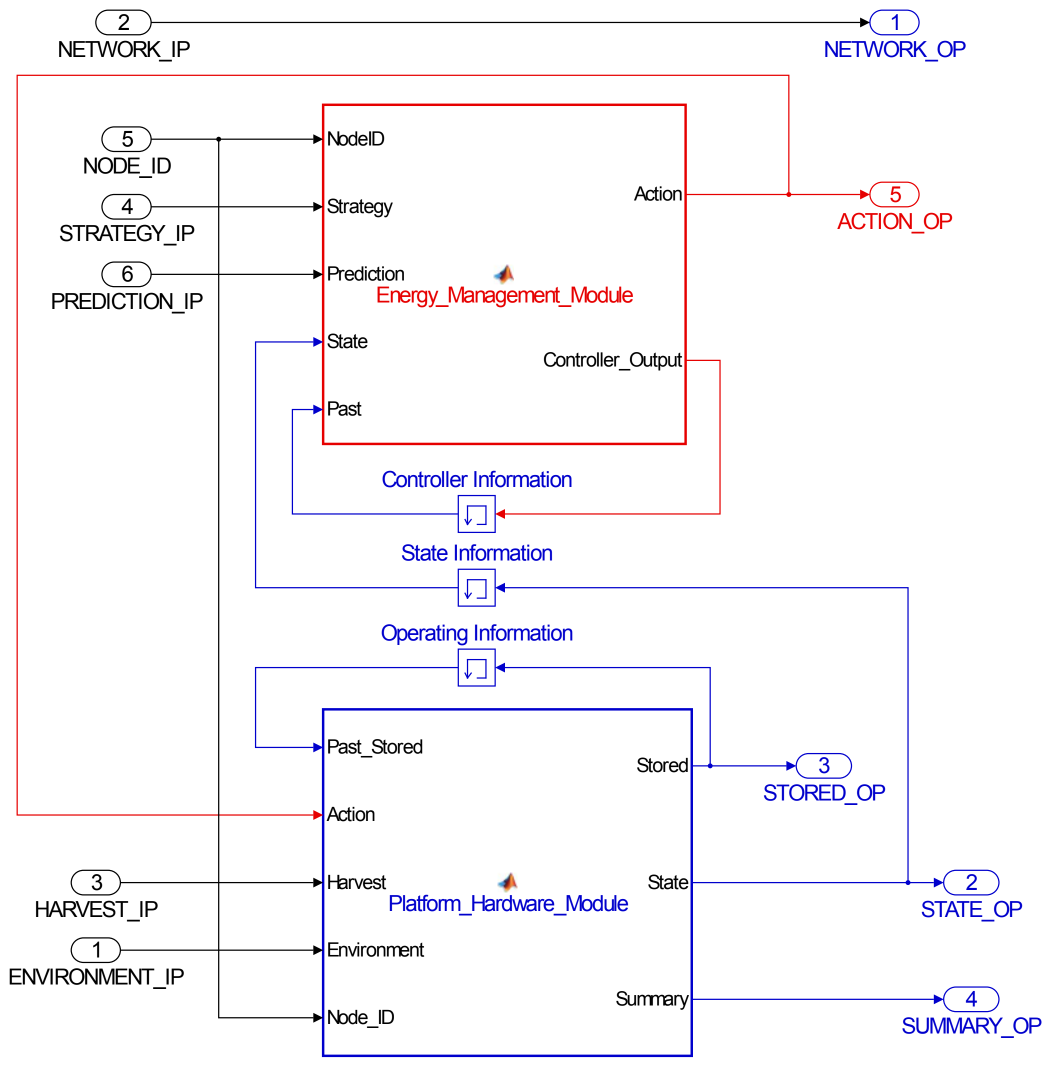


Figure 4.2: The two modules shown above make up the new simulation. The *Platform Hardware Module* (PHM) models the environmental monitor using physical variables. The PHM outputs relative state information for the second block, the *Energy Management Module* (EMM). The EMM makes control decisions for the simulation. Because it operates on state information rather than physical variables, it is abstracted from the hardware. The EMM returns the desired activity level of the hardware to the PHM. If the PHM cannot execute these actions, the device has failed.

plots. If the simulation were to expand, it would need to migrate to another high level language and align with more formal coding practice.

The motivation for moving towards this new simulation is discussed in [94]. The main points are broken down by subsection in Section III, “Controller Improvements”:

Shift Focus from the Time to the State Domain

The simulation should allow the control module to filter actions based upon changes in state rather than changes in time step. Using the *time* variable, which was not formally designated as a state variable, made the control system ineffective at filtering out environmental variation in its incoming energy signal.

Improve Load Model and Increase Controller Integration

Increasing the detail of the load model may allow the controller to improve performance, while increasing and centralizing the controller’s influence enables it to do so. This provides the energy management strategy more tools to work with.

Increase the Flexibility of Control System

Data and performance should determine how the fuzzy rules of the controller are composed; expert opinion is useful, but suboptimal. By abstracting control from the hardware into a separate module, there is more freedom in mapping states to actions.

Data-Centric Perspective on Fitness

The optimization process of the simulation and control system should be determined based upon the quality of data it can collect because that is the objective of remote monitoring. Earlier optimization attempts focused on the energy system, which can be effective, but are a less direct approach to this goal.

Generally, the simulation needed to be reorganized to favour the perspective of control systems and machine learning, rather than energy harvesting and storage. The largest change involved separating the *Energy Management Module* (EMM) from the rest of the SBE and basing its operation on relative state values. The *Platform Hardware Module* (PHM) is now free to operate using physical values because the EMM only requires state information. Though the EMM uses abstractions it can now dictate exactly what it wants the hardware to do and leave how to do it up to the PHM. It is free to develop in which ever way produces the highest performance.

This change was also prompted by the arrival of a new, more computationally capable, monitoring platform: the SolarNode, described in Section 4.1.2. The simulation of this new device is divided into three modules: *Input Data Processing*, *Energy Management*, and *Platform Hardware* [71, 72, 95]. These are described in following subsections.

4.3.1 Input Data Processing

The *Input Data Processing* module (IDP) prepares data for the other two modules of the simulation: the EMM and the PHM. All necessary environmental data is processed and passed on as either scalable energy signals or ambient environmental conditions for hardware. The energy signal must be scalable because harvester sizing is abstracted to the PHM. Environmental data that determines the ambient operating conditions of the hardware platform must also be packaged with the energy signal so that it can be automatically loaded while the simulation in running. This data must be easily accessible because it is used independently of the IDP by the rest of the simulation. For example, if the platform's storage efficiency is affected by ambient temperature, the PHM must be able to extract both the energy signal and the temperature data at simulation run time. Presently the standardized format includes a strict naming convention and hourly data points, but that can be easily changed to suit different hardware platforms, environmental data sets, and simulation time steps. This convention is compatible with the energy harvesting SFE split off from the SBE in Section 4.2.1. This allows work from the AWS to be directly passed on to the SolarNode with minimal modification. While this makes an effective estimate, it should be recomputed for accuracy.

This module requires continuous environmental data with no missing or irregular data points. Presently, the timestep of the input data must match the timestep of the simulation, which defaults to 1 hour intervals. This simulation uses hourly time steps for compatibility with most environmental data sets. If a data set has irregular or asynchronous time periods, it must be changed to synchronize with the simulation. The simulation is event driven and based on energy consumption, so in the future it could automatically adapt to this problem; however, at this time, missing or irregular points must be replaced with valid ones.

Points are filled in two ways: locally, based on the weather, and regionally, based on the climate. Filling points based on local data takes into account the weather of the area, as captured by the data set. This has the advantage of preserving weather extremes and keeping the environmental data related to that point in time. This works best for small gaps in the data set. Climate related filling looks at all available data sets to find seasonal trends associated with the area, rather than specific weather patterns. If a large chunk of a data is missing from a data set, but that data set cannot be discarded, filling that segment with a climatological average provides a functional estimate. While this eliminates the local variation associated with specific weather patterns, it allows data sets to be used that would otherwise be discarded. An example of this is presented in [95], when a large chunk of data is missing from an otherwise excellent time series. If the missing data segment was not filled, there would be no suitable datasets available for the region. This is described in more detail in Section 7.1.3.

The SolarNode uses a simple method of converting environmental data to an energy signal. The solar irradiance available to the simulated device is assumed to be identical to the solar irradiance from the environmental data set; this data set comes from field deployments. The incoming energy over the area is then scaled based upon the energy harvesting technology available. For the SolarNode, the solar panel area is 0.000846m^2 with an efficiency assumed to be 22% regardless of circumstance [72,95]. This is a significant departure from the high level of detail provided by the energy harvesting models of [64], but this highlights the difference in research focus. However, now that the simulation is modular, the detail and composition of this module could be improved, or completely changed, as long as its output remained compatible with the other modules in the simulation.

Standardizing the output of the IDP was necessary to interface with the other modules. The output of the IDP should have any automatic deductions which will not vary from simulation to simulation already included, but should leave those associated with power conversion to allow different platform storage and consumption topologies to be tested within the PHM. A “rule of thumb” to divide data processing between modules is: if the PHM cannot *observe* the process, like a power converter integrated into an energy harvester, and the EMM cannot *control* the process, say that power converter had no interface, then the IDP module should integrate the effect of the process into its output data set. If these two conditions are met, then the EMM and PHM have no way of interacting with the data set, and the data will not change between runs; by computing this with the IDP, computational effort is reduced later. The output of the IDP module is presently a power signal, in W/m^2, and the ambient temperature, in °C, with values every hour. The PHM is based on the energy consumed by events in the simulation, and internally converts power per timestep to total energy per timestep; the EMM operates with the same frequency as PHM events, not the timestep of IDP data. Genetic optimization requires automatic extraction of the IDP output, so a rigid naming convention is required for all variables. Power signals from each harvesting technology should be separated, along with all variables for ambient conditions. This way the PHM can extract and recombine them as it requires.

4.3.2 Energy Management Module

The *Energy Management Module* (EMM) acts as a controller for the simulation. It is dependent upon, and housed entirely within, the PHM. Dividing these modules allows control to be independent of physical variables and hardware configurations. Abstracting the EMM allows the simulation to interface with different management strategies, provided they can handle the same inputs and outputs. The EMM accepts state information from the PHM as an input and provides action information back to the PHM as an output; the state space must be completely mapped to the action space with no undefined points. Any energy management strategy that fulfils these requirements can act as a controller for this simulation. Those used by the EMM for the SolarNode are described in Section 6.3.2.

The PHM provides the EMM with relative state information for decision making. This is dependent upon the hardware platform that is modelled: the SolarNode uses an array of four variables. Presently these are the state variables that measure the percentage capacity used in the data buffer, the data reserve, the energy buffer, and the energy reserve. To determine what to do with this state information, the EMM loads a preconfigured state-to-action mapping, stored as a lookup table as described in Section 6.3.2, and operates accordingly. This allows the EMM to retrieve action information for the PHM. At this time it determines how often to expend energy collecting and transmitting environmental measurements, fulfilling the objective of the monitoring platform, and how often to empty the data buffer into the data reserve, protecting these measurements for later retrieval. These actions are passed to the PHM as the frequency of operations per timestep. A different hardware platform will provide different states and require different actions, but because these values are abstracted from physical values, the simulation is easy to change.

The EMM is highly configurable, and, in addition to receiving past and present state information, can optionally receive three other sets of variables, as shown in Figure 4.2: node identification, strategy configuration, and apriori predictions. *Node identification* can be used to pass relevant hardware or network specific settings to the control system; this could be a serial number or some preconfigured starting conditions. *Strategy configuration* lets the controller know which energy management strategy it should follow during operation. This allows several strategies to be set up at once and tested automatically. This is used in [72] to quickly switch between energy management strategies. Finally, *apriori predictions* can be used to pass information to the controller which it could not otherwise observe. This is done to capture the predictive elements of [64]. If it is included as additional state information, it must be made part of the state-to-action mapping of the energy management strategy. This information is critical to remote monitoring stations in extreme environments because it can warn the device of future problems, energy shortfall for example, before the device can observe them. Should the device fail, it would not be able to learn from the environment anyway, so a predictive signal may be required. However, in the present SolarNode simulation, these signals are largely unused.

4.3.3 Platform Hardware Module

The *Platform Hardware Module* (PHM) is responsible for all computations relating to physical variables in the simulation once the IDP has produced an output file. The number and complexity of these variables changes with the platform: the AWS required very few parameters to characterize its operation, while the SolarNode requires several scripts to prepare them all. No relative values are used inside this module and settings should be based on empirical results: either from the manufacturer, via datasheet, or from experimentation with the monitoring platform. PHM variables and settings change with the physical system; however, they are read from an external configuration file. This simplifies switching or reconfiguring platforms and enables automated testing. The file can be changed outside the Mathworks Simulink environment. Changing the PHM does not affect the other modules. Ideally, the IDP should be able to change based on available energy harvesting technologies and the PHM need only scale the power signal based on hardware limitations, such as solar panel size or number. The IDP is responsible for the one-way transfer of an output file to the PHM. If the IDP requires some sort of real-time interaction, it should be through the control system of the EMM.

In this module, the data buffer, data reserve, energy buffer, and energy reserve all use physical units: bytes or joules. It is often more convenient to work in data packets, the results of complete measurement operations, than bytes, but this is a simple conversion. The size of these storage layers are either configurable, like a removable SD card, or fixed by the platform, like the supercapacitors attached during fabrication. The amount of energy or data in each storage layer is divided by its total capacity to produce a percentage; this provides some relative measure of state for the EMM.

In its current state, the inner workings of the PHM are straight forward. To begin, the module pulls in its hardware operating variables, the action commands of the EMM, and the stored physical variables related to its previous state. The state information contains not only the number of bytes and joules stored by the device, but also past action commands. This is combined with new action information from the EMM to determine what to execute this timestep.

All actions are based on the idea of *operating frequency*. An action is executed when the sum of its internal frequency counter is greater than 1. For example, say that the EMM determines that the PHM should take 1 measurement every 2 hours: an action with a frequency of 0.5 operations per hour. The

Table 4.2: PHM action descriptions

Operate	The platform is neither out of energy (failure) or sleeping
Store	The platform transfers data from buffer to reserve
Upload	The platform transfers data to some external source, possibly emptying the data reserve
Transmit	The platform transfers data to some local source, such as another networked device or nearby user
Receive	The platform receives data from some local source, such as another networked device or nearby user
Measure	The platform engages its sensor array and captures environmental data

first time the command of *0.5 measurements* is received, nothing happens; the total value of the command is less than 1, so it is simply stored in the PHM. In the next timestep, that 0.5 is added to the incoming value of 0.5, so the PHM executes the measure command. This generates the measurement event, which consumes some amount of energy and produces some number of data packets. If the frequency of measurements is above 1.0 per timestep, the operation is executed repeatedly until the counter is reduced to below 1. Because the PHM may execute several different actions, this process is aggregated into something similar to a state machine. The current actions are: *operate*, *store*, *upload*, *transmit*, *receive*, and *measure*. While they are not all implemented by the SolarNode, ideally they operate as described in Table 4.2. The obvious extension to the *download* action is excluded. There is not enough detail on networking options and internal memory management for it, and other network functions, to operate as intended.

Each of these operations are treated as individual events that consume some amount of energy. This energy is based upon how much power the operation consumes, on average, and approximately how long it takes to execute. An additional cost parameter, referred to as *overhead*, is intended to capture all energy associated with starting, stopping, and organizing the event. This will help the controller learn if it is beneficial to batch together events like transmissions, which potentially have a high overhead energy cost. To ensure that the platform will not fail in the middle of an operation, the PHM compares its energy resources, the energy from the IDP and what it has stored, to the energy cost of the operation. Underpowered operations are not attempted. This is an idealization of platform operation but it simplifies the simulation. Should all operations be completed successfully, the total energy cost of that timestep is adjusted by any efficiency losses from power converters and added to the total incoming energy signal from the IDP module, which is also adjusted by energy conversion efficiency losses. Finally, this net energy consumption per timestep is reduced first from the energy buffer, then, if that is insufficient, from the energy reserve. Tiered storage reduces the wear on the energy reserve by buffering it from charge and discharge cycles. If there is not enough energy for the device to perform any operations, it remains in a sleeping, low power state; this is considered *device failure* because the PHM failed to execute the command of the EMM. Should the device have insufficient energy to sleep, it will be considered "dead" for the timestep. At the very end of the timestep, the internal losses of the energy storage mediums are computed; this means if the device "dies", it can still lose its remaining residual charge. This is a problem for supercapacitors as they must be recharged from empty if they are idle for a moderate length of time. Finally, all the information from that timestep is packaged and exported from the module.

The PHM passes out three signals: stored physical values to be used in the next time step, relative state values to be used by the EMM, and relevant diagnostic information to evaluate performance. The stored physical variables are not held within the Mathworks MATLAB code blocks because they must also be passed out to the user for analysis. Relevant diagnostic information includes how much energy and data are wasted, lost, or consumed, in each timestep. This is important for evaluating and optimizing the energy management strategy later on. As shown in [64], this information is very helpful for sizing energy storage and harvesting hardware.

The general "event driven" framework of the PHM allows the simulation to be quickly changed. The AWS could be ported into the present SolarNode simulation with little difficulty. In fact, given the modular nature of the present simulation, the SolarNode and AWS platforms could be run simultaneously in the same Simulink environment, if another set of configuration scripts were written and the model in Figure 4.2 was duplicated.

Chapter 5

Energy for Data Exchange

The theory behind component selection and sizing is covered in Chapter 2 and Chapter 3. The energy and data systems are considered separately before discussing energy management. Data considerations are focused on quality, production, and storage, while energy considerations are focused on production, consumption, and storage. Energy management infers its performance from data quality, while ensuring efficient energy consumption. Optimizing management moves the system towards the best possible energy for data exchange rate. The objective of this exchange is to produce the highest quality data using the smallest amount of energy. To succeed, the system must adapt to its environment, eliminate energy waste, and ensure that collected data is not lost or overwritten.

Simulation Note:

Simulations are helpful tools for investigating the energy for data exchange in energy harvesting environmental monitoring systems. Their results provide useful estimates of energy resources and hardware performance. This forms a basis for harvester and storage sizing, which would otherwise depend on less complete calculations. Without computational tools, accurately sizing parameters for platforms with adaptive energy consumption is challenging and inefficient. Selecting appropriate simulation parameters is important to successful operation. The monitor's energy and data systems are interconnected, so the simulation helps infer what sensors may be selected based on the energy available, or conversely, how much energy must be harvested to meet performance requirements. If harvester size is fixed, then the simulation can be used to estimate how the device will operate to ensure deployment is worthwhile. As data acquisition determines performance, the simulation helps refer this performance information back to the energy system. The goal of this chapter is to look at the practical side of implementing energy management theories within a simulation.

5.1 Data Considerations

The objective of autonomous sensing and monitoring devices is to acquire data. The data system of the monitor is responsible for generating value from the resources provided by the energy system. The *quality* of this data is determined by how well the captured data set meets the requirements of the monitoring application. Data production can be considered a process of investing energy to generate this value. Data storage provides energy management with the ability to delay tasks and improve performance.

5.1.1 Data Quality

Data quality is determined by how well a data set meets the requirements of an application. Determining the quality of the data set attaches a value to the product of the energy for data exchange. This value allows the energy for data exchange rate to be evaluated. The value of a data set depends on two things: what properties are relevant to its quality and how can their relative value be determined. The properties of the dataset can be divided into those inherent to the data itself, like accuracy, frequency, or completeness, and those related to how it is used, like availability, security, or applicability. The present application, "long term observation of climate variables to inform scientists, policy makers, and the public," requires accurate, consistent, long-term data sets for a complete look at weather and climate. Without these qualities, the application is not satisfied, so they determine the data's value. The application establishes the relative value of data properties with respect to one another.

For example, to demonstrate how data quality may be determined, consider analysing the solar energy resources of an area. For the data set to be valuable, its collection frequency must be high enough to capture relevant events without being so high that it becomes impractically large. Measurement frequencies lower than the diurnal cycle are not informative because the nature of the sun is not observed; alternately, millisecond accuracy would create huge, redundant data sets that are impractical to work with. Both of these extreme options are less valuable than a useful middle ground. For yearly intervals, solar irradiance measurements spaced a few minutes apart are close to ideal; they are the most suitable for the application and thus have the highest quality. That said, many other factors impact data quality as well. Climatological data does not need to be made available immediately after collection because the climate does not change fast enough for that to matter. Wireless transmission of environmental data does not need to be so secure that it cannot be read by a third party because the consequences of stealing it are minimal; that said, it does need to be secure enough to avoid loss or corruption during transmission. To lose data during transmission means that the energy used to produce it was wasted; wasting resources reduces exchange rate at which the data set is produced. Data that is not directly related to the study may still have some limited value: *photosynthetically active radiation* (PAR) is a limited portion of all solar radiation but is still informative of the solar energy resource; even though it is less than ideal, it still has some value. There is a hierarchy of value with respect to data quality within each application.

Consider another scenario, posed by [35]: some environment is monitored for intrusion. In this case, high accuracy is not important; the question is boolean: is it true an intruder is present? It may even be good to err on the side of caution and capture a false positive for intrusion, rather that accidentally permit an intruder. However, with this decrease in accuracy relative to the previous example application, comes an increase in required measurement frequency and data availability. It would not be acceptable if the measurement frequency was lower than the speed of intrusion or if intrusion detection was delayed. Access to environmental data can be delayed for a season: the environment is not going to run away. In this application, data security is much more valuable than in the previous example, because the intruder must be prevented from accessing or corrupting the data. This application has a different measure of data quality and a vastly different way of assigning value to data production than the previous example. Energy management must be specifically designed to invest resources appropriately in each case.

Prioritizing different data properties to improve data quality is demonstrated by the AWS results of Chapter 7, [93], and [94]. The AWS energy management strategy prioritizes the consistency and completeness of measurements before their availability and frequency. First, transmissions are delayed until spring to prevent device failures that would fragment the data set by storing them in the device's internal memory. The data is stored and safe but completely unavailable while the transmitter is off. Availability is less important than completeness: climate data is not required immediately after collection, so storing it allows that energy to be used to take more measurements. Second, the number of measurements per hour is reduced during the winter to use less energy so measurements of a lower frequency may be collected over more of the year. The more often measurements are taken, the more thoroughly the weather is recorded; however, a higher measurement rate consumes too much energy and leads to device failure and missing data points. This would provide a less complete understanding of the climate as a whole. Since the application is climate monitoring, a larger portion of the year covered by measurements is more important than a high measurement rate.

5.1.2 Data Collection

During the data collection process, the system consumes energy at a certain rate and generates value based upon data quality. It is important to consider the properties of the data being collected and the limitations of the system with respect to the data side of the energy for data exchange. Data is the metric for evaluating performance in sensing and monitoring applications and must always be considered in energy management. For environmental monitoring, performance is directly related to the value, and thus quality, of the produced data set.

The rate of data production is bound between the two operational extremes of the platform: MIN_{OP} and MAX_{OP}. Sensors do not need to produce redundant data; there is some maximum operational level for the sensing platform, MAX_{OP}, after which additional data collection is not useful. If a bottleneck in the data system prevents the monitoring device from reaching this data rate, it generally indicates a design error. Either the sensor's maximum rate of data capture, the processor's maximum rate of data processing, or the transmission hardware's maximum speed when evacuating the data buffer is too low. Data collection below the minimum operational threshold, MIN_{OP}, wastes energy and reduces the overall quality of the data set. A good example of a operating below MIN_{OP} would be when a monitoring system recovers from failure by harvesting energy, only to fail again by exhausting its energy supply before successfully completing the data collection process. Turning on the device when there is not enough energy to safely complete the measurement process expends energy, and risks producing no data or corrupting data which has already been stored; the last possibility could devastate the quality of the entire data set, rather than reducing the exchange rate by missing a single measurement.

When it comes to data production, sensor redundancy and sensor reliability are inversely related [5]. If sensor fidelity is scalable, small simple sensors may compensate for a larger more accurate one. Redundant sensing may provide enough low quality data that it becomes useful, or inexpensive simple sensors may be used to selectively operate a more capable power intensive sensor. Collecting less data means there is less redundancy in the system and each data point increases in value. Missing data points and device failure become less tolerable as the system loses redundant data production [5]. Processing low value data can increase its value; for example, averaging multiple points may produce a data point more useful than those that made it. Compressing data can combat system redundancy but increases the requirement for data security; compressed data points become far more important than individual data points. This process illustrates the trade-off between redundancy and reliability and its impact on data value [5].

Simulation Note:

The simulation assumes that sensors consume energy to produce measurements. Each sensor in the sensor array is not individually controllable at the present; it is assumed that they are duty cycled automatically by some fraction of their regular operational period. The simulation does not determine the effect of failure on the captured data set, aside from data loss or overwrite errors. It does anticipate energy related failures and will not perform partial measurements: operating below MIN_{OP} is forbidden. This means that the simulation cannot model devices that require long continuous access to volatile memory, because there is no means of handling possible faults. Devices that cannot handle energy-related failure should rely on appropriate system design, rather than energy management, to avoid these problems. While the data system of the simulation has come a long way, this area still requires more detail.

5.1.3 Data Storage

Modern data storage is small and cheap; environmental measurements occupy only a small number of bytes, so storage capacity is massive in comparison. There is no reason to risk undersizing data storage given its low price. Unless precise, high frequency measurements or some special file format is required, like images or video, storage is a simple problem. In the simulation of Section 4.3.3, data storage is divided into two levels, a data buffer and a data reserve. Environmental monitoring applications require non-volatile data storage, otherwise energy-related failure will cause data loss. While good design and energy management largely eliminate failures, nature can be unpredictable. A rodent eating a power cable should not cost an entire project months of volatile data. Because maintenance is limited, the responsibility for protecting data falls on the platform and its designers, not energy management.

The SolarNode, of Chapter 4, has a built in data buffer, so its capacity is fixed upon manufacture, but its data reserve is a removable SD Card. Just like how the energy reserve of the platform will change if different batteries are used, so the data reserve will change with a different SD Card. Given some rate of data collection and some time of deployment, an appropriately sized SD Card can ensure that *all* data collected by the monitoring platform, transmitted or not, is locally backed up and retrievable during the regular maintenance cycle. A large data reserve to safely store data increases the reliability of the system; local backups ensure that transmission problems do not reduce the quality of the data set. Data storage sizing is a simple, but critical task. By layering it, the platform can use small, cheap, low-power storage for small, time sensitive, operations, then move this data to higher capacity storage when the energy state is favourable. Including adequately sized storage is a simple design decision that provides the control system with very important tools for energy management. With data storage, the platform has the flexibility to delay energy intensive tasks, like data processing or transmission, which may be all that is necessary to prevent failure.

Within the simulator, data is treated as an unordered "pool". The simulation records the total number of packets, or bytes, it has in storage, but does not consider the contents of sensor measurements or incoming network transmissions. This is a weakness of the present simulation, but the simulation will quickly grow in complexity if each measurement is processed separately. However, to thoroughly model the network and data storage capabilities of the environmental monitoring device this needs to change. The simulator cannot implement data dependent power management techniques, like adaptive sampling, which uses the variability of sampled data to adjust collection rate, because it cannot observe the necessary state information. Additionally, because data storage is not organized, data received from the network cannot be prioritized with respect to local sensor data: the two data sources may overwrite each other unpredictably. At present, data storage acts like a counter, just like energy storage; the simulation knows how much data is present, but not what that data is. When the data buffer is full, the simulation responds by transmitting or storing measurements.

Controlling the movement of data is important. A networked device may need to share its data buffer with network traffic; however, should this buffer overflow, not only is the energy consumed to collect the local, now-overwritten measurements wasted, so is the energy used to determine which data had been lost and how to fix the problem. The importance of preventing buffer overflows is mentioned in [93] and [94]. Given how inexpensive data storage is recently, oversizing the data buffer will only marginally increment device cost and negligibly increase energy consumption. These costs are minor in comparison to the flexibility and security provided by a large data buffer. Even if some level of online performance is guaranteed, a simple mistake like unplugging the wrong battery could result in significant data loss without appropriate data storage.

5.2 Energy Considerations

The energy system of the environmental monitoring device provides the resource the data system needs to operate. Energy is consumed by the monitoring system to produce data. Three aspects of the energy system are considered here: energy production, energy consumption, and energy storage. Energy “production” is misleading: though energy harvesters produce energy from the perspective of the monitoring device, they are only converting environmental energy from one form (radiant, heat, kinetic, etc.) to electricity. Energy consumption is closely associated with data production. All of the platform’s systems require energy, and so must all be included in the rate of energy for data exchange. Energy storage is necessary for effective energy management because it provides a protective layer of energy security to the platform and ensures that the rates of energy consumption and energy production do not need to be perfectly matched.

5.2.1 Energy Production

Energy harvesting, discussed in Section 2.2.1, is an effective power supply for environmental monitoring platforms. Energy production, collecting environmental energy for use within the monitoring platform, allows the problem of energy management to change from a time and energy constrained problem to a power and storage constrained problem [29, 34–36]. The system does not need to initially store all the energy required for its deployment because a medium for energy to enter the system is available.

All the aggregated power collected from the environment, represented by $P_s(t)$ in Section 3.3, determines the maximum average power consumption of the hardware platform. If the average power consumed or lost by the system is greater than the average power entering through energy harvesting, the system will exhaust its resources and fail. The success of each application depends upon different energy production requirements. Harvester sizing, regardless of its impact on performance, is not discussed at length in this document. Sizing is assumed to be fixed, so the bounds of $P_s(t)$ are fixed; an energy management strategy is necessary to achieve desirable performance. Should this performance not be possible, then it is necessary to minimize failures and produce the best data set in that scenario. Performance below what is worth carrying out indicates a serious design flaw that is beyond the scope of a simulation-based investigation [39]. The fault-ridden results could then be used to rectify problems in the next iteration of design.

At the moment, solar panels dominate low-power energy harvesting. There are a huge variety of energy harvesting technologies, but only a limited number are developed enough for general commercial application. The intention of this work is to select a well known, commercialized technology and use energy management strategies to ensure good performance. The SolarNode has integrated solar cells; but any upgrades or expansions to them are beyond the scope of the present research. As for the AWS, the wind energy harvesters used in [64] were experimental and an example of an non-commercial product. Their performance was not well understood and required modelling work [66, 67]; there was no sizeable body of experience to determine how they would perform in extreme conditions. While the simulations in [64] utilized both wind and solar technologies, the "windbelt" is not suitable for the present simulation. If multiple energy harvesting technologies are incorporated, as in [54], sizing the solar panel is a different question than sizing the entire harvesting module. The average power of all energy harvesting technologies make up $P_s(t)$ and should be considered for the theory summarized in Chapter 3. A diverse set of environmental energy sources helps the monitoring platform overcome non-idealities in the environment at the cost of increased complexity. As the number of incoming energy signals increases, the complexity of modelling $P_s(t)$ quickly escalates.

Given that energy harvesting systems are ultimately responsible for all energy passing through the system, it is imperative that they are not undersized. While oversizing harvesters is expensive and wastes energy, undersizing harvesters guarantees failure. Even if waste it not an issue, harvesting technologies cost too much to oversize simply to ensure the load has plenty of resources to avoid energy management. Excessive overproduction may even need to be dissipated to prevent hardware damage.

Energy harvesting alone cannot deal with variations in the environment. By increasing the size of the harvesting module, the instantaneous power provided to the platform increases but this does not necessitate improved performance.

If there is no energy available for harvest, no increase in harvester sizing will prevent failure; this is a problem that can only be solved by energy storage. It is up to harvester module to ensure that there is sufficient energy entering the monitoring system to keep energy storage appropriately charged. The module as a whole must meet the energy requirements of the load, which means that the average power entering the system is greater than or equal to the energy exiting the system [35].

5.2.2 Energy Consumption

The energy consumption of the monitoring system is represented by $P_c(t)$ in Section 3.3; whether or not this includes loses, like leakage or self-discharge, is dependent upon the model. [35] treats energy loss as separate from consumption; however, since both intentional consumption and unintentional loss must be balanced with energy production, $P_s(t)$, they are discussed together. Energy consumption, including loss, is limited by the energy available in the environment through harvesting. If consumption and loss exceed "production", failure cannot be avoided. To ensure reliable operation, energy production must be slightly larger than the combined consumption and loss signal [5]. An ideal system would meet the *energy neutral* requirements discussed in Section 3.3; however, when designing a practical system, some redundancy, a safety factor or margin of error, is required to ensure performance.

Energy consumption has a different perspective on the energy system than the previous section. $P_s(t)$ is not controllable, but it is predictable, while $P_c(t)$ is not only predicable but also controllable. While production is based upon the environment, the platform can change its level of energy consumption.

Scaling energy consumption by changing the operational level of the system allows the platform to adapt to its environment [35]. Energy management decides how active the load should be: some techniques are summarized in Section 3.2. Extreme environments may require another step that involves estimating the future $P_s(t)$ so the present $P_c(t)$ may adapt to upcoming energy limitations in addition to the present environment [65]. The effort required to control consumption to prevent data loss due to device failure, is determined by the value of data collected; data value determines how reliable the system needs to be and it sets the level of operation of the device.

Repeating tasks, like measuring or transmitting data, that leave the system with nothing to do upon completion, can be grouped together to allow the operational level of the device to be divided into discrete levels or executed at different frequencies [72, 93–95]. The sensing action makes up a sizeable portion of the sensor platform's energy consumption, not only because its sensors consume energy, but because the hardware that supports them must also be left on. Even sensing elements that do not require power to operate result in energy consumption through data capture and processing [72]. Signal conditioning and measurement hardware, along with any necessary data processing, must be taken into account for the sensor energy consumption. The simulation has been designed to accommodate the energy overhead some sensing hardware may require to calibrate or configure before taking measurements. As long as the hardware platform consumes energy in a repetitive, predicable, event driven way, the present simulation can be used to accurately predict its energy consumption [71, 72, 95].

There are two useful parameters for describing the process of matching $P_s(t)$ and $P_c(t)$: the *minimum acceptable operational level*, MIN_{OP}, and the *maximum non-redundant operational level*, MAX_{OP}. These set two important boundaries for how the device operates. MIN_{OP} is the minimum set of operations the device can execute while remaining useful. Operating below this point is a waste of energy; it is preferable for the device to shut down or remain unused. MAX_{OP} determines when increasing or improving the executed operations stops producing increases in utility, or value; this is generally tied to data quality and is based upon data collection and the application. For the simulation, MIN_{OP} is set to one measurement per hour; if this action cannot be executed the device should shutdown.

As an example, consider the temperature control system in a home. The system must capture data often enough to provide a comfortable living environment. Operating the controller at rate so high that the home temperature does not fluctuate at all is redundant because the occupants could not perceive any improvements in comfort past a certain point. The minimum level of operation that is above the noticeable level of the occupants is MAX_{OP}, because any further increases in frequency do not provide perceivable improvements. On the other side of the scenario, if the heating system cannot operate often enough to maintain an adequate temperature inside the home, then it is not useful and must be replaced. The point before the system loses its value is MIN_{OP}. This point is not ideal, but is tolerable; the ideal operating point, given sufficient resources, is MAX_{OP}.

5.2.3 Energy Storage

Background information on energy storage is provided in Section 2.2.2. Energy storage is a critical component of the energy system. It protects the monitor from failure when the platform's control system demands consumption beyond what is available in the environment. Storage only provides the energy management system with the flexibility to match consumption and production; it does not allow the system to consume more energy than the environment can provide. Suppose that the average power production and consumption of some ideal monitoring system are perfectly balanced after a very long period of operation. Unless production and consumption change at the same rate over time, then storage is required to match these processes.

Energy storage has two important properties from the perspective of energy for data exchange. Firstly, energy storage is required to allow energy transfer between the source and load without device failure; it buffers the incoming and outgoing energy signals and simplifies the *control* required to match them. Secondly, energy storage acts as a filter by reducing environmental noise in the incoming energy signal; this makes the state of the system easier to *observe*.

These two properties of storage are exemplified by solar energy. The sun provides plenty of energy during the day, but none at night. If storage was not available, the activity of the device during daylight would be high, but unused energy would be wasted. Once sunlight stops for the day, the device must cease its activities regardless of the day's excess energy. Capturing excess energy for use at night evens out the operational level of the device and facilitates consistent data capture. The device does not need to try to consume excess energy during sunlight to prevent waste because storage makes use of it later. As long as energy production and consumption have matching average power, and there is sufficient energy storage, the system can operate without failure. Idealized natural energy patterns, like the daily solar cycle, are constantly disturbed by other natural phenomena. A weather event, say, a rainstorm, can suddenly reduce the solar energy for a few days, while completely clear days may provide more energy than usual. These changes to environmental energy act like noise on the incoming power signal; energy storage provides a way to filter it for monitoring platform. The incoming energy signal can jump and shift with the clouds and rain but the hardware will be able to maintain its steady consumption of energy.

Energy storage may conceal design errors in other areas of the energy system if it is oversized. For example, if an energy harvester is too small, but storage is massive, then the error in harvester sizing is offset by the excess energy in storage; the system operates at an energy deficit by consuming stored energy and does not fail. Some margin of safety is required for a high-reliability system, so storage must be oversized to some degree, but this quickly becomes expensive. If storage is too small, failure and data loss are certain to appear. If other, non-storage related components have been oversized to compensate for undersized storage, the design problem will not appear until the system is stressed in the right way. A small device with a large solar panel will not fail until there is a serious energy shortfall that cannot be buffered. This is backed up by the theory presented in Section 3.3, when production is too "bursty" to be adequately stored [35], and energy neutrality is violated. Provided the budget of the monitoring project is sufficient, it is better to have too much energy storage than too little. [64] provides a sizeable amount of work on determining the ideal energy storage capacity for an AWS.

The simulation must take into account the hardware that manages energy storage. While these components are typically simple, their coordination can be complex. Oversizing storage capacity in a simulation can result in problems. As simulations are inexpensive and non-physical, the two main disadvantages to oversizing storage, bulk and cost, are eliminated. From an electrical perspective, there are plenty of advantages to oversizing storage. Oversized devices are not cycled as deeply, so they have a longer lifetime, plus, their larger capacity compensates for more failures. This must be considered while analysing the results of the simulation. For example, the results from the AWS simulator in [93] and [94] are based on a 300Ah battery. This is a *huge* battery; it is still inadequate for the application, as the simulation shows, but to use one larger than this is infeasible. That said, the simulator does not care, simulating a larger battery does not cost more money or processor time; however, if the results of the simulator were ever used, attention to realism is critical.

Simulation Note:

The idea of energy storage as a buffer or filter is key to producing useful simulations. Energy storage must be appropriately sized to buffer the incoming and outgoing energy signals of the system. The simulation can use MIN_{OP} to accurately size energy storage by ensuring that its capacity is large enough to provide the minimum useful operational level at all times. Because the simulations are dependent upon solar energy to replenish storage, they must run for longer than most natural environmental cycles to adequately inform the user of its performance. The simulation must ensure that the impact of seasonal changes to solar energy on storage is observable. A harvesting simulation that runs for less than a full day is useless because it cannot capture daily changes in solar energy. In the same way, the seasonal solar energy changes of the Arctic require a simulation to run longer than a year for their effects to be observed. The energy stored at the start of the simulation must not be necessary for the device to survive the winter or it will eventually fail once this transient resource is consumed; that would be an example of storage compensating for design errors in other parts of the system. The seasonal changes in solar energy throughout the year act like an envelope signal for the daily solar energy cycle; the simulation should be able to observe both variations to size components of the energy system. By simulating more than one period, the simulation can demonstrate that, on average, the collected solar energy is more than what is consumed by the load, and that these two processes are periodically buffered without failure. This also demonstrates that energy management, rather than additional stored energy, actually improved performance.

There are two different types of energy storage in this simulation: energy buffers and energy reserves. Regardless of whether the buffer uses chemicals or capacitance to store energy, the idea is that it can both supply and receive energy from the system. There can be multiple layers of energy buffer, but both platforms presented in Section 4.1 have only one layer. The energy reserve is typically a single large, cheap, reservoir that is, generally, not rechargeable. They are typically primary energy storage technologies, but their definition is not dependent upon their internal construction or capabilities, but on how they are used within the hardware platform and the simulation. Both these storage concepts have a grey area and are loose classifications for differentiating between concepts. The two storage types differ in function: buffers are discharged and recharged often with respect to the length of deployment, and energy reserves tend to be large and replaceable. The two hardware platforms described in Section 4.2.1 and Section 4.1.2 have examples of how these two storage strategies are used.

AWS Simulator Storage

In the simulator, the AWS has a very large, rechargeable, lead-acid battery bank. The primary objective of this battery bank is the help the device survive seasonal energy shortfall. While it does buffer the daily solar energy cycle, its real value is providing energy during the winter. Its main function is to buffer the seasonal solar energy signal by distributing excess energy harvested in the summer to the sunlight-absent winter. The AWS does not have tiered storage like the SolarNode, so its storage technology must take on the roles of both filter and buffer. The sealed lead-acid battery model stores current values related to the adjusted amp hour capacity of the battery based on temperature; its dependence upon temperature required that the hourly ambient temperature be provided to the simulator. The code for the AWS energy storage is unmodified from Appendix A2.6 of [64] and is not included here.

SolarNode Simulation Storage

The SolarNode clearly defines the roles between energy buffer and energy reserve. Two ultracapacitors buffer the reserve from the solar cycle by fulfilling the daily energy requirements of the platform. If an energy buffer is present, then it should be responsible for adapting to the environment and for regulating device operation. The buffer effectively filters the daily and short-term weather variation out of the incoming energy harvest. The energy reserve is only used when the energy buffer is exhausted. That said, the buffer does not make the energy state of the platform easier to observe; it oscillates with the daily energy cycle and must be preprocessed by the controller before selecting a control action. Taking a daily average of the energy buffer level captures the envelope of the daily power signal, and does a decent job of filtering out this periodic energy change.

The energy reserve for the SolarNode is a pair of AA batteries. While these batteries may be rechargeable, the SolarNode does not have this capability, so they must be replaced by regular maintenance once exhausted. In comparison to the storage capacity of the supercapacitor energy buffer, the battery based energy reserve is very large. With small environmental energy variations filtered out by the energy buffer, the energy reserve is only engaged to fill in energy shortfall due to seasonal variation. It is assumed that the reserve will operate until it fails.

The energy buffer of the SolarNode is about the right size to filter out short-term energy variation; however, it is integrated with the hardware and cannot be changed without refabricating the platform. The SolarNode uses energy values in all its computations, so the energy buffer is modelled as a "pool" of energy constrained in size by the operating range of the solar charge controller. This is mentioned in Section 4.1.2. The energy reserve is based on a datasheet for lithium AA batteries [15]. The amp hour capacity of the batteries, at a reasonable level of current draw, is converted to some amount of stored energy in Joules. In the future, a more detailed model of the supercapacitor energy buffer may be implemented [96]. Because the storage technologies used by the SolarNode are more tolerant to extreme conditions, lithium batteries and supercapacitors are largely unaffected by temperatures above $-40°C$, environmental variables are not passed to the SolarNode simulation.

5.3 Energy Management and Exchange Rate

The theoretical framework explained in Section 3.3 provides a basis for designing the monitoring system, while the techniques in Section 3.2 ensure that it efficiently invests energy while producing data. This will facilitate a reliable exchange of energy for data. Three topics are considered here: trivial cases of energy management, an analysis of the energy for data exchange rate, and a discussion on managing this exchange.

5.3.1 Trivial Solutions

Consider two energy management scenarios where environmental adaptation is trivial: one where no energy harvesting is present, and another, where there is never less energy collection than consumption, so storage is not required. The monitoring device without harvesting capabilities continually operates at an energy deficit until all energy is consumed, then it fails. The monitor is isolated from its environment because no harvesting module is present to bring energy into the system. Energy management does not need to adapt to the environment; the system begins its application with all the energy it will ever have and typically operates at a fixed rate until said energy is exhausted. From this perspective, a finite energy supply results in a finite data set with the quality of the data determined by the rate of energy for data exchange. The exchange rate is fixed and the value of the data points are generally fixed as well. Some management on the data side of the exchange may occur, but it cannot prevent device failure. In the other situation, where the monitoring device always harvests sufficient power to meet the demands of its load, the monitor never suffers a power or energy deficit and can operate without storage technology. Energy management is not required for this “naïve” control system [35]. The value of each data point is very low because there is no “energy scarcity”.

Neither of these conditions are dependent upon their environment. The first example is guaranteed to fail after depleting its energy reserves, regardless of where it is deployed and how much energy is available, because it cannot collect energy. The second example has so much energy available it is completely buffered from any changes in the environment. While the environment constantly changes, the monitoring device never *observes* scarcity; it is constantly wasting excess harvested energy. Energy management becomes interesting when the energy supply, in addition to storage, is sufficient to meet the demand of a device operating at some acceptable performance level. For the remainder of this paper, it is assumed that some form of energy harvesting technology is present to make energy management an interesting problem at the software level, as mentioned in Section 3.2. Just as economics is only interesting when there is some form of scarcity, so too is energy management only interesting when there is some “energy scarcity” to confound the supply of, and demand for, energy.

5.3.2 Exchange Rate

The energy for data exchange of a remote monitor consuming energy to capture data is analogous to an economy. In this analogy, energy is the resource used to produce data, while the “price” of data, the abstraction of its relative value, is set by supply and demand. The monitoring application determines the demand for data and the desirability of its properties, while the supply is based on how much of this data is produced by the monitoring system. "Energy scarcity" ensures that the supply of desirable data is limited. Energy management is the environmental monitor attempting to navigate the market. This explains why it is only interesting under the condition of energy scarcity, as postulated in Section 5.3.1: without scarcity, there is no economy because supply always exceeds demand. While the quality of the data collected by the monitoring device is evaluated relative to the application, the rate of exchange between energy investment and data collection is dependent upon how effectively the energy management strategy controls the device's actions [72,93,95]. The exchange rate is how much data, or what level of performance, is obtained for some amount of consumption. The maximum rate of energy for data exchange is limited by the monitoring system's hardware. This analogy provides a convenient way to model this: energy for data rather than money or resources for goods and services.

Ultimately energy scarcity forces energy management to the forefront of research in the field. As seen in Section 7.4.3 and [95], stable, energy-rich environments require only rudimentary management strategies; there is very little energy scarcity, so management has limited impact. Research in the field of energy management is required because it is too expensive to oversize energy harvesting and storage for an entire network of environmental monitoring platforms. The supply of data is limited, but the demand remains. Management improves how effectively the energy provided by the harvesting module is invested in the monitoring platform's task; ideally this allows the use of smaller, and thus possibly cheaper, harvester and storage modules. There is some suitable balance between storage and harvester sizing that provides enough average power for the platform, with sufficient storage to buffer its operations, while guaranteeing some level of performance with limited waste.

From the perspective of this energy and data market, the data collection process is largely responsible for establishing the rate of exchange. The monitoring platform's energy overhead needs to be amortized over all the measurements produced while deployed to accurately determine the cost of data production. Given the wide range of sensing technologies available, it may be possible to reduce the energy cost of measuring some environmental variables by changing sensors. Notice that the controllers, in Section 6.3.2, used to govern the energy-for-data exchange, all reduce the value of sensor measurements as data buffers approach their maximum capacity. The value of collecting data that cannot be stored is *negative*, because energy is invested to capture data which will overwrite preexisting data that also consumed energy. The new measurement produced one point at the cost of another point, and has likely fragmented an earlier part of the data set. Unless the new data is far more valuable than the old data, it would be better to save the energy for other operations. Given that the SolarNode simulation cannot presently determine the relative value of individual data points, as their content is hidden, this computation cannot be performed. This is why appropriately sizing data storage is so important. While two data sets may not be directly comparable, they can each be evaluated with respect to their rate of data production relative to energy consumption. These properties can be weighted and considered together when deciding how to best invest available energy [93, 94]. This is used in Section 7.4.3 to compare monitor performance in two different regions. An amusing side effect of this exchange model is that inefficient energy consumption could have a higher exchange rate than maximum energy efficiency, if it produced data of sufficiently high quality to compensate for the wasted energy.

This is similar to inefficiently produced luxury items commanding a high price on the market.

Secondary factors in determining data quality, such as cost and convenience should be included in determining the rate of exchange. Consider the spectrum of wireless communication options available today. Let there be two transmission media identical in every way except energy efficiency: one option is slightly less efficient than the other. If the less efficient medium is more widely used, making data transmission more convenient, then it provides data accessibility, which may make its data set more valuable than the other, more efficient, option. The rate of exchange favours the less energy efficient option because it increases data value more than it increases energy cost. Even though wireless transmission is a massive energy investment for the hardware, it may become a necessity simply because it is too much trouble to obtain the data otherwise. This may add only small increases in data accessibility with large increases in energy investment, but still adds enough value to the data set to make it worthwhile. Purchasing hardware or establishing wireless infrastructure is expensive, and though it may significantly improve data quality, financial limitations are always a concern.

Finally, waste is a relevant constraint as well. Oversized harvesters or undersized storage will result in wasted energy; energy which is not used to produce data, effectively increases the energy cost of the data which is produced. Even if the device completely eliminates all energy related failures and is operating at MAX_{OP}, waste still reduces its rate of exchange. Waste ultimately hurts monitoring projects in other ways: funding spent on oversized harvesters could have been invested in improving data quality in other ways.

5.3.3 Managing the Exchange

The objective of environmental monitoring is to produce high quality data. Energy management facilitates this by carefully investing the monitor's energy resource to improve device performance. Management becomes an interesting problem when the ideal performance level cannot be reached by the naïve strategy; a case where this could be achieved has a design problem, not an management one. When the naïve strategy does not meet the needs of the application, a strategy must exist to ensure resources are effectively consumed [72, 93, 95]. Providing a range of operational levels allows energy management to become flexible. While true, perpetual, *energy neutral* operation may be impossible, energy management allows some performance guarantees to be made. As long as the energy captured from the environment is sufficient to meet the demands of the load, while keeping the system load operating somewhere between MIN_{OP} and MAX_{OP}, then the system is guaranteed to function. Determining what level of activity is the best choice for a certain state requires value to be associated with each operational level. The exchange rate of energy for data is used to determine how valuable the energy consumed is, based upon the value of the data it produces.

This is especially true when dealing with finite energy reserves: without some MIN_{OP} it is difficult to estimate when to consume the limited reserve or when to save it for later. When keeping the device operational is the key, as it is in the Arctic [71], the goal should be to "maximize the minimum duty cycle" [43]. This goal can be expanded and applied to the yearly cycles of the application: to maximize the minimum duty cycle on a periodic basis. With this policy in place, the energy in the reserve is now as valuable as the data produced from the load at MIN_{OP}, in addition to the loss of value accrued if the reserve was not engaged; this would be the “opportunity cost” of failing then, and using the energy later. To meet energy requirements that are outside the capabilities of the energy buffer, it is important to size the energy reserve so that it can achieve MIN_{OP} without failure. Data produced at MIN_{OP} without using the energy reserve has slightly less value than when it is produced using the energy reserve, but that is because imminent failure is not a factor.

This is only one possible strategy, and the operational level of the device need not be constrained to MIN_{OP} when operating on reserve energy. However, it is important that some MIN_{OP} be defined to give other operations some value in the energy-for-data exchange. Consider a circumstance where the device is operating at its maximum operational level, MAX_{OP}, and has an unforeseen, but brief, energy buffer failure. If the energy reserve is engaged to prevent device failure, the device should not immediately drop to MIN_{OP}. This makes the data set inconsistent because the failure would be temporary and the device will snap back to MAX_{OP} upon recovery. The energy management strategy should be able to determine that is worth the extra short-term expenditure of reserve energy to provide the extra few points of consistent, high quality data while the buffer recovers, based upon how data consistency is weighted relative to the value of the data set. This decision can be made because some value has been attached to the energy in the reserve that allows it to be compared to the energy in the buffer and the rate at which data is produced by the monitoring platform.

Optimizing the energy management strategy will improve the energy for data exchange rate. Preventing energy waste and eliminating device failures should reduce the energy required to produce data and increase the total value of the data set, improving the exchange rate. The goal of optimizing the rate of exchange is to produce the highest quality data set for a specific application using the smallest possible amount of energy. The fitness of an energy management strategy should be determined by data production [94]. Optimization can take the form of improving the quality of data production more than a corresponding increase in energy cost, or a reduction in energy cost for some constant level of data quality. Optimization can still reduce waste even when the device is certain to avoid failure, so it is generally a good idea.

One method of optimization eliminates energy waste by aligning energy over-consumption with energy over-production. Management confines and concentrates energy intensive operations to periods of energy surplus. This avoids losing energy to storage overflows, so there is an effective reduction in the energy cost to the set of actions because waste is reduced. The data quality may not change, but the impact on its energy cost is significant. If the energy management strategy uses excess storage to stockpile energy to take advantage of "economies of scale" by performing actions in batches, then it has turned a potential sizing problem into an advantage and improved the exchange rate. It has also discovered an interesting energy saving behaviour. This is demonstrated by offloading the communication function of the AWS to periods of extended sunlight to conserve energy during the long nights of the winter in Section 7.2.2 and 7.3.1 [93, 94].

This can be extended to any delay tolerant tasks [35]. Delaying transmission slightly inconveniences the user, but allows efficient energy investment. The same could be said for other computationally intensive, but not time sensitive, operations. Estimating energy availability in an environment could be confined to day time when solar energy is plentiful, while after sundown only measurement actions are executed. This indicates that wasted energy harvest *must* be included into the rate of energy for data exchange to some degree, as mentioned in Section 5.3.2, and supports the inclusion of “break-even cycles” from [5], to accurately asses the effect of optimization on the exchange rate.

The previous example reduced the energy cost of a group of actions by executing them more efficiently. Consider another way to improve the exchange rate: increasing the value of a set of actions faster than increasing its cost. Providing wireless transmission capabilities to a monitoring platform is energy intensive and costly; however, wireless access to remote data easily outweighs the energy cost of the transmitter. This is particularly true in the Canadian Arctic. The value of the data set, due to the improvement of the accessibility quality, is increased more than the increase in energy investment. Both the addition of the transmitter, and the management techniques associated with it, contribute to improving the rate of energy for data exchange.

With this in mind, what other techniques may improve performance on the data end of the exchange? Limitations on data considerations in environmental monitoring are different from those of energy considerations. There is some overlap, but the major difference is that while energy management struggles with limited resources, data management struggles with limited logistics. The data system may not be capturing, processing, or transporting data in the best way. If data storage or network bandwidth are limited, but computational resources are not, processing measurements may be a useful strategy. This can reduce the number of bytes transmitted and stored without significantly changing their measurement information. It produces a smaller data point of higher value that demands less energy to manipulate. This is similar to the “economies of scale” approach on the energy side, except instead of a resource stockpile used to efficiently execute actions, a “value stockpile” of compressed, or otherwise value-enhanced, data can be moved about using less energy.

Chapter 6

Fuzzy Control for Energy Management

Control is the basis of energy management. As discussed in Section 2.3, fuzzy systems provide a simple, robust technique for selecting appropriate actions based on state information. This chapter explains all the control strategies used to generate the results in Chapter 7.

6.1 Naïve Control

Naïve control is an uninformed process for determining how an environmental monitoring device should operate; it does not adapt to its environment and does not take into account state information. Naïve control refuses to make the performance-complexity trade-off required for adaptive energy management strategies. The simulation results in Section 7.2.1 show that this trade-off is favourable, and that only a small increase in control complexity can result in massive performance improvements. From this perspective, naïve control is not a good idea, but addressing it is necessary to provide a complete basis for

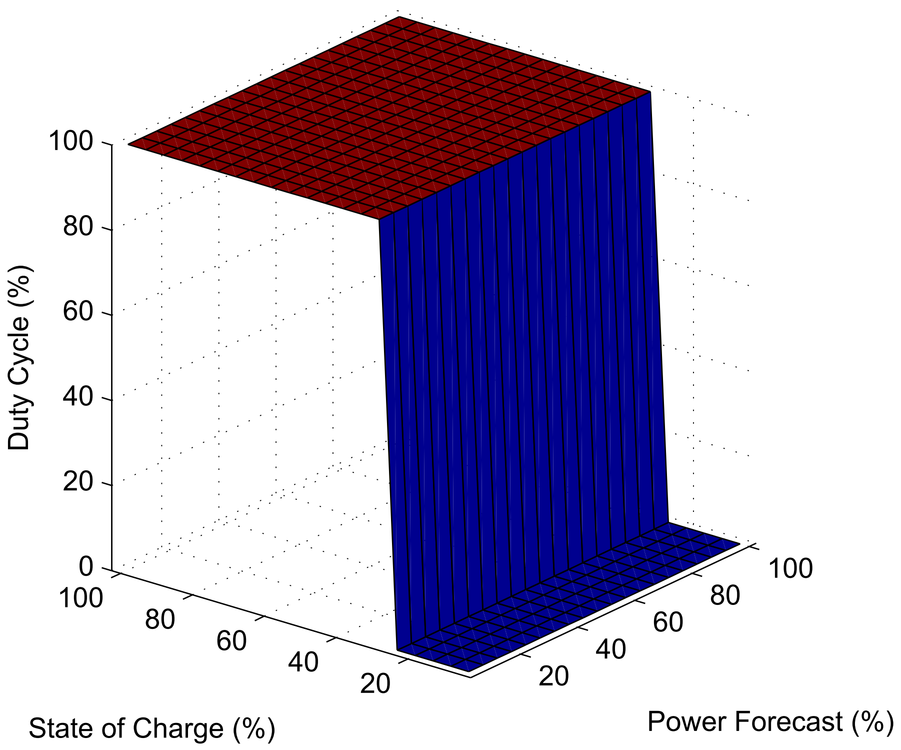


Figure 6.1: This surface shows the input to output mapping of the energy management strategy for Naïve control. This is not a lookup table or controller like other plots in this chapter, but rather a mapping of what control actions the current design of the AWS will execute in a given state. When executing this strategy no state observations are made.

further discussion. With the shift in sensor networks towards simple, cheap, highly redundant hardware platforms, energy management cannot be ignored given its many benefits for such a small increase in complexity.

AWS The naïve energy management strategy used with the AWS simulator of Section 4.2 is simple: operate at the maximum useful duty cycle, MAX_{OP}, until the battery state of charge (SOC) drops to a point where it is unsafe to continue, less than 25%, then completely shutdown. Additionally, the satellite

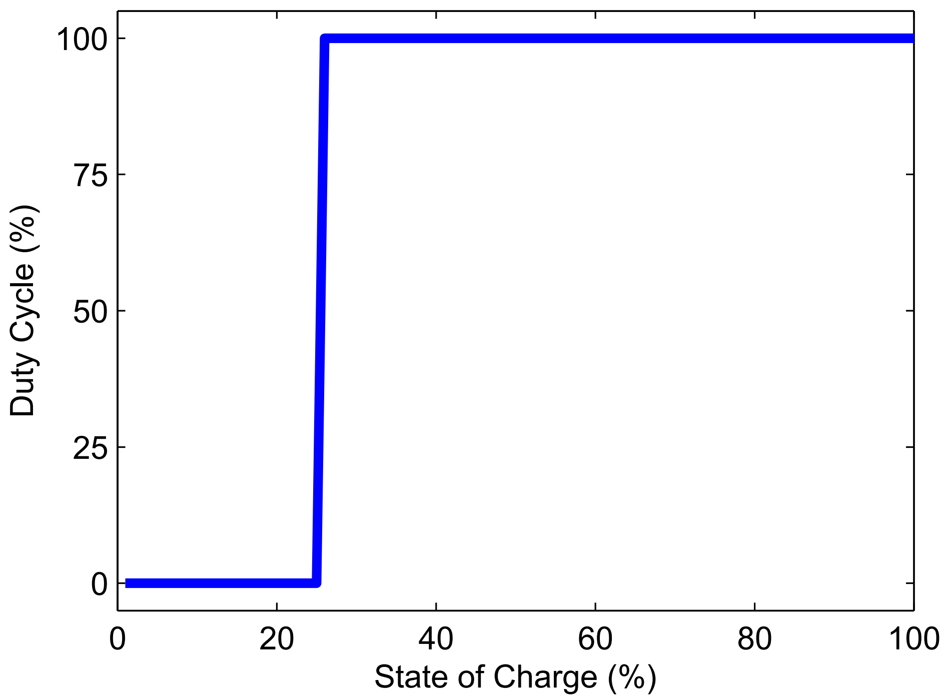


Figure 6.2: This figure is a compressed version of Figure 6.1. The output is the same for all states in the "Power Forecast" dimension, so that dimension may be removed. Remember, under the Naïve strategy, the AWS never uses this input to output mapping; this is presented to summarize how the AWS has been designed to operate.

transmitter immediately returns all collected data. If there is sufficient energy available through harvest and storage, then this strategy provides the best possible data accuracy, consistency, and accessibility; however, once this energy is exhausted, the availability and consistency of the collected data drops to undesirable levels, below MIN_{OP}, and the platform enters shutdown mode. This problem is illustrated in Figure 7.8. The system attempts to resume its maximum level of operation, MAX_{OP}, at any time when the available energy exceeds the shutdown condition: a battery SOC less than or equal to 25%. The naïve strategy cannot adapt to different environmental energy profiles by changing its duty cycle, and thus its data quality, regardless of how valuable data properties are in relation to one-another [93]. A plot of this energy management strategy, with respect to system state, is provided in Figures 6.1 and 6.2. There is no control module to use this information; the figures only shown how the hardware is configured to operate. All devices, no matter how simple, have some energy management strategy designed into them.

SolarNode There are two naïve strategies used with the SolarNode simulation of Section 4.3. These strategies, which attempt to achieve some design goal using a static operational setting, do not adapt their actions to their observed hardware state. The first naïve strategy is referred to as "Minimum": it attempts to avoid device failure by operating at the minimum acceptable level, MIN_{OP}, and consuming the smallest amount of energy possible. It does not adapt to the environment and wastes a large amount of harvested energy in the summer due to the limited storage capacity of the energy buffer and its low energy consumption. It collects the least amount of data, only one measurement per hour, but rarely fails. Figure 6.3 shows the energy manage-

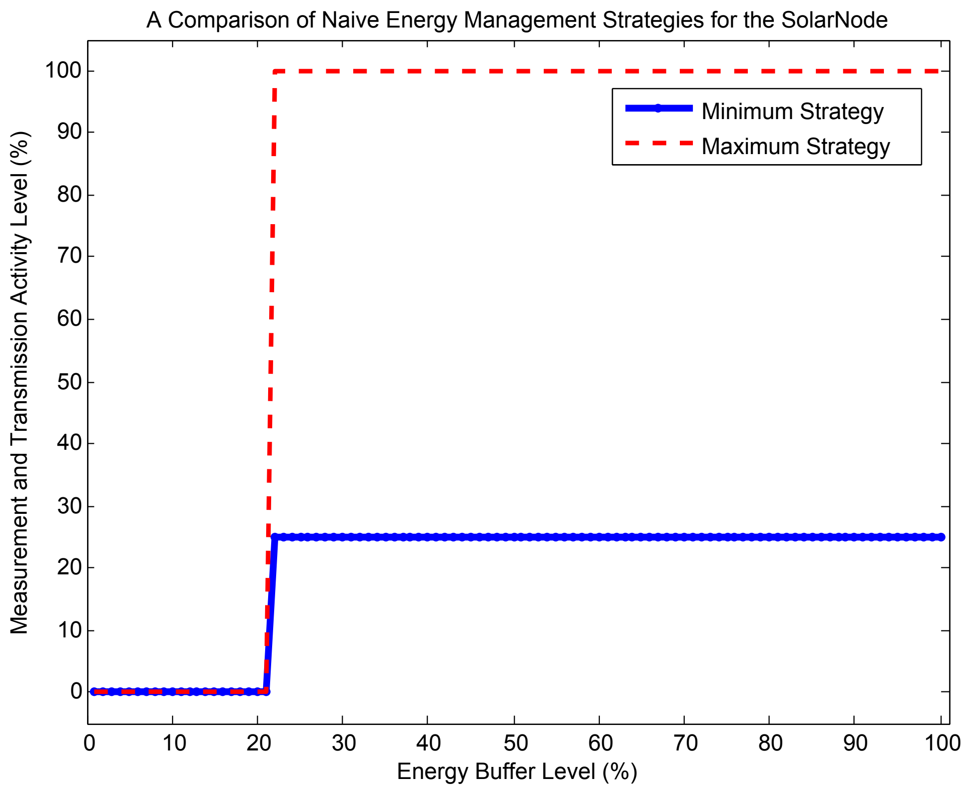


Figure 6.3: This plot displays the naïve energy management strategies with respect to the energy buffer. The outputs do not change with respect to the data buffer, so that dimension has been left out. The naïve strategy does not observe state information, so this surface is not used by the control module; rather, it is a mapping of what the platform hardware has been designed to execute upon deployment. The Maximum (MAX) strategy executes 4 measurements per hour unless the energy buffer is at, or below, 21%, at which point the hardware automatically disables the energy buffer. The Minimum (MIN) strategy executes a similar plan, but it only collects 1 measurement per hour while operating.

ment strategy of the device with respect to its state, even though a control module does not manage its operation. The second naïve strategy is referred to as "Maximum": it attempts to maximize the amount of data it collects by taking measurements as often as desirable, MAX_{OP}, and storing them frequently. This consumes a great deal of power, but captures far more data than the "Minimum" strategy and reduces the amount of harvested energy wasted during the summer. The "Maximum" strategy is not concerned with how often its operational level causes it to fail, so it is vulnerable during the winter months when environmental energy is low. The "Maximum" and "Minimum" strategies capture four measurements per hour and one measurement per hour, respectively [72].

The AWS naïve strategy is tested with Arctic data from Section7.1.1 and the simulator from Section 4.2.1. Its results are provided in Section 7.2.1. The SolarNode naïve strategies are tested with Boreal data from Section 7.1.2 and the simulation from Section 4.3. The results of the simulations are also presented in Section 7.2.1 and expand on the information provided by the AWS simulation. They are compared to the competing adaptive strategies in Section 7.2.

6.2 Applied Fuzzy Control

Fuzzy controllers for energy management on environmental monitoring systems map some input conditions (controller states), to some device operation (controller actions). The state describes what is happening to the monitoring system. Not all properties of the system are considered part of its state; only a limited number of variables can be internally measured by the hardware. On top of that, only a few of those variables are relevant to the control process. The state informs the controller, which determines the actions of the system. Classical control considers an action yet another state variable, but here the term from machine learning is used instead. In machine learning, “actions” are what an agent, in this case the controller, does after observing a state.

Fuzzy controllers operate using fuzzy sets covering the input and output dimensions of the control space. These sets are organized into a fuzzy inference system, which associates them using fuzzy rules. At present, an “expert” designs and tunes the fuzzy rule base. Because the universe of discourse of all input and output dimensions must be fixed before any computations can take place, all possible inputs and outputs to the controller are accessible when the control system is offline. The entire input space of the controller can be passed to the fuzzy inference system before the controller is activated, allowing all possible input states to be mapped to output actions. Once this is completed, the controller can be implemented as a simple lookup table. The indices of the lookup table are determined by state information, while the entries in the lookup table are predetermined controller actions.

The required activities of the sensor platform determine its energy consumption; however, the environment, and thus its energy supply, is constantly changing. A good fuzzy RBS will map all possible states the remote monitor could observe to the actions that will produce good quality environmental measurements and avoid device failure [93]. The objective of the energy management strategy is to allocate the energy available to the device, harvested from the environment or previously stored in the energy buffer, to the activities which will allow the best quality data to be collected by the remote monitoring device.

6.2.1 Observable State Space

AWS The AWS platform has two main state variables: the SOC of the attached sealed lead acid battery, in percent, and a prediction of future power available to the energy harvesting module of the platform. The battery SOC informs the controller how much energy is available and allows the controller to estimate the time of year. If the SOC is consistently high, then it is either summer, or the energy harvester module is oversized; both circumstances indicate a low risk of energy-related failure. The future average power available to the device, or power forecast, is based on climatological data, a typical meteorological year (TMY), collected and averaged over a period of thirty years. When the future average power is provided as a percentage, it warns the controller of a change in season. If it drops to a low percentage of its maximum, then the controller knows that the climate is expected to change for the worse: winter is coming. The controller can then reduce the operational level of the device to pre-emptively save energy to survive the change in season.

The only state variable that provides feedback to the controller is the battery SOC; the average power forecast is apriori information from device memory so it cannot be changed by device operation. As the controller modifies the operational level of the device by selecting actions, the SOC is depleted or charged at different rates. This rate determines the next SOC state value, and thus future control decisions. This provides stable, closed-loop control of the system illustrated in Figure 6.4.

The fullness of the internal device memory, in percent, is not included in the state-action mapping of the fuzzy controller, but is still important to the simulation. The satellite transmissions to empty this data buffer are very energy intensive. To save energy during the winter by safely delaying satellite transmission, the AWS simulation must observe the data buffer level to prevent buffer overflows. A SOC value acts as a threshold to determine when the transmitter should be used, while the percentage of data in the internal memory determines the number of transmissions that should occur when above the SOC transmission threshold. When more data stored in the buffer, more energy is dedicated to making transmissions to ensure the buffer is emptied safely. Once the SOC falls below the transmission threshold, the satellite transmitter is deactivated to conserve energy. A similar strategy is implemented in Section 6.8 of [64]; however, it makes transmissions based upon excess harvested energy rather than observing the hardware's state. The present control system does not observe the excess harvested energy, so the transmission scheme had to be changed to its present form.

SolarNode The SolarNode controller also observes two state dimensions: the amount of energy in storage and the amount of data in storage. Both these values measure, in percent, their respective buffer level. These two state variables allow the SolarNode controller to determine how much energy is available to be invested in data collection, and how much data has been collected recently. The SolarNode does not consider its energy or data reserves in its control system at the present. While both of the reserve state variables could be measured, the current energy management strategy avoids these extra control dimensions for simplicity. The data buffer value can be passed directly to

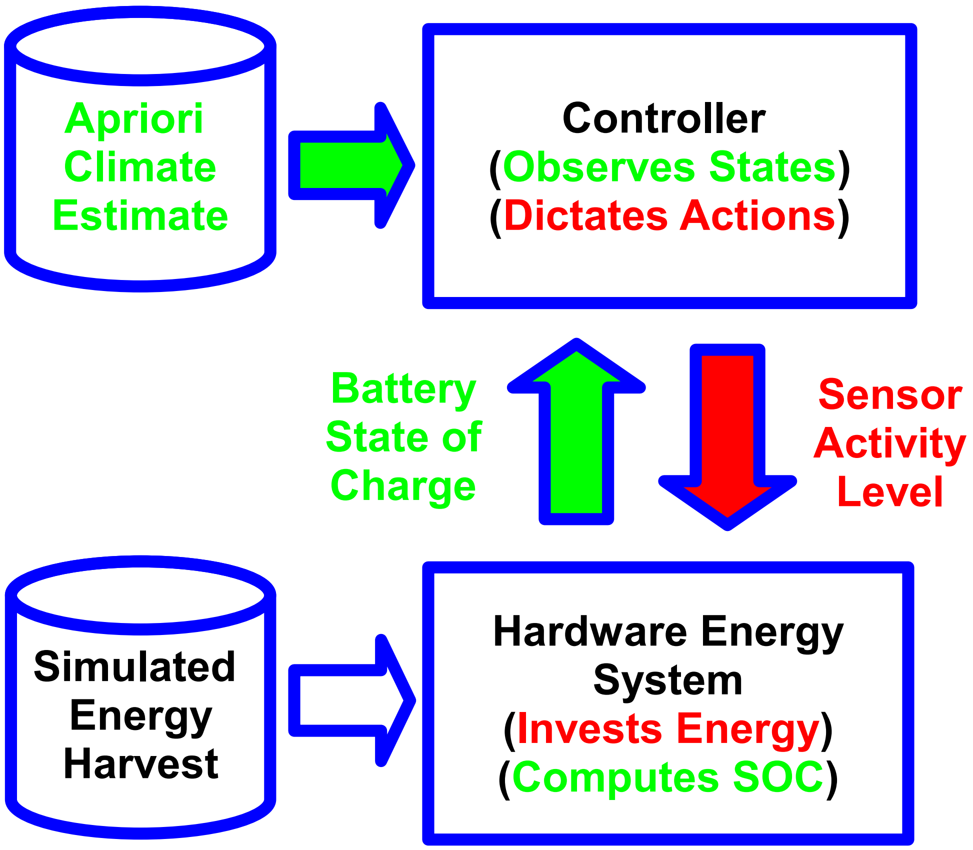


Figure 6.4: This a simple model of the control loop linking the AWS controller and the hardware. This simplification does not reflect the actual Simulink model, as shown in Figure 4.1, but rather how the controller interacts with its surroundings. Information and signals relating to states are in green, while those relating to actions are in red. The "Simulated Energy Harvest" is not state information, as it is not observed. To the controller it appears as noise injected into the system that affects the SOC state value as the battery is charged by the solar panel. The control loop is formed by the association of "Battery State of Charge" state information with the "Sensor Activity Level" action information.

the controller. Because the capacity of the energy buffer is similar to the daily energy consumption of the device, the change in the average energy level of the buffer can be used to estimate the energy harvested from the environment throughout the year. This allows the energy management strategy to select actions that ration energy resources throughout the year. To prevent the diurnal solar cycle from occluding seasonal changes to the energy state variable, due to the size of the energy buffer relative to the harvesting module, the energy buffer level is averaged over a period of 24 hours.

The controller receives closed-loop feedback in both of its state dimensions. The state of the energy and data buffers inform the actions of the platform, which cost energy to collect and manipulate data. This, in turn, affects the energy and data state of the next time step, closing the control loop. The environmental energy harvest is the primary influence on the energy state of the platform, but it cannot be directly observed or controlled by the SolarNode. The SolarNode waits until environmental energy has entered the energy buffer before taking it into account. Its control system is focused on using clearly *observable states* to inform *controllable actions* and, unlike the AWS, requires no apriori knowledge or auxiliary state information to function.

It is important that the controller's state variables are abstracted from hardware values. As mentioned above, the control systems for both the AWS and the SolarNode use percentages of observed hardware values as state information, rather than raw physical values [64]. An energy buffer at 50% capacity lets the control system know there is still energy left to invest, but tasks that are not critical should be delayed to save energy. A raw energy value, of 1000J for example, does not provide context to the controller, which is necessary to operate in a constantly changing computational environment.

This is important for atleast two reasons: modularity and design convenience. First of all, the controller must be able to account for subtle changes in parameter sizing within the simulation, the context mentioned before, if it is going to be optimized computationally. A genetic algorithm may test a huge number of differently sized energy buffers; in each case 50% of storage capacity provides useful information to the controller, while some amount of energy in joules does not have a predicable meaning. This formalizes the format of state information and allows the control module to be transferred between different platforms with identically formatted state information. This separates the control and hardware modules and is essential to modern object oriented programming techniques and software design methodologies. Additionally, it allows different control techniques to be used for the same hardware platform in the future. Because the controller is a module, it can be moved about like any other standardized part. The second reason that the state information is passed to the controller as a percentage is for design convenience. The fuzzy controller requires that the entire state space be covered by the fuzzy rule based system. If the controller and hardware are to be computationally optimized, then the size of the state space must be predictable and standardized to ensure its whole range is covered. Regardless of the actual size of the data or energy buffers, the fuzzy controllers can still be designed before simulation because the state variables have fixed ranges.

Processing Both the AWS and the SolarNode filter their control information. Filtering state information is a necessity for two reasons: raw state values vary with the environment and discrete controllers have discontinuous action spaces. Because both platforms are dependent upon the sun for energy, state information is tied to the diurnal cycle. This pattern is randomized by weather and climate, so it is constantly changing. If a continuous controller manages device operation, selected actions continuously vary with the state information; however, in this application, the state information accesses discrete values in a lookup table. This is unavoidable because actions are discrete events, so these small variations in state can cause large operational changes when nearby states are mapped to different actions. As the monitor approaches a discontinuity in the discretized control surface, the daily cycling of harvested energy pushes the controller back and forth over the change in the action space. This causes the output of the controller to oscillate and degrades the consistency of data collection. It also demonstrates that control is not effectively exerted over the device. This was a serious problem for the AWS that was addressed by [94]. Controllers for the AWS solve this problem by enforcing some minimum Hamming distance on the state space between action transitions to ensure than the state trajectory of the platform passes over the discontinuity in the action space rather than oscillating around it. This is demonstrated in Figure 6.5. The controller now assertively changes its actions based on significant changes in state. The SolarNode faces a similar problem. Because the energy buffer is similar in size to the daily energy requirements of the platform, the energy buffer may completely cycle over the period of one day. This will push the state trajectory of the platform completely around the state space and cause its output to "bounce" throughout the day as it attempts to compensate for the nightly energy shortfall. To solve this problem, and ensure that the SolarNode controller actually adapts to its environment, the average energy buffer level of the last 24 hours is passed to the controller, rather than the instantaneous energy buffer level. Though this has not been tuned extensively, it helps limit controller oscillations. Both strategies make a tradeoff: the sensitivity of the controller is reduced to provide some immunity to the noise of the environmental energy signal the platform is tracking. The "expert" assumes that reducing the sensitivity of the device will not result in disaster; weather changes often, but within some predictable bounds.

6.2.2 Controllable Action Space

The AWS has two controllable actions: data collection frequency and transmission size. The data collection frequency is bound between 4 times per hour, MAX_{OP}, and once per hour, MIN_{OP}. The action is reduced to zero should it be below MIN_{OP}. The values must be whole numbers, as partial measurement cycles are invalid, and an action must be assigned to every possible point in the state space. This means that the controller must intentionally decide to collect no data when it is in a state where collection is impossible; it may not be left to some hardware safety setting as it was in [64]. If control actions and hardware parameters disagree, it can create internal control conflicts. This sort of conflict makes the fuzzy controller invalid, as it is analogous the the hardware having fuzzy control rules that contradict those of the control module. The actions space of the controller must completely reflect all necessary actions of the hardware platform. This is done by designing the fuzzy rule base such that it respects, for example, the hardware requirement of a sealed lead acid battery to avoid complete charge depletion [94].

In addition to the number of measurements collected per timestep, the AWS is also able to control the number of satellite transmissions it attempts

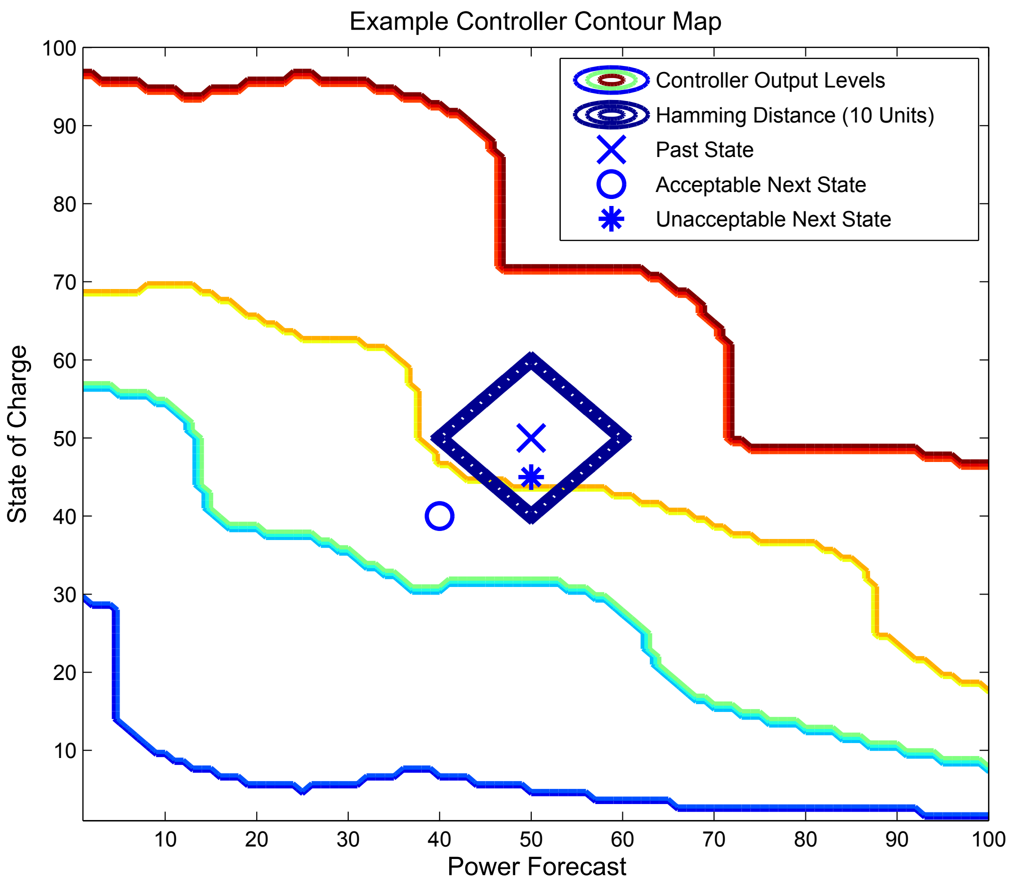


Figure 6.5: An example of how Hamming distance is used to filter state oscillations. Three states are shown: a past state, an acceptable next state, and an unacceptable next state. If the next state is unacceptable (within the large diamond) then the controller will not change its output. This image comes from Figure 4 of [94] (©2013 IEEE). The contour lines represent discrete level changes in the example controller. For example, oscillating over this region may change the measurement frequency between, say, 4 measurements per hour and 2 measurements per hour, depending upon how the controller is constructed.

per timestep. This was not fully explored, but it should be mentioned as it is controllable by the AWS. The effects of data storage and delaying satellite transmission are shown in Figures 7.9 and 7.10.

The SolarNode has more detailed, low-level control of its operations than the AWS, so its controllable action space is potentially much larger. The number of actions will increase if networking applications and capabilities are brought into the model. At this time however, the SolarNode has only two controllable action dimensions that make up the energy allocation process. The first dimension involves the frequency per timestep that the node executes a measurement cycle, transmits the result to the local network, and stores it in the data buffer, while the second dimension is what percentage of the data buffer is transferred to the data reserve. While the fuzzy inference system is used to build the controller, to the control module, the energy management strategy appears as two lookup tables that determine the monitor's actions: how often to take sensor measurements and transmit them wirelessly, and how full the data buffer should be before it is emptied into long term data storage. Each operation within the action dimensions may not be executed independently: measurement and transmission are dictated by the same control output, while storage is a separate process. The SolarNode will measure and wirelessly transmit data at whatever interval is dictated by the lookup table in its memory. Data storage operations move blocks of data from the buffer to long term data storage: the data reserve. To produce desirable data, the remote monitor must collect data at a certain energy cost; however, if the storage operation is neglected to save energy, and the data buffer overflows, data is lost and the energy to collect it is wasted. The energy management strategy must select an acceptable balance between these two actions based upon the available state information. Should more sophisticated energy management strategies be implemented in the future, a wider range of actions will be available.

The data buffer of the SolarNode protects measurements from device failure, but must be periodically emptied into a larger, more energy intensive, data storage device using the storage action. Regularly emptying the data buffer is desirable because it makes room for more measurements, but doing so when there is very little energy available results in device failure. Ultimately, every data point must be stored, so only the time at which it is stored can be modified; emptying the data buffer during periods of excess environmental energy helps reduce the amount of energy wasted when the energy buffer is full, while providing space in the data buffer for delaying tasks when the energy buffer is low.

6.3 Applied Control Examples

The applied examples of this energy management strategy are taken from [94] and [93], when regarding AWS controllers, and from [72], [95], and [71], when regarding SolarNode controllers. A sample fuzzy control surface, like those shown in this chapter, is built in Appendix A. This includes the steps required to map a point in the input space to the output space, which constructs a continuous control surface.

6.3.1 Energy Management for Arctic Weather Stations

There have been two fuzzy controllers developed for the AWS: one for Resolute, Nunavut, and one for Inuvik, Northwest Territories (NWT). The Inuvik controller was constructed second, so it is more refined than the one from Resolute. The surface for Resolute appears in Figure 6.6.

The control surface created by the fuzzy RBS is later compressed to five discrete measurement actions ranging from 0% to 100% duty cycle. The two inputs act as indices for the lookup table used as a discrete controller, providing a predetermined operational duty cycle to the hardware. The content of the lookup table dictates the system's actions based upon how much energy is presently available and how much energy should be available in the future. It attempts to balance the energy demands of data collection with the energy profile predicted by the energy forecast. Because the controller is generated outside of the simulation and stored in memory, this method requires no complex computations to make decisions at run time. Simplicity is paramount in this approach as other strategies may be incompatible with legacy hardware. The continuous control surface for Inuvik is shown in Figure 6.7a. To demonstrate the discretization process, a three dimensional surface of the lookup table, as stored in the AWS memory, is available in Figure 6.7b. Note that the hardware controlled low-battery shutdown has been built into the Inuvik control surface; it is the area of 0% duty cycle in Figure 6.7a. The hardware shutdown is not controllable, so it must be applied directly to surface, overwriting previous values; contradictory control signals are not permitted in hardware or software.

The Resolute controller assumes that the satellite transmitter must always send a data point each timestep, regardless of state information. Providing ac-

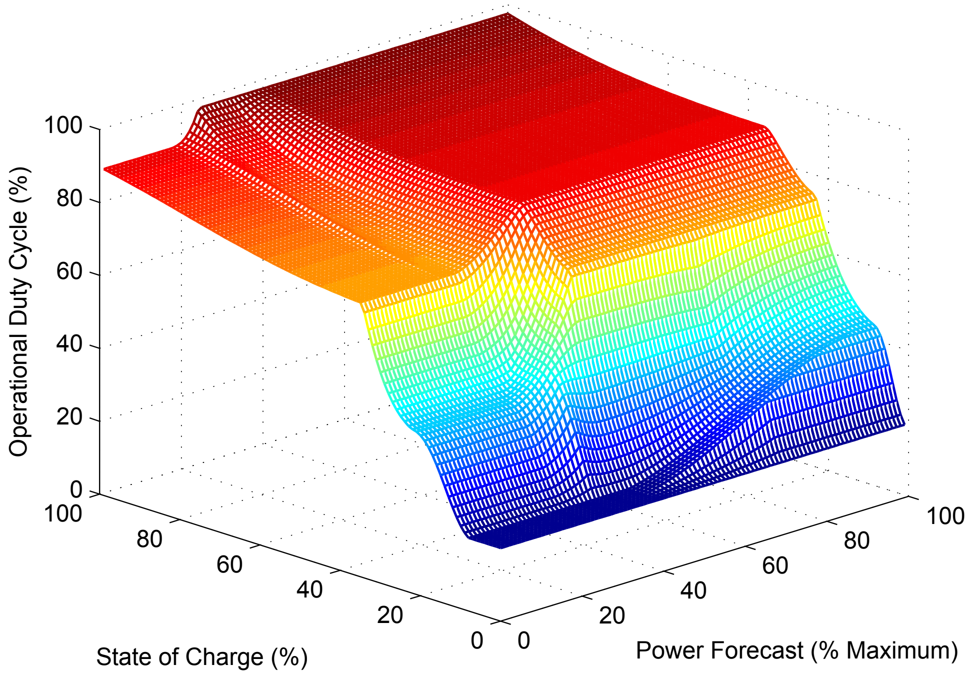
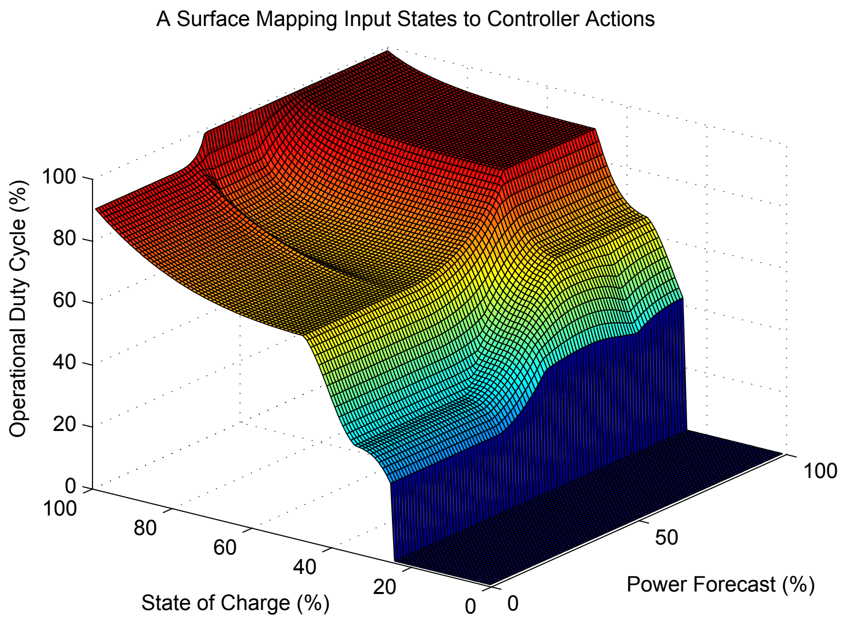
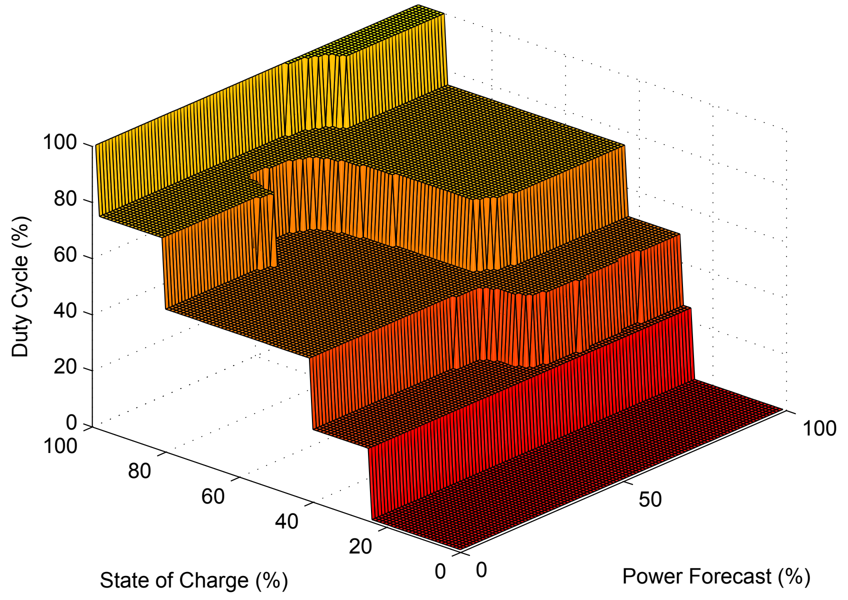


Figure 6.6: This image is taken from Figure 3 of [94] (©2013 IEEE). It is a continuous control surface that maps input state variables to output activity levels. The large, flat top of the surface ensures that many consistent measurements are collected during the spring and summer when the SOC and power forecast are both high. The lower, rectangular portion where the power forecast is low but the SOC is still high, reduces the measurement frequency in the fall in preparation for winter. When the SOC is low, the activity level of the platform must be reduced to protect the battery. The power forecast does not provide energy to the system; it only lets the system know whether or not energy will soon be available. When the SOC recovers during the spring, the power forecast is at its maximum, due to the abundance of summer energy, which ensures that the activity level of the device quickly moves to its maximum at the end of spring. The results of this control system are available in Section 7.4.1.



(a) This image comes from Figure 2 of [93] (©2013 IEEE). This surface is continuous, aside from the hardware shutdown at low SOC. The shape of the control surface is determined by a genetic algorithm.



(b) This surface is the discretized form of Figure 6.7a. Discretization favoured rounding down continuous activity values.

Figure 6.7: This controller, from Inuvik, NWT, shows the discretization process from [93]. There are noticeable similarities between it and the surface from Figure 6.6, as both are for Arctic locations. This controller was evolved in stages to ensured it invested energy conservatively.

cess to the data through satellite transmission is costly to the energy budget of the monitor, but important to prevent data loss. No other data management was attempted within the Resolute AWS simulation. Unlike the Resolute energy management strategy, the Inuvik strategy attempts some data-related energy management by setting a threshold SOC where it switches between the always-transmit and always-store satellite policies. If solar power is abundant, the SOC is high and the always-transmit strategy ensures that energy is consumed rather than wasted to storage overflow. During polar night, the SOC drops and the system selects the always-store strategy to invest energy in data collection rather than spend it on data availability. In Inuvik, the action of the satellite transmitter is determined by a gene from the genetic algorithm; for the simulations shown here, the value of this threshold is 82.5%, as shown in Table 6.1.

Both the Resolute and Inuvik control systems suppress oscillatory actions to prevent wasting energy while in a vulnerable state; this ensures that the simulated monitor only resumes operation after failure when it can provide consistent data capture, rather than repeatedly failing and restarting, which would produce a fragmented timeseries. The Hamming filter used by the AWS, as discussed earlier, is set to 17 units for the Inuvik simulations, as shown in Table 6.1. The effectiveness of this approach is discuss in Section 7.4.1.

6.3.2 Energy Management for the SolarNode

The SolarNode is tested in three locations: Fairview, Chamela, and Inuvik. Fairview and Inuvik are in Canada, while Chamela is in Mexico. The controllers for all three locations were created in the same way and have a similar

Table 6.1: Inuvik simulation parameters and control variables

Battery Capacity	300Ah
Minimum SOC	25%
State Distance Between Action Transitions	17 (Hamming Distance)
Minimum Transmission SOC	82.5% SOC
Length of Power Forecast	5855 Hours

format: a fuzzy RBS creates a mapping from two hardware state variables to two possible hardware actions. This required two control surfaces, one for each action. Fuzzy controls allowed "expert opinion", intuitive knowledge on how the remote monitor should allocate its energy and data resources, to be quickly and easily passed to the controller. The fuzzy RBS covered each of the two state dimensions with five triangular membership functions, sized by the "expert", to fuzzify the incoming state information. Each of the two output action dimensions were also covered by fuzzy sets; they make up the consequent of the RBS by providing fuzzy information on the activity levels of the output. The input and output membership functions were related using twenty-five automatically generated rules. The "expert" determined which membership functions should be used in each fuzzy rule and tuned them using trial and error [65]. The two control surfaces are computed, discretized, and stored as lookup tables in memory before the simulation begins.

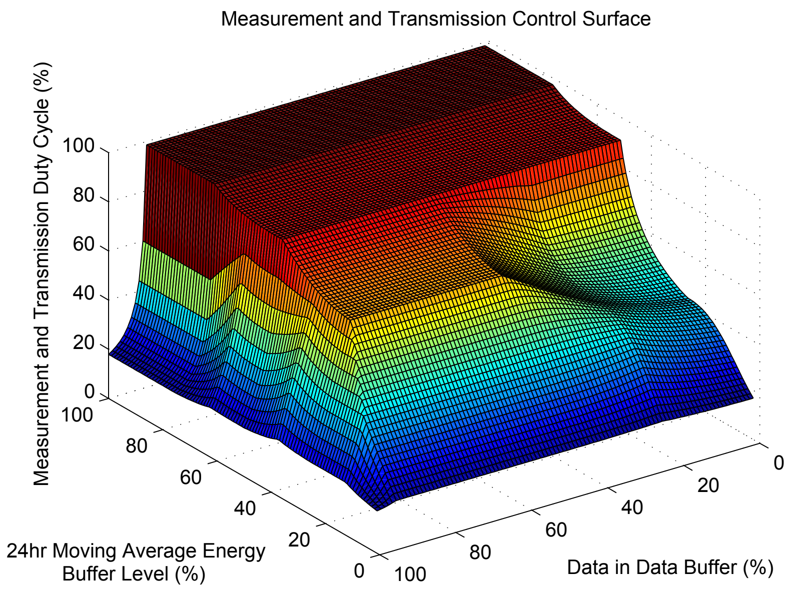
The continuous form of the control surfaces used in each location are provided in Figures 6.8 to 6.10. As mentioned earlier, these used relative percentage values as state inputs and provide a relative percentage activity level as an output. When the continuous controllers are discretized, the outputs take on the actual values the hardware platform will use to execute the energy management strategy. In each location, the storage action operates in the same way, but the measurement and transmission action has a different maximum frequency. All locations have a MIN_{OP} of 1 measurement and transmission per timestep, but the MAX_{OP} is 4 per timestep in Fairview, 7 per timestep in Chamela, and 8 per timestep in Inuvik. These values changed between locations as the project and control technique evolved. In its present state, the simulator assumes that any timestep when atleast one measurement cannot be taken is a device failure. Four measurements and transmissions per timestep were selected for the Fairview controller to facilitate comparison to the naïve management strategies shown in Section 7.2.1 and discussed in Section 6.1. The controllers for the SolarNode have not been optimized, but are still useful for this application; this is validated in Chapter 7.

The relative activity levels of the continuous controllers are replaced with actions immediately before simulation. The surfaces produced for each location by the fuzzy RBS are discretized by grouping certain percentages of activity levels into distinct actions. For example, everywhere the measurement surface for Chamela was above 35%, but below 50%, the lookup table was set to take 1 of the 7 possible measurements and transmissions per timestep. The discretization process is shown in Figure 6.7, for an AWS controller, and Figures 6.10 and 6.11, for the SolarNode's Chamela controller. This way a relative activity level can use state information to *directly* determine what to do. These lookup tables hold all relevant information for the energy management strategy, and provide the control module with a sufficiently complete perspective on the device. The sensor platform only needs to look up one value from each of these two tables to know what it is supposed to do. Using this technique, control is always a simple process for the environmental monitor, while the complexity of determining a good strategy can be abstracted to iterative development using a simulation or passed off to a human.

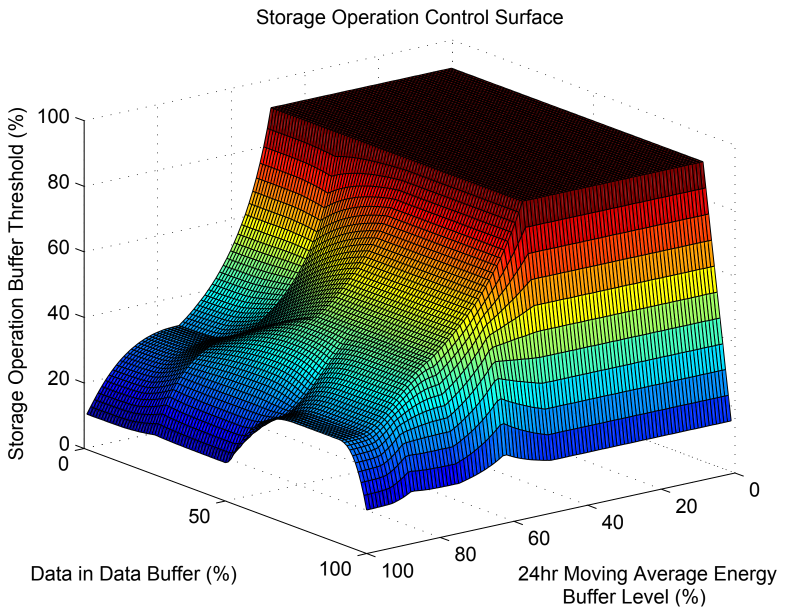
Figure 6.11, shows the "ideal" two actions for the Chamela environment in every possible input state. The state information in Figures 7.6 and 7.7 are used directly by the lookup tables in Figure 6.11. These two tables were originally the two control surfaces shown in Figure 6.10. The results from this simulation are also presented in Figures 7.6 and 7.7. The simulated results of applying all the other controllers shown in this chapter are discuss in Chapter 7 as well.

An energy reserve was added to the SolarNode simulation for the Arctic test location of Inuvik, Canada. As shown in Chapter 7, the other two locations, Fairview and Chamela, do not require significant long term energy reserves to operate in their environment for the current simulation settings. However, in Inuvik, the energy buffer of the device, coupled with energy harvesting, was no longer sufficient to keep the device operational. This was intentional; the objective of the Arctic simulation was to pressure the monitoring device into using its tiered storage capabilities and show the importance of its finite energy reserve. As discuss in Section 4.3 and Section 4.1.2, the energy buffer of the platform is a rechargeable set of super capacitors while the energy reserve is a finite set of batteries. These batteries cannot be recharged by the hardware platform, but as long as their extra energy is available, they provide a great deal of operational security. They can be used to provide performance guarantees that otherwise could not be made. To govern this new capability, a single, new policy has been added to the energy management strategy of Inuvik that is not present for the other locations.

The energy reserve is not observed and is only used when the energy buffer and incoming solar energy are below a useful level that would otherwise result

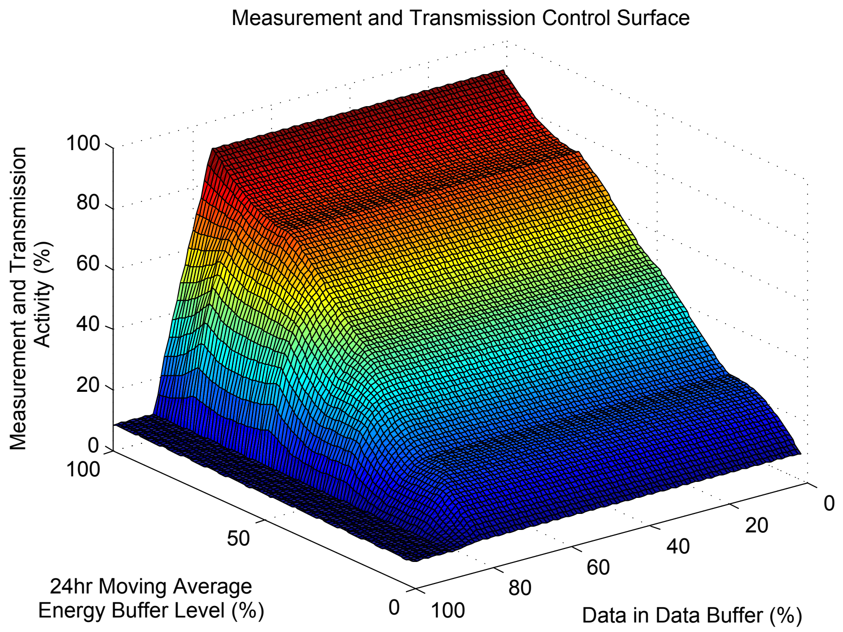


(a) Continuous control surface for measurement and transmission activity level.

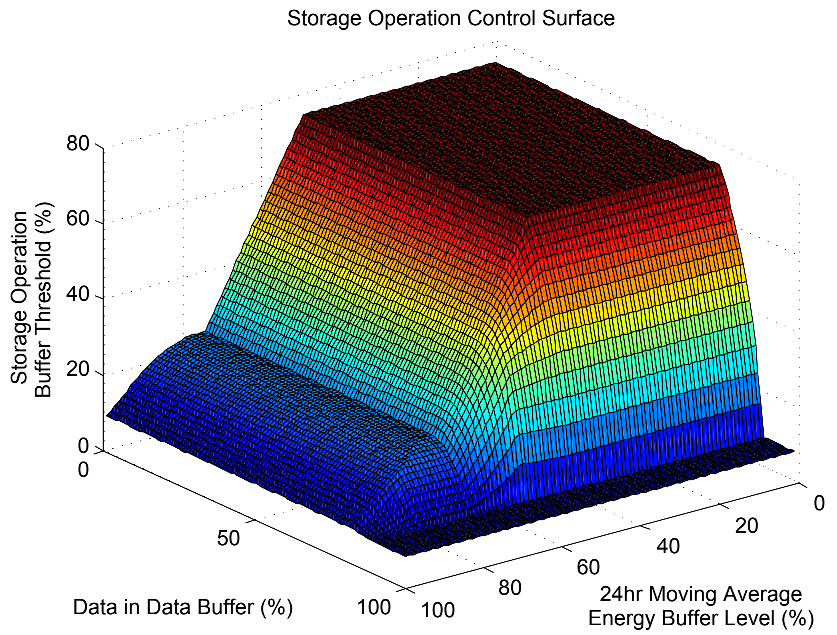


(b) Continuous control surface for storage operation threshold. If the amount of data in the data buffer crosses this threshold, the buffer is emptied.

Figure 6.8: These surfaces show how the controller inputs, the energy and data state of the hardware module, have been mapped to controller outputs: the measurement and transmission activity level and the data storage threshold. Both of these figures must be used simultaneously, but are presented separately for clarity. The results provided in Section 7.2.3 and 7.3.2 for Fairview, Canada, used this controller. This figure is based on Figures 3 and 4 of [72].

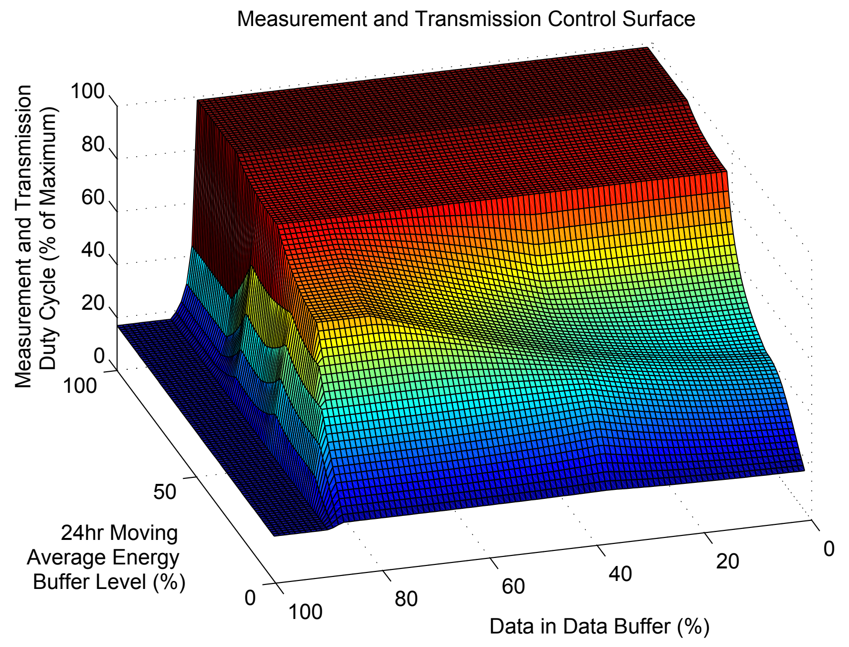


(a) Continuous control surface for measurement and transmission activity level.

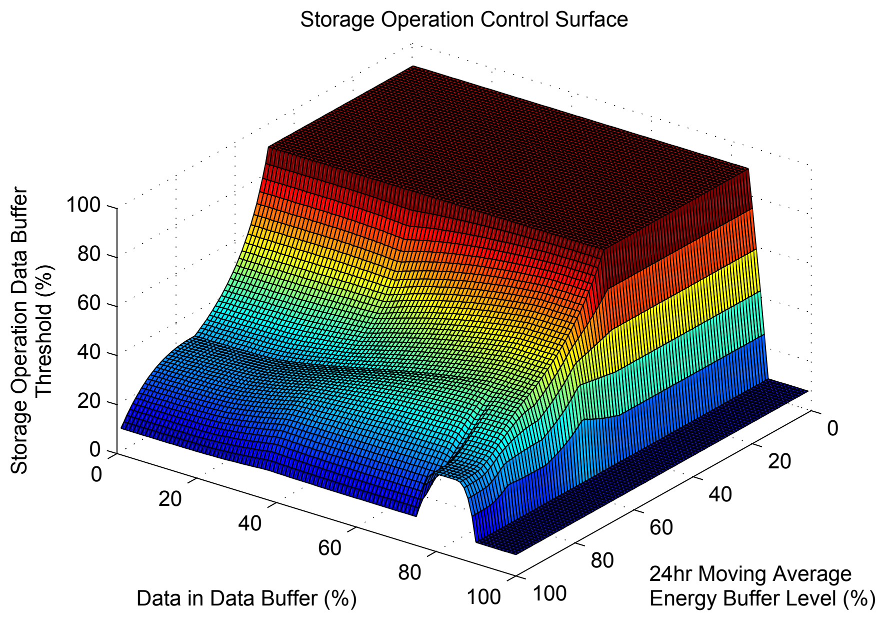


(b) Continuous data buffer threshold for the storage operation.

Figure 6.9: These control surfaces are employed in a similar manner to those in Figure 6.8. The long slope of the measurement surface allows the activity level to be scaled down over the whole range of the energy buffer. The storage surface is set to avoid moving data when energy is scarce, then store it in bulk when energy is is available.

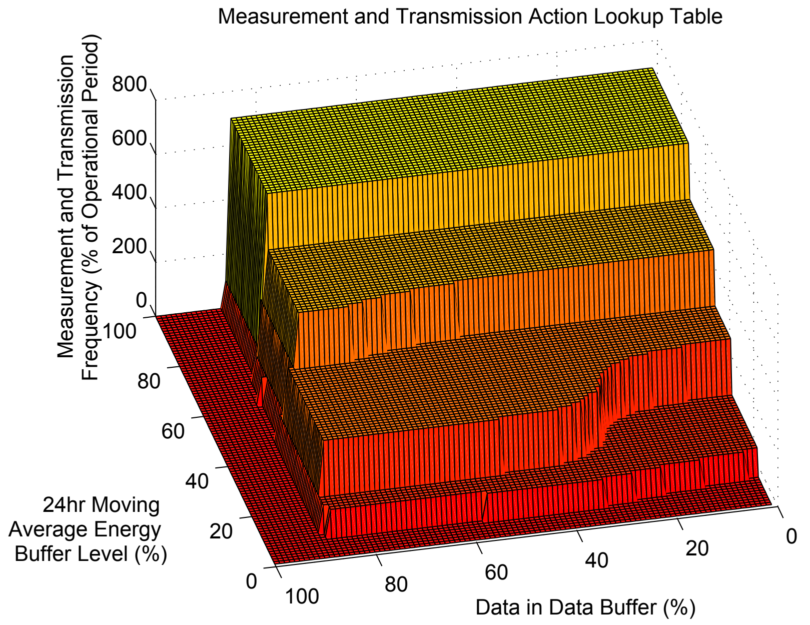


(a) Continuous fuzzy RBS output for measurement and transmission activity level.

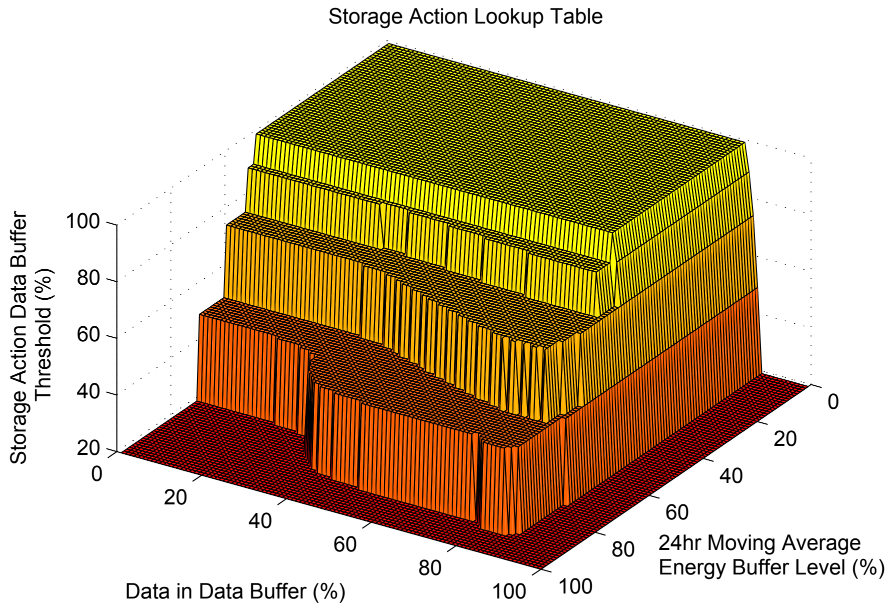


(b) Continuous fuzzy RBS output threshold for storage actions by the hardware module.

Figure 6.10: The control surfaces above are made in a similar manner to those of Figure 6.8, but modified to suit the environment of Chamela, Mexico. This figure is based upon Figures 4 and 5 of [95]. This discrete versions of these surfaces are presented in Figure 6.11.



(a) Discretized measurement and transmission duty cycle plot.



(b) A 3D representation of the storage threshold operation; if the data buffer is more full than this value, it is emptied.

Figure 6.11: This figure is based upon Figures 6 and 7 of [95]. These 3D plots are the discretized forms of the surfaces in Figure 6.10. The input state and output actions during simulation are shown in Figures 7.6 and 7.7.

in device failure. Rather than allowing the device to fail, the energy reserve provides the energy necessary to keep the system operating at MIN_{OP}, which has been defined as one measurement and transmission per timestep. If the energy reserve was included as a state in the control system, like the energy and data buffers, more fuzzy sets and rules would be required to govern the monitor's actions as the input space increased in dimension. This extra control condition in the Arctic energy management strategy is not visible in the figures in this section, but its influence appears in the results of Section 7.3.1.

6.4 Genetic Optimization

The controllers used by the SolarNode simulation were not optimized; instead they were built by "expert opinion", then tuned by trial and error. This works well to quickly implement a controller for energy management, but tuning controllers by hand is neither systematic nor efficient. This describes the controllers from [72], [95], and [71].

The control surfaces used by the energy management strategies of the AWS are optimized, to some degree, by a genetic algorithm. The genetic algorithm was written specifically for the purpose of optimizing AWS control surfaces and was intended to expand on the work of Pimentel in Chapter 6 of [64]. The preconfigured Mathworks MATLAB genetic algorithm was not convenient for optimizing controller and other necessary variables. This motivated the creation of custom software.

6.4.1 Implementation

The genetic algorithm written to optimize the energy management strategy of the AWS is explained in the order it is executed. There are effectively four parts to the algorithm: population generation, population evaluation, population selection, and population recombination.

Population Generation The population of the genetic algorithm represents all individual candidate solutions to the problem; it is stored as a MATLAB array with individuals in each row. The number of individuals is easily changed by adding or removing rows from the array. Each individual has a number of parameters, its genotype, that determine its characteristics, its phenotype, and, eventually, its fitness within the population. Each individual has: a power forecast length, a state filter Hamming distance, a satellite transmitter threshold, and several fuzzy set parameters to determine the control surface. The filter and threshold values have already been discussed; the power forecast length is the number of hours of apriori climatological data used to compute the power forecast state values. A smaller number is more sensitive to immediate weather changes, but provides a less complete look at the energy future of the device.

The individual has 9 genes, 3 sets of 3, for its controller: one set of genes for each input or output dimension. Each gene is the distance between the apexes of the two nearest fuzzy sets. All fuzzy sets are triangular, and the edges of each set's base are attached beneath the apexes of the two adjacent sets. For 5 triangular fuzzy sets, 4 of these numbers are required; the last number is inferred by subtracting the sum of the other three from the total range of the universe of discourse. This ensures that the UOD for each variable is always completely covered, and that the controller uses as few genes as possible. These genes for each controller were selected randomly, so each individual had to be checked for validity before it was accepted into the population to ensure that the sum of all 4 controller genes per dimension was equal to the UOD's range.

Population Evaluation The main step in evaluating the population is running the simulator. The simulator is loaded with environmental data from 1995 to 2005 for one of the three Arctic locations compatible with the AWS. The individual's auxiliary genes, those involved in the operation of the simulator but not part of the control surface, are loaded as well. The power forecast length uses the Typical Meteorological Year data to build the state timeseries for the power forecast. The controller is then constructed and discretized. After the simulator has finished, it outputs sets of diagnostic data that are used to compute the fitness of the individual. Every individual in the population that does not presently have some fitness value undergoes evaluation. The fitness function takes several variables into account, as shown below, but uses a user-determined weighting to compress this information into a single dimensional optimization problem. True multidimensional optimization would require fitness to be used for selection before this weighting takes place.

$$\begin{aligned}\text{Fitness} = \; & \text{(Weighted value of collected data points)} + \\ & \text{(Weighted sum of changes in data consistancy)} + \\ & \text{(Weighted penalty for battery related hardware damage)} + \\ & \text{(Weighted penalty for device failure)}\end{aligned} \tag{6.1}$$

Population Selection The population is sorted by each individual's performance during the evaluation stage. The selection process is based upon roulette selection proportional to rank or to fitness. Rank selection was used to select the next generation, as it was too easy for fit solutions to dominate the population and cause premature convergence. One or two of the most fit candidates were considered elites and exempt from the selection process. This helped ensure that fit solutions, which would otherwise dominate fitness-proportional selection, were carried to the next generation without causing the population to converge. An entirely new population was created by selecting fit solutions from the previous generation of individuals.

Population Recombination The selected individuals were subjected to crossover and recombination to create new individuals and explore the problem space. Rather than manipulate arrays of binary values, the genes of these individuals used whole numbers. Crossover allowed the problem space between two individuals to be explored. The two individuals for crossover were selected randomly from the new population. Multipoint crossover, with some level of blending determined by the user, was used to create a new individual, which then replaced the first randomly selected parent, though elites were exempt from replacement. This was repeated until some user defined percentage, generally between 30% and 50%, of the parent population remained. The individuals were mutated by scaling their genes by a percentage randomly chosen within some user defined bounds. Typically 10% of the genes would undergo mutation, with the a change no greater than 40%. This allows a fair amount of local exploration around each gene.

After this point the population begins a new generation and starts the cycle again. The genetic algorithm executes for 40 to 100 generations, based upon user preference, unless the convergence condition is met. The convergence condition stops the algorithm if all individuals in the population become very similar.

6.4.2 Discussion

In [94], the genetic algorithm loads environmental data from random locations, with random years, without regard for the high variation between these data sets. This led to the flat, angular, top of the controller in Figure 6.6. The results of this controller are presented in Section 7.4.1.

In [93], the genetic algorithm is sent through three different sets of generations with the same population: one set using only Resolute data, one set using only Inuvik data, and the last set using only Whitehorse data. The intention was to push the population into a semi-converged group, which could then be adjusted by changing locations. The population was pushed together in generations using data from Resolute; its harsh climate removed poor solutions. After Resolute, the population spent some generations using Inuvik data. After repeated tests, Inuvik appeared to be the hardest location for the simulator, so it was used to promote individuals that avoided failure by conserving energy in the fall and winter. Finally, Whitehorse data had much more energy than the other two locations; it was used to make sure the individuals that succeeded would not be too conservative during the summer. Up until this point, avoiding failure during the winter was critical to fitness; however, after some generations using Whitehorse data, it became important to collect lots of data during the summer as well. To succeed everywhere an individual needed to conserve energy in the winter, but aggressively collect data during the summer.

The custom genetic algorithm code for this problem allowed the controllers to be optimized, but departed from the reliability of the built-in Mathworks MATLAB algorithm. The other auxiliary energy management variables, besides the controller, were included in the genetic algorithm because of their importance in device performance. They could have been manually tuned, but the optimization process was already automated. By eliminating them from the genetic algorithm, the user would be biasing the algorithm's results by tuning them. At that point, the user might as well just tune the controller. As demonstrated in Chapter 7, a user tuned controller based on trial and error can quickly produce a reasonable energy management strategy. Additionally, these auxiliary variables had a large impact on the data quality, which was the focus of the genetic algorithm's fitness function. For example, the Hamming distance used to suppress oscillations was directly measured by the data consistency value in the fitness function, and would grow very quickly if data collection began to oscillate. By focusing the fitness function on some composite of various data qualities, the user could automatically generate energy management strategies that met their objectives. For example, if the consequences of device failure were increased relative to all other values in the fitness function, the optimization process would push the controllers to be more conservative and ensure that failure did not occur. A more balanced approach was selected for the controller generated for [93], but its goal was to demonstrate the importance of energy management trade-offs in the energy-for-data exchange, rather than strictly prohibit failure. This data centric perspective on fitness follows the recommendations of [94].

The genetic approach is infeasible outside of the computational environment provided by a simulation. The simulation allows many possible controllers to be tested quickly at a low cost, without risking hardware damage [93]. Physical processes are slow and expensive. Capturing feedback from a single AWS would require a whole year of deployment. Should the strategy fail, time and money are wasted. That said, this simulator's settings and results have not been validated with hardware testing or applied in the field. While this is a necessary step in the future, a physical process in the optimization procedure destroys the possibility of using computational intelligence techniques. Validation is an important step, but it should be introduced slowly and used to fix problems within the simulator.

Chapter 7

Simulation Results and Discussion

The simulation results presented in this chapter are taken from various publications over the last two years. In chronological order these publications are: [94], [93], [72], [95], and [71]. The first two publications, [94] and [93], focus on the AWS and use a modified version of the simulator presented in [64], while the final three publications, [72], [95], and [71], use an entirely new simulation based on the SolarNode.

These results can also be separated by year and location. The first study, [94], used data from 1995, in Resolute, Nunavut. Results from 1995, in Inuvik, specifically refer to work from [93] and its associated control strategy, while results using the TMY data in Inuvik are presented in [71]. The discussion of results from 2011, in Fairview, are presented in [72] and compared, in [95], to results from 2008 in Chamela.

7.1 Data Preparation and Processing

All simulation results are based on recorded environmental data; this data grounds the simulation in reality. The results from [94], [93], and [71] are all based upon data from the Canadian Arctic. The results from [72] and [95] are based on data from the Boreal forests of northern Alberta. An additional location in Mexico, using data from a Tropical Dry forest, is also included in [95].

7.1.1 The Canadian Arctic

The Arctic locations use environmental data from the CWEEDS, CWEC, TMY3, and NSRDB databases; these data sets were curated for, and described in, Chapter 5 of [64]. Only Arctic sites within Canada were considered: Resolute in Nunavut, Whitehorse in the Yukon, and Inuvik in the Northwest Territories. They are shown in Figure 7.1. The data set consisted of one *Typical Meteorological Year* (TMY) and ten years of hourly weather data, from July 1, 1995, to June 30, 2005, with the following information: direct normal radiation, diffuse horizontal radiation, air temperature, and wind speed. This information was parsed by year based on date and location, with the TMY data treated as its own special "year". Using simulator conditions determined in [64] to be suitable for each location, all these data sets were converted to MAT-files for the *simulator back-end* (SBE). Each year and location provided two MAT-files: a *climate* MAT-file, which stored the input data to the *simulator front-end* (SFE), and an *output* MAT-file, which stored the power data from each harvesting source for the SBE. The *output* MAT-file also held the ambient temperature for the SBE.

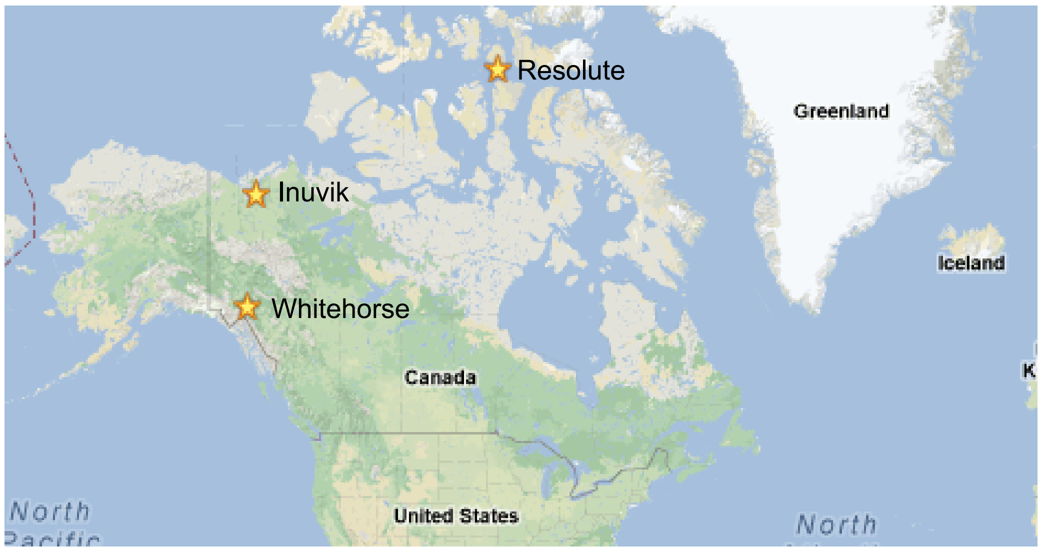


Figure 7.1: A map indicating the three locations for simulation data in the Canadian Arctic. These are, from north to south: Resolute, Inuvik, and Whitehorse. This image was created using Google Maps.

The scripts and simulator modules for processing this data are accessible by Appendix B. The SFE was modified to remove harvester scaling; solar power, for example, now has its output in W/m^2 and must be scaled by the size of the solar panel to be used by the SBE. Figure 7.2 shows how much solar energy the simulator predicts the SolarNode will harvest during an Inuvik deployment using TMY data. The SFE allows energy data to be computed from environmental data while the simulator is offline. Pre-computing the predicted power data allowed a great deal of the previous simulator's complexity to be abstracted away from the SBE. This facilitated the increase in SBE complexity, and allowed it to move to its present, modular, form.

Of these three locations, Inuvik and Resolute had the largest seasonal changes in energy availability due to their distance above the Arctic Circle. While Whitehorse is far north, it is not above the Arctic Circle, so it had far more energy available in the winter than the other two locations. The results

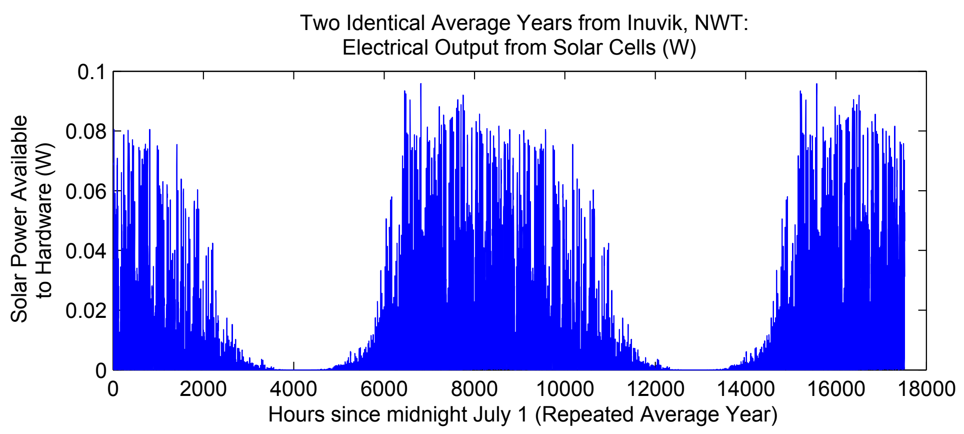


Figure 7.2: The SolarNode's incoming energy harvest during its two year simulated trial in Inuvik using TMY data. Notice the extended period of limited energy harvest during the winter due to Polar Night.

of any simulations from Inuvik or Resolute heavily depend upon harvester sizing and energy storage to succeed.

7.1.2 Boreal Forests of Northern Alberta

The locations from northern Alberta, Canada, were selected because they were close to the Ecosystem Management Emulating Natural Disturbance project (EMEND) site [73]. Two locations, Fairview AGDM and Manning AGDM, were selected from the Agro-Climatic Information Service (ACIS) website because their available data sets had all the required environmental variables while remaining within an acceptable distance of the target [1, 2]. The location of Fairview is shown in Figure 7.3. Both locations have hourly solar irradiance measurements in W/m^2 and are continuous from July 1, 2011, at midnight, until June 31, 2013, at 23:00 [72]. In addition to the solar energy data, the data sets include ambient temperature in degrees Celcius. ACIS Fairview AGDM is located at 56.0815° Latitude, -118.4395° Longitude, and

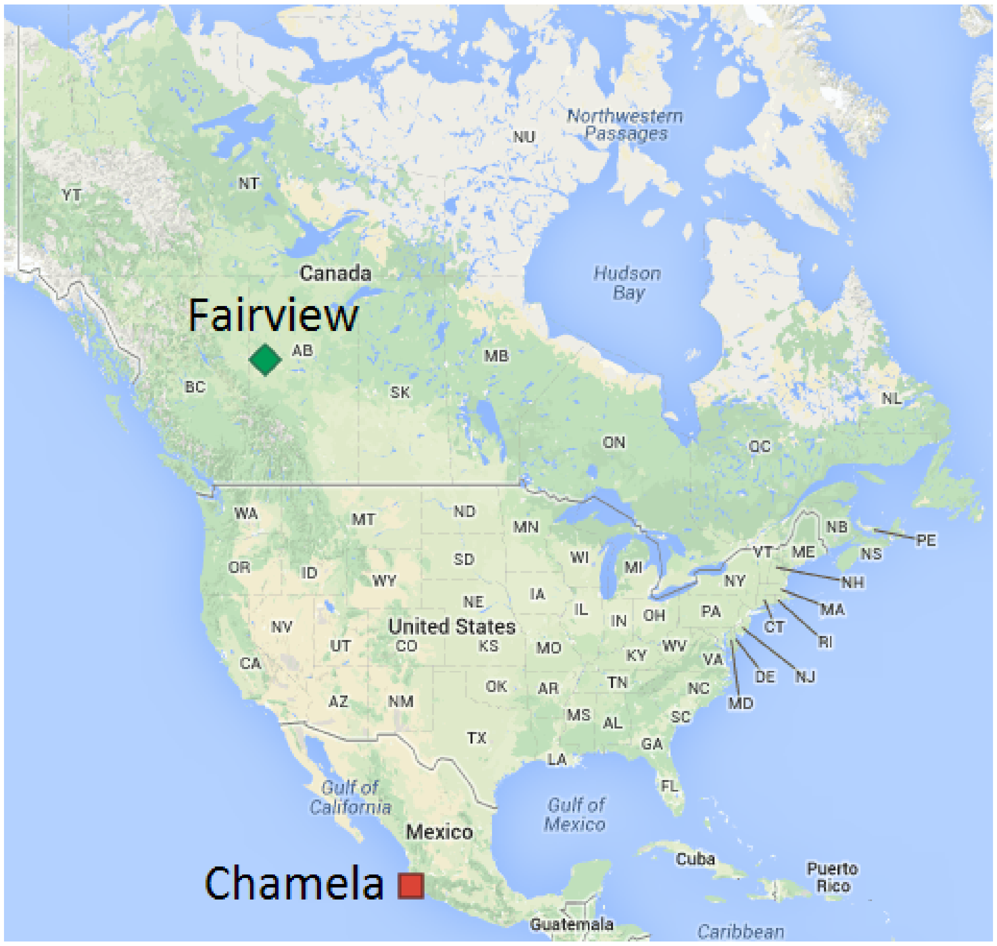


Figure 7.3: The environmental data for the two forested regions came from these two locations. The data set from Fairview, Canada, is marked by a green diamond. The data set from Chamela, Mexico, is marked by a red square. This image was made using Google Maps.

655.00m elevation, while ACIS Manning AGDM is located at 56.9738° Latitude, -117.4508° Longitude, 457.00m elevation. *AGDM* identifies the station as property of Alberta Agriculture and Rural Development [4].

The data sets were downloaded in monthly batches from the ACIS website. The files were moved into two, continuous, hourly data sets based on location. The incoming solar energy data was scaled by 22%: the estimated panel efficiency of the SolarNode platform. This data, along with ambient

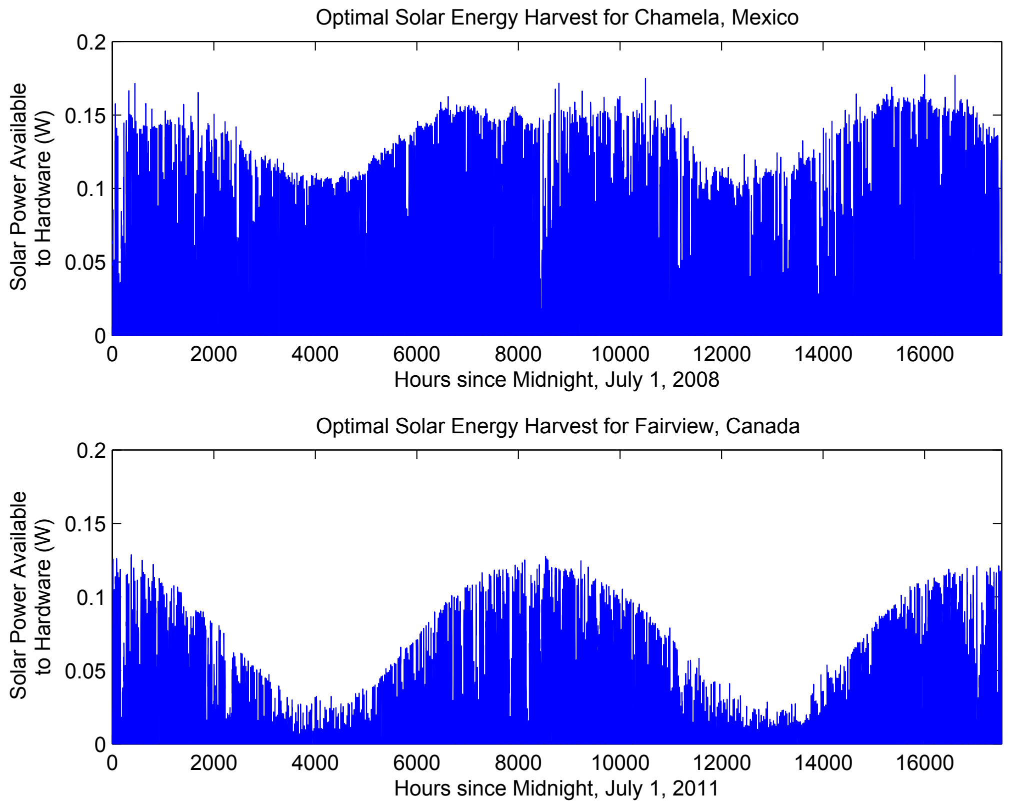


Figure 7.4: The maximum amount of electrical energy provided by the SolarNode's solar cells for Fairview, Canada, and Chamela, Mexico. The figure appears solid due to the diurnal cycle of the data set. The plots in this figure have had their magnitude scaled to down based on solar cell size and efficiency. This image is from Figure 8 of [95] (©2014 IEEE).

temperature information, was then moved to an output MAT-file that could be automatically read by the simulation. ACIS had already filled missing data points, so the files did not require further processing.

7.1.3 Tropical Dry Forests of Mexico

The Chamela location was selected for its proximity to the Tropi-Dry research site in the Chamela-Cuixmala Biosphere Reserve of Jalisco, Mexico [77, 91].

The location is shown in Figure 7.3. The tropical dry forests of Mexico have a unique climate and contrast the other regions in this section. The Tropi-Dry research site is located at 19.4877° Latitude, -104.995° Longitude, and 250.00m elevation, with solar irradiance measurements in W/m^2 from a pyranometer every 30 minutes from March 21, 2008, at 19:30, to April 18, 2013, at midnight [95]. The data sets originate from the Enviro-Net portal [18, 56, 57, 95].

The original file was filled with large and unpredictable gaps of missing data. To provide a data set suitable for comparison with the other simulated locations, the most complete two year segment, July 1, 2008, at midnight, to June 30, 2010, at 23:30, was selected for repair [95]. Missing data points were filled by spreading out the data set based on timestamps, then packing empty segments with invalid, dummy data. The yearly data sets were arranged into a multilayer matrix, then all valid data was averaged to create an estimate of the typical yearly climate. No other climate data was available, so this average was the best estimate for repairing the data set. If no data existed for a point in any year, that point was filled with a local estimate for the typical climate data set. This local estimate took the average of points exactly 2 and 3 weeks on either side of the missing point to fill the hole. These four points were spread over this period to avoid neighbouring missing data, and to ensure that short-term weather events did not interfere with the filling process. After this step the climate estimate was complete.

All invalid data points in the original yearly data sets were filled with their corresponding values from the new average climate data set. Of the now-filled 5 years of data, the most regular 2-year segment of the data set was selected for the simulation. Other years had a tendency to look discontinuous when the average climate data was inserted. The starting point of this process and

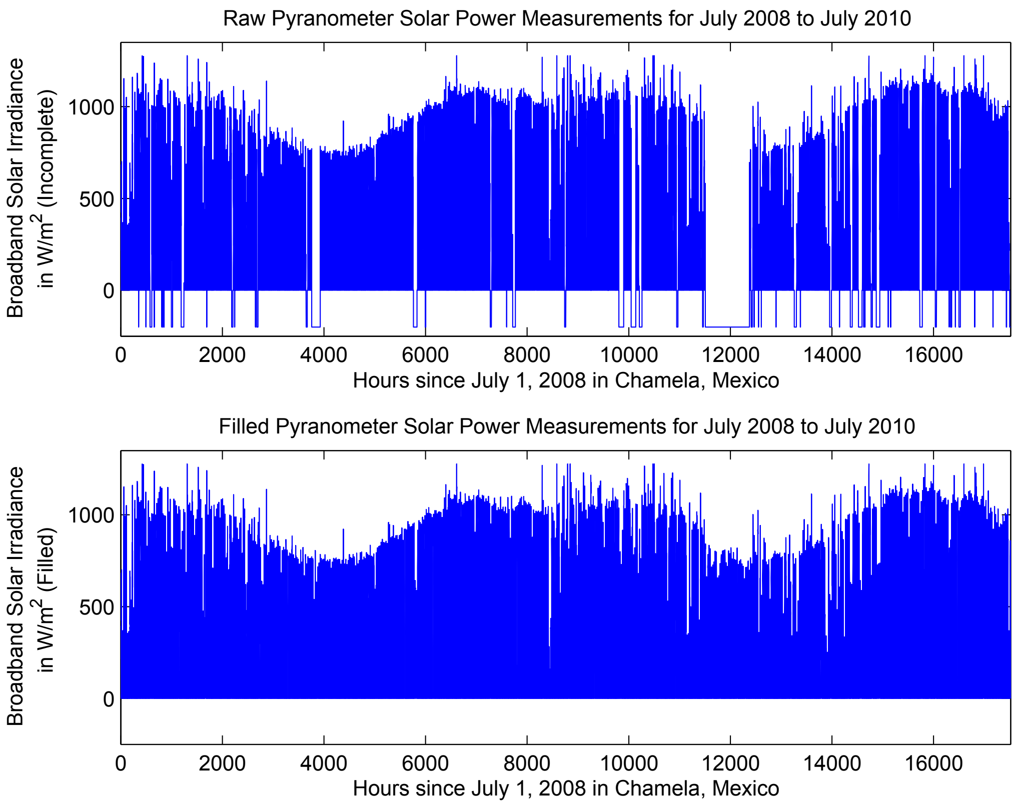


Figure 7.5: The upper plot shows the solar power data from Chamela, Mexico, with missing points represented by invalid negative values. Every point with a negative value was filled with an estimate of the average, yearly solar power. The lower plot shows the final data set used by the simulator [95]. This image is from Figure 2 of [95] (©2014 IEEE).

its final result are presented in Figure 7.5. Combining the yearly data with the average climate estimate was more desirable than using the climate estimate alone, because it forced the simulation to deal with the isolated extremes of individual weather events [95]. Averaged data may let the simulation adapt to the average reduction in solar energy from weather events, rather than a full reduction in solar energy due to a single weather event [95].

The Chamela data set was also scaled by the estimated panel efficiency of the SolarNode. This data set did not include the ambient temperature of the location because the SolarNode is designed not to be temperature sensitive.

7.2 Environmental Adaptation

Two of the three simulation regions, the Arctic and Boreal forest, required the environmental monitoring platform to adapt to the local energy profile. Arctic results are provided by [93], while [72] and [95] provide the results from the Boreal region. Because the Tropical region has a very stable environment, it is discussed in Section 7.2.1 with content from [95].

7.2.1 Consequences of Static Strategies

Naïve energy management strategies are employed by both the AWS and SolarNode simulations to demonstrate the limitations of static energy management and the importance of adaptive device operation. These strategies are outlined in Section 6.1. This section is made up of their results for each of the simulation regions.

The first set of results, from Chamela, Mexico, serve as an example where a static energy management strategy is largely successful. As shown in Figure 7.5 and Figure 7.4, the climate of a tropical dry forest favours solar energy harvesting. Solar energy resources are far higher and more consistent than either the Boreal or Arctic regions. Because the energy resource is consistently plentiful, the survival of the monitor does not depend on adapting to its environment. The environmental energy profile is effectively static, so adapting to it will result in a static energy management strategy anyway; the device does not need to reduce its energy consumption, or make any sort of trade-offs, to

Table 7.1: AWS simulation results for the naïve energy management strategy in Inuvik, 1995 [93]

Strategy:	Naïve
Total Energy Consumed	780Wh
Total Data Points Lost	2748
Percent Data Points Lost	31.4%
Useful Data Points Captured	63.3%
Average Duty Cycle	68.6%

meet its operational requirements. The control required is limited, with the selected output action almost always the same. From Figures 7.7 and 7.6, it is apparent that the predominantly static environment can be controlled by a predominantly static energy management strategy.

The Boreal and Arctic locations do not have a static environmental energy profile. The results of the Arctic and Boreal simulations are provided in Tables 7.1 and 7.2, respectively. These results clearly show that these static strategies require improvement. Attempting to collect data at a consistent rate based on a single part of the year has a disastrous impact on data quality. The naïve strategies for the Arctic and Boreal regions are described in Section 6.1. Later sections show that even a minute increase in complexity, like the addition of a lookup table for control, can result in massive performance improvements.

The results of the AWS under the naïve strategy are shown in Table 7.1 and Figure 7.8. Performance is excellent during the summer because there is more energy than the device is designed to consume; data production is very high. This produces accurate, consistent data, which is constantly made available by satellite for the first $3,236$ hours. Once fall and winter appear, environmental energy becomes scarce, and the device loses the ability to consis-

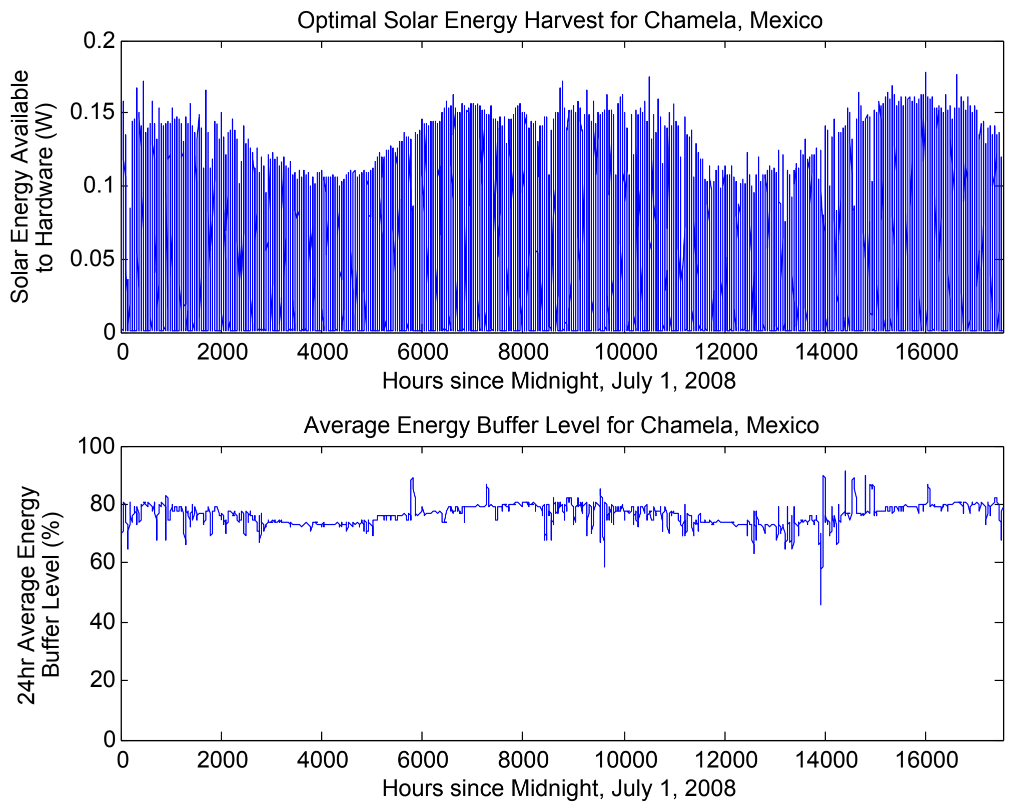


Figure 7.6: This figure shows two signals related to the energy system of the SolarNode for the Chamela, Mexico, simulation data. The upper plot shows the optimal energy harvest for the SolarNode. This assumes that the SolarNode captures precisely 22% of the incoming solar energy, consistent with the efficiency of its solar panels. It shows that there is plenty of energy for the SolarNode throughout the two year period. The upper plot is similar to the upper plot in Figure 7.4 and the lower plot in Figure 7.5. The lower plot shows the average energy buffer level, in percent, over a two year period. The energy buffer of the SolarNode spends the entire two year period in a high, stable state, indicating that the environment is static, and adaptation is unnecessary. This is confirmed by Figure 7.7 [95].

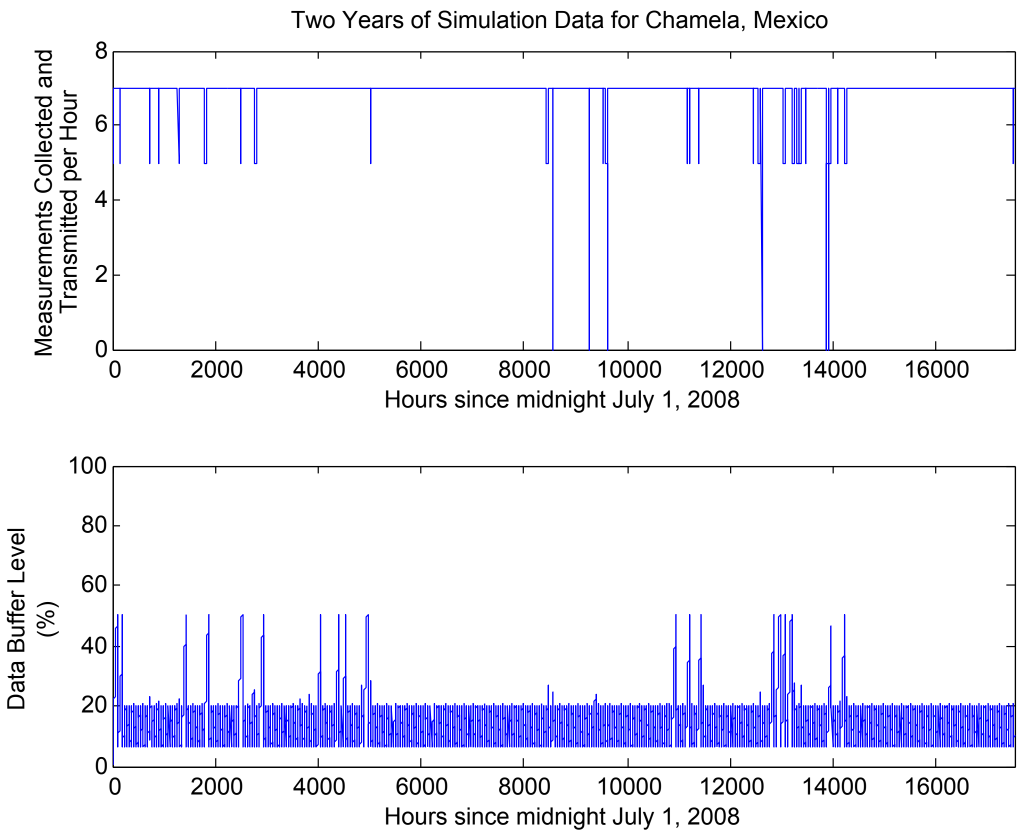


Figure 7.7: These two plots show the actions of the SolarNode while using the Chamela, Mexico, data. The upper plot shows the measurement and transmission frequency selected by the controller. Throughout the two year period the controller predominantly selects 7 operations per hour, with minimal failures and few required reductions in activity. The lower plot shows the data buffer level of the SolarNode throughout the simulation. Storage operations are common, which keep the buffer mostly empty, but there are a few places where storage needed to be delayed. The inclusion of additional redundancy in the energy system, an energy reserve for example, would completely eliminate failures. The monitor succeeds in its environment by maintaining a static level of operation to match the static energy profile of the location [95].

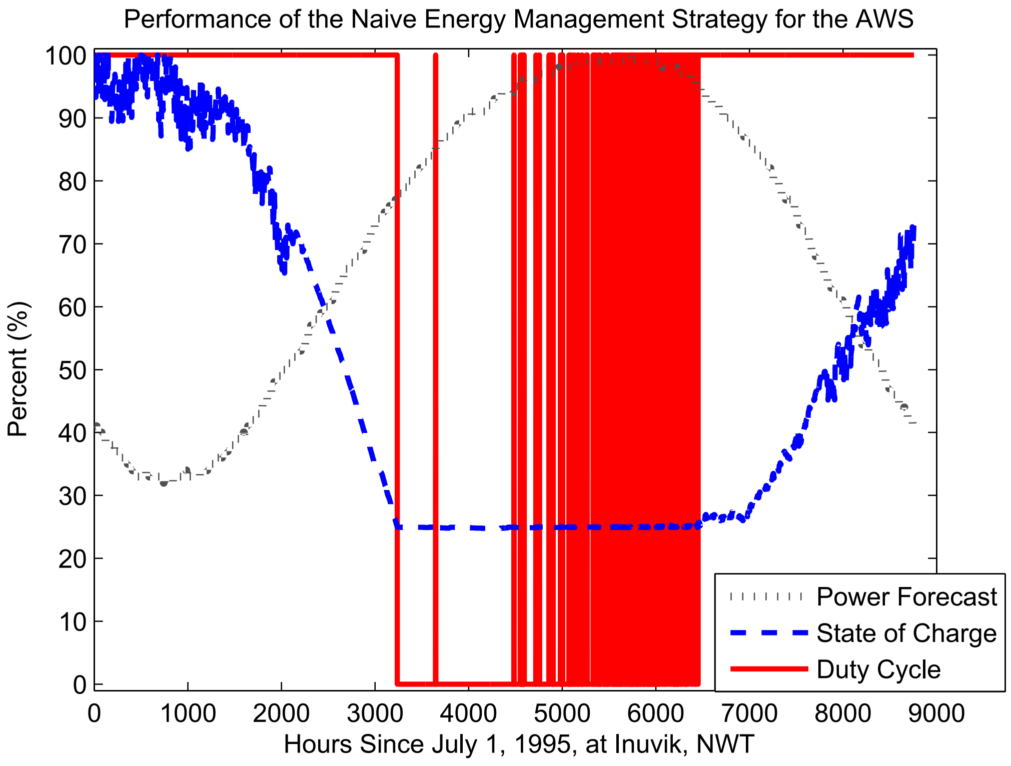


Figure 7.8: This image is based on Figure 1 of [93]. It shows the results of the AWS simulation using data from Inuvik, in 1995, while executing the naïve strategy. During the summer the activity of the device is at its maximum, as shown by the 100% duty cycle; however, during the winter, energy becomes scarce and the AWS fails until solar energy returns. Because the naïve strategy does not observe its state, it cannot detect its repeated failures during the spring. This appears as the large solid section in the duty cycle plot. The AWS is wasting energy in this area; it is spending energy but collecting data at inconsistent intervals.

Table 7.2: SolarNode Boreal simulation results for naïve energy management strategies in Fairview, 2011 [72]

Strategy	Minimum	Maximum
Total Time Period	2 years	2 years
Measurements per Hour	1	4
Maximum Possible Measurements	17520	70080
Collected Measurements	17439	49104
Number of Failures	81	5088
Failure as Percent of Collected	0.4645%	10.36%
Consecutive Failure Score	1349	377850

tently collect data until 6, 453 hours. The randomly interspersed high accuracy data points collected in this time are not useful due to their poor consistency. To become effective again, the system must wait until its energy buffer has been replenished to the point where full system operation may resume without interruption. This naïve strategy fails to collect 31.4% of all data points and spends 36.7% of the simulated year functioning at undesirable levels [93]. The winter energy shortfall of the yearly environmental energy profile exceeds the design capabilities of the device. Because this strategy is unable to make any trade-offs in the energy-for-data exchange, due to its static nature, it is trapped. This is analogous to a material that is too brittle shattering due to physical shock, where a material with more flexibility would survive an impact without damage. Sections 7.2.2 and 7.2.3 discuss the importance of adaptability in control and flexibility in making operational trade-offs.

When the SolarNode simulation uses data from the Boreal region, which is less extreme than the Arctic, the naïve strategies, "Minimum" and "Maximum" from Figure 6.3, still have difficulty. The "Minimum" strategy uses a low measurement frequency, so it collects far less data than it potentially could, given its energy resources. Though the device rarely fails, it produces data of limited utility and wastes energy, which indicates poor energy management. The “Minimum” strategy does not collect data at a sufficient rate to determine high frequency weather events. The “Maximum” strategy is too reckless with its data quality and its operation is plagued by device failures. The *Consecutive Failure Score*, of Appendix C, indicates that the dataset is punctuated by patches and chunks of missing data rather than singular missing points; this makes the data set harder to repair. On top of this, the “Maximum” strategy fails 10.36% of the time. While much lower than the 31.4% of data points lost in the Arctic, is still unacceptable given that nearly all those failures can be eliminated with an adaptive energy management strategy.

7.2.2 Adapting to the Arctic

With Polar Day in the summer and Polar Night in the winter, the Canadian Arctic has extreme seasonal energy changes over the year. Both the AWS and SolarNode simulations from Chapter 4 have been tested on data from this region. Both simulations demonstrate the importance of adapting the device to the variable solar energy resource. The control strategies for the AWS and SolarNode results presented here are available in Sections 6.3.1 and 6.3.2, respectively.

The *Inuvik Fuzzy Strategy* (IFS) for the AWS, of Section 6.3.1, was compared to its corresponding naïve strategy using data from Inuvik, NWT, between July 1, 1995 and June 30, 1996. Inuvik was selected for the particularly harsh conditions of its data set, which deprive the AWS of environmental power for a long continuous part of the year. The importance of energy management becomes apparent under these circumstances. The naïve strategy, described in Section 7.2.1, is commonly used throughout environmental sensing, so this simulation shows why it fails and how to solve its problems. For this simulation, any point where the device does not have enough energy to take a measurement is considered a failure. The *best data quality* for this application shall be when 100% duty cycle is achieved throughout the year. If the device does fail, consecutive failures negatively impact data quality more than individual failures. The IFS allows the environmental monitoring system to dynamically change its operational level between MIN_{OP} and MAX_{OP}. The fuzzy controller is constructed using the technique shown in Section 6.3. The control surface is available in Figure 6.7a. Table 6.1 lists the simulator settings to produce the results shown in Figures 7.9 and 7.10 [93].

Figure 7.9 and Figure 7.10 also show the performance of data storage and the satellite transmission strategy. When solar power is abundant the battery SOC is high and the always-transmit strategy utilizes excess energy effectively. During Polar Night, the SOC drops and the system selects the always-store strategy so energy is invested in data collection rather than spent on data availability [93]. Filling the data buffer, by delaying the transmission task, is like creating a debt of energy, which must then be paid off during the summer when energy is plentiful. The figures show that this is mostly successful. This flexibility allows the controller to make trade-offs in the energy-for-data exchange, ensuring the most important data properties are protected, while those less important are temporarily given up. Energy is invested differently than before, but in a way that reflects the priorities of the monitoring application. A device that depends on solar energy must adapt to its environment to operate in a desirable manner and minimize energy related failures. Strategies that are not suited to the environment produce less desirable data sets than

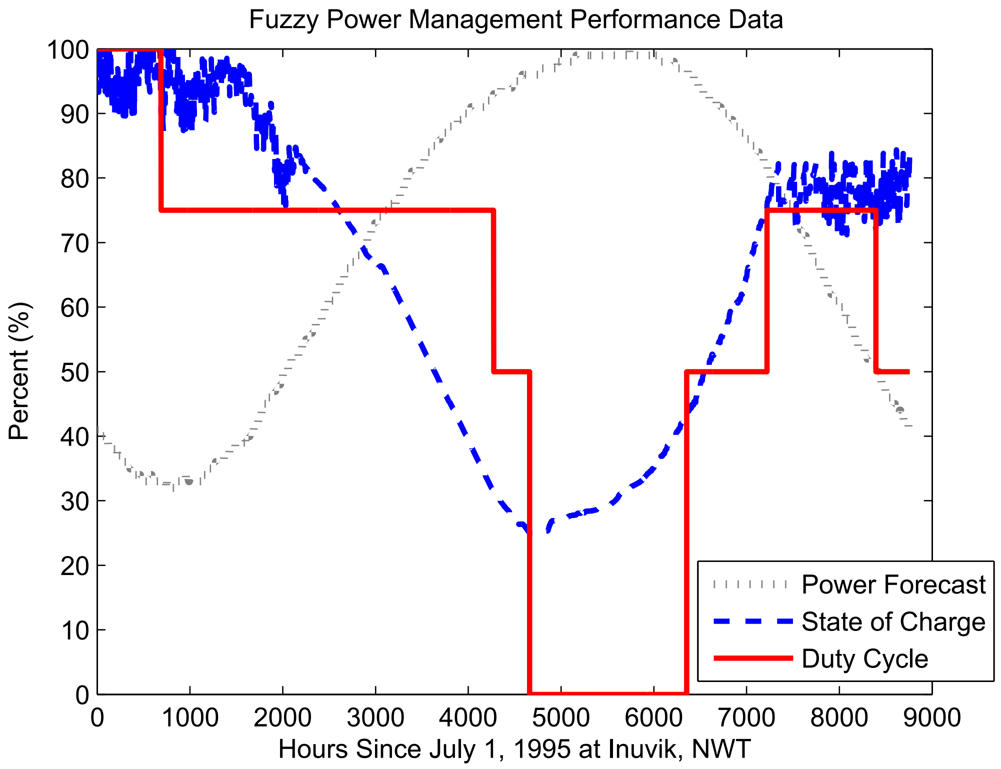


Figure 7.9: AWS simulation results using data from Inuvik, NWT, in 1995, based on Figure 3 of [93]. The blue dashed line and grey dotted line represent the state information for the system. The solid red line is the controller's selected actions. The large smooth trough in the dashed blue line, the battery state of charge, indicates there is a long, continuous period of energy scarcity. Even though the AWS fails for a segment, shown by the Duty Cycle dropping to 0%, it maintains a consistent and decreasing measurement frequency to ration energy in the fall and early winter. When spring comes the controller turns the AWS back on once constant operation can be guaranteed for the rest of the year. By reducing the duty cycle of the device, the fuzzy controller reduces the number of data points it collects, but increases its period of operation. It does this in a stable manner without oscillating or measuring at inconsistent intervals.

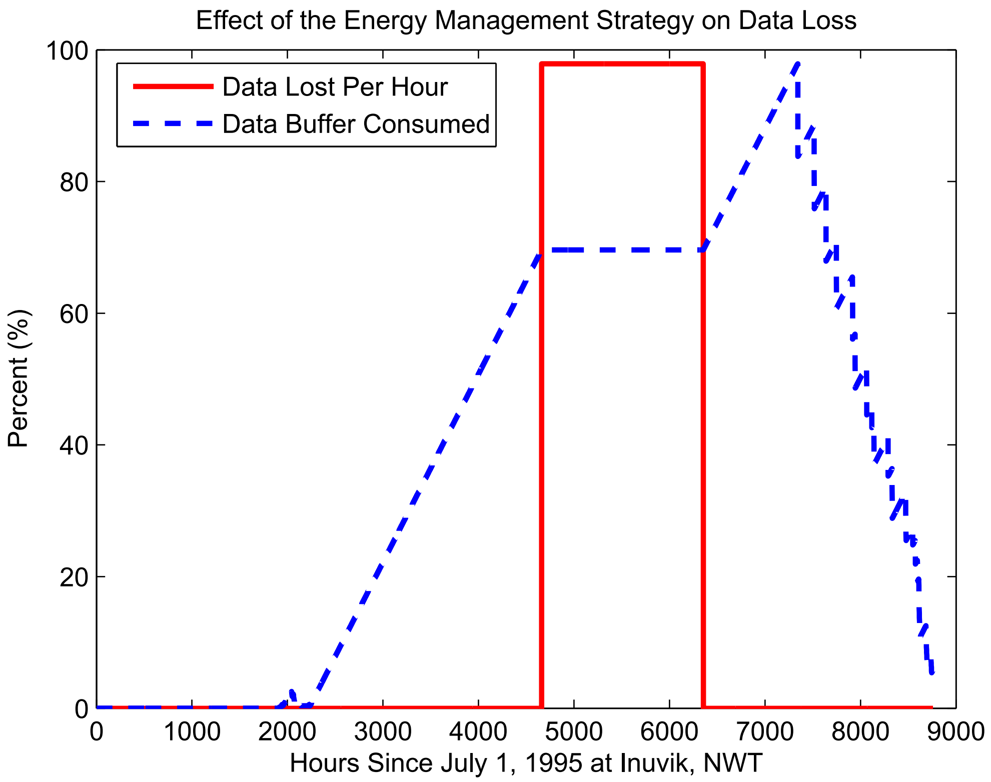


Figure 7.10: This plot is the data-side counterpart of Figure 7.9 and is constructed from the same simulation, based on Figure 4 from [93]. The blue dashed line shows the state of the data buffer. When the bold red line is high, data is lost because the monitor is shut down, so the blue dashed line does not move. The data buffer can hold a maximum of 3500 measurements, so no data was lost to overflow during the winter. Once spring comes, the data buffer is emptied while the battery state of charge has a chance to recuperate. Given additional time, the data buffer could be cleared and the SOC could be high enough for another yearly cycle. By delaying the transmission task, the AWS has reduced energy waste and the number of failures during the winter. Table 7.3 has the performance data for comparing the IFS and naïve energy management strategies.

those that change the operational level of the device to match available energy resources [72].

The results of the naïve strategy, shown in Figure 7.8, can be compared with those of the IFS in Figures 7.9 and 7.10. These figures show that the IFS sacrifices data accuracy by reducing its duty cycle to 75% for the early part of simulation. According to Table 7.3, this allows the IFS system to collect over a thousand more data points than the naïve strategy. In addition to this, the IFS controller does not oscillate when it selects actions, which ensures it collects data at a consistent frequency [93]. Figure 7.10 shows that delaying transmissions and shifting the satellite's energy consumption to the spring and summer helped reduce the amount of time the IFS failed to 19.33% of the year. The naïve strategy performed poorly for 36.7% of the year and its intermittent high duty cycle measurements were not valuable.

The IFS also took better care of the platform's hardware: the battery is only fully discharged once per year, then allowed to recover, before resuming operation in the spring. Stressing the battery for a long time shortens its lifespan [7]. Under the IFS, the final SOC of the system does not return to 100%, which is undesirable, but it does stay consistently high. The results show that the present controller is successfully investing its resources. The remaining summer and fall months in the next year should allow the system to collect the energy it needs to survive the next winter. To improve performance, the AWS must collect and store more energy. Table 7.3 compares some performance metrics of the two strategies [93].

Table 7.3: Performance comparison of energy management strategies for Inuvik, 1995 [93]

Strategy:	Naïve	IFS
Total Energy Consumed	780Wh	724Wh
Total Data Points Lost	2748	1693
Percent Data Points Lost	31.4%	19.3%
Useful Data Points Captured	63.3%	80.7%
Average Duty Cycle	68.6%	57.8%

7.2.3 Adapting to Northern Alberta

The SolarNode simulation in Fairview, Canada, demonstrates the importance of adapting to seasonal energy changes in the Boreal region of Northern Alberta. The two naïve SolarNode strategies of Section 7.2.1, "Minimum" and "Maximum", are intended for comparison to a third, adaptive energy management strategy, described in Section 6.3.2, that can respond to the observed state of its hardware. Because the simulation uses data from Fairview, Canada, and it adapts to its environment using the controller from Figure 6.8, it will be referred to as the *Fairview Adaptive Strategy* (FAS) [72]. Two years of data from the Fairview location were fed into the simulator to compare the "Minimum" and "Maximum" naïve strategies to the FAS. For the FAS, the controller must balance operation between the SolarNode's MIN_{OP} and MAX_{OP} settings; simply reducing operation to MIN_{OP} is not an appropriate response because it unnecessarily reduces the monitor's performance to an undesirable level. In this application, collecting the most data points over a two year period with as few device failures as possible produces the *best quality data*. By adapting to the environmental energy profile the controller can capture almost as much data as the static MAX_{OP} strategy while reducing the total number of failures to almost that of the MIN_{OP} strategy. This is confirmed by the

Table 7.4: Simulation results for three different energy management strategies

Property	Minimum	FAS	Maximum
Total Time Period	2 years	2 years	2 years
Measurements per Hour	1	1 to 4	4
Maximum Possible Measurements	17520	70080	70080
Collected Measurements	17439	61731	49104
Number of Failures	81	91	5088
Failure as Percent of Collected	0.4645%	0.1474%	10.36%
Consecutive Failure Score	1349	1297	377850

simulation results presented in Table 7.4.

The simulator output for the FAS is shown in Figure 7.11 and Figure 7.12. The first figure shows the two inputs to the control surfaces: the data buffer and the 24 hour moving average of the energy buffer. The effects of the two possible actions are visible on the data buffer plot. During the summer, the data buffer is quickly filled by the maximum number of measurement actions and emptied by frequent store actions, while in the winter, the data buffer is slowly filled by the reduced measurement rate and emptied less often. Delaying the storage action allows the device to wait for more environmental energy and avoid consecutive failures. The energy buffer plot shows that energy intensive actions are reduced while environmental energy is low [72].

Figure 7.12 helps clarify what is observed in Figure 7.11. The number of measurements taken per hour is at a maximum for most of the year, and only drops for a brief period during the winter. Any points on this plot where the number of measurements drops to zero are considered failures. The lower plot in Figure 7.12 grows quickly during the summer, when measurements and stores are common, but flattens out during the winter, when less energy is available. The most interesting periods on all four plots are around 4000

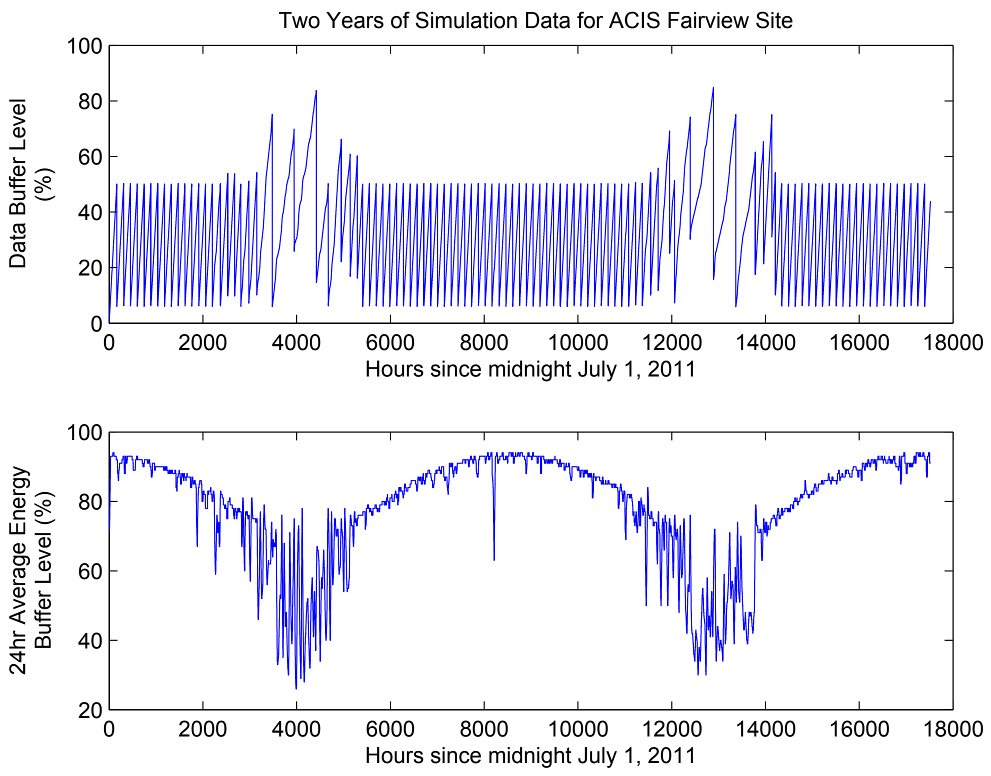


Figure 7.11: This image is taken from Figure 5 of [72] (©2014 IEEE). The upper and lower plot make up the data buffer and energy buffer state variables, respectively, for the control surfaces in Figure 6.8. The data buffer level in the upper plot is filled quickly, and emptied at half capacity, for most of the year, except winter, where it is filled slowly and permitted to pass half capacity to save energy. The sharp vertical drops in the data buffer state are the storage actions. In the lower plot the 24 hour average energy buffer level forms two arches with their maximums at the peak of summer and their minimums in the middle of winter. The energy buffer saturates in the summer, hence the high, consistent average, but occasionally fails in the winter when it is vulnerable. The simulation uses two complete years of data to show that the controller can provide this level of performance over a long time period.

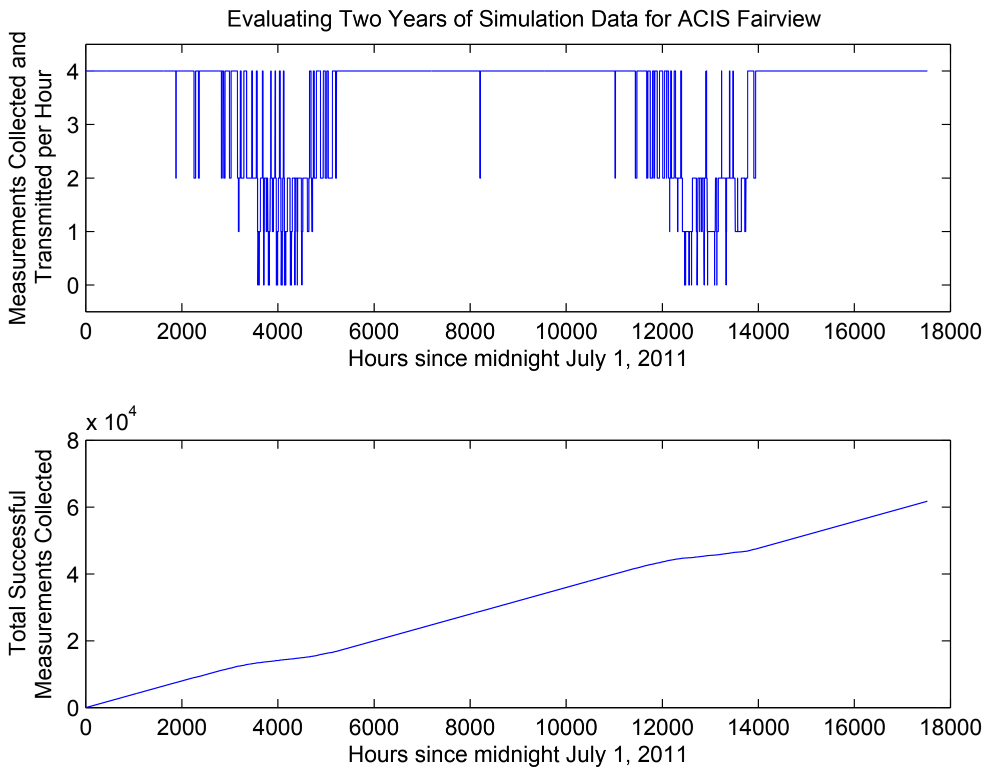


Figure 7.12: This image is taken from Figure 6 of [72] (©2014 IEEE). It shows complementary data to the plots in Figure 7.11. The top plot shows the measurement and transmission activity of the platform. It has a maximum frequency of 4 measurements and transmissions per hour, which was determined to be the most desirable amount. The device could capture data more often at all times other than winter, but this rate was set as MAX_{OP} for comparison to the region's naïve strategies. During the winter, when the average energy buffer level is low and unstable, the activity level of the device drops to avoid failure. This reduces the rate at which the data reserve is filled. The data reserve, shown in the lower plot, is massive, and will last a long time before reaching capacity; it is filled in blocks during storage actions.

and 13000 hours, corresponding to the winter periods of the two consecutive years [72].

By changing between the “Maximum” strategy, when energy is plentiful, and the “Minimum” strategy, when energy is scare, the FAS eliminates the majority of device failures associated with excessive energy consumption in the winter, while collecting the majority of data points during the summer. This allows the FAS to outperform the other two static strategies. While the “Minimum” strategy technically fails less often than the FAS, Table 7.4 shows that the number of failures compared to the total amount of data captured by the device heavily favours the FAS. Both the *Failure as Percent of Collected* and *Consecutive Failure Score* show how seriously exceeding environmental energy availability harms data collection, particularly for the “Maximum” strategy, which ends up collecting less data than the FAS [72]. The *Consecutive Failure Score* is a simple operation to determine whether device failures are clustered together or spread throughout the data set. It is described in Appendix C.

This simulation demonstrates that an adaptive energy management strategy is more desirable for the environmental monitoring platform than a static energy management strategy. The strategy can effectively invest the energy resources available in the environment because of the flexibility provided by the fuzzy controller. Should this controller be tuned further, the performance will continue to increase, but at a more modest rate. For the rest of the chapter, the importance of energy management will be taken for granted.

7.3 Storage and Sizing

Storage and sizing are important properties for energy and data management. Arctic locations are largely concerned with storage because environmental energy drastically changes with the season. This is demonstrated for both the AWS, in [93], and the SolarNode, in [71], using simulations with data from Inuvik in 1995 and from Inuvik's TMY, respectively. The Boreal and Tropial regions are exclusively explored with the SolarNode in [72] and [95]. They mostly focus on sizing, as those environments are more hospitable to monitoring hardware and energy harvesting.

7.3.1 Arctic Considerations

Arctic environmental monitoring systems depend upon energy storage to survive the energy-scare winter months. As environmental energy decreases in consistency, energy storage increases in importance. The AWS results presented in [93] demonstrate the importance of adequate energy storage. This is related to harvester sizing as well. Another Arctic simulation, detailed in [71], demonstrated that the additional storage provided by an energy reserve allows the SolarNode to survive Polar Night without any failures; adding a layer of redundant energy storage to the system improved performance.

To prevent failure, storage for the AWS is critical. It provides the flexibility the energy management strategy requires to make trade-offs in the energy-for-data exchange. First of all, sufficient energy storage capacity is necessary to transfer excess solar energy from Polar Day to Polar Night, for consumption when energy harvesting is not an option. Without diversifying energy harvesting technologies, storage is the only option for the platform. Secondly, a sizeable energy buffer protects against changing environmental conditions. When the energy storage capacity of the device is much higher than periodic energy changes in the environment, energy storage effectively becomes a low pass filter for the controller. This is visible in the battery SOC signal in Figures 7.9 and 7.10; the diurnal solar energy cycle has a minuscule effect on the energy buffer and the platform's actions, unless it is near a discontinuity in the control surface.

The harvester and storage components must be sized at the same time because they are interdependent parts of the energy system. The harvester module must be large enough that the average power harvested from the environment is greater than the average power consumed by the platform on a yearly basis. The solar panel of device must produce enough surplus energy in the summer for device to operate during the winter; energy storage must have the capacity to transfer this energy surplus between these two seasons.

It may be difficult to see problems with harvester sizing if energy storage is very large, as shown in Figure 7.9. The period of failure in the figure indicates that energy storage was not adequately sized to meet the energy demands of the system at its selected operational level. Whether or not that operational level is appropriate is a different design question. Looking at the battery SOC in the last quarter of the timeseries, during the spring, it appears that the harvesting module is also slightly undersized. It struggles to refill the battery during the energy intensive process of emptying the data buffer. While the solar panel would have a couple months to recharge the battery if the simulation had continued, any energy shortfall would harm the monitor's performance in the next year.

Figures 7.9 and 7.10 show the importance of data storage to the platform's performance. The data storage capabilities of the AWS, though primitive in the simulator, allow its energy management strategy to learn to prioritize different data properties. Data availability, via satellite transmission, is not a factor in the fitness function of the genetic optimization; this identifies data availability as a very low priority. Device failures and controller oscillations are important to fitness, so energy management delays transmission when there is an energy shortage to focus on collecting more data and avoiding these two problems. This data is transmitted later when there is an energy surplus to prevent wasting energy. Appropriately sized data storage allows failure prevention to be spread between the energy and data systems, rather that concentrating all the responsibility for good performance on the energy system; without appropriate data storage capacity, data related tasks cannot not be delayed.

Figure 7.10 shows that if the AWS had not shut down briefly during the winter, its data buffer would have overflowed. The data buffer's capacity is too small for this application, but the inadequacy of energy buffer prevents it from overflowing. If the capacity of the energy buffer and the area of the energy harvester were increased exactly enough to prevent energy related failure in the winter, the performance of the system would not improve because stored data would be lost when the data buffer overflowed. The interrelated storage systems of the device must be sized together. This is not a simple problem, and has not been fully addressed by this simulation.

Device sizing is a complex problem within the simulator for two reasons. First of all, the simulation is not constrained by the cost and physical dimensions of energy storage, so it is easy to eliminate failure by increasing storage capacity. However, this is just changing numbers in a program and does not actually consider the limitations of the project. Second of all, as systems become more integrated, like the SolarNode, it may not be possible to arbitrarily resize components. The AWS can change its battery bank and solar panel, but its logger module has a fixed internal data storage capacity. The SolarNode cannot change the size of either of its data or energy buffers, but has more freedom when selecting energy and data reserves. Adjusting simulation parameters without consideration for practical limitations does not help design the remote monitoring system.

The SolarNode has a very different energy and data storage topology than the AWS. While the AWS has large buffers for energy and data relative to its daily needs, the SolarNode has two layers of storage, with the buffers relatively the same size as its daily energy and data needs. Because of their size, these buffers do not have the same filtering properties as those of the AWS, and risk becoming saturated if proper control is not exerted. The SolarNode is more dependent upon control than the AWS; without a controller, the data buffer would quickly overflow and the energy buffer might fail at night due to bad weather during the day. However, both the buffers of the SolarNode can be supported by a layer of large, redundant, reserve capacity to improve the reliability of the platform.

To showcase the importance of storage to the SolarNode, the simulation uses the Inuvik TMY data to subject the platform to Polar Night, which forces it to rely on its energy reserve. The SolarNode was configured for operation in the Arctic and equipped with a large, disposable energy reserve to supplement its small energy buffer. The SolarNode operates under the most conservative consumption policy when using the reserve; only one measurement is collected

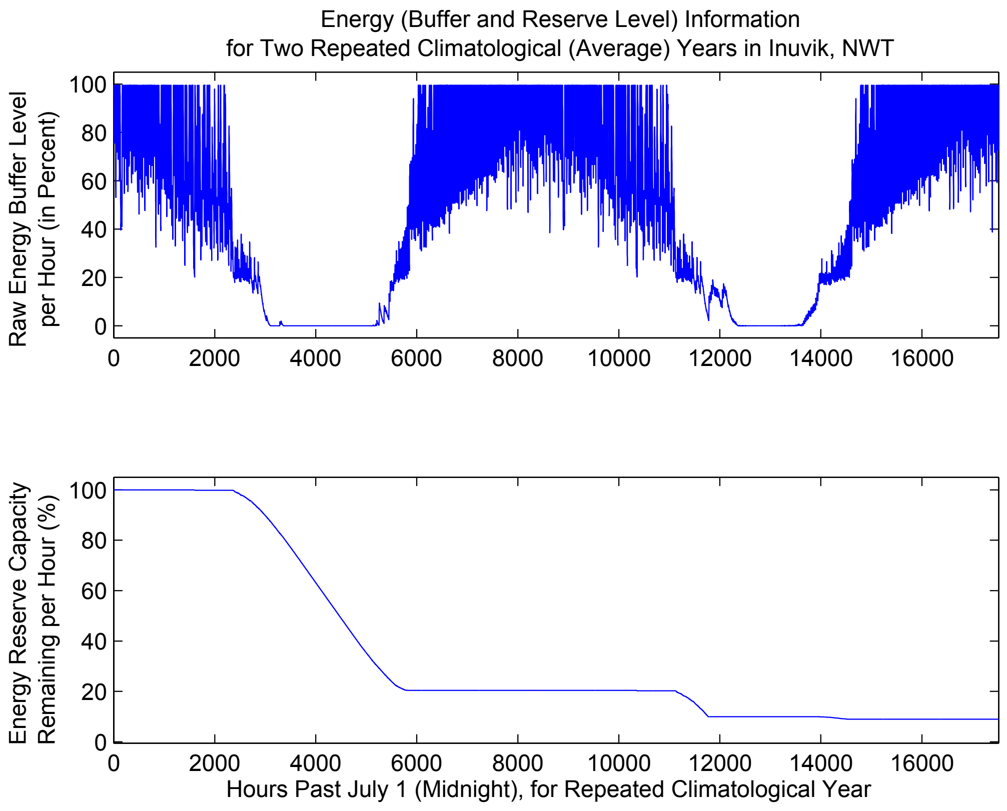


Figure 7.13: This image is based on Figure 3 of [71]. The two plots show the state of the SolarNode's energy system over the two year simulation period. The upper plot is the unprocessed energy buffer level for the SolarNode; it is not averaged and is not used as a state variable. The plot is dense where the buffer level changes with the diurnal cycle. This is most apparent in spring and fall. The flat top of the plot during the summer shows that the energy buffer saturates and excess energy is wasted. This shows there is more than enough energy for the platform during the summer. Polar Night appears as the low, flat segment of the year when there is no solar energy is available for harvest. The lower plot is the SolarNode's disposable energy reserve level. It is appropriately sized for one year of operation, but the simulation runs for two years. Exhausting the energy reserve during the second year results in massive winter failure. This shows how important appropriately sized buffers and reserves are to performance, as confirmed by Table 7.5.

per hour, MIN_{OP}, when using battery power to collect the longest, uninterrupted data set possible. Figure 7.13 shows the state of the two energy storage layers of the SolarNode during a two year simulation using Inuvik TMY data. The winter period, the region of limited energy harvest in Figure 7.2, completely consumes the energy stored in the buffer and forces the platform to depend upon the reserve until spring, as shown in Figures 7.13 and 7.14.

The winter periods of the two sequential years are very different. During the first year, the energy reserve eliminates all failures and guarantees a complete year of operation. The activity of the device changes from MAX_{OP}, 8 measurements and transmissions per hour, during Polar Day, to MIN_{OP}, 1 measurement and transmission per hour, while operating on reserve energy during Polar Night. Winter energy investment leaves the reserve depleted, so it fails at the beginning of the subsequent winter. The small amount of energy remaining in the reserve is quickly consumed, and then the SolarNode must wait until spring for its supercapacitors to be recharged. This leaves a large, undesirable gap in data set. The end of Polar Night and the arrival of spring allow the SolarNode to recharge its energy buffer, so the energy reserve is no longer used or required. According to this simulation, the SolarNode should be able to guarantee continuous data sets if it had a yearly maintenance cycle. Table 7.5 compares the results of the two simulation years to show the importance of the energy reserve.

The two years of data in Table 7.5 break down the performance of the monitoring device by year. The first thing to notice is the similarity in the number of measurements between the years. The year without the energy reserve has only 7.95% less measurements, but does not cover of 28.78% of the year. The small increase in the number of measurements due to the energy

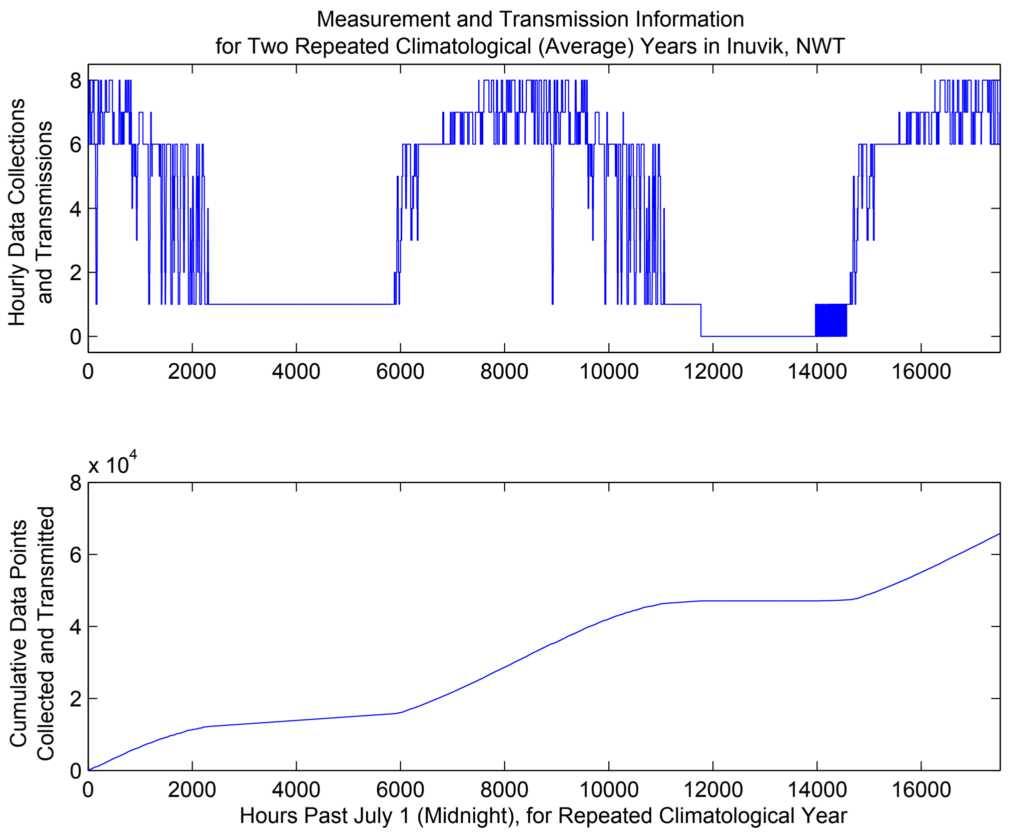


Figure 7.14: This image is based on Figure 4 of [71]. This figure shows the movement of data through the SolarNode during the two year simulation. The upper plot shows data entering the system through measurement. During the first year, Polar Night reduces the system to MIN_{OP}, one measurement per hour, while operating on reserve energy. During the second year, the reserve is depleted and the device fails for the remainder of the winter. Polar Day allows the SolarNode to collect up to 8 measurements per hour, MAX_{OP}, without failing throughout the summer. The lower plot shows the level of the data reserve; it acts like an integral of the upper plot because all measurements are eventually stored there. The slope of the plot is high in the summer, but low in the winter, because of the changing measurement rate. Its slope drops to zero during the second winter when the device fails for an extended period.

Table 7.5: SolarNode energy reserve simulation results for Inuvik [71]

Reserve State by Year Plot Start Index	**Available** Hour 1	**Depleted** Hour 8761
Hours Failed per Year	No Failures	2519
Measurements per Year	34140	31666
Failure as % of Measurements	0 %	7.95 %
Failure as % of Hours	0 %	28.78 %
Total Energy Collected	95.487 Wh	95.487 Wh
Reserve Energy Consumed	6.051 Wh	0.869 Wh
Energy-for-Data Exchange Rate	336.228 Measures per Wh	328.634 Measures per Wh

reserve provides a massive increase in data completeness and coverage. Given the small size of the energy buffer relative to the energy reserve, reducing the performance of the SolarNode in the fall to ration energy is not an effective strategy because the buffer does not have the capacity to store energy for long term operation. Even operating at MIN_{OP} during the fall to conserve the buffer's energy supply would prove insufficient.

The size of the reserve itself is very important. It is appropriately sized for a single winter of operation, but its value would drop sharply if it was any smaller. Remember that the size of the reserve is dependent upon the platforms rate of consumption; this reserve is appropriately sized for only the lowest level of activity. Any energy consumption higher than MIN_{OP} would quickly result in device failure. The energy reserve must be sized according to the needs of the application. Because the energy reserve is not rechargeable, excess energy harvested during the summer is wasted. This waste is apparent in Figure 7.13, where the raw energy buffer level reaches its maximum for most of the summer, making the top of the plot look flat. This is also evident in the 24 hour moving average energy buffer state information; this average tries to filter out environmental noise for the controller, and replaces the flattened peaks of the raw energy buffer level with a long curve over the summer. Figure 7.15 shows the daily average energy buffer level jumps to between 80% and 100% during the summer: it spends most of its time saturated and wasting energy. If the reserve was rechargeable, the energy reserve at its current size could guarantee year-round performance by collecting wasted energy in the summer for use in the winter, just like the AWS.

The rate of energy-for-data exchange between the two years is affected by the presence of the energy reserve. Firstly, when the reserve is present it helps collect extra data points, which improves the exchange rate. Secondly, a great deal of energy is wasted during the summer in both years, which hurts the rate of exchange, but in fall and spring, energy is also wasted by the supercapacitor energy buffer. The energy buffer cannot be used below a certain percent capacity, due to hardware constraints, but this does not prevent the supercapacitors from leaking this charge. Excess solar energy trapped in the super capacitors while the device is not functioning properly is lost when the reserve is not around to keep the device operational; the supercapacitors still leak this energy, but at least a measurement is taken if the reserve is present. This happens when the SolarNode repeatedly wakes up, then fails, during the second year of Figure 7.14. It appears as the dark, solid square in the upper plot.

These figures show the differences in value between the yearly data sets. Given that the energy buffer saturates during the summer of both years, data collection could be increased during that time. Their current summer rate is impressive, but it does not compensate for poor performance in the winter. Collecting data 8 times an hour is not that much more impressive than col-

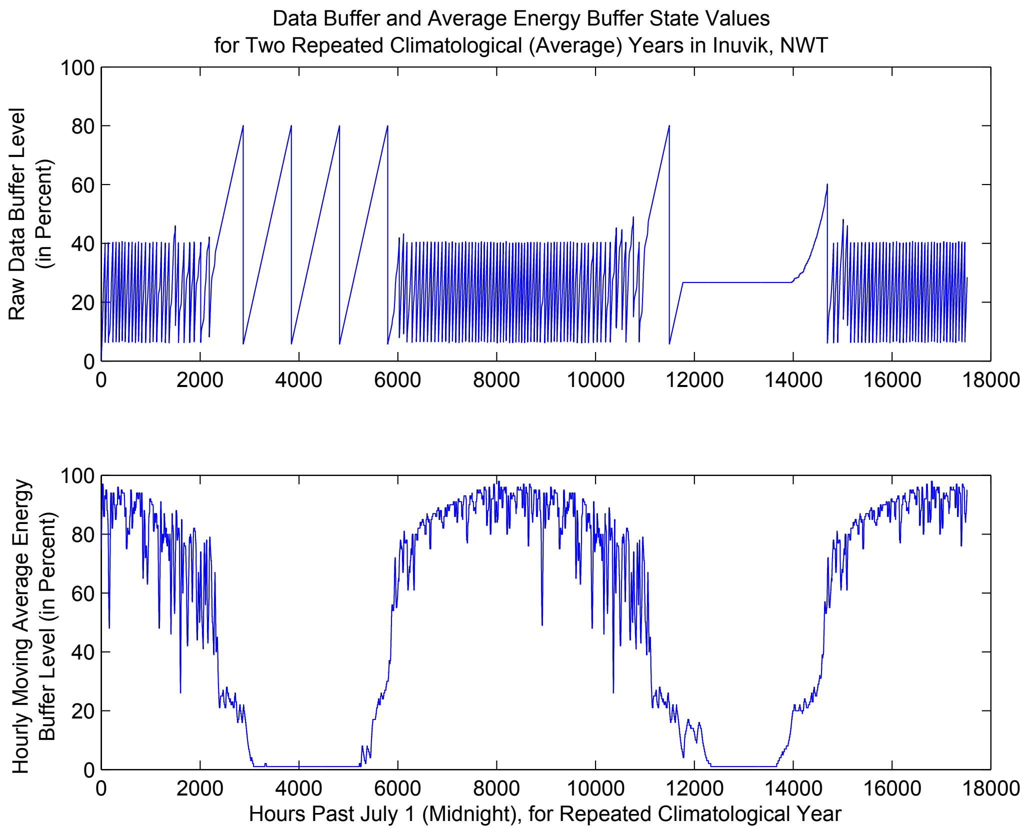


Figure 7.15: This image is based on Figure 5 of [71]. These plots show the state information for the SolarNode's two year simulation using Inuvik TMY data. The upper plot shows the data buffer level. When using the energy buffer, the high rate of data collection quickly fills the data buffer, which is emptied at 40% capacity. Once the controller switches to MIN_{OP} while on the energy reserve, the data buffer fills slowly and is only emptied before it overflows. The long flat segment during the winter of the second year is the extended period of failure after the energy reserve is depleted. The lower plot shows the 24 hour average energy buffer level, which is the processed version of the lower plot in Figure 7.13. The flat top of that plot becomes the high value arches of this plot during the summer. The energy buffer spends most of the time during the day saturated, so it has a high value after being averaged. Because the reserve is not rechargeable, this excess energy is wasted. The sharp drop in buffer value as solar energy disappears shows how small the buffer is in comparison to the platform's rate of energy consumption.

lecting data 6 times an hour, or all that less impressive than collecting data 10 times an hour. The consistency of the data set throughout the year must improve for its value is to increase. Rather than allowing the platform to collect data at the highest rate solar energy will allow, the controller can improve the value of the data set by selecting the highest rate it can achieve consistently, then ensuring this rate is maintained. The energy reserve should be increased in size, not necessarily to increase the number of years the platform can survive, but to provide a higher average level of data capture during the winter. The first year performs much better than the second, as some data is better than no data, but the measurement rate is still too low, particularly when compared to the summer rate. The SolarNode cannot take advantage of the massive solar resource of Polar Day. To improve performance in the Arctic, the SolarNode needs some way to shift its energy surplus in the summer to its energy deficit in the winter, improving its collection rate throughout the year.

7.3.2 Forested Locations

The SolarNode simulations using data from the two forested regions, Fairview and Chamela, are covered in [72] and [95]. These regions do not have the challenges of Polar Night, but energy storage and sizing are still important. The size of the SolarNode's energy buffer and solar panel are both fixed and cannot be changed; however, the SolarNode may equip an energy reserve and it has more control over its activities than the AWS.

Simulation results based on data from Fairview indicate that the capacity of the energy buffer is too close in size to the daily energy consumption of the SolarNode platform. Figure 7.11 and Figure 7.12 show how inconsistent the operation of the platform becomes when environmental energy is scarce. Daily changes in solar energy regularly cause the energy buffer to jump between 10% and 80% capacity during the winter. While averaging the energy buffer level over a 24 hour period helps reduce the noise of this state variable to a point where it useful to the controller, it does not directly attack the problem. Reducing the sensitivity of the system to daily trends reduces oscillations in activity, but these trends are still too extreme, particularly when environmental energy is low. This is shown by Figure 7.16. When the state information changes this much this often, it is very difficult for the controller to select stable actions. The average for the energy buffer level could be extended beyond 24 hours, to filter the signal further, but this reduces the sensitivity of the device. The size of the energy buffer is fixed by the hardware and cannot be increased to act as a low-pass filter like the AWS. Since it is not desirable to reduce the sensitivity of the control system further, this problem should be solved by increasing redundancy in the energy system. Improving the device's robustness to compensate for environmental noise requires the addition of an energy reserve. While the Boreal region does not suffer from Polar Night, the energy reserve can fill in for daily or weekly energy instability, rather than seasonal changes. Instead of eliminating failures by filling in data at MIN_{OP}, the reserve can buffer the present operational level, improving data quality and providing performance guarantees.

A similar issue is present in the Chamela simulation, though it is less pronounced due to the large surplus of energy. The intermittent sharp reductions in performance scattered throughout the year are associated with short term variations in the environment. Because the Chamela simulation does not have any redundancy in the energy system, it cannot ignore these weather

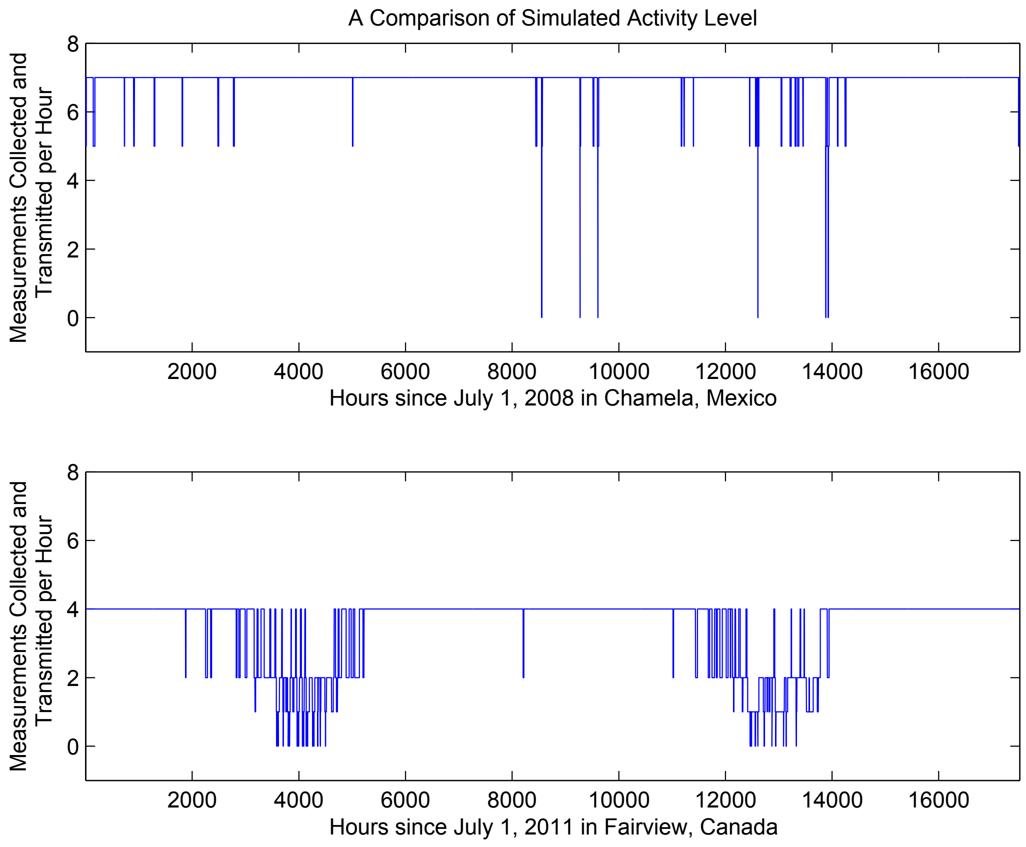


Figure 7.16: This image is from Figure 11 of [95] (©2014 IEEE). These plots compare the measurement frequencies of the two forested regions over a two year simulation with the SolarNode. The upper plot shows the measurement frequencies of the SolarNode using the Chamela data set. Aside from the few points where the platform fails, its activity level is high and consistent throughout the year. The platform will be able to guarantee uninterrupted performance with the addition of an energy reserve. Because these failures are so short and spread out, it will take only a little energy to fill them in and the reserve will last a long time. The lower plot is the measurement activity of the SolarNode using the Fairview data set. The activity level of the device is reduced during the winter, which is desirable, but it oscillates between its selected actions throughout the season. Averaging the energy buffer signal stopped these oscillations from happening daily, but the energy buffer alone cannot eliminate them. It is not large enough to buffer the energy demand from the energy supply for more than a couple days, even at low activity levels. While both regions perform well using only the energy buffer, both would perform better with an energy reserve.

changes and trust that storage will compensate for them. The high activity level the platform holds throughout the year shows that the fixed size of the solar panel is more than sufficient for this application. Given how few failures occur during the night, the energy buffer is approximately the correct size as well. That said, the controller is not sensitive enough to compensate for short term weather events because of the 24 hour moving average separating the energy buffer level from the controller's state information; this is the disadvantage of a simple averaging filter. Without an energy reserve, the buffer is too small to support high levels of performance during sudden drops in energy collection. A reserve will eliminate all energy related failures in both Chamela and Fairview, while allowing more consistent data sets to be collected. When comparing the energy demands of Fairview and Chamela to surviving the whole winter in the Arctic, it is clear that a much smaller energy reserve is required for these regions. Maintenance is inevitable with this sensing equipment, so as long and the replaceable reserve is appropriately sized, then its inclusion will not significantly increase the cost or complexity. However, it will significantly improve the reliability of the energy system of the SolarNode.

The problems that appear in these two regions are not necessarily cause by the size of the energy buffer; the problem is a reliability issue with the whole energy system. The energy buffer is appropriate for these applications, but it cannot be expected to support the entire platform by itself without some help. These observations justify using simulations; the design problems discovered in this phase of the project, which is based on assumptions and estimates, will be very useful in the future.

7.4 Enhancing Performance

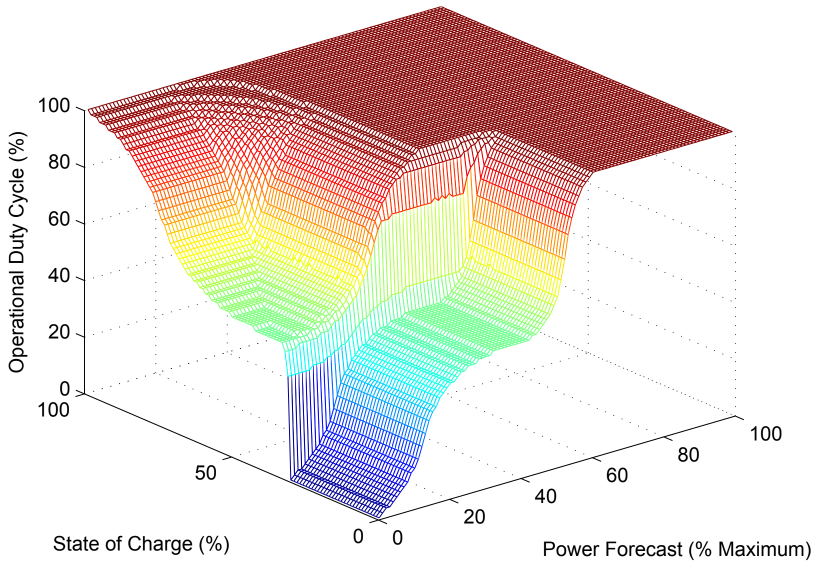
Some discussion on improving the performance of the SolarNode in its various simulated environments has been scattered throughout this chapter. Those points have been omitted to avoid repetition. This section collects some remaining points from [94], [95], and [72], on how to improve performance within the simulation.

7.4.1 The Arctic

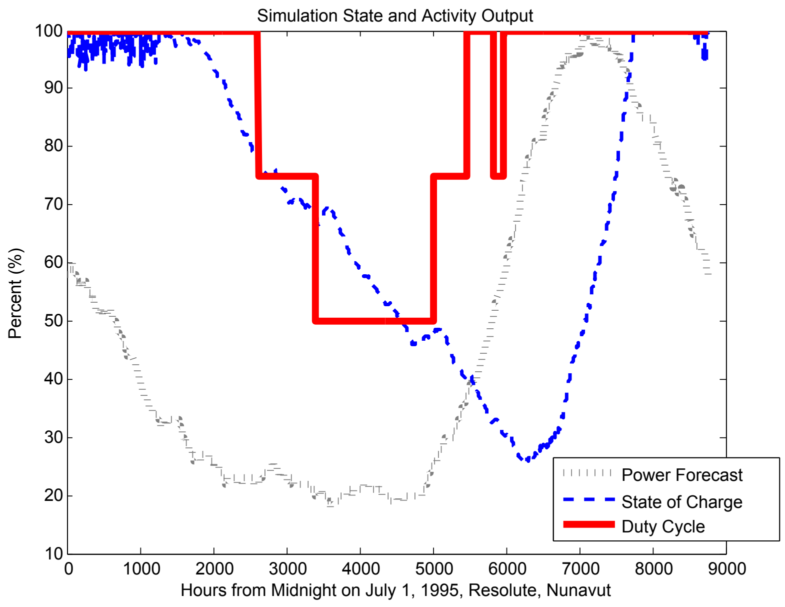
The Arctic simulations involved both the AWS and the SolarNode and took place over more than a year. There were several performance issues that needed to be addressed along the way.

Proper control design, while appearing obvious, was an important first step. Building these controllers can be a quick process and is intended to be a simple as possible. However, if they are designed under incorrect assumptions, it causes serious performance problems for both the simulations and eventual deployed devices. Take, for example, two AWS simulations in the Arctic, both using identical environmental data from July, 1995, in Resolute, Canada. Both simulations use a Hamming distance threshold of 5 units between states before an action transition is allowed. The control surfaces, followed by their corresponding state outputs from the simulator, are provided in Figures 7.17 and 7.18.

These figures show the importance of controller design. Both succeed at adapting to the climate to survive the winter, and both succeed at eliminating oscillations from the action space. However, the controller in Figure 7.17

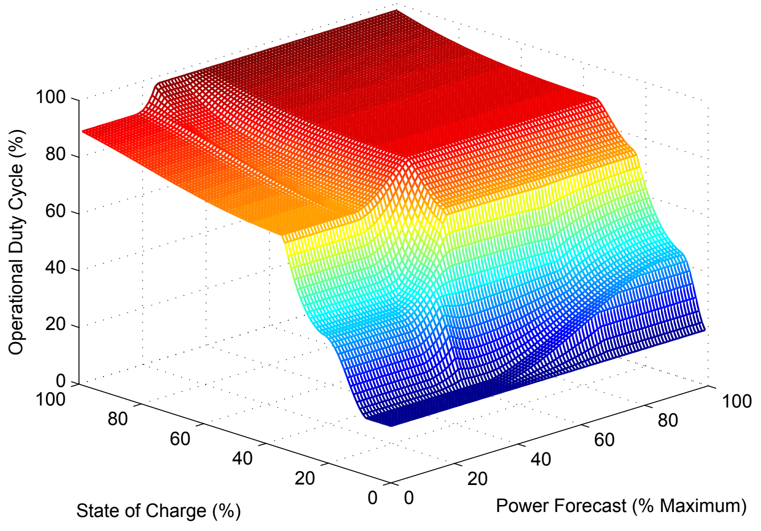


(a) Continuous control surface. This image is from Figure 2 of [94].

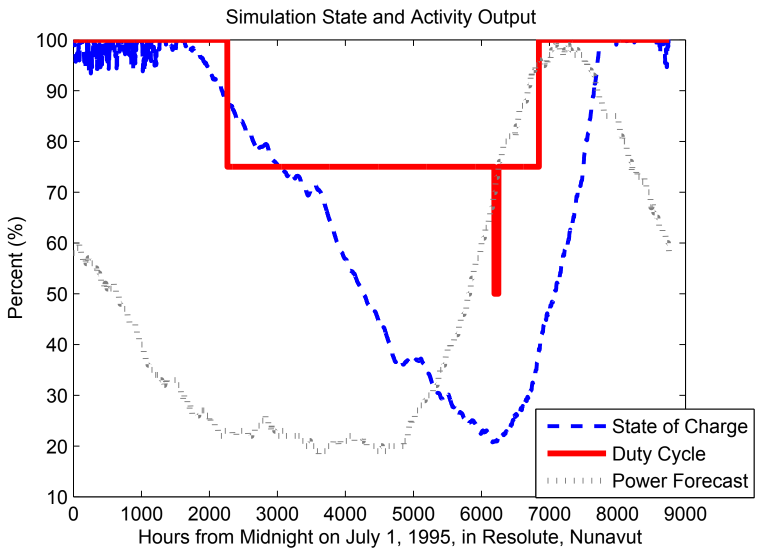


(b) AWS Simulation output. This image is based on Figure 7 of [94].

Figure 7.17: Figure 7.19a (©2013 IEEE) concentrates activity reduction to the bottom corner of the control surface where both the SOC and the power forecast are low. This controller saves energy by cutting the activity level in half for a short time, rather than spreading energy conservation out over the year. Figure 7.19a shows that the controller will attempt to operate at high duty cycle when the SOC is low but the power forecast is high. This is a common state in the spring and can result in multiple repeated failures when the battery is depleted and vulnerable.



(a) Continuous control surface. This image is from Figure 3 of [94].



(b) AWS Simulation output. This image is based on Figure 8 of [94].

Figure 7.18: This control surface was generated automatically using a genetic algorithm. The parameters of the fitness function determined that energy consumption should be reduced over a longer period of time to improve the minimum duty cycle during the winter. The brief drop in duty cycle to 50% at the end of the winter is due to a slight sizing error: if the battery was marginally larger, or if the year had slightly more energy, a consistent duty cycle of 75% could have been achieved for all energy scarce periods. All unnecessary changes in activity would be removed. This controller is more battery concious than the one in Figure 7.17; it does not risk high duty cycles at low battery SOC under any circumstances and it smoothly charges and discharges the battery only once per year. (©2013 IEEE)

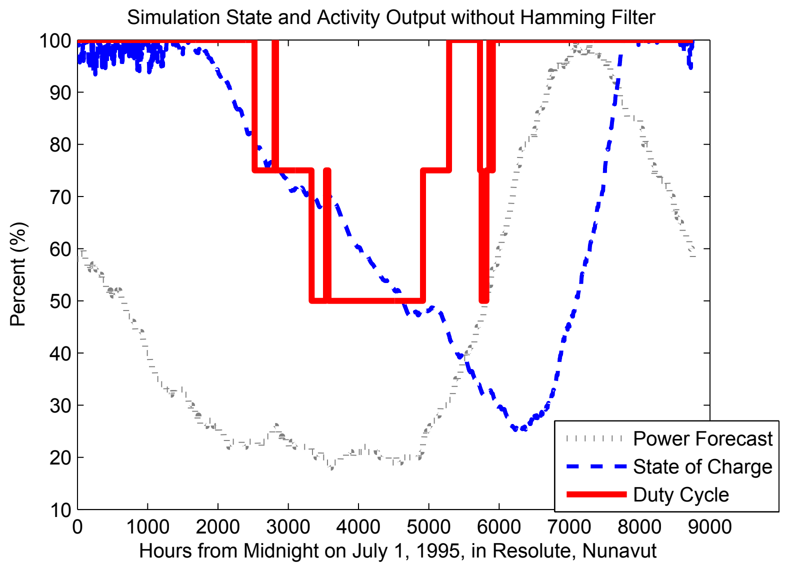
makes a dangerous assumption about the system's response to winter recovery during the spring. The controller assumes that when the SOC is low but the power forecast is high, the monitor should operate with a high duty cycle; this is shown in the top right corner of the controller. This state occurs in the early part of spring, when the surge of summer energy is immanent, but the battery is still discharged from the winter. If the power forecast is too optimistic, then the platform will repeatedly attempt, and fail to reach, this high duty cycle. This prevents the platform from effectively recovering in the spring, but does not appear in the results of Figure 7.19b, because it does not happen every year. The controller in Figure 7.18 takes this into account by keeping the duty cycle low when the SOC is low, and waiting until the battery recovers before risking a high duty cycle.

The results from Figure 7.18 consistently maintain a higher level of performance than Figure 7.17 using the same resources. The difference in performance is not overwhelming; the leap in performance occurs when a controller and filter are added in the first place. The key difference is that the second controller, from 7.18, has been optimized to minimize state transitions and to value consistent, high quality data. The number and magnitude of output transitions from the controller were heavily penalized by the fitness function that guided its development [94]. The first controller, of Figure 7.17, was tuned by an "expert" through trial and error, as most controllers were. The controller could not adapt directly to data from the simulation like the surface in Figure 7.18. Flexibility within the control system is important to allow good trade-offs in the energy-for-data exchange. It is also necessary for data-centric optimization, which avoids the bias of the "expert" tuning process.

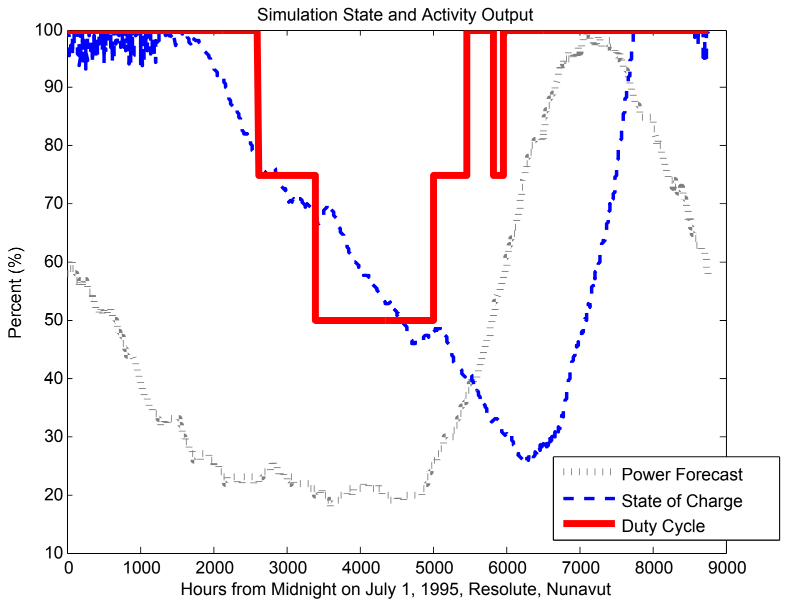
Other variables, like the length of the Hamming filter, can be added to the optimization process to adjust the sensitivity of the controller based upon the available data. Tuning this filter while adjusting control parameters would be difficult without computational assistance. Small changes to this, or other, parameters included in the optimization, directly affect the performance of the controller. Without data-centric optimization, there will not be enough simulations to shown how they interact with each other and their effect on performance.

Filtering state information is an important part of improving the performance of devices in the Arctic. To minimize the computational effort of the controller when making decisions, the distance between state transitions is taken as the Hamming distance between the previous state and the present one. This ensures that the control system maintains simplicity while directly addressing the source of the problem in the state domain by "locking out" excess changes in controller output. Consider the comparison in Figure 7.19. Once again, these figures are from identical conditions to those in Figures 7.17 and 7.18. The Hamming filter directly attacks the problem of AWS measurement consistency in the state domain, and solves it with very few operations, remaining simple and easy to implement.

The simulations using Inuvik data were intended to push the controller to its limits. The period of failure in Figures 7.9 and 7.10 has a more subtle effect on the device than just turning it off. Notice that the data buffer is very close to its maximum level at the end of the spring. Had the device not failed and collected data throughout the winter, that data would have been overwritten and lost, wasting both energy and data. Genetic optimization has made the controller provide the best quality data possible, using the given simulator settings, on its current energy budget. This subtle reduction of waste is not



(a) State output of the AWS simulation without Hamming filter. The image is based on Figure 5 of [94] (©2013 IEEE).



(b) State output of the AWS simulation using a Hamming distance of 5 units between action transitions. This image is based on Figure 7 of [94] (©2013 IEEE).

Figure 7.19: These plots show the difference between filtered and unfiltered controller actions using simulator data for Resolute, Canada. Because the controller is successful at eliminating failure, it is not forced towards its weak points; however, the advantage of the simple filtering operation is still apparent. A slightly larger filter may have eliminated the small drop in duty cycle just before spring. The filter allows the controller to change its activity level without a period of inconsistency.

easily achieved by the trial and error of an “expert”.

When it comes to using the SolarNode, rather than the AWS, with the Arctic data, the filtering problem changes. The energy buffer level is averaged for 24 hours of data. This value was selected arbitrarily to match the daily solar cycle, but no testing for other lengths has been done. All SolarNode results have highly variant action spaces when adapting to environmental conditions. Unlike the AWS, which has a large energy buffer to provide a great deal of energy security, the SolarNode has a small buffer that cannot use the Hamming filter. Because the average energy buffer spends most of its time near maximum capacity in Figure 7.15, it shows that SolarNode does not have enough ways to invest its excess energy during the summer, and a large amount of energy is wasted. This cannot be solved by increasing the sampling rate of the sensors, because that only reduces waste for part of the year; a glut of summer measurements do not make up for a dearth of winter ones. The SolarNode needs more ways to shift energy consumption from the energy-scarce winter to the energy-rich summer, or to store excess summer energy for consumption during the winter.

7.4.2 The Boreal Forests

The results using Fairview data, from the Boreal region, show a flaw in controller design. The high, smooth curve of the average energy buffer level during the summer, shown in Figure 7.11, indicates that the maximum operation setting of the device is far too low. A significant amount of energy is wasted that could be used to increase the data collection rate during the summer. This is demonstrated by the flat collection rate of 4 measurements per hour during all points of the year outside of winter in Figure 7.12. Far more data could have been collected if the controller been structured differently. That said, these results were generated for comparison with the naïve strategies discussed in Section 7.2.1, so that the minimum operational level of the controller would match the "Minimum" strategy and the maximum operational level of the controller would match the "Maximum" strategy. Increasing the maximum operational level of the device would result in better performance during the summer, but would have made the corresponding "Maximum" naïve strategy pathetically dysfunctional.

Increasing the energy consumption of the SolarNode in Fairview could be combined with an additional energy reserve, and some sort of filter for the action space, to produce some impressive results. The poor data consistency of the winter months could be eliminated by the filter, while the corresponding reduction in controller sensitivity could be solved by the energy reserve. This would help reduce energy scarcity in the winter, and improve the overall average activity level of the device. The system would be able to provide some guarantee of performance to the user in this application, creating a robust monitoring solution.

7.4.3 The Advantage of the Tropics

The climate of a tropical dry forest is ideal for solar energy based environmental monitors. The SolarNode thrives in Chamela, Mexico, and the consistent climate requires minimal adaptation to environmental conditions. Even while capturing a considerable amount of data, Figure 7.7 shows the average energy buffer level hovering around 79%. The largest increase in performance

Table 7.6: A comparison of remote monitor performance for Tropical Dry forest and Boreal forest [95]

Property	Fairview	Chamela
Measurements per Hour	1 to 4	1 to 7
Maximum Possible Measurements	70080	122640
Collected Measurements	61731	121696
Number of Failures	91	8
Failure as Percent of Collected	0.1474%	0.0066%
Optimal Total Solar Harvest	348.193Wh	591.297Wh
Measurements per Watt Hour	177.3	205.8

for simulations in this area will come from the addition of an energy reserve. It will easily eliminate all inconsistencies in measurement rate and will last a long time because it is only needed to fill in for short term weather problems. Table 7.6 shows that the device failed for 8 data points, all of which are easily filled by battery powered operation.

In a whole year the solar cell in Chamela can provide approximately 591.30Wh, while the solar cell in Fairview can only provide 348.19Wh. The Fairview location only has 59% of the solar energy in Chamela to invest in data collection, storage, and transmission. Given the significant difference in solar energy resources between the two locations, a direct comparison between their number of measurements and transmissions is inappropriate as a measure of comparative performance and does not provide insight into the energy management strategy at work. Using the data from Chamela, 121696 data points are captured, rather than the 61731 data points from using the Fairview data. The Chamela data set provides about 1.7 times more solar energy, which allows about 2 times the data to be collected. Given that Fairview has its rate of data collection limited to four measurements per hour during the summer, this difference is not surprising.

Comparing the performance of these two contrasting environments requires the idea of an energy for data exchange. The environmental data from Chamela provides more resources and allows more data to be collected than the environmental data from Fairview; however, from the energy management perspective, it is not the *amount* of energy that is important, but how that energy *utilized*. To analyse the results of these two simulations, the number of data points collected *per unit energy* should be considered. Fairview has less energy, and given that energy is the limiting factor in the energy for data exchange, it produces less data. The simulation using the data from Chamela captures 205.81 data points per watt hour, while the simulation using the data from Fairview only captures 177.29 data points per watt hour. The energy management strategy for the Chamela simulation is still the more effective of the two, but it is only better by a factor of 1.16. This is not as good as it initially appeared: collecting double the data using double the energy is only marginally better than the simulation from Fairview. Considering that the controller for the Fairview simulation did not permit the monitor to collect measurements at a rate higher than 4 per hour, it is clear that if the controllers of both locations were optimized, their difference in performance will shrink as the Fairview controller improves.

Section 7.2 showed that adapting the duty cycle of the simulated device to its environment significantly improved performance. It is important for the duty cycle of the device to suit its environmental energy profile. Section 7.2.1 shows that a static energy management strategy performs well in Chamela, because Chamela has static energy profile. The duty cycle in Fairview must be dynamic because its energy profile is dynamic.

Regardless of optimization technique, the peak value of measurements per unit energy is presently dependent upon the parameters of the simulation, but when a remote monitor is deployed, this value will depend the hardware platform. To remain useful these simulations must be a part of the iterative development of the whole monitoring project.

Chapter 8

Conclusion

8.1 Summary

. Environmental monitoring is necessary to understand natural systems. These systems are constantly changing and interacting, with profound consequences to humanity. To investigate global phenomena, data collection requires a massive spatial and temporal scope, while retaining the resolution necessary to observe environmental processes. The efficacy of this data centric approach, and our understanding of the environment, is directly related to the quality of the data collected by environmental monitoring.

Historically, environmental monitoring has been constrained by instrument precision and human resources; however, automated monitoring systems have proven to be an effective data collection methodology. Collecting data in an area for extended periods typically requires the monitoring equipment to be embedded within environment for the duration of the study. The hardware must be independent and robust to operate without access to infrastructure or maintenance. It also becomes responsible for its own energy supply. All automated monitoring systems have some energy management strategy that determines how they invest their energy resources. This may be intentionally designed before hand, or unintentionally enforced by the limitations of the hardware. All too often, monitoring devices are left with an energy management strategy that cannot adapt to their environment.

A good energy management strategy will ensure the monitor's resources are invested in a way that collects high quality data. In an energy constrained environment, the monitoring device with the best energy management strategy will collect the highest quality data. Some recent theoretical frameworks for energy management consider the addition of energy harvesting devices to support monitoring operations; these frameworks help guide and formalize the design process. Initially, theoretical work was centralized on energy problems, and avoided the data side of the problem. However, the production of data, through collection, is the objective of energy management, and the monitor's performance is ultimately judged by the merits of its data set. This naturally extends to examining the rate of energy-for-data exchange, which is a slightly different perspective. Data collection is just as important as energy investment: respectively comprising the *ends* and *means* of environmental monitoring.

Developing and testing energy management strategies in the field can be risky and expensive. The opportunity cost of investing resources in experimental trials involves the time and money expended to deploy devices, along with the data that may have been collected if the management strategy fails. Risk is integral to advancement, so to mitigate it, and explore the problem space, a simulation has been created. It allows energy management to be studied without sacrificing equipment and provides a convenient computational medium for system optimization. Simulation allows the value of data produced in the energy for data exchange to be linked to the resources invested by the monitor's energy management strategy. This informs design choices: performance is determined by data, but enabled by energy; the simulation allows these to be easily connected. This allows energy management strategies to be tested and optimized by trial and error, or some other computational method, reducing the risk and expense of the project. The simulations provide some indication of what will work and what will not.

An energy management strategy can be divided into the control system and the techniques of energy investment. These techniques are the tools the monitoring equipment can use to invest its energy resources, while the control system observes the state of the monitor and determines the best actions for it to carry out. Fuzzy systems provide a direct way to construct the robust, yet simple, control system this application requires. They can use a combination of expert knowledge and computational optimization to adapt control for may different environments. Extreme environments show massive performance improvements when including a fuzzy control system in their energy management strategy.

The results from the simulation focus on three main areas: adaptation, design, and iterative improvement. When a monitoring system depends upon an environmental energy source, it must adapt how it operates to the resources available. The monitoring system has no way of changing its environment, so it must tolerate energy shortages and inhospitable climates. By matching energy consumption to energy collection, and storing energy when it is abundant to compensate for when it is scarce, the monitoring system efficiently produces quality data. Providing the components and control required to adapt to the environment increases the complexity of the monitoring solution. It is important for the energy and data systems of the monitoring platform to be correctly sized. Appropriately sized energy collection, through energy harvesting, ensures there is enough power, on average, for the platform to operate, while appropriately sized energy storage ensures that this collected energy can be distributed throughout the operating period. Elements of the energy and data systems may be oversized to increase redundancy against failure. This provides operational security but increases the bulk and cost of the monitoring platform. If these systems are designed correctly, they will provide enough flexibility to the energy management strategy for it to adapt to its environment and change its priorities in real time. Developing an energy management strategy is an iterative process; environmental monitoring systems and their controllers are never truly optimal. Looking at the simulated results always shows some way to improve performance; the simulator must be continually improved as well.

The final step, and true test of this work, involves deploying one of these environmental monitoring devices with an energy management strategy. Thus far, the simulations are built from simplifications, estimates, and data sheets, but deployment will tie all this back to the physical world and will validate this work. Improving the simulation has, and will continue to be, an iterative process. The first field deployments will result in significant changes to the simulation, but this will strengthen this research, not detract from it.

8.2 Contributions

This work makes some contributions to the field of energy management for environmental monitoring.

First, the beginning of the document collects and summarizes information on the current state of energy management for environmental monitoring. This includes the hardware required for energy harvesting and storage, along with the computational techniques of fuzzy systems, for simple controller design, and genetic algorithms, for optimization. Additionally, the predominant ideas on energy management are summarized and explained.

Second, the software simulation tools used to investigate energy management are explained in detail. Background information on monitoring hardware platforms are provided for some grounding in the physical world. The simulation itself is an important tool for predicting how these hardware platforms will interact with their surroundings, and how their energy management strategies will affect performance. Once this simulation is updated by feedback from deployment, it will become a powerful design tool. The methodology to design the controller for the management strategy is included; it shows how "expert knowledge" can be turned into a useful part of software. This makes the content more available to fields of environmental research outside of engineering and computer science that are not involved with control theory. The simulation, in its current form, is advanced enough to be changed to suit other monitoring platforms and configurations outside of the ones described here.

Finally, the applied controller and simulation sections conclusively demonstrate important results related to energy management theory. First, energy harvesting is necessary for long-term low-maintenance environmental monitoring; it is the only effective way to establish an energy for data exchange.

Second, the monitoring platform must be designed with a high degree of flexibility to adapt to its environment, or a high degree of reliability to buffer it from the environment. These conclusions are supported by the results of comparing naïve, or static, energy management to adaptive energy management. Adaptive strategies perform much better in all cases except where the naïve strategy already matches the environmental energy profile of the region; in this case no significant change in performance was anticipated. These results show weaknesses in current design practices and explains why they are flawed. Energy management significantly improves performance for remote monitoring devices while only marginally increasing project complexity and system overhead. The use of computational methods for designing, implementing, and optimizing these energy management strategies show that simple, "expert-based" knowledge from the field can be passed on to automated devices to provide better quality data collection. This may allow specialists in the field of environmental monitoring to instruct energy management to make good trade-offs in the energy-for-data exchange.

8.3 Future Work

Future work on this project can go in several different directions. Improvements and updates to the simulation will make it a powerful and accurate design tool; this could involve increasing its internal control and detail, or expanding its scope to networked platforms. A final speculation on the energy for data exchange closes the section.

The Simulation

The first and most important improvement for the simulation is field testing. When the simulation was first developed, field testing seemed a long way off, but with work on the hardware platform since the simulation's creation, the device is up and running and has seen some short term deployments in the field already. With this in mind, the numbers and settings the simulator uses to estimate the energy consumption and data production of the device need to be corrected. Some laboratory testing of the SolarNode hardware platform has already allowed the first iteration of updates to be brought into the simulation [70]. As this continues, it will provide the simulation with useful, predictive capabilities.

Actions within the simulation are treated as independent events, and the overhead to execute them is largely ignored. Once the simulation considers the overhead of activities, the control system will be able to learn interesting energy saving behaviours. For example, network communication is not as simple as dumping data onto the network at a certain energy cost. Connections need to be established and maintained, which costs energy but does not directly involve the transfer of collected data. By transmitting data in batches, the energy cost of this overhead is distributed over each packet of data. Including these subtleties in the model allows the controller to operate the hardware more efficiently. The simulation generally lacks detail, but it will require data from the field to determine where this additional detail will have a meaningful impact. It is possible some portions of the simulator will end up begin simplified. As another example, the sensors used by the simulated device are not individually controllable, which will become a serious problem if the number and variety of sensors increase. Energy intensive sensing technologies will require their own special control. Additionally, the energy storage technologies are less detailed than they should be. This is a problem because supercapacitors, in particular, exhibit complex behaviour when it comes to charging and leakage [96].

The simulation needs to migrate out of the Mathworks environment into something more accessible and portable. While MATLAB and Simulink are convenient, they are locked behind licences and restrict spreading the simulation to other people and computing platforms. Changing platforms allows a more modular and accessible approach for other people to explore the energy management issues discuss in this document. It would also be good to separate the timestep of the simulation from the timestep of the environmental data set. This timestep is common between modules, but it would be interesting to see modules interact asynchronously; they could become far more independent of one another.

The movement of data within the simulation requires an additional level of detail: metadata. When the simulation collects data, it cannot be stored as disorganized heap. This is currently a convenient simplification, but for the simulator to eventually incorporate networking features and advanced sensing techniques, there needs to be some metadata associated with data points. The control system will need to know the difference between data from the network and data from its own sensors. This will facilitate adaptive sensing techniques, where the activity of the platform adapts to phenomena it monitors, as well as its energy resources. This upgrade will be expedited by increasing the detail of sensors within the simulation.

The Management Strategy

The energy management strategies used in the simulator require improvement. The current fuzzy system is rigidly structured and only covers a small part of the potential state and action spaces. As the control system enters a higher dimensional problem space, there will be more opportunities for performance improvements and interesting learning opportunities for the energy management strategy. So far there have only been offline attempts at controller design and optimization; the controller is built before a round of simulation is started, and updates only occur between rounds of simulation. While the controller does operate in real time, it would be an interesting addition to see it adapt in real time as well. This may require moving towards neurocomputing or reinforcement and machine learning. Alternately, evolutionary strategies could be used instead of genetic algorithms, to modify the population of controllers rather than replacing then. Both of these options require an online reward or value system in place, to evaluate the fitness of the simulation's data production in real time.

Given the opportunity to repeat this project, three things would be done differently. First, the simulation would immediately leave the confines of the Mathworks environment. This would make diversifying the project easier and open up more computational options. It would also make it easier to integrate this work into other simulation software. Secondly, the simulation would focus on modularity to allow parallel computing for testing and optimization. Finally, optimization and fitness evaluation would be changed. Genetic or evolutionary optimization would become truly multiobjective to allow data properties and device operation to be independently optimized. Fitness would be evaluated online either for learning, or to facilitate data-centric device management in the future. Both of these options open interesting research opportunities into computational intelligence, but would be difficult to approached within the confines of the Mathworks environment.

Upon leaving the Mathworks environment, it would be interesting to see this simulation interfaced with other software for wireless monitoring. This work largely avoids networking issues and instead focuses on hardware management. In that sense, it is unique: computer science has surged into the area of efficient networking but seems to avoid the detailed electronics of monitoring systems. The focus in wireless sensor networks appears to be complexity at the network level, which is more relevant to computer systems. It will also be interesting to see the simulation adopt more hardware platforms, rather than just the SolatrNode or the AWS. This will require new energy management strategies to handle these different control systems and, perhaps, different energy harvesting techniques.

A Final Thought

To close, there is one, final, hypothetical indulgence to put forward. It has floated in the background throughout the project. Networked environmental monitoring systems operate a lot like a small economy; an economy where intelligent, distributed agents collect and invest resources for some sort of mutual benefit. Within environmental monitoring, energy is the resource or capital, while data is the goods or services. The perceived value of this data is the nebulous *reward* concept for optimization and evaluation. This is similar, in a sense, to the fiat currency used day to day.

Consider, for example, a small network of devices around a central base station, which is connected to a data base in a remote server. The server is evaluating the data provided by the network in real time and assigning each data point some value based upon the application; perhaps this is how important the data is to research, what a company will pay for the data, or what computational resources are required to process the data.

If one node in the network provides data in a continuous static signal, the same point, over and over, the value of this data point is very low, because it can be easily predicted. The external server will *buy it* for very little *reward* from the base station, so the base station provides very little *reward* for this data to the node. The node reduces its sampling rate to help keep its average reward per energy expenditure favourable.

Now, consider another node sampling highly variant data that is spatially distinct from the other nodes; this data may even be critical to some application. The data rate for this device would be inflated by high rewards from the server, or user, via the base station, and energy would be invested to procure additional reward. Effectively, a supply chain has been created, carrying valuable data forward to the server, and thus the user, and a signal carrying the *reward* backward to the node. Because each layer of nodes passes data forward and *rewards* backward, the network is distributed, modular, and self-organizing, just like a small market, establishing the value for each data set. Links in the network that are energy poor cannot *buy* data they cannot afford to transmit, which should prevent network fragmentation by scaling down performance. This provides an interesting model for wireless sensor networks, but also an amusing commentary on economics. Applications in machine learning would extend from environmental science to finance and associate environmental sensing with the new data economy that is appearing with the Internet of

Things, the rise of data analytics, and the advent of digital currencies.

Bibliography

[1] ACIS. Agroclimatic information service. `http://agriculture.alberta.ca/acis/about.jsp`, April 2014.

[2] ACIS. Current and historical Alberta weather station data viewer. `http://agriculture.alberta.ca/acis/alberta-weather-data-viewer.jsp`, April 2014.

[3] I.F. Akyildiz, Weilian Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *Communications Magazine, IEEE*, 40(8):102–114, Aug 2002.

[4] Alberta Agriculture and Rural Development. Agriculture and rural development: Ropin' the web. `http://www.agric.gov.ab.ca/app21/rtw/index.jsp`, April 2014.

[5] C. Alippi, G. Anastasi, M. Di Francesco, and M. Roveri. Energy management in wireless sensor networks with energy-hungry sensors. *Instrumentation Measurement Magazine, IEEE*, 12(2):16–23, April 2009.

[6] Atmel Corporation, San Jose, CA. *8-bit AVR Microcontroller with Low Power 2.4GHz Transceiver for ZigBee and IEEE 802.15.4*, February 2013.

[7] Henrik Bindner, Tom Cronin, Per Lundsager, James F. Manwell, Utama Abdulwahid, and Ian Baring-Gould. Lifetime modelling of lead acid batteries. Technical Report Risø-R-1515(EN), Roskilde, Denmark, April 2005. [Electronic Resource].

[8] S. Boon, D. O. Burgess, R. M. Koerner, and M. J. Sharp. Fourty-seven years of research on the Devon Island Ice Cap, Arctic Canada. *Arctic*, 63(1):13–29, March 2010. [Electronic Resource].

[9] D. Brunelli, C. Moser, L. Thiele, and L. Benini. Design of a solar-harvesting circuit for batteryless embedded systems. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 56(11):2519–2528, Nov 2009.

[10] A. Cerpa and D. Estrin. Ascent: adaptive self-configuring sensor networks topologies. *Mobile Computing, IEEE Transactions on*, 3(3):272–285, July 2004.

[11] PVPerformance Modelling Collaborative. Modelling steps. `http://pvpmc.org/modeling-steps/`, 2013.

[12] Cooper Bussmann, Boca Raton, FL. *PowerStor Supercapacitors HV Series*, 2013.

[13] Waltenegus Dargie. Dynamic power management in wireless sensor networks: State-of-the-art. *IEEE Sensors Journal*, 12(5):1518–1528, May 2012.

[14] Energizer Battery Manufacturing, Inc., Westlake, OH. *Lithium Iron Disulfide Batteries Product Safety Datasheet*, January 2014.

[15] Energizer Holdings, Inc., Westlake, OH. *Energizer L91 Ultimate Lithium Product Datasheet*, January 2014.

[16] A. P. Engelbrecht. Fuzzy controllers. In *Computational Intelligence: An Introduction*, chapter 22, pages 475–479. John Wiley and Sons, Ltd., Chichester, England, 2nd edition, 2007.

[17] A. P. Engelbrecht. Genetic algorithms. In *Computational Intelligence: An Introduction*, chapter 9, pages 143–176. John Wiley and Sons, Ltd., Chichester, England, 2nd edition, 2007.

[18] Enviro-Net. University of Alberta's Enviro-Net home page. `http://www.enviro-net.org/`, April 2014.

[19] D.B. Fogel. An introduction to simulated evolutionary optimization. *Neural Networks, IEEE Transactions on*, 5(1):3–14, Jan 1994.

[20] Michael F. Gard. A status report on environmental monitoring. *IEEE Transactions on Instrumentation and Measurement*, 51(4):782 – 785, August 2002.

[21] G. Gascon, M. Sharp, and A. B. G. Bush. Changes in melt season characteristics on the Devon Ice Cap, Canada, and their association with the Arctic atmospheric circulation. *Annals of Glaciology*, 54(63):101–110, 2013.

[22] J.J. Grefenstette. Optimization of control parameters for genetic algorithms. *Systems, Man and Cybernetics, IEEE Transactions on*, 16(1):122–128, Jan 1986.

[23] F. Guély and P. Siarry. Gradient descent method for optimizing various fuzzy rule bases. In *Proceedings of the Second IEEE International Conference on Fuzzy Systems*, pages 1241–1246, 1993.

[24] Romain Guigon, Jean-Jacques Chaillout, Thomas Jager, and Ghislain Despesse. Harvesting raindrop energy: Theory. *Smart Materials and Structures*, 17:1–8, January 2008.

[25] Jane K. Hart and Kirk Martinez. Environmental sensor networks: A revolution in the earth system science? *Earth-Science Reviews*, 78:177–191, May 2006.

[26] W.B. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *Wireless Communications, IEEE Transactions on*, 1(4):660–670, Oct 2002.

[27] A.O. Hero and D. Cochran. Sensor management: Past, present, and future. *Sensors Journal, IEEE*, 11(12):3064–3075, Dec 2011.

[28] M. Horowitz, T. Indermaur, and R. Gonzalez. Low-power digital design. In *Low Power Electronics, 1994. Digest of Technical Papers., IEEE Symposium*, pages 8–11, Oct 1994.

[29] J. Hsu, S. Zahedi, A. Kansal, M. Srivastava, and V. Raghunathan. Adaptive duty cycling for energy harvesting systems. In *Low Power Electronics and Design, 2006. ISLPED'06. Proceedings of the 2006 International Symposium on*, pages 180–185, Oct 2006.

[30] IXYS Corporation, Milpitas, CA. *IXOLAR High Efficiency SolarBIT*, Aug. 2011.

[31] J.-S.R. Jang. ANFIS: adaptive-network-based fuzzy inference system. *Systems, Man and Cybernetics, IEEE Transactions on*, 23(3):665–685, May 1993.

[32] Ravindra Jejurikar, Cristiano Pereira, and Rajesh Gupta. Leakage aware dynamic voltage scaling for real-time embedded systems. In *Proceedings of the 41st Annual Design Automation Conference*, DAC '04, pages 275–280, New York, NY, USA, 2004. ACM.

[33] Xiaofan Jiang, Joseph Polastre, and David Culler. Perpetual environmentally powered sensor networks. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, IPSN '05, Piscataway, NJ, USA, 2005. IEEE Press.

[34] A. Kansal, J. Hsu, M. Srivastava, and V. Raghunathan. Harvesting aware power management for sensor networks. In *Design Automation Conference, 2006 43rd ACM/IEEE*, pages 651–656, 2006.

[35] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava. Power management in energy harvesting sensor networks. *ACM Transactions on Embedded Computer Systems*, 6(4), September 2007. [Electronic Resource].

[36] A. Kansal, D. Potter, and M. B. Srivastava. Performance aware tasking for environmentally powered sensor networks. *ACM SIGMETRICS Performance Evaluation Review*, 32(1):223–234, June 2004. [Electronic Resource].

[37] A. Kansal and M.B. Srivastava. An environmental energy harvesting framework for sensor networks. In *Low Power Electronics and Design,*

2003. ISLPED '03. Proceedings of the 2003 International Symposium on, pages 481–486, Aug 2003.

[38] Safa O. Kasap. Photovoltaic devices. In *Optoelectronics and Photonics Principles and Practices*, chapter 6, pages 254–274. Prentice-Hall Inc., Upper Saddle River, New Jersey, 2001.

[39] Colin J. Legg and Laszo Nagy. Why most conservation monitoring is, but need not be, a waste of time. *Journal of Environmental Management*, 78:194 – 199, 2006.

[40] E.H. Mamdani. Application of fuzzy algorithms for control of simple dynamic plant. *Electrical Engineers, Proceedings of the Institution of*, 121(12):1585–1588, December 1974.

[41] E.H. Mamdani and N. Baaklini. Prescriptive method for deriving control policy in a fuzzy-logic controller. *Electronics Letters*, 11(25):625–626, December 1975.

[42] Microchip Techology Inc., Chandler, AZ. *1024K I2C CMOS Serial EEPROM*, 2005.

[43] C. Moser, L. Thiele, D. Brunelli, and L. Benini. Adaptive power management for environmentally powered systems. *Computers, IEEE Transactions on*, 59(4):478–491, April 2010.

[44] Harini Nagendra, Catherine Tucker, Laura Carlson, Jane Southworth, Mukunda Karmacharya, and Birendra Karna. Monitoring parks through remote sensing: Studies in Nepal and Honduras. *Environmental Management*, 34(5):748 – 760, October 2004.

[45] Michael Nagnevitsky. Fuzzy expert systems. In *Artificial Intelligence: A Guide to Intelligent Systems*, chapter 4, pages 87–128. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 2 edition, 2004.

[46] Natural Resources Engineering Laboratory (NREL). Vehicles & fuels research: Ultracapacitors. `http://www.nrel.gov/vehiclesandfuels/energystorage/ultracapacitors.html`, September 2009.

[47] Natural Resources Engineering Laboratory (NREL). Learning about renewable energy: Batteries. `http://www.nrel.gov/learning/eds_batteries.html`, May 2012.

[48] Natural Resources Engineering Laboratory (NREL). Learning about renewable energy: Supercapacitors. `http://www.nrel.gov/learning/eds_supercapacitors.html`, May 2012.

[49] Natural Resources Engineering Laboratory (NREL). National center for photovoltaics. `http://www.nrel.gov/ncpv/`, May 2013.

[50] Natural Resources Engineering Laboratory (NREL). Research cell efficiency records. `http://www.nrel.gov/ncpv/images/efficiency_chart.jpg`, May 2013.

[51] Natural Resources Engineering Laboratory (NREL). Vehicles & fuels research: Batteries. `http://www.nrel.gov/vehiclesandfuels/energystorage/batteries.html`, January 2013.

[52] M.S. Olsen, T.V. Callaghan, J.D. Reist, L.O. Reiersen, D. Dahl-Jensen, M.A. Granskog, B. Goodison, G.K. Hovelsrud, M. Johansson, R. Kallenborn, J. Key, A. Klepikov, W. Meier, J.E. Overland, T.D. Prowse, M. Sharp, W.F. Vincent, and J. Walsh. The changing Arctic cryosphere

and likely consequences: An overview. *AMBIO*, 40:111–118, 2011. [Electronic Resource].

[53] Elinor Ostrom and Harini Nagendra. Insights on linking forests, trees, and people from the air, on the ground, and in the laboratory. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 103, pages 19224 – 19231, December 2006.

[54] Chulsung Park and P.H. Chou. Ambimax: Autonomous energy harvesting platform for multi-supply wireless sensor nodes. In *Sensor and Ad Hoc Communications and Networks, 2006. SECON '06. 2006 3rd Annual IEEE Communications Society on*, volume 1, pages 168–177, Sept 2006.

[55] Camille Parmesan and Gary Yohe. A globally coherent fingerprint of climate change impacts across natural systems. *Nature*, 421:37 – 42, January 2003.

[56] G. Z. Pastorello, G. Arturo Sanchez-Azofeifa, and M. A. Nascimento. Enviro-net: From networks of ground-based sensor systems to a web platform for sensor data management. *Sensors*, 11(6):6454–6479, 2011.

[57] Gilberto Z. Pastorello, G. Arturo Sanchez-Azofeifa, and Mario A. Nascimento. Enviro-net: A network of ground-based senors for tropical dry forests in the americas. In *Proceedings of the* 34^th *Internation Symposium on Remote Sensing of Environment*, April 2011.

[58] Steven G. Paulsen, Robert M. Hughes, and David P Larsen. Critical elements in describing and understanding our nation's aquatic resources. *Journal of the American Water Resource Association*, 34(6):995 – 1005, October 1998.

[59] W. Pedrycz and F. Gomide. From logic expressions to fuzzy logic networks. In *Fuzzy Systems Engineering: Toward Human Centric Computing*, chapter 12, pages 335–382. John Wiley and Sons, Inc., Hoboken, NJ, 1st edition, 2007.

[60] W. Pedrycz and F. Gomide. Fuzzy modeling: Principles and methodology. In *Fuzzy Systems Engineering: Toward Human Centric Computing*, chapter 10, pages 252–275. John Wiley and Sons, Inc., Hoboken, NJ, 1st edition, 2007.

[61] W. Pedrycz and F. Gomide. Fuzzy relations. In *Fuzzy Systems Engineering: Toward Human Centric Computing*, chapter 6, pages 139–156. John Wiley and Sons, Inc., Hoboken, NJ, 1st edition, 2007.

[62] W. Pedrycz and F. Gomide. Operations and aggregations of fuzzy sets. In *Fuzzy Systems Engineering: Toward Human Centric Computing*, chapter 5, pages 101–138. John Wiley and Sons, Inc., Hoboken, NJ, 1st edition, 2007.

[63] Richard Perez, Robert Seals, Pierre Ineichen, Ronald Stewart, and David Menicucci. A new simplified version of the Perez diffuse irradiance model for tilted surfaces. *Solar Energy*, 39(3):221–231, 1987.

[64] D. Pimentel. *Energy Management for Automatic Monitoring Stations in Arctic Regions*. Ph.D. dissertation, Electrical and Computer Engineering, University of Alberta, 2012.

[65] D. Pimentel and P. Musilek. Fuzzy power management for automatic monitoring stations in the Arctic. In *Proceedings of the Twenty-first (2011) International Offshore and Polar Engineering Conference*, Maui, HI, 2011.

[66] D. Pimentel, P. Musilek, and A. Knight. Energy harvesting simulation for automatic Arctic monitoring stations. In *Proceedings of the Electric Power and and Energy Conference (EPEC)*, Halifax, NS, 2010.

[67] D. Pimentel, P. Musilek, A. Knight, and J. Heckenbergerova. Characterization of a wind flutter generator. In *Proceedings of the 9th International Conference on Environmental and Electrical Engineering*, Prague, 2010.

[68] John Porter, Peter Arzberger, Hans-Werner Braun, Pablo Bryant, Stuart Gage, Todd Hansen, Paul Hanson, Chau-Chin Lin, Timothy Kratz, William Michener, Sedra Shapiro, and Thomas Williams. Wireless sensor networks for ecology. *BioScience*, 55(7):561 – 572, July 2005.

[69] John H. Porter, Paul C. Hanson, and Chau-Chin Lin. Staying afloat in the sensor data deluge. *Trends in Ecology and Evolution*, 27(2):121–129, February 2012.

[70] Michal Prauzek, P. Musilek, and Asher G. Watts. Fuzzy algorithm for intelligent wireless sensors with solar harvesting. In *IEEE Symposium on Intelligent Embedded Systems (IES) (submitted)*, December 2014.

[71] Michal Prauzek, P. Musilek, Asher G. Watts, and Marketa Michalikova. Powering environmental monitoring systems in Arctic regions: A simulation study. In *Electronics 2014, KTU 18th International Conference*, June 2014.

[72] Michal Prauzek, Asher G. Watts, P. Musilek, L. Wyard-Scott, and Juri Koziorek. Simulation of adaptive duty cycling in solar powered environmental monitoring systems. In *Proceedings of the 2014 Canadian Conference on Electrical and Computer Engineering (CCECE)*, May 2014.

[73] EMEND Project. Ecosystem-based research into boreal forest management. `http://www.emendproject.org/`, 2008.

[74] V. Raghunathan, S. Ganeriwal, and M. Srivastava. Emerging techniques for long lived wireless sensor networks. *Communications Magazine, IEEE*, 44(4):108–114, April 2006.

[75] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M.B. Srivastava. Design considerations for solar energy harvesting wireless embedded systems. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 457–462, April 2005.

[76] Ibrahim Reda and Afshin Andreas. Solar position algorithm for solar radiation applications. Technical Report NREL/TP-560-34302, Golden, Colorado, January 2008. [Electronic Resource].

[77] Arturo Sanchez-Azofeifa, Julio Calvo, Mario Marcos do Espirito Santo, Geraldo W. Fernandes, Jeniffer Powers, and Mauricio Quesada. *Tropical Dry Forests in the Americas: Ecology, Conservation, and Management*, chapter Tropical Dry Forests in the Americas: The Tropi-Dry Endeavour. CRC Press, Boca Raton, FL, 2013.

[78] G Arturo Sánchez-Azofeifa, Cassidy Rankine, Mario Marcos de Espirito Santo, Rob Fatland, and Milton Garcia. Wireless sensor networks for environmental monitoring: Two case studies from tropical forests. In *Seventh IEEE International Conference on eScience*, pages 70 – 76, 2011.

[79] Bruno Scrosati. Challenge of portable power. *Nature*, 373(6515):557–558, February 1995.

[80] M. Sharp, D.O. Burgess, J.G. Cogley, M. Ecclestone, C. Labine, and G.J. Wolken. Extreme melt on Canada's Arctic ice caps in the 21st century. *Geophysical Research Letters*, 38:1–5, 2011. [Electronic Resource].

[81] Y. Shi, M. Mizumoto, N. Yubazaki, and M. Otani. A learning algorithm for tuning fuzzy rules based on the gradient descent method. In *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems*, pages 55–61, 1996.

[82] F. Simjee and P. H. Chou. Everlast: Long-life, supercapacitor -operated wireless sensor node. In *Proceedings of the 2006 International Symposium on Low Power Electronics and Design*, 2006.

[83] T. Simunic, L. Benini, A. Acquaviva, P. Glynn, and G. De Micheli. Dynamic voltage scaling and power management for portable systems. In *Design Automation Conference, 2001. Proceedings*, pages 524–529, 2001.

[84] Amit Sinha and Anantha Chandrakasan. Dynamic power management in wireless sensor networks. *IEEE Des. Test*, 18(2):62–74, March 2001.

[85] P. Stanley-Marbell and D. Marculescu. An 0.9 x 1.2", low power, energy-harvesting system with custom multi-channel communication interface. In *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*, pages 1–6, April 2007.

[86] S. Sudevalayam and P. Kulkarni. Energy harvesting sensor nodes: Survey and implications. *Communications Surveys Tutorials, IEEE*, 13(3):443–461, Third 2011.

[87] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-15(1):116–132, Jan 1985.

[88] J. Taneja, Jaein Jeong, and D. Culler. Design, modeling, and capacity planning for micro-solar power sensor networks. In *Information Processing in Sensor Networks, 2008. IPSN '08. International Conference on*, pages 407–418, April 2008.

[89] Texas Instruments Incorporated, Dallas, Texas. *High Efficiency Single Inductor Buck-Boost Converter with 1-A Switches*, March 2012.

[90] Texas Instruments Incorporated, Dallas, Texas. *Ultra Low Power Boost Converter with Battery Management for Energy Harvester Applications*, September 2012.

[91] Tropi-Dry. Tropidry: Building bridges between research and conservation. `http://tropi-dry.eas.ualberta.ca/`, 2012.

[92] Gian-Reto Walther, Eric Post, Peter Convey, Annette Manzel, Camille Parmesan, Trevor J. C. Beebee, Jean-Marc Fromentin, Ove Hoegh-Guldberg, and Franz Bairlein. Ecological responses to recent climate change. *Nature*, 416:389 –395, March 2002.

[93] AG. Watts, P. Musilek, and L. Wyard-Scott. Managing the energy-for-data exchange in remote monitoring systems. In *Electrical Power Energy Conference (EPEC), 2013 IEEE*, pages 1–4, Aug 2013.

[94] AG. Watts, P. Musilek, and L. Wyard-Scott. Optimizing fuzzy control of energy harvesting remote monitoring systems. In *IFSA World Congress*

and NAFIPS Annual Meeting (IFSA/NAFIPS), 2013 Joint, pages 661–666, June 2013.

[95] Asher G. Watts, Michal Prauzek, P. Musilek, Emil Pelikán, and Arturo Sanchez-Azofeifa. Fuzzy power management for environmental monitoring systems in tropical regions. In *Neural Networks (IJCNN), The 2014 International Joint Conference on*, July 2014.

[96] A.S. Weddell, Geoff V. Merrett, T.J. Kazmierski, and B.M. Al-Hashimi. Accurate supercapacitor modeling for energy harvesting wireless sensor nodes. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 58(12):911–915, Dec 2011.

[97] Lotfi A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

Appendix A

Controller Generation

A.1 Sample Fuzzy Controller Code

```
%Notes:
%   eb          energy buffer
%   db          data buffer
%   m           measure & transmit
%   s           store
%
%   uod         universe of discourse
%   mf          membership function

% Must use 101, not 100,
% as last set number, else tmf does not work properly.

% EXAMPLE MEMEBERSHIP FUNCTIONS:
%DB_PARAMETER
MF_SET(:,:,1) = [   [0 0 25];
                    [0 25 50];
                    [25 50 75];
                    [50 75 101];
                    [75 100 101]    ];

%EB_PARAMETER
MF_SET(:,:,2) = [   [0 0 20];
                    [0 20 40];
                    [20 40 70];
                    [40 70 90];
                    [70 100 101]    ];

```

```
%M_PARAMETER
MF_SET(:,:,3) = [   [0 0 25];
                    [0 25 50];
                    [25 50 75];
                    [50 75 101];
                    [75 100 101]    ];

%S_PARAMETER
MF_SET(:,:,4) = [   [0 0 25];
                    [0 25 50];
                    [25 50 75];
                    [50 75 101];
                    [75 100 101]    ];

DB_PARAMETER    = MF_SET(:,:,1);
EB_PARAMETER    = MF_SET(:,:,2);
M_PARAMETER     = MF_SET(:,:,3);
S_PARAMETER     = MF_SET(:,:,4);

% Build uod for each dimension (2IP, 2OP)
eb_uod  = 1:1:100;  % 1 - 100%
db_uod  = 1:1:100;  % 1 - 100%
m_uod   = 1:1:100;  % 1 - 100%
s_uod   = 1:1:100;  % 1 - 100%

%Build all input Memebership functions
for i = 1:1:100
    EB_MF(1,i) = aTRI_MEM(i,EB_PARAMETER(1,:)); %OFF
    EB_MF(2,i) = aTRI_MEM(i,EB_PARAMETER(2,:)); %LOW
    EB_MF(3,i) = aTRI_MEM(i,EB_PARAMETER(3,:)); %MED-LOW
    EB_MF(4,i) = aTRI_MEM(i,EB_PARAMETER(4,:)); %MED-HIGH
    EB_MF(5,i) = aTRI_MEM(i,EB_PARAMETER(5,:)); %HIGH
end

for i = 1:1:100
    DB_MF(1,i) = aTRI_MEM(i,DB_PARAMETER(1,:)); %EMPTY
    DB_MF(2,i) = aTRI_MEM(i,DB_PARAMETER(2,:)); %LOW
    DB_MF(3,i) = aTRI_MEM(i,DB_PARAMETER(3,:)); %MED
    DB_MF(4,i) = aTRI_MEM(i,DB_PARAMETER(4,:)); %HIGH
    DB_MF(5,i) = aTRI_MEM(i,DB_PARAMETER(5,:)); %FULL
end

%Build output membership functions - Can substitue for ...
    weights!
for i = 1:1:100
    M_MF(1,i) = aTRI_MEM(i,M_PARAMETER(1,:)); %OFF
    M_MF(2,i) = aTRI_MEM(i,M_PARAMETER(2,:)); %LOW
    M_MF(3,i) = aTRI_MEM(i,M_PARAMETER(3,:)); %MED-LOW
    M_MF(4,i) = aTRI_MEM(i,M_PARAMETER(4,:)); %MED-HIGH
    M_MF(5,i) = aTRI_MEM(i,M_PARAMETER(5,:)); %HIGH
end

```

```
for i = 1:1:100
    S_MF(1,i) = aTRI_MEM(i,S_PARAMETER(1,:)); %NONE
    S_MF(2,i) = aTRI_MEM(i,S_PARAMETER(2,:)); %25
    S_MF(3,i) = aTRI_MEM(i,S_PARAMETER(3,:)); %50
    S_MF(4,i) = aTRI_MEM(i,S_PARAMETER(4,:)); %75
    S_MF(5,i) = aTRI_MEM(i,S_PARAMETER(5,:)); %100
end

%OUTPUT WEIGHTS

%RULES
RULE = zeros(25,100);
M_SURFACE_con = zeros(100,100);
S_SURFACE_con = zeros(100,100);

%MEASUREMENT AND TRANSMISSION SURFACE
for db_i = 1:1:100
    for eb_i = 1:1:100
    %RULE = (INPUT * INPUT = ANT.) * (OUTPUT = CONS.)
    RULE(1,:) = DB_MF(1,db_i) * EB_MF(1,eb_i) * M_MF(1,:);
    RULE(2,:) = DB_MF(1,db_i) * EB_MF(2,eb_i) * M_MF(2,:);
    RULE(3,:) = DB_MF(1,db_i) * EB_MF(3,eb_i) * M_MF(3,:);
    RULE(4,:) = DB_MF(1,db_i) * EB_MF(4,eb_i) * M_MF(5,:);
    RULE(5,:) = DB_MF(1,db_i) * EB_MF(5,eb_i) * M_MF(5,:);

    RULE(6,:) = DB_MF(2,db_i) * EB_MF(1,eb_i) * M_MF(1,:);
    RULE(7,:) = DB_MF(2,db_i) * EB_MF(2,eb_i) * M_MF(2,:);
    RULE(8,:) = DB_MF(2,db_i) * EB_MF(3,eb_i) * M_MF(3,:);
    RULE(9,:) = DB_MF(2,db_i) * EB_MF(4,eb_i) * M_MF(5,:);
    RULE(10,:)= DB_MF(2,db_i) * EB_MF(5,eb_i) * M_MF(5,:);

    RULE(11,:)= DB_MF(3,db_i) * EB_MF(1,eb_i) * M_MF(1,:);
    RULE(12,:)= DB_MF(3,db_i) * EB_MF(2,eb_i) * M_MF(2,:);
    RULE(13,:)= DB_MF(3,db_i) * EB_MF(3,eb_i) * M_MF(3,:);
    RULE(14,:)= DB_MF(3,db_i) * EB_MF(4,eb_i) * M_MF(5,:);
    RULE(15,:)= DB_MF(3,db_i) * EB_MF(5,eb_i) * M_MF(5,:);

    RULE(16,:)= DB_MF(4,db_i) * EB_MF(1,eb_i) * M_MF(1,:);
    RULE(17,:)= DB_MF(4,db_i) * EB_MF(2,eb_i) * M_MF(2,:);
    RULE(18,:)= DB_MF(4,db_i) * EB_MF(3,eb_i) * M_MF(3,:);
    RULE(19,:)= DB_MF(4,db_i) * EB_MF(4,eb_i) * M_MF(4,:);
    RULE(20,:)= DB_MF(4,db_i) * EB_MF(5,eb_i) * M_MF(5,:);

    RULE(21,:)= DB_MF(5,db_i) * EB_MF(1,eb_i) * M_MF(1,:);
    RULE(22,:)= DB_MF(5,db_i) * EB_MF(2,eb_i) * M_MF(1,:);
    RULE(23,:)= DB_MF(5,db_i) * EB_MF(3,eb_i) * M_MF(1,:);
    RULE(24,:)= DB_MF(5,db_i) * EB_MF(4,eb_i) * M_MF(2,:);
    RULE(25,:)= DB_MF(5,db_i) * EB_MF(5,eb_i) * M_MF(3,:);

    M_SURFACE_con(db_i,eb_i) = ...
        defuzz(db_uod,max(RULE),'centroid');
    end
```

```
end


%STORAGE SURFACE
for db_i = 1:1:100
    for eb_i = 1:1:100
    %RULE = (INPUT * INPUT = ANT.) * (OUTPUT = CONS.)
    RULE(1,:) = DB_MF(1,db_i) * EB_MF(1,eb_i) * S_MF(1,:);
    RULE(2,:) = DB_MF(1,db_i) * EB_MF(2,eb_i) * S_MF(1,:);
    RULE(3,:) = DB_MF(1,db_i) * EB_MF(3,eb_i) * S_MF(2,:);
    RULE(4,:) = DB_MF(1,db_i) * EB_MF(4,eb_i) * S_MF(4,:);
    RULE(5,:) = DB_MF(1,db_i) * EB_MF(5,eb_i) * S_MF(5,:);

    RULE(6,:) = DB_MF(2,db_i) * EB_MF(1,eb_i) * S_MF(1,:);
    RULE(7,:) = DB_MF(2,db_i) * EB_MF(2,eb_i) * S_MF(1,:);
    RULE(8,:) = DB_MF(2,db_i) * EB_MF(3,eb_i) * S_MF(3,:);
    RULE(9,:) = DB_MF(2,db_i) * EB_MF(4,eb_i) * S_MF(4,:);
    RULE(10,:)= DB_MF(2,db_i) * EB_MF(5,eb_i) * S_MF(5,:);

    RULE(11,:)= DB_MF(3,db_i) * EB_MF(1,eb_i) * S_MF(1,:);
    RULE(12,:)= DB_MF(3,db_i) * EB_MF(2,eb_i) * S_MF(2,:);
    RULE(13,:)= DB_MF(3,db_i) * EB_MF(3,eb_i) * S_MF(3,:);
    RULE(14,:)= DB_MF(3,db_i) * EB_MF(4,eb_i) * S_MF(5,:);
    RULE(15,:)= DB_MF(3,db_i) * EB_MF(5,eb_i) * S_MF(5,:);

    RULE(16,:)= DB_MF(4,db_i) * EB_MF(1,eb_i) * S_MF(2,:);
    RULE(17,:)= DB_MF(4,db_i) * EB_MF(2,eb_i) * S_MF(3,:);
    RULE(18,:)= DB_MF(4,db_i) * EB_MF(3,eb_i) * S_MF(4,:);
    RULE(19,:)= DB_MF(4,db_i) * EB_MF(4,eb_i) * S_MF(5,:);
    RULE(20,:)= DB_MF(4,db_i) * EB_MF(5,eb_i) * S_MF(5,:);

    RULE(21,:)= DB_MF(5,db_i) * EB_MF(1,eb_i) * S_MF(3,:);
    RULE(22,:)= DB_MF(5,db_i) * EB_MF(2,eb_i) * S_MF(4,:);
    RULE(23,:)= DB_MF(5,db_i) * EB_MF(3,eb_i) * S_MF(5,:);
    RULE(24,:)= DB_MF(5,db_i) * EB_MF(4,eb_i) * S_MF(5,:);
    RULE(25,:)= DB_MF(5,db_i) * EB_MF(5,eb_i) * S_MF(5,:);

    S_SURFACE_con(db_i,eb_i) = ...
        defuzz(eb_uod,max(RULE),'centroid');
    end
end

% AT THIS POINT THE CONTROL SURFACES ARE CONTINUOUS ...
    AND RELATIVE.
% THEY MUST BE DISCRETIZED BASED UPON THE HARDWARE'S ...
    CAPABILITIES.
```

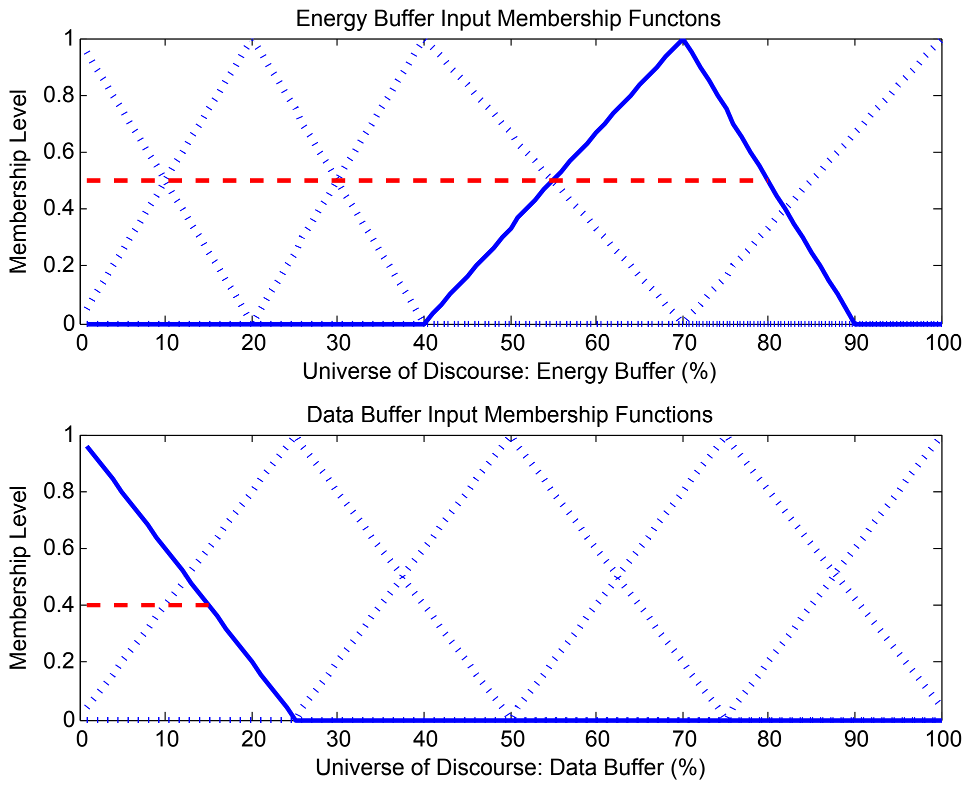


Figure A.1: Above are the two input dimensions that make up the antecedent for the simple fuzzy controller. The fuzzy sets that are involved in "Rule 4" from the "Measurement and Transmission Surface" from Appendix A.1 are shown in solid blue, while sets that are not included in "Rule 4" are shown in dotted lines. The red dashed line shows the level of activation of the fuzzy sets for the inputs for 15% for the data buffer and 80% for the energy buffer. Their values are 0.4000 and 0.5000 respectively.

A.2 Sample Fuzzy Controller Computations

Consider a point in the input space where the data buffer level is 15% and the energy buffer level is 80%. The sample script used to generate the continuous fuzzy control surface, shown in Appendix A.1, is executed. The fuzzy sets that cover the two input dimensions and their activation levels for "Rule 4" of the "Measurement and Transmission Surface" from the two inputs are shown in Figure A.1. The results from the fuzzy inputs sets are related using the product operation to form the antecedent. This is shown in Equation A.1.

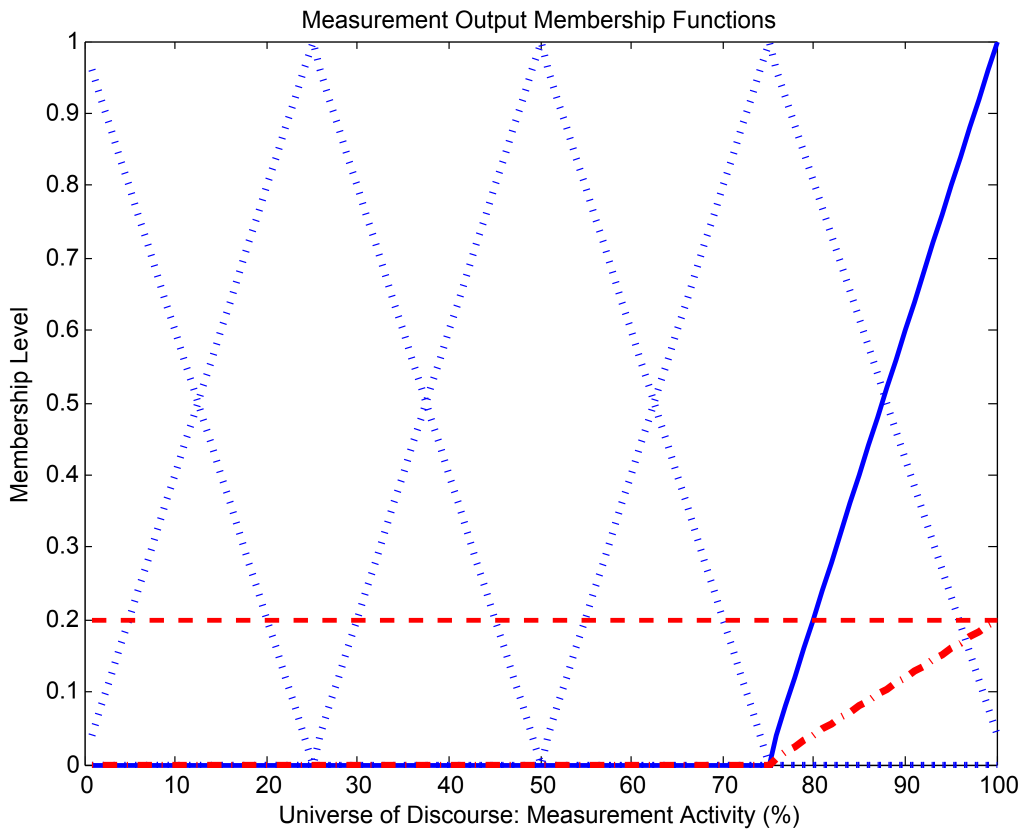


Figure A.2: The fuzzy sets covering the "Measurement and Transmission" output space are shown above. Fuzzy sets that are not involved in the consequent of "Rule 4" have dotted blue lines, while the fuzzy set that is used, is in solid blue. When the antecedent of 0.2000, shown as a dashed red line, is applied to the consequent, it is scaled down to the red dash-dot line in the bottom right.

$$\begin{aligned} &IF\ (80\ \ in\ \ energyBuffer)\ AND\ (15\ \ in\ \ dataBuffer) \\ &\qquad THEN\ (MeasureAndTransmit\ \ is\ \ high) \end{aligned} \tag{A.1}$$

The two inputs, the fuzzy sets labelled "EMPTY" for the data buffer and "MED-HIGH" for the energy buffer, are activated to membership levels of 0.4000 and 0.5000 respectively, as shown in Figure A.1. This creates an an-

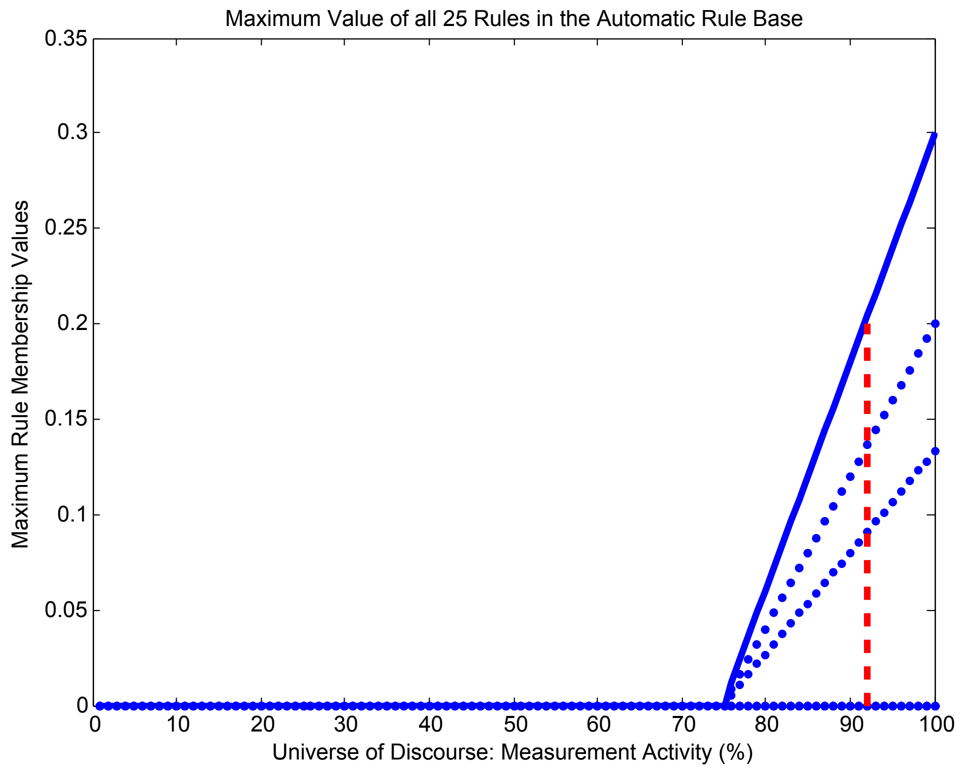


Figure A.3: The maximum value of all rules used in the computation of the output is shown in solid blue. Rules that were computed but below the maximum, and thus not used, are shown in blue dots. The scaled consequent from Figure A.2 is one of the rules which is below the maximum. The dashed red line is the computed centroid value of 92%. This is the activity level output mapped to the two input points.

tecedent value of 0.2000 by the product operation. This is then applied to the consequent, once again by the product rule, as shown in Figure A.2. This produces the output fuzzy set for "Rule 4".

Once all rules are computed in this fashion, then the maximum operator is used to with the "centroid" operation in the defuzzification process. This creates an output value, shown in Figure A.3, that would be used in the state-action mapping of the fuzzy controller. From this set of operations, the inputs, a data buffer level of 15% and an energy buffer level of 80%, are mapped to a 92% activity level for the measurement and transmission output dimension. The number of transmissions that are actually executed is dependent on the hardware and the discretization process. For example, if the MAX_{OP} of the hardware was 4 measurements per hour, an activity level of 92% would likely be mapped to 4 measurements per hour.

Appendix B

Simulation Requests

The simulation work used in this thesis is available by request from the research group. Providing a copy of the Simulink simulation, supporting Matlab scripts, and environmental data within the thesis is impractical. The simulation has continued to evolve past what has been presented here, but is stored as several versions corresponding to each publication. By storing it like this, the most current version will be available by request along with other previous versions.

Requests for information should be made to the principle investigator, who is included in all the following publications: [94], [93], [72], [95], [71], and [70].

Appendix C

Quantizing Failure

The *consecutive failure score* comes from [72]. It is used to measure the output data set of the simulation and determine how fragmented it is. The idea is that consecutive failures are less desirable than intermittent short-term failures. Intermittent short-term failures are like operating at a lower duty cycle, they are less than ideal, but the environment is still monitored to some degree. Consecutive failures are a void of knowledge; there is a large chunk of data missing that is harder to fill, particularly if it is periodic between years. Intermittent failures could be random, and a few of them is hard to avoid. If there were large blocks of repeated failures in the output of the simulator, this typically indicated a component sizing problem or an oversight in the energy management strategy.

To compute the *consecutive failure score*, the simulation output is used to determine which measurement periods did not produce data. Each period where data is collected, and the device did not fail, is assigned a value of zero. The first failure to occur is assigned a value of one. Any subsequent failures, after this first failure, are assigned an incremental value until a point is found where data is collected; the process then repeats. Once this is completed for the entire year, all values are squared. The sum of these squared values over the whole year is the *consecutive failure score.* This heavily penalizes long, continuous periods of failure, but is moderate to intermittent short-term failures.

For the first example, consider a measurement period of one week. Assume that one measurement is taken per day, and that no measurement constitutes a failure. In the following sequence of numbers, 1 shall be considered a failure, while 0 shall be the successful collection of data.

$$[0, 1, 0, 1, 0, 1, 0]$$

The device fails every other day. There are 3 failures, none of which are consecutive, so the *consecutive failure score* is 3.

$$0 + 1 + 0 + 1 + 0 + 1 + 0 = 3$$

For the second example, the first example is repeated, but with failures group together in the center of the week.

$$[0, 0, 1, 1, 1, 0, 0]$$

Consecutive failures are incremented based upon previous failures; this creates the following series.

$$[0, 0, 1, 2, 3, 0, 0]$$

This results in a *consecutive failure score* of 14.

$$0 + 0 + 1^2 + 2^2 + 3^2 + 0 + 0 = 14$$

This value, when used with the total number of failures, helps compare the performance of two energy management strategies. It mentioned in Chapter 7.