Scheduling Dependent Tasks for Energy-Neutral Operation of Sensor Nodes

Vom Promotionsausschuss der
Technischen Universität Hamburg

zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

genehmigte Dissertation

von

Lars Hanschke

aus

Hamburg, Deutschland

2022

Date of Oral Examination	February 8th, 2022
Chair of Examination Board	Prof. Dr.-Ing. Görschwin Fey Institute of Embedded Systems, Hamburg University of Technology
First Examiner	Prof. Dr.-Ing. Bernd-Christian Renner Institute for Autonomous Cyber-Physical Systems, Hamburg University of Technology
Second Examiner	Prof. Dr. rer. nat. Volker Turau Institute of Telematics, Hamburg University of Technology

Acknowledgment

In the process of working towards this dissertation, a long list of supportive people evolved. To these people, I would like to express my deep gratefulness.

In first place, I want to thank Prof. Christian Renner for lively technical and non-technical discussions, continuous input and feedback as well as for sharpening my eye for the small details required by scientific work. I would also like to thank my second reviewer, Prof. Volker Turau for his valuable comments on this dissertation.

For all the support and encouragement by my colleagues, I would like to express my deep thankfullness. Explicitly, I would like to thank Jan Heitmann for teaching me circuit board design as well as for sharing my passion for well-prepared coffee. Additionally, I would like to express my gratefulness to Fabian Steinmetz, Peter Oppermann, and Christian Busse for their encouragement and support during my work at the institute.

For all the inspiration I obtained by working with them, I would like to thank my students. Especially, I would like to thank Jannick Brockmann and his team members for the work on the solar-harvesting testbed. Furthermore, my gratefulness is addressed to Frank Zimdars for sharing countless hours with my harvesting prototypes in the laboratory.

To balance my work on this dissertation with joy of life, I could always rely on my friends and family. I would like to thank Johannes Timmermann and Fritz Miekautsch for exchange and discussions on the daily business as research assistant. For keeping me motivated, focused and relaxed when needed, I would like to thank Robin Hannemann, Steffen Just, and Kolin Schunck. Furthermore, I would like to express thankfulness to my parents for supporting me in all my decisions.

In the end, I would like to thank the most important people who supported me. Johan and Emil, thank you for teaching me the most essential aspects in my life and for cheering me up countless times. Finally, the last words are reserved for Lisa: Thank you, for being the most valuable critic and unconditional supporter in the best way I could ever wish for.

Abstract

Wireless sensor networks became ubiquitous in modern life since they collect fine-grained data for a plethora of applications, e.g., environmental monitoring and infrastructure health assessment. Moreover, sensor networks are a central part of production processes—e.g., to monitor pollution—and are, hence, a driver of the fourth industrial revolution. To supply the growing number of sensor nodes with power, typically wires or disposable batteries are used. While the former harm installation and arrangement flexibility, the latter impose high maintenance effort and increase the ecological impact. Energy harvesting—recharging the sensor node with ambient energy—gains considerable research interest since it promises to cure the burden of frequent battery exchange. However, modern applications pose new demands on energy-harvesting sensor nodes which are yet not covered by current literature. Applications, such as fine dust monitoring at outside production plants, demand timely sampling of dependent sensor values which is yet not enforced by scheduling algorithms. Additionally, applications need to describe times of interest via quality-of-service levels, e.g., health assessment of bridges at peak time during rush hour. This dissertation, hence, addresses these main challenges and presents tools and algorithms to advance knowledge on scheduling for energy-harvesting sensor nodes.

Time dependencies between tasks of program structures are already described but not enforced by scheduling algorithms of energy-harvesting sensor nodes. The key requirement remains to ensure perpetual operation to ensure continuous data sampling. If perpetual operation can only be ensured via long sleep times to recharge the storage, timeliness of sensor values is risked—if only timeliness is targeted, energy may be spent greedily, and the storage may deplete. A new algorithm is developed that balances between these two poles. First, a mathematical model is derived that integrates task-based programs and distributes the tasks in time without violation of time constraints and without excess of a targeted energy consumption. Second, a heuristic is developed that parameterizes scheduling of energy consumption to target different application needs. The simplified definition of program structures and their energy-aware scheduling broadens the range of potential program developers and, hence, facilitates development of new applications.

Quality-of-service is a central part of new sensor network applications with the goal to increase utility of data sampling. In this dissertation, a new energy management algorithm—`EmRep`—is developed that couples energy-aware scheduling with a description of important monitoring times. `EmRep` uses utility profiles to spend energy primarily at times which benefit the application. Additionally, the energy efficiency is increased since surplus energy in times of high intake is exploited. As consequence, `EmRep` improves utility across a variety of harvest estimation techniques, storage size and utility profiles.

The developed methods and algorithms are compared against state-of-the-art approaches via simulation studies, testbed investigations, and real-world experiments. The algorithms, the simulation framework and the instructions of the testbed are published as open-source software and, hence, simplify program development and performance assessment. The algorithms in this dissertation outreach existing literature for energy-harvesting systems, improve energy-efficient operation of sensor networks, and will, as such, enrich application variety.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The world of Wireless Sensor Networks has grown from early deployments to an indispensable part of today's Internet of Things. Sensor networks constantly monitor our environment, gather information, e.g., about pollution, or help with heating buildings more efficiently. Wireless Sensor Networks not only help governments to increase quality of life for inhabitants but also allow companies to lower their environmental impact and hence contribute to a greener Planet. As new areas of deployment evolve, the number of connected devices increases year after year.

Supplying all these sensor nodes with power, however, still entails the choice between wired supplies or batteries. While the former causes high installation cost and lowers installation flexibility, the latter imposes high maintenance cost due to change of batteries and raises the environmental impact due to toxic materials used in disposable batteries.

Energy harvesting—generating electrical energy from ambient resources for sustained operation—promises an unlimited lifetime of sensor nodes without the need for human intervention. Hence, energy harvesting is a vital competitor to power Wireless Sensor Networks; especially solar energy harvesting draws considerable research interest. The converted energy is either used to supply the sensor node directly or to charge rechargeable batteries and (super-)capacitors. While the former ties node operation to energy availability, the latter entails careful use of energy to prevent storage depletion. The powerful vision of endless lifetime moves the implications of energy harvesting into the focus of research but imposes new challenges.

Moreover, the duties of Wireless Sensor Networks shift from solely transmitting raw data values to transmitting *information*. However, to provide accurate information, sensors need to be aware of the *sense* of data values, e.g., dependencies between subsequently collected values. A decision on the air quality level, for example, requires raw data values of multiple sensors—as well as the present authority regulations. Up until now, this contradicts with energy management techniques: sensor values split apart in time, imposed by long sleep times, may eliminate information. Timeliness for sensor values has been investigated for

real-time systems and for intermittently-powered—loosing supply voltage multiple times per minute—devices; but literature lacks support for small-sized energy-harvesting sensor nodes. This dissertation closes this gap and introduces a lightweight scheduling algorithm that enforces timeliness for sensor values in harvesting systems. The scheduler allows program definition by graphs and annotations for timeliness which is easy to use and to operate. Hence, this dissertation contributes to lower the entry barrier for application developers for energy-harvesting systems.

The availability of energy is highly varying, e.g., solar energy is influenced by many factors, such as clouds, local shadowing, dust, and seasonal changes. Hence, sensor nodes estimate the future energy intake and careful energy management is extremely important. The key role of energy management—adjusting the node activity—is to prevent energy storage depletion and to maximize the Quality-of-Service for the served application. Up until now, energy management misses a definition of Quality-of-Service for small-sized energy harvesters. Energy is spent when it is available, without increasing the value for the application—this wastes energy and lowers efficiency. This gap is closed by this dissertation which supports applications in defining times of interest and usefulness of information. The developed energy-aware activity management enables applications to define a certain activity level, e.g., gathering a flexible number of sensor values per time interval. This dissertation hence advances the current state of the art for energy management under Quality-of-Service with depletion safe harvesting sensor nodes.

1.1 Dissertation Goal

This dissertation aims at solving the identified open challenges for sensor nodes equipped with energy harvesters. Open challenges embrace depletion safety in-hand with timely sensor measurements and user-defined utility. In this dissertation, developers that require their applications to fulfill above aspects, find new tools, techniques and algorithms. More specifically, the contributions are as follows.

- Based on the application demands, the open challenges are deduced; depletion safety, timely-gathered sensor values and user-defined utility.
- Previous work on energy management for sensor nodes with energy harvesters is examined and evaluated; specifically targeted at the identified challenges.
- The open challenges—depletion safety, timely sensor values, and user-defined utility—are analyzed, and necessary actions are deduced.
- For reliable testing of the sensor node, measurements are undertaken, a simulation toolkit is introduced and a testbed is developed.

- A new approach for describing time relations in programs of energy harvesting systems is presented and mathematically described.
- A new heuristic to schedule task-based program structures for sensor nodes with energy harvesters is developed and evaluated in detailed simulations and in a real-world experiment.
- A new energy management algorithm for energy-neutral sensor nodes that integrates user-defined utility is detailed and analyzed.
- In the light of this, future research directions are identified and outlined.

1.2 Dissertation Structure

The remainder of this dissertation is organized as follows. Chapter 2 details relevant literature on Wireless Sensor Networks in general, operation on limited energy, challenges imposed by usage of solar energy harvesting as well as the latest trends on energy management and operation for Wireless Sensor Networks. Chapter 3 analyzes the main challenges for future energy management in solar energy harvesting systems and showcases gaps in existing literature. The challenges, namely providing depletion safety, supporting timely scheduling of task-based program structures and incorporating user-defined utility into energy management for solar harvesting systems, build the basis for the core contributions of this dissertation. In Chapter 4 the tools for reliable testing are introduced; namely a solar harvester with supercapacitor, the closely connected simulation framework, and the controlled lighting testbed for harvesting condition emulation. Chapter 5 details the first core contribution: the mathematical foundations for describing arbitrary program structures as graphs and for solving the entailing scheduling problem with a standard mathematical solver is developed. Chapter 6 presents a heuristic algorithm to solve the problem for task-based program scheduling in energy-harvesting systems under dynamically varying harvest conditions. Chapter 7 presents `EmRep`, an energy management algorithm relying on state-of-charge prediction. It enables usage of user-defined utility profiles that enhance the value for applications relying on solar energy harvesting sensor nodes. The dissertation finishes with Chapter 8 by providing a summary and conclusion on solved challenges, as well as implications for future research directions.

Chapter 2

State of the Art

This chapter details and examines existing literature. First, Wireless Sensor Networks are introduced and open challenges are highlighted, c.f. Section 2.1. Section 2.2 examines the opportunities and challenges that arise from working with the scarce resource energy in Wireless Sensor Networks. The entailing topic of energy management is discussed in Section 2.3, especially targeted at deficiencies of previously published literature. In Section 2.4, common barriers and pitfalls while operating with solar energy harvesting are detailed. The chapter ends with Section 2.5 in which future research areas are highlighted and major challenges are identified.

2.1 Wireless Sensor Networks

The term Wireless Sensor Network (WSN) became ubiquitous in the past years; in industry, government, and also private life. WSNs consist of multiple spatially distributed autonomous devices, called *sensor nodes*. They are used to monitor the environment, gather new information and even influence the environment [AMAT+18]. WSNs gained popularity in applications such as environmental monitoring [WCPnC18], animal tracking [SGJ+20], disaster management [CLW+13], and health monitoring of civil infrastructure [GRHJ+18].

According to [SBH16] sensor nodes mostly consist of the main components: a wireless communication module, a microcontroller, data storage, a power supply and a sensor or an actuator. Once deployed, sensor nodes collect information about their surrounding, e.g., temperature and moisture level, and transmit the data towards a central instance, typically a gateway connected to a remote server or the Cloud. Deployments regularly contain hundreds or thousands of nodes, hence sensor nodes are low-cost devices that only have limited computation and storage resources. WSNs work autonomously without preinstalled infrastructure and ideally arrange themselves after deployment.

In industrial context, WSNs are a central driver of the *fourth industrial revolution* (“Industry 4.0“). Here, WSNs evolve into Cyber-physical Systems (CPS), where interaction

with the physical world is enhanced. In CPS, sensors and actuators are integrated into a larger system, fulfill only a small designated task and are embedded into large infrastructure, e.g., a production plant surveillance and control system [SWYS11]. If sensor networks are additionally provided with Internet access, they are typically referred to as part of the Internet of Things (IoT). Smart buildings, smart roads and smart production areas are unimaginable without the IoT [AIM10].

Applications

Local authorities such as the *Hamburg Port Authority (HPA)* aim at redesigning and preparing the harbor infrastructure for the future. This involves multiple projects, e.g., smart roads, smart maintenance and parking space management; as detailed in [Aut14]. Moreover, local governments such as the "Behörde für Wirtschaft, Verkehr und Innovation" in Hamburg propose concepts [fVudIB] for innovations for mobility in the city of Tomorrow. According to [YKK+20], a key requirement for autonomous driving is a robust infrastructure that safely aids cars and provides reliable data about surroundings. Equipped with sensor technology, these smart roads are an essential part of future transportation [TSCM20].

According to the World Health Organization (WHO) [Org13], air pollution is one of the major risks for human health in cities. As a consequence, all members of the European Union (EU) signed programs, e.g., for Germany [FMftENSB19], that require local governments to track and take countermeasures against pollution with ozone [MSHT16], fine particle matter [BEMRB13], and others [MZT18]. Platforms such as OpenSense [LFS+12] (deployed in Zurich) can help the governments by providing high spatial resolution data at a fraction of the cost for fixed deployments.

The applications—in which WSNs are envisioned—change: from animal-tracking to an essential part of future infrastructure and transportation. This raises several new questions and topics; such as Quality-of-Service (QoS), security and privacy. Furthermore, as sensors become ubiquitous, and integrated in structures, the problem of powering these nodes worsens. How should batteries be changed in a sensor device which is embedded at the surface of a smart road? Who can change batteries from hundreds of sensors deployed to assess the health of a high bridge?

According to [SBH16] and [AMAT+18] one of the main challenges that WSNs are projected to face in the future is energy-efficient computing. This requires a deep understanding how energy as primary resource influences the operation of WSNs.

	TelosB Mote	Adafruit Feather	Espressif ESP8266
Source	[Mot04]	[STM19]	[Esp]
MCU	TI MSP430	Atmel ARM Cortex M0+	RISC L106
clock frequency	8 MHz	48 MHz	80 MHz
RAM, Flash	10 kB, 48 kB	32 kB, 256 kB	80 kB, 4 MB
radio	TI CC2420	Semtech SX127x	ESP8266EX Wifi
radio technology	IEEE 802.15.4	LoRa	IEE 802.11n
light sleep (µW)	60	14	2700
deep sleep (µW)	3	9	60
active (mW)	5	11	45
radio listening (mW)	56	40	168
radio transmission (mW)	52	96	360

Table 2.1: Comparison of common sensor nodes with different radio technologies and their power consumption.

2.2 Powering the Internet of Things

The number of connected IoT devices is projected to outreach 27 billion by 2026[1]. At this level, providing electrical power imposes huge efforts and cost. While communication via wireless transmission has been vastly adapted for IoT deployments, providing electrical energy is still an open issue [SBH16]. Even if feasible, wired power supplies for millions of devices impose huge deployment costs and decrease placement flexibility. Additionally, in remote deployments, e.g., the Matterhorn [WBF+19] and volcanoes [WALR+06], wires are not an option; batteries are used instead. This, however, makes the energy resource finite due to the limits of the battery capacity.

As detailed in Table 2.1, the power consumption between microcontrollers and their peripherals varies vastly. Microcontroller Units (MCUs) of connected devices are offered in a wide range of computational power and power consumption. Typical devices draw between 5 mW (MSP430) [Tex17] and 45 mW (Espressif ESP8266). With typical non-rechargeable AA batteries [Dur14], this limits the operation time to 250 d (MSP430) and 3 d (Espressif ESP8266). This high-frequent battery exchange is economically not feasible since human-intervention is typically the most costly part of maintenance. Additionally, battery exchange translates to interrupting the regular operation: productions flows are suspended and environmental conditions may be influenced. Furthermore, non-rechargeable batteries contain potentially hazardous materials such as heavy metals and their recycling process is complex [BFB+12]. Thus, various methods exist to tackle energy as scarce resource in WSNs.

[1] https://www.ericsson.com/en/mobility-report/mobility-visualizer

2.2.1 Operation with Limited Energy

As also detailed in [KHZS07], the electrical energy E is the power P consumed over a time interval $[0, \mathrm{T}]$ and is described by the integral:

$$E = \int_0^{\mathrm{T}} P(t)\,\mathrm{d}t. \tag{2.1}$$

To prolong the operation time of sensor nodes, the consumed energy has to be as little as possible which directly translates to reducing the consumed power or reducing the time when power is spent. The most power-hungry components of sensor nodes are typically the radio module, for communication with the rest of the network, and the MCU for computations. Additionally, sensor nodes are equipped with the sensors themselves, e.g., temperature sensors, humidity sensors or GPS. To operate on limited energy, precise knowledge of the energy consumption of every component is required. Next, a brief overview of influencing factors of energy consumption is given.

As detailed in Table 2.1, modern MCUs offer multiple operation states; hence the key objective is to operate as little time as possible in energy-costly active states. The ratio between active time and overall time is commonly known as duty cycle DC:

$$\mathrm{DC} = \frac{T_{\text{active}}}{T_{\text{active}} + T_{\text{sleep}}}. \tag{2.2}$$

In the original definition of the duty cycle, T_{active} includes all operational states except sleeping. Other definitions, such as the radio duty cycle where T_{active} only includes operational time of the radio module, are possible as well.

The power demand of radio modules depends on the used communication standard, transmission power but also used hardware such as signal amplifiers and antennas. Over the past, many communication standards for WSNs evolved [ESC16]:

- IEEE 802.15.4, adopted by popular IoT protocols such as ZigBee;
- LoRa, targeted at low data rate and long range communication;
- WiFi, offering seamless integration in existing infrastructure;
- or 5G, competing for fusion of mobile communication and WSNs.

While standards such as IEEE 802.15.4 leave choices on transmission power or modulation to the operator, others require strict definitions. For example, WiFi certified devices define modulation and coding schemes with exact definition of data rate, modulation technique and transmission power. As detailed in Table 2.1, the power consumption of radio modules can differ drastically between used techniques, e.g., the WiFi module consumes seven times

more power during transmission than a IEEE 802.15.4 module. Hence, using low power radio communication is beneficial. However, lower energy consumption can also be achieved if the time with active radio modules, or power-hungry components in general is reduced, c.f. Eq. (2.1).

For radio communication, this is achieved by optimizing the Medium Access Control (MAC) protocol. Literature has shown that prior methods, e.g., CSMA/CA as used in WiFi, require long time with active radio modules. Hence, various protocols are introduced to target energy-efficiency of communication, e.g., S-MAC [YHE02] and B-MAC [PHC04]. Furthermore, initiated communication, known as wake-up radios, evolve that mitigate the need for idle-listening completely [BXNX19, PSLRC18].

Still, communication protocols and duty-cycling only *prolong* the lifetime. To reduce the burden of changing batteries completely, there is no way around harvesting energy from the environment.

2.2.2 Energy Harvesting

Since changing batteries imposes huge maintenance cost and environmental impact, a solution for *recharging* the batteries is needed. Various sources and opportunities of harvesting ambient energy are discovered: solar energy, mechanical energy, temperature variation energy, as well as hydro and wind energy; to only name a few. An overview of recently used harvesting techniques is given in Table 2.2.

Solar Harvesting

Outdoor solar harvesting offers the highest power density with up to 100 mW/cm^2. Small solar panels are used to harvest energy during daylight hours. Especially on bright sunny days, harvested energy often exceeds consumable energy by magnitudes. However, many factors influence the performance of solar energy harvesting. The irradiation level of sunlight varies throughout the year [DHM75] and is influenced by daily weather. Furthermore, local shadowing due to trees or buildings, light reflection and diffusion impacts solar irradiation so that even neighboring locations do not experience the same irradiation level. Additionally, solar panel efficiency is harmed by accumulating dust, pollen, and snow. Regardless of the challenges, a plethora of prototypes for solar harvesting evolved, e.g., Heliomote [RKH$^+$05], Ambimax [PC06], Everlast [SC06], and Renner et. al. [RUTR14]. The specific challenges and existing solution approaches to outdoor solar energy harvesting are detailed in Section 2.4.

Solar panels can also be used inside, but generally offer lower energy intake; i.e., one magnitude lower [SKS20]. At indoor locations, energy can be harvested from artificial lightning sources—ceiling or desk lamps—and reflected natural light from the sun. Recent observations [SGT19] show that indoor harvesting offers good opportunities as well, especially

harvest method	principle	power density	characteristics	source
solar energy (outdoor)	photovoltaic	15 – 100 mW/cm^2	ambient, predictable, uncontrollable	Sun
solar energy (indoor)	photovoltaic	0.15 – 100 mW/cm^2	artificial, controllable	lights
mechanical energy	micro generator, piezoelectric	4 – 800 µW/cm^3	artificial	machines, cars, human motion
temperature variations	pyroelectric	10 µW/cm^3	ambient, uncontrollable	waste energy (heaters, vehicle exhausts)
radio freqency	electromagnetic induction	0.1 – 125 µW/cm^3	ambient	WiFi signals
hydro, wind energy	triboelectric generator	60 – 400 µW/cm^3	ambient, unpredictable, uncontrollable	wind, water flow

Table 2.2: Non-exhaustive comparison of energy harvesting techniques and example sources; power density varies by magnitudes; based on [SKS20].

if rooms with significant outdoor radiation are used. For installed sensors, indoor illumination levels offer good predictability, e.g., caused by switched-on lights in office hours, but wearable sensors suffer from unpredictable local shadowing, e.g., caused by surrounding people. Recently indoor solar harvesting gained more interest but needs more investigation, e.g., on the interplay between solar panel and artificial light sources [MBO20].

Vibration Harvesting

Applications in which kinetic energy can be used for electrical conversion are numerous: human motion, machine vibration, bridge vibration or even raindrops [WDMM14]. The principle remains roughly the same: kinetic energy induces motion on a proof mass (usually oscillation). This proof mass may be surrounded by coils such that an electrical field is induced or is used to deform a cantilever-shaped structure. The excitation of the proof mass influences the amount of harvested power; hence, using the harvester at resonant frequency is beneficial since the amplitude of oscillation increases in resonance. However, according to [WJ17], the most limiting factor is that harvested power drops significantly above and below the resonant frequency. Hence, the resonant frequency of the vibrating structure has to be determined very exactly in beforehand and the harvester has to be tuned. While this may be simple in some cases, e.g., machines with known rotational speed, it requires more effort in other cases, e.g., for bridges. As consequence, harvesting source and harvester are coupled very tightly which makes it hard to develop a one-fits-all-solution. The power density of up

to 800 µW/cm^3, according to [SKS20], allows to power sensor nodes, but falls behind solar harvesting. Additionally, the harvested power is only available at a low amplitude AC signal, which entails additional circuits for voltage rectification and up-conversion and thus increases harvester complexity.

Nevertheless, various projects evolve which use vibration energy for powering the IoT. In [GRHJ+18] a self-tuning vibration harvester is used to power MSP430-based sensor nodes. The harvester generates 1 mW in average on the Forth Road Bridge (a 2.5 km suspension bridge in Scotland) and allows health assessment of the bridge. In [KRCZ17] a harvester for wearables is presented. Mounted at the knee-joint, it generates energy from the walking human and powers a sensor node with IEEE 802.15.4 radio.

Other Sources

Additionally, also radio frequency waves, temperature variations and wind and hydro energy can be used to power sensor nodes. However, their availability is even more unsteady and the power density is lower. According to [AMAT+18], a main challenge is to harvest and store sufficient energy to enable networked wireless communication at defined service levels. An interesting prototype for usage of radio frequency harvesting is presented by the authors of [NPK+15]: the WISPcam. It is based on the WISP platform [SSP+06] which implements a passive Radio-Frequency Identification (RFID) tag and harvests energy from the communication with the RFID reader. The camera accumulates charge in capacitor, takes a snapshot and transmits the image back to the reader. On-demand data gathering and transmission is therefore not possible and prevents active networking.

Thermoelectric Generators (TEGs) can be used to transform temperature gradients—usually two adjacent materials—into electrical energy. An appealing platform is presented with DoubleDip [MCS12] which powers the sensor node by the temperature gradient between a waste water pipe and ambient air. Furthermore, the jump in temperature gradient when new water runs through the pipe is used to wake up the sensor node. Although being an active research topic, e.g., in [TSM+17] and [AA20], prototypes with wireless communication capabilities are yet to come.

Multi-source Harvesting

Since all sources of ambient energy vary, combining multiple sources into one harvester seems beneficial. Ideally, harvesting sources with non-overlapping high-intake phases promise increase of average consumption. Popular prototypes combine a highly-varying high-power source—e.g., solar radiation—with a constantly available but low-power source, e.g., a vibrating structure. The first popular approach is Ambimax [PC06] which is evaluated with a solar panel and a wind turbine, but also supports exchange of harvesting sources. Three

different sources of energy—solar, radio frequency, and vibration—are used in a prototype presented in [ATGC+15]. Although the claimed power output of 241 mW seems promising, the authors of [ATGC+15] admit that most power is generated by the large solar panel at ideal conditions. Only few prototypes evolve which is mainly caused by the additional complexity of harvesting circuitry including power point tracking of multiple sources. As also identified by the authors of [WMM+13] the promised benefits have to outweigh the loss of simplicity of single-source harvesting.

Conclusion

Opportunities for energy harvesting can be found in a plethora of applications, e.g., monitoring air quality for workers in outside production plants powered by solar panels. Also, commercially available products, e.g., the EnOcean push button [EnO20], show that energy harvesting leaves the niche of academic prototypes. A mature and well-investigated source is solar energy harvesting which is also thoroughly discussed in Section 2.4.

2.2.3 Energy Storage

The previous chapter shows that wireless sensor nodes need opportunities to recharge their energy storage to prolong lifetime. Although the majority of sensor nodes is equipped with non-rechargeable batteries, those batteries have no relevance in Energy-harvesting Wireless Sensor Networks (EH-WSNs) due to the lack of recharging capabilities [AMAT+18]. Choosing the right energy storage, however, is not trivial. Important aspects to consider are physical dimensions, capacity, charge and discharge characteristics as well as cycle life. A brief introduction of the essential aspects is given, followed by a characterization of the most commonly used storage types in EH-WSNs.

Criteria

Sensor nodes exist in a plethora of physical *dimensions* which are mainly dictated by the deserved purpose. Hence, the served purpose limits the physical dimension (volume) of the energy storage, e.g., flying foxes can only carry small storage to prevent disturbance of the animal [SGJ+20]. At constant physical dimensions, the capacity of a rechargeable battery—usually measured in Ah—is limited by the *energy density* of the used materials. Choosing materials with high energy density promises high capacity at low volume but also bears risks when not treated correctly, e.g., flammable lithium batteries [LLCA18]. Furthermore, charge and discharge characteristics have to be taken into account. The voltage level of rechargeable batteries often differs when discharged with high peak currents which makes modeling the exact behavior overly complicating [RFG13]. Furthermore, *charge and discharge* currents

	NiMH	LiIon	LiPo	EDLC
manufacturer	Panasonic	UltraLife	MikroElektronika	Samwha
model	BK-3HCC	UBBP01	SR674361P	DB5U207M3
source	[Pan20]	[Ult14]	[Mik20]	[Sam20]
nominal voltage (V)	1.2	3.7	3.7	2.7
operational range (V)	1.0...1.4	3.0...4.2	2.5...4.2	0...2.7
nominal capacity	2550 mAh	1800 mAh	2000 mAh	200 F
energy (calculated)	3.06 Wh	6.6 Wh	7.4 Wh	0.2 Wh
nominal energy density	295 Wh/L	317 Wh/L	n.a.	6.37 Wh/L
charge (recom.)	250 mA	900 mA	400 mA	(max.) 12.3 A
discharge (recom.)	500 mA	max. 1.8 A	max. 2 A	(max.) 96.4 A
self-discharge	2% per year	<10% per month	<10% per month	≈50% per week
recharge cycles	500	500	300	500 k
volume (cm^3)	8.3	21.2	19.2	35.3
weight (g)	30	46	45	38

Table 2.3: Energy storage used in harvesting systems; supercapacitors offer high power density and recharge cycles but suffer from high leakage.

typically differ by magnitudes, e.g., charging with a solar panel with up to 300 mW and discharging with a node at sleep mode with 10 µW. This behavior makes State-of-Charge (SoC) estimation regularly too complex to be executed efficiently on microcontrollers. Additionally, the cycle count is important to assess the lifetime of the sensor node. The cycle count—as commonly defined in literature—is the number of full recharge cycles, i.e., increasing the storage level from near zero to fully charged. With only a few hundreds of *charge cycles*, rechargeable batteries may die too early which reverses the benefits of energy harvesting to the opposite.

A comparison of commonly used energy storage technologies is depicted in Table 2.3. In the following a comparison between rechargeable batteries and supercapacitors is presented.

Rechargeable Batteries

Early prototypes for energy harvesting systems such as the Heliomote [RKH+05] are equipped with Nickel Metal Hydrydide (NiMH) batteries, typically in the widely available AA or AAA size formats. Compared to Lithium-ion (LiIon) batteries, NiMH batteries offer a comparable energy density but suffer from moderate memory effect. Commonly, NiMH batteries are used in series to increase the voltage. LiIon batteries are very popular [LLCA18], since they offer a high energy and power density. Furthermore, they do not suffer from the memory effect and improve the lifetime due to increased number of recharge cycles. In recent deployments, e.g.,

the Signpost platform [AGJ+18], even large batteries with up to 576 Wh are used. Lithium-ion Polymer (LiPo) batteries improve the lithium-based batteries by replacing the liquid electrolyte with a polymer which allows arbitrary shapes of batteries. Both lithium-based batteries share the burden that this technology is extremely sensitive: undercharging can possibly make the cell useless; overcharging can lead to expansion; puncturing or short-circuiting can lead to fires. However, lithium-based batteries are extremely popular nowadays—especially due to their low price.[2]

Supercapacitors

Supercapacitors, or also called ultracapacitors, are the second group of energy storage devices used in EH-WSNs. The most common type used in EH-WSNs is an Electric Double-Layer Capacitor (EDLC) which make use of two charged layers forming at the electrodes. Compared to rechargeable batteries, they offer less energy density but much higher power density. Furthermore, they operate over a wider temperature range and survive larger number of recharge cycle counts, c.f. Table 2.3. In contrast to rechargeable batteries, supercapacitors experience high leakage currents which manifests in high self-discharge [dFCMV20]. The remaining charge of supercapacitors can be determined by voltage measurements which simplifies SoC assessment. Simulation of discharge and charge behavior, however, is studied thoroughly in [WMKAH11] and [RT10] to obtain accurate estimations on future charge level.

Supercapacitors are suitable to replace rechargeable batteries in a variety of deployments. The Everlast platform [SC06] operates a low power MCU with radio module powered by a 100 F supercapacitor. Even power-hungry WiFi sensor nodes can be equipped with large supercapacitors and operated perpetually on solar energy [HHR16].

The drawback of using large capacitors is that they recharge slowly. If supercapacitors are used to power the MCU directly, a distinct turn-on voltage has to be reached before the system is usable, e.g., the common MSP430 MCU [Tex17] needs at least 1.8 V to start. To mitigate this problem, boost converters can be used to stabilize the supply voltage; this increases the usable voltage range and hence more energy can be extracted. Still, the large capacitance entails the difficulty of slow recharge—if the turn-on voltage should be reached quickly, small capacitors should be favored [HSS15].

Consequently, the use of capacitor banks which incorporate multiple, but smaller capacitors is advocated [HSS15]. Further approaches even tie single capacitors to peripherals [HS17a] to decouple platform recharging.

[2] Bloomberg Report, accessed February 10th, 2021.

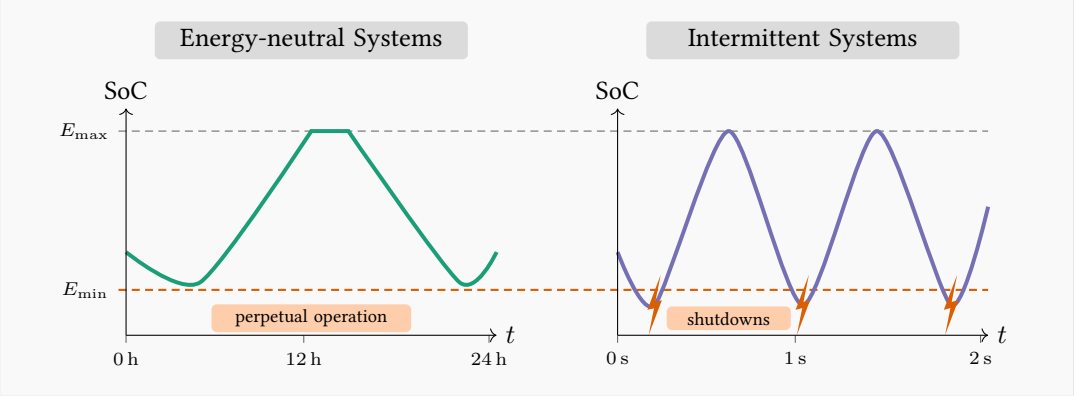


Figure 2.1: Example of state-of-charge course of energy-neutral and intermittently-powered systems. Please note the different timescale. Energy-neutral systems adapt their activity when energy is scarce to ensure perpetual operation. Intermittently-powered system, however, operate with high consumption until the energy is fully depleted and the sensor shuts down; when the energy level is sufficient again, operation returns to normal and the process repeats.

2.3 Energy Management in EH-WSNs

As highlighted before, operation on limited energy entails thoughtful use of available energy. Here, regulating the activity of the sensor node is the prior objective—without neglecting the application goal. First, this section motivates relevant parameters for energy management of Energy-neutral Operation systems, c.f. Section 2.3.1. Second, the upcoming generation of Intermittent Computing devices is presented and examined. An example for Energy-neutral Operation systems compared to Intermittent Computing systems is depicted in Fig. 2.1. The section ends with a comparison between the two classes and their ability to fulfill the presented criteria.

2.3.1 Energy-neutral Operation

The term Energy-neutral Operation (ENO) is defined and examined in [KHZS07] and describes the relation between the system's power consumption $P_\mathrm{n}(t)$ and the harvest power $P_\mathrm{h}(t)$. To reach energy neutrality with ideal and infinite energy storage, the following condition has to be ensured:

$$\int_0^T P_\mathrm{n}(t)\,\mathrm{d}t \leq \int_0^T P_\mathrm{h}(t)\,\mathrm{d}t + C_0 \quad \forall \quad T \in [0,\infty)\,. \tag{2.3}$$

Publication

Parts of Section 2.3.1 are published in [HR20].

Here, C_0 denotes the initial capacity of the ideal energy storage and T is an arbitrary non-negative point in time. Energy management for ENO systems, hence, optimizes $P_\mathrm{n}(t)$ such that Eq. (2.3) holds. Although this sounds intuitive, fulfilling Eq. (2.3) in practice is non-trivial due to the following factors. *Consumption uncertainty*: $P_\mathrm{n}(t)$ has to be obtained by in-lab measurements. However, it is also influenced by conversion efficiencies [ABB+19] and varies with used peripherals, e.g., radio modules which additionally have external dependencies such as MAC protocols. *Harvest uncertainty*: as outlined in Section 2.4, the harvesting source is not predictable; however a pattern can be learned. *Storage uncertainty*: energy storage typically has charging and discharging characteristics, c.f. Section 2.2.3 and even obtaining the actual energy capacity—compared to the nominal capacity—is complex [RT10]. Hence, the in-lab assumptions have to be adjusted continuously during the deployment to obtain an accurate estimation on the energy-neutrality. In the light of this and the application requirements presented in Section 2.1, the following criteria are used to examine the abilities of energy management algorithms for ENO systems:

- **Horizon.** The timespan for which the schedule is computed in advance.
- **Resolution.** To limit the computation effort, schedules are created and calculated at discrete points in time. The horizon is thus split apart, e.g., hours for diurnal cycles.
- **Harvest Estimation.** Estimating the amount of harvested energy is crucial to plan energy expenses; best to be incorporated by design.
- **User-defined Utility.** Enables the application to express the value of information in a certain time-interval.
- **Task-based Programs.** Programs are seen as sequence of individual tasks; instead of purely duty-cycle-centric approaches.
- **Online only.** Harvesting patterns are extremely location dependent; hence adaption during runtime is crucial.
- **Real-World experiments.** Simulations without connection to real hardware often neglect factors such as storage leakage, conversion efficiencies; furthermore energy consumption related to the radio module is hard to model due to interference.

In the past, two different classes of energy management techniques for ENO systems evolved. *Planning* approaches assume that the activity of the sensor node can be planned in advance. Usually, requirements of the application, e.g., a certain sampling rate, are given and included for energy management. The general idea is to match or outreach the application requirement while simultaneously ensuring energy-neutrality. In contrast to

planning approaches, *real-time* approaches assume the node activity to appear spontaneously in small parts (i.e., tasks), often described by statistic models of arrival rates. Tasks have to be scheduled as fast as possible to prevent to miss the deadline but without neglecting energy consumption.

Table 2.4 lists relevant literature in both classes and classifies them based on the presented criteria. A detailed description of the relevant literature is given in the following; starting with *planning* approaches and finishing with *real-time* approaches.

Planning

One of the first approaches presented in [KHZS07] solely relies on adjusting the duty cycle based on future energy income to ensure ENO. The approach aims to maximize the duty cycle but neglects the variety of different power consumption states of a sensor node, apart from *sleep* and *active* state. However, for sensor nodes with different tasks and more complex program structures, this approach is not suitable.

The work presented in [VGB07] borrows the definition of [KHZS07] in which energy neutrality can only be achieved if node consumption does not exceed harvest. The authors of [VGB07] claim that a node should always aim at keeping its initial charge level to keep energy neutrality. A linear-quadratic (LQ) tracker is borrowed from control theory to minimize the deviations from the initial voltage level. Since this may lead to abrupt changes in node activity, the LQ-tracker also tries to minimize duty cycle variance.

A controller-based approach for adjusting the activity rate of the sensor node is presented in [MTBB10]. The authors formally describe their controller design which is solved by Integer Linear Programming (ILP). While controller generation is done offline, the authors reduce the complexity of the online approach by only determining the control region of the actual system state. Although their approach allows for rate adaption of different tasks, they do not take dependencies among tasks into account which might lead to energy wastage in more complex applications.

Harvest estimation, c.f. Section 2.4.3, is always prone to errors, which lead the authors of [PL14] to develop P-FREEN. Instead, they use the currently observed charging rate and storage level to solve an average-duty-cycle maximization problem. P-FREEN lacks support for user-defined utility and its performance in times of low-intake remains unclear.

Targeted at long-term sensing (usually year-around), the authors of [BSBT14a] present LT-ENO. It incorporates an astronomical model which maps changes of solar irradiation during the year. However, it assumes a uniform utility and aims to maintain a stable duty cycle throughout the year which is adapted if harvested energy and estimated energy differentiate.

The authors of [RUTR14] present a lightweight algorithm based on binary search. Based on the equivalent circuit, also depicted in Fig. 2.2, a consumption model is developed. The

Model	Horizon	Resolution	Harvest Estim.	U.d. Utility	Tasks	Online	Real-world
Real-Time							
Moser et al. [MBTB07]	t.d.	ms	⊕	⊖	⊖	⊕	⊖
Steck et al. [SR09]	t.d.	ms	⊕	○	⊕	⊖	⊖
HOLLOWS [PBAR10]	hours	ms	⊕	⊖	⊖	⊕	⊕
STAM and STFU [AMMW11]	t.d.	n.a.	○	⊖	⊕	○	⊖
EDeg [GCC11]	t.d.	ms	○	⊖	⊖	⊕	⊖
DEOS [ZMPT12]	t.d.	ms	⊕	⊖	⊖	⊕	⊕
EnOS [WDJ+18]	ms	ms	⊖	⊕	⊕	⊕	○
Planning							
Jiang et al. [JPC05]	n.a.	n.a.	⊖	⊖	⊖	⊕	⊕
Kansal et al. [KHZS07]	day	minutes	⊕	○	⊖	⊕	⊕
LQ-Tracker [VGB07]	day	minutes	⊕	○	⊖	⊕	○
Moser et al. [MTBB10]	day, weeks	minutes, hours	⊕	⊖	⊕	⊖	○
Renner et al. [RUTR14]	day	minutes	⊕	⊖	⊖	⊕	⊕
P-FREEN [PL14]	day	minutes	⊖	⊖	⊖	⊕	○
LT-ENO [BSBT14a]	year	days	⊕	○	⊖	⊕	⊕
Yang et al. [YTR16]	day	hours	⊕	⊕	⊖	⊕	⊖
Hanschke and Renner [HR18]	hours	minutes	⊕	⊖	⊕	⊖	⊖
`PreAct` [GKJZ19]	year	days	⊕	⊕	⊖	⊕	○
SPM & FHC [ABD+19]	days	hours	⊕	○	⊖	○	⊕
Hanschke and Renner [HR20]	hours	minutes	⊕	⊖	⊕	⊕	⊕
`EmRep` [HR22]	days	hours	⊕	⊕	⊕	⊕	○

Table 2.4: Comparison of different energy management algorithms. Existing literature splits into *planning* approaches and *real-time* approaches. t.d. = task dependent.

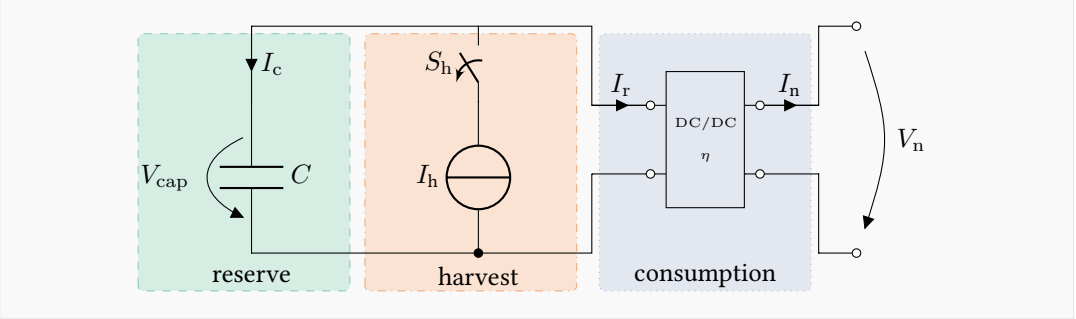


Figure 2.2: Simplified, equivalent circuit serving as basis for the system model of [RUTR14].

goal is to find the maximum average node power consumption restricted by a given policy. Although the depletion safe policy (`DS`)—which keeps the SoC above depletion limit—ensures uninterrupted operation and stable duty cycle, the approach lacks support for time-varying utility. A detailed analysis of this approach is provided in Section 3.1.

The authors of [YTR16] identified that sampling rates of sensors are typically not known beforehand in research experiments. Thus, the authors develop an interactive algorithm which allows change of sampling rate upon user request at runtime considering energy limitations. The authors ensure ENO by keeping track of the saved energy in under-sampling phases and allowing to spend saved energy at a later point in time. However, they do not check for task dependencies while varying the different sampling rates.

The authors of [GKJZ19] present `PreAct`, a long-term energy management algorithm designed for time-varying utility. `PreAct` outperforms the algorithms presented in [VGB07] and [BSBT14a], and can be considered as state-of-the art approach for ENO systems which incorporate user-defined utility. It uses an underlying Proportional-integral-derivative (PID) controller to follow an optimal SoC curve and dynamically corrects harvest mismatch; however, the performance in rapidly changing weather conditions and on low-capacity energy storage remains unclear. The working principle, and its strengths and weaknesses are also analyzed in Section 3.3.

The authors of [ABD+19] present a two-folded approach: first, a finite control horizon (FHC) problem is solved that maximizes the duty cycle for one year in advance. Second, a statistical model is developed based on historical data which provides stochastic guarantees for minimum harvest. SPM and FHC outperform [VGB07] in simulation and real-world testing. The approach does not model user-defined utility and requires offline learning of the stochastic harvest model; the performance in online-only deployments which additionally experience weather-induced changes, remains unclear.

Real-Time

The work presented in [MBTB07] resolves the shortcomings of real-time Earliest Deadline First (EDF) algorithms by presenting two classes of Lazy Scheduling Algorithms (LSAs). Arriving tasks are scheduled as late as possible but with a margin that allows to schedule potentially arriving new tasks without missing their deadline. Unfortunately, LSA relies on independent and preemptive tasks. Furthermore, the approach assumes the power consumption of the node to be continuously adaptable. Since the power consumption has discrete levels, determined by the state of the microcontroller and peripherals, the approach is not directly applicable in practice.

By using Directed Acyclic Graphs (DAGs) and two corresponding ILP approaches, [SR09] describes the tasks of the used platform. The approach tries to find one path in the graph which minimizes the sum of execution times or maximizes the usefulness of tasks, respectively. Again, a rate-based duty-cycling ensures ENO. Their graph-based task model does not include time distances between tasks: each task is executed directly after its predecessor, although this might not be valuable in practice. Furthermore, energy savings in practice remain unclear, since the rate adaption requires re-evaluating the task graph before each new iteration. Since this can be time-consuming on low-power microcontrollers, it is not clear if this leads to energy savings in real-world experiments.

To efficiently schedule tasks of sensor nodes, the strategy of [PBAR10] inserts tasks into prioritized queues. During earlier experiments, the authors found that task arrival times are exponentially distributed, which allows development of a queuing model. Again, the approach does not consider dependencies between tasks, e.g., if information of a task with higher priority is valuable without the information of a task with lower priority.

Two approaches for tackling the scheduling problem of energy harvesting are presented in [AMMW11]. By introducing the concept of *virtual tasks*, the authors opt for smoothing the average power consumption or achieving full utilization. While they show that their concept outperforms LSA and EDF in simulations with static schedules, it is unclear how their approach is influenced by dynamically changing weather conditions and thus a dynamically changing schedule. Furthermore, the approach lacks support for dependencies between tasks.

A real-time scheduling approach called Earliest Deadline with Energy Guarantees (EDeg) is presented in [GCC11] as a variation of the classical EDF algorithm. By simulations, the authors show that EDeg outperforms energy-oblivious scheduling approaches in terms of system runtime and deadline miss rate. One drawback of EDeg is that it assumes tasks to be preemptive without loss of energy. In practice, this is hard to justify when communicating with external devices, e.g., sensors or other nodes in the network. Restarting a task, e.g., polling data from an external sensor, involves repetition and thus wastes energy.

The authors of DEOS [ZMPT12] present a dynamic task-based scheduler which treats energy as primary objective. DEOS analyzes present tasks and check if they share common resources, e.g., the radio, and if they can be split-up in even smaller tasks. Subtasks can be rearranged and combined afterwards with the objective to eliminate redundant interactions with the hardware and allow for concurrent operations, e.g., performing calculations while waiting for an external interrupt. Additional information such as task priority, available energy and task energy consumption allows for a schedule optimization based on previously created subtasks. The superiority of DEOS is demonstrated with a comparison against a simple Weighted Earliest Deadline First (WEDF) algorithm; DEOS outperforms WEDF in both task count and missed deadlines in short outdoor and indoor real-world tests. Unfortunately, DEOS lacks user-defined utility and the performance in times of low-energy intake remains unclear; only cumulative task counts are used as metrics.

In [WDJ+18], a new operating system kernel for ENO systems—EnOS—is introduced and demonstrated. The key idea of EnOS is to divide operation into different *energy modes* which translate to SoC levels. For each energy mode, a different task set is executed—if energy availability is high, tasks are executed at higher rate. Energy modes can be switched only when the system is idle, but switching is optimized to be fast since energy losses due to switching should be low. EnOS lacks support for dependent tasks, is harvest-agnostic and offers only implicit user-defined utility (configuration of task sets).

2.3.2 Intermittent Computing

Recently, a new take on energy-harvesting devices and their operation principle has evolved. Rather than running complex scheduling algorithms on top of potentially unreliable harvest estimation and instead of relying on bulky, expensive, and hazardous batteries, Hester and Sorber advocate Intermittent Computing (IC) on the basis of battery-less devices [HS17b]. In contrast to ENO, devices are only operational during times of abundant energy and their outage is accepted deliberately. Moreover, most IC devices such as [HS17a] are powered directly off capacitors without voltage regulator.

The switch from ENO to IC triggered completely new execution models due to the immanent and deliberate risk of power loss. To mitigate the effects of consistent shutdowns, various models exist to save the program state to non-volatile memory before power-loss [BWM+15, ABA+19, BM17]. To mitigate effects of finding the right spot to save a program's state during execution, task-based execution for intermittently-powered devices is introduced [CL16, MCL17]. The key idea is to split programs into atomic tasks. This is typically left to the programmer or provided by analysis solutions [CL18, MDM+20, KYB+20]. Tasks are restarted if they have not run to completion before power failure, so residual energy assessment is not required. In Mayfly [HSS17], the programmer defines atomic tasks and (time) constraints

between them to keep the value among connected tasks. After power-failure, a device determines if the next task should be executed (time constraints met) or if the previous task has to be repeated (time constraints violated). A notable achievement of Mayfly is that it supports a sense of QoS (execution timeliness and data validity). As the model of Mayfly offers potential, also for ENO systems, it is discussed in detail in Section 3.2.

Above concepts are generally computing-centric; i.e., they assume long operation on obtained data, whereas reading and sharing sensor data is short and can, hence, be put in a single task each. Local communication with peripherals is not considered, frequently fails and has to be restarted. This topic is part of active research, but operation is still tied to niche solutions [BMAS19, RL19, YMP+18]. The type and capacity of the energy buffer are critical in battery-less systems [HSS15]. Recharge times heavily depend on capacity; hence, software is tightly coupled to the hardware; e.g., large capacitors allow more tasks to run uninterrupted but impose long recharging. Hardware with capacitor banks [HS17a, CRL18] lower the burden of choosing the right capacitor size in hardware design phase. A framework automatically determines which capacitor to charge dependent on the demand of the running application.

2.3.3 Replacement or Coexistence?

Recent literature sees a large and fruitful horizon for new work and new problems discovered with IC [RZ19, HS17b]. Often, IC is seen as counterpart of ENO systems which are replaced better today than tomorrow. In the following, the capabilities of IC compared to ENO are examined which underlines that both co-exist instead of IC replacing ENO systems.

IC devices by design operate on very small capacitors and low-power harvesting methods which allows for tiny-sized sensors. Hence, they are suitable for wearables and other areas where seamless integration into the surroundings is key. Additionally, IC devices operate mostly harvest-agnostic which makes them suitable for harvesting sources, such as Radio Frequency (RF) harvesting which offers limited potential for future harvest estimation. When harvest is available, IC devices are capable of transmitting current data values or do calculations without time constraints. Guarantees or estimations for on-times or times of availability, however, are hard to give which complicates communication between intermittently-powered devices. Up until now, communication happens to always-on gateways; multi-hop communication for large-scale deployments is not possible.

In contrast, ENO devices are comparably bulky—dependent on the size of the energy storage. Although low-capacity supercapacitors allow for small-sized footprint in short-term ENO systems, long-term systems designed for year-around sensing still rely on large-scale batteries. The spare energy capacity, however, allows for constant sensing and estimation on future duty cycle and hence availability of the system. Even if ENO systems spend a

large amount of operation time in sleeping mode they can react spontaneously to occurring events causing interrupts—which are missed without present supply voltage. Combining this together allows to offer a certain degree of QoS, e.g., due to user-defined utility. Additionally, participation in large-scale deployments is already possible; ENO systems have been shown to be a reliable partner, even in multi-hop communication, e.g., as presented in [GRHJ+18].

Up until now, ENO and IC solve challenges which partially overlap—but mostly serve different purposes [SCW+19]. This underlines that IC and ENO are rather be used together instead of serving different applications. Concepts that envision ENO systems in hand with IC devices enrich the world of applications powered by ambient energy even further.

2.4 The Challenges of Solar Energy Harvesting

As outlined in Section 2.2.2, solar energy harvesting is the preferred choice in a variety of energy-harvesting deployments. Solar panels are easy to set up and offer high harvest during daytime. Still, operating a solar harvesting system has several pitfalls which are detailed in the following.

2.4.1 Operation Characteristics of Solar Panels

Solar panels, also known as Photovoltaic (PV) panels, convert the energy from the sun into electrical energy using the photoelectric effect. For a deep dive into the physics of solar cells, the reader is referred to [Wür08]. Commonly used solar cells operate at efficiency levels of around 20%. The power which can be generated by the solar cell depends on a variety of factors: the size, the illumination level, efficiency and load.

The *size* of a solar panel is restricted by the operation scenario of the sensor node, e.g., wearables demand very small panels.

The *illumination* level is influenced by controllable and uncontrollable factors. Uncontrollable factors include time of the year, and weather conditions; controllable factors of illumination include deployment site and orientation of the solar panel. As detailed in Section 2.4.3, estimating future illumination level is an active research area.

The *efficiency* of a solar panel is influenced by manufacturing parameters, such as used materials, thickness of cell layers, coating etc., and environmental parameters. Especially environmental parameters, e.g., snow, dust, temperature and aging, influence the efficiency of solar panels throughout the lifetime and are an active research topic [FNSSk21]. Environmental parameters are, however, hard to predict and hence often neglected in simple solar cell models. The *load* characteristics of solar cells, however, are widely studied. Based on semi-conductive materials, solar cells share non-linear electrical characteristics with diodes. Figure 2.3 sketches the I-V-curve as well as the P-V curve of the solar panel. The current decays slightly with

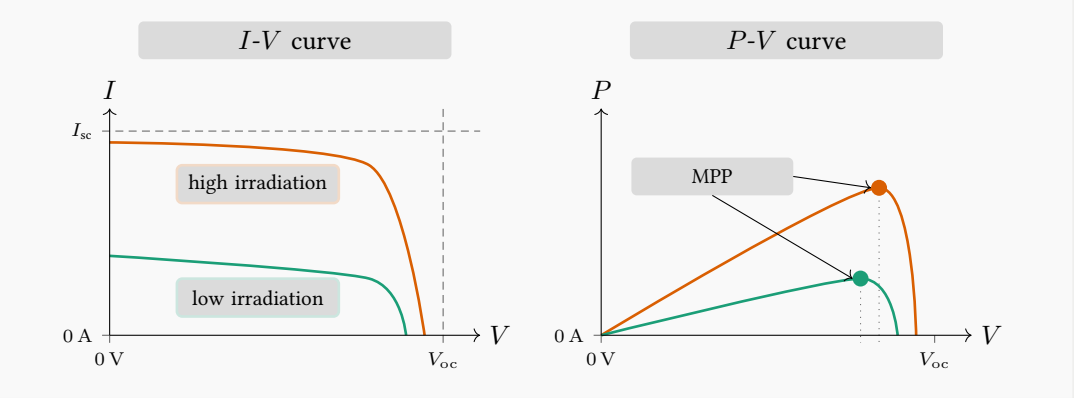


Figure 2.3: Qualitative and simplified characteristics of a solar cell. Higher illumination yields higher solar current. With increasing load voltage, harvest current decays before dropping significantly. The point of peak power is called Maximum Power Point (MPP).

increasing voltage at the solar panel output. Before reaching the open circuit voltage V_{oc}, the current drops significantly and reaches zero. Hence, the power $P = U \cdot I$ increases nearly linearly with increasing voltage. The point of peak power is called Maximum Power Point (MPP) and is commonly considered as targeted operating point. According to [JC12] the MPP mainly depends on the illumination level and is hard to configure statically beforehand. Recharging batteries or supercapacitors is, hence, challenging: If the storage is attached directly to the solar panel, the voltage level of the storage influences the amount of harvested power. This leads to the unwanted situation that the low storage level (low voltage) operates the harvester at a unfavorable operating point and less power is harvested compared to full storage (high voltage). Hence, techniques for the so-called *input load matching* evolve. The authors of [PKK13] present a solution which boosts the voltage at the solar panel output to a more favorable level. Since, the MPP varies with lighting conditions, performing static input load matching may be suboptimal. The solution to this problem is called Maximum Power Point Tracking (MPPT) which deploys a dedicated circuit to keep the harvester near the favorable operating point. An overview of existing techniques can be found in [RBR17].

The need for input load matching is a trade-off between efficiency and harvested power. Direct charging offers high efficiency without conversion losses but requires storage and harvester to operate in the same—potentially narrow—voltage band. Input load matching can broaden this range but either increases circuit complexity or adds additional, energy-consuming, components to the harvesting circuit.

2.4.2 Harvest Profiling

The solar irradiation varies throughout the year due to the Earth's path around the Sun and throughout the day due to the Earth's rotation. Hence, the power harvested from the solar panel

varies drastically, which forces energy-harvesting sensor nodes to keep track of incoming energy and to estimate future intake. Keeping track of present conditions allows the sensor node to save energy harvested during high-intake periods (summer or daylight) and to enable activity in low-intake periods (winter or night). In literature, representing the observed harvest in discrete time intervals—known as timeslots S—is an established approach. The length of the timeslots T_S depends on the profile horizon T_H. Systems which track the diurnal cycle usually use a profile horizon of 24 h, while long-term systems often use a yearly repeating horizon. The T_H is sliced into timeslots of equal length T_S. For short-term systems, T_S usually ranges between 30 – 240 min while long-term systems usually use $T_\mathrm{S} = 1\,\mathrm{d}$. In each timeslot, J samples μ_s^j are collected. To reduce information overhead, a slot value $\mu_\mathbf{s}$ is determined that represents the harvest in the timeslot. An established approach is using the average $\bar{\mu}_\mathbf{s}$ as $\mu_\mathbf{s}$, which is calculated by

$$\bar{\mu}_\mathbf{s} \leftarrow \frac{1}{\mathrm{J}} \sum_{j=0}^{j=\mathrm{J}-1} \mu_\mathrm{s}^j. \tag{2.4}$$

Using an average for the slot values entails additional problems introduced by non-exact representation. The average may be misleading, since it fails to incorporate harvest imbalance within the slot. The harvest may lie above the average at the beginning of the timeslot and the storage fills up early, which mitigates the risk of depletion. The opposite holds if harvest arrives late in the time slot: although this distribution has the *same* average, energy is taken from the storage which has not been harvested yet; hence the risk of depletion increases. This is especially severe when the storage level is near the minimum. An intuitive option is that smaller timeslots offer more knowledge about the harvest profile. Here, benefits have to outweigh increased memory demand and possibly higher computational effort. Literature [RT12], however, found that timeslot lengths below 30 min do not increase the benefit for energy-harvesting systems. Approaches which use variable timeslot length, i.e., shorter timeslots in highly varying conditions, show to increase benefits [CPS16, RT12]. However, in most deployments, simple solutions are preferred due to lower program complexity. To further increase the benefit for energy management, an estimation of future energy intake is needed: the length of the low-intake period has to be known accurately to prevent early depletion.

2.4.3 Harvest Estimation

As the solar harvest varies both yearly and daily, estimation models differ dependent on the estimation horizon. Long-term estimation models are usually employed on energy harvesting systems with large energy buffer so that year-around fluctuations can be compensated, and daily variations can be neglected. Short-term models operate with forecast horizons of hours, spanning up to one day. Here, the daily variations due to daylight as well as local shadowing

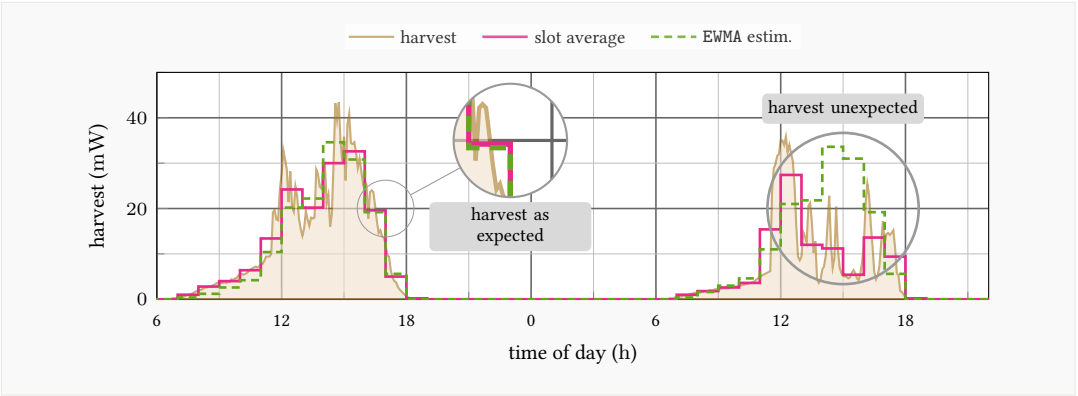


Figure 2.4: Harvest at two subsequent days; harvest on the first day is similar to learned pattern; weather changes on afternoon of second day; EWMA fails to accurately represent second day.

are extremely important. On those systems, energy buffers are often only large enough to cover one low-intake period: the night. Here, local obstructions due to trees or buildings or changing weather conditions severely influence harvest conditions. Multiple algorithms evolved which use past observations to estimate future harvest conditions.

A commonly used approach is to use an Exponentially-weighted Moving Average (EWMA) filter to estimate future conditions; also highlighted in Fig. 2.4. At the end of each slot, the recent slot average is used to approximate the future harvest in the same slot on the next day $\hat{\mu}_\mathbf{s}$ by the formula

$$\hat{\mu}_\mathbf{s} = \alpha \cdot \bar{\mu}_\mathbf{s} + (1-\alpha) \cdot \mu_\mathbf{s}, \qquad (0 < \alpha < 1)\,. \tag{2.5}$$

The EWMA approach is memory-efficient and has low complexity, thus, it is used for many low-power systems, e.g., [KHZS07] and [SGJ+20]. A drawback of EWMA estimation is that a rapidly changing slot value, e.g., due to fundamental weather change, only changes the harvest estimation on the next day—the succeeding slot value remains untouched.

This lead the authors of Weather-conditioned Moving Average (WCMA) [RPBASR09] to develop an algorithm which scales succeeding slot values with currently changing conditions. Changing conditions are expressed relatively to previously present conditions and influence the estimation of succeeding timeslots. While WCMA improves short-term estimation accuracy, 1 – 3 h, daily estimation performance suffers because implications between far-distant timeslots are small. The idea of using past observations of weather conditions to improve harvest estimation is also used by the authors of [CPS16] in Profile Energy Prediction Model (Pro-Energy). The Pro-Energy approach stores multiple *typical* representations of daily harvest, e.g., a sunny morning followed by cloudy afternoon. Continuously, Pro-Energy compares current harvest with the stored representation and estimates future

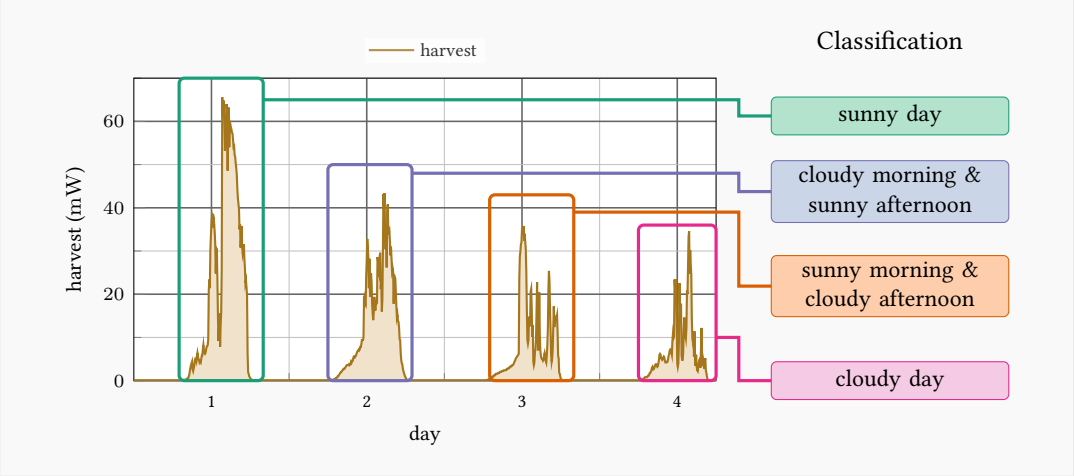


Figure 2.5: Principle of `Pro-Energy`; daywise patterns are learned and stored as separate profiles; harvest estimation for next timeslots based on most similar profile.

values by assuming the current day behaves as one of the most similar representations. The principle of `Pro-Energy` is also highlighted in Fig. 2.5. The authors of `Pro-Energy` only evaluate their approach for estimation horizons up to 2 h, hence the performance for a larger horizon—e.g., half a day (12 h)—remains unclear. This may be particularly concerning for state-of-charge estimation for the succeeding part day since uncertainty of estimation influences the accuracy of state-of-charge estimation.

Numerous other approaches exist, e.g., by incorporating cloud cover forecast [SGI+14, Ren13], atmospheric transmittance [DALT18], and machine learning techniques [ATMA18], which aim to minimize the estimation error. However, an ongoing debate exists about the necessity of—possibly—erroneous harvest forecast. A variety of energy harvesting systems neglect harvest estimation by design, e.g., [WDJ+16] and [YTJ+19]. Those systems tie their operation to present harvest and risk early depletion.

2.4.4 Conclusion

Using PV cells for supplying energy harvesting systems draws considerable research interest since it offers high energy intake in outdoor deployments. Solar panels are relatively easy to install and operate. Dependent on the horizon, e.g., year or day, the per-node prevailing harvesting pattern can differ drastically due to local shadowing. However, monitoring the prevailing pattern of energy intake is necessary to equalize energy storage throughout low- and high-intake periods. Harvest intake estimation may extend the knowledge about future intake—taking into account learned patterns from the past. Numerous approaches exist which mitigate the influence of weather-induced harvest variability. Monitoring harvest intake,

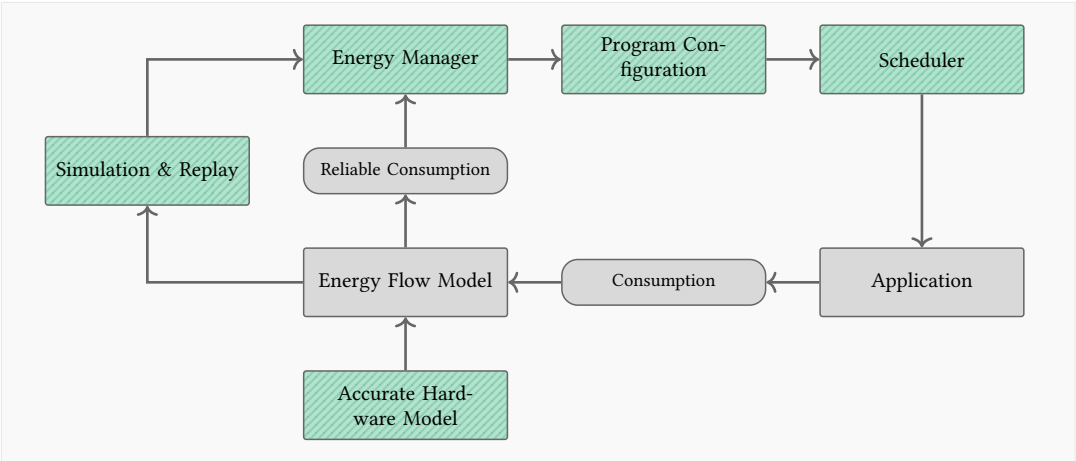


Figure 2.6: Interplay between necessary components for operation and development of energy-neutral sensor node; rectangular boxes are major components; colored boxes indicate components altered and improved in this dissertation.

incorporating harvest estimation, and closely modeling solar cell characteristics is, hence, key for careful adaption of the activity of the sensor node.

2.5 Open Research Directions

The ever-increasing popularity of the IoT opens new directions for applications and deployments. While the number of sensor nodes is still increasing and deployments face increasing sensor densities, powering those sensor nodes efficiently still poses new challenges. Operating billions of sensor nodes on wired supply or—possibly—hazardous disposable batteries is economically and environmentally unfeasible. Duty-cycling alone only stretches the lifetime of a sensor node, but leaves it limited. Only energy harvesting enables near-endless operation of once deployed sensor nodes. Solar energy is a high-density source that allows for usage in a variety of deployments; however, rapidly changing weather conditions and non-linear charging characteristics complicate operation.

Energy management for ENO devices—but also IC devices—still faces open challenges in areas becoming more important for future applications. Energy management for uninterrupted operation supports—until now—only programs on sensor nodes that assume uniform power consumption. Large spikes in power consumption, as imposed by advanced sensors, need energy management which incorporates this by design. The challenge, hence, is to design an *energy manager* that can be adjusted via *program configuration* to adapt the sensor node energy consumption, c.f., Fig. 2.6. Energy management is even more complex if applications demand different levels of activity throughout the node operation, e.g., varying utility.

Furthermore, the *scheduler* needs to take account of timely task execution. As also presented in Section 2.3.2, measured physical phenomena depend on each other and are, hence, connected by time constraints. This view on sensor node programs for ENO system does not exist yet.

The basis for development in energy management and scheduling is an *accurate hardware model* because it enables the sensor node to improve its energy usage. Only if accurate information of hardware power consumption is available to the sensor node, an algorithm can be developed that improves energy-efficient load adaption. *Simulation and replay* frameworks and hardware additionally enable development of these algorithms.

An in-depth analysis of the open challenges is presented in the following chapter.

Chapter 3

Analysis and Contributions

As highlighted in Section 2.3, energy management for energy-harvesting systems draws considerable research interest. However, evolving applications pose new demands which are yet to be completely covered by energy management. This chapter addresses the major challenges:

- depletion safety, i.e., perpetual operation under varying harvest conditions, (Section 3.1),
- support for task-based programs, i.e., scheduling for graph-based structures (Section 3.2), and
- time-varying utility, i.e., usefulness expression for energy-aware scheduling (Section 3.3).

For each of the major challenges, a detailed motivation, introduction to the current state-of-the-art approach, and an analysis of present weaknesses as well as open research directions is provided. Based on this analysis, the core contributions of this dissertation are identified and detailed in Section 3.4.

3.1 Depletion-safe Operation of Energy-harvesting Sensor Nodes

As highlighted in Section 2.3.3, the most compelling argument for ENO systems compared to IC devices is the uninterrupted operation and availability. The goal to prevent outages by adjusting the activity of the sensor node is commonly known as depletion safety. Prevention of outages needs an estimate of the harvest intake as well as good knowledge about energy storage and node consumption characteristics. In [RUTR14], three strategies for perpetual operation are presented—of which two are suitable–that couple harvest and node consumption for energy management. The assumptions, characteristics and used hardware are motivated and detailed in the following.

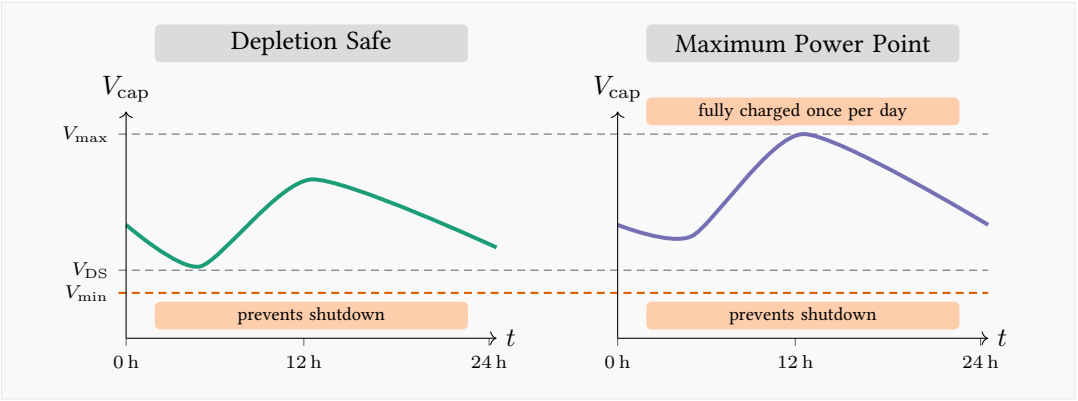


Figure 3.1: Comparison of the two presented policies—*depletion safe* and *maximum power point*. The depletion safe policy ensures capacitor voltages above a threshold. The maximum power point policy additionally aims at charge to capacitor fully once per day. As consequence, voltage levels of the depletion safe policy are lower in general. However, the achievable average node power consumption is higher in this example.

Relevance

The foundation to provide sustained operation of energy-harvesting devices is deep knowledge and accurate modeling of the underlying hardware. As highlighted in Section 2.4, supplying sensor nodes with solar harvesting has several pitfalls. The benefits of the simple hardware architecture, c.f. Fig. 2.2, are low cost and less electronic components potentially experiencing defects. Since the node has to adapt on changing environmental conditions, SoC estimation hence poses a trade-off between estimation accuracy and computation effort: it should be lightweight enough to run on low-power hardware, but accurate enough to model real-world conditions. Furthermore, depletion safety is key since downtimes make nodes unreachable which may impose severe consequences, e.g., missed control signals.

Existing Strategy

In [RUTR14], the authors present a consumption model and energy management algorithm to adjust the activity of a sensor node. The simplified equivalent circuit, found in Fig. 2.2, allows the authors to simulate the SoC course of the energy storage. Together with a model from [RT12] the course of the capacitor voltage can be estimated by:

$$C \cdot \dot{V}_{cap} + I_h = \frac{I_n \cdot V_n}{V_{cap} \cdot \eta\left(I_n, V_{cap}\right)}. \tag{3.1}$$

Here, the energy is stored in a capacitor with nominal capacity C at voltage level V_{cap} and harvested with the current I_h. The efficiency of the boost converter η depends on the node load current I_n at node voltage V_n and voltage level of the capacitor. This equation is solved

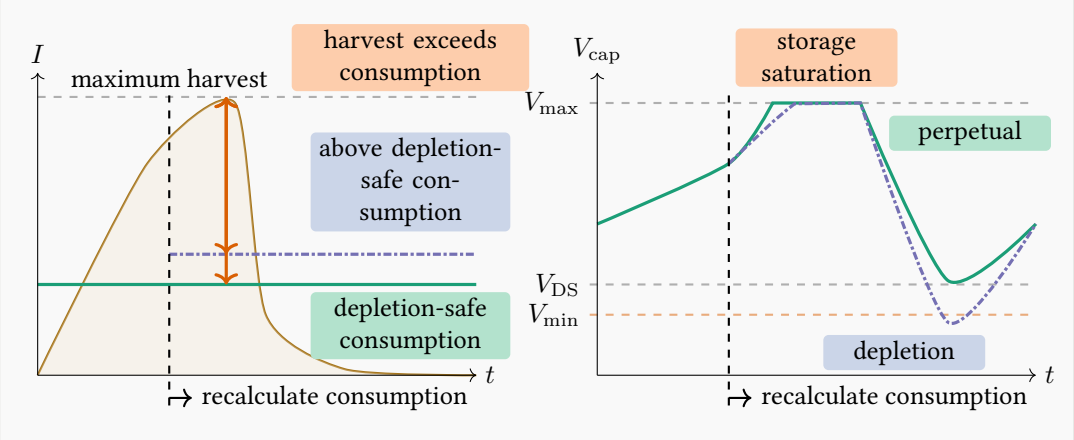


Figure 3.2: Illustration of prevailing storage saturation. In times of harvest surplus, maximum harvest exceeds average node consumption by far. The depletion safe budget cannot be increased without risking depletion. Energy storage saturates and harvest excess is not used for additional activity.

for the course in time of V_{cap}. This voltage simulation is based on a binary search algorithm. The authors suggest two suitable policies: depletion safe `DS`, and maximum power point `MPP`, also depicted in Fig. 3.1. The former ensures a capacitor voltage $V_{cap} \geq V_{DS}$ within the horizon, while the latter additionally aims to achieve $V_{cap} = V_{max}$ at least once per day. The algorithm runs at the end of the timeslot for the next horizon; and gives *one* maximum compliant load I_n^*—also called budget B. While the `DS` approach tends towards low voltages throughout the day, the `MPP` policy leads to significantly higher voltages throughout the whole day. Although voltages well above V_{min} further decrease risk of depletion—caused by erroneous harvest estimation—it also increases the time with full capacitor; especially in times of high harvest. In those situations, the overcharge protection disconnects the solar panel and incoming energy is neither used for charging nor for supplying the sensor node.

Open Challenges

Although the presented approach prevents depletion efficiently, it leaves room for performance improvements—especially in times of high harvest. As the primary objective is to prevent the capacitor voltage from dipping below V_{min}, it is generously accepted that the capacitor is fully charged for large parts of the day. This issue is depicted in Fig. 3.2 and detailed in the following.

Challenge 1

Storage saturation at harvest surplus without increased activity.

While this may not appear problematic from the perspective of depletion safety, it is inefficient since incoming energy is neither stored nor used for activity—it is wasted. The reason for this is that harvest during daytime is usually one magnitude higher than the average node consumption throughout the day. Even the maximum average consumption B is much lower than harvest in bright sunny conditions. Still, the node prevents increasing B because of imminent depletion during the night—the SoC minimum at the end of the night hence dictates the maximum node consumption.

Furthermore, the `DS` approach assumes the node to have a near-stable average power consumption.

Challenge 2

Depletion safety for task-based programs with high power demand.

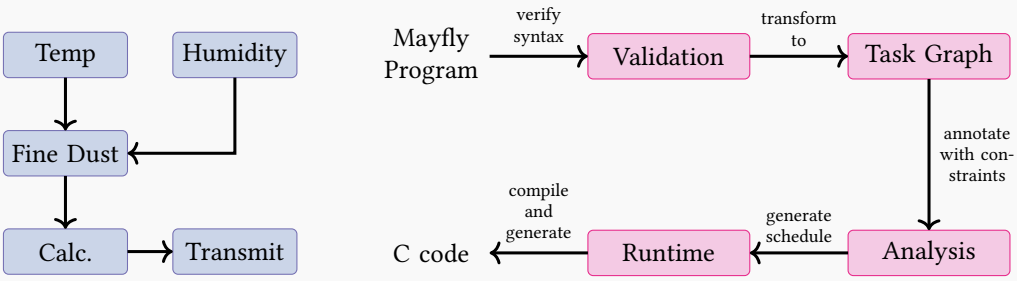
While this holds for simple sensors, e.g., temperature or humidity, more power-hungry sensors such as fine dust, lead to high peak power consumption peaks. These peaks drive impose significant storage level drops which are not modeled yet. This goes hand-in-hand with the lack of support for modern task-based programs, as presented in Section 3.2. Here, tasks may occur in batches which may influence model accuracy since the energy storage level decreases within a short time window.

3.2 Task-based Program Structures

Newly evolving applications such as outlined in Section 2.1 exceed the horizon of classical duty-cycle-centric scheduling approaches. For intermittent computing, the Mayfly language presented in [HSS17] allows the programmer to describe time relations between different steps of the program. The motivation and actual strategy, as well as open challenges are detailed in the following.

Relevance

If offering a specific meaning for an application, e.g., providing a classification of air quality, involves multiple steps, their time relation between each other is important. The process of fine-dust monitoring, for example, involves multiple atomic sub parts, called tasks. Figure 3.3(a) highlights a potential program design of a fine-dust sensor node. Here, in-time measurements of temperature and humidity are needed before the fine-dust sensor can be sampled; afterwards, calculations and a transmission of the data packet follows. Even up-to-date fine dust sensors draw up to 360 mW which is around $6\times$ more than a typical sensor node power consumption of 60 mW. This produces a considerable drop in energy storage level which may risk depletion



(a) Fine-dust sampling exemplarily shown as task-based program flow.

(b) From writing the program in the Mayfly language to C code generation. Schedule generation involves constraint checking and leads static prioritized ordering. Adapted from [HSS17].

Figure 3.3: Fine dust sampling presented as graph and process for generating a Mayfly executable.

safety; hence the node has to spend time to recharge the energy storage afterwards. This, however, bears the risk of waiting/sleeping too long—and transmit potentially outdated values afterwards. Therefore, a language to describe time constraints between atomic tasks is needed.

Existing Strategy

The Mayfly language requires the program developer, to divide its program into atomic subparts called *tasks*. These tasks should ideally complete without emptying the battery. Data collected or processed within the task is stored in non-volatile memory. Furthermore, program tasks are connected by edges so that the program is represented as a graph—called *program flow*. Tasks are connected via edges which contain time constraints for connected tasks. Tasks should not be executed too fast after each other, but fast enough so that measurements are not outdated. Furthermore, multiple samples within a time interval shall be collected. Once a valid schedule (ordering) for the program is created, the Mayfly *runtime* ensures timeliness while the node is deployed. Before the next task is started, the runtime checks if the defined time constraints are met. The process of generating a Mayfly executable is depicted in Fig. 3.3(b).

Open Challenges

Although Mayfly offers an efficient way to describe time relations between tasks, it has the considerable drawback that these constraints are only checked *after* execution. In intermittently-powered systems, this may be sufficient but bears the risk of energy wastage since tasks may have to be restarted. ENO systems have the benefit of potentially uninterrupted operation

and spare energy for scheduling. This allows to simulate the node activity in beforehand, and hence check if task execution is feasible *before* execution.

Challenge 3

Task-based program structures for ENO systems.

This allows to increase energy efficiency since tasks are only executed if they benefit the application, i.e., meet time requirements. Additionally, Mayfly constraints offer a certain degree of scheduling freedom, since they define time intervals. By design, Mayfly does not exploit this freedom, e.g., stretching the time distance between tasks.

Challenge 4

Parameterized scheduling heuristic for task-based programs.

Here, a greater safety margin can be introduced which may decrease risk for early depletion. Furthermore, Mayfly only generates the schedule for the program at compile-time. Hence, the program flow can not be changed while the node is already deployed. This limits program design flexibility and increases testing efforts since misconfiguration can only be resolved by writing the entire program to the sensor node again.

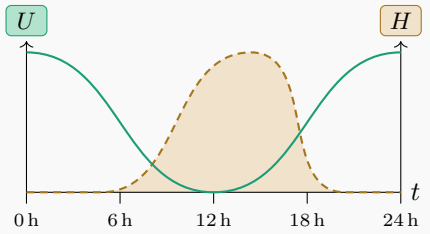
Additionally, the authors of Mayfly do not offer a simulation toolkit to test the performance of their program. Especially the degree of atomicity, i.e., if tasks are small enough to run without interruption, is prone to misconfiguration and requires deep knowledge by the program developer. A flexible simulation framework which allows easy change of program flow parameters allows developers to test their programs for energy harvesting system in beforehand—and hence help to develop more solutions efficiently.

3.3 User-defined Time-varying Utility

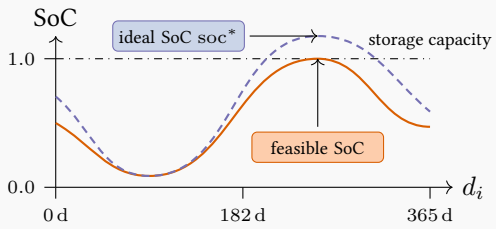
As outlined in Section 2.1, applications served by energy-harvesting systems exceed *best-effort* sensing policies. In energy-harvesting systems, best-effort strategies lead to high activity when harvest is high, and low activity when harvest is low. This, however, may not be beneficial for every application which is outlined in the following.

Relevance

If times of interest and times of high harvest mismatch, c.f. Fig. 3.4(a), energy is spent without increasing the benefit for the application. The term to describe the importance for an application is known as the dimensionless *utility*. This utility can be user-defined and



(a) Example for mismatched harvest H and utility U.



(b) SoC and ideal SoC in principle of `PreAct`.

Figure 3.4: Example scenarios for harvest and utility mismatch and the working principle of state-of-the-art approach `PreAct`.

hence allows the application to express the level of interest. Figure 3.4(a) shows an example scenario: harvest and utility mismatch which complicates energy management.

Existing Strategy

The most recent state-of-the-art approach which incorporates time-varying utility into energy management for ENO systems is `PreAct` [GKJZ19]. `PreAct` targets ENO systems with large batteries which are designed for year-around sensing. `PreAct` uses the harvest estimation algorithm presented in [BSBT14b] and a user-defined utility to calculate an ideal SoC course for the year. As highlighted in Fig. 3.4(b), the ideal SoC curve may not be feasible since it exceeds the battery capacity. Hence, the curve of the ideal SoC is scaled by its amplitude, i.e., the difference between maximum SoC and minimum SoC within the year. The result is a target SoC which fits the storage capacity and is used as input for the PID controller. To reach the target SoC on the next day, the PID controller calculates the duty cycle for the next day.

Open Challenges

As `PreAct` targets long-term ENO systems, it operates only at day resolution. While the structure of `PreAct` is flexible enough to allow for change of horizon and resolution, the performance remains unclear; especially due to faster changing harvest conditions. State-of-the-art estimation techniques, such as `Pro-Energy`, however, rapidly alter their estimate based on experienced weather changes. This leads to frequent changes of the ideal SoC which may influence the performance of the underlying PID controller. Additionally, `PreAct` relies on large batteries. With small energy storage, however, energy availability varies drastically within a day.

Challenge 5

User-defined utility for short-term ENO systems with varying harvest conditions and small storage.

Furthermore, the underlying PID controller is fine-tuned for the given hardware; it is unclear how it performs without a priori knowledge of control parameters. Finally, the definition of the duty cycle and spent energy remains unclear. While the authors state that spent energy cannot exceed harvest energy in long term to fulfill ENO, it is unclear how this definition works with shorter horizons of one day.

3.4 Contributions

Based on the previous explanations and discovered research areas, the core contributions are detailed subsequently.

Contribution 1

Development of a mathematical formulation for incorporating task-based programs into energy management of ENO systems.
Addresses: Challenge 3.
Addressed in: Chapter 5.

The mathematical formulation uses the concept of task-based program structures of the Mayfly language. The approach advocates usage of the same language centered around timeliness for IC devices, as well as ENO devices. Graph-based program structures and their underlying time constraints are used to generate starting times for time-slotted scheduling. All time constraints are transformed to the standard canonical form which enables usage of a standard ILP solver which outputs the schedule. By using randomized program structures and time constraints, performance-determining factors are identified. It can be shown that the number of tasks but especially the average node power consumption lead to a strong increase of execution time for defining and solving the ILP problem. Furthermore, estimations on microcontroller performance are presented which indicate that the approach is feasible, but the efficiency leaves room for improvement.

Contribution 2

Extension of the depletion safe strategy with a task-based parameterized scheduling heuristic.
Addresses: Challenges 2, 3 and 4.
Addressed in: Chapter 6.

The time constraints of the mathematical formulation are used to develop a scheduling heuristic based on Mayfly and the `DS` strategy. Arbitrary task-based program structures are integrated into online scheduling for energy-harvesting sensor nodes. Based on Kahn's algorithm, used for topological sorting of graphs, the heuristic finds starting times within a timeslot for multiple program structures. The scheduling heuristic can be parameterized in multiple ways by altering two key parameters: the scheduling `STRATEGY` and `BALANCE`. The scheduling `STRATEGY` can be used to group tasks greedily together to finish tasks early; lazy if preventing outages has high priority; and to match the average power consumption best. Furthermore, `BALANCE` targets the already discussed problem of present harvest imbalance throughout the slot duration; and hence allows the algorithm to arrange tasks at the beginning, at the end of the timeslot, or to distribute them uniformly. The heuristic is deeply integrated into a simulation toolkit which offers effortless exchange of harvesting condition, program configuration and hardware power consumption. It enables future researches in EH-WSNs to test their programs or sensing systems *before* deployment and hence speeds up system installment. By altering capacitor size and harvest estimation technique, the extensive simulation results show the influence on outages and activity. It is shown that the scheduling heuristic mitigates common difficulties faced with solar energy harvesting systems, e.g., the discussed early-morning depletion. Additionally, the activity can be increased by up to 28% without loosing depletion safety. A short real-world case-study based on solar-powered fine dust monitoring is presented which underlines the effectiveness of the scheduling heuristic.

Contribution 3

Integration of utility into short-term ENO energy management in hand with SoC estimation.
Addresses: Challenges 1 and 5.
Addressed in: Chapter 7.

The energy manager `EmRep` integrates time-varying utility into short-term ENO system together with SoC estimation. `EmRep` builds upon the depletion safe approach combined with in-day varying utility. It targets the discussed storage saturation problem at high harvest intake and increases node activity without risking depletion safety. `EmRep` uses SoC estimation to identify low and high energy intake phases—and uses two different management techniques. In

the low-intake phase, depletion safety is ensured via the DS algorithm. In high-intake phases, the *free* strategy is used: here, the primary objective is to increase utility for the application but to simultaneously ensure that the start of the DS phase is entered with the same SoC as before. This avoids storage saturation in high-intake phases by design and achieves the same utility level—but with smaller sized storage and hence footprint. Extensive simulation results study the influence of capacitor size, harvest estimation technique and utility profile. Across various configurations, EmRep increases the effective utility for the application. Furthermore, a short case-study for bridge health monitoring is presented in which EmRep increases the performance significantly.

Chapter 4

Reliable Testing of Energy-harvesting Sensor Nodes

To test and evaluate the algorithms presented in this dissertation, repeatable evaluation of energy-harvesting sensor nodes is needed. The motivation for reliable testing is outlined in Section 4.1. Details and efficiency measurements of the harvester based on a solar panel and supercapacitor are presented in Section 4.2. The simulation toolkit introduced in Section 4.3 simulates the presented harvester and closely emulates its behavior. In Section 4.4, the testbed is introduced and examined. The testbed is able to emulate and replay harvesting traces accurately and hence helps with developing energy management algorithms.

4.1 Introduction

The deployment of sensor networks has several pitfalls, e.g., as described in [LBV06], which slow down system setup. This does not stop for sensor nodes operating on energy harvesting: every bit of energy has to be spent carefully. Hence, software malfunction or unexpected hardware operation has to be reduced. Especially for harvesting systems, hardware and software are closely connected, which demands reliable testing of the system before deployment. Since every Joule counts, in-lab measurements of the hardware are needed to assess power consumption in every hardware state, e.g., sleeping and transmitting data. Additionally, harvester, node and energy storage often operate at different voltage levels; conversion, however, imposes additional losses that need to be modeled. Especially with power-hungry sensors (e.g., fine dust), and radio modules (e.g., WiFi), the existing platform does not suffice. Hence, the solar harvesting circuit is updated to match more power-hungry hardware, c.f. Section 4.2. To keep the reliability of the consumption model detailed in Section 3.1 with the updated hardware, new measurements are performed with respect to changed conversion efficiencies.

Long-term experiments for energy harvesting systems are time-consuming and hard to reproduce. Solar harvesting sensor nodes usually employ harvest estimation techniques which need a distinct time until a pattern is learned, and a consistent performance can be achieved. Furthermore, harvesting conditions vary dramatically, e.g., due to local shadowing as highlighted in Section 2.4; hence, even neighboring nodes experience different conditions. Moreover, different storage size, harvester size and estimation algorithms lead numerous combinations which are infeasible to evaluate time-efficiently in experiments. Hence, a simulation framework is developed, c.f. Section 4.3, which emulates the presented harvester with real-world harvesting traces.

Additionally, real hardware ages due to environmental effects, e.g., dust, pollen and micro defects on solar panels. This complicates fair repetition of real-world experiments. Hence, a testbed is developed and presented in Section 4.4 that uses controllable light sources to emulate previously recorded harvesting conditions. Moreover, artificial traces can be fed into the lighting testbed to evaluate corner cases and highlight coherence between real-world hardware and simulation setup.

4.2 Hardware Architecture

The harvester is based on the simplified equivalent circuit depicted in Fig. 2.2. In the following, the harvester and its components are presented in detail, necessary measurements for output voltage stabilization efficiency are examined, and the leakage current of the supercapacitor is obtained.

4.2.1 Overview

The original circuit is extended since prior measurements [HHR16] revealed an unstable supply voltage at high load power consumption. The harvesting circuit is designed for supercapacitors with maximum voltage of $V_{\max} = 2.7\,\mathrm{V}$ and solar panels with open circuit voltage of $V_{\mathrm{oc}} = 4\,\mathrm{V}$. Tests have been performed with supercapacitors with capacity of 50 – 400 F and solar panels rated with maximum current of 25 – 250 mA. Directly integrated on the harvesting board shown in Fig. 4.1, is an Analog to Digital Converter (ADC) with 12 bit resolution, which is attached via I2C to the microcontroller. The ADC allows voltage measurements at four channels with variable reference voltage for dynamic accuracy. The ADC supports bipolar measurements—between positive and negative reference voltage—but also supports unipolar measurements. With unipolar measurements, resolution is reduced to 11 bit which lowers measurement accuracy. However, the accuracy still suffices for the dedicated purposes; hence, the commonly-used, low cost and low power TI ADS 1015 is used for the harvester platform. The ADC is used to measure the capacitor voltage and the charge current of the solar panel I_{h}.

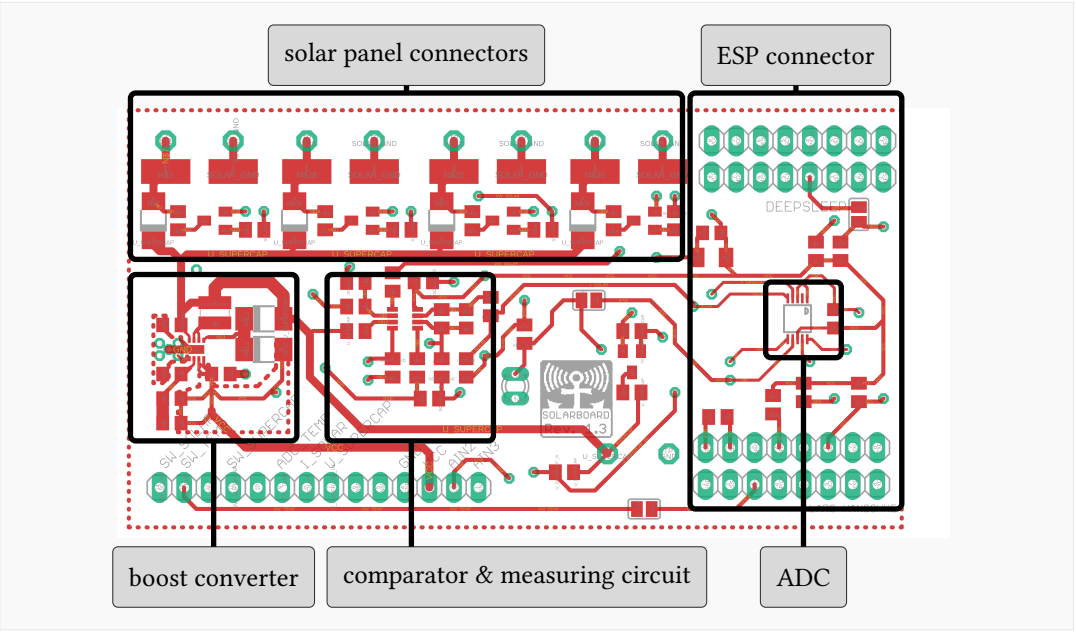
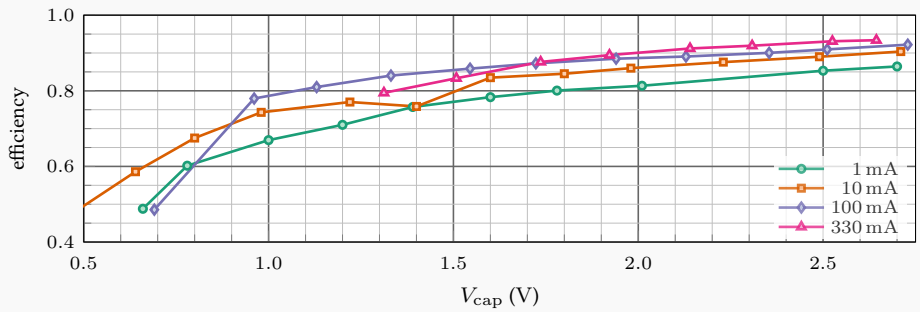


Figure 4.1: Layout of the adapted harvesting board; size is 8.5 × 4.2 cm; microcontroller can be easily swapped via custom breakout board.

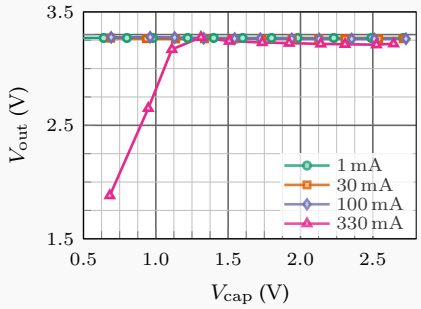
While the voltage of the capacitor can be measured directly, obtaining the harvest current requires more effort. Since very low currents have to be measured, an amplifier circuit based on a TI OPA 2369 is used. The harvest current flows through an 1 Ω measuring shunt; the resulting voltage is amplified and measured by the ADC. The used configuration of resistors ensures that harvest currents of up to 250 mA can be measured against the reference voltage. Exploiting the full measurement range, the ADC achieves a—theoretical—measurement accuracy with $2^{11} - 1$ samples for I_h of 0.12 mA. Furthermore, a comparator circuit monitors the capacitor voltage and disconnects the solar panel from charging if the capacitor reaches $V_{max} = 2.7\,V$. Attached to the supercapacitor is a boost converter—a TI TPS 61021A—that is configured to provide a stable output voltage of $V_s = 3.3\,V$. Note that the upconversion efficiency highly depends on load current I_n and the capacitor voltage V_{cap}; this dependency is investigated later. The harvesting board is designed to fit one of the two microcontrollers—Espressif ESP32 and ESP8266—which integrate via breakout board. An overview of measured power consumption of both microcontrollers in relevant states can be found in Table 4.1. The harvester board alone without load attached consumes 109.9 µW at the converter output. Further measurements of characteristics of the boost converter are presented in the following.

	Espressif ESP8266	Espressif ESP32
deep sleep (µW)	85	86
active (mW)	49	112
WiFi transmitting (mW)	352	357

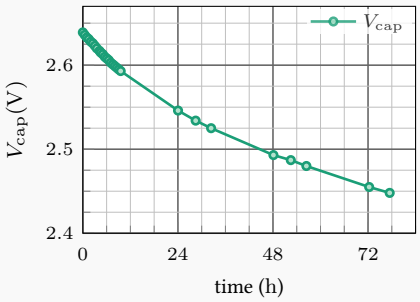
Table 4.1: Measurements of used Espressif microcontrollers; obtained with TI INA 139 measurement amplifier.



(a) Efficiency of converter



(b) Stability of converter



(c) Self discharge

Figure 4.2: Efficiency and output stability of boost converter and supercapacitor leakage trace for the presented prototype.

4.2.2 Supply Voltage

The power conversion efficiency η of the circuit for typical load currents is determined. A controlled laboratory supply is connected at the input of the converter to emulate different levels of V_{cap}. On the output side, different resistors are attached to emulate different load power consumption. The results are shown in Fig. 4.2(a). For $V_{cap} \geq 1.5\,V$, η ranges from 78 – 93 % with relatively low dependency of the load current. Only for input voltages close to V_{min}, η drops significantly for high load currents. However, the minimum input voltage V_{min} to operate the converter safely depends on the required output current. $V_{min} = 1.3\,V$ for the used hardware is obtained for the converter with load currents from 1 mA to 330 mA to mirror

consumption from sleep state over reading sensors to WiFi usage, c.f, Fig. 4.2(b). Given the relatively low variation of η and in order to speed up energy management, a constant $\eta = 70\%$ is used for simulation unless otherwise noted.

4.2.3 Capacitor Leakage

Furthermore, the supercapacitor leakage current is measured for inclusion in the simulation model. For this purpose, a supercapacitor with nominal capacity of 400 F is charged to its maximum voltage of 2.7 V until the charging current diminishes. In the following, the voltage drop is monitored with a high-impedance multimeter for 75 h as shown in Fig. 4.2(c). All experiments are taken at room temperature, i.e., 22 °C. Due to physical characteristics of the capacitor, high leakage currents of up to 680 µA are obseverd at high energy levels, settling to 143 µA at around 2.5 V. Through another charge and discharge experiment with a steady charging current and an ohmic load, respectively, a real (or empirical) capacitance of 382 F for the same capacitor is obtained.

4.3 Simulation Toolkit

For accurate simulations, a toolkit is developed that accurately emulates the harvester presented in Section 4.2. Figure 4.3 depicts the simulation architecture and highlights exchanged information via interfaces between the modules. The toolkit consist of three main compartments: configuration, simulation and on-node software.

4.3.1 Structure

The *configuration* part houses the parameter set for the evaluated approaches, e.g., the settings for harvest estimation, harvest database and task configuration. The main dataset is a 194-day solar trace collected with a miniature energy harvester similar to the prototype hardware at resolution of 5 min. Most common data sets only provide samples with resolution of 30 – 60 min which limits insights on timeslot harvest imbalance, as highlighted in Section 2.4. With the task configuration, arbitrary program structures can be loaded via CSV file and hence easily be changed. The task configuration also contains necessary power consumption at V_s to enable energy-aware management and scheduling.

The *simulation* itself contains the main simulation loop with discrete time steps of 5 min. Furthermore, the harvester is emulated, and the voltage course is simulated with the formulas

Publication

Parts of Section 4.3 are published in [HR20].

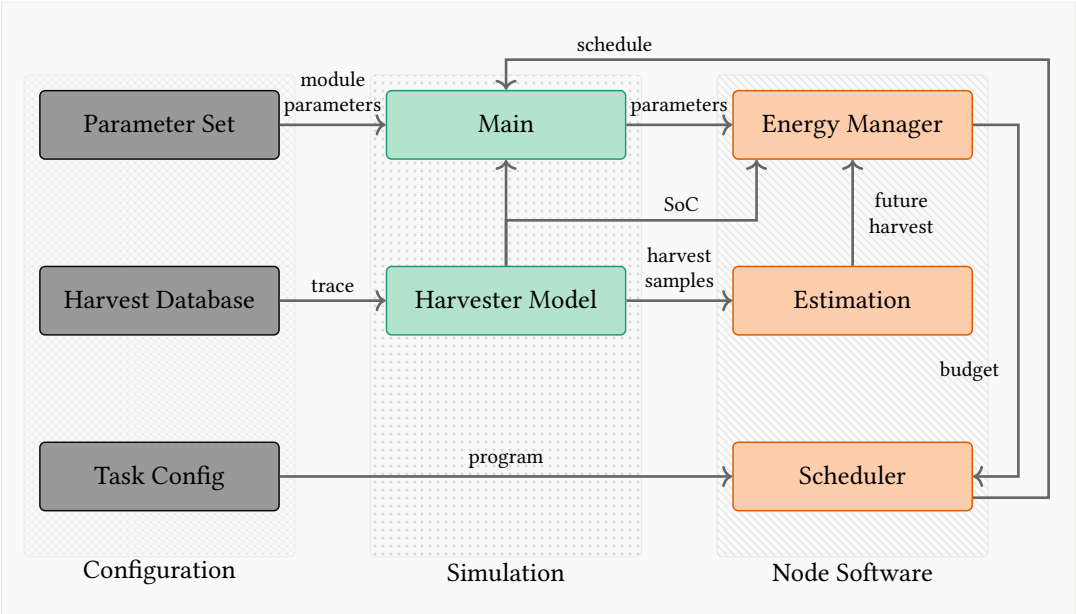


Figure 4.3: Overview of the used simulation framework; boxes represent modules used in software; arrows indicate implemented interfaces between the modules.

presented in Section 3.1. The simulation toolkit is modified by adding a model for the efficiency of the boost converter, and by replacing its simple consumption model by task execution, where tasks have a duration and consumption analog to the program model used in Chapter 5. Moreover, the supercapacitor leakage current is included by adding it into sleep consumption. Finally, simulation resembles node failure, when the supercapacitor voltage falls below a critical threshold V_{min} that is insufficient to operate the boost converter any longer. Similar to the real hardware, the converter powers up again, only if the supercapacitor voltage rises above V_{en}. The data loss occurring with voltage memory storage solution is also modeled, e.g., the harvest pattern is lost and has to be learned again.

4.3.2 Implementation

The *node software* is written in C++ and can be easily ported to common microcontrollers. It implements different modules, namely the energy manager, the estimation algorithm and the scheduler. All modules share common base types and interfaces so that they can be easily exchanged.

4.4 Lighting Testbed

To enable repeatable experiments in controlled lighting conditions, a light box is developed and examined.

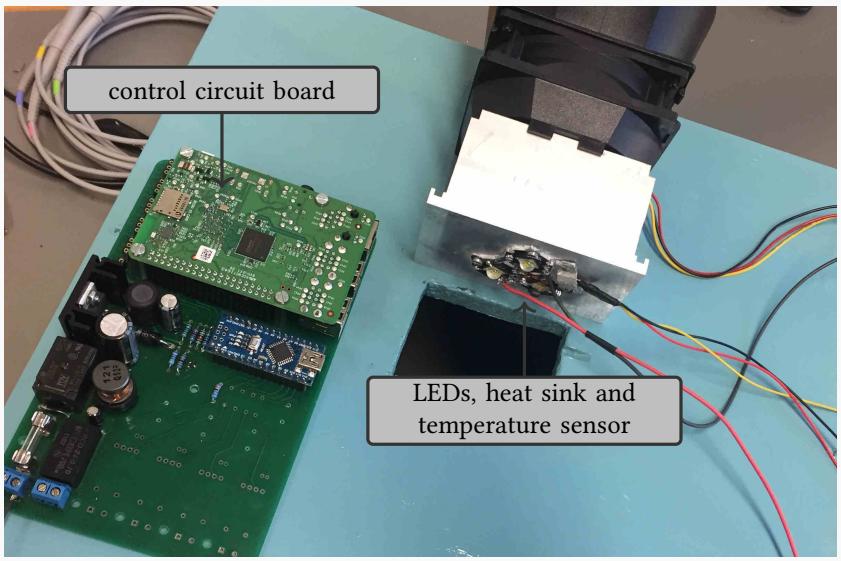


Figure 4.4: Light box with high-power LEDs; full brightness yields 75 mA at the solar panel with the used harvester; self-developed control circuit and temperature monitoring.

Comparing energy-harvesting algorithms under equal conditions is hardly feasible in practice: days with equal solar radiation profile are rare and impossible to predict. Especially situations of further interest, e.g., an unexpectedly decreasing energy intake, are hardly reproducible. However, for debugging and optimization, understanding the potentially not ideal behavior of algorithms is key for improvements. Consequently, the sensor node's harvester has to be fed by an entity offering: stable conditions in repeated experiments, accuracy within the limits of the sensor node's sensing circuitry and easy-to-use exchange of replayed scenarios.

Replaying already happened corner cases; e.g., a failed long-term test, is hence essential for evaluation. If this situation can be exactly replayed afterwards in the laboratory as observed by the sensor node, additional options of debugging can be applied and consequently modifications in the algorithms can be tested under controlled circumstances.

While reproducing current-voltage traces is sufficient in most cases, practical issues of solar cells and reaction of the sensor nodes cannot be modeled: e.g., dust or pollen may reduce energy intake in reality and also smaller damages can only be investigated in laboratory with

Repository

The simulation framework developed in this dissertation is available open-source via https://collaborating.tuhh.de/e-24/public/wsn/eh-scheduling-and-management.

the solar cell. Therefore, a manual emulation of these real-world issues should be possible. Additionally, solar cells experience slow transient processes in rapidly changing lighting conditions; investigating these issues without the need for complex changes in hardware structure leads to a deeper insight of the behavior of solar-energy harvesters. Thus, a solution is needed for replaying recorded real-world lighting conditions providing the same solar energy-harvesting conditions, which should be as closely to the scenario seen by the energy-aware sensor node.

This leads to the development of the light box, depicted in Fig. 4.4, which uses high-power LEDs, is remotely configurable and uses affordable off-the-shelf hardware components. The light box is able to replay already recorded lighting conditions from the environment but also artificial traces through current and voltage traces by a sensor node to enable subsequent testing of energy-aware and -predictive algorithms in the laboratory.

The prototype harvester presented in Section 4.2 is used to calibrate the platform to ensure the replayed scenario is as close to the originally recorded real-world situation as possible. In consequence, no special sensing equipment is needed for the presented accuracy: only a communication channel between lighting control and harvester inside the box is needed to calibrate and start emulations.

The rest of the section is structured as follows: First, the goals as well as hard- and software structure of the light box are introduced in Section 4.4.1. Second, techniques are shown in Section 4.4.2 to achieve the accuracy in the range of the harvesting platform and third, the performance is evaluated in Section 4.4.3 of the light box concerning reaction of the energy-aware sensor node.

4.4.1 Hard- and Software

The goal of the light box is to take the real-world data as input and reproduce them so accurately that the sensor node inside the light box experiences the same situation as outside. Furthermore, recorded cases shall be replayed multiple times to alleviate uncontrollable effects, such as varying wireless channel quality.

The system structure of the light box and the utilized hard- and software are presented in the following.

Publication

Parts of Section 4.4 are published in [HRB+17].

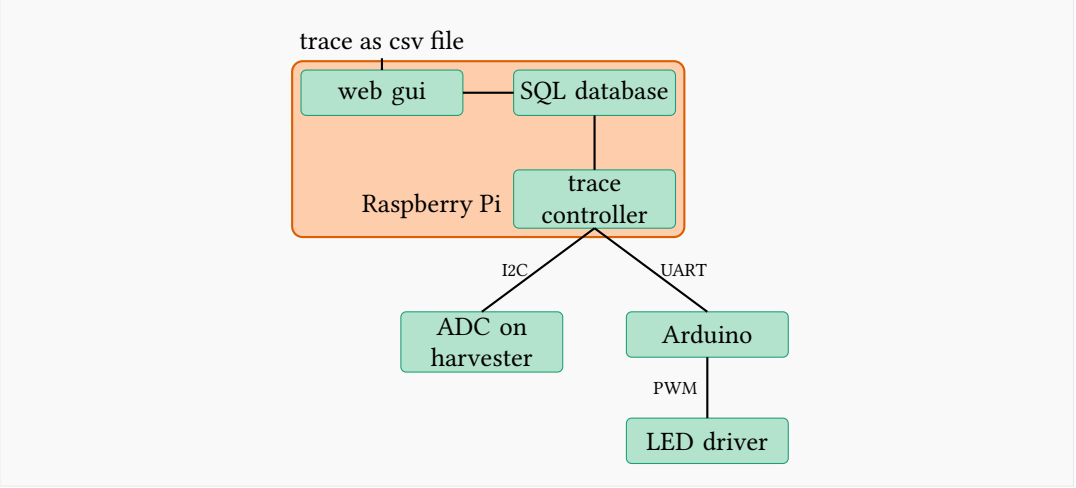


Figure 4.5: System structure of the light box; input of traces via web page, storage and trace control are hosted by a Raspberry Pi; hardware PWM for LED control is provided by an Arduino Nano.

System Structure

The harvester detailed in Section 4.2 is used to record real-world lighting conditions through solar current and capacitor voltage traces during a long-term test. It is also possible to feed custom traces to evaluate corner cases or performance in situations of special interest. These traces contain triples of delivery time, solar current and capacitor voltage.

As shown in Fig. 4.5, these traces serve as input for the trace controller, which runs on a Raspberry Pi. It is powerful enough to host the easy-to-use web GUI and the database.

However, the PWM signal for controlling the LEDs might be unstable if it is generated in software by an operating system handling other processes. Thus, an Arduino Nano is integrated, which only receives commands via UART and adjusts its output PWM duty cycle upon reception of the configuration command. The control circuit and its components are explained in the following.

Repository

The source code and schematics of the light box are available open-source via `https://collaborating.tuhh.de/e-24/public/wsn/light-insight`.

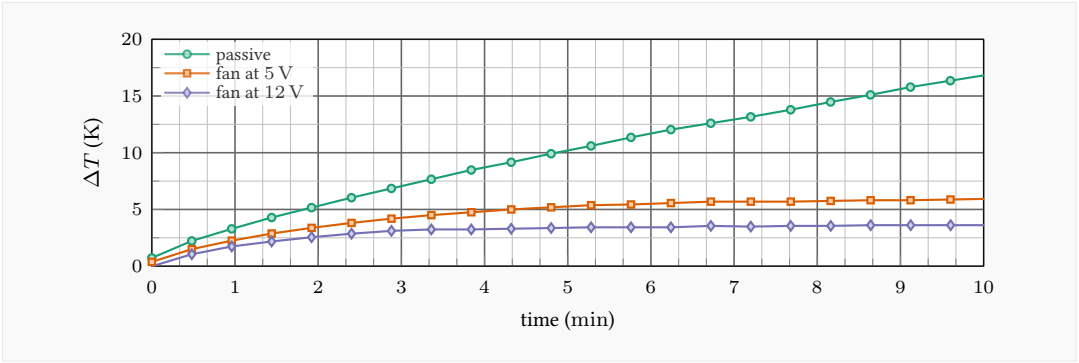


Figure 4.6: Temperature difference to ambient temperature at heat sink; junction temperature at LED is expected to be 60 K higher; passive cooling insufficient.

Hardware

The control circuit board of the light box hosts four major components: a Raspberry Pi, an Arduino Nano, a DC-converter and an LED driver. The supply voltage for the Arduino is directly provided by the Raspberry Pi. Control commands are sent via UART to the Arduino Nano, which generates the PWM signal with adjustable duty cycle to control the LED driver. This driver controls the current through the LEDs and consequently the illumination level.

The two used high-power LED's have a light temperature of 6200 K, which is close to what is observed in direct sunlight (6000 – 6500 K). Integration of LEDs with different light temperatures is possible, but influence on harvest is still discussed, c.f. Section 2.4. With an additional reflector and two high power LEDs, the light box is able to achieve a harvester output current of up to 75 mA.

To keep the internal temperature of the box at a low level, an efficient heat dissipation from the LED junction is needed. The light box setup uses a conventional CPU heat sink with active cooling. As displayed in Fig. 4.6, the difference between heat sink and ambient temperature quickly rises beyond 15 K without active cooling. Since this already influences the lifetime of the LED and the brightness level, an active cooling solution is implemented. Two different voltage settings are tested: the fan runs at its maximum speed at 12 V but 5 V are directly available on the control circuit board. Both voltages provide a sufficient cooling, but it is decided to use 5 V to save additional voltage converting circuits. With control modules and two LEDs at maximal brightness, the box consumes 21 W. The total bill of material is around 110 €.

Software

A key requirement of the light box is its remote configurability. The Raspberry Pi hosts a web page with GUI for the light box control, which is depicted in Fig. 4.7. In the control panel

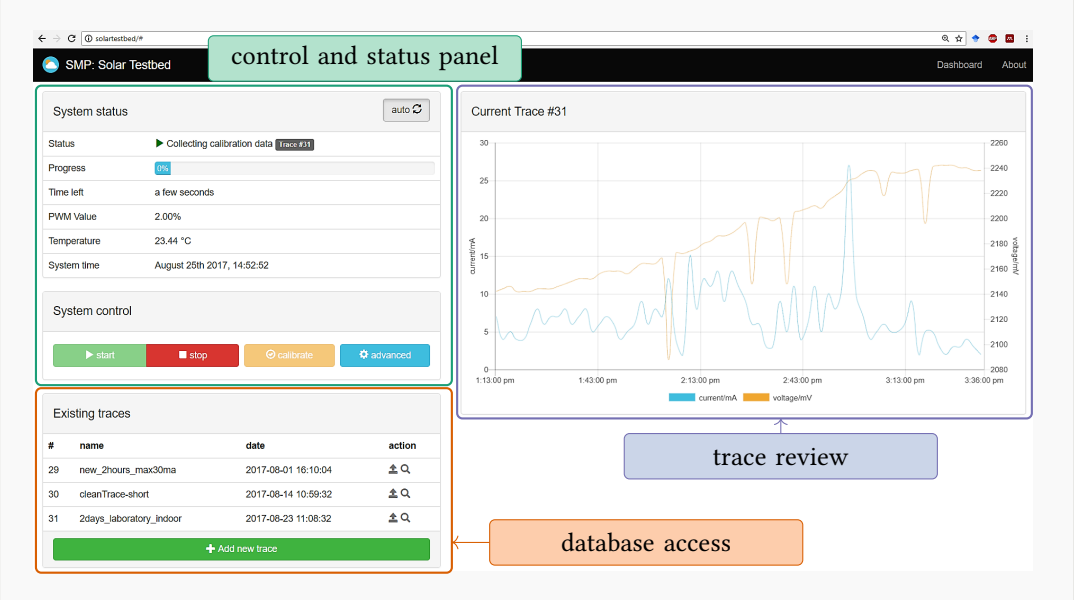


Figure 4.7: GUI allows comfortable upload of recorded traces via web browser as CSV files; traces can be reviewed after upload; system control allows monitoring of progress and experiment control.

area, the user can start and stop the experiment, perform a new calibration or shut the whole system down. Before starting a newly loaded experimental trace, the box is calibrated, c.f. Section 4.4.2, to match the PWM values and the current. To replay recorded and artificial traces accurately, further steps during calibration are necessary which are detailed in the following.

4.4.2 Techniques for Accurate Replay

To replay recorded situations, the same hardware platform is used for recording, calibrating and replaying. Comparability is achieved by applying two techniques before starting any experiment which is explained in the following.

Accurate Calibration

The lighting conditions in the box are controlled by applying a distinct PWM duty cycle to the LED driver, which regulates the current through the LEDs. The resulting light generates a current at the solar panel, which charges the supercapacitor and can be monitored by the harvester of the sensor node inside the box.

Before starting the experiment, a calibration is needed to map PWM values to the desired lighting conditions. The calibration curve differs upon used solar cell, distance between LEDs and solar panel and positioning of the solar cell inside the box or the illuminated area on the

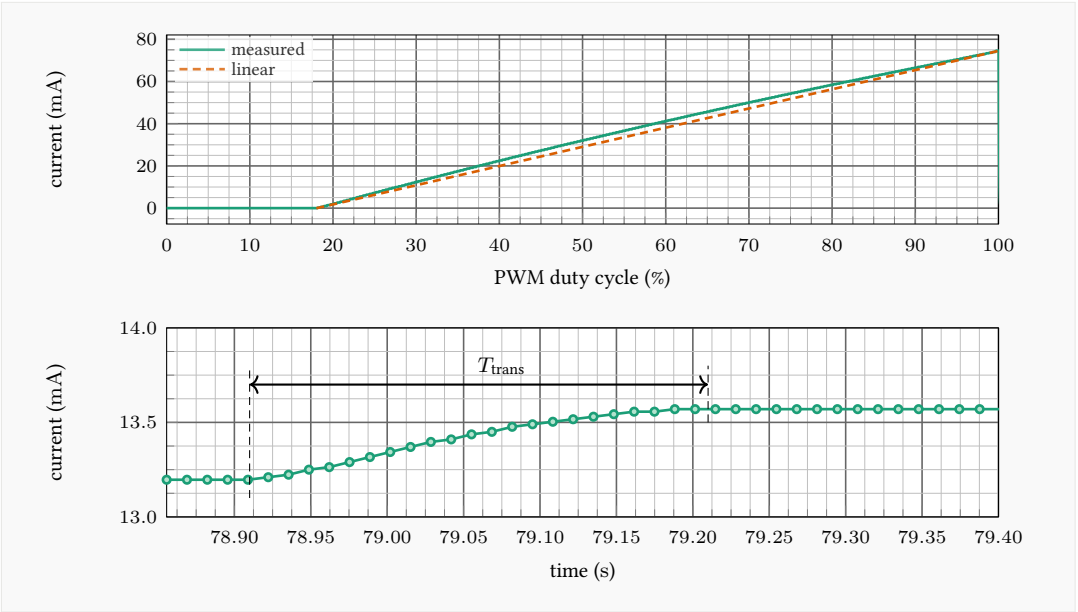


Figure 4.8: Calibration trace links PWM duty cycle and solar current; LEDs start working above 18% PWM duty cycle; between PWM steps, harvesting circuit experiences transient behavior manifesting in T_{trans}. $T_{\text{trans}}^{\text{max}} = 2.93\,\text{s}$ was observed at most.

solar panel, respectively. Since exact placement of the harvester cannot be ensured between experiments, e.g., after flashing a new software on the sensor node, it is decided to calibrate the lighting conditions before each experiment.

The calibration procedure works as follows: each PWM value is applied for a calibration time T_{calib}. At the end of this timespan, the Raspberry Pi reads the solar current and the capacitor voltage from the ADC of the harvester and stores them in the so-called PWM-current map. The resulting calibration trace is depcited in Fig. 4.8. The difference between measured and ideal curve is mainly caused by non-linear behavior of the LEDs, which is also treated in the data sheet [Cre17]. Due to the minimum current of the LEDs, a PWM duty cycle of approximately 18% is needed for the lowest illumination level. As the box uses an Arduino Nano to achieve real-time PWM control, values between 0 and 255 are available for adjusting the duty cycle and the lighting level in the box, respectively. Even with wasting a fifth of the PWM values, at full brightness a step size of 0.36 mA is achieved, which is equal to 3 steps of the ADC of the harvester.

Due to the already studied transient effects of solar cells during calibration a careful adjustment of the calibration time is needed. The transient time T_{trans} is defined as the time difference between applying a new PWM signal and stabilization of the solar current. A stable current is assumed when five subsequent samples at 100 ms sampling interval give the same ADC value. During the calibration tests, a median and maximum transient time

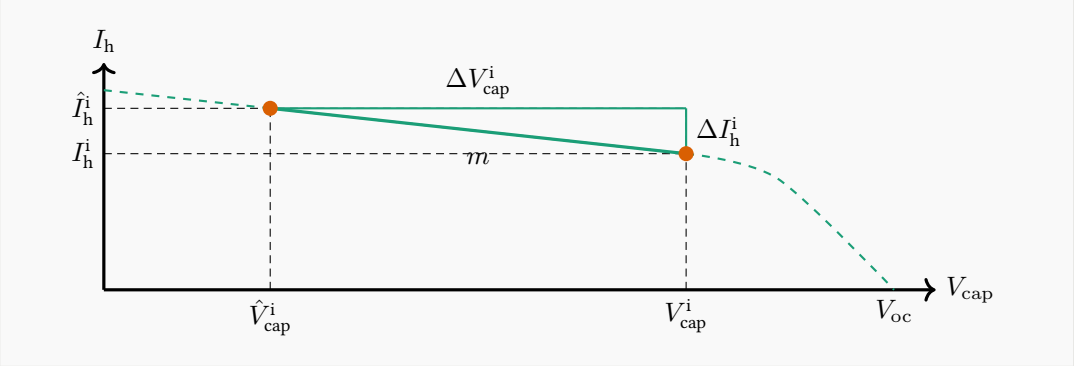


Figure 4.9: Capacitor voltage $\hat{V}^\mathrm{i}_\mathrm{cap}$ during calibration might differ from voltage $V^\mathrm{i}_\mathrm{cap}$ in replayed trace; instead of wanted current I^i_h, $\hat{I}^\mathrm{i}_\mathrm{h}$ is generated; before replay, the trace is compensated by applying an offset factor $\Delta I^\mathrm{i}_\mathrm{h}$ to each trace point.

of $T^\mathrm{med}_\mathrm{trans} = 1.86\,\mathrm{s}$ and $T^\mathrm{max}_\mathrm{trans} = 2.93\,\mathrm{s}$ are observed, respectively. Applying a calibration time of $T_\mathrm{calib} = 3\,\mathrm{s}$ to each of the 256 PWM values yields an overall calibration time of 13 min.

Compensating Power Point Shifting

The prototype harvester uses a direct charging circuit for the supercapacitor; thus, the voltage of the capacitor influences the power point of the solar panel and consequently the amount of harvested power.

During calibration, the generated power at the solar cell is influenced by the voltage of the supercapacitor, c.f. Section 2.4. During replay, the capacitor voltage might differ from the voltage during calibration; thus, the generated current is different from the value stored in the database. During replay, a PWM value with an expected current is applied, but since the capacitor voltage changed, the observed current differs from the expected current due to the calibrated PWM-current map. During compensation, the current difference between calibrated value and recorded value is wanted: in consequence, a higher or lower PWM value has to be chosen than originally expected. This issue in is depicted in Fig. 4.9. The recorded solar traces consist of triples containing time, solar current I^i_h and capacitor voltage $V^\mathrm{i}_\mathrm{cap}$. The PWM-current map contains triples, containing PWM value, the capacitor voltage $\hat{V}^\mathrm{i}_\mathrm{cap}$ and the generated solar current $\hat{I}^\mathrm{i}_\mathrm{h}$. Since it is expected that $V^\mathrm{i}_\mathrm{cap} \neq \hat{V}^\mathrm{i}_\mathrm{cap}$, $I^\mathrm{i}_\mathrm{h} - \hat{I}^\mathrm{i}_\mathrm{h} = \Delta I^\mathrm{i}_\mathrm{h}$ is observed. Consequently, the observed solar current during replay may be smaller or larger than expected. If $V^\mathrm{i}_\mathrm{cap}$ is not closely located to the open circuit voltage V_oc, an approximately linear dependency can be found: the known voltage difference $\Delta V^\mathrm{i}_\mathrm{cap}$ can be used to calculate

$$\Delta I^\mathrm{i}_\mathrm{h} = m \cdot \Delta V^\mathrm{i}_\mathrm{cap} = m \cdot (V^\mathrm{i}_\mathrm{cap} - \hat{V}^\mathrm{i}_\mathrm{cap}). \tag{4.1}$$

After calculating $\Delta I^\mathrm{i}_\mathrm{h}$, a compensated PWM value for the current $I^\mathrm{i}_\mathrm{h} + \Delta I^\mathrm{i}_\mathrm{h}$ is selected.

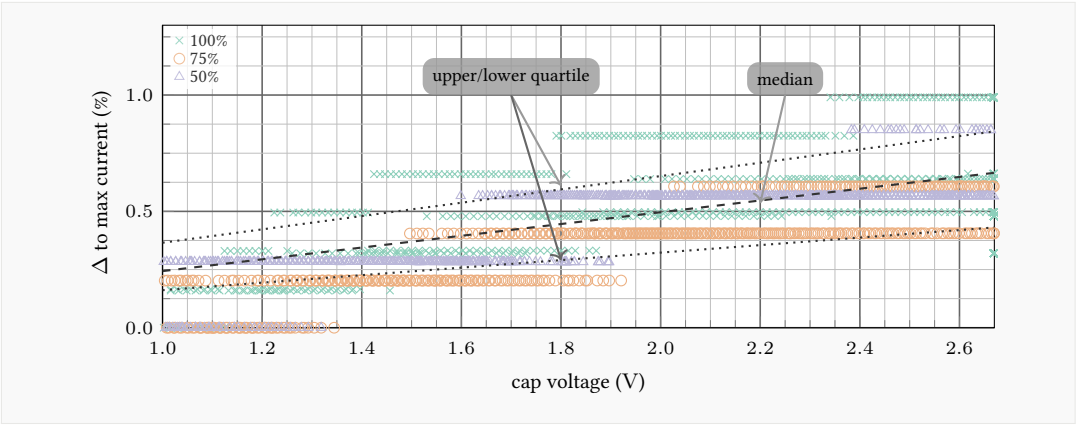


Figure 4.10: Relative difference between maximal current and observed current at shown capacitor voltage; less current can be drawn from the solar cell with increasing voltage; linear curve of the median gives gradient for compensation.

To obtain the gradient m, current-voltage traces at different illumination levels are recorded. The results in are displayed in Fig. 4.10. Note that the values are normalized to the maximum solar current of the corresponding illumination level to compare them side-by-side. The switching regulator of the harvesting platform works with capacitor voltages down to 1.0 V. The solar cell has an open-circuit voltage of 4 V whilst the supercapacitor voltage is rated at a maximum voltage of 2.7 V. This ensures the condition that $V_{\mathrm{cap}}^{\mathrm{i}}$ is far off V_{oc}.

To compare the different illumination levels, the difference towards the current at 1.0 V capacitor voltage is calculated, and the difference is normalized by dividing by the maximum current. Based on all data points depicted in Fig. 4.10, the linear curve fitting yields a median relative gradient m' of 0.26 %/V. Note that m' has to be multiplied by the maximum current to obtain m which is mandatory for our compensation technique. A higher relative gradient is observed for higher illumination levels, e.g., for 100% up to 0.45 %/V are observed. Using $\Delta V_{\mathrm{cap}}^{\mathrm{i}} = 1.7\,\mathrm{V}$ and a current of 75 mA yields $\Delta I_{\mathrm{h}}^{\mathrm{i}} = 0.57\,\mathrm{mA}$. Since this is already in the range of the step size and thus emulation capability, the influence is expected to be minor compared to the right choice of the calibration time.

4.4.3 Results and Practical Merit

In this section, the impact of the different calibration and compensation methods are presented and the repetition accuracy for subsequent experiments is examined.

Calibration Methods

The benefit of the compensation method is presented in Fig. 4.11. The overall charge fed into the charging circuit is compared during the replay against the charge of the recorded

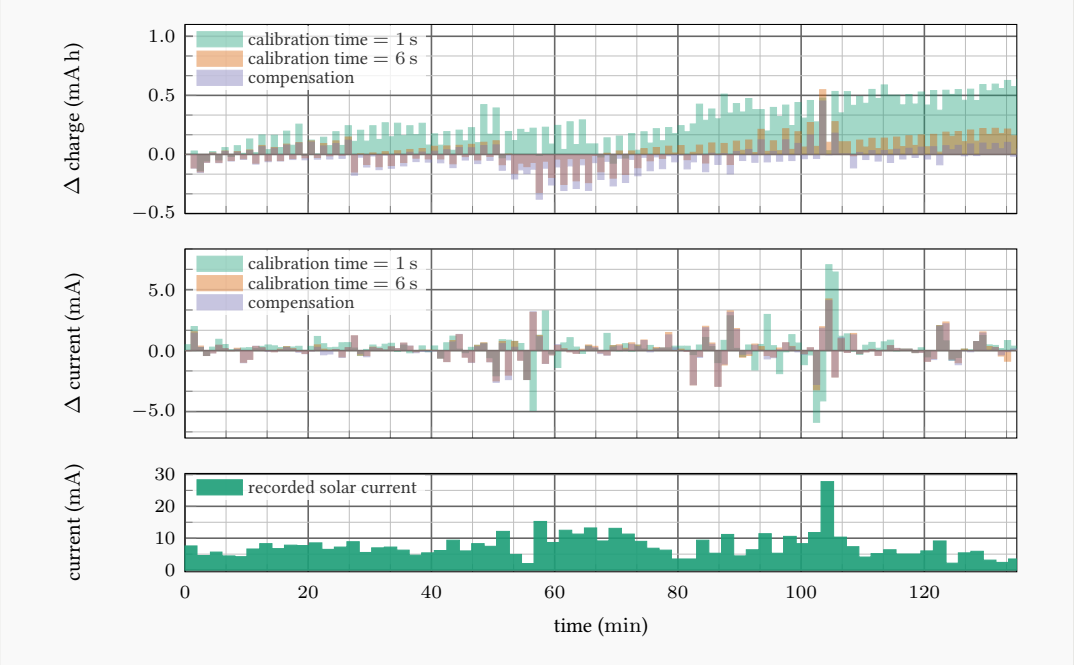


Figure 4.11: Difference between replayed current trace and observed input at harvester; charge is summed up error during replay; a higher calibration time and compensation of capacitor voltage difference is needed to increase accuracy.

trace. Since most energy-aware duty-cycling algorithms evaluate the energy storage level of a node, the amount of charge fed into the storage unit at the end of the experiment is especially interesting. The same sensor node and harvester is used throughout all experiments. With short calibration time of 1 s, the charge difference at the end of the simulation is 0.52 mAh, which results in a voltage difference of 37.44 mV with a 50 F capacitor. Using a calibration time per PWM value of 6 s yields a charge error of 0.01 mAh. With the total amount of charge fed during the emulation of 17.23 mAh, this results in a relative error of 0.58%. The voltage difference with the used 50 F capacitor is 7.2 mV. The compensation method presented in Section 4.4.2 yields a charge error of 0.001 mAh, leading to a capacitor voltage error of only 0.72 mV. Since the ADC compares the capacitor against a reference voltage of ±4.096 V, this result is already below the accuracy which is used for the energy-aware algorithms.

The practical merit is also shown in the difference to the original current traces: the observed trace with the developed compensation in mean only differs by 0.62 mA, which is in the range of the measurement accuracy. The on-board 12 bit-ADC has a resolution of 0.12 mA. This underlines that, within the accuracy range of the used harvester, the light box is able to reproduce environmental lighting conditions accurately.

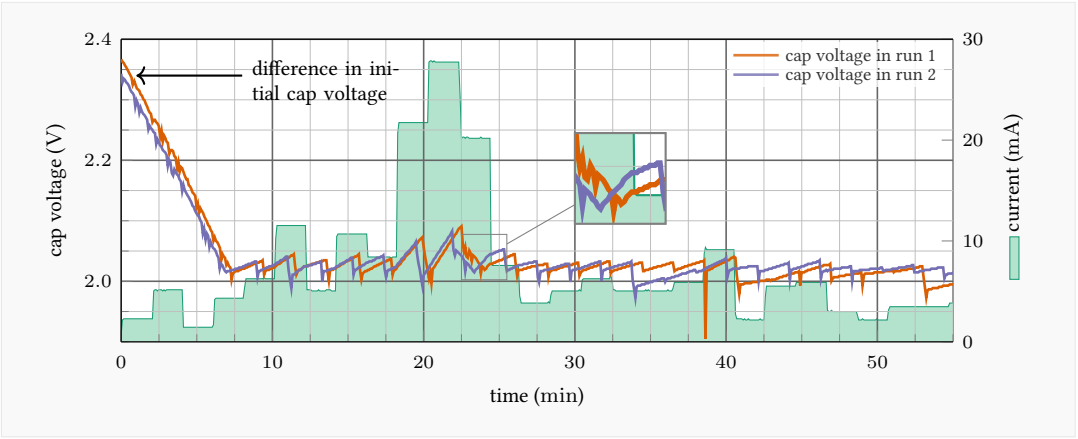


Figure 4.12: Repeated experiment; different initial capacitor voltage hinders exactly reproducible voltage traces; reaction of harvester is repeatable, i.e., gradient of capacitor voltage.

Repetition Accuracy

To alleviate uncontrollable conditions such as varying radio channel conditions, subsequent repetitions of experiments are needed. Two experiment runs are compared using an energy-aware but simple program on the sensor node. The node operates as long as it has energy and enters a sleep mode if detected otherwise. The sensor node queries the ADC of the harvesting platform four times a second. If the capacitor voltage is larger than 2.0 V it stays active, otherwise it goes to sleep for 2 min to save energy waiting for the capacitor to charge. Another harvester with the same hardware is connected to the inputs of the ADC inside the box and samples values every 100 ms to obtain a timely fine-grained image without influence of the wireless channel.

The resulting capacitor voltage trace of both runs and the replayed solar trace are portrayed in Fig. 4.12. Since the initial capacitor voltages differ, the curves do not exactly overlap. A small change of just 20 mV results in unequal decisions of the sensor node. If the nodes are in the same state, their power consumption is equal as well as the solar current which charges the capacitor. This translates to an earlier point in time, at which the threshold voltage of 2.0 V is reached and the sensor node shuts down at an earlier point in time. Consequently, more effort has to be spent to adjust the initial capacitor voltage, which is hard to do manually. The key observation can be seen in the magnification figure. When two nodes in the different runs are in the same state and consequently have the same power consumption, the gradient of the two voltage curves is the same; especially when the illumination level changes rapidly. For example between approximately 24 and 25 min, the capacitor voltage increases in both runs by 1 mV each six seconds. The Root-Mean-Squared Error (RMSE) of the solar current between both runs is 0.07 mA, which is below the measurement accuracy of the solar harvester.

Conclusion

An affordable light box, which is able to reproduce environmental conditions repeatedly in the laboratory, was developed and examined. It was shown that the box achieves measurement accuracy within the range of the harvesting platform which underlines that it is sufficient for investigating algorithms relying on typical low-power platforms. The box can be easily extended with additional LEDs for larger solar cells or higher light intensity. The light box is also used in Section 6.4.2 to show coherence between real-world hardware and the simulation framework.

Chapter 5

Task-based Program Structures for Energy-harvesting Sensor Nodes

As highlighted in Section 3.2, task-based program structures offer an easy way to describe time relations in programs for EH-WSNs. Hence, this chapter introduces the entailed scheduling problem and proposes a solution based on a standard mathematical solver.

5.1 Introduction

In the following, the motivation for defining a scheduling problem for task-based programs as graphs is outlined. Furthermore, the core contributions for energy-harvesting sensor nodes are illustrated.

5.1.1 Motivation

The popularity of WSNs, based on still increasing computational power, new sensing capabilities or sharing of WSNs, offers a plethora of new application scenarios, c.f. Section 2.1. Sensor nodes are equipped with multiple sensors, e.g., fine dust, humidity or ozone, or different radio interfaces, e.g., LoRa, WiFi or IEEE 802.15.4. With an increasing number of different peripherals, the complexity of the underlying program structure grows steadily. Each of these radio interfaces or peripherals imposes a new task on the sensor node. Since these tasks are influenced by physical phenomena, e.g., detecting a change or reacting on a control command, they are not independent of each other. The sampling rate has to be adapted to the physical phenomenon, but additional information of the surroundings has to be considered,

Publication

Parts of this chapter are published in [HR18].

e.g., a gas sensor typically needs to be compensated for temperature or humidity. If power supply and energy constraints are neglected, this complexity is manageable with today's approaches: generating a schedule for all envisaged tasks and performing these at a constant rate is sufficient. However, if the number of sensor nodes steadily grows, installation costs for wiring or maintenance cost and effort for battery-supplied nodes increase enormously.

As detailed in Section 2.3, many classes of energy-driven devices evolved: continuously operating or even intermittently-powered. Recent work, detailed in Section 3.2 emphasizes to include time constraints between tasks in scheduling for sensor networks. Although this approach concentrates on intermittently-powered devices, it offers high potential—also for ENO systems. Especially more complex sensors, as gas sensors, need up-to-date information of their surroundings, e.g., the temperature or humidity, before a meaningful sensor reading can be obtained. Thus, sampling a gas sensor without querying related sensors in-time wastes energy.

From the presented literature in Section 2.3 it can be concluded that future sensor nodes powered by ambient energy need a new scheduling approach: without consideration of time constraints and dependencies between tasks, energy is wasted and thus operation is inefficient. This chapter shows how dependencies, time and energy constraints can be used for task graph generation and scheduling.

5.1.2 Problem Analysis

In order to illustrate the shortcomings of existing approaches and to show the benefits of the presented solution, a practical example is depicted in Fig. 5.1. High aerial fine dust concentration in European cities is a major influencing factor of human health, c.f. Section 2.1. Responsible authorities for inner cities, residential areas but also larger production plants need to monitor their air quality finely grained to take countermeasures. Especially in port areas, where ships are the main contributor to fine dust pollution, flexible placement of fine dust sensors is desired. Interruption of production processes for sensor maintenance is time-consuming and cost-intensive; thus, supplying these sensors by ambient energy is a promising approach.

Low-cost and small sensors to measure fine dust particles, i.e., $PM_{2.5}$ and PM_{10}[1], are studied in [RID+11] and [BEMRB13]. However, solely supplying the sensor nodes with ambient energy requires careful adaption of the sensing activities, because of the high power consumption compared to the harvest. A cheap fine dust sensor as the Nova SDS011 draws 70 mA at 5 V, which depletes the energy stored in a 100 F supercapacitor in just 9 min. The relative humidity compensation requires a humidity and temperature sensor to be sampled before a

[1] $PM_{2.5}$ and PM_{10} are air particles with diameter less than 2.5 µm and 10 µm respectively.

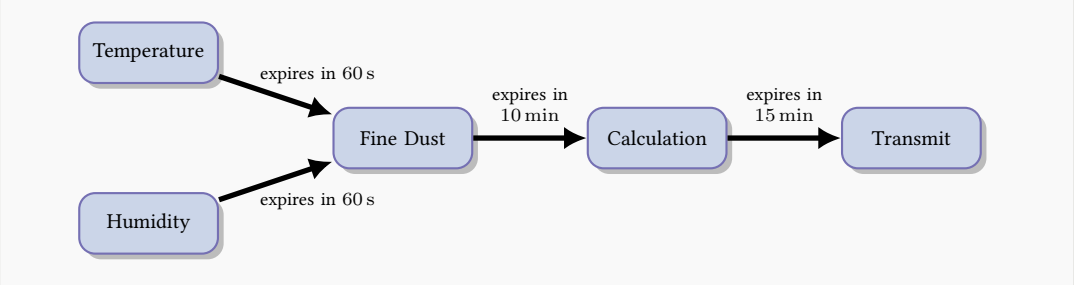


Figure 5.1: Process of fine dust sampling; limited energy storage prevents consecutive execution of all tasks at once.

valid fine dust measurement can be processed. Commercial fine dust sensors offer internal humidity compensation, but they are orders of magnitudes more expensive.

As highlighted in Section 2.3.2, it is not desirable to supply energy harvesting sensor nodes with large capacitors. Due to non-deterministic events, the voltage level of the capacitor might drop below the minimum voltage. Since recharge time is much longer with large capacitors, the system is unavailable for a longer time. Unfortunately, the need for smaller capacitors entails that energy-intensive tasks cannot be completed in one active phase. Entering a low-power (sleep) state between sampling sensors allows the capacitor to recharge but requires consideration of time constraints among tasks.

Figure 5.1 illustrates this behavior: temperature and humidity have to be sampled shortly before the fine dust sensor to ensure correlation between sensor values. However, once sensor values are gathered, calculation and transmission of these values can be delayed. Most applications require a certain amount of information per time interval, e.g., two times per hour. Thus, the margin for calculation and transmission allows for recharging of the capacitor, ensuring perpetual operation in the future.

5.1.3 Contributions

An important aspect of task scheduling is still missing in existing literature: tasks are not independent of each other and are strongly related in time. Thus, this approach is presented which ensures the following:

- Description of a program structure satisfying time constraints.
- Definition of the mathematical problem.
- Automated scheduling for dependent tasks.
- Evaluation of the performance and indications for solving the problem on low-power microcontrollers used in WSNs.

This approach allows complex sensors to operate with ambient energy, especially with size-restricted energy storage units. This paves the ground for running multipurpose sensors, with different sensors and radio interfaces, perpetually for fine-grained environmental monitoring.

5.1.4 Structure

The task model and problem definition are introduced in Section 5.2. Furthermore, the implementation is described in Section 5.3 and investigated in Section 5.4. Section 5.5 finishes with closing thoughts and directions for future research.

5.2 Task Model

As existing scheduling techniques lack the support for describing relations between tasks for WSNs, e.g., sensing, actuating, processing or sending, the underlying mathematical model of the approach is detailed. The presented model aims at task planning, so tasks are considered as non-preemptive and known in beforehand. First, the program structure of nodes are described as a DAG and the definitions of nodes and edges of the graph are introduced. Second, the time and third, the energy constraints are defined which are solved via ILP. Fourth, the problem definition and choice of the objective function are elaborated.

5.2.1 Program Flow

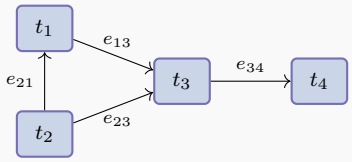
The time constraints between different tasks of a sensor node are also described by the Mayfly language, c.f. Section 3.2. The authors of Mayfly claim that tasks can be divided into three categories or can be described with three properties:

- `misd`: minimum inter-sample distance; the time after which new data is valuable;
- `expires`: the time after which a sensor value looses its meaning;
- `collect`: describes a distinct `number` of samples to be collected within a certain `timeframe`.

These definitions allow the programmer to describe the program flow similar to the one presented in Section 5.1.2. `misd` and `expires` describe the relation between consecutive activities, i.e., the time difference. Additionally, `collect` embraces numerous activities, which all have to be scheduled into a certain `timeframe`.

Beyond Mayfly

However, the schedule constructed from the Mayfly compiler only ensures a sequence of task execution. Since intermittently-powered devices may experience power outages between



(a) Task structure as DAG

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

(b) Adjacency matrix

Figure 5.2: Example for describing complex program structures by the use of a DAG.

two consecutive tasks, they cannot ensure that execution of every task maintains its meaning. If task information is outdated, is only checked *after* execution. Devices ensuring ENO have the benefit of deciding the meaningfulness of tasks *before* execution by accurate task planning. The basis for the presented task scheduling is the program flow described as DAG; see also Fig. 5.2.

The task graph $\mathcal{G} = (\mathcal{T}, \mathcal{E})$ is defined by the set of vertices, here tasks, $\mathcal{T} = \{t_1, t_2, ..., t_N\}$ with $N \in \mathbb{N}^+$ and edges $\mathcal{E} \subseteq \{e_{vw} \mid (v,w) \in \mathcal{T}^2 \text{ and } v \neq w\}$. Since $\mathcal{G}$ is a directed graph, each edge $e_{vw} \in \mathcal{E}$ is represented by a 2-tuple of starting vertex v and end vertex w. Tasks which are connected by edges are called *adjacent*; thus, the relationships in $\mathcal{G}$ can be described by an adjacency matrix $\mathbf{A}$. The entries of $\mathbf{A} = [a_{i,j}]$ are defined as

$$a_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{else.} \end{cases} \tag{5.1}$$

Tasks

To develop an energy-efficient schedule, more information about the tasks and the edges of the graph is needed. Typical microcontrollers, e.g., the ARM Cortex series or the low-cost ESP8266, have different hardware states, which vary in computing power, usable internal components and consequently power consumption. Additionally, the nodes power external peripherals, such as sensors and actuators, to interact with their surroundings. Thus, a task t_n is described by its power consumption $P_{\mathrm{t,n}}, \mathrm{n} \in \mathbb{N}^+$.

To query a sensor or to perform calculations, a certain time duration is required. For task t_n, this duration d_n can be obtained or updated at runtime by internal timers or external devices [DFPC08]. Together with the power consumption, this allows to compute the energy consumption of the node but also enables overlapping-free scheduling of tasks. To develop the schedule, all s_n (starting time of t_n) have to be calculated.

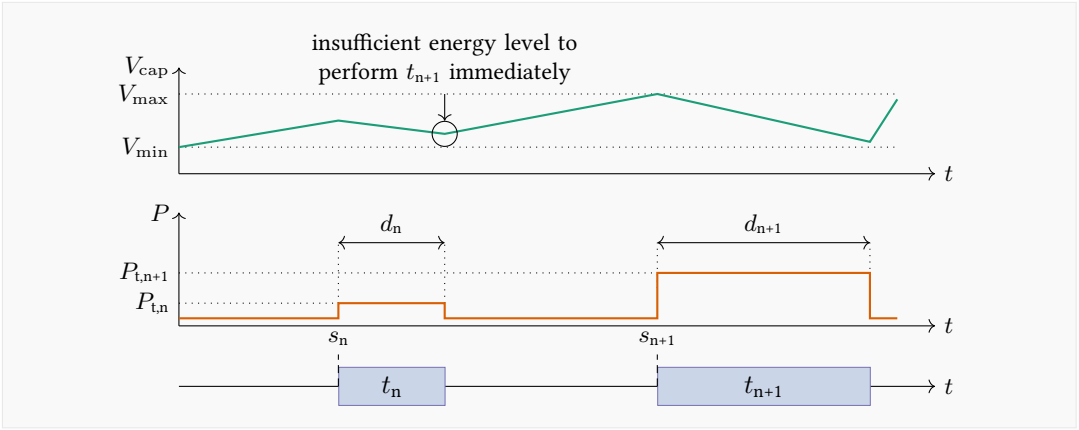


Figure 5.3: To prevent depletion of the capacitor, i.e., a voltage below V_{min}, sleeping before execution of t_{n+1} is needed.

Edges

Apart from its start vertex v and end vertex w, each edge comprises the constraints for the relation between v and w. Following the definitions of Mayfly, edges are described by a 4-tuple $e_{vw} = (\texttt{misd}, \texttt{expires}, \texttt{timeframe}, \texttt{number})$. This 4-tuple describes the constraints which have to be fulfilled by a valid schedule so that a following task w can work with up-to-date data. Not every task has to fulfill all parameters; e.g., if only one sensor value has to be sampled, no value for `timeframe` is defined.

5.2.2 Constraints

To compute a valid but also energy efficient schedule, numerous definitions are needed to satisfy the constraints. The constraints are also visualized in Fig. 5.3.

Time

In the following, k and l denote indices of their corresponding tasks in $\mathcal{T}$, with $k \neq l$ which are connected by an edge, so that $a_{k,l} = 1$. A mandatory requirement is that connected tasks of the program flow are consecutive and that their execution times do not overlap:

$$s_l \geq s_k + d_k. \tag{5.2}$$

Furthermore, the time constraints described by the connecting edges have to be fulfilled. Thus, the difference between starting times has to be:

$$\texttt{misd}_{kl} \leq s_l - s_k \leq \texttt{expires}_{kl}. \tag{5.3}$$

If one task requests a certain `number` of samples to be collected within `timeframe`, the scheduler ensures a greater time distance towards the following tasks, i.e.,

$$s_k + \texttt{timeframe}_k \leq s_l. \tag{5.4}$$

Energy

Sensor nodes, which are powered from ambient solar energy, mostly run energy estimation algorithms, e.g., `WCMA` and `Pro-Energy`, which give an approximation of the harvest within the next prediction horizon (usually 24 h in timeslots of 1 h). This is used to obtain an approximation of the maximum allowed energy consumption per timeslot within the prediction horizon, c.f. Section 3.1. Typically, this approximation prevents depletion of the energy storage, e.g., the voltage of a supercapacitor does not drop below a threshold. Within the prediction horizon, not all tasks can be executed subsequently without breaks and thus require sleep times in between to ensure recharging of the capacitor. If less energy is available, the time difference of the tasks are spread further to the boundaries of `expires`; if more energy is available, further to `misd`.

The energy consumption of the sensor node within one timeslot of the prediction horizon is given by the energy consumed by all executed tasks. If there is a budget B which can be taken from the energy storage without depletion within the next slot with length T_S of the prediction horizon, the energy consumption of the sensor node has to satisfy:

$$\sum_{n=1}^{N} \mathrm{rep}_\mathrm{n} \cdot d_\mathrm{n} \cdot P_\mathrm{t,n} + (T_\mathrm{S} - \sum_{n=1}^{N} \mathrm{rep}_\mathrm{n} \cdot d_\mathrm{n}) \cdot P_\mathrm{q} \leq B \cdot V_\mathrm{s} \cdot T_\mathrm{S}. \tag{5.5}$$

The budget B describes a constant current drawn by the sensor node at its supply voltage V_s. Here, rep_n denotes the number of executions of task t_n—if the budget is high, the task graph may be executed more than once. Additionally, P_q denotes the quiescent power consumption in sleep state. For simplicity, one sleep state is considered, but the use of different sleep states is possible in general.

5.2.3 Problem Definition and Objective Function

To solve the scheduling problem, the presented constraints are written together in standard canonical form: The definition in the canonical form is used:

$$\begin{aligned} \text{maximize} \quad & \mathbf{f}^T\mathbf{s} \\ \text{subject to} \quad & \mathbf{C}\mathbf{s} \leq \mathbf{c} \\ \text{and} \quad & \mathbf{s} \geq 0 \end{aligned} \tag{5.6}$$

The vector of starting times $\mathbf{s} = \{s_1, s_2, \ldots s_N\}$ needs to be determined by the scheduler. A feasible solution of the problem is only found if the presented constraints, c.f., Section 5.2.2, are met. All constraints can be rewritten so that the inequality $\mathbf{Cs} \leq \mathbf{c}$ is ensured. Hence, any solution of the problem develops a schedule that fulfill time constraints given by `misd`, `expires`, `timeframe`, and `number`. The optimal solution maximizes $\mathbf{f}^T\mathbf{s}$. The parameters `misd`, `expires`, `timeframe`, and `number` do not specify the objective function $\mathbf{f}$.

Commonly used objective functions for scheduling problems are chosen to minimize lateness, delay, or a serving time. Other objective functions embrace maximizing the usefulness of a schedule by assigning a utility or priority to each task. However, the goal of this approach is to ensure uninterrupted operation of the sensor node–as long as time requirements are met. Hence, the optimal solution is the solution that decreases the likelihood of downtimes due to empty energy storage. The amount of energy that can be consumed by the sensor node is given by an arbitrary energy management algorithm in the form of Eq. (5.5), i.e., a constant power consumption throughout a fixed time interval. Hence, the number of tasks that can be executed is out of control of the scheduler—only the point in time when the task is executed, can be controlled. As also shown in Section 2.4.2, the idea of a constant harvest throughout the timeslot simplifies the real world conditions. Harvest may arrive late in the slot, or at an early point in time. Most energy intake estimation algorithm, c.f., Section 2.4.3, however assume a uniform harvest during the timeslot. If this assumption is not true, e.g., nearly no harvest at the beginning of the timeslot, and the sensor node spend most of its energy at the beginning, the energy storage depletes drastically. This imposes a high risk of early energy storage depletion and hence risks perpetual operation of the sensor node. If perpetual operation is the primary goal, it may be beneficial to execute tasks as late as possible. This decreases the risk of depletion, since energy is mostly spend *after* it is harvested (within the timeslot). Therefore, the objective function is constant, i.e., $\mathbf{f} = \{1, 1, \ldots, 1\}$, which favors a solution that arranges tasks at the end of the timeslot. Arguably, other choices for $\mathbf{f}$ are possible. Section 6.3.3 discusses this choice in more detail.

Microcontrollers offer a plethora of timers of different resolutions, typically limited by the clock frequency of the MCU. Often, the time resolution is reduced to milliseconds. Hence, describing the starting time as integers has practical relevancy since the determined solution can be directly used as timer inputs. However, ILPs are generally NP-complete. However, modern solvers, e.g., MATLAB's `intlinprog`, automatically relax the problem first, i.e., converting it to an LP, and hence calculate solutions in an acceptable timeframe. If the scheduling problem has to be computed without access to modern libraries, however, than the ILP should be transformed to an LP first, e.g., allowing the starting time to be real numbers and round them after an optimal solution is found. More details on the implementation are presented in the following.

5.3 Implementation

To assess the performance of the approach, the problem definition, i.e., the constraints as defined in Section 5.2.2 and Section 5.2.3, are implemented and fed into a MATLAB ILP solver. For the problem definition, only two inputs are needed: the task graph and the budget of the timeslot for schedule generation. The goal is to find the number of entries in $\mathbf{s}$ as well as matrix $\mathbf{C}$ and its corresponding vector $\mathbf{c}$ to solve Eq. (5.6). To obtain the number of entries in $\mathbf{s}$, tasks are incrementally added to the list of tasks to be executed within this timeslot as long as the budget is not exceeded. Once the number of starting times is known, constraints can be added incrementally. First, it is checked if the task has to be executed at the current starting time. Second, the adjacency matrix $\mathbf{A}$ is checked for neighbors of the tasks of the current starting time. Neighbors can be predecessors or successors in the task graph—for both of them, the constraints as defined in Eqs. (5.2) to (5.4) are added to $\mathbf{C}$ and $\mathbf{c}$. If tasks are executed more than once, a constraint is added as well to ensure the first task of the next repetition of the task graph is executed after the last task of the current execution. Last, it is ensured that starting times are upper bounded by the length of a timeslot.

To speed up computation and reduce memory footprint, the sparse matrix representation for $\mathbf{C}$ is used. The function `intlinprog` from the optimization toolbox of MATLAB is used for solving. As `intlinprog` uses several pre-processing techniques, i.e., reducing the size of the problem and solve a relaxed (non-integer) problem first, the time to solve the problem only gives a hint on complexity-influencing factors, not on the execution time itself. Still, it can be shown whether the approach generates feasible schedules. Additionally, influencing factors can be highlighted.

5.4 Results

First, generation of an energy-aware schedule generated for the example of Section 5.1.2 is shown. Second, the performance of this approach is analyzed and the influence of the number of tasks in the graph and an increasing budget on execution time is investigated. Third, implications for microcontroller implementation are given. For the simulations in MATLAB an Intel Core i7-6700 with 16 GB of RAM is used. Execution times are measured by the provided `cputime` function.

5.4.1 Example

The task graph shown in Fig. 5.1 is used as input to generate a schedule. Additionally, real-world budget traces obtained from a long-term experiment with a solar energy har-

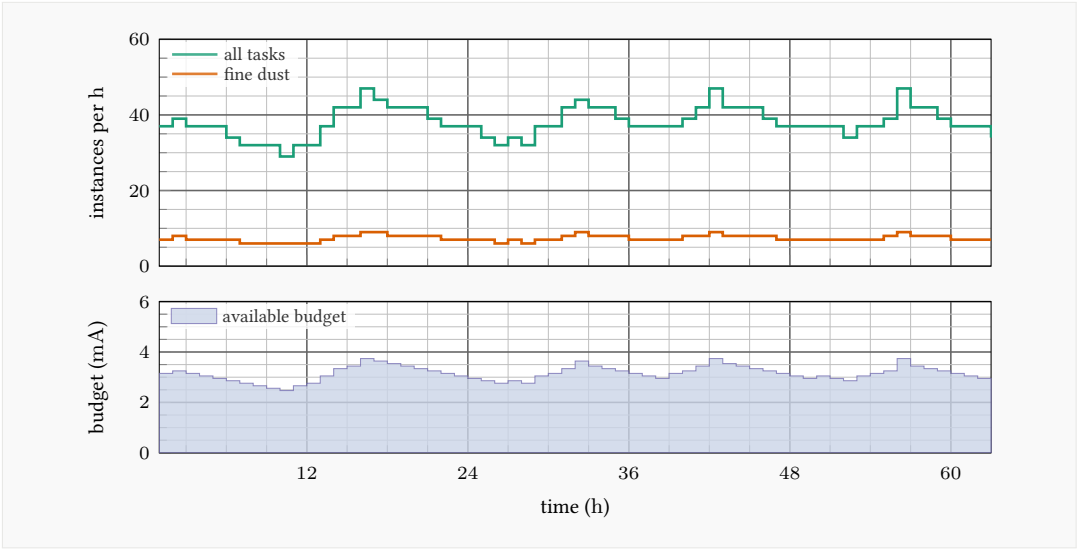


Figure 5.4: Energy-adapting task scheduling of the fine dust example; number of instances per h varies upon budget; budget trace obtained from real-world tests.

vester [HHR17] are fed into the simulation. For simplicity, only the first 64 h are depicted in Fig. 5.4.

Depending on the budget, it can be seen that instances per hour of the fine dust sampling task ranges between 6 and 9. The number of instances of all tasks shows a similar behavior, ranging between 29 and 47. This shows that even more complex program structures defined by a task graph and with time constraints can be scheduled being energy-aware simultaneously.

5.4.2 Performance

The main metric for comparing the performance is the execution time, i.e., the time to define and solve the problem. Although energy-aware scheduling is necessary for uninterrupted operation, it is still only a tool for performing the designated task. Consequently, the energy spent for calculating a schedule imposes an energy overhead and thus the time spent for scheduling should be as short as possible. Figure 5.5 depicts how the number of tasks in the task graph influences the execution time. Fully connected task graphs are randomly created with fixed number of tasks, varying number of edges and a constant budget. For each number of tasks, 500 different task graphs are generated.

Although the complexity of the task graph increases due to increased number of tasks and edges, the execution time remains relatively constant. Due to the fixed budget, an increasing number of tasks does not increase the number of variables—the number of variables is equal if 15 tasks are executed four times each or if 20 tasks are executed three times each.

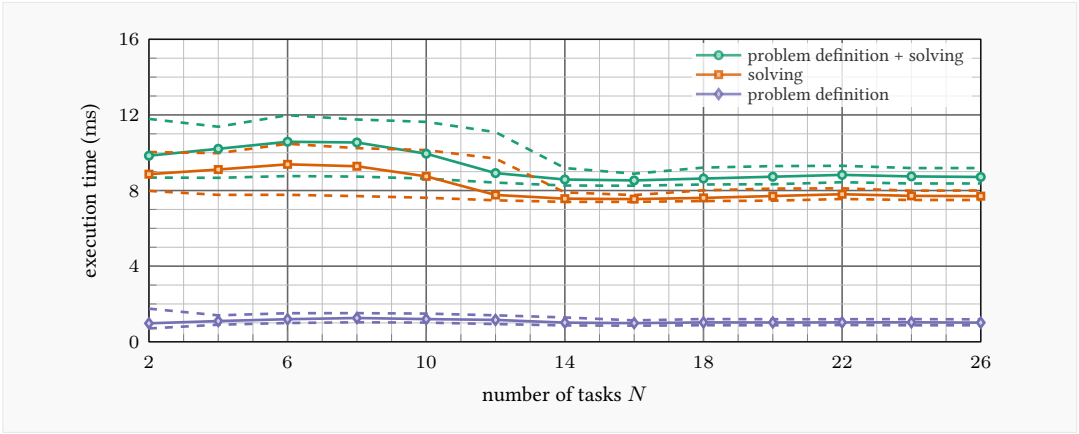


Figure 5.5: Median of execution times influenced by the number of tasks in task graph; dashed lines indicate upper and lower quartiles respectively; constant budget of 5 mA; since budget stays constant, number of instances in total and thus complexity of the problem remains relatively steady.

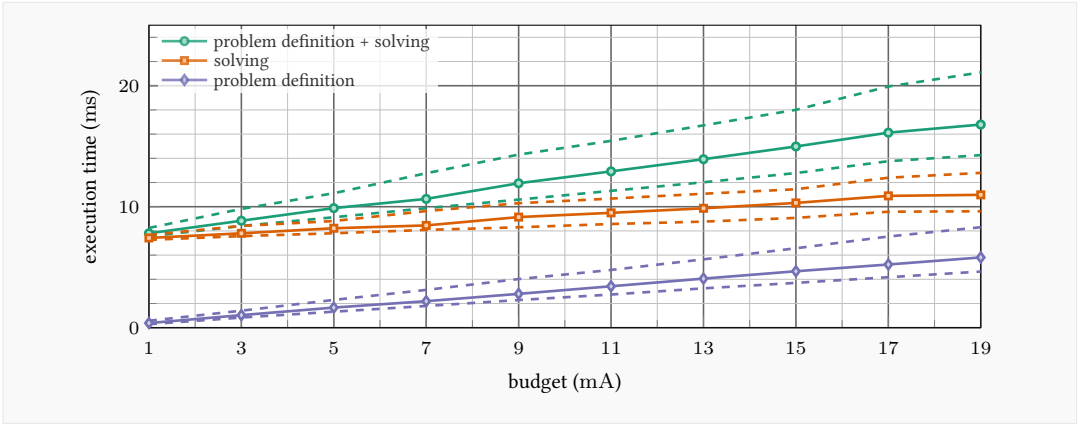


Figure 5.6: Median of execution times influenced by increasing budget. Dashed lines indicate upper and lower quartiles respectively; increased complexity influences both problem definition and solving; however, increase in execution time primarily influenced by increasing time for problem definition.

However, if the budget increases, also the execution time because more variables have to be determined. Figure 5.6 shows how the increased budget influences the performance. Again, the task graphs are randomly created, and a schedule is computed with an increasing budget each. This process is repeated 500 times.

The execution time strongly increases with budget, e.g., increasing the budget from 3 – 9 mA demands 30% more execution time. Although this seems noteworthy, the increase is good-natured: an increasing budget means good energy-harvesting conditions and thus tolerates a larger overhead for scheduling purposes. However, it is important to mind this aspect for microcontroller implementations.

5.4.3 Discussion

The investigations show that a high-performance PC can generate a schedule for 10 tasks and 20 edges for a typical budget of 3 mA in 8.8 ms. However, the computational power of microcontrollers is orders of magnitudes lower. To have a rough estimation on the energy overhead, the DMIPS[2] of a common microcontroller, the ARM Cortex M0+ (c.f. Section 2.2.1) and the simulation PC are compared. The Intel Core i7 measures 11.28 DMIPS/MHz, while a Cortex M0+ offers just 0.95 DMIPS/MHz. Including the clock frequencies in the calculation, the approximated time for executing the schedule for one timeslot on a Cortex M0+ is roughly 11 s. While this sounds relatively long, the energy overhead is still only 0.7%, assuming a current consumption of 6.85 mA, a budget of 3 mA and a timeslot length of 1 h. Surely, these numbers will differ in practice, but they indicate that the approach is feasible and leaves room for improvement.

5.5 Conclusion

The increasing capabilities of WSNs offer a plethora of new applications but also pose new demands on energy-aware task scheduling. It was shown that using task graphs, which describe dependencies between tasks, and defining time and energy constraints, allows for calculation of schedules for sensor nodes. However, the measured execution times to solve the problem indicate that it is rather energy-costly for online calculation; hence a heuristic for faster execution times is beneficial.

[2] DMIPS = Dhrystone Million Instruction per Seconds; common performance measure generated by the Dhrystone benchmark.

Chapter 6

Heuristic for Time- and Energy-Aware Scheduling

As outlined in Chapter 5, task-based program structures offer an easy-to-use definition of program structures that helps to spread the usage of EH-WSNs. For online microcontroller usage, however, a heuristic is needed that solves the scheduling problem energy-efficiently.

6.1 Introduction

In the following, the scheduling problem for task-based program structures is motivated and detailed. The core contributions and structure of this chapter is presented afterwards.

6.1.1 Motivation

The still growing popularity of EH-WSNs is fostered by increasing computation power, new sensing capabilities, and sharing of WSNs by multiple parties. As highlighted in Section 2.1, this opens a plethora of new and more complex applications, e.g., neural networks, air quality monitoring or multi-tenant sensor platforms.

As shown in a concept in the field of intermittently-powered devices, programs on sensor nodes can be split into atomic subparts that serve a particular purpose and consume and produce data from and for other subparts, respectively. A possibly complex structure of data dependencies (flows) and time constraints evolves. These subparts are referred to as *tasks* in the remainder of this chapter.

Publication

Parts of this chapter are published in [HR20].

This concept can be transferred to ENO systems, and this approach is the first that makes this step. Unfortunately, WSN applications are frequently implemented as monolithic programs without isolated, well-defined tasks and without an appropriate, versatile scheduler. All dependencies are buried in the source code. Energy-aware scheduling—i.e., adapting operation to ambient energy conditions—boils typically down to changing the execution rate of tasks or (component) duty cycles through relatively simple or application-specific schedulers (c.f. Section 2.3.1), which make use of energy harvest estimation (c.f. Section 2.4). For WSN applications, this approach implies hard-coded and difficult to change behavior; and profound knowledge of the interplay between energy consumption of the sensor node, task scheduling and available ambient energy is required.

This limits the circle of potential developers and companies willing to operate more complex sensors supplied with ambient energy. Not only program development is complex, but also advanced knowledge about the reliability of the system is hard to obtain. As detailed in Section 2.3 many factors influence the operational characteristics of EH-WSNs: energy storage size, energy harvest and its pattern, power consumption of used sensors, and also the program structure. To estimate operational characteristics, developers and researchers mostly opt for a very specific solution tailored to their application's needs. Still, miscalculation or misjudged interplay often result in high downtimes—depleted energy resources leaving the node dead for some time—or over-provisioned energy storage.

The Mayfly language, c.f. Section 3.2, shows how WSN programs may look, if they split functionality from meta information, i.e., time constraints between atomic tasks. As also detailed in Section 5.1.2, more complex sensors, such as gas sensors, require up-to-date compensation information of their surroundings, e.g., the temperature or humidity, before a meaningful sensor reading can be obtained.

6.1.2 Problem Analysis

Typically, task scheduling or task planning algorithms for energy-harvesting sensor nodes are rate-oriented. A set of tasks is grouped together to a program which is assumed to run uninterruptedly. Based on the available energy, this program is adapted in its rate, i.e., the number of executions per time period. However, the assumption that the complete program can be executed without interruption might not hold if (a) the complete program is energy-hungry, (b) the energy buffer is small or (c) the harvest is low. Emerging applications such as fine-dust monitoring interfere with (a); small capacitors for desirable short outage-times add to (b) while newly used sources of environmental energy (c.f. Section 2.2.2) lead to (c). This deficiency is illustrated in Fig. 6.1. Three tasks form a simple program in which tasks 1 and 2 have to be executed before task 3. The tasks differ in duration, i.e., width of the boxes on the timeline, and power consumption P, i.e., height of the boxes. A program executing all tasks

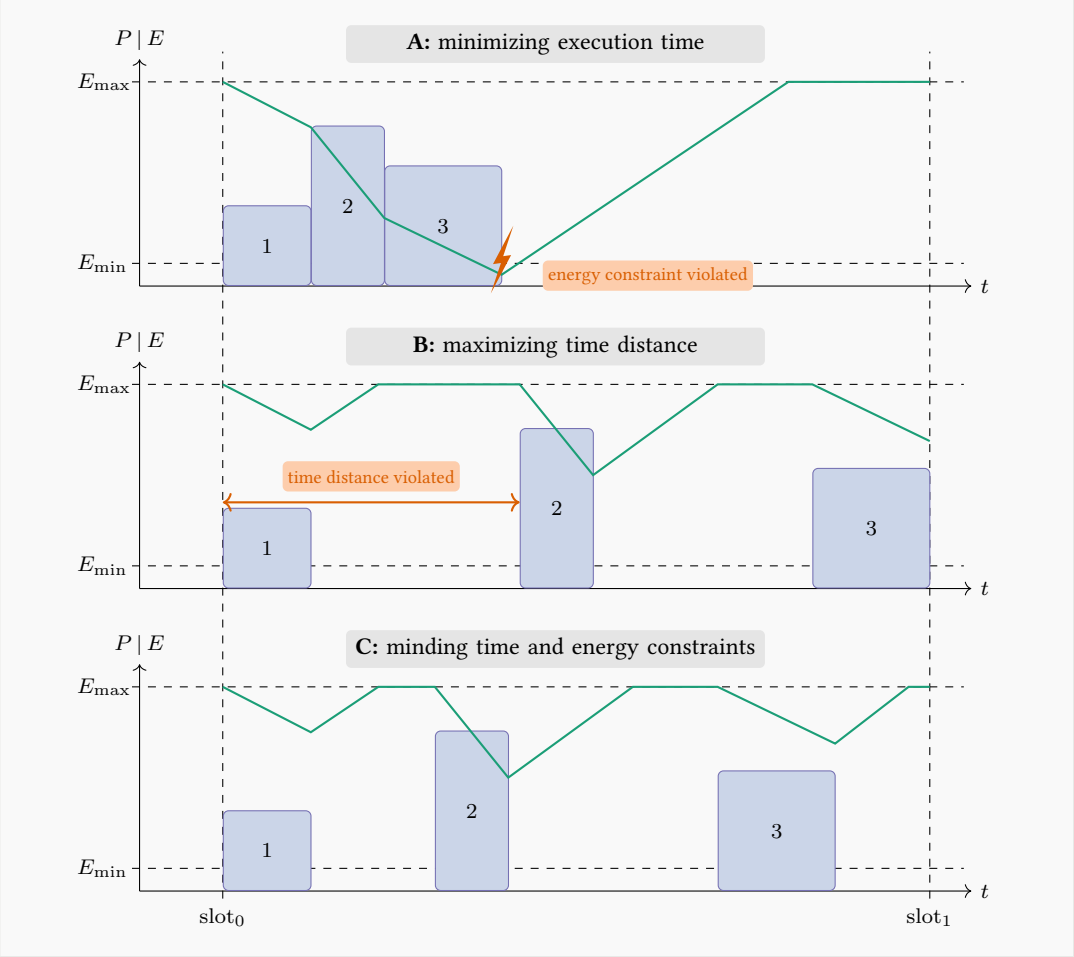


Figure 6.1: Scheduling approaches for three dependent tasks: **A** risks fast depletion of the energy buffer while **B** violates time constraints imposed by physics; only combining both in **C** satisfies time *and* energy constraints.

consecutively (approach **A**) risks depletion of the energy buffer. As a result, the node turns off and, dependent on the size of the energy buffer, needs time to recharge. Intuitively, the time distance between tasks can be increased to allow the energy buffer to recharge. However, sensing tasks in sensor systems rely on physical phenomena, e.g., temperature and humidity compensation for particulate matter readings, and thus are related in time. Consequently, the recharge time is limited by a time distance given by the underlying characteristics of the phenomenon. If the recharge time is maximized, as shown in **B**, the risk for violation of the time constraints increases. The potential consequences are reduced value for the application due to a distorted sensor reading, e.g., failed compensation, or wasted energy, since the task has to be executed again. Consequently, a new approach **C** is needed minding both time and energy constraints.

6.1.3 Contributions

This chapter argues that a solution is missing for (a) scheduling programs broken into atomic tasks depending on each other with time constraints, and (b) to pre-evaluate the performance of the interplay between program and EH-WSN hardware. Finding a solution for (a) is key to schedule tasks flexibly without energy-wastage. Missing the time constraints inherited from the physical quantity leads to lost meaning of sensor values and imposes an additional sensor reading or decreases the value for the application. A tool for (b) helps to spread the usage of EH-WSNs, since it allows programmers to assess—before deployment—if the existing hardware, e.g., energy buffer, is sufficient for the desired sensor accuracy and how large the energy buffer needs to be to prevent long downtimes at a range of sampling rates.

Hence, this chapter introduces a new approach which fulfills both (a) and (b) by offering a scheduler using graph-based program structures for determining the activity of an EH-WSN node. Furthermore, it is detailed how the simulation toolkit, presented in Section 4.3, enables researchers to pre-evaluate downtimes and achievable task cycles to dimension their energy buffer and select a scheduling scheme for their application. In particular, this chapter

- adopts the concept from Mayfly for ENO sensing systems;
- introduces and discusses light-weight scheduling strategies for graph-based programs of recurring task sets;
- analyzes the performance of scheduling strategies based on extensive (replay) simulation of a real-world sensor node and real-world energy harvesting traces;
- shows coherence of simulation and real-world behavior; and
- highlights the usage of the simulation toolkit to assess the validity of task design and to help to determine sufficient energy supplies before deployment.

6.1.4 Structure

The rest of this chapter is organized as follows: Section 6.2 introduces the used program models and highlights given time- and energy constraints. In Section 6.3, Kahn's Algorithm (KA) is introduced which allows topological sorting of graphs efficiently; additionally the adaption is detailed in Section 6.3.1 in which time-and energy constraints are incorporated. Furthermore, the approach is parameterized with the scheduling `STRATEGY`, c.f. Section 6.3.2, and the scheduling `BALANCE`, c.f. Section 6.3.3. Benchmark applications, simulation model accuracy and parameters, as well as metrics are introduced in Section 6.4. Furthermore, the coherence between real-world harvester and simulation toolkit is shown in Section 6.4.2. An extensive simulation study is carried out in Section 6.5 which highlights the influence of capacitor

size and timeslot lengths on downtimes and activity; furthermore, a real-world experiment is provided in Section 6.5.5. Section 6.6 discusses the simulation performance and shows the prototype microcontroller implementation; additionally possible limitations and learned key messages are provided. This chapter finishes with a summary and indications for future research Section 6.7.

6.2 Model Description

The support for describing relations between tasks for WSNs, e.g., sensing, actuating, processing or sending, is missing in commonly used scheduling techniques, c.f. Section 2.3.1. In consequence, atomic subtasks are typically hard coded in the program, either chained or realized via state machines. This decreases not only the flexibility of adapting the program structure, but it also increases the entry barrier for application developers in the field of EH-WSNs. Breaking program structures down into atomic tasks thus bears the potential for a new and user-friendly approach for composing programs as side effect. If pre-defined program elements can be reused, even non-experts can develop complex programs. Thus, task scheduling with graphs and atomic tasks as input helps the spreading of EH-WSNs.

6.2.1 Time Awareness

Time constraints between tasks of a sensor node are described by the Mayfly language and detailed in Chapter 5. However, the Mayfly language is designed for intermittently-powered devices and not for ENO systems. Still, above parameters help to describe the inherited physical phenomena, logical program flow, and timing and thus help to develop a scheduling scheme with high benefit for the application. For details on the mathematical model, the reader is referred to Chapter 5.

For energy-aware task scheduling, precise knowledge about energy consumption of a sensor node is required. Normally, a task involves different peripherals, e.g., querying a sensor consumes energy for powering both sensor and microcontroller to read the sensor and process data. The corresponding consumption values could be preconfigured—e.g., through lab measurements or data sheet information—or even be determined at runtime. ENO systems typically employ converters to stabilize the energy storage voltage. As long as the increased operating voltage range outplays losses due to conversion efficiency, boost converters are the preferred choice. To support adapting changes at runtime, tasks are modeled by their power consumption $P_{\mathrm{t,n}}$ and their execution time or duration d_{n}. Here, constant and known execution times are targeted. However, the approach enables task scheduling for varying execution times—e.g., communication tasks rely on other nodes and change with channel utilization. In this case, initial values of d_{n} have to be set at the beginning but can be adapted at runtime

by internal timers or external devices. Finally, a schedule defines all starting times s_n of tasks t_n within a scheduling horizon. If a complete series of tasks, i.e., the complete DAG, is scheduled, this series is called a *cycle*. A cycle is repeated, if and as long as sufficient energy is available (cf. Section 6.3.3).

6.2.2 Energy Awareness

Keeping track of energy harvest to estimate its future course is key to bridge timespans of low energy harvest. As long as a pattern exists—e.g., machine working hours for vibration harvesting—a model of future harvest can be derived. Here, the primary source is solar energy, since it is commonly available and literature knows many well-established estimation schemes, e.g., `WCMA` and `Pro-Energy`. However, the scheduling approach is compatible with other sources.

Harvest estimation algorithms produce a forecast for a prediction horizon (usually 24 h in timeslots of 30 min to 4 h for the diurnal cycle of solar energy). This, in turn, can be used to obtain—e.g., as presented in Section 3.1—an upper bound of energy consumption per timeslot, the *energy budget*, within the prediction horizon, which ensures that the energy buffer of the node is not depleted or even retains backup energy to cater for inaccurate harvest estimation.

The cumulated energy consumption of all tasks executed within the prediction horizon thus has to be balanced against the budget B, where the latter is valid within the next time slot of the prediction horizon. Again, Eq. (5.5) has to be enforced. If the budget is large, multiple cycles of the task graph can be executed.

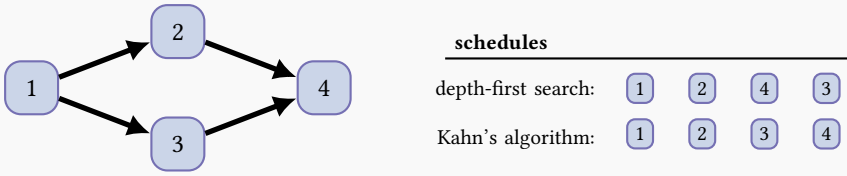
6.3 Algorithm Design

For task scheduling of graphs, three aspects must be considered: (a) ordering of tasks, (b) time distance between tasks, and (c) balancing consumption within a time slot to meet the budget.

By presenting an adapted topological sort algorithm, (a) is satisfied with low complexity. Since the time constraints between atomic tasks are the prior concern for valid readings, (b) is integrated by introducing different scheduling strategies. And, to highlight the influence of distribution of spent energy, the requirements of (c) are fulfilled with different balancing schemes.

6.3.1 Adaption of Kahn's Algorithm

Despite adequate time differences between the tasks for scheduling, the predominant question is how to choose an execution order of tasks. The algorithm has to ensure that a task is only scheduled if all predecessors in the graph have been scheduled already. In graph theory, this problem is known as *topological sorting*. If the graph structure is decreased into a one



(a) Example program (b) Possible schedules

Figure 6.2: In contrast to DFS, Kahn's Algorithm enforces that predecessors have already been scheduled. Hence, only Kahn's Algorithm ensures correct ordering of tasks. Note that task 2 and 3 can be exchanged, depending on their assigned priority. Here, it is assumed that lower numbers have higher priority.

Algorithm 1 Kahn's algorithm

```
1: S ← empty list for sorted elements
2: U ← set of nodes without incoming edge
3: while U not empty do
4:     remove a node n from U
5:     add n to tail of S
6:     for each succeeding node m and edge e_{n,m} do
7:         remove e_{n,m}
8:         if m has no incoming edges then
9:             insert m into U
```

dimensional structure, e.g., a list, topological sorting ensures that parent nodes of a child node appear prior in that list. Once all nodes are added to that list, a valid sorting order exists and task starting times can be determined.

Topological sorting of DAGs can be accomplished with low complexity via Depth-first Search (DFS) or Kahn's Algorithm (KA) [Kah62]. However, DFS is only suitable for graphs with no diversions, or independent paths in the program. Figure 6.2 illustrates a graph with two dependent tasks, which need to be completed before the final task. A DFS schedules the first path and generates starting times, even for the last task—although this starting time might violate constraints of the second path. While this behavior might be wanted in some cases, e.g., finishing single paths as soon as possible, the solution is chosen which integrates the whole task graph instead of focusing on sub-parts. The pseudocode in Algorithm 1 describes the behavior of KA. By design, KA only adds a task to the schedule, if all its predecessors have been added to the schedule S. The complexity of KA is $\mathcal{O}(|\mathcal{T}| + |\mathcal{E}|)$. The sorting is not unique, especially if the initial list U has more than one entry, but generates a list which can be used for scheduling. However, two problems hinder the direct use for the presented task model: Edges are deleted when tasks are added to the set of sorted elements, and time and energy constraint are not taken into account. Thus, a modified version of KA is developed

Algorithm 2 Adapted Kahn's algorithm

```
1:  S ← empty list for sorted elements
2:  U ← set of nodes without incoming edge
3:  while U not empty do
4:      pick n from U
5:      add n to tail of S
6:      determine s_n based on STRATEGY & BALANCE
7:      for each succeeding node m and edge e_{n,m} do
8:          decrement in-degree but keep information of e_{n,m}
9:          if m has no other incoming edges then
10:             insert m into U
11: while energy left && time left do
12:     add cycle of task graph to schedule
```

which is shown in Algorithm 2. The original KA deletes an edge $e_{n,m}$ after use, so that time constraints are lost. This is solved by counting the number of incoming edges for each node, called in-degree, which adds another $\mathcal{O}(|\mathcal{E}|)$ to complexity. Instead of deleting the edges, the in-degree of the edge's end task is only decremented when the algorithm adds the start task to the schedule. If the in-degree reaches zero, node m is added to U and can be scheduled in the next iteration. Furthermore, if a complete cycle is generated—i.e., all tasks are at least present once in the schedule S, but there is still energy and time left—a complete new cycle can be added to S. This can be done with low complexity, since the task graph is only sorted once—other cycles can be simply added to the overall schedule. For Algorithm 2, Line 7 shows a central behavior: determination of the starting time of the newly added task. Two aspects are of utmost importance: choosing the starting time for a task *within the cycle* and choosing the starting time of the cycle *within the timeslot*. As explained previously, the former is adapted by a `STRATEGY`, the latter by a `BALANCE`. Both are subsequently discussed in detail.

6.3.2 Scheduling Strategies

As shown in Fig. 6.1, determining the correct sleep time between tasks is overly important, especially if the energy budget is restricted and tasks are energy-hungry. However, sleep times are highly application-specific and their importance differs. As an example, Delay-tolerant Networks (DTNs) tolerate large gaps between data gathering and transmitting. Thus, long sleep times can be favored to increase longevity of the network. As another example, control loops usually pose high timing requirements on the participants in the loop. Hence, sleep times need to be balanced carefully to maintain reactivity of the system. Three different strategies are identified to determine the starting time within the adapted KA.

Greedy Strategy

In certain situations, e.g., if a researcher found that a high sampling rate is needed, it is desirable to achieve a fast utilization of the available energy. For these applications or situations, the `greedy` strategy is implemented. Thus, when checking the edges to the predecessors, the scheduler tries to execute the new task as fast as possible but not faster than the minimum-inter-sample distance of its predecessors allows. Consequently, the starting time is $s_\mathrm{n} = \max(\texttt{misd}_\mathcal{P})$ with $\mathcal{P}$ denoting the set of predecessors. In general, the last task in the schedule is not necessarily the predecessor of task t_n. Consequently, s_n may be smaller than the start time of the last task in the schedule. In this case, without recalculating the schedule, the new task t_n has to be scheduled directly after the last task in the schedule. This aspect is also highlighted in Section 6.6. Taking the maximum `misd` of all predecessors ensures that the requirements of the underlying physical phenomenon is still considered. On top of the increased risk of depletion, this strategy consumes energy in bursts since the complete budget is spent in a very short period. While this leads to high sampling rates, it also may decrease the reachability of the system in the rest of the prediction horizon: since all energy is spent in a very short timespan, the rest of the prediction horizon may be spent sleeping.

Lazy Strategy

In contrast, conservative application developers may want to spread their energy consumption over a long time span. This is especially important if avoiding power-outages is a primary concern, e.g., if the node only supports volatile storage solutions or writing to non-volatile storage, e.g., Flash, is energy-expensive. The `lazy` strategy supports this conservative approach by spreading the tasks in time. Particularly, the starting time of a task is $s_\mathrm{n} = \min(\texttt{expires}_\mathcal{P})$, which is the highest possible time distance between predecessor and successor without violating the deadlines of the other predecessors in $\mathcal{P}$. While the reachability of the system can be increased with this approach, it also inherits the risk of not spending available energy. Defining high values of `expires` only allows few repetitions of the cycle, although the budget allows this—the duration of the prediction horizon limits the number of repetitions.

Match Strategy

Especially for small capacitor size, which is favorable due to the short recharge time, consuming energy in bursts leads to sudden voltage drops of the capacitor. Additionally, energy estimation schemes (c.f. Section 2.4.3) provide an average budget for the prediction horizon.

Reducing the number of timeslots decreases the time spent for calculating a budget but inherits the risk of missing that the energy intake may be unevenly distributed in the estimation

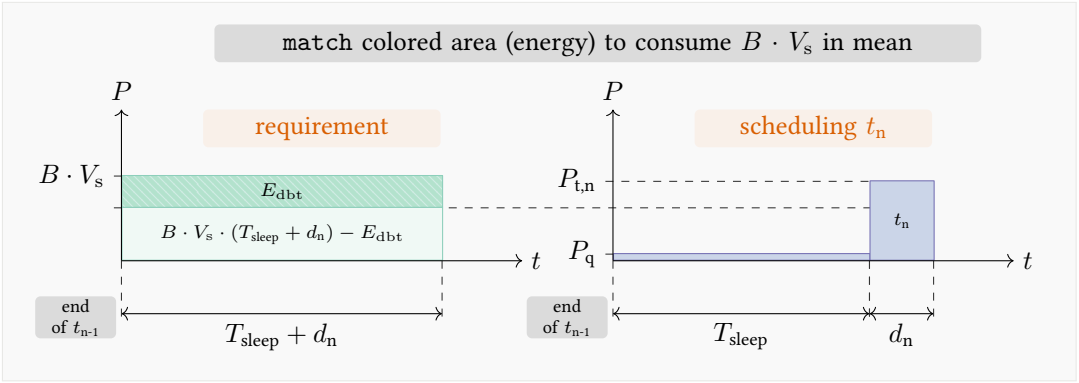


Figure 6.3: Energy debt E_{dbt} from previously scheduled task leads to adjusted sleep time T_{sleep}. In this example E_{dbt} is positive.

horizon (average over long time period). Thus, spending energy in bursts violates the average current consumption in short term, although it is fulfilled for the remaining timeslot. Spending all energy late in the time slot decreases the risk of depletion but imposes a long time period at the beginning of the slot where the node is unreachable.

The `match` strategy, also highlighted in Fig. 6.3, tackles this issue as follows. Before adding new task t_n to the schedule, the scheduler determines the sleep time T_{sleep} required before execution of t_n to match the mean current consumption given by the budget:

$$\begin{aligned} V_s \cdot B \cdot (T_{sleep} + d_n) - E_{dbt} &= P_q \cdot T_{sleep} + P_{t,n} \cdot d_n \\ \rightarrow T_{sleep} &= \frac{E_{dbt} + d_n \cdot (P_{t,n} - V_s \cdot B)}{V_s \cdot B - P_q}. \end{aligned} \tag{6.1}$$

Here, E_{dbt} is the difference of the energy budget and the actual energy consumption of the current schedule (before insertion of t_n). Next, the scheduler derives the earliest starting time, $\max(\texttt{misd}_{\mathcal{P}})$, and latest starting time, $\min(\texttt{expires}_{\mathcal{P}})$, to fulfill the constraints of all predecessors of t_n. If T_{sleep} is within these boundaries, it is accepted by the scheduler and t_n is scheduled T_{sleep} after the end of the current schedule. In this case, E_{dbt} will be zero for the next task. Otherwise, T_{sleep} is discarded and the scheduler adopts either the earliest starting time (if T_{sleep} is less) or the latest starting time. In both cases, E_{dbt} becomes non-zero, because the scheduler is forced to schedule t_n later or earlier than would be ideal. The `match` strategy tracks the energy debt E_{dbt}, and if greater than zero saves energy for the remainder of the timeslot. Regardless of the employed strategy, the scheduler only schedules a single task cycle and then repeats this cycle. Schemes for repeating or distributing cycles in the horizon are introduced next.

6.3.3 Consumption Balance

As highlighted in Section 2.4, harvest estimation methods based on timeslots have, apart from estimation errors, the drawback of averaging energy harvest in the prediction horizon. Since harvest may fluctuate heavily or exhibit a slope within a timeslot, the harvest forecast may deviate considerably from reality. Splitting the diurnal cycle into timeslots hence imposes the challenge of carefully thinking about when to spend the energy budget. This aspect is considered by presenting different `BALANCE` schemes which can be adapted according to the application's need.

To satisfy the requirements of applications that need their information as fast as possible, the `front` scheme is suggested. All cycles are grouped at the front, which yields early execution of all task cycles yet entails an energy deficit—i.e., average consumption exceeds the budget—early in the timeslot, which is expected to equal out until the end of the timeslot. Consequently, this scheme works better with accurate harvest estimation, e.g., vibration harvesting on production machines, and suffers from error-prone estimation in varying harvesting conditions, e.g., solar radiation.

To prevent an energy deficit, the `end` scheme is recommended which groups all repetitions of the cycle at the end of the timeslot. In direct-charging solar-panel circuits, this scheme also has the benefit that the capacitor has a higher voltage throughout the timeslot which drives the solar panel to a better power point, i.e., the power generated at the solar panel is higher.

Both balancing schemes `front` and `end` have the drawback of a long inactive phase: cycles are grouped together and the node is either active at the front or at the end of the timeslot. Applications needing information regularly suffer in these schemes. The `stretch` scheme solves this issue by dividing the timeslot into subparts with equal length determined by the repetitions of cycles. Tasks are spread nearly uniformly distributed along the timeslot.

6.4 Simulation Setup and Model

In the following, the simulation setup and benchmarking applications are used to assess the presented scheduling strategies, and balancing schemes.

6.4.1 Benchmark Application

Based on a planned real-world fine-dust application that is described subsequently, two benchmark applications are devised that are used for the following evaluation. During early tests, it was realized that random task sets allow for limited insight only.

task	device	repetitions	$P_{t,n}$ (µW)	d_n (ms)
temperature	MCP9808 + μC	1	112 300	150
humidity	Sensirion SHT30 + μC	1	114 200	50
fine dust	Nova SDS011 + μC	1 / 5	492 600	5 000
calculation	Espressif ESP32 μC	1	112 200	500
transmit	Espressif ESP32 WiFi	1	357 000	3 000

Table 6.1: Benchmark application details.

Fine Dust Sensing Example

Low-cost and small sensors to measure fine dust particles are deployed by local authorities to monitor air quality, c.f. Section 5.1.2. The resulting program structure, already shown in Fig. 5.1, illustrates this behavior: temperature and humidity sensors are queried shortly before the fine dust sensor to ensure correlation between sensor values. However, once sensor values are gathered, calculation and transmission of these values can be delayed. Relevant power consumption numbers are obtained by manual measurements with a prototype built at the institute. However, in-field use is hindered by the lack of adequate scheduling mechanisms. Additionally, this example is chosen since the energy demand is high, implying that a task cycle can be executed infrequently and in varying intervals. This adds additional corner cases for the scheduling strategies and balancing schemes.

Benchmark Applications

The first benchmark application (FDS = Fine Dust Sampling) is the fine-dust example. The second one (FDS-5) is a variant of FDS that takes five fine dust samples to create a longer and more power-hungry program. Table 6.1 provides details on task duration and consumption. Consumption values of sensor tasks always include power consumption of the ESP32 μC of the prototype (cf. Section 4.2). Consumption in sleep mode (when no task is executed) is 86 µW. Durations have been determined empirically. Time constraints are equivalent to those in the example. For the five-time fine-dust sampling, `misd` = 15 s and `expires` = 5 min is used. To increase scheduling flexibility, `misd` = 1 s for the temperature and humidity sensors are added.

Execution of a task cycle is 7.5 s to 8.1 min for the original example (FDS) and 1.1 – 28.1 min for the extended one (FDS-5). Energy consumption (excluding sleep consumption) per cycle is 2.3 Ws and 9.5 Ws, respectively.

6.4.2 Simulation Environment

The original implementation of the framework presented in Section 4.3 is modified to support scheduling policies as follows. When a new budget has been determined, a new schedule is calculated based on the chosen scheduling strategy, balancing scheme, and the budget (average consumption) for the next time slot. This means that schedules are revised S times per day. Consumption of the node is simulated based on this schedule and the provided power consumption and duration of its tasks. Here, it is assumed that both parameters are known. This knowledge may have been determined in the laboratory or has been measured by the node at runtime. If the node is not executing a task, it is sleeping with low consumption. In the event of $V_{\text{cap}} < V_{\text{min}}$ at the beginning of a new time slot, no tasks are scheduled within this slot. The simulator also takes care of a depleted energy buffer; i.e., when the actual capacitor voltage V_{cap} falls below V_{min}. This condition is evaluated at the beginning and the end of every task, but at least every 5 min. According to the real hardware, the node restarts only when the supercapacitor voltage rises above 1.6 V, which is checked every 5 min simulation time.

Model Accuracy

To verify that simulation follows real-world behavior closely, a 7-day experiment was run with an artificial harvest trace in the light box presented in Section 4.4. The fine-dust benchmark application with the scheduling approach using `match` and `stretch` was used. To avoid accidental depletion of the capacitor due to WiFi instability, connection attempts to the WiFi network are limited to 5 s, which was used as task duration for the scheduler. The simulation framework detailed in Section 4.3 is used throughout all experiments. For realistic simulation, the average of actual connection times recorded during the experiment was used to simulate consumption. Results are portrayed in Fig. 6.4 and show that simulation and experiment are almost identical. Due to the fact that actual consumption is lower than expected (due to WiFi connection being shorter than 5 s on average), the budget rises throughout the night, because capacitor voltage is higher than expected. In order to show what would happen if consumption was known precisely, simulations are repeated accordingly and included in the figure (labeled *ideal simulation*).

6.4.3 Simulation Parameters

Simulation parameters are chosen to resemble the harvester presented in Section 4.2 with common supercapacitor sizes. Parameters of algorithms (not subject to particular evaluation in this work) are selected based on previous findings; e.g., the choice of maximum V_{cap} and `EWMA` filter coefficients. Note that the latter are used in a way that larger values favor historic

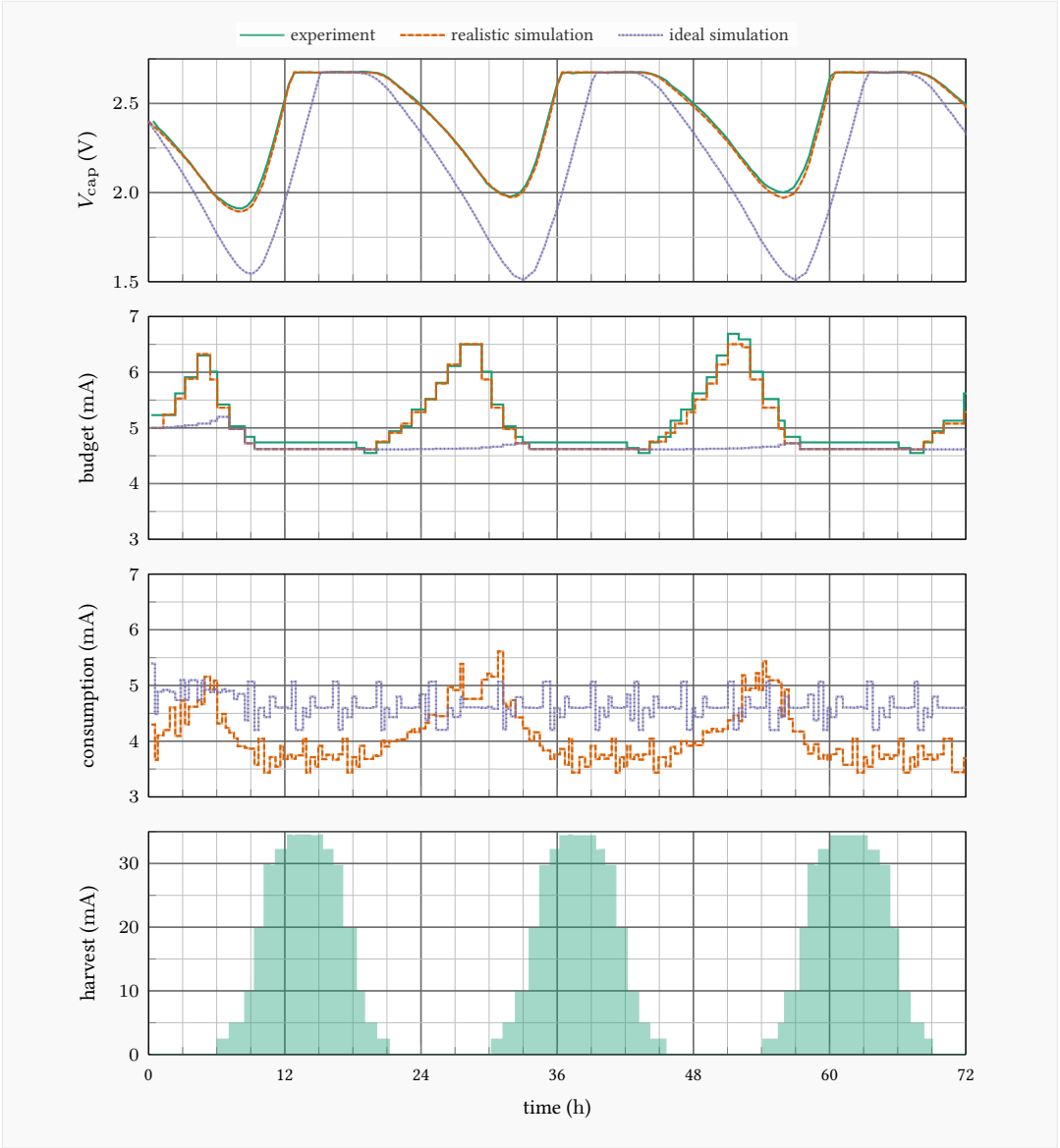


Figure 6.4: Comparison of simulated and real behavior in a controlled light box. *Realistic simulation* uses actual mean WiFi connection times (from the experiment) for consumption but not for budgeting (where a maximum time is used, before connecting is aborted); whereas *ideal simulation* uses real connection times for budgeting and consumption.

parameter	value(s)
simulation duration	194 d
harvest resolution	5 min
solar cell	up to 35 mA (and 4 V)
EWMA smoothing factor	0.8
number of time slots	6, 12, 24, 48
supercapacitor size	25, 50, 100, 200, 400 F
supercapacitor voltage V_{cap}	up to 2.7 V
depletion-safe voltage V_{DS}	1.5 V
cut-off voltage V_{min}	1.3 V
turn-on voltage	1.6 V

Table 6.2: Summary of simulation parameters.

values over more recent ones. As highlighted in Section 2.4.3, an EWMA filter coefficient of $\alpha = 0.8$ yields low error rates for most data sets. Consumption and duration of tasks are based on measured values for real-world cases. Table 6.2 summarizes all parameters used in the evaluation.

6.4.4 Metrics

The evaluation is based on the following metrics.

- **Number of executed task cycles:** The count of successfully completed task cycles. It is assumed that task execution is only helpful, if the entire cycle has been completed. An interruption within a cycle implies that work was only partially completed and all measurements and results cannot be used in the next cycle.
- **Downtime:** The amount (percentile) of time that the node is off due to inaccurate harvest estimation and consumption profile.
- **Cycle Distribution:** The distribution of time intervals between task cycles in terms of the start time of the last task in a cycle.
- **Task Distribution:** The (time) distribution of tasks within cycles.

While monitoring the number of task cycles directly translates to assessing a node's activity, keeping the downtimes at low levels is key. Downtimes are of particular interest, since it is hard to ensure they happen at convenient times, e.g., if the node would sleep anyway. However, monitoring the environment often demands detecting rare events which might be possible during sleeping, e.g., by interrupt-driven wake-up, but is definitely not possible at

downtimes. Moreover, traces are studied in order to explain effects, advantages, and problems of the scheduling and balancing approaches.

6.4.5 Comparison

To highlight deficiencies of existing approaches, the WEDF scheduling approach is additionally implemented, similarly to the baseline of DEOS, c.f. Section 2.3.1. WEDF is a simple yet established approach which is used to compare the algorithm to previously developed work. It underlines the need for a new scheduling approach for dependent task scheduling for EH-WSN. Correct ordering of tasks is ensured by applying the adapted KA at the beginning of the timeslot. Based on the available energy, the number of cycles of the task graph is computed. Following, the deadlines are calculated ensuring subsequent task execution without overlapping. As a consequence, tasks group at the beginning of the timeslot.

6.5 Results

Next, the results of the extensive simulation and real-world experiments are presented and discussed.

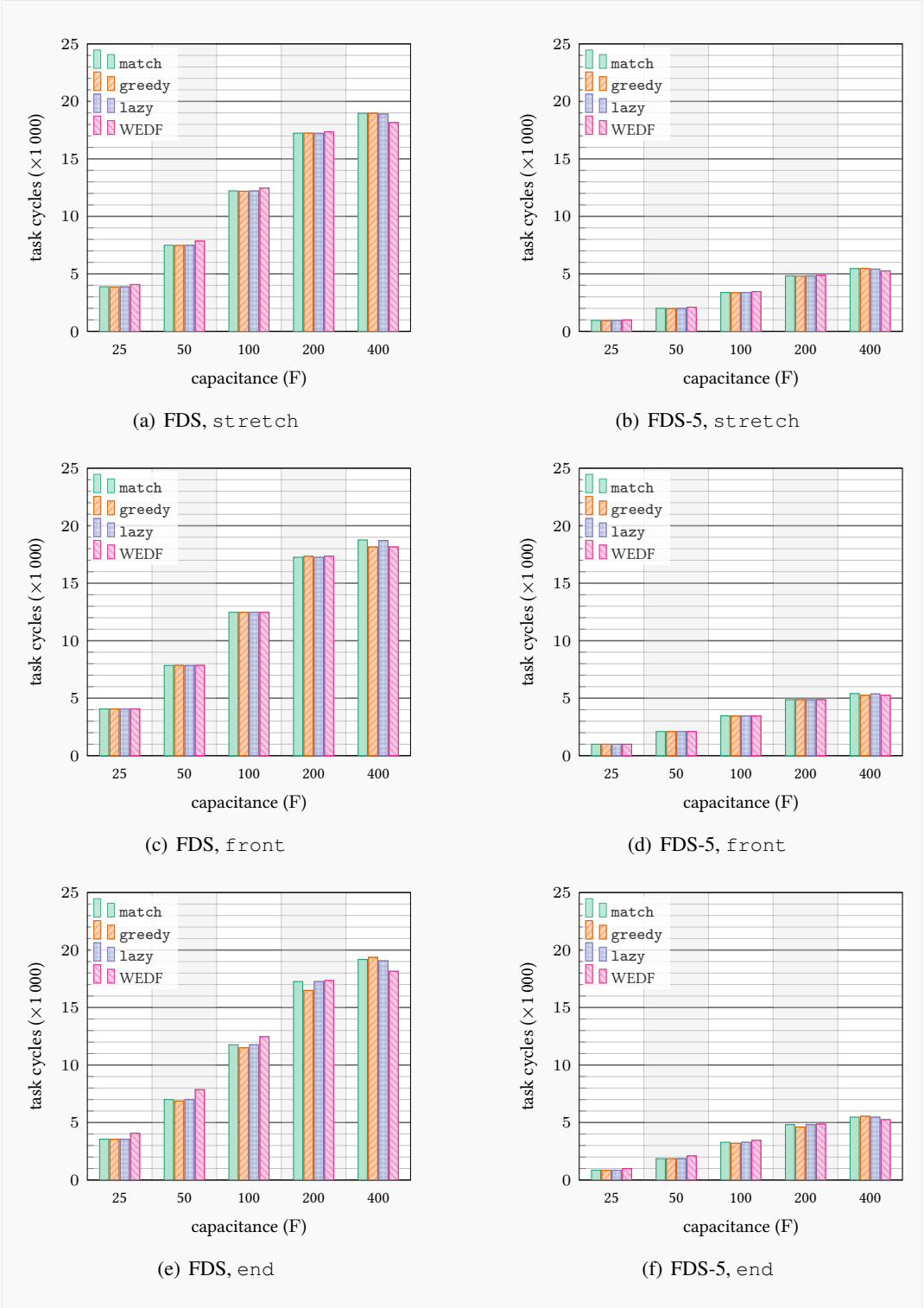
6.5.1 Number of Completed Task Cycles

First, the number of completed task cycles is studied. Larger numbers translate to more sensor data and higher utility of harvested energy, whereas time-resolution and sampling rates may also increase, depending on the balancing scheme. Figure 6.5 gives an overview of the number of completed cycles. The main findings from this figure and other setups are discussed subsequently. All strategies are investigated for different capacitor sizes in F and for different number of timeslots S.

In general, larger capacitors allow for more cycle completions, and are not notably affected by the balancing scheme. The scheduling strategies perform similar as well, with the exception that `match` has a tendency to allow more cycles for the largest capacitance used.

However, it is observed that the number of completed cycles grows slower than the capacitance. The reason is that the available budget is limited by capacitance *and* energy harvest. For small capacitance, the former is the dominant limiter, whereas for larger capacitance, harvest is. For the given harvest, the merit of an increased capacitance is hence decreasing and diminishes at some point.

An expectation is that the number of cycles for FDS is approximately $4.1\times$ higher than for FDS-5. As detailed in Section 6.4.1, the needed energy to complete one cycle is $4.1\times$ higher for FDS-5 compared to FDS. Hence, the overall number of scheduled cycles for FDS-5 is $4.1\times$ less compared to FDS with the same energy budget. For those energy-hungry applications,



(a) FDS, `stretch` (b) FDS-5, `stretch`

(c) FDS, `front` (d) FDS-5, `front`

(e) FDS, `end` (f) FDS-5, `end`

Figure 6.5: Number of completed task cycles for both sample applications and different balancing. `EWMA` predicted harvest and $S = 6$. Note that WEDF makes no use of balancing.

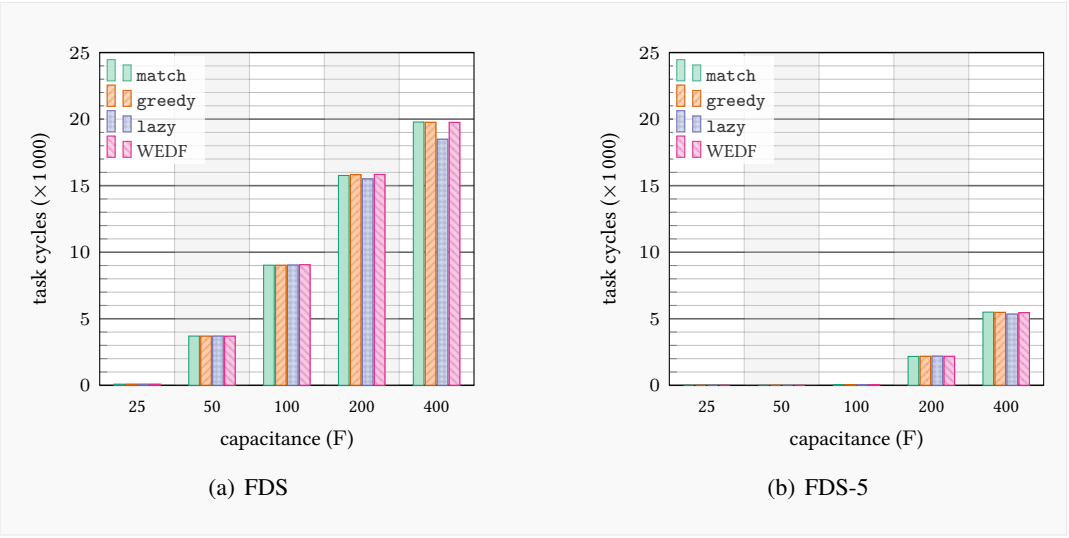


Figure 6.6: Number of executed task cycles for `stretch` balancing. `EWMA` predicted harvest and $S = 48$.

however, this only holds for large capacitors and longer timeslots. For a 25 F capacitor and $S = 12$, a factor of more than 10 is found. The main reason is that cycles are repeated as a whole only, hence giving a discrete number of repetitions and therefore the energy used is a multiple of a single cycle's energy consumption. Moreover, repetitions are scheduled for a single timeslot only, so that the available energy budget is smaller for short slots. If consumption is large w.r.t. a slot's budget, only few repetitions are scheduled. If consumption and budget are close, fewer cycles are scheduled and completed. For small capacitors, this may lead to many cases in which the capacitor is filled in a timeslot and excess energy cannot be stored, so that it cannot be used in the next timeslot. An extreme situation is when slots are so short that the budget does not suffice to schedule a cycle at all. By construction of the budgeting algorithm, the energy budget is a function of slot length, whereas the current budget is not. The consequence of this problem is revealed by Fig. 6.6. For small capacitors, the number of completed cycles is drastically reduced compared to Fig. 6.5. In case of FDS-5, no cycles are schedule at all for 50 F and 100 F.

An in-depth study based on FDS-5 is depicted in Fig. 6.7. Here, the situation arises that scheduling even one task cycle within a 30 min timeslot ($S = 48$) exceeds the budget (or average allowed consumption) in most cases, whereas scheduling in a 60 min timeslot ($S = 24$) is possible most of the time. The underlying problem is that scheduling is not intended to stretch over multiple timeslots. The traces reveal yet another curiosity, which is that for $S = 48$, most cycle executions are in the early morning, which is a peculiarity of the budgeting algorithm analyzed in Section 3.1.

WEDF produces overall an equal number of task cycles; with a tendency to work better with

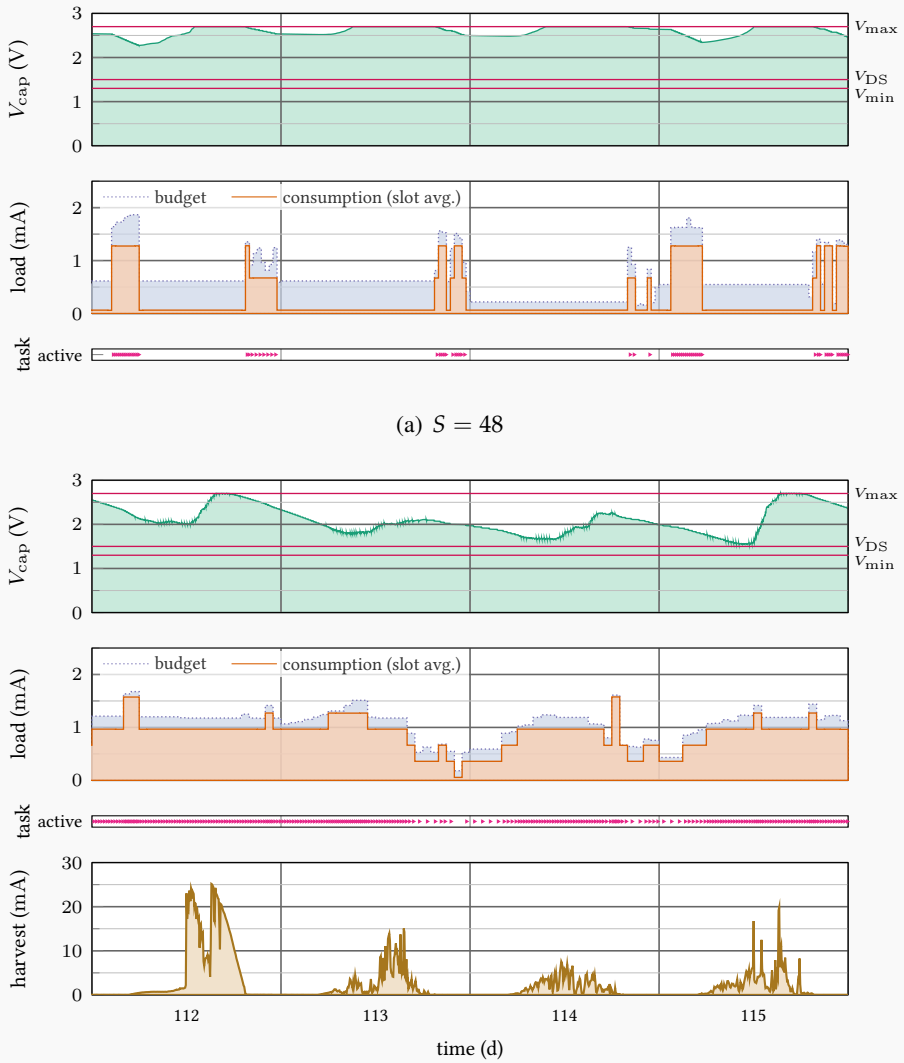


Figure 6.7: Study of simulated task schedules showing the impact of short timeslots. Harvest is equal in both plots and hence only shown once. Marks in the task activity plots indicate beginnings and ends of cycles. FDS application with 100 F capacitor, `match` scheduling with `stretch` balancing for best presentation.

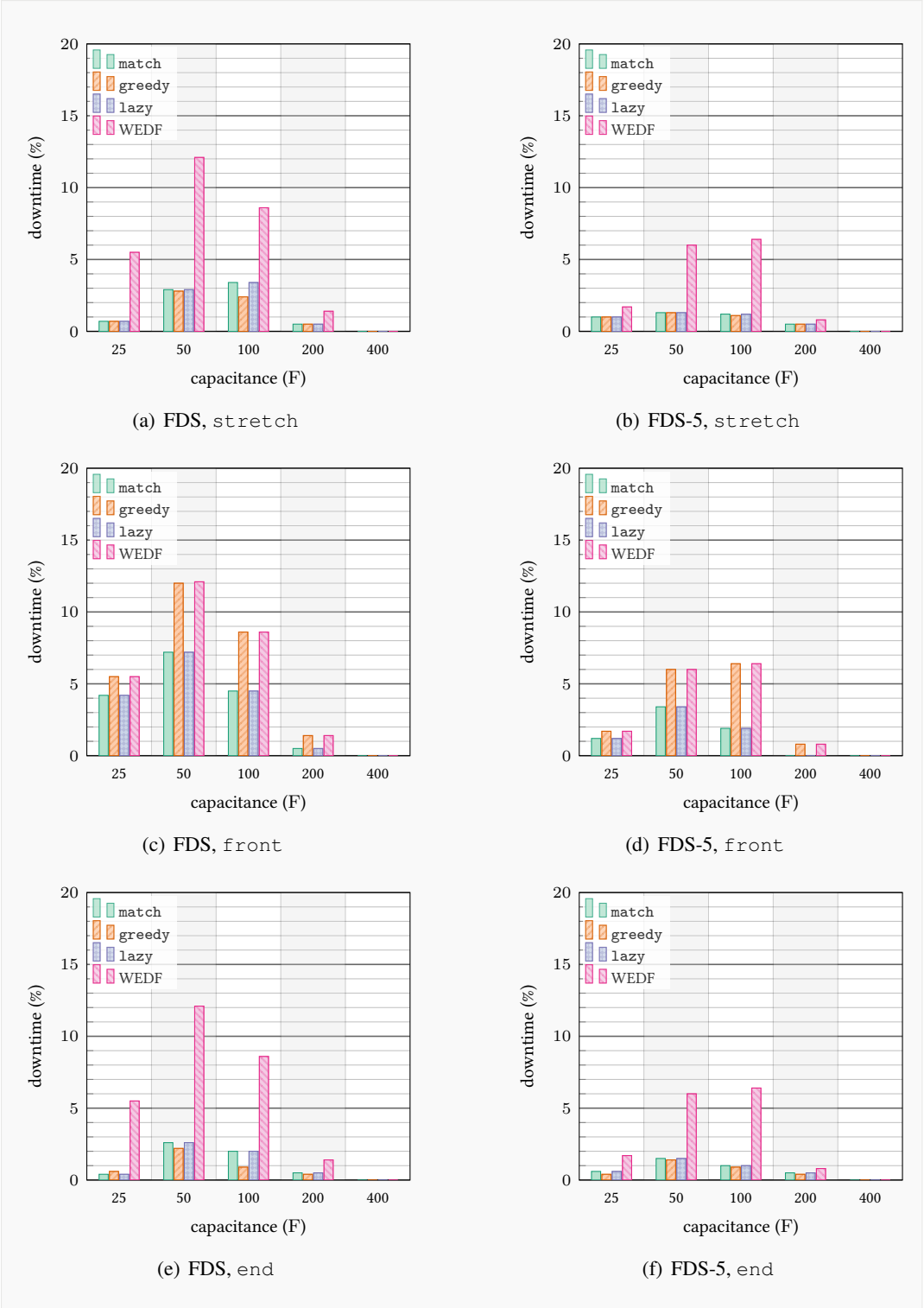
smaller capacitors. Still, the number of executed task cycles alone misses to illustrate the whole picture. High task cycles are not beneficial for times without energy supply, i.e., downtimes (c.f. Section 6.5.2). Rare events might be missed and learned energy harvesting intake patterns—an essential part for prediction algorithms—are lost without persistent memory.

The analysis hence underlines the substantial shortcoming identified in Section 3.1 of the budgeting `DS` algorithm. As a consequence, it is appealing to increase the scheduling horizon but not necessarily increase slot lengths of harvest estimation, because this may have a negative influence on estimation accuracy, as reported in Section 2.4.3.

6.5.2 Downtime

Another essential aspect is that of depleted energy buffers causing node downtimes. The major observation is that downtimes are only notable in case of few, long slots, i.e., $S = 6$. For $S = 12$, downtimes are below 4% in all cases and essentially 0% for all other configurations. This is an important difference to the `DS` algorithm, stemming from the fact that the budget is less frequently exceeded with the presented scheduling approach. The latter only occurs in case of inaccurate harvest estimation, which is timely handled (in terms of rescheduling) at the beginning of the next timeslot.

However, downtimes still pose a serious problem in certain cases, which is illustrated in Fig. 6.8. The figure reveals that downtimes for FDS are less than for FDS-5, which is a direct consequence of less completed task cycles and less effectively used budgets. This adds an unintended safety margin. Downtimes are particularly high for medium-sized capacitors. The reason for this is relatively complex. In general, a larger capacitor increases the safety margin (at constant V_{DS}) in terms of buffering energy before depletion. However, a larger capacitor also implies a longer recharging time in case of depletion. Moreover, the budget is increased (cf. Section 6.5.1). In summary, the chance of depletion is reduced with increasing capacity while the recharging time is increasing. While the number of depletion events decreases, its impact (downtime) decreases. At 50 F, two odds sum up. For smaller capacities, more depletion events happen, but they are shorter. For larger capacities, the number of depletion events is reduced drastically. This discussion pinpoints the sensitivity to several factors (capacitance, harvest, scheduling intervals), and that this particular aspect demands for a tool to inspect the health of a system and task design before deployment. Most importantly yet, the figure illustrates the influence of task (cycle) distribution within the timeslots. Particularly if most consumption happens early (`front`, Fig. 6.8(c) and Fig. 6.8(d)), depletion may occur before harvest arrives. As a consequence, downtimes of up to 12% are experienced; foremost WEDF with an increase of 5×. In case of long slots, the lowest depletion times have been recorded for `end`, because harvest happens before consumption. Figure 6.9 illustrates the effect of task distribution with a depletion occurring early in slot 3 and 4 for `greedy`

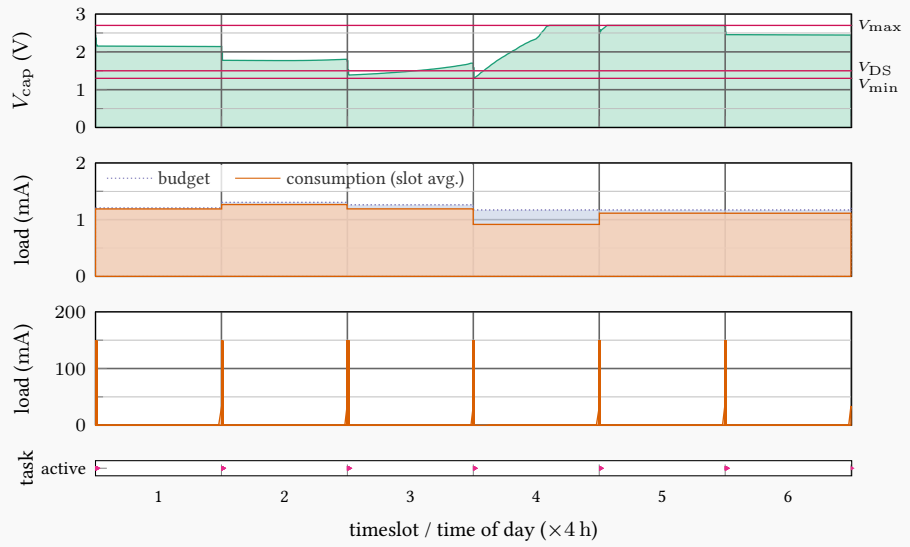


(a) FDS, `stretch` (b) FDS-5, `stretch`

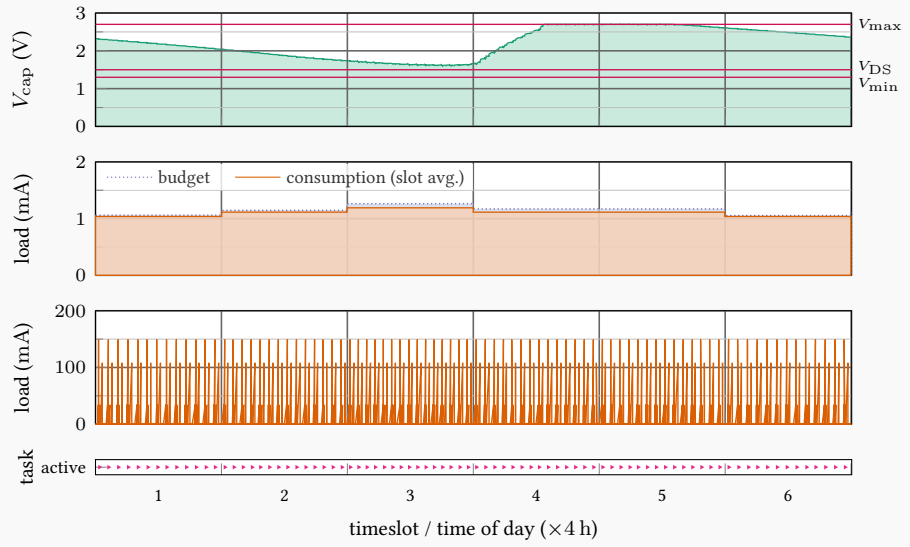
(c) FDS, `front` (d) FDS-5, `front`

(e) FDS, `end` (f) FDS-5, `end`

Figure 6.8: Percentage of downtime for different balancing options. `EWMA` predicted harvest and $S = 6$. Note that balancing schemes do not apply to WEDF.



(a) `greedy` with `front` (depletion at the beginning of slot 3 and 4)



(b) `match` with `stretch` (no depletion)

Figure 6.9: Study of simulated task schedules showing the impact of scheduling strategy and balancing scheme on day 52. Harvest is equal in both cases and omitted for sharpness. Marks in the task activity plots indicate beginnings and ends of cycles. FDS application with 100 F capacitor, and $S = 6$.

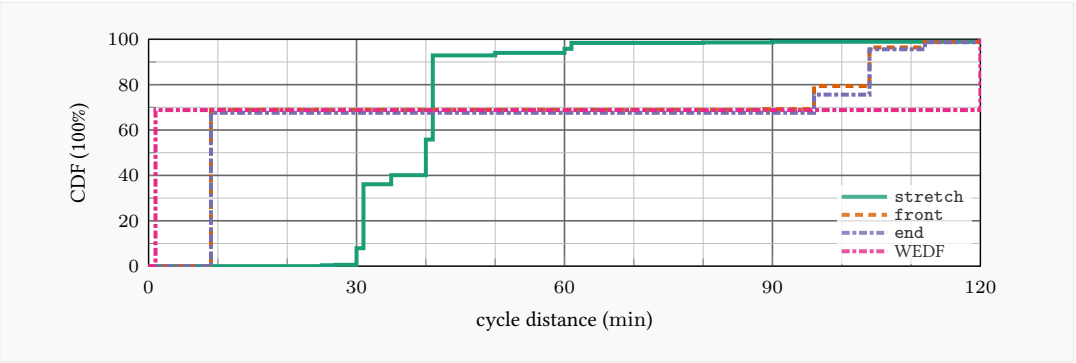


Figure 6.10: Cycle distribution (CDF) for `match`, 50 F, $S = 12$, and FDS. Time resolution is 1 min. The number of scheduled task cycles is similar (ca. 13 500).

combined with `front` (Fig. 6.9(a)). Due to high consumption early in the slot, harvested energy arrives too late to keep the voltage above the critical threshold V_{min}. In contrast, `match` with `stretch` leads to an even (matched) consumption distribution that prevents depletion. The performance of WEDF pinpoints an interesting finding: although downtimes are significantly higher, task cycles drop only slightly. With WEDF, all energy is spent at the beginning which suffices to perform all cycles but fails to keep the capacitor voltage above V_{min}. The node is thus unable to perform any unexpected tasks, e.g., an interrupt-driven event, which can harm performance for certain applications. Combining `front` and `greedy` is thus not advisable for small and medium capacitors, particularly for infrequent schedule updates.

6.5.3 Cycle Distribution

The task cycle distribution is investigated in order to assess the influence of balancing. Figure 6.10 portrays the results for scheduling of the FDS application with parameters leading to low (but non-zero) downtime and a relatively low budget. On average, 3.8 cycles were scheduled per timeslot. For `front` and `end`, 70% of tasks are scheduled with an interval of 8.1 min, the maximum cycle length (cf. Section 6.4.1). Using WEDF even worsens the situation: cycles group at the beginning of the timeslot, leaving the rest of the slot nearly uncovered. Applications which favor an evenly distributed activity, e.g., for regularly receiving control commands, suffer with WEDF due to long inactive periods. The remaining task intervals are much longer, with the majority within 95 – 120 min. In contrast, `stretch` is able to maintain a more narrow and compact distribution with more than 95% of all tasks having an interval within 30 – 45 min. The results pinpoint, that in applications demanding for a smoother sampling (or task cycle execution) rate, `stretch` is to be favored.

	Run 1			Run 2		
Metric	WEDF	`match`	Gain	WEDF	`match`	Gain
Cycles	2.66	3.14	+18.0%	4.65	4.53	-2.6%
Budget	0.94 mA	1.01 mA	+6.8%	1.23 mA	1.20 mA	-2.3%
Harvest	1.68 mA	1.85 mA	+10.3%	4.18 mA	3.94 mA	-5.7%

Table 6.3: Key metrics (mean values) and gain of `match` in real-world experiment.

6.5.4 Task Distribution

Since the distribution of tasks within a cycle is inherent to the scheduling schemes, this aspect is investigated only briefly. In many cases, however, it was noted that `lazy` and `match` lead to similar cycles, because the benchmark applications mainly consist of power-hungry tasks that need long sleep intervals in between. In most cases, these intervals are stretched towards the `expires` constraint. In less power-hungry conditions (or larger budgets due to, e.g., a larger solar cell), `match` will produce more compact cycles. If closer timing between tasks is beneficial (for measurement quality), `match` should be used over `lazy`.

Another important aspect to consider is that `lazy` fails in poorly specified cycle specifications. The most notable variant is if time constraints are very relaxed and will lead to a cycle schedule longer than a timeslot. However, such situations can be detected before deployment by applying the test suite offline.

6.5.5 Real-World Experiment

The simulation results clearly highlight the advantages of the approach over traditional scheduling schemes with harvest adaption such as WEDF. To verify the results practically, an 18-days real-world experiment was run with two ESP32 sensor nodes and the FDS application. Instead of sensors, switched resistors are used to simulate equal consumption, and constant WiFi connection time of 5 s is ensured. Both nodes were equipped with the prototype harvester, a 100 F supercapacitor, and were placed on a window sill. One node was running WEDF and the other one `match` with `stretch`. Both nodes employed `EWMA` harvest prediction with $S = 12$ and $\alpha = 0.8$. After 8 days, the nodes are swapped in order to exclude hardware biases of the harvesters and supercapacitors.

Overall results (per-slot averages) are displayed in Table 6.3. The harvest figures indicate that one harvester produced slightly more energy. This advantage is reflected by the average budget. However, `match` is able to improve over WEDF in terms of average executed cycles per slot. In run 1, `match` turns a 10.3% harvest surplus into 18.1% more executed task cycles. In run 2, `match` turns a 5.7% harvest deficit into only 2.6% less executed task cycles.

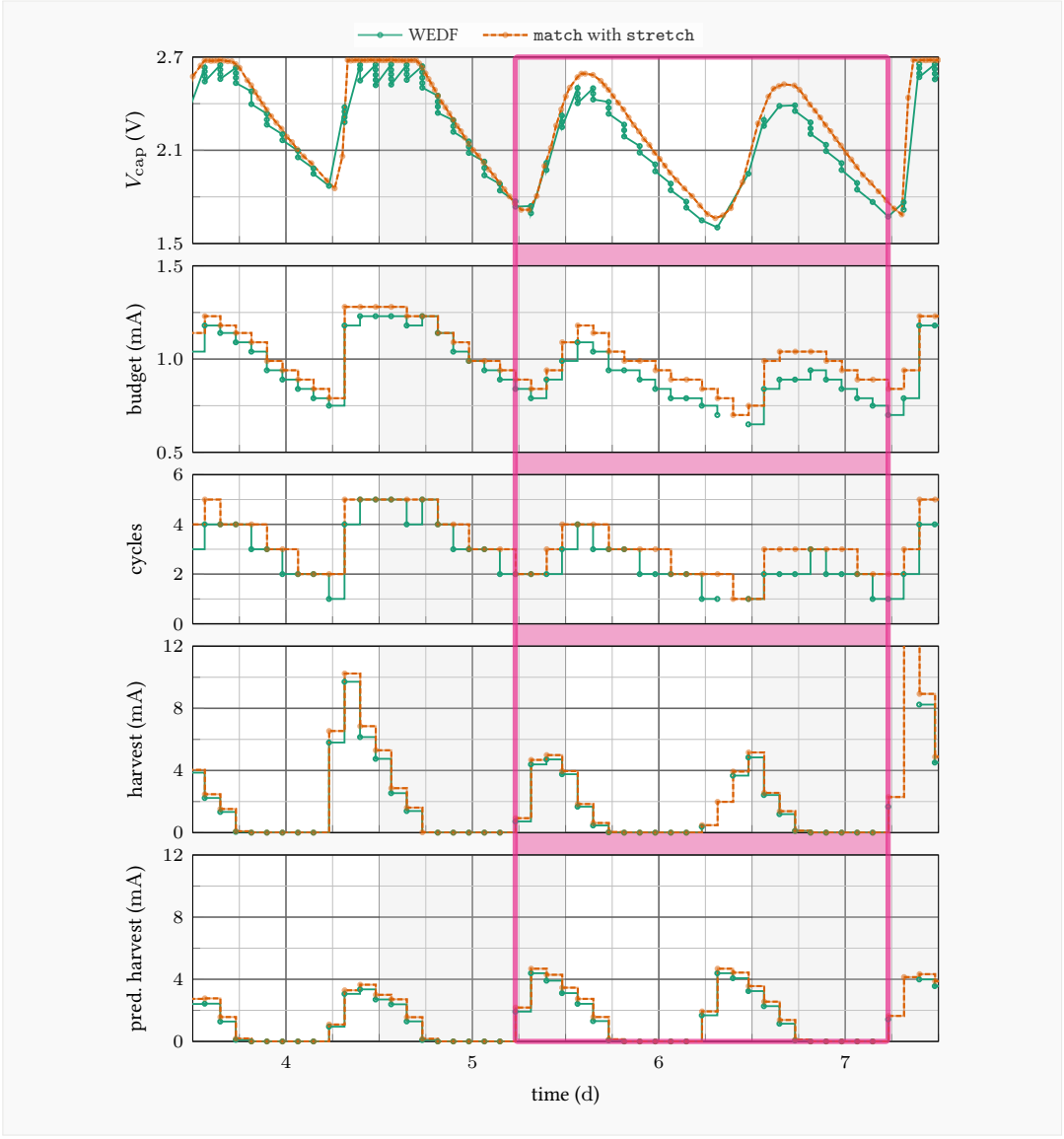


Figure 6.11: Excerpt of real-world experiment. Highlighted area shows days of less harvest than predicted.

A closer look into the first run in Fig. 6.11 showcases the reason for this improvement. While `match` distributes cycles and individual tasks evenly across a slot (trying to obey the budget at all times), WEDF greedily schedules all tasks early in the slot. As a result, capacitor voltage dips notably early in each slot. Even if cycles were stretched, the close timing of tasks would produce notable dips. This can be compensated when harvest exceeds consumption—e.g., at noon of day 4—but produces lower residual energy (lower V_{cap} for WEDF) at the beginning of a new slot in the highlighted area, where predicted harvest is too high. In consequence, the new budget is lower for WEDF as are the number of scheduled task cycles. In these two days, despite a harvest surplus of 9.4%, `match` elevates the budget by 12.0% (25.6% after deduction of sleep consumption) and by a 29.2% extra executed task cycles (2.7 vs. 2.1 cycles on average). Moreover, the voltage dips produced by WEDF increase the risk of depletion at low capacitor voltages, as this situation—as indicated in the figure—typically occurs after the night when harvest is expected to rise but sets in late in the first sunny slot.

6.6 Feasibility and Discussion

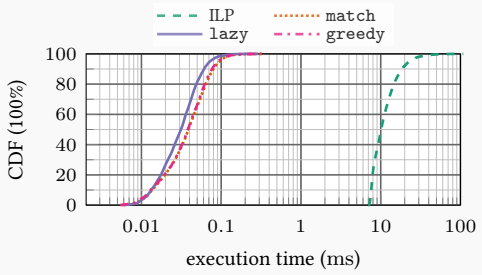
The simulation results and the real-world experiment portray the advantages of the approach. Here, the scheduling overhead and limitations are discussed, and the potential of the approach is underlined.

6.6.1 Runtime Analysis

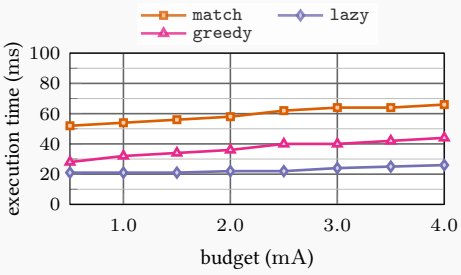
Simulations

To assess the performance in terms of execution time and achievable duty cycle the heuristic is compared with the MATLAB implementation detailed in Section 5.3. To ease the influence of single program configuration, 200 program configurations are generated randomly with achievable constraints.

Figure 6.12(a) portrays the Cumulative Distribution Function (CDF) of different scheduling strategies and the ILP. The heuristic is one to two orders of magnitude faster than the ILP solution. `lazy` needs less time because fewer activities can be performed due to larger time spacing; `match` takes longer, since additional calculations for sleep times have to be performed. Apart from the `lazy` strategy, it is ensured that the achieved duty cycle of the strategies are very close to the solution of the ILP. The smaller energy overhead for scheduling outweighs the loss in duty cycle.



(a) CDF of execution times (simulation); varying budget and tasks.



(b) Microcontroller implementation; strategies vary in complexity.

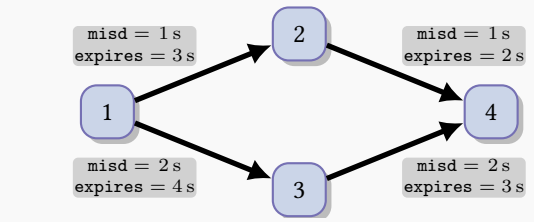
Figure 6.12: Schedule computation for a 60 min horizon.

Microcontroller Implementation

The C++ implementation is compatible with many modern microcontrollers. To verify this claim, the scheduler is compiled and analyzed for a low-power Cortex M0+ microcontroller (c.f. Section 2.2.1) where only 17 kB Flash memory are occupied. The time spent for scheduling a timeslot of 60 min is measured with a clock frequency of 4 MHz, cf. Fig. 6.12(b). Execution time rises slightly with increasing budget; e.g., for the `greedy` strategy by factor 1.42 for a budget increase of factor 8. Due to the additional calculations for sleep time, lower and upper bound for starting times, the execution time for `match` is generally higher. Still, even for a low budget of 0.5 mA the overhead for scheduling is small: with a current consumption of 6.85 mA, a Cortex M0+ spends a mere 0.07% of the budget on scheduling.

6.6.2 Limitations

Describing program structures by graphs and atomic tasks offers an intuitive and flexible way to develop applications for EH-WSN. Jointly with the presented scheduling strategies, it can be used for automatically creating time- and energy-aware schedules. However, unintentional misuse of the flexibility of this approach may result in non-achievable schedules. Figure 6.13 illustrates an example for this: The task graph contains four tasks connected by four edges. For simplicity, it is assumed that all tasks have duration smaller than 1 s. For both scheduling strategies, t_1, t_2 and t_3 do not cause any problems; however, scheduling t_4 leads to constraint violation. When the scheduler sets a starting time s_4, it has to satisfy the constraints of predecessors t_2 and t_3. In case of `greedy`, $\max(\texttt{misdp})$ is chosen, which is 2 s w.r.t. the starting time s_3. The resulting $s_4 = 4\,\text{s}$ looses the information of t_2, since it already expired. The same holds for the `lazy` strategy: only $s_4 = 5\,\text{s}$ is possible, which leads to the violation of edge e_{34}.



(a) Graph

schedule

tasks:	1	2	3	4	
greedy:	0	1	2	4	$\texttt{expires}_{24}$ violated
lazy:	0	3	4	5	$\texttt{misd}_{34}$ violated

(b) Schedule

Figure 6.13: Misconfiguration can lead to deadline violation independent of the used scheduling strategy.

Until now, constraints are not checked for these conflicts before scheduling, because misuse of an application developer is out of the scope of this work. However, it is possible to remove this limitation, e.g., by using a shortest path algorithm before applying the adapted KA.

Right now, only per-node scheduling is considered and network-inherited activity, e.g., due to MAC, is neglected. Although the approach is flexible enough to support these tasks, e.g., regularly receiving beacons, a thorough analysis in this area is needed.

6.6.3 Key Messages

Evolving applications for EH-WSNs overcharge existing scheduling techniques, since they cannot react on burst energy spending. It was shown that keeping the eye on *when* to spend the energy and *how fast* to spend the energy decrease downtimes significantly, e.g., $5\times$ for a fine dust sensor equipped with 50 F capacitor. It was also showcased in practice that—even if downtimes are similar—the approach leads to higher activity of the node of up to 28%.

6.7 Conclusion

The evolving, potentially complex program structures of modern EH-WSN applications impose new challenges on program specification and task scheduling. The separation of task implementation and definition of dependencies, as already present in intermittent systems was advocated. This change of paradigm requires sophisticated energy- and constraint-aware task scheduling. The presented approach supports a variety of scheduling techniques, ensuring

ENO while still satisfying dependent deadlines of atomic tasks. Through a detailed study with real-world solar traces and practical evaluation, the suitability and characteristics of these scheduling strategies are shown and their advantage over existing techniques is demonstrated. Still, there is no ideal strategy for each scenario; thus pros and cons are pinpointed, and the usage of the simulation toolkit is highlighted to assess the validity of task design before deployment and to dimension the energy supply. For future work, it is key to extend the influence of the application developer by incorporating user-defined utility into scheduling.

Chapter 7

Time-varying Utility for Short-term ENO systems

As detailed in Section 3.3, existing literature lacks techniques to incorporate time-varying utility into short-term ENO systems. Furthermore, the presented algorithm `EmRep` resolves the energy storage saturation problem as highlighted in Section 3.1.

7.1 Introduction

This section highlights the need for an energy management algorithm that resolves two common problems: energy storage saturation in times of high harvest and lack of support for user-defined utility. This topic is motivated first, followed by a presentation of the existing problems. Core contributions are highlighted afterwards as well as an introduction into the chapter structure.

7.1.1 Motivation

Over the past years, energy harvesting has been a vital competitor for powering WSNs in many applications. Perpetual operation without human intervention is especially important in critical places, e.g., bridges and mountains, where access is time-consuming and costly. As detailed in Section 2.3, the varying nature of ambient energy entails careful management of activity to use energy efficiently. The evolving field of IC (c.f. Section 2.3.2) poses new challenges on rethinking hardware, program design, and energy management. Still, their operation is tied to

Publication

Parts of this chapter are published in [HR22] and received a funding of the *Funding Program "Open Access Publishing"* of Hamburg University of Technology (TUHH).

sufficient harvest so usage of IC is limited to applications where harvest and times of interest overlap.

Hence, ENO devices—which bridge phases of low energy intake—are indispensable in applications where uninterrupted operation (depletion safety) is mandatory, such as bridge health monitoring. Here, Vibroacoustic Modulation (VAM) [ODRR21] and others, require external vibrations by trains or cars; hence, measurements are only possible or meaningful at specific times of the day. This leads to time-varying utility which may not correlate with energy harvest and has to be incorporated into energy management. Approaches such as `PreAct` model year-around utility but the performance on a day-basis is suboptimal (c.f. Section 7.4). Here, small storage and uncertain harvest forecasts may lead to unexpected shutdowns. Traditional energy management, however, on day-basis does not yet model time-varying utility, c.f. Section 3.3.

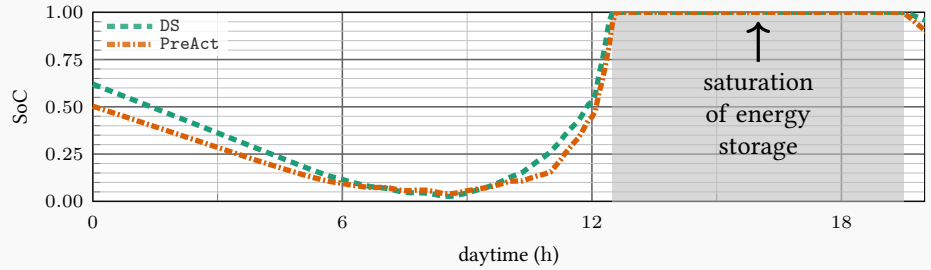
The requirements of modern applications—also including industrial context—demand a new way of managing energy for EH-WSNs. Hence, this chapter presents `EmRep`[1] which integrates utility into energy management. `EmRep` uses SoC estimation to identify low-intake and high-intake phases, and then decouples energy management in these two phases. This allows to utilize the surplus energy in high-intake phases without sacrificing depletion safety in low-intake phases.

Especially in industrial applications, perpetual operation with defined service levels are required. Previously published energy management algorithms (c.f. Section 2.3.1) aim to provide a stable duty throughout a finite horizon without being optimized for time-varying activity. Real-time systems with energy consideration pass the burden of varying utility to the application, but still energy is mostly spent when it is available. Hence, new energy management needs to incorporate time-varying utility to provide sufficient service levels at all times with a large variety of energy storage sizes.

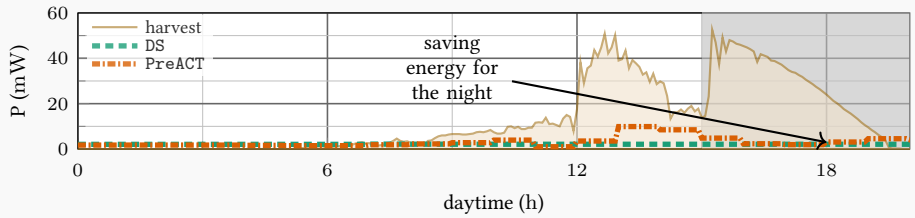
7.1.2 Problem Analysis

As stated before, up until now, no energy management algorithm exists which incorporates time-varying utility and is targeted at small-sized storage and short-term horizon. Small storage is assumed to charge once per day; hence the capacity limits the usable energy per day. Especially when a stable duty cycle is preferred, times of no or low harvest intake dictate the desired duty cycle. However, even with small solar cells, e.g., $35 \times 35\,\mathrm{mm}$, the harvest during the day often exceeds the node consumption by a magnitude, c.f. Fig. 7.1(b). The energy surplus leads to saturation of the energy storage, the solar panel is short-circuited, and incoming energy is wasted.

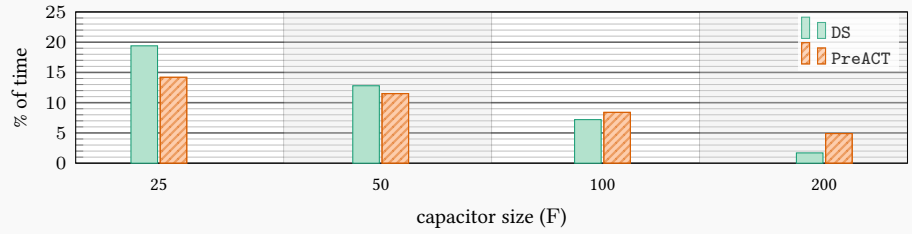
[1] `EmRep` is an (**e**nergy)-analogy to the workout type **AMRAP**, where the goal is to execute **A**s **M**any **R**epetitions **A**s **P**ossible.



(a) Exemplarily: SoC of two common energy management algorithms with small 50 F supercapacitor.



(b) Corresponding power consumption shows saving for the low-intake phase; although SoC and harvest are high; `PreAct` tries to compensate while `DS` keeps stable power consumption.



(c) Percentage of time spent with full energy storage (saturation); simulation of 194 d.

Figure 7.1: Study of saturated energy storage; storage is full up to 19.4% of time for smaller storage; incoming energy at harvester is wasted.

Figure 7.1(a) showcases one day with different energy management algorithms. The used energy storage is a 50 F supercapacitor with 35 × 35 mm solar panel. The node uses `Pro-Energy`, a state-of-the-art solar-harvesting estimation algorithm. However, both supplied management schemes, i.e., `DS` and `PreAct`, fail to utilize all energy in the afternoon: nearly seven hours of high energy availability is wasted.

This behavior adds up, as highlighted in Fig. 7.1(c). For small-sized storage, e.g., a 25 F supercapacitor, in up to 19% of the time in simulations surplus energy is wasted. In total this adds up to 903 h or an average of 4.65 h/d. Hence, `EmRep` is designed which targets storage saturation mitigation and aims at efficiently using harvested energy under time-varying utility.

7.1.3 Contributions

This chapter hence

- integrates time-varying utility into energy management for ENO systems;
- exploits energy surplus which lowers storage saturation by up to 32% without increasing risk of depletion;
- increases performance across all application-specific utility profiles;
- enables downsizing of storage size even with solar harvest estimation; and
- performs a case study for VAM in which `EmRep` increases performance by 80%.

7.1.4 Structure

The remainder of this chapter is organized as follows. Section 7.2 introduces the underlying system model, and introduces `EmRep` as well as its principles based on identification of harvest regions and utility optimization. In Section 7.3 necessary changes in the simulation framework are highlighted, relevant metrics are defined, and real-world connected utility profiles are developed. Section 7.4 showcases the extensive simulation study on lowered storage saturation times, interplay of storage capacity and utility, effect of utility profiles, and the impact of non-ideal harvest forecast. Furthermore, Section 7.4.5 performs a case study based on bridge health monitoring, while Section 7.4.6 elaborates on limitations of `EmRep`. This chapter finishes with Section 7.5 in which achieved goals are evaluated and related to future research directions.

7.2 Design of EmRep

As highlighted in Section 7.1.2, one dominating problem is identified with existing energy management schemes: bridging low-intake phases limits the management flexibility in high-intake phases. This entails saturation of energy storage and hence wastes surplus energy.

7.2.1 System Model

The system model is similar to `PreAct` and is depicted in Fig. 7.2. In this model, the *node* includes the microcontroller, a radio module and other attached hardware such as sensors and actuators. The Energy Harvesting (EH) circuit side consists of two essential entities: the harvester and the energy storage. External devices such as ADCs or coulomb counters gather information about the harvest H throughout a time horizon T_{H}, and the SoC of the energy storage.

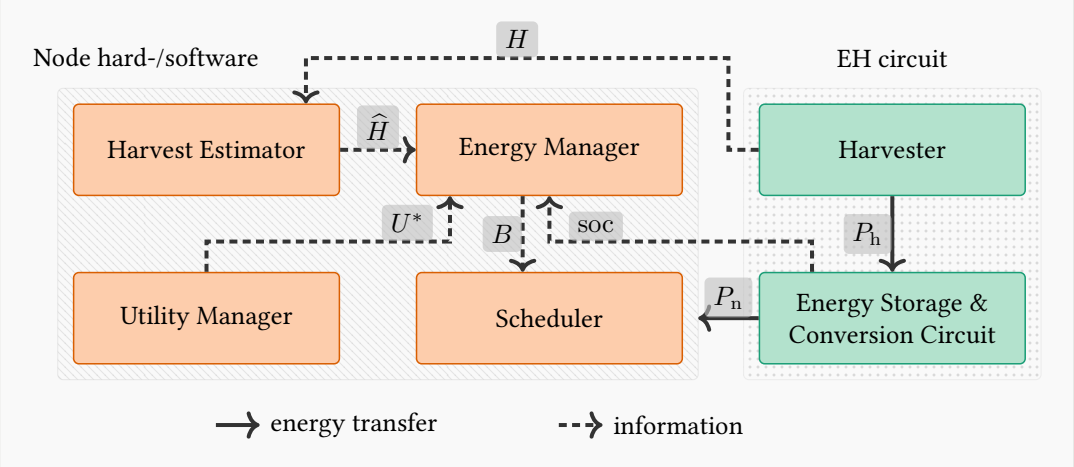


Figure 7.2: Modeling of the harvester system; harvested energy is stored before used by the node; energy manager joins harvest estimation $\widehat{H}$ and utility profile U^* to calculate energy-aware budget B; energy storage and conversion inherits multiple efficiencies; time-dependencies are omitted for readability.

Although `EmRep` is designed with solar energy harvesting in mind, the energy source is exchangeable in general. However, it is assumed that a certain pattern of the harvest source can be learned. For solar harvesting, this pattern may be influenced by buildings, trees and angle to the sun, whilst for vibration harvesting, machine times or rush hour traffic influence the pattern. The energy storage entity may be one or multiple supercapacitors or a battery. The generated power at the harvester P_h translates to an energy transfer into the storage. The node is attached to the energy storage and converter circuit of the harvester with a power consumption of P_n. Harvest power, energy storage and converter circuit inherit different efficiencies that are omitted here for model simplicity since these are highly platform-specific. The used toolkit, c.f. Section 4.3, includes efficiencies for charging and output power conversion. For maximum efficiency, the storage can be connected directly to the node and, hence, serves as supply. If conversion losses can be accepted, a boost converter can be used to stabilize supply voltage. Furthermore, more energy from the capacitor can be used since it can be discharged to lower voltage levels. For common supply voltage levels, e.g. $V_\mathrm{s} = 3.3\,\mathrm{V}$, the conversion efficiency is usually above 80% (c.f. Section 4.2) so that benefits outweigh. As a consequence, the power demand of a sensor node $P_\mathrm{n}(t)$ simplifies to

$$P_\mathrm{n}(t) = V_\mathrm{s} \cdot I_\mathrm{n}(t), \tag{7.1}$$

with $I_\mathrm{n}(t)$ being the current drawn by the node. This current varies with used peripherals, e.g., radio or MCU, and has to be obtained by in-lab measurements or on-line. Over a time-interval Δt, $I_\mathrm{n}(t)$ regularly fluctuates but literature for energy management (c.f. Section 3.1)

has shown that the time average over Δt can be used without sacrificing accuracy.

The software side of the model consists of four entities: harvest estimator, utility manager, scheduler and energy manager. As discussed in Section 7.1, applications may demand time-varying use of resources, e.g., prefer high sampling rates in the morning. These demands are fed into the energy manager with a utility profile $U^*(s)$. Since ENO requires spent energy to be not greater than harvest energy for a finite time horizon T_H, it is crucial to estimate harvest intake. For solar energy, numerous algorithms exist, e.g. `Pro-Energy`, `WCMA` and `Delta-T`. Such algorithms try to learn prevailing patterns and estimate future harvest $\widehat{H}$ with as low error as possible. This information is used by the energy manager to decide on an appropriate duty cycle $\mathrm{DC}(s)$, i.e., the ratio between active times and overall time. Please note that EH devices usually divide the horizon T_H into slots of fixed time, with time slots $s \in 1,\ldots,S$ usually lasting 30 – 240 min for a diurnal cycle. Given a known active node power consumption P_n, the duty cycle $\mathrm{DC}(s)$ can be used to calculate the target power consumption

$$P_\mathrm{n}^*(s) = V_\mathrm{s} \cdot I_\mathrm{n} \cdot \mathrm{DC}(s) = V_\mathrm{s} \cdot B. \tag{7.2}$$

In literature, the product of active power consumption of the node and duty cycle is often referred to as budget B.

`EmRep` implements the energy management entity; and thus merges information about future harvest, current SoC and instructions for the scheduler.

7.2.2 Utility

Literature suggests different definitions of the utility. Either as a relative metric at which a portion of the overall available energy is distributed. E.g. `PreAct` implicitly defines the utility as a portion of the mean expected energy intake. A more application-specific approach, e.g., as presented in Section 2.3.1, directly relates the utility to a sampling rate. Since applications usually require a specific amount of activity, e.g., structural health monitoring with desired number of measurements per hour, the latter approach is more promising. Hence, the achieved utility $U(s)$ is given by

$$U(s) = \frac{B(s)}{\mathrm{B}_\mathrm{max}}, \quad s \in 1,\ldots,S \tag{7.3}$$

with B_max denoting the maximum achievable budget and correspondingly sampling rate and $B(s)$ the current average node consumption. B_max is the slot average of the node consumption, which directly translates to the sampling rate if the energy consumption of the sampling activity is known. Correspondingly a utility profile $U^*(s)$ is defined which declares

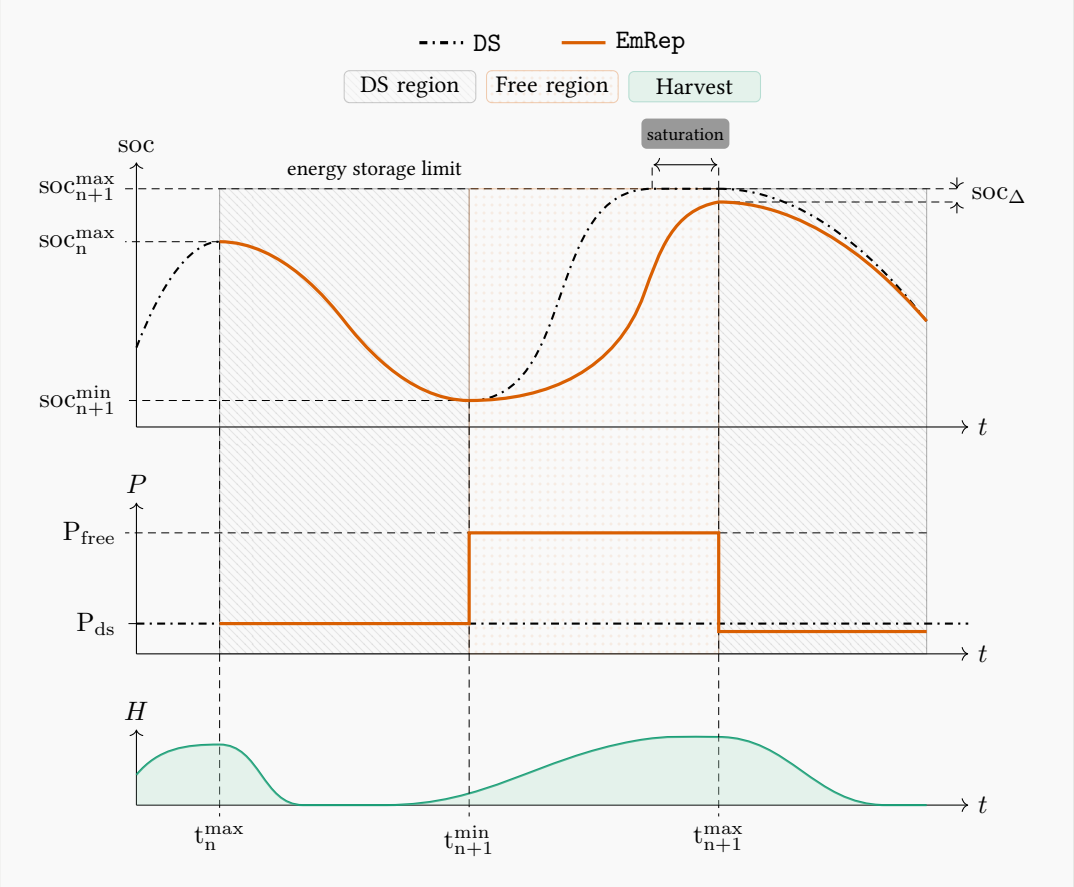


Figure 7.3: Behavior of EmRep; management regions are determined using SoC estimation; in DS region power consumption P_{ds} ensures uninterrupted operation; in Free region the energy surplus is exploited to provide a higher P_{ds}.

a desired utility throughout the horizon T_H. Next, the main characteristics of EmRep are introduced.

7.2.3 Algorithm

As briefly introduced in Section 7.1, EmRep uses SoC estimation to identify high-intake and low-intake phases, manages energy separately in these phases together with aiming at utility maximization.

SoC Curve

The SoC of the presented harvester depends on the usable energy in the capacitor. Ideally, the energy in the capacitor can be calculated via $E_{cap}^{*} = \frac{1}{2} \cdot C_{cap} \cdot {V_{cap}}^2$. In practice, however,

energy from the capacitor below V_{min} is not feasible, c.f. Section 4.2. Hence, the usable energy in the capacitor is

$$E_{\text{cap}} = E^{*}_{\text{cap}} - E^{\text{min}}_{\text{cap}} = \frac{1}{2} \cdot C_{\text{cap}} \cdot \left({V_{\text{cap}}}^2 - {V_{\text{min}}}^2 \right). \tag{7.4}$$

For the used harvester with supercapacitor, the SoC is the ratio between E_{cap} and $E^{\text{max}}_{\text{cap}}$ and hence is calculated by:

$$\text{soc}(V_{\text{cap}}) = \frac{E_{\text{cap}}}{E^{\text{max}}_{\text{cap}}} = \frac{{V_{\text{cap}}}^2 - {V_{\text{min}}}^2}{{V_{\text{max}}}^2 - {V_{\text{min}}}^2}. \tag{7.5}$$

The SoC curve of devices powered by solar energy usually follows the trend depicted in Fig. 7.3: a minimum—usually at the end of the low energy intake phase—and a maximum—during or at the end of the high energy intake phase. Those two extrema naturally divide the prediction horizon T_{H} into two regions. In the low energy region, little or now harvest can be expected; thus preventing depletion of the energy storage is key. In the second region, the harvesting intake often exceeds node consumption which either saturates energy storage or gives more freedom to spend energy. In `EmRep`, these two regions are called the DS region and the Free region; cf. Fig. 7.3.

Directly after passing $\text{t}^{\text{max}}_{\text{n}}$, `EmRep` estimates the SoC for the next prediction horizon T_{H}. The employed SoC estimation strategy highly depends on the used platform; e.g., if a supercapacitor or regular battery is used. Two two-tuples for the next minimum $(\text{t}^{\text{min}}_{\text{n+1}}, \text{soc}^{\text{min}}_{\text{n+1}})$ and maximum $(\text{t}^{\text{max}}_{\text{n+1}}, \text{soc}^{\text{max}}_{\text{n+1}})$ are stored.

For SoC estimation, the depletion safe strategy, analyzed in Section 3.1, is used. The simplified equivalent circuit, found in Fig. 2.2, allows the authors to simulate the SoC course of the energy storage. When the sun shines, the harvesting current is $I_{\text{h}} > 0$. The consumption part consists of a boost converter with efficiency $\eta\left(I_{\text{n}}, V_{\text{cap}}\right)$ which provides the output voltage V_{n}. The sensor node draws a current I_{n} from the boost converter output; the boost converter draws the current I_{r} at it's input. If I_{h} exceeds I_{r}, the capacitor is charged with current I_{c}. To prevent overcharging, the switch S_{h} disconnects the solar panel whenever $V_{\text{cap}} > V_{\text{max}}$ is detected. Since the current flowing into the reserve is the difference between harvest current and boost converter current,

$$I_{\text{c}} = I_{\text{h}} - I_{\text{r}} \tag{7.6}$$

holds. According to [RUTR14], the converter efficiency equation (equal power at in- and output side)

$$I_{\text{r}} \cdot V_{\text{cap}} \cdot \eta\left(I_{\text{n}}, V_{\text{cap}}\right) = I_{\text{n}} \cdot V_{\text{n}} \tag{7.7}$$

and the capacitor model $I_c = C \cdot \dot{V}_{cap}$, form the final formula for the system model together as:

$$C \cdot \dot{V}_{cap} + I_h = \frac{I_n \cdot V_n}{V_{cap} \cdot \eta\left(I_n, V_{cap}\right)}. \tag{7.8}$$

Simulating the time-dependent capacitor voltage course with this formula, however, is not feasible due to the additional hardware characteristics. The boost converter is only able to provide a stable output voltage if a capacitor voltage $V_{cap} > V_{min}$ is achieved; additionally $V_{cap} \leq V_{max}$ due to the overcharging protection. The solution however, is possible for discrete time intervals (timeslots) since I_h and I_n can be assumed to be piece-wise constant within the timeslot. This results in a first-order differential equation whose solution is approximated with the Newton's method and a binary search algorithm.

It finds the maximum power consumption $P_n^*(s) = P_{ds}$ of the node throughout the complete horizon without depleting the energy storage. This behavior is also shown in Fig. 7.1(b), where `DS` keeps a stable power consumption neglecting energy surplus. `EmRep` uses this power consumption as baseline in the DS region. Based on P_{ds} the SoC curve for the next prediction horizon is calculated.

Regions

In the DS region, the node consumes the previously calculated P_{ds}. Up until t_{n+1}^{min}, `EmRep` only checks if the target soc_{n+1}^{min} will be reached and slightly lowers/increases P_{ds} if necessary; i.e., the SoC estimation reruns until t_{n+1}^{min} if estimation and reality differentiate.

After t_{n+1}^{min} passed, the Free region starts with a new iteration of the voltage course estimation as lower limit for P_{free}. Still, not for T_H but for

$$T_{free} = t_{n+1}^{max} - t_{n+1}^{min}. \tag{7.9}$$

Here, to keep depletion safety for the whole horizon T_H, it is crucial that soc_{n+1}^{max} is met at t_{n+1}^{max}. A certain SoC deviation soc_Δ is allowed to increase consumption flexibility. Large values of soc_Δ allow for a higher power consumption during the Free region but increase the risk of downtime, and lower the power consumption in the following DS region. The parameter evaluation shows that soc_Δ should range between 1% and 19%; above and below this limit, the performance suffers significantly. It was found that $soc_\Delta = 5\%$ offers a good balance across storage size and prediction methods.

As stated in Section 7.1, `EmRep` is tailored to small size storage which is assumed to recharge significantly during the horizon. If T_{free} is small, or the margin between SoC extrema is small, the flexibility for `EmRep` is small and hence P_{free} is only slightly higher than P_{ds}. For harvesting patterns which may lead to two SoC peaks, `EmRep` still works and the DS region

and the Free region occur multiple times per horizon. However, in such situations T_{free} and the difference between SoC extrema is small and so is the surplus energy; hence the benefit of `EmRep` towards `DS` decreases.

Utility Optimization

As stated in Section 7.2.2, utility is defined as ratio between a maxium budget and achieved budget. This linear correlation allows for optimization of the budget; hence a maximum budget also maximizes the utility. Additionally, it is not beneficial for the application to provide a budget higher than B_{max}. In the SoC curve estimation, `EmRep` takes this into account and uses the utility profile $U^*(s)$ together with B_{max} as upper bound for the budget in slot s.

Schedule Generation

As detailed in Fig. 7.2, `EmRep` implements the energy manager entity. To develop a schedule which can be executed by the application, however, a scheduler is needed, similarly to the one presented in Chapter 6. `EmRep` is compatible with schedulers that keep a target average consumption B, equivalent to a duty cycle as stated in Eq. (7.2). For evaluation, a scheduler is needed that transforms the average consumption into a uniformly distributed set of activities within the timeslot. The simple scheduler, which is called *classical* scheduler, assumes a program to consist of only one part with duration d_{task} and an average power consumption P_{task}. First, the classical scheduler determines the overall energy available in the slot and calculates the maximum number of repetitions $N \in \mathbb{N}^+$ of the program which can be executed so that the assumption

$$B \cdot V_s \cdot T_S \geq N \cdot d_{\text{task}} \cdot P_{\text{task}} + (T_S - N \cdot d_{\text{task}}) \cdot P_q \tag{7.10}$$

holds. Here, T_S denotes the length of the timeslot and P_q the power drawn in the used sleep mode. Afterwards, starting times for N tasks are generated by distributing the times uniformly throughout the slot. To ease the influence of the task distribution within the timeslot, the classical scheduler is used in favor of the version presented in Chapter 6.

7.3 Simulation Tools and Metrics

In this section, energy managers used for comparison are introduced, changes in the simulation tools are detailed, the used metrics are outlined and the relevancy of utility profiles is motivated. For comparison, the authors of `PreAct` kindly provided an implementation of their state-of-the-art energy manager for long-term ENO systems. Since depletion safety is key for `EmRep`,

parameter	value(s)
simulation duration	194 d
harvest resolution	5 min
solar cell	up to 35 mA (and 4 V)
`EWMA` parameters	$\alpha = 0.8$
`WCMA` parameters	$\alpha = 0.8$, $D = 10$, $K = 5$
`Pro-Energy` parameters	$\alpha = 0.5$, $D = 14$, $K = 3$, $G = 5$
number of time slots	6, 12, 24
supercapacitor size	25, 50, 100, 200, 400 F
supercapacitor voltage V_{cap}	up to 2.7 V
depletion-safe voltage V_{DS}	1.5 V
shutdown voltage V_{min}	1.3 V
turn-on voltage	1.6 V
maximium budget B_{max}	2.73 mA

Table 7.1: Summary of simulation parameters.

a natural competitor is the `DS` algorithm. Please note that `DS` is utility-agnostic whilst both `PreAct` and `EmRep` include time-varying utility into their management by design.

7.3.1 Simulation Tools

The presented simulation toolkit, c.f. Section 4.3, enables to ensure repeatable conditions and a fair comparison. The coherence between hardware prototype and simulation toolkit is shown in Section 6.4. Additionally to the provided `EWMA` harvest estimation, an implementation of the `WCMA` estimation as well as the state-of-the art estimation algorithm `Pro-Energy` is added. Moreover, and *ideal* estimator is added which contains the slotted averages of the original trace. Furthermore, the extended simulation framework contains a utility manager which gives control about the utility profiles. Please note that the framework also simulates node shutdown, i.e., a voltage dropping below V_{min}; the utility is consequently set to zero. Table 7.1 gives an overview of the used simulation parameters. The parameters for `WCMA` and `Pro-Energy` are chosen based on suggestions in the corresponding papers. However, the goal is not to compare prediction methods but to investigate the influence of differently designed prediction methods on energy managers. Only the subset of simulation results with most benefit for the reader is shown.

7.3.2 Metrics

To evaluate the performance of the different energy managers, utility is used as the primary metric as presented in Section 7.2.2. The utility is used to define the specific amount of activity for the sensor node; hence it is assumed that a utility in a timeslot s greater than the specified utility $U^*(s)$ is not beneficial. Therefore, the *effective* utility is defined as:

$$U_{\text{eff}}(s) = \frac{\min(U(s), U^*(s))}{U^*(s)}. \tag{7.11}$$

Enabled by this metric, it can be judged how well the desired utility profile is met. As stated in Section 7.2.2, the utility is defined based on a maximum budget $\mathrm{B}_{\max}$ which can be achieved by the sensor node. Defining $\mathrm{B}_{\max}$ is non-trivial: since the sensor has to fulfill ENO condition, it can not spend more than it harvested. Hence, the average current fed into capacitor is chosen as maximum budget resulting in $\mathrm{B}_{\max} = 2.73\,\mathrm{mA}$. However, this value can also be changed to an application-specific value, e.g., representing a budget needed for desired sampling rate. To reveal differences between configurations, a relative utility U_{rel} is defined:

$$U_{\text{rel}}(s) = \frac{U_{\text{ref}(s)} - U_{\text{eff}}(s)}{U_{\text{ref}}(s)}. \tag{7.12}$$

The reference effective utility $U_{\text{ref}}(s)$ is specific to a configuration, e.g., for a specific capacitor size.

7.3.3 Utility Profiles

To highlight the influence of time-varying utility on energy-management, four different utility profiles are selected, which are depicted in Fig. 7.4, and motivated in the following. Please note that the average utility throughout T_{H} is equal for all profiles since it directly influences the results of the effective utility. All utility profiles are defined with hourly resolution, which was found to be most intuitive. If higher resolution is required, it can be changed in the simulation.

The *Const* profile serves as baseline for management, as it replicates utility-agnostic behavior. Each slot is equally important and hence no saving of energy is performed. However, it is totally different from greedy algorithms or IC devices as the goal is to provide a constant duty cycle—also in nighttime. An example application is a weather surveillance station which constantly gathers temperature and humidity in the air.

The *Workday* profile represents a classic case at which harvest is aligned with interested of the application. Between 8 h and 20 h, the interest of the application is very high, while evening, night and very early mornings are out of interest. This should be the profile where `EmRep` performs best since it profits most from application interest in the Free region. An

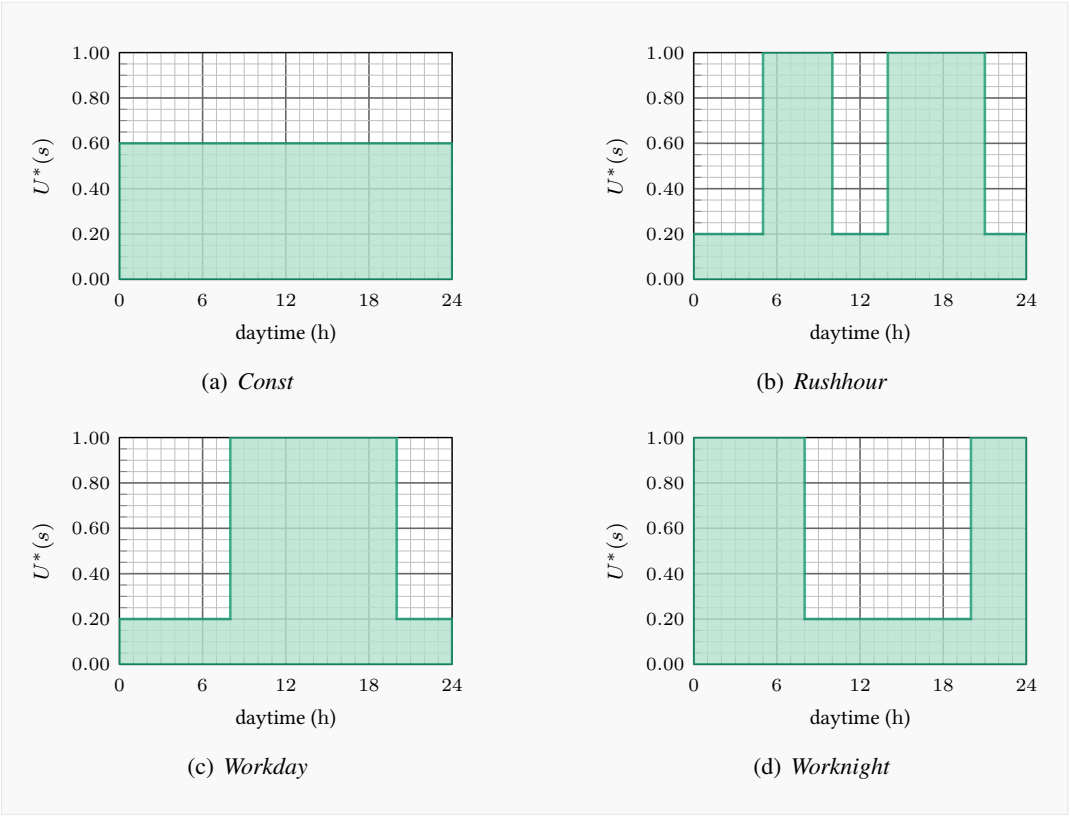


Figure 7.4: Overview of simulated utility profiles; average requested utility is equal for all profiles; utility profiles can take arbitrary shapes on houly resultion; *Rushhour* and *Worknight* are most challenging, since harvest and utility mismatch; *Rushhour* is also used for case-study.

example application is a gas sensor at outside work plants, e.g., harbors. Here, during normal work hours, the interest is high to ensure safe working conditions for employees while interest outside the working hours is of limited interest.

The *Worknight* profile is the most demanding for energy managers working on solar-harvesting system since application interest is misaligned with harvest. An example application is a surveillance camera system. During the day, workers are present, so the risk by burglars is limited, while nighttime is lonely, hence gathering images is important.

The *Rushhour* profile is used by the VAM case study. Here, early mornings and evenings are of most interest. This applies also to traffic surveillance systems since traffic is highest during these times. For energy managers, providing high utility in early morning is usually most challenging since the energy storage is at very low levels.

7.4 Results

In this section, the following investigations are showcased:

- The influence on SoC curve and saturation times (Section 7.4.1).
- The interplay of storage capacity and absolute and relative utility (Section 7.4.2).
- The effect of utility profiles on different energy managers (Section 7.4.3).
- The impact of non-ideal estimation harvest forecast (Section 7.4.4).
- A case study for structural health monitoring on bridges (Section 7.4.5).
- The limitations of `EmRep` (Section 7.4.6).

7.4.1 Benefits of Decoupling

To highlight the benefits of decoupling low- and high-energy intake periods for energy management, SoC curves and resulting utilities are analyzed in Fig. 7.5. All energy managers are compared with the same `EWMA` filtered harvest estimation for the presented dataset. The day is divided into $S = 24$ slots; hence the accuracy of forecast and utility calculation is 60 min. For completeness, the harvest at the solar panel of the four-day excerpt is shown as well. The size of the capacitor is 50 F.

At the beginning of day 35, all energy manager share a nearly equal utility of around $U = 0.15$ which is limited by the amount of energy stored capacitor. The latter was fully charged before and since no harvest is expected during nighttime, equals an average consumption of $I_\mathrm{n} = 0.45\,\mathrm{mA}$. Since the predicted harvest is slightly lower than expected, all managers have to correct their average consumption slightly immediately following the minimum SoC.

After passing the minimum, position (a), `EmRep` evaluates the choice of B again to reach the maximum at 17 h, position (b). Since SoC is still low at 0.1, B is even lower than before to the minimum. This, however, changes with increasing harvest at which B can be adjusted so that an average utility of $U = 0.32$ between 11 – 17 h is reached. Still, after 17 h at day 35, position (b), the utility during the night is en par with the other energy managers since the SoC is similar late in the afternoon.

The same trend follows for day 36 to 38: the time with full energy storage roughly decreases by 1 h per day. This translates to a reduction of saturation time by 20% compared to the five hours of `DS`. More importantly, the ability to optimize B only towards the end of high harvest phase at 18 h allows `EmRep` to increase utility to $U = 0.63$ between 11 – 18 h at day 37. At the same time, `PreAct` averages $U = 0.2$ and `DS` achieves $U = 0.17$.

This advantage also manifests when comparing the saturation times with different energy managers. In Fig. 7.6 the number of hours in the 194 d lasting data set which are spent at full

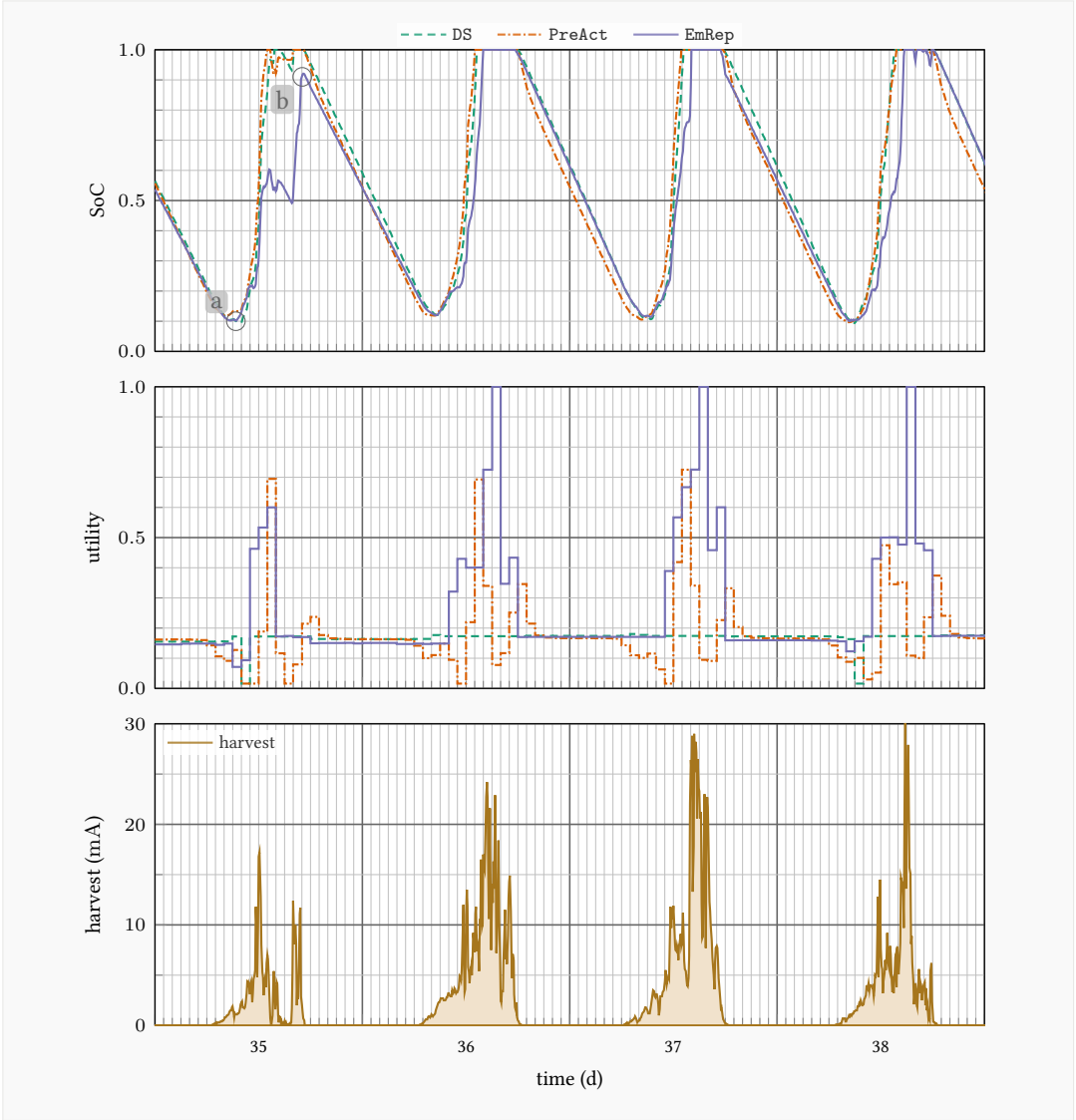
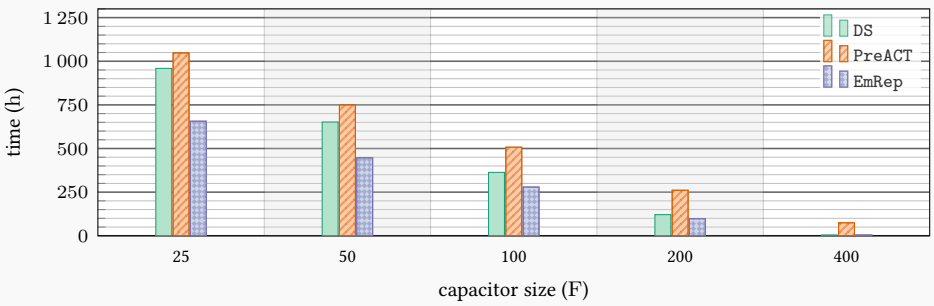
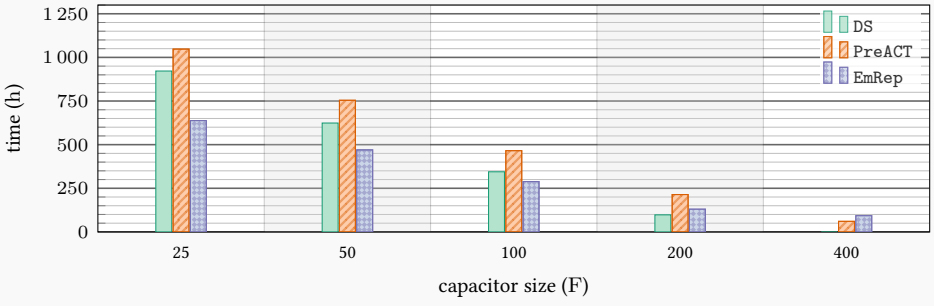


Figure 7.5: Comparison of SoC curve and utility for different energy managers; simulation uses `EWMA` estimation with $S = 24$ and $C = 50\,\mathrm{F}$; `DS` shows nearly constant utility while `PreAct` pushing towards the benefits of surplus energy; still `EmRep` is the only energy manager that achieves full utility together with higher average utility.



(a) Without utility profile



(b) *Workday* utility

Figure 7.6: Average saturation duration of energy storage for small capacitors significantly reduced; results include various estimation methods, slot sizes with *Const* profile; with 25 F capacitor, `EmRep` decreases time with saturated storage at minimum by 31.5%.

storage are depicted. The effect of reducing saturation by design is clearly visible: `EmRep` decreases time in saturation by up to 37.3% at 25 F compared to `PreAct`. This yields an absolute reduction of 391 h or an average of 2.01 h/d. As storage size increases, the problem of saturation lessens, e.g., at 200 F `EmRep` reduces saturation time by 19.1% compared to `DS`. Next, the influence of storage size on utility is discussed.

7.4.2 Influence of Storage Capacity

Absolute Results

The influence of supercapacitor capacity on the effectiveness of the different energy managers is depicted in Fig. 7.7. The ordinate shows the average of the slot-wise computed effective utility. The different subplots show an increasing number of timeslots, which affects utility and estimation accuracy as well as management reactivity. Figure 7.7 showcases the situation with ideal slot-wise estimation. This means that energy managers work on the correct slot average; however misalignment of harvest throughout the timeslot can influence the results, c.f. Fig. 2.4. For example, the slot average can be equal if high harvest arrives late in the slot compared

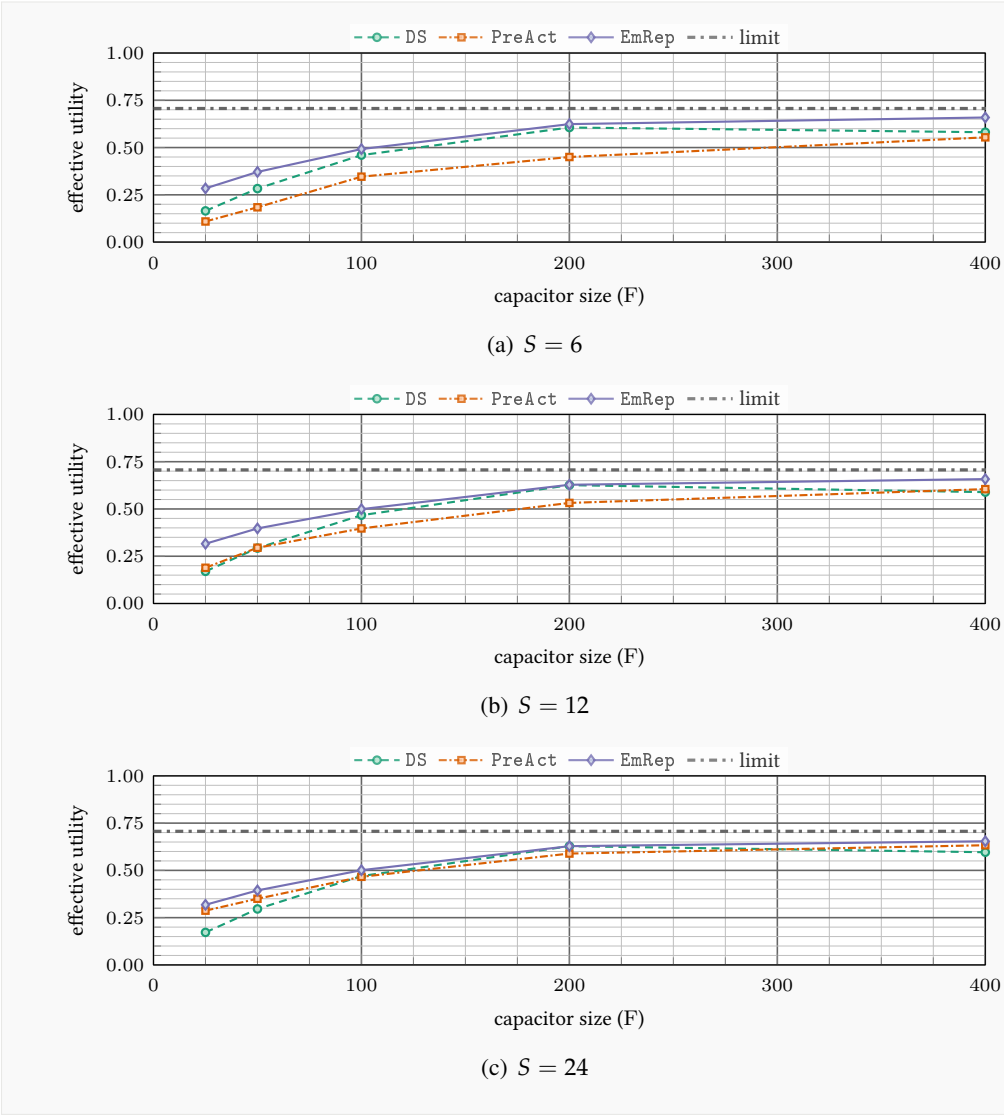


Figure 7.7: Average effective utility with different storage capacity and timeslot lengths with ideal slot-wise estimation; *Const* utility is required throughout the day; `EmRep` outperforms `PreAct` and `DS` at long timeslots and small-sized storage; benefits diminish at 400 F and $S = 24$; best performance should meet the practical limit (average harvest power).

to stable (lower) harvest throughout the slot—the effect on the sensor node, however can be totally different since it could have been shut down before the end of the slot. This is especially critical at early mornings as SoC is at a very low level. A realistic limit is also depicted in the simulations: as stated in Section 4.2, the capacitor voltage influences the power point of the solar panel. The limit shows the average power which could have been generated at the solar panel if the SoC was always soc $= 0.5$. The closer an energy manager is able to get to this line, the better the performance is.

Clearly, only increasing the energy storage does not linearly increase the effective utility. Increasing the capacity by eight, yields an increase in utility of 2.03× for `EmRep`, of 2.41× for `PreAct`, and 3.60× for `DS` at $S = 24$. Interestingly, both `PreAct` and `EmRep` show an increase up until a capacity of 200 F. Doubling the storage capacity additionally, does not increase the utility effectively for all energy managers. The reason behind this is the size, and the power respectively, of the solar panel. The small 35×35 mm panel does not suffice to recharge the capacitor completely; hence no benefit in utility can be seen. This underlines again that solely increasing the storage capacity—over-dimensioning—only makes sense when the size of the solar panel is increased as well.

The number of timeslots, however, has a minor effect for `EmRep` and `DS`. Although the slot length increases from 1 h to 4 h, the effective utility only decreases by 7% for `DS` and 2% for `EmRep` for a 100 F capacitor. `PreAct` on the other hand is sensitive to reduction in slot length which manifests in a loss of 32%. This, however is mainly caused by extreme reactions of the underlying PID-controller. These high fluctuations in duty cycle can be corrected with short timeslots and high-capacity storage, c.f. Fig. 7.7(c), but otherwise can have severe consequences. Adjusting the PID parameters accordingly does not improve results; a more detailed investigation may improve the performance of `PreAct`. However, these configurations are not ideally suitable since `PreAct` is mainly designed for long-term ENO systems with large-sized storage.

Relative Results

Typically, small-sized energy storage saturates during times of high energy intake. With `EmRep`, mitigating this effect and enhance the effective utility is a central goal. Hence, the influence of capacitor size on the performance of energy managers is highlighted in Fig. 7.8. Here, the difference in effective utility in the simulation setup is shown and compared to the reference, a 200 F capacitor. The 200 F capacitor is chosen as reference, since increasing the storage further offers only little improvements for two of the three energy managers.

Across all simulation runs, the energy manager loose between 13.6% and 60.3% of their performance when decreasing the size of the capacitor from 200 F to 25 F. However, while

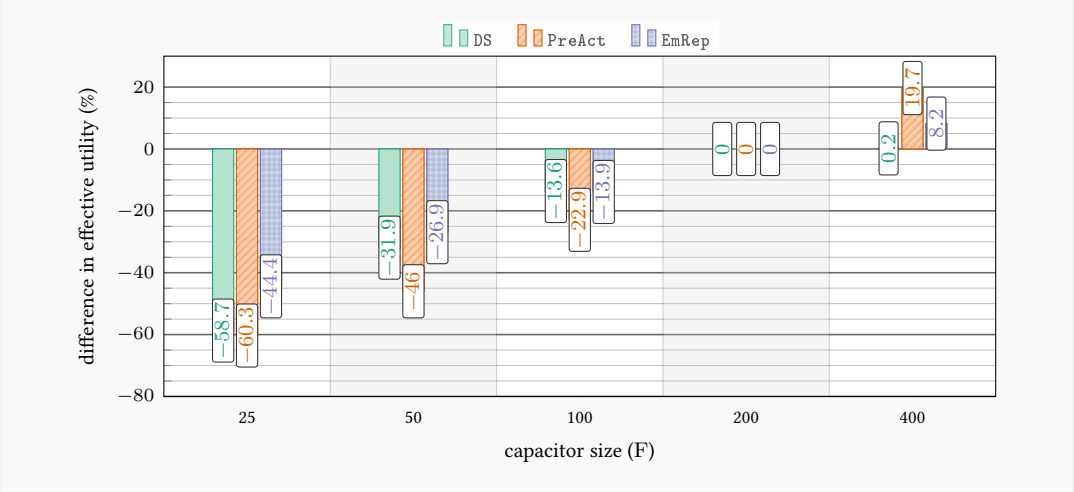


Figure 7.8: Relative difference in utility, compared to 200 F capacitor; `EmRep` looses significantly less utility with capacitors smaller than 100 F.

`DS` and `PreAct` loose 58.7% and 60.3%, `EmRep` only looses 44.4%. Still, `EmRep` keeps an average utility level of 0.36 which is 1.42× higher than `DS` and 1.7× higher than `PreAct`.

Interestingly, `DS` is not able to improve the utility in use of a 400 F capacitor. This is a direct consequence of the depletion safe algorithm, which tends to operate at lower SoC levels. In a direct charge circuit, the operating point of the solar panel is hence suboptimal and the generated power is lower. `PreAct`, however, tends to keep high SoC levels which enables the system to profit from over-sized storage. `EmRep` sits in the middle; building up on `DS` it also tends to favor low SoC but profiting also from non-saturating storage.

7.4.3 Utility Profiles

Next, the influence of the different utility profiles are presented in Section 7.3.3 and the resulting utility is showcased in Fig. 7.9. Each bar incorporates 60 different simulation configurations with varying estimator, capacitor size, and number of time slots.

The baseline for the other profiles is the *Const* profile. Here, `EmRep` outperforms both `DS` and `PreAct`. In this profile, each time of the day is equally important; hence surplus activity during daytime is beneficial. However, as visible by the performance of `PreAct`, spending available energy too greedily harms the overall performance.

The performance of the *Worknight* profile shows how well energy manager are able to maintain activity during nighttime. Here, the `DS` algorithm as a base serves very well since it aims at maintain equal performance level throughout the day. Although this profile structure contrasts the design of `EmRep`—being designed to use surplus energy—it still matches or outperforms the utility level of `DS`. However, increasing the utility with *Worknight* profile requires a larger cap; e.g., with a 50 F capacitor `DS` and `EmRep` both reach at utility of 0.48.

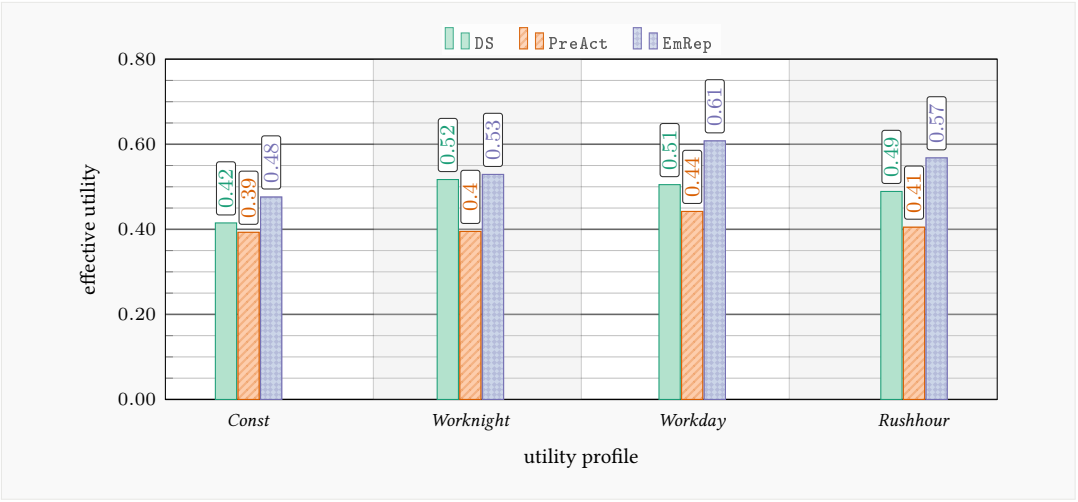


Figure 7.9: Mean utility with different profiles; mean includes capacitor sizes, various estimation methods and slot length included; `EmRep` profits most from valued activity at daytime; `EmRep` matches performance of `DS` at suboptimal profile *Worknight* and outperforms `PreAct`.

This image changes with a 400 F capacitor at which `DS` reaches $U = 0.6$ and `EmRep` achieves 0.65. Please note, that in this profile, saturation times are generally higher since surplus energy during daytime is not used to increase activity

The *Workday* and *Rushhour* profiles are ideally suited for algorithms that make use of surplus energy. `EmRep` is able to increase utility by 0.13 (*Workday*) and 0.10 (*Rushhour*). Also `PreAct` reaches its best performance here; although it still suffers to match the performance of `DS`.

7.4.4 Estimation Influence

Since a fundamental basis of `EmRep` is the time of SoC extrema, the used harvest estimation method is important for the overall performance. Simulation results for three different short-term estimation methods for solar energy harvesting are conducted and depicted in Fig. 7.10. For comparison, the results with original data, i.e., ideal estimation, are also shown. However, even the ideal estimation uses slot averages hence the deviation within the time slot is still affecting results.

Under ideal conditions but also with all used estimation methods, `EmRep` maintains the top-end performance across the simulation configurations. `WCMA` and `EWMA` only produce a small drop in utility—for all energy managers—but with `Pro-Energy` estimation, the performance drops notably. This is due to the fact that the estimation of `Pro-Energy` may completely change throughout the day. Out of a pool of similar harvest profiles, `Pro-Energy` continuously selects the profile most similar to the already observed profile. While this

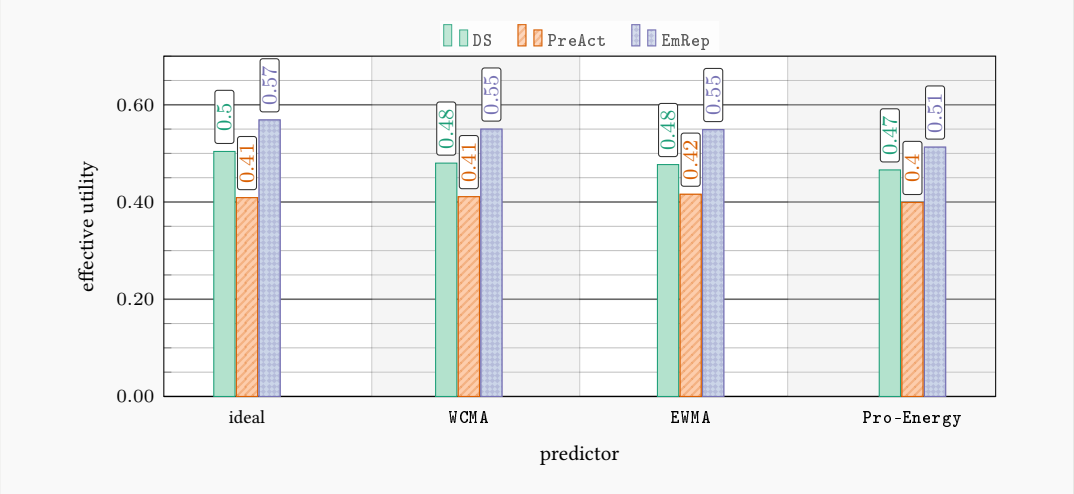


Figure 7.10: Mean utility with different estimators; influence of estimation algorithm is minor, implicating that simple algorithms (`EWMA`) may be favored; although `EmRep` performs worst in combination with `Pro-Energy`, it still tops performance of other energy managers with ideal estimation.

decreases the estimation error, it also changes the SoC curve completely; hence `EmRep` works suboptimal. This is en-par with findings presented with `Pro-Energy` where error rates increase with increasing forecast horizon above 2.5 h.

Interestingly, `PreAct` shows a very stable performance across estimation methods and is even capable of improving its performance with the use of an `EWMA`-filtered estimation. This effect occurs with long timeslots, i.e., 4 h, at which harvest deviation has a high impact. In early mornings, the harvest may rise only at the very end of the timeslot which increases the slot average. Since `PreAct` often uses a vast part of incoming energy directly, it suffers from shut-downs in early mornings with little capacitors. This effect is softened by an `EWMA` filter since the slot average generally is lower; and hence also the used budget.

In general, the influence of the estimation algorithm is limited for all energy manager. Although estimation-error-centric work founds difference in Mean Absolute Percentage Error (MAPE), the actual implications onto a deployed system remains unclear. This is due to the fact that estimation errors frequently cancel out throughout the day; i.e., the implication for the SoC is smaller than the MAPE. Hence, the presented investigations imply that simple estimation methods suffice; even if their MAPE is higher.

7.4.5 Case Study: Vibro-acoustic Modulation

Structural health monitoring of bridges or buildings offers great insights to aging or possible defects of materials but entails high installation cost of wired power supply and high maintenance effort. Supplying these sensor nodes with solar energy lowers maintenance costs with

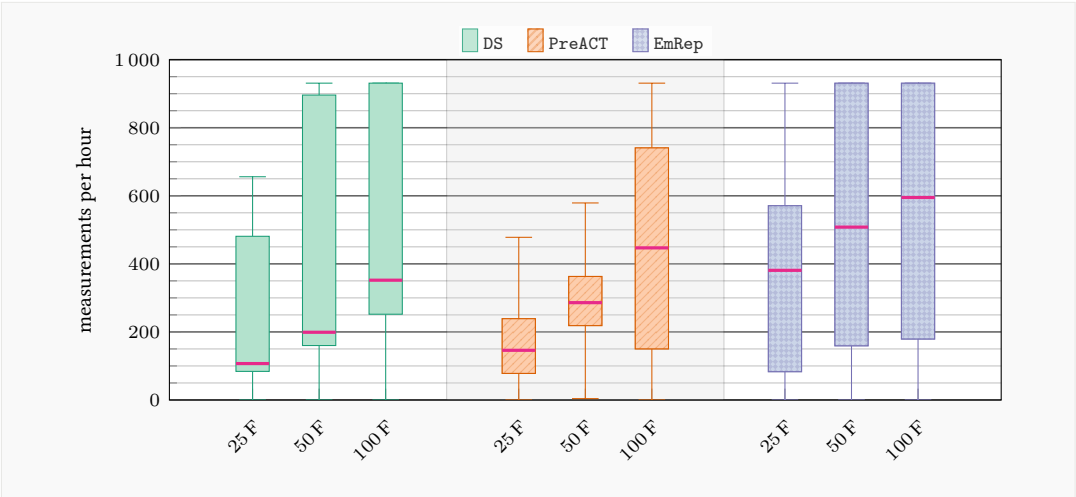


Figure 7.11: Boxplot for VAM case study; *Rushhour* utility profile with `EWMA` estimation with $S = 24$; even with a 50 F capacitor `EmRep` outperforms `DS` and `PreAct` in median measurements per hours; for 25 F `EmRep` increases measurements per hour by at least 2.5×.

low impact on measurement performance. VAM, uses piezoelectric inducers to generate high frequency pulses. The measurement method requires structural vibration at low frequencies, which are typically caused by cars or trains. Hence, only with a combination of low and high frequency pulses, useful measurements can be obtained. Therefore, it is ideally suited for the *Rushhour* utility profile, at which energy is spent when high traffic, and hence low frequency vibrations, is present. Each measurement contains stimulation of different frequencies typically lasting 100 ms each. Good quality results can be obtained with 50 different frequencies; but the quality of the measurements strongly increases with number of tested frequencies.

The deployed piezoelectric transducer uses sinusoidal signals with amplitude up to 12 V. The signal generator typically draws a current of 20 mA. These parameters are fed into the simulation framework and the results are highlighted in Fig. 7.11. For simplicity, only hourly resolution and simple harvest estimation with `EWMA` is simulated. The box plot shows the number of individual frequency measurements per hour for different capacitor sizes. With the small 25 F capacitor, `DS` achieves 110 measurements in 50% of the cases whilst `PreAct` provides 280 and `EmRep` 380 measurements. Hence, `EmRep` enables the VAM method either to increase frequency depth by 35% or up to two additional complete readings with 50 frequencies each, compared to `PreAct`. Even more interestingly, `EmRep` achieves a higher median performance with a 50 F capacitor than `DS` and `PreAct` with a 100 F capacitor. This allows for a much smaller package—supercapacitor volume usually scale linearly with capacity—without loosing performance.

7.4.6 Limitations and Discussion

EmRep shows a strong performance across various harvest estimation patterns, slot size, utility profile and capacitor size. The presented investigations, however, also reveal deficiencies. The performance benefit in the *Worknight* profile is limited, compared to DS. This is dominated by diminishing benefits at small size storage, although EmRep is tailored to small size storage. However, buffering energy for the night requires a certain size, e.g., a 50 F capacitor only allows for an average budget of 0.8 mA during the night. The options of shifting energy here are very limited. Furthermore, investigations on estimation methods reveal that EmRep needs an estimator with stable estimation for T_{H}. In particular, Pro-Energy which achieves very low MAPE for forecasts up until 2 h, is no good partner for EmRep since varying extrema estimation hurts performance. Also, large-size storage let the benefits of EmRep fade: relatively small solar cells combined with large capacitors lead to low variations in SoC. If energy storage is held at 50% of SoC, all incoming energy can be stored and no energy surplus can be used for increasing utility additionally.

7.5 Conclusion

This chapter presented EmRep, a new energy manager for ENO systems. It uses SoC extrema prediction to decouple energy management in phases where depletion safety is important from phases where energy surplus exists. This allows EmRep to increase effective utility across a plethora of combinations, e.g., doubling utility with 25 F and long timeslots. Moreover, the VAM case study showed that measurement frequency can be increased by 2.5× for small size storage. EmRep paves the ground for smaller sensor nodes powering the EH-WSN. In future work, tests of EmRep with different harvest sources and real-world test at harbor bridges can be performed.

Chapter 8

Conclusion

8.1 Summary

Wireless Sensor Networks collect information about their surroundings and, hence, offer fine-grained insights without prior installation of costly infrastructure. Moreover, sensors nodes that blend into Cyber-physical Systems, are a central driver of the fourth industrial revolution and aid applications such as autonomous driving. As the number of nodes increases, supplying these sensors by wires or disposable batteries is neither economically nor environmentally feasible. Hence, energy harvesting—which recharges sensor nodes to prolong lifetime—draws considerable research but struggles to cope with imposed challenges of future research topics.

This dissertation has identified main challenges that need to be addressed: depletion safety for perpetual operation as key requirement but especially timely gathering of sensor values and incorporation of user-defined utility. For a prototype harvester with solar panel and supercapacitor, software for efficient energy management has been implemented and demonstrated. In this dissertation, scheduling and management algorithms for energy-harvesting sensor nodes have been evaluated in detailed simulations with practical relevancy and real-world tests have been conducted. Furthermore, a lighting testbed that helps developers to study behavior of their energy-aware algorithms has been demonstrated and evaluated. This dissertation, hence, addressed the following challenges and exceeds existing literature: support for task-based program structures and incorporation of Quality-of-Service into energy management.

Up until now, no scheduling approach existed that translates graph- and task-based program structures with time-constraints into schedules for—carefully planning—energy management of energy-neutral sensor nodes. Task-based structures split up programs in subparts which are otherwise glued together in hard-to-maintain source code; edges between tasks allow the programmer to describe time relations between tasks. The work in this dissertation has simplified program definition by providing developers a tool to describe time relations between tasks—and to enforce them as basis for energy-aware scheduling. A dedicated toolkit helps

to assess performance prior to deployment by detailed simulations in coherence with the prototype harvesting platform and hence accelerates system development.

Moreover, this dissertation has contributed to the active research topic Quality-of-Service; especially targeted at small-scale sensor nodes equipped with solar harvesters and supercapacitors. The presented energy manager `EmRep` is the first that uses user-defined utility profiles to enhance energy management on hourly-basis for short-term energy-neutral devices. `EmRep` allows the application developers to express times of interest and proactively adjusts activity to increase to benefit for the application. This dissertation, hence, has helped energy-harvesting sensor networks to be programmed and operated more easily. The work has enabled program designers to express timeliness of sensor values and Quality-of-Service constraints which paves the ground for future applications powered by ambient energy. The approaches in this dissertation have outreached existing literature for short-term energy-neutral system equipped with solar harvester and supercapacitors and hence advance public knowledge on energy-efficient operation of Wireless Sensor Networks. Additionally, the developed algorithms and tools are available open-source and will, as such, enrich application variety for energy-harvesting systems.

8.2 Outlook

This dissertation has shown that energy efficiency of sensor networks is a still-to-solve major challenge. It has been outlined that future applications rely on depletion safety, timely gathered sensor values, easy-to-maintain programs and user-defined utilities. However, the unpredictable nature of ambient energy still entails that guarantees on activity levels are hard to give. This hinders spreading of energy harvesting in more applications, e.g., operators of production plant have to monitor pollution continuously—otherwise, they are fined by authorities. As computation power of sensor nodes increases, they may monitor their activity and apply computational-complex statistical models. Right now, Quality-of-Service is included into scheduling and energy management, but sudden weather changes throw plans overboard. If, however, nodes learn from previously happened weather changes they can offer a likelihood for future service levels. This makes their activity more predictable which also allows other nodes to adapt their own scheduling, e.g., to decrease communication rendezvous time. As consequence, cooperation in the network can be increased since communication is more efficiently.

Furthermore, applications that rely on Artificial Intelligence (AI) can be implemented for Energy-harvesting Wireless Sensor Networks. Since the duty of sensor nodes shift from transmitting *raw data* to transmitting *information*, machine learning can help on decision. Models obtained from learned patterns, e.g. sensor value oddities, increase the accuracy of

decision. At first, these models have to be learned offline, but ideally those model are directly build online.

Moreover, the upcoming class of energy harvesting devices—Intermittent Computing—should be actively integrated. Concepts, such as strategic energy management can be transferred and hence increase scheduling efficiency for intermittently-powered devices. Furthermore, energy-neutral systems are a great communication partner for Intermittent Computing devices, since Intermittent Computing devices still require an always reachable gateway in near reach. Since energy-neutral systems generally have more energy to spend, they are an ideal counterpart as portable gateway, e.g., equipped with 5G technology. Together, new applications such as on-body monitoring solely powered by ambient energy are imaginable. Consequently, more devices equipped with disposable batteries can be replaced with harvesting devices, and the environmental footprint of future Wireless Sensor Networks can be lowered.

Bibliography

[AA20] Alaa Attar and Faisal Albatati. Wearable Thermoelectric Generators as Energy Harvesters for Wireless Body Sensors. *International Journal of Energy and Environmental Engineering (IJEEE)*, 12, November 2020. http://dx.doi.org/10.1007/s40095-020-00365-x.

[ABA+19] Saad Ahmed, Naveed Bhatti, Hamad Alizai, Junaid Siddiqui, and Luca Mottola. Efficient Intermittent Computing with Differential Checkpointing. In *Proceedings of the 20th International Conference on Languages, Compilers, and Tools for Embedded Systems*, LCTES '19, Phoenix, AZ, US, June 2019. http://dx.doi.org/10.1145/3316482.3326357.

[ABB+19] Saad Ahmed, Abu Bakar, Naveed Anwar Bhatti, Muhammad Hamad Alizai, Junaid Haroon Siddiqui, and Luca Mottola. The Betrayal of Constant Power× Time: Finding the Missing Joules of Transiently-powered Computers. In *Proceedings of the 20th International Conference on Languages, Compilers, and Tools for Embedded Systems*, LCTES '19, Phoenix, AZ, US, June 2019. http://dx.doi.org/10.1145/3316482.3326348.

[ABD+19] Rehan Ahmed, Bernhard Buchli, Stefan Draskovic, Lukas Sigrist, Pratyush Kumar, and Lothar Thiele. Optimal Power Management with Guaranteed Minimum Energy Utilization for Solar Energy Harvesting Systems. *Transactions on Embedded Computing Systems (TECS)*, 18, June 2019. http://dx.doi.org/10.1145/3317679.

[AGJ+18] Joshua Adkins, Branden Ghena, Neal Jackson, Pat Pannuto, Samuel Rohrer, Bradford Campbell, and Prabal Dutta. The Signpost Platform for City-scale Sensing. In *Proceedings 17th International Conference on Information Processing in Sensor Networks*, IPSN '18, Porto, Portugal, April 2018. http://dx.doi.org/10.1109/ipsn.2018.00047.

[AIM10] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of Things: A Survey. *Computer Networks*, 54(15), October 2010. http://dx.doi.org/10.1016/j.comnet.2010.05.010.

[AMAT+18] Kofi Sarpong Adu-Manu, Nadir Adam, Cristiano Tapparello, Hoda Ayatollahi, and Wendi Heinzelman. Energy-Harvesting Wireless Sensor Networks (EH-WSNs). *Transactions on Sensor Networks (TOSN)*, 14(2), July 2018. http://dx.doi.org/10.1145/3183338.

[AMMW11] David Audet, Neil MacMillan, Dimitri Marinakis, and Kui Wu. Scheduling Recurring Tasks in Energy Harvesting Sensors. In *The 30th International Conference on Computer Communications Workshops*, INFOCOM WKSHPS '11, Shanghai, China, April 2011. http://dx.doi.org/10.1109/infcomw.2011.5928823.

[ATGC+15] R. Ambrosio, R. TorreAlba, J. F. Guerrero-C, V. González, A. Limon, and M. Moreno. Energy Harvesting Combining Three Different Sources For Low Power Applications. In *Proceedings of the 12th International Conference on Electrical Engineering, Computing Science and Automatic Control*, CCE '15, Mexico City, Mexico, October 2015. http://dx.doi.org/10.1109/ICEEE.2015.7357943.

[ATMA18] Faisal Ahmed, Gert Tamberg, Yannick Le Moullec, and Paul Annus. Adaptive LINE-P: An Adaptive Linear Energy Prediction Model for Wireless Sensor Network Nodes. *Sensors*, 18(4), April 2018. http://dx.doi.org/10.3390/s18041105.

[Aut14] Hamburg Port Authority. Hamburger Hafen—Digitales Tor zur Welt, 2014. https://www.iaph2015.org/downloads//smartPORT-Brosch%C3%BCren/broschuere_smartportlogistics_web.pdf. Last accessed: 2021/05/02.

[BEMRB13] Matthias Budde, Rayan El Masri, Till Riedel, and Michael Beigl. Enabling Low-cost Particulate Matter Measurement for Participatory Sensing Scenarios. In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia*, MUM '13, Lulea, Sweden, December 2013. http://dx.doi.org/10.1145/2541831.2541859.

[BFB+12] Heather A. Barrett, Alyssa Ferraro, Chris Burnette, Amanda Meyer, and Mark P. S. Krekeler. An Investigation of Heavy Metal Content from Disposable Batteries of Non-U.S. Origin from Butler County, Ohio: An Environmental Assessment of a Segment of a Waste Stream. *Journal of Power Sources*, 206, May 2012. http://dx.doi.org/10.1016/j.jpowsour.2012.01.008.

[BM17] Naveed Anwar Bhatti and Luca Mottola. HarvOS: Efficient Code Instrumentation for Transiently-Powered Embedded Sensing. In *Proceedings of the 16th International Conference on Information Processing in Sensor Networks*, IPSN '17, Pittsburgh, PA, USA, April 2017. http://dx.doi.org/10.1145/3055031.3055082.

[BMAS19] Adriano Branco, Luca Mottola, Muhammad Hamad Alizai, and Junaid Haroon Siddiqui. Intermittent Asynchronous Peripheral Operations. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, SenSys '19, New York, NY, USA, November 2019. http://dx.doi.org/10.1145/3356250.3360033.

[BSBT14a] Bernhard Buchli, Felix Sutton, Jan Beutel, and Lothar Thiele. Dynamic Power Management for Long-term Energy Neutral Operation of Solar Energy Harvesting Systems. In *Proceedings of the 12th Conference on Embedded Network Sensor Systems*, SenSys '14, Memphis, TN, USA, November 2014. http://dx.doi.org/10.1145/2668332.2668333.

[BSBT14b] Bernhard Buchli, Felix Sutton, Jan Beutel, and Lothar Thiele. Towards Enabling Uninterrupted Long-Term Operation of Solar Energy Harvesting Embedded Systems. In *Proceedings of the 11th European Conference on Wireless Sensor Networks*, EWSN '14. Porto, Portugal, February 2014. http://dx.doi.org/10.1007/978-3-319-04651-8_5.

[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. *Embedded Systems Letters*, 7(1), March 2015. http://dx.doi.org/10.1109/LES.2014.2371494.

[BXNX19] Hilal Bello, Zeng Xiaoping, Rosdiadee Nordin, and Jian Xin. Advances And Opportunities In Passive Wake-up Radios With Wireless Energy Harvesting For The Internet Of Things Applications. *Sensors*, 19(14), July 2019. http://dx.doi.org/10.3390/s19143078.

[CL16] Alexei Colin and Brandon Lucia. Chain: Tasks and Channels for Reliable Intermittent Programs. In *Proceedings of the International Conference on Object-Oriented Programming, Systems, Languages, and Applications*, OOPSLA '16, Amsterdam, The Netherlands, November 2016. http://dx.doi.org/10.1145/3022671.2983995.

[CL18] Alexei Colin and Brandon Lucia. Termination Checking and Task Decomposition for Task-Based Intermittent Programs. In *Proceedings of the 27th International Conference on Compiler Construction*, CC '18, New York, NY, USA, February 2018. http://dx.doi.org/10.1145/3178372.3179525.

[CLW+13] Dan Chen, Zhixin Liu, Lizhe Wang, Minggang Dou, Jingying Chen, and Hui Li. Natural Disaster Monitoring with Wireless Sensor Networks: A Case Study of Data-intensive Applications upon Low-Cost Scalable Systems. *Mobile Networks and Applications*, 18(5), August 2013. http://dx.doi.org/10.1007/s11036-013-0456-9.

[CPS16] Alessandro Cammarano, Chiara Petrioli, and Dora Spenza. Online Energy Harvesting Prediction in Environmentally Powered Wireless Sensor Networks. *Sensors Journal*, 16(17), September 2016. http://dx.doi.org/10.1109/jsen.2016.2587220.

[Cre17] Cree. *Datasheet Cree XLamp MK-R LEDs*, 2017. http://www.cree.com/led-components/media/documents/XLampMKR.pdf. Last accessed: 2021/05/02.

[CRL18] Alexei Colin, Emily Ruppel, and Brandon Lucia. A Reconfigurable Energy Storage Architecture for Energy-harvesting Devices. In *Proceedings 23rd International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '18, Williamsburg, VA, USA, March 2018. http://dx.doi.org/10.1145/3296957.3173210.

[DALT18] Stefan Draskovic, Rehan Ahmed, Cong Lin, and Lothar Thiele. A Case For Atmospheric Transmittance: Solar Energy Prediction In Wireless Sensor Nodes. In *Proceedings of the International Conference on Green Computing and Communications*, GreenCom '18, Halifax, NS, Canada, July 2018. http://dx.doi.org/10.1109/cybermatics_2018.2018.00097.

[dFCMV20] Roberto de Fazio, Donato Cafagna, Giorgio Marcuccio, and Paolo Visconti. Limitations and Characterization of Energy Storage Devices for Harvesting Applications. *Energies*, 13(4), February 2020. http://dx.doi.org/10.3390/en13040783.

[DFPC08] Prabal Dutta, Mark Feldmeier, Joseph Paradiso, and David Culler. Energy Metering for Free: Augmenting Switching Regulators for Real-time Monitoring. In *Proceedings 7th International Conference on Information Processing in Sensor Networks*, IPSN '08, St. Louis, MO, USA, April 2008. http://dx.doi.org/10.1109/ipsn.2008.58.

[DHM75] Jitendra V. Dave, P. Halpern, and H. J. Myers. Computation of Incident Solar Energy. *IBM Journal of Research and Development*, 19(6), November 1975. http://dx.doi.org/10.1147/rd.196.0539.

[Dur14] Duracell. *Datasheet Duracell Ultra Power MX 1500 AA (LR6) Battery*, 2014. https://docs.rs-online.com/2a27/0900766b814ef4c0.pdf. Last accessed: 2021/05/02.

[EnO20] EnOcean. *Datasheet PTM21x Pushbutton transmitter modules*, July 2020. https://www.enocean.com/de/produkte/enocean_module/ptm-210ptm-215/data-sheet-pdf/. Last accessed: 2021/05/02.

[ESC16] Mahmoud Elkhodr, Seyed Shahrestani, and Hon Cheung. Emerging Wireless Technologies in the Internet of Things: A Comparitive Study. *International Journal of Wireless and Mobile Networks (IJWMN)*, 8(5), October 2016. http://dx.doi.org/10.5121/ijwmn.2016.8505.

[Esp] Espressif Systems. *Datasheet Espressif ESP8266EX*. https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf. Last accessed: 2021/05/02.

[FMftENSB19] Nature Conservation Federal Ministry for the Environment and Germany Nuclear Safety (BMU). National Air Pollution Control Programme, May 2019. https://www.umweltbundesamt.de/sites/default/files/medien/1410/dokumente/luftreinhalteprogramm_bericht_bf.pdf. Last accessed: 2021/03/21.

[FNSSk21] Mahsa Z. Farahmand, M. E. Nazari, S. Shamlou, and Miadreza Shafie-khah. The Simultaneous Impacts of Seasonal Weather and Solar Conditions on PV Panels Electrical Characteristics. *Energies*, 14(4), February 2021. http://dx.doi.org/10.3390/en14040845.

[fVudIB] Bundesministerium für Verkehr und digitale Infrastruktur (BMVI). Automatisiertes und vernetztes Fahren. http://www.bmvi.de/SharedDocs/DE/Artikel/DG/automatisiertes-und-vernetztes-fahren.html. Last accessed: 2021/05/02.

[GCC11] Hussein El Ghor, Maryline Chetto, and Rafic Hage Chehade. A Real-time Scheduling Framework for Embedded Systems with Environmental Energy Harvesting. *Computers & Electrical Engineering*, 37(4), July 2011. http://dx.doi.org/10.1016/j.compeleceng.2011.05.003.

[GKJZ19] Kai Geissdoerfer, Brano Kusy, Raja Jurdak, and Marco Zimmerling. Getting More Out of Energy-harvesting Systems: Energy Management under Time-varying Utility with PREAcT. In *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*, IPSN '18, Porto, Portugal, April 2019. http://dx.doi.org/10.1145/3302506.3310393.

[GRHJ+18] Andrea Gaglione, David Rodenas-Herraiz, Yu Jia, Sarfraz Nawaz, Emmanuelle Arroyo, Cecilia Mascolo, Kenichi Soga, and Ashwin A. Seshia. Energy Neutral Operation of Vibration Energy-harvesting Sensor Networks for Bridge Applications. In *Proceedings of the International Conference on Embedded Wireless Systems and Networks*, EWSN '18, Madrid, Spain, February 2018. http://dx.doi.org/10.5555/3234847.3234849.

[HHR16] Lars Hanschke, Jan Heitmann, and Christian Renner. Challenges of WiFi-Enabled and Solar-Powered Sensors for Smart Ports . In *Proceedings of the 4th International Workshop on Energy Neutral Sensing Systems*, ENSsys '16, Stanford, CA, USA, November 2016. http://dx.doi.org/10.1145/2996884.2996887.

[HHR17] Lars Hanschke, Jan Heitmann, and Christian Renner. Stop Waiting: Mitigating Varying Connecting Times for Infrastructure WiFi Nodes. In *Proceedings of the 16th GI/ITG KuVS Fachgespräch "Sensornetze"*, FGSN '17, Hamburg, Germany, October 2017. http://dx.doi.org/10.48441/4427.4.

[HR18] Lars Hanschke and Christian Renner. Time- and Energy-aware Task Scheduling in Environmentally-powered Sensor Networks. In *Proceedings of the 14th International Symposium on Algorithms and Experiments for Wireless Networks* , AlgoSensors '18, Helsinki, Finnland, August 2018. http://dx.doi.org/10.1007/978-3-030-14094-6_9.

[HR20] Lars Hanschke and Christian Renner. Scheduling Recurring And Dependent Tasks In EH-WSNs. *Sustainable Computing: Informatics and Systems (SUSCOM)*, 27, September 2020. http://dx.doi.org/10.1016/j.suscom.2020.100409.

[HR22] Lars Hanschke and Christian Renner. EmRep: Energy Management Relying on State-of-Charge Extrema Prediction. *IET Computers & Digital Techniques*, 16(4), July 2022. http://dx.doi.org/10.1049/cdt2.12033.

[HRB+17] Lars Hanschke, Christian Renner, Jannick Brockmann, Tobias Hamann, Jannes Peschel, Alexander Schell, and Alexander Sowarka. Light Insight — Emulation of Radiation Traces for Analysis and Evaluation of Solar-Harvesting Algorithms. In *Proceedings of the 5th International Workshop on Energy Neutral Sensing Systems*, ENSsys '17, Delft, The Netherlands, November 2017. http://dx.doi.org/10.1145/3142992.3142994.

[HS17a] Josiah Hester and Jacob Sorber. Flicker: Rapid Prototyping For The Batteryless Internet-of-Things. In *Proceedings of the 15th Conference on Embedded Network Sensor Systems*, SenSys '17, New York, NY, USA, 2017. http://dx.doi.org/10.1145/3131672.3131674.

[HS17b] Josiah Hester and Jacob Sorber. New Directions: The Future of Sensing is Batteryless, Intermittent, and Awesome. In *Proceedings of the 15th Conference on Embedded Network Sensor Systems*, SenSys '17, Delft, The Netherlands, November 2017. http://dx.doi.org/10.1145/3131672.3131699.

[HSS15] Josiah Hester, Lanny Sitanayah, and Jacob Sorber. Tragedy of the Coulombs: Federating Energy Storage for Tiny, Intermittently-Powered Sensors. In *Proceedings of the 13th Conference on Embedded Network Sensor Systems*, SenSys '15, Seoul, South Korea, November 2015. http://dx.doi.org/10.1145/2809695.2809707.

[HSS17] Josiah Hester, Kevin Storer, and Jacob Sorber. Timely Execution on Intermittently Powered Batteryless Sensors. In *Proceedings of 15th Conference on Embedded Network Sensor Systems*, SenSys '17, Delft, The Netherlands, November 2017. http://dx.doi.org/10.1145/3131672.3131673.

[JC12] Jaein Jeong and David Culler. A Practical Theory of Micro-solar Power Sensor Networks. *Transactions on Sensor Networks (TOSN)*, 9(1), November 2012. http://dx.doi.org/10.1145/2379799.2379808.

[JPC05] Xiaofan Jiang, Joseph Polastre, and David Culler. Perpetual Environmentally Powered Sensor Networks. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, IPSN '05, Los Angeles, CA, USA, April 2005. http://dx.doi.org/10.1109/IPSN.2005.1440974.

[Kah62] Arthur Kahn. Topological Sorting of Large Networks. *Communications*, 5(11), November 1962. http://dx.doi.org/10.1145/368996.369025.

[KHZS07] Aman Kansal, Jason Hsu, Sadaf Zahedi, and Mani Srivastava. Power Management in Energy Harvesting Sensor Networks. *Transactions on Embedded Computing Systems (TECS)*, 6(4), September 2007. http://dx.doi.org/10.1145/1274858.1274870.

[KRCZ17] Yang Kuang, Tingwen Ruan, Zheng Jun Chew, and Meiling Zhu. Energy Harvesting During Human Walking to Power a Wireless Sensor Node. *Sensors and Actuators A: Physical*, 254, February 2017. http://dx.doi.org/10.1016/j.sna.2016.11.035.

[KYB+20] Vito Kortbeek, Kasim Sinan Yildirim, Abu Bakar, Jacob Sorber, Josiah Hester, and Przemysław Pawełczak. Time-sensitive Intermittent Computing Meets Legacy Software. In *Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '20, Lausanne, Switzerland, March 2020. http://dx.doi.org/10.1145/3373376.3378476.

[LBV06] K. Langendoen, A. Baggio, and O. Visser. Murphy Loves Potatoes: Experiences from a Pilot Sensor Network Deployment in Precision Agriculture. In *Proceedings of the 20th International Parallel & Distributed Processing Symposium*, IPDPS '06, April 2006. http://dx.doi.org/10.1109/ipdps.2006.1639412.

[LFS+12] Jason Jingshi Li, Boi Faltings, Olga Saukh, David Hasenfratz, and Jan Beutel. Sensing the Air We Breathe: The OpenSense Zurich Dataset. In *Proceedings of the 26th Conference on Artificial Intelligence*, AAAI '12, Toronto, Ontario, Canada, July 2012. https://ojs.aaai.org/index.php/AAAI/article/view/8163.

[LLCA18] Matthew Li, Jun Lu, Zhongwei Chen, and Khalil Amine. 30 Years of Lithium-Ion Batteries. *Advanced Materials*, 30(33), June 2018. http://dx.doi.org/10.1002/adma.201800561.

[MBO20] Xinyu Ma, Sebastian Bader, and Bengt Oelmann. Power Estimation for Indoor Light Energy Harvesting Systems. *Transactions on Instrumentation and Measurement*, 69(10), October 2020. http://dx.doi.org/10.1109/tim.2020.2984145.

[MBTB07] Clemens Moser, Davide Brunelli, Lothar Thiele, and Luca Benini. Real-time Scheduling for Energy Harvesting Sensor Nodes. *Real-Time Systems*, 37(3), December 2007. http://dx.doi.org/10.1007/s11241-007-9027-0.

[MCL17] K. Maeng, A. Colin, and B. Lucia. Alpaca: Intermittent Execution Without Checkpoints. In *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications*, OOPSLA '17, Vancouver, Canada, October 2017. http://dx.doi.org/10.1145/3133920.

[MCS12] Paul Martin, Zainul Charbiwala, and Mani Srivastava. DoubleDip: Leveraging Thermoelectric Harvesting for Low Power Monitoring of Sporadic Water Use. In *Proceedings of the 10th Conference on Embedded Network Sensor Systems*, SenSys '12, 2012. http://dx.doi.org/10.1145/2426656.2426679.

[MDM+20] Amjad Yousef Majid, Carlo Delle Donne, Kiwan Maeng, Alexei Colin, Kasim Sinan Yildirim, Brandon Lucia, and Przemysław Pawełczak. Dynamic Task-Based Intermittent Execution for Energy-Harvesting Devices. *Transactions on Sensor Netwworks (TOSN)*, 16(1), February 2020. http://dx.doi.org/10.1145/3360285.

[Mik20] MikroElektronika. *Datasheet MikroElektronika SR674361P LiPo*, December 2020. https://s3-us-west-2.amazonaws.com/shop.mikroe.com/downloads/batteries/li-polymer-battery-37v-

2000mah/SR674361P%20RevB%20-%20Updated%20datasheet.pdf. Last accessed: 2021/05/02.

[Mot04] Moteiv Corporation. *Datasheet Telos B Mote*, May 2004. http://www2.ece.ohio-state.edu/~bibyk/ee582/telosMote.pdf. Last accessed: 2021/05/02.

[MSHT16] Balz Maag, Olga Saukh, David Hasenfratz, and Lothar Thiele. Pre-Deployment Testing, Augmentation and Calibration of Cross-Sensitive Sensors. In *Proceedings of the International Conference on Embedded Wireless Systems and Networks*, EWSN '16, Graz, Austria, February 2016. http://dx.doi.org/10.5555/2893711.2893735.

[MTBB10] Clemens Moser, Lothar Thiele, Davide Brunelli, and Luca Benini. Adaptive Power Management for Environmentally Powered Systems. *Transactions on Computers (TC)*, 59(4), April 2010. http://dx.doi.org/10.1109/tc.2009.158.

[MZT18] Balz Maag, Zimu Zhou, and Lothar Thiele. A Survey on Sensor Calibration in Air Pollution Monitoring Deployments. *Internet of Things Journal*, 5(6), December 2018. http://dx.doi.org/10.1109/jiot.2018.2853660.

[NPK+15] Saman Naderiparizi, Aaron Parks, Zerina Kapetanovic, Benjamin Ransford, and Joshua Smith. WISPCam: A battery-free RFID camera. In *International Conference on RFID*, RFID '15, San Diego, CA, USA, April 2015. http://dx.doi.org/10.1109/rfid.2015.7113088.

[ODRR21] Peter Oppermann, Lennart Dorendorf, Marcus Rutner, and Christian Renner. Nonlinear Modulation with Low-power Sensor Networks Using Undersampling. *Structural Health Monitoring*, January 2021. http://dx.doi.org/10.1177/1475921720982885.

[Org13] World Health Organization. Health Risks of Air Pollution in Europe – HRAPIE Project: Recommendations for Concentration-response Functions for Cost-benefit Analysis of Particulate Matter, Ozone and Nitrogen Dioxide. *UN City: Copenhagen, Denmark*, 2013. https://www.euro.who.int/__data/assets/pdf_file/0006/238956/Health_risks_air_pollution_HRAPIE_project.pdf.

[Pan20] Panasonic. *Datasheet Eneloop Pro AA NiMH*, July 2020. https://asset.conrad.com/media10/add/160267/c1/-/en/001221220DS01/datenblatt-1221220-panasonic-eneloop-pro-hr06-mignon-aa-akku-nimh-2500-mah-12-v-4-st.pdf. Last accessed: 2021/05/02.

[PBAR10] Joaquin Recas Piorno, Carlo Bergonzini, David Atienza, and Tajana Simunic Rosing. HOLLOWS: A Power-aware Task Scheduler for Energy Harvesting Sensor Nodes. *Journal of Intelligent Material Systems and Structures*, 21(13), September 2010. http://dx.doi.org/10.1177/1045389x10377033.

[PC06] Chulsung Park and Pai Chou. Ambimax: Autonomous Energy Harvesting Platform for Multi-supply Wireless Sensor Nodes. In *3rd Annual Communications Society on Sensor and Ad Hoc Communications and Networks*, COMSOC '06, Reston, VA, USA, September 2006. http://dx.doi.org/10.1109/sahcn.2006.288421.

[PHC04] Joseph Polastre, Jason Hill, and David Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, Baltimore, MD, USA, November 2004. http://dx.doi.org/10.1145/1031495.1031508.

[PKK13] Naser Khosro Pour, François Krummenacher, and Maher Kayal. Fully Integrated Solar Energy Harvester and Sensor Interface Circuits for Energy-Efficient Wireless Sensing Applications. *Journal of Low Power Electronics and Applications (JPEA)*, 3(1), February 2013. http://dx.doi.org/10.3390/jlpea3010009.

[PL14] S. Peng and C. P. Low. Prediction Free Energy Neutral Power Management For Energy Harvesting Wireless Sensor Nodes. *Ad Hoc Networks*, 13, February 2014. http://dx.doi.org/10.1016/j.adhoc.2013.08.015.

[PSLRC18] L. Perilli, E. Franchi Scarselli, R. La Rosa, and R. Canegallo. Wake-up Radio Impact In Self-sustainability Of Sensor And Actuator Wireless Nodes In Smart Home Applications. In *Proceedings of the 9th International Green and Sustainable Computing Conference*, IGSC '17, Pittsburgh, PA, USA, October 2018. http://dx.doi.org/10.1109/igcc.2018.8752164.

[RBR17] J. Prasanth Ram, T. Sudhakar Babu, and N. Rajasekar. A Comprehensive Review on Solar PV Maximum Power Point Tracking Techniques. *Renewable and Sustainable Energy Reviews*, 67, January 2017. http://dx.doi.org/10.1016/j.rser.2016.09.076.

[Ren13] Christian Renner. Solar Harvest Prediction Supported by Cloud Cover Forecasts. In *Proceedings of the1st International Workshop on Energy Neutral Sensing Systems*, ENSsys '13, Rome, Italy, November 2013. http://dx.doi.org/10.1145/2534208.2534210.

[RFG13] Christian Rohner, Laura Marie Feeney, and Per Gunningberg. Evaluating Battery Models in Wireless Sensor Networks. In *Proceedings of the 11th International Conference on Wired/Wireless Internet Communication*, WWIC '13, St. Petersburg, Russia, June 2013. http://dx.doi.org/10.1007/978-3-642-38401-1_3.

[RID+11] Ario Alberto Ruprecht, Giovanni Invernizzi, Bertrand Dautzenberg, Luke Clancy, Josè Precioso, Cinzia De Marco, Roberto Boffi, Roberto Mazza, Maria Josè Lopez, and Hanns Moshammer. Mass Calibration and Relative Humidity Compensation Requirements for Optical Portable Particulate Matter Monitors: The IMPASHS (Impact of Smoke-free Policies in EU Member States) Wp2 Preliminary Results. *Epidemiology*, 22(1), January 2011. http://dx.doi.org/10.1097/01.ede.0000392314.24613.c6.

[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*, IPSN '05, Boise, ID, USA, April 2005. http://dx.doi.org/10.1109/ipsn.2005.1440973.

[RL19] Emily Ruppel and Brandon Lucia. Transactional Concurrency Control for Intermittent, Energy-harvesting Computing Systems. In *Proceedings of the 40th Conference on Programming Language Design and Implementation*, PLDI '19, June 2019. http://dx.doi.org/10.1145/3314221.3314583.

[RPBASR09] Joaquin Recas Piorno, Carlo Bergonzini, David Atienza, and Tajana Simunic Rosing. Prediction and Management in Energy Harvested Wireless Sensor Nodes. In *Proceedings of the 1st International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology*, VITAE '09, Aalbord, Denmark, May 2009. http://dx.doi.org/10.1109/wirelessvitae.2009.5172412.

[RT10] Christian Renner and Volker Turau. CapLibrate: Self-Calibration of an Energy Harvesting Power Supply with Supercapacitors. In *Proceedings of the International Conference On Architecture Of Computing Systems*, ARCS '10, Hanover, Germany, February 2010. https://ieeexplore.ieee.org/abstract/document/5759026.

[RT12] Christian Renner and Volker Turau. Adaptive Energy-harvest Profiling to Enhance Depletion-safe Operation and Efficient Task Scheduling. *Sustainable Computing: Informatics and Systems (SUSCOM)*, 2(1), March 2012. http://dx.doi.org/10.1016/j.suscom.2012.02.001.

[RUTR14] Christian Renner, Stefan Unterschütz, Volker Turau, and Kay Römer. Perpetual Data Collection with Energy-Harvesting Sensor Networks. *Transactions on Sensor Networks (TOSN)*, 11(1), September 2014. http://dx.doi.org/10.1145/2566675.

[RZ19] Christian Renner and Matteo Zella. The Internet Of Intermittent Things, A Land Of Low-hanging Fruits. In *Proceedings of the 7th International Workshop on Energy Harvesting & Energy-Neutral Sensing Systems*, ENSsys '19, New York, NY, USA, 2019. http://dx.doi.org/10.1145/3362053.3363493.

[Sam20] Samwha. *Datasheet Samwha GreenCap EDLC*, December 2020. https://asset.conrad.com/media10/add/160267/c1/-/en/000451438DS01/datenblatt-451438-samwha-db5u207m30045ha-green-cap-kondensator-200-f-27-vdc-20-o-x-l-30-mm-x-45-mm-1-st.pdf. Last accessed: 2021/05/02.

[SBH16] Farzad Samie, Lars Bauer, and Jörg Henkel. IoT Technologies for Embedded Computing. In *Proceedings of the 11th International Conference on Hardware/Software Codesign and System Synthesis*, CODES '16, Pittsburgh, PA, USA, October 2016. http://dx.doi.org/10.1145/2968456.2974004.

[SC06] Farhan Simjee and Pai Chou. Everlast: Long-Life, Supercapacitor-Operated Wireless Sensor Node. In *Proceedings of the 12th International Symposium on Low Power Electronics and Design*, ISLPED '06, Tegernsee, Germany, October 2006. http://dx.doi.org/10.1145/1165573.1165619.

[SCW+19] Sivert T. Sliper, Oktay Cetinkaya, Alex S. Weddell, Bashir Al-Hashimi, and Geoff V. Merrett. Energy-driven Computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 378(2164), December 2019. http://dx.doi.org/10.1098/rsta.2019.0158.

[SGI+14] Navin Sharma, Jeremy Gummeson, David Irwin, Ting Zhu, and Prashant Shenoy. Leveraging Weather Forecasts in Renewable Energy Systems. *Sustainable Computing: Informatics and Systems (SUSCOM)*, 4(3), September 2014. http://dx.doi.org/10.1016/j.suscom.2014.07.005.

[SGJ+20] Philipp Sommer, Kai Geissdoerfer, Raja Jurdak, Branislav Kusy, Jiajun Liu, Kun Zhao, Adam McKeown, and David Westcott. Energy- And Mobility-aware Scheduling For Perpetual Trajectory Tracking. *Transactions on Mobile Computing*, 19(3), March 2020. http://dx.doi.org/10.1109/tmc.2019.2895336.

[SGT19] Lukas Sigrist, Andres Gomez, and Lothar Thiele. Dataset: Tracing Indoor Solar Harvesting. In *Proceedings of the 2nd Workshop on Data Acquisition To Analysis*, DATA '19, New York, NY, USA, November 2019. http://dx.doi.org/10.1145/3359427.3361910.

[SKS20] Jaspreet Singh, Ranjit Kaur, and Damanpreet Singh. Energy Harvesting in Wireless Sensor Networks: A Taxonomic Survey. *International Journal of Energy Research*, 45(1), September 2020. http://dx.doi.org/10.1002/er.5816.

[SR09] Jamie Bradley Steck and Tajana Simunic Rosing. Adapting Task Utility in Externally Triggered Energy Harvesting Wireless Sensing Systems. In *Proceedings of the 6th International Conference on Networked Sensing Systems*, INSS '09. IEEE, 2009. http://dx.doi.org/10.1109/inss.2009.5409959.

[SSP+06] Joshua R. Smith, Alanson P. Sample, Pauline S. Powledge, Sumit Roy, and Alexander Mamishev. A Wirelessly-Powered Platform for Sensing and Computation. In *Proceedings of the 8th International Conference of Ubiquitous Computing*, UbiComp '06, Orange County, CA, USA, September 2006. http://dx.doi.org/10.1007/11853565_29.

[STM19] STMicroelectronics. *Datasheet STM32L072X8*, September 2019. https://www.st.com/resource/en/datasheet/stm321072cz.pdf. Last accessed: 2021/05/02.

[SWYS11] Jianhua Shi, Jiafu Wan, Hehua Yan, and Hui Suo. A Survey of Cyber-Physical Systems. In *International Conference on Wireless Communications and Signal Processing*, WCSP '11, Nanjing, China, November 2011. http://dx.doi.org/10.1109/wcsp.2011.6096958.

[Tex17] Texas Instruments. *Datasheet MSP430FR573x*, December 2017. https://www.ti.com/lit/gpn/msp430fr5739. Last accessed: 2021/05/02.

[TSCM20] Chai K. Toh, Julio A. Sanguesa, Juan C. Cano, and Francisco J. Martinez. Advances in Smart Roads for Future Smart Cities. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 476(2233), January 2020. http://dx.doi.org/10.1098/rspa.2019.0439.

[TSM+17] Moritz Thielen, Lukas Sigrist, Michele Magno, Christofer Hierold, and Luca Benini. Human Body Heat for Powering Wearable Devices: From Thermal Energy to Application. *Energy Conversion and Management*, 131, January 2017. http://dx.doi.org/10.1016/j.enconman.2016.11.005.

[Ult14] UltraLife. *Datasheet UltraLife UBBP01 LiIon Battery*, October 2014. https://www.mouser.de/datasheet/2/419/TDS_UBBP01-972369.pdf. Last accessed: 2021/05/02.

[VGB07] Christopher M. Vigorito, Deepak Ganesan, and Andrew G. Barto. Adaptive Control Of Duty Cycling In Energy-harvesting Wireless Sensor Networks. In *Proceedings of the 4th Annual Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, SAHCN '07, San Diego, CA, USA, June 2007. http://dx.doi.org/10.1109/sahcn.2007.4292814.

[WALR+06] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Welsh. Deploying a Wireless Sensor Network on an Active Volcano. *Internet Computing*, 10(2), March 2006. http://dx.doi.org/10.1109/mic.2006.26.

[WBF+19] Samuel Weber, Jan Beutel, Reto Da Forno, Alain Geiger, Stephan Gruber, Tonio Gsell, Andreas Hasler, Matthias Keller, Roman Lim, Philippe Limpach, Matthias Meyer, Igor Talzi, Lothar Thiele, Christian Tschudin, Andreas Vieli, Daniel Vonder Mühll, and Mustafa Yücel. A Decade of Detailed Observations (2008–2018) in Steep Bedrock Permafrost at Matterhorn Hörnligrat (Zermatt, CH). *Earth System Science Data*, 11(3), February 2019. http://dx.doi.org/10.5194/essd-2019-14.

[WCPnC18] Daniel A. Winkler, Miguel Á. Carreira-Perpiñán, and Alberto E. Cerpa. Plug-and-play Irrigation Control At Scale. In *Proceedings of the 17th International Conference on Information Processing in Sensor Networks*, IPSN '18, Porto, Portugal, April 2018. http://dx.doi.org/10.1109/IPSN.2018.00008.

[WDJ+16] Peter Wägemann, Tobias Distler, Heiko Janker, Phillip Raffeck, and Volkmar Sieh. A Kernel for Energy-Neutral Real-Time Systems with Mixed Criticalities. In *Proceedings of the Real-Time and Embedded Technology and Applications Symposium*, RTAS '16, Vienna, Austria, April 2016. http://dx.doi.org/10.1109/rtas.2016.7461320.

[WDJ+18] Peter Wägemann, Tobias Distler, Heiko Janker, Phillip Raffeck, Volkmar Sieh, and Wolfgang Schröder-Preikschat. Operating Energy-Neutral Real-Time Systems. *Transactions on Embedded Computing Systems (TECS)*, 17(1), January 2018. http://dx.doi.org/10.1145/3078631.

[WDMM14] Chin-Hong Wong, Zuraini Dahari, Asrulnizam Abd Manaf, and Muhammad Azman Miskam. Harvesting Raindrop Energy with Piezoelectrics: a Review. *Journal of Electronic Materials*, 44(1), October 2014. http://dx.doi.org/10.1007/s11664-014-3443-4.

[WJ17] Chongfeng Wei and Xingjian Jing. A Comprehensive Review on Vibration Energy Harvesting: Modelling and Realization. *Renewable and Sustainable Energy Reviews*, 74, July 2017. http://dx.doi.org/10.1016/j.rser.2017.01.073.

[WMKAH11] Alex S. Weddell, Geoff V. Merrett, Tom J. Kazmierski, and Bashir M. Al-Hashimi. Accurate Supercapacitor Modeling for Energy Harvesting Wireless Sensor Nodes. *Transactions on Circuits and Systems II: Express Briefs*, 58(12), December 2011. http://dx.doi.org/10.1109/tcsii.2011.2172712.

[WMM+13] Alex S. Weddell, Michele Magno, Geoff V. Merrett, Davide Brunelli, Bashir M. Al-Hashimi, and Luca Benini. A Survey of Multi-Source Energy Harvesting Systems. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition*, DATE '13, Grenoble, France, March 2013. http://dx.doi.org/10.7873/date.2013.190.

[Wür08] Peter Würfel. *Physics of Solar Cells*. Wiley, 2008. https://cds.cern.ch/record/1609973.

[YHE02] Wei Ye, J. Heidemann, and D. Estrin. An Energy-efficient MAC Protocol for Wireless Sensor Networks. In *Proceedings of the 21st Annual Joint Conference of the Computer and Communications Societies*, INFOCOM '02, New York, NY, USA, June 2002. http://dx.doi.org/10.1109/infcom.2002.1019408.

[YKK+20] Ibrar Yaqoob, Latif U. Khan, S. M. Ahsan Kazmi, Muhammad Imran, Nadra Guizani, and Choong Seon Hong. Autonomous Driving Cars in Smart Cities: Recent Advances, Requirements, and Challenges. *Network*, 34(1), January 2020. http://dx.doi.org/10.1109/mnet.2019.1900120.

[YMP+18] Kasim Sinan Yildirim, Amjad Yousef Majid, Dimitris Patoukas, Koen Schaper, Przemyslaw Pawelczak, and Josiah David Hester. InK: Reactive Kernel for Tiny Batteryless Sensors. In *Proceedings of the 16th Conference on Embedded Network Sensor Systems*, SenSys '18, Shenzhen, China, November 2018. http://dx.doi.org/10.1145/3274783.3274837.

[YTJ+19] Fan Yang, Ashok Samraj Thangarajan, Wouter Joosen, Christophe Huygens, Danny Hughes, Gowri Sankar Ramachandran, and Bhaskar Krishnamachari. AsTAR: Sustainable Battery Free Energy Harvesting For Heterogeneous Platforms And Dynamic Environments. In *Proceedings of the International Conference on Embedded Wireless Systems and Networks*, EWSN '19, Beijing, China, February 2019. http://dx.doi.org/10.5555/3324320.3324329.

[YTR16] Jinseok Yang, Sameer Tilak, and Tajana Simunic Rosing. An Interactive Context-aware Power Management Technique for Optimizing Sensor Network Lifetime. In *Proceedings of 5th International Confererence on Sensor Networks*, SENSORNETS '16, Rome, Italy, February 2016. http://dx.doi.org/10.5220/0005728600690076.

[ZMPT12] Ting Zhu, Abedelaziz Mohaisen, Yi Ping, and Don Towsley. DEOS: Dynamic Energy-oriented Scheduling for Sustainable Wireless Sensor Networks. In *Proceedings of the Annual Joint Conference of the Computer and Communications Societies*, INFOCOM '12, Orlando, FL, USA, May 2012. http://dx.doi.org/10.1109/infcom.2012.6195625.

List of Acronyms

ADC Analog to Digital Converter

BFS Breadth-first Search

CDF Cumulative Distribution Function

CPS Cyber-physical Systems

DAG Directed Acyclic Graph

DFS Depth-first Search

DTN Delay-tolerant Network

DVS Dynamic Voltage Scaling

DVFS Dynamic Voltage Frequency Scaling

EDeg Earliest Deadline with Energy Guarantees

EDF Earliest Deadline First

EH Energy Harvesting

EH-WSN Energy-harvesting Wireless Sensor Network

EDLC Electric Double-Layer Capacitor

ENO Energy-neutral Operation

EWMA Exponentially-weighted Moving Average

FRAM Ferroelectric Random Access Memory

IC Intermittent Computing

ILP Integer Linear Programming

IoT Internet of Things

KA Kahn's Algorithm

LiIon Lithium-ion

LiPo Lithium-ion Polymer

LP Linear Programming

LSA Lazy Scheduling Algorithm

LT-ENO Long-term Energy-Neutral Operation

MAC Medium Access Control

MAPE Mean Absolute Percentage Error

MCU Microcontroller Unit

MPP Maximum Power Point

MPPT Maximum Power Point Tracking

NiCd Nickel-Cadmium

NiMH Nickel Metal Hydrydide

PHY Physical Layer

PID Proportional-integral-derivative

Pro-Energy Profile Energy Prediction Model

Pro-Energy-VLT Profile Energy Prediction Model with Variable-length Timeslots

PV Photovoltaic

QoS Quality-of-Service

RF Radio Frequency

RFID Radio-Frequency Identification

RMSE Root-Mean-Squared Error

RTC Real-time Clock

SoC State-of-Charge

TEG Thermoelectric Generator

VAM Vibroacoustic Modulation

WCMA Weather-conditioned Moving Average

WEDF Weighted Earliest Deadline First

WSN Wireless Sensor Network