Diss. ETH No. 25456

Design and Specification of Batteryless Sensing Systems

A thesis submitted to attain the degree of

Doctor of Sciences of ETH Zurich
(Dr. sc. ETH Zurich)

presented by
ANDRES GOMEZ
M.Sc. ALaRI, USI

born on 25.06.1986
citizen of
United States of America
República de Colombia

accepted on the recommendation of
Prof. Dr. Luca Benini, examiner
Prof. Dr. Lothar Thiele, co-examiner
Prof. Dr. Geoff Merrett, co-examiner

2018

Institut für Technische Informatik und Kommunikationsnetze
Computer Engineering and Networks Laboratory

TIK-SCHRIFTENREIHE NR. 176

Andres Gomez

Design and Specification of Batteryless Sensing Systems

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

A dissertation submitted to
ETH Zurich
for the degree of Doctor of Sciences

DISS. ETH NO. 25456

Prof. Dr. Luca Benini, examiner
Prof. Dr. Lothar Thiele, co-examiner
Prof. Dr. Geoff Merrett, co-examiner
Examination date: September 21, 2018

To my heavenly father, whose guidance enlightens my path every day.
Para mi padre terrenal, cuyo amor infinito e incondicional lo hizo todo posible.

Abstract

Over the past few decades, batteries played a central role in the design of wireless sensing systems. Large storage devices provide a stable energy supply, ensuring long system lifetimes even when energy demands are highly variable. Current trends point towards the deployment of billions of interconnected sensing devices gathering information from their surrounding, also known as the Internet-of-Things (IoT). While energy flow is absolutely necessary for IoT devices to function, large energy storage capacity is not. Minimized energy provisioning will make the IoT more economically viable and environmentally friendly. It also restricts the use of high-power peripherals and introduces intermittence, raising new challenges in application development.

In this dissertation, we address how rich data sensing systems can be designed for robust and efficient operation with a minimized transducer and storage element. Batteryless systems are particularly suited for energy-driven sensing applications, where energy and information are simultaneously available in the environment. To this end, we design a general-purpose power subsystem compatible with transducers whose output cannot directly sustain system operation. Then, we introduce a data aggregation scheme which can drastically reduce average transactional costs to power hungry peripherals. Finally, we present a specification model for intermittence-tolerant applications. With it, developers can write arbitrarily long (single-core) sensing applications and automatically calculate the optimal energy burst partitioning with optimized data state retention and restoration. Specifically, the following contributions are presented in this dissertation.

- We design an Energy Management Unit (EMU), which acts as a broker between a low-power transducer

and a high-power application circuit while keeping their operation independent from each other. The EMU uses an application-specific capacitor to buffer energy for a single execution cycle or burst. This absorbs the transducer's I-V variability and guarantees atomic execution. Compared to existing solutions, this enables energy-proportional operation down to tens of microwatts of harvested power at wide voltage ranges.

- We develop the Non-Volatile Memory Hierarchy (NVMH) for long-term data logging of rich data sensors. Current approaches to power-hungry peripherals greedily access them regardless of their initialization cost. By temporarily storing multiple data samples in an energy-efficient nonvolatile memory, NVMH can significantly reduce average storage costs of large density memories such as SD cards.

- We propose a data-flow-based specification model for batteryless sensing applications. By composing multiple atomic kernels with explicit data dependencies, an entire application can be optimally partitioned into bursts and minimize energy cost, given a capacity bound. As opposed to existing solutions, this method provides an application with energy guarantees required for atomic tasks, like sending radio packets, and automatically saves/restores the minimal amount of non-volatile data for each execution unit.

From the thorough experimental evaluation, it is concluded that batteryless applications based on architectures with tunable energy guarantees can operate robustly and efficiently even in low and intermittent energy harvesting scenarios.

Sommario

Negli ultimi decenni, le batterie hanno giocato un ruolo centrale nella progettazione dei sistemi sensoriali senza fili. Un dispositivo di stoccaggio energetico di grandi dimensioni può fornire un'alimentazione constante, garantendo un lungo funzionamento del sistema anche quando le sue necessità energetiche sono molto variabili. Le attuali tendenze indicano una diffusione di miliardi di sistemi sensoriali interconnessi che raccolgono informazioni dall'ambiente circostante, noti anche come Internet of Things (IoT). Sebbene un constante flusso di energia sia assolutamente necessario per il corretto funzionamento dei dispositivi IoT, la disponibitilità di una grande capacità di immagazzinamento energetico non lo è. Un approvvigionamento energetico ottimizzato renderebbe l'IoT più redditizio e più ecologico. Inoltre, tale approccio limiterebbe l'uso di periferiche ad alta potenza ed introdurrebbe un funzionamento intermittente, ponendo nuove sfide nello sviluppo applicativo.

In questa dissertazione affrontiamo come i sistemi sensoriali possano essere progettati per il funzionamento robusto ed efficiente minimizzando sorgente e stoccaggio energetico. I sistemi senza batteria sono particolarmente adatti per applicazioni di monitoraggio in cui l'energia e l'informazione sono disponibili simultaneamente nell'ambiente. A tal fine, presentiamo un sottosistema generico per la generazione di potenza, compatibile con trasduttori che altrimenti non potrebbero garantire il corretto funzionamento nel tempo. Inoltre, nel presente lavoro viene introdotto uno schema di aggregazione dei dati che può ridurre drasticamente i costi transazionali medi per le periferiche ad alto consumo energetico. Infine, presentiamo anche un modello di specifiche per applicazioni con tolleranza all'intermittenza. Grazie ad esso, gli sviluppatori possono scrivere applicazioni sequen-

ziali di lunghezza arbitraria e calcolare automaticamente la ripartizione ottimale degli *energy bursts* (pacchetti di energia) al fine di minimizzare il costo energetico per la conservazione dei dati in memoria. In particolare, i seguenti contributi sono presentati in questa tesi.

- La progetazione di una *Energy Management Unit (EMU)* (Unità di Gestione Energetica), che funge da intermediario tra un trasduttore a bassa potenza ed un circuito applicativo ad alta potenza, mantenendo il loro funzionamento indipendente. La EMU utilizza un condensatore, dimensionato in base all'applicazione, per immagazzinare l'energia necessaria per un singolo ciclo di esecuzione, o *energy burst*. Questa EMU assorbe la variabilità I-V del trasduttore e garantendo l'esecuzione atomica dell'applicazione. In confronto alle soluzioni esistenti, ciò consente un funzionamento lineare rispetto alla potenza raccolta fino a decine di microwatt con un'ampia gamma di tensioni.
- Lo sviluppo di una *Non-Volatile Memory Hierarchy (NVMH)* (Gerarchia delle Memorie Non Volatili) per il salvataggio a lungo termine delle informazioni provenienti da sensori che producono un'elevata mole di dati. I tradizionali approcci nella gestione delle periferiche ad alta potenza non tengono conto del loro costo di inizializzazione. Grazie alla conservazione temporanea di più campioni di dati in una memoria non volatile ad alta efficienza energetica, la nostra NVMH è in grado di ridurre significativamente i costi medi di memorizzazione per le memorie a grande densità, come le schede SD.

- Un modello di specifiche basato sul flusso di dati per applicazioni sensoriali senza batterie. Attraverso il raggruppamento di più kernel atomici con esplicite dipendenze di dato, un'applicazione può essere suddivisa in diversi *energy bursts*. Il dimensonamento degli stessi può essere ottimizzato al fine di minimizzare i costi energetici dell'applicazione, rispettando la limitata capacità di immagazzinamento energetico del sistema.

Questo è reso possibile dal salvataggio e dal ripristino automatico della quantità minima dei dati non volatili per ogni unità di esecuzione. A differenza delle soluzioni esistenti, questo metodo fornisce ad un'applicazione le garanzie energetiche necessarie per l'esecuzione atomica dei kernel, come l'invio di pacchetti radio.

Attraverso un'attenta valutazione sperimentale, si è concluso che le applicazioni senza batteria basate su architetture dove il livello di garanzia energetica è configurabile, possono operare in modo robusto ed efficiente anche in scenari in cui la sorgente energetica risulta essere limitata ed intermittente.

Acknowledgments

As my long and joyful student life comes to a close, I feel an immense sense of gratitude towards all who played a role in it. Prof. Benini, Prof. Thiele, I am very grateful for the opportunity to be a part of not one, but two research groups. It was a privilege that I never took for granted, and a responsibility that I always took seriously. Besides providing me with a thought-provoking research problem, you both listened to my (sometimes repetitive) ideas, identified their best parts, and greatly improved them. I would also like to thank Prof. Merrett for agreeing to review my thesis and giving insightful feedback.

My entry into academic research was heavily influenced by my master thesis advisors: Lars and Pratyush. Under their combined supervision, I learned that patience and diligence go hand in hand when pursuing ideas of scientific interest and scholarly value to the highest standards. During my PhD, I have strived to reproduce their attention to detail, impeccable efficacy and industrious efficiency. It was their willingness to supervise me that first made my dream of doing research at ETH possible.

Ai miei cari colleghi UniBo ringrazio per una collaborazione indimenticabile. Dall'inizio mi sono sentito accolto nel gruppo ed è stato lì, a Bologna, dove ho trovato la mia indipendenza professionale. Ad Andrea B., Andrea M., Davide, Christian, Filippo, Alessandro e Mohammad vi ringrazio per avermi insegnato tanto su microelectronica, architettura, modellamento di sistemi multiprocessore, e sistemi indossabili. Francesco, anche se non abbiamo laborato molto insieme, sono *molto grato* per il nostro lavoro con machine learning. Michele, grazie per insegnarmi su microcontrollori, sensori e trasduttori. Senza queste basi, la mia ricerca sarebbe stato impossibile. Daniele, a te ringrazio per la tua introduzione ai temi di computer vision e nano/femto UAV's. Mi hai fatto ridere sempre con i tuoi scherzi e grazie a te conosco meglio il *sermo vulgaris*. Grazie mille a tutti!

Pascal, dein neugieriger Geist und experimentelles Know-how waren sehr wichtig für unsere industrielle Zusammenarbeit und Studienprojekte. Du hast immer die Arbeit verbessert und interessanter gemacht. Lukas C., obwohl wir für eine relative kurze Zeit unseren Büro geteilt haben, es war eine Freude. Unsere Diskussionen über Transient, System Entwurf, Kapitalismus und Akademischeverordnungen waren immer spannend. Danke euch beiden für alles!

Lukas S., thank you for choosing to work on the transient project. Were it not for your dexterity with electronic circuits, the results in this dissertation would not have been as precise or obtained as quickly as they were. It was always a pleasure to discuss with you topics as varied as transient computing, quantum mechanics, influential architects, fire-breathing techniques and entrances to Dutch bars. Andreas, thank you for your work on Ladybirds. Though it wasn't obvious at the time, it ended up being an important tool for my research. Georgia, I am grateful for the group traditions you created, your company during working Sundays, and for sharing my appreciation for Colombia. Pengcheng, our discussions were always technically rigorous and gave me a different point of view. I hope I can someday repay the wonderful trip you organized to China, it was an amazing experience. To my more theoretically-minded colleagues, Rehan and Stefan, I greatly appreciate our joint work with mathematical formalisms that pushed the boundaries of my comprehension.

In every research group, there is a lot of essential work done behind the scenes. I would like to express my gratitude to Alfonso Blanco, Beat Muheim and Frank Gürkaynak for all their support with PCB design and VLSI lectures; Beat Futterknecht, Friederike Brütsch, Susann Arreghini, and Christine Haller for all their help with administrative tasks; Thomas Steingruber, Benny Gächter, Stefan Schindler, and Christoph Wicki for the smooth-running infrastructure; and Hansjörg Gisler for an organized and well-maintained laboratory. During my PhD I had the opportunity to co-supervise over 30 bachelor/master students. It was thanks to their commitment and enthusiasm that we explored so many different ideas. In particular, I would like to thank Thomas Schalch, Seraina Schweizer, Alexander Sage, Praveenth Sanmagujarah, and Patroklos Anagnostou. Their work, or some iteration thereof, greatly contributed to the topics presented in this dissertation.

Durante un doctorado es importante tener personas que le recuerden la importancia de la vida más allá del laboratorio. Mi vida en el edificio ETZ no hubiera sido la misma sin Laura, El Hamiltoniano y Paulina. Mi vida fuera del edificio ETZ tampoco hubiera sido la misma sin Jeanine, Almu, Crespo, las Vanesas y Salsina. A mis hermanos argentinos, Pablo y Mariano, les agradezco el proceso de latinización cuyas raíces van más allá de un asensor en Lugano. Aprecio mucho haber compartido con todos ustedes cumpleaños, fiestas, paseos, proyectos de construcción casera, matrimonios, embarazos y nacimientos.

Finalmente, la base de todo logro en mi vida se debe a mi núcleo familiar. A Diego y Patricia les debo no sólo la oportunidad de reencarnar en esta existencia, sino también el amor parternal que provee todo al hijo. Gracias Isabella por la guía y figura materna que ha enriquecido mi vida durante décadas. Gracias Ivanna por tu dulce esencia que nos ha aportado más de una carcajada. Gracias a los Gómez por darme la base de una familia que apesar de la distancia, permanece unida bajo la enseñanza del Señor. Georgi, gracias por ser la compañera ideal que llena mi vida de paz, alegría y amor.

Gracias totales y sin baterías.

The work presented in this thesis has been partially supported by the Swiss National Science Foundation under grant number 157048: Transient Computing Systems. This support is gratefully acknowledged.

Contents

List of Figures

List of Tables

1

Introduction

Information is one of the key economic factors in many sectors including healthcare, energy, transportation, infrastructure, supply chain networks, among many others [JW14]. When this information can only be obtained in the physical realm, sensing systems need to be deployed to acquire, process, store and transmit relevant data. Designing these systems for long-term deployments is a difficult challenge given size, cost, reliability and maintenance restrictions. As the emerging Internet of Things (IoT) envisions information and communication technologies as services available 'anytime, anywhere', these challenges will only get more difficult [UHM11].

This vision of ubiquitous sensing systems has in part been enabled by the advances in microelectronics, material sciences and related fields. They have not only reduced the costs and energy consumption of digital devices but also allowed their miniaturization to $1\,mm^3$ scale [LBL+13]. These ultra-low-power systems are not only capable of ambient sensing and wireless transmission, but also efficient acquisition and parallel processing of complex biomedical signals [GHS+17]. Sensing systems have been successfully used in a wide range

application scenarios like healthcare [AE10], infrastructure monitoring [ELE06], precision agriculture [GVMNGPG14], surveillance [CMB15] and assisted living [WSV+08] among many others.

Among the most fundamental problems to solve when designing long-term deployments is how to supply sensing systems with electrical energy. For many decades, primary batteries have been a popular solution to supply wireless sensor and actuator systems [WHSC01]. These non-rechargeable batteries can have lifetimes spanning several years, in part because of their low current drain and very high energy density [Hug10]. Researchers have also developed techniques including dynamic power management [BCMS01, SC01] and low-power design [PBB98, CSB92] to reduce the average power and energy requirements for many applications. This has relaxed some design constraints and extended system lifetimes even further. Due to stringent reliability requirements, a few specific applications (e.g. pacemakers [MIR04]) will remain battery-powered for the foreseeable future. In many other application scenarios, however, energy harvesting has been successfully introduced to reduce costs and extend system lifetimes even further [RKH+05]. This trend has also been fueled by advances in secondary (rechargeable) battery technology [SPM15].

1.1 Energy harvesting-based systems

Figure 1.1 depicts the energy flow in all harvesting-based systems, starting from primary (ambient) energy into electrical energy to useful work. Ambient energy can be found in many different forms including solar, thermal and vibrational [SK11]. Even though virtually every environment has some primary energy to offer, all energy availability exhibits some temporal- and spatial-dependent behavior. Since transducers can only convert ambient energy to electrical energy, this dynamic behavior directly influences the system's energy budget. Besides the environment, transducer output depends

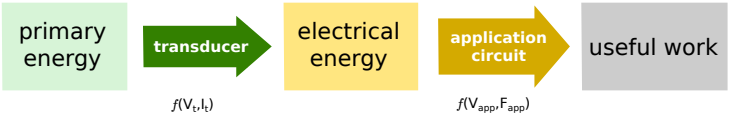


Figure 1.1: All harvesting-based systems convert primary energy into electrical energy, which can then be buffered and used by application circuits to do useful work. Transducer output is a function of its voltage and current, while that of the application depends on its voltage and operating frequency.

on other factors including its size (e.g. area, volume or weight), its I-V curve and its operating point [SPM15]. While sensing systems are typically designed to provide minimum service level guarantees, energy sources rarely behave in a consistent and predictable manner. Consequently, energy produced by transducer must be stored and conditioned before it can be used effectively [RKH+05]. How much energy a system can generate and store depends on many interconnected factors which can radically change resulting behavior [SGL+17]. There are different ways to cope with harvesting variability, among them service adaptation [MTBB10], power subsystem capacity planning [BSBT14], and multi-model energy harvesting [WMM+13].

1.1.1 Multi-harvesting

Harvesting-based systems which are exposed to a single source of primary energy can work efficiently with an appropriate transducer. In certain application domains, like wearable systems [MBS+16], they can be exposed to different types of primary energy. Given that primary energy availability is highly space and temporal dependent, coupling different sources of energy can increase the total harvested energy [DRF+15]. Multi-modal harvesting can be costly since it requires multiple transducers and larger form factors, but it can increase the energy reliability of systems with small harvesting and storage capabilities [BJJ18]. To do this efficiently, however, independent impedance matching and power point tracking systems must be implemented for each source [BC11]. Even

if multiple sources are being used, the dependence on the environment and its potentially long periods of energy unavailability might require further measures. Some systems increase overall storage capacity by combining different types of energy storage (e.g. fuel cells, batteries, and supercapacitors) and their different benefits [WMM+13].

1.1.2 Energy neutral operation

In the best-case scenario, a system is able to harvest and store enough energy to continuously fulfill application requirements. When this occurs, it is said that the system has energy neutral operation. Realistic deployments depend on high-power periodic source (i.e. the sun) to provide enough for operation. To design energy neutral systems, it is necessary to have a detailed energy generation [SPC12], storage [BAB13] and consumption model [HZK+06]. Among the difficulties in designing such systems is to provide closed-form expressions for minimum harvesting requirements depending on application parameters [BSB+13]. One of the simplest parameters to analyze is the excess time (t_{excess}), defined as the maximum theoretical time the system can operate without harvesting any energy (see Figure 1.2). If the time during which no energy is harvested ($t_{unavailable}$) can be bounded, then a necessary but not sufficient condition for energy neutrality is that $t_{excess} > t_{unavailable}$. Having enough energy storage capacity is crucial for energy neutrality. But many energy storage technologies have a low number of recharge cycles, which limits the overall system lifetime.

1.1.3 Energy storage

Batteries are often one of the most expensive components of a sensing system in terms of form-factor and cost considerations [ZGL13] as well as environmental impact [VdBVVM+06]. Though certain battery technologies do have high energy densities, their lifetimes can be restricted due to their limited number of charge cycles and high self-discharge rates [NHP03].

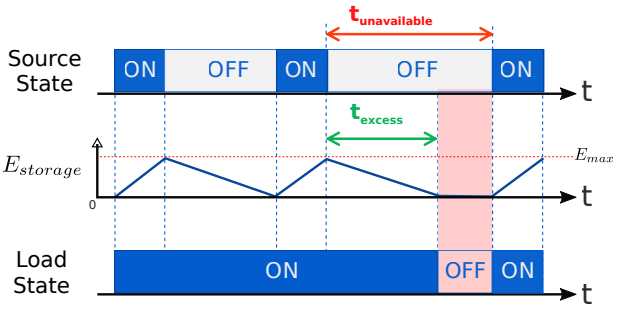


Figure 1.2: Operation of harvesting-based systems depends on the energy source, energy storage and load. For energy neutral operation, it is necessary (but not sufficient) that storage capacity be enough to supply the system when primary energy is not available ($t_{excess} > t_{unavailable}$).

In addition, certain technologies [ST 17] impose current peak limits that would hinder the use of power-hungry peripherals. Broadly speaking, storage capacity and system lifetime do not scale well with cost and energy efficiency. For this reason, current trends [HSS15a, BWM+15, WCK+14, GSM+16] point towards the optimization of the energy buffer's size according to application-specific parameters. In doing so devices become smaller, cheaper and more efficient but time-driven behavior is sacrificed for event-driven behavior. When energy requirements are small enough, electrostatic energy is an advantageous form of storage. Compared to electrochemical storage elements, small (SMD) capacitors have much higher power densities [Hug10]. In addition, these passive devices have small form factors and are made from cheap materials [Cra17]. Though their energy density is lower than batteries, they have very high reliability. In fact, some technologies like solid aluminum [KEM18] and tantalum [BBMP13] capacitors do not have aging effects within given temperature, voltage, and current limits. Referring back to Figure 1.2, one of the key parameters is again t_{excess} and designing it to ensure functional correctness, even with power-hungry peripherals. If a sensing system is able to run reliably and efficiently with these small

energy storage devices, they could have a potentially infinite useful lifetime.

1.1.4 Batteryless Systems

In an ideal scenario, energy and information come from the same source. When this is the case, harvesting-based systems with limited storage capacity can efficiently obtain information from an event using energy from the event itself. There are several application scenario where this principle of *energy-driven* operation can be applied. For example, cameras [NSF15] and UV sensors [HBD13] can be powered by light. In the case of movement, ferroelectret insoles can be used to power fitness tracking [RBL+17], vibrations inside a tire can be used to monitor tire pressure [LKWH07], and vibrations from earthquakes, wind and traffic can be harvested to monitor their effects on infrastructure [ELE06]. In all of these applications, the usefulness of information is tied to the energy availability. For example, if a camera takes a picture in the dark or a fitness tracker measures stillness, there is little to no information in the data. As soon as there is light or movement, the camera or the fitness tracker can both harvest energy and produce valuable information. Batteryless sensing systems can fulfill application requirements in a wide variety of scenarios, and they can do so with minimized cost, area, and energy overheads. However, there are many challenges which need to be solved for such systems to be efficient.

1.2 Challenges in batteryless operation

The design and optimization of batteryless sensing systems must take into account some general restrictions in order to execute applications in an efficient and reliable manner. The main challenges we have identified are the following:

1. *Transducers have low conversion efficiency and power density* Depending on the technology being used, transducers

have very different conversion efficiencies and power densities (see Figure 1.3). Thermoelectric generators (TEGs), for example, have conversion efficiencies of 0.1-3% and power densities ranging from 60 to 3000 $\mu W/cm^2$ in wearable ($\delta T < 10°K$) and industrial scenarios ($\delta T \geq 100°K$), respectively [SPM15]. Photovoltaic cells have conversion efficiencies ranging from 10 to 25%[BASM16] and power densities ranging from 10 to 1500 $\mu W/cm^2$ in indoor and outdoor environments, respectively [GB08]. Even transducers with high power densities can have an insufficient open circuit voltage and closed circuit current for power-hungry peripherals such as thermal cameras [FLI18]. To operate reliably under these conditions, the system needs to have separate voltage domains for the source and load. In this way, the power subsystem can become transducer-independent and allow impedance matching for maximum power transfer without affecting the load.

2. *Environmental energy is difficult to predict and can be adversarial*
   Micro-level energy harvesting, which is capable of supplying energy to low-power devices, has received considerable attention in recent years [BASM16]. One important characteristic of all ambient energy sources is their dynamic behavior. It is typically infeasible to predict the harvested power from sources like light, temperature and wind in a fine-grained and accurate manner due to the amount information it would require [KLCL16]. Wearable harvesters, for example, exhibit highly user-dependent behavior[MBS+16], which can vary significantly. Since the system designer has no control over ambient conditions, a conservative but realistic assumption is that the source could fail *at any moment*, even during application execution. This means that for any design to be reliable *and* efficient, it must be able to both adjust to time-dependent primary energy and tolerate its sudden disappearance.

3. *Loads have highly variable current, voltage and energy requirements*
   low-power sensing systems typically have components such as microcontrollers, memories, and peripherals (e.g. sensors and wireless transceivers). Microprocessors can be efficiently power scaled using voltage and frequency [BCMS01, PBB98] within a narrow range. However, external peripherals such as radios [Dig09] can have different voltage and current requirements, which are significantly higher than digital CMOS logic [Tex15]. In fact, these peripherals typically demand to be used in an atomic fashion, since their functionality depends on it. During packet transmission, for example, a radio has a precise power dissipation which must be satisfied for the packet to be sent successfully. These power and energy requirements can pose a challenge when supplying loads from transducers in dynamic environments. Unless the complete sensor node and its peripherals are fully scalable in terms of voltage and current, the load's operating point can be incompatible with the transducer's.

These combined challenges must all be addressed from the system design perspective. Conventional low-power design dictates that any harvesting-based system should, at all times, maximize the energy input and minimize the energy output. To maximize the harvested energy in dynamic scenarios, the source's maximum power point needs to be tracked. To minimize the load's energy, its operating point needs to be continuously adjusted to meet dynamic application requirements. These two criteria can differ significantly, especially when variable environmental conditions are taken into account. This variability, in turn, demands that the design use energy storage to provide some minimal energy guarantees, otherwise no application progress can be guaranteed. Designing an architecture that addresses these issues requires innovative methods that combine both hardware and software aspects. The main challenge is to design systems that can operate efficiently, have minimized

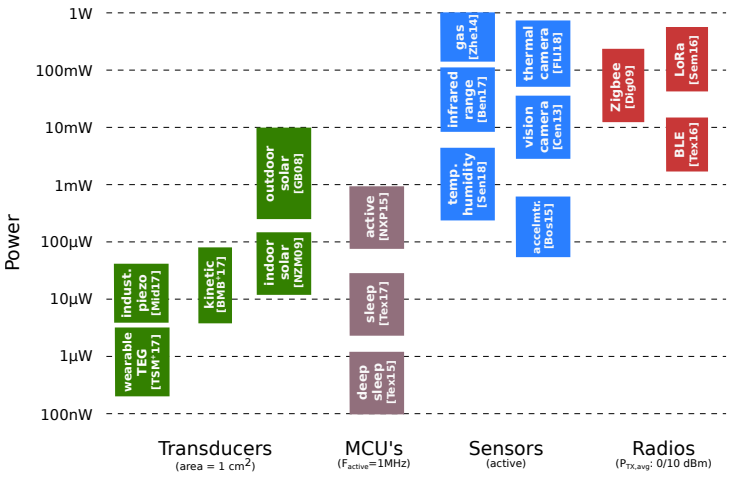


Figure 1.3: Conventional transducers, sensors, microcontrollers and transceivers exhibit a discrepancy between energy production and energy consumption capability.

storage and start-up costs given these conditions.

1.3 Thesis Outline and Contributions

This thesis proposes several building blocks for the design and specification of batteryless sensing systems. Two of the most important design parameters are the storage capacity and application energy cost. Though both are ideally minimized, there are fundamental trade-offs between these objectives. Figure 1.4 shows an overview of the proposed building blocks in the two-dimensional design space. In Chapter 2, we begin by introducing the notion of reliable execution in sense-process-store applications. We define reliable execution as being able to guarantee the completion of an activation cycle once it is begun. Since our energy sources can be adversarial (e.g. they can disappear at any moment), the energy storage device must be able to supply the energy necessary for one activation cycle.

Whether this cycle runs an entire application or just a part of it, the system performance and overhead will vary. In Chapter 3, we address aggregation as a means to reduce the average energy cost logging data. Since certain peripherals like high-density non-volatile memories can have high initialization costs, it is very costly duty-cycle them for single data items. By aggregating multiple measurements, the high initialization cost can be efficiently amortized among many data items at the expense of a larger required storage capacity. Our final building block for batteryless system design, discussed in Chapter 4, is partitioning. When processing tasks dominate application energy, they can be effectively distributed among many small activation cycles to reduce storage requirements. To do this efficiently, advanced state retention techniques need to be employed such that only the input and output data of these small processing tasks are transferred between volatile and non-volatile data.

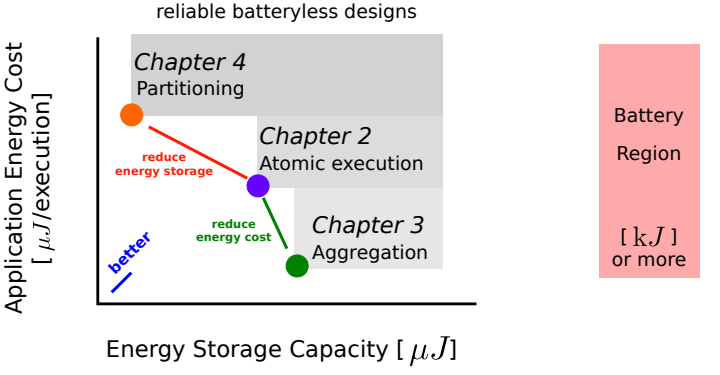


Figure 1.4: Overview of the building blocks presented in this thesis. Chapter 2 presents reliable task-based execution of sensing applications. Chapter 3 then includes aggregation to reduce the average energy cost of the same application. Chapter 4 introduces partitioning to reduce the energy storage required for task-based execution.

In the following, we present the main contributions of the thesis.

Chapter 2: Reliable execution in adverse environments
In this chapter, we present an energy-efficient Energy Management Unit (EMU) to supply generic loads when the transducer voltage and current are lower than required for sustained system operation. By slowly building up charge to a pre-defined energy level, the EMU can generate short energy bursts reliably, even under variable harvesting conditions. Furthermore, we propose a dynamic energy burst scaling (DEBS) technique to adjust the supply voltage of these bursts in accordance with the load's requirements. Using a simple digital interface, the load can dynamically configure the EMU to supply small bursts of energy at its optimal power point, independent from the harvester's operating point. Using two implementations of vision-based sensing systems, we demonstrate the general applicability of traditional low-power design to batteryless operation. Extensive theoretical and experimental data demonstrate the high energy efficiency of our approach, reaching up to 73.6% even when harvesting only 110 μW and supplying a load of 3.89 mW.

Chapter 3: Reducing energy costs through aggregation
Sensing systems using an Energy Management Unit (EMU) and Dynamic Energy Burst Scaling (DEBS) can generate small bursts of energy to execute sense-process-store applications in an energy efficient manner. However, energy efficiency alone does not encompass other important systems parameters such as application energy cost and storage capacity. In this chapter, we will demonstrate a fundamental trade-off between these last two parameters. Using a solar-powered image logging system, we propose and evaluate a Non-Volatile Memory Hierarchy (NVMH) to reduce the average energy cost per store image, at the expense of a larger energy storage device. Furthermore, we study the energy proportional aspects of EMU-based designs, both from the harvester and the load perspectives. Experimental data show that, regardless of its configuration, the EMU can supply energy bursts to a 43.4 mW load with energy efficiencies of up to 79.7% and can work with input power levels as low as 140 μW. When the system is configured to use both DEBS and NVMH, the total energy cost

of acquiring, processing and storing an image can be reduced by 77.8%, at the price of increasing the energy buffer size by 65%.

Chapter 4: Reducing energy storage through partitioning
Using EMU-based designs, entire applications can be easily executed within a single activation cycle in an efficient and reliable manner. However, this approach is not scalable as applications grow in data and energy consumption. If complex tasks can be partitioned among multiple energy bursts, a much smaller storage capacity will be required. At the same time, data consistency between bursts needs to be guaranteed. These state retention systems can incur significant energy costs as they transfer data between volatile and non-volatile domains. In this chapter, we study the optimization of complex processing applications following an energy burst execution scheme. Our method, *Julienning*, automates the energy optimization of batteryless applications based on a novel specification model. With this model, developers can define applications as a set of atomically-executed kernels with explicit data dependencies. By leveraging inter-kernel data dependencies, our optimizer can partition arbitrarily long applications into small execution cycles with minimized data overheads. We validate our methodology with transient cameras running CNN-based applications. Results show that compared to ad-hoc solutions, our method can reduce the required energy storage by 17.4× while incurring a negligible 0.08% energy overhead.

2

Reliable sensing in adverse environments

2.1 Introduction

Over the past decade, there has been a considerable research effort to reduce the energy consumption of electronic devices. While there has been considerable progress, the lifetime of battery-based devices remains the bottleneck in their development. Broadly speaking, the problem of how to supply low-power embedded systems with the energy they require in an efficient, low-cost, long-term, scalable, and self-sustainable manner has not yet been adequately solved. Over-provisioning with large energy harvesting and storage elements can be prohibitively expensive in many application scenarios such as wearable, distributed, miniaturized or "smart dust" systems. Fortunately, a purely harvesting-driven system can still meet application requirements in many of these scenarios.

Batteryless sensing systems acquire and store/transmit environmental data using environmental energy. However, they are supplied by volatile energy sources which can, at most,

directly power the system for only a limited amount of time. During this time, the energy harvesting rate might not be high enough to complete even one atomic task execution. We define an atomic task as a set of instructions whose functionality depends on its timely and uninterrupted execution. Common examples of atomic tasks include reading a sensor, transmitting a wireless packet, or transferring a block of data from a volatile to a non-volatile domain. Consequently, batteryless sensing systems need to be able to buffer at least the amount of energy needed to bridge the environmental power deficit and guarantee the completion of an atomic task. In this application domain, attempting to use a buffer that stores large amounts of energy inevitably leads to high losses due to self-discharge, idle currents and converter inefficiencies. Though energy can be stored in many forms: thermal, mechanical, electrostatic and electrochemical [Hug10], only the last two are commonly found in low-power sensing systems. Electrochemical devices such as lithium-ion batteries and supercapacitors are particularly unsuited because they are either expensive in terms of cost and area, have limited recharge cycles, high self-discharge rates, or are not easily integrated onboard [ZGL13]. Electrostatic devices such as ceramic or tantalum capacitors can only store limited amounts of energy, but have virtually unlimited recharge cycles, low self-discharge rates, are cheaper and can be easily integrated on board due to their smaller size.

Typical low-power cyber-physical systems have components such as microcontrollers [Tex15, Tex17], memories [Cyp15], and peripherals, for example, sensors[Bos15, Sen18] and transceivers [Sem16, Dig09]. Microcontrollers usually have a wide operating voltage range, but on-chip converters operate most efficiently at lower supply voltages [GPB+15]. External peripherals such as sensors and radios can have substantially different voltage requirements, but system designers usually avoid multiple voltage domains to reduce converter losses and simply choose the highest minimum voltage required to supply the entire system. In order to design a flexible platform that is able to efficiently harvest energy

from different sources, it is necessary to decouple the source and load voltages, allowing each to operate at their respective optimal operating point. We argue that batteryless sensing systems should take all of these issues into consideration. More precisely, we believe these systems should have the following properties in order to be considered useful and efficient:

1. Source and load operating points are decoupled.
2. System has a minimally-sized energy buffer.
3. Load receives required energy at a supported voltage.

The first property ensures functionality and maximum power point tracking [BBMT08] for a wide range of input power and voltage. The second limits the energy that the system can buffer to the absolute minimum since anything larger would not improve functionality but would increase energy losses during cold-start. We define this minimum to be the energy known to be required for the execution of one atomic task. If an application consists of several tasks, the maximum energy level allowed corresponds to the task with the highest energy requirement. In this way, the system has the ability to guarantee functionally correct application progress regardless of environment and transducer dynamics. The third property implies that when the load is activated to execute a given atomic task, it can receive said energy while respecting minimum voltage requirements.

This chapter presents an Energy Management Unit (EMU) which allows a system with limited energy buffering to operate predictably and efficiently, even under very lower power harvesting conditions. Existing works [Yak11, AM15, LPRR10] have looked at low-power systems with energy harvesting and storage capabilities. However, these systems are extremely expensive in terms of harvesting and storage requirements for long-term, efficient functionality under transient power conditions. Our proposed EMU has an optimally sized capacitor which minimizes the required start-up time and energy from zero while maintaining a low cost, small form factor, high efficiency and virtually unlimited charge cycles.

Furthermore, we propose the novel concept of Dynamic Energy Burst Scaling (DEBS) to track the load's optimal power point and minimize its *per task* energy consumption. To evaluate our proposed methodology, we have implemented two vision-based sensor nodes which acquire, process (and store) images. Our experimental results are based on extensive characterization and evaluation of both systems. We summarize the main contributions of this chapter as follows:

- Energy Management Unit that efficiently converts low power levels to short, high power energy bursts.
- Feedback-based Dynamic Energy Burst Scaling technique to track the load's optimal power point.
- Accurate model to optimize system's application-specific parameters for low input power scenarios.
- The design and implementation of two energy-proportional vision sensor nodes.
- Experimental validation of the high energy efficiency and proportionality of the proposed transfer scheme using two separate vision-based sensing applications.

2.1.1 Roadmap

The remainder of this chapter is structured as follows: In Section 2.2n we identify other works in batteryless system design. Our main building block, the Energy Management Unit (EMU), is described in Section 2.3. While the EMU can operate in stand-alone mode, a feedback technique called Dynamic Energy Burst Scaling (DEBS) to minimize application energy is presented in Section 2.4. In Section 2.5, we present two low-power vision sensors that perform sense-process-store applications. These applications are optimized for low energy consumption and adapted to energy-driven, batteryless operation. Our experimental evaluation using both vision sensors is presented in Section 2.6. Lastly, we summarize our findings in Section 2.7.

2.2 Related Work

2.2.1 System architectures

Cyber-physical systems have traditionally been used in conjunction with energy harvesting and energy storage. More recently, the research community has focused on systems with very limited energy storage capacity. In the most extreme case, energy storage is so limited that guaranteed application progress occurs at a very fine granularity, possibly down to a single instruction per activation cycle. Depending on the environment, transducer and application, different types of circuits might be need to supply the system with the energy necessary for program progress at a supported voltage range. Generally speaking, there are three types of architectures for batteryless, or transiently powered systems:

Directly-Coupled

When the energy source has an I-V curve compatible with the load, they can be directly connected. These systems typically use a small decoupling capacitance (<20 μF) to buffer small amounts of energy. If the energy storage is too small, atomic tasks such as sensor measurements and radio transmission cannot be guaranteed unless the environment and the transducer can meet the power and energy requirements of such tasks. The authors of [BWM+15, JRR14] have proposed a combined HW/SW approach to perform computation when the source can directly sustain a computational load during short periods of time. These works use volatile logic that requires state-retention mechanisms. An approach to federating energy proposed in [HSS15a] increases the computational ability by using multiple independent capacitors, each dedicated to a specific peripheral. In [LC15, WLW+15, WCK+14, KCWP10], the authors present storage-less and converter-less harvesting systems in which the load uses frequency scaling to track the maximum power point of the source. While frequency scaling can maximize the energy input in CPU-based applications, it does not minimize the load's energy consumption and

is limited to a narrow active power range. Even though directly-coupled systems avoid converter losses, if the power input is below this narrow active range, the load cannot be powered and the system's efficiency immediately drops to 0%. Unfortunately, this is often the case in batteryless systems. When the energy source and load have incompatible operating points, decoupling them with converters becomes a necessity. As opposed to traditional, battery-based systems, decoupled transient systems have a limited energy buffer between the source and load.

Boost Converter Only

In [DBL+15, DLBL+15], the authors propose a low-power management system that requires very low input voltage and current. Using a large buffer capacitor at the converter input, they are able to start the energy conversion at very low input power level. However, both approaches suffer from cold-start times of at least 18 minutes due to charging a large input capacitance of 140 mF at a constant input power of 2.5 μW. As will be explained in Section 2.3.4, our capacitance is chosen to minimize the cold-start energy and time.

Boost Buck Converter Combination

The authors of [NPK+15] also use a boost converter for optimal power point tracking. However, their proposed system utilizes RF harvesting to accumulate charge in a supercapacitor and then power a camera application with a buck converter. The boost/buck converter topology with an energy buffer also serves as a basis for the approach presented in this work. While a charge-state model is used to characterize the capacitor's self-discharge rate, energy losses such as impedance matching and converter inefficiencies are neglected. More importantly, the system has a large startup cost and can only supply the load with bursts of a constant size and voltage. In Section 2.6, it will be shown that this approach can lead to a substantially higher energy consumption, larger storage elements and longer start-

up times.

To summarize, we propose an Energy Management Unit (EMU) to decouple the load from the source, and efficiently build up charge regardless of the load's operating point. In addition, we propose a feedback-loop technique called Dynamic Energy Burst Scaling (DEBS) that optimizes an application's energy requirements by supplying the minimum voltage *per task*. In this way, a smaller storage can still guarantee the application's completion with a smaller startup time and energy.

2.2.2 Vision Sensors

In this chapter, we will present two wearable vision sensors that acquire pictures and perform image processing. One of these can be used to estimate the walking speed of the user wearing the device. This is done by running an optical flow algorithm. Optical flow, as introduced by Gibson [Gib50] is a basic concept in visual perception that describes the apparent velocities of patterns in the perceived image [HS81]. Since these concepts were first proposed, they have become fundamental in computer vision and signal processing, with applications ranging from video compression [Bar04] to fluid measurements [Far01] among others. For a survey on optical flow algorithms, interested readers are encouraged to look at [BB95]. Due to the applicability of optical flow, researchers have been implementing algorithms on a variety of systems. V. More et al. proposed a visual navigation system for UAVs based on optical flow estimation using the Lucas-Kanade method and an ultrasonic sensor [MKK+15]. To account for the high computational effort for the Lucas-Kanade method, the system runs on an onboard Linux computer (Odroid-U3), which consumes up to 10 W. Computing the optical flow with the Lucas-Kanade method can also be done on computationally constrained microcontrollers such as the Atmel ATmega2560. K. Schneider et al. [SCN13] took this approach and used the same Stonyman vision chip from Centeye as in this work. However, low-power microcontrollers

have very limited memories and, in their case, it could only handle camera resolutions up to 28×28 pixels due to the memory intensive Lucas-Kanade method.

In general, vision sensors have either required too much power, bandwidth or offered too little performance to design functional batteryless devices. In [OHP15], for example, a stereo vision system for a micro aerial vehicle capable of processing 640x480 frames at 60 fps, requiring 5 W and 50 grams for the FPGA subsystem alone. In [HMTP13], the authors use a PX4FLOW optical flow sensor to provide velocity and position estimation at high update rates for mobile robot navigation. The system is not only vision-based but also includes a gyroscope and an ultrasonic sensor. The power consumption of this system is specified as 575 mW. To design a batteryless system with these power requirements would require a solar panel with an area of 100's cm^2, which is no longer wearable. Thanks to our low-power components and aggressive energy optimizations, we can perform many measurements per minute in ambient lighting conditions.

2.3 Reliable Execution with Energy Bursts

In this section, we present the model and architecture of a batteryless sensing system with the proposed Energy Management Unit (EMU). Since the EMU can work with a wide variety of sources, this section omits the transducer and focuses on the EMU and the load. In Section 2.6 we will discuss evaluate our proposed EMU with two different loads to demonstrate how it is compatible with different application requirements.

2.3.1 Energy-driven operation

In this section, we have so far presented two low-power vision-based sensing systems. While these could indeed be battery-powered, having a multi-month or multi-year lifetime would require a large energy storage device and the device would no

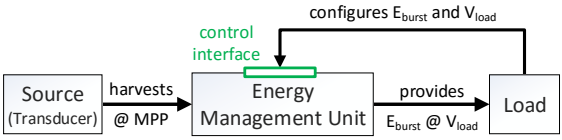


Figure 2.1: Feedback loop for Dynamic Energy Burst Scaling.

longer be wearable or affordable. Current trends in harvesting-based systems point towards a significant reduction in storage capability due to cost, size and environmental considerations. The trade-off in doing so is that a minimal service cannot be guaranteed for long periods of time. This is because as storage capability decreases, the behavior of these systems becomes more immediately influenced by the environment. Since all environments show some form of spatial or temporal dependence, energy availability will vary and affect system operation. Duty-cycling is a common technique which allows a system to adjust its average energy consumption by introducing low-power sleep states. However, in order to perform single tasks such as reading a sensor value or transmitting a data packet, these systems need to be able to buffer the required energy. Otherwise, environmental conditions can rapidly change and turn off the load before it completed its task. Consequently, we argue that a novel Energy Management Unit (EMU) is needed to provide energy guarantees in such disadvantageous scenarios in an efficient, transducer-agnostic manner. Due to the limited energy intake in batteryless systems, the unit should self-start requiring as little time and energy as possible. During those short periods of limited energy intake, it maximizes the energy build-up by harvesting at the source's optimal power point. When powering the load with short energy bursts, it should provide a control interface to the load so its optimal power point can be tracked. In this work, we present an EMU that satisfies these requirements, shown in Figure 2.1.

The proposed Dynamic Energy Burst Scaling (DEBS) technique aims to exploit the EMU's control interface by

closely following the application's minimum required power envelope. To illustrate with an example, imagine a simple sensing application with two tasks: 1) acquisition and 2) transmission. The first requires a sensor supplied with 3 V, while the second requires only 2 V. One approach, used in [NPK+15], uses bursts of constant size and supply voltage. Using DEBS, our proposed EMU is able to produce one burst at 3 V for sensing, and another burst at 2 V for transmission, thus minimizing the total energy.

2.3.2 System Model

We now describe our model of the proposed Energy Management Unit (EMU), shown in Figure 2.3. One of the main goals is to derive equations which can apply to a wide variety of energy sources and loads. The proposed model will then be used to optimize important system parameters, namely the EMU's start-up costs and the load's energy. The accuracy of the proposed model will be experimentally validated in Section 2.6.

2.3.3 Energy Buffering and Losses

The amount of energy buffered in the EMU depends on several parameters including the input and load power, and the system's non-idealities. The equation governing the time-dependent energy level in a capacitor is as follows:

$$E'_{cap}(t) = \frac{d}{dt}E_{cap}(t) = \eta_{boost}\left(V_{in}(t), I_{in}(t)\right) \times P_{in}(t) - P_{load}(S_i)/\eta_{buck} - P_{leak}(t) \tag{2.1}$$

In this equation, the positive term represents the energy intake, while the negative ones represent the energy consumption.

Input Power: The system has only one power input, $P_{in}(t)$, supplied by the transducer. We focus on the adverse scenario where $P_{in} < P_{load}$ and $V_{in} < V_{load,min}$. This means that directly coupling the transducer to the load is not possible since it would not meet voltage requirements. Furthermore,

the batteryless sensing system can be placed in dynamic environments. In these cases, maximizing the system's overall energy flow demands that the source's maximum power point be tracked.

Load Power: In the proposed model, the load can have two states (S_i): active or inactive. When active, the load is characterized by three quantities: $E_{burst,i}, V_{load,i}, P_{load,i}$; where $E_{burst,i}$ defines the energy burst size required for one execution of task i, $V_{load,i}$ its supply voltage and $P_{load,i}$ the power consumption during the execution of task i. These parameters were characterized experimentally. In the inactive state, the load is in deep sleep and awaits the trigger from the energy management unit. Though the actual power consumption during deep sleep depends on the hardware, many systems achieve nW levels [Tex15]. If the deep sleep power is higher, it will simply take longer for the EMU to accumulate the energy necessary for the next burst.

Converter Efficiencies: Since decoupled systems can have the source and load operating at different voltages, converters are needed. This step, while necessary, introduces non-negligible losses, which are represented by boost and buck converter efficiencies $\eta_{boost}(V, I)$ and η_{buck}. The boost converter's efficiency is particularly sensitive to the operating voltage and current, meaning it must be parameterized. These efficiencies were also characterized experimentally, and a simple look-up table is used for simulations.

Other Energy Losses: Unfortunately, converter inefficiencies are not the only sources of energy losses. The maximum power point tracking unit and the control circuit also consume energy. The consumption of the control circuit I_{ctrl} and buck converter I_{buck} consists of a constant current and resistive component and hence depends on V_{cap}. For the energy buffer, a capacitor of size C_{cap}, a resistive leakage R_{cap} is assumed. Considering these components, the system leakage is summarized as:

$$\begin{aligned} P_{leak}(t) = & V_{cap}(t) \times \left(I_{ctrl}\left(V_{cap}(t)\right) + I_{buck}\left(V_{cap}(t)\right)\right) \\ & + V_{cap}(t)^2 / R_{cap}. \end{aligned} \tag{2.2}$$

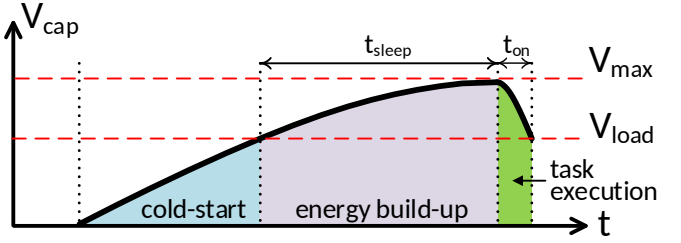


Figure 2.2: Start-up time and cold-start energy overhead.

Equations 2.1 and 2.2 can accurately describe the time evolution of the system's energy levels, as will be shown in Section 2.6.3. They will be used in the remainder of this section to estimate how different parameters impact the system's losses, to then calculate the optimal parameters that minimize the losses.

2.3.4 Minimizing Cold-Start Energy and Start-up Time

Given the system model presented above, we can start optimizing the cold-start energy and start-up time. By definition, this is the fixed start-up cost to turn a batteryless system on. Figure 2.2 shows that after a period of energy unavailability, the capacitance first needs to be recharged to the level of V_{load}. In order to minimize these fixed costs for a given input power, we need to minimize the start-up time defined as:

$$t_{start\text{-}up} = \left\{ t \mid V_{cap}(t) = \sqrt{\frac{2\int_0^t E'_{cap}(\tau)\, d\tau}{C_{cap}}} = V_{load} \right\} \tag{2.3}$$

However, the minimum capacitance is limited by the EMU's maximum supported voltage swing, as shown in the following equation:

$$C_{min,i} = \frac{2E_{load,i}}{\eta_{buck}(V_{max}^2 - V_{load,i}^2)}, \tag{2.4}$$

where $E_{load,i}$ and $V_{load,i}$ are the energy and voltage required to execute task i, and V_{max} is the EMU's maximum supported

voltage. The optimal capacitor value is then selected as the highest $C_{min,i}$ among all tasks i.

2.3.5 Regulating Load Voltage

To show the advantages of our EMU's boost-buck architecture compared to the boost-only architecture, let us consider the case of supplying a constant current load. Assuming the load has a maximum supply voltage tolerance from V_{max} down to V_{min}, we have the following power consumption: for boost-only architecture, the average power of a task is $P_A = (V_{min} + V_{max})/2 \cdot I_{load}$, while the buck has a constant power of $P_B = (V_{min} \cdot I_{load})/\eta_{buck}$. By comparing these two power consumptions, it directly follows that buck converter reduces the load's power consumption if the following condition for the buck converter efficiency holds:

$$\eta_{buck} > \frac{2V_{min}}{V_{min} + V_{max}} \tag{2.5}$$

To illustrate with a numerical example, suppose a load has a voltage tolerance of 3 to 5 V. This means that a buck converter has a lower power consumption if $\eta_{buck} > 75\%$. Furthermore, the use of a buck adds the possibility of tracking the load's optimal power point. When an application consists of multiple tasks with different voltage requirements, we can use Dynamic Energy Burst Scaling (DEBS) to minimize the load's energy.

2.3.6 Energy Management Unit Architecture

The Energy Management Unit (EMU) is tasked with building up energy and producing short bursts to power the load. The EMU provides a control interface to dynamically adjust the bursts' size and voltage, as shown in Figure 2.3. Our proposed Dynamic Energy Burst Scaling (DEBS) technique exploits this by using a feedback loop to track the load's optimal power point and minimize its energy consumption.

Converters. The harvesting part of the system is based on the commercial bq25505 energy harvesting chip. This chip uses a boost converter to convert the input voltage to

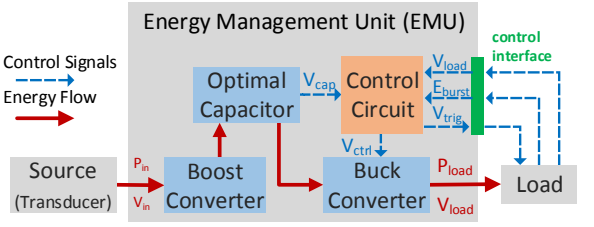


Figure 2.3: Architecture of the proposed Energy Management Unit and its interconnection to the source and load.

a level where the energy can be stored in a storage device. Using its integrated maximum power point tracking (MPPT), the boost converter adjusts the input impedance such that the power source always operates at its optimal power point to maximize the harvested energy. To provide the required output voltage to the load, the TPS62740 buck converter is directly connected to the energy buffer. This buck converter was chosen for two reasons: 1) its high efficiency, and 2) its digitally controlled adjustable output voltage. Given that the measured η_{buck} is $\geq$ 85% for both applications circuits and the EMU supports a maximum voltage swing from 5.5-1.8 V, the inequality 2.5 confirms that using a buck converter is more efficient. Furthermore, it allows tracking the load's power point, which will be discussed in Section 2.4.

Energy Buffer. The energy buffer between the input voltage boosting and output voltage regulation guarantees voltage and current separation of the source and load, thus allowing independent optimization of these parts. Although actual energy requirements are highly application-specific, the fact that we only need to supply a single, short burst of activity means that energies are relatively small. Typical applications scenarios like sensing [Sen18, Sil13] and transmitting [Tex16, Dig09] have activity cycles less than one second. SMD capacitors (ceramic, tantalum and electrolytic) are capable of storing enough energy for many low-power applications. They are also very suitable for batteryless applications because of their high power density [CC00] and

virtually unlimited number of charge cycles [Col18]. When the energy is minimized per task, the smallest capacitor capable of guaranteeing atomicity can be used. In this case, it can also be said that such capacitor also minimizes the EMU's start-up time and energy cost from zero.

Control Circuit The control circuit manages the burst size as well as the output voltage and oversees the energy accumulation in the buffer. For the first, the battery OK signal of the bq25505 is used to trigger the activation of the load, once the capacitor voltage reached the threshold level V_{th} at which the requested energy burst was accumulated. The threshold voltage V_{th} is configured using a resistor network, which is controlled by a digital switch to dynamically adjust V_{th} and therefore the burst size. The load supply voltage V_{load} can be controlled using the TPS62740 buck converter's digital input.

2.3.7 Application Circuit (Load)

All sensing systems have an application circuit, or load, which includes the components that actually perform the sensing application. They include a microcontroller and peripherals (e.g. sensors and radios). In Section 2.5, the two sample sensing systems used in the evaluation were presented. Regardless of the specific components being used, all sensing applications can be decomposed into tasks, depending on the peripherals they use. For example, one application with two tasks can determine an environmental parameter with a sensor, and transmit the result wirelessly with a radio. To be compatible with batteryless execution, the application circuit must have at least two states: *on* and *off*. Since batteryless execution is energy-driven, the EMU needs to control over *when* state transition happens, otherwise, task completion cannot be guaranteed.

For higher energy efficiency, the sensing system's execution flow, shown in Figure 2.4, is more intricate. Instead of a single $V_{load} = 0\,V$ *off* state, there is an additional low-power *deep sleep* state. If the environment has a long period of energy unavailability, V_{load} will inevitably enter cold-start and reach

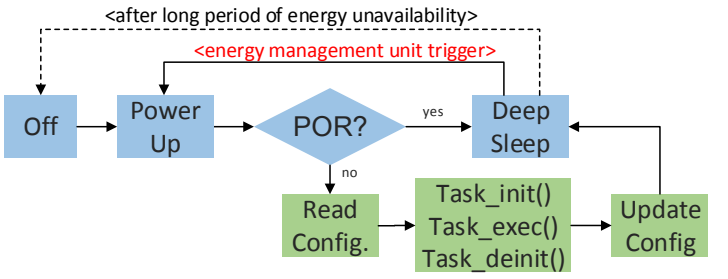


Figure 2.4: Execution flow in all duty-cycled systems has low-power modes (blue) and an active mode (green). For energy-driven execution, the transition between states is handled by the EMU.

$0\,V$. It is said that the system exits cold-start once V_{load} reaches the minimum load voltage and checks the Power-On-Reset (POR) flag. Then the microcontroller performs some basic initialization and immediately enters deep sleep. This deep sleep is preferably ultra-low-power, in the nW range, since it directly influences the buildup of energy for the next burst. When the next burst is generated, the EMU triggers a control signal to wake up the load. The system then reads the next task configuration from non-volatile memory and starts its execution after initializing the peripherals needed for that task. At the end of the task, the configuration is updated and the next required burst is configured. Afterward, the load enters deep sleep again and waits for the next energy burst to build up.

2.4 *Per Task* Energy Minimization

As was discussed in Section 2.5, there are application circuits whose optimal power point varies according to the task. This can occur when tasks use peripherals with different voltage requirements. For this scenario, our proposed EMU provides a control interface to dynamically adjust the burst size and voltage. Our proposed Dynamic Energy Burst Scheduling (DEBS) technique is based on a feedback loop (Figure 2.1) that

allows the load to configure the EMU to supply an energy burst to execute a task at the task's optimal operating point. This configuration only requires changing the value of 4 digital I/O pins.

There are multiple benefits to tracking the load's operating point. For starters, it reduces the energy required to execute a single application. In single-burst applications, these energy savings directly translate to a smaller storage device. However, if it is possible to split the application into multiple bursts, the required storage can be reduced even further. The minimal required energy storage depends on the largest atomic task, and not on the entire application.

To illustrate the potential energy savings from DEBS, let us use a simple application consisting of two tasks T_1 and T_2. First, let us assume that they require the same voltage ($V_{1,2}$=3.3 V), current ($I_{1,2}$=10 mA) and execution time ($\tau_{1,2}$=100 ms). These values are representative of typical sense and transmit tasks. The total application energy, E_{app}, can be calculated as follows:

$$E_{app} = E_{T_1} + E_{T_2} = V_1 \times I_1 \times \tau_1 + V_2 \times I_2 \times \tau_2 \qquad (2.6)$$

If we now assume the voltage requirements for T_2 are now reduced, we can apply DEBS to reduce the application energy. T_2's energy consumption will be reduced if the supply By how much depends on how the actual component's current scales. We evaluate two conservative possibilities: the current is constant, and the current decreases linearly with the voltage. Figure 2.5 shows the potential reduction in application energy (E_{app}) by using DEBS. The X-axis is the minimum voltage requirement of T_2, or V_{min,T_2}. For each V_{min,T_2}, we calculate application energy when using DEBS and we normalize it by the non-optimized version with V_2=3 V. The blue dashed line was computed with the assumption that T_2's current is constant (I_1=I_2=10 mA), while the green dotted line assumes linear scaling ($I_2 = V_{min,T_2}/V_1 * I_1$). As expected, the application energy is proportional to V_{min,T_2}. In the case of constant I_2 current, DEBS can reduce the application energy by up to 23 %. If I_2 scales linearly with V_2, the reduction is almost 36 % at V_2=1.8 V. Finally, it should be noted that compared to a single

burst scheme, the optimized two-burst application requires an energy storage element only half as big (dominated by E_1).

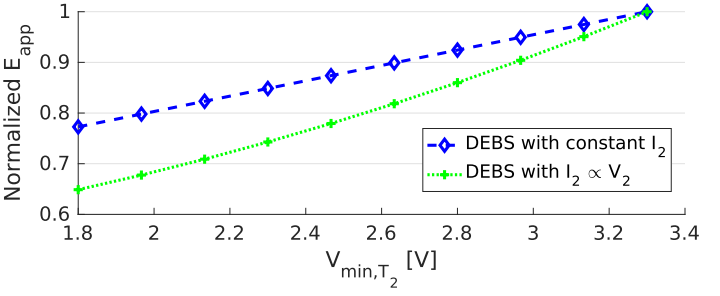


Figure 2.5: Normalized application energy with and without Dynamic Energy Burst Scaling (DEBS). Energy reduction depends mainly on the operating voltage difference between tasks. In this plot, $V_{T1} = 3.3\,\text{V}$ and $V_{T2} \in [1.8, 3.3]\,\text{V}$.

2.5 Low-Power Sensing Systems

For over a decade, there has been significant research and development into low-power sensing systems. These systems are typically designed to obtain environmental information, process it, and transmit wirelessly. Though they were originally meant to run for multiple years on a small battery, current trends point towards systems that run entirely from harvested energy, without long-term storage. Vision sensors, which acquire and process images, are typically high-bandwidth devices which require substantial computational resources. For this reason, it was not until recently that it became feasible to have harvesting-based vision sensors in a wearable form factor, thanks to new paradigms in energy harvesting systems. Wearable vision sensors are a nascent field with many potential applications in fields including healthcare, security, leisure and the quantified self [BDR17]. In [MBS+16] and [CPA+17], for example, vision data partly used to determine the user's context. In [OVV17], a wearable camera

is shown to have lower false positive rates than accelerometer-only systems for detecting falls of elderly users. This section presents two low-power vision-based applications which were implemented and evaluated for energy-driven operation. The results presented here were obtained in collaboration with Lukas Sigrist, Thomas Schalch and Michele Magno.

In this work, we implemented two low-power vision sensor nodes which can acquire, process and store images. The general overview of both vision sensors can be seen in Figure 2.6. Both are based on the Centeye Stonyman sensor [Cen13], which takes 114×114 pixel black-and-white images and features low energy consumption and ultra-low-power deep sleep mode. Each sensor node uses its own microcontroller from the MSP family: MSP432 [Tex17] and MSP430FR [Tex15]. The former has a 32-bit ARM Cortex M4 CPU with a floating point unit, 64 KB of SRAM and 256 KB of Flash. The latter is an ultra-low-power 16-bit microcontroller with 64 KB of on-chip FRAM. For high-density data storage, an external 2 GB Class 10 microSD card [San04] is used. With these vision sensors, we will implement two sensing applications: Sense-Process and Sense-Process-Store. Sense-Process applications obtain sensor data and perform some processing of the data to either filter, compress, or otherwise prepare data. Sense-Process-Store application can also save data in non-volatile memory for future use.

2.5.1 Walking Speed Estimation (Sense-Process-Store)

Vision sensors have been studied for many years, and there has been a significant research effort to use them for optical navigation of unmanned aerial vehicles (UAV) or robots. V. More et al. proposed a visual navigation system for UAVs based on optical flow estimation using the Lucas-Kanade method and an ultrasonic sensor [MKK+15]. K. Schneider et al. [SCN13] took this approach and used the same Stonyman vision chip from Centeye as in this work. However, low-power microcontrollers have very limited memories and in their case, it could only handle camera resolutions up to 28×28 pixels

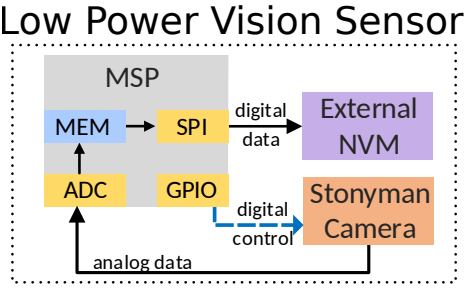


Figure 2.6: Overview of implemented low-power vision sensors. Two microcontrollers from the MSP family (MSP430FR [Tex15] and MSP432 [Tex17]) will be used to read images from Centeye's Stonyman sensor [Cen13]. There is also the possibility to store data in external non-volatile memory.

due to the memory intensive Lucas-Kanade method. Here, we propose a low-power wearable vision sensor based on the generic architecture shown in Figure 2.6. This sensor is attached to a user's glasses to estimate and log the walking speed of the person. Figure 2.7 shows the generic function of a visual velocity estimation sensor node, which can be divided into three parts: *Image Acquisition and Compensation*, *Image Processing* and *Storage*.

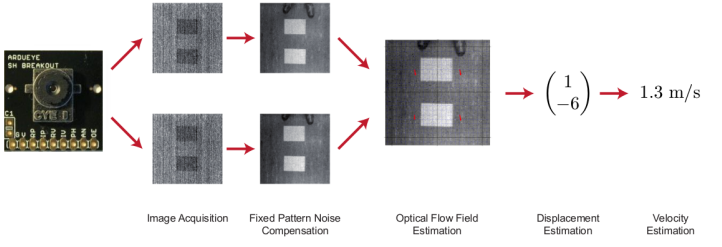


Figure 2.7: Overview of Sense-Process-Store application. After acquiring two images and estimating user displacement, the walking speed can be estimated. Both pictures, as well as the velocity estimation, are stored in the external microSD card.

Image Acquisition and Compensation. The image acqui-

sition task takes two images in a sequence and compensates for the fixed pattern noise (FPN). In order to minimize the computational effort of the motion estimation algorithm, the time difference between the two acquired images should be minimal. This was achieved by operating the vision sensor at the maximum frame rate of 37.5 frames per second. To achieve this frame rate, the CPU frequency has to be at least 24 MHz. Using a CPU frequency of 48 MHz even allows compensating for the FPN on-line during acquisition, while still achieving the same frame rate. This task is executed at 3 V since it is the Centeye camera's minimum supported voltage. The time and energy characterization can be seen in Table 2.1. For all characterizations, the circuit was supplied a DC source and its power consumption recorded for the energy calculation. The evaluation reveals that configuration with the lower CPU frequency is beneficial in a batteryless system since the overall energy consumption is 42% lower.

FPN compensation	**CPU Freq.**	**Executions**	$E_{task,avg}$	$t_{task,avg}$
on-line	48 MHz	1152	930 μJ	**53 ms**
sequential	24 MHz	1829	**537 μJ**	61 ms

Table 2.1: Energy consumption and execution time of the image acquisition task with compensation.

Image Processing. Using the two acquired image frames, a velocity estimation needs to be calculated. This is done using a motion estimation algorithm which will be discussed below. The result of the motion estimation algorithm is a so-called optical flow field, a vector field in which every vector indicates the displacement of the corresponding part of the image between the two frames. This optical flow field is then reduced to one single displacement vector indicating the displacement between the two frames. Together with the user's height, camera view angle and image acquisition times, this displacement can be scaled to a final velocity estimation.

Here, we implement a block-based algorithm. This heuristic estimates the motion between two images by comparing them for all possible displacements and determining where the images "fit best". Such a method only works, if there is a pure

translation and only a negligibly small rotation between the images. To relax this condition, one could think of dividing the image into blocks and then search for each block the position in the next image where the block "fits best". The smaller the block size, the more robust the method is to rotations. The result of the algorithm is an independent translation vector for each block, which represents an estimate for the optical flow of the pixels within the considered block. Block-based methods are not restricted to small displacements like the differential methods and are therefore able to recognize displacements of many pixels.

For the task characterization, the block size of the block-matching algorithm is set to 48px and the search area to ±3 and ±8 in x- and y-direction respectively. The supply voltage for this task was set to 2.2 V since it is the MSP432's minimum supported voltage for the required frequency range. Table 2.2 compares the energy consumption and the execution time of the processing task for different CPU frequencies. As expected, higher CPU frequencies lead to shorter execution times. However, shorter execution times do not necessarily result in lower energy consumption, because the MSP432 MCU consumes much more power for higher clock frequencies [Tex17]. As the results in Table 2.2 show, the energy consumption of the processing task is lowest at a CPU frequency of 24 MHz.

CPU Freq.	**Executions**	E_{task} **(mean)**	t_{task} **(mean)**	P_{task} **(mean)**
48 MHz	1079	757 μJ	**56 ms**	13.5 mW
24 MHz	1254	**635 μJ**	110 ms	5.8 mW
12 MHz	770	1371 μJ	388 ms	3.5 mW

Table 2.2: Energy consumption, execution time and power consumption of the processing task.

Storage Task. The storage task takes the result from the image processing, as well as the two acquired pictures, and stores it in non-volatile memory (NVM). Each image has a size of 12544 bytes, while the resulting velocity is only 2 bytes. Although the SD card only requires 2.7 V, its high current peaks made it necessary to raise the operating voltage. For

our specific system, the safe operating voltage was empirically determined to be 3 V. The energy required for storing both images and the walking speed estimation was 6684 μJ.

Power Management and Non-Volatility. To support batteryless operation, a sensor node needs to be able to reduce its activity and, consequently, its power consumption. The Stonyman vision chip supports a software shut-down and can be connected directly to the supply voltage. The FRAM chip is connected to the supply voltage via a load switch (TPS22960) controlled by GPIO pins of the MSP432. This allows the MCU to activate the microSD card only when needed. The MSP432 is optimized for low-power applications and features various low-power modes (LPMs) [Tex17]. In the deepest low-power mode (LPM4.5), all peripherals including the CPU and the SRAM bank are disabled. For this reason, it is imperative to perform all three tasks within a single activation cycle, otherwise, on-chip data memory is lost. During deep sleep, the GPIOs can be configured to lock their previously configured state which will be kept as long as there is power. This is of particular importance if the sensor node needs to interface with other external components. The current consumption in LPM4.5 is only 25 nA [Tex17], allowing the device to significantly reduce its average power consumption.

2.5.2 Image Capture (Sense-Process)

Following our vision-based sensing examples, we now present a simpler Sense-Process application. Using the MSP430FR5969-based vision sensor, we implement an image capture application. This microcontroller features 64 KB of FRAM, a high endurance, low-power non-volatile memory (NVM). For this application, the images are simply stored (and overwritten) in the on-chip NVM. The application consists of two tasks: 1) an image acquisition task, and 2) a basic image processing task, which are present in all vision-based sensor nodes. During the image acquisition task, the raw image from the Stonyman's analog output is read with the ADC. Because the sensor node has only one power domain and the

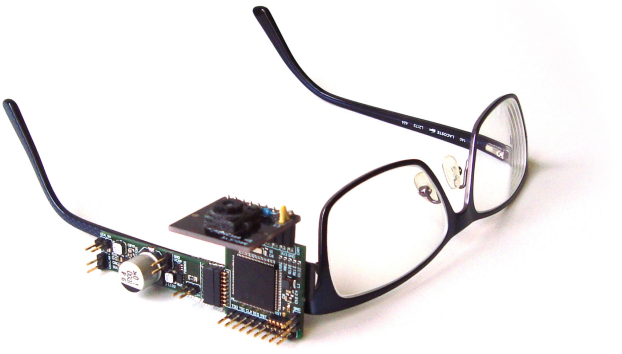


Figure 2.8: Image of the wearable, batteryless vision sensor prototype mounted on conventional glasses (no transducer shown).

sensor's minimum voltage requirement, this task requires a supply of 3 V. The processing task, however, only requires the microcontroller which can operate at 3.0 V, but also at the reduced voltage of 2 V. Table 2.3 shows the characterization of the task energies. For these measurements, the circuit was supplied a DC source and its power consumption recorded for the energy calculation.

Task	**Voltage [V]**	t_{task} **[ms]**	E_{task} **[μJ]**
Image Acquisition	3.0	47	**174.0**
Image Processing	3.0	50.6	220.8
Image Processing	2.0	50.6	**143.7**

Table 2.3: Energy consumption at different operating voltages of the Image Capture application.

Power Management and Non-Volatility. The MSP430FR5969 has a deep sleep power consumption of less than 50 nW, making it very efficient for aggressive duty-cycling. In addition, the IO state lock mechanism and non-volatile FRAM features of the microcontroller enable it

to keep the interface state during deep sleep and maintain a well-defined state with external components. Due to this microcontroller's unified memory system, the need for state retention mechanisms is mitigated. Valuable data can be stored and processed entirely in the non-volatile domain, thus saving expensive transitions between volatile and non-volatile memory.

2.6 Experimental Evaluation

2.6.1 Methodology

This section evaluates the costs, performance, and efficiency of the Energy Management Unit (EMU). To do this, we utilize the two low-power vision sensors and their initial characterization first presented in Section 2.5. They implement two different applications: Sense-Process-Store, and Sense-Process. In the former, we highlight the ability of the EMU to supply the application circuit with the voltage necessary to minimize the application per task. In the latter, we emphasize additional system parameters like cold-start energy and startup times as well as the energy savings from using Dynamic Energy Burst Scaling (DEBS). To do this, we will test the performance of the EMU by running the Sense-Process application using two execution profiles: 1) Dynamic Bursts (using DEBS), and 2) Constant Bursts (no DEBS). Both execution profiles will be tested under a) constant power input, and b) variable power input. These last experimental results will also be compared to a discrete-time simulation of the model presented in Section 2.3.2 using Matlab. With both applications, we will demonstrate the energy efficiency and proportionality of batteryless execution.

Experimental Setup

Both application circuits are interfaced with the EMU. Flexible solar panels from PowerFilm were used as transducers for both systems. The Sense-Process-Store application used the MP3-37

version, with an area of 42 cm^2. The Sense-Process application used the MP3-25 version, with an area of 28.5 cm^2. In both scenarios, controlled lighting was used to expose the system to different illumination levels typically representative of low light levels (<1000 lux).

Figures of Merit

For the system performance analysis, the following metrics are used in all experiments:

- $E_{in} = \int_0^{T_{exp}} P_{in}(t)\, dt$, for the total input energy,
- $E_{app,j} = \sum_{i=1}^{N_{tasks}} \int_{t_{active,i,j}} P_{load}(t)\, dt$, for active energy consumed by the j-th application execution,
- $E_{load} = \sum_j E_{app,j}$, the total energy consumed by the load for all application executions,
- $\eta_{sys} = E_{load}/E_{in}$, the total system efficiency, and
- $R_{exec} = N_{bursts}/(N_{tasks} \cdot T_{exp})$, the application execution rate.

In the formulas above, $t_{active,i,j}$ denotes the time interval of task i in the j-th application execution, N_{tasks} the number of tasks in the application, and N_{bursts} the number of bursts during the experiment of duration T_{exp}. By measuring the currents and voltages at the relevant points, we can accurately measure P_{in} and P_{load}, and experimentally calculate these metrics.

2.6.2 Sense-Process-Store Application

The walking speed estimation system consists of an MSP432-based vision sensor which takes two pictures in sequence, estimates the user's displacement, and stores the images as well as the walking speed estimation on an external microSD card. The simplest execution profile for this application is to have a single activation cycle which performs all of these tasks.

EMU-based execution requires the configuration of energy bursts, which supply the load with a predefined amount of energy at a given voltage. For this application, only one energy burst is required.

Single-Burst Configuration. During this burst, the vision sensor will perform all of the aforementioned tasks (Sense-Process-Store). The two EMU configuration parameters which are missing are operating voltage and the capacitance value. The vision node's energy optimizations were presented in Section 2.5.1. The minimized energy for the entire application is 7.856 mJ. To guarantee this energy between the 5 V-3 V voltage gap, a 1 mF capacitor can be used. This way, reliable execution of the walking speed estimation can be guaranteed, even with highly variable energy sources.

Motion Estimation Accuracy. To find an optimal value for the block size, a sample image data set was first acquired from a development prototype with an SD card to continuously record images of a controlled displacement. These images were then post-processed to compare the estimated and the measured displacements. A Matlab model of the block-matching algorithm tested different block sizes between 96 pixels (using the whole image as one single block) and 16 pixels (dividing the whole image into 36 blocks). The size of the search area was set to ±8px in x- and y-direction. A block size of 48px showed the best combination of sensitivity against rotations and high x-direction accuracy. After a traveled distance of 42 m, its estimated position deviates only 57 cm from the reference value, which corresponds to only 1.4% of the traveled distance. Though this accuracy does depend on the type of floor surface, we assume it also holds for our real-world experiment, which has an equivalent experimental set-up.

Real-World Velocity Estimation. In order to demonstrate the velocity estimation in a typical wearable scenario, the wearable vision sensor prototype was attached to the glasses (see Figure 2.8), such that the Stonyman vision chip faces downwards and captures the floor in front of the person. For this experiment, the user tested a stand-walk pattern for a few seconds each and with different walking speeds. The prototype

was continuously powered to acquire as much data as possible.

The vision sensor executes a loop of acquiring two images, estimating the displacement between two images and writing the images and the results onto the NVM. Because changing the clock frequencies during runtime causes some overheads and for optimizing the application for speed, a CPU frequency of 48 MHz is used for the whole application. The search area of the block-matching algorithm is configured to ±3 pixels in x-direction and ±8 pixels in y-direction. The block size of the block-matching algorithm is set to 48px, which was found to be optimal according to the pre-characterization.

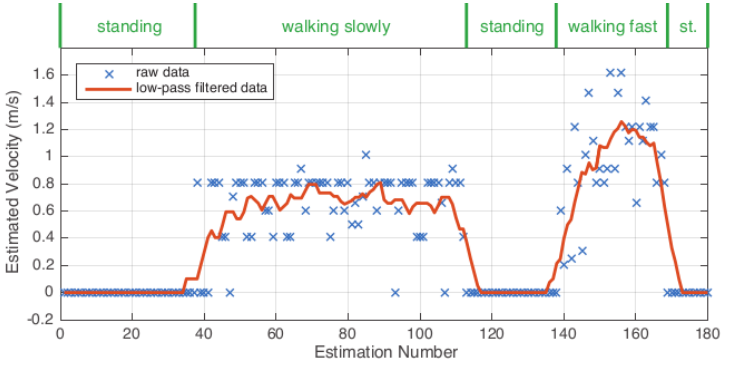


Figure 2.9: Velocity of the walking person estimated by the vision sensor prototype (blue crosses) and post-processed in Matlab using a low-pass filter (red line).

Figure 2.9 presents the walking speed estimations depicted as blue crosses, as well as the low-pass filtered data using a moving-average FIR filter in Matlab (solid line). The different phases of the experiment can clearly be recognized. Furthermore, the velocity estimation sensor is able to distinguish between different walking speeds. Batteryless operation will not always guarantee that the maximum execution rate is always achieved, as this depends on the harvesting conditions which can be highly variable. However, EMU-based designs operate on the notion of energy guarantees, which means that measurements are only begun

when their completion is guaranteed. Furthermore, the load's operating point can, for this brief moment of time, be guaranteed to be *per task* optimal.

Energy Efficiency and Proportionality

For this experiment, the batteryless sensing system consisting of the solar panel, the EMU, and the MSP432-based application circuit was exposed to different illumination levels. Each measurement lasts between 300 s (for high input powers) and 900 s (for low input powers) to ensure capturing a representative number of burst executions. The energy efficiency, calculated by measuring the energy harvested by the EMU and consumed by the vision sensor, can be seen in Fig. 2.10. The same plot also shows the average number of estimations per minute which were recorded during each experiment.

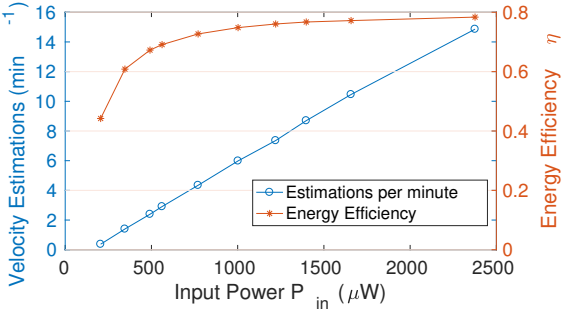


Figure 2.10: Measured energy efficiency and execution rate as a function of the input power.

As expected, both velocity estimations per minute and the system's energy efficiency increase with the input power. In both cases there is X-intercept is greater than zero since the EMU has a minimum input power in the μW range before any application circuit can be effectively turned on. Once the minimum input power is surpassed, the number of velocity estimations is proportional to the input power, and the slope

is determined by the average energy needed for one velocity estimation. Though there is an upper bound to the number of velocity estimations, it was not reached due to the duty-cycling required in low energy harvesting conditions. The overall energy efficiency, however, does saturate rather quickly. This can be explained by the product of the EMU's boost and the buck converter non-ideal efficiencies. The EMU's maximally achievable efficiency for this circuit was measured to reach up to 78.7%. For input power levels above 600 μW, the efficiency is higher than 70%. The lowest input power level in the experiment is 198 μW and achieves an efficiency of 43% when supplying the vision sensor with an average power consumption of 6.85 mW.

2.6.3 Sense-Process Application

The second vision-based application we have presented in this chapter was the image capture. The MSP430FR-based system acquired and processed an image on on-chip NVM. This application consists of two tasks: acquisition and processing, whose energy optimization was presented in Section 2.5.2. We will evaluate this application with two EMU-based execution profiles. The first is to execute the entire application within a single energy burst. The second is to generate one energy burst per task using DEBS.

Single-Burst Configuration. With this configuration, the vision node will perform the entire application within a single activation cycle. Consequently, reliable execution under variable harvesting conditions requires that the entire application energy be buffered before it is consumed. This is the only way to guarantee completion if the source stops harvesting energy when the application starts. Without any feedback-based control loop, a constant supply voltage would need to be selected. Due to the voltage requirements of the vision sensor, this constant supply voltage was set to 3 V. The required energy burst size, according to the data presented in Table 2.3, should be at least 394.8 μJ. The minimum capacitance supported by the EMU is 80 μF, which can provide this energy

(plus a safety margin) in the 5 V to 3 V voltage gap. Since all activation cycles are the same in terms of energy, we will call this configuration *constant bursts*.

Multi-Burst Configuration. The image capture application can also be executed with two separate bursts using DEBS. During the first burst, only 174 μJ at 3 V are consumed for image acquisition. Once enough charge has been built up, the EMU's control circuit configures the buck converter's digital input to set the output to 3 V and triggers the load to acquire the image. Afterward, the load uses the EMU's interface again and configures the second burst by a new wake-up threshold and operating voltage. The energy burst should supply 143.7 μJ to the sensor (plus any start-up overhead) at 2 V.

Thanks to the MSP430FR's unified memory, the application can be easily split amongst multiple energy bursts. Application data already resided in non-volatile memory at the end of the first burst, meaning no addition state-retention techniques are necessary. This allows the microcontroller to easily use DEBS to supply separate bursts.Since these bursts can have variable energies and voltages, both load-configurable, we call this configuration *dynamic bursts*. Without DEBS, the EMU would generate one burst at 3 V to acquire *and* process the image, similar to the approach proposed in [NPK+15], which can lead to larger bursts sizes due to the non-optimal operating point. These two approaches, dynamic and constant bursts, will be evaluated experimentally in the following section.

Start-Up Time and Cold-Start Energy Costs

As was discussed in Section 2.3.4, the energy buffer was optimized to minimize the cold-start's required energy and start-up time and still guarantee the completion of atomic tasks. To characterize these costs, the capacitor was completely discharged, and the flexible solar panel was exposed to constant illumination level until the cold-start phase ended. The measured start-up time and cold-start energy as a function of the input power can be seen in Figure 2.11. The maximum cost, which occurs at the minimum input power of 20 μW was

118 s and 2.49 mJ. This was expected since the harvester, by definition, cannot operate efficiently in this region. It should be noted that with an input power of 400 μW, the start-up costs go down to 3.9 s and 1.54 mJ. This translates to reduced reaction times and energy costs in batteryless systems, and highlights the importance of a minimized capacitance.

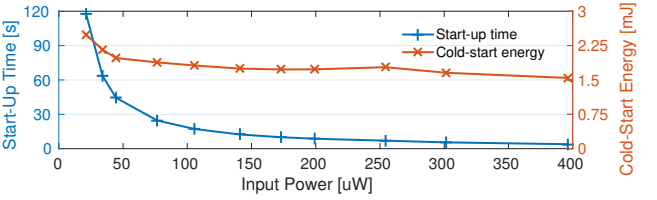


Figure 2.11: Measured start-up time and cold-start energy of image capture application. Even in the low-power harvesting scenario of 50 μW, the system starts up from zero Volts in less than a minute.

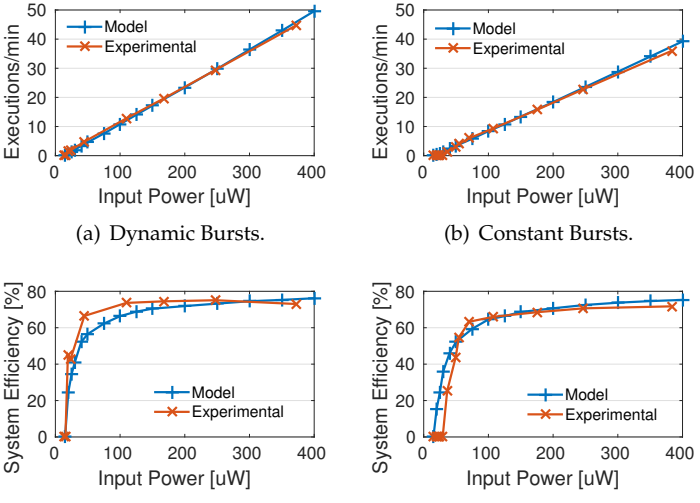
Energy Efficiency and Proportionality

In this experiment, the flexible solar panel was exposed to a constant illumination level for 5 min, supplying the EMU with a constant power. This experiment was repeated for different power levels using Dynamic and Constant Bursts. The system was then analyzed using the previously discussed performance metrics dependent on the input power level.

The resulting analysis of the task execution rate and system efficiency is shown for both task execution profiles in Figure 2.12, together with model-based simulation results for the same scenarios. The results show up to 50 and 39 task executions per minute when using Dynamic and Constant Bursts, respectively. For both profiles, the system efficiency η_{sys} reaches more than 70% for a wide range of input power, with a peak system efficiency of up to 75.1% for Dynamic Bursts. It should be noted that the system model presented in Section 2.3.2 allows accurate simulation of the number of task executions as well as system efficiency. However, some additional non-linear leakage effects of the boost converter at

the very low input power of $< 50\,\mu$W at high buffer capacitor voltages cannot be captured by the model, and results in an optimistic efficiency for Constant Bursts.

The experimental results show that execution rate R_{exec} when using Dynamic Bursts is on average 27% higher than Constant Bursts. Further, Dynamic Bursts lowers the minimal system operating input power down to $19\,\mu$W compared to $36\,\mu$W for Constant Bursts. Lastly, the system efficiency η_{sys} is increased across the whole input power range for Dynamic Bursts, with significant improvements for input powers of $200\,\mu$W and below.



(a) Dynamic Bursts. (b) Constant Bursts.

(c) Dynamic Bursts. (d) Constant Bursts.

Figure 2.12: Experimental evaluation of image capture application under constant input power conditions: execution rate R_{exec} and system energy efficiency η_{sys}.

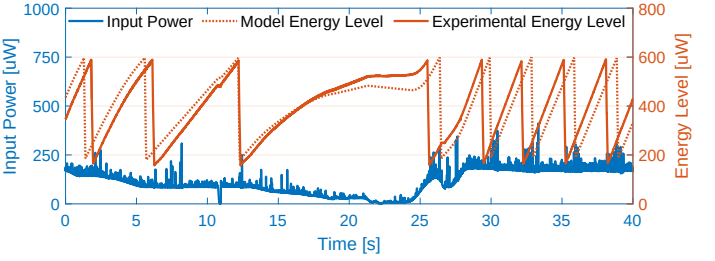


Figure 2.13: Time domain comparison between model simulation and experimental evaluation of image capture application using Dynamic Bursts under variable input power.

Variable Harvesting Conditions

In this experiment, the system performance was evaluated in an indoor real-world scenario, again for both task execution profiles. The two execution profiles were each evaluated with a 15 min experiment that included walking around with the setup in the office hallway partly illuminated by natural and artificial light, walking in a dimly lit basement and sitting at a well-illuminated office desk.

Burst Size	Avg. P_{in}	Metric	Simulation	Experiment	Error
Dynamic	92.3 μW	R_{exec}	9.93 min^{-1}	10.33 min^{-1}	-3.9%
		avg. $E_{app,j}$	368.4 μJ	369.0 μJ	-0.2%
		E_{load}	54.9 mJ	57.2 mJ	-4.0%
		η_{sys}	66.11%	68.82%	-3.9%
Constant	111.9 μW	R_{exec}	9.87 min^{-1}	9.93 min^{-1}	-0.7%
		avg. $E_{app,j}$	460.8 μJ	459.7 μJ	+0.2%
		E_{load}	68.2 mJ	68.5 mJ	-0.4%
		η_{sys}	67.76%	68.01%	-0.4%

Table 2.4: Execution rate and efficiency of image capture application for variable harvesting power. Even though Dynamic Bursts had a lower average input power, their execution rate was higher thanks to the reduced energy cost.

The experimental metrics for Dynamic and Constant Bursts under variable input power conditions are shown in Table 2.4. The first thing to note is that Dynamic Bursts reduces the

average energy per application execution by 19.7% when compared to Constant Bursts. Even though the Dynamic Bursts experiment had on average a lower input power P_{in}, both the execution rate and energy efficiency η_{sys} are still higher. This can be explained by the lower energy consumption per task execution due to DEBS' minimization of load energy. Normalized to the average input power, this results in 22% more task executions with Dynamic Bursts compared to Constant Bursts, which shows the considerable advantage of using DEBS.

The table also compares the experimental results to the model simulation that uses the experimental P_{in} data as input. Here, the comparison to experimental values shows that even in a real-world scenario with variable input power, the model is able to predict the system behavior with a maximum error of 4% for both task execution profiles. This fact is also reflected in Figure 2.13: it shows the input power, simulated and measured energy level of the buffer capacitor during a 40 second sample time window of the Dynamic Bursts experiment. Beside a small time drift in the energy accumulation during very low input power, where not all effects can be represented by our model, it tracks the buffer's energy level and bursts with high accuracy. This high accuracy results only in small deviation in the time diagram, despite the accumulation of simulation errors in the time domain.

2.6.4 Result Discussion

Sense-Process-Store. The hardware and firmware of the vision sensor have been highly optimized for low energy consumption. In particular, the first two tasks have strict dependencies between operating voltage and operating frequency. There is, however, almost no connection between execution time, average power, and energy consumption in the image acquisition and processing tasks. In converterless systems [LC15, WCK+14], there is no chance to guarantee the operating conditions necessary for energy minimization. Thanks to the decoupling guaranteed by the EMU, and the high

power density of capacitors, this vision-based sensing system achieves high performance and energy efficiency. Experiments show that the vision sensor prototype can produce reliable velocity estimations in an energy-proportional manner.

Sense-Process. The results from the constant power characterization and variable input experiment highlight the four main advantages of our proposed approach. First, thanks to our minimized energy buffer, the cold-start energy and start-up time are minimized: at $400\,\mu W$, they were only 1.54 mJ and 3.9 s, respectively. Second, in the very common low-power harvesting scenario for batteryless systems, the EMU completely decouples the source's and the load's power points. Even though the harvested power never surpassed $400\,\mu$W, the EMU still provided the 3.83 mW load with a 75.1% energy efficiency using DEBS. With direct coupling, it is simply impossible to power the same load. Third, the proposed DEBS technique uses the EMU in a feedback loop to track the load's optimal power point and significantly reduce its energy consumption. Fourth, the proposed model is able to accurately predict the experimental results. This validates the minimization of our model parameters, namely the minimized energy buffer and start-up costs.

2.7 Summary

In this chapter, we have presented an energy management unit (EMU) that minimizes the cold-start energy and start-up time for batteryless sensing systems. By only accumulating the energy necessary for an activation cycle in an optimally-sized buffer, the EMU is able to supply generic loads predictably and efficiently, even when harvesting only a small fraction of the load's power. Furthermore, we proposed a Dynamic Energy Burst Scaling (DEBS) technique to track the load's optimal power point. Using a simple interface consisting of only a few digital inputs, the EMU is able to dynamically adjust the burst size and voltage according to an application's needs, thus minimizing the load's energy consumption and reducing

the required storage element size. We have applied these techniques to two different vision-based sensing devices and our proposed model is able to predict the system's performance and energy efficiency within 4.0% of the experimental values, even under variable power input conditions. With our Sense-Process and (-Store) applications, we have demonstrated that batteryless execution can leverage low-power optimization techniques, independent of the transducer's operating point.

3

Reducing energy costs by aggregation

3.1 Introduction

Over the past decade, there has been a considerable research effort to reduce the energy consumption of electronic devices. While there has been considerable progress, the lifetime of battery-based devices remains a bottleneck in their development. Supplying low-power embedded systems with the energy they require in an efficient, low-cost, long-term, scalable, and self-sustainable manner remains an open question. Over-provisioning with large energy harvesting and storage elements is either infeasible or unnecessary in many application scenarios such as wearable, autonomous, miniaturized or "smart dust" systems. Fortunately, a purely harvesting driven system can still meet application requirements in many of these scenarios.

Batteryless sensing systems are supplied by variable energy sources which can, at most, directly power the system for only a limited amount of time. Even then, the energy harvesting

rate might not be high enough to complete one atomic task execution, such as performing a sensor reading or transmitting a radio packet. Consequently, such systems need to be able to buffer at least the amount of energy needed to bridge this power deficit and thereby to guarantee the completion of any single execution cycle. The developer has the liberty to define what this single execution cycle achieves. In the previous chapter, we studied how an application can be executed either atomically or in multiple bursts. By splitting the application over multiple bursts we were able to achieve two important goals. The first was minimizing the energy storage required for executing the application reliably. The reduction in storage depends on the ratio between the number of tasks and their energy ratios.. The second goal was a reduction of the energy cost per application cycle. These energy savings were achieved by adjusting the operating voltage to the minimum supported *per task*. The techniques presented in Chapter 2 result in a Pareto-optimal point on the *application energy cost* vs *energy storage capacity* design space. In this chapter, we will present additional techniques, applicable to tasks with high initialization costs, to determine another point in the Pareto front. Compared to previous techniques, they can be used to reduce the average energy cost of an image logging application but with a higher energy storage capacity.

3.1.1 Batteryless System Challenges

Batteryless systems must be able to tolerate highly volatile sources and still guarantee program progress. In order for these systems to operate reliably and efficiently, they have to accumulate harvested energy until enough is available for the execution of a predefined activation cycle, also called a burst. Choosing what an activation cycle consists of has a significant impact on two important metrics: the application's energy cost and the system's required energy storage capacity. In Chapter 2, we saw one scenario where both metrics were simultaneously minimized. In applications where activating peripherals incurs a significant energy overhead, there is

an inherent trade-off between energy storage capacity and application energy cost. Batteryless, or transiently powered systems, thus face multiple challenges that lead to design trade-offs:

- Constraint (1) **Bounded Storage Capacity** Energy storage is necessary for reliable execution with adversarial harvesting conditions. Though desirable, minimizing energy storage and application energy cost is not always possible. A trade-off exists when tasks have high initialization overheads.
- Constraint (2) **Time-dependent Tasks** There is no control of the time interval between two bursts since this depends on the available input power. If necessary, the application needs to be split into separate bursts *with no temporal dependencies*.
- Constraint (3) **Data Consistency** Data-logging applications need *non-volatile memory* (NVM) to guarantee long-term data consistency. Many batteryless sensing applications also require NVM for functionality. If one application cycle spans multiple bursts, NVM is the only way to guarantee program progress on a system that can loose all power between bursts.

3.1.2 Transient System Configurations

As has been previously discussed, the different properties of batteryless sensing systems require novel approaches to operate efficiently in such disadvantageous scenarios. In this work, we argue that an additional Energy Management Unit (EMU), shown in Figure 3.1, is needed to maximize the harvested energy, minimize the load's power, and provide the load with the energy guarantees necessary for program progress. Due to the limited energy intake in transiently powered systems, the unit should self-start requiring as little time and energy as possible. During those periods of limited energy intake, it maximizes the energy build-up by harvesting at the source's optimal power point. When powering the load

with short energy bursts, it should provide a control interface to the load such that the application circuit's optimal power point can be tracked.

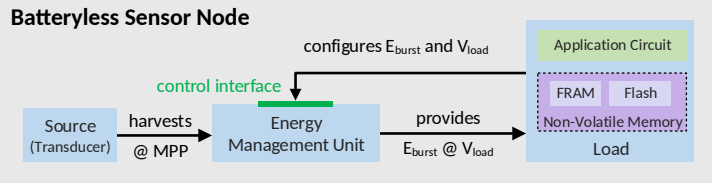


Figure 3.1: Overview of EMU-based systems with a DEBS feedback loop. In this chapter, we will focus on external non-volatile memories for long-term logging applications.

While existing works [MMB+12, Yak11, AM15, LPRR10] have looked at low-power systems with energy harvesting and storage capabilities, these are expensive in terms of harvesting and storage requirements for long-term, efficient functionality under transient power conditions. State of the art transient system design [BWM+15, JRR14], connects the energy source directly to the load, without any other intermediaries. However, these works only focus on non-atomic execution of processing tasks and only work when specific harvesting conditions generate a safe operating voltage on the solar panel. The Energy Management Unit (EMU), presented in Chapter 2, was first proposed to decouple the operating point of the source from the load. Additionally, the EMU can apply Dynamic Energy Burst Scaling (DEBS) to track the load's optimal power point. The EMU allows a system to operate predictably and efficiently with limited energy buffering, even under very low power harvesting conditions where the harvested power is much lower than the load's minimum power requirement. In this work, we focus on specific design aspects of batteryless vision sensors performing long-term logging. Vision-based systems have the property of guaranteed information and energy availability since darkness provides neither energy nor information and light provides both.

3.1.3 Contributions

Rich data sensors such as cameras bring their own challenges to batteryless system design. Logging applications are particularly costly, due to the large volume of data that rich data sensors produce. We thus propose a novel Non-Volatile Memory Hierarchy (NVMH), which increases the energy efficiency of rich data sensor logging applications. We will show how the addition of the NVMH introduces a trade-off between the energy cost per stored byte and the minimum energy buffer size. Our proposed EMU-based design uses an optimally sized capacitor which minimizes the required start-up time and energy from zero, while maintaining a low cost, small form factor, high efficiency and virtually unlimited charge cycles.

The main contributions of this work are summarized as follows:

- A Non-Volatile Memory Hierarchy (NVMH) that reduces the average energy costs of reliable sensing applications with high-power peripherals (e.g. microSD cards).
- Accurate model to optimize system's application-specific parameters for low input power scenarios, including energy and data buffer sizes as well as harvester's dimension.
- Experimental validation of the high energy efficiency and proportionality of the proposed transfer scheme in long-term image acquisition application.
- Experimental validation of the trade-off between energy cost per image stored and required energy storage capacity.

3.1.4 Roadmap

The remainder of this chapter is organized as follows. In Section 3.2, we will give a detailed overview of the state of the art in batteryless and state-retentive systems, as well as

external non-volatile memory technologies. The model and architecture of the EMU are presented and discussed in detail in Section 3.3. The main application scenario, long-term logging of images, as well as the architecture of our baseline vision sensor is explained in Section 3.4. Our novel energy-efficient Non-Volatile Memory Hierarchy (NVMH), composed of both Flash and FRAM memories is presented in Section 3.5. The optimized system design, which includes the NVMH for long-term logging, is described in detail, along with its design trade-offs, in Section 3.6. The experimental evaluation of our different load configurations and analysis of the energy efficiency and proportionality are shown in Section 3.7. Lastly, we summarize our work in Section 3.8.

3.2 Related Work

Cyber-physical systems have traditionally been used in conjunction with energy harvesting and energy storing. When coupled with aggressive duty-cycling techniques, they are able to significantly reduce their average power consumption, possibly to the point of self-sustainability. Due to the prohibitive costs of storing energy, there is a new trend to design systems with minimized storage capacity. As a consequence of this limited capacity and the variability of energy harvesting, Non-Volatile Memory (NVM) is required to ensure data consistency. For long-term logging applications, this poses a challenge to store large amounts of data. While there are many different memory technologies, each has its own characteristics in terms of storage density, read/write cycles, and power consumption. In this section, we will summarize the state of the art of transient systems and non-volatile memory technologies.

3.2.1 Reliable execution

The authors of [NPK+15] propose a decoupling-based system which utilizes RF harvesting to accumulate charge in a

supercapacitor and then powers a camera application with a buck converter. The boost/buck converter topology with an energy buffer also serves as the basis for the approach presented in this work. While a charge-state model is used to characterize the capacitor's self-discharge rate, energy losses such as impedance matching and converter inefficiencies are neglected. More importantly, the system has a large startup cost and can only supply the load with bursts of a constant size and voltage. Due to the controllability of the RF source, this startup cost is not incurred often. In fact, this constant energy availability is one of the key differences to our work, which focuses on adverse environments that can stop generating energy at any point in time. Furthermore, energy might not be available for prolonged periods of time, leading the system to inevitably discharge down to zero Volts. In Section 3.7, it will be shown that constant bursts can lead to a substantially higher energy consumption, larger storage elements and longer start-up times.

In the previous chapter, we presented the feedback-based DEBS technique (Section 2.4). Assuming the system has available non-volatile memory, applications can be split up into multiple bursts. Though this might incur some state retention overhead, the required energy storage can be significantly reduced by up to the number of bursts. Additionally, when tasks operate at different voltages, DEBS can introduce large energy savings by dynamically adjusting the supply voltage. However, when single tasks like storing or transmitting incur a high initialization cost, it is not energy efficient to have many activation cycles. In this chapter, we will introduce the NVMH to reduce the average cost of accessing a microSD card.

3.2.2 State-Retentive Systems

To have efficient long-term data retention, a system necessarily needs non-volatile elements. Fully non-volatile systems are those where both logic and data memory are built exclusively from non-volatile elements. While these architectures have guaranteed data retention by definition, there are many differ-

ent trade-offs related to the low-level technology. Compared to standard CMOS technologies, non-volatile devices exhibit reduced performance, high power consumption and larger die area [MV16]. Furthermore, novel manufacturing processes are expensive, not yet scalable and can suffer from reduced reliability [TBCS13]. Nonetheless, the promise of efficient, fine-grained duty-cycling with zero leakage has received a lot of attention. Researchers have explored many different technologies to design fully non-volatile processors, for a survey we recommend [Xie16].

Recent designs based on FRAM include [BKC+13, WLL+12] and feature very low energy per read or write. Spin-torque transfer magnetoresistive RAM (STT-MRAM), which offers higher access speeds, has also been used to design non-volatile processors [STN+14, GIS10]. Magnetic-tunnel-junction (MJT) memory is yet another technology used for fully non-volatile architectures and has been successfully demonstrated using 90 nm CMOS/100 nm MJT processes [OMTH15]. All non-volatile designs are highly resilient to frequent power outages, even with tiny storage devices. Nonetheless, this resilience comes at an increased energy cost per instruction and is limited to processing tasks. Should larger energy storage be allowed, longer activity cycles could be guaranteed and volatile logic would become more energy efficient. For this reason, we will use digital systems with volatile logic and complement them with (additional) external non-volatile memories. In the particular case of the MSP430FR5969, there are 64 KB of on-chip FRAM which can be used for data storage, but it is too limited to hold pictures for long-term data logging, thus requiring high density external non-volatile memories.

3.2.3 External Non-Volatile Memories

In the design of sensing systems, one of the key considerations is the choice of Non-Volatile Memory (NVM). Due to the inherent power cycling of transient nodes, any data saved in volatile memory will be lost. Consequently, the choice of NVM technology is closely related to the application's power

envelope, reliability and storage requirements.

One of the most mature non-volatile technologies used in cyber-physical systems today is Flash Memory. While it offers one of the highest storage densities available, it suffers from two main drawbacks: power consumption and reliability. SD Cards alone, for example, can have a capacity in the order of 100's of GB and a power consumption in the range of 50-100 mW. Each memory block also has a restrictive read/write cycle limit of 10^4 - 10^5 [TBCS13].

For many years, researchers have been actively searching for new NVM technologies that can (ideally) offer ultra-low-power consumption, unlimited read/write cycles, ultra-high densities, and compatibility to standard fabrication processes. Unfortunately, no global optimum has been found yet, and commercially available technologies offer different trade-offs between these important parameters. We will focus on one specific technology, Ferro-electric Random Access Memory (FRAM), which is a promising candidate for unified (instruction and data) memory due to its high endurance (10^{15} read/write cycles), and its low power consumption [Tex15]. The main limitation of FRAM for long-term logging of rich data sensors is its capacity since the largest commercially available capacity is in the order of 100's of KB [Cyp15].

3.3 EMU-Based System Operation

In this section, we review the model and architecture of the Energy Management Unit (EMU), which will be the basis of our batteryless execution profile.

One of the main goals is to derive an analytical model which can be applied to a wide variety of energy sources and loads. The model will then be used to optimize important system parameters, namely the EMU's start-up costs and the load's energy. The accuracy of the model will be experimentally validated in Section 3.7.

3.3.1 Modeling Energy Buffering and Losses

The amount of energy buffered in the EMU depends on several parameters including the input power and load powers, and the system's non-idealities. The equation governing the time-dependent energy level in a capacitor is as follows:

$$E'_{cap}(t) = \frac{d}{dt}E_{cap}(t) = \eta_{boost}\left(V_{in}(t), I_{in}(t)\right) \times P_{in}(t) - P_{leak}(t) - P_{load}(S_j)/\eta_{buck}(V_{load}(t), I_{load}(t)) \quad (3.1)$$

In this equation, the positive term represents the energy intake, while the negative ones represent the energy consumption.

Input Power. The system has only one power input, $P_{in}(t)$, supplied by the harvester's transducer. This work focuses on the scenario where $P_{in} < P_{load}$. In order to maximize the transducer's efficiency, the maximum power point must be tracked to account for variable harvesting conditions.

Load Power. In our model, the load can have two states (S_j): active or inactive. When active, the load is characterized by three quantities: $E_{burst,i}, V_{load,i}, P_{load,i}$; where $E_{burst,i}$ defines the energy burst size required for one execution of task i, $V_{load,i}$ its supply voltage and $P_{load,i}$ the power consumption during the execution of task i. These parameters will be characterized experimentally. In the inactive state, the load is in deep sleep, consumes very little power, and awaits the trigger from the energy management unit.

Converter Efficiencies. Since decoupled systems can have the source and load operating at different power points, voltage converters are used. This step, while necessary, introduces non-negligible losses, which are represented by boost and buck converter efficiencies $\eta_{boost}(V, I)$ and $\eta_{buck}(V, I)$. The boost converter's efficiency is particularly sensitive to the operating voltage and current, meaning it must be parameterized. For these efficiencies, a look-up table is used for simulations. The overall system efficiency of the EMU will be bounded by the product of the boost and buck converter efficiencies. While this depends on both the input and output voltage/current, broadly speaking for our application domain, it goes up to ~75%, see our experimental results in Section 3.7.

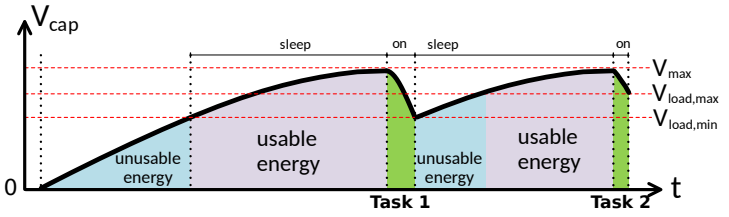


Figure 3.2: When the application consists of bursts at different supply voltages, the bands of energy usable by the load change according to the task.

Other Energy Losses Unfortunately, converter inefficiencies are not the only sources of energy losses. The maximum power point tracking unit and the control circuit also consume energy. The consumption of the control circuit I_{ctrl} and buck converter I_{buck} consists of a constant current and a resistive component and hence depends on V_{cap}. For the energy buffer, a capacitor of size C_{cap}, a resistive leakage R_{cap} in parallel is assumed. Considering these components, the system leakage is summarized as:

$$\begin{aligned} P_{leak}(t) = & V_{cap}(t) \times \left(I_{ctrl}\left(V_{cap}(t)\right) + I_{buck}\left(V_{cap}(t)\right)\right) \\ & + V_{cap}(t)^2 / R_{cap}. \end{aligned} \tag{3.2}$$

Equations 3.1 and 3.2 can accurately describe the time evolution of the system's energy levels, as will be shown in Section 3.7.4. They will be used in the remainder of this section to estimate how different parameters impact the system's losses, to then calculate the optimal parameters that minimize these losses.

3.3.2 Optimizing Cold-Start Energy and Start-up Time

Given the system model presented above, we can start optimizing the cold-start energy and start-up time. By definition, this is the fixed start-up cost to turn a transient system on. Figure 3.2 shows that after a period of energy unavailability, the capacitance first needs to be recharged to

the level of $V_{load,min} = \min_i\{V_{load,i}\}$. In order to minimize these fixed costs for a given input power, we need to minimize the start-up time. Assuming $V_{cap}(t = 0) = 0$, this start-up time is defined as:

$$t_{start\text{-}up} = \left\{ t \mid V_{cap}(t) = \sqrt{\frac{2\int_0^t E'_{cap}(\tau)\, d\tau}{C_{cap}}} = V_{load,min} \right\} \tag{3.3}$$

However, the minimum capacitance is limited by the EMU's maximum supported voltage swing, as shown in the following equation:

$$C_{min,i} = \frac{2E_{load,i}}{\eta_{buck}(V_{max}^2 - V_{load,i}^2)}, \tag{3.4}$$

where $E_{load,i}$ and $V_{load,i}$ are the energy and voltage required to execute task i, and V_{max} is the EMU's maximum supported voltage. These values must be known at design time, such that the optimal capacitor value can be selected as the highest $C_{min,i}$ among all tasks i, i.e. $C_{optimal} = \max_i\{C_{min,i}\}$. For the implementation we selected the next higher available capacitor size C_{buffer} to guarantee task completion. It should be noted that whenever the load has multiple operating voltages, a small phase difference will be introduced between the tasks. This is due to the fact that the bands of usable energy are different between tasks. This gap depends on the supply voltages, $V_{load,min}$ and $V_{load,max} = \max_i\{V_{load,i}\}$. This time is defined as:

$$t_{phase} = \left\{ t \mid V_{cap}(t) = \sqrt{\frac{2\int_{t_{start-up}}^t E'_{cap}(\tau)\, d\tau}{C_{cap}}} = V_{load,max} \right\} \tag{3.5}$$

3.3.3 Minimizing Load Energy

To show the advantages of our EMU's boost-buck architecture compared to the boost-only architecture, let us consider the case of supplying a constant current load, consuming I_{load}. The

harvesting power of a transiently powered system which is typically much smaller than the load's power consumption, therefore has a negligible impact on the linear voltage decrease during the time in which the load is supplied with an energy burst. Assuming the load has a maximum supply voltage tolerance from V_{max} down to V_{min}, this results in the following power consumption: for the boost-only architecture the average power of a task is $P_A = (V_{min} + V_{max})/2 \times I_{load}$, while the buck converter provides a constant power of $P_B = (V_{min} \times I_{load})/\eta_{buck}$ to the load. By comparing these two power consumptions, it directly follows that a buck converter reduces the load's power consumption if the following lower bound for its conversion efficiency holds:

$$\eta_{buck} > \frac{2V_{min}}{V_{min} + V_{max}} \tag{3.6}$$

To illustrate with a numerical example, suppose a load has a voltage tolerance of 3 to 5 V. This means that a system using a buck converter has a lower power consumption if $\eta_{buck} > 75\%$. Furthermore, the use of a buck converter adds the possibility of tracking the load's optimal power point for all tasks by dynamically switching the voltage level. When an application consists of multiple tasks with different voltage requirements, we can use Dynamic Energy Burst Scaling (DEBS) to minimize the load's energy.

3.3.4 EMU Architecture

The Energy Management Unit (EMU) controls the buildup of energy from the source, and controls the energy transfer to the load, or application circuit. A brief summary of the main components will now be discussed.

Harvesting and Buffering

The harvesting part of the system is based on the commercial BQ25505 energy harvesting chip. This chip uses a boost converter to accumulate charge on buffer capacitor. In doing

so, both the transducer's voltage and current are independent from the capacitor's voltage and discharge current. The BQ has integrated maximum power point tracking (MPPT) which adjusts the input impedance such that the power source always operates at its optimal power point to maximize the harvested energy. This is done in a duty-cycled fashion, by sampling the open circuit for 256 ms voltage every 16 s. Although this means it won't be able to harvest energy 1.6% of the time, it will be able to track dynamic environments.

Since application circuits need a regulated voltage to operate efficiently, we selected the TPS62740 buck converter for its high efficiency in a wide current range. The input of the buck converter is directly connected to the energy buffer to provide a single, regulated voltage domain. Though it is possible to use multiple converters to create separate voltage domains [DXS+18], we argue that only one is necessary to execute batteryless applications in a cost-efficient manner.

The energy buffer between the input voltage boosting and output voltage regulation guarantees complete separation of the harvesting and load supply unit and therefore allows independent optimization of these parts. As was shown in Chapter 2, minimizing the storage element is an application-specific process. In this chapter, we will show that two different implementations of the same application can exhibit a trade-off between storage size, and the amount of work completed with the same energy budget.

Control Circuit

The control circuit manages the burst size as well as the output voltage and oversees the energy accumulation in the buffer. For the first, the battery OK signal of the bq25505 is used to trigger the activation of the load, once the capacitor voltage reached a threshold level V_{th}. At this voltage, the energy level is reached at which enough energy has been accumulated to provide the requested energy burst to the load. The variable burst size dependent threshold voltage V_{th} is configured using a resistor network. This comparator threshold can be switched

digitally from the control circuit by selecting between different resistor networks. Besides very large resistor values, the bq25505 control circuit uses duty cycling to reduce the energy consumption of the comparator and resistor network. The load supply voltage V_{load} can be controlled directly using the TPS62740 buck converter's digital input.

Requirements for EMU Operation

Thanks to its inherent decoupling of source and load power points, the minimum requirements for EMU operation are conceptually independent from the load and are only tied to the EMU's circuit implementation. In our case, the first requirement is a minimum input voltage of 330 mV, which is required to turn on a diode in the BQ harvester. If this requirement is met, charge is transferred from the solar panel to a small capacitance. After a certain voltage on this capacitance is reached, the main boost converter is turned on. This transition requires a minimum input current of ~60 μA. This means that so long as the input power is greater than ~20 μW, the EMU is guaranteed to exit the cold-start phase and enter the energy build-up phase. During this phase, the charge on the capacitor will increase as long as $\overline{P_{in}} > P_{leak,max}$, where

$$P_{leak,max} = \left\{ P_{leak}(t') \mid V_{cap}(t') = V_{max} \right\}. \tag{3.7}$$

After some time, which depends on the input power, enough charge will be accumulated in the capacitor to guarantee a task completion. This makes only the frequency of task activations dependent on the harvesting conditions, but not the task itself. Once triggered, the actual task execution is guaranteed regardless of the harvesting conditions.

3.4 Long-Term Logging of Rich Data Sensors

Up to now, we have discussed how energy can be efficiently buffered, even in low power harvesting scenarios. How this

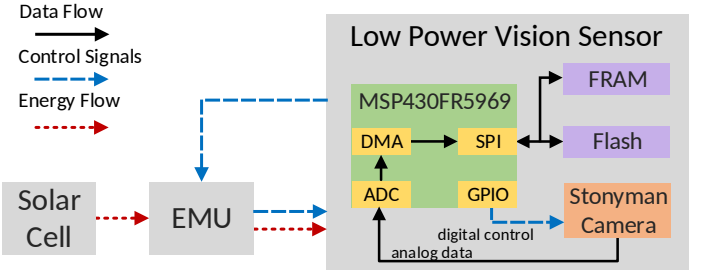


Figure 3.3: Architecture of the proposed batteryless vision sensor. Using a solar cell and an EMU, the MSP430FR959-based vision sensor can capture and store images in different non-volatile memories.

energy is consumed, however, is a function of the application circuit itself. To highlight the flexibility and efficiency of our approach, we will focus on nodes which do long-term logging of rich data sensors. More specifically, we focus on batteryless vision sensors, which acquire and store images in Non-Volatile Memory (NVM).

3.4.1 Sensing System Architecture

The architecture of our proposed system can be seen in Figure 3.3. The transiently powered vision sensor is composed of an MSP430FR5969 microcontroller and a Centeye Stonyman image sensor, both of which feature low-power consumption and ultra-low-power deep sleep modes. The IO state lock mechanism and the microcontroller's non-volatile features are important to keep the interface state of the energy manager during deep sleep and maintain the task configuration across periods of energy unavailability. In order to store long-term data, external memories with high capacities are needed. One of the most commonly used technologies for this purpose is an SD Card, which is based on Flash. As will be highlighted in Section 3.5, SD Cards generally suffer from very high power and energy consumption but offer the highest densities.

3.4.2 Energy Burst Configuration

In order to configure the EMU for correct operation, it is important to characterize the application's voltage and energy needs. For our rich data sensor logger, the baseline application consists of three tasks: 1) image acquisition to read the sensor, 2) basic image processing, and 3) image storage to copy data from volatile to non-volatile memory. Table 3.1 shows the energy burst configuration (V_{load}, E_{burst}) for each task. As was previously mentioned in Section 3.3.2, the EMU's capacitor has to be dimensioned according to the largest task: transferring an image to the SD Card. The energy cost for this transfer has two components, a constant initialization cost and a transfer cost which depends on the amount of data transferred. For the baseline application using DEBS, which acquires, processes and transfers a single image, the costliest task is the image transfer which requires 11.67 mJ. The minimum available capacitance that can store this energy between 5.1 V and 2.7 V is 1 470 μF. If DEBS is not used then all tasks would need to execute in a single burst at a constant voltage, requiring a minimum capacitance of 2 200 μF.

Task	Voltage (V_{load})	Energy (E_{burst})
Acquire One Image	3.0 V	156 μJ
Process One Image	2.0 V	527 μJ
SD Card Initialization	2.7 V	10 536 μJ
SD Card Data Transfer (per Image)	2.7 V	1 137 μJ

Table 3.1: Baseline voltage requirements and energy costs for individual task execution.

3.4.3 Software Execution Flow

The execution flow of an EMU-based sensing system is shown in Figure 3.4. Once the system has exited cold-start, after 2 V, the Power-On-Reset (POR) flag is checked. Since the reset was not triggered from the EMU pin, the microcontroller performs some basic initialization and immediately enters deep sleep. With a measured power consumption of < 600 nW, deep sleep minimizes losses during the buildup of energy

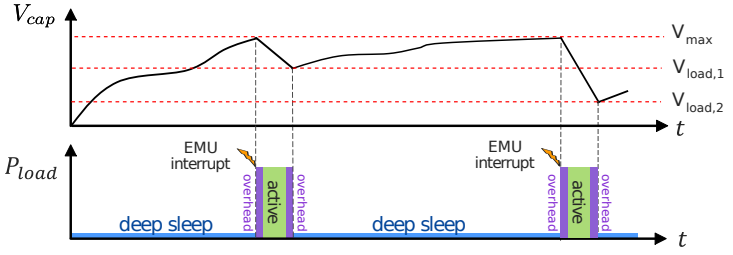


Figure 3.4: Sample execution with an arbitrary V_{cap} trace. Once $V_{cap} = V_{max}$, the application circuit can be triggered for one activation cycle, incurring some overheads for booting up, initializing and shutting down.

for the next burst. The system will stay in this state until the EMU has accumulated enough energy to reliably execute a task. When this happens, the system then reads the next task configuration and starts its execution after initializing the peripherals needed for that task. There are some unavoidable energy overheads involved in the peripheral initialization since this might include a power switch enabling the peripheral power domain. At the end of the task, the configuration is updated, unnecessary power domains are disabled and the next required burst is configured. The load then enters deep sleep again and waits for the next energy burst to build up. For all of our evaluated applications, all of the tasks will be repeatedly executed in a static schedule. For example, the baseline application will repeatedly execute the following tasks: 1) acquire one image, 2) process one image, 3) store one image on the SD Card.

3.4.4 Feedback Control for Dynamic Energy Burst Scaling (DEBS)

As was discussed in Section 3.3.3, there are many application scenarios where the load has a varying optimal power point. This occurs when tasks use peripherals with substantially different voltage requirements. For this scenario, the EMU provides a control interface to dynamically adjust the burst size

and voltage. Our DEBS technique is based on a feedback loop (Figure 3.1) that allows the load to configure the EMU to supply the energy burst at the optimal operating point. Following our baseline image acquisition example, when using DEBS the EMU generates three bursts, one for each task. During the first burst, 156 μJ at 3 V were requested. Once enough charge has been built up, the EMU's control circuit configures the buck converter's digital input to set the output to 3 V and triggers the load to acquire the image. Afterward, the load uses the EMU's interface again and requests the second burst (image processing) by setting the energy and voltage to 527 μJ and 2 V, respectively. Lastly, at the end of the second burst, the load requests the third burst for storing an image with energy and voltage set to 11.67 mJ and 2.7 V, respectively. So long as the EMU's buffer has energy, the buck converter will maintain this output voltage until the next burst is generated, the next task executed, and the load requests the next energy burst size and voltage.

Without DEBS, the EMU would only be able to generate bursts at a constant voltage of 3 V. This results in an approach similar to the one proposed in [NPK+15], where one large burst would be used to acquire, process, and store one image. This approach leads to significantly larger burst sizes due to the grouping of tasks with a non-optimal operating point. These two approaches, single burst and DEBS based bursts, will be evaluated experimentally in Section 3.7.

3.5 Non-Volatile Memory Hierarchy

In order to make long-term logging of rich data sensors viable, the energy cost per unit of data stored needs to be as small as possible. To this end, we propose the use of a novel Non-Volatile Memory Hierarchy (NVMH) that combines the best characteristics of different technologies. The use of a NVMH is independent from Dynamic Energy Burst Scaling (DEBS). The former is an application-side optimization to reduce the cost of storing data to non-volatile memory, while the latter is EMU-

based feedback technique to split a single transient application into task-based burst executions. Over the following sections, we will discuss how to combine both techniques to optimize the system design of a transient image sensor for long-term logging applications. The insights presented here were obtained in collaboration with Lukas Sigrist and Thomas Schalch.

3.5.1 Non-Volatile Memories (NVM)

As was discussed in Section 3.2.3, Flash is the most mature NVM technology available and boast very large storage capacities, but suffers from high energy consumption. Contrarily, Ferro-electric Random Access Memory (FRAM) offers ultra-low-power/energy consumption, but is limited to very low storage capacities.

Flash

While Flash memories can have a very large capacity (up to 100's of GB), they have a high power consumption during the initialization and write phases. The time-consuming initialization procedure results in a very high constant energy overhead as it is needed before the actual memory can be accessed. In burst-powered transient systems, this initialization has to be paid for every energy burst that requires SD Card access and can lead to prohibitively large overheads. Furthermore, Flash memories alone have low durability (around 100 000 read/write cycles) and typically use a controller to spread the wear of individual memory cells evenly, also called wear-leveling [HXZ+13]. Unfortunately, wear-leveling leads to additional delays and energy consumption and introduces additional variability in the total overhead.

FRAM

In recent years, FRAM has emerged as a viable alternative with very low-power consumption. FRAM provides ultra-low-power read and write actions at high read and write speeds. In

addition to high energy efficiency, FRAM also provides very high durability. Unfortunately, due to the incompatibility with standard manufacturing processes, FRAM components have relatively low storage capacities (~ 100's KB).

In the following section, we show how these two memory technologies can be combined in long-term logging applications to exploit the advantages of each technology: Flash's high density and FRAM's energy efficiency.

3.5.2 Non-Volatile Memory Hierarchy (NVMH)

Novel NVM technologies have not reached densities that allow them to replace Flash in data-intensive applications. However, since they consume significantly lower energy than traditional Flash, they open the door to reducing the energy requirements of long-term logging.

We propose to use energy efficient FRAM to increase the efficiency of transiently powered logging applications that rely on an SD Card as a large storage device for long-term logging. More specifically, we introduce a small FRAM data cache before the SD Card to distribute its high initialization cost among several image transfers. As shown in Figure 3.3, this can be added to the sensor node as an additional component on the SPI bus. Thanks to the ultra-low energy read and write operations in FRAM, multiple images can be cheaply buffered before transferring them in one single batch to the SD Card. Compared to writing the images one by one on the SD Card, as described in Section 3.4, the high SD Card initialization cost is required only once for writing *all* images buffered in the FRAM to the SD Card. This reduces the SD Card initialization cost by a factor determined by the FRAM buffer size. Because of the high energy overhead for initialization, this non-volatile memory hierarchy allows significant reductions in the energy needed to acquire, process and store one image. The experimental results described in Section 3.7 will show that on average 294.8% more images can be stored in Flash when the baseline application discussed in Section 3.4.2 uses a 10 image FRAM buffer, compared to the same application without it.

It is important to note that the non-volatility property of the FRAM is a key requirement in this memory hierarchy, because this buffer is only supplied with energy when an acquired image is buffered or the full buffer is flushed to the SD Card. In between bursts with memory access, it is powered off to reduce leakage losses. Deploying any volatile component in the memory hierarchy, like SRAM, is not be suitable for a transiently powered system because the content would be lost at the point where the memory is turned off, or when the system enters cold start due to low input power.

The proposed memory hierarchy combines the advantages of low energy consumption available in novel NVM technologies with the high density of traditional Flash. This allows building long-term rich data sensor logging applications with large storage requirements and high energy efficiency.

3.5.3 Energy Cost vs Minimum Capacitor Size Trade-Off

The key design parameter of the proposed memory hierarchy is the size of its intermediate FRAM buffer. The larger the buffer, the more images can be buffered before flushing them to the SD Card. By distributing the initialization cost of the SD Card, the average energy requirement per image decreases. The energy overhead for reading and writing the image to the FRAM buffer is negligible compared to the SD Card: even with a buffer of only 2 images, the energy saved for one initialization is larger than the additional energy cost for the FRAM. However, because of the larger transfer size from FRAM to the SD Card during the flush operation, the amount of energy that needs to be guaranteed by the EMU also increases. This results in a trade-off between the required energy buffer size and energy cost of the storing data in NVM. In this work, which focuses on long-term logging of rich data sensors, our primary concern is minimizing the energy cost of non-volatile data. How one can select the optimal parameter values at design time will be discussed in Section 3.6.1.

3.6 Optimized System Design

In this section we demonstrate how the Energy Management Unit (EMU) model with Dynamic Energy Voltage Scaling (DEBS) presented in Section 3.3.1 is used during the design of a transient system for determining individual system parameters. For that, we use the following two use-cases: 1) selecting the FRAM size in the Non-Volatile Memory Hierarchy (NVMH), and 2) dimensioning the solar panel for a desired application performance.

3.6.1 FRAM Buffer Size and Cost/Capacitance Trade-Off

As was previously mentioned, the main focus of this work is to minimize the total energy cost E_{img} required to acquire, process and store image data in non-volatile memory for long-term logging applications. It has already been shown that for generic applications, using DEBS reduces the load's energy requirements through task-level optimizations. Now we will focus on optimizing the non-volatile storage component of E_{img} through the use of both DEBS and NVMH. To design the NVMH, we will first select the FRAM buffer size, since this has a direct impact on the energy cost reduction. As a second step, we will select the value for the capacitance such that all other costs like form factor and start-up time/energy are minimized.

As long as Flash memory remains the cheapest and densest technology available for embedded systems, it will always be required in long-term image logging applications. In the transient application scenario, which involves duty-cycling the SD Card, it is inevitable to pay the SD card's initialization cost before transferring any data. Since this initialization cost is constant, the main purpose of the NVMH is to divide this cost, on average, over as many images as possible. Because of this, the first relevant parameter is how large the FRAM buffer can be. Currently, the main limitation is technology scaling, which has so far prevented FRAM to have capacities greater than 100's KB. At the time of writing, the largest commercially available FRAM can only buffer 10 images from our vision sensor.

Once the FRAM buffer size has been determined, the only remaining question is how small the buffer capacitance can be such that the energy savings are as high as possible for a given FRAM buffer size. To minimize the capacitance, it is important to understand how much energy is needed to flush the entire FRAM buffer to Flash. To simplify the discussion, we assume that once the FRAM buffer is full, the application schedule ensures the buffer is fully flushed before acquiring new pictures. Depending on the capacitor size, which limits the maximum burst size, flushing the entire buffer might take one or more bursts. The data-dependent energy cost to transfer a block of one or more images from the FRAM to the SD card is given by

$$E_{trans}(N_{img}) = N_{img} \times (E_{FRAM,img} + E_{Flash,img}), \tag{3.8}$$

where N_{img} is the total number of images that can be buffered in FRAM before being flushed to the SD card and $E_{x,img}$ is the data-dependent cost of storing one image in memory X. As mentioned earlier, if the EMU buffer capacitor is not big enough to guarantee that the FRAM can be flushed in one single burst, then several bursts will be needed for the complete data transfer. The number of bursts N_{burst} required to complete the FRAM flush can be calculated as

$$N_{burst}(N_{img}, E_{buff}) = \left\lceil \frac{E_{trans}(N_{img})}{E_{buff} - E_{mem,init}} \right\rceil, \tag{3.9}$$

where $E_{mem,init}$ is the initialization cost of both FRAM and Flash memories, and $E_{buff} = \frac{1}{2} C_{buff}(V_{max}^2 - V_{min}^2)$ is the energy stored in the EMU buffer capacitor C_{buff} between the flush task's lowest operating voltage of V_{min}=2.7 V and the EMU's highest buffer voltage of V_{max}=5.1 V. UsingEq. (3.8) and Eq. (3.9), it is possible to calculate the total energy E_{flush} required to flush the FRAM buffer of N_{img} images as

$$\begin{aligned} E_{flush}(N_{img}, E_{buff}) =& N_{burst}(N_{img}, E_{buff}) \\ & \times E_{mem,init} + E_{trans}(N_{img}) \end{aligned} \tag{3.10}$$

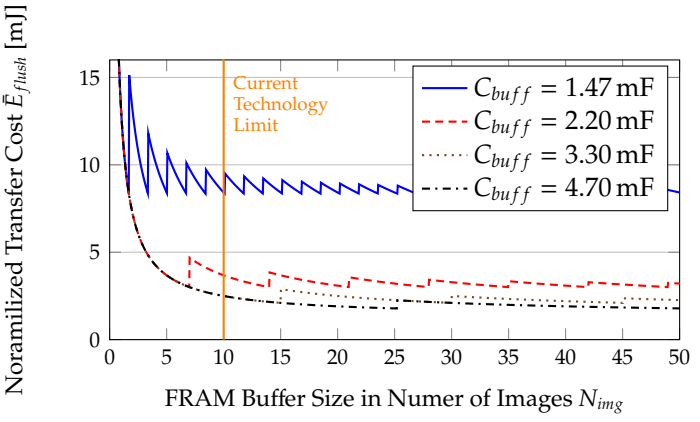


Figure 3.5: DEBS+NVMH simulation results show how a given FRAM buffer size can have very different energy cost per image stored if the capacitor size is not chosen appropriately.

Figure 3.5 plots the normalized energy per image stored transfer cost $\bar{E}_{flush} = E_{flush}/N_{img}$. Each line in the plot represents a specific EMU configuration with a different C_{buff}. As discussed earlier, this has a direct influence on the number of bursts required to flush the FRAM buffer. For a given capacitance value, the image cost decreases monotonically up until the point where the data flush needs to be split into more than one burst. This step increase in $E_{flush,img}$ is introduced by the need of an additional Flash initialization for the subsequent bursts and decreases as the transfer size becomes multiple of the specific buffer size. From Eq. (3.10) it follows that all E_{trans} minima of a given C_{buff} are equal, since the NVMH mechanism amortizes the SD Card initialization cost by the same ratio. The optimized C_{buff} is the minimum capacitance that has not reached its first minimum for a given FRAM buffer size, since any larger capacitance will not introduce higher savings but would require larger start-up costs.

Optimized Burst Configuration

As mentioned earlier in Section 3.4.2 for the baseline DEBS application, it is important to characterize the application's voltage and energy needs in order to guarantee correct operation of the EMU. With the introduction of the Non-Volatile Memory Hierarchy (NVMH), the application scheduling needs to be modified to 1) image acquisition to read the sensor, 2) basic image processing, 3) buffer processed image in the FRAM, and 4) when the buffer is full, flush the entire buffer to the SD Card. With respect to the energy costs presented in Table 3.1, there is only the additional task of buffering one image in FRAM. This task was characterized to require 32 μJ at 2.0 V. Since the energy cost of transferring data to the SD card has a data proportional term, the new cost of writing to Flash is much larger than without NVMH. This is because a new transfer of 10 images is needed to flush the entire FRAM buffer, instead of transferring a single picture to Flash. Compared to the DEBS Only configuration, the minimum capacitance increases from 1 470 μF to 3 300 μF, while the energy cost per stored image was lowered from 11.11 mJ to 2.73 mJ when using DEBS+NVMH. This is the fundamental trade-off presented by NVMH: significantly reduced energy costs of stored images at the expense of a larger capacitance.

Design Space

So far, three critical parameters for a transient vision sensor with NVMH have been discussed: FRAM buffer size, EMU buffer capacitance, and energy cost per image stored. Figure 3.5 shows that when using DEBS+NVMH, the minimum energy cost and start-up time/energy is achieved when the FRAM buffer is maximized and the capacitance minimized for that buffer size. Figure 3.6 shows the energy cost per image stored vs the minimum capacitor size of all four configurations in a Pareto plot. It should be noticed that whenever DEBS is applied to a baseline configuration, the result is a reduced capacitor *and* a reduced energy cost per image stored. In fact, the Pareto front consists of all configurations that employ DEBS. By contrast,

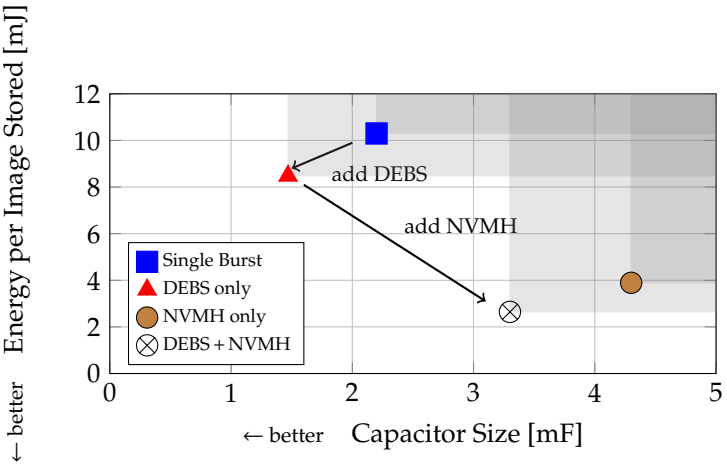


Figure 3.6: Pareto plot of the trade-off between minimal buffer size and energy used to acquire, process and store an image. Regions dominated by the shown design points are grayed out. The design goal is to minimize both the energy per image stored and the capacitor size.

whenever NVMH is added to a baseline application, the energy cost per image stored always decreases, but at the expense of a larger buffer capacitor. This corresponds to the expected behavior from previously presented models and highlights the fact that there is a trade-off between energy per image stored and buffer capacitance.

3.6.2 Minimum Harvester Size

In order to calculate the necessary size of the solar panel for an application, it is important to determine the desired performance. Depending on the lighting conditions the sensor node will be exposed to, the size of the solar panel might need to be bigger or smaller. Simulation can be used to optimize the solar panel area.

Four different PowerFilm flexible solar panels with sizes

ranging from 12.7 mm×64 mm to 37 mm×114 mm, with corresponding areas from 8 cm^2 to 42 cm^2, were considered as harvesting sources. We first evaluated the output power of the individual solar panels for illuminance levels in the range from 100 lx up to 2000 lx. To do so, the solar panels were connected to the bq25505 harvester with maximum power point tracking (MPPT). The buffer capacitor of the harvesting circuit was replaced by a source meter (Keithley SMU 2450), which was configured to keep the buffer voltage constant at the expected average capacitor voltage of 4 V. For measuring the actual power extracted from the solar panel a small shunt resistor of 10 Ω was inserted between the solar panel and the bq25505 to measure the current. The power was then calculated using the measured current and the solar panel voltage.

The input power derived from these measurements was then used to simulate the application's behavior under the given constant illuminance using the model presented in Section 3.3.1. For this use-case, the application configuration that includes both Dynamic Energy Burst Scaling (DEBS) and the Non-Volatile Memory Hierarchy (NVMH) with a FRAM buffer size of 10 images was used in the model. The application was simulated for a time window of one hour and the output is then analyzed to determine the average number of images that can be acquired, processed and stored per minute.

In Figure 3.7, the results of these simulations are shown for solar panels with different areas A_{panel} depending on the illuminance level. It can be seen that for a typical indoor illuminance (<1000 lux) a solar panel with an area of 42 cm^2 can acquire, process and store up to 11 images per minute. The linear relationship between application performance and the solar panel area is also clearly visible. Considering an example where the expected room illuminance is 750 lux and the application should acquire 6 images per minute (one every 10 seconds on average), the results show that the solar panel with A_{panel} = 42 cm^2 should be used to get the desired performance. This is also the solar panel that will be used for experimental evaluation in Section 3.7.

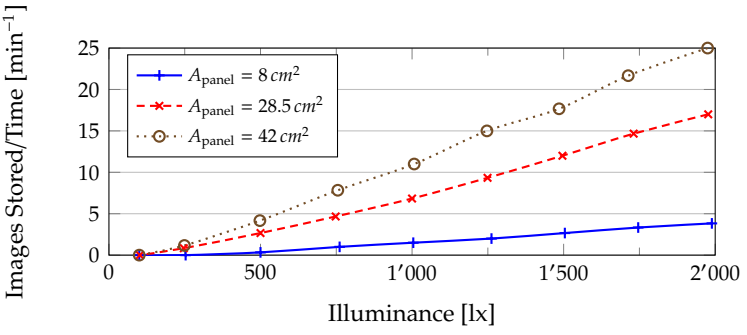


Figure 3.7: The application's performance is shown as the average number of stored images per minute vs the illuminance level for different solar panel areas A_{panel}.

3.7 Experimental Evaluation

This section evaluates the costs, performance and efficiency of an EMU-based transient vision sensor in four different configurations. These configurations represent all the possible combinations of our two main contributions: Dynamic Energy Burst Scaling (DEBS) and Non-Volatile Memory Hierarchy (NVMH). We will compare the performance vs efficiency trade-offs that these different combinations introduce. For the purposes of our sample life-logging application, our main goal is to minimize the energy costs/requirements since it will allow the transient camera to acquire, process and store the greatest number of pictures during a day.

3.7.1 Experimental Setup

The performance of our wearable prototype will be tested with four different configurations. Experiments will be done in both controlled (constant) and real-world (variable) harvesting conditions. All evaluation results shown in this section were done with the same MP3-37 flexible solar panel from PowerFilm, which has an area of 42 cm^2. For the analysis of the

system and application performance, all the relevant voltages and currents in the source, EMU and load were measured. To compare the experimental and simulation results, the input power traces were also recorded and used as input to our Matlab model.

In order to identify the effect of both DEBS and NVMH, different configurations will be individually tested for prolonged periods of time. The performance of each configuration, under different harvesting conditions, is measured, compared and contrasted. Their main characteristics are summarized in Table 3.2.

Single Burst This configuration is the baseline for all comparisons. It buffers the energy for one entire application execution and does it in a single burst with constant voltage. This means that within a single burst, one picture is taken, processed and stored in Flash directly.

DEBS Only This configuration, described in Section 3.4.2, uses Dynamic Energy Burst Scaling for each task. This means that the first burst does acquisition, the second does processing, and the third saves to Flash. Each of these bursts is configured to its optimal voltage.

NVMH Only This configuration introduces the Non-Volatile Memory Hierarchy (NVMH) with a FRAM buffer size set to ten, as was described in Section 3.6.1. This means that, conceptually, each burst executes an application cycle. The first 9 bursts write only to the FRAM buffer: each burst doing acquisition+processing+buffering. The 10th burst performs an additional SD card flush, which transfers the ten buffered images from FRAM to Flash.

DEBS + NVMH This configuration combines the previous two, but with one task executed per burst. This means that it takes 30 bursts (acquisition, processing and buffering 10 times each) to fill up the FRAM buffer. One additional burst transfers the buffered data to Flash. Again, each burst is executed at its optimal voltage.

Configuration Name	Burst Size	Execution Profile	DEBS	NVMH
Single Burst	$\sum_i E_{task,i}$	Entire Application	X	X
DEBS Only	$\max_i\{E_{task,i}\}$	Single Tasks	✓	X
NVMH Only	$\sum_i\{E_{task,i}\}$	Entire Application	X	✓
DEBS + NVMH	$\max_i\{E_{task,i}\}$	Single Tasks	✓	✓

Table 3.2: Description of the evaluated transient configurations.

Performance Metrics

In order to compare the performance of different transient configurations, the following metrics are calculated for all experiments:

- $E_{in} = \int_0^{T_{exp}} P_{in}(t)\, dt$, for the total input energy,
- $E_{app,j} = \sum_{i=1}^{N_{tasks}} \int_{t_{active,i,j}} P_{load}(t)\, dt$, for active energy consumed by the j-th application execution,
- $E_{load} = \sum_j E_{app,j}$, the total energy consumed by the load for all application executions,
- $\eta_{sys} = E_{load}/E_{in}$, the total system efficiency,
- $E_{image} = E_{load}/N_{image}$, the average energy cost for acquiring, processing and storing one image in NVM, and
- $\Theta_{app} = N_{image}/T_{exp}$, the average number of images acquired, processed and stored per time.

In the formulas above, $t_{active,i,j}$ denotes the execution time of task i in j-th application execution, N_{tasks} the number of tasks in the application, and N_{image} the number of images acquired, processed and stored in Flash memory during the experiment of duration T_{exp}.

3.7.2 Start-Up Time and Cold-Start Trade-Offs

As was discussed in Section 3.3.2, each configuration requires a different minimized capacitor that guarantees the completion of its largest atomic task. This minimized capacitor in turn

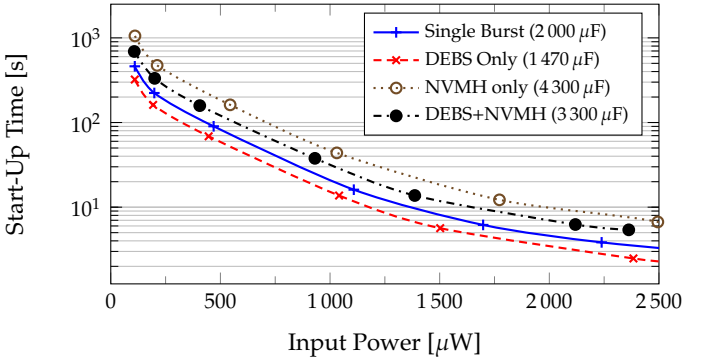


Figure 3.8: The cold-start time vs the input power for the EMU with different storage capacitor sizes.

minimizes the required energy and start-up time for each configuration's cold-start. To characterize these costs that occur after an input power loss and depletion of the buffer capacitor, the buffer capacitor was completely discharged, and the flexible solar panel was exposed to constant illuminance level until the cold-start phase ended. The time measured to go through this cold-start phase as a function of the input power is shown in Figure 3.8 for all four configurations. As expected, the start-up time for all configurations decreases with higher input power. More specifically, a maximum start-up time of 1055 s for the NVMH configuration (C_{buff} = 4 300 μF) was reached at the minimum measured input power of 110 μW. For an input power higher than 1 000 μW, all start-up times decrease to values below 44 s.

The start-up time overhead analysis shows the need for a minimized buffer capacitor, since this also minimizes the time/energy overhead of a given configuration. So long as the harvesting scenario provides enough energy, the transient node can exit cold-start and begin executing its application. In our specific life-logging scenario, we assume that typical human activities satisfy the requirements for EMU operation and last long enough for the transient vision node to start storing images.

3.7.3 Constant Input Power

In this part of the evaluation, the solar panel was exposed to a constant illumination level, resulting in an energy management unit with constantly supplied power. The experiments lasted for 10 min. For low input power levels of 200 μW and lower this time was extended to 15 min to observe a sufficient number of application triggers. For each application configuration, *Single Burst*, *DEBS only*, *NVMH Only* and *DEBS + NVMH*, the experiment was repeated for constant power levels ranging from 145 μW up to 1875 μW. Measuring the currents and voltages at the EMU's input, output and buffer capacitor, as well as the load supply, the system's state and energy flow can be tracked to later calculate the performance metrics of the experiment. With these measurements, the previously introduced metrics system efficiency η_{sys} and the number of images stored per time Θ_{app} were calculated. The results of these metrics are analyzed depending on the different input power levels and discussed in detail in the following sections.

System Efficiency

The analysis of the system efficiency η_{sys} is shown in Figure 3.9 for the four configurations mentioned earlier. The results show a consistent behavior: despite small variations between individual configurations, they all show a system efficiency η_{sys} that reaches at least 70% when the input power is greater than 1 000 μW. At the higher end of the evaluated input power the efficiencies asymptotically approach the maximum efficiency dictated by the product of the boost and buck converter efficiencies. During the experiments, the highest observed efficiency reached a value of 78.6% at 1 875 μW, the highest input power for which the system was evaluated. Also common for all configurations is the fact that the efficiency drops sharply for input power levels close to the minimum required input power of 140 μW. During experiments below that minimum input power level, no task executions were observed, resulting in a system efficiency η_{sys} of 0%.

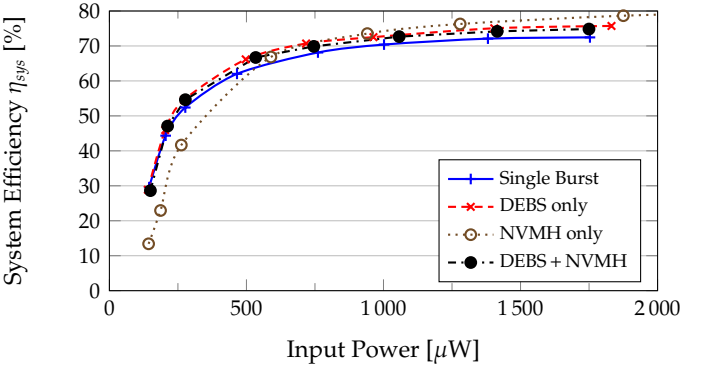


Figure 3.9: Evaluation of the EMU's system efficiency at different input power levels.

Number of Stored Images

While the results for the system efficiency are very consistent, a large difference can be observed in the application performance that is characterized by the average number of stored images per time Θ_{app}. The results for this metric as a function of the input power level are shown in Figure 3.10. The most noticeable difference is the increase in the number of images stored per minute when deploying *DEBS + NVMH* instead of *NVMH Only* or *DEBS only* instead of *Single Burst*: in either case, the inclusion of DEBS significantly increases the number of images stored per minute. In the case of DEBS Only, the increase was 30.1% at an input power of 1 750 μW, compared to Single Burst. Using the NVMH Only configuration offers a significant performance boost of 294.8% on average compared to Single Burst. It is the DEBS+NVMH combination, however, that clearly offers the highest number of stored images per time, thanks both of the proposed enhancements. The properties that all configurations have in common are the minimum input power of 140 μW and, once that power level is reached, the energy proportional increase of the number of stored images. Comparing the slopes of Θ_{app} for the individual configurations, it is visible that deploying DEBS instead of using the very basic

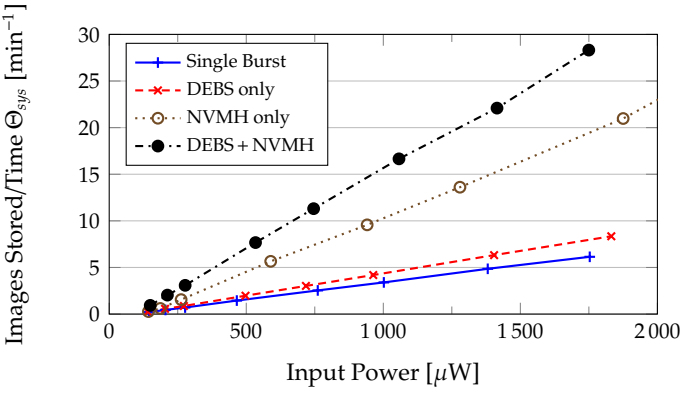


Figure 3.10: Evaluation of the average number of images stored on the SD Card per time at different input power levels.

Single Burst configuration already results in an improvement of 26% on average. Making use of DEBS+NVMH has the highest impact and increases the application performance by 268% or 365% when compared to the *DEBS only* or *Single Burst* configuration, respectively.

These experiments show the performance gain of deploying not only a dynamic energy burst scheme, but also an efficient memory hierarchy design for transiently powered logging applications.

3.7.4 Variable Input Power

The experiments discussed in this subsection were performed in an indoor real-world scenario, again for all three configurations. Each configuration was evaluated with a 15 min experiment that included walking around with the setup in the office hallway illuminated by artificial light, walking in a dimly lit basement and sitting at an office desk well illuminated by natural and artificial light.

The experimental metrics for *Single Burst*, *DEBS Only*, *NVMH only*, and *DEBS + NVMH* under variable input power conditions are shown in Table 3.3. The first thing to note is that

Configuration	Avg. P_{in}	C_{buffer}	Metric	Simulation	Experiment
Single Burst	731.94 μW	2 000 μF	Θ_{app}	2.39 min^{-1}	2.13 min^{-1}
			avg. E_{img}	12.06 mJ	12.58 mJ
			η_{sys}	63.51%	60.93%
DEBS Only	706.98 μW	1 470 μF	Θ_{app}	2.62 min^{-1}	2.62 min^{-1}
			avg. E_{img}	10.54 mJ	11.12 mJ
			η_{sys}	64.48%	68.59%
NVMH Only	663.93 μW	4 300 μF	Θ_{app}	5.96 min^{-1}	6.28 min^{-1}
			avg. E_{img}	4.28 mJ	4.35 mJ
			η_{sys}	63.99%	68.65%
DEBS + NVMH	607.71 μW	3 300 μF	Θ_{app}	7.78 min^{-1}	8.29 min^{-1}
			avg. E_{img}	3.27 mJ	2.79 mJ
			η_{sys}	64.48%	63.53%

Table 3.3: Results for variable input power experiments: average number of images stored per minute Θ_{app}, average of energy cost per image E_{img} and EMU efficiency η_{sys}.

adding DEBS to a baseline configuration reduces the average energy per image costs (E_{img}). This is the case for DEBS Only, which is an enhancement of the Single Burst configuration, as well as DEBS+NVMH, which is an enhancement of the NVMH Only configuration. This is expected since DEBS optimizes the load's operating point to minimize its energy requirements per task, simultaneously reducing the load's energy requirements as well as the minimum required capacitance. Comparing the buffer capacitances of the configurations without NVMH we see that the minimum required capacitor increases when NVMH is used. This is the fundamental trade-off of NVMH, and it is expected since the improved energy performance requires a larger energy guarantee for the SD Card flush task. Compared to the *Single Burst* configuration, the DEBS+NVMH configuration uses a 65% larger capacitor, but is able to reduce the average energy cost per stored image by 77.8%, down to only 2.79 mJ per image. It should also be noticed that even though the average input power during the *DEBS + NVMH* experiment was only 607.71 μW, the average number of images stored per minute was almost 4× that of Single Burst, up to 8.29 images per minute.

Table 3.3 also compares the experimental results to the model simulation that takes the measured harvested power

as input. Here, the comparison to experimental values shows that even in a real-world scenario with variable input power, the model is able to predict the system behavior with a maximum error of ~6% for most performance metrics. This fact is also reflected in Figure 3.11: it shows the input power, simulated and measured energy levels of the buffer capacitor during a 350 second sample time window of the *DEBS + NVMH* experiment. Besides a small time drift in the energy accumulation during very low input power, where not all effects can be represented accurately by our model, it tracks the buffer's energy level and bursts with high accuracy. This high accuracy results only in small deviation in the time diagram, despite the accumulation of simulation errors in the time domain.

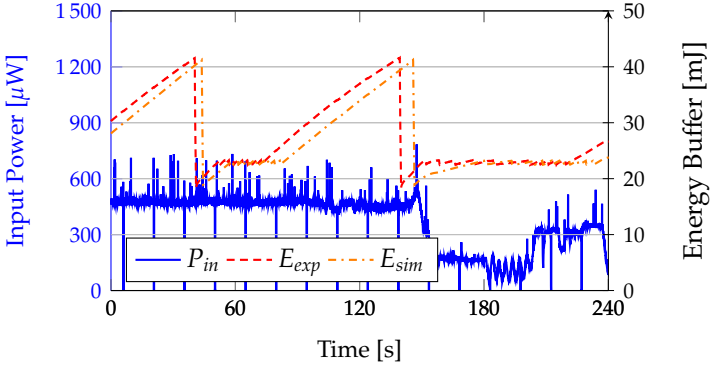


Figure 3.11: Time domain comparison between model simulation and experimental evaluation of image capture application using DEBS and NVMH under variable input power.

3.8 Summary

In this chapter, we have presented an EMU-based vision sensor that acquires and stores images in external non-volatile memory. By accumulating only the minimum amount of energy in an optimally-sized capacitance, the EMU is able to supply the sensing system reliably and efficiently, even when it harvests only a small fraction of the load's active power. Dynamic Energy Burst Scaling (DEBS) can be used with the EMU to track the load's optimal power point and minimize an application's energy, which in turn minimizes the start-up time and energy.

In long-term logging applications, where non-volatile memory can dominate the application's energy requirements, novel memory technologies such as FRAM can be introduced to form a Non-Volatile Memory Hierarchy (NVMH) that reduces the average energy cost of storing data. We show evidence of an important design trade-off between the energy cost per image and the minimum required capacitance. By adding only a ten-image FRAM buffer, 77.8% of the energy cost per image can be saved, though the minimum required capacitance grows 65%.

The EMU model can be used to dimension the solar panel to achieve a minimum performance for a given lighting condition. Experimental results show that a 42 cm^2 solar panel under indoor lighting conditions of 870 lux can be used to acquire, process, and store more than 11 images per minute on an SD Card. The EMU powered the 43.4 mW load at 69.90% efficiency requiring only 746 μW input power.

4

Reducing energy storage by partitioning

4.1 Introduction

In previous chapters, we have seen how the EMU can provide guaranteed energy bursts at specified voltages over a wide input power and voltage range. Furthermore, we have seen how different energy storage capacities can have an impact on important metrics like average energy cost per unit of work. We studied aggregating multiple transactions into a larger one can reduce the average energy cost. We will now study the same trade-off but instead of grouping multiple atomic transactions together, we will now decompose previously atomic tasks (i.e. data processing) into a set of smaller atomic kernels which can be executed separately without affecting the task's functionality. By partitioning applications with long processing tasks, we will ensure that they can be correctly and efficiently executed in wide range of energy storage capacities. Furthermore, the tools we present will be able to determine the minimum required energy storage capacity for reliable

execution, as well as the minimum application energy for a given energy storage capacity. As opposed to the methods presented in Chapter 2, the data processing task can be arbitrarily long and not dominate the size of the energy storage device. Using our specification model, sense-process-transmit applications can be optimized for burst-based execution such that truly atomic tasks like sensing or transmitting determine the lower bound for the energy storage capacity.

Recent work has identified two approaches for batteryless operation: atomic and state-retentive execution. Atomic execution of tasks [NPK+15, GSM+16, GSS+17b] relies on having enough storage capacity to complete either full applications or individual tasks without preemptions. In Chapter 2, we have seen how dynamic adjustments of the supply voltage can be used to reduce the application energy. If the system has unified memory schemes like the MSP430FR[Tex15], most application data is already non-volatile. We leveraged this fact in Section 2.5 with our low-power vision sensor. The sense-and-process application was manually split into two separate bursts without any state retention overhead. For architectures without unified memory schemes, data will need to be transferred between volatile and non-volatile memory (NVM). Data-intensive applications will thus require optimized data transfers for energy-efficient execution. If a system's energy storage capacity cannot guarantee the atomic execution of the application, then state-retentive execution becomes necessary. As opposed to atomic execution, state-retentive execution needs to account for preemption, typically due to a power-critical interrupt. These systems [BWM+15, JRR14, RBMW18] can be very efficient when the environment can sustain computation since it can avoid checkpointing. However, applications with power-hungry peripherals like cameras or radios might not be able to complete if their energy requirements are not met.

Sensing applications can have widely varying energy requirements depending on their tasks. Some tasks (e.g. sensing and transmitting) cannot be interrupted and require a precise energy storage capacity. Other tasks (e.g. data

processing) can be interrupted but data needs to be transferred between volatile and non-volatile domains. Applications with both types of tasks can be difficult to optimize for multiple reasons. Reliable execution depends on the ability to determine energy bursts sizes, which depend on both tasks sizes and the amount of data which needs to be saved and restores from NVM. Determining memory requirements of conventional code is not feasible, and conservative estimates would lead to large energy overheads. Our proposed optimization scheme, called *Julienning*, is based on a data-flow specification model where programmers declare atomic kernels with explicit data dependencies. Our optimizer can then partition an arbitrarily long sense-process-transmit application into bursts with a bounded energy size and minimal energy overheads. In doing so, atomic tasks can have the energy guarantees necessary for application progress and precious harvested energy will be efficiently used by the application itself.

4.1.1 Contributions

Complex processing tasks have two main hurdles for efficient batteryless execution. First, large and energy-hungry applications would require a very large energy storage capacity for the atomic execution of the entire application. In reality, large applications consisting of multiple tasks only require task-based atomicity for functionally correct execution with minimized energy cost per unit of work. Second, batteryless execution of data-intensive raise specific challenges since preempting a task due to energy unavailability would incur in a large penalty to transfer data from the volatile to the non-volatile domain. We will use head counting as a sample batteryless sense-process-transmit application, and demonstrate how our design tools can efficiently execute an energy-demanding and data-intensive application. To this end, two head counting systems were designed: one based on a normal vision sensor, the other based on a thermal (infrared) sensor. To detect the number of heads in each type of image, we trained CNN's and we implemented them in a low-power

microcontroller and applied *Julienning*. By varying the storage capacity bound, we will identify the Pareto front of both the thermal and the vision-based systems.

The main contributions presented in this chapter are the following:

- Design and implementation of two head-counting embedded systems, based on visual and thermal images.
- Trained CNN's with small memory footprint. Software was implemented using a data-flow specification model with explicit data dependencies.
- The *Julienning* optimization flow which partitions a large sequential application into bursts with a bounded energy size and minimum energy overhead.
- Experimental evaluation of both head-counting applications and their batteryless execution using the minimum feasible storage capacity.
- Design space exploration using *Julienning* over a wide storage capacity range.

4.1.2 Roadmap

The remainder of this chapter is structured as follows: In Section 4.2, we summarize current approaches to preempting batteryless applications. Section 4.3 introduces our general approach to specifying atomic kernels with explicit data dependencies. Our application and partitioning models are presented in Section 4.4. Section 4.5 presents the *Julienning* optimization algorithms. Two head-counting embedded systems optimized for batteryless operation are presented in Section 4.6. These systems will be used to evaluate our proposed optimization flow in Section 4.7. Finally, we summarize our work in Section 4.8.

4.2 Related Work

The intermittent power produced by transducers and its impact on the reliable execution of sensing applications has been studied in recent years. When the energy storage device is treated as a design variable, it can be dimensioned to guarantee the non-preemptive execution of a single sense-process-transmit cycle. While this approach avoids any run-time overheads, it typically requires a large energy storage device. If neither the energy source nor energy storage can guarantee one full application cycle, then the application will inevitably be preempted, thus requiring special techniques guarantee progress and functional correctness. We identify two general approaches which have been developed. If an application contains tasks with atomic execution requirements (e.g. sensing and transmitting), then sufficient energy storage must be provided for these tasks to execute reliably under adversarial harvesting conditions. When applications do not have task atomicity requirements, they can rely on mechanisms to detect power-critical instances to then consistently transfer data from volatile to non-volatile memory before powering down. These software-based mechanisms focus on developing robust code that can withstand constant resets and still have consistent program progress.

4.2.1 Energy storage-based program progress

When information is short-lived, like in sense-and-send applications, a correctly dimensioned storage device can guarantee the uninterrupted execution of a single application cycle, independent of transducer intermittence. If minimizing state retention overheads is the only objective, this solution would be optimal since the data lifetime starts and ends within one activation cycle. Examples of such systems include cameras that wirelessly transmit pictures[NPK+15], a wearable camera that can estimate a user's walking speed [GSS+17b] and ambient sensors that transmit measurements via BLE [MB11] and LoRa [MB16]. Dimensioning energy storage for uninterrupted application cycles is not a scalable approach

as it increases linearly with application requirements. Other works [CRL18] have proposed a reconfigurable energy storage architecture, which can adjust to dynamic application energy demands. By contrast, our *Julienning* approach uses the energy storage capacity as an optimization constraint. This allows a designer to explore a wide range of energy storage bounds to find the minimum capacitance necessary for guaranteeing atomic tasks, not entire application cycles. In this way, the capacitance can be greatly reduced and our optimization model minimizes the energy overhead given this energy storage bound.

4.2.2 Software-based program progress

When application requirements exceed a system's storage capacity, software execution requires specialized methods for guaranteeing consistent program progress. Backing up data in non-volatile memory is a common support mechanism for data consistency in batteryless applications, and a lot of work has gone into optimizing this process. Architectural support for checkpointing includes specialized data transfer between volatile flip-flops and shadow non-volatile flip-flops, for example scan-chain based methods [HFdGB17]. Furthermore, advanced data tracking techniques restrict data transfer to registers which changed after previous backup [HXZ+13]. FRAM-enabled designs have been shown to have costs of 3.44 pJ per bit [QAC14] and <400-ns wake-up time [KBC+14] using 130 nm technology, while a 65 nm ReRAM-based design [LWL+16] has 20 ns restore time, operating at 100 MHz. SW-based methods are based on an external signal which warns the system of an imminent undervoltage condition, triggering a data transfer from the volatile to the non-volatile domain. Software libraries have been developed to intermittently execute processing tasks by automatically saving/restoring volatile data based on voltage thresholds [JRR14, BWM+15]. In [RBMW18], support for re-configuring external peripherals was added. These works have been demonstrated to work on 16-bit MSP430 microcontrollers

with on-chip FRAM. Automatic checkpointing techniques have also been demonstrated on flash-based systems [RSF11], 32-bit Cortex M3 systems [BM16] along with energy-aware optimizations [BM17]. Additional state-retention policies have also been proposed to exploit the different properties of both FRAM and Flash to find the most efficient policy/platform [VBM18], Besides checkpointing, applications have also been decomposed into tasks which are then executed individually [GSM+16, HSS15b, GSS+17a]. These systems require manually transferring data to non-volatile memory. While state-retentive systems excel at minimizing the required energy storage for executing arbitrarily long processing tasks, they have a fundamental limitation from their small storage. Large atomic tasks using power-hungry sensors or transceivers are simply not supported.

Specialized languages and runtimes have been developed for batteryless systems. The Drewdrop [BGW11] runtime, developed for flash-based MSP430's, is able to dynamically adjust the system's wake-up voltage to find the lowest value for reliable task execution. Due to the high costs of flash memory access, Dewdrop does not support state retention. The DINO[LR15] programming and execution model which breaks down applications into sets of instructions manually defined task boundaries. DINO can guarantee consistency by tracking both volatile and non-volatile states, and re-executing any interrupted task from the most recent task boundary. With Chain [CL16], developers can specify applications as static task graphs with statically multi-versioned channels, with restricted access to volatile and non-volatile memory domains. Chain is able to reduce checkpointing costs, with guaranteed consistency, by marshaling data and allocating multiple copies of data in non-volatile memory. CleanCut [CL18] can automatically decompose applications into tasks, by placing task boundaries such that they can be executed with a given energy storage. CleanCut works by analyzing the application's control-flow graph and using a statistical energy model to avoid non-terminating path bugs. The Mayfly [HSS17] language and runtime, developed for MSP430's with

on-chip FRAM, allows developers to declare tasks with time-dependent data flows. In this way, the runtime can decide whether or not to execute tasks based on the age of sensed data.

All of these programming models assume an inherently unreliable hardware layer that can reset the system at any point. In contrast, we build on top of an Energy Management Unit (EMU), which can guarantee a specified amount of energy, regardless of the variability in a transducer's voltage and current. This opens the door for a programming model with guaranteed program progress with controlled, consistent preemption at predefined program points. This allows us to run energy-hungry tasks, even if their power requirements are beyond the limits of the source. Furthermore, our programming model can automatically minimize the total application energy using a transducer-independent, energy burst execution model. Related work embraces a greedy execution model contingent on energy source behavior. Missing.

4.3 General Approach

In this chaper, we will use sample head-detecting embedded systems to demonstrate and evaluate our proposed *Julienning* approach. Though the implementation details will be discussed in greater detail in Section 4.6, we will introduce our custom specification model with a generic sense-process-transmit application. This specification model is based on *Ladybirds* [Tre18]. *Ladybirds* was specifically designed for efficient data exchange for parallel applications in multi-core architectures. There are three phases to using *Ladybirds*: front-end, optimization, and code generation. During the front-end phase, developers specify the application with a C dialect. During the optimization phase, the application and its energy costs are analyzed to determine the optimal partitioning for a given storage bound. During the back-end phase, platform-specific C code is generated. A custom

back-end was developed for the low-power LPC5400 platform [NXP15]. We will now discuss in more detail the specification model used in the front-end. The next section will focus on the optimization phase.

The specification model distinguishes between kernels and metakernels. Kernels can be thought of as conventional C functions with explicitly specified inputs and outputs of fixed sizes [Tre18]. Kernels implement a specific functionality, while metakernels interconnect kernels to build up applications.

A kernel describes how input data is transformed into output data in the front-end. A call to a kernel is called a task. In Listing 4.1, we have a front-end representation of a sense-process-transmit application. In this example, we have defined three kernels with explicit data dependencies and one meta-kernel joining them. Though this simple example would generate one task per kernel, this is not always the case. A hierarchy of metakernels will be flattened out by *Ladybirds* and depending on the data dependencies, multiple tasks can be instantiated. In the case of our head counting application, for example, we have specified 7 processing kernels, 3 of which are CNNs. The code generated from this specification includes over 5400 tasks, the vast majority being CNNs analyzing different image segments.

For our batteryless execution model, kernels (and tasks) will be executed atomically. When an application needs to be partitioned into multiple bursts, *Julienning* can only possibly to preempt execution in between tasks, where memory usage was explicitly declared. Conceptually, this is similar to cooperative scheduling[BBY13], where the developer can choose when a thread *yields* control of the CPU. Our specified application is basically a vector of tasks. Partitioning is then the process of selecting which tasks to execute together before preempting. For these execution bursts to be reliable, they need to respect the system's storage capacity. This means that the energy required to load input data, execute tasks and save output data needs to be known. We will propose a burst energy model for calculating these costs based on pre-characterization data. By default, *Julienning* can generate one task per burst partitioning

Listing 4.1: Sample definition of atomic kernels with explicit dependencies.

```
#define Dx 80
#define Dy 60

kernel sense( out uint8_t img[Dx][Dy]) {
    camera_enable();
    AcquireImage(img);
    camera_disable();
}

kernel process( in uint8_t img[Dx][Dy], out uint8_t headCount) {
    //Picture is processed, result saved to 'headCount'
    headCount = runCNN(img);
}


kernel transmit(in uint8_t headCount) {
    radio_enable();
    BLE(headCount);
    radio_disable();
}

metakernel main() {
    uint8_t img[Dx][Dy];
    uint8_t headCount;

    //Acquire picture and store it in 'img'
    sense(Dx,Dy,img);
    //Process picture and save result in 'headCount'
    process(Dx,Dy,img,headCount);
    //Transmit result via BLE
    transmit(headCount);
}
```

to facilitate energy characterization measurements using a DC source. For the state retention, *Julienning* can determine the amount of data which needs to be transferred, and a linear model is used to determine the energy cost. The systems energy storage capacity is an independent design parameter to be chosen by the developer. This will have a clear impact on how, if at all, the application will be partitioned. Using the application model and energy characterization, *Julienning* will then find the optimal burst partitioning such that the burst size respects the energy bound, and the total overhead is minimized.

4.4 System Model

In this section, we introduce our model for specifying computationally intensive applications. This methodology is the result of a collaboration with Andreas Tretter, Pascal Alexander Hager, and Praveenth Sanmugujarah. Our proposed specification model is composed of discrete tasks, or kernels, with well defined inputs and outputs. All discrete tasks have guaranteed reliable execution if the EMU's storage capacity was dimensioned properly. However, their input and output data needs to be loaded and stored in Non-Volatile Memory (NVM) to tolerate power outages. Accessing NVM is more expensive than on-chip SRAM, and its use needs to be optimized in order to keep energy overheads manageable. We will first describe application and energy model with optimized data transfers.

4.4.1 Application model

A sequential application $\mathcal{A}$ is defined a 4-tuple (T, M, In, Out) with a vector of tasks $T = \{t_1, t_2, ..., t_{N_{tasks}}\}$, a set of memory blocks $M = \{m_1, m_2, ..., m_{N_{mem}}\}$, a data access function to obtain input data memory blocks $In : T \rightarrow \mathcal{P}(M)$ [1] and a data access function to obtain output memory blocks $Out : T \rightarrow \mathcal{P}(M)$.An

[1] $\mathcal{P}(M)$ denotes the powerset of M, or the set of all the subsets of M.

application consists of tasks which can have multiple inputs and outputs, kept in separate memory blocks. By definition, every task $t \in T$ can only access memory blocks enclosed in $In(t) \cup Out(t) \subseteq M$. The size of each memory block $m \in M$ can be obtained with the function $s : M \rightarrow \mathbb{N}$.

4.4.2 Partitioning Model

If an entire application can be directly executed from energy storage, then there is no need to partition an application. However, as batteryless applications grow in complexity and energy requirements, energy storage bounds will play a decisive role in their development. Partitioning enables an application to be divided into smaller units of execution. Using a burst-based execution model, these units of execution have guaranteed reliable execution. Since we focus on sequential applications, whose tasks need to be executed in a predefined order, the process of partitioning is reduced to defining the starting point of the application's N_{bursts} bursts. To distinguish which tasks belong to which bursts, we define an *execution configuration*. An *execution configuration* of an application is a vector $\mathbf{R}$ of starting task indices $\mathbf{R}$= $(r_1, r_2, ..., r_{N_{bursts}})$ and $r_k < r_{k+1}$ $\forall k | 1 \leq k < N_{bursts}$. The auxiliary function *burstBounds* determines the starting and ending indices of the i-th burst given an *execution configuration* ($\mathbf{R}$). It is defined as follows

$$burstBounds : (i, \mathbf{R}) \rightarrow (\mathbf{r_i}, \mathbf{e_i}) \tag{4.1}$$

where $i \in \{1, ..., N_{bursts}\}$ and $r_n < e_n$.

Optimizing NVM Data Transfer

As mentioned in the introduction of this section the considered system needs to regularly save results of the executed tasks on an external non-volatile memory, to guarantee that data is available for the subsequent burst after a power loss. A burst needs to load data from an external non-volatile memory into the main memory, before the system can start executing tasks. One possibility is to load all the input memory blocks from the

non-volatile memory and storing all the output memory blocks on the non-volatile memory of each task. This is not efficient in terms of energy since each transfer consumes energy and usually not all of the data needs to be stored and/or loaded. Another possibility is to only consider memory blocks that are needed. This leads to the minimum number of memory blocks that have to be transferred for functional correctness. Two access functions $Mem_{load} : T \rightarrow \mathcal{P}(M)$ and $Mem_{store} : T \rightarrow \mathcal{P}(M)$ are introduced to reduce NVM data transfer.

$Mem_{load}(t_k, n)$ determines which memory blocks need to be loaded for the task t_k in the n-th burst with $(i, j) = burstBounds(n, \mathbf{R})$ and can be described as follows:

$$Mem_{load}(t_k, n) = \{In(t_k) \setminus \cup_{l=i}^{k-1}(Out(t_l) \cup In(t_l))\} \tag{4.2}$$

Not all the input memory blocks of task t_k need to be considered for loading. Some might be generated within the burst and others might have been already loaded by previous tasks in the burst. In both cases, data transfer can be omitted since memory blocks will already be in SRAM. Therefore $Mem_{load}(t_k, i)$ has to consider only all the input memory blocks that are not generated within the i-th burst before the task t_k and exclude also input memory blocks that will be loaded from previous tasks within the i-th burst. All the common input and output memory blocks of the previous tasks within the bursts are removed from all the input memory blocks of the task t_k. The results set of memory blocks are those which need to be loaded from the external memory for the task t_k.

$Mem_{store}(t_k, n)$ similarly determines which memory blocks need to be stored after task t_k is executed in the n-th burst with $(i, j) = burstBounds(n, \mathbf{R})$, as follows:

$$\begin{aligned} &Mem_{store}(t_k, n) = \\ &\qquad \left\{ m \in Out(t_k) \middle| \text{findIn}(m, j) < \text{findOut}(m, k) \right\} \end{aligned} \tag{4.3}$$

Just like with loading, not all the output memory blocks of the task t_k have to be considered for data transfer. Some of the output memory blocks might only be used within the

same burst. Only if the memory blocks are the input of tasks in subsequent bursts will those blocks need to be transferred to NVM. However, we still need to check whether other tasks within the same burst also write to those blocks, since we only want to transfer blocks once they are no longer being used within the burst. We use two additional functions to determine this:

$$\text{findIn}(m,k) = \min\left\{j \in \{k+1,...,N_{tasks}\} | m \in In(t_j)\right\} \quad (4.4)$$

$$\text{findOut}(m,k) = \min\left\{j \in \{k+1,...,N_{tasks}\} | m \in Out(t_j)\right\} \quad (4.5)$$

$findIn(m,k)$ returns the index of the first task after the k-th which also uses the memory block as an input memory block, or $N_{tasks}+1$ if there are no tasks. Similarly, $findOut(m,k)$ is used for the output memory block. With the help of these functions, $Mem_{store}(t_k, i)$ returns the set of memory blocks which needs to be stored on the external memory for future bursts.

4.4.3 Execution of Partitioned Applications

Algorithm 1 gives an overview of burst execution using *Julienning*. Given an application and partitioning $\mathbf{R}$, Algorithm 1 will execute all tasks using burst with minimized energy overheads. The outer loop iterates through every burst in the application. To keep track of the burst, a counter b is used. Each burst entails the following steps: loading data, executing tasks and storing data. After every burst, the system goes to sleep and waits for an external interrupt from the EMU.

At the beginning of each burst, the burst counter is loaded from the NVM into the main memory using the function *loadState*() and the $burstBounds(b, \mathbf{R})$ function calculates the indices of the starting and the ending tasks of each burst. When tasks require input data, Mem_{load} gives the optimized list of memory blocks that needs to be loaded for a specific task.

As soon as all the required inputs for the bursts' tasks have been loaded, all tasks will be executed sequentially. Once

Algorithm 1: Execution of partitioned application with *Julienning*

```
   Input  : 𝒜, R
1  foreach b in R do
2      (i,j) = burstBounds(b,R);
3      foreach k in (i,j) do
           /* load input data from NVM                    */
4          load(Mem_load(t_k, b))
5      end
6      foreach k in (i,j) do
           /* execute tasks in burst                      */
7          execute(t_k);
8      end
9      foreach k in (i,j) do
           /* store output data to NVM                    */
10         store(Mem_store(t_k, b));
11     end
12     b = b + 1;
13     storeState(b);
14     sleep();
15     b = loadState();
16 end
```

all tasks have been completed, Mem_{store} provides the list of memory blocks that need to be stored. Auxiliary functions, `load` and `store`, transfer the data to and from the NVM. Lastly, the algorithm increases the `burst-round` number and stores it on NVM using the function *storeState*().

Thanks to the specification of the explicit data dependencies, *Julienning* can calculate the required memory blocks to load and to store. Non-optimized specifications models would need to load and store the entirety of the application data, since it is not known a priori which data will be read or modified.

4.4.4 Energy Model

As was shown in Algorithm 1, applications executed over multiple bursts have the following four stages: boot-up, loading of input data, task execution, and storing of output data. Naturally, single-burst applications would not have loading/storing stages since all data is produced and consumed within the same burst. For all other burst configuration, we need to calculate both individual burst costs as well as total energy costs.

Individual Energy Burst Costs

Of the four stages we have identified, we can distinguish between "useful" and overhead energy. The former is the energy which goes towards task execution, while the latter are costs which cannot be avoided. Both energies need to be modeled to have an accurate notion of a burst's energy. The energy necessary to boot-up the system, $E_{start-up}$, depends on many parameters (voltage, operating frequency, etc). The energy consumed for data transfers depends on the type of memory, its energy cost per byte ($E_{readByte}$ and $E_{writeByte}$) and the amount of data being transferred. Which specific memory blocks need to be transferred from/to NVM depends on the set of tasks in each burst. The input and output memory blocks of the n-th burst with $(i, j) = burstBounds(n, \mathbf{R})$ can be determined as follows:

$$MemInput_{partition} = \cup_{k \in (i,j)} Mem_{load}(t_k, n) \tag{4.6}$$

$$MemOutput_{partition} = \cup_{k \in (i,j)} Mem_{store}(t_k, n) \tag{4.7}$$

To find the input memory blocks, we apply the union over the inputs to all tasks belonging to the same burst. The same procedure is also applied for output memory blocks. Lastly, the burst energy (E_{burst}) is calculated by determining the size of these blocks and adding all of the tasks energies.

$$E_{read} = E_{readByte} \cdot \text{bytes}(In_{partition}) \tag{4.8}$$

$$E_{write} = E_{writeByte} \cdot \text{bytes}(Out_{partition}) \tag{4.9}$$

$$E_{burst} = E_{start-up} + E_{read} + E_{write} + \sum_{k \in \mathcal{B}} E_{task,k} \tag{4.10}$$

Total Energy Costs

We have seen so far the energy costs for an individual burst. We now study the energy costs for running entire an application over multiple bursts. The energy required to execute the application, without any overheads, is simply the sum of all tasks: $E_{app} = \sum_{i=1}^{N_{tasks}} E_{task,i}$. As we have seen, partitioned applications incur energy overheads to turn on the system and transfer data between on-chip and off-chip memories. We distinguish overhead between start-up costs and data transfer costs as follows. For an application executed over N_{bursts} bursts, the system boots-up once per burst incurring a cost of $E_{start-up}$.

To obtain the energy consumption for the transferred data $S : \mathcal{P}(M) \rightarrow \mathbb{N}$ is introduced. It returns the number of Bytes that is contained in a set of memory blocks. This relation can be described as follows:

$$S(M) = \sum_{m \in M} s(m) \tag{4.11}$$

The energy overhead from data transfers can be decomposed in two: energy for reading (optimized) data inputs, and energy for writing (optimized) data outputs. The burst counter, of size S_{ctr}, is the one variable that is always loaded, updated and stored, as it is essential for program progress.

$$E_{transfer} = E_{load} + E_{store} \tag{4.12}$$

Since the amount input and output data can be different, they are modeled separately. We calculate the energies by multiplying the amount of data (in bytes) times the read/write energy per byte. Note that many NVMs have asymmetrical read/write energies.

$$E_{load} = \left(\sum_{i \in \{1,\ldots,N_B\}} \sum_{t \in T} S(Mem_{load}(t,i)) + S_{ctr} \cdot N_{bursts} \right) \cdot E_{readByte} \tag{4.13}$$

To obtain the energy consumption for writing data the same procedure like for obtaining the energy consumption for reading is applied.

$$E_{store} = \left(\sum_{i \in \{1,\ldots,N_B\}} \sum_{t \in T} S(Mem_{store}(t,i)) + S_{ctr} \cdot N_{bursts} \right) \cdot E_{writeByte} \tag{4.14}$$

The total energy consumption can be described as follows where the energy of the application is also considered:

$$E_{Total} = E_{app} + E_{overhead} \tag{4.15}$$

$$= E_{app} + E_{transfer} + N_{bursts} * E_{boot-up} \tag{4.16}$$

4.5 Optimal Partitioning

The *partitioner* takes a sequence of tasks and segments it into certain *bursts*. The aim of the partitioner is on the one hand to reduce the required energy guarantee for an application and on the other hand to find a partition such that the total energy consumption of the application is minimized. In the previous section, we have seen that data transfer between volatile and non-volatile memory can be optimized thanks to the explicit data dependencies in *Julienning*. However, this alone is not enough to optimize the total energy consumption. In sequential applications, partitioning is the only possible way to reduce the required storage capacity. When this is combined with the aforementioned data transfer optimizations, partitioning can be done with very low energy overheads.

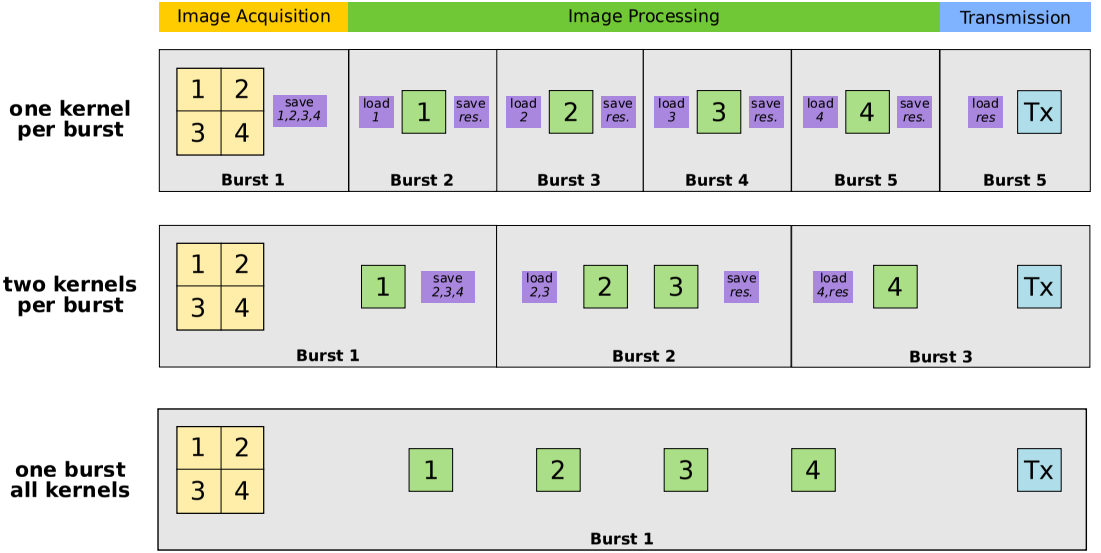


Figure 4.1: Three basic partitioning schemes for a simplified image processing application consisting of 6 kernels. Top scheme generates one energy burst per kernel, incurring save/load overheads. Middle scheme generates one burst per two kernels, decreasing the save/load overheads. Bottom scheme generates only one burst for all kernels, avoiding save/load overheads but requiring a large burst and storage element. It should be noted that *saving* and *loading* have already been optimized to each burst.

Figure 4.1 shows examples of fixed task partitioning. Following our sense-process-transmit application, we have defined 1 acquisition, 4 processing and 1 transmit kernels. Each processing kernel processes one frame of the image. The simplest partitioning is no partitioning at all, where one burst contains all kernels. Though this has the advantage of having zero data being transferred between volatile and non-volatile memory, it requires a large storage capacity. By splitting the application into multiple bursts, we hope to reduce this required capacity. When one kernel is assigned an individual burst, one can note several effects. The number of bursts is maximized, as is the amount of data transfer. If the kernels dominate the burst energy, then burst size is also minimized. Note that when data transfer is more expensive, bursts sizes can get larger. For our application and NVM technology, however, this is not the case.

Fixed partitioning takes a constant number of tasks per burst. Even though this is fairly simple to implement, it can lead to great inefficiencies. Since all bursts need to be executed atomically, the storage capacity needs to match the largest bursts. Ideally, all bursts can be the same size but since kernels are specified by application designers, they will very likely have different energy costs. Consequently, we need an automated way to calculate optimal partitions that can make the most of a bounded storage capacity and also minimizes energy overheads.

This section explains how an application $\mathcal{A}$ can be partitioned into bursts such that the burst sizes are bounded, and the total application energy is minimized. For a given sequential application there are $2^{|N_{tasks}|-1}$ possibilities to partition it into bursts. An *exhaustive search* for finding the optimal partition is computationally infeasible for arbitrarily long applications. To solve the partitioning problem, *Julienning* transforms it into a shortest path problem, which can be solved in polynomial time.

The following example illustrates the steps to achieve the shortest path-problem transformation. A sequential application with three tasks and explicit data dependencies is

shown in step 1) of Figure 4.2. In a second step, we introduce $N_{tasks} + 1$ new states (S_0 to S_3).

S_0 indicates the state before the application is executed, while S_3 indicates the state after the application is executed. States in between indicate a sleep state where the system is switched off. In this state graph, a burst is defined as a change of states. The edges between the states indicate the energy that is needed to advance the state. For instance, to get from S_0 to S_1, the system must turn on, execute task t_1, store d_a in NVM, and then go to sleep. To get from S_0 to S_4, the system needs to wake up and execute all tasks (no data transfers necessary). To fully populate this graph, the following energy costs are needed:

- energy cost of each task E_{task}
- energy cost of transferring one byte $E_{readByte}$ and $E_{writeByte}$
- energy cost for booting up the system $E_{start-up}$

Step 2) in Figure 4.2 illustrates all the possible transitions in the new state graph. If there were no energy storage bounds, then the graph could be left as is. The shortest path between S_0 and S_4 would simply be $\{t_1, t_2, t_3\}$. However, with *Julienning*, designers can specify an arbitrary storage bound and still find the partitioning that minimizes the total energy. To do so, all the edges that exceed the storage bound Q_{max} are removed from the graph. In this example, we color the $\{t_1, t_2, t_3\}$ edge to denote a removed path. Afterward, in step 3) we find the shortest path between S_0 and S_4 using *Dijkstra*'s algorithm, which finds the optimal path in polynomial time. Each node on the obtained path indicates the last task of a burst. Lastly, in step 4) we convert the node index into starting indices to get the optimal execution configuration **R**.

4.5.1 Minimizing Storage Capacity

At this point, we have already solved the problem of finding the optimal partition that minimizes the total energy and satisfies a storage bound. However, our execution configuration

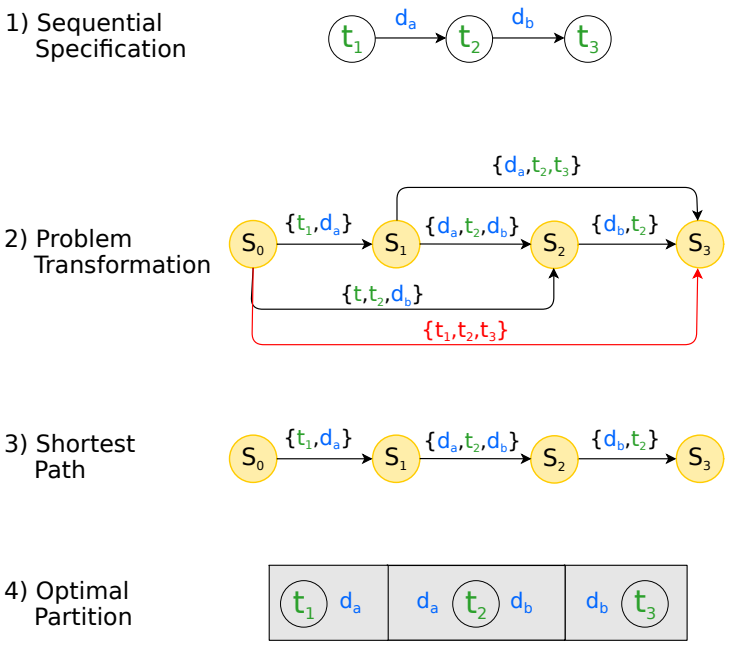


Figure 4.2: Optimization flow using *Julienning* starts with a high-level specification. This is then transformed to state-based graph with energy burst costs. Partitioning with minimum energy is found using the shortest path algorithm.

might require a much smaller storage capacity than the actual bound. There are several factors which determine this, namely the distribution of kernel energies and their ratio to storage overheads. Ultimately, the total energy does not have a continuous range and will experience possibly large gaps. To find the actual minimum capacity Q_{min} of an application with an execution configuration $\mathbf{R}$, we adapt *Dijkstra*'s algorithm. Instead of summing up all the weights on a path, we simply find the highest weighted edge of that path. This is the actual storage capacity needed for this path/partitioning.

4.6 Head Counting Embedded System

Detecting the number of people occupying an environment is an important use case for surveillance in public spaces such as airports, stations and squares, but also for smaller environments such as classrooms (e.g. to track occupation of classrooms). Using visible imaging for this task is often suboptimal because 1) it potentially violates user privacy 2) to have a good final count, high-resolution cameras are required. Long-wave infrared imaging is a viable solution to both these issues. Here, we present a people counting algorithm on thermal images based on convolutional neural networks (CNNs) small enough to be executed on a limited-memory low-power platform. This algorithm results from a collaboration with Francesco Conti.

4.6.1 Detection Algorithm

Convolutional Neural Networks (CNN's) [LBBH98], are a popular method for many image recognition tasks. CNN's use a set of filter kernels which are convoluted with the input image to extract certain features from it in a succession of multiple *layers*, each convolving its own filter kernels with the output of the previous one, thus extracting increasingly higher-level features. CNN's have successfully achieved better-than-human performance in a variety of computer vision problems in the visible light spectrum. Due to their high spatial resolution, these images contain texture details consistent with the human visual system [MML18]. For this reason, visible image recognition tasks are usually deployed on high-performance hardware with an abundance of memory and computing power. By contrast, thermal or infrared imaging can make objects stand out due to their temperature, making them more immune to weather, lighting conditions or body pose.

Dataset collection and tagging. In order to effectively train any Neural Network, a large set of tagged training data is required. We collected a dataset targeted at people recognition in the context of a classroom by setting up five Raspberry Pi

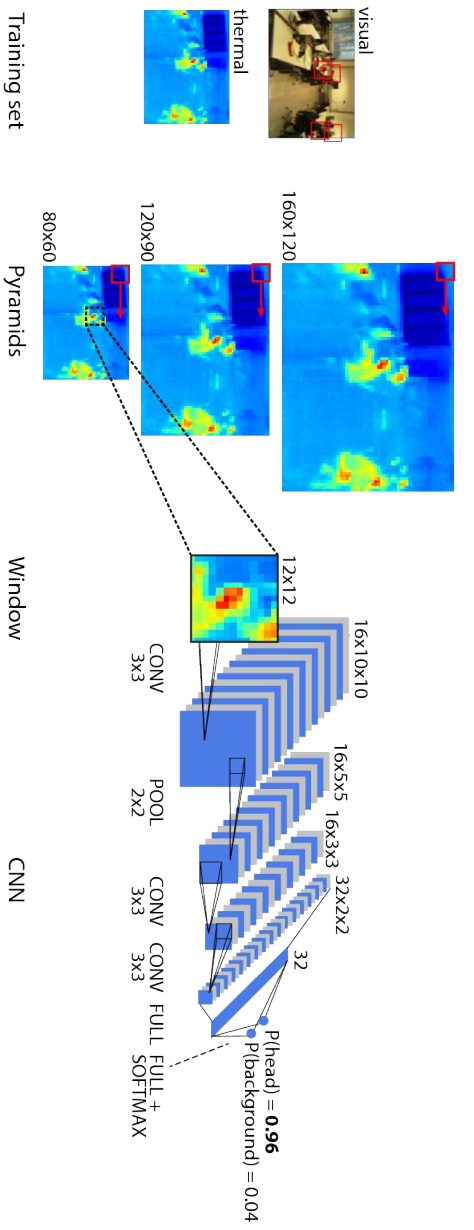


Figure 4.3: Overview of the proposed thermal head detection algorithm.

single-board computers in a student workroom. Each of the Rasperry Pi's was fit with both a thermal and a visible light camera and set up to capture the room from different angles.

The low-power thermal camera[FLI18] encodes each image pixel as a 16-bit value between 0 and 65536 proportional to the impinging amount of infrared radiation; each thermal image has 80×60 pixel resolution, considerably smaller than typical visual cameras. To collect the dataset, we coupled the low-power thermal camera with an off-the-shelf visual camera, whose collected output was scaled to 80×60 to allow for a fair comparison between the two approaches.

We developed a small Python tool to aid with the manual tagging of these images, which was performed based on the visual images, which are much better recognizable from a human's perspective. The tagged dataset was then shuffled and divided in a training set with 2089 images, a validation set with 446 images and a test set with 450 images. Due to the different image resolutions, aspect ratios and possible slight differences in camera orientation, the tags from visual images cannot directly be used for the corresponding thermal image. To achieve this, we fit a transformation of the form

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \underbrace{\begin{pmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \end{pmatrix}}_{T} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{4.17}$$

to a list of coordinates (x, y) on the visual image and (x', y') on the thermal image corresponding to the same point. Using the resulting transformation matrix T, we were able to reconstruct accurate thermal tags out of the visual ones.

Head detection and counting. The detector we developed focused on detecting people in a student's work room, where people are often partially occluded by the desk they are sitting at. This occlusion of body parts makes it unreasonable to try a full-body detection but favors detecting only the heads. This can be justified by the fact that the head is usually the most visible body part in such a setting, and also one which radiates a high amount of heat that makes them even more visible

through a thermal camera.

We decided to detect each head individually by sliding a windowed classifier on top of the input image and classifying each window as a head or background rather than use a direct regression approach on the full image. This approach adds robustness, is easier to train and has an overall lower memory footprint. The lower memory footprint originates from the fact that only small portions (windows) of the input image are fed to the CNN, which greatly reduces the size of the feature maps that have to be held in memory. The simplicity of a binary head or background classification also implies a simpler overall structure for the employed CNN, meaning less layers and smaller convolution kernels, thus reducing the number of weights needed.

The general topology of the applied CNN was thus inspired by the work of Li et al. [LLS+15]. Their CNN consists of a total of six stages, where calibration stages follow detection stages to correct the position of windows classified as faces. These corrected or calibrated windows are then passed to the next classification stage which has a more complex topology and analyses the window at a higher resolution than the previous one. In our work, we build on the first and simplest CNN they propose, using 12×12 *detection window* as input and 3×3 convolution kernels to predict whether it contains a head or not, performing binary classification.

At the native 80×60-pixel resolution, 12×12 already covers an area larger than the biggest expected head size on the image produced by the thermal camera. To be able to detect smaller details without increasing the size of the window, during detection we upscale the input image, creating a pyramid of three images sized 80×60, 120×90 and 160×120 pixels, respectively. To increase numerical stability during the training procedure, the images in the pyramid are normalized to a range of $[0, 1]$, using maximum and minimum values collected from the entire training set.

A 12×12 detection window is slid along each of the pyramid images using a stride of 2 pixels in each direction. The collected detection windows are then fed to the CNN-based classifier.

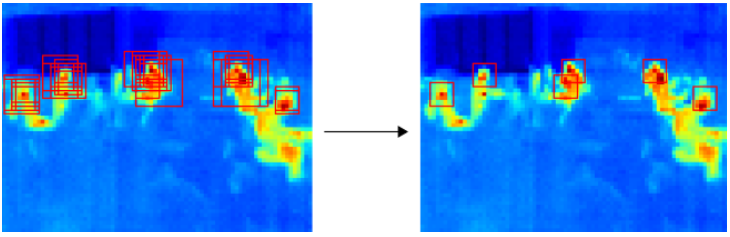


Figure 4.4: Example of non-maximum suppression reducing the initial 45 detections on the left to the 6 ones on the right.

Similarly to Li et al [LLS+15], the network uses the ReLU activation function, max pooling after the convolutional layer and a final softmax activation[2] for the output. Figure 4.3 shows the overall methodology illustrated in this section, from the dataset images up to the proposed classifier topology.

The output of the sliding window classifier is an array of confidence values ranging in $[0, 1]$, each indicating how confident the CNN is that the corresponding image patch contains a head. However, one particular head on the image will usually still be detected by multiple windows at different positions and scales making it necessary to determine the most confident one of all of these overlapping windows. This is done by applying *non-maximum suppression* as defined in Felzenszwalb et al. [GFM]. It greedily takes the detection with the highest confidence and eliminates all others with significant overlap, then proceeds to the next highest until only the local maxima are left. One example of detections before and after the application of non-maximal suppression is shown in Figure 4.4. After this step, only the correct detections remain, so the remaining windows can be counted to obtain the final people count in the image.

For the purpose of training the CNN head detector, the

[2]The softmax function converts a number of output values to values in the range (0, 1) that add up to one and can be interpreted as a probability distribution. It is defined as $\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$ where z is a vector of N output values.

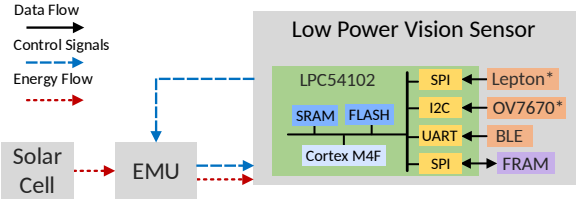


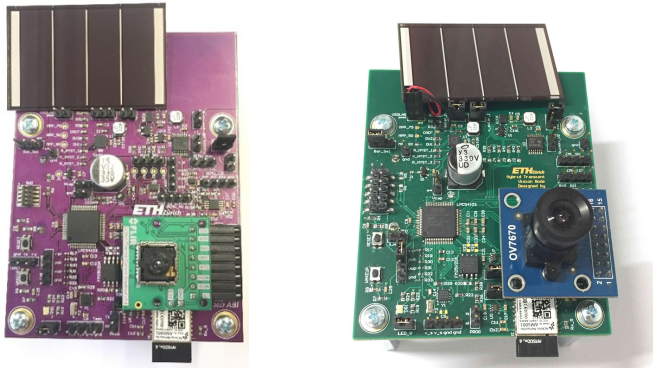
Figure 4.5: Overview of the ultra-low-power people recognition platform. One system uses a visual sensor (OV7670 [Omn06]), while another uses an infrared sensor (Lepton [FLI18]).

presented people counting algorithm was implemented in Python targeting the Keras [Cho15] deep learning framework with the TensorFlow [AAB+15] backend. The CNN training set was created using 4203 head patches cut out from the full training dataset and 5000 randomly selected backgrounds, while a CNN validation set using all patches from the full validation dataset was used to select the "best" result from the training. Batch normalization was used after each convolutional layer and dropout layers were inserted before the two fully connected ones to aid with training and minimize overfitting.

4.6.2 Embedded Implementation

Two standalone head counting systems were developed with commercial-off-the-shelf components. The vision-based version used the OV7670 [Omn06] camera as sensor input, while the thermal-based version used the FLIR Lepton [FLI18] sensor. The head-counting algorithm (same for both versions) was implemented on a LPC54102 microcontroller [NXP15]. A BLE radio [ACK14] was used to transmit the results of the head counting process. Lastly, an external FRAM memory [Cyp15] was selected as a high endurance, low-power non-volatile memory.

In Figure 4.5 we show a simplified diagram of the full platform, which is battery powered. The LPC54102 contains an ARM Cortex-M4F core with 512 kB of Flash memory



(a) Thermal image version. (b) Visual image version.

Figure 4.6: Head counting system prototypes with integrated EMU.

and 104 kB of on-chip SRAM, with no data caches. This poses severe memory constraints for the embedded CNN implementation, which must be able to fit all weights within the 512 kB of Flash and all data (including intermediate results between CNN layers) within the 104 kB of local memory. In both implementations, the voltage supply was 2.8 V, the minimum required by the Lepton, and compatible with all other peripherals including the LPC microcontroller, the OV7670 vision sensor, and the BLE radio. To facilitate interfacing with peripherals, the platform runs at a frequency of 80 MHz. The thermal image acquisition via the camera's SPI interface, while the visual image is acquired via 8 (parallel) GPIO pins. In both cases, the acquired image is stored in SRAM. The prototypes of both the thermal and visual based detector prototypes can be seen in Figure 4.6.

We implemented the full algorithm described in Section 4.6.1 in bare-metal embedded C targeted at the deployment on the LPC54102. In the case of atomic application execution, the head detection CNN runs directly on SRAM data without transferring any data to NVM. The CNN itself uses a pure C implementation of convolutional and densely

connected layers, without machine-dependent optimizations or special instructions.

4.7 Experimental evaluation

4.7.1 Methodology

This section evaluates the proposed optimization flow for batteryless sensing systems. More precisely, we focus on applications with energy-intensive processing tasks since these can be easily partitioned into separate bursts of execution. To this end, we first evaluate the head counting embedded system presented in Section 4.6.1. We will evaluate the accuracy and energy costs of the same system using a thermal (infrared) sensor [FLI18] and a tradition vision sensor [Omn06]. Afterward, we will evaluate three partitioning schemes, including our optimal *Julienning* method. Lastly, we perform a design space exploration of both the thermal and the visual applications. Using *Julienning*, we can effectively sweep a range of storage capacities and determine the partitioning that minimized the total energy.

Set-Up

All energy characterizations were made using an external DC power supply. The open source RocketLogger measurement device[SGL+17] was connected for low side current measurements. GPIO pins were used to mask power traces and thus determine the energy consumption.

For the partitioning evaluation, the energy costs for the kernels and the external NVM were used. Our optimization tool can then calculate the different figures of merit.

Figures of Merit

For the analysis of the system performance, the following metrics are used in all experiments:

N_{bursts} the number of bursts necessary to execute the entire application.

$E_{start-up}$ the energy required to boot up the system.

E_{load} the energy required to load the input data for all bursts of a given partitioning.

$E_{task,i}$ (kernel energy) the energy required to execute the i-th kernel.

E_{save} the energy required to save the output data for all bursts of a given partitioning.

$E_{app} = \sum_i E_{kernel,i}$ (application energy) the energy required to execute the entire application atomically (without state retention overheads).

$E_{total} = E_{start-up} * N_{bursts} + E_{load} + E_{save} + E_{app}$ the total energy required the execute an application with a given partitioning.

Q_{max} (energy storage bound) the maximum storage capacity allowed by the system

Q_{min} (minimum storage) the minimum storage capacity needed to reliably execute an application with a given partitioning.

4.7.2 Head-counting embedded system

Detection Accuracy

Similar to many of the image recognition challenges out there, like the Face Detection Data Set and Benchmark (FDDB) [JLM10], the bounding boxes produced by the detection algorithm were compared to the original annotations (tags) by calculating their overlap. If a detection overlaps with a tag by more than 30%, it is accepted as correct, otherwise, it is counted as a false positive. This enables the creation of a realistic accuracy statistic over all the full images in the validation dataset. We trained the topology shown in Figure 4.3

for 300 epochs, using the Adam optimizer with learning rate 5×10^{-5}. We reached a final validation accuracy of 97.6%. As the validation set is used during the training phase, a new "untouched" test set is built using 67540 background windows and 872 head windows from the full-image test set. The CNN achieves a 95.9% accuracy on this set; non-maximum suppression with a hard confidence threshold calibrated at 0.9997 yields a net improvement to accuracy up to 99%.

While this final post-training error is low (~1%), the CNN is applied many times to each image, and even a single error can drop the overall algorithmic accuracy. To quantify this phenomenon, we evaluated the overall counting accuracy on the full image test set. The algorithm predicts the correct count on 53.7% of all the test images, and in 84.4% of the images the error is bound within ±1. The non-maximum confidence threshold was calibrated so that false positives are of similar cardinality as false negatives.

As a point of comparison, we also trained a similar CNN to that shown in Figure 4.3 using the collected visual images as input (in full-color, but downscaled to 80×60). We used the same training methodology and parameters as in the thermal case. Our results have shown that the features are typically too small and the images too cluttered for the CNN to be able to converge to a decently discriminating model; in fact, in most iterations, they simply converge to a local minimum where all patches, regardless of their content, were predicted as backgrounds. To highlight the differences between the two results, in terms of discrimination between heads and backgrounds and of correct overall count, we split the two test sets. One was a subset for empty rooms, where the correct prediction is always 0 people, and the other subset was for occupied room, where the number of people varies from image to image. Ideally, both the thermal-based and visual-based algorithms should perform well on either subset. Instead, while the two algorithms perform similarly on pure backgrounds, their results are dramatically different on the occupied rooms. Whereas the thermal-based algorithm is able to discriminate between heads and backgrounds leading to a

correct count in 45% of the images and an error bound within ±1 head for 81% of the predictions, the visual-based algorithm can identify the correct number of people only in ~10% of the subset images.

Even though the detection accuracy using visual images is poor, we will continue the same CNN as a point of comparison for batteryless execution. From an energy perspective, the only difference between the versions using thermal and visual images is the cost of the image acquisition kernel.

Memory

There are three key data elements: the image itself, the weights of the CNN and the intermediate results. The Lepton sensor produces an 80×60 matrix of the type `int16_t`, while the weights and intermediate results are `float`. The code was compiled with the `-Os` optimization flag to minimize its size. The memory breakdown of the compiled application can be seen in the left column of Table 4.1. It should be noted that the biggest section, Text, contains all of the constants for the CNN filters. The BSS section is almost one fourth the Text size and fits comfortably in the available 104 kB SRAM.

Section	Size [B]
Text	$444{\times}10^3$
BSS	$63{\times}10^3$
Data	186

Table 4.1: Memory requirements for people recognition application running on the LPC54102 (stride 3×3).

Kernel characterizations

To characterize the energy requirements for different kernels, we used a DC source to supply 2.8 V. Once again, the Rocketlogger was connected for low side measurements and GPIO flags were used to mask the power trace and calculate

the kernel energy. For our two sensor prototypes, the only difference is in the image acquisition kernel; all other kernels are the same. Table 4.2 shows the energy costs of kernels using external peripherals. As expected, the thermal sensor consumes much more energy than the visual camera. It should be noted that this energy already includes the overhead for turning on the camera. In the case of the Lepton camera, this start-up energy actually represents over 90% of the total cost.

The energy breakdown of the compiled application can be seen in the right column of Table 4.1. The execution of the CNN has a computational complexity of ~50k multiply-accumulate operations for each window in the input pyramid, for a total of 16k windows for the 2×2 stride considered in Section 4.7.2; ~7300 windows if we consider a bigger 3×3 stride. For batteryless operation, we select the 3×3 stride since it has lower processing needs.

Table 4.3 shows the energy necessary to perform the full algorithm including the bulk of CNN computation as well as the pyramid construction, the non-maximum suppression and the final thresholding. These measurements were based on running the kernels over 100 times. In the case of the CNN kernels, they were executed over 1000 times. For all kernels, we took the maximum measured energy.

Kernel	Energy per Kernel [mJ]
Thermal Image Acquisition	131
Visual Image Acquisition	4.4
BLE Transmission	0.086

Table 4.2: Energy costs for kernels using external peripherals. Measurements were done with V_{DD}=2.8 V.

4.7.3 Partitioning Results

Once the energy requirements of the individual kernels are known, we can proceed to choose the execution configuration. We will evaluate three different algorithms to partition the sequential head-counting application into bursts. We will

Kernel	E_{kernel} [mJ]	N_{tasks}	E_{sum} [mJ]
Normalize	0.043	1	0.043
Initialize	0.003	1	0.003
CNN1	0.396	4125	1 633.5
CNN2	0.396	936	370.7
CNN3	0.403	391	157.6
Sort	0.010	1	0.010
NMS	0.006	1	0.006
Total head-counting			2 162

Table 4.3: Energy cost of processing kernels during one complete head-counting application execution (3×3 stride). Measurements were done with V_{DD}=2.8 V @ 80 MHz.

compare *Julienning* to two fixed partitioning schemes: *Single Task* and *Whole Application*. These partitioning schemes were discussed in Section 4.5. In *Single Task* partitioning, each kernel execution is assigned its own burst and state retention is not optimized, meaning every burst will save and restore all application data. With *Whole Application*, all kernels are assigned to a single burst. Due to the image acquisition kernel, which is the energy-dominant kernel in both versions of the application, a minimum storage capacity of $E_{acquistion} + E_{save}$, is required to acquire an image and save it in NVM reliably.

Figure 4.7 shows the results for the three partitioning schemes in different figures of merit. *Single Task* partitioning uses the lowest possible storage capacity, but due to its inefficient state retention scheme, it will end up transferring over 438 MB of data over its 5458 bursts. The energy overhead ends up being larger than the application energy itself. *Whole Application* minimizes the data overhead since everything is executed within a single burst no state retention is necessary. However, the overhead is actually the required storage capacity. Since batteryless sensing systems are designed to minimize storage capacity, *Whole Application* scheme is not a scalable solution. *Julienning* is able to have the minimum feasible storage capacity, like *Single Task*, and have a very low data and burst overhead, similar to *Whole Application*. Since

every kernel was specified with explicit data dependencies, every burst loads the data required for its own kernels. Furthermore, our optimization algorithm can group together multiple kernels together such that all energy bursts are as close a possible to the minimum storage capacity. In the end, this reduces the number of bursts to just 16, as opposed to 5458 with *Single Task*. Boot-up and data overheads are only 2 mJ, or less than 0.1% of the application energy.

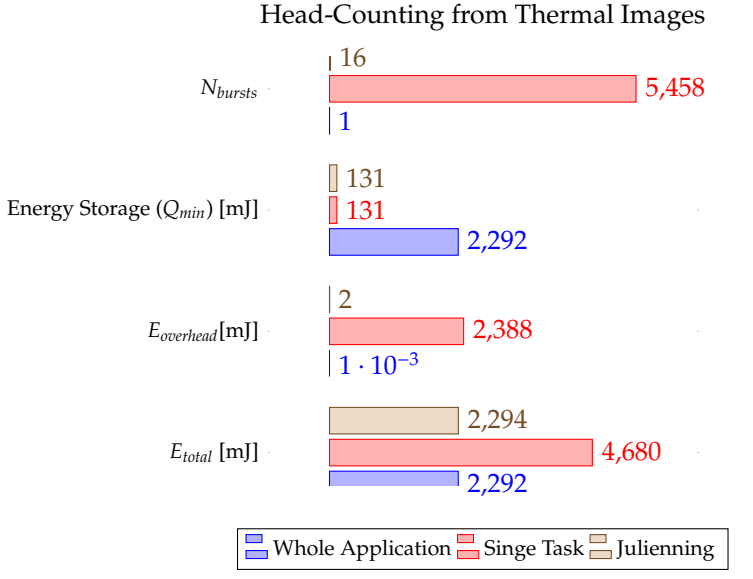


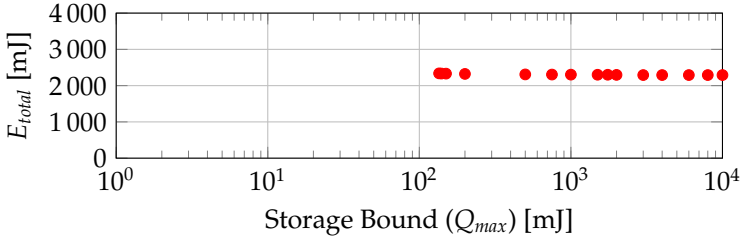
Figure 4.7: Figures of merit when partitioning the thermal head detection application using three algorithms. Julienning was evaluated with Q_{max}=131 mJ, or the largest atomic task. Results show that by splitting the application into 16 optimized bursts, Julienning reduces energy storage by more than 17× compared to *Whole Application* partitioning, and increases the total energy by only 0.08%.

Design Space Exploration

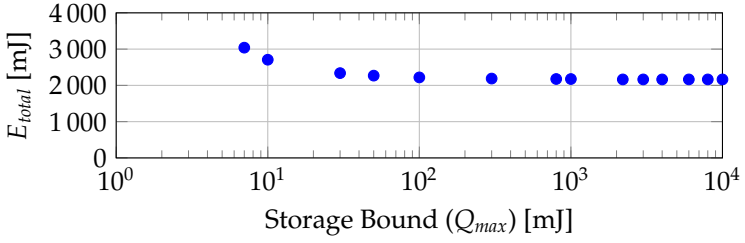
To further illustrate the flexibility of our *Julienning* method, we now look at the design exploration of both the visual and thermal head-counting applications. The energy characterization of all individual kernels has been shown in Table 4.2 and Table 4.3. In previous sections, we saw how basic partitioning schemes are kernel-based. In other words, a certain number of kernels can be grouped in the same burst. These fixed kernel schemes were shown to be inefficient, leading either to large data overhead or very large storage capacities. With *Julienning*, we can select an energy burst bound (Q_{max}) and it will find the partitioning that will minimize the total application energy within that bound. In addition, *Julienning* can find the minimum required energy storage capacity (Q_{min}), which can be lower than the bound. These values depend on several parameters including the weights of the kernels, the data transfer costs, and the boot-up costs. In the following, we will see how the Q_{max} parameter will impact the main design metrics, namely: E_{total}, N_{bursts}, and the Q_{min}. For each application, each point at a given Q_{max} belongs to the same solution.

Figure 4.8 shows the results from *Julienning* when applied to the thermal (Figure 4.8(a)) and visual (Figure 4.8(b)) head counting applications. The X axis is the Q_{max} used in each optimization run. Each point represents the optimized partitioning and its resulting E_{total}, which includes boot-up, data transfer and application energy. It should first be noted that the visual application has a wider feasibility range. This is due to the fact that the dominant kernel, the visual image acquisition, requires relatively little energy. In the thermal image version, the minimum required storage is already quite high (131 mJ). As expected, the results show that as the storage bound is increased, the total energy is reduced, and it levels off when the entire application is executed within a single burst.

Figure 4.9 shows the design space exploration using *Julienning* on the thermal (Figure 4.9(a)) and visual (Figure 4.9(b)) head counting applications. The X-axis is the Q_{max} used in each optimization run, and the Y-axis is the optimal N_{burst}.



(a) Thermal head counting design space.

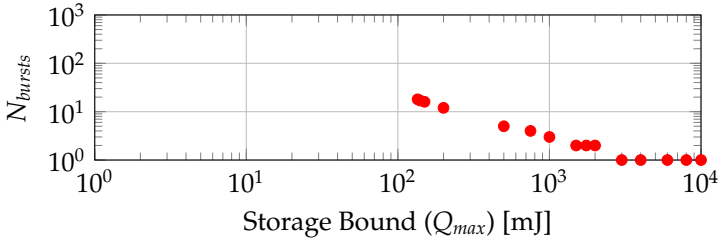


(b) Visual head counting design space.

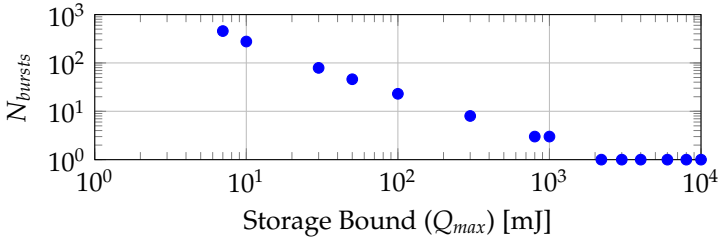
Figure 4.8: Total energy obtained when applying *Julienning* to thermal and visual head counting applications.

Once again, the visual application exhibits a wider feasibility range due to its fine-grained kernels. This allows *Julienning* to partition the visual application into 456 bursts, while incurring an 875.6 mJ overhead. N_{bursts} decreases monotonically with Q_{max}, which is expected as doing so reduces boot-ups and data transfers. Once $Q_{max} > E_{app} + E_{boot-up}$, the optimal N_{bursts} is always 1.

Lastly, Figure 4.10 depicts the design space exploration using *Julienning* on the thermal (Figure 4.10(a)) and visual (Figure 4.10(b)) head counting applications. The Y-Axis shows the normalized storage Q_{min}/Q_{max} as a function of the storage bound (Q_{max}). The behavior here once again has the boundary condition at $Q_{max} = E_{app} + E_{boot-up}$. If Q_{max} is below this threshold, *Julienning* is able to make full use of the storage bound. This, however, strictly depends on the granularity of



(a) Thermal head counting design space.



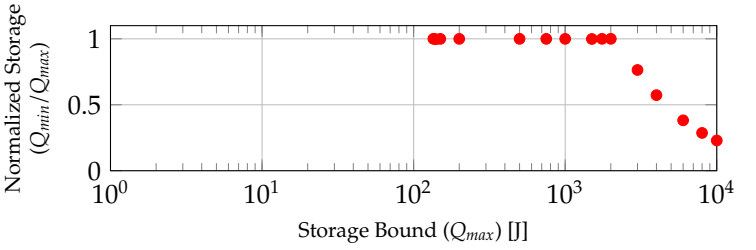
(b) Visual head counting design space.

Figure 4.9: N_{bursts} obtained when applying *Julienning* to thermal and visual head counting applications.

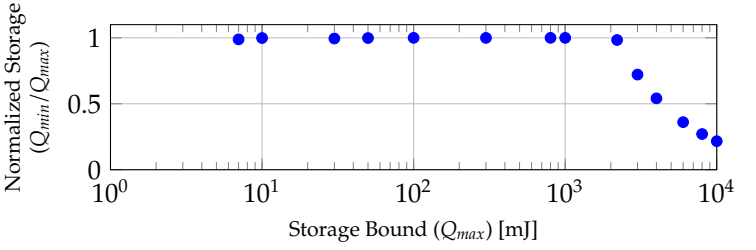
the kernels. In both of our head counting applications, the CNN's constitute the vast majority of the kernels and when added also dominate the total energy consumption. As such, most energy bursts are very close in size. In other applications which much greater variability, this would not be the case. If Q_{max} is above the application threshold, then Q_{min} will always stay the same.

4.7.4 Analysis of Partitioning Results

EMU-based designs are guaranteed to have reliable execution thanks to the safely calculated storage capacity. In previous chapters, we have seen applications which consisted of few tasks whose energy costs were in the sub-mJ range. Here, we face a more daunting challenge with a CNN-based head



(a) Thermal head counting design space.



(b) Visual head counting design space.

Figure 4.10: Normalized storage resulting from applying *Julienning* to thermal and visual head counting applications.

detection application consisting of thousands of kernels and consuming a few Joules. Burst-based execution follows a traditional initialize-run-deinitilize cycle, as was shown in Figure 2.4. Using *Julienning*, we automate the generation of optimized code for batteryless operation.

Our optimization model takes as input the application graph with data dependencies and energy costs. Our tool then determines the partitioning which minimizes the total energy, satisfying a storage capacity bound. This capacity bound can be used by application designers to scan the entire design space and easily prune the application graph of infeasible partitions. Furthermore, *Julienning* can determine the storage device for the optimal partition, which can be significantly below the capacity bound.

Our proposed methodology has been demonstrated with two prototypes of the head detection system. They have very different hardware requirements since the thermal version uses a power-hungry infrared sensor, and the vision-based uses a lower power device. The relative energy cost of these kernels has a direct impact on the capacity reduction. In our application, loading and saving has a low energy cost compared to the actual kernels. This is due to both the low-power FRAM as well as the power intensive convolutional workload. Since kernels are executed atomically, the maximum reduction is given by the following ratio:

$$\frac{max\left\{E_{load,i} + E_{kernel,i} + E_{save,i}\right\}}{\sum_i E_{kernel,i}} \tag{4.18}$$

The numerator is the largest possible (single-kernel) burst, and the denominator is the total application energy. For this reason, the thermal application had a more limited feasibility range than the visual application. It should be noted that if the kernels are very small and $E_{load} + E_{save}$ dominate burst energy, single-kernel bursts will not be as energy efficient as multi-kernel bursts with lower data transfers.

The generality of EMU-based designs allowed both head detection systems to be developed independently from transducer characteristics and guarantee reliable execution. In fact, the only difference between them is the storage capacity (i.e. capacitor size). We have seen how different capacity bounds affect important figures of merit like energy overhead, number of bursts and minimum storage. Thanks to *Julienning*, arbitrarily long applications can be executed in reliable energy bursts with optimized data transfers and minimized energy.

4.8 Summary

In this chapter, we have seen the design and implementation of two head counting applications: one based on visual images and another on infrared images. These applications were implemented in a low-power platform with stringent memory

constraints. To enable EMU-based batteryless operation, we have developed *Julienning*, an optimization flow to partition large sequential applications into energy bursts with bounded size and minimized energy overhead. To use *Julienning*, software must be specified using a data-flow model with explicit data dependencies. Using experimentally characterized energy costs, we transform the partitioning problem into a shortest path problem, easily solvable in polynomial time. Due to explicit data dependencies, *Julienning* optimizes the data transfer for every burst, loading only the data required for burst execution and saving only the data required by future kernels. In batteryless systems, where storage capacity needs to be minimized, *Julienning* can be used to rapidly explore the design space of an arbitrarily long sequential application. Using the thermal head detection as a benchmark, our proposed methods can reduce the energy storage by 17× compared to no partitioning, while incurring less than 0.1% energy overhead.

5

Conclusion and Outlook

This chapter summarizes the contributions of this thesis and outlines possible future research directions.

5.1 Contributions

The aim of this thesis is to show that reliable execution of batteryless sensing applications is possible, even when harvesting conditions are not compatible with sustained system operation. Though the designer cannot directly control how much energy is available, it is possible to control under what conditions the application executes. By buffering small amounts of energy, designers can control the operating voltage and maximize the amount of work done per unit of energy. To support these claims, we have proposed four main building blocks and applied them to multiple vision-based sensing systems.

Energy Management Unit (EMU)
The power generation of transducers depends on both their size and their environment. Even large transducers can have voltage and current ranges that are far below the requirements

of low-power sensing systems. In these scenarios, it is necessary to decouple the transducer's voltage and current from that of the load. Our proposed Energy Management Unit (EMU) uses a boost-buck topology to effectively decouple the source from the load. In doing so, it also allows for the possibility to independently adjust the voltage of both the source and the load. On the source side, the commercial boost converters already implement maximum power point tracking algorithms. The EMU efficiently builds up charge in a small capacitor to predefined energy levels and supply the load with a short, high power energy burst. By using an optimized capacitor, the load can reliably execute power-hungry tasks of up to 1 W even if the harvested power was intermittent, as long as it can generate at least 20 μW.

Dynamic Energy Burst Scaling (DEBS)
While Energy Management Units (EMUs) can efficiently accumulate charge in dynamic and adverse environments, by default they generate energy bursts of constant size and at a constant supply voltage. In certain applications consisting of a single atomic task with only one voltage requirement, this configuration is enough for efficient operation and minimized capacity. However, in many application circuits have external peripherals like sensors, non-volatile memories and radios, which can have very different operating voltage ranges. In addition, applications composed of multiple atomic tasks can be manually split into multiple tasks

Non-Volatile Memory Hierarchy (NVMH)
In many applications peripherals with very high initialization costs can dominate an application's energy requirements. Long-term logging using Flash memory is just one example. By using novel memory technologies such as FRAM, we can form a Non-Volatile Memory Hierarchy (NVMH) that reduces the average energy cost of storing data. We demonstrate an important design trade-off between the average energy cost per logged measurement and the minimum required capacitance to reliably execute the logging application.

Julienning
This optimization framework can be applied to batteryless applications. The applications consisting of an arbitrarily long chain of computational kernels can be automatically optimized for minimal application energy, given a storage element bound. Using a custom specification language, data dependencies between finely-grained kernels can be accurately calculated such that only the essential data is transferred between volatile and non-volatile domains. Our optimization formulation allows the designer to efficiently explore the design space of batteryless applications to identify the Pareto front.

5.2 Possible Future Directions

Sensing systems offer the possibility of obtaining valuable data from physical environments. Batteryless sensing systems can do the same with unparalleled scalability thanks to a reduction of monetary, environmental and maintenance costs brought by using tiny storage elements. We have shown how vision sensors, which are typically energy and data-intensive, can effectively run many complex applications in a batteryless fashion. In places where the presence of light correlates with the presence of people, as is the case in many indoor environments, batteryless vision sensors can play an important role. There is, however, the need to identify other scenarios where true deploy-and-forget batteryless systems can leverage simultaneous availability of energy and information. Though this thesis has introduced key building blocks in the design and specification of such systems, much work can be done to improve the responsiveness and efficiency in specific scenarios.

Improving cold-start
Cold-start is a critical phase of any harvesting-based system. By definition, the energy harvested during this phase is not transferred to the load, but it is used by the harvesting system itself before entering a steady operating state. Electronically speaking, it is difficult to design circuits that behave well down to 0 V. Current designs clamp the transducer's voltage

at 330 mV and rely on there being enough current to charge up internal capacitors. This charging phase is typically inefficient and can lead to long charge times depending on the storage element being used. Novel converter circuits are necessary to increase the energy efficiency during this critical phase. Systems with optimized cold-start would be much more responsive, which could be a critical factor in highly dynamic systems.

Multi-source harvesting

Up until now, all of our EMU designs have been single-source. As we have seen, adding different types of energy sources would increase the probability of harvesting more energy. For example, a small weather station with a solar panel and a windmill could, in theory, measure luminosity and wind speed in a batteryless and self-sustainable fashion. Building such a system and optimizing it for inexpensive and efficient operation is challenging. AC and DC sources have different needs in rectification, voltage conversion, impedance matching and tracking. Pooling the energy is a single storage might not always be the best option. Specialized circuitry will then be needed for generating energy bursts from multiple storage elements.

Dynamic energy management

Applications with multiple tasks and peripherals can have multiple operating points. Not only can different tasks have different voltage ranges, but their power consumption can also be very different. The average active power ($\bar{P}_{load,active}$) naturally depends on the individual power levels of different tasks. When $P_{in} << \bar{P}_{load,active}$, the EMU can very efficiently duty-cycle the load using energy bursts. However, if $P_{in} \approx \bar{P}_{load,active}$ the EMU can, for a moment, saturate its storage element because it was executing a lower power task. When this saturation happens, the boost converter shuts off and no energy is harvested to protect the storage element. There is no way to avoid this is the load is statically managed,

meaning the order in which tasks are executed is constant. The only way to avoid this saturation is to introduce some level of power awareness, possibly by sampling the capacitor voltage and estimating the input power. With this information, a dynamic scheduler can choose an appropriate task such that the load's power is always higher than the input power. Assuming P_{in} is always less than $P_{load,max}$, dynamic scheduling can effectively guarantee that saturation never happens. For the application to dynamically adapt its task execution, and thus its power consumption, there needs to be a large enough memory buffer such that the outputs of a particular task can be stored. Input power awareness can also be useful to identify low power scenarios. Assuming the load is compatible with approximative computing, there is a correlation between result quality and spent energy. A dynamic scheduler would then be able to choose a lower quality result when energy is scarce. In effect, this would allow a batteryless system to adjust its service in quality and not just duty-cycle adjustment.

Bibliography

[AAB+15] M. Abadi, A. Agarwal, P. Barham, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[ACK14] ACKme Networks. *AMS001/AMS002 Datasheet*, 2014. Preliminary Datasheet.

[AE10] H. Alemdar and C. Ersoy. Wireless sensor networks for healthcare: A survey. *Computer networks*, 54(15):2688–2710, 2010.

[AM15] K. Ahmed and S. Mukhopadhyay. A 190na bias current 10mv input multi-stage boost regulator with intermediate node control to supply rf blocks in self-powered wireless sensors. *IEEE Trans. Power Electronics*, PP(99), 2015.

[BAB13] B. Buchli, D. Aschwanden, and J. Beutel. Battery state-of-charge approximation for energy harvesting embedded systems. In *European Conference on Wireless Sensor Networks*, pages 179–196. Springer, 2013.

[Bar04] A. Barjatya. Block matching algorithms for motion estimation. *Trans. Evolution Computation*, 2004.

[BASM16] N. A. Bhatti, M. H. Alizai, A. A. Syed, and L. Mottola. Energy harvesting and wireless transfer in sensor network applications: Concepts and experiences. *ACM Transactions on Sensor Networks (TOSN)*, 12(3):24, 2016.

[BB95] S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM computing surveys (CSUR)*, 1995.

[BBMP13] J. Bates, M. Beaulieu, M. Miller, and J. Paulus. Reaching the Highest Reliability for Tantalum Capacitors. Technical report, 2013.

[BBMT08] D. Brunelli, L. Benini, C. Moser, and L. Thiele. An Efficient Solar Energy Harvester for Wireless Sensor Nodes. In *Proc. DATE Conf.*, 2008.

[BBY13] G. C. Buttazzo, M. Bertogna, and G. Yao. Limited preemptive scheduling for real-time systems. a survey. *IEEE Transactions on Industrial Informatics*, 9(1):3–15, Feb 2013.

[BC11] S. Bandyopadhyay and A. P. Chandrakasan. Platform architecture for solar, thermal and vibration energy combining with mppt and single inductor. In *VLSI Circuits (VLSIC), 2011 Symposium on*, pages 238–239. IEEE, 2011.

[BCMS01] L. Benini, G. Castelli, A. Macii, and R. Scarsi. Battery-driven dynamic power management. *IEEE Design & Test of Computers*, 18(2):53–60, 2001.

[BDR17] M. Bolanos, M. Dimiccoli, and P. Radeva. Toward storytelling from visual lifelogging: An overview. *IEEE Transactions on Human-Machine Systems*, 47(1):77–90, 2017.

[Ben17] Benewake Co., Ltd. *TFmini Infrared Module Specification*, 2017. Version A00.

[BGW11] M. Buettner, B. Greenstein, and D. Wetherall. Dewdrop: an energy-aware runtime for computational rfid. In *Proc. USENIX NSDI*, pages 197–210, 2011.

[BJJ18] Y. Bai, H. Jantunen, and J. Juuti. Energy harvesting research: The road from single source to multisource. *Advanced Materials*, page 1707271, 2018.

[BKC+13] S. C. Bartling, S. Khanna, M. P. Clinton, S. R. Summerfelt, J. A. Rodriguez, and H. P. McAdams. An 8MHz 75uA/MHz zero-leakage non-volatile logic-based Cortex-M0 MCU SoC exhibiting 100% digital state retention at VDD=0V with <400ns

wakeup and sleep transitions. In *Proc. ISSCC Conf.* IEEE, feb 2013.

[BM16] N. Bhatti and L. Mottola. Efficient state retention for transiently-powered embedded sensing. In *International Conference on Embedded Wireless Systems and Networks*, pages 137–148, 2016.

[BM17] N. A. Bhatti and L. Mottola. Harvos: Efficient code instrumentation for transiently-powered embedded sensing. In *Proc. IPSN Conf.*, New York, NY, USA, 2017. ACM.

[BMB+17] R. Bolt, M. Magno, T. Burger, A. Romani, and L. Benini. Kinetic ac/dc converter for electromagnetic energy harvesting in autonomous wearable devices. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 64(12):1422–1426, 2017.

[Bos15] Bosch Sensortech. *Digital, triaxial acceleration sensor*, 2015. Version 1.0.

[BSB+13] H. Besbes, G. Smart, D. Buranapanichkit, C. Kloukinas, and Y. Andreopoulos. Analytic conditions for energy neutrality in uniformly-formed wireless sensor networks. *IEEE transactions on wireless communications*, 12(10):4916–4931, 2013.

[BSBT14] B. Buchli, F. Sutton, J. Beutel, and L. Thiele. Towards enabling uninterrupted long-term operation of solar energy harvesting embedded systems. In *European Conference on Wireless Sensor Networks*, pages 66–83. Springer, 2014.

[BWM+15] D. Balsamo, A. S. Weddell, G. V. Merrett, B. M. Al-hashimi, D. Brunelli, and L. Benini. Hibernus : Sustaining Computation during Intermittent Supply for Energy-Harvesting Systems. *Embed. Syst. Lett. IEEE*, 7(1), 2015.

[CC00] T. Christen and M. W. Carlen. Theory of ragone plots. *Journal of power sources*, 91(2):210–216, 2000.

[Cen13] Centeye Inc. *Stonyman and Hawksbill vision chips*, 2013. Rev. 1.0.

[Cho15] F. Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[CL16] A. Colin and B. Lucia. Chain: Tasks and channels for reliable intermittent programs. In *Proc Object-Oriented Programming, Systems, Languages, and Applications Conf.* ACM, 2016.

[CL18] A. Colin and B. Lucia. Termination checking and task decomposition for task-based intermittent programs. In *Proc. Compiler Construction Conference*, New York, NY, USA, 2018. ACM.

[CMB15] L. Cavigelli, M. Magno, and L. Benini. Accelerating real-time embedded scene labeling with convolutional networks. In *Proceedings of the 52Nd Annual Design Automation Conference*, DAC '15, pages 108:1–108:6, New York, NY, USA, 2015. ACM.

[Col18] A. Colin. *System Support for Intermittent Computing*. PhD thesis, Carnigie Mellon University, 2018. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2018-06-14.

[CPA+17] F. Conti, D. Palossi, R. Andri, M. Magno, and L. Benini. Accelerated visual context classification on a low-power smartwatch. *IEEE Transactions on Human-Machine Systems*, 47(1):19–30, 2017.

[Cra17] G. M. Crawley. *Energy Storage*. World Scientific, 2017.

[CRL18] A. Colin, E. Ruppel, and B. Lucia. A reconfigurable energy storage architecture for energy-harvesting devices. In *Proc. Architectural Support for Programming Languages and Operating Systems Conf.* ACM, 2018.

[CSB92] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen. Low-power cmos digital design. *IEICE Transactions on Electronics*, 75(4):371–382, 1992.

[Cyp15] Cypress Semiconductor Corportation. *1-mbit Serial F-RAM Datasheet*, 8 2015. Rev. E.

[DBL+15] E. Dallago, A. Barnabei, A. Liberale, P. Malcovati, and G. Venchi. An interface circuit for low-voltage low-current energy harvesting systems. *IEEE Trans. Power Electronics*, 30(3), 2015.

[Dig09] Digi International. *XBee ®RF Modules Product Manual*, 9 2009.

[DLBL+15] E. Dallago, A. Lazzarini Barnabei, A. Liberale, G. Torelli, and G. Venchi. A 300 mv low-power management system for energy harvesting applications. *IEEE Trans. Power Electronics*, PP(99), 2015.

[DRF+15] M. Dini, A. Romani, M. Filippi, V. Bottarel, G. Ricotti, and M. Tartagni. A nanocurrent power management ic for multiple heterogeneous energy harvesting sources. *IEEE Transactions on Power Electronics*, 30(10):5665–5680, 2015.

[DXS+18] R. Dekimpe, P. Xu, M. Schramme, D. Flandre, and D. Bol. A battery-less ble iot motion detector supplied by 2.45-ghz wireless power transfer. In *Proc. PATMOS Symposium*. IEEE, 2018.

[ELE06] N. G. Elvin, N. Lajnef, and A. A. Elvin. Feasibility of structural monitoring with vibration powered sensors. *Smart materials and structures*, 15(4):977, 2006.

[Far01] G. Farnebäck. Very high accuracy velocity estimation using orientation tensors, parametric motion, and simultaneous segmentation of the motion field. In *Proc. ICCV Conf.* IEEE, 2001.

[FLI18] FLIR Systems, Inc. *Lepton Engineering Datasheet*, 2018. Rev. 200.

[GB08] J. M. Gilbert and F. Balouchi. Comparison of energy harvesting systems for wireless sensor networks. *International Journal of automation and computing*, 5(4):334–347, 2008.

[GFM] R. B. Girshick, P. F. Felzenszwalb, and D. McAllester. Discriminatively trained deformable part models, release 5. `http://people.cs.uchicago.edu/~rbg/latent-release5/`.

[GHS+17] F. Glaser, Q. Huang, P. Schonle, P. Meier, J. Bosser, N. Brun, T. Burger, S. Fateh, G. Rovere, and L. Benini. Towards a mobile health platform with parallel processing and multi-sensor capabilities. In *2017 Euromicro Conference on Digital System Design (DSD)*, pages 462–469. IEEE, 2017.

[Gib50] J. J. Gibson. The perception of the visual world. 1950.

[GIS10] X. Guo, E. Ipek, and T. Soyata. Resistive computation: Avoiding the power wall with low-leakage, stt-mram based computing. In *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ISCA '10, pages 371–382, New York, NY, USA, 2010. ACM.

[GPB+15] A. Gomez, C. Pinto, A. Bartolini, D. Rossi, L. Benini, H. Fatemi, and J. P. de Gyvez. Reducing energy consumption in microcontroller-based platforms with low design margin co-processors. In *Proc. DATE*, 2015.

[GSM+16] A. Gomez, L. Sigrist, M. Magno, L. Benini, and L. Thiele. Dynamic energy burst scaling for transiently powered systems. In *Proc. DATE Conf.*, pages 349–354. EDA Consortium, 2016.

[GSS+17a] A. Gomez, L. Sigrist, T. Schalch, L. Benini, and L. Thiele. Efficient, long-term logging of rich data sensors using transient sensor nodes. 2017.

[GSS+17b] A. Gomez, L. Sigrist, T. Schalch, L. Benini, and L. Thiele. Wearable, energy-opportunistic vision sensing for walking speed estimation. In *Proc. SAS Symp.*, pages 1–6. IEEE, 2017.

[GVMNGPG14] J. Gutiérrez, J. F. Villa-Medina, A. Nieto-Garibay, and M. Á. Porta-Gándara. Automated irrigation system using a wireless sensor network and gprs module. *IEEE transactions on instrumentation and measurement*, 63(1):166–176, 2014.

[HBD13] S. M. Hatch, J. Briscoe, and S. Dunn. A self-powered zno-nanorod/cuscn uv photodetector exhibiting rapid response. *Advanced Materials*, 25(6):867–871, 2013.

[HFdGB17] P. A. Hager, H. Fatemi, J. P. de Gyvez, and L. Benini. A scan-chain based state retention methodology for iot processors operating on intermittent energy. In *Proc. DATE Conf.* EDA Consortium, 2017.

[HMTP13] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys. An open source and open hardware embedded metric optical flow cmos camera for

indoor and outdoor applications. In *Proc. ICRA Conf.* IEEE, 2013.

[HS81] B. K. Horn and B. G. Schunck. Determining optical flow. In *Artificial Intelligence 17*, 1981.

[HSS15a] J. Hester, L. Sitanayah, and J. Sorber. Tragedy of the coulombs: Federating energy storage for tiny, intermittently-powered sensors. In *Proc. SenSys Conf.*, pages 5–16, 2015.

[HSS15b] J. Hester, L. Sitanayah, and J. Sorber. Tragedy of the coulombs: Federating energy storage for tiny, intermittently-powered sensors. In *Proc. Embedded Networked Sensor Systems Conf.* ACM, 2015.

[HSS17] J. Hester, K. Storer, and J. Sorber. Timely execution on intermittently powered batteryless sensors. In *Proc. SenSys Conf.*, 2017.

[Hug10] R. A. Huggins. *Energy Storage*, volume 391. Springer, 2010.

[HXZ+13] J. Hu, C. J. Xue, Q. Zhuge, W.-C. Tseng, and E. H.-M. Sha. Write activity reduction on non-volatile main memories for embedded chip multiprocessors. *ACM Trans. Embed. Comput. Syst.*, 12(3):77:1–77:27, April 2013.

[HZK+06] J. Hsu, S. Zahedi, A. Kansal, M. Srivastava, and V. Raghunathan. Adaptive duty cycling for energy harvesting systems. In *Proceedings of the 2006 international symposium on Low power electronics and design*, pages 180–185. ACM, 2006.

[JLM10] V. Jain and E. Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst, 2010.

[JRR14] H. Jayakumar, A. Raha, and V. Raghunathan. QUICKRECALL: A Low Overhead HW/SW Approach for Enabling Computations across Power Cycles in Transiently Powered Computers. *Proc. Int. Conf. VLSI Design*, 2014.

[JW14] S. John Walker. Big data: A revolution that will transform how we live, work, and think, 2014.

[KBC+14] S. Khanna, S. C. Bartling, M. Clinton, S. Summerfelt, J. A. Rodriguez, and H. P. McAdams. An FRAM-Based Nonvolatile Logic MCU SoC Exhibiting 100% Digital State Retention at VDD=0 V Achieving Zero Leakage With <400-ns Wakup Time for ULP Applications. *IEEE J. Solid-State Circuits*, 49(1), jan 2014.

[KCWP10] Y. Kim, N. Chang, Y. Wang, and M. Pedram. Maximum power transfer tracking for a photovoltaic-supercapacitor energy system. In *Proc. ISLPED*, pages 307–312, 2010.

[KEM18] KEMET Electronics Corporation. *T52X/T530 Polymer Electrolytic Capacitors*, 2018.

[KLCL16] M.-L. Ku, W. Li, Y. Chen, and K. R. Liu. Advances in energy harvesting communications: Past, present, and future challenges. *IEEE Communications Surveys & Tutorials*, 18(2):1384–1412, 2016.

[LBBH98] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LBL+13] Y. Lee, S. Bang, I. Lee, Y. Kim, G. Kim, M. H. Ghaed, P. Pannuto, P. Dutta, D. Sylvester, and D. Blaauw. A modular 1 mm^3 die-stacked sensing platform with low power i2c inter-die communication and multi-modal energy harvesting. *IEEE Journal of Solid-State Circuits*, 48(1):229–243, 2013.

[LC15] H. G. Lee and N. Chang. Powering the iot: Storage-less and converter-less energy harvesting. In *Proc. ASP-DAC Conf.*, 2015.

[LKWH07] M. Löhndorf, T. Kvisterøy, E. Westby, and E. Halvorsen. Evaluation of energy harvesting concepts for tire pressure monitoring systems. *Proceedings of Power MEMS*, pages 331–334, 2007.

[LLS+15] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *Proc. CVPR Conf.*, June 2015.

[LPRR10] C. Lu, S. P. Park, V. Raghunathan, and K. Roy. Efficient power conversion for ultra low voltage

micro scale energy transducers. In *Des. Autom. Test Eur. Conf. Exhib. (DATE), 2010*, 2010.

[LR15] B. Lucia and B. Ransford. A simpler, safer programming and execution model for intermittent systems. In *Proc. Programming Language Design and Implementation Conf.*, PLDI '15, 2015.

[LWL+16] Y. Liu, Z. Wang, A. Lee, F. Su, C.-P. Lo, Z. Yuan, C.-C. Lin, Q. Wei, Y. Wang, Y.-C. King, C.-J. Lin, P. Khalili, K.-L. Wang, M.-F. Chang, and H. Yang. A 65nm ReRAM-enabled nonvolatile processor with 6x reduction in restore time and 4x higher clock frequency using adaptive data retention and self-write-termination nonvolatile logic. In *Proc. ISSCC Conf.* IEEE, jan 2016.

[MB11] M. Meli and U. Beerli. Indoor battery-less temperature and humidity sensor for Bluetooth Low Energy. Technical Report November, Zürcher Hochschule für Angewandte Wissenschaften (ZHAW), 2011.

[MB16] M. Meli and P. Bachmann. Powering Long Range Wireless Nodes with Harvested Energy. Technical Report November, Zürcher Hochschule für Angewandte Wissenschaften (ZHAW), 2016.

[MBS+16] M. Magno, D. Brunelli, L. Sigrist, R. Andri, L. Cavigelli, A. Gomez, and L. Benini. Infinitime: Multi-sensor wearable bracelet with human body harvesting. *Sustainable Computing: Informatics and Systems*, 11:38–49, 2016.

[Mid17] Midé Technology Corporation. *PPA-1011 Product Datasheet and User Manual*, 1 2017. Rev. 3.

[MIR04] V. S. Mallela, V. Ilankumaran, and N. S. Rao. Trends in cardiac pacemaker batteries. *Indian pacing and electrophysiology journal*, 4(4):201, 2004.

[MKK+15] V. More, H. Kumar, S. Kaingade, P. Gaidhani, and N. Gupta. Visual odometry using optic flow for unmanned aerial vehicles. In *Proc. CCIP Conf.* IEEE, 2015.

[MMB+12] M. Magno, S. Marinkovic, D. Brunelli, E. Popovici, B. O'Flynn, and L. Benini. Smart power unit with ultra low power radio trigger capabilities for wireless sensor networks. In *Proc. DATE Conf.*, 2012.

[MML18] J. Ma, Y. Ma, and C. Li. Infrared and visible image fusion methods and applications: A survey. *Information Fusion*, 2018.

[MTBB10] C. Moser, L. Thiele, D. Brunelli, and L. Benini. Adaptive power management for environmentally powered systems. *IEEE Transactions on Computers*, 59(4):478–491, 2010.

[MV16] S. Mittal and J. S. Vetter. A survey of software techniques for using non-volatile memories for storage and main memory systems. *IEEE Trans. Parallel Distr. Syst.*, 27(5):1537–1550, May 2016.

[NHP03] G. Ning, B. Haran, and B. N. Popov. Capacity fade study of lithium-ion batteries cycled at high discharge rates. *Journal of Power Sources*, 117(1-2):160–169, 2003.

[NPK+15] S. Naderiparizi, A. N. Parks, Z. Kapetanovic, B. Ransford, and J. R. Smith. WISPCam : A Battery-Free RFID Camera. In *Proc. IEEE RFID*, 2015.

[NSF15] S. K. Nayar, D. C. Sims, and M. Fridberg. Towards self-powered cameras. In *2015 IEEE International Conference on Computational Photography (ICCP)*, pages 1–10, April 2015.

[NXP15] NXP Semiconductors. *LPC5410X Product Data Sheet*, 2015. Rev 2.2.

[NZM09] A. Nasiri, S. A. Zabalawi, and G. Mandic. Indoor power harvesting using photovoltaic cells for low-power applications. *IEEE Transactions on Industrial Electronics*, 56(11):4502–4509, 2009.

[OHP15] H. Oleynikova, D. Honegger, and M. Pollefeys. Reactive avoidance using embedded stereo vision for mav flight. In *Proc. ICRA Conf.* IEEE, 2015.

[Omn06] OmniVision. *OV7670/OV7171 CMSO VGA CameraChip Sensor Preliminary Datasheet*, 2006. Version 1.4.

[OMTH15] N. Onizawa, A. Mochizuki, A. Tamakoshi, and T. Hanyu. A sudden power-outage resilient nonvolatile microprocessor for immediate system recovery. In *Proc. Nanoscale Architectures Symp.*, July 2015.

[OVV17] K. Ozcan, S. Velipasalar, and P. K. Varshney. Autonomous fall detection with wearable cameras by using relative entropy distance measure. *IEEE Trans. Human-Machine Systems*, 47(1):31–39, 2017.

[PBB98] T. Pering, T. Burd, and R. Brodersen. The simulation and evaluation of dynamic voltage scaling algorithms. In *Proceedings of the 1998 international symposium on Low power electronics and design*, pages 76–81. ACM, 1998.

[Pow] Powerfilm Solar. *MP3-25 and MP3-37 Product Pages.* https://www.powerfilmsolar.com. Accessed July 2018.

[QAC14] M. Qazi, A. Amerasekera, and A. P. Chandrakasan. A 3.4-pj feram-enabled d flip-flop in 0.13-μm CMOS for nonvolatile processing in digital systems. *IEEE J. Solid-State Circuits*, 49(1):202–211, jan 2014.

[RBL+17] A. Rodriguez, D. Balsamo, Z. Luo, S. P. Beeby, G. V. Merrett, and A. S. Weddel. Intermittently-powered energy harvesting step counter for fitness tracking. In *Sensors Applications Symposium (SAS), 2017 IEEE*, pages 1–6. IEEE, 2017.

[RBMW18] A. Rodriguez Arreola, D. Balsamo, G. V. Merrett, and A. S. Weddell. RESTOP: Retaining External Peripheral State in Intermittently-Powered Sensor Systems. *Sensors*, 18(1), 2018.

[RKH+05] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. Srivastava. Design considerations for solar energy harvesting wireless embedded systems. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, page 64. IEEE Press, 2005.

[RSF11] B. Ransford, J. Sorber, and K. Fu. Mementos: System support for long-running computation on rfid-scale devices. *SIGPLAN Not.*, 46(3), March 2011.

[San04] SanDisk Corporation. *SanDisk SD Card Product Manual*, 11 2004. Version 2.2.

[San06] Sanyo. *AM-1417 Amorphous Solar Cell Datasheet*, 2006.

[SC01] A. Sinha and A. Chandrakasan. Dynamic power management in wireless sensor networks. *IEEE Design & Test of Computers*, 18(2):62–74, 2001.

[SCN13] K. Schneider, J. Conroy, and W. Nothwang. Computing optic flow with ardueye vision sensor. Technical report, DTIC Document, 2013.

[Sem16] Semtech Corporation. *SX1276MB1xAS - 137 MHz to 1020 MHz Low Power Long Range Transceiver Data*, 8 2016. Rev. 5.

[Sen18] Sensirion. *Datasheet SHT3x-DIS*, 2018. Version 5.

[SGL+17] L. Sigrist, A. Gomez, R. Lim, S. Lippuner, M. Leubin, and L. Thiele. Measurement and validation of energy harvesting iot devices. In *Proc. DATE Conf.*, pages 1159–1164. IEEE, 2017.

[Sil13] Silicon Labs. *Si1145 Proximity/UV/Ambient Light Sensor IC with I2C Interface Datasheet*, 2013. Rev. 1.1 12/13.

[SK11] S. Sudevalayam and P. Kulkarni. Energy harvesting sensor nodes: Survey and implications. *IEEE Communications Surveys Tutorials*, 13(3):443–461, Third 2011.

[SPC12] D. Spenza, C. Petrioli, and A. Cammarano. Pro-energy: A novel energy prediction model for solar and wind energy-harvesting wireless sensor networks. In *2012 IEEE 9th International Conference on Mobile Ad-Hoc and Sensor Systems (MASS 2012)*, pages 75–83. IEEE, 2012.

[SPM15] P. Spies, M. Pollak, and L. Mateu. *Handbook of energy harvesting power supplies and applications*. CRC Press, 2015.

[ST 17] ST Microelectronics. *EnFilm - rechargeable solid state lithium thin film battery*, 9 2017. Rev 2.

[STN+14] N. Sakimura, Y. Tsuji, R. Nebashi, H. Honjo, A. Morioka, et al. 10.5 A 90nm 20MHz fully nonvolatile microcontroller for standby-power-critical applications. In *Proc. ISSCC Conf.* IEEE, 2014.

[TBCS13] L. Torres, R. M. Brum, L. V. Cargnini, and G. Sassatelli. Trends on the application of

emerging nonvolatile memory to processors and programmable devices. In *Proc. ISCAS*, pages 101–104, May 2013.

[Tex15] Texas Instruments. *MSP430FR59xx Mixed-Signal Microcontrollers*, 3 2015. Rev. E.

[Tex16] Texas Instruments. *CC2650 SimpleLink™ Multistandard Wireless MCU*, 7 2016.

[Tex17] Texas Instruments. *MSP432P401R Mixed-Signal Microcontrollers*, 9 2017.

[Tre18] A. Tretter. *On Efficient Data Exchange in Multicore Architectures*. PhD thesis, ETH Zurich, 2018.

[TSM+17] M. Thielen, L. Sigrist, M. Magno, C. Hierold, and L. Benini. Human body heat for powering wearable devices: From thermal energy to application. *Energy Conversion and Management*, 131:44–54, Jan. 2017.

[UHM11] D. Uckelmann, M. Harrison, and F. Michahelles. An architectural approach towards the future internet of things. In *Architecting the internet of things*, pages 1–24. Springer, 2011.

[VBM18] T. D. Verykios, D. Balsamo, and G. V. Merrett. Selective policies for efficient state retention in transiently-powered embedded systems: Exploiting properties of NVM technologies, jul 2018.

[VdBVVM+06] P. Van den Bossche, F. Vergels, J. Van Mierlo, J. Matheys, and W. Van Autenboer. Subat: An assessment of sustainable battery technology. *Journal of power sources*, 162(2):913–919, 2006.

[WCK+14] C. Wang, N. Chang, Y. Kim, S. Park, Y. Liu, H. G. Lee, R. Luo, and H. Yang. Storage-less and converter-less maximum power point tracking of photovoltaic cells for a nonvolatile microprocessor. In *Proc. ASP-DAC Conf.*, 2014.

[WHSC01] A. Wang, W. B. Heinzelman, A. Sinha, and A. P. Chandrakasan. Energy-scalable protocols for battery-operated microsensor networks. *Journal of VLSI signal processing systems for signal, image and video technology*, 29(3):223–237, 2001.

[WLL+12] Y. Wang, Y. Liu, S. Li, D. Zhang, B. Zhao, M.-F. Chiang, Y. Yan, B. Sai, and H. Yang. A 3us wake-up time nonvolatile processor based on ferroelectric flip-flops. In *Proc. ESSCIRC Conf.* IEEE, Sept. 2012.

[WLW+15] Y. Wang, Y. Liu, C. Wang, Z. Li, X. Sheng, H. Lee, N. Chang, and H. Yang. Storage-less and converter-less photovoltaic energy harvesting with maximum power point tracking for internet of things. *IEEE Trans. CAD*, PP(99), 2015.

[WMM+13] A. S. Weddell, M. Magno, G. V. Merrett, D. Brunelli, B. M. Al-Hashimi, and L. Benini. A survey of multi-source energy harvesting systems. In *Proc. DATE Conf.*, March 2013.

[WSV+08] A. D. Wood, J. A. Stankovic, G. Virone, L. Selavo, Z. He, Q. Cao, T. Doan, Y. Wu, L. Fang, and R. Stoleru. Context-aware wireless sensor networks for assisted living and residential monitoring. *IEEE network*, 22(4), 2008.

[Xie16] Y. Xie. *Emerging Memory Technologies. Design, architecture, and applications.* Springer, 2016.

[Yak11] A. Yakovlev. Energy-modulated computing. In *Proc. DATE Conf.* IEEE, 2011.

[ZGL13] P. Zhang, D. Ganesan, and B. Lu. QuarkOS: Pushing the operating limits of micro-powered sensors. *Proc. 14th USENIX Conf. Hot Top. Oper. Syst.*, 2013.

[Zhe14] Zhengzhou Winsen Electronics Technology Co., Ltd. *MQ-6 Flammable Gas Sensor Manual*, 5 2014. Version 1.3.

List of Publications

The following list includes publications that form the basis of this thesis. The corresponding chapters are indicated in parentheses.

Gomez A, Sigrist L, Magno M, Benini L, Thiele L. **Dynamic Energy Burst Scaling for Transiently Powered Systems.** *In Proceedings of Conference on Design, Automation & Test in Europe (DATE).* EDA Consortium. 2016. (Chapter 2)

Gomez A, Sigrist L, Schalch T, Benini L, Thiele L. **Wearable, Energy-Opportunistic Vision Sensing for Walking Speed Estimation.** *In Proceedings of Sensors Applications Symposium (SAS).* IEEE. 2017. (Chapter 2)

Gomez A, Benini L, Thiele L. **Designing the Batteryless IoT.** *Design, Automation & Test in Europe (DATE) PhD Forum.* EDA Consortium 2017. (Chapter 2)

Gomez A, Sigrist L, Schalch T, Benini L, Thiele L. **Efficient, Long-Term Logging of Rich Data Sensors using Transient Sensor Nodes.** *Transactions on Embedded Computing Systems.* ACM. 2017. (Chapter 3)

Gomez A, Benini L, Thiele L. **Designing Reliable Transient Applications.** *In Proceedings of the IDEA League Doctoral School on Transiently Powered Computing.* 2017. (Chapter 3)

Gomez A, Conti F, Benini L. **Thermal Image-Based CNN's for Ultra-Low Power People Recognition.** *In Proceedings of LP-EMS Workshop.* ACM 2018. (Chapter 4)

Gomez A, Tretter A, Hager P, Sanmugarajah P, Benini L, Thiele L. **Optimizing the Energy Cost of Transient Applications through Partitioning.** *To be submitted.* 2018. (Chapter 4)

The following list includes publications that were written during the PhD studies, yet are not part of this thesis.

Anagnostou P, Gomez A, Hager PA, Fatemi H , Pineda de Gyvez J, Thiele L, Benini L **Torpor: A Power-Aware HW Scheduler for Energy Harvesting IoT SoCs.** *In Proceedings of Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS).* IEEE. 2018.

Palossi D, Gomez A, Draskovic S, Benini L, Thiele L. **Extending the Lifetime of Nano-Blimps via Dynamic Motor Control.** *Journal of Signal Processing Systems.* Springer. 2018.

Sigrist L, Gomez A, Lim R, Lippuner S, Leubin M, Thiele L. **Measurement and validation of Energy Harvesting IoT Devices.** *In Proceedings of Conference on Design, Automation & Test in Europe (DATE).* EDA Consortium. 2017.

Palossi D, Gomez A, Draskovic S, Keller K, Benini L, Thiele L. **Self-sustainability in Nano Unmanned Aerial Vehicles: A Blimp Case Study.** *In Proceedings of Computing Frontiers Conference.* ACM. 2017.

Sigrist L, Gomez A, Lim R, Lippuner S, Leubin M, Thiele L. **RocketLogger: Mobile Power Logger for Prototyping IoT Devices: Demo Abstract.** *In Embedded Network Sensor Systems (SenSys) Conference.* ACM. 2017.

Gomez A, Bartolini A, Rossi D, Kara BC, Fatemi H, de Gyvez JP, Benini L. **Increasing the Energy Efficiency of Microcontroller Platforms with Low-Design Margin Co-Processors.** *Microprocessors and Microsystems.* Elsevier. 2016.

Gomez A, Magno M, Lagadec MF, Benini L. **Precise, Energy-Efficient Data Acquisition Architecture for Monitoring Radioactivity using Self-Sustainable Wireless Sensor Nodes.** *Sensors Journal.* IEEE. 2016.

Magno M, Brunelli D, Sigrist L, Andri R, Cavigelli L, Gomez A, Benini L. **InfiniTime: Multi-sensor wearable bracelet with human body harvesting.** *In Proceedings of Future Access Enablers of Ubiquitous and Intelligent Infrastructures*

Conference. Springer. 2016.

Magno M, Polonelli T, Casamassima F, Gomez A, Farella E, Benini L. **Energy-Efficient Context Aware Power Management with Asynchronous Protocol for Body Sensor Network.** *Mobile Networks and Applications.* Springer. 2016.

Sigrist L, Giannopoulou G, Huang P, Gomez A, Thiele L. **Mixed-criticality Runtime Mechanisms and Evaluation on Multicores.** *Sustainable Computing: Informatics and Systems.* Elsevier. 2016.

Gomez A, Magno M, Wen X, Benini L. **Extending Body Sensor Nodes' Lifetime Using a Wearable Wake-up Radio.** *In Proceedings of Future Access Enablers of Ubiquitous and Intelligent Infrastructures Conference.* Springer. 2015.

Gomez A, Lagadec MF, Magno M, Benini L. **Self-powered Wireless Sensor Nodes for Monitoring Radioactivity in Contaminated Areas using Unmanned Aerial Vehicles.** *In Proceedings of Sensors Applications Symposium (SAS).* IEEE. 2015.

Gomez A, Pinto C, Bartolini A, Rossi D, Benini L, Fatemi H, de Gyvez JP. **Reducing Energy Consumption in Microcontroller-Based Platforms with Low Design Margin Co-Processors.** *In Proceedings of Conference on Design, Automation & Test in Europe (DATE).* EDA Consortium. 2015.

Gomez A, Schor L, Kumar P, Thiele L. **SF3P: A Framework to Explore and Prototype Hierarchical Compositions of Real-Time Schedulers.** *In Proceedings of Rapid System Prototyping (RSP) Symposium.* IEEE. 2014.