Bachelor Thesis

Energy-Aware and Prediction-Based Scheduling for Energy-Harvesting Sensor Nodes

by

Florian Meier

September 2011

First Examiner | Prof. Dr. Volker Turau
Institute of Telematics
Hamburg University of Technology

Second Examiner | Christian Renner
Institute of Telematics
Hamburg University of Technology

Institute of Telematics
Hamburg University of Technology
TUHH

Declaration by Candidate

I, FLORIAN MEIER (student of Informatik-Ingenieurwesen at Hamburg University of Technology, matriculation number 20836390), hereby declare that this thesis is my own work and effort and that it has not been submitted anywhere for any award. Where other sources of information have been used, they have been acknowledged.

Hamburg, 1st September, 2011

Florian Meier

Table of Contents

List of Symbols

Λ	series of performance levels ranging from 1 to λ_{max}
N	number of slots
l_s	length of slots s
τ_s	starting time of slots s
T	prediction horizon
$P_h(t)$	power produced by the harvesting device
$\widetilde{P}_h(t)$	predicted power produced by the harvesting device
$I_h(t)$	current produced by the harvesting device
$\widetilde{I}_h(t)$	predicted current produced by the harvesting device
$I_c(t)$	capacitor current consumption
$V_c(t)$	capacitor voltage
$E_c(t)$	capacitor energy
$\widetilde{V}_c(t)$	predicted capacitor voltage
$P_c(t)$	power flow into capacitor
$I_n(t)$	sensor node current consumption
V_n	sensor node supply voltage
$P_n(t)$	sensor node power consumption
$\overline{P}_n$	sensor node power consumption averaged over one slot
η_B	coulombic efficiency of a battery
η	efficiency of DC-DC converter

Chapter 1

Introduction

Monitoring is a frequent application in today's life. Engineers observe bridges and historical buildings to detect signs of fraction early enough to take countermeasures. Scientists desire to observe animals in the wild, volcanoes or glaciers to gain insight into these phenomena. These are only a handful of examples.

Conventional monitoring of the environment either requires laying cables to connect each sensor to the base station, which collects all data from the individual sensors. If this is impracticable, all data is stored by each sensor and collected manually. Both methods are cost and time intensive, thus rendering them infeasible for many applications. Moreover, they may interfere with the subject of interest, caused by the invasive deployment or by the frequent presence of humans: This may be impossible due to legal restrictions (e.g., in monument conservation) or may lead to a biased result (e.g., in wildlife monitoring). In order to tackle these problems alongside the high costs in terms of time and financial outlay, wireless sensor networks have been developed. They consist of so-called sensor nodes, which are tiny and cheap electronic devices with restricted memory and processing resources. They are also equipped with low-power radio for communicating among each other. This enables collaborative tasks and automatic, wireless data collection. The tiny size and low price of sensor nodes provides the opportunity of non-invasive monitoring and of large-scale deployment to achieve fine-grained spatial resolution.

A sensor node is typically powered by small batteries as energy source. This limits the theoretical lifetime of the sensor nodes and produces high maintenance costs,

because the batteries have to be replaced regularly. The lifetime ranges from a few days up to some years depending on the a-priori scheduled load of each node. A false estimation or other unforeseen problems that arise from large-scale deployments may lead to a considerably reduced lifetime.

Therefore, much effort has been put into reducing the energy consumption of the sensor nodes, e.g. low power radio communication and energy-aware routing to name the most active areas of research. Yet, this only prolongs lifetime but cannot prevent battery replacement. Furthermore, the time space between depleted batteries and replacement will produce measurement gaps. This can reduce the significance of the data or even render it useless. To be dependent on maintenance is especially inappropriate for wireless sensor networks in harsh and difficult to reach environments, such as mountains or for arctic observations.

In these environments, however, it is often possible to utilize regenerative energy sources, e.g., solar, wind or thermal energy. This technique is referred to as energy harvesting. Installing a harvesting device, i.e., a solar cell or a wind generator, allows for sustainable and self-sufficient node operation. The harvested energy can be buffered with rechargeable batteries or supercapacitors, which can compensate times with no energy intake, as regenerative energy sources neither provide a steady nor an evenly distributed energy supply.

Conventional harvesting-powered sensors have large harvesting devices and energy buffers in order to prevent them from running out of energy in any case, which make them expensive. Much of this dearly paid energy is wasted, because there have to be enough reserves for times with poor energy conditions. Moreover, to keep the benefits of tiny sensor nodes alive, the harvesting devices have to comply with their size.

In many applications of wireless sensor networks the high cost is a knock-out criterion as these networks often consist of many hundred sensor nodes, hence a low price per unit is required. Costs can be reduced by scaling the harvesting devices and energy buffers down and compensate this by adapting the system's performance to the available energy. Thus, few energy is wasted, because the harvesting device produces less excess energy, which on the other hand can be transferred into useful benefit. The goal to strive for is energy neutral operation, i.e. harvested and consumed energy energy are balanced.

This bachelor thesis develops an algorithm that accomplishes this task and can run on state-of-the-art sensor nodes. The algorithm adapts itself continuously for

many reasons: First the weather itself is highly dynamic. A behavior that is suitable at summer can be poor at winter. Second it is very likely that different nodes in a sensor network face different energy conditions, e.g., because one could be located under a tree while another one is directly exposed to the sun. Third these conditions could change due to environmental changes (e.g., growing plants) or aging effects of the hardware.

The algorithm uses the concept of performance levels, defined by the application designer. In contrast to existing solutions, the algorithm thus allows for an application-specific adaption of the performance to meet the energy conditions of a sensor node. To achieve this goal it utilizes three concepts proposed in the literature: Assessment of the residual energy, prediction of future energy intake and the expected consumption in each performance level.

This thesis evaluates the performance of the algorithm via simulation using real-world solar traces and compares it with another solution proposed in literature. It shows that the developed algorithm can adapt the system performance quickly to the current energy conditions, while showing an adequate low variance during a day. The evaluation in this thesis also shows that other algorithms are inapplicable for the used hardware, either because they can not be adapted to the hardware or are unstable with it.

Alongside the development of an algorithm a ready-to-use software for sensor nodes based on the TinyOS platform, a popular operating system for wireless sensor networks, is implemented. It demonstrates the usefulness and applicability of the algorithms and is tested in a real-world deployment.

Chapter 2

State of the Art

This chapter gives an overview about characteristics and basic concepts of wireless sensor networks. Furthermore, techniques needed for implementing an energy-aware scheduler for wireless sensor networks are introduced. Finally existing algorithms for energy-aware scheduling are presented.

2.1. Sensor Networks and Deployments

2.1.1. Sensor Nodes

A sensor node is a tiny electrical sensing device. It contains various sensors and typically a low-power 8-bit microcontroller and a radio chip with a transmission speed of 250 KBit/s. Modern sensor nodes are also equipped with so-called energy-harvesters, i.e. modules harvesting regenerative energy from the environment. Examples are Enviromote [KSS+07], Trio [DHJ+06] and the one presented in [RJT09] that was used in this thesis. The energy harvester is supported by a battery or a supercapacitor buffering spare energy in order to ensure sustainable operation even in times where no harvesting is possible.

2.1.2. Sensor Network Deployments

Today many applications of wireless sensor networks are found in the field. The environments range from glaciers with no settlement within kilometers [MPE+06] to downtown Tokyo [OII+07]. In these two scenarios, coin sized hardware driven by lithium button cells were used. Much bigger lead acid batteries supported by solar panels driving a sensor network for warning about rainfall-induced landslides in India [Ram09].

Some wireless sensor networks are developed for a special research and therefore designed for finite lifetime in the first place. They are perfect for investigation of microclimate with high spatial resolution, examples being redwood trees [TPS+05] or potato fields [LBV06]. They permit deferred message sending, e.g., as reported by the authors, the data from the potato field is only needed once in the morning, when the treatment of the field is planned. The potato plants induced a problem, because the growing plants degraded the radio range. Furthermore, the large-scale deployment raised some problems with the software caused by wrong assumptions: The radio protocol assumed non moving sensor nodes. This is virtually correct, because the sensor nodes were mounted at fixed positions at the field. The problem aroused from the fact that they were turned on before mounting them on the field.

In [SMP+04] sensor nodes were used for non-invasive habitat monitoring at Great Duck Island. Among others, sensor nodes were placed in breeding burrows in order to detect residing birds. All of the three last named papers used batteries and report on premature node failure, either because software bugs led to a higher energy consumption or the energy consumption was estimated inaccurately.

2.2. Residual Energy and Energy Consumption

Energy-aware algorithms rely on information about a node's energy consumption. It is necessary to know how much energy each job, e.g, measurement, transmitting a message, consumes in order to create an execution plan. Each job draws a specific amount of energy, as the execution time and the used hardware components (e.g., microprocessor or radio) vary.

Furthermore, the residual energy left in the energy buffer, must be predicted. For a capacitor, the voltage can be measured resulting in the residual energy

$$E_c(t) = \frac{1}{2} C V_c(t)^2 \tag{2.1}$$

2.2.1. Premeasured Energy

The first approach is to calibrate each job before deployment. An execution of a job is measured externally, e.g, by an oscilloscope, before deployment. These values serve as fixed configuration for the scheduler. Therefore, it neither needs software nor hardware support. This needs high manual effort before deployment. The other problem is that the consumed energy may change. For example, the energy consumed by the radio depends on link quality, e.g., because of messages that have to be retransmitted. Furthermore, because of low cost hardware, the consumption may vary between different sensor node. This would require measurements for each single sensor node.

2.2.2. Componentwise Energy Measurement

An approach for on-line energy measurement was introduced in [DOTH07]. A component (e.g., the radio) can have many states (e.g. sending, listening). Each component has at least two states (e.g., on and off). The power a component c drains in state b is denoted $P_{c,b}$. This power can be measured without knowledge of the concrete software implementation. The problem of measuring the energy consumption is converted into the problem of measuring the time $t_{c,b}$ a job resides in a component's state with high resolution as introduced in [SDS10]. The resulting energy is

$$E_j = \sum_c \sum_b t_{c,b} \cdot P_{c,b}. \tag{2.2}$$

This takes care of packet retransmission etc. and does not need a manual configuration of each job, but introduces overhead in terms of calculation, because there is a multiplication at every state change.

2.2.3. iCount

The last solution is hardware based. Dutta et al. [DFPC08] presented an energy measurement approach for sensor nodes driven by a switching regulator. A switching regulator transfers small amounts of energy in recurring cycles. The energy is buffered by an inductor. Each cycle delivers the energy

$$E_{cycle} = \frac{1}{2} L i^2, \tag{2.3}$$

where i denotes the peak current in the inductor. It is thus possible to count the number of cycles and deduce the total energy transferred. This can be achieved by wiring the switching regulator to a counter of the microcontroller. This method can be used to estimate overall consumption or the per job consumption: At the beginning of a job the counter is reset to zero. The value of the counter M at the end of the job yields the energy consumption

$$E_j = M \cdot E_{cycle}. \tag{2.4}$$

2.2.4. Conclusion

Table 2.1 summarizes the different approaches.

	Advantages	Disadvantages
Premeasured	■ Simple implementation	■ Inaccurate ■ High manual effort
Componentwise	■ Accurate (if well calibrated)	■ Calculational overhead ■ It is necessary to calibrate the components
iCount	■ Accurate ■ Simple implementation	■ Hardware modification and calibration needed

Table 2.1.: Advantages and disadvantages for various energy consumption approaches

2.3. Prediction Techniques

At a point in time t the regenerative energy source has a harvesting potential $H(t)$. Depending on the hardware used, this could be a power $P_h(t)$ or a current $I_h(t)$. The actual benefit from this potential depends on the current system state. It is not even guaranteed, that this potential can be used by the system, because it could have a full energy buffer.

Many energy aware algorithms use a prediction unit for predicting future energy intake. Its prediction lasts a time T into the future. This time duration is referred to as prediction horizon. For solar-powered devices a prediction horizon of one day is reasonable, because the sun has a periodicity of one day.

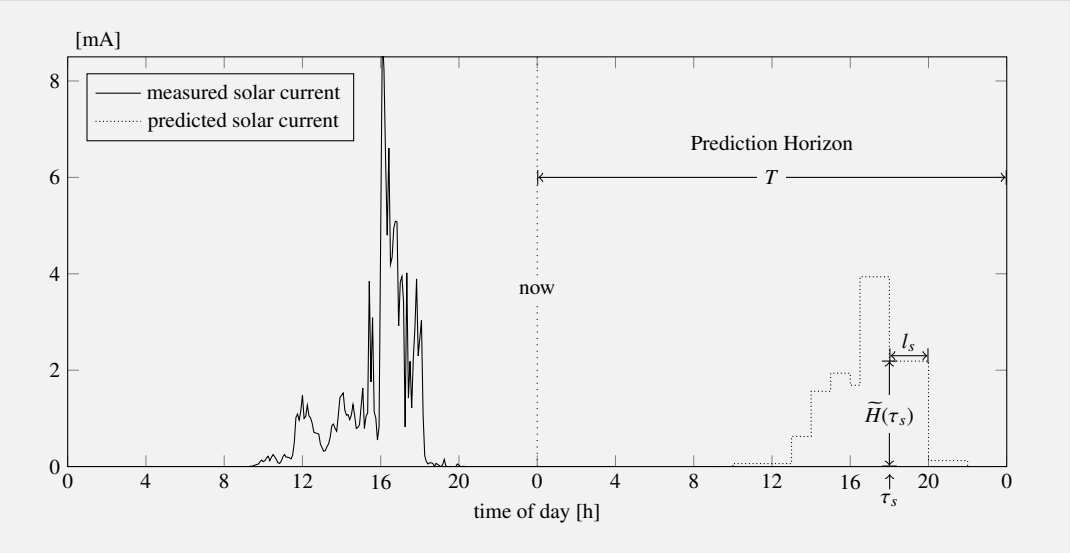


Figure 2.1.: Sample profile with measured solar current and prediction for next day.

A frequently used technique is to partition the prediction horizon into N intervals $[\tau_s, \tau_{s+1})$, also referred to as slot, with $\tau_N = \tau_0 + T$, where τ_0 is the time at the start of each period, e.g., twelve o'clock at night for solar powered harvesters. Each slot has a length $l_s = \tau_{s+1} - \tau_s$ and a predicted mean value μ_s, so the predicted harvesting potential at time t is

$$\widetilde{H}(t) = \sum_{s=0}^{N-1} \begin{cases} \mu_s & \text{if } t \in [\tau_s, \tau_{s+1}) \\ 0 & \text{else} \end{cases} . \tag{2.5}$$

Note that $\widetilde{H}(\tau_s) = \mu_s$. This notation is depicted in 2.1 on the preceding page. The use of slots saves memory compared to the complete storage of the measured values. At a specific number of slots, the error resulting from this is smaller than the error that arises from the uncertainty of the weather.

2.3.1. Exponentially Weighted Moving Average

In order to find the prediction values μ_s, an approach that was introduced by [KHZS07] is based on slots with a fixed length and maintains an exponential weighted moving average for each slot. After a slot ends, the prediction for this slot is updated by means of the measured mean value $\overline{H}(s-1)$

$$\mu_s \leftarrow \alpha\mu_s + (1-\alpha) \cdot \overline{H}(s-1), \qquad 0 < \alpha < 1 \tag{2.6}$$

2.3.2. Weather Conditioned Moving Average

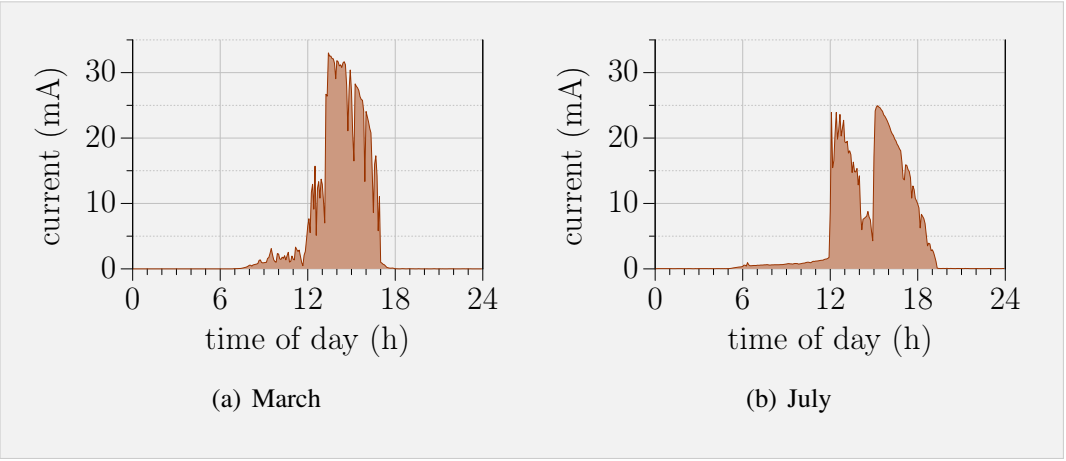
This technique was enhanced by Piorno et al. [PBAR09] by taking the current day's condition into account. The authors state that a day with poor condition at the morning will likely be poor for the rest of the day. Therefore, the prediction for the following slot is scaled by the prediction error in the directly preceding slots. Using one preceding slot, the prediction of the following slot is denoted

$$\tilde{\mu}_s \leftarrow \alpha\tilde{\mu}_s + (1-\alpha) \cdot \overline{H}(s-1), \qquad 0 < \alpha < 1 \tag{2.7}$$

$$\mu_s \leftarrow \beta\tilde{\mu}_s \cdot \frac{\overline{H}(s-1)}{\tilde{\mu}_{s-1}} + (1-\beta) \cdot \overline{H}(s-1), \qquad 0 < \beta < 1 \tag{2.8}$$

2.3.3. Variable Slot Lengths

Another approach presented in [Ren10] is based on the observation, that the harvesting potential varies a lot at some times, while it varies less (or even is zero) at other times. In order to exploit this fact, variable slot lengths are introduced in order to reduce the variance and with it the prediction error within one slot. The adaption of the slot lengths is done online in order to adapt to environmental changes. This is necessary, because the harvesting profile is changing during the year as depicted in Fig. 2.2 on the next page.



(a) March (b) July

Figure 2.2.: Harvesting profiles at different times of the year recorded by the same sensor node with unaltered position [Ren10]

During the day, the errors resulting from the averaging of each slot are measured. At the end of the day, the slot with the highest error is splitted, while two consecutive slots with small errors are merged, if the total error is reduced by this operation.

2.4. Energy-Aware Load Adaption

This section describes some existing approaches for energy-aware load adaption. At time t the sensor node draws a power $P_n(t)$. We assume that the underlying hardware utilizes a DC-DC converter to guarantee a fixed input voltage. The DC-DC converter has an efficiency η. Thus, the energy unit has to provide a power $\frac{P_n(t)}{\eta}$. It is supplied by the harvester or, if this not suffices, by battery or supercapacitor.

The algorithms presented in this chapter result in a assignment of duty cycles D_s to future slots. A duty cycle is the ratio where the system is in an energy consuming active state $P_n \approx 50\,\text{mW}$ or more, compared an idle state with low power consumption $P_n \approx 30\,\mu\text{W}$. This is often only related to the radio, because it is the heaviest consumer of the system, but can also include other system components (e.g., duration of calculations or measurement rate).

The aim to strive for is energy neutrality introduced by Kansal et al. in [KHZS07]. The outgoing energy must not outrun the incoming energy (harvested plus initial

energy). For a system without energy buffer this means

$$P_h(t) \geq \frac{P_n(t)}{\eta}, \qquad \forall t \in [0;\infty) \tag{2.9}$$

By using an ideal energy buffer (infinite capacity, no loss), the sensor node can be supplied, even if no power is harvested, but the buffer has enough energy:

$$\underbrace{\int_0^T P_h(t)\,\mathrm{d}t + E_c(0)}_{\text{harvested \& initial energy}} \geq \underbrace{\int_0^T \frac{P_n(t)}{\eta}\,\mathrm{d}t}_{\text{consumed energy}}, \qquad \forall T \in [0;\infty) \tag{2.10}$$

with $E_c(0)$ being the initial energy stored in the buffer. A not empty energy buffer at all times t is sufficient in order to reach energy neutrality:

$$E_c(t) > 0, \qquad \forall t \in [0;\infty) \tag{2.11}$$

If this holds true, then there is no time, where the total consumed energy exceeds the harvested plus the initial energy. The original definition allows for the case where $E_c(t) = 0$, but there is enough harvesting power to drive the system. This is excluded in the last statement, because it is an unstable case: Coming from $E_c(t) > 0$ this case can only be reached if the consumed power exceeds the harvested power, but this would lead to system failure as soon as $E_c(t) = 0$ is reached.

2.4.1. Harvesting-Aware Power Management

One of the first systematic approaches towards power management in energy harvesting sensor networks is [KHZS07]. The authors based their considerations on a system model as depicted in Fig. 2.3. It matches their hardware named HelioMote [RKH+05]. It has the possibility to bypass the battery in order to prevent losses due to the charging circuit and the batteries efficiency η_B ($0 < \eta_B < 1$). The authors report on a typical value of $\eta_B \approx 70\,\%$ for NiMH batteries. The power from the harvester $P_h(t)$ can be used directly by the sensor node with efficiency η. Excess energy is stored in the battery, which then can only be used with an overall efficiency $\eta_B \cdot \eta$.

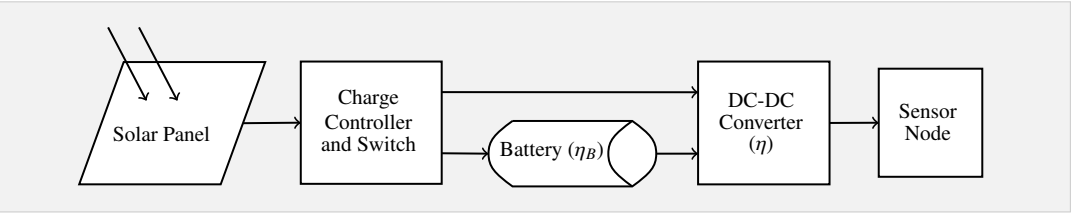


Figure 2.3.: System model by Kansal et al.

The algorithm developed in [KHZS07] searches an optimal series of duty cycles $\langle D_0, \ldots, D_{N-1} \rangle$ in such a way, that the sum is maximized:

$$\begin{aligned} &\max \sum_{s=0}^{N-1} D_s, \\ &D_{min} \leq D_s \leq D_{max} \quad \forall s \in [0, N-1]. \end{aligned} \tag{2.12}$$

The slots are assigned to two sets depending on whether the battery is charging ($\mathcal{S}_c$) or discharging ($\mathcal{S}_d$), i.e.

$$\mathcal{S}_c = \{s \mid \widetilde{P}_h(\tau_s) \geq P_n\} \tag{2.13}$$

$$\mathcal{S}_d = \{s \mid \widetilde{P}_h(\tau_s) < P_n\}. \tag{2.14}$$

As the algorithm tries to reach energy neutrality, the incoming energy should balance the outgoing energy

$$\begin{aligned} \sum_{s=0}^{N-1} l_s \cdot \widetilde{P}_h(\tau_s) = &\sum_{s \in \mathcal{S}_c} l_s \cdot D_s \cdot \frac{P_n}{\eta} \\ &+ \sum_{s \in \mathcal{S}_d} l_s \cdot D_s \cdot \left(\frac{P_n}{\eta_B \cdot \eta} + \widetilde{P}_h(\tau_s) \left(1 - \frac{1}{\eta_B \cdot \eta} \right) \right). \end{aligned} \tag{2.15}$$

For $\eta_B < 1$ and for hardware as depicted in Fig. 2.3 it is always better to use as much energy as possible without buffering. In order to maximize (2.15) the authors are using the initial assignment

$$D_s = D_{min} \quad \forall s \in \mathcal{S}_d, \tag{2.16}$$

$$D_s = D_{max} \quad \forall s \in \mathcal{S}_c. \tag{2.17}$$

If this leads to a too high energy consumption, the duty cycle in the charging slots is reduced uniformly. If there is excess energy, the discharging slots with the largest coefficients will be set to D_{max}, too. According to the authors this algorithm leads to the optimal solution of (2.12) on the preceding page for their hardware.

Dynamic Duty Cycling

Due to the fact, that there always is a error in the prediction of energy intake, the algorithm presented in the previous section is supported by an additional algorithm. It compares predicted values $\widetilde{P}_h(\tau_s)$ and observed values $P_h(\tau_s)$ and adjusts the duty cycles of following slots accordingly. The excess energy is defined as

$$E_X = \begin{cases} \widetilde{P}_h(\tau_s) - P_h(\tau_s) & \text{if } P_h(\tau_s) > P_n \\ \widetilde{P}_h(\tau_s) - P_h(\tau_s) - \left(\widetilde{P}_h(\tau_s) - P_h(\tau_s)\right)\left(1 - \frac{1}{\eta}\right) D_s & \text{if } P_h(\tau_s) \leq P_n \end{cases} \tag{2.18}$$

For $E_X < 0$ it is necessary to reduce the duty cycle in subsequent slots. It will reduce the duty cycle in slots with lowest harvesting potential first. For $E_X > 0$ the algorithm increases the duty cycle. In order to account for better energy utilization this is done in slots with highest harvesting potential first. This algorithm complements the one described in the last section, which is executed once a day in order to find appropriate duty cycles for the next day. The algorithm presented in this section accounts for prediction errors during the day. However, the prediction error can not completely compensated, because duty cycles in elapsed slots can not be corrected.

2.4.2. Linear Quadratic Tracking

The so-called LQ-Tracker, developed in [VGB07], uses a model-free approach that needs no knowledge about the hardware and does not rely on an explicit prediction technique. Furthermore, reduction of duty cycle stability is addressed.

A constant energy level V_c^* is assumed as optimal course by the authors. The objective is to reduce the mean square error of the voltage $V_c(t)$ at discrete time steps t

$$min \lim_{N\to\infty} \frac{1}{N} \sum_{t=1}^{N} (V_c(t) - V_c^*)^2. \tag{2.19}$$

This matches a linear-quadratic (LQ) tracking problem as described in [KV86]. The sporadic energy income is modeled as noise. The authors derive the optimal control law for (2.19) on the preceding page

$$D = \frac{V_c^* - (a+c) \cdot V_c + c \cdot V_c^*}{b}, \tag{2.20}$$

while the parameters a, b and c are estimated online with gradient descent techniques [KV86]. An additional part of the algorithm accounts for reducing duty-cycle variance. This is achieved by introducing an exponentially weighted moving average

$$\overline{D}_t = (1-\alpha) \cdot \overline{D}_{t-1} + \alpha \cdot D, \qquad 0 < \alpha \leq 1. \tag{2.21}$$

The averaged duty cycle is weighted with the current, optimal duty cycle

$$\widehat{D}_t = \beta \cdot D + (1-\beta) \cdot \overline{D}_t, \qquad 0 < \beta \leq 1. \tag{2.22}$$

Appropriate parameters α and β were determined by experiment in [VGB07], but depend on the deployment.

2.4.3. Multiparametric Linear Programming

The authors of [MTBB07] set their focus on implementing a wide range of linear programs with restricted resources. An example would be to maximize the execution rate of a job, such as a measurement, or to maximize the minimal duty cycle. The associated linear program is to maximize λ subject to

$$D_s \geq \lambda, \qquad \forall\, s \in [0, N-1] \tag{2.23}$$

$$0 \leq E_c + \sum_{r=0}^{s-1} l_r \cdot \left(\widetilde{P}_h(\tau_r) - \frac{P_n}{\eta} D_r \right) \qquad \forall\, s \in [1, N]. \tag{2.24}$$

The first inequality formulates the minimum duty cycle λ to use. (2.24) calculates the predicted energy level in slot s. Furthermore it contains an inequality to take account for the fact that the energy buffer can not contain negative energy. E_c denotes

the current energy level. The linear program can be extended by other constraints (e.g., used memory buffer for radio packets).

It is too complex to solve this linear program on a sensor node with standard techniques like simplex algorithm, as the dimension of the solution space is N (the number of slots, e.g., 12). The authors developed an algorithm to solve such a linear program on sensor nodes by precalculating some parts offline before deployment.

Hierarchical Control Design

An extension was introduced in [MTBB08]. The algorithm consists of two layers with one linear program each: A worst-case layer which maximizes the minimal available energy per day and an average-case layer, which is responsible for short-term power saving. The calculated energy per day is used by the average case algorithm. This way the complexity is reduced compared to a single linear program and it is robust against poor weather conditions.

The algorithms by Moser et al. are not dependent on a specific system model, although they were developed for a system like the one in [KHZS07] with a by-passable charging circuit as depicted in Fig. 2.3 on page 13. They can be adapted to different hardware as long as it does not contain any nonlinearities, but a complex system model results in a large state space with high computational complexity.

2.4.4. Energy Management for Time-Critical Applications

Besides duty cycle adaption there are other concepts to control the node's energy consumption, e.g, Dynamic Voltage Scaling (DVS) and Dynamic Modulation Scaling (DMS). Zhang et al. [ZSA10] developed the concept of Harvesting Aware Speed Selection (HASS). HASS maximizes the minimal energy level of each node, while maintaining the required performance of the network. This is useful if an energy buffer for energy intensive tasks is desirable ("emergency situation").

2.4.5. Directive-Based Energy Management

While the previous approaches are algorithms to reach energy neutrality in harvesting systems, the authors in [JTO+07] set their focus on energy policies as a set of directives that should be met at run-time in order to reach a given lifetime with battery driven

systems. The directives have priorities so that the system can disregard the least important directives if it cannot satisfy all of them. The least important directive is often an optimization objective such as "maximize sensing rate". It is restricted to one optimization only, because the authors state, that it is difficult to optimize two objectives with no logical link at the same time.

Chapter 3

Requirement Analysis

This chapter analyzes the requirements for job scheduling on energy-aware wireless sensor nodes. The demands for such a system are discussed and the energy aware algorithms presented in the previous chapter are analyzed. Finally, the concept of performance levels are introduced.

3.1. Demands for Wireless Sensor Networks

Perpetual operation is the most important thing to ensure for a wireless sensor node. This leads to the demand of energy neutrality (cf. 2.4 on page 11). Finding appropriate system settings manually is difficult, because it is threatened by software errors or wrong assumptions about energy consumption and node work load as reported in Sect. 2.1.2 on page 6. Due to the fact that this can never totally excluded, the system should counteract against drainage by lowering the performance dynamically. A lower performance is much better than a short period of high performance followed by a system failure.

Yet, dynamic performance adaption provides even more comfort: Conventionally, the optimal measurement rate has to be found by iterative evaluation and testing. This is not be necessary, as a rough estimation for dimensioning the hardware suffices for the system to run at best effort. While this is also true for non-harvesting systems (cf. Sect. 2.4.5 on page 16), it is more important for harvesting systems, because of the high variance in harvesting potential. A set of settings that is necessary in times with

low harvesting potential in order to prevent system failure would underperform in times with high harvesting potential.

Therefore, the second objective is to maximize the systems performance. This term is application specific: In some deployments the sampling rate of a sensor is to be maximized, while the transmission of data can be deferred (cf. potato field in 2.1.2 on page 6). In other deployments an immediate message transmission is essential (cf. hill slide in 2.1.2). Another example would be different sensors with different energy consumption and different precision. The imprecise sensor should be used at low energy conditions, the better sensor at good energy conditions. This can be indefinitely continued. Thus, an easy specification of the needs is to be aimed at.

The approach by Kansal et al. splits time into discharging times, where the consumption is minimized, and in charging times, where the consumption is maximized (considering energy neutrality). For solar powered systems, this leads to low performance at night and high performance at day. As the authors of the LQ-Tracker state, this effect is often unwanted. In contrast they ague that a low duty-cycle variance is useful in many application settings, e.g., as additional measurements at day often cannot compensate missing measurements at night. In particular it is problematic if time-specific patterns are emerging (e.g., high performance at day, low performance at night):

As an example, consider the Great Duck Island deployment with the goal of observing behavioral patterns of birds (cf. 2.1.2 on page 6): If the sensor node misses the bird only because it has a low measurement rate, this could lead to the wrong assumption that the animal was more often at this place at day than at night. It would be possible to interpolate between measurement points, but a higher measurement rate at night would be much better, even though the measurement rate at day had to be lowered.

3.2. Analysis of Hardware Requirements

The system model for the hardware used in this thesis is depicted in Fig. 3.1 on the facing page. The hardware was developed in the Institute of Telematics at Hamburg University of Technology and is presented in [RJT09].

In contrast to the hardware used by Kansal et al., it uses a supercapacitor rather than a NiMH batteries as energy buffer. This has consequences: A supercapacitor

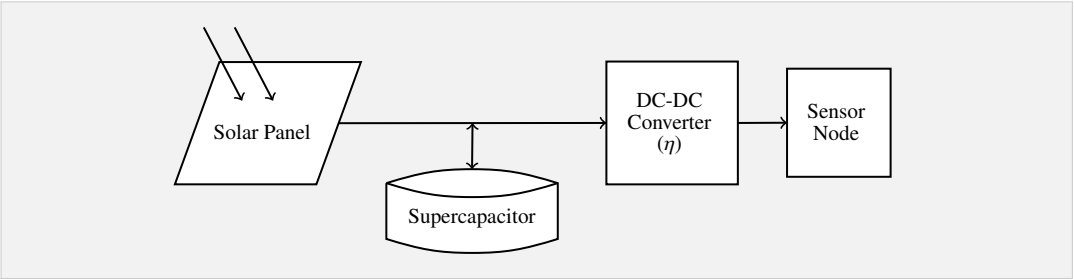


Figure 3.1.: System model of the used hardware.

has a much higher efficiency (99 % according to [SM11]) and only a simple, efficient and cheap charging circuit is needed: A Zener diode to prevent overcharging and a Schottky diode to avoid discharging over the solar panel at night suffices. Therefore, there is no need for a special battery bypass (cf. Sect. 2.4.1 on page 12), which is the fundament of the algorithm by Kansal et al.

Furthermore, as the solar panel is directly connected to the supercapacitor, the power increases with the capacitor voltage at a fixed solar current. This can be assumed, since the point where the current of the solar panel decreases is higher than the breakdown voltage of the supercapacitor and will therefore never reached.

This implies that the more energy is already buffered, the more energy is harvested. It introduces a nonlinearity (cf. Sect. 4.2.1 on page 28). The algorithm by Moser et al. needs a formulation as a linear program (cf. Sect. 2.4.3 on page 15). Therefore, it can not be used for this platform. The energy aware algorithm developed in this thesis allows for such a nonlinearity. In fact it supports every hardware whose energy course can be modeled by an (at least implicit solvable) differential equation.

The LQ-Tracker is not based on a system model, therefore it can be applied to this hardware and will be compared with the algorithm developed in this thesis in Sect. 6.2 on page 46.

3.3. Performance Levels for Energy-Aware Scheduling

Most existing algorithms are based on continuous reward functions, often directly coupled with a given duty cycle. This is acceptable for simple systems, e.g. sampling a single sensor, but will be less user-friendly in case of a more complex system, because

the application designer has to formulate a complicate reward function as dependencies between different jobs arise. Those dependencies can be easily formulated in form of performance levels:

A performance level consists of a set of settings resulting in different energy consumption. Each performance level has an specified reward, therefore the performance level with the highest reward that complies with the energy conditions is chosen.

As an example a traffic monitoring application is introduced: A sensor node repeatedly takes an image of a road and transmits it to a web server, so that drivers can plan its driving route accordingly. In order to reduce the amount of transmitted images, a sensor node counts the number of cars and only transmits the image if the difference between the current number of cars and the one of the last transmitted image reaches or exceeds a threshold ΔC_{th}. Furthermore, the system should send a small alive-packet (consuming much less energy than sending an image) in times with low image rates, so that the operator knows, that the system is still running.

There are three adjusting screws to adapt the systems performance to the available energy: Rate of image acquisition, rate of alive packets and the threshold. In order to let the operator adjust those settings easily, they are written into a table such as 3.1. Each row defines a performance level. The higher the level is, the better is the performance of the system, but the higher is the energy consumption.

Reward	Image acquisition rate	Alive packet rate	ΔC_{th}
5	60	0	4
4	20	0	10
3	8	6	20
2	3	12	0[1]
1	0	12	0

Table 3.1.: Example for a performance table, rates are in number per hour

For the prediction of future energy consumption the execution rate of each job (image acquisition, alive packet transmission, image processing, and image transmission) has to be predicted. The rate of the first two is directly given. The rate of the image processing is given by the rate of image acquisition. The rate of the image transmission has to be estimated by taking the acquisition rate and the transition

[1] Saves the image processing at such a low acquisition rate.

probability from one number of cars to another into account. The algorithm should choose the maximum performance level that the sensor node can meet for a defined time interval without violation of energy neutrality.

A performance level λ can be composed of any settings that could affect energy consumption. This includes fixed execution rates, depending execution rates, but also other settings like radio duty cycle or transmission power, as long as the system is able to calculate the consumed energy per hour. The series of energy levels $\mathcal{L} = \langle 1, \ldots, \lambda_{max} \rangle$ has a monotonically increasing energy consumption. Every given set of performance levels can be automatically transferred into a series that achieves this: At first the energy consumption of each level is calculated (cf. 4.3 on page 35). Then the levels are sorted by this energy consumption. Levels with lower benefit, but higher energy consumption than another level are deleted.

Chapter 4

Energy-Aware and Prediction-Based Scheduling

This chapter describes the algorithm for energy-aware scheduling in sensor networks developed in this thesis. It is based on discrete performance levels as introduced in the previous chapter.

The algorithm determines the highest performance level for the next prediction horizon that complies with the energy constraints given by energy policies. Voltage courses resulting from different performance levels are offered to the energy policy, which accepts or rejects it. A voltage course is given by a series of voltages at each slot beginning:

$$\widetilde{\mathcal{V}} = \left\langle \widetilde{V}_c(\tau_0), \ldots, \widetilde{V}_c(\tau_{N-1}) \right\rangle \tag{4.1}$$

where τ_s is the starting time of slot s as presented in Sect. 2.3 on page 9. Slot s_0 is the slot at the beginning of a period. For solar powered nodes with a period of one day s_0 would start at 0:00. The voltage change within one slot is calculated as presented in 4.2 on page 28. It is executed for each slot of the prediction horizon, while the voltage calculated in the last slot serves as input for the calculation in the next slot.

The algorithm is executed at each slot start, as soon as a new prediction exists. If it would be possible to predict the future (within a given horizon, e.g., a day) perfectly, one execution per prediction horizon suffices. In order to react on prediction errors

(both in energy intake as in energy consumption), the algorithm decisions must be checked and possibly revised. This is called receding horizon control.

At all times holds $V_c > 0$, because the DC-DC converter will cease running below a minimum voltage V_{min} and no more energy is consumed. Furthermore, the voltage can not rise above the breakdown voltage V_{max} of the capacitor, because this is avoided by a protective circuit. The simulation takes this into account by resetting V_c to V_{max} if this voltage is exceeded.

4.1. Energy Policies

The following policies are derived from the demands presented in Sect. 5.2.3 on page 43.

EnergyNeutrality

At first it is important to achieve perpetual operation. This implies that the supercapacitor never runs out of energy. Therefore the policy has to include

$$EnergyNeutrality(\widetilde{\mathcal{V}}) \quad := \quad \forall\, \tau : \widetilde{V}_c(\tau) > V_{alarm}. \tag{4.2}$$

V_{alarm} should be chosen slightly higher than V_{min} in order to protect from sudden changes in energy intake. As long as this predicate holds, the system maintains energy neutrality (cf. Sect. 2.4 on page 11). It is depicted in Fig. 4.1.

ReachTop

In order to utilize the full voltage range (resulting in better energy efficiency (cf. Sect. 4.2.1 on page 28), a second policy pictured in Fig. 4.2 on the facing page is introduced that requests a full energy buffer within the prediction horizon:

$$ReachTop(\widetilde{\mathcal{V}}) \quad := \quad \exists\, \tau : \widetilde{V}_c(\tau) = V_{max} \quad \wedge \quad EnergyNeutrality(\widetilde{\mathcal{V}}) \tag{4.3}$$

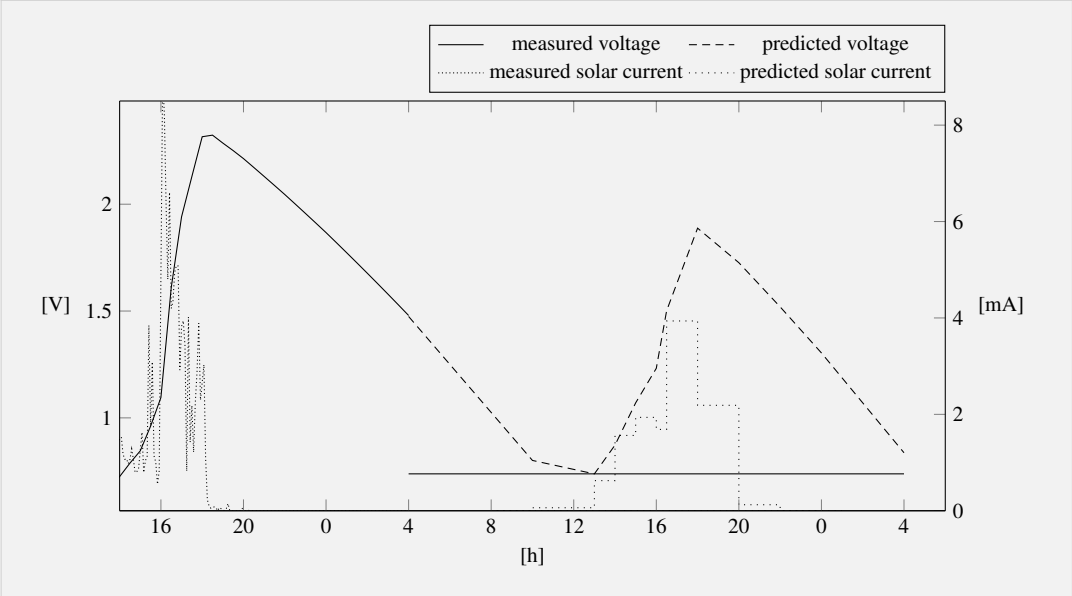


Figure 4.1.: The EnergyNeutrality policy requests a non empty energy buffer.

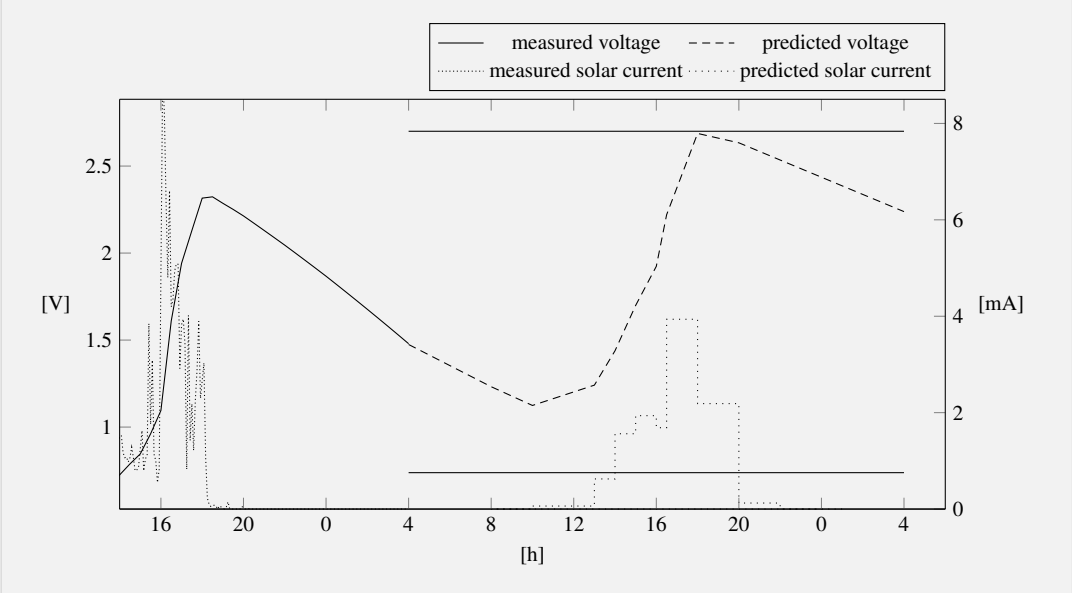


Figure 4.2.: The ReachTop policy requests the energy to be full once within the prediction period, while ensuring energy neutrality.

Equalize

A third policy does not depend on a static goal, but rather tries to utilize the full energy consumed in the upcoming prediction horizon by requesting the same voltage at the end of the prediction horizon as depicted in Fig. 4.3.

$$Equalize(\widetilde{\mathcal{V}}, V_c, s) \;:=\; V_c = \widetilde{V}_c(\tau_{s+N}) \;\wedge\; EnergyNeutrality(\widetilde{\mathcal{V}}) \tag{4.4}$$

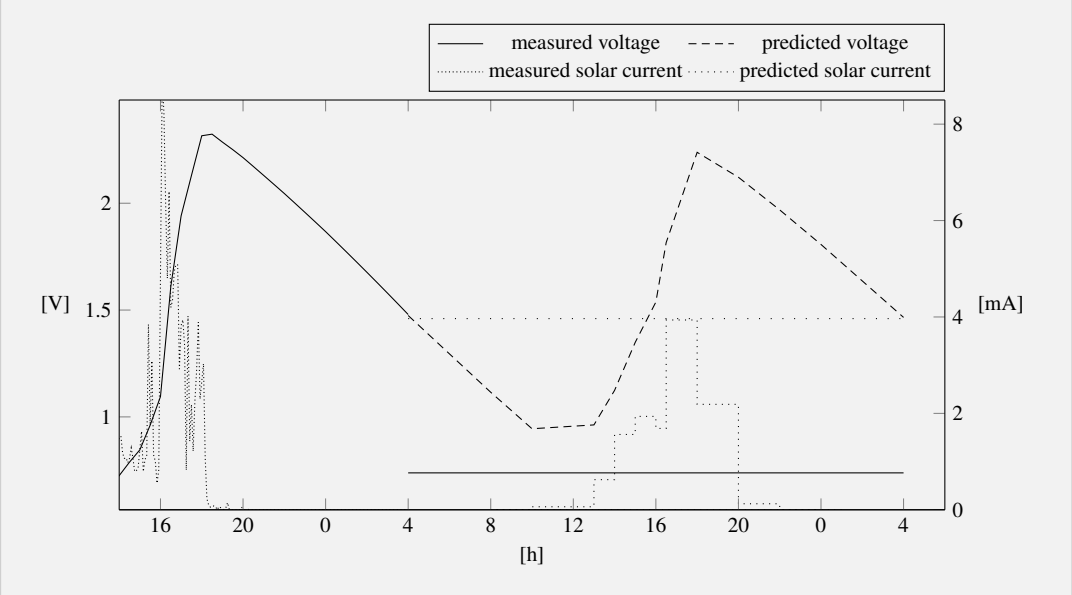


Figure 4.3.: The Equalize policy requests the voltage to be equal at the end of the prediction horizon, while ensuring energy neutrality.

4.2. Energy-Progress Simulation

For simulating the voltage course at a given performance level, a differential equation is derived. It will be interpreted in context of prediction and solved.

4.2.1. System Model

The hardware consists of a solar panel directly connected to an electric double-layer capacitor which powers the microcontroller with a DC-DC converter (cf. Sect. 3.2 on page 20). A simplified circuit is provided in Fig. 4.4.

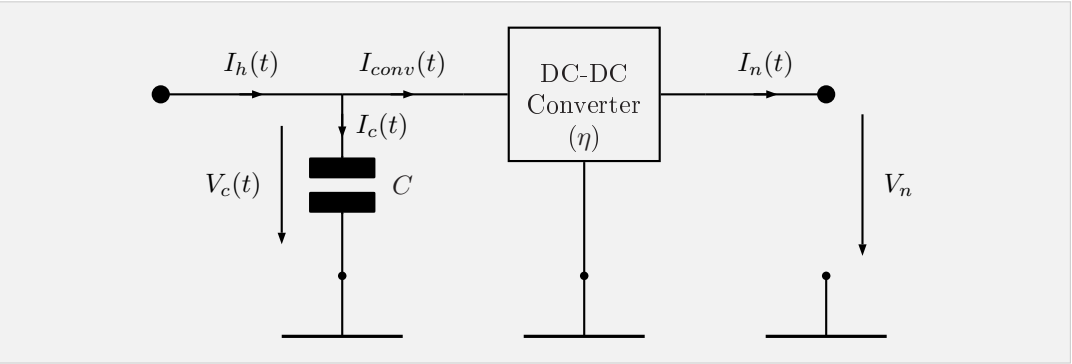


Figure 4.4.: Energy Supply Circuit

At time t, a current $I_h(t) \geq 0$ is harvested. At the same time the microcontroller consumes the current $I_n(t) \geq 0$ at a constant output voltage V_n which is provided by the DC-DC converter. The current into the capacitor $I_c(t)$ is given by Kirchhoff's current law as

$$I_c(t) = I_h(t) - I_{conv}(t). \tag{4.5}$$

Note that $I_h(t)$ is negative, if $I_{conv}(t)$ exceeds $I_h(t)$. The incoming power of a DC-DC converter equals the outgoing power reduced by the efficiency η:

$$I_{conv}(t) \cdot V_c(t) \cdot \eta = I_n(t) \cdot V_n = P_n(t) \tag{4.6}$$

Inserting (4.6) into (4.5) yields

$$I_c(t) = I_h(t) - \frac{P_n(t)}{V_c(t) \cdot \eta}. \tag{4.7}$$

The increase of the energy stored in the capacitor equals the power at the capacitor:

$$\begin{aligned} \frac{\mathrm{d}E_c(t)}{\mathrm{d}t} &= I_c(t) \cdot V_c(t) \\ &\overset{(4.7)}{=} I_h(t) \cdot V_c(t) - \frac{P_n(t)}{\eta} \end{aligned} \tag{4.8}$$

This confirms the proposition that the harvested energy increases with rising state of charge (cf. Sect. 3.2 on page 20). The energy stored in a capacitor is directly related to the capacitor voltage by

$$\frac{1}{2} \cdot C \cdot V_c(t)^2 = E_c(t) \tag{4.9}$$

Differentiation yields the differential equation for the voltage at the capacitor:

$$C \cdot V_c(t) \cdot \frac{\mathrm{d}V_c(t)}{\mathrm{d}t} = \frac{\mathrm{d}E_c(t)}{\mathrm{d}t} \tag{4.10}$$

$$\frac{\mathrm{d}V_c(t)}{\mathrm{d}t} = \frac{1}{C} \cdot \left(I_h(t) - \frac{P_n(t)}{\eta \cdot V_c(t)} \right) \tag{4.11}$$

This equation is the basis for forecasting the voltage course (4.1) on page 25. It's solution will be subsequently developed.

4.2.2. Adaptions for Simulation

Due to the nature of common, state-of-the-art prediction schemes for energy intake, the equation can be solved on a per-slot basis. Recall from 2.3 on page 9 that the prediction of $\widetilde{I}_h(t)$ is constant within a slot.

$$\widetilde{I}_h(t) = const. \quad \tau_s \leq t < \tau_s + l_s \tag{4.12}$$

Node power $P_n(t)$ varies between a few µA in times while the microcontroller is in sleep state up to several mA in times of activity (e.g., calculation, radio). The times of high power are short compared to the duration of the sleep states while being equally distributed in time. In order to provide a feasible simulation, $P_n(t)$ is assumed to be constant during a slot.

In times of $I_h(t) = 0$ this induces no error because from (4.8) it follows that

$$\frac{\mathrm{d}E_c(t)}{\mathrm{d}t} = -\frac{P_n(t)}{\eta} \tag{4.13}$$

$$E_c(\tau_{s+1}) = E_c(\tau_s) - \int_{\tau_s}^{\tau_{s+1}} \frac{P_n(u)}{\eta}\,\mathrm{d}u. \tag{4.14}$$

Thus the stored energy at time t only depends on the integral of the power and not on its temporal distribution. This does not hold if $I_h(t) \neq 0$, as the harvested energy is dependent on the voltage of the supercapacitor (cf. (4.8)).

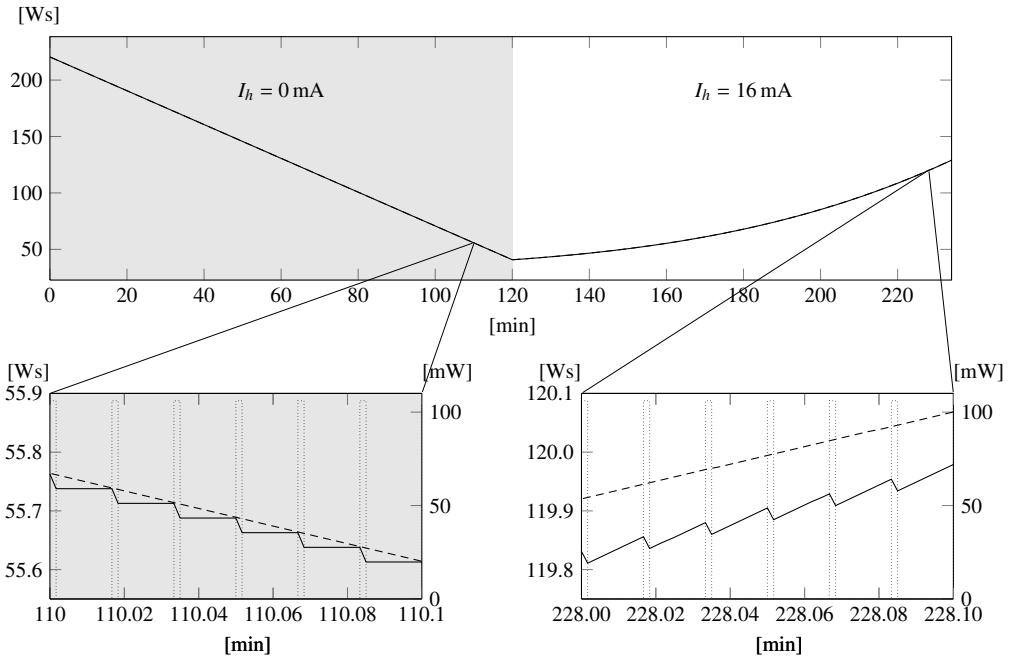


Figure 4.5.: Comparison of assumed (dashed) and real energy course (solid) of the energy buffer. In the upper plot, they are not distinguishable. The lower plots show magnifications of the upper plot and illustrate different errors at different harvesting currents. The dotted line represents the consumed power.

In Fig. 4.5 a sample energy course of two hours with $I_h(t) = 0\,\mathrm{mA}$ followed by two hours with $I_h(t) = 16\,\mathrm{mA}$ is shown. The simulation (dashed line) with constant $\overline{P_n}(t)$ and the energy course with variable energy consumption (solid line) both show the same course on the upper figure. Only with higher resolution, a difference can be seen. The real energy course is staircase-shaped, while the simulation generates a steady course. At $I_h(t) = 0\,\mathrm{mA}$, both courses meet once a step, indicating no error that builds up over time. At $I_h(t) = 16\,\mathrm{mA}$, an error can be detected, because in reality, the energy drops at each execution cycle, while in the remaining time only the harvester influences the supercapacitor. Though the same energy is drawn by the microcontroller, the buffered energy (and therefore the voltage) is lower most of the time within a cycle. This results in lower power by the harvester. The error builds up over time, but the steps are small enough ($\approx 0.04\,\mathrm{W\,s}$) to produce a absolute error

smaller 0.1 W s withing two hours, corresponding to an absolute voltage error as small as 10^{-3} V. In this application such an error is non significant, because achievable voltage measurement resolution is in the same order of magnitude (for a 10 bit ADC).

4.2.3. Solution of Differential Equation

With the made assumptions introduced in the last section, (4.11) is a first order nonlinear ordinary differential equation:

$$\frac{\mathrm{d}y}{\mathrm{d}t} = a - \frac{b}{y} \qquad \text{with } y(t) = \widetilde{V}_c(t),\ a = \frac{\widetilde{I}_h}{C},\ b = \frac{\overline{P}_n}{\eta \cdot C} \tag{4.15}$$

The coefficients a and b are constant (cf. previous section) and positive, because otherwise the sensor node is a consumer and the harvester only produces energy. Furthermore, $y > 0$ as the voltage will not drop below V_{alarm}. Given the current voltage $y(\tau_s) = V_c(\tau_s)$, the solution of the differential equation at time τ_{s+1} can be divided into 3 cases:

Equilibrium operation

In this case the energy consumption and energy intake are balanced:

$$\frac{\widetilde{I}_h}{V_c(\tau_s)} = \frac{\overline{P}_n}{\eta} \quad \Rightarrow \quad a = \frac{b}{y} \tag{4.16}$$

The harvested energy is directly used and the energy of the capacitor is kept constant. From (4.15) it follows that

$$\Rightarrow \quad \frac{\mathrm{d}y}{\mathrm{d}t} = 0 \quad \Rightarrow \quad V_c(\tau_{s+1}) = V_c(\tau_s) = \text{const.} \tag{4.17}$$

Discharging only

In times where the harvester produces no energy ($\widetilde{I}_h = 0 \ \Rightarrow \ a = 0$), (4.15) boils down to

$$\frac{\mathrm{d}y}{\mathrm{d}t} = -\frac{b}{y} \tag{4.18}$$

This can be solved by separation of variables:

$$\begin{aligned} y \cdot \frac{\mathrm{d}y}{\mathrm{d}t} &= -b \\ \int_{\tau_s}^{\tau_{s+1}} s \cdot \frac{\mathrm{d}s}{\mathrm{d}u}\,\mathrm{d}u &= \int_{\tau_s}^{\tau_{s+1}} -b\,\mathrm{d}u \\ \int_{y(\tau_s)}^{y(\tau_{s+1})} s\,\mathrm{d}s &= \int_{\tau_s}^{\tau_{s+1}} -b\,\mathrm{d}u \\ \frac{1}{2}y(\tau_{s+1})^2 - \frac{1}{2}y(\tau_s)^2 &= -b \cdot (\tau_{s+1} - \tau_s) \\ y(\tau_{s+1}) &= \sqrt{y_0^2 - 2bl_s} \end{aligned} \tag{4.19}$$

Back substitution yields

$$V_c(\tau_{s+1}) = \sqrt{V_c(\tau_s)^2 - 2\frac{\overline{P}_n l_s}{\eta C}}. \tag{4.20}$$

Mixed operation

The last case contains all other physical possibilities, in which the harvesting device provides power not equal to the consumed power. In fact it contains two separate cases: If energy intake is greater than energy consumption, the supercapacitor will monotonously charge. If more energy is consumed than provided by the harvester, the amount of buffered energy monotonously decreases. Both cases are strictly separated, as they will diverge from the equilibrium point with increasing speed. Therefore, it can be assumed, that each occurrence of $a - \frac{b}{y}$ will have the same sign at a given a and b (thus $ay - b$, too, as $y > 0$). Furthermore, it is assumed, that $a \geq 0$ and $a - \frac{b}{y} \neq 0$, because these cases are handled above. The solution can be found by separation of variables:

$$\begin{aligned} \frac{\mathrm{d}y}{\mathrm{d}t} &= a - \frac{b}{y} \\ 1 &= \frac{1}{a - \frac{b}{y}} \cdot \frac{\mathrm{d}y}{\mathrm{d}t} \\ \int_{\tau_s}^{\tau_{s+1}} 1\,\mathrm{d}u &= \int_{y(\tau_s)}^{y(\tau_{s+1})} \frac{1}{a - \frac{b}{s}}\,\mathrm{d}s \end{aligned} \tag{4.21}$$

$$
\begin{aligned}
\tau_{s+1} - \tau_s &= \int_{y(\tau_s)}^{y(\tau_{s+1})} \frac{s}{as-b}\,\mathrm{d}s = \int_{y(\tau_s)}^{y(\tau_{s+1})} \frac{as-b+b}{a\cdot(as-b)}\,\mathrm{d}s \\
l_s &= \int_{y(\tau_s)}^{y(\tau_{s+1})} \frac{as-b}{a\cdot(as-b)} + \frac{b}{a\cdot(as-b)}\,\mathrm{d}s \\
l_s &= \int_{y(\tau_s)}^{y(\tau_{s+1})} \frac{1}{a}\,\mathrm{d}s + \int_{y(\tau_s)}^{y(\tau_{s+1})} \frac{b}{a\cdot(as-b)}\,\mathrm{d}s \\
l_s &= \frac{y(\tau_{s+1})}{a} - \frac{y(\tau_s)}{a} + \frac{b}{a^2}\int_{y(\tau_s)}^{y(\tau_{s+1})} \frac{a}{as-b}\,\mathrm{d}s \\
a\cdot l_s &= y(\tau_{s+1}) - y(\tau_s) + \frac{b}{a}\cdot\ln\left(\frac{ay(\tau_{s+1})-b}{ay(\tau_s)-b}\right)
\end{aligned}
$$

This implicit equation cannot be solved explicitly for $y(\tau_{s+1})$. It is necessary to use a numerical approximation, such as Newton's method:

$$
\begin{aligned}
y_{n+1} &= y_n + \frac{f(y_n)}{f'(y_n)} \qquad (4.22)\\
\text{with}\quad f(y_n) &= y_n + \frac{b}{a}\cdot\ln\left|\frac{ay_n-b}{ay_0-b}\right| - y_0 - a\cdot l_s \\
\text{and}\quad y_0 &= y(\tau_s) \\
\Rightarrow\quad y_{n+1} &= y_n + \frac{y_n + \frac{b}{a}\cdot\ln\left(\frac{y_n-\frac{b}{a}}{y_0-\frac{b}{a}}\right) - y_0 - a\cdot l_s}{1+\frac{b}{a}\left(y_n-\frac{b}{a}\right)^{-1}} \qquad (4.23)
\end{aligned}
$$

As starting value $y_0 = y(\tau_s) = V_c(\tau_s)$ was chosen, because it serves as an rough approximation to the resulting voltage as high voltage changes are rare. The terms $\psi := \frac{b}{a} = \frac{\overline{P}_n}{\eta\cdot\widetilde{I}_h}$ and $\phi := y(\tau_s) + a\cdot l_s = V_c(\tau_s) + \frac{\widetilde{I}_h l_s}{C}$ are constant during the iterations, so they can be precalculated.

$$
y_{n+1} = y_n + \frac{y_n + \psi\cdot\ln\left|\frac{y_n-\psi}{y_0-\psi}\right| - \phi}{1+\psi\left(y_n-\psi\right)^{-1}} \qquad (4.24)
$$

This calculation will be executed until the target accuracy ε is reached:

$$
\varepsilon > \left|y_{n+1} - y_n\right|. \qquad (4.25)
$$

It is chosen as $\varepsilon = 0.01\,\mathrm{V}$, resulting from the resolution of the voltage measurement. This results in approximately 4 calculation steps.

For the sake of completeness, back substitution according to (4.15) yields

$$V_{cn+1} = V_{cn} + \frac{V_{cn} + \frac{\overline{P}_n}{\eta \cdot \widetilde{I}_h} \cdot \ln \left| \frac{V_{cn} - \frac{\overline{P}_n}{\eta \cdot \widetilde{I}_h}}{V_{c0} - \frac{\overline{P}_n}{\eta \cdot \widetilde{I}_h}} \right| - V_{c0} - \frac{\widetilde{I}_h l_s}{C}}{1 + \frac{\overline{P}_n}{\widetilde{I}_h \cdot \eta} \cdot \left(V_{cn} - \frac{\overline{P}_n}{\eta \cdot \widetilde{I}_h} \right)^{-1}} \tag{4.26}$$

4.3. Energy Consumption

The mean power in level λ can be calculated as T

$$\overline{P}_n = \sum_{j \in \mathcal{J}} r_{\lambda,j} \cdot E_j. \tag{4.27}$$

with $\mathcal{J}$ being the set of jobs, E_j being the average energy consumed by a job for a single execution and $r_{\lambda,j}$ being the execution rate of job j in level λ. The rates build up the matrix $\mathbf{R}$ with Λ being the set of levels:

$$\mathbf{R} = (r_{\lambda,j}),\ \mathbf{R} \in \mathbb{N}^{|\Lambda| \times |\mathcal{J}|} \tag{4.28}$$

Some jobs like communication have a small jitter in the energy consumption. For example it is dependent on channel load or link quality. This is taken into account by filtering the measured value $E_{j,\text{measured}}$ by a moving average

$$E_j \leftarrow \ \alpha \cdot E_j + (1 - \alpha) \cdot E_{j,\text{measured}} \quad \alpha \in [0, 1]. \tag{4.29}$$

α is chosen in such way, that only long-term variations (e.g., environmental changes) affect the result. Due to the fact, that the jitter is small most of the time, $\alpha = 0.5$ is a fair choice.

4.4. Algorithm

By means of the simulation, the optimal level of the set of levels Λ is found by binary search as described in Alg. 1 on the following page. The compliance with the policy

(cf. 4.1 on page 26) is used as search criterion. If no valid level can be found, the lowest level is chosen, implying that this level must describe a level in which it is unlikely to drain the supercapacitor until better energy conditions are experienced. For solar harvesting, operation for a few days should be guaranteed.

Data: Set of levels Λ, Series of slot lengths $\mathcal{L}$ and predictions of energy intake $\widetilde{\mathcal{I}}_h$, current capacitor voltage V_c, matrix of execution rates R, current time t_{curr} and slot s_{curr}, vector of energy consumptions $E_{\mathcal{J}} = \left(E_0, \ldots, E_{|\mathcal{J}|-1}\right)^T$
Result: Optimal level for the current prediction horizon

$\lambda_{lower} \leftarrow 1$
$\lambda_{upper} \leftarrow \lambda_{max}$
while $\lambda_{lower} < \lambda_{upper}$ **do**
  $\lambda \leftarrow \left\lfloor \frac{\lambda_{lower} + \lambda_{upper} + 1}{2} \right\rfloor$
  $s \leftarrow s_{\text{curr}}$
  $\tau \leftarrow t_{\text{curr}}$
  while $\tau < t_{curr} + T \ \wedge \ V_c >= V_{alarm}$ **do**
    $s \leftarrow (s+1) \bmod N$
    $\tau \leftarrow \tau + \mathcal{L}[s]$
    $\overline{P}_n \leftarrow \sum_{j \in \mathcal{J}} r_{\lambda,j} \cdot E_j$
    if $\widetilde{I}_h \tau_s = 0$ **then**
      $V_c \leftarrow \sqrt{V_c(\tau_s)^2 - 2\frac{\overline{P}_n l_s}{\eta C}}$
    else
      $V_c(\tau_s) \leftarrow V_c$
      $\psi \leftarrow \frac{\overline{P}_n}{\eta \cdot \widetilde{I}_h[s]}$
      $\phi \leftarrow V_c(\tau_s) + \frac{\widetilde{I}_h[s]\mathcal{L}[s]}{C}$
      repeat
        $V_{c,\text{old}} \leftarrow V_c$
        $V_c \leftarrow V_c + \frac{V_c + \psi \cdot \ln\left|\frac{V_c - \psi}{V_c(\tau_s) - \psi}\right| - \phi}{1 + \psi(y_n - \psi)^{-1}}$
      until $|V_c - V_{c,old}| < \epsilon$
    if $V_c > V_{max}$ **then**
      $V_c \leftarrow V_{max}$
    $\widetilde{\mathcal{V}}[s] \leftarrow V_c$
  if Policy$(\widetilde{\mathcal{V}}, V_c, s_{curr})$ *satisfied* **then**
    $\lambda_{lower} \leftarrow \lambda$
  else
    $\lambda_{upper} \leftarrow \lambda - 1$
return λ_{upper}

Algorithm 1: Algorithm for finding the optimal performance level

Chapter 5

Software Design and Implementation

The software is implemented in nesC for the TinyOS platform. The use of TinyOS arises from its support for the used IRIS platform and the availability of needed profound know-how and elaborate software modules for this system.

5.1. Design Considerations

The software consists of the following components: The energy manager for implementing the algorithm presented in Chap. 4 on page 25. It controls a set of jobs for measurement and communication. They are executed by the energy-aware scheduler. They make use of the following modules already developed in the Institute of Telematics at Hamburg University of Technology: An energy tracker for measurement of the consumed energy according to Sect. 2.2.2 on page 7, a solar predictor for partitioning the time into slots and prediction of future solar current according to Sect. 2.3 on page 9 and a hardware abstraction layer for communication with the solar harvesting board used for measuring solar current, capacitor voltage, light, and temperature.

Figure 5.1 on the next page illustrates their interaction.

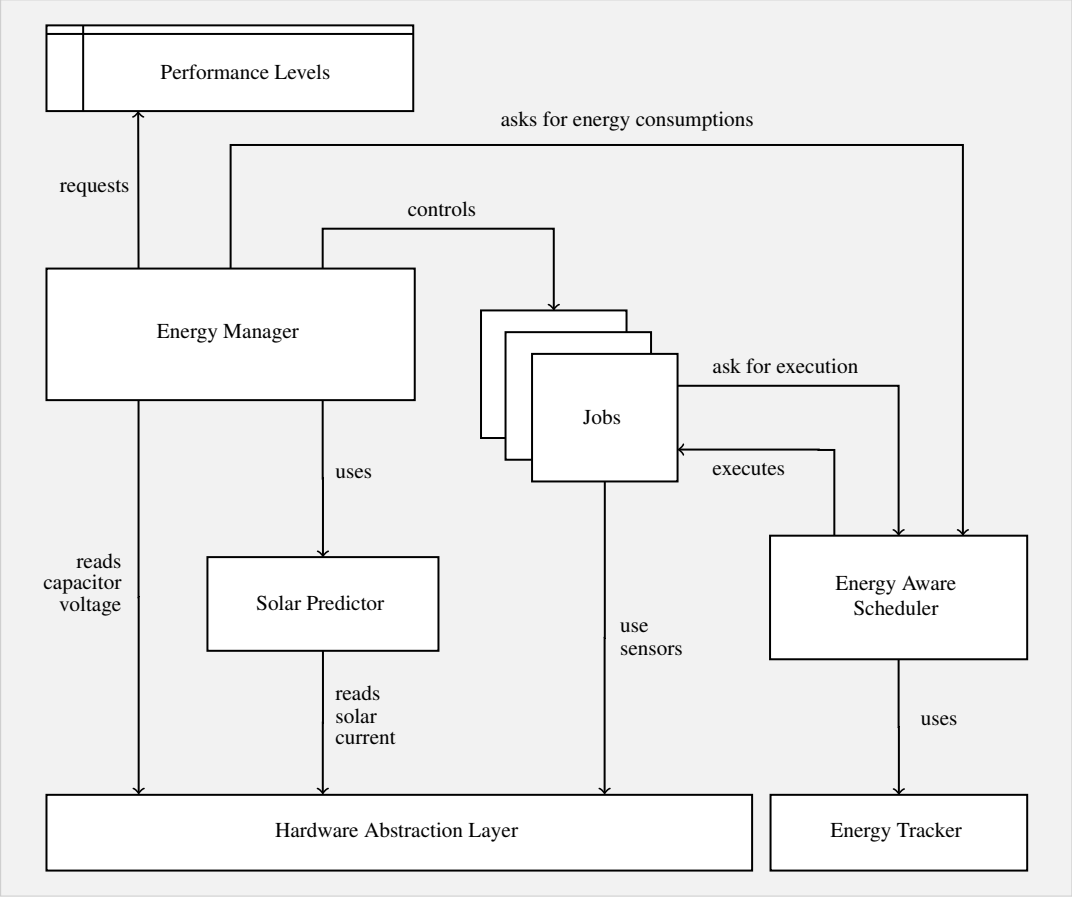


Figure 5.1.: Interaction between modules.

5.1.1. Job Concept

A job is an abstraction of an activity, such as sampling a sensor value or sending a radio packet. It is one file of source code describing the implementation of the activity. Jobs are not preemptive; a new job will be delayed until the running job is done. Of course it is possible for the microprocessor to interrupt the execution of a job for interrupt handlers for timers or other hardware. This is similar to the task concept in TinyOS, but at a higher level: A task is a sequence of commands that is scheduled as soon as no other task is executed. It is finished at the end of the command sequence, while jobs can include split phases: In a split phase a sequence of commands ends and the job waits for an interrupt (e.g., a finished sensor reading). While being in a split phase, the processor can idle, but no other job can start. In fact, a job is implemented as a composition of tasks and interrupt handlers. At first a task is called by the scheduler.

It can start one or more split phases, that will return later. The scheduler assumes the job runs until it signals the done event.

There are many kinds of jobs, some are called at a fixed rate, some (so called consumers) depend on the execution of another job (producers) or have to be scheduled before a given deadline.

Jobs are non-preemptive, to allow for an easy measurement of the consumed energy: This energy is measured by means of the energy tracker, which permanently sums up the consumed energy of all hardware components. This approach was chosen, because it provides a sufficient accuracy and does not depend on hardware support (cf. 2.2.4 on page 8).

The accumulated energy between starting and stopping the job execution is equivalent to the energy drawn by a job as long as no other energy-consuming job runs at the same time. This introduces a small delay if two jobs want to be executed at the same time, but it is negligible in the majority of cases, because job durations are small compared to the times where no job is executed. For example a job sending a packet normally takes a few milliseconds for execution. Sampling the temperature is less than one millisecond. In contrast, the execution cycle lasts from a few seconds up to hours.

5.1.2. Energy Aware Scheduler

The scheduler is responsible for executing jobs at the desired times given by the current performance level. A performance level consists of a set of settings stored in a central module. It is defined by the user as a static array of structs. For the traffic monitoring example presented in Sect. 3.3 on page 21 an excerpt of the implemented performance levels is

```
typedef struct {
  uint16_t imageAquisitionRate; // rates are given in 1/hour
  uint16_t alivePacketRate;
  uint8_t carDifferenceThereshold;
} Level;

Level levels[5];
....
levels[1].imageAquisitionRate = 3;
levels[1].alivePacketRate = 12;
levels[1].carDifferenceThreshold = 0;

levels[0].imageAquisitionRate = 0;
```

```
levels[0].alivePacketRate = 12;
levels[0].carDifferenceThreshold = 0;
```

It is necessary to map such a set of settings to an execution plan and to the predicted energy consumption. The possibility of dependencies between jobs (producer and consumer) further complicates the situation. Different design considerations were taken into account for solving this problem:

External job control ("wise scheduler/dumb jobs") would mean that applying these settings to each job is done by the scheduler. The jobs are separated into categories, e.g., periodic jobs, producer or consumer, so that the scheduler knows how to handle each job in each level. The predicted energy can be calculated efficiently because all data and all calculation resides in one place. The drawback is that the scheduler must be extended each time a new job category is introduced or a special energy prediction is needed, e.g., the calculation of the estimated image transmission rate based on the threshold (cf. Sect. 3.3 on page 21 is to be implemented in the scheduler. Moreover, it is necessary to maintain a mapping between settings and jobs.

Internal job control ("dumb scheduler/wise jobs") would mean that each job itself is responsible for maintaining its own execution plan. After an execution the job itself decides if and when it wants to be executed the next time. This point in time is sent to the scheduler. One job can access the whole set of settings and can pick the settings important for itself. This avoids the need of a mapping and each job has the possibility to calculate its own energy consumption, because it can rely on the settings for other jobs, too. Hence, it is possible that the same calculation is executed by two jobs, because both rely on the same setting. This and the fact that more data than needed is shared, results in slightly worse performance. In fact, this leads to a negligible overhead of four additional integer divisions at each slot end in the implementation.

Internal job control with proxies can be used to avoid another drawback of the internal job control: The same functionality (e.g., reposting a periodic job) must be implemented in multiple jobs making it prone to errors and produces hard to maintain source code. A job proxy acts as a job toward the scheduler and as a scheduler toward

	External	Internal	Internal with proxies
flexibility	⊖⊖	⊕	⊕⊕
maintainability	⊕	⊖⊖	⊕
performance	⊕	○	○

Table 5.1.: Different design considerations for implementing the job control

the assigned job. For example, a module provides the reposting functionality for a job which itself only sets the desired execution rate. The application designer can choose if a job fits to an existing proxy, if he should write a new proxy, or if the job is such a special job that he should implement all functionality in the job itself.

After consideration of the advantages and disadvantages as presented in Table 5.1, the internal job control with proxies was chosen for implementation.

5.2. Implementation

The implementation is written for TinyOS 2.1 based on nesC 4.5. The developed software can run in two environments: On real hardware as well as in a simulator (TOSSIM). The hardware used is a IRIS sensor node from Crossbow Technology equipped with an ATmega1281 microcontroller and an AT86RF230 radio chip.

The total memory consumption of the system (excluding the operating system) is 31800 bytes of ROM and 1800 bytes of RAM, including some modules that would not be necessary for a real-world application (e.g., test code). The implementation embraces 4000 lines of code.

5.2.1. Energy Manager

The energy manager is responsible for finding the optimal performance level for the next slot (cf. Chap. 4 on page 25). It is executed at each start of a slot.

The energy manager requests the energy consumption and execution rates from each job. Alg. 1 on page 36 is executed and the determined level is distributed to each job. The calculation is implemented in floating point arithmetics. Fixed point arithmetic would be faster, but for the price of much higher implementation effort. One execution of the algorithm takes 88 ms and draws 1666 µW s. The averaged power over the day

is 0.23 µW, since it is executed only at each slots end. For comparison: Sending one radio packet each minute draws an average power of 6.75 µW.

5.2.2. Simulation

In addition to running on real hardware, the software execution can be simulated on a desktop computer. This way it is possible to run extended periods, e.g., a whole month, within minutes giving the opportunity to test the software and find appropriate performance levels before deployment with real-world harvesting traces. Harvesting device and supercapacitor can be dimensioned this way, too.

Due to the fact that the desktop computer has no hardware interface to a harvesting device and that the software should run faster than on hardware, the hardware and the energy consumption must be simulated, too. The current capacitor voltage is recalculated perpetually. The power needed for each job was measured before by the software running on real hardware. Data with sampled solar readings serves as incoming energy. The calculation of the voltage is based on the same differential equation (4.11) on page 30, but is calculated in a more fine-grained manner and waives the simplifications in 4.2.2 on page 30, especially the consumed power is not assumed to be constant during one slot.

Each time a job starts or stops, the voltage course since the last event is calculated stepwise by means of the Runge-Kutta method. It was preferred to Newton's method used in 4.2.3 on page 32, because interim values needed for later analysis of the precise voltage course are already calculated. Furthermore, it serves as a validation for the developed algorithm (cf. Chap. 4 on page 25). Note that the developed algorithm only determines one value per slot (minutes to hours), while the Runge-Kutta method provides a new value each millisecond or less (and is correspondingly much slower).

5.2.3. Sample Jobs

Several sample jobs serve as a rule for implementing a real-world application. They include some jobs for measuring temperature, light, capacitor voltage and solar current. Furthermore, some jobs for transmitting the current systems state and measurement values to a base station are implemented. Due to the lack of appropriate camera hardware, an imaging job is simulated by playing Conway's Game of Life. The

associated transmission job has the possibility for deferred message sending (cf.). Therefore, it maintains a message queue and sends a message at the point in time with highest capacitor voltage within a given deadline or at a full queue.

Chapter 6

Evaluation

In this chapter, the developed algorithm is evaluated and compared to the LQ-Tracker (cf. Sect. 2.4.2 on page 14). The simulation covers $d = 40$ days of real-world solar current measured at the Institute of Telematics at Hamburg University of Technology. The performance levels were chosen with a linear increasing power consumption ranging from 12.8 µW to 4800 µW. For better comparability the reward was chosen as proportional to the energy consumption. The converter efficiency is $\eta = 85\,\%$. The voltage of the capacitor ranges from $V_{min} = 0.5\,\mathrm{V}$ to $V_{max} = 2.71\,\mathrm{V}$. The alarm threshold was chosen as 0.9 V.

Deferred message sending (cf. Sect. 5.2.3 on page 42) was not further reviewed, because the energy intake permits a high sampling and transmission rate of up to once a second. This quickly leads to a full message queue, even in times with poor energy conditions and a queue with ineligible size for the used hardware.

6.1. Metrics

The following metrics yield from Sect. 5.2.3 on page 43. Most important the voltage must not drop below $V_{min} = 0.5\,\mathrm{V}$. Otherwise the sensor node would stop to work.

The mean level of day x starting at t_x relative to the maximum level λ_{max} is calculated as

$$\overline{\lambda}_d(x) = \frac{1}{T} \int_{t_x}^{t_x+T} \frac{\lambda(u)}{\lambda_{max}} \, \mathrm{d}u \tag{6.1}$$

The standard deviation within a day x is

$$\sigma_d(x) = \sqrt{\frac{1}{T} \int_{t_x}^{t_x+T} \left(\frac{\lambda(u)}{\lambda_{max}} - \overline{\lambda}_d(x) \right)^2 \mathrm{d}u} \tag{6.2}$$

The mean level over all days

$$\overline{\lambda} = \frac{1}{d} \sum_{x=1}^{d} \overline{\lambda}_d(x) \tag{6.3}$$

should be as high as possible, while the average standard deviation

$$\overline{\sigma} = \frac{1}{d} \sum_{x=1}^{d} \sigma_d(x) \tag{6.4}$$

should be as low as possible, in order to prevent time specific patterns (cf. Sect. 5.2.3 on page 43). They are correlated, as can be seen in Fig. 6.1 on the facing page. It shows the above values of the results below (and some more). A high standard deviation implies better adaption to the actual situation, but leads to a less smooth operation.

6.2. LQ-Tracker

The LQ-Tracker is based on a continuous duty-cycle (and therefore energy consumption). This is not possible for a real-world application, as it can only adapted in discrete steps, while the resolution depends on the hardware. As a approximation, 200 performance levels were chosen.

Figure 6.2 on the next page shows the result of the algorithm without smoothing ($\alpha = \beta = 1$), $V_c^* = 1.5\,\mathrm{V}$ and $C = 100\,\mathrm{F}$. A high average level of 0.528 is reached, but the level is at λ_{max} most of the time when $V_c > V_c^*$ and at λ_{max} when $V_c < V_c^*$, leading to a high standard deviation of 0.344. The algorithm tries to force the voltage

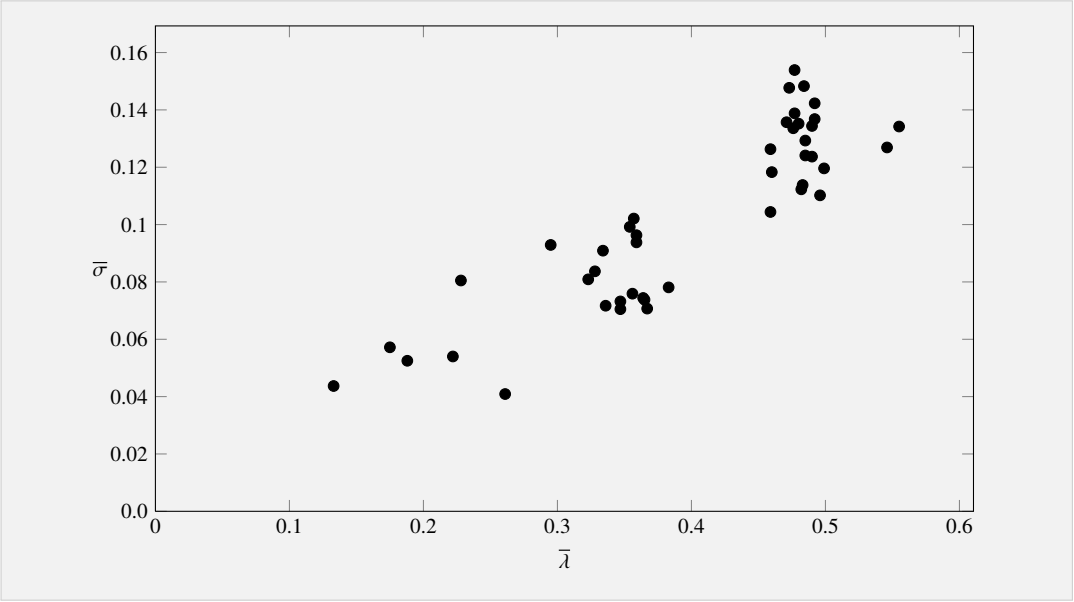


Figure 6.1.: Scatter plot showing correlation of standard deviation and mean value

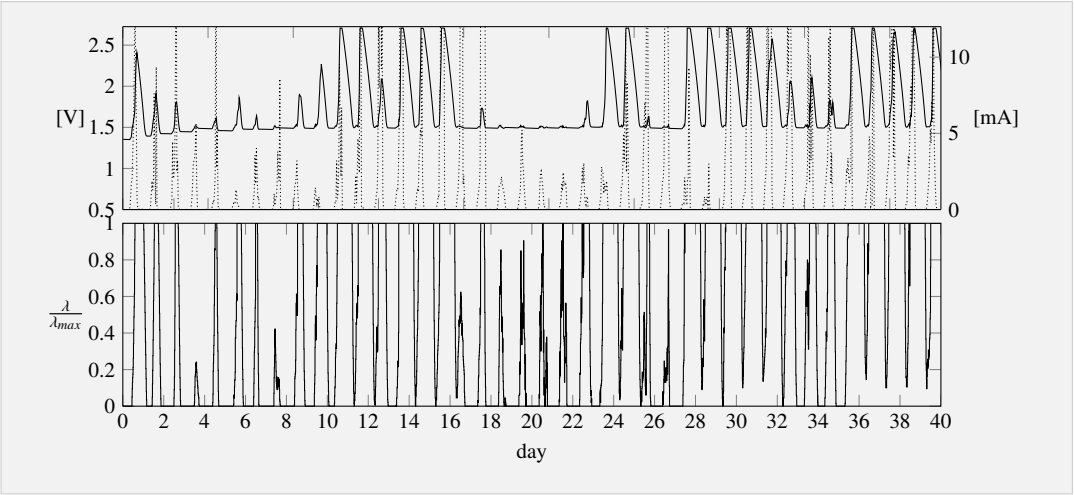


Figure 6.2.: LQ-Tracker with no smoothing. The upper plot shows the voltage at the capacitor (solid) and the harvesting current (dotted). The lower plot shows the chosen level relative to the maximum level.

to a static goal, even though it is slightly softened by use of the control law. A static voltage is a improper assumption, as the harvester shows a high variance and therefore, the voltage course has to vary a lot during a day in order to reach a low variance in the level.

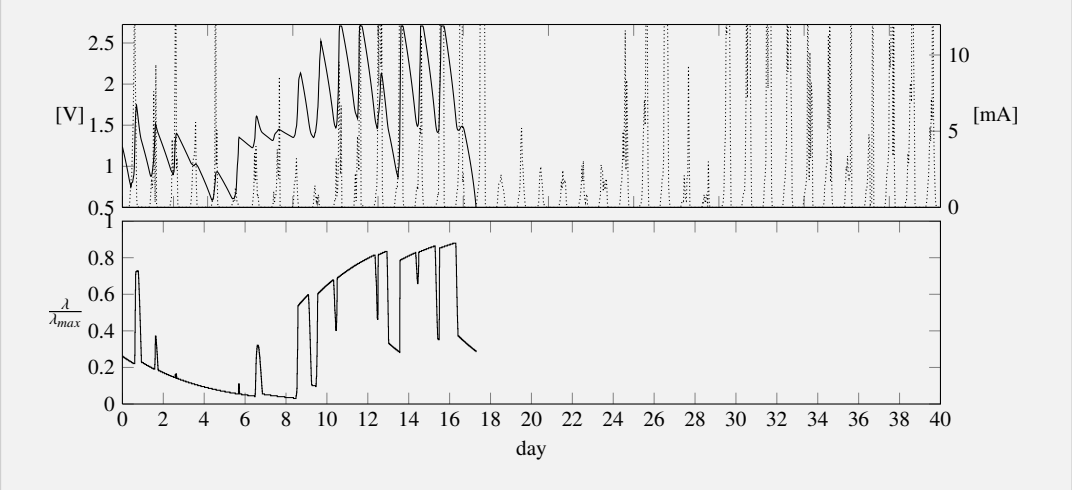


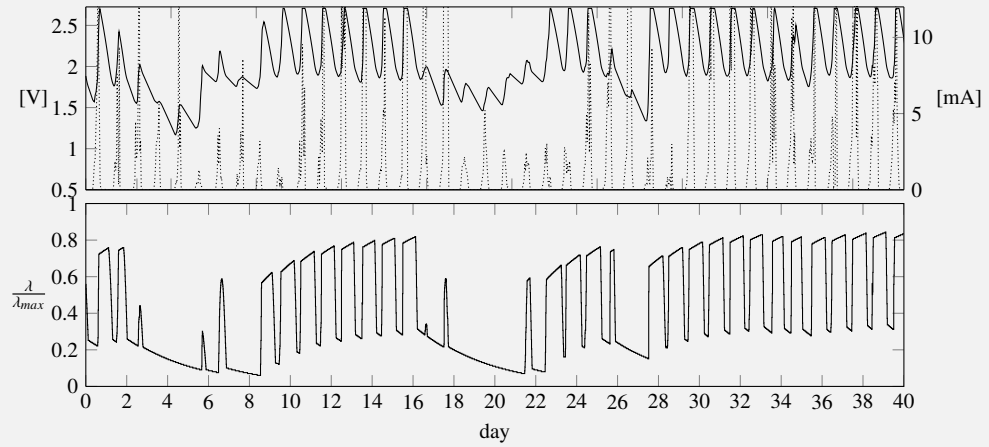
Figure 6.3.: LQ-Tracker with $\alpha = 0.001$ and $\beta = 0.5$

Therefore, the authors of [VGB07] introduced an exponentially-weighted moving average. With $\alpha = 0.001$ and $\beta = 0.5$, Fig. 6.3 occurs. It achieves a low standard deviation of 0.099 with mean 0.49, but takes quite long until it adapts to new energy conditions and most important: It results in a system failure at day 18. It was not possible to find appropriate parameters, that ensure energy neutrality as well as a smooth operation. Some exemplary results are depicted in Fig. 6.4 on the next page.

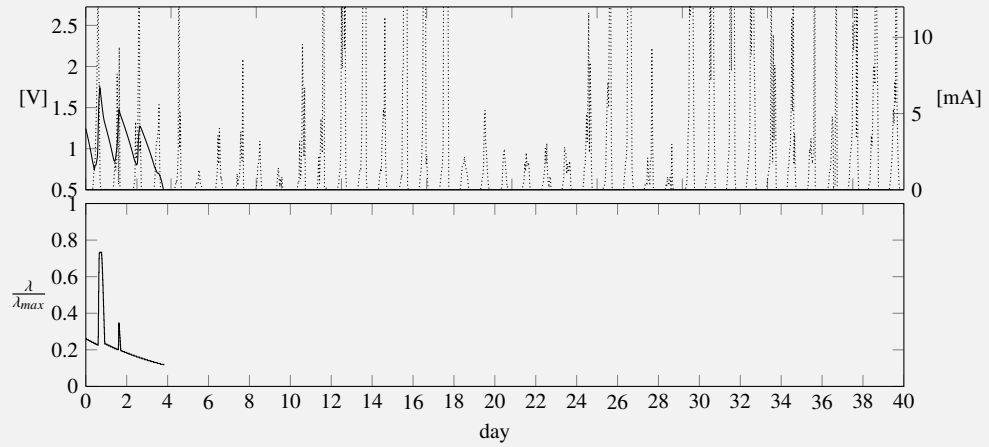
From this concludes, that the LQ-Tracker is not applicable for a system with a small energy buffer, e.g., a supercapacitor. A much bigger energy buffer would be needed in order to compensate the low adaption ability of the LQ-Tracker. Even with a bigger energy buffer, appropriate parameters have to be found.

6.3. Policy Assessment

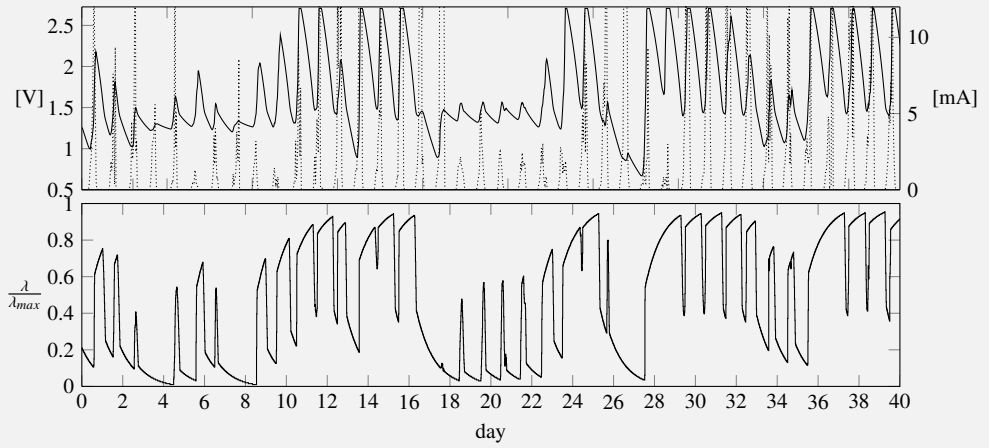
In the following, the simulation results for the algorithm developed in this thesis are presented. The simulation was executed with $C = 100\,\mathrm{F}$, an adaptive predictor with $\alpha = 0.8$ and 20 performance levels. These parameters are reviewed in Sect. 6.4 on page 52.



(a) $\alpha = 0.001$, $\beta = 0.5$ and $V_c^* = 2.0\,\mathrm{V}$



(b) $\alpha = 0.0008$, $\beta = 0.5$ and $V_c^* = 1.5\,\mathrm{V}$



(c) $\alpha = 0.004$, $\beta = 0.5$ and $V_c^* = 1.5\,\mathrm{V}$

Figure 6.4.: Behavior of LQ-Tracker at various parameters. None leads to a sustainable as well as smooth operation.

EnergyNeutrality

Figure 6.5 shows the simulation of the EnergyNeutrality policy. As requested by the policy, the voltage course returns to the minimum voltage at each day where this is possible. It tries to consume as much energy instantly, thus resulting in small time-specific patterns, while reaching $\overline{\lambda} = 0.473$ and $\overline{\sigma} = 0.148$.

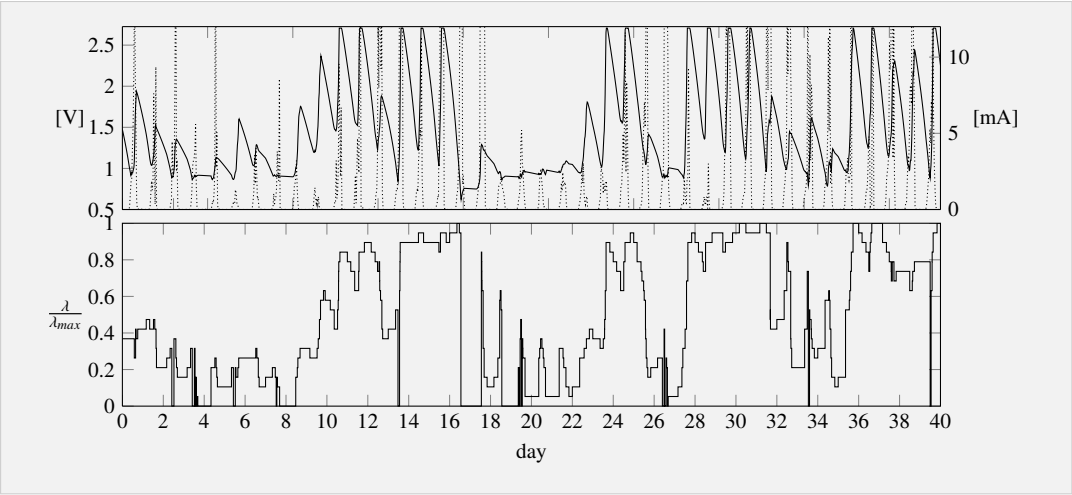


Figure 6.5.: Simulation with EnergyNeutrality

ReachTop

The second policy tries to reach the maximum voltage each day. Figure 6.6 on the facing page results from the simulation of this policy with $C = 100\,\mathrm{F}$. The resulting mean level is 0.490 and the standard deviation is 0.134. As soon as the supercapacitor is full, the performance level is inflated. This leads to a higher standard deviation, but is fine, because otherwise the energy would get wasted.

Equalize

The last policy aims to consume the same energy that will be harvested within the next prediction horizon, as it tries to reach the same voltage at the end of the prediction horizon. This is depicted in Fig. 6.7 on the next page. It does not take the preceding course into account as it does only rely on future energy intake, producing an mean value of 0.460 and a standard deviation of 0.118. It has the smallest standard deviation of all policies at the cost of the smallest mean value.

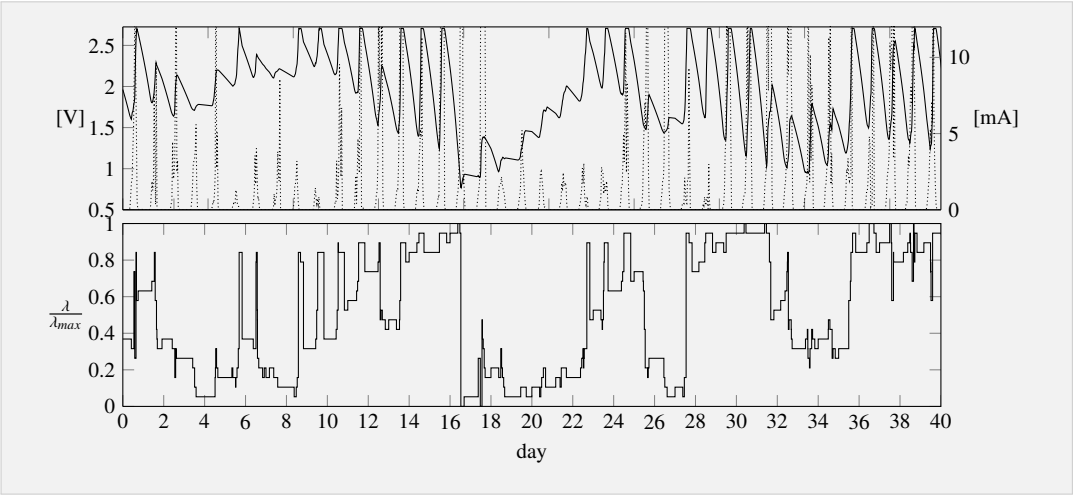


Figure 6.6.: Simulation with ReachTop

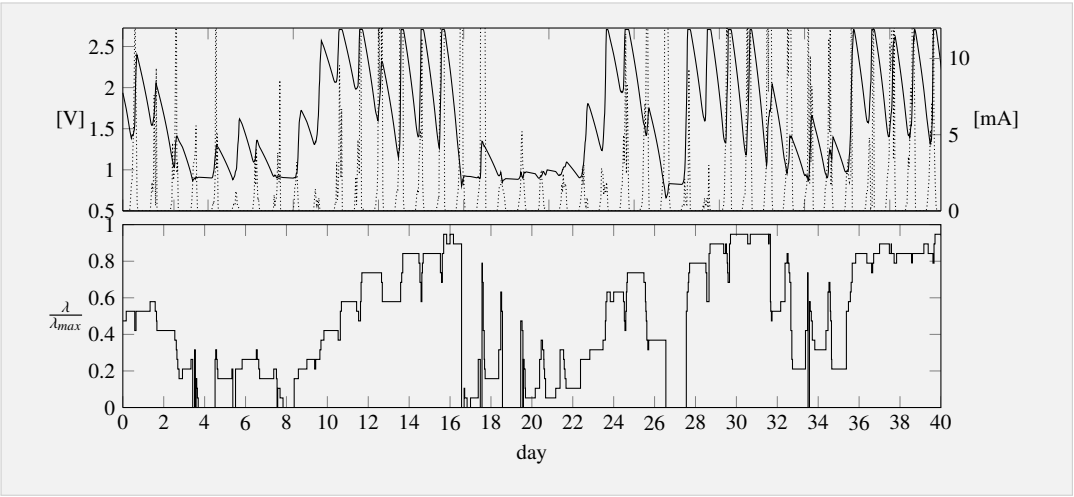


Figure 6.7.: Simulation with Equalize

Comparison

Figure 6.8 on the following page compares the different policies and they are related to the LQ-Tracker. It shows the cumulated hourly performance level. This is equal to the gained reward, because it is proportional to the level. It can be seen, that the LQ-Tracker results in a low reward, even before it runs out of energy. Equalize has a high reward in the first days, but it can not keep up this good start. ReachTop shows the best reward over time, because it stays at a high voltage most of the time, resulting in better energy efficiency.

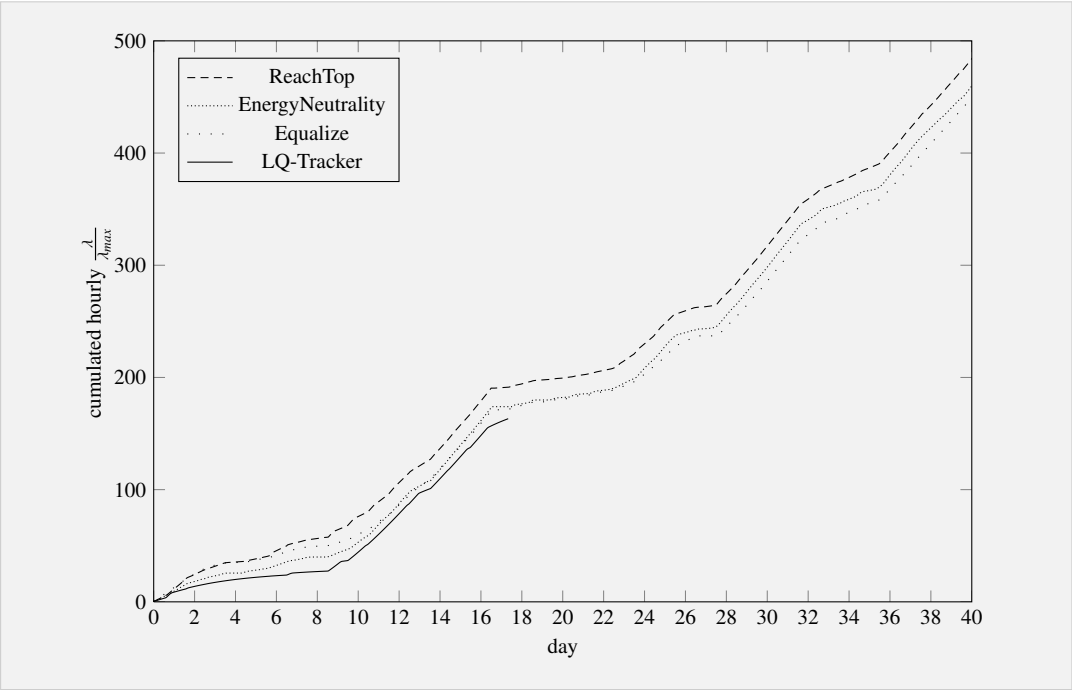


Figure 6.8.: Cumulated hourly performance level for each policy and the LQ-Tracker

6.4. Influence of System Model Parameters

The result is influenced by various parameters. They are presented and analyzed in the following. The used policy was ReachTop , because it has turned out to have the best performance for these conditions.

6.4.1. Prediction

The prediction has a great impact on the result of the algorithms. Table 6.1 compares the energy intake prediction with fixed and variable slot lengths for ReachTop with $C = 100\,\mathrm{F}$.

	Adaptive Prediction			Static Prediction		
α	0.6	0.8	0.95	0.6	0.8	0.95
$\overline{\lambda}$	0.484	0.490	0.499	0.485	0.476	0.447
$\overline{\sigma}$	0.148	0.134	0.120	0.132	0.134	0.141

Table 6.1.: Results for prediction techniques with ReachTop and $C = 100\,\mathrm{F}$

It can be seen, that the adaptive prediction results into higher levels, while maintaining a low standard deviation. The results from the fact, that the prediction error is smaller, therefore a accurate level is chosen more often.

6.4.2. Number of Performance Levels

The number of given performance levels affects how exactly the system can adapt to the current energy conditions. Table 6.2 shows the resulting mean values and average standard deviations for different number of levels. It can be seen that the relative mean level rises with the number of levels, but also the standard deviation rises. Therefore, the concrete number of levels is application specific, as it is a trade-off between a high mean level and a low standard deviation. It is an adequate procedure to start with a small number of levels, run the simulation and increment the number of levels, if the result is not acceptable.

λ_{max}	5	10	20	40
$\overline{\lambda}$	0.482	0.485	0.490	0.492
$\overline{\sigma}$	0.112	0.124	0.134	0.142

Table 6.2.: Results for different λ_{max} with ReachTop and $C = 100\,\mathrm{F}$

6.4.3. Capacity of Supercapacitor

The capacity of the supercapacitor shows a similar behavior, as can be seen in Table 6.3. It has an impact on the energy that can buffered. However, in this case a higher capacity implies higher costs.

C	25 F	50 F	100 F	150 F
$\overline{\lambda}$	0.222	0.359	0.490	0.555
$\overline{\sigma}$	0.054	0.094	0.134	0.134

Table 6.3.: Results for different capacities with ReachTop and $\lambda_{max} = 20$

6.5. Real-World Deployment

The applicability of the implemented system was tested in a real-world deployment. A preliminary software was used. Therefore, the above results can not be directly compared to the real-world application. Due to the fact, that a system failure would not be acceptable, because of the short time span to test the system, a high energy reserve of $V_{alarm} = 1.6\,\mathrm{V}$ was chosen. The used policy was Equalize while the size of the supercapacitor was 50 F and the number of performance levels was 10. In Fig. 6.9 the result from 8 days of measurement are presented, showing the principle applicability of the algorithm.

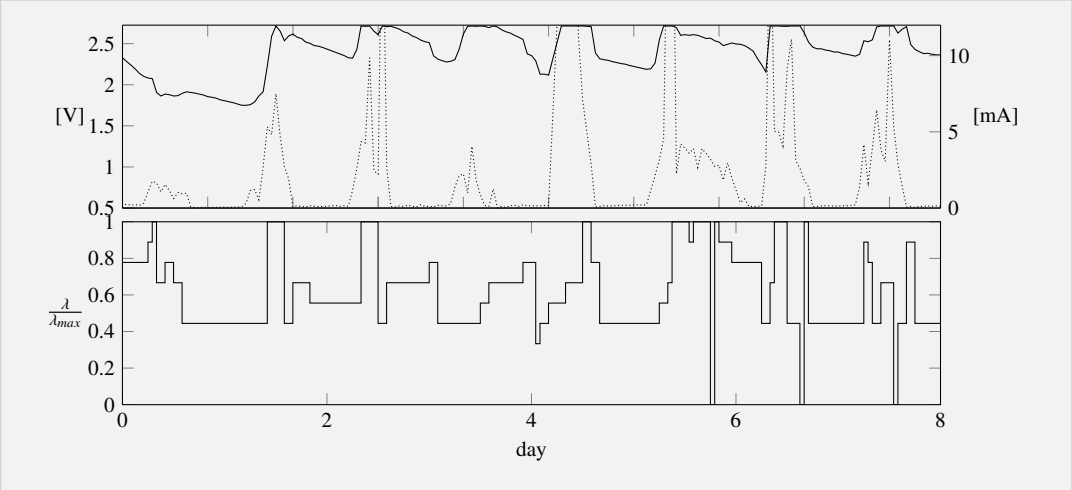


Figure 6.9.: 8 days of real-world deployment

6.6. Limitations

This chapter demonstrates the applicability to the target hardware in reaching energy neutrality and adequate low standard deviation. However, it is not possible to derive general guidelines for parameter justification, because a trade-off between accurate adaptation and smooth operation must be made. Furthermore, a good prediction of future energy is needed. Therefore, the algorithm shows a problematic behavior at successive days having a high deviation in energy intake, as the system will likely consume too much or too few energy.

Chapter 7

Conclusion

Wireless sensor networks are widely-used for non-intrusive data-gathering. The advantages are the tiny size of the sensor nodes, an easy deployment and low costs. Yet, the lifetime of a sensor network depends on the available energy. Purely battery driven sensor nodes will refuse to run as soon as the battery is empty. Therefore, many research has been carried out in order to prolong a sensor node's lifetime. Continuing on this, a further approach is the use of energy harvesting devices, that harvests energy from the environment. This allows for sustainable operation.

However, the size of the harvesting device should comply with the tiny size of the sensor nodes. Therefore, the sensor nodes should be equipped with equally tiny harvesting devices. Since most harvesting sources, e.g., the sun, yield a varying, unsteady energy intake, the missing energy in times of poor harvesting conditions should be balanced by adapting the systems performance dynamically. Hence, online energy management, i.e., job scheduling and duty cycle adaption, is mandatory for achieving perpetual and depletion-safe operation.

In this thesis a new scheduling approach was developed and implemented in order to reach this goal for a given hardware platform. Therefore, existing state-of-the-art algorithms were analyzed at first. From existing real-world sensor network deployments, demands were derived and formulated. A user-friendly specification of the system designer's demands is achieved by the introduction of discrete performance levels. Many existing algorithms show a lack of a user-friendly specification of demands,

but rather depend on a single linear reward function. This is appropriate for simple applications, but can not be applied to more complex user demands.

These demands and the existing algorithms were compared showing some deviations that lowers the adaptability to other hardware or user demands. This is especially important, because the hardware used in this thesis includes a nonlinearity, because the harvested energy depends on the current energy stored in a supercapacitor. The algorithm can handle such a nonlinearity making it applicable to a wide range of hardware, since it depends on a mathematical model of the target platform as a differential equation. The algorithm solves this differential equation online with sufficient accuracy, while taking the low computational power of a sensor node into account.

The algorithm is dependent on the prediction of future energy intake exploiting the varying nature of the environment. Based on these predictions, the voltage course of the energy buffer is simulated and different performance levels are investigated for their compatibility with defined energy policies, such as a nonempty energy buffer.

The second part of this thesis addresses the implementation of a complete energy-aware framework that can run on state-of-the-art sensor nodes and was tested in a real-world deployment. Besides the use on sensor nodes, the framework can simulate the behavior of the system offline. This was used to evaluate the developed algorithm, as well as it can serve as a tool for a system designer in order to find appropriate settings for his deployment.

The algorithm developed in this thesis was compared to the LQ-Tracker, a state-of-the-art scheduling approach. It was shown, that the LQ-Tracker is not applicable on a system with a small energy buffer. The evaluation demonstrates, that the developed algorithm can adapt the system performance to the energy conditions quite well. One direction of future research would be to enhance the algorithm for gaining an even smoother operation, e.g., by avoiding recalculation if the predicted and the real voltage course do not differ considerabily. Furthermore, the scheduling of infrequent, but energy-consuming operations, e.g., calibration tasks, should be considered as their energy efficiency varies with the time of execution.

The framework developed in this thesis aims at application in single-hop networks with direct radio connection to a base station. Future work has to be done in order to take energy conditions of a multi-hop network into account: Decisions concerning energy consumption may affect other sensor nodes. Therefore, these decisions should

be made in a network-wide manner, e.g., sensor nodes not acting as routers and with improving energy conditions should not raise their sampling rate if forwarding nodes have poor energy conditions and thus do not support forwarding additional radio packets.

Bibliography

[DFPC08] Prabal Dutta, Mark Feldmeier, Joseph Paradiso, and David Culler. Energy Metering for Free: Augmenting Switching Regulators for Real-Time Monitoring. In *Proceedings of the International Conference on Information Processing in Sensor Networks*, IPSN '08, St. Louis, Missouri, USA, April 2008.

[DHJ+06] Prabal Dutta, Jonathan Hui, Jaein Jeong, Sukun Kim, Cory Sharp, Jay Taneja, Gilman Tolle, Kamin Whitehouse, and David Culler. Trio: Enabling Sustainable and Scalable Outdoor Wireless Sensor Network Deployments. In *Proceedings of the International Conference on Information Processing in Sensor Networks*, IPSN '06, Nashville, Tennessee, USA, April 2006.

[DOTH07] Adam Dunkels, Fredrik Österlind, Nicolas Tsiftes, and Zhitao He. Software-Based On-Line Energy Estimation for Sensor Nodes. In *Proceedings of the IEEE Workshop on Embedded Networked Sensors*, EmNets '07, Cork, Ireland, June 2007.

[JTO+07] Xiaofan Jiang, Jay Taneja, Jorge Ortiz, Arsalan Tavakoli, Prabal Dutta, Jaein Jeong, David Culler, Philip Levis, and Scott Shenker. An Architecture for Energy Management in Wireless Sensor Networks. *ACM Special Interest Group on Embedded Systems (SIGBED) Review*, 4, July 2007.

[KHZS07] Aman Kansal, Jason Hsu, Sadaf Zahedi, and Mani B Srivastava. Power Management in Energy Harvesting Sensor Networks. *ACM Transactions on Embedded Computing Systems (TECS)*, 6, September 2007.

[KSS+07] V. Kyriatzis, N. S Samaras, P. Stavroulakis, H. Takruri-Rizk, and S. Tzortzios. Enviromote: A New Solar-Harvesting Platform Prototype for Wireless Sensor Networks / Work-in-Progress Report. In *Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, PIMRC '07, Athens, Greece, September 2007.

[KV86] P. R. Kumar and Pravin Varaiya. *Stochastic Systems: Estimation, Identification and Adaptive Control*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1986.

[LBV06] Koen Langendoen, Aline Baggio, and Otto Visser. Murphy Loves Potatoes: Experiences from a Pilot Sensor Network Deployment in Precision Agriculture. In *Proceedings of the International Conference on Parallel and Distributed Processing*, IPDPS '06, Rhodes Island, Greece, April 2006.

[MPE+06] Kirk Martinez, Paritosh Padhy, Ahmed Elsaify, Gang Zou, A. Riddoch, Jane K. Hart, and H. L. R. Ong. Deploying a Sensor Network in an Extreme Environment. In *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, SUTC '06, Taichung, Taiwan, June 2006.

[MTBB07] Clemens Moser, Lothar Thiele, Davide Brunelli, and Luca Benini. Adaptive Power Management in Energy Harvesting Systems. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '07, Nice, France, April 2007.

[MTBB08] Clemens Moser, Lothar Thiele, Davide Brunelli, and Luca Benini. Robust and Low Complexity Rate Control for Solar Powered Sensors. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '08, Munich, Germany, March 2008.

[OII+07] Takahiro Ono, Hiroki Isizuka, Kanoko Ito, Yasuyuki Ishida, Shohei Miyazaki, Oru Mihirogi, and Yoshito Tobe. UScan: Towards Fine-Grained Urban Sensing. In *Proceedings of the International Workshop on Real Field Identification*, RFId '07, Tokyo, Japan, November 2007.

[PBAR09] J. R Piorno, C. Bergonzini, D. Atienza, and T. S Rosing. Prediction and Management in Energy Harvested Wireless Sensor Nodes. May 2009.

[Ram09] Maneesha V. Ramesh. Real-Time Wireless Sensor Network for Landslide Detection. In *Proceedings of the International Conference on Sensor Technologies and Applications*, SENSORCOMM '09, Athens/Glyfada, Greece, June 2009.

[Ren10] Christian Renner. Energy-budgeting sensor networks with renewable energy sources. In *Proceedings of the SenSys Doctoral Colloquium*, SenSys DC, Zurich, Switzerland, November 2010.

[RJT09] Christian Renner, Jürgen Jessen, and Volker Turau. Lifetime Prediction for Supercapacitor-powered Wireless Sensor Nodes. In *Proceedings of the GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze"*, FGSN '09, Hamburg, Germany, August 2009.

[RKH+05] Vijay Raghunathan, Aman Kansal, Jason Hsu, Jonathan Friedman, and Mani Srivastava. Design Considerations for Solar Energy Harvesting Wireless Embedded Systems. In *Proceedings of the International Symposium on Information Processing in Sensor Networks*, IPSN '05, Los Angeles, California, April 2005.

[SDS10] Thomas Schmid, Prabal Dutta, and Mani B. Srivastava. High-Resolution, Low-Power Time Synchronization an Oxymoron no More. In *Proceedings of the International Conference on Information Processing in Sensor Networks*, IPSN '10, Stockholm, Sweden, April 2010.

[SM11] R. Saravana Kumar S. Millika. Review on Ultracapacitor Battery Interface for Energy Management System. *International Journal of Engineering and Technology (IJET)*, 3(1), February 2011.

[SMP+04] Robert Szewczyk, Alan Mainwaring, Joseph Polastre, John Anderson, and David Culler. An Analysis of a Large Scale Habitat Monitoring Application. In *Proceedings of the International Conference on Embedded Networked Sensor Systems*, SenSys '04, Baltimore, MD, USA, November 2004.

[TPS+05] Gilman Tolle, Joseph Polastre, Robert Szewczyk, David Culler, Neil Turner, Kevin Tu, Stephen Burgess, Todd Dawson, Phil Buonadonna, David Gay, and Wei Hong. A Macroscope in the Redwoods. In *Proceedings of the International Conference on Embedded Networked Sensor Systems*, SenSys '05, San Diego, California, USA, November 2005.

[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 IEEE Communications Society Conference on Sensor, Mesh, and Ad Hoc Communications and Networks*, SECON '07, San Diego, California, USA, June 2007.

[ZSA10] Bo Zhang, Robert Simon, and Hakan Aydin. Energy Management for Time-critical Energy Harvesting Wireless Sensor Networks. In *Proceedings of the International Conference on Stabilization, Safety, and Security of Distributed Systems*, SSS '10, NewYork, NY, USA, September 2010.

Appendix A

Content of the DVD

The attached DVD contains the implemented framework as well as a bundle of simulation results. Finally, an electronic version of this thesis together with it's source code is contained.

The directory structure consists of the following parts: In the directory *implementation/EnergyBudget* the implementation as presented in this thesis is stored. Some other directories in *implementation* provide additional modules needed for compilation. In order to compile the implementation there exist two shell scripts *implementation/EnergyBudget/test/tossim/compile.sh* and *implementation/EnergyBudget/test/realworld/compile.sh*. Both require an installed TinyOS environment. Appropriate settings can be made inside the shell scripts. The first shell script results in an executable file *implementation/EnergyBudget/test/tossim/simulation* that writes its simulated values to standard output. The second shell script will result in a binary file that can be flashed to a sensor node as usual in TinyOS.

The second directory *simulation* includes some simulation results, while the last one *latex* contains the source code for compiling this thesis. An already compiled version of this thesis can be found in the root directory.