Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Institut für Informatik
Lehrstuhl V

SNoW-RA: Reap and Allot

Energieneutraler Betrieb des SNoW^5-Sensorknotens durch Gewinnen und Verteilen von Solarenergie

Diplomarbeit
zur Erlangung des akademischen Grades
Diplom-Informatiker (Dipl.-Inform.)
an der Fakultät für Mathematik und Informatik
der Bayerischen Julius-Maximilians-Universität Würzburg

vorgelegt von
Andreas Engel

bei
Prof. Dr. Reiner Kolla
Lehrstuhl 5 für Informatik

Würzburg, den 9. Juli 2009

Inhaltsverzeichnis

KAPITEL

1 Einleitung

1.1 Zielstellung und Aufbau der Arbeit

Ein wesentlicher Aspekt bei der Konzipierung von drahtlosen Sensornetzen ist die Energieversorgung der einzelnen Sensorknoten. Die Versorgung über eine externe, theoretisch unerschöpfliche Energiequelle (Netzteil) ist im Allgemeinen nicht realisierbar, da dies die Mobilität und den möglichen Einsatzbereich der Knoten stark einschränkt.

Stattdessen werden endliche Energiespeicher (Batterien, Akkumulatoren oder Kondensatoren) verwendet, wobei die erhöhte Mobilität mit einer Limitierung der wartungsfreien Systemlaufzeit erkauft wird. Durch die Minimierung des Energiebedarfs der einzelnen Knoten und einer gezielten Lastverteilung im Sensornetz kann die Systemlaufzeit zwar maximiert werden [30], für viele Langzeitanwendungen ist diese Art der Energieversorgung aber trotzdem ungeeignet, vor allem, wenn der manuelle Zugriff auf die Sensorknoten besonders aufwendig ist. Beispiele hierfür sind die Erfassung von Messdaten an schwer zugänglichen Stellen (etwa im All, im Gebirge oder in den Polarregionen) sowie Anwendungen mit zufällig, weitläufig oder ortsveränderlich ausgebrachten Sensoren (bspw. bei der Klimaforschung oder der Beobachtung von Tieren in ihrem natürlichen Lebensraum).

Mit der vorliegenden Arbeit wird ein Konzept zum energieneutralen Betrieb des SNoW^5-Sensorknotens [3, 19] vorgestellt. Ziel ist dabei die vollständige und dauerhafte Abdeckung des Energiebedarfs mit Solarenergie. Da diese starken zeitlichen Schwankungen unterworfen ist, bedarf es einer Anpassung des Verbrauchs an die zur Verfügung stehende Energiemenge. Die Grundlage dieser Verbrauchsadaption bildet dabei die Veränderung der Ausführungshäufigkeit periodisch anfallender Aufgaben, wie etwa das Erfassen, Verarbeiten und Speichern bzw. Versenden von Messdaten. Für Anwendungen ohne solche periodisch auszuführenden Aufgaben ist das vorgestellte System daher nicht geeignet.

Der Rest dieses Kapitels widmet sich der Identifikation der notwendigen Komponenten für den energieneutralen Betrieb. Außerdem werden die Verbrauchscharakteristiken des SNoW^5-Sensorknotens sowie die Eigenschaften der Solarenergie beschrieben. Das Kapitel schließt mit einer Übersicht der Realisierungen des Solarbetriebs verschiedener anderer Sensorplattformen.

Kapitel 2 beschäftigt sich mit den Besonderheiten bei der Ansteuerung von Solarzellen. Hierbei wird ein Regler untersucht, der den Arbeitspunkt der Zellen in Abhängigkeit von der Beleuchtungsstärke für eine maximale Leistungsausbeute optimiert. Die Arbeitsweise des Reglers wird mit einer PSpice-Simulation [7] veranschaulicht.

Für die Glättung der zeitlichen Varianz der Solarenergiequelle wird ein wiederaufladbarer Energiespeicher benötigt. In Kapitel 3 wird dafür das Konzept eines hierarchischen Energiespeichers beschrieben, der die Vorteile kleiner, kapazitiver Speicher mit denen großer, elektrochemischer Speicher kombiniert.

Kapitel 4 beschreibt das Konzept zur gezielten Anpassung des Energieverbrauchs des Sensorknotens und damit den eigentlichen Kern dieser Arbeit. In den folgenden Abschnitten 5 bis 9 wird die Implementierung der einzelnen Teilmodule vorgestellt, wobei die Realisierung und Steuerung der periodischen Aufgaben, das Erlernen ihres Energiebedarfs und die Regelung eines energieneutralen Betriebs zu den wichtigsten Aspekten gehören.

Im Kapitel 10 erfolgt die Evaluation der Implementierung und Kapitel 11 beschließt die Arbeit mit einer kurzen Zusammenfassung und Bewertung.

1.2 Grundlagen des energieneutralen Betriebs

Mobile Sensorknoten, deren wartungsfreie Laufzeit nur durch die physikalische Lebensdauer der Hardwarekomponenten begrenzt wird, müssen Energie aus ihrer Umgebung gewinnen. Abbildung 1.1 veranschaulicht das Grundprinzip eines solchen Systems, angelehnt an die Ausführungen von [9, 17]. Es besteht im Wesentlichen aus einer dynamischen Energiequelle mit einer

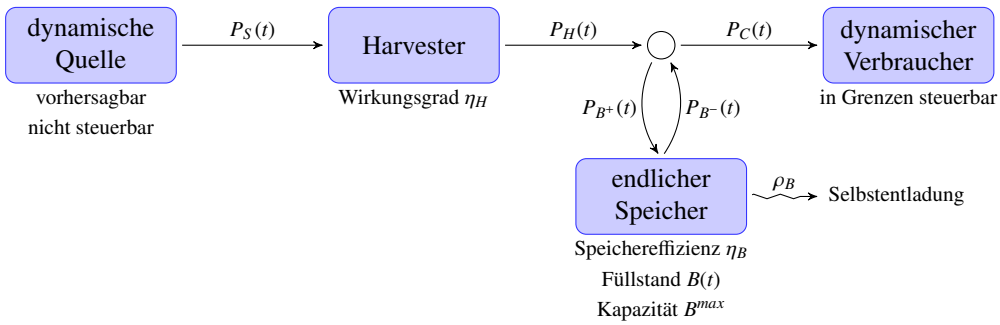


Abbildung 1.1: Versorgung eines Sensorknotens mit Umgebungsenergie

Quellleistung $P_S(t)$, einem Ernteverfahren (*Harvester*) mit Wirkungsgrad η_H zum Umwandeln der Quellleistung in elektrische Leistung $P_H(t) = \eta_H P_S(t)$ sowie dem dynamischen Verbraucher (Sensorknoten) $P_C(t)$.

Durch die unterschiedliche Dynamik von Verbraucher und Quelle entstehen Zeiten mit Leistungsüberschuss $P_{B^+}(t) = [P_H(t) - P_C(t)]^+$ sowie Leistungsdefizit $P_{B^-}(t) = [P_C(t) - P_H(t)]^+$, wobei $[x]^+ := \max(0, x)$. Diese Schwankungen müssen durch den Einsatz eines endlichen, wiederaufladbaren Energiespeichers gepuffert werden, der wiederum Verluste in Form von Selbstentladung ρ_B und Speichereffizienz η_B verursacht.

Die Entwicklung des Speicherfüllstandes während eines endlichen Zeitintervalls T kann daher als

$$B(t+T) = B(t) + \int_t^{t+T} (\eta_B P_{B^+}(\tau) - P_{B^-}(\tau) - \rho_B)\, d\tau \tag{1.1}$$

beschrieben werden. Das System arbeitet energieneutral, wenn

$$0 \le B(t) \le B^{max} \qquad \forall\, t \ge 0. \tag{1.2}$$

In [17] werden hinreichende Bedingungen für die Energieneutralität angegeben. Dafür müssen zunächst die langfristige mittlere Leistung ρ_H bzw. ρ_C sowie die maximalen Abweichungen σ_H^1 und σ_H^2 bzw. σ_C von diesem Mittelwert bestimmt werden. Dabei muss für jedes Zeitintervall $I = [t, t+T]$

$$\int_I P_H(\tau)\, d\tau \;\in\; [\rho_H T - \sigma_H^1,\ \rho_H T + \sigma_H^2] \tag{1.3a}$$

$$\int_I P_C(\tau)\, d\tau \;\in\; [0,\ \rho_C T + \sigma_C] \tag{1.3b}$$

gelten. Stimmt man nun Energiequelle, -speicher und -verbraucher so aufeinander ab, dass

$$\rho_C \;\le\; \eta_B\, \rho_H - \rho_B \tag{1.4a}$$

$$B^{max} \;\ge\; \eta_B\, (\sigma_H^1 + \sigma_H^2) + \sigma_C \tag{1.4b}$$

$$B(0) \;\ge\; \eta_B\, \sigma_H^1 + \sigma_C \tag{1.4c}$$

erfüllt wird, dann ist ein energieneutraler Betrieb gewährleistet.

1.3 Charakterisierung des Verbrauchers

Der SNoW^5-Sensorknoten [3, 19] bildet den Verbraucher des Systems, wobei verschiedene Subkomponenten zum Gesamtverbrauch beitragen.

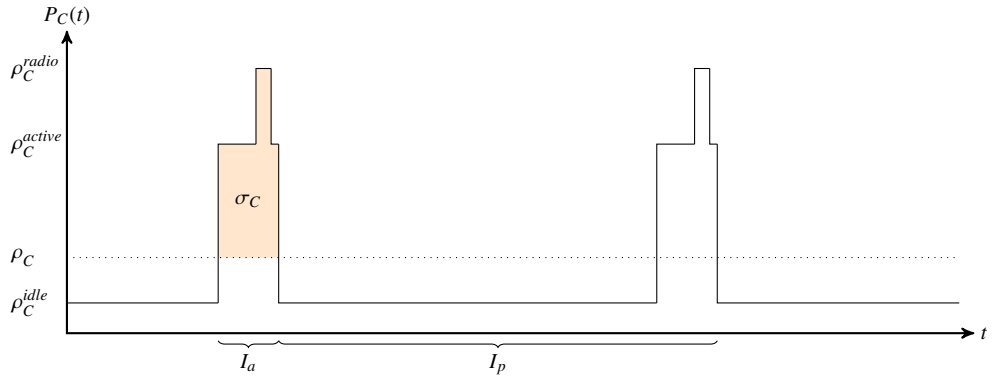


Abbildung 1.2: Ermittlung der Modellparameter für den Verbraucher

Sein MSP430-*Microcontroller* wird vom verwendeten Betriebssystem SmartOS in zwei verschiedenen Modi betrieben. Während der Ausführung von Programmcode müssen die *CPU* und der Haupttaktgeber aktiviert sein. Warten alle *Tasks* auf das Eintreten eines Ereignisses, dann wird die *MCU* in den *idle* Modus versetzt, um durch Abschalten der *CPU* und des Haupttaktgebers den Energieverbrauch während der Wartezeit zu reduzieren. Bei 3 V Versorgungsspannung und einer 8 MHz Taktung verbraucht der Sensorknoten $\rho_C^{active} = 26{,}4$ mW im aktiven Modus und $\rho_C^{idle} = 17{,}1$ mW im *idle*-Modus [2].

Eine weitere wichtige Verbraucherkomponente ist der CC1100 Funkchip für die drahtlose Kommunikation der Sensorknoten. Er erhöht den Verbrauch im aktiven Modus auf etwa $\rho_C^{radio} = 65$ mW [2], wobei der konkrete Wert von der Übertragungsrichtung, der Funksignalstärke und der Funkfrequenz abhängig ist.

Je nach Anwendung können noch weitere Verbraucher hinzukommen. Insbesondere Sensoren und Aktoren, welche über die Erweiterungsschnittstelle mit dem Knoten verbunden sind, können den Energiebedarf stark erhöhen.

Für die Ermittlung der Modellparameter des Verbrauchers nach (1.3b) ist eine genaue Kenntnis der Anwendung nötig, die vom Sensorknoten ausgeführt wird. Aus einem Leistungsprofil wie in Abbildung 1.2 kann man dann zunächst die mittlere Verbrauchsleistung

$$\rho_C = \lim_{t\to\infty} \frac{1}{t} \int_0^t P_C(\tau)\, d\tau \tag{1.5a}$$

bestimmen. Bei einem periodischen Leistungsprofil genügt dabei die Betrachtung eines beliebigen Intervalls I_p mit entsprechender Periodenlänge, wodurch (1.5a) zu

$$\rho_C = \frac{1}{|I_p|} \int_{I_p} P_C(\tau)\, d\tau \tag{1.5b}$$

vereinfacht wird. Die aktive Phase I_a, die den mittleren Verbrauch am stärksten übersteigt, bestimmt den zweiten Modellparameter zu

$$\sigma_C = \int_{I_a} (P_C(\tau) - \rho_C)\, d\tau. \tag{1.6}$$

1.4 Wahl der Energiequelle

An die Energiequelle für das autarke System werden im Wesentlichen drei Anforderungen gestellt. Zum einen muss ein Ernteverfahren in der bautechnischen Größenordnung des Sensorknotens ($50 \times 85 \times 7$ mm^3 [3]) so realisierbar sein, dass die mittlere Ausbeute nach Abzug aller Verluste den mittleren Leistungsbedarf des Sensorknotens abdeckt (vgl. Ausdruck (1.4a)). Die im Mittel erwirtschaftete Leistung ρ_H sollte daher im Bereich von 50 bis 100 mW liegen. Zum zweiten sollten die Schwankungen im zeitlichen Profil der Energiequelle, also die Modellparameter $\sigma_H^{1,2}$ möglichst gering sein, um den Energiespeicher und den damit verbundenen Energieverlust möglichst klein halten zu können (vgl. Ausdruck (1.4b)). Um die Verluste am Energiespeicher beim Puffern von Überschüssen bzw. Ausgleichen von Defiziten zu minimieren, muss das Verbrauchsprofil so gut wie möglich an den Verlauf des Energiegewinns angepasst werden. Dazu sind Vorhersagen über den zukünftig zu erwartenden Gewinn nötig, die Energiequelle muss also hinreichend genau modelliert werden können.

Die potentiellen Energiequellen sind je nach Anwendungs- und Einsatzgebiet der Sensorknoten vielfältig, jedoch werden oben genannte Anforderungen nicht immer voll erfüllt. So lässt sich grundsätzlich jede Form von mechanischer Bewegungsenergie mittels Gleichstromgeneratoren nutzbar machen (vgl. Fahraddynamo), jedoch ist die Gewinnvorhersage für windgetriebene Rotoren weitaus schwieriger als bei kontinuierlich angetriebenen Wasserrädern. Letztere hingegen sind wohl für die wenigsten Anwendungsgebiete eine mögliche Option. Das erhöhte Verschleißpotential der beweglichen Teile solcher mechanisch angetriebenen Systeme muss ebenfalls berücksichtigt werden.

Ohne bewegliche Teile kommen Thermogeneratoren aus, welche nach dem Seebeck-Effekt Energie aus einem Temperaturgradienten erzeugen. Die erforderliche Temperaturdifferenz zum Erzeugen der notwendigen Leistung ist aber recht hoch. So bietet *Micropelt* [27] einen Generator mit wenigen Quadratmillimetern Grundfläche an, der selbst unter optimalen Bedingungen bzgl. Wärmeleitung und Lastanpassung bei über 50 K Temperaturgradienten weniger als 3 mW Leistung liefert [4]. Zur Versorgung eines Verbrauchers in der Größenordnung des SNoW^5-Sensorknotens werden daher Temperaturdifferenzen von mehreren 100 K benötigt, wie sie etwa zwischen der Abgasanlage und dem Kühlkreis von Kraftfahrzeugen zu finden sind.

Ebenfalls interessant für den Einsatz in Kraftfahrzeugen sind Piezogeneratoren, die Vibrationen in Wechselspannung wandeln und diese gleichrichten. Die erzeugbaren Leistungen im Mikrowattbereich [34] sind für den SNoW^5-Sensorknoten allerdings nicht ausreichend.

Wesentlich höhere Leistungen lassen sich mittels solarthermischer und photovoltaischer Prozesse aus der Sonnenstrahlung gewinnen. Solarthermische Kraftwerke erhitzen eine geeignete Flüssigkeit mittels gebündelter Sonnenstrahlen und treiben mit dem entstehenden Dampf Turbinen an. Dieses Verfahren ist jedoch nicht effizient im Maßstab des Sensorknotens realisierbar [46]. Ohne die Zwischenstufe der kinetischen Energie wandeln dagegen Solarzellen die eingestrahlte Energie direkt in elektrische Energie, wobei die erzeugte Leistung durch die Wahl der aktiven Fläche des Solarmoduls in weiten Bereichen skaliert werden kann. Der nächste Abschnitt wird zeigen, dass Solarzellen in der Größe des SNoW^5-Sensorknotens bereits ausreichend Energie liefern können. Das größte Problem bei der Verwertung der Sonnenenergie sind die tages- und jahreszeitlichen Schwankungen ihrer Verfügbarkeit. Neben einer geeigneten Modellierung zur Vorhersage bedarf es eines großen Energiespeichers, um den im Sommer erwirtschafteten Energieüberschuss in den Wintermonaten zur Verfügung zu stellen.

Eine perfekte Energiequelle, welche die oben genannten Anforderungen voll erfüllt und für alle erdenklichen Anwendungsgebiete geeignet ist, kann es also nicht geben. Stattdessen muss von Fall zu Fall unterschieden werden, welche Energiequelle für den Einsatzort des Sensorknotens am besten geeignet ist. Die Szenarien, mit denen in Abschnitt 1.1 die Notwendigkeit des energieneutralen Betriebs motiviert wurde, sind in erster Linie *outdoor*-Anwendungen, für welche der Einsatz von Solarenergie prädestiniert ist. Im Folgenden sollen die Charakteristiken dieser Energiequelle daher näher untersucht werden.

1.5 Charakteristiken der Solarenergie

Abbildung 1.3 zeigt die bundesweite Verteilung der jährlich eingestrahlten Solarenergie pro Flächeneinheit. In unseren Breitengraden kann man demnach mit einer mittleren Bestrahlungsstärke zwischen $\frac{1100}{365 \cdot 24}$ kW/m^2 = 126 W/m^2 und $\frac{1400}{365 \cdot 24}$ kW/m^2 = 160 W/m^2 rechnen. Diese Daten basieren auf einem 10-Jahres-Mittel (1981 bis 1990) [29, 41] für direkt beleuchtete Solarmodule. Die Minderung der eingestrahlten Energie durch meteorologische Bedingungen ist dabei bereits berücksichtigt, während lokale Verschattungen durch Vegetation oder Gebäude zu weiteren Verlusten führen können.

Die Solarmodule müssen außerdem optimal ausgerichtet sein, um die effektive, also senkrecht bestrahlte Fläche zu maximieren. Die automatische Ausrichtung nach dem aktuellen Sonnenstand ist für Anwendungen im unteren Leistungsbereich nicht realisierbar, da der Mehrverbrauch für die dazu notwendigen Aktoren und deren Ansteuerung den zusätzlichen Energiegewinn übersteigt. Stattdessen muss eine fixierte Ausrichtung gefunden werden, welche die Summe der Energieeinträge über das gesamte Jahr maximiert. Auf der Nordhalbkugel sind die Solarzellen nach Süden auszurichten, um die am höchsten stehende Mittagssonne voll zu nutzen. Bei der Wahl des Inklinationswinkels α zwischen der Zellenebene und der Horizontalen muss die um $\varepsilon = 23{,}4°$ gekippte Rotationsachse der Erde berücksichtigt werden, da diese eine Variation der Höhe der Mittagssonne über dem Horizont im Laufe eines Jahres verursacht. Abbildung

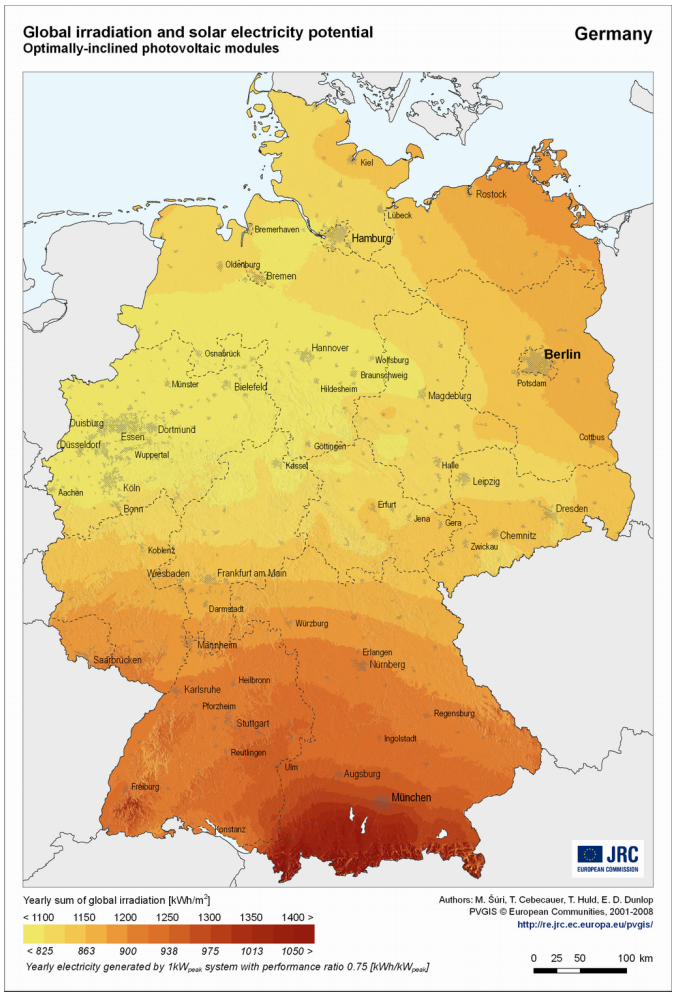


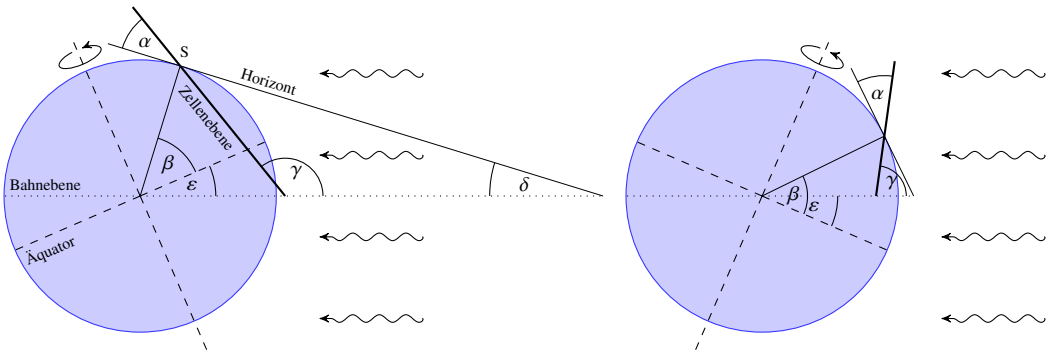
Abbildung 1.3: bundesweite Verteilung des Solarenergieeintrags im 10-Jahres Mittel [29, 41]

1.4a zeigt die Ausrichtung eines Solarmoduls im Winter am Standpunkt S mit β Grad nördlicher Breite. Die parallel zur Umlaufbahn der Erde um die Sonne verlaufenden Strahlen treffen die Solarzellen unter einem Winkel von

$$\gamma^{Winter} = 180° - \alpha - \delta = 180° - \alpha - (90° - (\beta + \varepsilon)) = 90° - \alpha + \beta + \varepsilon. \tag{1.7a}$$

Im Sommer hingegen ist die Nordhalbkugel der Sonne zugewandt (Abbildung 1.4b) und der Bestrahlungswinkel wird zu

$$\gamma^{Sommer} = 90° - \alpha + \beta - \varepsilon. \tag{1.7b}$$

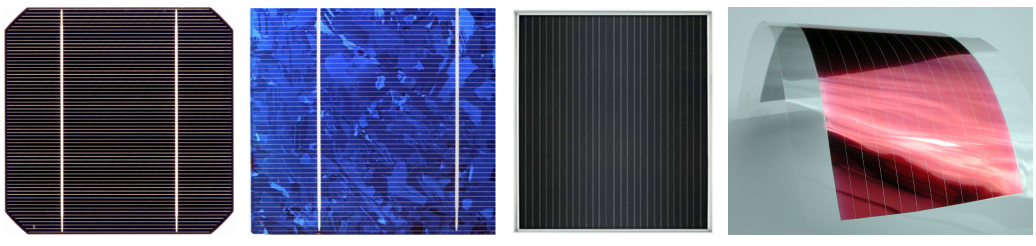


(a) Winter **(b)** Sommer

Abbildung 1.4: Einfall der Strahlen der Mittagssonne auf eine im Standpunkt S aufgestellte, nach Süden ausgerichtete und um α gegen die Horizontale geneigte Solarzelle

Für Würzburg ($\beta = 49{,}5°$) gibt [29] einen optimalen Inklinationswinkel $\alpha = 34°$ an. Damit fallen die Mittagssonnenstrahlen im Sommer unter einem Winkel von 82,1° und im Winter unter 128,9° auf die Solarzellen ein.

Ein weiterer entscheidender Faktor für den Energiegewinn ist der Wirkungsgrad η_H bei der Umwandlung der Solarenergie in elektrische Energie. Der Großteil der heute erhältlichen Solarzellen besteht aus kristallinem Silizium und weist einen Wirkungsgrad zwischen 14 % (polykrisallin, Abbildung 1.5b) und 16 % (monokristallin, Abbildung 1.5a) auf. Unter Laborbedingungen wurden aber auch schon Werte jenseits von 24 % erzielt, so dass auch für industriell gefertigte Module in naher Zukunft noch Effizienzsteigerungen zu erwarten sind. Einen wesentlich geringeren Wirkungsgrad (6 bis 8 %) haben Dünnschichtzellen aus amorphem Silizium (Abbildung 1.5c). Dieses Material ist dafür kostengünstiger produzierbar und wird besonders bei *indoor*-Anwendungen mit diffusen bzw. schwachen Beleuchtungsverhältnissen eingesetzt, da es konstruktionsbedingt weniger empfindlich auf Verschattung reagiert. Um die Produkti-



(a) **(b)** **(c)** **(d)**

Abbildung 1.5: Solarzelltypen: (a) monokristallines Silizium [40], (b) polykristallines Silizium [40], (c) amorphes Silizium [36], (d) organische Solarzellen [46]

onskosten weiter zu senken, wird inzwischen an Technologien geforscht, die ohne den teuren Rohstoff Silizium auskommen. Der Wirkungsgrad solcher organischen Solarzellen (Abbildung 1.5d) liegt jedoch noch unterhalb von 5 %. Die aktuell effizienteste Technologie stapelt III-V-Verbindungshalbleiter (bspw. Galliumarsenid) in unterschiedlichen Zusammensetzungen, um ein breites Frequenzspektrum des einfallenden Lichtes zu verwerten. Zusammen mit einer Lichtkonzentration durch Sammellinsen wurde ein experimenteller Wirkungsgrad von über 35 % erzielt [46].

Die folgende Abschätzung der Modellparameter (ρ_H, $\sigma_H^{1,2}$) des *Harvesters* basiert auf den Daten aus Tabelle A.5, welche den mittleren tageszeitlichen Verlauf der Bestrahlungsstärke $E_S^m(t)$ [W/m^2] auf ein in Würzburg aufgestelltes, nach Süden ausgerichtetes und um 34° gegen die Horizontale geneigtes Solarmodul für verschiedene Monate m beinhaltet. Für die Berechnung des Profils der daraus erzeugbaren elektrischen Leistung

$$P_H^m(t) = A_H \eta_H E_S^m(t) \tag{1.8}$$

soll eine kristalline Siliziumzelle mit einem Wirkungsgrad von $\eta_H = 15\ \%$ und einer aktiven Fläche von $A_H = 42{,}5\ \mathrm{cm}^2$ (Größe des SNoW^5-Sensorknotens) angenommen werden. Diese Werte sind für industriell gefertigte und dadurch kostengünstig einsetzbare Solarzellen realistisch.

Bedingt durch die Erdrotation und die Neigung der Rotationsachse beim Umlauf um die Sonne ist die erwirtschaftete Solarleistung starken tageszeitlichen und saisonalen Schwankungen unterworfen. Die Energieeinnahmen müssen daher über ein ganzes Jahr gemittelt werden. Wendet man die Ausdrücke

$$\rho_H^m = \frac{1}{24\ \mathrm{h}} \int_{24\ h} P_H^m(\tau)\,\mathrm{d}\tau \tag{1.9a}$$

$$\text{und} \qquad \rho_H = \frac{1}{365} \sum_{m=Jan}^{Dez} |m| \cdot \rho_H^m \tag{1.9b}$$

auf die Daten aus Tabelle A.5 an, so erhält man eine mittlere Jahresleistung von $\rho_H = 86$ mW, wenn $|m|$ die Anzahl der Tage des Monats m bezeichnet. Dies lässt genügend Spielraum für die Ineffizienzen des Energiespeichers (η_B, ρ_B), um (1.4a) für den energieneutralen Betrieb zu erfüllen.

Abbildung 1.6 zeigt die tageszeitlichen Schwankungen der vom *Harvester* gewonnenen Leistung. Während der Nacht kann keine Energie gewonnen werden, so dass das System vollständig aus dem Energiespeicher versorgt werden muss. In den Sommermonaten sind diese Phasen auf etwa acht bis zehn Stunden beschränkt, während im Winter fast zwei Drittel des Tages ohne Energieeinnahmen überbrückt werden müssen. Weiter zeigt die Abbildung, dass der dynamische Bereich der Verbrauchsleistung P_C im Sommer sehr klein im Vergleich zu den Schwankungen

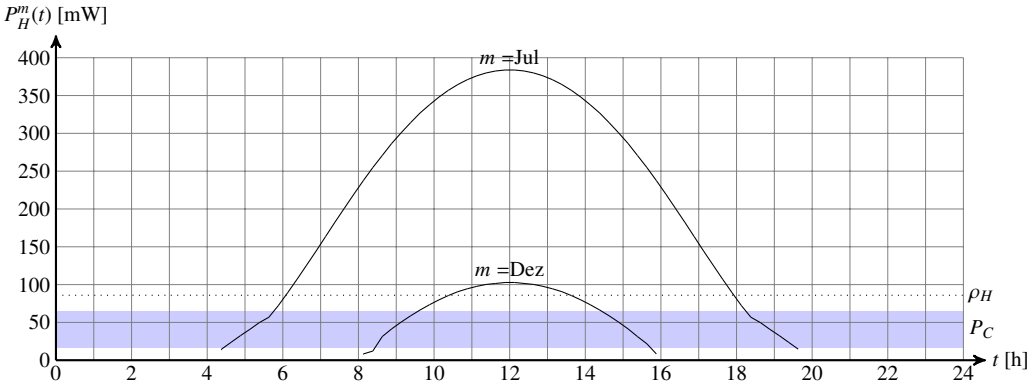


Abbildung 1.6: tägliche Variation der gewonnenen Leistung im ertragreichsten bzw. -ärmsten Monat Juli bzw. Dezember, verglichen mit dem Bereich P_C des SNoW^5-Sensorknotens

der erwirtschafteten Leistung P_H ist. Dadurch entstehen täglich Einnahmeüberschüsse, die gespeichert werden müssen, um die Defizite im Winter zu kompensieren.

Die Modellparameter $(\sigma_H^{1,2})$ für die maximalen Abweichungen der eingenommenen Energie vom Mittelwert ρ_H müssen daher aus den saisonalen Schwankungen bestimmt werden, welche in Abbildung 1.7 dargestellt sind. Da ρ_H die im Jahresmittel erwirtschaftete Leistung beschreibt, müssen sich die Flächen zwischen ρ_H und dem Leistungsprofil gerade ausgleichen. Das kritische Intervall, in dem überdurchschnittlich viel Energie gewonnen wird, erstreckt sich von April bis September, so dass die Modellparameter mit

$$\sigma_H := \sigma_H^1 = \sigma_H^2 = \sum_{m=Apr}^{Sep} |m| \cdot 24\ \text{h} \cdot (\rho_H^m - \rho_H) = 164\ \text{Wh} \tag{1.10}$$

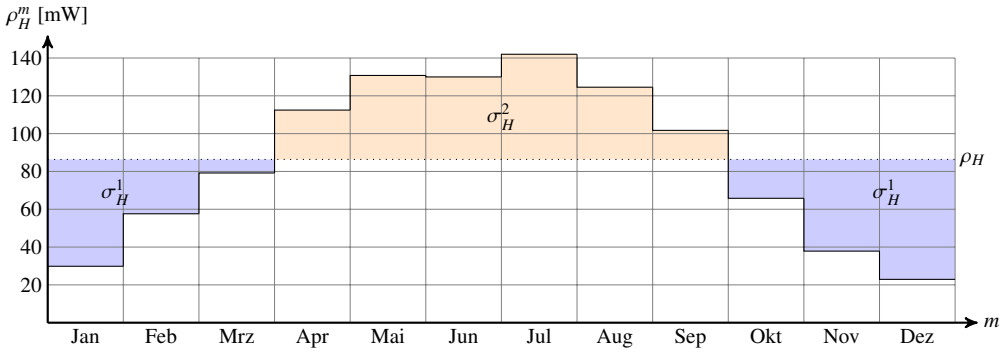


Abbildung 1.7: saisonale Variation der im Monatsmittel gewonnenen Leistung ρ_H^m

abgeschätzt werden können.

Verglichen mit der Kapazität konventioneller Akkumulatorzellen von wenigen Wattstunden ist es unrealistisch anzunehmen, dass ein Energiespeicher in der bautechnischen Größenordnung des Sensorknotens realisiert werden kann, um solche Schwankungen zu kompensieren. Wegen

$$\rho_H, \sigma_H \overset{(1.8) \text{ bis } (1.10)}{\propto} A_H \tag{1.11}$$

kann man mit der Fläche A_H des Solarmoduls die Schwankungen zwar reduzieren, die mittlere Einnahmeleistung ρ_H würde aber um den selben Faktor gesenkt werden. Da in jedem Fall $\rho_H > \rho_C^{idle}$ gelten muss, kann aber das Solarmodul und damit die zu kompensierenden Schwankungen nicht beliebig verkleinert werden.

Eine weitere Stellschraube ist die Ausrichtung (Inklination) α des Solarmoduls. Die bisher angenommenen 34° Neigung gegen die Horizontale maximieren ρ_H, weil die Sonnenstrahlen während der langen Sommertage, unter einem steilen Winkel auf das Solarmodul treffen, während in den kurzen Wintertagen der Einfallswinkel viel flacher ist (vgl. Abbildung 1.4). Verwendet man für die Abschätzung der Modellparameter ρ_H und σ_H die Daten für $\alpha \in \{0, 10, \ldots, 90\}$ aus Anhang A, so erhält man die in Abbildung 1.8a dargestellten Beziehungen zwischen den Modellparametern und dem Inklinationswinkel. Demnach werden neben dem Mittelwert auch die saisonalen Schwankungen mit steigendem Inklinationswinkel verringert. Erhöht man gleichzeitig die Solarmodulfläche um den Faktor $\frac{86\text{ mW}}{\rho_H(\alpha)}$, um die mittlere Jahresleistung konstant zu halten, wie in Abbildung 1.8b gezeigt, dann werden die Schwankungen nicht mehr so stark reduziert wie in 1.8a. Dennoch kann die benötigte Speicherkapazität bei der vertikalen Montage ($\alpha = 90°$)

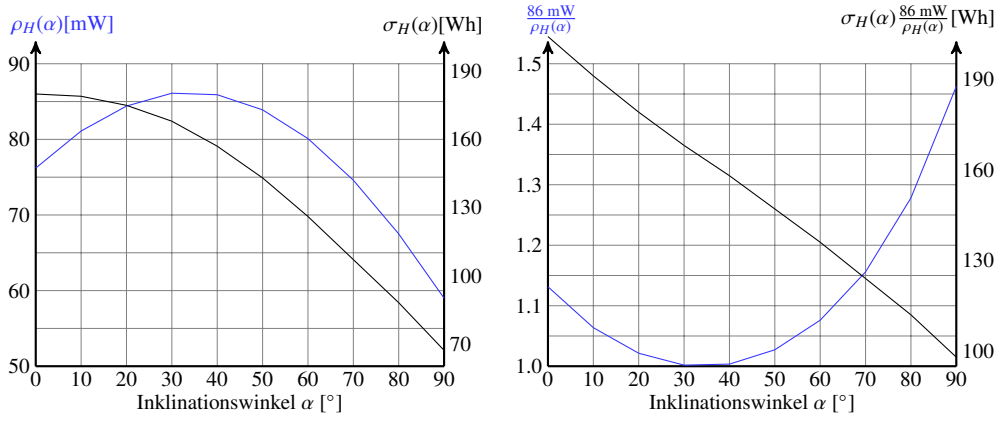


Abbildung 1.8: Einfluss der Ausrichtung des Solarmoduls auf die Modellparameter

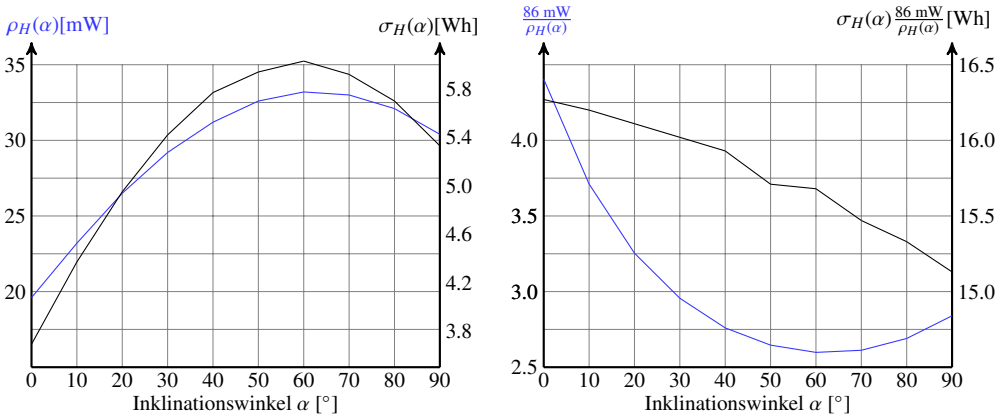


Abbildung 1.9: Einfluss der Ausrichtung des Solarmoduls auf die Modellparameter bei einer Beschränkung auf die Monate November bis Januar

eines um etwa 45 % vergrößerten Solarmoduls um etwa 40 % gesenkt werden, ohne Einbußen bei der insgesamt eingenommenen Energie hinnehmen zu müssen.

Dennoch sind die verbleibenden Schwankungen nicht durch einen kleinen, leichten und kostengünstigen Speicher kompensierbar. Um den Sensorknoten trotzdem über mehr als ein Jahr ausschließlich mit Solarzellen zu versorgen, muss das System so dimensioniert werden, dass die in den Wintermonaten im Mittel gewonnene Leistung bereits der mittleren Verbrauchsleistung entspricht. Berücksichtigt man bspw. nur die Monate November, Dezember und Januar bei der Bestimmung der Modellparamter, so erhält man die in Abbildung 1.9 gezeigte Abhängigkeit von der Modulausrichtung. Der optimale Inklinationswinkel liegt dann bei etwa 60° und das Solarmodul muss mehr als die zweieinhalbfache Fläche des Sensorknotens haben, um während des betrachteten Zeitraums im Mittel 86 mW zu liefern. Der Schwankungsparameter σ_H liegt dann noch bei etwa 16 Wh. Im Sommer könnte ein derart überdimensionertes System dann aber permanent auf maximaler Leistung laufen, was eine Verbrauchssteuerung überflüssig macht.

Richtet man sich andererseits bei der Dimensionierung des Systems nur nach den Sommermonaten, wie in Abbildung 1.10 gezeigt, dann genügt bereits ein um 20° geneigtes Solarmodul mit etwa 60 % der Fläche des Sensorknotens, um in dieser Zeit eine mittlere Leistung von 86 mW zu erzielen. Daraus würden dann zwar auch sehr geringe Schwankungswerte von etwa 5 Wh resultieren, den Winter könnte ein solches System aber nicht überstehen.

Zusammenfassend muss man festhalten, dass die ab Kapitel 4 beschriebenen Konzepte für einen energieneutralen Betrieb nur für Systeme mit einem deutlich geringeren Grundverbrauch sinnvoll anwendbar sind, solange die saisonalen Schwankungen der Energieeinnahmen dominieren.

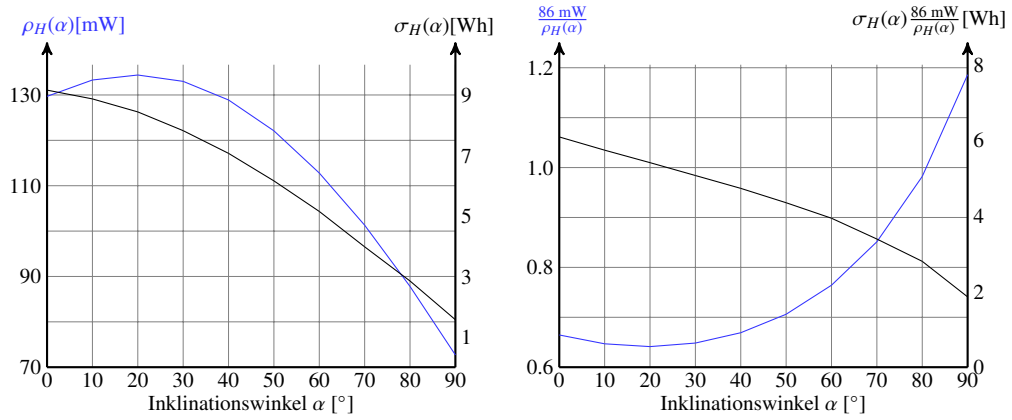


Abbildung 1.10: Einfluss der Ausrichtung des Solarmoduls auf die Modellparameter bei einer Beschränkung auf die Monate Mai bis August

Geht man hingegen von einem Tageszeitenklima aus, wie es bei Anwengungsgebieten in Äquatornähe zu finden ist, so müssen nur die täglichen Schwankungen der Energieeinnahmen gepuffert werden. Diese können dann mit $\sigma_H < 24\,\text{h} \cdot \rho_H$ abgeschätzt werden, selbst wenn der Hauptteil der Energie in einem sehr kurzen Tagesabschnitt eingenommen wird. Ein Energiespeicher von wenigen Wattstunden Kapazität ist für Schankungen in dieser Größernordnung ausreichend.

1.6 Übersicht bereits existierender Sensorplattformen mit Solarversorgung

Der Trend zur Nutzung regenerativer Energiequellen hat den Forschungsbereich der drahtlosen Sensornetze bereits vor einigen Jahren erreicht. Da sich die verschiedenen, bereits entwickelten *Harvester* Plattformen im Wesentlichen hinsichtlich ihrer Implementierung des Energiespeichers unterscheiden, soll hier je ein Vertreter der Plattform-Typen beschrieben werden. Tabelle 1.1 fasst deren Dimensionierungen zusammen. Dabei fällt auf, dass die Sensorknoten der vorgestellten Systeme im *idle*-Modus deutlich weniger Energie verbrauchen als SNoW^5. Dadurch können deren Solarmodule und Energiespeicher kleiner dimensioniert werden.

1.6.1 HelioMote

HelioMote [31] wurde 2005 an der UC Los Angeles entworfen und zeichnet sich durch einen relativ einfachen Hardwareentwurf aus (Abbildung 1.11). Eine Plattform mit ähnlichem Aufbau ist HydroSolar [42].

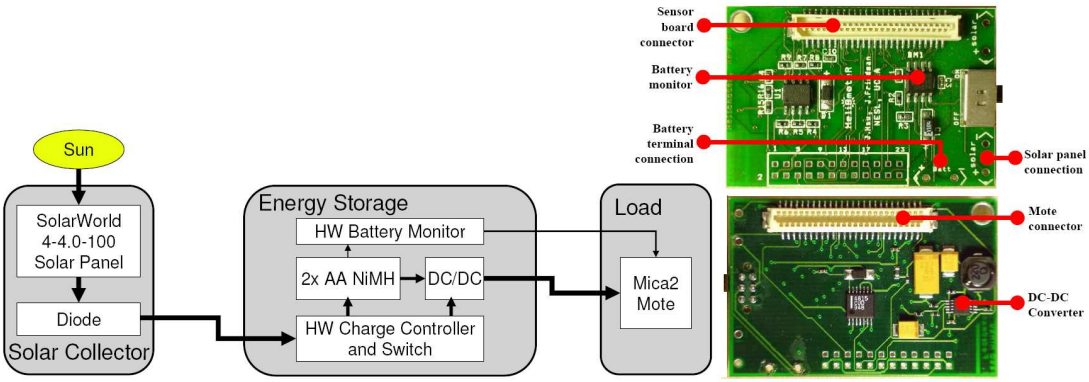


Abbildung 1.11: Modell und Implementierung von HELIOMOTE [15, 31]

Zwischen Solarpanel und Energiepuffer setzt HELIOMOTE lediglich eine Diode ein, um den Ladungsrückfluss vom Speicher zum Solarpanel bei fehlender Sonneneinstrahlung zu verhindern. Energieüberschüsse werden direkt in zwei seriell verschalteten NiMH-Akkumulatoren gespeichert, deren Füllstände über einen Batteriemonitor abgefragt werden können. Die Speicherspannung wird über einen Spannungswandler auf 3 V reguliert. Ein zweiter Überwachungs-IC erkennt automatisch Unter- (2.2 V) und Überspannungen (2.8 V) am Energiespeicher und verhindert dessen Beschädigung durch Abschalten des Spannungswandlers bzw. Abziehen des Solarstroms.

Der wesentliche Vorteil dieses Konzepts ist die Autonomie des *Harvesters*. Der MICA2 Sensorknoten hat zwar die Möglichkeit, den Ladestrom und die Zellenspannung des Energiespeichers über den Batteriemonitor zu ermitteln und seinen Energieverbrauch danach anzupassen, er muss HELIOMOTE aber nicht aktiv steuern. Dies erleichtert den Softwareentwurf sowie den Start und den Wiederanlauf mit leerem Energiespeicher.

HELIOMOTE verzichtet auf eine separate Ansteuerung des Solarmoduls, wodurch dessen Arbeitspunkt durch die Speicherspannung und den Spannungsabfall über der Eingangsdiode festgelegt wird. Mit der Begrenzung der Speicherspannung wird daher auch der Arbeitsbereich des Solarmoduls eingeschränkt, so dass die zur optimalen Leistungsausbeute notwendige Solarzellenspannung von 3,3 V (vgl. Tabelle 1.1) gar nicht erreicht wird [15].

Die permanente Belastung der NiMH-Akkus bei der Schwebeladung (Quelle, Puffer und Verbraucher parallel) wirkt sich negativ auf die maximal erreichbare Lebensdauer der Speicherzellen aus. Wegen der geringen Lade-/ Entladeeffizienz der NiMH-Technologie von etwa 66 % [42] ist die gezielte Verbrauchssteuerung zur Reduktion von Energieüberschüssen und -defiziten besonders wichtig. HELIOMOTE generiert dafür eine Einnahmeprognose aus den gemessenen Energieeinnahmen der vergangenen Tage und reguliert danach den Verbrauch des Systems. Die Verbrauchssteuerung erfolgt dabei durch das Anpassen der Aktivität, also dem Verhältnis der Lauf-

zeiten im aktiven und im energiesparenden *idle*-Modus. Dieses Grundkonzept verfolgt auch SNoW^5-RA und wird daher ab Kapitel 4 detailliert beschrieben.

1.6.2 Prometheus

PROMETHEUS [16] wurde 2005 an der UC Berkeley entworfen und 2006 in TRIO [11] integriert. Das herausstechendste Merkmal dieser Plattform ist die Aufteilung des Energiepuffers in Primär- und Sekundärspeicher (Abbildung 1.12), an dem sich auch SNoW^5-RA orientiert. Plattformen mit ähnlichem Aufbau sind TINYNODE [10] und ZEBRANET [47].

Als Primärspeicher verwendet PROMETHEUS zwei serielle Doppelschichtkondensatoren (SuperCaps), welche im Gegensatz zu elektrochemischen Akkumulatoren kaum einer Alterung unterliegen und somit nahezu unbegrenzt viele Ladezyklen ohne Kapazitätsverlust überstehen. Der Sekundärspeicher ist ein Li-Po-Akkumulator und wird softwaregesteuert über einen analogen Schalter und einen Spannungswandler mit Strombegrenzung aus dem Primärspeicher geladen. Über einen digitalen Schalter wählt der TELOS-Sensorknoten die Energiequelle, mit welcher er selbst versorgt werden soll. Bei der TRIO-Integration wurde eine *pull down* Logik für diesen Schalter hinzugefügt, die dafür sorgt, dass auch nach kompletter Entleerung des Sekundärspeichers der Primärpuffer automatisch wieder zur Versorgung des Sensorknotens genutzt wird. Ansonsten würde neu gewonnene Energie zwar in den SuperCap geladen, könnte aber nicht genutzt werden, da der abgeschaltete Sensorknoten nicht wieder auf den Primärspeicher wechseln kann.

Auch Prometheus verzichtet auf eine gezielte Ansteuerung des Solarpanels. Im Vergleich zu den beiden NiMH-Zellen von HELIOMOTE erlauben die beiden seriellen SuperCaps aber Solarzellenspannungen von bis zu 5 V, so dass der optimale Arbeitspunkt des Solarmoduls erreicht werden kann.

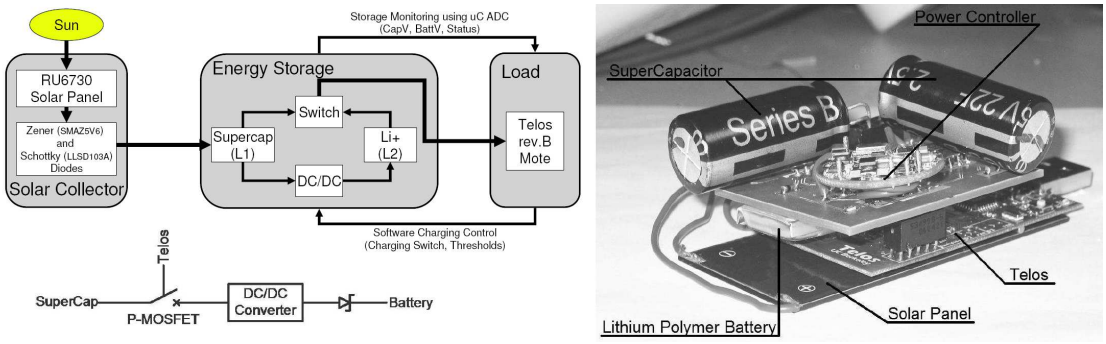


Abbildung 1.12: Modell und Implementierung von PROMETHEUS [15, 16]

Der Vorteil des hierarchischen Speichers besteht in der geringeren bzw. selteneren Belastung des Akkus und der damit einhergehenden Verlängerung der Laufzeit des Gesamtsystems. Da beim korrekten Laden von Lithium-Akkus außerdem kaum Energieverluste entstehen (vgl. Tabelle 3.1), ist dass Energiemanagement weniger auf die direkte Nutzung der gewonnenen Energie angewiesen.

Nachteilig ist hingegen das Abwälzen der gesamten Steuerlogik (Ladevorgang und Speicherauswahl) auf den Sensorknoten. Die Energieversorgung funktioniert damit nicht mehr eigenständig, es entstehen wechselseitige Abhängigkeiten und die Rechenzeit für die eigentlichen Anwendungen wird verringert.

1.6.3 Everlast

EVERLAST [37] wurde 2006 an der UC Irvine entwickelt und ist eine der wenigen Plattformen, die eine leistungsoptimierte Ansteuerung der Solarzellen implementiert und ganz auf Akkumulatoren verzichtet (Abbildung 1.13). Primäres Ziel dieses Konzeptes ist die Realisierung von Systemlaufzeiten von über 20 Jahren.

Kern der Plattform ist ein kleiner Eingangskondensator $C_{in} = 1\ \mu F$, ein Schalter S_2, eine Diode, und der eigentlichen SuperCap C_{load}. Bei geöffnetem S_2 kann das Solarpanel C_{in} sehr schnell aufladen, während die Diode verhindert, dass sich der SuperCap entlädt. Die so gewonnene Ladung wird beim Schließen des Schalters auf den SuperCap übertragen, wobei der Ladestrom von C_{in} nach C_{load} sehr viel größer ist als der Strom aus dem Solarpanel, weshalb die Spannung U_{solar} über C_{in} augenblicklich einbricht.

Die Spannung U_{solar} bestimmt den Arbeitspunkt der Solarzelle und kann durch die Schaltzeiten von S_2 gezielt beeinflusst werden. Dies übernimmt ein PFM-Regler in Form eines Kompa-

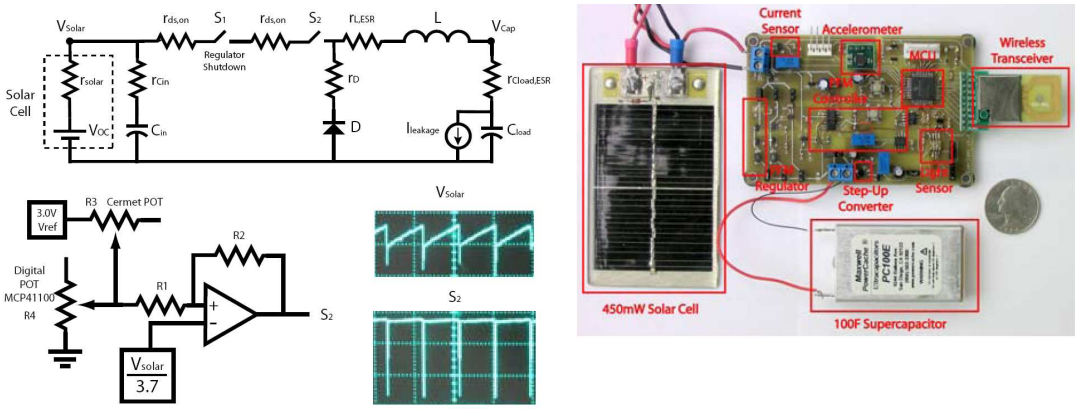


Abbildung 1.13: Modell und Implementierung von EVERLAST [37]

rators mit Hysterese, dessen Referenzspannung über ein Digitalpotentiometer (R_4) von EVERLAST vorgegeben werden kann. Der Komparator vergleicht diese Referenzspannung mit der Solarzellenspannung und steuert den Schalter S_2 so, dass U_{solar} um die Referenzspannung schwankt.

Da der optimale Arbeitspunkt U_{mpp} von der Beleuchtung und der Temperatur des Solarmoduls abhängig ist (vgl. Kapitel 2), wird dieser in regelmäßigen Abständen von EVERLAST neu bestimmt und als Referenz für den PFM-Regler vorgegeben. Aus empirischen Beobachtungen geht hervor, dass ein konstanter Zusammenhang zwischen U_{mpp} und der Spannung U_{oc} am unbelasteten Solarpanel über weite Teile des Betriebsbereichs angenommen werden kann. Als Proportionalitätsfaktor wird $F_{mpp} := \frac{U_{mpp}}{U_{oc}} = 0,7$ verwendet. Um U_{oc} zu bestimmen, trennt EVERLAST das Solarpanel und den Eingangskondensator über einen weiteren Schalter S_1 von der restlichen Schaltung und wartet, bis C_{in} maximal, also auf U_{oc} geladen ist.

Die Abhängigkeit des *Harvesters* von der MCU ist also relativ gering. Kann oder will EVERLAST die Referenzspannung des PFM-Reglers nicht mehr steuern, weicht der eingestellte Arbeitspunkt schlechtestenfalls vom aktuellen Optimum ab. In Kapitel 2 wird eine Erweiterung des EVERLAST-Konzepts um eine Abtast-/Halteschaltung untersucht, welche den Sensorknoten komplett von der Optimierung des Arbeitspunktes des Solarmoduls entbindet.

Tabelle 1.1: Dimensionierung bereits existierender Solarharvester [16, 31, 37]

Plattform	HELIOMOTE	PROMETHEUS		EVERLAST
Verbraucher	MICA2	TELOS		integrierte MCU
Betriebsspannung [V]	3	3		3
ρ_C^{active} [mW]	24	60		2,8
ρ_C^{idle} [µW]	45	15		75
Solarpanel	SW4-4.0-100	RU6730		
Größe [mm^2]	95×63	82×37		
max. P_H [mA×V]	60×3,3	40×3,4		90×5
Energiespeicher		primär	sekundär	
	2×NiMH	2×SuperCap	Li-Po	SuperCap
Spannung [V]	2,2 .. 2,8	≤ 5	3,4 .. 3,6	≤ 2,5
Kapazität	1800 mAh	11 F	500 mAh	100 F

Teil I

Hardware

KAPITEL

2 Leistungsoptimierte Ansteuerung von Solarzellen

2.1 Solarzellencharakteristiken

Für das Verständnis der Eigenschaften einer Solarzelle ist eine kurze Erläuterung ihrer Funktionsweise hilfreich. Letztlich sind Solarzellen für den Lichteinfall optimierte Halbleiterdioden und bestehen somit aus einer p- und einer n-dotierten Halbleiterschicht. Durch die Diffusion freier Ladungsträger in die jeweils andere Schicht entsteht am p-n-Übergang eine Raumladungszone, also ein elektrisches Feld zwischen dem nun positiv geladenen n-Gebiet (Elektronendefizit) und dem negativ geladenen p-Gebiet (Elektronenüberschuss). In diese Grenzschicht einfallende Photonen können Elektronen aus ihren Bindungen lösen (Photoeffekt), welche dann durch das elektrische Feld zur positiven Raumladung beschleunigt werden. Die zurückbleibenden Defektelektronen werden entsprechend zur negativen Raumladung beschleunigt. Erreichen beide Ladungen die an der jeweiligen Halbleiterschicht angeschlossene Elektrode und besteht außerdem eine leitende Verbindung zwischen beiden Elektroden, so resultiert ein Stromfluss in Sperrrichtung der Diode. Dieser Photostrom steigt mit der Anzahl der Photonen, die pro Zeiteinheit auf die Grenzschicht treffen und wird daher von der Zellenfläche und der Lichtintensität beeinflusst.

An einem angeschlossenem Lastwiderstand verursacht der Photostrom einen Spannungsabfall, also eine positive Zellenspannung zwischen p- und n-Gebiet. Das resultierende externe elektrische Feld ist dem internen Feld der Diode entgegen gerichtet. Bei kleinen Zellenspannungen wirkt sich dies kaum auf den Photostrom aus, solange immer noch fast alle durch den Photoneneinfall erzeugten freien Ladungsträger die Elektroden erreichen. Mit steigender Zellenspannung wird der Photostrom verringert und beim Erreichen der Durchbruchspannung der Diode wird

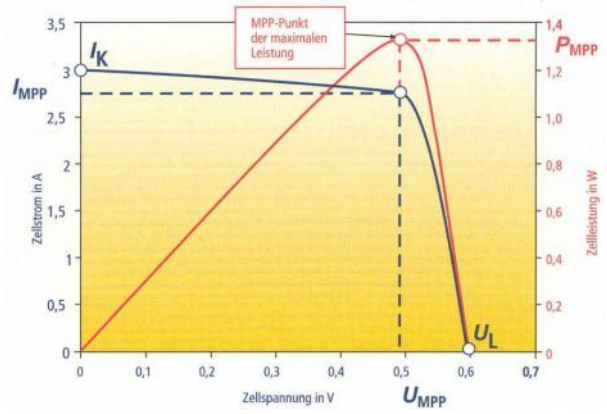
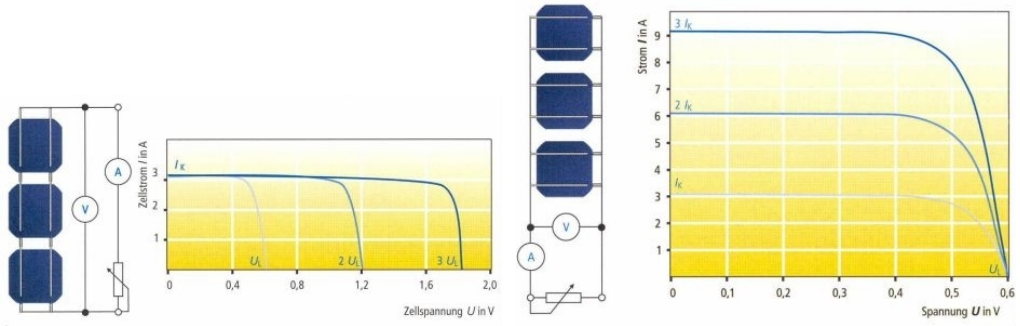


Abbildung 2.1: Strom- und Leistungskennlinie einer Solarzelle [28]

die Raumladungszone komplett aufgehoben, so dass kein Photostrom mehr fließen kann. Diese Spannung wird daher auch als Leerlaufspannung U_L bzw. U_{oc} (*open circuit*) der Solarzelle bezeichnet. Der maximale Photostrom wird dagegen bei verschwindender Last bzw. Zellenspannung erreicht und wird daher als Kurzschlussstrom I_K bzw. I_{sc} (*short circuit*) bezeichnet. Der Arbeitspunkt, bei dem die Solarzelle die maximale Leistung erzeugt, heißt MPP (*maximum power point*). Abbildung 2.1 zeigt beispielhaft die Kennlinie einer Solarzelle.

Im Gegensatz zu Batterien und Akkumulatoren sind Solarzellen also keine Spannungsquellen, sondern spannungsbegrenzte Stromquellen [31]. Die Leerlaufspannung von Siliziumzellen liegt bei etwa 0,6 V. Um höhere Spannungen zu erzielen, müssen mehrere Zellen seriell zu einem Solarpanel bzw. -modul verschaltet werden. Für einen höheren Photostrom können entsprechend mehrere parallele Zellen eingesetzt werden. Diese beiden in Abbildung 2.2 gezeigten Varianten



(a) serielle Zellenverschaltung **(b)** parallele Zellenverschaltung

Abbildung 2.2: Verschaltung von Solarzellen zu Modulen [28]

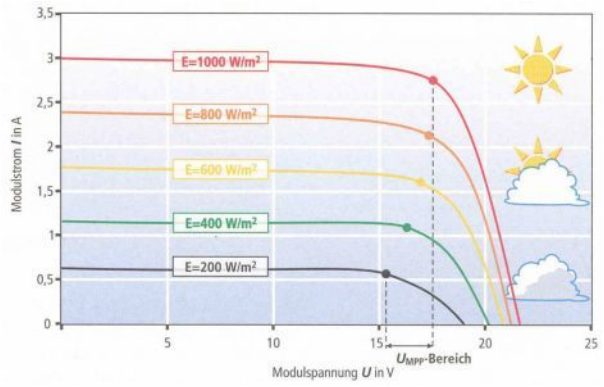


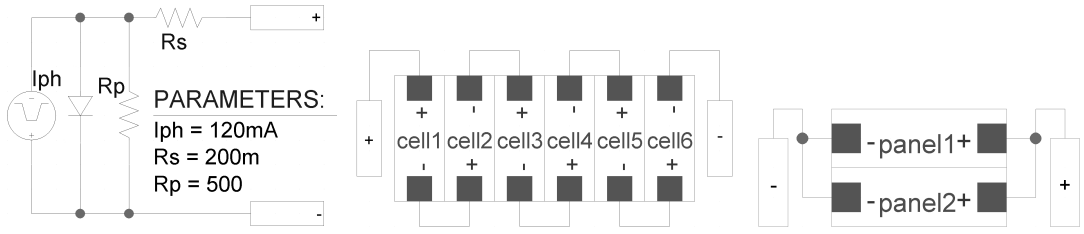
Abbildung 2.3: Abhängigkeit des MPP von der Beleuchtung des Solarmoduls [28]

der Modulbildung können auch kombiniert werden, um sowohl den Solarstrom als auch die erreichbare Solarspannung zu erhöhen.

Mit dem Photostrom ist auch der MPP von der Beleuchtung des Solarmoduls abhängig, wie Abbildung 2.3 zeigt. Um die Leistungsausbeute zu optimieren, muss der Arbeitspunkt der Solarzelle also regelmäßig den Beleuchtungsverhältnissen angepasst werden. Wie im Abschnitt 1.6 beschrieben, wird ein solches, als MPP-Tracker bezeichnetes Verfahren beispielsweise von Everlast [37] implementiert. In den nächsten Abschnitten wird der dabei verwendete PFM-Regler analysiert und um eine Komponente erweitert, die ein aktives Eingreifen des Sensorknotens in den Prozess der Arbeitspunkteinstellung überflüssig macht.

2.2 Simulationsmodell der Energiequelle

Um das MPPT-Verfahren von Everlast [37] mit PSpice nachvollziehen zu können, wird zunächst das Modell einer Solarzelle benötigt. Wie im letzten Abschnitt beschrieben, verhält sich eine beleuchtete Solarzelle wie eine Diode, welche ohne angelegte Spannung einen Stromfluss in



(a) Zelle **(b)** Panel **(c)** Modul

Abbildung 2.4: PSpice-Modell der Energiequelle

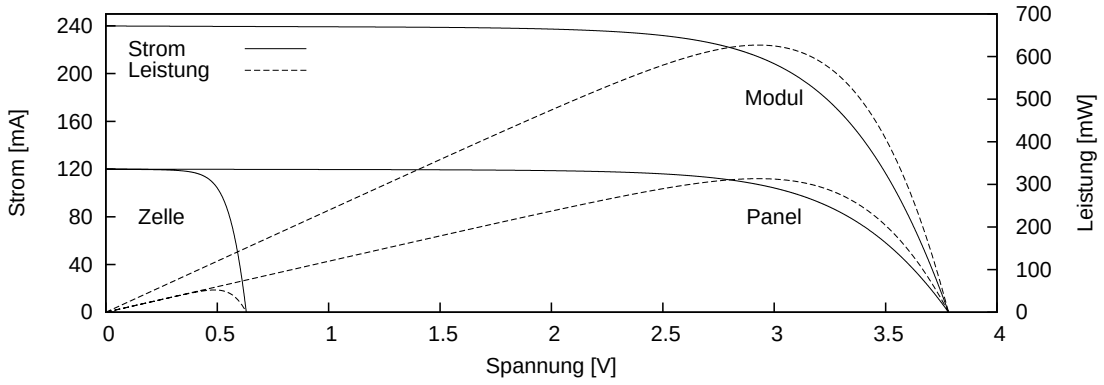
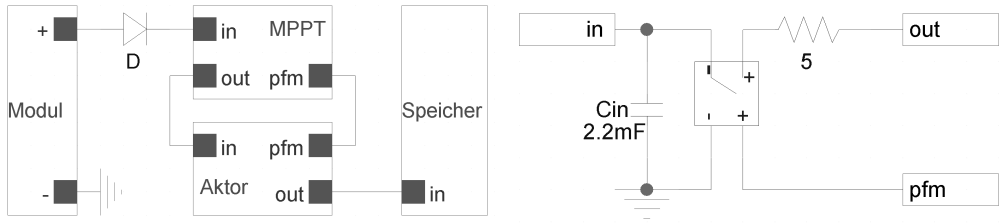


Abbildung 2.5: Ausgabecharakteristiken der Energiequelle

Sperrrichtung generiert. Nach [33] kann das Ein-Dioden-Modell aus Abbildung 2.4a zur Simulation dieses Verhaltens verwendet werden. Neben der Diode und der Photostromquelle werden ein serieller und ein paralleler Widerstand eingesetzt, um Abweichungen vom idealen Diodenverhalten zu modellieren. Die Dimensionierung des Modells orientiert sich an der monokristallinen XOD17-34B [14] Siliziumzelle mit 3,4 cm^2 aktiver Fläche und einem Wirkungsgrad von 16.6 %. Unter Standardtestbedingungen (100 mW/cm^2 Beleuchtungsstärke) erreichen diese Zellen einen Kurzschlussstrom von 120 mA und eine Leerlaufspannung von 0,63 V. Die maximale Leistung von 57 mW wird bei etwa 0,5 V ausgegeben. Abbildung 2.5 zeigt die Simulation dieser Kenndaten mit dem Ein-Dioden-Modell.

Da die niedrige Zellenspannungen in keinem Fall ausreicht, um den Sensorknoten zu versorgen, müssen mehrere Zellen seriell zu einem Solarpanel verschaltet werden. Für den im Kapitel 3 beschriebenen Energiepuffer werden 2,5 V benötigt, um den Primärspeicher vollständig zu laden. Außerdem fallen über der Schutzdiode zwischen Solarmodul und Energiespeicher noch einige 100 mV ab, so dass wenigstens sechs serielle Solarzellen zu einem Panel kombiniert werden müssen (Abbildung 2.4b).

Die etwa 350 mW Spitzenleistung, welche ein solches Panel unter den Standardtestbedingungen generiert, werden beim realen Einsatz nicht erreicht. Nach Tabelle A.5 kann man selbst bei optimaler Ausrichtung in den Sommermittagsstunden höchstens 600 W/m^2, also 60 % der Standardleistung erwarten. Sollte dies für eine konkrete Anwendung nicht ausreichen, so müssen mehrere Panels parallel zu einem Modul verschaltet werden (Abbildung 2.4c), um den generierten Strom und damit auch die Leistung zu vergrößern (Abbildung 2.5). Die namentliche Unterscheidung zwischen Panel und Modul ist dabei lediglich dem hierarchischen Aufbau der PSpice Simulation geschuldet. Im Folgenden wird die Energiequelle generell als Solarmodul bezeichnet, unabhängig von der Anzahl seriell und parallel verschalteter Zellen.



(a) Toplevel der Simulation **(b)** Aktor

Abbildung 2.6: Konzept der PFM-Regulation der Solarmodulspannung

2.3 Regulierung der Eingangsspannung

Um den Arbeitspunkt der Solarzellen optimieren zu können, muss ein Regler zwischen das Solarmodul und den Energiespeicher bzw. den Verbraucher gesetzt werden. Abbildung 2.6a zeigt das Konzept der zweistufigen Realisierung eines solchen Systems. Nach der Diode D, welche das Solarmodul vor Stromrückfluss aus dem Energiespeicher schützt, wird die um den Spannungsabfall über D verminderte Modulspannung an den *Maximium Power Point Tracker* angelegt, dessen Realisierung im nächsten Abschnitt beschrieben ist. Er generiert ein pfm-Signal, mit dem der Schalter im PFM-Aktor (Abbildung 2.6b) gesteuert wird. Solange dieser Schalter geöffnet ist ($pfm = GND$), wird der Eingangskondensator C_{in} vom Solarstrom aufgeladen, welchen der MPPT von *in* nach *out* durchschleift. Dabei steigt die Spannung über dem Eingangskondensator. Wird der PFM-Schalter geschlossen ($pfm = U_{cc}$), so entlädt sich C_{in} über den strombegrenzenten 5 Ω Widerstand in den eigentlichen Energiespeicher. Dabei fällt die Spannung über dem Eingangskondensator wieder ab. Da die Solarmodulspannung von der Spannung über C_{in} abhängt, kann der Arbeitspunkt des Solarmoduls durch das pfm-Signal vom MPPT beeinflusst werden.

Abbildung 2.7 zeigt die Entwicklung der Solarspannung, des Solarstroms und der gewonnenen Leistung im regulierten und im unregulierten Fall. Bleibt der PFM-Schalter permanent geöffnet, so begrenzt die ansteigende Modulspannung den Solarstrom. Beim Erreichen der Leerlaufspannung versiegt der Solarstrom vollständig, so dass C_{in} nicht weiter aufgeladen werden kann. Mit dem Solarstrom geht dann auch die gewonnene Leistung auf Null zurück. Wird der Schalter dagegen mit einer kleinen Verzögerung geöffnet, sobald die Modulspannung U_{mpp} übersteigt und wieder geschlossen, sobald sie unter U_{mpp} sinkt, so schwanken Solarspannung und Solarstrom um ihre MPP-Werte, so dass die gewonnene Leistung maximiert wird.

Das verzögerte Öffnen und Schließen des PFM-Schalters kann durch einen Komparator mit Hysterese, wie etwa den MAX965 [25] realisiert werden. Die Hysteresebreite kann bei diesem IC durch die externe Beschaltung (Abbildung 2.8) auf maximal ±50 mV konfiguriert werden. Legt man die aktuelle Modulspannung auf den positiven Eingang des Komparators und die angestreb-

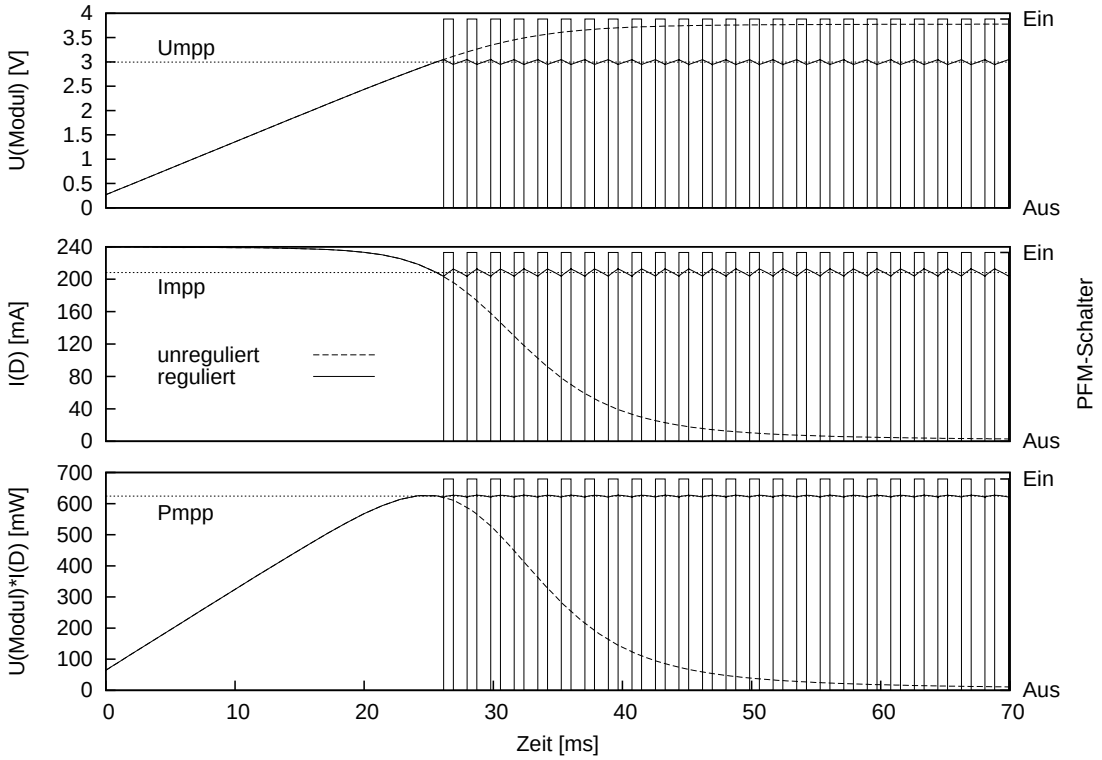


Abbildung 2.7: Regulation der Solarmodulspannung am MPP

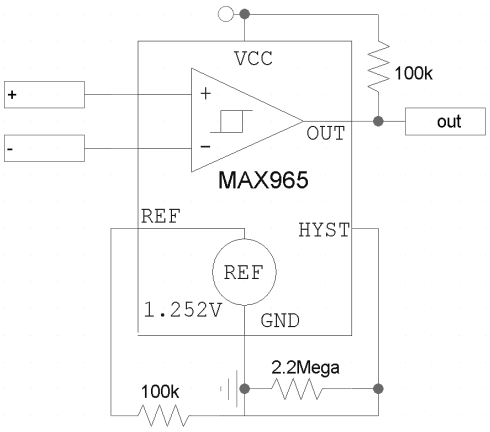


Abbildung 2.8: Komparator zur Generierung des pfm-Signals

Abbildung 2.9: Hysterese am Komparator

te Zielspannung U_{mpp} auf den invertierten Eingang, so entspricht die Ausgabe des Komparators nach Abbildung 2.9 gerade dem benötigten pfm-Signal.

2.4 Realisierung eines MPPT

Für den MPPT bleibt nun noch die Aufgabe, den optimalen Arbeitspunkt U_{mpp} für die aktuellen Beleuchtungsverhältnisse zu bestimmen. Der Kurzschlussstrom einer Solarzelle ist linear von der Beleuchtungsstärke, also der Anzahl der einfallenden Photonen abhängig. Im Solarzellenmodell (Abbildung 2.4a) wird der Kurzschlussstrom durch die Stromquelle I_{ph} bestimmt. Simuliert man nun das vollständige Aufladen des Eingangskondensators, also den unregulierten Fall von Abbildung 2.7 für verschiedene Kurzschlussströme, so kann man aus den gewonnenen Daten die in Abbildung 2.10 dargestellte Abhängigkeit von U_{oc} und U_{mpp} vom Kurzschlussstrom bzw. von der Beleuchtungsstärke ermitteln. Das obere Diagramm zeigt dabei die Spannungen am Solarmodul. Diese können aber vom MPPT wegen des Spannungsabfalls an der Schutzdiode D (vgl. Abbildung 2.6a) nicht beobachtet werden. Das untere Diagramm von Abbildung 2.10 zeigt deswegen die Eingangsspannung des MPPT zum Zeitpunkt der maximalen und optimalen Modulspannung.

Der vom Solarzellenhersteller angegebene Kurzschlussstrom von 120 mA wird bei einer Beleuchtungsstärke von 100 mW/cm^2 = 1000 W/m^2 [14] erreicht. Wie bereits erwähnt werden aber unter realen Bedingungen höchstens 60 % davon erreicht. Bei einem aus zwei parallelen Panels bestehendem Solarmodul sind demnach Kurzschlussströme bis etwa 140 mA für den MPPT von besonderem Interesse.

Nach Abbildung 2.10 ist U_{mpp} über weite Teile des Arbeitsbereichs linear von der Beleuchtungsstärke abhängig und bricht erst bei schwacher Beleuchtung stärker ein. Ignoriert der MPPT den Bereich kleinster Kurzschlussströme, so sind die dabei entstehenden Leistungsverluste mit

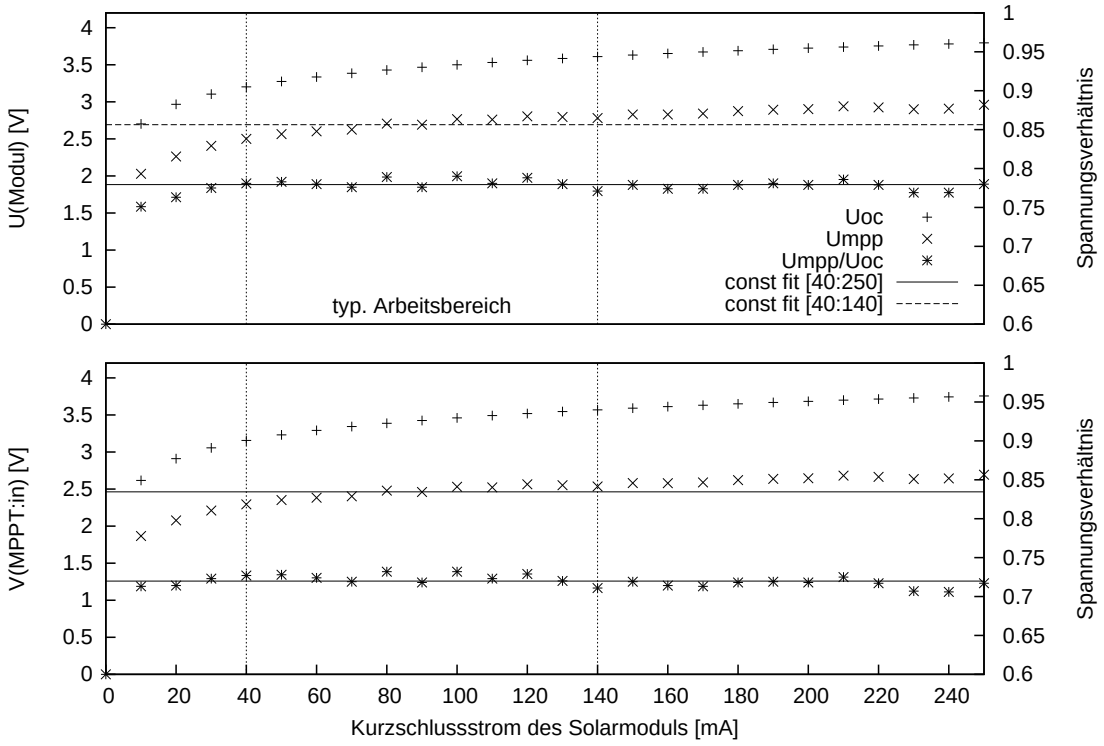


Abbildung 2.10: Abhängigkeit der maximalen (U_{oc}) und optimalen (U_{mpp}) Spannungen am Solarmodul und am MPPT

dem Betrag der maximal erzielbaren Leistung limitiert und fallen daher kaum ins Gewicht. Beschränkt man den betrachteten Betriebsbereich daher auf Kurzschlussströme von 40 bis 140 mA, so liegt U_{mpp} am Eingang des MPPT zwischen 2,3 und 2,6 V. Die Referenzspannung für den PFM-Regler kann daher mit 2,45 V in nullter Ordnung genähert werden, wenn man auf eine Adaption des Arbeitspunktes zur Laufzeit ganz verzichten will. Zu beachten ist dabei die Abhängigkeit dieser Simulationsdaten vom Spannungsabfall über der Schutzdiode.

Außerhalb des betrachteten Arbeitsbereichs steigen die Abweichungen des optimalen Arbeitspunktes von der konstanten Näherung an. Everlast verwendet das Verhältnis $F_{mpp} := \frac{U_{mpp}}{U_{oc}}$, um U_{mpp} aus dem relativ einfach ermittelbaren U_{oc} zu bestimmen. Dieses Verhältnis lässt sich nach Abbildung 2.10 in einem größeren Arbeitsbereich als konstant annehmen, als dies für die Spannungen selbst der Fall ist.

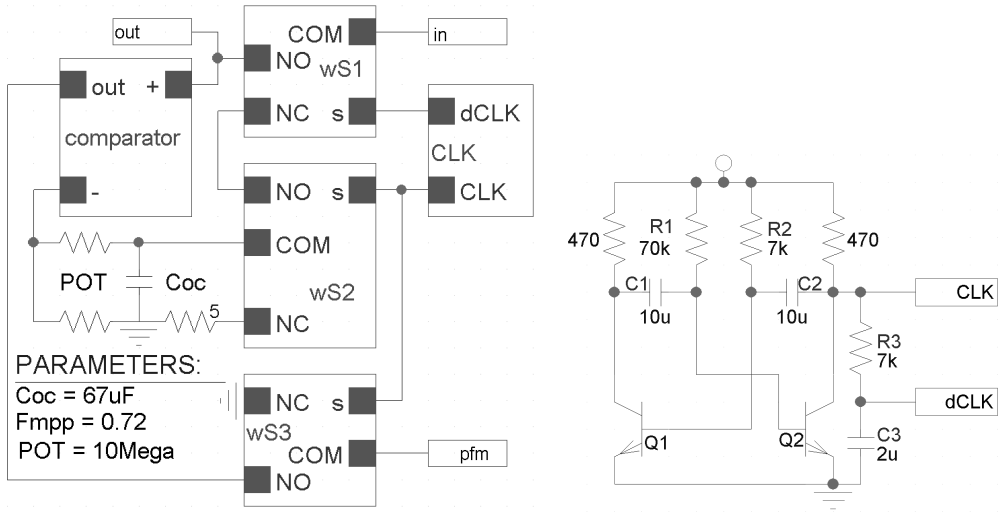
Um U_{oc} zu bestimmen, muss die PFM-Regulierung unterbrochen werden, da diese das Ansteigen der Modulspannung verhindert. Statt nun aber einen Analog-Digital-Wandler des Sensorknotens zum Messen von U_{oc} sowie ein digitales Potentiometer zum Einstellen von U_{mpp} =

$F_{mpp} \cdot U_{oc}$ zu verwenden, kann der MPPT durch Einsatz einer Abtast- und Halteschaltung nach Abbildung 2.11a vollständig hardwaregesteuert realisiert werden.

Die Grundidee dabei ist, einen Kondensator C_{oc} regelmäßig auf U_{oc} aufzuladen und die Multiplikation mit F_{mpp} durch eine Potentiometerschaltung zu realisieren. Gesteuert wird die Auffrischung von U_{oc} durch einen asymmetrischen Taktgeber und drei SPDT (*single pole double throw*) Wechselschalter wS1-3. Ein solcher Schalter verbindet COM mit NC (*normally connected*), wenn seine Steuerspannung s auf *GND* liegt. Bei einer Ansteuerung mit U_{cc} wird COM hingegen mit NO (*normally open*) verbunden.

Der Taktgeber wird durch die astabile Kippschaltung in Abbildung 2.11b realisiert. Ist bei dieser Schaltung Q_1 geschlossen und C_1 entladen, so wird die Basis von Q_2 über R_1 auf U_{cc} gezogen und somit Q_2 geöffnet. Der CLK-Ausgang liegt damit auf *GND* und C_2 wird über R_2 geladen, bis die Spannung ausreicht, um Q_1 zu öffnen. Dabei wird dann die Basis von Q_2 über den noch ungeladenen C_1 und die Kollektor-Emitter-Strecke von Q_1 auf *GND* gezogen, so dass Q_2 schließt und damit CLK nicht mehr auf *GND* gezogen wird. Außerdem wird C_2 über die Basis von Q_1 entladen. Damit wurde die Ausgangssituation umgekehrt und es wird nun C_1 über R_1 geladen, bis dessen Spannung ausreicht, um Q_2 wieder zu öffnen. Über die beiden Zeitkonstanten R_1C_1 und R_2C_2 kann die Dauer der *high*- und *low*-Phasen des CLK-Ausgangs konfiguriert werden.

Zusammen mit dem über ein weiteres RC-Glied verzögerten Ausgang dCLK versetzt der Taktgeber den MPPT in einen von vier möglichen Zuständen, die in Abbildung 2.12 gezeigt sind.



(a) MPPT **(b)** CLK

Abbildung 2.11: MPPT mit einer Abtast-/Halteschaltung zur Bestimmung von U_{oc}

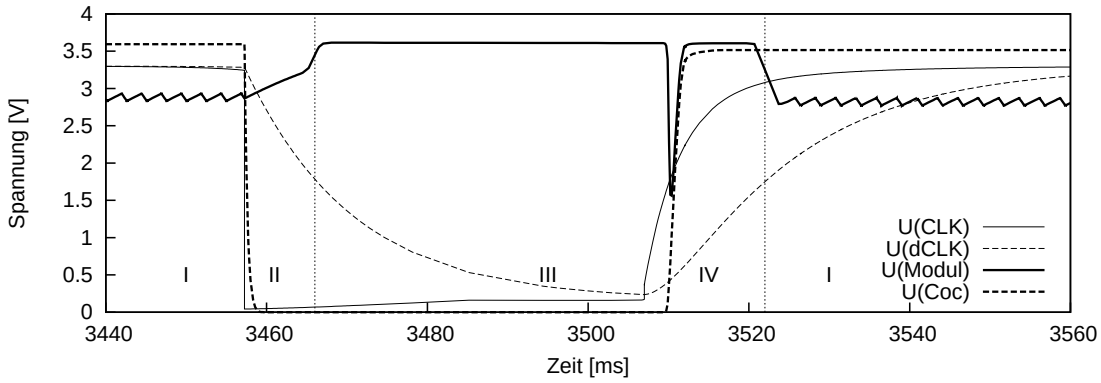


Abbildung 2.12: Auffrischung von C_{oc} auf die aktuelle Leerlaufspannung des Solarmoduls

I: CLK *high* und dCLK *high*
: wS1 verbindet *in* mit *out* und reicht damit den Solarstrom zum Laden des Eingangskondensators C_{in} an den PFM-Aktor weiter (vgl. Abbildung 2.6). wS2 verhindert das Auf-/Entladen von C_{oc} und wS3 gibt das Steuersignal des Komparators an den PFM-Aktor weiter. Dies ist also die eigentliche Arbeitsphase des PFM-Reglers. Sie dauert daher wesentlich länger als die anderen Phasen.

II: CLK *low* und dCLK *high*
: Diese Phase leitet die Auffrischung von C_{oc} ein, indem sie diesen über wS2 und den strombegrenzenden Widerstand vollständig entlädt, denn die Leerlaufspannung des Solarmoduls könnte seit der letzten Auffrischung gefallen sein. Ein direktes Verbinden des Solarmoduls mit C_{oc} würden dessen Restladung dann nicht vermindern. Da die Referenzspannung am Komparator nun auf Null fällt, würde der PFM-Regler auch den Eingangskondensator vollständig entladen. Das Wiederaufladen von C_{in} nach der Auffrischung von C_{oc} würde aber den Start der nächsten I-Phase unnötig verzögern. Über den Wechselschalter wS3 wird das pfm-Steuersignal deswegen auf GND gezogen und damit das Entladen von C_{in} verhindert.

III: CLK *low* und dCLK *low*
: C_{oc} und das pfm-Signal werden über wS2 und wS3 weiter auf GND gehalten, während wS1 *in* von *out* trennt, so dass C_{in} nicht weiter vom Solarmodul aufgeladen werden kann. Das Solarmodul ist nun vom Rest der Schaltung isoliert, so dass die Modulspannung auf U_{oc} steigt.

IV: CLK *high* und dCLK *low*
: C_{oc} wird über wS1 und wS2 mit *in*, also mit dem Solarmodul verbunden und somit bis auf den Spannungsabfall über der Schutzdiode auf U_{oc} aufgeladen.

Abbildung 2.13 zeigt das Verhalten des MPPT, wobei die veränderliche Beleuchtungsstärke wieder durch eine Variation des Kurzschlussstroms simuliert wird. Man erkennt, wie der MPPT den steigenden und fallenden Beleuchtungsstärken folgt. Für mittlere Kurzschlussströme zwischen

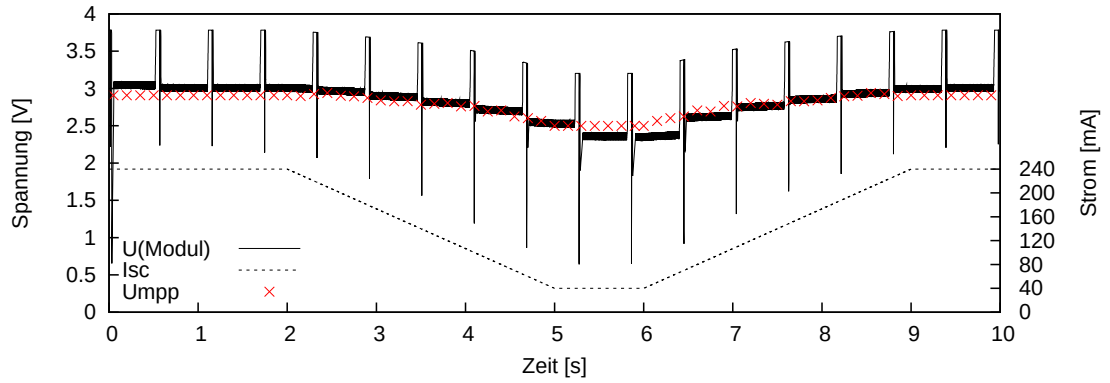


Abbildung 2.13: Anpassung des Arbeitspunktes an die Beleuchtungsstärke

80 und 160 mA wird der optimale Arbeitspunkt durch das Hystereseintervall des PFM-Reglers erfasst. Die Abweichungen der eingeregelten Modulspannung vom tatsächlichen MPP wird erst an den Rändern des Betriebsbereichs größer als die Hystereseweite, da hier die konstante Näherung von F_{mpp} nicht mehr ausreichend genau ist.

Bei der Dimensionierung der MPPT-Komponenten ist darauf zu achten, dass C_{oc} in Phase IV auch bei einem kleinen Photostrom auf etwa 3 V geladen werden muss. Reicht die Verzögerung von dCLK dafür nicht aus, so kann die Kapazität von C_{oc} verkleinert werden, damit dieser schneller geladen wird. In jedem Fall muss dabei das Produkt von C_{oc} mit den Potentiometerwiderständen wesentlich größer sein als die Dauer der I-Phase, damit U_{oc} bis zur nächsten Auffrischung annähernd konstant gehalten wird. Die in obiger Simulation verwendete Länge der I-Phase von unter einer Sekunde ist dabei lediglich der Verkürzung der Simulationsdauer geschuldet. Für reale Anwendungen sollte diese Arbeitsphase wesentlich länger sein, damit der relative Anteil der für den tatsächlichen Energiegewinn nicht nutzbaren Phasen II bis IV minimiert wird.

Ob der Einsatz eines solchen MPPT den Hardwareaufwand rechtfertigt, der Leistungszugewinn also den Energiebedarf der Abtast-/Halteschaltung übersteigt, hängt letztlich von der Dimensionierung des Systems ab. Für die meisten Anwendungen dürfte die Vorgabe einer festen Referenzspannung am Komparator des PFM-Reglers einen sinnvollen Kompromiss zwischen Aufwand und Nutzen darstellen. Erst wenn die erwarteten Schwankungen der Beleuchtungsverhältnisse und die maximale Modulleistung ausreichend groß sind, kann die zusätzliche MPPT-Hardware zur Steigerung des Engergiegewinns eingesetzt werden.

KAPITEL

3 Realisierung des Energiespeichers

Als Energiepuffer für das autarke System kommen grundsätzlich Doppelschichtkondensatoren (SuperCaps) und Akkumulatoren verschiedener chemischer Zusammensetzung in Frage, deren wichtigste Eigenschaften in Tabelle 3.1 zusammengefasst sind.

Die Wahl des Speichertyps muss sich an der Charakteristik der zu puffernden Energiemenge orientieren. Aus Abbildung 1.6 wird ersichtlich, dass am Tag überschüssige Energie gespeichert werden muss, um sie in der Nacht wieder entnehmen zu können. Daraus resultiert mindestens ein Lade-/Entladezyklus pro Tag. Kommt es zusätzlich zu lokalen Verschattungen, welche die direkte Bestrahlung des Solarmoduls mehrmals täglich verhindern, können noch wesentlich mehr, wenn auch flachere Ladezyklen anfallen. Die Angabe der Ladezyklen der verschiedenen Speichertypen in Tabelle 3.1 bezeichnet die Anzahl der vollständigen Lade- und Entladevorgänge, nach denen die Kapazität der Speicher auf etwa 80 % des Ausgangswertes gefallen ist. Zwar steigt diese Anzahl mit fallender Zyklentiefe, dennoch sind die elektrochemischen Energiespeicher kaum für derart häufige Energietransporte geeignet, wie sie für SNoW^5-RA nötig sind. Außerdem benötigen die Ladeverfahren der Akkumulatoren stabilisierte Strom- bzw. Spannungsquellen (CC- bzw. CV-Verfahren), um die ablaufenden chemischen Reaktionen zu steuern und

Tabelle 3.1: Eigenschaften verschiedener Energiespeichertypen [8, 26, 42]

	Blei	NiCd	NiMH	Li-Ion	SuperCap
nominelle Zellenspannung [V]	2.0	1.2	1.2	3.6	<2,5
Energiedichte [Wh/l]	90	100	240	400	6
Ladezyklen	250-500	300-700	300-600	1000	500000+
Lade-/Entladeeffizienz [%]	70-92	70-90	66	99.9	97-98
Selbstentladung [% pro Monat]	4-8	15-20	15-25	2	6 % pro Tag

somit Beschädigungen der Elektroden- und Elektrolytmaterialien zu verhindern [12]. Das direkte Aufladen der elektrochemischen Speicher aus der instabilen Solarstromquelle ist daher nicht ratsam, wenn man auf lange Akku-Laufzeiten angewiesen ist.

Kondensatoren zeichnen sich hingegen durch eine hohe Zyklenfestigkeit und Ladeeffizienz aus. Sie sind daher besonders geeignet, um die kurzfristigen Unterschiede zwischen dem Energieeinnahme- und dem Verbrauchsprofil zu glätten. Ihre geringe Energiedichte und die hohe Selbstentladungsrate verhindern allerdings die langfristige Speicherung großer Energiemengen, welche zur Kompensation der saisonalen Schwankungen der Energieeinnahmen notwendig ist (vgl. Abschnitt 1.5).

Die Realisierung großer, persistenter Speicher mit möglichst schnellem bzw. flexiblem Zugriff auf kleine Speichereinheiten erfolgt bei Datenspeichern durch den Einsatz von *Cache*-Systemen, also der Kombination aus einem langsamen Speicher mit großer Kapazität, einem schnellen Speicher mit kleiner Kapazität sowie einer geeigneten Strategie zum Datenaustausch zwischen beiden Speichern. Überträgt man dieses Konzept auf die Speicherung von Energie, so muss neben dem Kondensator als Primärspeicher ein elektrochemischer Sekundärspeicher eingesetzt werden, wie dies auch beim PROMETHEUS-Projekt [16] der Fall ist. Die dabei notwendige *Cache*-Verwaltung ist durch die Gleichwertigkeit aller Speichereinheiten (Energieportionen) im Vergleich zur Verwaltung der adressierten Einheiten hierarchischer Datenspeicher stark vereinfacht.

Energieüberschüsse werden nur im Primärspeicher abgelegt. Ist dieser voll, so wird eine bestimmte Teilmenge der enthaltenen Energie in den Sekundärspeicher transportiert. Die bei Datenspeichern an dieser Stelle verwendeten Strategien zur Auswahl der zu verdrängenden Einheiten wird beim Energiespeicher auf die Frage reduziert, wie viele Einheiten verdrängt, also wie weit der Primärspeicher entladen werden soll (vgl. Abschnitt 5.3). Statt der regenerativen Energiequelle dient nun also der wesentlich stabilere Primärspeicher als Energiequelle für das Ladeverfahren des Akkumulators.

Der Versuch, Energie aus einem entleerten Primärspeicher zu entnehmen, entspricht einem *Cache-Miss*. Bei Datenspeichern wird dies durch den Transport der angefragten Einheit vom Sekundärspeicher in den Primärspeicher behandelt, um künftige Anfragen auf dieselbe Einheit schneller, also ohne Sekundärspeicherzugriff beantworten zu können. Bei den hierarchischen Energiespeichern würde das dem Aufladen des Primärspeichers aus dem Sekundärspeicher entsprechen. Wenn aber anschließend wieder Überschüsse der Energiequelle im Primärspeicher eingelagert werden sollen, so muss dieser erst wieder (teilweise) in den Sekundärspeicher entladen werden. Da mit jedem Umladevorgang aber auch Energieverluste verbunden sind, ist es sinnvoller, bei entleertem Primärspeicher das System direkt aus dem Sekundärspeicher zu versorgen, und somit die zwischen den Speichern transportierte Energiemenge zu reduzieren. Dafür wird ein SPDT-Wechselschalter benötigt, mit dem jeweils ein Speicher zur Versorgung des Systems ausgewählt werden kann.

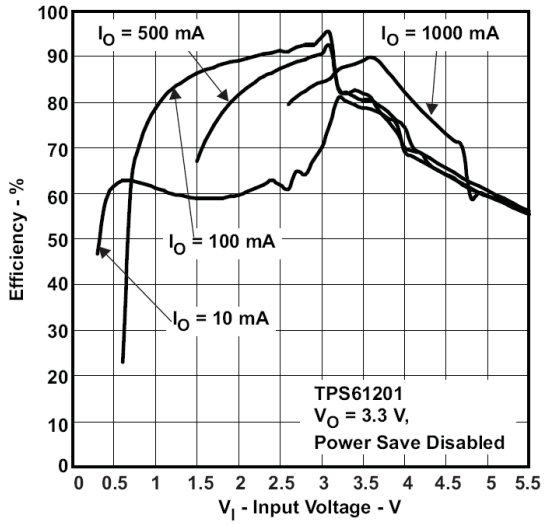


Abbildung 3.1: Abhängigkeit der Effizienz des TPS61201 Spannungswandlers von der Eingangsspannung für verschiedene Lastströme [43]

Da der SNoW^5-Sensorknoten mit einer konstanten Betriebsspannung von 3,3 V versorgt werden muss, die Zellenspannungen der Energiespeicher aber von der Menge der enthaltenen Energie abhängig ist, wird außerdem ein Gleichspannungswandler benötigt. Eine wichtige Eigenschaft solcher Spannungswandler ist ihre Mindesteingangsspannung, denn sie beschränkt den Spannungsbereich des Kondensators nach unten und limitiert somit die effektiv nutzbare Kapazität des Primärspeichers. Die meisten dieser Spannungswandler sind für den Betrieb mit Batterien oder Akkumulatoren ausgelegt und benötigen wenigstens 1 V Eingangsspannung, womit wegen $E_{cap} \propto U_{cap}^2$ nur etwa 84 % der Primärkapazität nutzbar wären. Mit der Verbreitung von Brennstoff- und Solarzellen, deren Zellenspannungen bei weit unter einem Volt liegen, werden auch die Spannungswandler für immer kleinere Eingangsspannungen konzipiert. So kann der TPS61201 [43] von Texas Instruments die 3,3 V Ausgangsspannung bereits aus 0,3 V Eingangsspannung erzeugen. Nach Abbildung 3.1 können in diesem untersten Spannungsbereich aber nur sehr geringe Lasten versorgt werden. Für den aktiven Modus des SNoW^5-Sensorknotens würden diese 10 mA zwar ausreichen, um aber auch den Betrieb des Funkchips oder anderer aktiver Sensoren zu ermöglichen, sollte die Kondensatorspannung nicht unter $U_{cap}^{min} := 0{,}7$ V fallen.

Gibt man außerdem $U_{cap}^{max} := 2{,}4$ V vor, um den Doppelschichtkondensator sicher vor einer Überladung zu schützen (eine parallele Zenerdiode mit entsprechender Durchbruchspannung garantiert die Einhaltung dieser Obergrenze), so kann im Primärspeicher maximal

$$E_{prim} = \frac{1}{2} C_{cap} \left((U_{cap}^{max})^2 - (U_{cap}^{min})^2 \right) \tag{3.1}$$

gepuffert werden. Für eine Abschätzung der notwendigen Kondensatorkapazität muss außerdem der Wirkungsgrad des Spannungswandlers beachtet werden, der bei den niedrigen Lastströmen nach Abbildung 3.1 nur etwa 60 % beträgt. Will man also einen 20 mW Verbraucher für 8 h aus dem Primärspeicher versorgen, so muss E_{prim} wenigstens 267 mWh und C_{cap} daher mindestens 364 F betragen. Kapazitäten dieser Größenordnung können etwa mit den *Boostcaps* von Maxwell [26] in der Größe einer Monozelle ($33 \times 33 \times 62$ mm^3) realisiert werden. Für längere Überbrückungszeiten oder stärkere Verbraucher wird eine entsprechend größeren Kapazität benötigt.

Aus Abbildung 3.1 wird auch ersichtlich, dass der TPS61201 nicht nur als *boost converter* zur Spannungserhöhung, sondern auch als *buck converter* zur Spannungsverkleinerung eingesetzt werden kann. Der notwendige Betriebsmodus wird dabei vom Spannungswandler automatisch erkannt und angewendet. Dies ermöglicht den Einsatz eines Li-Ion Akkumulators, ohne dessen Zellenspannung durch einen separaten Spannungswandler verringern zu müssen. Alternativ wären zwar auch drei serielle Nickel-basierte Akkumulatoren denkbar, aber die Lithium-Technologie ist diesen in allen in Tabelle 3.1 gezeigten Eigenschaften überlegen. Besonders die hohe Ladeeffizienz und die geringe Selbstentladung sind nach Ausdruck (1.4a) entscheidende Faktoren bei der Dimensionierung der Energiequelle für den energieneutralen Betrieb. Außerdem muss der Sekundärspeicher aus dem sehr viel kleineren Primärspeicher geladen werden. Der Akkumulator wird also nicht vollständig auf und wieder entladen, sondern in sehr flachen Ladezyklen betrieben. Die nutzbare Kapazität der Nickel-basierten Akkumulatoren würde daher durch den Memory-Effekt (NiCd) bzw. den Batterieträgheitseffekt (NiMH) verringert werden [8]. Nicht zuletzt ist die Verwendung einer einzelnen Li-Ion-Zelle der seriellen Verschaltung von drei NiMH-Zellen vorzuziehen, weil damit der Zustand einer unbalancierten Energieverteilung ausgeschlossen wird.

Zum Laden von Li-Ion-Akkumulatoren wird das CC-CV-Verfahren nach Abbildung 3.2 verwendet. Dabei wird zunächst die Zellenspannung durch die Zuführung eines konstanten Ladestroms (CC) bis zu einer Obergrenze erhöht. Bei NiMH-Akkumulatoren entspricht dies etwa 1,4 V und wird als Ladeschlussspannung bezeichnet, weil der Speicher dann bereits voll geladen ist[1]. Li-Ion-Akkus sind dagegen erst zu etwa 70 % geladen, wenn sie ihre maximale Zellenspannung von 4,2 V erreichen [5]. Die restliche Energie muss dann in der zweiten Phase mit konstanter Spannung (CV) und fallendem Ladestrom zugeführt werden. Der Ladevorgang endet, wenn der Ladestrom unter einen Schwellwert fällt.

Die Höhe des Ladestroms wird als C-Rate im Verhältnis zur nominalen Kapazität des Speichers angegeben. So entspricht der initiale Ladestrom von 1800 mA in Abbildung 3.2 gerade 1 C, weil die Kapazität des Li-Ion-Akkus 1800 mAh beträgt. Verwendet man statt der üblichen Laderaten von 0,5 C bis 1 C ein langsames Ladeverfahren mit 0,1 C bis 0,2 C, so kann man das

[1] exaktere NiMH-Ladeverfahren erkennen das Ende der Ladekurve an einem leichten Spannungsabfall oder dem Anstieg der Zelltemperatur [12]

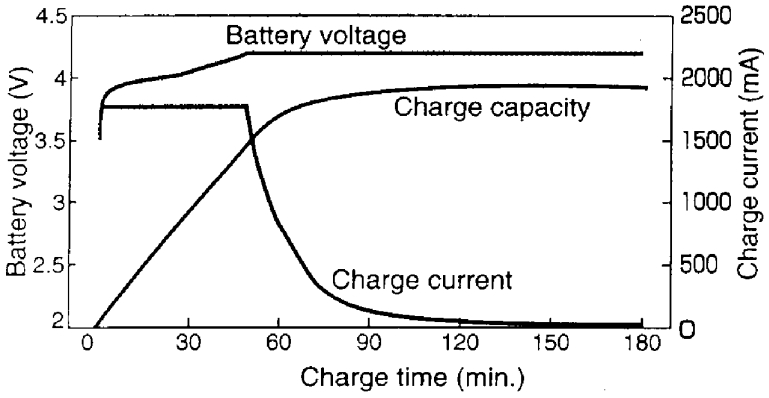


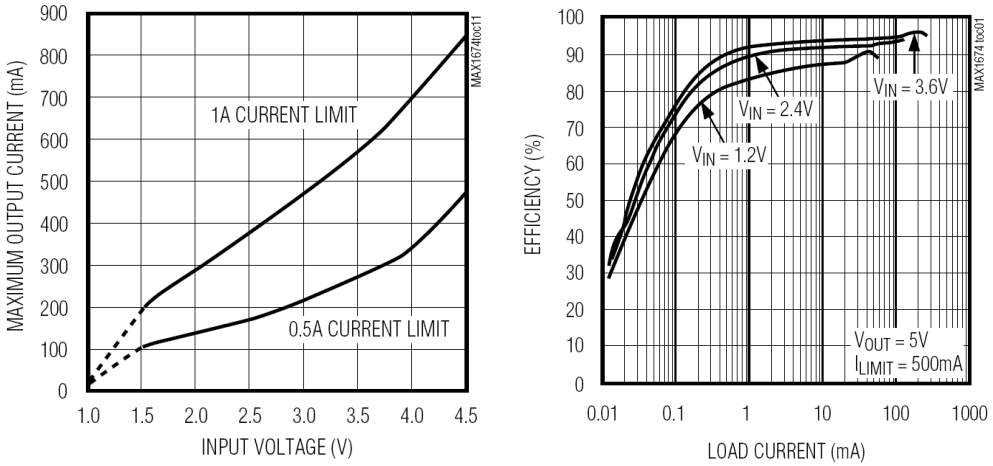
Abbildung 3.2: CC-CV Ladeschema für einen 1800 mAh Li-Ion-Akkumulator [8]

CC-CV-Schema durch das einfachere CC-Schema annähern, weil der Akku dann am Ende der verlängerten ersten Phase bereits fast vollständig geladen ist [8].

Für den Energietransport vom kleinen Primär- zum großen Sekundärspeicher ist das langsamere CC-Ladeverfahren mit kleinen C-Raten gut geeignet, denn bei höheren Strömen würde die Primärspannung schneller abfallen, so dass die zum Beenden des Ladevorgangs notwendige Ladelogik (vgl. Abschnitt 5.2) mit höheren Abtastraten betrieben werden müsste. Die Verlängerung der Ladedauer durch die kleineren Ladeströme ist für SNoW^5-RA nicht von Bedeutung. Da die Ladedauer außerdem in erster Linie durch die Kapazität des Primärspeichers begrenzt wird, ist eine Überladung des Sekundärspeichers kaum möglich.

Da die 3,3 V Ausgabespannung des TPS61201 nicht ausreichen, um den Li-Ion-Akku zu laden, wird ein weiterer Spannungswandler benötigt. Hier bietet sich die Verwendung des MAX1675 [24] *boost converters* von Maxim an, denn zum einen verfügt dieser über eine für das CC-Ladeverfahren notwendige Strombegrenzung und zum anderen kann er über einen *Shutdown Pin* an- und abgeschaltet werden, wodurch sich der Ladevorgang steuern lässt. Seine Ausgabespannung von 5 V kann durch einen Spannungsteiler auf die notwendigen 4,2 V angepasst werden. Weil Li-Ion-Akkus aber bereits bei einer Überladung von 50 mV geschädigt werden können und im vollen Ladezustand generell schneller altern [5], ist es sogar sinnvoll, die Sekundärspannung auf etwa 4 V zu begrenzen.

Aus Abbildung 3.3a ist ersichtlich, dass der maximale Ausgabestrom des Spannungswandlers mit fallender Eingangsspannung sinkt. Um einen konstanten Ladestrom zu gewährleisten, wird daher nicht die Kondensatorspannung selbst, sondern die bereits auf 3,3 V stabilisierte Ausgangsspannung des TPS61201 zur Versorgung des MAX1675 verwendet. Damit wird der Ladestrom auf etwa 250 mA begrenzt, was für den 2600 mAh Li-Ion-Akkumulator L18650 [35] weniger als 0,1 C, also einer langsamen und damit sicheren Laderate entspricht. Die Effizi-



(a) Strombegrenzung **(b)** Effizienz

Abbildung 3.3: MAX1675 *boost converter* [24]

enz des MAX1675 beträgt bei dieser Konfiguration nach Abbildung 3.3b etwa 95% , so dass die Verluste beim Energietransport vom Primär- zum Sekundärspeicher in erster Linie durch den TPS61201 Spannungswandler vorgegeben werden. Aus Abbildung 3.1 kann man diese mit 15-30 % abgeschätzten, solange die Primärspannung nicht unter $U_{cap}^{dl} := 1{,}5$ V fällt. Darunter sinkt die Effizienz des Spannungswandlers stark ab, so dass der Ladevorgang an dieser Stelle abgebrochen werden muss, um die Energieverluste zu begrenzen. Aus der Primärspannungsänderung $\Delta U_{cap} = U_{cap}^{max} - U_{cap}^{dl} = 0.9$ V, dem Ladestrom $I_{cap} = 250$ mA und der Kondensatorkapazität $C_{cap} \approx 350$ F kann die Dauer des Ladevorgangs auf $\frac{C_{cap}\Delta U_{cap}}{I_{cap}} \approx 21$ min abgeschätzt werden.

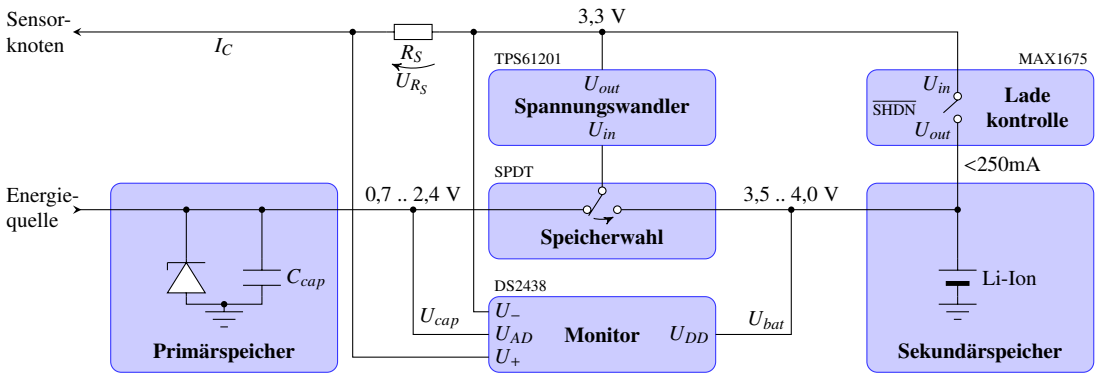


Abbildung 3.4: hierarchischer Energiespeicher mit Sensoren und Aktoren

Abbildung 3.4 fasst das hierarchische Speicherkonzept zusammen. Der SPDT-Wechselschalter zur Speicherauswahl und der *Shutdown Pin* ($\overline{\text{SHDN}}$) des MAX1675 dienen als Aktoren zur Cache-Verwaltung und werden von SNoW^5-RA über GPIO-Pins angesteuert (vgl. Abschnitt 5.2). Die dafür notwendigen Informationen über die aktuellen Speicherspannungen werden mit Hilfe eines DS2438 Batteriemonitors [23] gewonnen. Zwar könnten die Spannungen auch direkt mit den Analog-Digital-Wandlern der MSP430-MCU bestimmt werden, der Batteriemonitor besitzt aber einen zusätzlichen Ladungsakkumulator, mit dem der Energieverbrauch des Sensorknotens aus der Summe der durch den Messwiderstand R_S zum Sensorknoten transportierten Ladung bestimmt werden kann. Die Details der Ansteuerung des Monitors werden in Abschnitt 5.1 beschrieben, während die Implementierung des dafür notwendigen seriellen Kommunitationsprotokolls (Dallas 1-Wire) im Anhang B zu finden ist.

Teil II

Software

KAPITEL

4 Konzept zur Steuerung des Energieverbrauchs

4.1 Möglichkeiten zur Beeinflussung des Energieverbrauchs

Kapitel 1 zeigt, dass der Energieverbrauch des Sensorknotens an die aus der Umgebung gewonnene Energiemenge angepasst werden muss, um einen optimalen energieneutralen Betrieb zu erreichen. Zum einen darf im langfristigen Mittel nicht mehr verbraucht werden, als die Energiequelle liefern kann. Auf der anderen Seite muss jede Energieeinheit, die nicht direkt verbraucht wird, im verlustbehafteten Speicher abgelegt werden. Da das System zur Laufzeit keinen Einfluss auf die Energieeinnahmen hat, bleibt nur die gezielte Steuerung des Verbrauchers. Dazu gibt es verschiedene Ansätze, die allesamt mit einer Performanzskalierung des Systems einhergehen. Außerdem kann der Verbrauch nur in endlichen Grenzen reguliert werden ($0 < \rho_C^{min} \leq \rho_C \leq \rho_C^{max}$), wenn man von einer vollständigen Deaktivierung des Systems absieht.

Nach [6] ist der dynamische Verbrauch einer mit der Versorgungsspannung U_{cc} betriebenen und mit f_{CLK} getakteten CMOS Schaltung

$$\rho_C \propto U_{cc}^2 \cdot f_{CLK}. \tag{4.1}$$

Die Ursache für diese Abhängigkeit ist der Energieverlust beim Umladen der Gate-Kapazität der Transistoren ($\Delta E \propto \Delta U^2$) bei jeder Zustandsänderung eines Logikgatters, also potentiell in jedem Takt. Die Änderung der Taktrate f_{CLK} (*dynamic frequency scaling*) beeinflusst somit den Energieverbrauch pro Zeiteinheit, aber nicht den Energieverbrauch pro Instruktion und damit auch nicht den Energieverbrauch für ein auszuführendes Programm. Um den Verbrauch langfristig zu senken, ist es bei Systemen mit unterschiedlichen Betriebsmodi (vgl. Abschnitt 1.3) sogar sinnvoller, die größtmögliche Taktrate zu verwenden, um die Zeit im aktiven Modus zu

minimieren und schnellstmöglich wieder in den *idle*-Modus zu wechseln. Eine Änderung der Versorgungsspannung (*dynamic voltage scaling*) wirkt sich auf die Umladeströme und damit auf die Reaktionszeit der Transistoren aus [6]. Da die maximale Taktrate aus der Länge des kritischen Pfads der Schaltung resultiert, muss mit der Spannung auch die Frequenz reduziert werden. Die Verringerung des Energieverbrauchs durch das Absenken der Versorgungsspannung ist durch die Schwellenspannung zum Öffnen der Transistoren begrenzt.

Die wichtigsten Komponenten (MCU und Funkchip) des SNoW^5-Sensorknotens können mit Spannungen zwischen 1,8 und 3,6 V betrieben werden [3]. Auch unterstützt die MSP430 MCU eine grobe Anpassung der Taktfrequenz zur Laufzeit. Für das Echtzeitbetriebssystem SmartOS sind solche Änderungen in der Reaktionszeit des Systems aber problematisch und daher ist *dynamic voltage/frequency scaling* für den SNoW^5-Sensorknoten eher ungeeignet.

Eine weitere Stellschraube zur Modifikation des Energieverbrauchs ist die Signalstärke des Funkchips, falls die Kommunikationskosten einen signifikanten Anteil am Gesamtverbrauch ausmachen und die erhöhte Übertragungsfehlerwahrscheinlichkeit bei niedrigen Funksignalstärken für die jeweilige Anwendung akzeptabel ist.

Die MCUs der meisten Sensorknoten unterstützen verschiedene Betriebsmodi (LPM), welche durch die Deaktivierung von Teilkomponenten (CPU, Taktgeber oder anderer Peripherieeinheiten [44]) die verbrauchte Leistung auf Kosten der Handlungsfähigkeit des Systems reduzieren. Auch der SNoW^5-Sensorknoten wird von seinem Betriebssystem (SmartOS) in einem aktiven und einem schlafenden (*idle*) Modus betrieben (siehe Abschnitt 1.3). Über den Anteil $D \in [0, 1]$ der Laufzeit im aktiven Modus an der Gesamtlaufzeit kann der Verbraucher gezielt beeinflusst werden, denn

$$\rho_C^{idle} \leq \rho_C(D) = D \cdot \rho_C^{active} + (1 - D) \cdot \rho_C^{idle} = \rho_C^{idle} + D \cdot (\rho_C^{active} - \rho_C^{idle}) \leq \rho_C^{active}. \tag{4.2}$$

Da diese als *Aaptive Duty Cycling* [13] bezeichnete Form der Verbrauchsskalierung vom Betriebssystem des SNoW^5-Sensorknotens bereits weitgehend unterstützt wird und nahezu anwendungsunabhängig realisiert werden kann (es ist nicht wichtig, was in den aktiven Phasen berechnet wird), soll seine konkrete Umsetzung im Folgenden näher betrachtet werden.

4.2 *Adaptive Duty Cycling* durch Skalierung periodischer Tasks

Ein Kernkonzept von SmartOS ist eine in sich abgeschlossene Programmeinheit mit einer Einsprungadresse aber ohne Rücksprung [3]. Bis zu 255 solcher *Tasks* können definiert werden, wobei zu jedem Zeitpunkt genau einer aktiv ist. Diesem steht dann die Rechenleistung der CPU zur Verfügung, um seine Instruktionen der Reihe nach auszuführen. Ein Taskwechsel, also die Suspendierung des aktiven Tasks und Aktivierung eines anderen wird vom *Scheduler* des Betriebssystems gesteuert und basiert auf *Events*, *Deadlines* und den Prioritäten der Tasks. So wird ein Task suspendiert, wenn er auf das Eintreten eines Events oder das Erreichen einer Deadline wartet, oder wenn er von einem Task mit höherer Priorität verdrängt wird (siehe Abbildung 4.1).

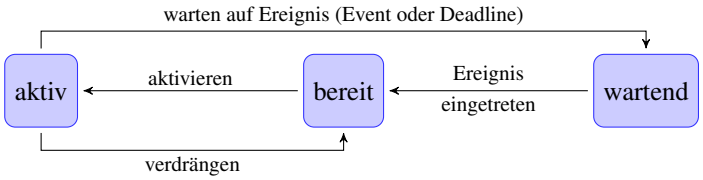


Abbildung 4.1: Zustandsübergänge von SmartOS Tasks (in Anlehnung an [9])

Verdrängte Tasks sind sofort wieder bereit zur Aktivierung, während wartende Task erst nach Eintreten des Ereignisses (Event oder Deadline) wieder bereit sind. Ein Task wird aktiviert, wenn er dazu bereit ist und kein Task mit höherer Priorität aktiv oder bereit ist.

Events werden explizit durch Aufruf einer entsprechenden Funktion oder implizit durch Freigabe einer Ressource ausgelöst und vom darauf wartenden Task konsumiert. Ein einmalig ausgelöstes Event kann also nicht mehrere Tasks aktivieren. Während die implizite Auslösung nur aus einem Task (dem Besitzer der freizugebenden Ressource) heraus erfolgen kann, ist das explizite Auslösen auch aus einer ISR möglich. Diese werden vom Betriebssystem beim Eintreten externer Ereignisse (*Interrupts*) in bestimmten Peripheriekomponenten (wie Timer, DMA-Controller oder Kommunikationsschnittstellen) aufgerufen.

Der vordefinierte *idle*-Task mit niedrigster Priorität wird immer dann aktiv, wenn alle anderen Tasks warten müssen. Er versetzt die MCU durch Abschalten der CPU und des Haupttaktgebers (MCLK) in den *idle*-Modus (LPM1), um den Energieverbrauch zu reduzieren (siehe Abschnitt 1.3). Für das *Adaptive Duty Cycling* zur Steuerung des Energieverbrauchs müssen also die Ausführungsdauer des *idle*-Tasks bzw. die Aktivitäten der übrigen Tasks beeinflusst werden können. Im Folgenden wird eine Möglichkeit zur Klassifikation von Tasks untersucht, mit der die Aktivität des gesamten System möglichst unabhängig vom Anwendungsszenario, also von der konkreten Implementierung der Tasks, beeinflusst werden kann.

Da Tasks keinen Rücksprung besitzen, müssen sie in einer Endlosschleife organisiert sein, wobei der Schleifenrumpf die eigentliche Aufgabe des Tasks realisiert. In der Regel werden die Schleifenrümpfe aber nicht ununterbrochen nacheinander ausgeführt, weil dies Tasks mit niedrigerer Priorität unterdrücken würde.

Definition 4.1 (Triggerung von Tasks)**.** *Ein Task wird von einem Ereignis (Event oder Deadline) getriggert, wenn sein Schleifenrumpf beim Eintreten dieses Ereignisses ausgeführt wird.*

Als Trigger sind also nur die Ereignisse zu verstehen, welche die Abarbeitung der vom Task realisierten Aufgabe anstoßen. Da SmartOS das gleichzeitige Warten auf ein Event oder eine Deadline erlaubt, kann ein Task auch von beiden Ereignissen getriggert werden.

Definition 4.2 (Klassifikation von Tasks)**.** *Ein Task t heißt periodisch, wenn er ausschließlich durch den Ablauf einer Deadline getriggert wird. Die Zeit $c \in \mathbb{R}^+$ zwischen zwei aufeinanderfolgenden Ausführungen heißt Periode von t. Ein periodischer Task heißt skalierbar, wenn seine Periode zwischen einer unteren und einer oberen Grenze $c^{\min}, c^{\max} \in \mathbb{R}^+$ gewählt werden kann ($c^{\min} \leq c \leq c^{\max}$), wobei $c^{\min} < c^{\max}$ gefordert wird. Nicht periodische Tasks heißen sporadisch und werden bis auf den* idle*-Task durch ein Event getriggert. Jeder Task bzw. jedes externe Ereignis, das dieses Event auslösen kann, heißt Trigger des sporadischen Tasks.*

Es seien $\mathcal{T} := \{t_0, \ldots, t_{n-1}\}$ die Tasks des Systems und o.B.d.A.

- *t_0 der* idle*-Task,*
- *$\mathcal{T}_P := \{t_1, \ldots, t_p\} \subset \mathcal{T}$ die Menge der periodischen sowie*
- *$\mathcal{T}_S := \{t_1, \ldots, t_s\} \subseteq \mathcal{T}_P$ die Menge der skalierbaren Tasks.*

Ein skalierbarer Task könnte beispielsweise alle fünf bis zehn Minuten einen Temperatursensor abfragen, die Messdaten in einen temporären FIFO-Speicher schreiben und ein Event auslösen, falls der Speicher voll ist. Ein sporadischer Task wartet auf genau dieses Event, um den Inhalt des Speichers an eine zentrale Datensenke zu versenden, den Speicher anschließend zu leeren und auf seine erneute Auffüllung zu warten. In diesem Sinne realisieren nicht extern getriggerte sporadische Tasks wiederverwendbare Teilaufgaben, die zur Realisierung komplexerer Aufgaben beitragen. Nach [9] können Tasks somit in einer Hierarchie gruppiert werden, bei der die periodischen Tasks die oberste Ebene bilden, weil die von ihnen realisierte Aufgabe eigenständig und nicht Teil einer komplexeren Aufgabe ist. Solche Hierarchien können durch einen Graphen veranschaulicht werden.

Definition 4.3 (Triggergraph)**.** *Sei $\mathcal{I}$ die Menge der externen Ereignisse, welche einen (sporadischen) Task triggern können. Ein Triggergraph $G_{trig} = (V, E)$ ist ein endlicher, gerichteter Graph mit*

- *Knotenmenge $V = \mathcal{T} \cup \mathcal{I}$ und*
- *Kantenmenge $E = \{(a, b) \in V \times \mathcal{T} : a \text{ ist Trigger von } b\}$.*

Die transitive Erreichbarkeit in G_{trig} wird durch

$$H_{trig} : a \in \mathcal{T} \mapsto \{b \in \mathcal{T} : a = b \vee \exists\, (c, b) \in E : c \in H_{trig}(a)\}$$

beschrieben und sei als

$$H_{trig} : i \in \mathcal{I} \mapsto \bigcup_{(i,a) \in E} H_{trig}(a)$$

$$H_{trig} : A \subseteq V \mapsto \bigcup_{a \in A} H_{trig}(a)$$

auf externe Ereignisse und Knotenteilmengen erweitert.

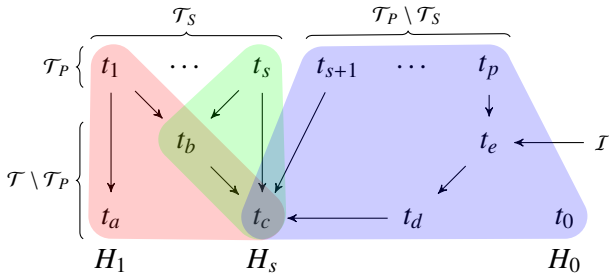


Abbildung 4.2: Beispiel für einen Triggergraph der SmartOS Tasks

Ein Beispiel für eine solche Hierarchie zeigt Abbildung 4.2. Der *idle*-Task t_0 hat weder ausgehende noch einlaufende Kanten, da er kein Event auslöst und auf kein Event wartet. Periodische Tasks können sporadische Tasks triggern, haben aber selbst keine einlaufenden Kanten, da sie nicht von anderen Tasks getriggert werden. Die skalierbaren Tasks bieten nun die Möglichkeit, die Aktivität des Systems in gewissen Grenzen zu beeinflussen. Mit jeder Ausführung von $t_i \in \mathcal{T}_S$ ist auch die Ausführung der sporadischen Tasks aus $H_i := H_{trig}(t_i)$ und damit ein bestimmter Energieverbrauch w_i verbunden. Wäre t_i nicht ausgeführt worden, dann hätte der Sensorknoten die entsprechende Laufzeit der Tasks aus H_i im *idle*-Modus verbracht. Daher repräsentiert w_i nicht den absoluten Energiebedarf für eine Ausführung von t_i (bzw. H_i), sondern den Mehrverbrauch über *idle*-Niveau in der selben Zeit. Durch eine Veränderung der Periode c_i von t_i wird w_i mehr oder weniger oft pro Zeiteinheit verbraucht und damit die mittlere Verbrauchsleistung reguliert. Die Ausführung der Tasks aus $H_0 := H_{trig}(\{t_0\} \cup \mathcal{I} \cup (\mathcal{T}_P \setminus \mathcal{T}_S))$ kann hingegen nicht beeinflusst werden und führt zur Grundverbrauchsleistung ρ_0. Für einen einheitlichen Formalismus ist es sinnvoll, diesen Grundverbrauch durch einen imaginären Task zur repräsentieren.

Definition 4.4 (Repräsentation des Grundverbrauchs als imaginärer Task)**.** *Statt des idle-Tasks bezeichne t_0 von nun an einen imaginären, pseudoskalierbaren Task mit fester Periode $c_0^{\min} := c_0^{\max} := c_0 \in \mathbb{R}^+$. Mit jeder Ausführung dieses Tasks sei der Energieverbrauch $w_0 := \rho_0\, c_0$ verbunden.*

In einem Zeitraum der Länge T entspricht der durch die $\frac{T}{c_0}$-fache Ausführung von t_0 verursachte Verbrauch $\frac{T}{c_0} w_0 = T\rho_0$ gerade dem Grundverbrauch in dieser Zeit. Da t_0 die Anforderung $c_0^{\min} < c_0^{\max}$ von Definition 4.2 nicht erfüllt, gilt weiterhin $t_0 \notin \mathcal{T}_S$. Wenn im Folgenden von skalierbaren Tasks die Rede ist, so bezieht dies aber t_0 mit ein, solange nicht anderweitig angegeben.

Der Verbrauchsvektor $\boldsymbol{w} := (w_0, \ldots, w_s) \in \mathbb{R}^{s+1}$ ist a priori unbekannt und muss zur Laufzeit durch Energiebilanzierung erlernt bzw. angenähert werden. Dabei ist w_i im Allgemeinen nicht

für jede Ausführung von t_i gleich, da die Entscheidung, ob ein Task ein Event auslöst und damit einen anderen Task triggert, von den verschiedensten Laufzeitfaktoren abhängig sein kann. So würde der periodische Messtask aus obigem Beispiel mehrfach ausgeführt werden, ohne den sporadischen Task zum Versenden des Speicherinhalts zu triggern. Die Messung, die zum Volllaufen des Speichers führt, ist somit energetisch teurer als die vorherigen Messungen. Zu dieser Abhängigkeit vom konkreten Kontrollfluss kommen für die Grundverbrauchskomponente w_0 noch die Beeinflussung durch die im Allgemeinen unregelmäßig auftretenden externen Ereignisse hinzu. Für eine repräsentative Ausmittelung von $\boldsymbol{w}$ muss der als *Slot* bezeichnete Zeitraum T, über den die Energiebilanzierung erfolgt, daher ausreichend lang gewählt werden. Um zwischen dem tatsächlichen Verbrauchsvektor und der von SNoW^5-RA erlernten Annahme zu unterscheiden, wird letztere als $\widetilde{\boldsymbol{w}}$ gekennzeichnet.

Ein Arbeitspunkt des Systems wird als Periodenvektor $\boldsymbol{c} := (c_0, \ldots, c_s) \in \mathbb{R}^{s+1}$ beschrieben und liegt wegen der Beschränkung

$$c_i^{\min} \leq c_i \leq c_i^{\max} \quad \forall\, i \in \{0, \ldots, s\} \tag{4.3}$$

in einem abgeschlossenen s-dimensionalen Hyperquader[1] (vgl. [9]). Wird der Arbeitspunkt $\boldsymbol{c}$ während eines Slots der Länge T nicht verändert, dann wird jeder skalierbare Task t_i während dieses Slots $\frac{T}{c_i}$-fach ausgeführt, so dass

$$W(\boldsymbol{c}, \boldsymbol{w}, T) = T \sum_{i=0}^{s} \frac{w_i}{c_i} \tag{4.4}$$

den Energieverbrauch des Sensorknotens beschreibt. Für gegebenes $\boldsymbol{w}$ und T liegen alle Arbeitspunkte $\boldsymbol{c}$, die nach (4.4) zu einem bestimmten Gesamtverbrauch W führen, auf einer $(s-1)$-dimensionalen Hyperebene[1]. Abbildung 4.3 veranschaulicht diesen Sachverhalt für zwei skalierbare Tasks im Periodenraum und im dazu inversen Frequenzraum ($f_i = \frac{1}{c_i}$).

Schneidet die Hyperebene den Hyperquader, wie es in der Abbildung bei W^b der Fall ist, dann kann jeder Arbeitspunkt auf der eingeschlossenen Schnittfläche verwendet werden, um den gewünschten Verbrauch zu erzielen. Gibt es hingegen keinen Schnittpunkt (vgl. W^a und W^c), so liegt der Zielverbrauch außerhalb des regulierbaren Bereichs, welcher durch

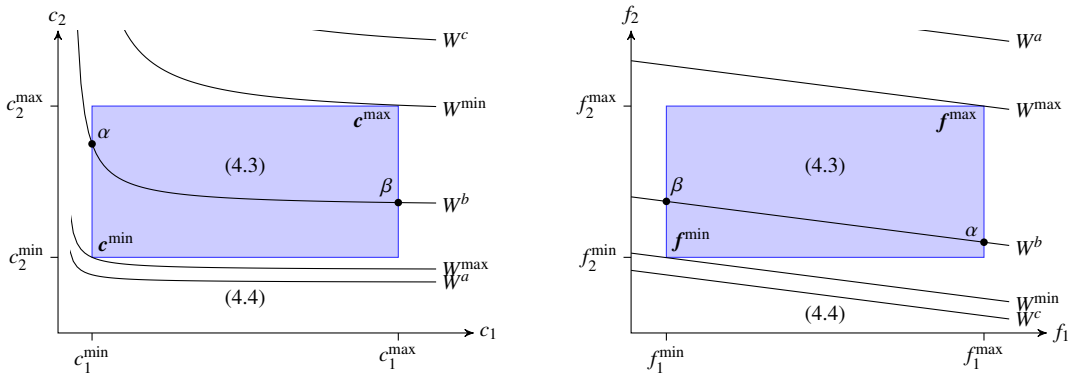
$$W^{\min} := W(\boldsymbol{c}^{\max}, \boldsymbol{w}, T) \qquad \text{und} \qquad W^{\max} := W(\boldsymbol{c}^{\min}, \boldsymbol{w}, T) \tag{4.5}$$

begrenzt wird, und kann nur durch Einstellung des entsprechenden extremalen Arbeitspunktes

$$\boldsymbol{c}^{\max} := (c_0^{\max}, \ldots, c_s^{\max}) \qquad \text{bzw.} \qquad \boldsymbol{c}^{\min} := (c_0^{\min}, \ldots, c_s^{\min}) \tag{4.6}$$

angenähert werden.

[1] wegen der Fixierung von c_0 entfällt eine Dimension



(a) Periodenraum **(b)** Frequenzraum

Abbildung 4.3: Hyperquader (4.3) und Hyperebenen (4.4) für verschiedene Verbrachswerte und zwei skalierbare Tasks

4.3 Nutzen skalierbarer Tasks

Um die Wahl des Arbeitspunktes auf dem Schnitt zwischen Hyperquader und Hyperebene nicht willkürlich zu treffen, wird ein weiteres Auswahlkriterium benötigt. Mit der Ausführungsfrequenz $f_i = \frac{1}{c_i}$ eines skalierbaren Tasks steigt der Nutzen des Tasks für das System bzw. die Qualität der von ihm erfassten Messdaten, da c_i als Reaktionszeit von Überwachungsaufgaben bzw. f_i als Abtastrate von Messungen interpretiert werden kann [17].

Definition 4.5 (relativer energetischer Nutzen skalierbarer Tasks)**.** *Für jeden skalierbaren Task $t_i \in \mathcal{T}_S$ sei $b_i \in \mathbb{N}$ der relative energetische Nutzen von t_i, wobei $b_i > b_j$ genau dann gelten soll, wenn die Erhöhung von f_i für das System einen größeren Nutzen hat als die Erhöhung von f_j.*

Nach [9] kann man den Gesamtnutzen des Systems bei der Wahl des Arbeitspunktes durch einen Algorithmus mit linearer Laufzeitkomplexität maximieren. Dieser Greedy-Algorithmus startet in einem extremalen Arbeitspunkt und läuft auf dem Rand des Hyperquaders in Richtung Hyperebene. Eine Bewegung auf dem Rand entspricht der Veränderung einer einzelnen Komponente c_i des Arbeitspunktes bis zu dessen Begrenzung ($c_i^{\min}$ oder $c_i^{\max}$) oder bis zum Erreichen der Hyperebene. Die Reihenfolge, in der die Komponenten geändert werden, also die Folge der abzulaufenden Kanten des Hyperquaders wird durch den Nutzenvektor $\boldsymbol{b} := (b_1, \ldots, b_s) \in \mathbb{N}^s$ bestimmt. So wird bei der Erhöhung des Energieverbrauchs zuerst die Periode des Tasks mit höchstem Nutzen minimiert. Umgekehrt wird beim Verringern des Energieverbrauchs zuerst die Periode des Tasks mit niedrigstem Nutzen maximiert. Der Algorithmus terminiert nach höchstens s betrachteten Kanten beim Erreichen der Hyperebene oder eines extremalen Arbeitspunktes. In Abbildung 4.3 würde für $b_1 > b_2$ der Arbeitspunkt α und für $b_1 < b_2$ der Arbeitspunkt β bestimmt werden, wenn W^b als Zielverbrauch vorgegeben ist.

Die Grenzen $\boldsymbol{c}^{\min}$ und $\boldsymbol{c}^{\max}$ und damit auch der Hyperquader (4.3) bleiben zur Laufzeit des Systems unverändert. Die Hyperperebene (4.4) ist aber abhängig vom Zielverbrauch W des Sensorknotens und somit veränderlich. Bei einer Verschiebung der Hyperebene durch eine Änderung des Zielverbrauchs um ΔW ist daher eine Nachführung des Arbeitspunktes $\boldsymbol{c}$ notwendig. Dabei kann der Greedy-Algorithmus beim letzten Arbeitspunkt aufsetzten und die Verschiebungsrichtung durch $\operatorname{sgn}(\Delta W)$ sowie die Verschiebungsweite durch $|\Delta W|$ bestimmen. Kleine Änderungen im Zielverbrauch können daher durch die Veränderung weniger Taskperioden ausgegelichen werden. Die Implementierung dieser relativen Verschiebung wird im Abschnitt 7.2 beschrieben.

Nun kann man sich ohne weiteres Anwendungen vorstellen, bei denen der energetische Nutzen einzelner skalierbarer Tasks zur Laufzeit veränderbar ist. So ist der Nutzen hoher Abtastraten für Beleuchtungsmessungen in freier Umgebung während des Tages sicherlich höher als in der Nacht. Um solche dynamischen Prozesse zu unterstützen, muss eine Veränderung von $\boldsymbol{b}$ jederzeit möglich sein und unmittelbar zu einer Umverteilung der zur Verfügung stehenden Energie führen. Diese Umverteilung entspricht der Verschiebung des Arbeitspunktes zu einem anderen Schnittpunkt zwischen Hyperebene und dem Rand des Hyperquaders und wird nur notwendig, wenn der Arbeitspunkt nicht bereits extremal ist und die Änderung an $\boldsymbol{b}$ auch tatsächlich zu einer veränderten Nutzenordnung der skalierbaren Tasks geführt hat.

4.4 Slotverwaltung zur Energiebilanzierung

Ausdruck (4.4) beschreibt die Beziehung zwischen dem Arbeitspunkt $\boldsymbol{c}$ und dem Gesamtverbrauch W des Sensorknotens. Dabei gehen auch die Verbrauchskomponenten $\boldsymbol{w}$ ein, welche dem System aber nur als Näherung $\widetilde{\boldsymbol{w}}$ bekannt sind. Dies macht eine Differenzierung in der Notation des Gesamtverbrauchs notwendig. Mit W wird fortan nur noch die tatsächlich verbrauchte Energiemenge bezeichnet, während $\widetilde{W}$ den erwarteten, also auf $\widetilde{\boldsymbol{w}}$ beruhenden Energieverbrauch beschreibt. Des Weiteren sei $\widehat{W}$ der Verbrauch, der zum Erreichen bestimmter Ziele (etwa der Energieneutralität) von SNoW^5-RA vorgegeben wird.

Das Erlernen der Verbrauchskomponenten, also das Annähern des Annahmevektors $\widetilde{\boldsymbol{w}}$ an den tatsächlichen Verbrauchsvektor $\boldsymbol{w}$ erfolgt durch verschiedene Lernstrategien (siehe Kapitel 9), die auf einer Energiebilanzierung beruhen. Dafür muss am Ende eines jeden Slots bekannt sein, wie viel Energie W der Sensorknoten insgesamt verbraucht hat und wie oft die einzelnen skalierbaren Tasks ausgeführt wurden. Die Bilanzierung muss außerdem über Slots einer gewissen Mindestlänge erfolgen, da die Verbrauchswerte ausgemittelt werden müssen (vgl. Abschnitt 4.2) und die Messung von W mit einer endlichen Auflösung bzw. Messungenauigkeit behaftet ist, die bei kürzerer Slots und somit geringerem Verbrauch stärker zum Tragen kommt. Wegen der Abhängigkeit des Arbeitspunktes $\boldsymbol{c}$ vom Nutzenvektor $\boldsymbol{b}$ und dessen Änderbarkeit zur Laufzeit können die Zeitabschnitte, in denen $\boldsymbol{c}$ unverändert bleibt, aber beliebig kurz werden. Dies macht eine Energiebilanzierung nach jeder Änderung des Arbeitspunktes unmöglich, wodurch eine Unterscheidung zwischen Haupt- und Subslots notwendig wird.

Die Länge der Hauptslots ist konstant und wird fortan mit T^{main} bezeichnet. Die Energiebilanzierung erfolgt ausschließlich über die Hauptslots, so dass die Verbrauchsannahmen $\widetilde{\boldsymbol{w}}$ sowie der angestrebte Gesamtverbrauch $\widehat{W}$ während eines Hauptslots konstant bleiben. Wählt man für die Periode des imaginären Tasks $c_0 := T^{\text{main}}$, so wird mit $\widetilde{w}_0$ der Grundverbrauch des Sensorknotens während eines Bilanzierungsslots erlernt. Subslots sind echte Teilintervalle eines Hauptslots und entstehen nur aufgrund von Veränderungen des Arbeitspunktes nach Umsortierung des relativen Nutzens der skalierbaren Tasks. Innerhalb eines Subslots bleibt der Arbeitspunkt $\boldsymbol{c}$ unverändert.

Lemma 4.1. *Wird ein Hauptslot mit Verbrauchsannahme $\widetilde{\boldsymbol{w}}$ und Zielverbrauch $\widehat{W}$ in k Subslots der Länge $T^1, \ldots, T^k$ mit den Arbeitspunkten $\boldsymbol{c}^1, \ldots, \boldsymbol{c}^k$ unterteilt, dann gilt*

$$\sum_{j=1}^{k} W(\boldsymbol{c}^j, \widetilde{\boldsymbol{w}}, T^j) = \widehat{W}.$$

Beweis. Um überhaupt Subslots zu generieren, müssen die Änderungen des Nutzenvektors zu Verschiebungen des Arbeitspunktes führen. Die Hyperebene (4.4) muss den Hyperquader (4.3) demnach schneiden und der Zielverbrauch im regulierbaren Bereich

$$\widetilde{W}^{\min} = W(\boldsymbol{c}^{\max}, \widetilde{\boldsymbol{w}}, T^{\text{main}}) < \widehat{W} < W(\boldsymbol{c}^{\min}, \widetilde{\boldsymbol{w}}, T^{\text{main}}) = \widetilde{W}^{\max} \tag{A}$$

liegen. Zu Beginn des Hauptslots wird unter Vorgabe von $\widehat{W}$ der erste Arbeitspunkt $\boldsymbol{c}^1$ so gewählt, dass $W(\boldsymbol{c}^1, \widetilde{\boldsymbol{w}}, T^{\text{main}}) = \widehat{W}$ gilt, da von den später ausgelösten Subslots noch gar nichts bekannt ist. Damit ist aber die Hyperebene festgelegt, auf der die späteren Arbeitspunkte nur noch verschoben werden können, es gilt also

$$W(\boldsymbol{c}^j, \widetilde{\boldsymbol{w}}, T^{\text{main}}) = \widehat{W} \quad \forall\ j \in \{1, \ldots, k\}. \tag{B}$$

Damit folgt

$$\begin{aligned}
\sum_{j=1}^{k} W(\boldsymbol{c}^j, \widetilde{\boldsymbol{w}}, T^j) &\overset{(4.4)}{=} \sum_{j=1}^{k} T^j \sum_{i=0}^{s} \frac{\widetilde{w}_i}{c_i^j} \\
&= \sum_{j=1}^{k} \frac{T^j}{T^{\text{main}}} T^{\text{main}} \sum_{i=1}^{s} \frac{\widetilde{w}_i}{c_i^j} \\
&\overset{(4.4)}{=} \sum_{j=1}^{k} \frac{T^j}{T^{\text{main}}} W(\boldsymbol{c}^j, \widetilde{\boldsymbol{w}}, T^{\text{main}}) \\
&\overset{(B)}{=} \sum_{j=1}^{k} \frac{T^j}{T^{\text{main}}} \widehat{W} = \frac{\widehat{W}}{T^{\text{main}}} \sum_{j=1}^{k} T^j = \frac{\widehat{W}}{T^{\text{main}}} T^{\text{main}} = \widehat{W}.
\end{aligned}$$

□

Dieses Lemma zeigt, dass die Arbeitspunktverschiebung durch die Subslots die Steuerung des Gesamtverbrauchs während des Hauptslots nicht beeinflusst, wenn der Arbeitspunkt für jeden Subslot so eingestellt wird, als würde er für den gesamten Hauptslot gelten (vgl. Aussage (B)).

Wegen der möglichen Verschiebung des Arbeitspunktes während eines Hauptslots können die für die Energiebilanzierung notwendigen Ausführungshäufigkeiten der skalierbaren Tasks nicht mehr aus deren Perioden abgeleitet werden. Es muss daher ein Zähler e_i für jeden Task $t_i \in \mathcal{T}_S$ am Anfang jedes Hauptslots initialisiert und bei jeder Ausführung von t_i inkrementiert werden. Einzig $e_0 = 1$ kann direkt aus der obigen Festlegung $c_0 = T^{\text{main}}$ geschlossen werden. Der so entstehende Ausführungsvektor $\boldsymbol{e} := (e_0, \ldots, e_s) \in \mathbb{N}^{s+1}$ wird am Ende eines Hauptslots unter Berufung auf folgendes Lemma für die Energiebilanzierung verwendet.

Lemma 4.2. *Wird ein Hauptslot mit Verbrauchsannahme $\widetilde{\boldsymbol{w}}$ und Zielverbrauch $\widehat{W}$ in k Subslots der Länge $T^1, \ldots, T^k$ mit den Arbeitspunkten $\boldsymbol{c}^1, \ldots, \boldsymbol{c}^k$ unterteilt und am Ende des Hauptslots der Ausführungsvektor $\boldsymbol{e}$ festgestellt, dann gilt*

$$\boldsymbol{e} \cdot \widetilde{\boldsymbol{w}} = \widehat{W}.$$

Beweis. Die Ausführungshäufigkeit von t_i im Subslot j beträgt $e_i^j = \frac{T^j}{c_i^j}$ und es folgt

$$\boldsymbol{e} \cdot \widetilde{\boldsymbol{w}} = \sum_{i=0}^{s} e_i \widetilde{w}_i = \sum_{i=0}^{s} \left(\sum_{j=1}^{k} \frac{T^j}{c_i^j} \right) \widetilde{w}_i = \sum_{j=1}^{k} T^j \sum_{i=0}^{s} \frac{\widetilde{w}_i}{c_i^j} \overset{(4.4)}{=} \sum_{j=1}^{k} W(\boldsymbol{c}^j, \widetilde{\boldsymbol{w}}, T^j) \overset{\text{Lemma 4.1}}{=} \widehat{W}.$$

□

4.5 Module der Implementierung

Die Implementierung des in den letzten Abschnitten vorgestellten Konzepts wird in mehrere Teilmodule zerlegt, um die Übersichtlichkeit und Wiederverwendbarkeit zu erhöhen. Da die in Abbildung 4.4 gezeigten Module in den nächsten Kapiteln detailliert beschrieben werden, soll hier ein kurzer Überblick über deren Aufgaben und Interaktionen genügen.

Das *Harvester Board* ist als Erweiterungsplatine des Sensorknotens konzipiert und enthält im Wesentlichen das Solarmodul mit leistungsoptimierter Ansteuerung (Kapitel 2) sowie den in Abbildung 3.4 dargestellten hierarchischen Energiespeicher (Kapitel 3). Letzterer enthält außerdem einen über die 1-Wire Kommunikationsschnittstelle (DOW) ansprechbaren DS2438 Batteriemonitor zur Überwachung der Speicherspannungen und zum Messen des Energieverbrauchs des Sensorknotes. Die Schalter zur Regulierung des Energietransports zwischen Primär- und Sekundärspeicher und zum Sensorknoten werden mittels GPIO Pins gesteuert.

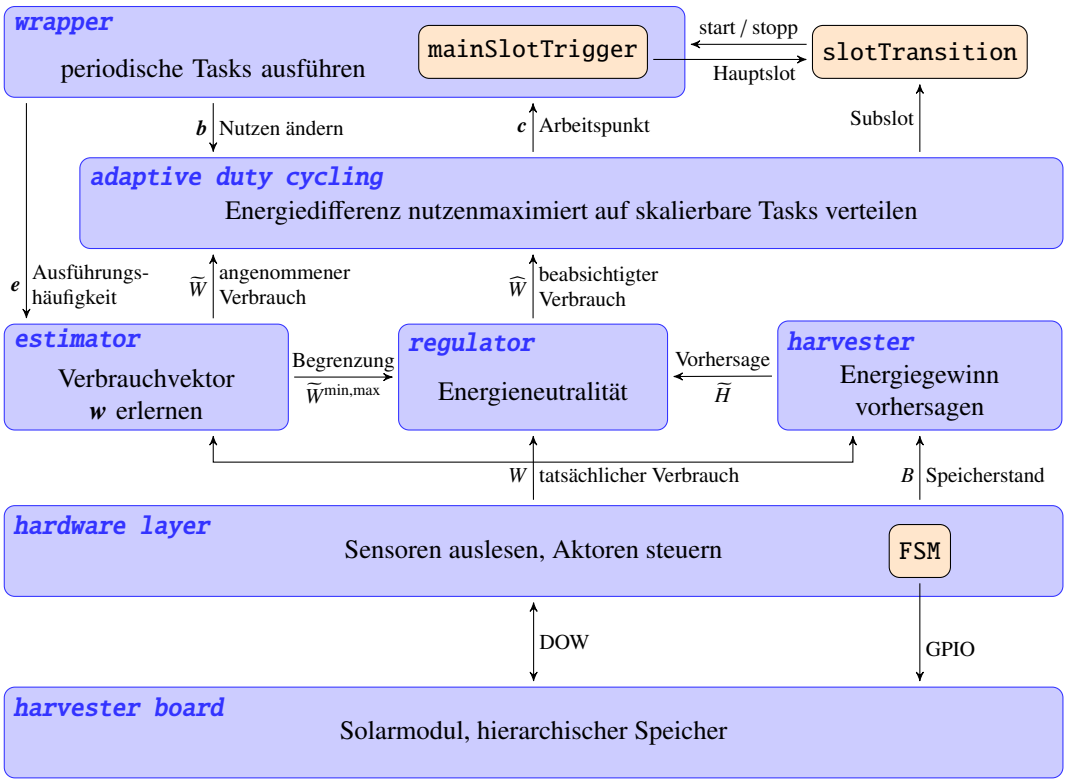


Abbildung 4.4: Modularisierung der Implementierung von SNoW^5-RA

Das *Harvester Board* wird vom *Hardware Layer* (Kapitel 5) gekapselt, um die zur Verfügung stehenden Messdaten des Speichermonitors auszulesen und für die weitere Verwendung aufzubereiten. Neben den Speicherspannungen, aus denen auch der Speicherfüllstand abgeleitet wird, ist die durch den Sensorknoten verbrauchte Energiemenge von besonderem Interesse für die Regulierung des energieneutralen Betriebs. Eine weitere Aufgabe des *Hardware Layers* ist die Überwachung der beiden Speicherspannungen sowie deren Beschränkung auf einen bestimmten Arbeitsbereich. Dazu steuert ein endlicher Zustandsautomat (FSM) über die GPIO-Pins die Auswahl des Energiespeichers sowie den Energietransport vom primären zum sekundären Speicher.

Der *Estimator* (Kapitel 9) ist für die Energiebilanzierung, also das Annähern der Verbrauchsannahmen $\widetilde{\boldsymbol{w}}$ an den tatsächlichen Verbrauchsvektor $\boldsymbol{w}$ zuständig. Dazu vergleicht es am Ende jedes Hauptslots die per *Hardware Layer* ermittelte, tatsächlich verbrauchte Energiemenge W mit der erwarteten Menge $\widetilde{W}$. Es werden verschiedene Lernstrategien kombiniert, welche unter anderem die Ausführungshäufigkeiten $\boldsymbol{e}$ der einzelnen skalierbaren Tasks während des letzten Hauptslots benötigen.

Der *Harvester* (Abschnitt 8.1) beobachtet die Füllstandsentwicklungen des Energiespeichers sowie die vom Sensorknoten verbrauchte Energie und leitet daraus eine 24-stündige Vorhersage über die Menge der geernteten Energie ab.

Diese Vorhersage für das Energieeinnahmeprofil benutzt der *Regulator* (Abschnitt 8.2) für die Steuerung des Energieverbrauchsprofils, also die Auswahl des Zielverbrauchs $\widehat{W}$ innerhalb der vom *Estimator* vorgegebenen Grenzen $\widetilde{W}^{\min}$ und $\widetilde{W}^{\max}$. Ziel der Regulierung ist neben der langfristigen Energieneutralität die Minimierung der Belastung des Energiespeichers.

Die Differenz zwischen dem vom *Regulator* vorgegebenen Zielverbrauch $\widehat{W}$ und dem aktuell angenommenen Energieverbrauch $\widetilde{W}$ wird beim *Adaptive Duty Cycling* (Kapitel 7) durch Verschiebung des Arbeitspunktes $\boldsymbol{c}$ ausgeglichen. Dazu wird eine Adaption des nutzenoptimierenden Greedy-Algorithmus aus [9] verwendet.

Der *Wrapper* (Kapitel 6) übernimmt dem aktuellen Arbeitspunkt gemäß die Ausführung der periodischen Tasks. Es setzt auf dem SmartOS-*Scheduler* auf und erfordert daher keine direkten Eingriffe in das Betriebssystem des Sensorknotens.

Ein sporadischer Task (`ra_slotTransition`) koordiniert die einzelnen Module und generiert dadurch Haupt- und Subslots nach Algorithmus 4.1. Die Hauptsloterzeugung wird dabei von einem periodischen Task (`ra_mainSlotTrigger`) mit der Periode T^{main} ausgelöst, welche über die Compilezeitkonstante `RA_MAIN_SLOT_LENGTH` spezifiziert werden kann. Subslots werden nur bei einer Änderung des Nutzenvektors $\boldsymbol{b}$ durch einen beliebigen Anwendungstask ausgelöst. Während des Übergangs von einem Slot zum nächsten wird die zyklische Ausführung der periodischen Tasks unterbrochen, da mit der Änderung des Arbeitspunktes, also der Perioden der skalierbaren Tasks auch deren Deadlines neu berechnet werden müssen.

Algorithmus 4.1 : Slotwechsel

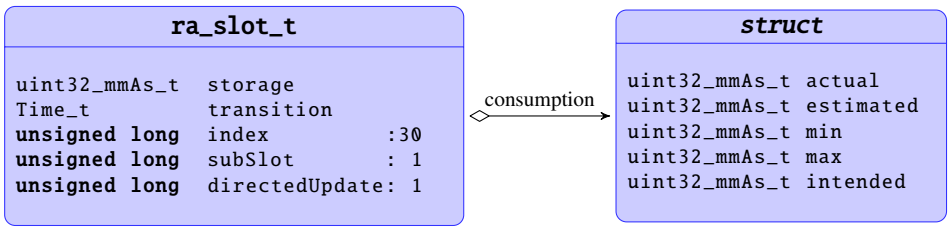
Eingabe : Art des neuen Slots
Wirkung : Anpassung von $\boldsymbol{c}$ und bei Hauptslotgenerierung auch $\widetilde{\boldsymbol{w}}$ und $\widehat{W}$
stoppe periodische Taskausführung;
wenn *Subslot generiert werden soll* **dann**
    verschiebe $\boldsymbol{c}$ gemäß neuem Nutzenvektor $\boldsymbol{b}$; `// Algorithmus 7.5`
sonst
    Ermittle Energieverbrauch W und Speicherfüllstand B ; `// Hardware Layer`
    Verbessere Annahme $\widetilde{\boldsymbol{w}}$ durch Energiebilanzierung ; `// Algorithmus 9.1`
    Bestimme die Energieeinnahmen während des letzten Hauptslots ; `// Algorithmus 8.1`
    Bestimme $\widehat{W}$ für den nächsten Hauptslot ; `// Algorithmus 8.3`
    Verschiebe $\boldsymbol{c}$ um $\widehat{W} - \widetilde{W}$ zur neuen Hyperbene ; `// Algorithmus 7.2`
starte periodische Taskausführung;

4.6 Datenorganisation

Rechenleistung und Datenspeicher sind bei einem Mikrocontroller wie dem SNoW[5]-Sensorknoten stark begrenzt, was eine effiziente Organisation der verwalteten Daten verlangt.

Einen Beitrag zur Reduktion von Rechenaufwand und Programmspeicherbedarf leistet der Verzicht auf Gleitkommaarithmetik. Die Datenstrukturen zur Repräsentation physikalischer Einheiten wie Energie und Spannung werden stattdessen als ganze Zahlen mit ausreichend kleiner Basiseinheit dargestellt. So repräsentiert der Typ `uint16_cV_t` eine Spannung zwischen 0 und 65535 cV = 655,35 V. Wegen der annähernd konstanten Versorgungsspannung des Sensorknotens werden dessen Verbrauchswerte nicht als Energie, sondern als Ladung vom Typ `uint32_mmAs_t`, also mit µAs als Basiseinheit dargestellt.

Um den Speicherbedarf von `ra_slotTransition` zu reduzieren, werden alle Slotparameter in einer globalen Datenstruktur `ra_slot` (Abbildung 4.5) gesammelt und von den einzelnen Modulen direkt gelesen bzw. geändert, anstatt sie per Parameter an entsprechende Unterfunktionen durchreichen zu müssen. Unnötige Mehrfachberechnungen werden damit ebenfalls vermieden.



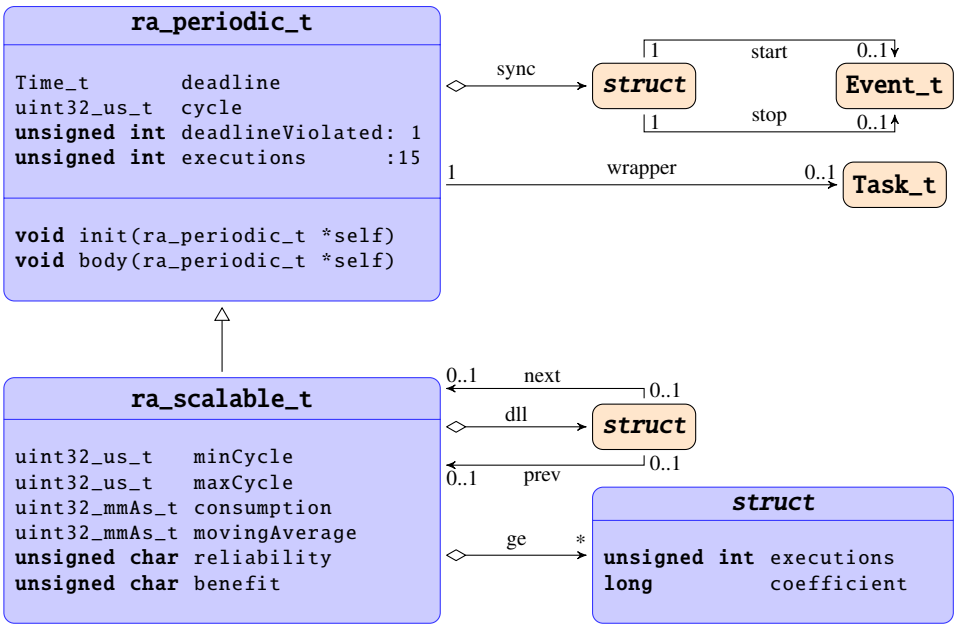
ra_slot_t	
storage	Speicherfüllstand B beim Slotübergang
transition	Zeitpunkt des Slotübergangs
index	Zähler für Hauptslots
subSlot	Flag für Generierung eines Subslots
directedUpdate	Flag für Unterdrückung der Arbeitspunktverschiebung (vgl. Abschnitt 9.4)

ra_slot_t.consumption	
actual	tatsächlicher Verbrauch W während des letzten Hauptslots
estimated	$\widetilde{W} = W(\boldsymbol{c}, \widetilde{\boldsymbol{w}}, T^{\text{main}})$
min	$\widetilde{W}^{\text{min}} = W(\boldsymbol{c}^{\text{max}}, \widetilde{\boldsymbol{w}}, T^{\text{main}})$
max	$\widetilde{W}^{\text{max}} = W(\boldsymbol{c}^{\text{min}}, \widetilde{\boldsymbol{w}}, T^{\text{main}})$
intended	Zielverbrauch $\widehat{W}$ für den nächsten Hauptslot

Abbildung 4.5: global zugreifbare Slotdaten

So müssen $\widetilde{W}^{\min}$ und $\widetilde{W}^{\max}$ vom *Estimator* nur einmalig bei der Änderung von $\widetilde{\boldsymbol{w}}$ neu berechnet werden und können dann jederzeit vom *Regulator* benutzt werden.

Die Breite der Vektoren $\boldsymbol{c}$, $\boldsymbol{c}^{\min}$, $\boldsymbol{c}^{\max}$, $\boldsymbol{e}$ und $\widetilde{\boldsymbol{w}}$ ist von der Anzahl der periodischen bzw. skalierbaren Tasks und somit von der konkreten Anwendung abhängig. Daher sind ihre Komponenten in die Kontrollblöcke der Tasks integriert (Abbildung 4.6), die bei der Deklaration sporadischer bzw. skalierbarer Tasks automatisch angelegt werden (vgl. Abschnitt 6.1). Der *Linker* (`makelib/os.lds`) ordnet diese Kontrollblöcke zu linearen Listen (`ra_periodic` und `ra_scalable`) und berechnet deren Längen (`ra_periodic_count` und `ra_scalable_count`). Somit wird automatisch nur der Speicher alloziert, der für die jeweilige Anwendung benötigt wird.



ra_periodic_t	
`deadline`	Zeitpunkt der nächsten Ausführung
`cycle`	c_i
`deadlineViolated`	Flag zum Anzeigen einer nicht eingehaltenen Deadline
`executions`	e_i
`sync.start/stop`	Events zur Synchronisation mit den Slotübergängen
`wrapper`	Verbindung zum SmartOS *Scheduler*
`init`	einmalig auszuführende Initialisierungsfunktion
`body`	periodisch auszuführende Funktion

ra_scalable_t	
`minCycle`	$c_i^{\min}$
`maxCycle`	$c_i^{\max}$
`consumption`	$\widetilde{w}_i$
`movingAverage`	Mittelwert der Lösungen für w_i aus der Gauß-Elimination
`reliability`	Anzahl der Werte, die zum Mittelwert beitragen
`benefit`	b_i
`dll.prev/next`	Vorgänger/Nachfolgerindex p_i/n_i in doppelt verketteter, nutzensortierte Taskliste
`ge[].executions`	Anzahl der Ausführungen dieses Tasks während der letzten Hauptslots
`ge[].coefficient`	Koeffizienten für Gauß-Elimination

Abbildung 4.6: Taskkontrollblöcke periodischer bzw. skalierbarer Tasks (vgl. Abschnitt 9.3 für die Verwendung der Daten zur Gauß-Elimination)

KAPITEL

5 Hardware Layer

5.1 Lesen der Sensoren

Die wichtigste Schnittstelle zwischen der Software und der Hardware von SNoW^5-RA bildet das serielle Dallas 1-Wire Kommunikationsprotokoll (vgl. Anhang B), über welches der DS2438-Batteriemonitor des Energiespeichers angesprochen wird. In der Initialisierungsphase des *Hardware Layers* werden alle angeschlossenen 1-Wire-Komponenten über den DS2438-Treiber gesucht, deren Identifikationsnummern ausgelesen und auf die Zugehörigkeit zur DS2438-Familie überprüft. Sollten für eine zu SNoW^5-RA parallel laufende Anwendung weitere Monitore dieses Typs verwendet werden, so muss die Zuordnung der Monitore zu den Anwendungen entweder über die global eindeutigen Identifikationsnummern oder eine entsprechende Kennzeichnung in den EEPROMs der Monitore erfolgen.

Zum Auslesen bestimmter Messdaten werden die dafür notwendigen DS2438-Treiber Funktionen gekapselt. Ein Zugriff auf diese Funktionen kann aber erst nach der Zuordnung der 1-Wire-Komponenten sinnvolle Ergebnisse liefern. Die Funktionen zum Auslesen von Spannung, Ladung und Temperatur blockieren deswegen durch das Warten auf ein Event, welches erst am Ende der Initialisierungsphase des *Hardware Layers* gesetzt wird.

Der Batteriemonitor kann zwei verschiedene Spannungen U_{DD} und U_{AD} mit einer Auflösung von 10 mV messen [23]. Da der U_{DD} Pin auch zur Energieversorgung des Monitors genutzt wird und dafür mindestens 2,4 V benötigt werden, muss die Kondensatorspannung U_{cap} am U_{AD} Pin und die Akkuspannung U_{bat} am U_{DD} Pin anliegen, wie in Abbildung 3.4 gezeigt. Der für den *Harvester* notwendige Füllstand des Primärspeichers wird aus der Kondensatorspannung und der als Compilezeitkonstante `RA_HAL_CAP_SIZE` konfigurierten Kapazität C_{cap} des Kondensators als

$$E_{cap}(U_{cap}) = \frac{1}{2} C_{cap} U_{cap}^2 \tag{5.1}$$

bestimmt. Da der Primärspeicher nicht vollständig entladen werden kann, ohne die Versorgung des Sensorknotens zu unterbrechen, repräsentiert (5.1) nicht die effektiv nutzbare Energiemenge nach Ausdruck (3.1). Weil der *Harvester* aber nur relative Änderungen dieser Speicherfüllstandsgröße (B) betrachtet, ist eine Korrekturverschiebung um $\frac{1}{2}C_{cap}{U_{cap}^{min}}^2$ an dieser Stelle nicht nötig. Durch die Ungenauigkeit ΔU_{cap} der Spannungsmessung kann auch der Füllstand des Primärspeichers nicht exakt bestimmt werden. Wegen

$$\begin{aligned} \Delta E_{cap} &= \frac{1}{2}C_{cap}(U_{cap}+\Delta U_{cap})^2 - \frac{1}{2}C_{cap}U_{cap}^2 \\ &= \frac{1}{2}C_{cap}\Delta U_{cap}(2U_{cap}+\Delta U_{cap}) \\ &\overset{2U_{cap} >> \Delta U_{cap}}{\approx} C_{cap}U_{cap}\Delta U_{cap} \end{aligned}$$

kann die Größenordnung der Ungenauigkeit der Speichermessung mit $100\,\mathrm{F}\cdot 1\,\mathrm{V}\cdot 10\,\mathrm{mV} = 1\mathrm{Ws} < 1\,\mathrm{mWh}$ überschlagen werden.

Ein sinnvoller Wert für den Füllstand des Sekundärspeichers lässt sich dagegen kaum aus der gemessenen Zellenspannung ableiten. Zum einen ist die Entladekurve von der Umgebungstemperatur und dem Entladestrom abhängig und zum anderen würde selbst unter der Annahme eines linearen Zusammenhangs

$$E_{bat}(U_{bat}) = E_{bat}(U_{bat}^{min}) + (U_{bat} - U_{bat}^{min}) \cdot \frac{E_{bat}(U_{bat}^{max}) - E_{bat}(U_{bat}^{min})}{U_{bat}^{max} - U_{bat}^{min}} \tag{5.2}$$

zwischen gespeicherter Energie und Zellenspannung die Messungenauigkeit ΔU_{bat} der Speicherspannung zur Ungenauigkeit

$$\Delta E_{bat} = \Delta U_{bat} \cdot \frac{E_{bat}(U_{bat}^{max}) - E_{bat}(U_{bat}^{min})}{U_{bat}^{max} - U_{bat}^{min}} \tag{5.3}$$

führen. Diese kann mit $10\,\mathrm{mV} \cdot \frac{1\,\mathrm{Wh}}{100\,\mathrm{mV}} = 100\,\mathrm{mWh}$ überschlagen werden und fällt damit viel zu groß aus.

Der Batteriemonitor besitzt einen integrierten Temperatursensor mit einer nominalen Auflösung von 0,03125 °C und einem Betriebsbereich von -55 bis +125 °C [23]. Die tatsächliche Messgenauigkeit dieses zur Überhitzungserkennung gedachten Sensors ist aber wesentlich geringer, so dass die Temperaturdaten lediglich als Richtwerte verwendet werden können.

Der für SNoW^5-RA wichtigste Messwert ist der Energieverbrauch des Sensorknotens während eines Hauptslots der Länge T^{main}. Um diesen zu bestimmen, wird der Stromfluss I_C zum Sensorknoten über einen Messwiderstand R_S geleitet (vgl. Abbildung 3.4), und der Spannungsabfall über diesem Widerstand im ICA-Register des Batteriemonitors akkumuliert. Beim Auslesen

dieses ICA-Registers teilt der DS2438-Treiber die integrierten Voltstunden durch den im Nutzerdatenbereich des Monitors konfigurierten Nennwert des Messwiderstands, so dass die während eines Hauptslots verbrauchte Ladung

$$\frac{\Delta\mathrm{ICA}}{R_S} = \frac{1}{R_S}\int_{t}^{t+T^{\mathrm{main}}} U_{R_S}(\tau)\,\mathrm{d}\tau = \frac{1}{R_S}\int_{t}^{t+T^{\mathrm{main}}} I_C(\tau)R_S\,\mathrm{d}\tau = \int_{t}^{t+T^{\mathrm{main}}} I_C(\tau)\,\mathrm{d}\tau \tag{5.4}$$

bestimmt werden kann. Da SNoW^5-RA den Energieverbrauch des Sensorknotens immer genau einmal beim Übergang zwischen zwei Hauptslots abfragt (vgl. Algorithmus 4.1), wird das ICA-Register direkt nach dem Auslesen wieder gelöscht, so dass die gemessenen Werte den Verbrauch im jeweils letzten, gerade abgeschlossenen Hauptslot repräsentieren. Eine Multiplikation der verbrauchten Ladung mit der durch den Spannungswandler TPS61201 stabilisierten Versorgungsspannung liefert die verbrauchte Energie. Diese Skalierung um einen konstanten Faktor ist aber unnötig, solange die als Ladung W (bzw. w_i für einzelne Tasks) verwalteten Verbrauchswerte nicht mit den Füllstandsmessungen B des Energiespeichers verglichen werden.

5.2 Steuern der Aktoren

Eine weitere Aufgabe des *Hardware Layers* ist die Begrenzung der primären und sekundären Speicherspannungen auf einen bestimmten Bereich, also die Gewährleistung von

$$0{,}7\ \mathrm{V} = U_{cap}^{min} \leq U_{cap} \leq U_{cap}^{max} = 2{,}4\ \mathrm{V} \tag{5.5a}$$

$$\text{bzw.} \qquad 3{,}5\ \mathrm{V} = U_{bat}^{min} \leq U_{bat} \leq U_{bat}^{max} = 4{,}0\ \mathrm{V}. \tag{5.5b}$$

Die konkreten Grenzwerte ergeben sich dabei aus den Anforderungen der Speicherverfahren und der verwendeten Spannungswandler (vgl. Kapitel 3).

Wie im letzten Abschnitt beschrieben, können die Speicherspannungen über den Batteriemonitor ermittelt werden. Die Beeinflussung dieser Werte erfolgt durch die Ansteuerung der Aktoren des hierarchischen Enerergiespeichers. Dies ist zum einen der SPDT-Wechselschalter, mit dem zu jedem Zeitpunkt genau einer der beiden Speicher zur Versorgung des Systems ausgewählt wird und zum anderen der MAX1675 *boost converter*, über dessen *Shutdown* Pin der Energietransport vom Primär- zum Sekundärspeicher zugelassen ($\overline{\mathrm{SHDN}} = 1$) bzw. unterbunden ($\overline{\mathrm{SHDN}} = 0$) werden kann. Die Aktoren werden vom *Hardware Layer* über die beiden GPIO-Pins `selectBat` und `chargeBat` gesteuert, wobei die Ansteuerungslogik durch die in Abbildung 5.1a und 5.1b gezeigten, voneinander unabhängigen Moore-Automaten realisiert wird. Da beide Automaten jeweils aus nur zwei Zuständen bestehen, kann das Ausgabebit, also der Wert des angesteuerten Pins, für die Zustandskodierung verwendet werden.

Der `selectBat`-Pin wird bei der Initialisierung des *Hardware Layers* zurückgesetzt, so dass der Sensorknoten zunächst aus dem Primärspeicher versorgt wird. Erst wenn die Primärspannung

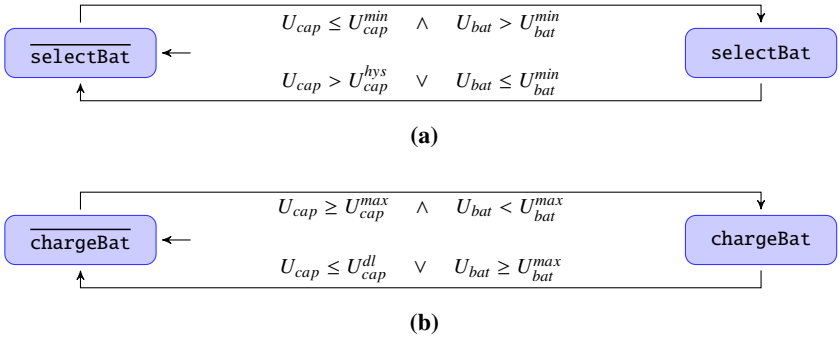


Abbildung 5.1: Moore-Automaten zum Steuern der Aktoren

durch mangelnde Einnahmen aus der regenerativen Energiequelle die Untergrenze erreicht, muss auf den Sekundärbetrieb umgeschaltet werden. Ist zu diesem Zeitpunkt allerdings auch der Sekundärspeicher maximal entladen, so können nicht mehr beide Untergrenzen eingehalten werden. Um eine Tiefenentladung des elektrochemischen Speichers zu verhindern, wird in einem solchen Fall der Primärspeicher weiter belastet. Fällt U_{cap} dann unter den zur Versorgung des Spannungswandlers notwendigen Wert, so kommt es zu dem in dieser Situation unvermeidlichen Systemausfall. Wenn das System hingegen vom Sekundärspeicher versorgt wird und das Solarmodul wieder genug Energie geliefert hat, um U_{cap} über einen Schwellwert U_{cap}^{hys} mit $U_{cap}^{min} < U_{cap}^{hys} < U_{cap}^{max}$ aufzuladen, dann kann wieder zurück auf den Primärbetrieb gewechselt werden. Durch diese Hysterese wird das ständige Wechseln zwischen beiden Betriebsmodi bei niedriger Kondensatorspannungen verhindert. Falls U_{bat} im Sekundärbetrieb seine Untergrenze erreicht, dann muss auch dann wieder zurück auf den Kondensator geschaltet werden, wenn dessen Spannung die Hysteresegrenze noch gar nicht erreicht hat. Auch hierbei steht wieder der Schutz des Sekundärspeichers vor einer Tiefenentladung im Vordergrund.

Der ebenfalls initial zurückgesetzte `chargeBat`-Pin sorgt dafür, dass der Energietransport vom Primär- zum Sekundärspeicher zunächst unterbunden wird. Erst wenn der Primärspeicher vom Solarmodul soweit aufgeladen ist, dass seine Speicherspannung die Obergrenze U_{cap}^{max} erreicht, der Sekundärspeicher aber noch nicht maximal geladen ist, wird der Pin gesetzt und somit der Ladevorgang gestartet. Die klassischen Ladeverfahren für elektrochemische Akkumulatoren unterscheiden sich nach der Methode zum Beenden des Ladevorgangs, also dem Verhindern einer Überladung [12]. Im vorliegenden Fall ist aber die Kapazität der Energiequelle (Primärspeicher) viel kleiner als die der Energiesenke (Sekundärspeicher), so dass der entladene Primärspeicher in der Regel den Energietransport beendet. In Kapitel 3 wurde gezeigt, dass die Energieverluste beim Laden stark ansteigen, sobald die Kondensatorspannung unter $U_{cap}^{dl} = 1{,}5$ V fällt. Der Ladevorgang wird daher beim Erreichen dieser Grenze beendet, wenn der Sekundärspeicher nicht bereits vorher seine maximale Spannung erreicht hat.

Für die Realisierung der Zustandsübergänge der endlichen Automaten ist der Einsatz eines peri-

odischen Tasks prädestiniert, welcher regelmäßig die Speicherspannungen ermittelt und danach die in Abbildung 5.1 angegebenen Fallunterscheidungen ausführt. Es bleibt also noch zu klären, mit welcher Periode dies geschehen sollte, also welche Abtastrate für die Überwachung der beiden Speicher notwendig ist. Die Primärspannung muss sich im Vergleich zur Sekundärspannung wegen der geringeren Speicherkapazität des Kondensators und dem größeren Primärspannnugsbereich schneller ändern. Wegen der endlichen Auflösung $\Delta U_{cap} = 10$ mV bei der Bestimmung der Spannung über den Batteriemonitor und

$$I_{cap}\Delta t = \Delta Q_{cap} = C_{cap}\Delta U_{cap} \tag{5.6a}$$

kann die minimale Abtastperiode

$$\Delta t = \frac{C_{cap}\Delta U_{cap}}{I_{cap}} \tag{5.6b}$$

aus dem maximalen Stromfluss I_{cap} vom oder zum Kondensator abgeschätzt werden, welcher in der Regel beim Energietransport zum Sekundärspeicher entsteht. Ein Überschlag der Größenordnungen zeigt, dass mit einer Abtastrate von $\frac{100\,\mathrm{F}\cdot 10\,\mathrm{mV}}{100\,\mathrm{mA}} = 10\ s$ gerechnet werden muss. Die Rechenleistung des Sensorknotens wird dadurch nicht über die Maßen beansprucht und nach Abschluss des Ladevorgangs kann die Abtastrate verringert werden, da der vom Sensorknoten oder Solarmodul verursachte Kondensatorstrom in der Regel kleiner ausfällt als der Ladestrom.

5.3 Verlängerung der Primärspeicherlaufzeit

Die Steuerung des Energietransports nach Abbildung 5.1a berücksichtigt ausschließlich die aktuellen Speicherspannungen. Wird der Primärspeicher in den Abendstunden in den Sekundärspeicher entladen, so reicht die unterhalb von U_{cap}^{dl} verbleibende Primärenergie in der Regel nicht mehr aus, um das System während der gesamten folgenden Nacht zu versorgen, so dass dann auf den Sekundärspeicher umgeschaltet werden muss.

In solchen Fällen ist es sinnvoller, eine Entladegrenze $U_{cap}^{ddl} \geq U_{cap}^{dl}$ so zu wählen, dass die verbleibende Energie ausreicht, um das System zu versorgen, bis wieder Energie gewonnen werden kann. Zum einen entstehen dadurch weniger Verluste beim Energietransport zwischen den Speichern, da insgesamt weniger Energie transportiert werden muss. Außerdem ist die Zahl der Lade-/Entladezyklen des Sekundärspeichers limitiert, so dass eine Reduktion der Laufzeit im Sekundärbetrieb die Gesamtlaufzeit des Systems verlängern kann.

Für die Berechung von U_{cap}^{ddl} werden die 24 h Vorschauprofile des *Harvesters* und des *Regulators* verwendet, deren Generierung in Kapitel 8 beschrieben ist. Während des Hauptslots j enthalten diese Profile den erwarteten Energiegewinn $\widetilde{H}^{j+k}$ und den geplanten Energieverbrauch $\widehat{W}^{j+k}$ im Hauptslot $j + k$, wobei die Vorschauweite k auf $0 \leq k < d := \frac{24\,\mathrm{h}}{T^{\mathrm{main}}}$ beschränkt ist.

Aus diesen Vorhersagen konstruiert Algorithmus 5.1 die erwartete Entwicklung des Primärspeicherfüllstandes innerhalb der nächsten 24 Stunden. Er startet dabei mit der Restenergie $E_{cap}(U_{cap}^{dl})$, welche nach einer maximalen Entladung des Kondensators verbleiben würde. Danach wird der Einnahmeüberschuss über das komplette Vorschaufenster akkumuliert (Zeile 3) und dabei der Speichertiefststand m ermittelt (Zeile 5). Wird zwischenzeitlich die maximale Kondensatorladung $E_{cap}(U_{cap}^{max})$ erreicht, kann die Akkumulation des Speicherfüllstandes abgebrochen werden, denn hier würde ein neuer Ladevorgang beginnen. Wurde die Untergrenze $E_{cap}(U_{cap}^{min})$ während der Vorschau nicht erreicht, so kann der Kondensator tatsächlich bis U_{cap}^{dl} entladen werden. Ansonsten ist der Speicherfüllstand nach der Entladung um $\Delta := E_{cap}(U_{cap}^{dl}) - m$ gefallen. Die Entladegrenze bzw. die damit verbundene Restenergie $r := E_{cap}(U_{cap}^{ddl})$ muss deshalb so weit angehoben werden, dass $r - \Delta > E_{cap}(U_{cap}^{min})$ gilt. Nun könnte man sehr konservativ vorgehen und hier ein großes Sicherheitspolster veranschlagen, da diese Berechnungen ja nur auf den Energievorhersagen beruhen. Dadurch würde aber die effektiv genutzte Kapazität des Primärspeichers sinken. SNoW^5-RA wählt r in Zeile 7 so, dass die Primärspannung nicht unter U_{cap}^{hys} fällt. Sollte diese Restenergie oberhalb der maximal speicherbaren Energie liegen, so wird die Entladegrenze einen Hystereseabstand (konkret 0,1 V) unterhalb der maximalen Primärspannung angesetzt (Zeile 9), um ein ständiges Flattern des `chargeBat`-Pins zu verhindern. Kann die notwendige Restenergie vom Primärspeicher bereitgestellt werden, so muss noch der entsprechende Spannungswert bestimmt werden (Zeile 11).

Algorithmus 5.1 : Entladegrenze berechnen

Eingabe : Index j des nächsten Hauptslots, Vorschauprofile $\widetilde{H}^j, \ldots \widetilde{H}^{j+d-1}$ und $\widehat{W}^j, \ldots \widehat{W}^{j+d-1}$
Ausgabe : U_{cap}^{ddl} mit $U_{cap}^{hys} \leq U_{cap}^{dl} < U_{cap}^{max}$

1 $s \leftarrow m \leftarrow E_{cap}(U_{cap}^{dl})$;
2 **für alle** $k \in \{0, \ldots, d-1\}$ `// Speicherentwickling abschätzen`
3 $\quad s \leftarrow s + \widetilde{H}^{j+k} - \widehat{W}^{j+k}$;
4 $\quad$ **wenn** $s \geq E_{cap}(U_{cap}^{max})$ **dann** beende Akkumulation;
5 $\quad m \leftarrow \min(m, s)$;
6 **wenn** $m > E_{cap}(U_{cap}^{min})$ **dann** beende mit $U_{cap}^{ddl} \leftarrow U_{cap}^{dl}$; `// maximale Entladung möglich`
7 $r \leftarrow E_{cap}(U_{cap}^{dl}) + E_{cap}(U_{cap}^{hys}) - m$; `// notwendige Restenergie`
8 **wenn** $r \geq E_{cap}(U_{cap}^{max})$ **dann**
9 $\quad U_{cap}^{ddl} \leftarrow U_{cap}^{max}$– Hystereseabstand;
10 **sonst**
11 $\quad U_{cap}^{ddl} \leftarrow \sqrt{\frac{2r}{C_{cap}}}$;

Da die Vorschauprofile nur während eines Hauptslotübergangs geändert werden, muss die Berechnung von U_{cap}^{ddl} auch nur während eines solchen Slotwechsels ausgeführt werden. Da die Vorschau aber erst nach den ersten 24 Slots vollständig inititalisiert ist, wird in dieser Anfangsphase $U_{cap}^{ddl} = U_{cap}^{dl}$ verwendet.

5.4 Warnung vor einem Systemausfall

Wie oben beschrieben, kann die vollständige Entladung des Speichers und der damit verbundene Systemausfall nicht immer verhindert werden. So könnte bspw. die im zeitlichen Mittel gewonnene Energie durch eine Verschmutzung des Solarmoduls dauerhaft sinken und somit nicht mehr zur vollständigen Versorgung des Sensorknotens ausreichen.

Um die Anwendung auf einen solchen bevorstehenden Systemausfall vorzubereiten, wird ein weiterer endlicher Automat mit dem Zustandsbit `energyLow` betrieben (vgl. Abbildung 5.2). Dieser wechselt vom initial unkritischen in den kritischen Zustand, sobald beide Speicherspannungen ihre Untergrenzen erreicht haben. Erst wenn die Kondensatorspannung den Hysteresewert wieder überschritten hat, wird Entwarnung gegeben.

Statt Aktoren anzusteuern, wie die beiden Moore-Automaten aus 5.1, ruft dieser Mealy-Automat bei jeder Zustandsänderung eine von der Anwendung registrierte Funktion (`lowEnergyHandler`) auf, und übergibt den neuen Energiestatus (kritisch oder nicht) als Parameter. Die Anwendung kann durch eine entsprechende Implementierung dieser Funktion entscheiden, was im Falle eines drohenden Systemausfalls zu tun ist. Denkbar wäre etwa das Sichern gepufferter Messdaten durch persistentes Speichern (Flash) oder durch Versenden an eine entsprechende Datensenke. Auch das Abmelden des Sensorknotens aus einem Kommunikationsnetz kann von Bedeutung sein, wenn damit die Freigabe von Netzwerkressourcen (etwa Funkkanäle oder Kommunikationsslots) verbunden ist.

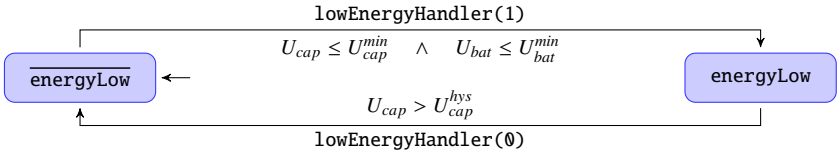


Abbildung 5.2: Mealy-Automat zur Warnung vor einem Systemausfall

KAPITEL

6 Realisierung periodischer Tasks

6.1 Deklaration

Skalierbare periodische Tasks bilden die Grundlage für das *Adaptive Duty Cycling* und somit für die Regulation des Energiebedarfs des Sensorknotens. Die Aufgaben, welche periodisch ausgeführt werden müssen, sind aber anwendungsspezifisch und können daher nicht durch SNoW^5-RA selbst vorgegeben werden. Auf der anderen Seite bedarf es einer Möglichkeit, die Ausführung periodischer Tasks verschiedenster Anwendungen einheitlich zu steuern. Daher werden die Makros aus Programmauszug 6.1 zur Deklaration und Definition (skalierbarer) periodischer Tasks bereitgestellt. Diese Makros bauen auf den Standardmechanismen von SmartOS zur Taskerzeugung auf, was einen Eingriff in das Betriebssystem überflüssig macht.

Die Basis der Deklaration periodischer Tasks bildet das Makro `RA_DECLARE_PERIODIC_TASK` mit den Parametern

- `_name`, `_stacksize`, `_priority`: Parameter für den eigentlichen SmartOS-Tasks,
- `_cycle`: Periode vom Typ `uint32_us_t` und
- `_init`: optionaler Zeiger auf eine Initialisierungsfunktion.

Mit den ersten drei Parametern wird ein normaler SmartOS-Task angelegt, der vom *Scheduler* des Betriebssystems verwaltet wird. Innerhalb dieses Tasks wird die periodische Ausführung der anwendungsspezifischen Aufgabe organisiert, weshalb er als *Wrapper* des periodischen Tasks bezeichnet wird. Die Priorität des *Wrappers* muss kleiner als 255 sein, damit die höchstpriorisierte Slotverwaltung nicht blockiert werden kann.

In Zeile 314 wird eine Funktion (`_name`) für die anwendungsspezifische Aufgabe deklariert. Diese *body*-Funktion erhält als Argument einen Zeiger auf den Kontrollblock des periodischen Tasks (vgl. Abbildung 4.6), um die Informationen über die periodische Ausführung (bspw. die nächste

```
312 #define RA_DECLARE_PERIODIC_TASK(_name, _stacksize, _priority, _cycle, _init)        \
313   OS_DECLARE_TASK(ra_wrapper_##_name, (_stacksize) + 6, _priority);                  \
314   static void _name(ra_periodic_t*);                                                 \
315   static OS_DECLARE_EVENT(ra_stop_##_name);                                          \
316   static OS_DECLARE_EVENT(ra_start_##_name);                                         \
317   ra_periodic_t ra_periodic_##_name __SECTION(".ra_periodic") = {                    \
318     .wrapper     = &__task_ra_wrapper_##_name,                                       \
319     .init        = _init,                                                            \
320     .body        = &_name,                                                           \
321     .cycle       = _cycle,                                                           \
322     .sync        = {.stop  = &ra_stop_##_name,                                       \
323                     .start = &ra_start_##_name}                                      \
324   }

334 #define RA_DECLARE_SCALABLE_TASK(_name, _stacksize, _priority, _benefit, _minCycle,
334                                  _maxCycle, _consumption, _init)                    \
335   RA_DECLARE_PERIODIC_TASK(_name, _stacksize, _priority, _minCycle, _init);          \
336   ra_scalable_t ra_scalable_##_name __SECTION(".ra_scalable") = {                    \
337     .periodic    = &ra_periodic_##_name,                                             \
338     .minCycle    = _minCycle,                                                        \
339     .maxCycle    = _maxCycle,                                                        \
340     .benefit     = _benefit,                                                         \
341     .consumption = _consumption                                                      \
342   }

346 #define RA_TASKENTRY(_name)                                                         \
347   OS_TASKENTRY(ra_wrapper_##_name) {ra_wrapper(&ra_periodic_##_name);}              \
348   static void _name(ra_periodic_t *self)
```

Programmauszug 6.1: include/lib/ra/ra.h

Deadline) auch auf der Anwendungsseite verfügbar zu machen. Dies entspricht dem Konzept von `self`- oder `this`-Zeigern aus objektorientierten Sprachen.

Die beiden *Events* `ra_stop_##_name` und `ra_start_##_name` werden für die Synchronisation der periodischen Taskausführung mit den Slotübergängen von SNoW^5-RA benötigt, die im nächsten Abschnitt näher beschrieben wird. Zuletzt legt `RA_DECLARE_PERIODIC_TASK` noch den Kontrollblock des periodischen Tasks vom Typ `ra_periodic_t` an und initialisiert ihn mit den übergebenen Parametern und den notwendigen Zeigern auf die übrigen Datenstrukturen. Die Kontrollblöcke aller periodischen Tasks werden vom Linkerscript (`makelib/os.lds`) in einem Abschnitt gesammelt, auf den über die Liste `ra_periodic[]` zugegriffen werden kann. Die Länge der Liste wird ebenfalls vom Linkerscript bestimmt und in der Variablen `ra_periodic_count` abgelegt.

Skalierbare Tasks sind Spezialisierungen periodischer Tasks. Das Makro `RA_DECLARE_SCALABLE_TASK` baut daher auf `RA_DECLARE_PERIODIC_TASK` auf und legt die zusätzlichen Informationen

- `_benefit`: relativer Nutzen $b_i \in \{0, \ldots, 255\}$ (vgl. Abschnitt 4.3),
- `_minCycle`, `_maxCycle`: Skalierungsgrenzen $c_i^{\min}, c_i^{\max}$ vom Typ `uint32_us_t` und

- `_consumption`: optionale Verbrauchsannahme $\widetilde{w}_i$ vom Typ `uint32_mmAs_t`

in einem weiteren Kontrollblock vom Typ `ra_scalable_t` ab. Der `periodic`-Zeiger vom skalierbaren zum periodischen Task simuliert dabei die Vererbungsbeziehung zwischen diesen beiden Konstrukten (vgl. Abbildung 4.6). Ist der Verbrauch w_i des Tasks etwa aus einer Testphase bekannt, so kann diese Information bei der Deklaration an SNoW^5-RA übergeben werden. Andernfalls muss der Parameter `_consumption` auf einen Standardwert (bspw. Null) gesetzt werden. Analog zu den Kontrollblöcken der periodischen Tasks werden auch die der skalierbaren Tasks vom Linkerscript in einer Liste `ra_scalable[]` der Länge `ra_scalable_count` gesammelt.

Nach der Deklaration eines (skalierbaren) periodischen Tasks muss dessen eigentliche Aufgabe mit Hilfe des Makros `RA_TASKENTRY` aus Zeile 346 von Programmauszug 6.1 definiert werden. Dieses wiederum erzeugt zunächst den Inhalt des *Wrapper*-Tasks durch den Aufruf der nicht zurückkehrenden Funktion `ra_wrapper` und leitet anschließend die Definition der *body*-Funktion des periodischen Tasks ein. Im Unterschied zur Definition der normalen SmartOS-Tasks darf mit `RA_TASKENTRY` also keine Endlosschleife definiert werden, da dies die Steuerung der Ausführung des Tasks durch SNoW^5-RA verhindern würde.

6.2 Ausführung

Wie im letzten Abschnitt gezeigt, werden periodische Tasks durch die Ausführung der `ra_wrapper` Funktion innerhalb eines normalen SmartOS-Tasks erzeugt. Diese Funktion ist nach Abbildung 6.1 in einer Endlosschleife organisiert und erhält als Argument einen Zeiger auf den Kontrollblock des zu generierenden periodischen Tasks t_i.

Neben der periodischen Ausführung von t_i realisiert der *Wrapper* auch die Synchronisation dieser Ausführung mit der Slotverwaltung, also das Pausieren vor dem ersten Slot und während des Übergangs von einem Slot zum Nächsten. Das Warten der periodischen Tasks auf den ersten Slot gibt SNoW^5-RA die Möglichkeit, seine Submodule zu initialisieren. Besonders die Registrierung des Batteriemonitors kann einige Zeit in Anspruch nehmen, muss aber abgeschlossen sein, bevor die skalierbaren Tasks anfangen, Energie zu verbrauchen, damit die Messung des Gesamtverbrauchs W den Slot auch vollständig erfasst. Die Unterbrechung des *Wrappers* beim Slotwechsel hängt mit einer Neuberechnung von *Deadlines* aufgrund veränderter Perioden zusammen.

Für die Slotsynchronisation werden die Events `sync.start` und `sync.stop` mit jedem Kontrollblock eines periodischen Tasks deklariert (vgl. Abbildung 4.6) und vom `ra_slotTransition` Task verwaltet. Am Ende eines Slots, also zu Beginn des Slotübergangs wird `sync.stop` gesetzt und `sync.start` zurückgesetzt. Analog wird zu Beginn eines Slots, also am Ende des Slotübergangs bzw. der Initialisierungsphase `sync.start` gesetzt und `sync.stop` zurückgesetzt.

Vor der Hauptschleife führt `ra_wrapper` die optionale Initialisierungsfunktion aus, falls diese bei der Deklaration von t_i angegeben wurde. Die Hauptschleife selbst beginnt mit dem Warten auf

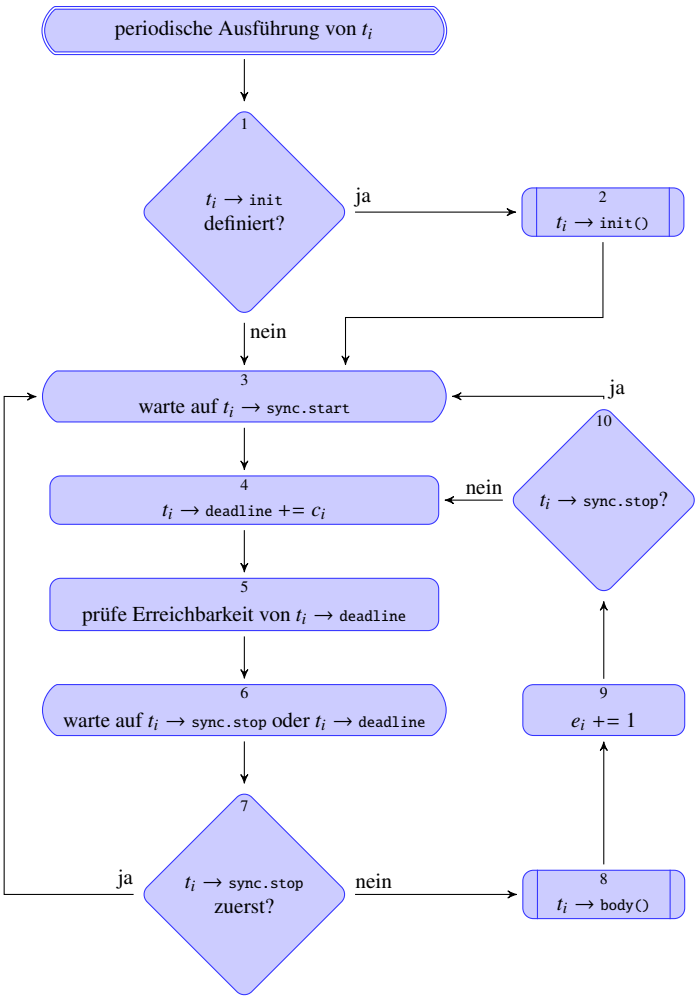


Abbildung 6.1: Programmablaufplan von `ra_wrapper`

den Start eines Slots. Danach wird die mit Null initialisierte *Deadline*, also der Zeitpunkt der nächsten Ausführung der *body*-Funktion des Tasks berechnet, indem die aktuell gesetzte Periodendauer zum Zeitpunkt der letzten Ausführung addiert wird. Danach wird überprüft, ob die neue Deadline überhaupt erreichbar ist, also in der Zukunft liegt. Ist dies nicht der Fall, so wird das `deadlineViolated`-Flag von t_i gesetzt, was der Anwendung bei der nächsten Ausführung der *body*-Funktion die Möglichkeit gibt, adäquat auf die Deadlineverletzung zu reagieren.

Die meiste Zeit verbringt der *Wrapper* im Punkt 6 des Programmablaufplans, dem Warten auf das Erreichen der *Deadline* oder dem Ende des aktuellen Slots. Der Kontrollfluss verzweigt

nach der Art des zuerst eingetretenen Ereignisses. Wird das Slotende vor der *Deadline* erreicht, so begibt sich der *Wrapper* direkt wieder in die Warteposition 3. Andernfalls wird die *body*-Funktion ausgeführt und danach der Ausführungszähler e_i inkrementiert. Um das Zurücksetzen dieses Zählers am Ende eines Hauptslots kümmert sich der `ra_slotTransition` Task. Wurde der alte Slot während der Ausführung der *body*-Funktion beendet und der Nächste nicht bereits gestartet, so erreicht die Kontrolle durch die Verzweigung 10 nun ebenfalls die Warteposition 3. Bei noch oder wieder laufendem Slot wird hingegen direkt mit der periodischen Ausführung, also der Berechnung der nächsten *Deadline* von t_i fortgefahren. Die zusätzliche Verzweigung 10 verhindert das Auslesen der Periode c_i zu einem Zeitpunkt, an dem der *Wrapper* nicht weiß, ob die Verschiebung des Arbeitspunktes innerhalb des Slotübergangs bereits erfolgt ist.

Abbildung 6.2a zeigt die Initialisierungsphase und den Beginn der periodischen Ausführung eines Tasks t_i im ersten Slot S^0. Das Verhältnis zwischen der Periode c_i und der Ausführungsdauer der *body*-Funktion ist dabei aus Gründen der Übersichtlichkeit verkürzt dargestellt. Durch das Warten auf absolute *Deadlines* werden Schwankungen in der Ausführungsdauer der *body*-Funktion kompensiert.

Die Abbildungen 6.2b bis 6.2c zeigen die drei möglichen Kontrollflüsse durch den Programmablaufplan aus Abbildung 6.1 beim Übergang von Slot S^j zu Slot S^{j+1}. Im ersten Fall erfolgt der vollständige Slotwechsel während der Ausführung der *body*-Funktion, so dass im Punkt 4 des Programmablaufplans die *Deadline* bereits um die aktualisierte Periode c_i^{j+1} erhöht wird. Im zweiten Fall hingegen endet die Ausführung der *body*-Funktion während des Slotwechsels. Der *Wrapper* muss daher zunächst die Aktualisierung von c_i^j auf c_i^{j+1} abwarten.

Der letzte Fall zeigt das Erreichen des Slotendes, während der *Wrapper* auf die nächste *Deadline* wartet. Auch hier wird zu Beginn des neuen Slots im Punkt 4 des Programmablaufplans die aktualisierte Periode c_i^{j+1} zur letzten *Deadline* addiert. Die *Deadline*, welche vor dem Slotwechsel gesetzt war, wurde aber nie erreicht, und so müssen zu Beginn jedes Slotwechsels alle noch ausstehenden *Deadlines* durch eine Subtraktion mit c_i^j zurückgesetzt werden. Wird in solchen Fällen allerdings die Periode verkürzt, so kann es passieren, dass die neue *Deadline* noch im alten Slot liegt und somit nicht eingehalten werden kann. Für $c_i^j > n \cdot c_i^{j+1}$ müsste die *body*-Funktion von t_i im neuen Slot schlimmstenfalls sogar $n \in \mathbb{N}$ mal direkt nacheinander ausgeführt werden, bis $t_i \rightarrow$`deadline` wieder die aktuelle Systemzeit überschreitet und damit Punkt 6 des Programmablaufplans zu einem echten Wartevorgang führt. Aus diesem Grund wird am Ende eines Slotwechsels die aktualisierte Periode so oft zur letzten *Deadline* addiert, bis eine weitere Addition (Punkt 4 des Programmablaufplans) einen Zeitpunkt im neuen Slot ergibt.

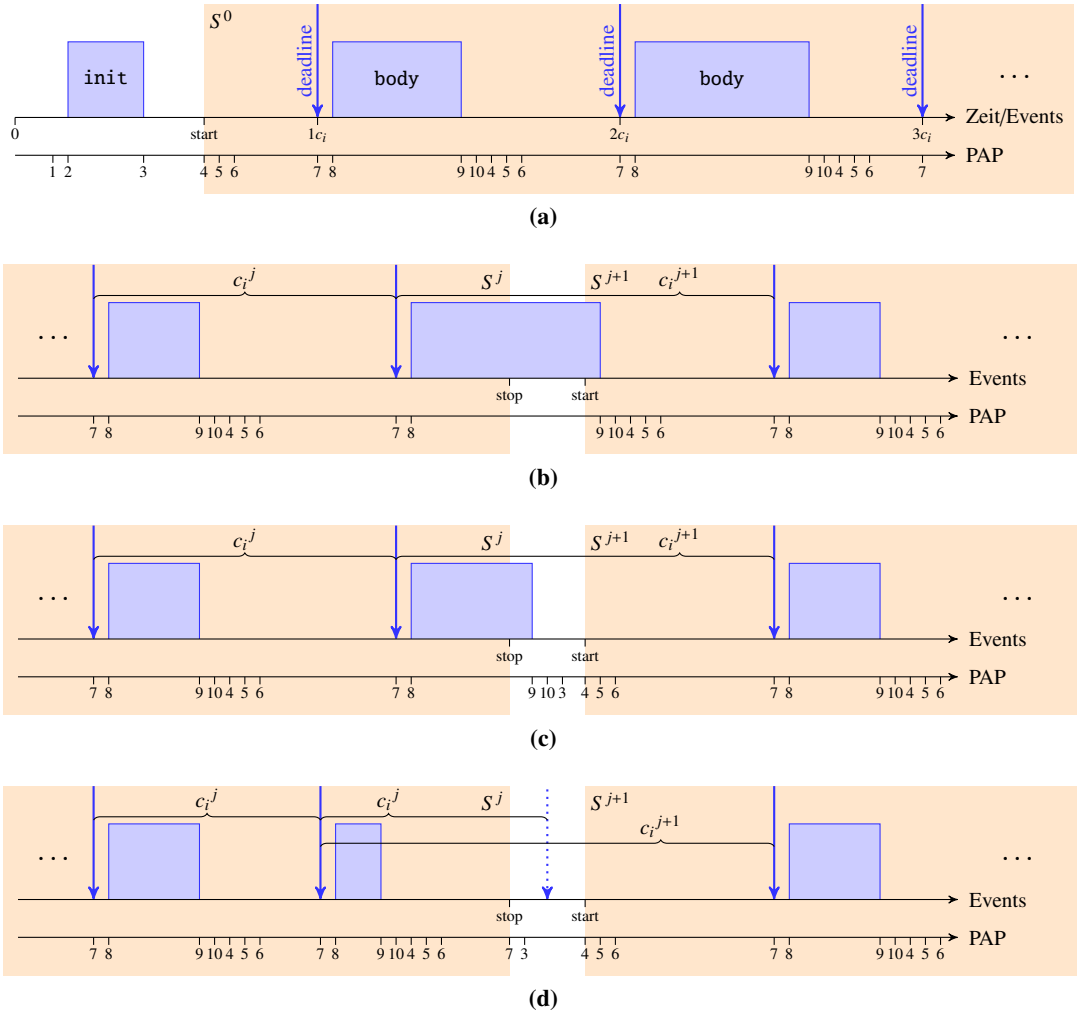


Abbildung 6.2: periodische Ausführung und Slotsynchronisation durch den *Wrapper*

KAPITEL

7 Adaptive Duty Cycling

7.1 Nutzensortierung in doppelt verketteter Liste

Im Abschnitt 4.2 wurde die Grundidee des *Adaptive Duty Cycling* zur Steuerung des Energieverbrauchs des SNoW^5-Sensorknotens durch die Skalierung der Ausführungshäufigkeiten periodischer Tasks vorgestellt. Im Abschnitt 4.3 wurde der relative energetische Nutzen eines skalierbaren Tasks eingeführt und die Mehrdeutigkeit des Arbeitspunktes $\boldsymbol{c}$ bei der Einstellung eines bestimmten Zielverbrauchs durch die Forderung nach einer nutzenoptimierten Energieverteilung aufgelöst. Weiterhin wurde der in [9] beschriebene Greedy-Algorithmus zur Bestimmung dieser nutzenoptimierten Energieverteilung vorgestellt.

In diesem Kapitel soll nun eine effiziente Implementierung dieses Greedy-Algorithmus beschrieben werden. Seine lineare Laufzeitkomplexität $O(s)$ setzt eine Vorsortierung der skalierbaren Tasks nach ihrem relativen Nutzen voraus [9]. Bei einer Vergrößerung des Gesamtverbrauchs erfolgt die Verringerung der Taskperioden in absteigender Nutzenreihenfolge, während beim Verringern des Gesamtverbrauchs eine Periodenvergrößerung in aufsteigender Nutzenreihenfolge notwendig ist. Da also in beide Richtungen traversiert werden muss, bietet sich die Verwaltung der Nutzensortierung als doppelt verkettete Liste an.

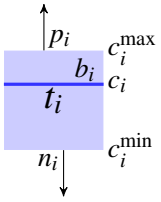


Abbildung 7.1: skalierbarer Task und seine Eigenschaften

Dafür verwaltet jeder skalierbare Task $t_i \in \mathcal{T}_S$ den Index p_i (`ra_scalable[i].dll.prev`) seines direkten Vorgängers $t_{p_i} \in \mathcal{T}_S$ und analog dazu den Index n_i (`ra_scalable[i].dll.next`) seines direkten Nachfolgers $t_{n_i} \in \mathcal{T}_S$ in der nutzensortierten Liste, wie in Abbildung 7.1 dargestellt. Da SmartOS höchstens 255 Tasks verwaltet, genügt ein Byte für die Darstellung eines Indexes, also die Hälfte des Speichers, die ein echter Zeiger auf dem 16 Bit Mikrocontroller benötigen würde. Die Tasks an den beiden Listenenden haben aber keinen Vorgänger bzw. Nachfolger, was durch den reservierten Index `NIL_ID` signalisiert wird. Die Sortierrichtung innerhalb der Liste kann beliebig gewählt werden, solange diese konsistent beibehalten wird. In folgender Definition wird die Sortierung nach absteigendem Nutzen festgelegt.

Definition 7.1 (nutzensortierte Taskliste)**.** *Die skalierbaren Tasks heißen nutzensortiert, wenn die Vorgänger- und Nachfolgerindizes aller $t_i \in \mathcal{T}_S$*

$$p_i \neq \texttt{NIL\_ID} \;\Rightarrow\; b_i \leq b_{p_i} \wedge n_{p_i} = i \qquad \textit{und} \qquad n_i \neq \texttt{NIL\_ID} \;\Rightarrow\; b_i \geq b_{n_i} \wedge p_{n_i} = i$$

erfüllen.

Liegt eine solche Sortierung vor, so können die direkten Nachbarschaftsbeziehungen zwischen den Tasks transitiv erweitert werden.

Definition 7.2 (Vorgänger und Nachfolger eines Tasks)**.** *In einer nutzensortierten Taskliste sind die (transitiven) Vorgänger- und Nachfolgermengen eines Tasks t_i als*

$$\mathcal{P}_i := \begin{cases} \emptyset & \textit{falls } p_i = \texttt{NIL\_ID}, \\ \{t_{p_i}\} \cup \mathcal{P}_{p_i} & \textit{sonst} \end{cases} \quad \textit{und} \quad \mathcal{N}_i := \begin{cases} \emptyset & \textit{falls } n_i = \texttt{NIL\_ID}, \\ \{t_{n_i}\} \cup \mathcal{N}_{n_i} & \textit{sonst} \end{cases}$$

definiert. Weiter sei $\mathcal{P}_{\texttt{NIL\_ID}} := \mathcal{N}_{\texttt{NIL\_ID}} := \emptyset$.

Der auf dem Simplexverfahren beruhende Greedy-Algorithmus aus Abschnitt 4.3 verschiebt den Arbeitspunkt $\boldsymbol{c}$ ausschließlich auf den Kanten des Hyperquaders (4.3). Zu jedem Zeitpunkt gibt es daher genau einen Task t_H, entlang dessen Kante der Arbeitspunkt verschoben wird und dessen Periode somit einen nicht-extremalen Wert annehmen kann. Da jede weitere Verschiebung des Arbeitspunktes immer von diesem Task ausgeht, wird dessen Index H als Zugriffspunkt (*Handle*) der doppelt verketteten Liste verwaltet (`benefitRanking.handle`).

Bei einer nutzenoptimierten Verteilung müssen alle Tasks t_i mit $b_i > b_H$ mit minimaler Periode, also maximaler Frequenz bzw. für $b_i < b_H$ mit maximaler Periode, also minimaler Frequenz betrieben werden. Nach [9] könnten die Perioden aller Tasks mit gleichem Nutzen gleichzeitig bzw. anteilig angepasst werden. Dabei würde man aber den Rand des Hyperquaders (4.3) verlassen und somit die Eindeutigkeit des *Handles* verlieren. Diese Sonderbehandlung wird von der hier vorgestellten Implementierung daher nicht unterstützt. Stattdessen ist die Periode c_i der Tasks $t_i \neq t_H$ mit $b_i = b_H$ ebenfalls minimal oder maximal, je nachdem, ob t_i vor oder nach

t_H in die doppelt verkettete Liste einsortiert wurde. Bei der Vergabe der relativen Nutzen sollte daher auf Eindeutigkeit geachtet werden, damit die Energieverteilung nicht von der konkreten Implementierung der Nutzensortierung abhängig ist. Die folgende Definition konkretisiert obige Überlegungen.

Definition 7.3 (Nutzenoptimierte Energieverteilung)**.** *Eine nutzensortierte Taskliste mit Handle H ist eine nutzenoptimierte Energieverteilung, wenn alle Vorgänger von t_H mit minimaler und alle Nachfolger von t_H mit maximaler Periode betrieben werden. Für alle $t_i \in \mathcal{T}_S$ muss also*

$$t_i \in \mathcal{P}_H \;\Rightarrow\; c_i = c_i^{\min} \qquad \textit{und} \qquad t_i \in \mathcal{N}_H \;\Rightarrow\; c_i = c_i^{\max}$$

gelten.

Abbildung 7.2 veranschaulicht die verschiedenen Ausprägungen nutzenoptimierter Energieverteilungen für $s = 4$ skalierbare Tasks mit $b_1 \geq b_2 \geq b_3 \geq b_4$. 7.2a stellt dabei den Fall minimalen Energieverbrauchs und damit die untere Skalierbarkeitsgrenze bzw. den extremalen Arbeitspunkt $\boldsymbol{c}^{\max}$ des Systems dar. Gäbe es einen Vorgänger $t_i \in \mathcal{P}_H$, so müsste dessen Periode c_i nach Definition 7.3 minimal sein. Da es keinen solchen Task gibt, ist $\mathcal{P}_H = \emptyset$ und somit $p_H =$ `NIL_ID` nach Definition 7.2. Das *Handle* muss also auf den ersten Task der nutzensortierten Liste zeigen. Analog dazu zeigt Abbildung 7.2d den Zustand bei maximalem Energieverbrauch. Das *Handle* ist dabei eindeutig auf den letzten Task der sortierten Liste festgelegt. Die beiden übrigen Abbildungen entsprechen je einem Arbeitspunkt innerhalb des skalierbaren Bereichs.

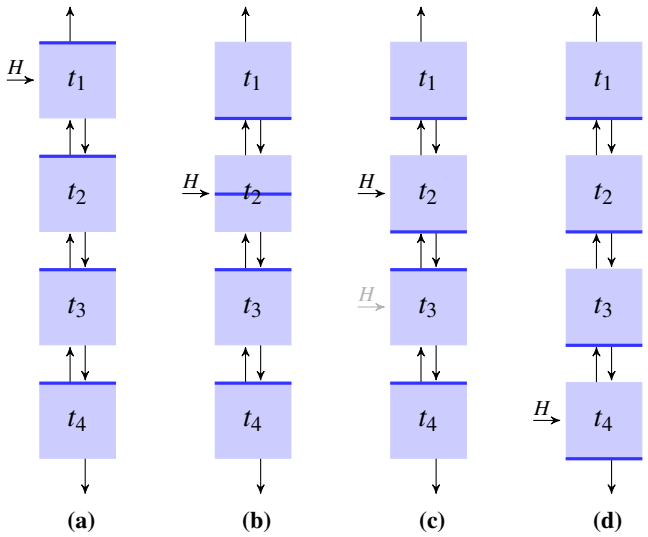


Abbildung 7.2: nutzenoptimierte Energieverteilungen für $b_1 \geq b_2 \geq b_3 \geq b_4$

Gibt es einen Task mit einer nicht-extremalen Periode wie in 7.2b, so kann dieser nicht Vorgänger oder Nachfolger von t_H, sondern nur das *Handle* selbst sein. Erreicht aber auch das *Handle* eine extremale Periode wie in 7.2c, so hat der Greedy-Algorithmus den Arbeitspunkt $\boldsymbol{c}$ gerade in einen Eckpunkt des Hyperquaders (4.3) verschoben. In diesem Fall kann jeder der beiden benachbarten Tasks der nutzensortierten Liste, welche den Übergang von maximaler zu minimaler Periode markieren, als *Handle* verwendet werden.

Die Sortierung der skalierbaren Task aus $\mathcal{T}_S$ nach ihrem energetischen Nutzen erfolgt durch Einfügen von $t_i \in \mathcal{T}_S$ in die sortierte Liste der Tasks aus $\mathcal{T}_S \setminus t_i$. Dieses Sortieren durch Einfügen hat bekanntermaßen eine quadratische Laufzeitkomplexität $O(s^2)$. Der Geschwindigkeitsnachteil gegenüber der Klasse der subquadratischen Sortieralgorithmen (wie etwa *Quicksort*) ist mit der geringen Anzahl der zu sortierenden Elemente beschränkt. Außerdem kann durch Extraktion und Wiedereinfügen eine Umsortierung der Liste bei der Nutzenänderung eines einzelnen Tasks effektiv realisiert werden (vgl. Abschnitt 7.3).

Das rekursive Zurückführen der Sortieraufgabe auf immer kleinere Listen kann bei der leeren Liste abgebrochen werden, da diese bereits sortiert ist. Sie wird durch $H = \texttt{NIL\_ID}$ repräsentiert und erfüllt somit auch Definition 7.3. Um einen konstanten Speicherbedarf zu gewährleisten, erfolgt die Sortierung aber nicht rekursiv sondern iterativ, sie beginnt also mit der leeren Liste und fügt sukzessive alle skalierbaren Tasks mit Hilfe von Algorithmus 7.1 ein. Dieser setzt neben der Erfüllung der Definitionen 7.1 und 7.3 vor dem Einfügen die Existenz höchstens eines Tasks mit nicht-extremaler Periode voraus. Durch die Initialisierung aller Tasks mit minimaler Periode (vgl. Abschnitt 6.1) und der Funktionsweise des Greedy-Algorithmus (vgl. Abschnitt 7.2) ist diese Voraussetzung stets erfüllt.

Der einfachste Fall ist das Einfügen von t_i in die leere Liste (Zeile 1) durch Setzen des *Handles* auf den Index des einzufügenden Tasks. Mit $n_H = p_H = \texttt{NIL\_ID}$ ist $\mathcal{N}_H = \mathcal{P}_H = \emptyset$, wodurch die Forderungen der Definitionen 7.1 und 7.3 erfüllt werden, da die Prämissen aller Implikationen für keinen Task erfüllt werden.

Ist die Liste, in welche t_i eingefügt werden soll, hingegen nicht leer (Zeile 4), so muss zunächst die passende Einfügeposition k gesucht werden, wobei die Suche immer beim *Handle* beginnt (Zeile 5). Des Weiteren muss festgelegt werden, ob t_i vor ($\delta = 0$) oder nach ($\delta = 1$) t_k eingefügt werden soll. Stimmt der Nutzen von t_i mit dem des *Handles* überein (Zeile 6), so ist mit $k = H$ bereits eine passende Einfügeposition gefunden. Für eine korrekte Nutzensortierung wären auch beide Einfügerichtungen möglich, so dass die restlichen Fallunterscheidungen (Zeilen 7-11) nur noch der Gewährleistung der nutzenoptimierten Energieverteilung dienen. Ist b_i extremal (Zeile 7 bzw. 8), so ergibt sich die Einfügerichtung direkt aus den Forderungen von Definition 7.3. Andernfalls ist t_i der einzige Task mit nicht-extremaler Periode und muss daher zum *Handle* werden (Zeile 10). Die Periode b_k des ursprünglichen *Handles* muss nun extremal sein und entscheidet über die Einfügerichtung (Zeile 11). Dabei ist zu beachten, dass bei minimalem b_k das neue *Handle* nach t_k eingefügt werden muss, damit $t_k \in \mathcal{P}_i = \mathcal{P}_H$ und somit Definition 7.3 gilt.

Algorithmus 7.1 : Einfügen von t_i in die doppelt verkettete nutzensortierte Liste

Eingabe : einzufügender Task t_i

Wirkung : Erweiterung der nutzensortierten Liste unter Einhaltung der Definitionen 7.1 und 7.3, ggf. Anpassung von *Handle* H oder der *Violator*-Daten (V, δ_V)

```
1  wenn H = NIL_ID dann                                   // in leere Liste einfügen
2      H ← i;
3      n_H ← p_H ← NIL_ID;
4  sonst
5      k ← H ;                                            // Einfügeposition k
6      wenn b_i = b_H dann                                // am Handle einfügen
7          wenn c_i = c_i^max dann δ ← 1 ;                // Einfügerichtung δ
8          sonst wenn c_i = c_i^min dann δ ← 0;
9          sonst
10             H ← i ;                                    // t_i wird Handle
11             δ ← c_k = c_k^min ? 1 : 0;
12     sonst
13         δ ← b_i < b_H ? 1 : 0;
14         j ← ▽_j^δ;
15         solange j ≠ NIL_ID und b_i ◇^δ b_j              // Einfügeposition suchen
16             (k, j) ← (j, ▽_j^δ);
17         wenn c_i ≠ c_i^δ dann                          // Gültigkeit prüfen
18             wenn k = H und c_H = c_H^(1−δ) dann H ← i sonst (V, δ_V) ← (i, δ)
19     (▽_i^δ, ▽_i^(1−δ), ▽_k^δ) ← (▽_k^δ, k, i) ;        // Umhängen der Verkettung
20     wenn (j ← ▽_i^δ) ≠ NIL_ID dann ▽_j^(1−δ) ← i;
```

Unterscheidet sich b_i vom Nutzen des *Handles* (Zeile 12), so muss die Einfügeposition durch eine lineare Suche bestimmt werden. Durch die Vorsortierung der Liste ohne t_i kann man die Suchrichtung δ durch einen Vergleich von b_i und b_H bestimmen (Zeile 13). Da sich die Such- und Einfügevorgänge in beide Richtungen nur durch wenige Details unterscheiden, werden parametrisierte Hilfsoperatoren für die Verkettungsindizes, den Nutzenvergleich und die Periodenbegrenzung verwendet.

Definition 7.4 (Richtungsparametrierte Operatoren)**.**

$$\triangledown^\delta := \begin{cases} n & \text{, für } \delta = 1 \\ p & \text{, für } \delta = 0 \end{cases} \qquad \diamond^\delta := \begin{cases} < & \text{, für } \delta = 1 \\ > & \text{, für } \delta = 0 \end{cases} \qquad c^\delta := \begin{cases} c^{\max} & \text{, für } \delta = 1 \\ c^{\min} & \text{, für } \delta = 0 \end{cases}$$

Zum besseren Verständnis wird das weitere Vorgehen nur für $\delta = 1$, also die Vorwärtssuche in

Richtung absteigenden Nutzens dargestellt.

Zur eigentlichen Suche der Einfügeposition wird ein weiterer Index j verwendet, welcher der Einfügeposition k immer einen Schritt in der Suchrichtung vorauseilt (Zeile 14-16). Endet die Suchschleife mit $j = \texttt{NIL\_ID}$, so zeigt k auf das Listenende. Dabei ist $b_i < b_k$, denn für $b_i \geq b_k$ wäre die Schleife wegen der zweiten Abbruchbedingung bereits im vorherigen Durchlauf terminiert. Der Nutzen von t_i ist somit kleiner als der aller anderen Tasks der Liste, was das Anhängen von t_i ans Listenende rechtfertigt. Endet die Suchschleife dagegen mit $j \neq \texttt{NIL\_ID}$, dann muss $b_i \geq b_j$ erfüllt sein. Es muss aber weiterhin $b_i < b_k$ gelten, da die Schleife sonst früher terminiert wäre. Mit $b_k > b_i \geq b_j$ ist demnach die korrekte Nutzensortierung durch das Einfügen von t_i zwischen t_k und t_j, also in Suchrichtung nach t_k gesichert.

Es fehlt nun noch die Überprüfung der Eigenschaften für die nutzenoptimierte Energieverteilung, welche beim Einfügen von t_i nach dem *Handle* ein maximales c_i erwarten. Ist dies nicht gegeben (Zeile 17), so können die Forderungen von Definition 7.3 trotzdem noch erfüllt werden, wenn t_i direkt nach dem *Handle* eingefügt werden soll und c_H selbst minimal ist (vgl. Abbildung 7.2c mit $H = k = 2$ und $j = 3$). Dann sind nämlich die Perioden aller Vorgänger von t_i minimal und eine Verschiebung des *Handles* auf t_i rettet die nutzenoptimierte Verteilung. Für $c_i \neq c_i^{\max} \wedge (c_H \neq c_H^{\min} \vee k \neq H)$ hingegen ist t_i ein Nachfolgertask von t_H mit nicht-maximaler Periode und t_H bzw. t_k Vorgängertasks von t_i mit nicht minimaler Periode. Daraus resultiert ein Konflikt mit Definition 7.3, der nicht durch eine Verschiebung des *Handles* aufgelöst werden kann. Stattdessen ist eine Energieumverteilung, also eine Verschiebung des Arbeitspunktes und damit das Auslösen eines Subslots notwendig, was durch die Markierung von t_i als *Violator V* (`benefitRanking.violator`) der nutzenoptimierten Energieverteilung signalisiert wird. Für die Behebung dieser Abweichung von der optimalen Verteilung (vgl. Abschnitt 7.3) wird außerdem die

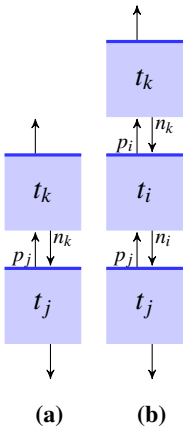


Abbildung 7.3: Situation (a) vor und (b) nach Einfügen von t_i zwischen t_k und t_j

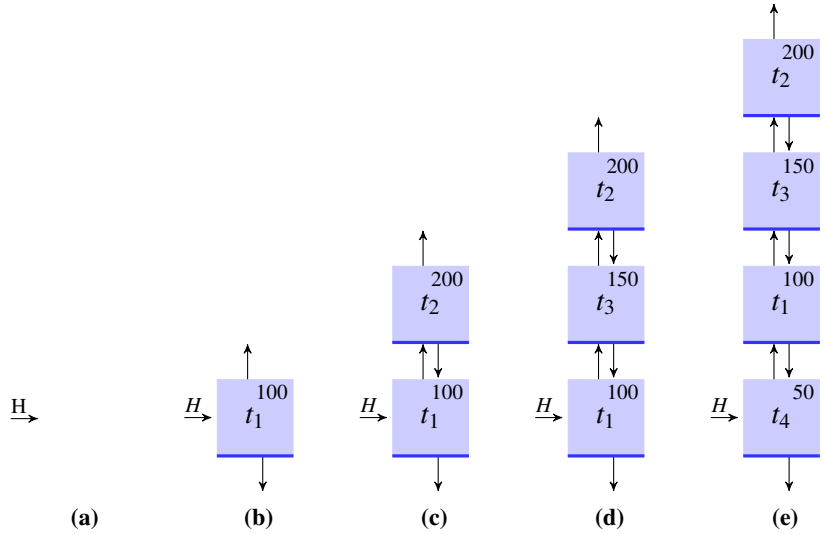


Abbildung 7.4: Zwischenschritte der initialen Sortierung

relative Richtung δ_V (`benefitRanking.violatorDir`) vom *Handle* zum *Violator* benötigt und daher in Zeile 18 gleich mit erfasst.

Nachdem die Einfügeposition und -richtung bestimmt wurden, muss das eigentliche Einfügen von t_i in die doppelt verkettete Liste durch Umhängen von Zeigern bzw. Ändern von Indizes wie in Abbildung 7.3 realisiert werden (Zeile 19-20). Das Setzen des letzten Zeigers von t_j nach t_i ist dabei nur notwendig, wenn t_i nicht ans Ende der Liste angehängt wird.

Bei der Initialisierung des Moduls für das *Adaptive Duty Cycling* wird die nutzenoptimierte Energieverteilung durch sukzessives Einfügen der skalierbaren Tasks in die anfangs leere Liste aufgebaut. Abbildung 7.4 veranschaulicht diesen Vorgang an einem Beispiel für $s = 4$. Da alle skalierbaren Tasks mit minimaler Periode initialisiert werden, kann eine Verletzung von Definition 7.3 während des Einfügens stets durch eine Verschiebung des *Handles* aufgehoben werden. Das Überprüfen der *Violator*-Markierung ist während der Initialisierung daher nicht notwendig.

7.2 Adaption des Arbeitspunktes

Die Hauptaufgabe des *Adaptive Duty Cycling* ist es, den angenommenen Energieverbrauch $\widetilde{W}$ des Sensorknotens bei jedem Hauptslotwechel an die Zielvorgaben $\widehat{W}$ des *Regulators* anzupassen. Dazu muss die Energiedifferenz $\Delta\widetilde{W} = \widehat{W} - \widetilde{W}$ durch eine Verschiebung des Arbeitspunktes $\boldsymbol{c}$ unter Berücksichtigung der Forderungen nach nutzenoptimierter Energieverteilung ausgeglichen werden.

Bei der Verschiebung des Arbeitspunktes entlang einer Kante des Hyperquaders (4.3) von $\boldsymbol{c}$ nach $\boldsymbol{c}'$ wird ausschließlich die Periode c_H des *Handles* zu c'_H verändert, während alle anderen Perioden unverändert bleiben, also $c_i = c'_i \; \forall \; i \neq H$. Eine solche Verschiebung führt nach (4.4) zu einer Veränderung des Gesamtverbrauchs von

$$\Delta\widetilde{W}_H = \widetilde{W}(\boldsymbol{c}', \widetilde{\boldsymbol{w}}, T^{\text{main}}) - \widetilde{W}(\boldsymbol{c}, \widetilde{\boldsymbol{w}}, T^{\text{main}}) = T^{\text{main}} \sum_{i=0}^{s} \left(\frac{\widetilde{w}_i}{c'_i} - \frac{\widetilde{w}_i}{c_i} \right) = \widetilde{w}_H T^{\text{main}} \left(\frac{1}{c'_H} - \frac{1}{c_H} \right). \quad (7.1a)$$

Die Auflösung von (7.1a) nach

$$c'_H = \frac{c_H \widetilde{w}_H T^{\text{main}}}{\widetilde{w}_H T^{\text{main}} + c_H \Delta\widetilde{W}_H} \quad (7.1b)$$

beschreibt die notwendige Periode des *Handles* zur Kompensation der Energiedifferenz $\Delta\widetilde{W}_H$, welche aber nicht zwangsläufig im skalierbaren Bereich des *Handle* liegen muss.

Der Greedy-Algorithmus 7.2 zur Kompensation von $\Delta\widetilde{W}$ beruht auf diesen beiden Ausdrücken und der Annahme, dass zum Zeitpunkt seines Aufrufs eine nutzenoptimierte Energieverteilung vorliegt. Wie bereits Algorithmus 7.1 nutzt auch dieser die mit der Laufrichtung δ parametrisierten Operatoren aus Definition 7.4, um die Unterscheidung zwischen einer Energievergrößerung und einer Energieverringerung auf den initialen Test des Vorzeichens von $\Delta\widetilde{W}$ zu reduzieren.

Algorithmus 7.2 : Adaption des Arbeitspunktes

Eingabe : auszugleichende Energiedifferenz $\Delta\widetilde{W}$
Wirkung : Verschiebung der Hyperebene (4.4) um $\Delta\widetilde{W}$ durch Anpassung des Arbeitspunktes unter Erhalt der Eigenschaften der nutzenoptimierten Energieverteilung

1 $\delta \leftarrow \Delta\widetilde{W} > 0 \; ? \; 1 : 0;$
2 **wiederhole**
3 $\quad \Delta\widetilde{W}_H \leftarrow \widetilde{w}_H T^{\text{main}} \left(\frac{1}{c_H^{1-\delta}} - \frac{1}{c_H} \right);$ // (7.1a)
4 $\quad$ **wenn** $|\Delta\widetilde{W}| < |\Delta\widetilde{W}_H|$ **dann** $\Delta\widetilde{W}_H \leftarrow \Delta\widetilde{W};$
5 $\quad c_H \leftarrow \frac{c_H \widetilde{w}_H T^{\text{main}}}{\widetilde{w}_H T^{\text{main}} + c_H \Delta\widetilde{W}_H};$ // (7.1b)
6 $\quad \Delta\widetilde{W} \leftarrow \Delta\widetilde{W} - \Delta\widetilde{W}_H;$
7 $\quad$ **wenn** $\nabla_H^\delta =$ `NIL_ID` *oder* $\Delta\widetilde{W} = 0$ **dann** *fertig*;
8 $\quad H \leftarrow \nabla_H^\delta;$
9 **bis** *fertig* ;

Jeder Schleifendurchlauf des Algorithmus realisiert die Verschiebung des Arbeitspunkts entlang der Hyperquaderkante des aktuellen *Handles*. Um zu ermitteln, wie weit c_H verschoben werden muss, wird zunächst die Energieänderung $\Delta\widetilde{W}_H$, welche aus einer maximalen Verschiebung resultieren würde, mittels (7.1a) bestimmt. Bei einer Energievergrößerung ($\delta = 1$) entspricht dies der Verschiebung von c_H bis $c_H^{\min}$, während bei einer Energievergrößerung nach $c_H^{\max}$ verschoben werden muss. Ist die zu kompensierende Energiedifferenz $\Delta\widetilde{W}$ kleiner als die mit dem *Handle* mögliche Änderung, trifft man also beim Verschieben entlang der aktuellen Kante auf die Zielhyperebene (vgl. Abbildung 4.3), so begrenzt $\Delta\widetilde{W}$ die Energieänderung $\Delta\widetilde{W}_H$. Mit Ausdruck (7.1b) wird dann die neue Periode c_H berechnet. Hat die Verschiebung ausgereicht, um $\Delta\widetilde{W}$ vollständig zu kompensieren oder ist das Ende der nutzensortierten Liste und damit eine Skalierbarkeitsgrenze des Systems erreicht, so endet der Greedy-Algorithmus. Andernfalls wird das *Handle* weiter entlang der Nutzensortierung verschoben, also die nächste Kante des Hyperquaders ausgewählt.

Sieht man vom einmaligen quadratischen Aufwand für die Nutzensortierung ab, die Algorithmus 7.2 voraussetzt, so ist dessen *Worst-Case* Laufzeit linear in der Zahl der skalierbaren Tasks, denn jede Hyperquaterkante muss höchstens einmal betrachtet werden. Wie im Kapitel 8 aber deutlich wird, sind zwischen den meisten Hauptslots nur kleine Unterschiede im angestrebten Gesamtverbrauch auszugleichen, wofür die Verschiebung des Arbeitspunktes entlang weniger Kanten ausreicht.

Es bleibt noch festzustellen, dass der Greedy-Algorithmus die Eigenschaften der nutzenoptimierten Energieverteilung nach den Definitionen 7.1 und 7.3 erhält. Da keinerlei Änderungen an den Zeigern der doppelt verketteten Liste vorgenommen werden, bleibt die Nutzensortierung bestehen. Bei jedem Schleifendurchlauf, der zur Änderung des *Handles* (Zeile 8) führt, muss vor Zeile 7 $\Delta\widetilde{W} \neq 0$ gelten. Dies setzt $\Delta\widetilde{W} \neq \Delta\widetilde{W}_H$ in Zeile 6 voraus, weshalb die bedingte Zuweisung aus Zeile 4 nicht ausgeführt wurde. Damit hat $\Delta\widetilde{W}_H$ in Zeile 5 noch den Wert, der in Zeile 3 zugewiesen wurde, so dass mit

$$c_H \leftarrow \frac{c_H \widetilde{w}_H T^{\text{main}}}{\widetilde{w}_H T^{\text{main}} + c_H \Delta\widetilde{W}_H} = \frac{c_H \widetilde{w}_H T^{\text{main}}}{\widetilde{w}_H T^{\text{main}} + c_H \widetilde{w}_H T^{\text{main}} \left(\frac{1}{c_H^{1-\delta}} - \frac{1}{c_H}\right)} = \frac{c_H}{1 + \frac{c_H}{c_H^{1-\delta}} - \frac{c_H}{c_H}} = c_H^{1-\delta} \quad (7.2)$$

die extremale Periode berechnet wird. Für $\delta = 1$ ist $c_H^{1-\delta} = c_H^{\min}$ und $\triangledown_H^{\delta} = n_H$, so dass die Perioden aller Vorgänger des neuen *Handles* nach der Verschiebung von Zeile 8 minimal sind. Für $\delta = 0$ ist analog $c_H^{1-\delta} = c_H^{\max}$ und $\triangledown_H^{\delta} = p_H$ und somit sind die Perioden aller Nachfolger des neuen *Handles* maximal. Beim letzten Schleifendurchlauf kommt es dagegen zu keiner Verschiebung des *Handles* mehr, wodurch die nach dem Induktionsprinzip bis dahin gültigen Eigenschaften von Definition 7.3 nicht mehr verletzt werden können.

Mit Abbildung 7.5 soll noch einmal die Funktionsweise des Greedy-Algorithmus 7.2 bei einer Verringerung des Gesamtverbrauchs ($\Delta\widetilde{W} < 0$) veranschaulicht werden. In 7.5a wird festgestellt, dass $|\Delta\widetilde{W}| \geq |\Delta\widetilde{W}_1|$ gilt und deshalb c_1 maximiert und das *Handle* auf $p_1 = 3$ aktualisiert

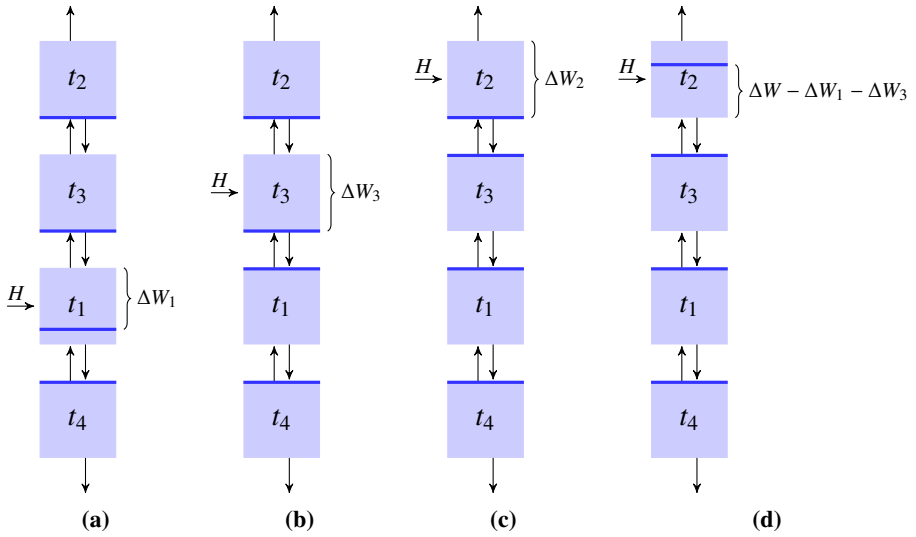


Abbildung 7.5: Zwischenschritte bei der Adaption des Arbeitspunktes

werden muss. Analoges gilt wegen $|\Delta\widetilde{W} - \Delta\widetilde{W}_1| \geq |\Delta\widetilde{W}_3|$ auch für 7.5b. Erst beim Task t_2 in 7.5d kann die Energiedifferenz wegen $|\Delta\widetilde{W} - \Delta\widetilde{W}_1 - \Delta\widetilde{W}_3| < |\Delta\widetilde{W}_2|$ vollständig ausgeglichen werden.

7.3 Nutzenänderung zur Laufzeit

In Abschnitt 4.3 wurde die Notwendigkeit der dynamischen Änderung des energetischen Nutzens skalierbarer Tasks motiviert. Dafür wird eine Möglichkeit zur Umsortierung der doppelt verketteten Liste sowie die Behandlung der eventuell daraus resultierenden Verletzungen der nutzenoptimierten Energieverteilung benötigt.

Die Umsortierung der Taskliste nach der Änderung des Nutzens b_i eines einzigen Tasks t_i ist durch Extraktion und Wiedereinfügen von t_i effektiv realisierbar, da die Liste ohne t_i korrekt sortiert ist und somit die Voraussetzungen für das Sortieren durch Einfügen erfüllt. Durch die Ver-

Algorithmus 7.3 : Extraktion eines skalierbaren Tasks t_i aus der doppelt verketteten Liste

Eingabe : zu extrahierender Task t_i
Wirkung : Reduktion der Liste um t_i und ggf. Anpassung des *Handles*
wenn $p_i \neq$ `NIL_ID` **dann** $n_{p_i} \leftarrow n_i$;
wenn $n_i \neq$ `NIL_ID` **dann** $p_{n_i} \leftarrow p_i$;
wenn $i = H$ **dann** $H \leftarrow (p_i \neq \texttt{NIL\_ID})\ ?\ p_i : n_i$;

waltung der Sortierung als doppelt verkettete Liste ist die Extraktion eines Tasks nach Algorithmus 7.3 durch Umhängen weniger Zeiger in konstanter Zeit realisierbar. Soll gerade das *Handle* extrahiert werden, so muss ein direkter Nachbar von t_i zum neuen *Handle* erklärt werden. Die Eigenschaften von Definition 7.3 bleiben dabei erhalten, da die Vorgänger- bzw. Nachfolgermengen des *Handles* bei dieser Verschiebung höchstens verkleinert werden.

Bei der Umsortierung einer nicht-extremalen Energieverteilung $\boldsymbol{c} \notin \{\boldsymbol{c}^{\min}, \boldsymbol{c}^{\max}\}$ können in bestimmten Fällen die Forderungen von Definition 7.3 nicht aufrecht erhalten werden. Wird b_i beispielsweise vom kleinsten zum größten Wert geändert, dann wird t_i mit $c_i \neq c_i^{\min}$ vor dem *Handle* wieder einsortiert. In Zeile 18 des Einfügealgorithmus 7.1 wird dann der Task, welcher zur Verletzung der optimalen Verteilung geführt hat, als *Violator V* markiert. Mit der Wiederherstellung einer optimalen Energieverteilung durch eine Verschiebung des Arbeitspunktes, also eine Veränderung der Taskperioden, wird auch die Neuberechnung der *Deadlines* mancher Tasks notwendig (vgl. Abschnitt 6.2). Diese Korrektur des Arbeitspunktes durch Algorithmus 7.5 wird daher durch das Auslösen eines Subslots auf den `ra_slotTransition` Task übertragen und damit der gesamte Stackbedarf der Slotverwaltung auf diesen einen Task konzentriert. Algorithmus 7.4 fasst die notwendigen Schritte zur Änderung des Nutzens eines skalierbaren Tasks zusammen. Diese Änderung muss unterbunden werden, wenn eine noch nicht behobene Verletzung der nutzenoptimierten Energieverteilung vorliegt oder der *Estimator* mit dem gesteuerten Lernen (vgl. Abschnitt 9.4) die Kontrolle über den Arbeitspunkt des Systems übernommen hat.

Algorithmus 7.4 : Ändern des Nutzens b_i eines skalierbaren Tasks t_i

Eingabe : zu ändernder Task t_i, neuer Nutzen b
Wirkung : Umsortierung der doppelt verketteten Liste und ggf. Auslösen eines Subslots bei resultierender Verletzung der nutzenoptimierten Energieverteilung

wenn $b_i \neq b$ *und* $V =$ `NIL_ID` *und gesteuertes Lernen nicht aktiv* **dann**
- Extrahiere t_i ; `// Algorithmus 7.3`
- $b_i \leftarrow b$;
- Füge t_i ein ; `// Algorithmus 7.1`
- **wenn** $V \neq$ `NIL_ID` **dann** löse Subslot aus;

Es bleibt nun noch die Restauration der nutzenoptimierten Energieverteilung nach Algorithmus 7.5 zu beschreiben. Dabei wird die Periode des *Violators V* auf den extremalen Wert gesetzt, den Definition 7.3 fordert. Da beim Setzten der *Violator*-Daten durch den Einfügealgorithmus 7.1 die Eigenschaften $\delta_V = 1 \Rightarrow t_V \in \mathcal{N}_H$ und $\delta_V = 0 \Rightarrow t_V \in \mathcal{P}_H$ sichergestellt werden, ist $c_V^{\delta_V}$ der Wert, auf den c_V verschoben werden muss. Die dadurch verursachte Veränderung des Gesamtverbrauchs wird schließlich durch eine weitere Verschiebung des Arbeitspunktes nach Algorithmus 7.2 kompensiert.

Insgesamt wird die Laufzeitkomplexität für eine Änderung von b_i also durch den Aufwand $O(s)$

zum Wiedereinfügen von t_i in Algorithmus 7.4 sowie $O(s)$ für die Kompensation von $-\Delta\widetilde{W}_V$ in Algorithmus 7.5 begrenzt, ist somit also ebenfalls linear. Durch das *Violator*-Konzept wird außerdem dafür gesorgt, das Nutzenänderungen nur dann einen Subslot und damit einen Eingriff in die periodische Taskausführung auslösen können, wenn sie tatsächlich zu veränderten Energieverteilungen führen.

Algorithmus 7.5 : Wiederherstellung der nutzenoptimierten Energieverteilung

Eingabe : *Violator*-Daten (V, δ_V)
Wirkung : Umverteilung des Energieüberschusses bzw. -defizits von t_V

$\Delta\widetilde{W}_V \leftarrow \widetilde{w}_V T^{\text{main}} \left(\frac{1}{\frac{\delta_V}{c_V}} - \frac{1}{c_V} \right)$; // (7.1a)

$c_V \leftarrow c_V^{\delta_V}$;
$V \leftarrow$ `NIL_ID`;
Verschiebe Arbeitspunkt um $-\Delta\widetilde{W}_V$; `// Algorithmus 7.2`

Zu guter Letzt soll mit Abbildung 7.6 der Ablauf der Nutzenänderung noch einmal beispielhaft skizziert werden. 7.6a zeigt die Situation vor der Erhöhung von b_1 von 100 auf 175. Die Extrahierung von t_1 in 7.6b erfordert eine Verschiebung des *Handles* auf einen direkten Nachbarn.

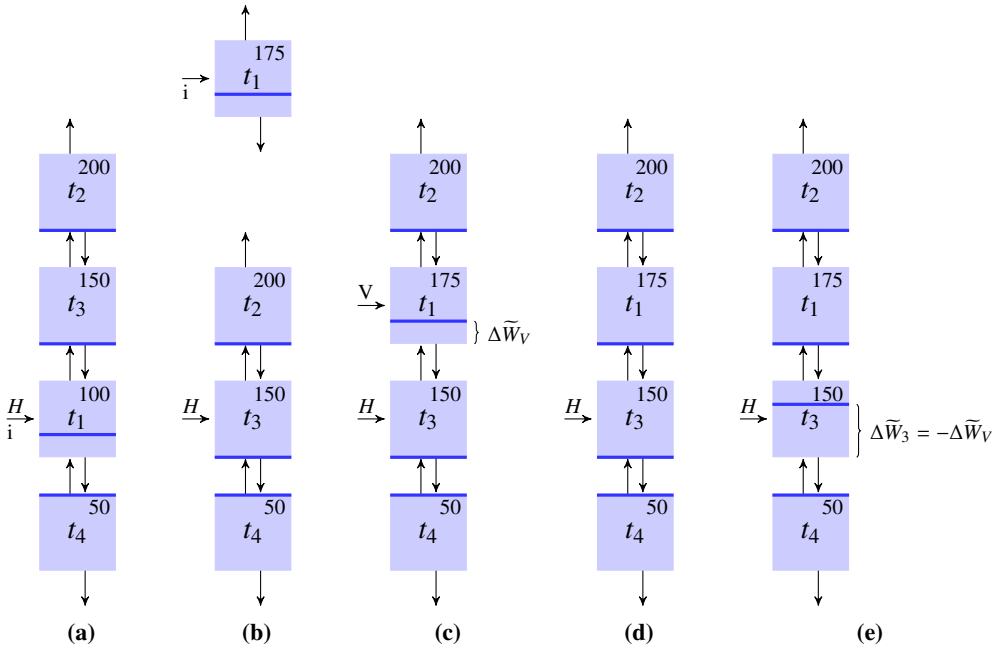


Abbildung 7.6: Zwischenschritte bei der Änderung von b_1 von 100 auf 175

Statt t_3 hätte dabei genauso gut t_4 gewählt werden können. Beim Wiedereinfügen von t_1 (7.6c) vor t_H ($\delta = 0$) entsteht eine nicht per Verschiebung des *Handles* auflösbare Verletzung der nutzenoptimalen Energieverteilung, denn für $H = 3$ ist $t_1 \in \mathcal{P}_H$ aber $c_1 \neq c_1^{\min}$ und für $H = 1$ ist $t_3 \in \mathcal{N}_H$ aber $c_3 \neq c_3^{\max}$. Die *Violator*-Daten $(V, \delta_V) = (1, 0)$ führen in 7.6d zur Verschiebung von c_1 nach $c_V^{\delta_V} = c_1^{\min}$, wodurch wieder eine nutzenoptimierte Energieverteilung erreicht wird. Da diese den Tasks nun aber $\Delta\widetilde{W}_V$ zu viel Energie zuordnet, muss der Arbeitspunkt vom *Handle* aus noch um $-\Delta\widetilde{W}_V$ verschoben werden, was in 7.6e resultiert. Es wurde also die Ausführungshäufigkeit von t_1 auf Kosten von t_3 erhöht und damit das Ziel der Nutzenänderung erreicht. Die Task t_2 und t_4 wurden wegen der Lokalität der Nutzenänderung zu keinem Zeitpunkt betrachtet.

KAPITEL

8 Anpassung des Verbrauchsprofils an das Einnahmeprofil

8.1 Ermittlung und Vorhersage der gewonnenen Energie

Für die Realisierung des energieneutralen Betriebs muss der mittlere Verbrauch des Sensorknotens an die im Mittel aus der regenerativen Energiequelle gewonnene Leistung angepasst werden. Um die Verluste beim Energietransport von der Quelle zum Speicher bzw. vom Speicher zum Verbraucher zu minimieren, muss auch der zeitliche Verlauf des Energieverbrauchs so gut wie möglich an den Verlauf des Energiegewinns angepasst werden. Dazu muss Letzterer aber erst einmal ermittelt werden.

Die Menge der eingenommenen Energie H^j während des Hauptslots S^j bestimmt der *Harvester* aus der Entwicklung $B^j \rightarrow B^{j+1}$ des Füllstandes des Energiespeichers und der Messung des Energieverbrauchs W^j des Sensorknotens während dieses Slots durch

$$H^j = B^{j+1} - B^j + W^j. \tag{8.1}$$

Damit werden sämtliche Verluste an der Hardware des *Harvester Boards* als Verminderung der gewonnenen Energie modelliert, denn aus Sicht des Verbrauchers ist es nicht relevant, ob die Quelle weniger Energie zur Verfügung stellt oder weniger der verfügbaren Energie nutzbar gemacht werden kann.

Wie bei jedem anderen denkbaren Messverfahren kann H^j auch mit Ausdruck (8.1) erst am Ende von S^j bestimmt werden. Da der Zielverbrauch $\widehat{W}^j$ aber bereits am Anfang des Slots vorgegeben werden muss, damit der Arbeitspunkt $\boldsymbol{c}$ entsprechend angepasst werden kann, wird eine Methode zur Bestimmung einer Einnahmevorhersage $\widetilde{H}^j$ für die gewonnene Energie benötigt. Dazu bedarf es einer adäquaten Modellierung der Dynamik der Energiequelle, was im

Falle von SNoW^5-RA dem tages- und jahreszeitlichen Verlauf der Sonneneinstrahlung entspricht.

Grundsätzlich kann der zeitliche Verlauf der Höhen- (Elevation) und Horizontalwinkel (Azimut) der Sonne für jeden beliebigen Standort auf wenige hundertstel Grad genau bestimmt [32] und wie in Abbildung 8.1 als Sonnenstandsdiagramm dargestellt werden. Diese Berechnungen erfordern aber neben der genauen Kenntnis des Standortes und der Ortszeit ein hohes Maß an Berechnungsgenauigkeit, insbesondere bei der Anwendung trigonometrischer Funktionen. Für die Mikrocontroller ohne Gleitkommaarithmetik ist diese Methode daher ungeeignet. Selbst mit vorberechneten und im Flash-Speicher des Sensorknotens abgelegten Datentabellen bliebe immer noch die Umrechnung vom Sonnenstand auf die verfügbare Energie zu realisieren. Dazu wäre die exakte Kenntnis von Fläche, Ausrichtung und Wirkungsgrad des Solarmoduls vonnöten, wobei sich die Fläche im Laufe der Zeit durch Verschmutzungen verringern kann und der Wirkungsgrad temperaturabhängig ist. Und selbst wenn diese Faktoren mit ausreichender Genauigkeit berücksichtigt wären, hätte man den Einfluss von meteorologischen Variationen und temporären Verschattungen noch nicht erfasst.

In [17] wurde eine weit weniger komplexe Vorhersagemethode vorgestellt, welche auf der Annahme beruht, dass sich der Eintrag der Solarenergie zu einer bestimmten Tageszeit nur geringfügig von den entsprechenden Werten der Vortage unterscheidet. Für den Sonnenstand als wichtigsten Faktor für die Dynamik der Energiequelle gilt diese Annahme, solange nur wenige Vortage berücksichtigt werden (vgl. Analemma-Kurven in Abbildung 8.1). Auch der zeitliche Verlauf des Schattenwurfs von stationären Objekten (etwa Gebäude oder Bäume) ist an aufeinander folgenden Tagen nahezu identisch. In Regionen mit stabilen Wetterlagen kann man sogar den meteorologischen Einflüssen ein gewisses Maß an Lokalität unterstellen.

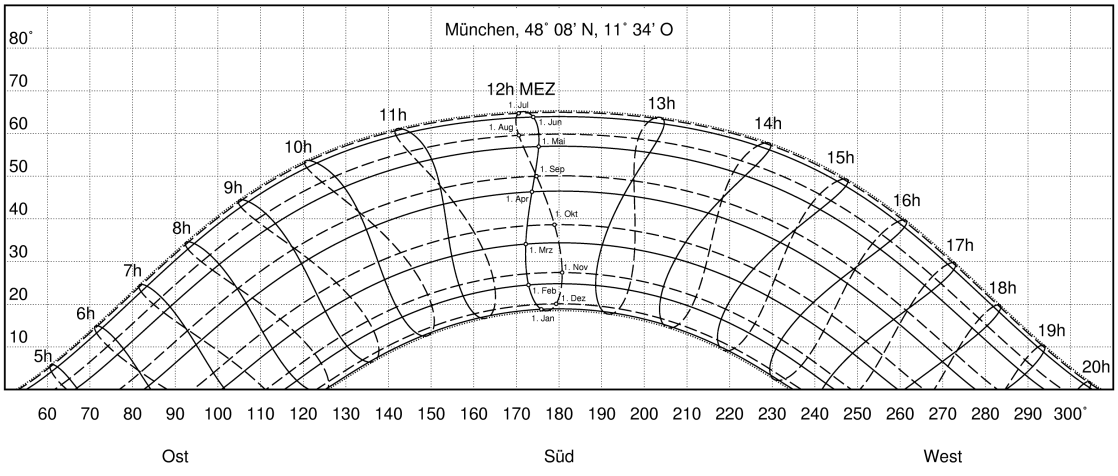


Abbildung 8.1: Sonnenstandsdiagramm (Elevation über Azimut) [1]

Um die fortlaufende Sequenz $(S^j)_{j\in\mathbb{N}}$ der Hauptslots so zu partitionieren, dass alle Slots einer Partition zur selben Tageszeit ausgeführt werden, muss T^{main} ein echter Teiler von 24 h sein, also $d := \frac{24\,\text{h}}{T^{\text{main}}} \in \mathbb{N}^+$ die exakte Anzahl der Hauptslots eines Tages bezeichnen. Alle Slots S^j mit gleichem $d_j := j \bmod d$ fallen dann in eine Partition und werden zur Vorhersage der Energieeinnahmen ihres spezifischen Tagesabschnitts herangezogen. Dabei spielt es keine Rolle, welche konkrete Uhrzeit die Slots $S^0, S^d, S^{2d}, \ldots$ repräsentieren, also zu welcher Tageszeit das System gestartet wurde. Eine Synchronisation der Systemzeit des Sensorknotens mit der tatsächlichen Ortszeit wird für dieses Vorhersageschema demnach nicht benötigt.

Ein einfacher Mittelwert aller bereits bestimmten Energieeinnahmen einer Partition wäre keine adäquate Vorhersage, denn mit zunehmendem Alter schwindet die Aussagekraft der Messdaten für die Vorhersage künftiger Einnahmen. Um dies zu modellieren, verwendet [17] eine exponentielle Glättung (EWMA) mit Glättungsfaktor $\alpha \in [0, 1]$ für die rekursive Vorhersage

$$\widetilde{H}^{j+d} := \begin{cases} H^j & \text{für } j < d \\ \alpha\widetilde{H}^j + (1-\alpha)H^j & \text{sonst.} \end{cases} \tag{8.2}$$

Lemma 8.1. *Der rekursive Ausdruck* (8.2) *ist äquivalent zur direkten Form*

$$\widetilde{H}^{j+d} = \alpha^{n_j} H^{d_j} + (1-\alpha)\sum_{k=1}^{n_j} \alpha^{n_j-k} H^{d_j+kd} \qquad \textit{mit } n_j := \left\lfloor \frac{j}{d} \right\rfloor$$

Beweis durch vollständige Induktion über $j \in \mathbb{N}$. Für $j < d$ ist $d_j = j$ und $n_j = 0$ und somit

$$\widetilde{H}^{j+d} \overset{(8.2)}{=} H^j = \alpha^0 H^j + (1-\alpha)\cdot \sum_{k=1}^{0} \alpha^{0-k} H^{j+kd} = \alpha^{n_j} H^{d_j} + (1-\alpha)\sum_{k=1}^{n_j} \alpha^{n_j-k} H^{d_j+kd},$$

was den Induktionsanfang beweist. Sei nun $j \geq d$. Dann ist $d_{j-d} = d_j$ und $n_{j-d} = n_j - 1$, so dass mit der Induktionsvoraussetzung (IV) für $j - d$ auch der Induktionsschritt folgt:

$$\begin{aligned}
\widetilde{H}^{j+d} &\overset{(8.2)}{=} \alpha\widetilde{H}^j + (1-\alpha)H^j \\
&= \alpha\widetilde{H}^{(j-d)+d} + (1-\alpha)H^j \\
&\overset{(IV)}{=} \alpha\left(\alpha^{n_{j-d}} H^{d_{j-d}} + (1-\alpha)\sum_{k=1}^{n_{j-d}} \alpha^{n_{j-d}-k} H^{d_{j-d}+kd}\right) + (1-\alpha)H^j \\
&= \alpha\left(\alpha^{n_j-1} H^{d_j} + (1-\alpha)\sum_{k=1}^{n_j-1} \alpha^{n_j-1-k} H^{d_j+kd}\right) + (1-\alpha)H^j \\
&\overset{d_j+n_jd=j}{=} \alpha^{n_j} H^{d_j} + (1-\alpha)\sum_{k=1}^{n_j-1} \alpha^{n_j-k} H^{d_j+kd} + (1-\alpha)\,\alpha^{n_j-n_j} H^{d_j+n_jd} \\
&= \alpha^{n_j} H^{d_j} + (1-\alpha)\sum_{k=1}^{n_j} \alpha^{n_j-k} H^{d_j+kd} \qquad \square
\end{aligned}$$

Lemma 8.1 zeigt zum einen die exponentiell fallende Gewichtung, mit welcher ältere Slots (kleinerer Index k) in den Mittelwert eingehen. Zum anderen kann man durch Summation aller Gewichte zu

$$\begin{aligned}\alpha^{n_j} + (1-\alpha)\sum_{k=1}^{n_j}\alpha^{n_j-k} &= \alpha^{n_j-0} + \sum_{k=1}^{n_j}\alpha^{n_j-k} - \sum_{k=1}^{n_j}\alpha^{n_j-k+1} \\ &= \sum_{k=0}^{n_j}\alpha^{n_j-k} - \sum_{k=0}^{n_j-1}\alpha^{n_j-k} = \alpha^{n_j-n_j} = 1\end{aligned} \tag{8.3}$$

die Korrektheit der Mittelwertbildung überprüfen.

Der über die Compilezeitkonstante `RA_HARVESTER_EWMA_SMOOTHING` spezifizierte Glättungsfaktor α bestimmt nun die Geschwindigkeit, mit der ältere Slots an Vorhersagekraft verlieren. Je kleiner α dabei wird, desto weniger werden ältere Slots bei der Vorhersage berücksichtigt und für $\alpha = 0$ geht mit $\widetilde{H}^{j+d} = H^j$ ausschließlich der jeweils letzte Messwert in die Vorhersage ein. Die konkrete Wahl von α ist anwendungsspezifisch und kann in einer Testphase durch die Forderung nach einem minimalen mittleren Vorhersagefehler optimiert werden. So wurde beispielsweise in [13] in einer 72 Tage dauernden Testphase mit $d = 48$ Slots pro Tag ein optimaler Glättungsfaktor von $\alpha = 15\ \%$ bestimmt.

Algorithmus 8.1 : Aktualisierung des Energieeinnahmeprofils

Eingabe : Index j des abgeschlossenen Slots, Gesamtverbrauch W während dieses Slots, aktueller Primärspeicherfüllstand B, Primärspeicherfüllstand b beim letzten Slotwechsel

Wirkung : Anpassung von $\widetilde{H}_p^{d_j}$ und b sowie Berechnung des Vorhersagefehlers ΔH des Slots

wenn *Sekundärspeicher während Slot j nicht verwendet* **dann** $H \leftarrow B - b + W$; // (8.1)
sonst wenn $0 < j < d$ **dann** $H \leftarrow \widetilde{H}_p^{d_{j-1}}$;
sonst $H \leftarrow \widetilde{H}_p^{d_j}$;

$b \leftarrow B$;
$\Delta H \leftarrow H - \widetilde{H}_p^{d_j}$;
wenn $j < d$ **dann** // (8.2)
  $\widetilde{H}_p^{d_j} \leftarrow H$;
sonst
  $\widetilde{H}_p^{d_j} \leftarrow \alpha\widetilde{H}_p^{d_j} + (1-\alpha)H$;

Mit der rekursiven Berechnungsvorschrift (8.2) kann ein 24 h Vorhersageprofil durch die zirkuläre Verwaltung eines linearen Speichers $\widetilde{H}_p^0, \ldots, \widetilde{H}_p^{d-1}$ nach Algorithmus 8.1 realisiert werden. Dabei repräsentiert $\widetilde{H}_p^{d_{j+k}}$ die Einnahmevorhersage $\widetilde{H}^{j+k}$, wenn j der Index des aktuellen Slots und $k < d$ die Vorschauweite ist.

Bei der Anwendung von (8.1) muss noch ein Problem beachtet werden, welches sich aus der Zweiteilung des Energiespeichers ergibt. Der Gesamtfüllstand des Speichers kann nicht ohne weiteres als Summe der Inhalte von Primär- und Sekundärspeicher betrachtet werden, denn beim Energietransport zwischen den beiden Speichern entstehen zusätzliche Verluste. H^j wäre somit vom Auftreten einer Energieumverteilung im Slot S^j abhängig. In Abschnitt 5.1 wurde außerdem die Messgenauigkeit bei der Bestimmung des Inhaltes des Sekundärspeichers mit 100 mWh abgeschätzt, was den Verbrauch W des Sensorknotens während eines Slots selbst bei $T^{\text{main}} = 1$ h deutlich übersteigt. Der *Harvester* beschränkt sich deshalb darauf, den Energiegewinn aus der Entwicklung des Füllstandes des Primärspeichers abzuleiten. Dies ist aber nur für Slots möglich, während derer kein Energietransport vom oder zum Sekundärspeicher stattgefunden hat. Andernfalls unterscheidet Algorithmus 8.1 zwischen einer Initialisierungsphase ($j < d$) und der anschließenden Phase mit bereits initialisierten Vorhersagewerten. Während der ersten Phase wird $\widetilde{H}_p^{d_j}$ unter der Annahme zeitlicher Lokalität mit der Vorhersage $\widetilde{H}_p^{d_{j-1}}$ des direkten Vorgängerslots initialisiert. In der zweiten Phase bleiben die Vorhersagen unverändert, wenn (8.1) nicht anwendbar ist. Da der Energietransport vom Primär- zum Sekundärspeicher nicht systematisch zu bestimmten Tageszeiten stattfindet, werden die Vorhersagen aller Slots regelmäßig aktualisiert.

Jede Vorhersage ist mit einem Vorhersagefehler $\Delta H^j := H^j - \widetilde{H}^j$ behaftet und muss bei der Vorgabe des Energieverbrauchs späterer Slots durch den *Regulator* kompensiert werden (vgl. nächster Abschnitt). Um den Speicherbedarf von SNoW^5-RA zu reduzieren, wird dabei nur der Vorhersagefehler ΔH des letzten Slots verwaltet, so dass die entsprechende Fehlerkompensation direkt an die Aktualisierung des Energieeinnahmeprofils anschließen muss.

8.2 Vorgabe des Verbrauchsprofils

Mit den im letzten Abschnitt vorgestellten Methoden verfügt SNoW^5-RA über eine 24-stündige Vorhersage für das Einnahmeprofil der regenerativen Energiequelle. In diesem Abschnitt wird daraus ein Verbrauchsprofil $\widehat{W}_p^0, \ldots, \widehat{W}_p^{d-1}$ mit dem Ziel abgeleitet, einen energieneutralen Betrieb bestmöglich anzunähern und dabei die Verluste am Energiespeicher zu reduzieren. Dass ein energieneutraler Betrieb nicht in jedem Fall erreicht werden kann, ergibt sich aus der Beschränkung der Skalierbarkeit des Verbrauchers, denn mit den extremalen Arbeitspunkten $\boldsymbol{c}^{\max}$ und $\boldsymbol{c}^{\min}$ ist nach (4.4) auch der Verbrauch während eines Hauptslots auf

$$\widetilde{W}^{\min} \leq \widehat{W}_p^k \leq \widetilde{W}^{\max} \qquad \forall\, k \in \{0, \ldots, d-1\}$$

begrenzt. Aus Abschnitt 1.5 ist aber bekannt, dass der dynamische Bereich der Energieeinnahmen diesen beschränkten Verbrauchsbereich über- und unterschreitet (vgl. Abbildung 1.6), so dass die Ausgangssituation für die Verbrauchsregulierung qualitativ durch Abbildung 8.2 beschrieben werden kann. In den Mittagsstunden steht demnach mehr Energie zur Verfügung, als überhaupt verbraucht werden kann, während nachts ein Energiedefizit entsteht. Nur in den

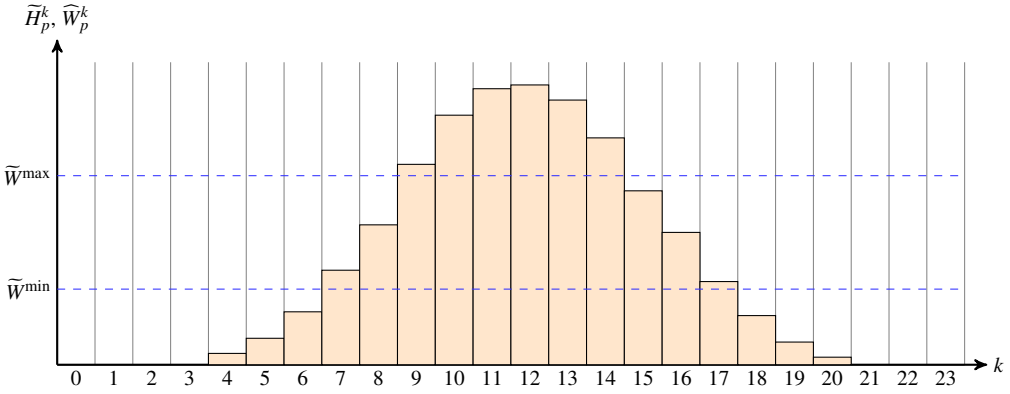


Abbildung 8.2: Ausgangssituation der Verbrauchsregulierung ($T^{\text{main}} = 1$ h)

Übergangsphasen liegt die gewonnene Energiemenge im Skalierbarkeitsbereich des Verbrauchers.

Liegt nun der Durchschnitt $\widetilde{H}_p := \frac{1}{d}\sum_{k=0}^{d-1} \widetilde{H}_p^k$ des Einnahmeprofils außerhalb des skalierbaren Bereichs, so kann ein energieneutraler Betrieb nur durch ein extremales Verbrauchsprofil $\widehat{W}_p^k = \widetilde{W}^{\min}$ bzw. $\widehat{W}_p^k = \widetilde{W}^{\max}$ $\forall\ k \in \{0, \ldots, d-1\}$ angenähert werden. Andernfalls muss die Fläche unter dem Verbrauchsprofil der Fläche unter dem Einnahmeprofil entsprechen, was sich am einfachsten durch ein konstantes Verbrauchsprofil $\widehat{W}_p^k = \widetilde{H}_p$ $\forall\ k \in \{0, \ldots, d-1\}$ wie in Abbildung 8.3 realisieren lässt.

Mit einem solchen konstanten Verbrauchsprofil wird der Energiespeicher aber unnötig stark belastet. So kann der Energietransport vom und zum Speicher reduziert werden, wenn in den Phasen niedriger Energieeinnahmen weniger und in in den Phasen mit ausreichenden Energieein-

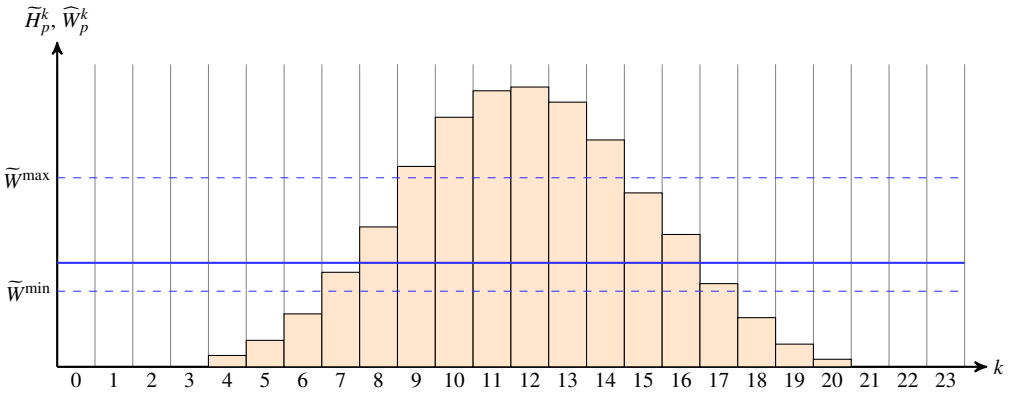


Abbildung 8.3: konstantes Verbrauchsprofil

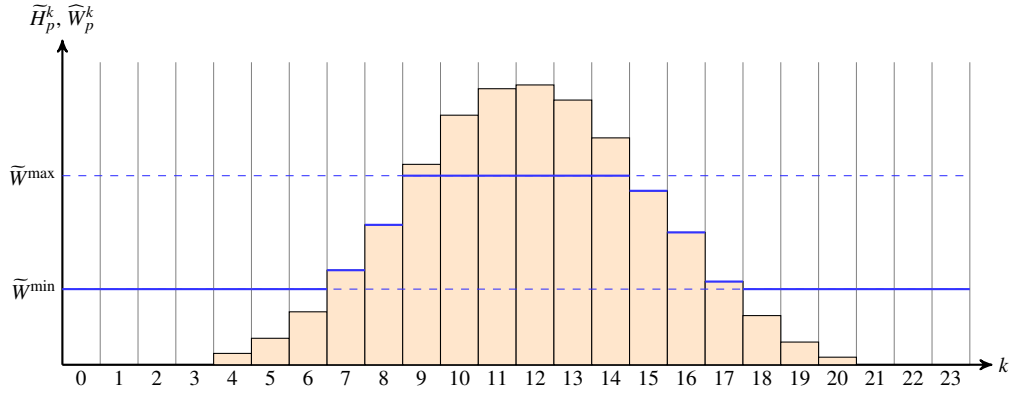


Abbildung 8.4: JIT-Verbrauch

nahmen entsprechend mehr verbraucht wird. Die geringste Speicherbelastung verursacht daher das in Abbildung 8.4 dargestellte und wie folgt charakterisierte Verbrauchsprofil.

Definition 8.1 (JIT-Verbrauch)**.** *Das Verbrauchsprofil mit minimaler Speicherbelastung*

$$\widehat{W}_p^k = \begin{cases} \widetilde{W}^{\max} & \textit{falls } \widetilde{H}_p^k \geq \widetilde{W}^{\max} \\ \widetilde{W}^{\min} & \textit{falls } \widetilde{H}_p^k \leq \widetilde{W}^{\min} \\ \widetilde{H}_p^k & \textit{sonst} \end{cases} \qquad \forall\, k \in \{0, \ldots, d-1\}$$

wird als JIT-Verbrauch (just in time) bezeichnet.

Im Allgemeinen wird mit dem JIT-Verbrauch aber kein energieneutraler Betrieb realisiert, so dass eine weitere Anpassung des Profils notwendig ist. Steht insgesamt mehr Energie zur Verfü-

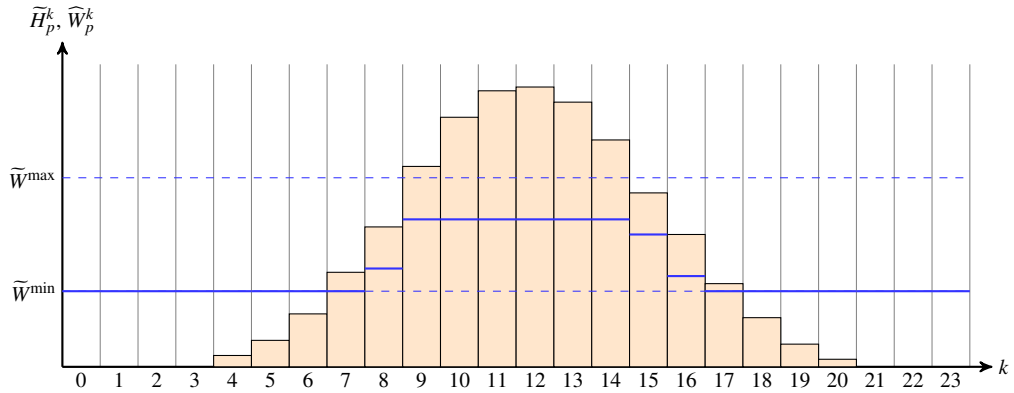


Abbildung 8.5: für Energieneutralität angepasstes JIT-Profil

gung, als mit dem JIT-Profil alloziert wird, so kann der Verbrauch in den Phasen niedriger Energieeinnahmen gleichmäßig über den Minimalverbrauch angehoben werden, bis die Verbrauchssumme der Summe der Einnahmen entspricht. Umgekehrt müssen im Falle eines Einnahmedefizits des JIT-Profils die Verbrauchswerte in den Phasen hoher Energieeinnahmen gesenkt werden, bis dieses Defizit kompensiert ist. Abbildung 8.5 veranschaulicht letzteren Fall. Die Anpassung des JIT-Profils endet spätestens bei der minimalen bzw. maximalen Verbrauchszuordnung, wenn der Durchschnitt der Einnahmen außerhalb des skalierbaren Verbrauchsbereichs liegt.

Bislang wurde vernachlässigt, dass sowohl das Einnahme- als auch das Verbrauchsprofil letztlich nur auf Schätzungen beruhen und dadurch mit Ungenauigkeiten einhergehen. So wird die tatsächliche Energieeinahme H^j in einem Slot S^j des Profils im Allgemeinen vom vorhergesagten Wert $\widetilde{H}^j$ abweichen. Ebenso kann der tatsächliche Verbrauch W^j vom angestrebten Verbrauch $\widehat{W}^j$ abweichen, da die Arbeitspunkteinstellung per *Adaptive Duty Cycling* nur mit dem erlernten Verbrauchsvektor $\widetilde{\boldsymbol{w}}$ arbeiten kann (vgl. Abschnitt 7.2), welcher sich im Allgemeinen vom tatsächlichen Verbrauchsvektor $\boldsymbol{w}$ unterscheidet. Am Ende von S^j sind nun aber die tatsächlichen Einnahme- und Verbrauchswerte bekannt, so dass die Fehlplanung bzgl. der Energieneutralität durch eine Anpassung des zukünftigen Verbrauchsprofils ausgeglichen werden kann. Die Methode, das Verbrauchsprofil basierend auf der Einschätzung zukünftiger Energieeinnahmen festzulegen und während der Abarbeitung des Profils erkannte Fehler durch Verbrauchsänderungen noch nicht abgearbeiteter Slots auszugleichen, beruht auf den Überlegungen von [13].

Vor der Beschreibung der konkreten Implementierung des *Regulators* gilt die letzte Vorüberlegung noch einmal der Beschränkung der Verbrauchsskalierung und der damit einhergehenden Schwierigkeit, einen energieneutralen Betrieb zu gewährleisten. Der 24 h Mittelwert $\widetilde{H}_p$ des Einnahmeprofils ist keineswegs konstant, sondern durch verschiedene mittel- (meteorologische Effekte) und langfristige (jahreszeitliche Änderung des Sonnenstandes, vgl. Abbildung 8.1) Entwicklungen beeinflusst. So kann es vorkommen, dass an einem Tag mit $\widetilde{H}_p > \widetilde{W}^{\max}$ selbst bei permanent maximiertem Verbrauch ein Energieüberschuss von $d \cdot (\widetilde{H}_p - \widetilde{W}^{\max})$ entsteht, welcher in den Profilen späterer Tage mit $\widetilde{H}_p < \widetilde{W}^{\max}$ berücksichtigt werden muss. Für einen langfristigen energieneutralen Betrieb ist daher der Ausgleich von Energieüberschüssen und -difiziten über einen beliebig langen Zeitraum notwendig. Der *Regulator* verwaltet dazu einen Übertrag Ψ^j, für den vor jedem Slot S^j

$$\Psi^j = \sum_{k=0}^{j-1} \left(H^k - W^k \right) + \sum_{k=0}^{d-1} \left(\widetilde{H}^{j+k} - \widehat{W}^{j+k} \right) \overset{(8.1)}{=} B^j - B^0 + \sum_{k=0}^{d-1} \left(\widetilde{H}^{j+k} - \widehat{W}^{j+k} \right) \tag{8.4}$$

gilt. Dabei beschreibt die erste Summe den bis zum aktuellen Zeitpunkt tatsächlich erwirtschafteten Energieüberschuss, während die zweite Summe das 24-stündige Vorschaufenster berücksichtigt. Um diesen Ausdruck direkt berechnen zu können, wäre die zusätzliche Speicherung aller H^k, W^k bis zum aktuellen Zeitpunkt notwendig. Da dies nicht realisierbar ist, muss Ψ^j

rekursiv berechnet bzw. akkumuliert werden. Alle Profildaten $\widetilde{H}_p^k$ und $\widehat{W}_p^k$ werden mit Null initialisiert, daher ist auch

$$\Psi^0 = \sum_{k=0}^{-1} \left(H^k - W^k\right) + \sum_{k=0}^{d-1} \left(\widetilde{H}^k - \widehat{W}^k\right) = 0. \tag{8.5}$$

Wenn nun beim Übergang von S^j zu S^{j+1} das Einnahmeprofil $\widetilde{H}_p^{d_j}$ durch Algorithmus 8.1 bereits auf $\widetilde{H}^{j+d}$ aktualisiert wurde, dann ist im Allgemeinen $\widetilde{H}^{j+d} \neq \widetilde{H}^j$. Zu diesem Zeitpunkt wurde das Verbrauchsprofil $\widehat{W}_p^j$ aber noch nicht verändert. Man kann dies so interpretieren, als ob der *Regulator* zunächst $\widehat{W}^{j+d} = \widehat{W}^j$ vorgibt, den alten Zielverbrauch also für die nächste Periode übernimmt. Würde das Verbrauchsprofil so bleiben, dann wäre

$$\begin{aligned}
\Psi^{j+1} &\overset{(8.4)}{=} \sum_{k=0}^{j} \left(H^k - W^k\right) + \sum_{k=0}^{d-1} \left(\widetilde{H}^{j+1+k} - \widehat{W}^{j+1+k}\right) \\
&= \sum_{k=0}^{j} \left(H^k - W^k\right) + \sum_{k=1}^{d} \left(\widetilde{H}^{j+k} - \widehat{W}^{j+k}\right) \\
&= \sum_{k=0}^{j-1} \left(H^k - W^k\right) + \sum_{k=0}^{d-1} \left(\widetilde{H}^{j+k} - \widehat{W}^{j+k}\right) + \left(H^j - W^j\right) + \left(\widetilde{H}^{j+d} - \widehat{W}^{j+d}\right) - \left(\widetilde{H}^{j+0} - \widehat{W}^{j+0}\right) \\
&\overset{(8.4)}{=} \Psi^j + \underbrace{\left(H^j - \widetilde{H}^j\right)}_{\Delta H} - \underbrace{W^j}_{W} + \underbrace{\widetilde{H}^{j+d}}_{\widetilde{H}_p^{d_j}} + \underbrace{\left(\widehat{W}^j - \widehat{W}^{j+d}\right)}_{0}.
\end{aligned}$$

Für die Aktualisierung des Übertrags werden also der Gesamtverbrauch W und der Vorhersagefehler ΔH für die Energieeinnahmen des gerade abgeschlossenen Slots sowie die vom *Harvester* aktualisierte Einahmevorhersage $\widetilde{H}_p^{d_j}$ benötigt.

Algorithmus 8.2 : Ändere Eintrag im Verbrauchsprofil

Eingabe : Profilindex k, neue Verbrauchsvorgabe $\widehat{W}$, Kompensationsvariable Δ
wenn $\widehat{W} > \widetilde{W}^{\max}$ **dann** $\widehat{W} \leftarrow \widetilde{W}^{\max}$;
wenn $\widehat{W} < \widetilde{W}^{\min}$ **dann** $\widehat{W} \leftarrow \widetilde{W}^{\min}$;
$\Delta \leftarrow \Delta - (\widehat{W} - \widehat{W}_p^k)$;
$\widehat{W}_p^k \leftarrow \widehat{W}$;

Um (8.4) weiter zu gewährleisten, muss nun jede Änderung des Verbrauchsprofils durch eine dazu additiv inverse Änderung des Übertrags kompensiert werden. Solche Energieverschiebungen vom Verbrauchsprofil in den Übertrag oder umgekehrt werden deswegen ausschließlich mittels Algorithmus 8.2 vorgenommen. Die hierbei verwendete Kompensationsvariable Δ entspricht

entweder dem globalen Übertrag Ψ oder einem anderen lokalen Akkumulator für Energiedifferenzen, der später auf den Übertrag umgelegt wird. Neben der Übertragsbehandlung übernimmt Algorithmus 8.2 auch die Begrenzung des Verbrauchsprofils auf den skalierbaren Bereich.

Die Annäherung an einen energieneutralen Betrieb entspricht der Reduktion von $|\Psi|$ durch eine gezielte Änderung des Verbrauchsprofils. Dazu wird Algorithmus 8.3 bei jedem Haupslotwechsel nach der Aktualisierung des Einnahmeprofils ausgeführt. Die erste Zeile realisiert dabei die oben beschriebene Aktualisierung des Übertrags.

Algorithmus 8.3 : Verbrauchsprofil für nächsten Slot vorbereiten

Eingabe : Index j, Vorhersagefehler ΔH und tatsächlicher Energieverbrauch W des abgeschlossenen Slots, Energieeinnahmeprofil $\widetilde{H}_p^0, \ldots, \widetilde{H}_p^{d-1}$

Ausgabe : Verbrauchsvorgabe $\widehat{W}$ für nächsten Hauptslot

1 $\Psi \leftarrow \Psi + \Delta H - W + \widetilde{H}_p^{d_j}$;
2 **wenn** $d_j = 0$ *oder* $j < d$ **dann**
3 **für alle** $k \in \{0, \ldots, d-1\}$ `// JIT-Verbrauch einstellen`
4 Ändere Verbrauchsvorgabe k zu $\widetilde{H}_p^k$ mit Kompensationsvariable Ψ;
5 Kompensiere Ψ;
6 **sonst**
7 $\Delta \leftarrow \Delta H - W + \widehat{W}_p^{d_j}$; `// Planungsfehler ermitteln`
8 **für alle** $k \in \{0, \ldots, d-1\}$ `// Verbrauchsbegrenzung einhalten`
9 Ändere Verbrauchsvorgabe k zu $\widehat{W}_p^k$ mit Kompensationsvariable Ψ;
10 $\Psi \leftarrow \Psi - \Delta$;
11 Kompensiere Δ ; `// Planungsfehler kompensieren`
12 $\Psi \leftarrow \Psi + \Delta$;
13 $\widehat{W} \leftarrow \widehat{W}_p^{d_{j+1}}$;

Am Anfang einer 24-Stunden Periode ($d_j = 0$) und während der Initialisierungsphase des *Harvesters* ($j < d$) erzeugt Algorithmus 8.3 zunächst ein JIT-Profil (Zeile 3) und versucht anschließend, den Übertrag $|\Psi|$ mittels Algorithmus 8.4 zu minimieren (Zeile 5). Dies entspricht dem Übergang von Abbildung 8.4 zu 8.5, also dem Streben nach Energieneutralität.

Der Übertrag Ψ ist das Softwarependant des Energiespeichers und schwankt bei nicht permanent über- oder unterdimensionierter Energiequelle im Verlaufe einer 24 h Periode, da in die erste Summe von Ausdruck (8.4) mal mehr und mal weniger Slots mit Energieüberschuss als mit Energiedefizit eingehen. Eine vollständige Reduktion von $|\Psi|$ nach jedem Slot hätte zur Folge, dass die in den Morgenstunden erzeugten Verbrauchsprofile kaum mehr als den Minimalverbrauch allozieren, da der Übertrag während der Nacht weit abgesunken ist. Erst nach einigen

Algorithmus 8.4 : Energiedifferenz durch Verbrauchsanpassung kompensieren

Eingabe : Energiedifferenz Δ

$\mathcal{S} \leftarrow \{k \in \{0, \ldots, d-1\} : (\Delta > 0 \wedge \widehat{W}_p^k < \widetilde{W}^{\max}) \vee (\Delta < 0 \wedge \widehat{W}_p^k > \widetilde{W}^{\min})\}$;

solange $\Delta \neq 0$ *und* $\mathcal{S} \neq \emptyset$

  $\delta \leftarrow \frac{\Delta}{|\mathcal{S}|}$;

  für alle $k \in \mathcal{S}$

    Ändere Verbrauchsvorgabe k zu $\widehat{W}_p^k + \delta$ mit Kompensationsvariable Δ;

    wenn $\widehat{W}_p^k$ *extremal* **dann** $\mathcal{S} \leftarrow \mathcal{S} \setminus \{k\}$;

Slots mit Einnahmeüberschuss wäre der Übertrag wieder soweit ausgeglichen, dass auch höhere Verbrauchswerte vorgegeben werden können. Der Verbrauch würde also zu spät gesteigert werden. Während der Initialisierungsphase existiert aber noch gar keine Einnahmevorhersage, so dass hier $|\Psi|$ nach jedem Slot vollständig kompensiert werden kann, um die nächste Periode möglichst mit $\Psi = 0$ zu beginnen.

Innerhalb der späteren 24 h Perioden ($d_j \neq 0 \wedge j \geq d$) kompensiert Algorithmus 8.3 dann nur noch die lokalen Planungsfehler, also die Abweichungen der Gewinnvorhersage und der Verbrauchssteuerung des gerade abgeschlossenen Slots S^j. Da die Annahmen über die Verbrauchskomponenten $\widetilde{\boldsymbol{w}}$ der skalierbaren Tasks und damit auch die Verbrauchsgrenzen $\widetilde{W}^{\min}$ und $\widetilde{W}^{\max}$ nach jedem Hauptslot durch den *Estimator* geändert werden können (vgl. Kapitel 9), muss die Beschränkung des Verbrauchsprofils auf diese Grenzen durch Anwendung von Algorithmus 8.2 auf jede Verbrauchsvorgabe (Zeile 8) gewährleistet werden.

Für die Kompensation einer Energiedifferenz Δ verwaltet Algorithmus 8.4 eine Menge $\mathcal{S}$ der Indizes der Slots, welche zur Änderung des Gesamtverbrauchs in die erforderliche Richtung beitragen können. Bei der gleichmäßigen Verteilung von Δ auf diese Slots kann es vorkommen, dass $\widehat{W}_p^k$ für ein $k \in \mathcal{S}$ nicht vollständig um $\frac{\Delta}{|\mathcal{S}|}$, sondern nur bis zur entsprechenden Skalierbarkeitsgrenze ($\widetilde{W}^{\min}$ bzw. $\widetilde{W}^{\max}$) geändert werden kann. Dieser Slot wird dann aus $\mathcal{S}$ entfernt und die verbliebene Energiedifferenz muss durch einen weiteren Schleifendurchlauf kompensiert werden. Wird $\mathcal{S}$ in einem Durchlauf nicht verkleinert, so konnte Δ vollständig auf die Slots aufgeteilt werden und der Algorithmus terminiert mit der ersten Abbruchbedingung der äußeren Schleife. Ansonsten muss nach spätestens d Iterationen $\mathcal{S} = \emptyset$ gelten, so dass Algorithmus 8.4 auch beim Erreichen eines extremalen Verbrauchsprofils terminiert. Die Ausführungshäufigkeiten der beiden verschachtelten Schleifen werden somit durch die Anzahl der Slots des Profils beschränkt, so dass die Laufzeitkomplexität für die Kompensation von Energiedifferenzen mit $O(d^2)$ abgeschätzt werden kann.

KAPITEL

9 Erlernen des Energieverbrauchs skalierbarer Tasks

9.1 Rahmenbedingungen möglicher Lernstrategien

Am Ende eines Hauptslots repräsentiert der Vektor $\boldsymbol{e} \in \mathbb{N}^{s+1}$ die Ausführungshäufigkeiten der skalierbaren Tasks. Während $e_1, \ldots, e_s$ durch direktes Abzählen der Ausführungen der Tasks $t_1, \ldots, t_s$ bestimmt werden, ist $e_0 := 1$ implizit festgelegt, damit die Verbrauchskomponente w_0 des imaginären Tasks gerade dem Grundverbrauch des Sensorknotens während des Hauptslots entspricht (vgl. Abschnitt 4.2). Außerdem ist am Ende eines Hauptslots die insgesamt vom Sensorknoten verbrauchte Energiemenge W durch Messungen des *Harvester Boards* bekannt.

Der Gesamtverbrauch entspricht der Summe der mit den Ausführungshäufigkeiten gewichteten Verbrauchskomponenten der skalierbaren Tasks, also

$$W = \sum_{i=0}^{s} e_i w_i = \boldsymbol{e} \cdot \boldsymbol{w}. \tag{9.1}$$

Wegen der Anzahl der Unbekannten $w_0, \ldots, w_s$ kann aber aus der Kenntnis von W und $\boldsymbol{e}$ für $s \geq 1$ nicht direkt auf den Verbrauchsvektor $\boldsymbol{w}$ geschlossen werden. Der Lösungsraum wird lediglich auf die Hyperebene

$$\mathcal{L}_{\boldsymbol{e},W} := \{\widetilde{\boldsymbol{w}} \in \mathbb{R}^{s+1} : W = \boldsymbol{e} \cdot \widetilde{\boldsymbol{w}}\} \tag{9.2}$$

eingeschränkt. Da jeder Punkt dieser Hyperebene den Energieverbrauch des Sensorknotens für einen bestimmten Ausführungsvektor exakt prognostiziert, kann $\mathcal{L}_{\boldsymbol{e},W}$ als Menge lokal optimaler Verbrauchsvektoren bezeichnet werden. Der gesuchte, also global optimale Verbrauchsvektor $\boldsymbol{w}$

liegt im Schnitt aller möglichen Hyperebenen und kann durch Lösung eines linearen Gleichungssystems

$$\mathbf{E} \cdot \boldsymbol{w} = \begin{pmatrix} \boldsymbol{e}^0 \\ \vdots \\ \boldsymbol{e}^s \end{pmatrix} \cdot \boldsymbol{w} = \begin{pmatrix} e_0^0 & \dots & e_s^0 \\ \vdots & & \vdots \\ e_0^s & \dots & e_s^s \end{pmatrix} \cdot \begin{pmatrix} w_0 \\ \vdots \\ w_s \end{pmatrix} = \begin{pmatrix} W^0 \\ \vdots \\ W^s \end{pmatrix} = \boldsymbol{W} \tag{9.3}$$

bestimmt werden. Für die Koeffizienten eines solchen Gleichungssystems werden die Ausführungsvektoren $\boldsymbol{e}^0, \dots, \boldsymbol{e}^s$ und die zugehörigen Verbrauchswerte $W^0, \dots, W^s$ aus $s+1$ verschiedenen Hauptslots benötigt.

Der Rang der Koeffizientenmatrix $\mathbf{E}$ begrenzt die Anzahl der eindeutig ermittelbaren Verbrauchskomponenten. Für die vollständige Bestimmung von $\boldsymbol{w}$ wird daher eine Koeffizientenmatrix mit vollem Rang $s+1$ benötigt.

Der vom *Regulator* vorgegebene Zielverbrauch liegt in der Nacht am unteren Ende des regulierbaren Bereichs, um das durch den Speicher auszugleichende Energiedefizit zu minimieren. Während der Mittagsstunden herrscht hingegen ein permanenter Energieüberschuss, der den *Regulator* dazu veranlasst, den Sensorknoten an seiner oberen Leistungsgrenze zu betreiben. Der Unterschied in den Verbrauchsvorgaben zwischen zwei aufeinander folgenden Slots ist in diesen beiden Phasen also sehr gering und wird beim *Adaptive Duty Cycling* durch Veränderung der Perioden weniger skalierbarer Tasks ausgeglichen. Da die Perioden der meisten skalierbaren Tasks demnach unverändert bleiben, werden diese im alten und neuen Slot gleich häufig ausgeführt. Die Funktionsweise von *Regulator* und *Adaptive Duty Cycling* sorgen also dafür, dass die Ausführungsvektoren aufeinander folgender Hauptslots häufig in vielen Komponenten übereinstimmen, was den Rang der daraus gebildeten Koeffizientenmatrix verringert.

Um diesem Problem zu begegnen, werden im Algorithmus 9.1 drei verschiedene Strategien kombiniert. Beim gesteuerten Lernen (Abschnitt 9.4) werden die Mechanismen des *Adaptive Duty Cycling* außer Kraft gesetzt und der Arbeitspunkt $\boldsymbol{c}$ stattdessen direkt vom *Estima-*

Algorithmus 9.1 : Verbrauchskomponenten erlernen

Eingabe : Ausführungsvektor $\boldsymbol{e}$, gemessener Gesamtverbrauch W, Verbrauchsannahme $\widetilde{\boldsymbol{w}}$
Wirkung : Annäherung von $\widetilde{\boldsymbol{w}}$ an $\boldsymbol{w}$
wenn $|\boldsymbol{e} \cdot \widetilde{\boldsymbol{w}} - W|$ *zu groß* **dann**
  Verwende gesteuertes Lernen;
sonst
  Löse lineares Gleichungssystem mit Koeffizienten der letzten Slots;
  Verschiebe $\widetilde{\boldsymbol{w}}$ auf kürzestem Weg zu $\mathcal{L}_{e,W}$;
Berechne $\widetilde{W}, \widetilde{W}^{\min}, \widetilde{W}^{\max}$ mit neuem $\widetilde{\boldsymbol{w}}$;
Speichere $\boldsymbol{e}$ und W als Koeffizienten für spätere Gleichungssysteme;

tor für die Erzeugung einer geeigneten Koeffizientenmatrix vorgegeben. Weil dabei aber weder Energieneutralität gewährleistet noch nutzenoptimierte Energieverteilung angestrebt wird, darf das gesteuerte Lernen nicht permanent zum Einsatz kommen. Es wird daher nur verwendet, wenn die Verbrauchsannahmen $\widetilde{\boldsymbol{w}}$ zu stark vom tatsächlichen Verbrauchsvektor $\boldsymbol{w}$ abweichen, also beispielsweise nach der Initialisierung des Systems ohne konkrete Verbrauchsannahmen oder nach gravierenden Veränderungen im Verbrauchsverhalten des Sensorknotens. Der Schwellwert für die Verwendung des gesteuerten Lernens wird durch die Compilezeitkonstante `RA_ESTIMATOR_THRESHOLD` bestimmt.

Im regulären Betrieb, also bei der Bestimmung des Arbeitspunktes $\boldsymbol{c}$ durch *Regulator* und *Adaptive Duty Cycling*, wird das Lösen linearer Gleichungssysteme (Abschnitt 9.3), das für stark variierende Ausführungsvektoren $\boldsymbol{e}$ geeignet ist, mit dem direkten Verschieben der Verbrauchsannahmen $\widetilde{\boldsymbol{w}}$ auf die aktuelle Hyperebene $\mathcal{L}_{\boldsymbol{e},W}$ (Abschnitt 9.2) verbunden. Letzteres sorgt in den Phasen kaum veränderlicher Ausführungsvektoren dafür, dass die Differenz zwischen angenommenem und tatsächlichem Gesamtverbrauch des Sensorknotens minimiert wird.

9.2 Fällen eines Lots auf die Hyperebene lokal optimaler Verbrauchsvektoren

Wie im letzten Abschnitt gezeigt, kann man am Ende eines Hauptslots die Hyperebene $\mathcal{L}_{\boldsymbol{e},W}$ der Verbrauchsvektoren bestimmen, die für den gegebenen Ausführungsvektor $\boldsymbol{e}$ zum Gesamtverbrauch W führen. Der tatsächliche Verbrauchsvektor $\boldsymbol{w}$ liegt auf dieser Ebene, der aktuell angenommene Verbrauchsvektor $\widetilde{\boldsymbol{w}}$ aber im Allgemeinen nicht. Abbildung 9.1 veranschaulicht diesen Sachverhalt.

Die Annäherung von $\widetilde{\boldsymbol{w}}$ an $\boldsymbol{w}$ entspricht der Suche eines Verschiebungsvektors $\Delta\widetilde{\boldsymbol{w}} \in \mathbb{R}^{s+1}$ mit

$$|\widetilde{\boldsymbol{w}} + \Delta\widetilde{\boldsymbol{w}} - \boldsymbol{w}| < |\widetilde{\boldsymbol{w}} - \boldsymbol{w}|. \tag{9.4}$$

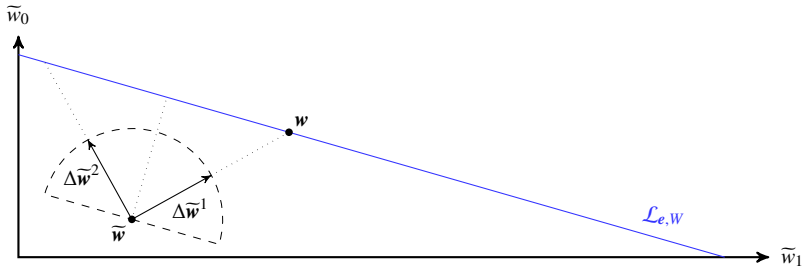


Abbildung 9.1: Hyperebene lokal optimaler Verbrauchsvektoren

Um diese Suche auf eine endliche Teilmenge $\mathcal{U} \subseteq \mathbb{R}^{s+1}$ einzuschränken, kann man beispielsweise die Länge $|\Delta\widetilde{\boldsymbol{w}}|$ festlegen und dann die Ausrichtung des Vektors mit fester Schrittweite variieren. Dies entspricht der Suche von $\widetilde{\boldsymbol{w}} + \Delta\widetilde{\boldsymbol{w}}$ auf einem Kreis um $\widetilde{\boldsymbol{w}}$.

Für die Auswahl des bestmöglichen Verschiebungsvektors ist eine Bewertungsfunktion $b_{\widetilde{\boldsymbol{w}}} : \Delta\widetilde{\boldsymbol{w}} \in \mathcal{U} \mapsto \mathbb{R}_0^+$ nötig. Die global optimale Bewertung

$$b_{\boldsymbol{w}}^{\boldsymbol{w}}(\Delta\widetilde{\boldsymbol{w}}) := |\widetilde{\boldsymbol{w}} + \Delta\widetilde{\boldsymbol{w}} - \boldsymbol{w}| \tag{9.5a}$$

kann aber nicht verwendet werden, da vom Vektor $\boldsymbol{w}$ nur bekannt ist, dass er sich irgendwo auf der Hyperebene $\mathcal{L}_{\boldsymbol{e},W}$ befindet. Somit bleibt als Bewertungsmaßstab nur die lokal optimale Bewertungsfunktion

$$b_{\widetilde{\boldsymbol{w}}}^{\mathcal{L}_{\boldsymbol{e},W}}(\Delta\widetilde{\boldsymbol{w}}) := \min_{\boldsymbol{v}\in\mathcal{L}_{\boldsymbol{e},W}} |\widetilde{\boldsymbol{w}} + \Delta\widetilde{\boldsymbol{w}} - \boldsymbol{v}|, \tag{9.5b}$$

also der Abstand von $\widetilde{\boldsymbol{w}} + \Delta\widetilde{\boldsymbol{w}}$ von der Hyperebene $\mathcal{L}_{\boldsymbol{e},W}$ übrig.

Der in Abbildung 9.1 gezeigte Verschiebungsvektor $\Delta\widetilde{\boldsymbol{w}}^1$ wäre in diesem Fall die global optimale Wahl. Aufgrund der lokalen Bewertungsfunktion lässt er sich aber nicht von $\Delta\widetilde{\boldsymbol{w}}^2$ unterscheiden, welcher aus globaler Sicht einen Verschlechterungsschritt darstellt.

Die lokale Bewertungsfunktion wird minimal für alle $\Delta\widetilde{\boldsymbol{w}}$, die zu einer Verschiebung der Verbrauchsannahme auf die Hyperebene $\mathcal{L}_{\boldsymbol{e},W}$ führen, denn

$$\begin{aligned}
& b_{\widetilde{\boldsymbol{w}}}^{\mathcal{L}_{\boldsymbol{e},W}}(\Delta\widetilde{\boldsymbol{w}}) = \min_{\boldsymbol{v}\in\mathcal{L}_{\boldsymbol{e},W}} |\widetilde{\boldsymbol{w}} + \Delta\widetilde{\boldsymbol{w}} - \boldsymbol{v}| = 0 \\
\Leftrightarrow \quad & \exists\, \boldsymbol{v} \in \mathcal{L}_{\boldsymbol{e},W} : \quad |\widetilde{\boldsymbol{w}} + \Delta\widetilde{\boldsymbol{w}} - \boldsymbol{v}| = 0 \\
\Leftrightarrow \quad & \exists\, \boldsymbol{v} \in \mathcal{L}_{\boldsymbol{e},W} : \quad \widetilde{\boldsymbol{w}} + \Delta\widetilde{\boldsymbol{w}} = \boldsymbol{v} \\
\Leftrightarrow \quad & \widetilde{\boldsymbol{w}} + \Delta\widetilde{\boldsymbol{w}} \in \mathcal{L}_{\boldsymbol{e},W}.
\end{aligned}$$

Wie oben argumentiert, können aber manche dieser lokal optimalen Verschiebungen auch zu globalen Verschlechterungen, also zu einer Vergrößerung des Abstandes zum tatsächlichen Verbrauchsvektor $\boldsymbol{w}$ führen. Das folgende Lemma schließt dies für einen bestimmten Verschiebungsvektor

$$\Delta\widetilde{\boldsymbol{w}}_L(\boldsymbol{e}, W, \widetilde{\boldsymbol{w}}) := \boldsymbol{e}\frac{W - \boldsymbol{e}\cdot\widetilde{\boldsymbol{w}}}{\boldsymbol{e}\cdot\boldsymbol{e}} \tag{9.6}$$

aus.

Lemma 9.1.

$$\widetilde{\boldsymbol{w}} \notin \mathcal{L}_{\boldsymbol{e},W} \quad \wedge \quad \Delta\widetilde{\boldsymbol{w}} = \Delta\widetilde{\boldsymbol{w}}_L(\boldsymbol{e}, W, \widetilde{\boldsymbol{w}}) \quad \Rightarrow \quad \widetilde{\boldsymbol{w}} + \Delta\widetilde{\boldsymbol{w}} \in \mathcal{L}_{\boldsymbol{e},W} \quad \wedge \quad |\widetilde{\boldsymbol{w}} + \Delta\widetilde{\boldsymbol{w}} - \boldsymbol{w}| < |\widetilde{\boldsymbol{w}} - \boldsymbol{w}|$$

Beweis. Zunächst ist

$$\begin{aligned}
\boldsymbol{e}\cdot(\widetilde{\boldsymbol{w}}+\Delta\widetilde{\boldsymbol{w}}) &\overset{\text{Def. Skalarprodukt}}{=} \sum_{i=0}^{s} e_i\left(\widetilde{w}_i + e_i\frac{W-\boldsymbol{e}\cdot\widetilde{\boldsymbol{w}}}{\boldsymbol{e}\cdot\boldsymbol{e}}\right)\\
&= \sum_{i=0}^{s} e_i\widetilde{w}_i + \frac{W-\boldsymbol{e}\cdot\widetilde{\boldsymbol{w}}}{\boldsymbol{e}\cdot\boldsymbol{e}}\sum_{i=0}^{s} e_ie_i\\
&\overset{\text{Def. Skalarprodukt}}{=} \boldsymbol{e}\cdot\widetilde{\boldsymbol{w}} + \frac{W-\boldsymbol{e}\cdot\widetilde{\boldsymbol{w}}}{\boldsymbol{e}\cdot\boldsymbol{e}}\boldsymbol{e}\cdot\boldsymbol{e} = \boldsymbol{e}\cdot\widetilde{\boldsymbol{w}} + (W-\boldsymbol{e}\cdot\widetilde{\boldsymbol{w}}) = W
\end{aligned}$$

und damit $\widetilde{\boldsymbol{w}}+\Delta\widetilde{\boldsymbol{w}} \in \mathcal{L}_{\boldsymbol{e},W}$ nach Ausdruck (9.2). Wegen

$$\begin{aligned}
&|\widetilde{\boldsymbol{w}}+\Delta\widetilde{\boldsymbol{w}}-\boldsymbol{w}| < |\widetilde{\boldsymbol{w}}-\boldsymbol{w}|\\
\overset{\text{Def. Norm}}{\Leftrightarrow}\quad & \sqrt{\sum_{i=0}^{s}(\widetilde{w}_i+\Delta\widetilde{w}_i-w_i)^2} < \sqrt{\sum_{i=0}^{s}(\widetilde{w}_i-w_i)^2}\\
\overset{\text{Monotonie der Wurzel}}{\Leftrightarrow}\quad & \sum_{i=0}^{s}(\widetilde{w}_i+\Delta\widetilde{w}_i-w_i)^2 < \sum_{i=0}^{s}(\widetilde{w}_i-w_i)^2\\
\overset{\text{Binom}}{\Leftrightarrow}\quad & \sum_{i=0}^{s}(\widetilde{w}_i-w_i)^2 + \sum_{i=0}^{s}2\Delta\widetilde{w}_i(\widetilde{w}_i-w_i) + \sum_{i=0}^{s}\Delta\widetilde{w}_i^2 < \sum_{i=0}^{s}(\widetilde{w}_i-w_i)^2\\
\overset{\text{Def. }\Delta\boldsymbol{w}}{\Leftrightarrow}\quad & \sum_{i=0}^{s}2e_i\frac{W-\boldsymbol{e}\cdot\widetilde{\boldsymbol{w}}}{\boldsymbol{e}\cdot\boldsymbol{e}}(\widetilde{w}_i-w_i) + \sum_{i=0}^{s}\left(e_i\frac{W-\boldsymbol{e}\cdot\widetilde{\boldsymbol{w}}}{\boldsymbol{e}\cdot\boldsymbol{e}}\right)^2 < 0\\
\Leftrightarrow\quad & 2\frac{W-\boldsymbol{e}\cdot\widetilde{\boldsymbol{w}}}{\boldsymbol{e}\cdot\boldsymbol{e}}\left(\sum_{i=0}^{s}e_i\widetilde{w}_i - \sum_{i=0}^{s}e_iw_i\right) + \left(\frac{W-\boldsymbol{e}\cdot\widetilde{\boldsymbol{w}}}{\boldsymbol{e}\cdot\boldsymbol{e}}\right)^2\sum_{i=0}^{s}e_ie_i < 0\\
\overset{\text{Def. Skalarprodukt}}{\Leftrightarrow}\quad & 2\frac{W-\boldsymbol{e}\cdot\widetilde{\boldsymbol{w}}}{\boldsymbol{e}\cdot\boldsymbol{e}}(\boldsymbol{e}\cdot\widetilde{\boldsymbol{w}}-\boldsymbol{e}\cdot\boldsymbol{w}) + \left(\frac{W-\boldsymbol{e}\cdot\widetilde{\boldsymbol{w}}}{\boldsymbol{e}\cdot\boldsymbol{e}}\right)^2(\boldsymbol{e}\cdot\boldsymbol{e}) < 0\\
\overset{\boldsymbol{e}\cdot\boldsymbol{e}>0}{\Leftrightarrow}\quad & 2(W-\boldsymbol{e}\cdot\widetilde{\boldsymbol{w}})(\boldsymbol{e}\cdot\widetilde{\boldsymbol{w}}-\boldsymbol{e}\cdot\boldsymbol{w}) + (W-\boldsymbol{e}\cdot\widetilde{\boldsymbol{w}})^2 < 0\\
\overset{(9.1)}{\Leftrightarrow}\quad & 2(W-\boldsymbol{e}\cdot\widetilde{\boldsymbol{w}})(\boldsymbol{e}\cdot\widetilde{\boldsymbol{w}}-W) + (W-\boldsymbol{e}\cdot\widetilde{\boldsymbol{w}})^2 < 0\\
\Leftrightarrow\quad & -2(W-\boldsymbol{e}\cdot\widetilde{\boldsymbol{w}})^2 + (W-\boldsymbol{e}\cdot\widetilde{\boldsymbol{w}})^2 < 0\\
\Leftrightarrow\quad & (W-\boldsymbol{e}\cdot\widetilde{\boldsymbol{w}})^2 > 0\\
\Leftrightarrow\quad & W \neq \boldsymbol{e}\cdot\widetilde{\boldsymbol{w}}\\
\overset{(9.2)}{\Leftrightarrow}\quad & \widetilde{\boldsymbol{w}} \notin \mathcal{L}_{\boldsymbol{e},W}
\end{aligned}$$

folgt auch der zweite Teil der Behauptung. □

Die definierende Gleichung $W = \boldsymbol{e}\cdot\widetilde{\boldsymbol{w}}$ von $\mathcal{L}_{\boldsymbol{e},W}$ ist eine Ebenengleichung in Normalenform und so haben der Normalenvektor der Ebene und der Verschiebungsvektor $\Delta\widetilde{\boldsymbol{w}}_L(\boldsymbol{e}^j, W^j, \widetilde{\boldsymbol{w}}^j)$ beide

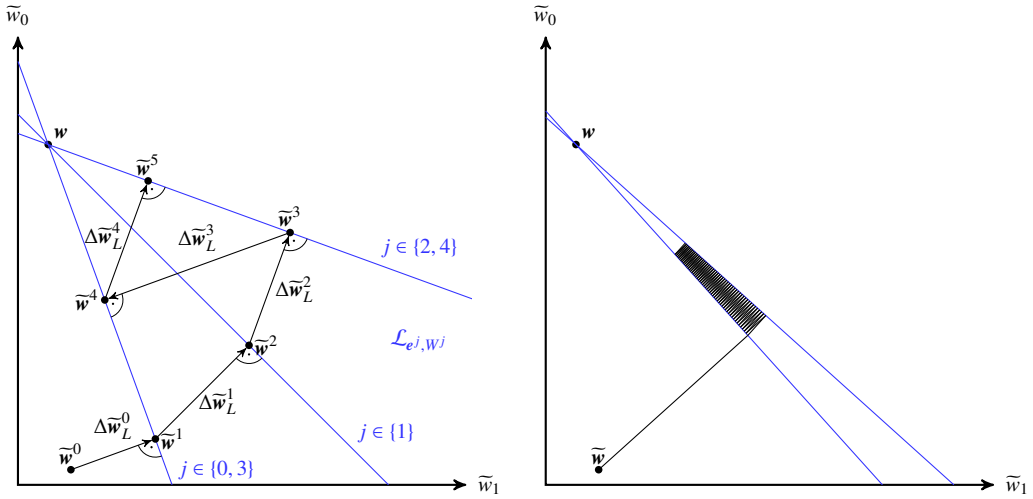


Abbildung 9.2: Annäherung der Verbrauchsannahmen $\widetilde{w}^j$ an den tatsächlichen Verbrauchsvektor w durch Fällen des Lots $\Delta\widetilde{w}_L^j = \Delta\widetilde{w}_L(e^j, W^j, \widetilde{w}^j)$ auf die lokal optimale Hyperebene $\mathcal{L}_{e^j,W^j}$

dieselbe Richtung e. Die Wahl dieses Verschiebungsvektors entspricht also dem Fällen des Lots durch $\widetilde{w}$ auf die Hyperebene. Abbildung 9.2a zeigt die Annäherung von $\widetilde{w}$ an w durch sukzessive Anwendung der Lotverschiebung über mehrere Hauptslots ($0 \leq j \leq 4$), also $\widetilde{w}^{j+1} = \widetilde{w}^j + \Delta\widetilde{w}_L(e^j, W^j, \widetilde{w}^j)$.

Aus Lemma 9.1 folgt auch, dass die Folge $(|\widetilde{w}^j - w|)_{j\in\mathbb{N}}$ monoton fällt. Wegen ihrer Beschränkung auf nicht-negative Werte muss sie daher auch konvergieren. Ob es sich dabei allerdings um eine Nullfolge handelt und $(\widetilde{w}^j)_{j\in\mathbb{N}}$ damit gegen w konvergiert, hängt von der Folge der Ausführungsvektoren ab. Kleine Änderungen der Ausführungsvektoren führen auch nur zu kurzen Verschiebungsvektoren (vgl. Abbildung 9.2b) und letztlich begrenzt die Festkommaarithmetik zur Berechnung der Verbrauchsvektoren die Genauigkeit der Darstellung.

Auf der anderen Seite sorgt die Lotverschiebung in solchen Phasen konstanter Ausführungsvektoren dafür, dass der Gesamtverbrauch des Sensorknotens richtig gesteuert wird, denn

$$\begin{aligned} e^j = e^{j+1} &\overset{(9.1)}{\Rightarrow} W^j = W^{j+1} \\ &\Rightarrow \widetilde{w}^{j+1} = \widetilde{w}^j + \Delta\widetilde{w}_L(e^j, W^j, \widetilde{w}^j) \overset{\text{Lemma 9.1}}{\in} \mathcal{L}_{e^j,W^j} = \mathcal{L}_{e^{j+1},W^{j+1}} \\ &\Rightarrow \widehat{W}^{j+1} \overset{\text{Lemma 4.2}}{=} e^{j+1} \cdot \widetilde{w}^{j+1} \overset{(9.2)}{=} W^{j+1}. \end{aligned}$$

9.3 Gauß-Elimination mit partieller Pivotisierung

Die im letzten Abschnitt vorgestellte Lernstrategie sorgt für minimale Abweichungen zwischen angenommenem und tatsächlichem Gesamtverbrauch des Sensorknotens, solange dieser an seiner oberen oder unteren Verbrauchsgrenze betrieben wird. Beim Übergang zwischen diesen beiden Phasen, also beim Erhöhen oder Verringern des Gesamtverbrauchs entstehen aber Fehleinschätzungen $W - \widetilde{W} = W - \boldsymbol{e} \cdot \widetilde{\boldsymbol{w}}$, die zur Annäherung von $\widetilde{\boldsymbol{w}}$ an $\boldsymbol{w}$ genutzt werden (vgl. Lemma 9.1). Nach Ausdruck (9.6) ist dabei die Annäherungskomponente $\Delta\widetilde{w}_i$ proportional zur Ausführungshäufigkeit e_i des entsprechenden Tasks t_i. Fehler in den Verbrauchsannahmen selten ausgeführter Tasks, wie etwa t_0, können daher auch nur langsam ausgeglichen werden.

Aus diesem Grund ist eine weitere Lernstrategie notwendig, welche in den Übergangsphasen mit veränderlichen Ausführungsvektoren schneller zur Verbesserung der Verbrauchsannahmen führt. Im Folgenden wird daher eine Implementierung des Gauß-Eliminationsverfahrens zum Lösen des linearen Gleichungssystems

$$\mathbf{E} \cdot \boldsymbol{x} = \begin{pmatrix} \boldsymbol{e}^0 \\ \vdots \\ \boldsymbol{e}^g \end{pmatrix} \cdot \boldsymbol{x} = \begin{pmatrix} e_0^0 & \dots & e_s^0 \\ \vdots & & \vdots \\ e_0^g & \dots & e_s^g \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ \vdots \\ x_s \end{pmatrix} = \begin{pmatrix} W^0 \\ \vdots \\ W^g \end{pmatrix} = \boldsymbol{W} \tag{9.7}$$

vorgestellt.

Die Ausführungsvektoren $\boldsymbol{e}^1, \dots, \boldsymbol{e}^g$ und Verbrauchsmessungen $W^1, \dots, W^g$ der letzten g Hauptslots werden in Schieberegistern verwaltet, in welche die Werte $\boldsymbol{e}^0 := \boldsymbol{e}$ und $W^0 := W$ des aktuellen Hauptslots eingespeist werden. Die Größe g der Schieberegister wird über die Compilezeitkonstante `RA_ESTIMATOR_GE_SIZE` festgelegt. Da das zu lösende System aus $g + 1$ Gleichungen besteht, muss dabei zwischen Speicheraufwand und Lernerfolg abgewogen werden. So wird für jede Gleichung 6s+10 Byte zusätzlicher Speicher alloziert:

- 6(s+1) Byte für einen Eintrag im Spaltenvektor (`ra_scalable_t.ge`, vgl. Abbildung 4.6)
- 4 Byte für einen Eintrag im Schieberegister der Verbrauchsvektoren

Auf der anderen Seite steigt mit g im Allgemeinen die Anzahl der Verbrauchskomponenten, welche durch das Gleichungssystem eindeutig bestimmt werden können. Wenn die Speicherbelegung für die Anwendung nicht kritisch ist, sollte daher $g = s$ verwendet werden.

9.3.1 Prinzip des Eliminationsverfahrens

Für eine einfachere Beschreibung wird das Gleichungssystem (9.7) ohne die Unbekannten $\boldsymbol{x}$ als erweiterte Koeffizientenmatrix

$$\mathbf{M} := \left(\begin{array}{ccc|c} \alpha_{0,0} & \dots & \alpha_{0,s} & \beta_0 \\ \vdots & & \vdots & \vdots \\ \alpha_{g,0} & \dots & \alpha_{g,s} & \beta_g \end{array}\right) := \left(\mathbf{E} \mid \boldsymbol{W}\right) \tag{9.8}$$

Tabelle 9.1: die Lösungsgesamtheit des Gleichungssystems erhaltende Transformationen

Zeilentausch	$Z_j \leftrightarrow Z_k$	$\beta_j \leftrightarrow \beta_k, \alpha_{j,i} \leftrightarrow \alpha_{k,i}$
Zeilenskalierung	$Z_j \leftarrow \lambda Z_j, \lambda \neq 0$	$\beta_j \leftarrow \lambda\beta_j, \alpha_{j,i} \leftarrow \lambda\alpha_{j,i}\ \forall\ i \in \{0,\ldots,s\}$
Zeilenaddition	$Z_j \leftarrow Z_j + \omega Z_k$	$\beta_j \leftarrow \beta_j + \omega\beta_k,$ $\alpha_{j,i} \leftarrow \alpha_{j,i} + \omega\alpha_{k,i}\ \forall\ i \in \{0,\ldots,s\}$

dargestellt. Mittels elementarer Zeilentransformationen aus Tabelle 9.1 können die Koeffizienten dieser Matrix gezielt manipuliert werden, ohne die Lösungsmenge des Gleichungssystems zu verändern [45].

Die Gauß-Elimination erfolgt nach Algorithmus 9.2 in zwei Phasen. Während der Vorwärts-

Algorithmus 9.2 : Gauß-Elimination

Eingabe : erweiterte Koeffizientenmatrix $\mathbf{M}$ für das lineare Gleichungssystem $\mathbf{E} \cdot \boldsymbol{x} = \boldsymbol{W}$
Ausgabe : $\forall\ i \in \{0,\ldots,s\}$: Lösung x_i oder Hinweis, dass x_i durch das Gleichungssystem nicht eindeutig bestimmt wird

$j \leftarrow 0$;
für $i \leftarrow 0,\ldots,s$ // Vorwärtselimination
  $\mathcal{P} \leftarrow \{\alpha_{k,i} \neq 0, j \leq k \leq g\}$; // potentielle Pivots
  wenn $\mathcal{P} \neq \emptyset$ **dann**
    Wähle Pivot $p \leftarrow \alpha_{k,i} \in \mathcal{P}$ mit größtem Betrag ; // partielle Pivotisierung
    $Z_k \leftrightarrow Z_j$;
    für $k \leftarrow j+1,\ldots,g$
      $Z_k \leftarrow Z_k - \frac{\alpha_{k,i}}{p} Z_j$; // Elimination von $\alpha_{k,i}$
    $j \leftarrow j+1$;
    wenn $j = g$ **dann** beende Vorwärtselimination
Markiere alle Unbekannten x_i als unbestimmt;
für $j \leftarrow g,\ldots,0$ // Rückwärtseinsetzen
  wenn *Zeile* j *enthält Pivot* $\alpha_{j,i}$ **dann**
    wenn $\nexists\ k > i : \alpha_{j,k} \neq 0$ *und* x_k *unbestimmt* **dann**
      Markiere x_i als bestimmt mit $x_i \leftarrow \frac{1}{\alpha_{j,i}} \left(\beta_j - \sum_{k=i+1}^{s} \alpha_{j,k} x_k\right)$;
  sonst
    wenn $\beta_j \neq 0$ **dann** // inkonsistentes System
      Markiere alle Unbekannten x_i als unbestimmt;
      terminiere mit Hinweis auf inkonsistentes Gleichungssystem;

elimination (erste äußere Zählschleife) wird $\mathbf{M}$ mittels elementarer Transformationen in eine Stufenform überführt. Dazu wird in jeder Spalte i zunächst ein von Null verschiedener Koeffizient $\alpha_{k,i}$ gesucht, mit dessen Hilfe die verbleibenden Koeffizienten der Spalte eliminiert werden können. Dieser Koeffizient wird als Pivot p der Spalte i und die Beschränkung seiner Auswahl auf die aktuelle Spalte als partielle Pivotisierung bezeichnet. Nach [39] sollte dabei der betragsgrößte Koeffizient gewählt werden, um den Einfluss von Rundungsfehlern zu minimieren. Um eine Stufenform zu erreichen, muss die Zeile k mit der Zeile j unterhalb des letzten gefundenen Pivots der vorherigen Spalten vertauscht werden. Anschließend werden alle Spaltenkoeffizienten $\alpha_{k,i}$ unter p durch eine Zeilenaddition eliminiert:

$$
\left(\begin{array}{ccccccc|c}
\vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots \\
0 & \dots & 0 & p & \alpha_{j,i+1} & \dots & \alpha_{j,s} & \beta_j \\
\vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots \\
0 & \dots & 0 & \alpha_{k,i} & \alpha_{k,i+1} & \dots & \alpha_{k,s} & \beta_k \\
\vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots
\end{array}\right)
$$

$$
\xrightarrow{Z_k \leftarrow Z_k - \frac{\alpha_{k,i}}{p} Z_j}
\left(\begin{array}{ccccccc|c}
\vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots \\
0 & \dots & 0 & p & \alpha_{j,i+1} & \dots & \alpha_{j,s} & \beta_j \\
\vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots \\
0 & \dots & 0 & \alpha_{k,i} - \frac{\alpha_{k,i}}{p}\alpha_{j,i} & \alpha_{k,i+1} - \frac{\alpha_{k,i}}{p}\alpha_{j,i+1} & \dots & \alpha_{k,s} - \frac{\alpha_{k,i}}{p}\alpha_{j,s} & \beta_k - \frac{\alpha_{k,i}}{p}\beta_j \\
\vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots
\end{array}\right)
$$

$$
\overset{p = \alpha_{j,i}}{=}
\left(\begin{array}{ccccccc|c}
\vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots \\
0 & \dots & 0 & p & \alpha_{j,i+1} & \dots & \alpha_{j,s} & \beta_j \\
\vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots \\
0 & \dots & 0 & 0 & \alpha_{k,i+1} - \frac{\alpha_{k,i}}{p}\alpha_{j,i+1} & \dots & \alpha_{k,s} - \frac{\alpha_{k,i}}{p}\alpha_{j,s} & \beta_k - \frac{\alpha_{k,i}}{p}\beta_j \\
\vdots & & \vdots & \vdots & \vdots & & \vdots & \vdots
\end{array}\right)
$$

Enthält die Spalte i ab Zeile j ausschließlich Null Elemente, dann kann kein Pivotelement gewählt und später auch nicht nach x_i aufgelöst werden. Nach der Vorwärtselimination kann die erweiterte Koeffizientenmatrix für $s = g = 4$ bspw. die Form

$$
\mathbf{M}_a = \left(\begin{array}{ccccc|c}
\alpha_{0,0} & \alpha_{0,1} & \alpha_{0,2} & \alpha_{0,3} & \alpha_{0,4} & \beta_0 \\
0 & \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \beta_1 \\
0 & 0 & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \beta_2 \\
0 & 0 & 0 & \alpha_{3,3} & \alpha_{3,4} & \beta_3 \\
0 & 0 & 0 & 0 & \alpha_{4,4} & \beta_4
\end{array}\right)
\quad \text{oder} \quad
\mathbf{M}_b = \left(\begin{array}{ccccc|c}
\alpha_{0,0} & \alpha_{0,1} & \alpha_{0,2} & \alpha_{0,3} & \alpha_{0,4} & \beta_0 \\
0 & \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \beta_1 \\
0 & 0 & 0 & \alpha_{2,3} & \alpha_{2,4} & \beta_2 \\
0 & 0 & 0 & 0 & \alpha_{3,4} & \beta_3 \\
0 & 0 & 0 & 0 & 0 & \beta_4
\end{array}\right)
$$

haben. Diese Stufenformen werden nun in der zweiten Phase der Gauß-Elimination durch Rückwärtseinsetzen (zweite äußere Zählschleife in Algorithmus 9.2) nach möglichst vielen Unbekannten aufgelöst. Zeilen j ohne Pivotelement, wie etwa die letzte Zeile in $\mathbf{M}_b$ entscheiden über die Konsistenz des Gleichungssystems. Wurde mit den Koeffizienten der linken Seite nicht auch die rechte Seite β_j eliminiert, so resultiert eine unerfüllbare Aussage $0 = \beta_j \neq 0$ und es gibt keinen Vektor $\boldsymbol{x}$, der (9.7) erfüllt. Alle Zwischenergebnisse der Gauß-Elimination werden daher verworfen. Spalten i ohne Pivotelement, wie etwa Spalte 2 in $\mathbf{M}_b$ repräsentieren freie Variablen x_i, nach denen nicht aufgelöst werden kann. Ebenso nicht bestimmbar sind Variablen, die von anderen unbestimmten Variablen abhängig sind.

9.3.2 Reduktion des Speicherbedarfs

Da die Ausführungshäufigkeit des imaginären Tasks t_0 in allen Hauptslots gleich ist, kann der erste Eliminationsschritt

$$\left(\begin{array}{cccc|c} 1 & e_1^0 & \dots & e_s^0 & W^0 \\ 1 & e_1^1 & \dots & e_s^1 & W^1 \\ \vdots & \vdots & & \vdots & \vdots \\ 1 & e_1^{g-1} & \dots & e_s^{g-1} & W^s \end{array}\right) \xrightarrow{Z_k \leftarrow Z_k - Z_0 \;\forall\, k>0} \left(\begin{array}{cccc|c} 1 & e_1^0 & \dots & e_s^0 & W^0 \\ 0 & e_1^1 - e_1^0 & \dots & e_s^1 - e_s^0 & W^1 - W^0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & e_1^g - e_1^0 & \dots & e_s^g - e_s^0 & W^s - W^0 \end{array}\right)$$

mit $\alpha_{0,0}$ als Pivotelement in die Initialisierung der erweiterten Koeffizientenmatrix aus den Schieberegistern integriert werden.

Die erste Zeile und Spalte wird während der Vorwärtselimination nun nicht mehr verändert. Wendet man die Gauß-Elimination auf die verbleibende $g \times (s+1)$ Matrix an und können bei deren Rückwärtseinsetzung alle Variablen $x_1, \dots, x_s$ bestimmt werden, dann kann auch

$$x_0 = W^0 - \sum_{i=1}^{s} e_i^0 x_i = W - \sum_{i=1}^{s} e_i x_i \tag{9.9}$$

berechnet werden. Damit ist die Speicherallokation für die $s + g + 2$ Koeffizienten β_0 und $\alpha_{0,0}, \dots, \alpha_{g,0}, \alpha_{0,1}, \dots, \alpha_{0,s}$ überflüssig.

9.3.3 Rationale Arithmetik

Iterative numerische Verfahren, wie die Gauß-Elimination, reagieren besonders empfindlich auf Rundungsfehler, da sich diese aufsummieren können. So wirkt sich die Ungenauigkeit eines Pivotkoeffizienten im Eliminationsschritt auf viele weitere Koeffizienten aus. Die partielle Pivotisierung erhöht zwar die numerische Stabilität des Verfahrens, da die Implementierung aber auf Gleitkommaarithmetik verzichten muss (vgl. Abschnitt 4.6) und mit der alternativen Festkommaarithmetik Rundungsfehler noch weniger kontrolliert werden können, basiert die hier vorgestellte Gauß-Elimination auf einer rationalen Arithmetik. Die Koeffizienten werden dabei als

32 bit Ganzzahlen dargestellt und Rundungsfehler bei Divisionen durch die Verwendung echter Teiler der Dividenden ausgeschlossen. Das wichtigste Werkzeug hierfür ist die Berechnung des größten gemeinsamen Teilers (ggT) ganzer Zahlen, für den nach [18] die Beobachtungen

$$\begin{aligned}
\mathrm{ggT}(a,1) &= 1 && \text{(9.10a)}\\
\mathrm{ggT}(a,0) &= |a| && \text{(9.10b)}\\
\mathrm{ggT}(a,a) &= |a| && \text{(9.10c)}\\
\mathrm{ggT}(a,b) &= \mathrm{ggT}(b,a) && \text{(9.10d)}\\
\mathrm{ggT}(a,b) &= \mathrm{ggT}(-a,b) && \text{(9.10e)}\\
\mathrm{ggT}(a,b) &= \mathrm{ggT}(b,a-qb)\ \forall\ q\in\mathbb{Z} && \text{(9.10f)}\\
\mathrm{ggT}(2a,2b) &= \mathrm{ggT}(a,b)\cdot 2 && \text{(9.10g)}\\
\mathrm{ggT}(2a,2b+1) &= \mathrm{ggT}(a,2b+1) && \text{(9.10h)}\\
\mathrm{ggT}(a,b,c) &= \mathrm{ggT}(\mathrm{ggT}(a,b),c) && \text{(9.10i)}
\end{aligned}$$

gelten.

Darauf basieren verschiedene iterative bzw. rekursive *ggT*-Schemata mit unterschiedlichen Laufzeiteigenschaften. So ist die Rekursionstiefe beim Euklidischen Algorithmus

$$\mathrm{ggT}(a,b) \leftarrow a = 0\ ?\ b : \mathrm{ggT}(b, a \bmod b) \qquad (9.11)$$

Algorithmus 9.3 : größter gemeinsamer Teiler

Eingabe : $a, b \in \mathbb{N}$
Ausgabe : $\mathrm{ggT}(a,b)$

Euklid
wenn $a < b$ **dann** $a \leftrightarrow b$; // (9.10d)
wenn $b = 0$ **dann** beende mit $\mathrm{ggT} \leftarrow a$; // (9.10b)
$a \leftarrow a \bmod b$; // (9.10f) mit $q = \left\lfloor \frac{a}{b} \right\rfloor$
wenn $a = 0$ **dann** beende mit $\mathrm{ggT} \leftarrow b$; // (9.10b)

Stein
$k \leftarrow 0$;
solange *a und b gerade* // (9.10g)
  halbiere a und b;
  $k \leftarrow k + 1$;
solange $a \neq b$ // (9.10c)
  wenn *a gerade* **dann** halbiere a ; // (9.10h)
  sonst wenn *b gerade* **dann** halbiere b ; // (9.10h)
  sonst wenn $a > b$ **dann** $a \leftarrow a - b$; // (9.10f) mit $q = 1$
  sonst $b \leftarrow b - a$; // (9.10f) mit $q = 1$
$\mathrm{ggT} \leftarrow a \cdot 2^k$; // (9.10g)

durch den Betrag des kleineren Arguments beschränkt [18]. Die Modulo-Operation muss auf Mikrocontrollern ohne Hardwareunterstützung für Divisionen per Software emuliert werden und ist daher so teuer, dass der binäre ggT-Algorithmus nach Stein [18, 38] vorzuziehen ist. Da dessen Laufzeit aber durch den Betrag des größeren Arguments beschränkt ist, kombiniert Algorithmus 9.3 beide Schemata.

Zunächst wird der erste Iterationsschritt nach Euklid ausgeführt, was eine einzelne Modulooperation und den Test der Parameter auf Null beinhaltet. Wenn der Algorithmus dabei nicht bereits terminiert, dann sind die beiden Parameter a und b und damit auch die Laufzeit des anschließenden Schemas nach Stein durch den ursprünglich kleineren Parameter beschränkt. Der binäre ggT-Algorithmus beruht auf dem Halbieren ganzer Zahlen (konstanter Rechstshift um eine Position) und der Prüfung ihrer Parität (Test des LSB). Dies sind effiziente Operationen, da sie im Instruktionssatz der MSP430 CPU enthalten sind [44]. Auch die abschließende Multiplikation mit 2^k entspricht einem Linksshift um k Stellen und ist daher effizient ausführbar. Die iterative Form garantiert außerdem einen konstanten Speicherbedarf. Wegen (9.10e) ist Algorithmus 9.3 nach Betragsbildung auch auf ganze Zahlen anwendbar.

Zum Einsatz kommt die rationale Arithmetik bei der Elimination des Spaltenkoeffizienten $\alpha_{k,i}$ unterhalb des Pivots $p = \alpha_{j,i}$. Die im Algorithmus 9.2 hierfür vorgesehene Zeilentransformation $Z_k \leftarrow Z_k - \frac{\alpha_{k,i}}{p} Z_j$ ist mit Rundungsfehlern behaftet, wenn $\alpha_{k,i}$ nicht ohne Rest durch p teilbar ist. Stattdessen erfolgt die Elimination mit Hilfe der beiden Transformationen $Z_k \leftarrow p\, Z_k$ und $Z_k \leftarrow Z_k - \alpha_{k,i}\, Z_j$. Dabei müssen die Spalten links des Pivots nicht mehr betrachtet werden, weil durch die vorherigen Eliminationen bereits $\alpha_{j,l} = \alpha_{k,l} = 0 \;\; \forall\, l < i$ gesichert ist. Es bleibt also

$$\beta_k \leftarrow p\,\beta_k - \alpha_{k,i}\,\beta_j \quad \text{und} \quad \alpha_{k,l} \leftarrow p\,\alpha_{k,l} - \alpha_{k,i}\,\alpha_{j,l} \quad \forall\, l \in \{i, \ldots, s\} \tag{9.12}$$

zu berechnen, wobei wegen $p = \alpha_{j,i}$ der Koeffizient $\alpha_{k,i} \leftarrow p\,\alpha_{k,i} - \alpha_{k,i}\,p = 0$ wie vorgesehen eliminiert wird. Bei der Berechnung der restlichen Koeffizienten der Zeile k entstehen wegen der fehlenden Divisionen keine Rundungsfehler.

Ein Koeffizientenüberlauf, also das Berechnen von Koeffizientenwerten außerhalb des darstellbaren Intervalls $\mathbb{Z}_{32} := \{-2^{31}, \ldots, 2^{31} - 1\} \subset \mathbb{Z}$ kann hingegen nicht ausgeschlossen werden. Daher muss durch eine abschließende Zeilenskalierung $Z_k \leftarrow \frac{1}{\lambda} Z_k$ mit einem geeigneten Divisor $\lambda \in \mathbb{N}^+$ die Zeile k soweit reduziert werden, dass alle verbleibenden Koeffizienten als 32 bit Ganzzahlen dargestellt werden können. Um auch hier Rundungsfehler zu vermeiden, wird zunächst

$$\lambda = \mathrm{ggT}(|\alpha_{k,i+1}|, \ldots, |\alpha_{k,s}|, |\beta_k|) \tag{9.13}$$

berechnet. Wegen (9.10a), (9.10b) und (9.10i) kann der größte gemeinsame Teiler mehrerer Koeffizienten durch die sukzessive Akkumulation $\lambda \leftarrow \mathrm{ggT}(\lambda, |\alpha_{k,l}|)$ bestimmt werden. Der Akkumulator wird mit $\lambda \leftarrow 0$ initialisiert und die Akkumulation bei $\lambda = 1$ abgebrochen. Erst wenn auch dieser Divisor nicht ausreicht, um alle Koeffizienten in den darstellbaren Bereich $\mathbb{Z}_{32}$ zu reduzieren, muss λ weiter erhöht werden. Dies führt dann unweigerlich zu Rundungsfehlern, da

es keinen größeren Divisor als (9.13) gibt, der alle Koeffizienten ohne Rest teilt. Da diese Überlaufkontrolle nur bei Koeffizienten mit großem Betrag notwendig wird, bleiben deren relative Rundungsfehler allerdings gering.

Für die Überlaufbehandlung werden temporär 64 bit Ganzzahlen benötigt. Die drei Einzeltransformationen der Elimination müssen daher koeffizientenweise vollständig ausgeführt werden, da nicht für jeden Koeffizienten ein solcher temporärer Speicher bereitgestellt werden kann. Da λ aber von den Zwischenergebnissen der ersten beiden Transformationen (9.12) aller Koeffizienten abhängt, wird ein Vorlauf zur Bestimmung von λ nach Algorithmus 9.4 benötigt, bei dem der neue Wert der Koeffizienten zwar temporär berechnet (Zeile 4 und 10), aber noch nicht in die Matrix **M** übernommen wird.

In diesem Vorlauf wird parallel zur ggT-Akkumulation in κ der für die Überlaufbehandlung mindestens notwendige Divisor λ bestimmt. Am Ende wird der Größere der beiden Werte verwendet (Zeile 13). Da κ mit jeder weiteren ggT-Berechnung höchstens kleiner wird, kann dessen Akkumulation auch abgebrochen werden, sobald ein von Null verschiedener Wert κ erreicht wurde, der nicht mehr größer als λ ist (Zeile 6 und 12). Die aufwendige ggT-Berechnung wird dadurch nicht häufiger ausgeführt als unbedingt notwendig. Die Sonderbehandlung beim Erkennen einer Nullzeile (alle Koeffizienten der linken Seite eliminiert) wird im nächsten Abschnitt behandelt.

Algorithmus 9.4 : Bestimmung des Reduktionsdivisors λ nach Elimination von $\alpha_{k,i}$

Eingabe : Indizes der Pivotspalte i, der Pivotzeile j und der Eliminationszeile $k > j$
Ausgabe : Reduktionsdivisor λ

```
1  λ ← 1 ;
2  κ ← 0 ;                                                  // ggT-Akkumulator
3  für l ← i + 1, ..., s
4  |  x ← |α_{j,i} α_{k,l} − α_{k,i} α_{j,l}| ;             // (9.12)
5  |  wenn x/λ ≥ 2^31 dann λ ← 2 · x/2^31 = x >> 30 ;       // Überlaufkontrolle
6  |  wenn κ = 0 oder κ > λ dann κ = ggT(κ, x) ;            // ggT-Akkumulation
7  wenn κ = 0 dann                                          // auf Nullzeile prüfen
8  |  λ ← |α_{j,i} α_{k,i}| ;                               // vgl. Abschnitt 9.3.4
9  sonst
10 |  x ← |α_{j,i} β_k − α_{k,i} β_j| ;                    // (9.12)
11 |  wenn x/λ ≥ 2^31 dann λ ← 2 · x/2^31 = x >> 30 ;       // Überlaufkontrolle
12 |  wenn κ > λ dann κ = ggT(κ, x) ;                       // ggT-Akkumulation
13 wenn κ > λ dann λ ← κ;
```

9.3.4 Interpretation der Lösungen

Eine weitere Adaption des im Abschnitt 9.3.1 beschriebenen Eliminationsverfahrens betrifft die Erkennung inkonsistenter Gleichungssysteme. Nach Algorithmus 9.2 muss für Nullzeilen k ($\alpha_{k,i} = 0 \; \forall \; i \in \{0, \ldots, s\}$) auch der Koeffizient β_k der rechten Seite verschwinden, damit das Gleichungssystem als konsistent betrachtet werden kann. Aufgrund einiger nicht idealer Eigenschaften bei der Bestimmung des Ausführungsvektors $\boldsymbol{e}$ und des Gesamtverbrauchs W, aus denen das Gleichungssystem generiert wird, ist der Test auf $\beta_k = 0$ ein zu scharfes Kriterium. So können sich die tatsächlichen Verbrauchswerte w_i eines Tasks t_i trotz der Durchschnittsbildung über die Länge eines Hauptslots von einem Slot zum nächsten mehr oder weniger stark unterscheiden. Ein Task t_i kann zum Zeitpunkt des Slotübergangs auch schon teilweise ausgeführt worden sein, ohne dass sich dieser Mehrverbrauch auf den Ausführungszähler e_i ausgewirkt hat. Des Weiteren führen die eingeschränkte Genauigkeit bei der Messung des Gesamtverbrauchs W und die nicht vollständig vermeidbaren Rundungsfehler bei der Vorwärtselimination zu weiteren Abweichungen vom ideal konsistenten Gleichungssystem. Aus diesen Gründen wird ein Toleranzwert `RA_ESTIMATOR_GE_CONSISTENCE_TOLERANCE` als Compilezeitkonstante spezifiziert und als Schwellwert für $|\beta_k|$ eines konsistenten Systems mit Nullzeile k interpretiert.

Würde man bei der Bestimmung des Reduktionsdivisors in Algorithmus 9.4 einfach den größten gemeinsamen Teiler aller Koeffizienten der Nullzeile ermitteln, so wäre $\lambda = |\beta_k|$. Damit würde der Koeffizient der rechten Seite in jedem Fall auf einen Wert innerhalb des Toleranzbereichs skaliert und inkonsistente Systeme könnten nicht mehr erkannt werden. Es ist daher eine geeignete Normierung bei der Reduktion von Nullzeilen notwendig.

Nullzeilen können ausschließlich nach einer Koeffizientenelimination entstehen, da in der initialen Matrix die erste Spalte mit Einsen besetzt ist. Sei also $\alpha_{j,i}$ der Pivotkoeffizient und $\alpha_{k,i}$ der zu eliminierende Koeffizient. Wegen der entstehenden Nullzeile k muss nach (9.12)

$$\alpha_{j,i}\,\alpha_{k,l} - \alpha_{k,i}\,\alpha_{j,l} = 0 \overset{\alpha_{j,i} \neq 0 \neq \alpha_{k,i}}{\Leftrightarrow} \alpha_{k,l} = \frac{\alpha_{k,i}}{\alpha_{j,i}}\,\alpha_{j,l} \quad \forall\, l \in \{i+1, \ldots, s\} \tag{9.14}$$

gelten. Wenn Zeile j im Laufe der Rückwärtseinsetzung aufgelöst werden kann, so ergibt dies die Lösung

$$x_i = \frac{1}{\alpha_{j,i}} \left(\beta_j - \sum_{l=i+1}^{s} \alpha_{j,l}\, x_l \right). \tag{9.15a}$$

Hätte man zur Auflösung stattdessen die Zeile k vor deren Elimination verwendet, so ergäbe dies bei inkonsistenten Systemen im Allgemeinen eine andere Lösung

$$x_i + \Delta x_i = \frac{1}{\alpha_{k,i}} \left(\beta_k - \sum_{l=i+1}^{s} \alpha_{k,l}\, x_l \right). \tag{9.15b}$$

Nach den ersten beiden Eliminationstransformationen hat die rechte Seite der Nullzeile somit die Form

$$\begin{aligned}
\beta_k &\leftarrow \alpha_{j,i}\,\beta_k - \alpha_{k,i}\,\beta_j \\
&\overset{(9.15a),\,(9.15b)}{=} \alpha_{j,i}\left(\alpha_{k,i}\,(x_i + \Delta x_i) + \sum_{l=i+1}^{s} \alpha_{k,l}\,x_l\right) - \alpha_{k,i}\left(\alpha_{j,i}\,x_i + \sum_{l=i+1}^{s} \alpha_{j,l}\,x_l\right) \\
&\overset{(9.14)}{=} \alpha_{j,i}\left(\alpha_{k,i}\,(x_i + \Delta x_i) + \frac{\alpha_{k,i}}{\alpha_{j,i}} \sum_{l=i+1}^{s} \alpha_{j,l}\,x_l\right) - \alpha_{k,i}\left(\alpha_{j,i}\,x_i + \sum_{l=i+1}^{s} \alpha_{j,l}\,x_l\right) \\
&= \alpha_{j,i}\,\alpha_{k,i}\,(x_i + \Delta x_i - x_i) + \alpha_{k,i} \sum_{l=i+1}^{s} \alpha_{j,l}\,x_l - \alpha_{j,l}\,x_l \\
&= \alpha_{j,i}\,\alpha_{k,i}\,\Delta x_i.
\end{aligned}$$

Nach der abschließenden Zeilenreduktion mit $\lambda = |\alpha_{j,i}\,\alpha_{k,i}|$ (vgl. Zeile 8 von Algorithmus 9.4) ist der Betrag der rechten Seite der Nullzeile k also ein geeignetes Maß für die Inkonsistenz des Systems. Entsteht die Nullzeile bereits bei der initialen Elimination (vgl. Abschnitt 9.3.2), so ist die explizite Normierung der rechten Seite wegen $\alpha_{j,i} = \alpha_{0,0} = 1$ und $\alpha_{k,i} = \alpha_{k,0} = 1$ nicht notwendig.

Mit `RA_ESTIMATOR_GE_CONSISTENCE_TOLERANCE` steigt nun also die Toleranz für die Unsicherheit Δx_i über die Genauigkeit der gefundenen Lösungen x_i. Um einzelne Ausreißer von systematischen Veränderungen in den Verbrauchskomponenten zu unterscheiden, wird x_i nicht direkt als $\widetilde{w}_i$ übernommen, sondern zunächst ein gleitender Mittelwert

$$\Omega_i^n := \frac{1}{n} \sum_{k=1}^{n} x_i^k \tag{9.16}$$

aus den sukzessive ermittelten Lösungen $x_i^1, \ldots, x_i^n$ aufeinanderfolgender Gauß-Eliminationen gebildet. Wegen

$$\Omega_i^{n+1} = \frac{1}{n+1} \sum_{k=1}^{n+1} x_i^k = \frac{1}{n+1}\left(x_i^{n+1} + \sum_{k=1}^{n} x_i^k\right) = \frac{1}{n+1}\left(x_i^{n+1} + n\,\Omega_i^n\right) \tag{9.17a}$$

kann eine neue Lösung x_i^{n+1} an den bestehenden Mittelwert Ω_i^n angehängt werden, ohne dass alle vorherigen Lösungen $x_i^1, \ldots, x_i^n$ gespeichert werden müssen. Lediglich der Zähler n muss neben dem eigentlichen Mittelwert Ω_i für jeden Task t_i verwaltet werden.

Eine Umformung von (9.17a) zu

$$\Omega_i^{n+1} - \Omega_i^n = \frac{x_i^{n+1} - \Omega_i^n}{n+1} \tag{9.17b}$$

Algorithmus 9.5 : gleitender Mittelwert über Lösungen sukzessiver Gauß-Eliminationen

Eingabe : neue Lösung x_i, bisheriger Mittelwert Ω_i, Anzahl n der zu Ω_i beitragenden Lösungen

Wirkung : Anpassung von Ω_i und n und ggf. auch $\widetilde{w}_i$

wenn $n = 0$ *oder* $\left|\frac{x_i - \Omega_i}{\Omega_i}\right| >$ *RA_ESTIMATOR_MA_THRESHOLD* **dann**

  $\Omega_i \leftarrow x_i$;

  $n \leftarrow 1$;

sonst

  $\Omega_i \leftarrow \frac{x_i + n\ \Omega_i}{n+1}$;

  $n \leftarrow n + 1$;

wenn $n \geq$ *RA_ESTIMATOR_MA_RELIABILITY* **dann** $\widetilde{w}_i \leftarrow \Omega_i$;

zeigt, dass Abweichungen neuer Lösungen vom bisherigen Mittelwert mit einem Dämpfungsfaktor $\frac{1}{n+1}$ auf die Mittelwertänderung übertragen werden und Ω_i^n mit steigendem n somit stabil gegen Ausreißer wird. Bei einer systematischen Änderung der Verbrauchskomponente w_i sorgt diese Stabilität allerdings dafür, dass sich Ω_i^n nur sehr langsam dem neuen Wert annähern kann. Aus diesem Grund wird die Mittelwertbildung mit $\Omega_i^1 \leftarrow x_i$ neu begonnen, wenn die relative Abweichung $\left|\frac{x_i - \Omega_i^n}{\Omega_i^n}\right|$ vom bisherigen Mittelwert den Schwellwert `RA_ESTIMATOR_MA_THRESHOLD` überschreitet.

Um nun Ausreißer von systematischen Änderungen zu unterscheiden, wird Ω_i^n erst als $\widetilde{w}_i$ übernommen, wenn ausreichend ($n \geq$ `RA_ESTIMATOR_MA_RELIABILITY`) Lösungen zum Mittelwert beitragen. Algorithmus 9.5 fasst die Verwaltung des gleitenden Mittelwerts zusammen.

9.4 Gesteuertes Lernen

Die in den Abschnitten 9.2 und 9.3 vorgestellten Lernstrategien beobachten die Entwicklung der Ausführungsvektoren $\boldsymbol{e}$ und die daraus resultierenden Verbrauchswerte W und versuchen daraus Schlüsse über die einzelnen Verbrauchskomponenten $\boldsymbol{w}$ zu ziehen. Sie sind in diesem Sinne passiv, da sie den Arbeitspunkt $\boldsymbol{c}$ und damit den Ausführungsvektor $\boldsymbol{e}$ nicht beeinflussen. Aktive Lernstrategien haben dagegen das höhere Potential der gezielten Beeinflussung des Verhaltens des Sensorknotens und können dabei die Rahmenbedingungen für die Ermittelung von $\boldsymbol{w}$ optimieren.

Genau dies geschieht beim gesteuerten Lernen. Der *Estimator* übernimmt für $s + 1$ Hauptslots die Kontrolle über den Arbeitspunkt, was durch das Flag `ra_slot.directedUpdate` (siehe Abbildung 4.5) angezeigt wird. Während dieser Zeit wird das *Adaptive Duty Cycling* deaktiviert, was auch die Änderung des Nutzens skalierbarer Tasks und die damit verbundene Auslösung

von Subslots einschließt. Während der $s+1$ Hauptslots werden nacheinander die Arbeitspunkte

$$\begin{aligned} \boldsymbol{c}^0 &= (c_0^{\max}, c_1^{\min}, c_2^{\min}, \ldots, c_s^{\min}) \\ \boldsymbol{c}^1 &= (c_0^{\min}, c_1^{\max}, c_2^{\min}, \ldots, c_s^{\min}) \\ &\vdots \\ \boldsymbol{c}^s &= (c_0^{\min}, c_1^{\min}, \ldots, c_{s-1}^{\min}, c_s^{\max}) \end{aligned} \tag{9.18}$$

eingestellt. Wenn mit $g = s$ die Matrixgröße der Gauß-Elimination ausreichend groß gewählt wurde, so resultiert wegen $e_i^{\min} = \frac{T^{\text{main}}}{c_i^{\max}}$ und $e_i^{\max} = \frac{T^{\text{main}}}{c_i^{\min}}$ die erweiterte Koeffizientenmatrix

$$\mathbf{M} = \left(\begin{array}{ccccc|c} 1 & e_1^{\max} & e_2^{\max} & \ldots & e_s^{\max} & W^0 \\ 1 & e_1^{\min} & e_2^{\max} & \ldots & e_s^{\max} & W^1 \\ 1 & e_1^{\max} & e_2^{\min} & & e_s^{\max} & W^2 \\ \vdots & \vdots & & \ddots & & \vdots \\ 1 & e_1^{\max} & e_2^{\max} & & e_s^{\min} & W^s \end{array}\right). \tag{9.19}$$

Bereits nach der Elimination der Koeffizienten unterhalb des ersten Pivots durch die Transformationen $Z_k \leftarrow Z_k - Z_0 \;\forall\, k \in \{1, \ldots, s\}$ wird die gewünschte Stufenform

$$\mathbf{M} = \left(\begin{array}{ccccc|c} 1 & e_1^{\max} & e_2^{\max} & \ldots & e_s^{\max} & W^0 \\ 0 & e_1^{\max} - e_1^{\min} & 0 & \ldots & 0 & W^0 - W^1 \\ 0 & 0 & e_2^{\max} - e_2^{\min} & & 0 & W^0 - W^2 \\ \vdots & \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & & e_s^{\max} - e_s^{\min} & W^0 - W^s \end{array}\right) \tag{9.20}$$

erreicht. Nach Definition 4.2 gilt $c_i^{\min} > c_i^{\max}$ für alle skalierbaren Tasks $t_i \in \mathcal{T}_S$ und damit ist auch $e_i^{\max} - e^{\min} \neq 0$. Die Koeffizientenmatrix hat somit $s+1$ Pivotkoeffizienten, also vollen Rang und kann zu

$$\widetilde{w}_i = \frac{W^0 - W^i}{e_i^{\max} - e_i^{\min}} = \frac{W^0 - W^i}{T^{\text{main}}} \frac{c_i^{\max} c_i^{\min}}{c_i^{\max} - c_i^{\min}} \;\forall\, i \in \{1, \ldots, s\} \tag{9.21a}$$

$$\text{und} \quad \widetilde{w}_0 = W^0 - \sum_{i=1}^{s} e_i^{\max} \widetilde{w}_i = W^0 - T^{\text{main}} \sum_{i=1}^{s} \frac{\widetilde{w}_i}{c_i^{\min}} \tag{9.21b}$$

aufgelöst werden.

Neben der eindeutigen Lösbarkeit des Gleichungssystems bietet die Koeffizientenmatrix (9.19) einige weitere Vorteile. Durch die Verwendung der minimalen und maximalen Ausführungshäufigkeiten ist die Differenz $W^0 - W^i$ maximal und somit der Einfluss von Messungenauigkeiten bei der Bestimmung des Gesamtverbrauchs des Sensorknotens reduziert. Außerdem laufen die skalierbaren Tasks während der $s+1$ Hauptslots zur Bestimmung von $\boldsymbol{W}$ fast permanent mit minimaler Periode, also mit maximalem Nutzen (vgl. Abschnitt 4.3).

Des Weiteren muss lediglich W^0 über mehrere Hauptslots gespeichert werden, da mit jedem weiteren W^i die Komponente w_i mittels (9.21a) direkt berechnet werden kann und die notwendigen Daten für (9.21b) bereits in `ra_scalable_t` enthalten sind. Die Auflösung von (9.19) kann daher mit der Ansteuerung von (9.18) verzahnt werden, wodurch das gesteuerte Lernen unabhängig von der Implementierung der Gauß-Elimination und der oben genannten Bedingung $g = s$ wird.

Da die Arbeitspunkte während des gesteuerten Lernens die Anforderungen für eine nutzenoptimierte Energieverteilung (Definition 7.3) nicht erfüllen, die Algorithmen des *Adaptive Duty Cycling* dies aber vorraussetzen, muss zum Abschluss des gesteuerten Lernen wieder eine nutzenoptimierte Verteilung hergestellt werden. Dazu genügt es, die Periode c_s des zuletzt untersuchten Tasks wieder auf sein Minimum zu setzten und damit den extremalen Arbeitspunkt $\boldsymbol{c}^{\min}$ wiederherzustellen. Das Modul für das *Adaptive Duty Cycling* setzt sein *Handle* während des gesteuerten Lernens selbsständig auf den letzten Task der nutzensortierten Liste, wodurch Definition 7.3 erfüllt wird (vgl. Abbildung 7.2d).

KAPITEL

10 Evaluation der Implementierung

10.1 Emulation

Für die Evaluation der Softwaremodule von SNoW^5-RA wird ein vom Sensorknoten selbst ausgeführter Emulator verwendet. Der Verzicht auf eine Hardwarerealisierung zugunsten einer Verhaltenssimulation wird dabei durch verschiedene Faktoren motiviert. So hat Kapitel 1 gezeigt, dass die Größe des Solarmoduls genau auf das erwartete Bestrahlungsprofil im Testzeitraum abgestimmt werden muss, damit die im Mittel erwirtschaftete Leistung zwischen dem maximalen und minimalen Verbrauch des Sensorknotens liegt. Ansonsten wäre das System über- oder unterdimensioniert und müsste permanent auf maximaler bzw. minimaler Leistung betrieben werden. Die Steuerungsmechanismen von SNoW^5-RA könnten damit nicht überprüft werden. Bei der Bestellung der XOD17-Solarzellen von Ixys [14], mit denen die generierte Leistung über die Anzahl der parallel und seriell verschalteten Zellen reguliert werden kann (vgl. Kapitel 2), kam es außerdem zu langfristigen Lieferschwierigkeiten. Alternative monokristalline Zellen mit etwa 16 % Wirkungsgrad waren nur als vollständige Module erhältlich, die zu einer Überdimensionierung des Systems geführt hätten.

Ein weiterer Vorteil eines Emulators ist die Möglichkeit, die Simulationsgeschwindigkeit zu beeinflussen, denn das Erlernen des Energieverbrauchs der skalierbaren Tasks und das Anpassen des Verbrauchsprofils an das Einnahmeprofil sind langwierige Mechanismen, die über einige Tage beobachtet werden müssen. Zur Verkürzung der Entwicklungs- und Evaluationszeit wird daher ein Zeitskalierungsfaktor `RA_EMU_SCALE` eingeführt, durch den alle *Deadlines* und Wartezeiten bei der Emulation geteilt werden.

10.1.1 Emulation des Verbrauchers

Um den von der Ausführungshäufigkeit der skalierbaren Tasks abhängigen Energieverbrauch des Sensorknotens zu simulieren, werden Testtasks definiert, welche durch den Aufruf der Makros `IDLE(t)` und `ACTIVE(I,t)` ein Aktivitätsprofil an den Emulator senden, statt eine konkrete Anwendung zu implementieren. Das erste Makro simuliert eine Phase der Länge t (ms) im energiesparenden *idle*-Modus durch Aufruf der `sleep` Funktion des Betriebssystems. Das `ACTIVE` Makro simuliert hingegen einen für die Dauer t (ms) konstanten Verbrauchsstrom I (mA), indem es im Emulator den Zähler T^{active} für die aktive Laufzeit um t und den Zähler W^{active} für den aktiven Verbrauch um $I \cdot t$ erhöht. Am Ende eines Hauptslots wird der Gesamtverbrauch des Sensorknotens als

$$W = W^{\text{active}} + I^{\text{idle}} \cdot (T^{\text{main}} - T^{\text{active}}) \tag{10.1}$$

an das *Hardware Layer* weitergegeben, da der Sensorknoten die nicht-aktive Zeit des Slots im *idle*-Modus verbracht haben muss. Der Stromfluss I^{idle} wird dabei als Compilezeitkonstante `RA_HAL_IDLE_CURRENT` für alle Tests auf 5,7 mA festgelegt [2].

Um eine Verbrauchsstreuung zu simulieren, wird eine normalverteilte Abweichung zu dem im `ACTIVE` Makro angegebenen Verbrauchsstrom addiert. Dazu muss zunächst eine gleichverteilte Pseudozufallsvariable $r \in \{-128, \ldots, 127\}$ auf einen normalverteilten Skalierungsfaktor

$$N(r) := \operatorname{sgn}(r) \cdot \begin{cases} 3 & \text{falls } 127 \leq |r| \\ 2 & \text{falls } 111 \leq |r| \leq 126 \\ 1 & \text{falls } 49 \leq |r| \leq 110 \\ 0 & \text{falls } |r| \leq 48 \end{cases} \tag{10.2}$$

umgerechnet werden. Abbildung 10.1 zeigt die annähernd standardnormalverteilten Wahrscheinlichkeiten der Werte dieses Skalierungsfaktors. Bei jedem Aufruf des `ACTIVE` Makros wird nun

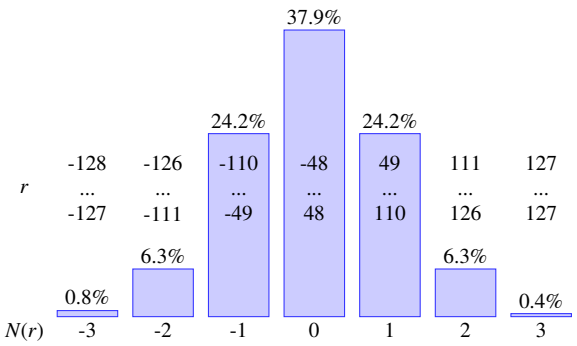


Abbildung 10.1: diskrete Wahrscheinlichkeitsverteilung bei der Verbrauchsstreuung

eine neue Pseudozufallszahl r bestimmt und die Abweichung

$$\Delta I(r) = I \cdot N(r) \cdot \delta_{streu} \tag{10.3}$$

zum Verbrauchsstrom I addiert, wobei δ_{streu} ein Konfigurationsparameter für die verschiedenen Simulationen ist. Solange dieser nicht explizit spezifiziert wird, ist $\delta_{streu} = 0$.

Programmauszug 10.1 zeigt die Grundkonfiguration des Verbrauchers, welche bis auf Abschnitt 10.4 für alle Tests verwendet wird. Dabei werden $p = 5$ periodische Tasks definiert, von denen $s = 4$ skalierbar sind und damit zur Steuerung des Gesamtverbrauchs beitragen. Nach Abschnitt 4.2 wird der Verbrauch w_i der einzelnen Tasks von SNoW^5-RA als Mehrverbrauch über *idle*-Niveau verwaltet. Wenn also $\mathcal{A}_i = \{(I^1, t^1), \ldots, (I^n, t^n)\}$ die Menge der aktiven Phasen des Aktivitätsprofils von t_i ist, dann entspricht

$$w_i = \sum_{(I^k, t^k) \in \mathcal{A}_i} (I^k - I^{\text{idle}}) \cdot t^k \tag{10.4}$$

der Verbrauchskomponente dieses Tasks. Der Verbrauch des nicht-skalierbaren Tasks t_5 innerhalb eines Hauptslots kann dagegen nicht gesteuert werden und zählt deshalb zum Grundverbrauch

$$w_0 = T^{\text{main}} \left(I^{\text{idle}} + \sum_{i=s+1}^{p} \frac{w_i}{c_i} \right). \tag{10.5}$$

Ein zusätzlicher periodischer Task ist der von SNoW^5-RA selbst definierte `mainSlotTrigger`, welcher aber nicht skalierbar ist und in der Simulation auch keinen Einfluss auf den Grundverbrauch hat, da er kein Aktivitätsprofil definiert.

Tabelle 10.1 fasst die Verbrauchskomponenten der Testkonfiguration neben dem relativen energetischen Nutzen und der Periodenbeschränkung der einzelnen Tasks für $T^{\text{main}} = 1$ h zusammen.

```
100 RA_DECLARE_SCALABLE_TASK(test1, 100, 250, 250, 5 RA_SECONDS, 20 RA_SECONDS, 0, NULL);
101 RA_TASKENTRY(test1) {ACTIVE(25, 300);}

103 RA_DECLARE_SCALABLE_TASK(test2, 100, 200, 200, 30 RA_SECONDS, 180 RA_SECONDS, 0, NULL);
104 RA_TASKENTRY(test2) {ACTIVE(20, 100); ACTIVE(30, 150); IDLE(1000); ACTIVE(35, 250);}

106 RA_DECLARE_SCALABLE_TASK(test3, 100, 150, 150, 60 RA_SECONDS, 90 RA_SECONDS, 0, NULL);
107 RA_TASKENTRY(test3) {ACTIVE(35, 900);}

109 RA_DECLARE_SCALABLE_TASK(test4, 100, 150, 150, 30 RA_SECONDS, 50 RA_SECONDS, 0, NULL);
110 RA_TASKENTRY(test4) {ACTIVE(21, 1200);}

112 RA_DECLARE_PERIODIC_TASK(test5, 100, 50, 25 RA_SECONDS, NULL);
113 RA_TASKENTRY(test5) {ACTIVE(22, 800);}
```

Programmauszug 10.1: `testbenches/testbench_ra/tasks/testbench.c`

Tabelle 10.1: Grundkonfiguration des Verbrauchers für $T^{\text{main}} = 1$ h

i	0	1	2	3	4
b_i		250	200	150	100
$c_i^{\min}$	1 h	5 s	30 s	60 s	30 s
$c_i^{\max}$	1 h	20 s	180 s	90 s	50 s
w_i	6,22 mAh	5,79 mAs	12,4 mAs	26,37 mAs	18,36 mAs
$\widetilde{w}_i$	5,7 mAh	0 mAs	0 mAs	0 mAs	0 mAs

Solange nicht anders spezifiziert, verwenden alle Simulationen diese Hauptslotlänge. Nach Ausdruck (4.5) lässt sich der Verbrauch innerhalb eines Hauptslots damit zwischen

$$W^{\min} = T^{\text{main}} \sum_{i=0}^{s} \frac{w_i}{c_i^{\max}} = 7{,}24 \text{ mAh} \tag{10.6a}$$

$$\text{und} \quad W^{\max} = T^{\text{main}} \sum_{i=0}^{s} \frac{w_i}{c_i^{\min}} = 8{,}84 \text{ mAh} \tag{10.6b}$$

variieren.

Die initialen Verbrauchsannahmen $\widetilde{w}_i$ der skalierbaren Tasks werden in der Grundkonfiguration von Programmauszug 10.1 nicht gesetzt und die Annahme über den Grundverbrauch initialisiert SNoW^5-RA standardmäßig mit $\widetilde{w}_0 = I^{\text{idle}} \cdot T^{\text{main}}$.

10.1.2 Emulation der Energiequelle

Für die Simulation der vom Solarmodul generierten Leistung wird ein 24 h-periodisches, skalierbares Einnahmeprofil $\boldsymbol{h} = (h_0, \ldots, h_{95}) \in \{0, \ldots, 255\}^{96}$ verwendet, dessen Definition Programmauszug 10.2 zeigt. Dieses Profil beruht auf den Daten des Photovoltaischen Geographischen Informationssystem [29] für ein in Würzburg aufgestelltes, nach Süden ausgerichtetes und um 20° gegen die Horizontale geneigtes Solarmodul (vgl. Abbildung 1.10). Der über jeweils einen Monat gemittelte tägliche Verlauf der Bestrahlungsstärke für ein solches Modul ist

```
5  static unsigned char input_profile[] = {
6            0,  0,  0,  0,     0,  0,  0,  0,     0,  0,  0,  0,     0,  0,  0,  0,
7            0, 11, 17, 23,    30, 37, 47, 58,    70, 81, 94,106,   118,130,142,154,
8          165,176,186,195,   204,213,220,227,   233,239,243,247,   250,253,254,255,
9          255,254,253,250,   247,243,102,101,   101, 99, 98, 96,    95,186,176,165,
10         154,142,130,118,   106, 94, 81, 70,    58, 47, 37, 30,    23, 17, 11,  0,
11           0,  0,  0,  0,     0,  0,  0,  0,     0,  0,  0,  0,     0,  0,  0,  0
12 };
```

Programmauszug 10.2: `libs/ra/profile/regulator_test_20Jul.sp`

in Tabelle A.3 angegeben. Das Einnahmeprofil $\boldsymbol{h}$ entspricht im Wesentlichen der Normierung der Juli-Werte dieser Tabelle auf den Wertebereich eines Bytes. Lediglich in der Zeit von 13:38 bis 15:08 ($h_{54}, \ldots, h_{60}$) werden niedrigere Werte verwendet, um eine Verschattung durch ein stationäres Objekt zu simulieren.

Die Tests aus Abschnitt 10.4 konfigurieren unterschiedliche Skalierungsfaktoren $h^{\max}$ als maximal erzeugte Solarleistung. Damit kann der Einsatz unterschiedlich großer Solarmodule simuliert werden, denn nach Ausdruck (1.8) ist die Modulfläche ein Proportionalitätsfaktor beim linearen Zusammenhang zwischen der Bestrahlungsstärke und der erzeugten Leistung. Die innerhalb eines Tages produzierte Energiemenge ergibt sich als

$$\rho_H \cdot 24\ \text{h} = h^{\max} \sum_{i=0}^{95} \frac{h_i}{256} \cdot 15\ \text{min} = h^{\max} \cdot 8{,}32\ \text{h}, \tag{10.7}$$

so dass die mittlere Harvesterleistung mit $\rho_H = 0,35 \cdot h^{\max}$ angegeben werden kann.

10.1.3 Emulation des Energiespeichers

Durch die Aktivitätsprofile der Testtasks der Verbrauchssimulation ist dem Emulator der zum Sensorknoten fließende Strom $I_C(t)$ zu jedem Zeitpunkt t bekannt. Bei einer konstanten Versorgungsspannung von $U_{cc} = 3{,}3$ V ergibt dies eine Verbrauchsleistung von

$$P_C(t) = I_C(t) \cdot U_{cc}. \tag{10.8a}$$

Genauso wie das Harvesterprofil

$$P_H(t) = \frac{h^{\max}}{256} \cdot h_{\left\lfloor \frac{t \bmod 24\ \text{h}}{15\ \text{min}} \right\rfloor} \tag{10.8b}$$

ist auch das Verbrauchsprofil stückweise konstant, so dass die Aktualisierung der simulierten Energiespeicherstände immer am Ende einer Phase konstanter Einnahmen und Ausgaben ausgeführt werden kann.

Am Ende einer solchen Phase der Länge Δt sei $P_C := P_C(t)$ und $P_H := P_H(t)$. Anhand des Zustandes der von den Moore-Automaten des *Hardware Layers* verwalteten GPIO-Pins für die Steuerung der Aktoren des Energiespeichers (vgl. Abschnitt 5.2) bestimmt der Emulator nun die Energieänderungen am Primär- und Sekundärspeicher nach Algorithmus 10.1. Die Fallunterscheidung zur Bestimmung der Leistungen an den beiden Speichern orientiert sich dabei an Abbildung 3.4. Auf eine Modellierung der Verluste von etwa 5 % am MAX1675 Spannungswandlers wird verzichtet, aber für den TPS61201 verwendet der Emulator zwei verschiedene Effizienzfaktoren:

- $\eta_0 := 60$ % bei niedrigen Lastströmen (nur zum Sensorknoten) und

- $\eta_1 := 80\ \%$ bei hohen Lastströmen (zum Sensorknoten und zum Sekundärspeicher).

Der Energietransport vom Primär- zum Sekundärspeicher wird als $P_{charge} = 4\ \text{V} \cdot 250\ \text{mA} = 1\ \text{W}$ simuliert und kann nur im Primärbetrieb auftreten, da sich der Akkumulator sonst selbst laden würde. Dieser vierte Fall (`selectBat=chargeBat=1`) muss vom Emulator daher nicht behandelt werden.

Die Ladeeffizienzen der beiden Speicher sowie die Selbstentladung des Sekundärspeichers können nach Tabelle 3.1 vernachlässigt werden. Die Selbstentladung des SuperCaps muss hingegen beachtet werden. Sie ergibt sich aus der aktuellen Kondensatorspannung U_{cap} und einem Leckstrom I_{cap}^{leak}, der beim BCAP0350 beispielsweise mit 1 mA spezifiziert wird [26].

Die Energieänderungen an den beiden Speichern werden nun über die gesamte Simulationsdauer akkumuliert und die Speicherfüllstände auf Anfrage des *Hardware Layers* in entsprechende Zellenspannungen umgerechnet. Für den Primärspeicher wird dabei Ausdruck (5.1) und für den Sekundärspeicher Ausdruck (5.2) verwendet. Der Emulator verhindert außerdem eine Erhöhung der Primärspannung über 2,5 V, um die Wirkung der Zenerdiode zum Schutz des SuperCaps zu simulieren (vgl. Abbildung 3.4). Bei einem Abfall der Primärspannung unter 0,3 V wird die Simulation außerdem mit einer Fehlermeldung beendet, um einen Systemausfall infolge einer Unterversorgung des TPS61201 zu simulieren.

Algorithmus 10.1 : Bestimmung der Änderungen der Energiespeicherstände

Eingabe : Einnahme- und Verbrauchsleistung P_H und P_C sowie die Dauer Δt seit der letzten Änderung eines dieser Werte, Zustand der Aktoren des Energiespeichers

Ausgabe : Energieänderungen ΔE_{cap} und ΔE_{bat} am Primär- und Sekundärspeicher

wenn *Primärbetrieb (`selectBat=0`) ohne Ladevorgang (`chargeBat=0`)* **dann**
- $P_{bat} \leftarrow 0$;
- $P_{cap} \leftarrow -\frac{P_C}{\eta_0}$;

sonst wenn *Sekundärbetrieb (`selectBat=1`) ohne Ladevorgang (`chargeBat=0`)* **dann**
- $P_{bat} \leftarrow -\frac{P_C}{\eta_0}$;
- $P_{cap} \leftarrow 0$;

sonst wenn *Primärbetrieb (`selectBat=0`) mit Ladevorgang (`chargeBat=1`)* **dann**
- $P_{bat} \leftarrow P_{charge}$;
- $P_{cap} \leftarrow -\frac{P_{charge}+P_C}{\eta_1}$;

$P_{cap} \leftarrow P_{cap} + P_H - U_{cap} \cdot I_{cap}^{leak}$;
$\Delta E_{cap} \leftarrow P_{cap} \cdot \Delta t$;
$\Delta E_{bat} \leftarrow P_{bat} \cdot \Delta t$;

10.2 Bestimmung und Anwendung des Arbeitspunktes

Die folgenden drei Simulationsläufe sind Konfigurationen von `testbenches/testbench_ra/tasks/testbench.c` (vgl. Tabelle 10.1). Sie sollen die Verschiebung des Arbeitspunktes $\boldsymbol{c}$ zur Einstellung eines Zielverbrauchs $\widehat{W}$ sowie die periodische Taskausführung beleuchten. Außerdem wird das Verhalten des Systems bei einer Änderung des Nutzenvektors $\boldsymbol{b}$ untersucht.

Das Erlernen der Verbrauchskomponenten $\boldsymbol{w}$ und die Regelung der Energieneutralität wird in den Abschnitten 10.3 und 10.4 betrachtet und soll daher zunächst ausgeblendet werden. Dazu werden die Verbrauchsannahmen $\widetilde{\boldsymbol{w}}$ mit den korrekten Werten $\boldsymbol{w}$ initialisiert und die Energiebilanzierung (Algorithmus 9.1) deaktiviert. Die Skalierbarkeitsgrenzen $\widetilde{W}^{\min}$ und $\widetilde{W}^{\max}$ des Gesamtverbrauchs bleiben damit während der Simulation konstant. Statt des *JIT-Regulators* wird ein Testregler mit einem Profil $\boldsymbol{r} \in \{0, \ldots, 255\}^n$ konfiguriert, der den Zielverbrauch im Slot j als

$$\widehat{W}^j = \widetilde{W}^{\min} + (\widetilde{W}^{\max} - \widetilde{W}^{\min})\frac{r_{j \bmod n}}{256} \tag{10.9}$$

vorgibt.

10.2.1 *Adaptive Duty Cycling*

$\widetilde{\boldsymbol{w}} = \boldsymbol{w}$	=	$(22397.76, 5.79, 12.4, 26.37, 18.36)$ mAs
$\boldsymbol{r}$	=	$(255, 245, 210, 80, 8, 0, 60, 95, 230)$
`RA_EMU_SCALE`	=	$64\times$

`testbenches/testbench_ra/tasks/adc/01.cfg`

Abbildung 10.2 zeigt, wie der Testregler die Verbrauchsvorgabe $\widehat{W}$ schrittweise von $\widetilde{W}^{\max}$ auf $\widetilde{W}^{\min}$ und danach wieder zurück auf $\widetilde{W}^{\max}$ ändert. Die Reaktionen des *Adaptive Duty Cycling* Moduls auf die Änderungen dieser Vorgaben zeigt Abbildung 10.3, in welcher der zeitliche Verlauf der Taskperioden dargestellt ist. Die Perioden der beiden nicht-skalierbaren Tasks bleiben korrekterweise während der gesamten Laufzeit unverändert.

Die skalierbaren Tasks sind in der Darstellung von oben nach unten nach absteigendem Nutzen sortiert. Das jeweilige *Handle* der Nutzenordnung wird durch einen Punkt markiert (vgl. Kapitel 7). Im Slot 0 sind alle Taskperioden minimiert, um den maximalen Gesamtverbrauch zu erzielen. Das *Handle* dieser Energieverteilung ist der Task t_4 mit niedrigstem Nutzen. Das Verringern des Zielverbrauchs im Slot 1 entspricht einer Verschiebung der Hyperebene (4.4) und wird durch eine Nachführung des Arbeitspunktes angepasst. Diese kleine Verbrauchsänderung kann dabei durch eine Erhöhung von c_4 kompensiert werden, so dass alle anderen Perioden unverändert bleiben und t_4 weiterhin als *Handle* markiert ist. Für die Kompensation der nächsten Verringerung des Zielverbrauchs im Slot 2 reicht die Vergrößerung von c_4 auf $c_4^{\max}$ allerdings nicht mehr aus, so dass t_3 zum *Handle* wird und seinerseits allozierte Energie durch eine Periodenvergrößerung

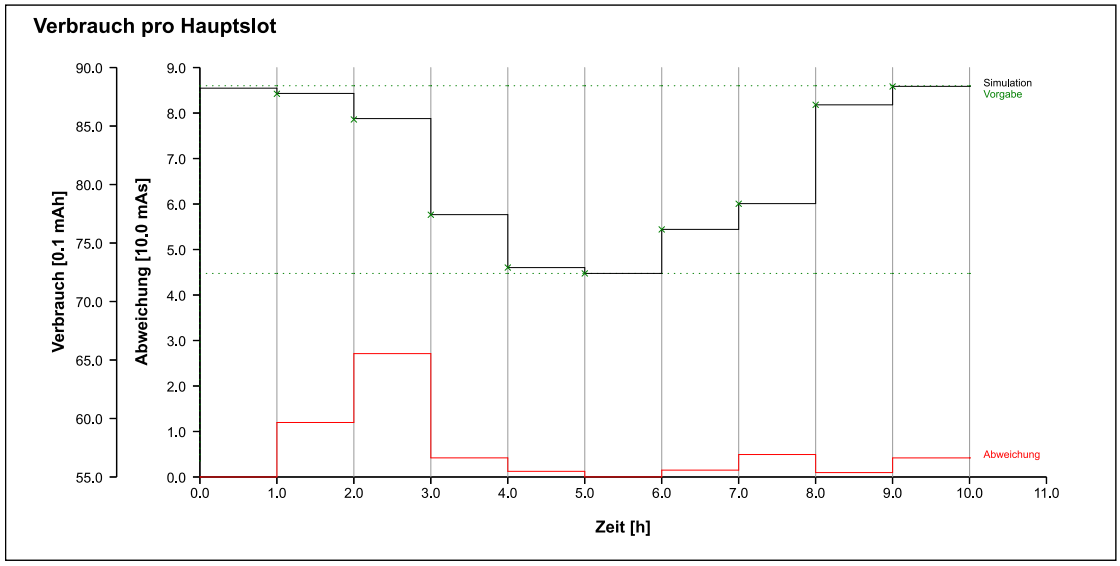


Abbildung 10.2: vorgegebener und tatsächlicher Gesamtverbrauch

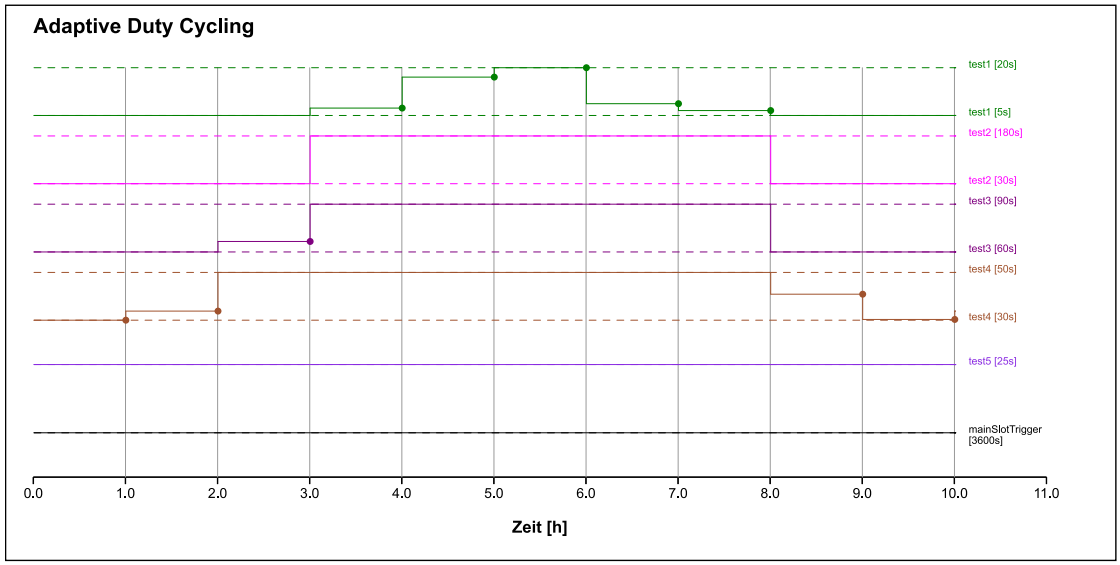


Abbildung 10.3: Arbeitspunktverschiebung

freigeben muss. Auf diese Weise werden die Perioden der Tasks mit immer größerem Nutzen erhöht, bis im Slot 5 der minimale Energieverbrauch erreicht ist und der Task t_1 mit größtem Nutzen das *Handle* darstellt. Da der Testregler den Verbrauch anschließend wieder steigern will,

werden die Taskperioden nun wieder verringert, diesmal aber mit umgekehrter Nutzenpriorität. Aus der Abbildung 10.3 ist auch ersichtlich, dass für die meisten Arbeitspunktanpassungen nur wenige Taskperioden geändert werden müssen.

Abbildung 10.2 zeigt neben den Verbrauchsvorgaben auch den simulierten Verbrauch während eines Hauptslots sowie den Betrag der Differenz aus beiden Größen. Die größte Abweichung beträgt dabei 27,1 mAs bei einem Verbrauchvorgabe von 8,556 mAh. Abweichungen dieser Größenordnung entstehen bereits durch die einmalige Nichtausführung eines einzelnen Tasks, dessen Periodendauer kein Teiler der Hauptslotlänge ist. Für ausreichend große Hauptslots sind solche Abweichungen aber nicht von Belang.

10.2.2 Periodische Taskausführung

$$
\begin{aligned}
T^{\text{main}} &= 2\ \text{min} \\
\widetilde{\boldsymbol{w}} = \boldsymbol{w} &= (746.592, 5.79, 12.4, 26.37, 18.36)\ \text{mAs} \\
\boldsymbol{r} &= (255, 245, 210, 80, 8, 0, 60, 95, 230) \\
\texttt{RA\_EMU\_SCALE} &= 2\times
\end{aligned}
$$

`testbenches/testbench_ra/tasks/adc/02.cfg`

Um die periodische Ausführung der Tasks durch den *Wrapper* sinnvoll darstellen zu können, muss T^{main} verkürzt werden. Damit verringern sich dann zwar auch der Grundverbrauch w_0 sowie die Verbrauchsgrenzen $\widetilde{W}^{\text{min}}$ und $\widetilde{W}^{\text{max}}$, da der Testregler aber relativ zu diesen Grenzen arbeitet, entspricht das in Abbildung 10.4 gezeigte Profil der Arbeitspunktverschiebung dem der letzten Simulation.

Abbildung 10.5 zeigt die Flusszeiten der einzelnen Tasks. Die *Peaks* entsprechen also den Zeiten, in denen sich der *Wrapper* des entsprechenden Tasks im Punkt 8 seines Programmablaufplans (vgl. Abbildung 6.1) befindet. Der Zusammenhang zwischen den Taskperioden in Abbildung 10.4 und der Ausführungshäufigkeit der Tasks, also der Dichte ihrer *Peaks* in Abbildung 10.5 ist offensichtlich.

Die mit einem Kreuz markierten *Peaks* stellen eine nicht eingehaltene Deadline dar. Solche Deadlineverletzungen können insbesondere in Folge einer Periodenverkleinerung entstehen. So ist die Deadline von t_2 am Ende von Slot 7 (16 min) auf 15 min + 180 s = 17,5 min gesetzt. Da beim Wechsel von Slot 7 auf Slot 8 die Periode c_2 aber auf 30 s verkürzt wird, liegt die neue Deadline von t_2 nun bei 15 min + 30 s = 15,5 min, also noch vor dem eigentlichen Slotwechsel und kann damit nicht eingehalten werden. Die nächste Ausführung von t_2 erfolgt daher frühestmöglich nach den Slotwechsel.

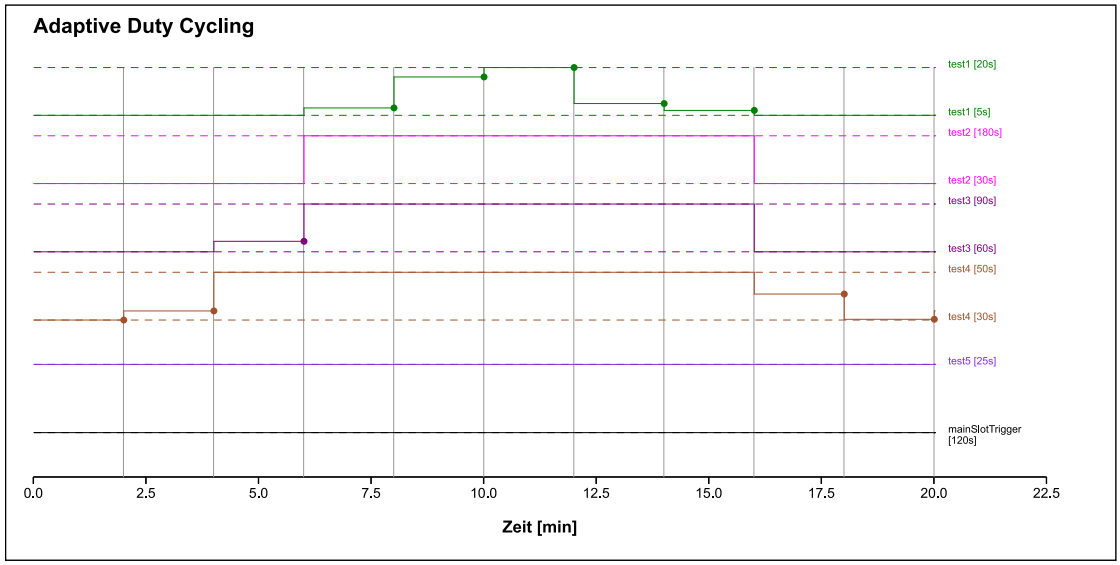


Abbildung 10.4: Arbeitspunktverschiebung

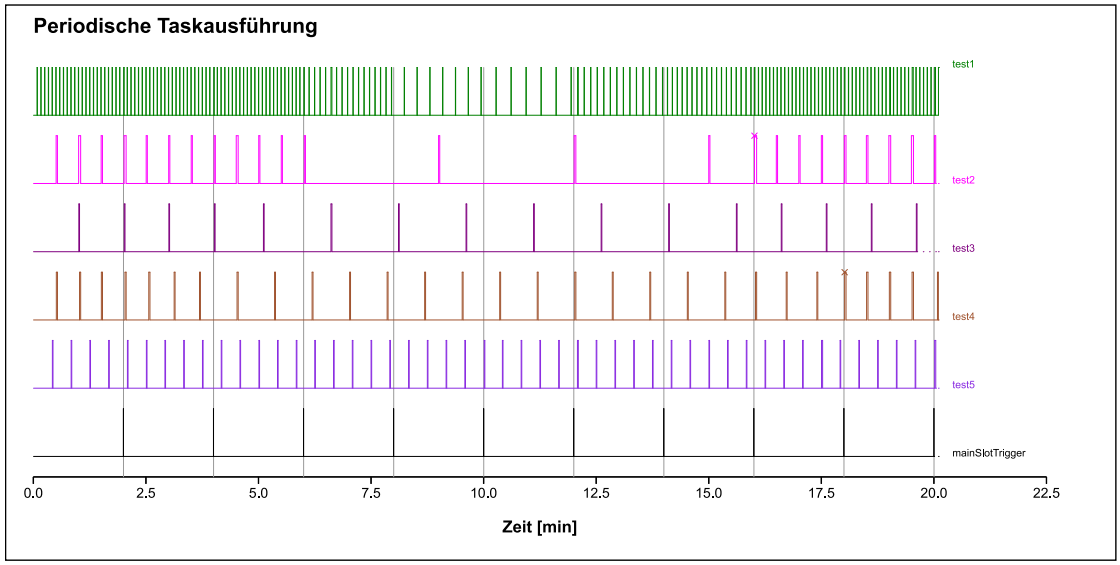


Abbildung 10.5: periodische Taskausführung

10.2.3 Nutzenänderung zur Laufzeit

$\widetilde{w} = w$	=	(22397.76, 5.79, 12.4, 26.37, 18.36) mAs
r	=	(128)
RA_EMU_SCALE	=	64×

testbenches/testbench_ra/tasks/adc/03.cfg

Mit dieser Simulation soll die Arbeitspunktanpassung nach einer Änderung der Nutzenordnung der skalierbaren Tasks untersucht werden. Die Testbench wird dabei so konfiguriert, dass dreimal pro Hauptslot der Nutzen b_i eines jeweils anderen Tasks $t_i \in \mathcal{T}_S$ auf einen zufälligen Wert aus $\{0, \ldots, 7\}$ gesetzt wird. Wenn es nach der Umsortierung der doppelt verketteten Taskliste (vgl. Kapitel 7) zu einer Verletzung der nutzenoptimierten Energieverteilung kommt, so wird der *Violator*-Task in Abbildung 10.6 mit einem Kreuz markiert. Weiter zeigt diese Abbildung die jeweils aktuellen Nutzenwerte und es ist ersichtlich, dass zu jedem Zeitpunkt eine korrekte Sortierung hergestellt wird.

Bei der ersten angezeigten Nutzenänderung (13,25 h) wird b_4 von 3 auf 2 gesetzt, was sich in keiner Weise auf die Tasksortierung auswirkt. Bei 13,5 h wird hingegen der Nutzen b_1 des *Handles* von 3 auf 7 erhöht. Bei der Extraktion von t_1 wird dann der darüber liegende Task t_3 zum neuen *Handle*. Da t_1 nun oberhalb von t_3 wieder eingefügt werden muss, seine Periode aber nicht minimal ist, wird er zum *Violator* und löst damit einen Subslot (gepunktete vertikale Linie) aus. Die Energie, die zur Minimierung seiner Periode eingesetzt wird, muss anschließend durch eine Vergrößerung der Taskperioden, beginnend beim *Handle* t_3, kompensiert werden. Da selbst eine Maximierung von c_3 nicht zur Kompensation ausreicht, wird auch c_1 wieder etwas vergrößert, so dass nach Abschluss der Arbeitspunktverschiebung t_1 wieder zum *Handle* wird. Das Ergebnis der Änderung von b_1 ist also eine Verringerung von c_1 unter Maximierung von c_3 und genau diese Energieumverteilung ist ja auch der Sinn der Nutzenänderung.

Bei 13,75 h wird b_2 von 3 auf 1 geändert, was zwar zu einer Umsortierung der Taskliste führt, aber keinen Subslot notwendig macht, da eine nutzenoptimierte Energieverteilung erhalten bleibt. Die restlichen Nutzenänderungen verlaufen nach jeweils einem dieser ersten drei Schemata.

Wie Abbildung 10.7 zeigt, hält der Testregler die Verbrauchsvorgabe während dieser Simulation auf $\widehat{W} = \frac{\widetilde{W}^{\min} + \widetilde{W}^{\max}}{2}$. Trotz der Energieumverteilungen innerhalb der Hauptslots werden diese Vorgaben bis auf wenige Milliamperesekunden erfüllt. Die geringen, bereits in Abschnitt 10.2.1 festgestellten Abweichungen werden also durch die Subslot-Behandlung nicht weiter verstärkt.

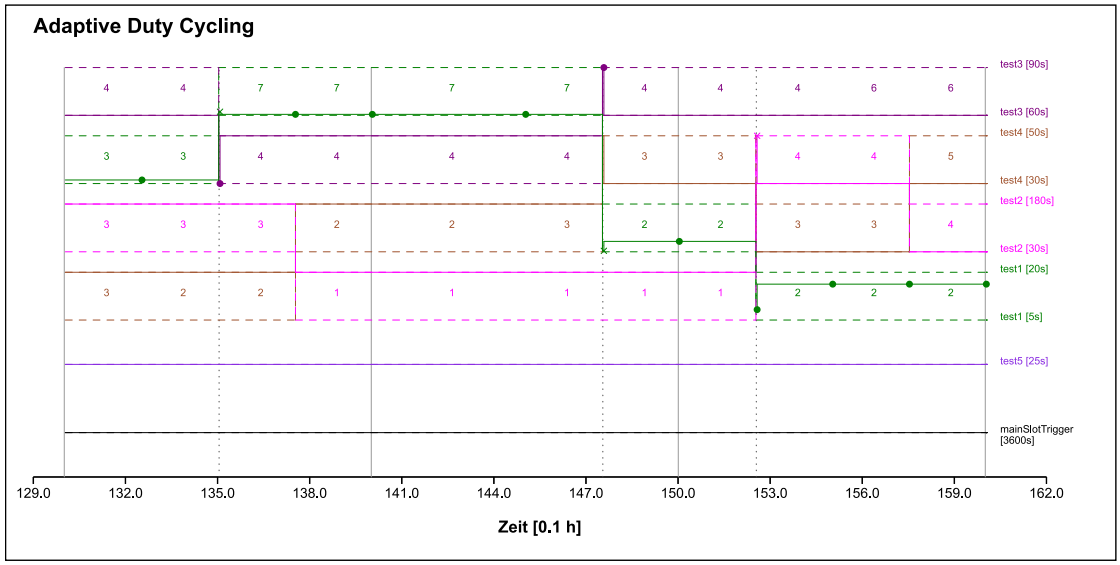


Abbildung 10.6: Nutzenänderung und Auslösung von Subslots

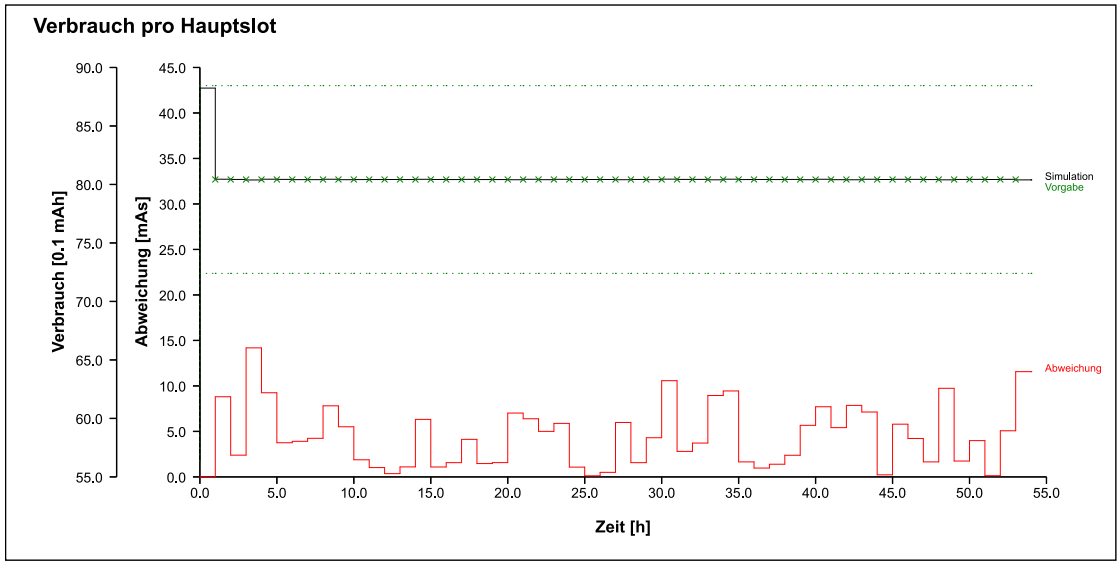


Abbildung 10.7: vorgegebener und tatsächlicher Gesamtverbrauch

10.3 Erlernen des Energieverbrauchs

Die folgenden Simulationen zeigen die Funktionsweise der verschiedenen Lernstrategien zur Annäherung der Verbrauchsannahmen $\widetilde{\boldsymbol{w}}$ an die tatsächlichen Verbrauchswerte $\boldsymbol{w}$. Sie verwenden weiterhin die Testtasks aus `testbenches/testbench_ra/tasks/testbench.c` mit $T^{\text{main}} = 1$ h. Für die Simulationen 10.3.1 bis 10.3.3 werden zunächst die Ergebnisse der Gauß-Elimination durch Konfiguration ausreichend hoher Werte für `RA_ESTIMATOR_MA_RELIABILITY` unterdrückt, um die beiden anderen Lernstrategien isoliert untersuchen zu können.

10.3.1 Gesteuertes Lernen

$$
\begin{aligned}
\delta_{streu} &= \tfrac{1}{8} \\
\boldsymbol{w} &= (22397.76, 5.79, 12.4, 26.37, 18.36)\ \text{mAs} \\
\widetilde{\boldsymbol{w}} &= (20520, 0, 0, 0, 0)\ \text{mAs} \\
\boldsymbol{r} &= (255, 210, 150, 80, 30, 0, 64, 120, 180, 245) \\
\texttt{RA\_EMU\_SCALE} &= 64\times
\end{aligned}
$$

`testbenches/testbench_ra/tasks/estimator/01.cfg`

Durch die fehlende Initialisierung der Verbrauchsannahmen der skalierbaren Tasks weicht der erwartete Energieverbrauch im ersten Slot so stark von der tatsächlich verbrauchten Energiemenge ab, dass Algorithmus 9.1 das gesteuerte Lernen (vgl. Abschnitt 9.4) bereits beim ersten Slotwechsel anstößt. Dabei übernimmt der *Estimator* die Kontrolle über die Arbeitspunkteinstellung, weshalb in den ersten Slots von Abbildung 10.8 keine *Handles* markiert sind.

Im Slot 1 werden alle Taskperioden auf ihrem Minimum gehalten, um den maximalen Verbrauch als Referenz W^0 für die weiteren Slots zu bestimmen. Theoretisch hätte dafür zwar auch der Verbrauch im Slot 0 verwendet werden können, durch die Initialisierungsphase von SNoW^5-RA wird der erste Slot aber verkürzt, so dass sein Verbrauchswert als Referenz ungeeignet ist.

In den folgenden Slots 2 bis 5 wird die Periode c_i je eines Tasks $t_i \in \mathcal{T}_S$ maximiert, während die restlichen $c_{j\neq i}$ minimiert bleiben. Am Ende eines solchen Slots wird der gemessene Verbrauch W^i verwendet, um $\widetilde{w}_i$ über Ausdruck (9.21a) zu bestimmen. Abbildung 10.9 zeigt, wie mit jedem weiteren Slot eine Verbrauchskomponente bestimmt wird. Dabei nähert sich auch die Annahme über den Gesamtverbrauch den simulierten Werten an. Am Ende von Slot 5 wird dann noch $\widetilde{w}_0$ über Ausdruck (9.21b) ermittelt.

Tabelle 10.2 zeigt die Ergebnisse der Ermittlung der Verbrauchskomponenten. Die Tasks t_3 und t_4 tragen weniger zum Gesamtverbrauch bei als t_1 und t_2. Da die Bestimmung von $\widetilde{w}_i$ über (9.21a) aber gerade auf diesem Beitrag $W^0 - W^i$ beruht, verursacht die Verbrauchsstreuung bei

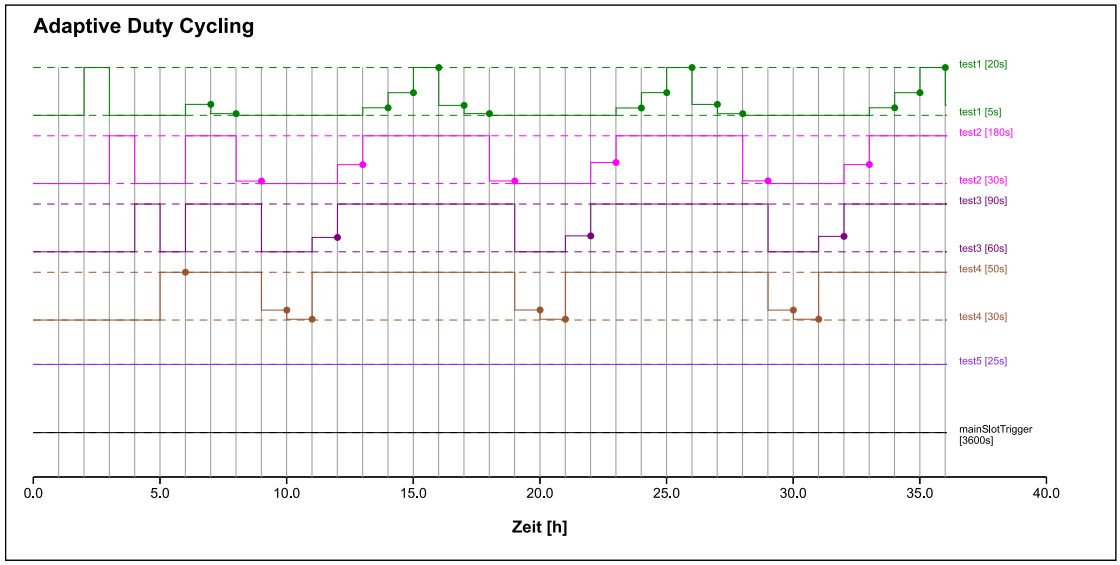


Abbildung 10.8: Vorgabe des Arbeitspunktes durch den Estimator in den Slots 1 bis 5

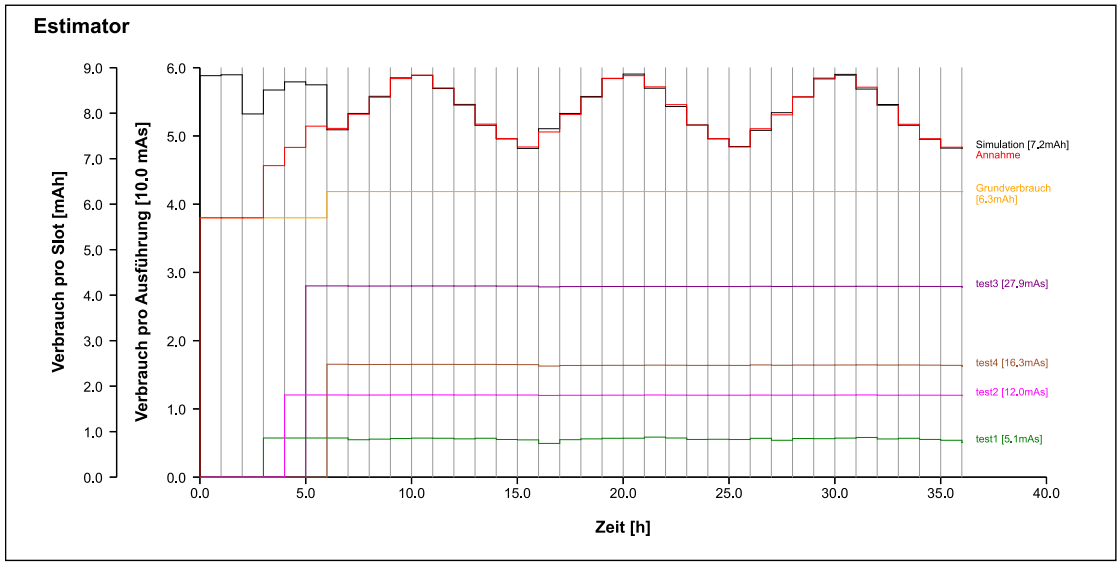


Abbildung 10.9: schrittweises Erlernen der Verbrauchskomponenten

diesen Tasks stärkere Abweichung zwischen der ermittelten und tatsächlichen Verbrauchskomponente. Andererseits können die relativen Fehler von $\widetilde{w}_3$ und $\widetilde{w}_4$ aber auch nur zu geringeren Fehleinschätzungen des Gesamtverbrauchs $\widetilde{W}$ führen. Die genaue Kenntnis von w_3 und w_4 ist

Tabelle 10.2: während des gesteuerten Lernens bestimmte Verbrauchsparameter

i	0	1	2	3	4
W^i [mAs]	31839,213	28741,887	30634,977	31278,741	31045,119
$\widetilde{w}_i$ [mAs]	22598,433	5,735	12,042	28,023	16,543
$\left\lvert\frac{\widetilde{w}_i - w_i}{w_i}\right\rvert$ [%]	0,9	0,95	2,89	6,27	9,90

für das System also weniger von Bedeutung als die von w_0, w_1 und w_2.

Nach Abschluss des gesteuerten Lernens übernimmt das *Adaptive Duty Cycling* wieder die Kontrolle über den Arbeitspunkt zur Verfolgung des vom *Regulator* vorgegebenen Verbrauchsprofils. Die dabei aus den verbleibenden Fehleinschätzungen der Verbrauchskomponenten resultierenden Abweichungen zeigt Abbildung 10.10. Die maximale Abweichung von 255 mAs im Slot 16 entspricht etwa einem Prozent des Grundverbrauchs w_0.

Die folgenden Simulationen werden zeigen, dass das gesteuerte Lernen den anderen beiden Lernstrategien bezüglich Genauigkeit und Geschwindigkeit überlegen ist. Wegen der Deaktivierung des *Adaptive Duty Cycling* können aber die Regelungsmechanismen für die Energieneutralität nicht greifen, so dass das gesteuerte Lernen eine Notfallmaßnahme für zu große Fehler in der Verbrauchsannahme bleiben muss.

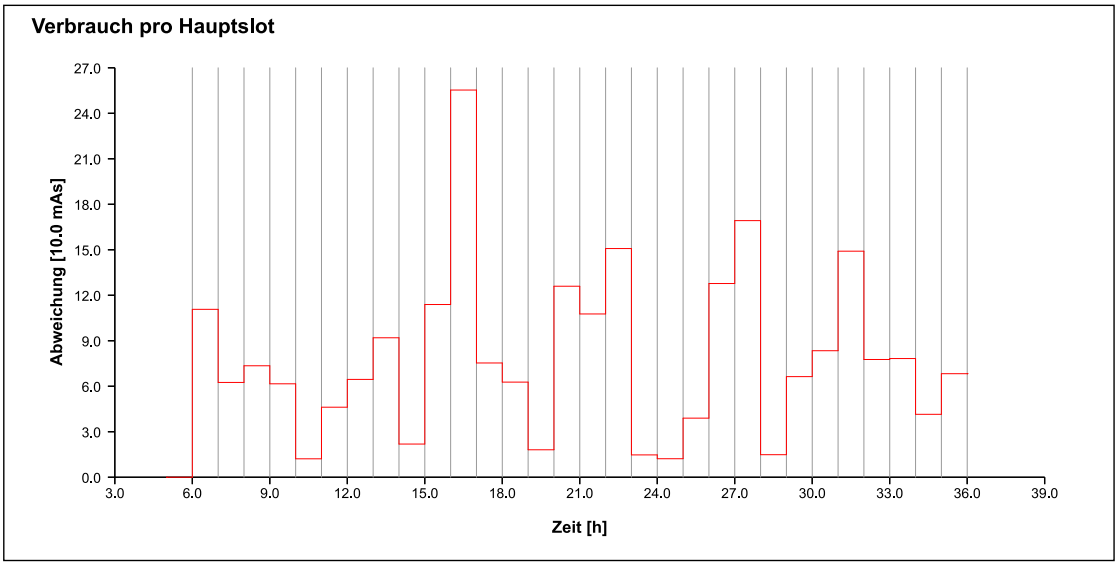


Abbildung 10.10: Abweichungen zwischen $\widehat{W}$ und W nach dem gesteuerten Lernen von $\widetilde{\boldsymbol{w}}$

10.3.2 Konvergenz der Lotverschiebung

$\boldsymbol{w}$	=	$(22397.76, 5.79, 12.4, 26.37, 18.36)$ mAs
$\widetilde{\boldsymbol{w}}$	=	$\boldsymbol{w} - (0, 0, 0, 0, 10)$ mAs
$\boldsymbol{r}$	=	$(255, 128, 32, 0, 40, 120)$
RA_EMU_SCALE	=	$128\times$

testbenches/testbench_ra/tasks/estimator/02.cfg

Um das langfristige Konvergenzverhalten der Lotverschiebung (vgl. Abschnitt 9.2) des Arbeitspunktes zu beobachten, wird die Verbrauchsannahme bis auf einen Fehlervektor korrekt initialisiert. Lediglich $\widetilde{w}_4$ wird dabei um 10 mAs zu niedrig angesetzt. Die daraus resultierende Fehleinschätzung des Gesamtverbrauchs ist zu klein, um das gesteuerte Lernen auszulösen. Da auch die Gauß-Elimination wie oben beschrieben deaktiviert ist, können die Abweichungen nur noch durch sukzessive Lotverschiebungen verringert werden.

Abbildung 10.11 zeigt, wie $\widetilde{w}_4$ im Laufe der Simulation angehoben und damit an w_4 angenähert wird. Gleichzeitig werden zwar auch die anderen Verbrauchsannahmen geändert, da der *Estimator* nicht weiß, von welchem Task die Fehleinschätzungen ausgehen, der euklische Abstand $|\boldsymbol{w} - \widetilde{\boldsymbol{w}}|$ wird dabei aber ständig verringert. Dadurch kann auch dem vorgegebenen Verbrauchsprofil mit immer kleiner werdenden Abweichungen gefolgt werden, wie Abbildung 10.12 zeigt.

Die Länge des Verschiebungsvektors $\Delta\widetilde{\boldsymbol{w}}_L(\boldsymbol{e}, W, \widetilde{\boldsymbol{w}})$ fällt nun aber nach (9.6) ebenfalls mit der Abweichung

$$W - \boldsymbol{e} \cdot \widetilde{\boldsymbol{w}} \overset{\text{Lemma 4.2}}{=} W - \widehat{W}. \tag{10.10}$$

Durch die Normierung von $\Delta\widetilde{\boldsymbol{w}}_L(\boldsymbol{e}, W, \widetilde{\boldsymbol{w}})$ mit $\boldsymbol{e} \cdot \boldsymbol{e}$ wird die µAs-Basiseinheit bei der Festkommarepräsentation der Verbrauchskomponenten bereits bei Abweichungen unterhalb weniger Hundert mAs unterschritten, denn

$$39584 = \boldsymbol{e}^{\min} \cdot \boldsymbol{e}^{\min} \leq \boldsymbol{e} \cdot \boldsymbol{e} \leq \boldsymbol{e}^{\max} \cdot \boldsymbol{e}^{\max} = 550801. \tag{10.11}$$

Die in der Simulation nach etwa 200 h erreichte Maximalabweichung von unter 100 mAs kann daher selbst nach weiteren 500 h nicht mehr entscheidend verringert werden.

Diese Simulation zeigt also, dass das Annähern der Verbrauchsannahme $\widetilde{\boldsymbol{w}}$ an den tatsächlichen Wert $\boldsymbol{w}$ sehr langwierig und in der Genauigkeit durch die Festkommarepräsentation der Verbrauchskomponenten beschränkt ist. Der eigentliche Vorteil und Grund für die Verwendung der Lotverschiebung wird in der nächsten Simulation deutlich.

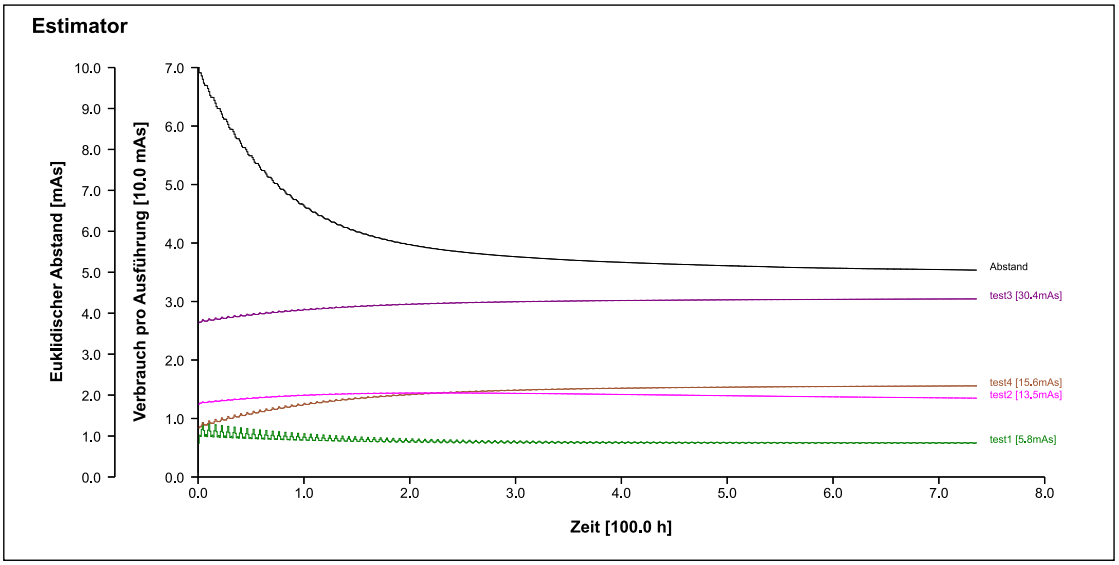


Abbildung 10.11: Ausgleich der Fehleinschätzung von $\widetilde{w}_4$ durch Lotverschiebungen

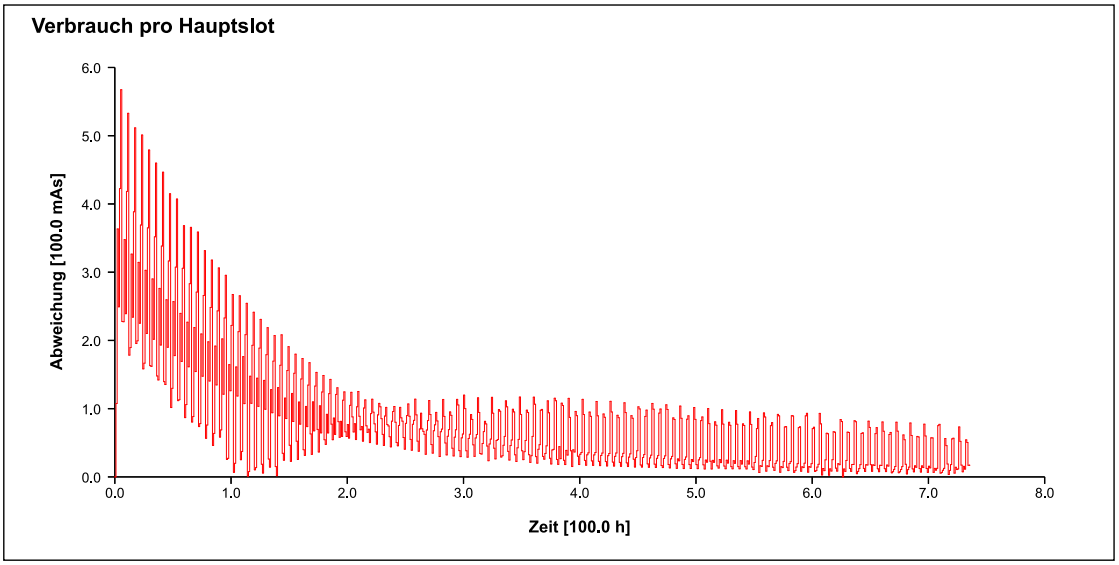


Abbildung 10.12: Verringerung der Abweichungen zwischen $\widehat{W}$ und W durch Lotverschiebungen

10.3.3 Lotverschiebung auf Verbrauchsplateau

$$
\begin{aligned}
\boldsymbol{w} &= (22397.76, 5.79, 12.4, 26.37, 18.36)\ \text{mAs} \\
\widetilde{\boldsymbol{w}} &= \boldsymbol{w} - (0, 0, 0, 0, 10)\ \text{mAs} \\
\boldsymbol{r} &= (255, 210, 150, 80, 30, 30, 30, 30, 30, 64, 180, 245, 245, 250, 255, 255, 255) \\
\texttt{RA\_EMU\_SCALE} &= 128\times
\end{aligned}
$$

`testbenches/testbench_ra/tasks/estimator/03.cfg`

Mit dieser Konfiguration simuliert der Testregler die für den *JIT-Regulator* typischen Verbrauchsplateaus. Dies sind die Phasen kaum veränderlicher Verbrauchsvorgaben, die entstehen, wenn der Zielverbrauch $\widehat{W}$ während der Nacht permanent an der Untergrenze $\widetilde{W}^{\min}$ bzw. tagsüber an der Obergrenze $\widetilde{W}^{\max}$ gehalten wird. Die Lotverschiebung sorgt in diesen Phasen dafür, dass der Verbrauch kaum noch von den Zielvorgaben abweicht, auch wenn die Verbrauchskomponenten noch nicht korrekt erlernt wurden.

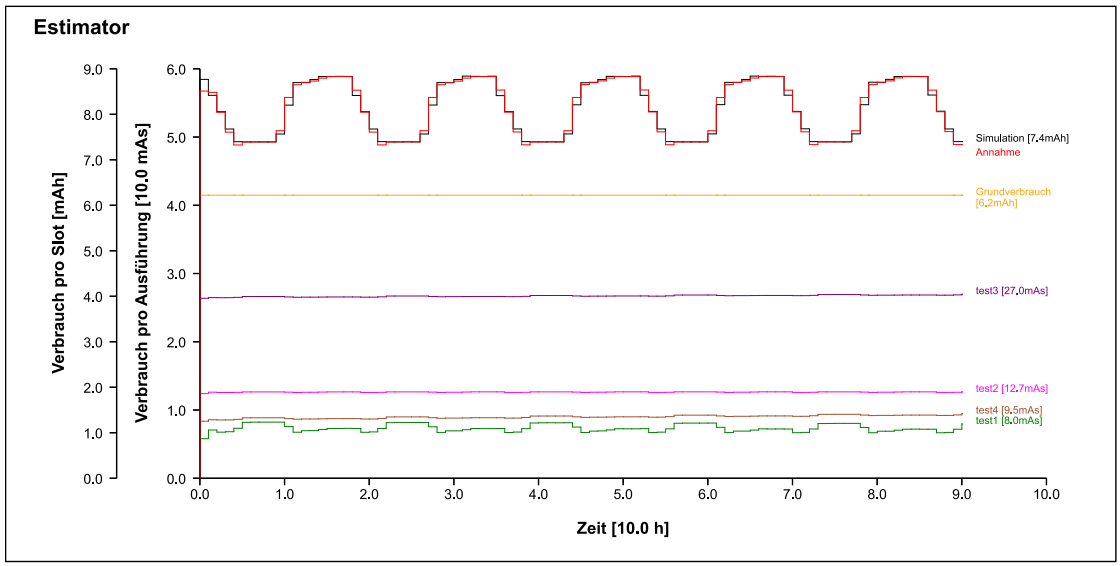


Abbildung 10.13: Lotverschiebung auf Verbrauchsplateau

10.3.4 Gauß-Elimination

$$
\begin{aligned}
\boldsymbol{w} &= \begin{cases} (22397.76, 5.79, 12.40, 26.37, 18.36)\ \text{mAs} & \text{bis 7 h} \\ (22397.76, 8.79, 12.40, 26.37, 35.16)\ \text{mAs} & \text{von 7 bis 17 h} \\ (20784.96, 8.79, 18.65, 26.37, 35.16)\ \text{mAs} & \text{ab 17 h} \end{cases} \\
\widetilde{\boldsymbol{w}} &= (22397.76, 5.79, 12.40, 26.37, 18.36)\ \text{mAs} \\
\boldsymbol{r} &= (255, 210, 150, 80, 30, 30, 30, 30, 30, 64, 180, 245, 245, 250, 255, 255, 255)
\end{aligned}
$$

RA_EMU_SCALE	=	64×
RA_ESTIMATOR_GE_SIZE	=	4
RA_ESTIMATOR_GE_CONSISTENCE_TOLERANCE	=	1 mAs
RA_ESTIMATOR_MA_RELIABILITY	=	2
RA_ESTIMATOR_MA_THRESHOLD	=	6,25 %

testbenches/testbench_ra/tasks/estimator/04.cfg

Beim Erlernen der Verbrauchskomponenten wird das Erstellen und Lösen eines linearen Gleichungssystems mit Hilfe der Gauß-Elimination als Mittelweg zwischen dem schnellen, aber nicht permanent einsetzbaren Konzept des gesteuerten Lernens und der langsameren Lotverschiebung verwendet. Ziel dieser Simulation ist die Beobachtung der Kombination der Gauß-Elimination mit der Lotverschiebung. Dazu wird der Verbrauchsvektor $\boldsymbol{w}$ zur Laufzeit nach 7 bzw. 17 h durch eine Änderung des Aktivitätsprofils einzelner Testtasks verschoben.

In Abbildung 10.14 werden die vom Eliminationsverfahren ermittelten Lösungen x_i als Kreuze dargestellt. Ein Rahmen um das Kreuz kennzeichnet außerdem die Verlässlichkeit der Lösung und wird gesetzt, sobald mindestens RA_ESTIMATOR_MA_RELIABILITY Werte in den gleitenden Mittelwert Ω_i eingehen (vgl. Abschnitt 9.3.4). Mit jeder verlässlichen Lösung wird $\widetilde{w}_i$ auf Ω_i gesetzt. Da die Lotverschiebung von Algorithmus 9.1 aber nach der Gauß-Elimination ausgeführt wird, kann diese die Verbrauchsannahmen noch einmal ändern. Die Lotverschiebung dominiert damit die Gauß-Elimination, baut aber auf deren Resultaten auf.

Ob und nach welchen Verbrauchskomponenten ein Gleichungssystem durch das Eliminationsverfahren aufgelöst werden kann, hängt maßgeblich von den Arbeitspunkteinstellungen der vergangenen Slots ab, da die Ausführungshäufigkeiten der Tasks durch ihre Periode bestimmt werden und als Koeffizienten in das Gleichungssystem eingehen. Abbildung 10.15 zeigt die Arbeitspunkteinstellungen während der gesamten Simulation. Direkt nach einem Wechsel zwischen dem unteren und dem oberen Verbrauchsplateau wurden die Perioden und damit die Ausführungshäufigkeiten aller Tasks geändert und die Wahrscheinlichkeit, dass die Zeilen des Gleichungssystems linear unabhängig sind, ist besonders hoch. Wann immer in Abbildung 10.14 nach allen Verbrauchskompenenten aufgelöst werden konnte, also bei 5, 22, 30 und 38 h Simulationszeit, dann fand unmittelbar zuvor ein solcher Plateauwechsel statt.

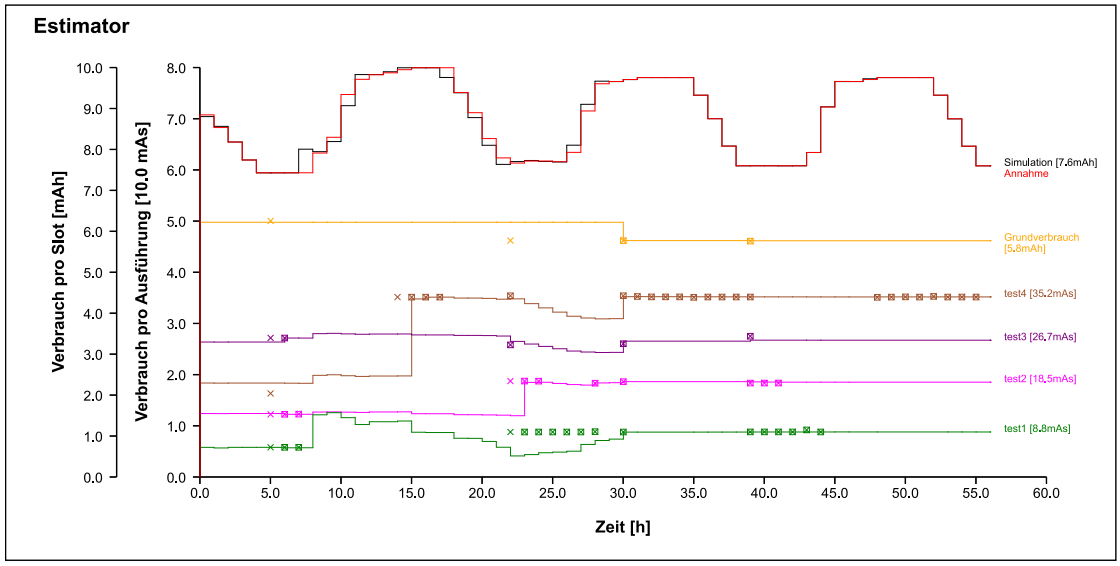


Abbildung 10.14: Kombinierte Lernstrategie aus Gauß-Elimination und Lotverschiebung

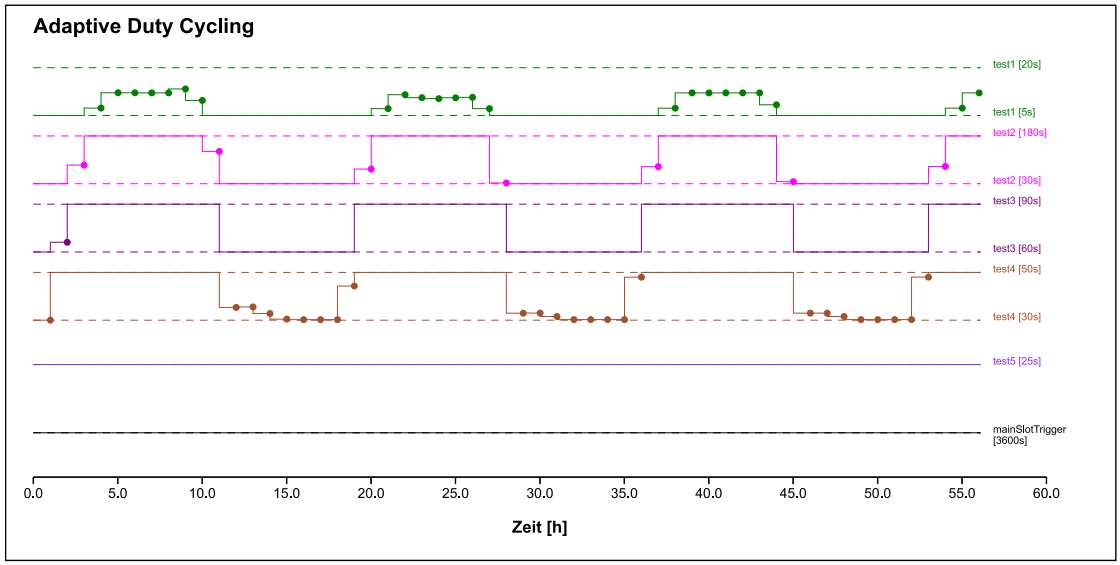


Abbildung 10.15: Arbeitspunktverschiebung

Innerhalb eines Plateaus wird bei der Verschiebung des Arbeitspunktes in der Regel die Periode höchstens eines Tasks geändert. In Abbildung 10.14 ist dies c_1 bei niedrigem bzw. c_4 bei hohem Verbrauch. Gegen Ende der Phasen konstanter Verbrauchsvorgaben kann das Glei-

chungssystem daher nur noch nach der Verbrauchskomponente dieses einen Tasks aufgelöst werden.

Bis zur ersten Verschiebung des Verbrauchsvektors stimmen der erwartete und der tatsächliche Verbrauch überein, da $\widetilde{\boldsymbol{w}}$ mit den korrekten Werten für diese erste Phase initialisiert wurde. Durch die Vergrößerung von w_1 und w_4 nach 7 h wird auch der Gesamtverbrauch W erhöht. Die Lotverschiebung versucht nun, die entstandene Differenz durch Erhöhung aller Verbrauchsannahmen $\widetilde{w}_i$ entsprechend ihrer Ausführungshäufigkeiten e_i zu kompensieren.

Durch die abrupte Änderung der beiden Verbrauchskomponenten sind die Gleichungssysteme der nächsten vier Slots in jedem Fall inkonsistent, weil die Koeffizienten in den Schieberegistern dann noch teils auf den alten und teils auf den neuen Werten beruhen. Am Ende des ersten Verbrauchsanstiegs (bei 13 h) kann daher auch nicht nach allen Verbrauchskomponenten aufgelöst werden. Da auf dem oberen Verbrauchsplateau nun aber nur noch c_4 geändert wird, können mehrere Lösungen für $\widetilde{w}_4$ in Folge gefunden werden. Bei oben angegebener Konfiguration werden die ersten beiden Lösungen bereits als verlässlich genug angesehen, um $\widetilde{w}_4 = 35{,}148$ mAs als neue Verbrauchsannahme zu übernehmen. Dadurch steigt aber die Annahme $\widetilde{W}$ des Gesamtverbrauchs über den tatsächlichen Verbrauch W, so dass die anschließende Lotverschiebung die Verbrauchsannahmen wieder dem Ausführungsvektor $\boldsymbol{e}$ entsprechend reduziert. Mit $\widetilde{\boldsymbol{w}} = (22397.776, 8.743, 12.363, 27.761, 34.786)$ mAs wird der Abstand zum tatsächlichen Verbrauchsvektor dabei auf $|\boldsymbol{w} - \widetilde{\boldsymbol{w}}| = 1{,}4$ mAs verkürzt.

Diese Prozedur wiederholt sich, wenn bei 17 h Simulationszeit die Vergrößerung von w_2 und Verkleinerung von w_0 in der Summe zu einem Abfall des Gesamtverbrauchs unter den erwarteten Wert führt. Die erste Kompensation dieser Abweichung erfolgt wieder durch die Lotverschiebung. Erst nach dem Übergang vom oberen zum unteren Verbrauchsplateau liefert die Gauß-Elimination wieder Lösungen, die zur Erhöhung von $\widetilde{w}_2$ auf 18,744 mAs führen. Die bei 22 h gefundene Lösung für $\widetilde{w}_0$ (20788,249 mAs) weicht zu stark von der zuvor bei 5 h gefundenen Lösung (22510,547 mAs) ab, denn

$$\frac{22510,547 - 20788,249}{22510,547} = 7{,}65\ \% > 6{,}25\ \% = \texttt{RA\_ESTIMATOR\_MA\_THRESHOLD} \tag{10.12}$$

Die Bildung des gleitenden Mittelwerts Ω_0 wird daher zurückgesetzt. Da eine einzelne Lösung aber noch nicht als verlässlich angesehen wird, kann $\widetilde{w}_0$ bei 22 h noch nicht reduziert werden. Die Gesamtverbrauchsannahme ist daher nach der Erhöhung von $\widetilde{w}_2$ zu groß und muss durch eine entsprechende Lotverschiebung korrigiert werden. Erst bei 30 h Simulationszeit kann eine weitere Lösung für $\widetilde{w}_0$ gefunden werden, die zur Verringerung der Grundverbrauchsannahme führt. Nach der anschließenden Lotverschiebung beträgt der Abstand der Verbrauchsannahmen $\widetilde{\boldsymbol{w}} = (20784.67, 8.77, 18.63, 26.54, 35.23)$ mAs vom tatsächlichen Verbrauchsvektor gerade noch 0,4 mAs.

10.3.5 Gauß-Elimination bei gestreutem Verbrauch

$$
\begin{aligned}
\delta_{streu} &= \tfrac{1}{64} \\
\boldsymbol{w} &= \begin{cases} (22397.76, 5.79, 12.40, 26.37, 18.36)\ \text{mAs} & \text{bis 7 h} \\ (22397.76, 8.79, 12.40, 26.37, 35.16)\ \text{mAs} & \text{von 7 bis 17 h} \\ (20784.96, 8.79, 18.65, 26.37, 35.16)\ \text{mAs} & \text{ab 17 h} \end{cases} \\
\widetilde{\boldsymbol{w}} &= (22397.76, 5.79, 12.40, 26.37, 18.36)\ \text{mAs} \\
\boldsymbol{r} &= (255, 210, 150, 80, 30, 30, 30, 30, 30, 64, 180, 245, 245, 250, 255, 255, 255)
\end{aligned}
$$

RA_EMU_SCALE	=	64×
RA_ESTIMATOR_GE_SIZE	=	4
RA_ESTIMATOR_GE_CONSISTENCE_TOLERANCE	=	200 mAs
RA_ESTIMATOR_MA_RELIABILITY	=	2
RA_ESTIMATOR_MA_THRESHOLD	=	6,25 %

testbenches/testbench_ra/tasks/estimator/05.cfg

Mit dieser Simulation soll der Einfluss gestreuter Verbrauchswerte auf die Lösungen der Gauß-Elimination untersucht werden. Um überhaupt ausreichend Lösungen zu generieren, muss die Toleranz des Eliminationsverfahrens gegen Inkonsistenzen erhöht werden (vgl. Abschnitt 9.3.4). Die daraus resultierende Verstärkung der Schwankungen der gefundenen Lösungen wird weiterhin durch die Bildung des gleitenden Mittelwertes kontrolliert, so dass ein einzelner Ausreißer nicht auf die Verbrauchsannahmen durchschlagen kann.

Bei den übrigen Parametern stimmt diese Konfiguration der Testbench mit der des letzten Abschnitts überein, so dass der in Abbildung 10.16 gezeigte Simulationsverlauf im Wesentlichen mit dem aus Abbildung 10.14 vergleichbar ist. So erfolgt die Anpassung der Verbrauchsannahme nach der ersten Verschiebung des Verbrauchsvektors ebenfalls nach 15 h Simulationszeit mit $\widetilde{\boldsymbol{w}} = (22397.780, 9.056, 11.471, 27.184, 34.070)$ mAs. Der verbleibende Abstand zu $\boldsymbol{w}$ ist mit 1,7 mAs nicht wesentlich größer als im vorherigen Falle ohne Verbrauchsstreuung.

Die Anpassung der Verbrauchsannahme nach der zweiten Verschiebung des Verbrauchsvektors bei 17 h ist bei dieser Simulation im Vergleich zur letzten um zwei Slots verzögert, weil bei 30 h der Mittelwert Ω_4 wegen eines Ausreißers zurückgesetzt werden muss. Nach zwei weiteren Slots wird dann aber $\widetilde{\boldsymbol{w}} = (21089.501, 8.382, 19.151, 24.214, 35.720)$ mAs erreicht. Der Restabstand von 305 mAs zu $\boldsymbol{w}$ wird vor allem durch die noch zu hohe Grundverbrauchsannahme bedingt. Durch weitere Eliminationslösungen und Lotverschiebungen wird dieser Abstand bis zum Ende der Simulation noch auf 166 mAs gesenkt.

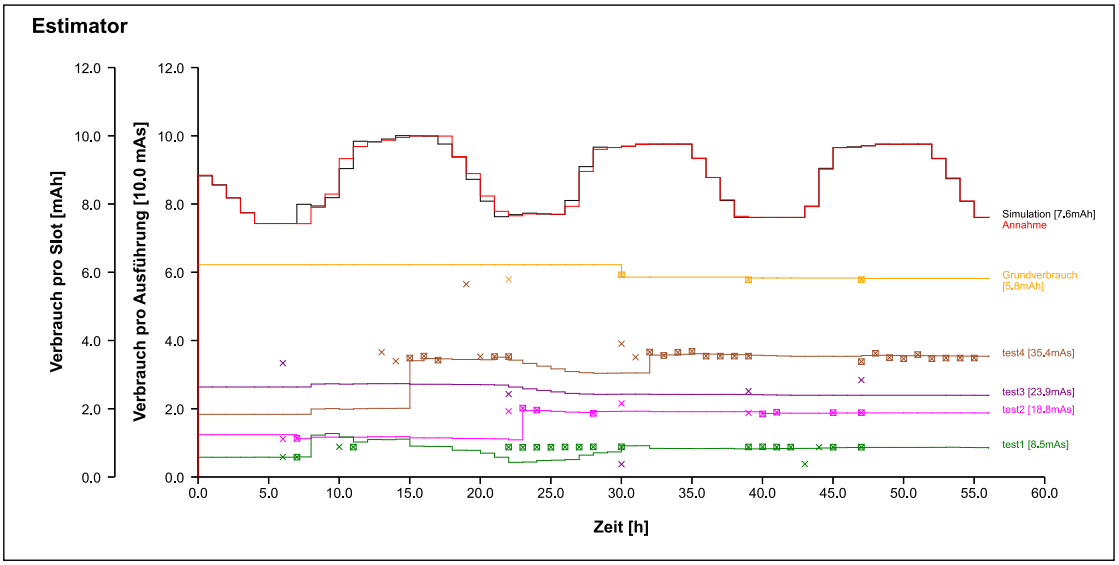


Abbildung 10.16: Gauß-Elimination bei gestreutem Verbrauch

10.4 Steuerung des Energieverbrauchs

Mit den folgenden drei Simulationen soll die Anpassung des Verbrauchsprofils an das simulierte Energieeinnahmeprofil durch den *JIT-Regulator* untersucht werden. Da für diesen Regler nur der Gesamtverbrauch des Sensorknotens, nicht aber die Verbrauchskomponenten der einzelnen Tasks von Bedeutung sind, kann die Simulationskomplexität durch die Verwendung eines einzelnen Testtasks reduziert werden. Programmauszug 10.3 zeigt die Definition dieses Tasks, dessen Verbrauchskomponente nach (10.4) 48,6 mAs beträgt. Zusammen mit dem Grundverbrauch $w_0 = 5{,}7$ mAh bei $T^{\text{main}} = 1$ h kann der Gesamtverbrauch des Sensorknotens zwischen

```
34 RA_DECLARE_SCALABLE_TASK(s, 200, 150, 150, 10 RA_SECONDS, 10 RA_MINUTES, 48600UL, NULL);
35 RA_TASKENTRY(s) {
36   ra_emu_active(30000L, 2000);
37 }
```

Programmauszug 10.3: `testbenches/testbench_ra/regulator/testbench.c`

$$W^{\min} = T^{\text{main}} \sum_{i=0}^{s} \frac{w_i}{c_i^{\max}} = 5{,}78 \text{ mAh} \tag{10.13a}$$

$$\text{und} \quad W^{\max} = T^{\text{main}} \sum_{i=0}^{s} \frac{w_i}{c_i^{\min}} = 10{,}56 \text{ mAh} \tag{10.13b}$$

reguliert werden. Bei 3,3 V Versorgungsspannung gilt daher

$$19{,}07 \text{ mW} \leq \rho_C \leq 34{,}85 \text{ mW} \tag{10.14}$$

für die mittlere Verbraucherleistung. Wegen der Verluste am Spannungswandler, dessen Effizienz bei der Versorgung des Sensorknoten mit 60 % simuliert wird, muss die im Mittel vom *Harvester* erwirtschaftete Leistung daher im Bereich

$$31{,}78 \text{ mW} \leq \rho_H \leq 58{,}08 \text{ mW} \tag{10.15}$$

liegen. Für das im Abschnitt 10.1.2 beschriebene Simulationsmodell der Energiequelle wird dafür nach (10.7) ein Skalierungsfaktor $h^{\max}$ mit

$$90{,}81 \text{ mW} \leq h^{\max} \leq 165{,}94 \text{ mW} \tag{10.16}$$

benötigt.

Im Kapitel 3 wurde eine Primärspeicherkapazität von 364 F bestimmt, welche notwendig ist, um einen 20 mW Verbraucher über 8 h ohne weitere Energieeinnahmen zu versorgen. Bei dem simulierten Einnahmeprofil aus Programmauszug 10.2 muss das System aber über 8,5 h am Stück ohne Einnahmen auskommen. Bei maximaler Verbrauchsleistung wären dann mindestens

$$C_{cap} = 364 \text{ F} \cdot \frac{8{,}5 \text{ h}}{8 \text{ h}} \cdot \frac{35 \text{ mW}}{20 \text{ mW}} = 677 \text{ F} \tag{10.17}$$

zur Überbrückung der Nacht im Primärbetrieb notwendig. Für die folgenden Simulationen wird eine Primärspeicherkapazität von 600 F angenommen, so dass ein permanenter Primärbetrieb nur möglich ist, wenn der Verbrauch in den Nachtstunden weit genug abgesenkt wird.

10.4.1 Verbrauchsadaption bei niedriger Einnahmeleistung

$h^{\max}$	=	121 mW
`RA_EMU_SCALE`	=	64×

`testbenches/testbench_ra/regulator/01.cfg`

Bei dieser Simulation liegt der Solarzellenskalierungsfaktor $h^{\max}$ in der unteren Hälfte seines nach (10.16) beschränkten Bereichs. Abbildung 10.17 zeigt den Ablauf der Simulation. Dabei sind der Primärspeicherfüllstand und die durch U_{cap}^{max}, U_{cap}^{min}, U_{cap}^{hys} und U_{cap}^{dl} vorgegebenen Grenzen im oberen Bereich, die Zustände der Steuerpins des Energiespeichers im mittleren Bereich und das Energieeinnahme- bzw. Verbrauchsprofil im unteren Bereich des Diagramms dargestellt.

Zu Beginn der ersten 24 h Periode hat der *Harvester* noch keine Informationen über den Verlauf der Energieeinnahmen. Der *Regulator* hält die Verbrauchsvorgabe $\widehat{W}$ daher zunächst am unteren Ende des skalierbaren Bereichs. Trotzdem wird dabei noch mehr Energie verbraucht als eingenommen, da das simulierte Einnahmeprofil um 0 *Uhr* startet. Da das System außerdem vom Primärspeicher versorgt wird (`selectBat` = 0), sinkt dessen Füllstand. Da die Energiequelle in den ersten Stunden keine Leistung liefert, der Harvester aber die Verluste am Spannnungswandler (vgl. Abschnitt 10.1.3) auf die Energieeinnahmen anrechnet, entstehen zunächst negative Einträge im Energieeinnahmeprofil.

Sobald die Energieeinnahmen ansteigen und die Ausgaben übersteigen, wird der Primärspeicher mit den Überschüssen geladen. Solange aber die Defizite aus der Anfangsphase diese Überschüsse übersteigen, hält der *Regulator* den Energieverbrauch weiter auf seinem Minimum. Nach etwa 13 h erreicht der Primärspeicherfüllstand seine Obergrenze, so dass ein Energietransport in den Sekundärspeicher (`chargeBat` = 1) ausgelöst wird. Da die im Abschnitt 5.3 beschriebenen Methoden zur Berechnung der optimalen Restenergie im Primärspeicher ein vollständig initialisiertes

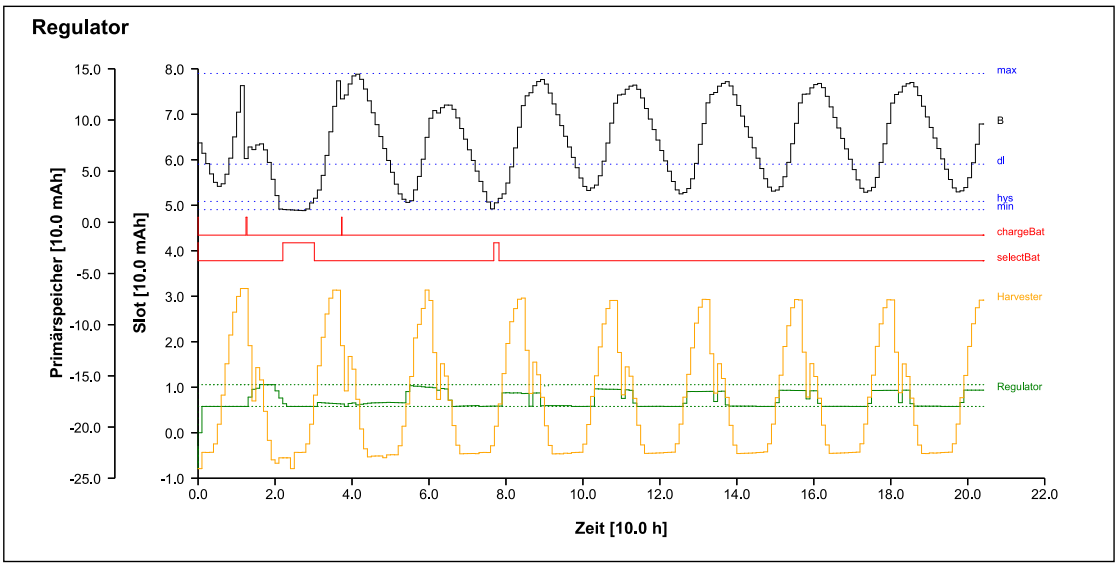


Abbildung 10.17: Anpassung des Energieverbrauchs an das Einnahmeprofil

Energieeinnahmeprofil voraussetzen und damit zu diesem Zeitpunkt nicht angewendet werden können, endet der Ladevorgang erst bei $U_{cap} = U_{cap}^{dl}$.

In der Initialisierungsphase versucht der *Regulator*, den Gesamtübertrag Ψ (vgl. Ausdruck (8.4)) nach jedem Slot auszugleichen. Dies bedingt den Anstieg des Verbrauchs zwischen 14 und 24 h Simulationszeit. Dieses Verbrauchsfenster ist durch die zunächst noch fehlende Einahmevorhersage im Vergleich zum Einnahmeprofil verspätet und wird in den folgenden Tagen weiter nach vorne verschoben, um die Energieüberschüsse und Defizite und damit die notwendige Speicherkapazität zu reduzieren.

Gegen Ende des ersten Tages fällt der Primärspeicherfüllstand auf sein Minimum, so dass auf den Sekundärspeicher umgeschaltet werden muss (`selectBat` = 1). Erst als am darauf folgenden Morgen die einsetzenden Energieeinnahmen den Primärspeicher über seine Hysteresegrenze laden, wird das System wieder aus dem Kondensator versorgt.

Wenn der Primärspeicher nach 37 h Simulationszeit zum zweiten Mal voll geladen ist und damit einen Energietransport bedingt, kann U_{cap}^{ddl} mit der nun vorhandenen Einnahme- und Verbrauchsvorhersage so berechnet werden, dass der Kondensator bis zum Ende der folgenden Nacht nur bis zu seiner Hysteresegrenze entladen wird. Das zwischenzeitliche Umschalten auf den Sekundärspeicher wird so verhindert.

Die absoluten Verluste am Spannungswandler (vgl. Kapitel 3) steigen mit dem Verbrauch des Sensorknotens. Der *Harvester* interpretiert diese Verluste als Verminderung der Energieeinnahmen und der *Regulator* kompensiert eine solche Einnahmeverminderung mit einer Reduktion des Verbrauchs. Diese Rückkopplung ist in Abbildung 10.17 im Verlauf der simulierten Tage erkennbar. In der zweiten Hälfte des ersten Tages wird viel Energie verbraucht, wodurch die im Mittel eingenommene Energie zu gering eingeschätzt wird. Das Verbrauchsprofil für den zweiten Tag fällt deswegen kleiner aus als eigentlich nötig. Dieser geringe Verbrauch am zweiten Tag reduziert die Verluste und vergrößert damit die Annahmen über den Energiegewinn. Darauf hin wird auch der Verbrauch am dritten Tag wieder gesteigert.

Da nur ein Anteil der Verbrauchsänderung über die Verluste zur Einnahmeänderung führt, ist diese Rückkopplung gedämpft und in der Simulation findet ab dem fünften Tag kaum noch eine Änderung des Verbrauchsprofils statt. Da der Primärspeicherfüllstand dann permanent um einen Mittelwert oszilliert, der langfristig weder steigt noch fällt, verhält sich das System in diesem Zustand energieneutral. Da der Energieverbrauch außerdem nur dann über sein Minimum angehoben wird, wenn die Energiequelle einen Überschuss produziert, wird auch die Amplitude der Primärspeicheroszillation und damit die für einen permanenten Primärbetrieb notwendige Kondensatorkapazität reduziert.

10.4.2 Verbrauchsadaption bei hoher Einnahmeleistung

$h^{\max}$	=	149 mW
RA_EMU_SCALE	=	64×

testbenches/testbench_ra/regulator/02.cfg

Abbildung 10.18 zeigt die Simulation der Verbrauchsadaption bei einem Solarzellenskalierungsfaktor im oberen Bereich der Beschränkung (10.16). Auch hier kann der erste Energietransport vom Primär- zum Sekundärspeicher erst bei der statischen Untergrenze der Primärspeicherspannung beendet werden, weil in den ersten 24 h die Einnahmevorhersage noch nicht vollständig ist.

Um den Verbrauchsdurchschnitt soweit anzuheben, dass die langfristig zur Verfügung stehende Energie auch verbraucht wird, muss $\widehat{W}$ auch in den Nachtstunden über den Minimalverbrauch gesteigert werden. Die Kapazität des Kondensators reicht dabei nicht mehr aus, um diesen erhöhten Verbrauch während der gesamten Nacht zu gewährleisten. Um den somit unvermeidlichen Sekundärbetrieb am Ende der Nacht so kurz wie möglich zu halten, wird U_{cap}^{ddl} nur wenig unterhalb von U_{cap}^{max} angesetzt. Es entstehen somit mehrere kurze Ladepeaks und der Primärspeicher bleibt so lange wie möglich annähernd voll geladen.

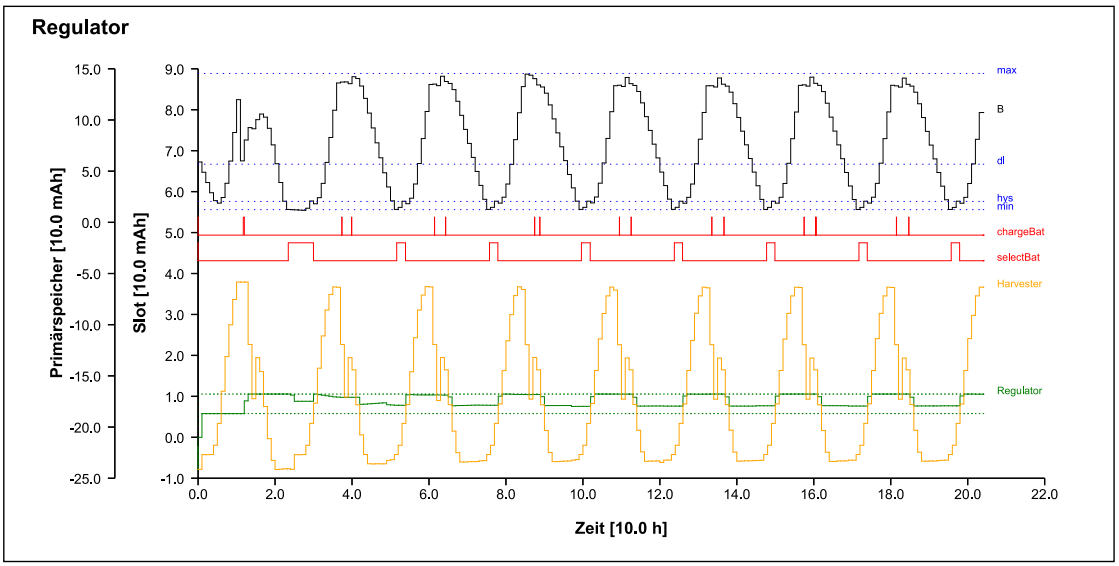


Abbildung 10.18: Anpassung des Energieverbrauchs an das Einnahmeprofil

10.4.3 Verbrauchsadaption bei variabler Einnahmeleistung

$$h^{\max} = \begin{cases} 115\ \text{mW} & \text{am Tag } 2, 6, 10, \ldots \\ 145\ \text{mW} & \text{am Tag } 4, 8, 12, \ldots \\ 130\ \text{mW} & \text{sonst} \end{cases}$$

$$\texttt{RA\_EMU\_SCALE} = 64\times$$

`testbenches/testbench_ra/regulator/03.cfg`

Mit dieser Simulation soll die Korrektur von Vorhersagefehlern durch den *JIT-Regulator* dargestellt werden. Dafür wird der Skalierungsfaktor $h^{\max}$ nach jedem Tag geändert, wobei die Sequenz (130, 115, 130, 145) mAs periodisch wiederholt wird. Zu jedem Stützpunkt der Einnahmesimulation wird außerdem ein zufälliger Wert zwischen -8 mW und +8 mW addiert.

Abbildung 10.19 zeigt das Ergebnis dieser Simulation. Das Verbrauchsprofil wird immer am Anfang einer 24 h Periode aus dem JIT-Profil abgeleitet, indem die Verbrauchsvorgaben am Tag gesenkt oder in der Nacht erhöht werden, bis der für die Energieneutralität notwendige Durchschnittsverbrauch erreicht ist. Wird nun aber während der Umsetzung des Profils festgestellt, dass die tatsächlichen Energieeinnahmen von der Vorhersage abweichen, so muss das Profil entsprechend angepasst werden. Für das Verbrauchsprofil des zweiten Tages wird im Wesentlichen das Einnahmeprofil des ersten Tages als Vorhersage verwendet. Dessen Durchschnitt würde es

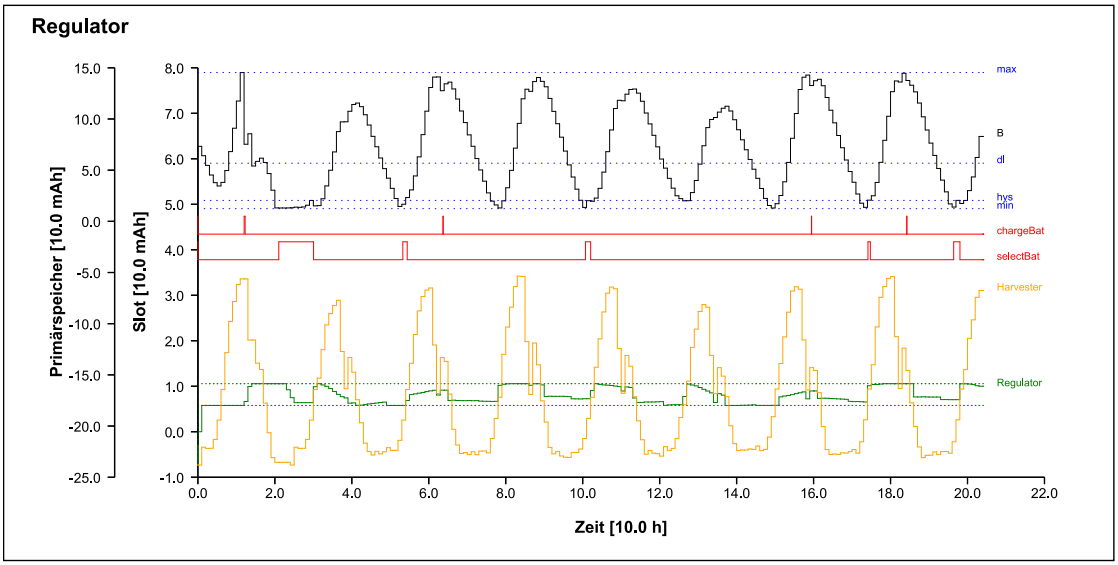


Abbildung 10.19: Anpassung des Energieverbrauchs an das Einnahmeprofil

ermöglichen, den Verbrauch tagsüber zu maximieren. Mit jedem Slot des zweiten Tages stellt der *Regulator* nun aber fest, dass weniger Energie eingenommen wird, als vermutet. Die gleichmäßige Verringerung des Verbrauchsprofils kann sich aber nur noch auf zukünftige Verbrauchsvorgaben auswirken. Da eine solche Kompensation nach jedem Slot des Tages notwendig wird, entsteht eine absteigende Treppe im Verbrauchsprofil.

Umgekehrt beruht das initiale Verbrauchsprofil des dritten Tages im Wesentlichen auf den geringen Einnahmen des Vortages und muss daher im Laufe des Tages schrittweise angehoben werden. Der Hauptteil der Energie wird nach Abbildung 10.19 aber weiterhin in den Zeiten des Einnahmeüberschusses verbraucht.

KAPITEL

11 Zusammenfassung

Im ersten Teil dieser Arbeit wurden Hardwarekonzepte zum Gewinnen und Speichern von Solarenergie vorgestellt und die Charakteristiken der langfristigen Schwankungen der zur Verfügung stehenden Solarenergie sowie deren Abhängigkeit von der statischen Ausrichtung des Solarmoduls untersucht. Dabei wurde festgestellt, dass die Speicherkapazität, die notwendig ist, um den Energieüberschuss der Sommermonate bis in den Winter zu puffern, zu groß ist, um sie mit herkömmlichen Speichertechnologien in der Größe des SNoW^5-Sensorknotens zu realisieren. Für einen mehrjährigen energieneutralen Betrieb des Sensorknotens müsste demnach zunächst dessen Grundverbrauch und damit die notwendige Größe des Solarmoduls und des Energiespeichers reduziert werden.

Im zweiten Teil der Arbeit wurde eine möglichst anwendungsunabhängige Möglichkeit zur Steuerung des Energieverbrauchs des Sensorknotens vorgestellt und dessen Implementierung beschrieben. Voraussetzung für die Anwendbarkeit dieses Konzepts ist die Existenz periodisch auszuführender Aufgaben, deren Ausführungshäufigkeit kontinuierlich zwischen einer unteren und einer oberen Grenze skaliert werden kann. Die zur Verfügung stehende Energie wird dann nutzenoptimiert auf die verschiedenen Aufgaben verteilt, wobei der relative energetische Nutzen der einzelnen Aufgaben zur Laufzeit angepasst werden kann.

Die Möglichkeit der Verbrauchssteuerung wird von einem Regler verwendet, um die Energieausgabe so gut wie möglich an das Energieeinnahmeprofil anzupassen und auf lange Sicht gerade so viel zu verbrauchen, wie von der Energiequelle bereitgestellt wird. Je größer dabei der Anteil des skalierbaren Verbrauchs am Gesamtverbrauch des Sensorknotens ist, desto besser können die Schwankungen der Solarenergie durch den Verbraucher adaptiert werden und desto kleiner ist die Kapazität des benötigten Energiespeichers. Da der Energiebedarf des SNoW^5-Sensorknotens im *idle*-Modus über 60 % des Bedarfs im aktiven Modus ausmacht [2], wird der nicht skalierbare Grundverbrauch immer einen signifikanten Anteil am Gesamtverbrauch aus-

machen. Die Energiequelle muss demnach genau dimensioniert werden, um eine permanente Über- oder Unterversorgung des Systems auszuschließen.

An dieser Stelle unterscheiden sich aber Theorie und Praxis. Während die Solarmodulgröße für die Emulationen in Kapitel 10 kontinuierlich skaliert werden kann, sind kommerziell vertriebene Solarzellen und Module nur in diskreten Leistungsabstufungen erhältlich. Zusammen mit den Problemen bei der Pufferung langfristiger Einnahmeschwankungen muss man realistischer Weise erkennen, dass die Umsetzung eines energieneutralen Betriebs des SNoW^5-Sensorknotens mittels einer Solarenergiequelle trotz funktionierender Verbrauchssteuerung mit einigen Schwierigkeiten bezüglich der Systemdimensionierung verbunden ist.

Mit einer Überdimensionierung kann man zwar ebenfalls für nahezu unbegrenzte Laufzeiten sorgen, eine Verbrauchsanpassung mit den in dieser Arbeit vorgestellten Methoden wäre dann aber höchstens noch zur Überbrückung der Nacht mit der verbleibenden Primärspeicherenergie sinnvoll. Durch die modulare Implementierung der SNoW^5-RA-Software ist es aber auch ohne weiteres möglich, die Verbrauchssteuerung über die Skalierung periodischer Tasks für andere Anwendungen zu nutzen. Für die Verwendung einer alternativen Energiequelle (bspw. Thermo- oder Piezogeneratoren) muss lediglich die Implementierung der *Harvesters* und ggf. des *Regulators* geändert werden, während das *Hardware Layer*, der *Estimator*, das *Adaptive Duty Cycling* und der *Wrapper* beibehalten werden können.

Anhang

ANHANG

A Solardaten

Die folgenden Tabellen zeigen den durchschnittlichen Tagesverlauf der solaren Bestrahlungsstärke auf eine in Würzburg (49°47′6″ N, 9°56′30″ E) aufgestellte, nach Süden ausgerichtete und um den Inklinationswinkel α gegen die Horizontale geneigte Fläche für die verschiedenen Monate eines Jahres. Die Daten stammen vom Photovoltaischen Geographischen Informationssystem (PVGIS) der Europäischen Kommission [29] und beruhen auf empirischen Beobachtungen im Zeitraum von 1981-1990 und räumlichen Interpolationsmodellen. Sie berücksichtigen neben der Dynamik des Einfallwinkels der Sonnenstrahlen auch Geländeverschattungen (Berge), reflektiertes und gestreutes Licht sowie den Einfluss atmosphärischer Störungen (Gase, Aerosole und Wolken).

Tabelle A.1: Bestrahlungsstärke [W/m^2] in Würzburg unter 0° Inklination [29]

Tageszeit	Jan	Feb	Mrz	Apr	Mai	Jun	Jul	Aug	Sep	Okt	Nov	Dez
04:08						21						
04:23						37	26					
04:38					26	57	46					
04:53					46	77	66					
05:08					67	98	88	27				
05:23				23	89	120	112	48				
05:38				43	112	142	137	70				
05:53				64	136	165	162	93	17			
06:08				86	161	189	188	117	33			
06:23			28	109	185	212	214	142	55			
06:38			47	133	210	234	240	168	77			
06:53			66	157	235	257	265	193	100	24		
07:08		18	86	181	259	279	290	218	123	41		
07:23		30	106	205	282	300	315	243	147	58		
07:38		48	126	228	305	320	338	267	170	75	16	
07:53	15	64	146	250	327	340	361	290	193	93	26	
08:08	24	81	165	272	347	358	383	313	216	111	40	
08:23	36	98	183	292	367	376	403	334	237	128	53	15
08:38	47	114	200	311	386	392	422	354	258	145	65	22
08:53	57	129	217	329	403	407	440	373	277	161	77	33
09:08	68	144	232	346	419	421	457	391	295	176	89	41
09:23	77	158	246	362	434	434	473	407	312	189	99	50
09:38	86	171	259	376	448	446	487	422	328	202	110	59
09:53	95	182	271	389	460	457	499	436	342	213	119	66
10:08	102	193	281	401	471	466	511	447	354	224	127	73
10:23	109	202	290	411	480	474	520	458	366	233	134	80
10:38	115	210	298	419	488	481	529	467	375	241	141	86
10:53	119	217	304	426	495	487	536	475	383	247	146	91
11:08	123	222	310	432	500	492	542	481	389	252	150	95
11:23	126	226	314	436	504	495	546	485	394	256	154	98
11:38	128	229	316	439	507	498	549	488	397	259	156	101
11:53	129	230	317	441	508	499	550	490	399	260	157	103
12:08	129	230	317	441	508	499	550	490	399	260	157	103
12:23	128	229	316	439	507	498	549	488	397	259	156	103
12:38	126	226	314	436	504	495	546	485	394	256	154	103
12:53	123	222	310	432	500	492	542	481	389	252	150	101
13:08	119	217	304	426	495	487	536	475	383	247	146	98
13:23	115	210	298	419	488	481	529	467	375	241	141	95
13:38	109	202	290	411	480	474	520	458	366	233	134	91
13:53	102	193	281	401	471	466	511	447	354	224	127	86
14:08	95	182	271	389	460	457	499	436	342	213	119	80
14:23	86	171	259	376	448	446	487	422	328	202	110	73
14:38	77	158	246	362	434	434	473	407	312	189	99	66
14:53	68	144	232	346	419	421	457	391	295	176	89	59
15:08	57	129	217	329	403	407	440	373	277	161	77	50
15:23	47	114	200	311	386	392	422	354	258	145	65	41
15:38	36	98	183	292	367	376	403	334	237	128	53	33
15:53	26	81	165	272	347	358	383	313	216	111	40	24
16:08	15	64	146	250	327	340	361	290	193	93	28	15
16:23		48	126	228	305	320	338	267	170	75	16	
16:38		30	106	205	282	300	315	243	147	58		
16:53		18	86	181	259	279	290	218	123	41		
17:08			66	157	235	257	265	193	100	24		
17:23			47	133	210	234	240	168	77			
17:38			28	109	185	212	214	142	55			
17:53				86	161	189	188	117	36			
18:08				64	136	165	162	93	17			
18:23				43	112	142	137	70				
18:38				23	89	120	112	48				
18:53					67	98	88	28				
19:08					46	77	66					
19:23					27	57	46					
19:38						39	27					
19:53						22						

Tabelle A.2: Bestrahlungsstärke [W/m^2] in Würzburg unter 10° Inklination [29]

Tageszeit	Jan	Feb	Mrz	Apr	Mai	Jun	Jul	Aug	Sep	Okt	Nov	Dez
04:08						21						
04:23						36	26					
04:38					26	48	42					
04:53					42	68	57					
05:08					60	88	77	27				
05:23				23	80	109	100	43				
05:38				39	103	131	125	65				
05:53				61	127	155	150	88	17			
06:08				83	153	178	177	113	32			
06:23			27	107	178	202	204	138	53			
06:38			47	133	204	226	232	165	79			
06:53			69	158	231	250	259	192	104	23		
07:08		17	91	184	257	274	286	219	129	43		
07:23		29	113	210	282	297	313	246	155	64		
07:38		53	135	235	307	319	339	273	181	85	16	
07:53	15	75	157	260	331	341	364	299	207	105	26	
08:08	24	94	178	284	354	361	388	324	232	126	46	
08:23	41	114	199	307	376	381	412	348	257	146	63	14
08:38	56	133	218	329	397	399	433	370	280	165	78	22
08:53	68	151	237	349	417	416	454	392	302	183	92	38
09:08	81	168	254	369	435	432	473	412	323	200	106	50
09:23	92	185	270	387	452	447	491	430	343	216	119	60
09:38	103	200	285	403	468	460	507	447	361	231	131	70
09:53	113	213	298	418	482	473	521	463	377	244	142	79
10:08	122	226	310	431	494	483	534	477	392	256	151	88
10:23	130	236	321	442	505	493	546	489	404	267	160	96
10:38	137	246	330	452	514	501	556	499	416	276	167	103
10:53	142	254	337	461	522	508	564	508	425	283	174	108
11:08	147	260	343	467	528	513	570	515	432	289	179	113
11:23	151	265	348	472	533	517	575	520	438	294	182	117
11:38	153	268	351	476	536	520	578	524	442	297	185	120
11:53	154	270	352	477	538	521	580	525	444	299	186	122
12:08	154	270	352	477	538	521	580	525	444	299	186	123
12:23	153	268	351	476	536	520	578	524	442	297	185	123
12:38	151	265	348	472	533	517	575	520	438	294	182	122
12:53	147	260	343	467	528	513	570	515	432	289	179	120
13:08	142	254	337	461	522	508	564	508	425	283	174	117
13:23	137	246	330	452	514	501	556	499	416	276	167	113
13:38	130	236	321	442	505	493	546	489	404	267	160	108
13:53	122	226	310	431	494	483	534	477	392	256	151	103
14:08	113	213	298	418	482	473	521	463	377	244	142	96
14:23	103	200	285	403	468	460	507	447	361	231	131	88
14:38	92	185	270	387	452	447	491	430	343	216	119	79
14:53	81	168	254	369	435	432	473	412	323	200	106	70
15:08	68	151	237	349	417	416	454	392	302	183	92	60
15:23	56	133	218	329	397	399	433	370	280	165	78	50
15:38	41	114	199	307	376	381	412	348	257	146	63	38
15:53	29	94	178	284	354	361	388	324	232	126	46	27
16:08	15	75	157	260	331	341	364	299	207	105	32	14
16:23		53	135	235	307	319	339	273	181	85	16	
16:38		29	113	210	282	297	313	246	155	64		
16:53		17	91	184	257	274	286	219	129	43		
17:08			69	158	231	250	259	192	104	23		
17:23			47	133	204	226	232	165	79			
17:38			27	107	178	202	204	138	53			
17:53				83	153	178	177	113	34			
18:08				61	127	155	150	88	17			
18:23				39	103	131	125	65				
18:38				23	80	109	100	43				
18:53					60	88	77	27				
19:08					42	68	57					
19:23					26	48	42					
19:38						36	26					
19:53						21						

Tabelle A.3: Bestrahlungsstärke [W/m^2] in Würzburg unter 20° Inklination [29]

Tageszeit	Jan	Feb	Mrz	Apr	Mai	Jun	Jul	Aug	Sep	Okt	Nov	Dez
04:08						20						
04:23						35	25					
04:38					25	50	40					
04:53					41	64	55					
05:08					56	77	70	26				
05:23				22	71	97	87	42				
05:38				36	93	118	111	59				
05:53				57	117	142	136	82	16			
06:08				79	142	166	163	106	31			
06:23			26	104	168	190	191	132	53			
06:38			48	130	195	215	220	160	79			
06:53			71	157	223	240	248	188	105	23		
07:08		17	94	184	250	265	277	217	133	46		
07:23		28	118	212	277	289	306	245	161	70		
07:38		59	142	239	304	313	334	274	189	93	16	
07:53	14	83	166	266	330	336	361	301	217	115	25	
08:08	23	106	189	291	355	358	387	328	245	138	53	
08:23	47	128	211	316	379	379	412	354	272	160	72	14
08:38	64	149	232	340	401	399	436	379	297	181	89	21
08:53	78	170	253	363	423	417	458	403	322	202	106	43
09:08	93	189	272	384	443	435	479	425	345	221	121	57
09:23	106	208	289	403	461	451	499	445	366	239	136	69
09:38	118	225	305	421	478	466	517	464	386	255	149	80
09:53	129	240	320	438	494	479	533	481	404	270	162	91
10:08	140	254	333	452	507	491	547	496	421	283	173	101
10:23	148	266	345	465	520	501	560	509	435	295	182	110
10:38	156	277	355	476	530	510	571	521	447	305	191	117
10:53	163	285	363	485	539	518	580	531	458	314	198	124
11:08	168	293	370	493	546	524	587	538	466	321	203	130
11:23	172	298	375	498	551	528	593	544	472	326	208	134
11:38	174	301	378	502	554	531	596	548	477	329	211	138
11:53	176	303	380	504	556	533	598	550	479	331	212	140
12:08	176	303	380	504	556	533	598	550	479	331	212	141
12:23	174	301	378	502	554	531	596	548	477	329	211	141
12:38	172	298	375	498	551	528	593	544	472	326	208	140
12:53	168	293	370	493	546	524	587	538	466	321	203	138
13:08	163	285	363	485	539	518	580	531	458	314	198	134
13:23	156	277	355	476	530	510	571	521	447	305	191	130
13:38	148	266	345	465	520	501	560	509	435	295	182	124
13:53	140	254	333	452	507	491	547	496	421	283	173	117
14:08	129	240	320	438	494	479	533	481	404	270	162	110
14:23	118	225	305	421	478	466	517	464	386	255	149	101
14:38	106	208	289	403	461	451	499	445	366	239	136	91
14:53	93	189	272	384	443	435	479	425	345	221	121	80
15:08	78	170	253	363	423	417	458	403	322	202	106	69
15:23	64	149	232	340	401	399	436	379	297	181	89	57
15:38	47	128	211	316	379	379	412	354	272	160	72	43
15:53	33	106	189	291	355	358	387	328	245	138	53	31
16:08	14	83	166	266	330	336	361	301	217	115	36	14
16:23		59	142	239	304	313	334	274	189	93	16	
16:38		28	118	212	277	289	306	245	161	70		
16:53		17	94	184	250	265	277	217	133	46		
17:08			71	157	223	240	248	188	105	23		
17:23			48	130	195	215	220	160	79			
17:38			26	104	168	190	191	132	53			
17:53				79	142	166	163	106	33			
18:08				57	117	142	136	82	16			
18:23				36	93	118	111	59				
18:38				22	71	97	87	42				
18:53					56	77	70	26				
19:08					41	64	55					
19:23					25	50	40					
19:38						35	25					
19:53						20						

Tabelle A.4: Bestrahlungsstärke [W/m^2] in Würzburg unter 30° Inklination [29]

Tageszeit	Jan	Feb	Mrz	Apr	Mai	Jun	Jul	Aug	Sep	Okt	Nov	Dez
04:08						19						
04:23						33	24					
04:38					24	47	38					
04:53					39	61	52					
05:08					53	74	66	25				
05:23				21	67	87	80	40				
05:38				36	82	104	95	55				
05:53				52	105	126	120	74	16			
06:08				74	129	150	147	98	30			
06:23			25	99	156	175	175	124	52			
06:38			48	125	183	200	204	152	79			
06:53			72	153	211	225	233	181	106	21		
07:08		16	96	181	239	251	263	210	134	49		
07:23		27	121	209	267	276	293	240	163	74		
07:38		64	146	238	295	300	322	269	193	99	15	
07:53	14	91	171	266	322	324	350	298	223	123	24	
08:08	22	115	195	293	348	347	378	327	252	148	58	
08:23	52	140	219	319	373	370	404	354	281	172	80	13
08:38	70	163	242	345	397	391	429	380	308	195	99	20
08:53	87	185	263	368	420	410	453	405	334	216	117	48
09:08	102	207	284	391	441	429	476	429	359	237	134	63
09:23	117	226	302	412	461	446	496	450	382	256	150	77
09:38	131	245	320	431	479	462	515	470	404	274	165	89
09:53	143	261	336	448	495	476	533	489	423	290	178	101
10:08	154	277	350	464	510	489	548	505	441	304	190	112
10:23	164	290	362	478	523	500	562	519	456	317	201	121
10:38	172	301	373	490	534	509	573	532	469	328	210	130
10:53	179	311	382	500	544	517	583	542	480	337	218	137
11:08	185	319	389	508	551	524	591	550	489	345	224	143
11:23	189	324	394	514	557	529	597	557	496	350	228	148
11:38	192	328	398	518	560	532	601	561	501	354	232	152
11:53	194	330	400	520	562	533	603	563	503	356	233	154
12:08	194	330	400	520	562	533	603	563	503	356	233	156
12:23	192	328	398	518	560	532	601	561	501	354	232	156
12:38	189	324	394	514	557	529	597	557	496	350	228	154
12:53	185	319	389	508	551	524	591	550	489	345	224	152
13:08	179	311	382	500	544	517	583	542	480	337	218	148
13:23	172	301	373	490	534	509	573	532	469	328	210	143
13:38	164	290	362	478	523	500	562	519	456	317	201	137
13:53	154	277	350	464	510	489	548	505	441	304	190	130
14:08	143	261	336	448	495	476	533	489	423	290	178	121
14:23	131	245	320	431	479	462	515	470	404	274	165	112
14:38	117	226	302	412	461	446	496	450	382	256	150	101
14:53	102	207	284	391	441	429	476	429	359	237	134	89
15:08	87	185	263	368	420	410	453	405	334	216	117	77
15:23	70	163	242	345	397	391	429	380	308	195	99	63
15:38	52	140	219	319	373	370	404	354	281	172	80	48
15:53	36	115	195	293	348	347	378	327	252	148	58	33
16:08	14	91	171	266	322	324	350	298	223	123	39	13
16:23		64	146	238	295	300	322	269	193	99	15	
16:38		27	121	209	267	276	293	240	163	74		
16:53		16	96	181	239	251	263	210	134	49		
17:08			72	153	211	225	233	181	106	21		
17:23			48	125	183	200	204	152	79			
17:38			25	99	156	175	175	124	52			
17:53				74	129	150	147	98	32			
18:08				52	105	126	120	74	16			
18:23				36	82	104	95	55				
18:38				21	67	87	80	40				
18:53					53	74	66	25				
19:08					39	61	52					
19:23					24	47	38					
19:38						33	24					
19:53						19						

Tabelle A.5: Bestrahlungsstärke [W/m^2] in Würzburg unter 34° Inklination [29]

Tageszeit	Jan	Feb	Mrz	Apr	Mai	Jun	Jul	Aug	Sep	Okt	Nov	Dez
04:08						19						
04:23						32	23					
04:38					23	46	37					
04:53					37	59	51					
05:08					52	72	64	24				
05:23				20	65	85	78	39				
05:38				35	78	98	89	53				
05:53				51	99	120	113	71	15			
06:08				72	124	143	140	94	29			
06:23			24	96	150	168	168	120	51			
06:38			48	123	177	193	197	148	78			
06:53			72	150	205	218	226	177	105	21		
07:08		15	96	179	233	244	256	207	134	50		
07:23		26	121	207	262	269	286	236	164	76		
07:38		66	147	236	290	294	315	266	194	101	14	
07:53	13	93	172	264	317	318	344	296	224	126	23	
08:08	21	118	197	292	343	341	372	324	254	151	60	
08:23	54	143	221	319	369	364	399	352	283	175	82	13
08:38	73	167	244	344	393	385	424	379	311	199	102	19
08:53	90	190	266	369	416	405	449	404	337	221	121	49
09:08	106	212	287	391	438	424	471	428	363	242	138	65
09:23	121	232	306	413	458	441	492	450	386	262	155	79
09:38	135	251	324	432	476	457	512	470	408	280	170	92
09:53	148	268	340	450	493	472	529	489	428	296	184	104
10:08	159	284	354	466	508	485	545	505	446	311	196	115
10:23	169	297	367	480	521	496	559	520	461	324	207	125
10:38	178	309	378	492	533	506	571	533	475	335	216	134
10:53	185	319	387	502	542	514	581	543	486	345	224	142
11:08	191	327	394	510	550	520	589	552	496	352	231	148
11:23	195	333	400	517	556	525	595	558	503	358	235	153
11:38	198	337	403	521	559	529	599	562	507	362	239	157
11:53	200	339	405	523	561	530	601	564	510	363	240	159
12:08	200	339	405	523	561	530	601	564	510	363	240	161
12:23	198	337	403	521	559	529	599	562	507	362	239	161
12:38	195	333	400	517	556	525	595	558	503	358	235	159
12:53	191	327	394	510	550	520	589	552	496	352	231	157
13:08	185	319	387	502	542	514	581	543	486	345	224	153
13:23	178	309	378	492	533	506	571	533	475	335	216	148
13:38	169	297	367	480	521	496	559	520	461	324	207	142
13:53	159	284	354	466	508	485	545	505	446	311	196	134
14:08	148	268	340	450	493	472	529	489	428	296	184	125
14:23	135	251	324	432	476	457	512	470	408	280	170	115
14:38	121	232	306	413	458	441	492	450	386	262	155	104
14:53	106	212	287	391	438	424	471	428	363	242	138	92
15:08	90	190	266	369	416	405	449	404	337	221	121	79
15:23	73	167	244	344	393	385	424	379	311	199	102	65
15:38	54	143	221	319	369	364	399	352	283	175	82	49
15:53	37	118	197	292	343	341	372	324	254	151	60	34
16:08	13	93	172	264	317	318	344	296	224	126	41	13
16:23		66	147	236	290	294	315	266	194	101	14	
16:38		26	121	207	262	269	286	236	164	76		
16:53		15	96	179	233	244	256	207	134	50		
17:08			72	150	205	218	226	177	105	21		
17:23			48	123	177	193	197	148	78			
17:38			24	96	150	168	168	120	51			
17:53				72	124	143	140	94	32			
18:08				51	99	120	113	71	15			
18:23				35	78	98	89	53				
18:38				20	65	85	78	39				
18:53					52	72	64	24				
19:08					37	59	51					
19:23					23	46	37					
19:38						32	23					
19:53						19						

Tabelle A.6: Bestrahlungsstärke [W/m^2] in Würzburg unter 40° Inklination [29]

Tageszeit	Jan	Feb	Mrz	Apr	Mai	Jun	Jul	Aug	Sep	Okt	Nov	Dez
04:08						18						
04:23						31	22					
04:38					22	44	35					
04:53					36	56	49					
05:08					49	69	61	23				
05:23				19	62	81	74	37				
05:38				33	75	92	86	50				
05:53				46	91	110	102	66	14			
06:08				68	115	133	128	89	27			
06:23			23	92	141	157	156	114	50			
06:38			47	118	168	182	185	142	77			
06:53			71	146	196	207	214	170	104	20		
07:08		15	96	174	224	232	244	200	133	50		
07:23		25	121	203	252	258	274	230	163	77		
07:38		68	147	232	280	283	304	260	194	103	14	
07:53	13	96	173	261	308	307	333	290	224	129	22	
08:08	20	122	198	289	335	330	361	319	255	154	63	
08:23	56	148	223	316	360	353	388	347	284	179	86	12
08:38	76	173	246	342	385	375	414	374	313	203	106	19
08:53	93	197	268	366	408	395	439	399	340	226	126	51
09:08	110	219	290	390	430	414	462	423	365	248	144	68
09:23	126	240	309	411	451	432	483	446	389	268	161	82
09:38	140	259	327	431	469	448	503	467	412	286	177	96
09:53	154	277	344	449	486	463	521	485	432	303	191	109
10:08	165	293	358	466	502	476	537	502	450	319	204	120
10:23	176	307	371	480	515	487	551	517	466	332	215	130
10:38	185	319	383	492	527	497	563	530	480	343	225	140
10:53	192	329	392	503	536	506	573	541	492	353	233	147
11:08	198	337	399	511	544	512	582	550	501	361	239	154
11:23	203	344	405	517	550	517	588	556	509	367	244	159
11:38	206	348	409	521	554	521	592	561	513	371	247	163
11:53	207	350	411	524	556	522	594	563	516	372	249	166
12:08	207	350	411	524	556	522	594	563	516	372	249	167
12:23	206	348	409	521	554	521	592	561	513	371	247	167
12:38	203	344	405	517	550	517	588	556	509	367	244	166
12:53	198	337	399	511	544	512	582	550	501	361	239	163
13:08	192	329	392	503	536	506	573	541	492	353	233	159
13:23	185	319	383	492	527	497	563	530	480	343	225	154
13:38	176	307	371	480	515	487	551	517	466	332	215	147
13:53	165	293	358	466	502	476	537	502	450	319	204	140
14:08	154	277	344	449	486	463	521	485	432	303	191	130
14:23	140	259	327	431	469	448	503	467	412	286	177	120
14:38	126	240	309	411	451	432	483	446	389	268	161	109
14:53	110	219	290	390	430	414	462	423	365	248	144	96
15:08	93	197	268	366	408	395	439	399	340	226	126	82
15:23	76	173	246	342	385	375	414	374	313	203	106	68
15:38	56	148	223	316	360	353	388	347	284	179	86	51
15:53	38	122	198	289	335	330	361	319	255	154	63	36
16:08	13	96	173	261	308	307	333	290	224	129	42	12
16:23		68	147	232	280	283	304	260	194	103	14	
16:38		25	121	203	252	258	274	230	163	77		
16:53		15	96	174	224	232	244	200	133	50		
17:08			71	146	196	207	214	170	104	20		
17:23			47	118	168	182	185	142	77			
17:38			23	92	141	157	156	114	50			
17:53				68	115	133	128	89	31			
18:08				46	91	110	102	66	14			
18:23				33	75	92	86	50				
18:38				19	62	81	74	37				
18:53					49	69	61	23				
19:08					36	56	49					
19:23					22	44	35					
19:38						31	22					
19:53						18						

Tabelle A.7: Bestrahlungsstärke [W/m^2] in Würzburg unter 50° Inklination [29]

Tageszeit	Jan	Feb	Mrz	Apr	Mai	Jun	Jul	Aug	Sep	Okt	Nov	Dez
04:08						16						
04:23						28	20					
04:38					20	40	32					
04:53					32	51	44					
05:08					45	62	56	21				
05:23				18	56	73	67	33				
05:38				30	68	84	78	46				
05:53				42	79	94	89	58	13			
06:08				61	99	114	108	78	25			
06:23			21	84	124	137	135	102	48			
06:38			46	109	150	161	163	129	73			
06:53			70	136	177	185	192	157	100	18		
07:08		13	94	164	205	210	221	186	129	51		
07:23		23	120	193	233	235	250	216	159	79		
07:38		71	146	222	260	260	280	245	190	105	12	
07:53	11	100	172	250	288	284	309	275	221	132	20	
08:08	18	127	197	278	314	307	337	304	251	158	66	
08:23	59	153	222	305	340	330	364	332	281	183	90	11
08:38	80	179	245	332	365	351	390	359	310	208	111	17
08:53	98	204	268	356	388	371	415	385	337	231	131	54
09:08	116	227	289	380	410	391	438	409	363	253	150	71
09:23	132	248	309	402	431	408	459	432	388	274	168	87
09:38	147	268	327	422	450	425	479	452	410	293	184	101
09:53	161	286	344	440	467	439	497	472	431	310	199	114
10:08	173	302	359	456	482	453	513	489	449	325	212	126
10:23	184	317	372	471	496	464	528	504	466	339	224	137
10:38	193	329	384	484	507	474	540	517	480	351	234	146
10:53	201	340	393	494	517	483	550	528	492	361	242	154
11:08	207	348	401	502	525	489	559	537	501	369	249	161
11:23	212	354	406	509	531	495	565	543	509	374	254	166
11:38	215	359	410	513	535	498	569	548	513	378	257	170
11:53	216	361	412	515	537	500	571	550	516	380	259	173
12:08	216	361	412	515	537	500	571	550	516	380	259	174
12:23	215	359	410	513	535	498	569	548	513	378	257	174
12:38	212	354	406	509	531	495	565	543	509	374	254	173
12:53	207	348	401	502	525	489	559	537	501	369	249	170
13:08	201	340	393	494	517	483	550	528	492	361	242	166
13:23	193	329	384	484	507	474	540	517	480	351	234	161
13:38	184	317	372	471	496	464	528	504	466	339	224	154
13:53	173	302	359	456	482	453	513	489	449	325	212	146
14:08	161	286	344	440	467	439	497	472	431	310	199	137
14:23	147	268	327	422	450	425	479	452	410	293	184	126
14:38	132	248	309	402	431	408	459	432	388	274	168	114
14:53	116	227	289	380	410	391	438	409	363	253	150	101
15:08	98	204	268	356	388	371	415	385	337	231	131	87
15:23	80	179	245	332	365	351	390	359	310	208	111	71
15:38	59	153	222	305	340	330	364	332	281	183	90	54
15:53	40	127	197	278	314	307	337	304	251	158	66	37
16:08	11	100	172	250	288	284	309	275	221	132	44	11
16:23		71	146	222	260	260	280	245	190	105	12	
16:38		23	120	193	233	235	250	216	159	79		
16:53		13	94	164	205	210	221	186	129	51		
17:08			70	136	177	185	192	157	100	18		
17:23			46	109	150	161	163	129	73			
17:38			21	84	124	137	135	102	48			
17:53				61	99	114	108	78	29			
18:08				42	79	94	89	58	13			
18:23				30	68	84	78	46				
18:38				18	56	73	67	33				
18:53					45	62	56	21				
19:08					32	51	44					
19:23					20	40	32					
19:38						28	20					
19:53						16						

Tabelle A.8: Bestrahlungsstärke [W/m^2] in Würzburg unter 60° Inklination [29]

Tageszeit	Jan	Feb	Mrz	Apr	Mai	Jun	Jul	Aug	Sep	Okt	Nov	Dez
04:08						14						
04:23						25	18					
04:38					18	35	29					
04:53					29	46	39					
05:08					40	56	50	18				
05:23				16	50	65	60	30				
05:38				27	60	75	70	41				
05:53				37	70	84	79	51	12			
06:08				54	79	92	88	62	22			
06:23			19	75	105	115	112	89	45			
06:38			44	99	130	137	138	114	69			
06:53			67	125	155	161	166	141	95	16		
07:08		12	91	152	182	184	194	169	123	51		
07:23		20	116	179	209	208	222	197	152	79		
07:38		72	141	207	235	232	250	226	182	105	11	
07:53	10	101	167	235	262	255	278	255	212	131	18	
08:08	16	128	192	262	288	278	306	283	242	158	68	
08:23	60	155	216	289	313	300	332	310	272	183	92	10
08:38	82	181	239	314	337	321	357	336	300	207	114	15
08:53	100	206	262	338	360	340	381	362	327	231	134	55
09:08	118	229	283	361	381	359	404	385	353	253	154	73
09:23	135	250	302	383	401	376	425	408	377	273	171	89
09:38	150	270	320	403	420	392	445	428	399	292	188	103
09:53	164	288	337	421	437	407	462	447	419	309	203	117
10:08	176	305	351	437	452	420	478	464	438	324	216	129
10:23	187	319	364	451	465	431	492	478	454	338	228	140
10:38	197	332	376	463	476	441	505	491	468	350	238	149
10:53	204	342	385	474	486	449	515	502	480	359	246	157
11:08	211	350	393	482	494	456	523	511	489	367	253	164
11:23	215	357	398	488	500	461	529	517	496	373	258	170
11:38	219	361	402	493	504	464	533	522	501	377	261	174
11:53	220	363	404	495	505	466	535	524	503	379	263	176
12:08	220	363	404	495	505	466	535	524	503	379	263	178
12:23	219	361	402	493	504	464	533	522	501	377	261	178
12:38	215	357	398	488	500	461	529	517	496	373	258	176
12:53	211	350	393	482	494	456	523	511	489	367	253	174
13:08	204	342	385	474	486	449	515	502	480	359	246	170
13:23	197	332	376	463	476	441	505	491	468	350	238	164
13:38	187	319	364	451	465	431	492	478	454	338	228	157
13:53	176	305	351	437	452	420	478	464	438	324	216	149
14:08	164	288	337	421	437	407	462	447	419	309	203	140
14:23	150	270	320	403	420	392	445	428	399	292	188	129
14:38	135	250	302	383	401	376	425	408	377	273	171	117
14:53	118	229	283	361	381	359	404	385	353	253	154	103
15:08	100	206	262	338	360	340	381	362	327	231	134	89
15:23	82	181	239	314	337	321	357	336	300	207	114	73
15:38	60	155	216	289	313	300	332	310	272	183	92	55
15:53	40	128	192	262	288	278	306	283	242	158	68	38
16:08	10	101	167	235	262	255	278	255	212	131	45	10
16:23		72	141	207	235	232	250	226	182	105	11	
16:38		20	116	179	209	208	222	197	152	79		
16:53		12	91	152	182	184	194	169	123	51		
17:08			67	125	155	161	166	141	95	16		
17:23			44	99	130	137	138	114	69			
17:38			19	75	105	115	112	89	45			
17:53				54	79	92	88	62	27			
18:08				37	70	84	79	51	12			
18:23				27	60	75	70	41				
18:38				16	50	65	60	30				
18:53					40	56	50	18				
19:08					29	46	39					
19:23					18	35	29					
19:38						25	18					
19:53						14						

Tabelle A.9: Bestrahlungsstärke [W/m^2] in Würzburg unter 70° Inklination [29]

Tageszeit	Jan	Feb	Mrz	Apr	Mai	Jun	Jul	Aug	Sep	Okt	Nov	Dez
04:08						13						
04:23						22	16					
04:38					16	31	25					
04:53					25	40	34					
05:08					35	49	43	16				
05:23				14	44	57	52	26				
05:38				23	53	65	61	36				
05:53				33	61	73	69	45	10			
06:08				42	70	81	77	54	19			
06:23			16	65	77	88	84	75	41			
06:38			41	87	108	112	112	98	63			
06:53			63	111	132	134	137	123	88	14		
07:08		10	86	136	156	156	163	149	114	50		
07:23		18	110	162	181	178	190	175	142	77		
07:38		72	134	188	206	200	216	202	171	103	10	
07:53	9	100	159	215	231	222	243	229	200	128	16	
08:08	14	127	183	241	256	243	268	256	228	154	68	
08:23	61	154	206	266	279	264	293	282	256	179	92	9
08:38	82	179	228	290	302	284	317	307	283	202	114	13
08:53	101	203	250	313	324	302	340	331	309	225	134	55
09:08	118	226	270	335	344	320	362	353	334	246	153	73
09:23	135	247	289	355	363	337	382	374	357	266	171	89
09:38	150	266	306	374	381	352	400	394	378	284	187	104
09:53	163	284	322	392	397	365	417	412	398	301	201	117
10:08	176	300	336	407	411	378	432	428	416	316	214	129
10:23	186	314	348	421	424	389	446	442	431	329	226	140
10:38	195	326	359	432	435	398	457	454	445	340	236	149
10:53	203	336	368	442	444	406	467	465	456	350	244	157
11:08	209	344	376	450	451	412	475	473	465	357	250	164
11:23	214	350	381	456	457	417	481	479	472	363	255	169
11:38	217	354	385	460	461	420	484	483	476	367	259	173
11:53	219	356	386	462	462	422	486	485	479	369	260	176
12:08	219	356	386	462	462	422	486	485	479	369	260	177
12:23	217	354	385	460	461	420	484	483	476	367	259	177
12:38	214	350	381	456	457	417	481	479	472	363	255	176
12:53	209	344	376	450	451	412	475	473	465	357	250	173
13:08	203	336	368	442	444	406	467	465	456	350	244	169
13:23	195	326	359	432	435	398	457	454	445	340	236	164
13:38	186	314	348	421	424	389	446	442	431	329	226	157
13:53	176	300	336	407	411	378	432	428	416	316	214	149
14:08	163	284	322	392	397	365	417	412	398	301	201	140
14:23	150	266	306	374	381	352	400	394	378	284	187	129
14:38	135	247	289	355	363	337	382	374	357	266	171	117
14:53	118	226	270	335	344	320	362	353	334	246	153	104
15:08	101	203	250	313	324	302	340	331	309	225	134	89
15:23	82	179	228	290	302	284	317	307	283	202	114	73
15:38	61	154	206	266	279	264	293	282	256	179	92	55
15:53	40	127	183	241	256	243	268	256	228	154	68	38
16:08	9	100	159	215	231	222	243	229	200	128	45	9
16:23		72	134	188	206	200	216	202	171	103	10	
16:38		18	110	162	181	178	190	175	142	77		
16:53		10	86	136	156	156	163	149	114	50		
17:08			63	111	132	134	137	123	88	14		
17:23			41	87	108	112	112	98	63			
17:38			16	65	77	88	84	75	41			
17:53				42	70	81	77	54	24			
18:08				33	61	73	69	45	10			
18:23				23	53	65	61	36				
18:38				14	44	57	52	26				
18:53					35	49	43	16				
19:08					25	40	34					
19:23					16	31	25					
19:38						22	16					
19:53						13						

Tabelle A.10: Bestrahlungsstärke [W/m^2] in Würzburg unter 80° Inklination [29]

Tageszeit	Jan	Feb	Mrz	Apr	Mai	Jun	Jul	Aug	Sep	Okt	Nov	Dez
04:08						11						
04:23						19	13					
04:38					13	27	22					
04:53					22	34	30					
05:08					30	42	37	14				
05:23				12	38	49	45	22				
05:38				20	46	56	52	31				
05:53				28	53	63	59	39	9			
06:08				36	60	69	66	46	17			
06:23			14	43	66	75	72	54	37			
06:38			38	74	73	81	78	81	57			
06:53			58	96	106	86	107	103	79	12		
07:08		9	80	118	128	126	130	126	104	48		
07:23		15	102	142	151	146	154	151	129	74		
07:38		70	125	166	174	166	179	175	156	98	8	
07:53	8	97	147	190	196	185	202	200	183	123	13	
08:08	12	123	170	214	219	205	226	224	209	147	67	
08:23	60	148	191	237	240	224	249	248	235	170	91	7
08:38	80	173	212	260	261	242	271	271	260	193	111	11
08:53	98	196	232	281	281	259	292	293	285	214	131	55
09:08	116	217	251	301	300	275	311	314	307	234	149	72
09:23	132	237	269	320	317	290	330	333	329	253	166	87
09:38	146	255	285	338	333	304	347	351	349	270	182	101
09:53	159	272	299	354	348	317	363	368	367	285	195	114
10:08	171	287	313	368	361	328	377	382	383	299	208	126
10:23	181	300	324	381	373	338	389	395	398	312	219	136
10:38	190	312	334	391	383	347	400	407	410	322	228	145
10:53	197	321	343	401	392	354	408	416	421	331	236	153
11:08	203	329	350	408	399	360	416	424	429	339	242	159
11:23	207	335	355	414	404	364	421	430	436	344	247	164
11:38	210	339	358	417	407	367	425	434	440	347	250	168
11:53	212	341	360	419	409	368	427	436	442	349	251	171
12:08	212	341	360	419	409	368	427	436	442	349	251	172
12:23	210	339	358	417	407	367	425	434	440	347	250	172
12:38	207	335	355	414	404	364	421	430	436	344	247	171
12:53	203	329	350	408	399	360	416	424	429	339	242	168
13:08	197	321	343	401	392	354	408	416	421	331	236	164
13:23	190	312	334	391	383	347	400	407	410	322	228	159
13:38	181	300	324	381	373	338	389	395	398	312	219	153
13:53	171	287	313	368	361	328	377	382	383	299	208	145
14:08	159	272	299	354	348	317	363	368	367	285	195	136
14:23	146	255	285	338	333	304	347	351	349	270	182	126
14:38	132	237	269	320	317	290	330	333	329	253	166	114
14:53	116	217	251	301	300	275	311	314	307	234	149	101
15:08	98	196	232	281	281	259	292	293	285	214	131	87
15:23	80	173	212	260	261	242	271	271	260	193	111	72
15:38	60	148	191	237	240	224	249	248	235	170	91	55
15:53	40	123	170	214	219	205	226	224	209	147	67	38
16:08	8	97	147	190	196	185	202	200	183	123	44	7
16:23		70	125	166	174	166	179	175	156	98	8	
16:38		15	102	142	151	146	154	151	129	74		
16:53		9	80	118	128	126	130	126	104	48		
17:08			58	96	106	86	107	103	79	12		
17:23			38	74	73	81	78	81	57			
17:38			14	43	66	75	72	54	37			
17:53				36	60	69	66	46	21			
18:08				28	53	63	59	39	9			
18:23				20	46	56	52	31				
18:38				12	38	49	45	22				
18:53					30	42	37	14				
19:08					22	34	30					
19:23					13	27	22					
19:38						19	13					
19:53						11						

Tabelle A.11: Bestrahlungsstärke [W/m^2] in Würzburg unter 90° Inklination [29]

Tageszeit	Jan	Feb	Mrz	Apr	Mai	Jun	Jul	Aug	Sep	Okt	Nov	Dez
04:08						9						
04:23						16	11					
04:38					11	22	18					
04:53					18	29	25					
05:08					25	35	32	12				
05:23				10	32	42	38	19				
05:38				17	39	48	44	26				
05:53				24	45	53	50	33	7			
06:08				30	51	59	56	39	14			
06:23			12	36	56	64	61	45	32			
06:38			35	60	62	69	66	51	50			
06:53			53	79	66	73	71	82	70	10		
07:08		8	72	99	99	77	75	102	91	45		
07:23		13	92	120	119	81	117	123	114	69		
07:38		67	113	141	138	129	138	145	138	92	7	
07:53	6	92	133	162	158	146	159	166	162	114	11	
08:08	10	116	153	184	178	163	180	188	186	137	65	
08:23	57	140	173	204	197	180	200	209	209	158	87	6
08:38	77	162	192	224	215	195	219	229	232	179	106	10
08:53	94	183	210	243	233	211	238	249	253	198	125	52
09:08	110	203	227	261	249	225	255	267	274	217	142	69
09:23	125	222	243	278	265	238	271	284	293	234	158	84
09:38	139	239	257	293	279	250	286	301	311	249	172	97
09:53	151	254	271	308	292	262	300	315	327	263	185	109
10:08	162	268	283	320	304	272	313	328	342	276	196	120
10:23	171	280	293	332	315	281	324	340	355	287	206	130
10:38	180	290	302	342	323	288	333	350	366	297	215	138
10:53	186	299	310	350	331	295	341	359	376	305	222	145
11:08	192	306	316	356	337	300	347	365	383	312	228	151
11:23	196	312	320	361	342	304	352	371	389	316	232	156
11:38	199	315	324	365	345	306	355	374	393	320	235	159
11:53	200	317	325	366	346	308	357	376	395	321	236	162
12:08	200	317	325	366	346	308	357	376	395	321	236	163
12:23	199	315	324	365	345	306	355	374	393	320	235	163
12:38	196	312	320	361	342	304	352	371	389	316	232	162
12:53	192	306	316	356	337	300	347	365	383	312	228	159
13:08	186	299	310	350	331	295	341	359	376	305	222	156
13:23	180	290	302	342	323	288	333	350	366	297	215	151
13:38	171	280	293	332	315	281	324	340	355	287	206	145
13:53	162	268	283	320	304	272	313	328	342	276	196	138
14:08	151	254	271	308	292	262	300	315	327	263	185	130
14:23	139	239	257	293	279	250	286	301	311	249	172	120
14:38	125	222	243	278	265	238	271	284	293	234	158	109
14:53	110	203	227	261	249	225	255	267	274	217	142	97
15:08	94	183	210	243	233	211	238	249	253	198	125	84
15:23	77	162	192	224	215	195	219	229	232	179	106	69
15:38	57	140	173	204	197	180	200	209	209	158	87	52
15:53	38	116	153	184	178	163	180	188	186	137	65	36
16:08	6	92	133	162	158	146	159	166	162	114	42	6
16:23		67	113	141	138	129	138	145	138	92	7	
16:38		13	92	120	119	81	117	123	114	69		
16:53		8	72	99	99	77	75	102	91	45		
17:08			53	79	66	73	71	82	70	10		
17:23			35	60	62	69	66	51	50			
17:38			12	36	56	64	61	45	32			
17:53				30	51	59	56	39	14			
18:08				24	45	53	50	33	7			
18:23				17	39	48	44	26				
18:38				10	32	42	38	19				
18:53					25	35	32	12				
19:08					18	29	25					
19:23					11	22	18					
19:38						16	11					
19:53						9						

ANHANG

B Dallas 1-Wire-Schnittstelle

Das Dallas 1-Wire-Interface [20] ist ein serielles asynchrones Kommunikationsprotoll, mit dem zahlreiche, für Sensorknoten besonders interessante, integrierte Schaltungen wie etwa

- Datenspeicher,
- Schalter,
- Temperatur- und Feuchtigkeitssensoren,
- AD-Wandler
- Zeitgeber und
- Batteriemonitore

angesteuert werden können. Abbildung B.1 zeigt die 1-Wire-Topologie, bei welcher ein *Master* und nahezu beliebig viele der als *Slaves* bezeichneten Komponenten parallel an eine einzige Datenleitung angeschlossen werden.

Da die 1-Wire-Schnittstelle vom Mikrocontroller des SNoW^5-Sensorknotens nicht nativ unterstützt wird, für die Verwendung des DS2438 Batteriemonitors im hierarchischen Energiespeicher von SNoW^5-RA aber notwendig ist (vgl. Abbildung 3.4), beschreibt dieser Abschnitt eine auf dem MSP430 basierende Realisierung des 1-Wire-*Masters*.

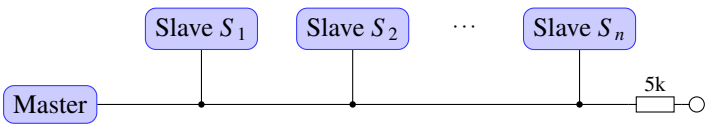


Abbildung B.1: 1-Wire-Topologie

Basis der Implementierung des *Masters* ist die Bitübertragungsschicht (`dow_link`), welche die Kodierung einzelner Bits auf der Datenleitung realisiert. Auch die Serialisierung zu übertragender Bytesequenzen und die Deserialisierung empfangener Bitsequenzen wird in dieser Protokollschicht umgesetzt.

Darauf aufbauend wird jede Datenübertragung zwischen *Master* und *Slave* durch folgende Sequenz realisiert

1. Zurücksetzen aller *Slaves* in einen Grundzustand
2. Auswahl eines der *Slaves* als Kommunikationspartner
3. Übertragung eines 8 bit-Kommandos vom *Master* zum *Slave*
4. kommandospezifische Datenübertragung

Eine solche Sequenz kann jeder Zeit durch das Zurücksetzen der *Slaves*, also dem Beginn einer neuen Sequenz, abgebrochen werden. Die Auswahl eines *Slaves* wird durch die Netzwerkschicht (`dow_network`) des Protokolls implementiert und basiert auf einer global eindeutigen 64 bit-Identifikationsnummer (*rom code*), die für jede 1-WIRE-Komponente werksseitig konfiguriert wird.

Die Anwendungsschicht, also die Menge der unterstützen Kommandos und der zugehörigen Datenübertragungsschemata, muss sich für die verschiedenen 1-WIRE-Komponenten unterscheiden, denn ein Datenspeicher bietet eine andere Funktionalität als ein Temperatursensor. Die höheren Ebenen des Protokollstacks müssen daher durch spezifische Treiber implementiert werden.

B.1 Bitübertragungsschicht

Solange weder der *Master* noch einer der *Slaves* den Bus aktiv auf *GND* ziehen, wird das Potential der Datenleitung über einen 5 kΩ *pullup*-Widerstand auf U_{cc} gehalten (vgl. Abbildung B.1). Dieser *idle*-Zustand des Datenbuses kann beliebig lange gehalten werden. Zur Übertragung eines Bits wird der Bus für eine bestimmte Zeit auf *GND* gezogen, wobei die Länge dieses Pulses den logischen Wert des Bits kodiert.

Abbildung B.2 zeigt das *Timing* der Bitkodierung aus Sicht des *Masters*, wobei das Potential der Datenleitung an den stark markierten Stellen durch den *Master*, an den mittelstark markierten Stellen durch einen *Slave* und an den schwach markierten Stellen durch den *pullup*-Widerstand bestimmt wird. Zum Versenden einer logischen 1 zieht der *Master* den Bus für höchstens 15 µs auf *GND*, während der Puls zum Versenden einer logischen 0 zwischen 60 und 120 µs lang sein muss. Bevor das nächste Bit übertragen werden kann, muss der Bus für wenigstens 1 µs im *idle*-Status gehalten werden [23].

Beim Lesen eines Bits werden zwar die gleichen Pulslängen verwendet, allerdings darf der *Slave* nicht den Zeitpunkt des Beginns eines Pulses, sondern nur dessen Länge bestimmen. Deshalb

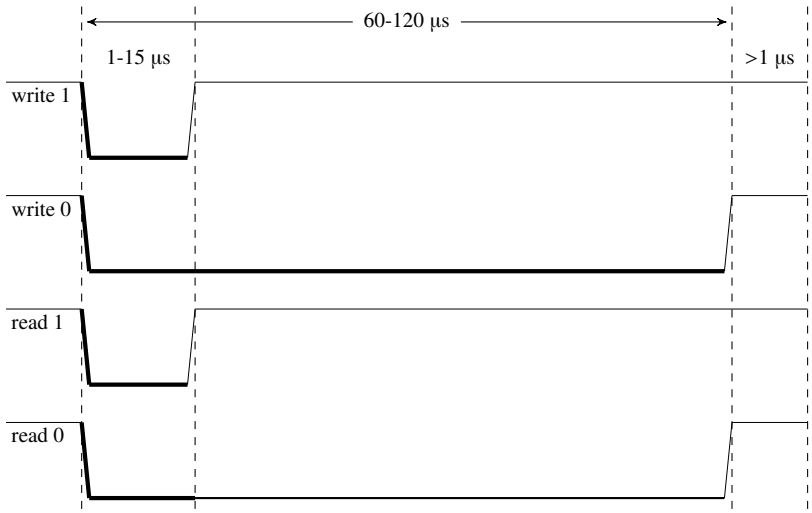


Abbildung B.2: Schreib- und Lesepulse

zieht der *Master* beim Lesen eines Bits den Bus wieder für höchstens 15 µs auf *GND* und gibt ihn danach wieder frei, wie dies bereits beim Schreiben einer 1 der Fall war. Die *Slaves* erkennen die fallende Flanke und wissen aufgrund des bisherigen Verlaufs der Kommunikationssequenz, ob sie als Kommunikationspartner selektiert sind und der *Master* ein Bit lesen möchte. Trifft dies beides zu und will der aktive *Slave* eine logische 0 an den *Master* senden, so hält er den Bus für den Rest des Pulses auf *GND*. Will er hingegen eine logische 1 senden, so lässt er den Bus durch den *pullup*-Widerstand wieder auf U_{cc} ziehen, nachdem der *Master* den Bus freigegeben hat. Für den *Master* entspricht der Lesevorgang also dem Schreiben einer 1 mit einer zusätzlichen Busabtastung kurz nach der Freigabe.

Unter bestimmten Umständen können auch mehrere Slaves gleichzeitig von der Netzwerkschicht aktiviert werden. Schreibt der *Master* dann Daten auf den Bus, so werden diese von allen aktiven *Slaves* gelesen und interpretiert. Liest der *Master* hingegen ein Bit vom Datenbus, so überlagern sich die Antworten aller aktiven *Slaves* zum *wired-AND*, denn eine einzelne 0 genügt, um den für den *Master* sichtbaren Puls zu verlängern.

Weil nur der *Master* die Zeitpunkte aller Pulse bestimmt, können Buskollisionen ausgeschlossen werden. Eine zusätzliche Arbitrierung ist daher nicht notwendig. Außerdem kommt die Leselogik ohne *Interrupts* und *Pollen* der Datenleitung aus.

Die oben beschriebene Kommunikationssequenz soll jederzeit durch Zurücksetzen aller *Slaves* abgebrochen werden können. Neben den Pulsen für eine logische 0 oder 1 wird daher noch ein *reset*-Puls benötigt, wie ihn Abbildung B.3 zeigt. Er muss mindestens 480 µs lang sein und wird von den *Slaves* mit einem *presence*-Puls beantwortet, an dem der *Master* erkennen kann, ob

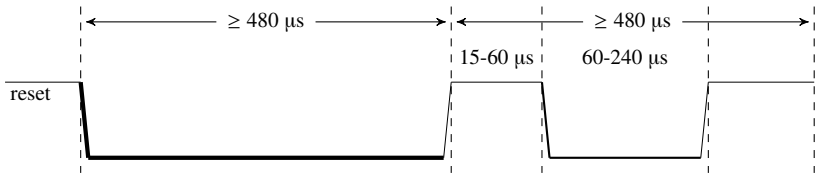


Abbildung B.3: *reset*- und *presence*-Puls

überhaupt 1-WIRE-Komponenten an den Datenbus angeschlossen sind.

Das in Abbildung B.2 und B.3 gezeigte Bustiming muss nun durch den MSP430 Mikrocontroller des SNoW[5]-Sensorknotens realisiert werden. Dabei wird das in [22] beschriebene und in Abbildung B.4 dargestellte Konzept zur Realisierung eines 1-WIRE-*Masters* mit Hilfe eines UART-Moduls umgesetzt, um die CPU nicht mit der Erzeugung der einzelnen Pulse zu belasten. Welcher der beiden UARTs des MSP430 dafür verwendet wird, kann über die Compilezeitkonstante `DOW_LINK_USART` konfiguriert werden.

Durch eine externe *Open-Collector*-Schaltung wird die 1-WIRE-Datenleitung (DOW) über den Transistor T2 auf *GND* gezogen, sobald der UART durch Versenden einer 0 am UxTX-Pin den Transistor T1 schließt. Beim Versenden einer 1 wird T1 hingegen geöffnet und zieht die Basis von T2 auf *GND*. Die Datenleitung wird dadurch freigegeben und kann von einem *Slave* auf *GND* oder vom *pullup*-Widerstand auf U_{cc} gezogen werden. Zum Abtasten der Datenleitung ist diese außerdem mit dem UxRX-Pin des UARTs verbunden, so dass eine TX-RX-Rückkopplung entsteht.

Der UART wird zur Verwendung von einem Startbit, acht Datenbits, einem Stoppbit und keinen Paritätsbits konfiguriert. Beim Versenden eines Bytes B^{tx} wird also zunächst das Startbit 0 übertragen und damit die Datenleitung auf *GND* gezogen. Durch die TX-RX-Rückkopplung wird somit auch ein Startbit am UxRX-Pin erkannt, so dass das UART-Modul mit jedem Schreibvorgang auch einen Lesevorgang startet. Anschließend wird B^{tx} mit dem am wenigsten signifikanten Bit zuerst übertragen und gleichzeitig B^{rx} empfangen, wobei

$$B_i^{tx} = 0 \Rightarrow B_i^{rx} = 0 \qquad \forall\, i \in \{0, \ldots, 7\} \tag{B.1}$$

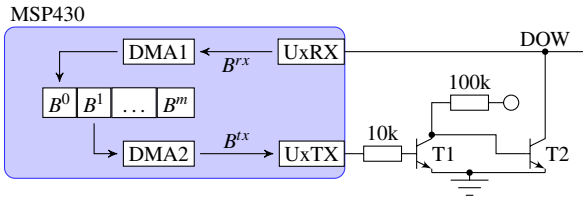


Abbildung B.4: Implementierung des 1-WIRE-Masters mittels UART und DMA-Controller

Tabelle B.1: Übersetzung zwischen 1-Wire-Bits und UART-Bytes

1-Wire Signal	Baudrate	B^{tx}	B^{rx}
write 1	115200	0xFF	0xFF
write 0	115200	0x00	0x00
read 1	115200	0xFF	0xFF
read 0	115200	0xFF	$\neq$ 0xFF
reset	9600	0xF0	0xF0
reset presence	9600	0xF0	$\neq$ 0xF0

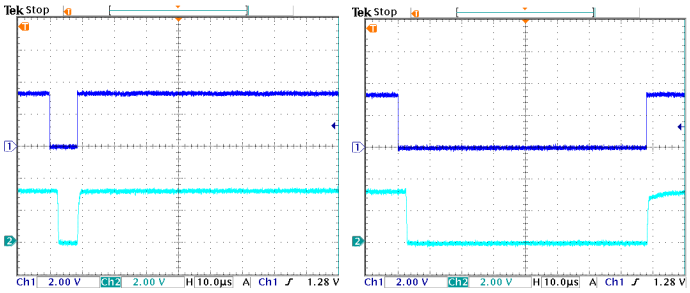
gelten muss. Wenn sich also das empfangene vom übertragenen Byte an einer Stelle i unterscheidet, dann muss $B_i^{tx} = 1$ und $B_i^{rx} = 0$ sein. In einem solchen Fall hat ein *Slave* den Datenbus auf *GND* gezogen, während er vom *Master* freigegeben war.

Zur Übertragung jedes 1-Wire-Bits muss nun ein UART-Byte nach Tabelle B.1 versendet werden. Bei einer Übertragungsbeschwindigkeit von 115200 baud $\approx$ 8,7 µs/bit erzeugt bereits das Startbit einen kurzen, zur Repräsentation der logischen 1 geeigneten Puls, so dass die eigentlichen Datenbits bei einem write 1 Signal allesamt auf 1 gesetzt werden müssen. Beim write 0 Signal wird der durch das Startbit begonnene Puls dagegen durch die acht Datenbits auf 78,3 µs verlängert, was zur Repräsentation der logischen 0 auf der Datenleitung ausreicht. Das abschließende UART-Stoppbit gibt die Datenleitung wieder frei und gewährt somit die in Abbildung B.2 geforderten Abschlussphase von über 1 µs.

Wie oben bereits beschrieben wird das Lesen eines 1-Wire-Bits durch den *Master* wie das Schreiben einer 1 realisiert, nur dass nun der Wert des gelesenen B^{rx} interpretiert werden muss. Sobald ein *Slave* die Datenleitung auf *GND* zieht, weicht das gelesene UART-Byte vom geschriebenen ab und kennzeichnet den Puls daher als read 0 Signal. Wird B^{rx} hingegen unverändert empfangen, so liegt ein read 1 Signal vor.

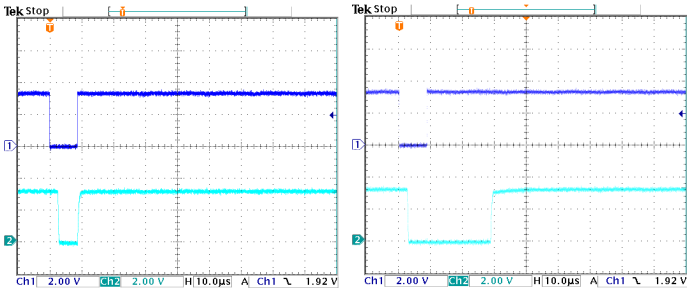
Für den wesentlich längeren *reset*-Puls muss die Baudrate des UARTs verringert werden. Bei 9600 baud $\approx$ 104 µs/bit entsprechen die vier niederwertigsten, auf 0 gesetzten Datenbits zusammen mit dem Startbit einer ausreichend großen Pulslänge von 521 µs. Die auf 1 gesetzten höherwertigen Datenbits von B^{tx} und das Stoppbit geben die Datenleitung für eine ebensolange Zeit frei, so dass ein *presence*-Puls registriert werden kann. Nur aus einer Abweichung zwischen B^{rx} und B^{tx} kann auf die Anwesenheit mindestens eines *Slaves* geschlossen werden.

Die Abbildungen B.5 bis B.7 zeigen die mittels eines Oszilloskops an den beiden UART-Pins beobachteten Signale bei der Erzeugung der einzelnen 1-Wire-Pulse. Als *Slave* diente dabei ein DS2438 Batteriemonitor.



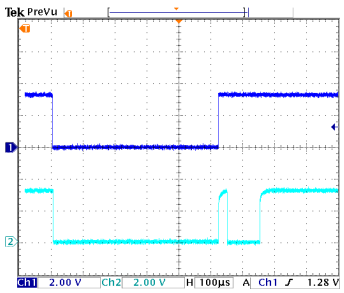
(a) write 1 **(b)** write 0

Abbildung B.5: 1-WIRE-Signalerzeugung mittels UART (Ch1=UxTX, Ch2=UxRx)



(a) read 1 **(b)** read 0

Abbildung B.6: 1-WIRE-Signalerzeugung mittels UART (Ch1=UxTX, Ch2=UxRx)



(a) reset presence

Abbildung B.7: 1-WIRE-Signalerzeugung mittels UART (Ch1=UxTX, Ch2=UxRx)

Beim Transport eines Bytes über die 1-WIRE-Schnittstelle wird dessen LSB zuerst übertragen. Dazu wird jedes Bit nach Tabelle B.1 in ein entsprechendes UART-Byte übersetzt und dieses dann in den Versendepuffer (UxTXBUF) des UARTs geschrieben. Nach dem Auslösen des *Interrupts* des UART-Eingangspuffers (UxRXBUF) wird das empfangene Byte wiederum nach Tabelle B.1 in-

terpretiert, falls es sich bei der Datenübertragung um einen Lesevorgang handelt. Die so empfangenen 1-WIRE-Bits werden wieder beim LSB beginnend zu einem 1-WIRE-Byte zusammengesetzt. Byteseqenzen werden auf die gleiche Weise seriell übertragen.

Das Übertragen eines 1-WIRE-Bits dauert 87 µs. Durch das interruptgesteuerte Warten auf den Abschluss der UART-Übertragung kann die CPU in dieser Zeit zwar für andere Aufgaben verwendet werden, allerdings wird für jeden Taskwechsel bereits etwa ein Viertel dieser Wartezeit benötigt. Während der Übertragung längerer Bytesequenzen wie etwa der 64 bit ID eines *Slaves*, kann die CPU für mehrere Millisekunden nicht mehr effektiv für andere Anwendungen verwendet werden.

Um diesem Problem zu begegnen, kann die Bitübertragungsschicht so konfiguriert werden, dass die serielle Übertragung von bis zu m Bits von zwei DMA-Controllern organisiert wird, wie in Abbildung B.4 gezeigt. Dabei wird ein Bytepuffer von der CPU zunächst mit den nach Tabelle B.1 notwendigen UART-Bytes initialisiert. Der DMA-Controller mit der niedrigeren Priorität (DMA2 in der Abbildung) ist dafür verantwortlich, jeweils ein UART-Byte in den Versendepuffer des UART zu schreiben, während der DMA-Controller mit der höheren Priorität (DMA1) die empfangenen Bytes zurück in den Bytepuffer kopiert. Beide Aktionen werden vom Datenempfang am UART getriggert, so dass die Einhaltung der korrekten Reihenfolge nur durch die DMA-Prioritäten gewährleistet wird. Beide Controller erhöhen automatisch ihre Quell- bzw. Zieladressen und stoppen nach der anfänglich konfigurierten Anzahl zu übertragender Bytes. Somit kann die CPU die Übertragung aller UART-Bytes auslösen, in dem sie das erste Byte in den Versendepuffer des UARTs schreibt. Bis zu dem beim Empfang des letzten Bytes vom DMA1 ausgelösten *Interrupt* steht die CPU daher vollständig für andere Anwendungen zur Verfügung. Abschließend muss der Bytepuffer nur noch nach Tabelle B.1 interpretiert und die empfangenen 1-WIRE-Bits wieder zu einer Bytesequenz zusammengesetzt werden.

B.2 Netzwerkschicht

Die Netzwerkschicht verwaltet für jeden angeschlossenen *Slave* einen Kontrollblock vom Typ `dow_network_slave_t`, der neben dem der Identifikationsnummer `rom` noch die beiden Flags `resumable` und `alerted` enthält. Das erste Flag ist dabei eine statische Eigenschaft, die vom spezifischen Treiber des *Slaves* gesetzt werden sollte, wenn dieser das Kommando `DOW_NETWORK_ROM_RESUME` unterstützt und damit der Slaveauswahlprozess vereinfacht werden kann. Das zweite Flag ist hingegen zur Laufzeit veränderlich und zeigt ein besonderes Ereignis am *Slave* an. Bei Temperatursensoren könnte dies bspw. das Überschreiten eines Schwellwerts sein.

Die Auswahl einzelner *Slaves* basiert auf deren 64 bit `rom` IDs. Damit der *Master* für verschiedene 1-WIRE-Anwendungen verwendet werden kann, ohne die IDs aller angeschlossenen *Slaves* von vorne herein kennen zu müssen, unterstützt das Kommunikationsprotokoll einen Suchalgorithmus [21], mit der die IDs zur Laufzeit bestimmt werden können. Neben der Standardsuche gibt es noch eine bedingte Suche, bei der nur die IDs der alarmierten *Slaves* gefunden werden.

Diese bedingte Suche wird von `dow_network_findAlerted()` verwendet, um die `alerted`-Flags der einzelnen *Slaves* zu setzten.

Bei der Selektion eines einzelnen *Slaves* mittels `dow_network_select(dow_network_slave_t*)` muss in der Regel das `DOW_NETWORK_ROM_MATCH` Kommando gefolgt von der ID des *Slaves* gesendet werden. In einigen Fällen kann der relative große Aufwand zur Übertragung der ID aber auch vermieden werden. Ist beispielsweise nur ein *Slave* an der Datenleitung angeschlossen, so genügt das `DOW_NETWORK_ROM_SKIP` Kommando für dessen Selektion. War der *Slave* andererseits bereits in der letzten Kommunikationssequenz aktiv und unterstützt er `DOW_NETWORK_ROM_RESUME`, so genügt die Übertragung dieses Kommandos zur Selektion des *Slaves*.

B.3 Verwendung der 1-WIRE Schnittstelle

Die 1-WIRE-Hardware wird durch die `DOW_RESOURCE` vor einem verschränkten Zugriff durch verschiedene Anwendungen geschützt. Mit dieser Ressource wird auch das UART-Modul der Bitübertragungsschicht initialisiert. Holt sich eine Anwendung die Ressource, so wird zum einen eine *Slave*-Suche durchgeführt, falls dies nicht bereits zuvor erfolgreich getan wurde. Außerdem werden alle *Slaves* in ihren Grundzustand zurückgesetzt und damit die erste Kommunikationssequenz eingeleitet. Ob dies erfolgreich war, also alle *Slaves* gefunden und zurückgesetzt wurden, kann mittels `DOW_RESOURCE.flags & fGet_ok` geprüft werden.

In einer Initialisierungsphase der Anwendung müssen dann die verschiedenen *Slaves* ihren spezifischen Treibern zugeordnet werden. Am einfachsten lässt sich dies durch die Überprüfung des niederwertigsten Bytes ihrer ID erreichen, welches für alle *Slaves* eines bestimmten Typs identisch ist (*family code*). Die meisten Treiberfunktionen müssen dann nur noch ein Kommando mit einer zugehörigen Datenübertragungssequenz kapseln und sollten nach dem Muster in Programmauszug B.1 realisiert werden.

```
getResource(&DOW_RESOURCE);
...
if (!dow_network_select(&slave)) //error handling
  dow_link_writeByte(command);
  /*kommandospezifische Datenübertragung mittels
      dow_link_readBit(),   dow_link_writeBit(),
      dow_link_readByte(),  dow_link_writeByte(),
      dow_link_readBytes(), dow_link_writeBytes() */
if (!dow_network_reset()) //error handling
...
releaseResource(&DOW_RESOURCE);
```

Programmauszug B.1: Realisierung einer 1-WIRE-Kommunikationssequenz

Abbildungsverzeichnis

Tabellenverzeichnis

Algorithmenverzeichnis

Programmauszugsverzeichnis

Abkürzungsverzeichnis

CC	constant current
CMOS	complementary metal oxide semiconductor
CPU	central processing unit
CV	constant voltage
DMA	direct memory access
DOW	dallas one wire
EEPROM	electrically erasable programmable read-only memory
EWMA	exponentially weighted moving average
FIFO	first in first out
FSM	finite state machine
GND	ground
GPIO	gereral purpose input/output
IC	integrated circuit
ICA	integrated current accumulator
ISR	interrupt service routine
JIT	just in time
LDO	low drop out
Li-Ion	Lithium-Ionen
Li-Po	Lithium-Polymer
LPM	low power mode
LSB	least significant bit
MCU	micro controller unit
MPP	maximum power point
MPPT	maximum power point tracker
NiCd	Nickel-Cadmium
NiMH	Nickel-Metallhydrid
o.B.d.A.	ohne Beschränkung der Allgemeinheit
PAP	Programmablaufplan
PFM	pulse frequency modulation
RX	receiver
SPDT	single pole double throw

TX ..... transmitter
UART ..... universal asynchronous receiver transmitter

Literaturverzeichnis

[1] *Sonnenstandsdiagramm für München*. http://de.wikipedia.org/wiki/Datei:Sonnenstandsdiagramm_Muenchen_300dpi.png. – aufgerufen am 31.05.2009

[2] Baunach, Marcel: SNoW^5 *Power Consumption Test Protocol*. July 2008

[3] Baunach, Marcel ; Kolla, Reiner ; Mühlberger, Clemens: SNoW^5: a modular platform for sophisticated real-time wireless sensor networking / Institut für Informatik. 2007 (399). – Forschungsbericht

[4] Bottner, H. ; Nurnus, J. ; Schubert, A. ; Volkert, F.: New high density micro structured thermogenerators for stand alone sensor systems. In: *Proc. 26th International Conference on Thermoelectrics ICT 2007*, 2007, S. 306–309. – ISSN 1094-2734

[5] Buchmann, Isidor: *Charging lithium-ion batteries*. http://www.batteryuniversity.com/partone-12.htm. – aufgerufen am 13.06.2009

[6] Burd, T.D. ; Brodersen, R.W.: Energy efficient CMOS microprocessor design. In: *Proc. Twenty-Eighth Hawaii International Conference on System Sciences* Bd. 1, 3–6 Jan. 1995, S. 288–297

[7] Cadence: *Cadence PSpice A/D and Advanced Analysis*. http://www.cadence.com/products/orcad/pspice_simulation/pages/default.aspx. – aufgerufen am 10.04.2009

[8] David Linden, Thomas R.: *Handbook of Batteries (3rd ed.)*. Kap. 22, 28, 29, 35, McGraw-Hill

[9] Denninger, Andreas: *Konzeption und Entwurf eines energieautarken Sensorknotens*, Julius-Maximilians-Universität Würzburg, Diplomarbeit, September 2007

[10] Dubois-Ferriere, H. ; Meier, R. ; Fabre, L. ; Metrailler, P.: TinyNode: a comprehensive platform for wireless sensor network applications. In: *Proc. Fifth International Conference on Information Processing in Sensor Networks IPSN 2006*, 2006, S. 358–365

[11] Dutta, P. ; Hui, J. ; Jeong, J. ; Kim, S. ; Sharp, C. ; Taneja, J. ; Tolle, G. ; Whitehouse, K. ; Culler, D.: Trio: enabling sustainable and scalable outdoor wireless sensor network deployments. In: *Proc. Fifth International Conference on Information Processing in Sensor Networks IPSN 2006*, 2006, S. 407–415

[12] Gold Peak Batteries: *NiMH Technical Handbook*. http://www.gpbatteries.com.hk/html/pdf/NiMH_technical.pdf. – aufgerufen am 07.06.2009

[13] Hsu, J. ; Zahedi, S. ; Kansal, A. ; Srivastava, M. ; Raghunathan, V.: Adaptive Duty Cycling for Energy Harvesting Systems. In: *Proc. International Symposium on ISLPED'06 Low Power Electronics and Design*, 4–6 Oct. 2006, S. 180–185

[14] IXYS: *IXOLAR High Efficiency Solar Cells*. http://ixdev.ixys.com/DataSheet/XOD17-Solar-Cell-Die-Datasheet_Mar-2008.pdf. – aufgerufen am 18.06.2009

[15] Jeong, Jaein ; Jiang, Xiaofan ; Culler, D.: Design and analysis of micro-solar power systems for Wireless Sensor Networks. In: *Proc. 5th International Conference on Networked Sensing Systems INSS 2008*, 2008, S. 181–188

[16] Jiang, X. ; Polastre, J. ; Culler, D.: Perpetual environmentally powered sensor networks. In: *Proc. Fourth International Symposium on Information Processing in Sensor Networks IPSN 2005*, 15 April 2005, S. 463–468

[17] Kansal, A. ; Hsu, J. ; Srivastava, M. ; Raqhunathan, V.: Harvesting aware power management for sensor networks. In: *Proc. 43rd ACM/IEEE Design Automation Conference*, 2006, S. 651–656

[18] Knuth, Donald E.: *Art of Computer Programming, Volume 2: Seminumerical Algorithms (3rd ed.)*. Kap. 4.5.2, Addison-Wesley Professional, November 1997

[19] Kolla, Reiner ; Baunach, Marcel ; Mühlberger, Clemens: SNoW^5: a versatile ultra low power modular node for wireless ad hoc sensor networking. In: Marrón, Pedro J. (Hrsg.): *5. GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze"* Institut für Parallele und Verteilte Systeme (Veranst.), July 2006, S. 55–59

[20] MAXIM: *1-Wire and iButton*. http://www.maxim-ic.com/auto_info.cfm. – aufgerufen am 01.07.2009

[21] MAXIM: *APPLICATION NOTE 187: 1-Wire Search Algorithm*. http://pdfserv.maxim-ic.com/en/an/AN187.pdf. – aufgerufen am 02.07.2009

[22] MAXIM: *APPLICATION NOTE 214: Using a UART to Implement a 1-Wire Bus Master*. http://pdfserv.maxim-ic.com/en/an/AN214.pdf. – aufgerufen am 01.07.2009

[23] MAXIM: *DS2438 Smart Battery Monitor*. http://datasheets.maxim-ic.com/en/ds/DS2438.pdf. – aufgerufen am 07.06.2009

[24] MAXIM: *MAX1674/MAX1675/MAX1676 High-Efficiency, Low-Supply-Current, Compact, Step-Up DC-DC Converters*. http://datasheets.maxim-ic.com/en/ds/MAX1674-MAX1676.pdf. – aufgerufen am 07.06.2009

[25] MAXIM: *MAX965-MAX970 Single/Dual/Quad, Micropower, Ultra-Low-Voltage, Rail-to-Rail I/O Comparators*. http://datasheets.maxim-ic.com/en/ds/MAX965-MAX970.pdf. – aufgerufen am 18.06.2009

[26] Maxwell Technologies: *BC Power Series BOOSTCAP Ultracapacitors*. http://www.maxwell.com/pdf/uc/datasheets/20090227_DATASHEET_BC_Series_1009643.5.pdf. – aufgerufen am 07.06.2009

[27] Micropelt: *Thin film thermoelectrics*. http://www.micropelt.com. – aufgerufen am 10.04.2009

[28] Photovoltaic Austria Federal Association: *PV-Auslegung*. http://www.pvaustria.at/content/page.asp?id=62. – aufgerufen am 16.06.2009

[29] Photovoltaic Geographical Information System: *Geographical Assessment of Solar Resource and Performance of Photovoltaic Technology*. http://re.jrc.ec.europa.eu/pvgis. – aufgerufen am 10.04.2009

[30] Raghunathan, V. ; Schurgers, C. ; Park, Sung ; Srivastava, M.B.: Energy-aware wireless microsensor networks. 19 (2002), March, Nr. 2, S. 40–50

[31] Raghunathan, Vijay ; Kansal, A. ; Hsu, J. ; Friedman, J. ; Srivastava, Mani: Design considerations for solar energy harvesting wireless embedded systems. In: *Proc. Fourth International Symposium on Information Processing in Sensor Networks IPSN 2005*, 15 April 2005, S. 457–462

[32] Reda, Ibrahim ; Andreas, Afshin: Solar Position Algorithm for Solar Radiation Applications. In: *Solar Energy* 76 (2003), S. 577–589

[33] Rindelhardt, Udo: *Photovoltaische Stromversorgung*. Kap. 3.2, S. 81–103, Teubner Verlag, November 2001

[34] Roundy, Shad ; Wright, Paul K. ; Rabaey, Jan: A study of low level vibrations as a power source for wireless sensor nodes. In: *Computer Communications* 26 (2003), July, Nr. 11, S. 1131–1144. – URL http://dx.doi.org/10.1016/S0140-3664(02)00248-7

[35] Samsung: *L18650 Li-Ion Akku*. http://www2.produktinfo.conrad.com/datenblaetter/250000-274999/251024-da-01-en-LI-ION_Akku_Samsung_L18650.pdf. – aufgerufen am 12.06.2009

[36] schott: *solar*. http://www.schott.com/photovoltaic/german/. – aufgerufen am 10.04.2009

[37] Simjee, F. ; Chou, P.H.: Everlast: Long-life, Supercapacitor-operated Wireless Sensor Node. In: *Proc. International Symposium on ISLPED'06 Low Power Electronics and Design*, 4–6 Oct. 2006, S. 197–202

[38] Stein, J.: Computational Problems Associated with Racah Algebra. In: *Journal of Computational Physics* 1 (1967), Februar, S. 397–+

[39] Strang, Gilbert: *Linear algebra and its applications (3rd ed.)*. Kap. 1.7, Harcourt Brace Jovanovich College Publishers, 1988

[40] sunways: *Photovoltaic Technology*. http://www.sunways.de. – aufgerufen am 10.04.2009

[41] Šúri, Marcel ; Hulda, Thomas A. ; Dunlopa, Ewan D. ; Ossenbrink, Heinz A.: Potential of solar electricity generation in the European Union member states and candidate countries. In: *Solar Energy* 81 (2007), S. 1295–1305

[42] Taneja, J. ; Jeong, Jaein ; Culler, D.: Design, Modeling, and Capacity Planning for Micro-solar Power Sensor Networks. In: *Proc. International Conference on Information Processing in Sensor Networks IPSN '08*, 22–24 April 2008, S. 407–418

[43] Texas Instruments: *TPS61200 / TPS61201 / TPS61202 low input voltage synchronous boost converter with 1.3A-switches*. `http://focus.ti.com/lit/ds/symlink/tps61201.pdf`. – aufgerufen am 11.06.2009

[44] Texas Instruments: *MSP430x1xx Family User's Guide*. Dallas (USA): , 2006

[45] Watkins, David S.: *Fundamentals of matrix computations (2nd ed.)*. Kap. 1.7. New York, NY, USA : John Wiley & Sons, Inc., 2002

[46] Weber, R.: Photovoltaik und Solarthermie - Energielieferanten der Zukunft. In: *FACH.JOURNAL* (2008), S. 36–44

[47] Zhang, Pei ; Sadler, Christopher M. ; Lyon, Stephen A. ; Martonosi, Margaret: Hardware design experiences in ZebraNet. In: *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA : ACM, 2004, S. 227–238. – ISBN 1-58113-879-2

Erklärung

Hiermit versichere ich, dass ich diese Diplomarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Stellen meiner Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, habe ich in jedem Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht. Dasselbe gilt sinngemäß für Tabellen, Karten und Abbildungen. Diese Arbeit hat in dieser oder einer ähnlichen Form noch nicht im Rahmen einer anderen Prüfung vorgelegen.

Würzburg, den 9. Juli 2009 Andreas Engel