Heinrich C. Mayr, Martin Pinzger (Hrsg.): INFORMATIK 2016,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2016 1917

# IoT und Sensornetzwerke: Entwurf und Programmierung von Niedrigstenergiesystemen anhand einer Metaarchitektur

Volker Skwarek[1], Thorsten Kistler[2], Mark Rawer[2] und Stefan Schauer[3]

**Abstract:** Ein wesentlicher Erfolgsfaktor von Sensornetzwerken wird in deren Autonomie und damit auch dem Energiebedarf liegen. Der Energiebedarf wird aber zunächst wenig vorhersagbar in allen Schichten des Hard- und Softwaredesigns beeinflusst. Daher muss der Energiebedarf für eine Niedrigenergieapplikation das primäre Entwurfsmerkmal sein. In diesem Beitrag wird ein Architekturkonzept vorgestellt, mit dem Energiebedarfe systematisch geplant werden können.

**Keywords:** Niedrigstenergiesysteme, Sensornetze, Architektur, Entwurfsrichtlinien

## 1 Einleitung

Die aktuellen Trends der x-4.0-Entwicklungen wie Industrie 4.0, Internet 4.0, o. Ä. setzen einen höheren Informationsgrad durch eine intensivere, vernetztere Auswertung von Informationen um. Diese Informationen werden dann beispielsweise über das Internet-of-Things (IoT) [MF10, S. 1] von *smart objects* eingesammelt werden, die aufgrund ihrer Größe, Beschaffenheit, Vielzahl, Redundanz aber auch aufgrund ihres individuellen Wertes als *smart dust* bezeichnet werden [DP10], [KKP99, S. 8]. Als größte Herausforderung zur Funktion solcher Sensoren und damit auch von deren Akzeptanz wird der Energiebedarf der Sensorknoten genannt:

> ”A major challenge is to incorporate all these functions while maintaining very low power consumption, thereby maximizing operating life given the limited volume available for energy storage. the dust mote power consumption cannot exceed roughly 10 microwatts. The functionality envisioned for Smart Dust can be achieved only if the total power consumption of a dust mote is limited to microwatt levels, and if careful power management strategies are utilized.“[KKP99, S. 271f.].

Schon in dieser Kernpublikation der drahtlosen Sensornetzwerke Ende der 1990er Jahre besteht also die elementare Anforderung eines sehr restriktiven Energiemanagements. Ob diese Arbeit Ursprung oder Teil eines neuen Trends war, lässt sich nicht eindeutig

[1] Hochschule für Angewandte Wissenschaften Hamburg, Ulmenliet 20, D-21033 Hamburg, volker.skwarek@haw-hamburg.de
[2] Cypress Semiconductor GmbH, Willy-Brandt-Allee 4, D-81829 München, thorsten.kistler@cypress.com, mark.rawer@cypress.com
[3] Texas Instruments Deutschland GmbH, Haggertystr. 1, D-85356 Freising, s-schauer1@ti.com

feststellen. Allerdings ist insbesondere zwischen 2000 und 2010 eine hohe Anzahl von Publikationen zum Energiemanagement und zur Energiebedarfsprädiktion festzustellen. Exemplarisch können folgende Bereiche benannt werden: Chipdesign, Energiespeicherung und -erzeugung, Befehlssätze, Architektur und Programmzyklensimulation oder Kommunikationsprotokolle.

Allen diesen Aspekten ist es gemeinsam, dass der jeweilige Schwerpunkt auf einzelne Systemschwerpunkte gelegt wurde, aber nicht auf das gesamte System. Hieraus entsteht dann das Problem, dass eine Energiebedarfsoptimierung immer nur ein Angebot zur Nutzung ist, nie aber verpflichtend, weshalb sie einfach umgangen werden kann. Folglich muss die Nutzung von energieminimierenden Eigenschaften durch das gesamte System hindurch geplant werden.

Dieser Planungsaspekt steht in dieser Publikation im Vordergrund: Anhand eines sehr einfachen Sensorknotens werden examplarisch hohe Energiebedarfe dargestellt (Kapitel 2.1) und mit Hilfe eines aus vielen Beispielen empirisch entwickelten Energieschichtenmodells unter dem Aspekt der Energiesystemplanung diskutiert (Kapitel 2.2). Im Kapitel 3 wird dann dieser Sensorknoten mit Hilfe des Schichtenmodells in Teilen neu geplant. In diesem Beitrag ist zu berücksichtigen, dass vollständige Anleitung und Umsetzung innerhalb des Workshops erfolgt und hier nur wenige Fokuspunkte betrachtet werden können

## 2 Schichtenmodell zur Energiebedarfskonzeption

Auf der Modellierungsebene sind bisher Energiemodelle bekannt, die konkrete komplexe Applikationen wie beispielsweise Prozessabläufe in Fabriken auf Systemebene abbilden [Fel+13] oder [SSS13]. Im Gegensatz dazu wird hier ein generisches Modell vorgestellt, das sich an Softwarearchitekturen bzw. dem Softwareentwicklungsprozess orientiert.

### 2.1 Energiebedarfsermittlung am Beispiel eines Lichtsensors

Zunächst soll als Beispiel eine Energiebedarfsanalyse für eine einfache Applikation herangezogen werden: ein autonomer Lichtsensor. Diese Anwendung ist aufgrund ihrer einfachen Architektur gewählt, da diese keine grundsätzliche Herausforderung an deren Entwicklung und Umsetzung darstellt. Die Architektur dieses Systems ist in der folgenden Abbildung 1 dargestellt. Der Programmablauf umfasst das Messen der Umgebungshelligkeit mit einem Photowiderstand 5 Mal pro Sekunde und das Senden des Messwertes per Bluetooth Low Energy (BLE) an ein zentrales Lichtsteuergerät. Der Betriebszustand wurde mit einer grünen Low-Power-LED angezeigt. Es wurde mit dem

*Energy Monitor* von EEMBC[4] der mittlere Strombedarf einzelner Komponenten oder Maßnahmen über 10 Sekunden ermittelt, indem diese sukzessive modifiziert wurden

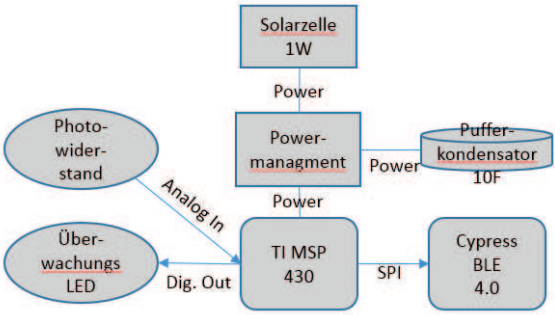


Abb. 1: Architektur eines Beispielsensors..

In Tabelle 1 wird deutlich, wie exemplarische Designmaßnahmen in Code, Hardware und Algorithmus jeweils zu signifikanten Verbesserungen im Energiebedarf des Systems geführt haben. Nach allen - teilweise in der Tabelle auch nicht aufgeführten – Maßnahmen konnte die über 10 Sekunden gemittelte Stromaufnahme von ca. 3mA auf 30µA zuzüglich des BLE-Kommunikationsmoduls gesenkt werden.

| **Maßnahme** | **Stromreduktion in µA** |
|---|---|
| Low Power LED | 1490 |
| Eingangskonfiguration als Input mit Pullup-Widerstand | 2070 |
| Aktivieren des Photowiderstandes über GPIO erst zum Messzeitpunkt | 40 |
| Aktivieren des Low-Power-Mode 3 von 4 mit Interrupt-steuerung | 227 |

Tab. 1: Mittlerer Strombedarf einzelner Komponenten und Strukturen des Sensors.

## 2.2 Empirisches Architekturmodell zur Energiebedarfsplanung

Die Energiearchitektur in Abbildung 2 wurde als Strukturmodell zur Energieplanung entworfen. Als Vorbild wurde die ISO/OSI-Kommunikationsarchitektur verwendet [ISO94, Abb. 11].

Die grobe Unterteilung sieht zunächst eine Separation des Problems in software- und hardwarespezifische Anteile vor, was dem oberen und unteren Ende des Modells in Abbildung 2 entspricht. Als Beispiel könnte ein fiktiver Signalvergleich per Korrelation herangezogen werden: Dieser erfolgt in der Regel über analoge Eingangspegel. Bei stark verrauschten Signalen würde aber eine Polaritätskorrelation - also eine binäre

[4] http://www.eembc.org/ulpbench/about.php

Korrelation der Vorzeichen in Bezug auf einen Mittelwert - zu vergleichbaren Ergebnissen führen [Pue15, Kapitel 6.3.4]. Selbst wenn an dieser Stelle noch immer eine softwarebasierte Korrelation erfolgt, was nicht zwingend erforderlich wäre, könnte der Energiebedarf dafür schon durch folgende Maßnahmen deutlich gesenkt werden:

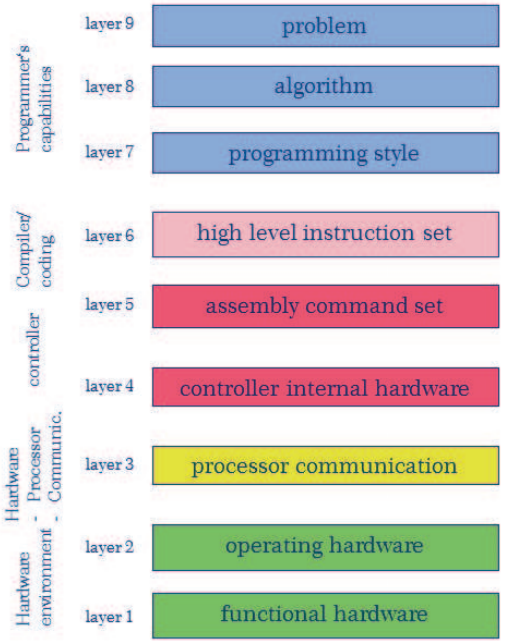


Abb. 2: Modell zu energieoptimierendem Systemdesign

- hardwareseitige Vorverarbeitung der Analogsignale in amplitudendiskrete Digitalsignale durch einen Schwellwertkomparator.
- Verzicht auf Analog-Digital-Wandler,
- Reduktion der prozessorinternen Berechnungsauflösung auf Ganz- oder sogar Binärzahlen sowie Verzicht auf Fließkommaoperationen.

Kerngedanke dieser Architektur ist also die genaue Analyse des erforderlichen Algorithmus hinsichtlich der Zielanwendung und -genauigkeit. Im Detail übernehmen die einzelnen Schichten folgende Funktionen, beginnend mit der unteren Schicht der funktionalen Hardware:

**Layer 1: functional hardware** Im Rahmen der algorithmischen Separation auf Soft- und Hardware wird überlegt, welche Bestandteile des Systems besser hardwareseitig realisiert werden können. Dies sind idealerweise solche, die die Auflösung des Signal zur weiteren Verarbeitung im Prozessor herabsetzen und

damit Rechenaufwand reduzieren, oder solche, die algorithmisches Anteile auf einfache physikalische mathematischen Gesetze zurückführen wie Multiplikationen durch Mischer oder Integrationen/Differentiationen auf rückgekoppelte Operationsverstärker. Hier ist allerdings unbedingt zu beachten, dass auch negative Rahmenbedingungen wie sie oft durch Temperaturabhängigkeiten von Hardwarekomponenten auftreten, mit betrachtet werden.

**Layer 2: operational hardware** beschreibt sämtliche Komponenten zum Betrieb eines Mikrocontrollers sowie dessen geeignete Auswahl selbst. Hier steht die Bewertung im Vordergrund, inwieweit der Controller zur Anforderung passt und welche Peripherie wie beschaltet werden muss. Klassische Auslegungsaspekte wie Speicherbedarf und Taktfrequenz treten hier eher in den Hintergrund. Wichtiger sind Aspekte, die das Sleep- und Wakeup-Timing sowie Umfang und Funktion von Niedrigenergiemodi betreffen. Da das Heruntertakten einen Kernaspekt der Niedrigenergieprogrammierung darstellt, sollten folgende Fragen beantwortet werden: welche Controllerkomponenten erfordern welche Timer zum Betrieb und wie weit lassen diese sich heruntertakten? Welche Komponenten funktionieren noch in welchem Niedrigenergiemodus und wie lang sind deren Wake-up-Zeiten? Ist gegebenenfalls ein kaskadiertes Aufwachkonzept erforderlich, das wiederum zu weiteren Latenzzeiten führt? Hieraus ergibt sich dann die Controllerauswahl sowie dier betriebserforderlichen Peripheriekomponenten und Beschaltungen.

**Layer 3: processor configuration** Unter der Vielzahl der drahtgebunden und drahtlosen Möglichkeiten, Daten aufzunehmen und nach außen zu kommunizieren, muss das der Messaufgabe angemessene energiesparendste Verfahren ausgewählt werden. Es sind dabei nicht nur die vordergründigen Aspekte wie Datenrate und Reichweite zu berücksichtigen, sondern auch das Kommunikationsbasisprotokoll, das die Nettodatenrate und den Overhead definiert, Handshake- und Latenzzeiten nach dem Wake-up aber auch die Konfigurierbarkeit von Protokollen oder Diensten.

**Layer 4: controller internal hardware** Aus der Selektion des Controllers sowie der Erfassungs- und Verarbeitungsaufgabe ergeben sich meist unterschiedliche Möglichkeiten, diese innerhalb des Controllers umzusetzen. Oft werden unreflektiert Basiskonfigurationen übernommen. Zu ungenutzten Anpassungsmöglichkeiten gehören unter anderem die Herabsetzung der Auflösung von AD-Wandlern auf den minimalerforderlichen Wert aber auch die Konfiguration vonWatch-Dogs, Capture/Compare-Strukturen oder das Setzen von ungenutzten Kommunikationsports auf hochohmige Eingänge. Kurz gefasst ist hier das intensive Auseinandersetzen mit der Beschreibung des Controllers erforderlich.

**Layer 5: assembly command set** Die (genutzte) Befehlsstruktur des Controllers definiert sehr genau, wie welche Komponenten der central processing unit (CPU) bzw. des Controllers verwendet werden. ÜBLICHERWEISEblicherweise kann dies sehr gut durch Compiler-Einstellungen sowie den Programmierstil definiert werden. Mangels einer Compilereinstellung ”energiesparend“ ist es hier aber

erforderlich, das Ergebnis der Codegenerierung kritisch zu analysieren und gegebenenfalls in Assembler nachzuoptimieren.

**Layer 6: high level instruction set** Heutzutage ist die Programmierung eines Mikrocontrollers in Assembler eher selten, sondern es wird auf C zurückgegriffen. Grundsätzlich ist hier aber weiterhin eine systemnahe Programmierung erforderlich, die beispielsweise durch Pointer und Bitmanipulation realisiert werden kann. In modernen Prozessorarchitekturen wird daher eine Abstraktionsschicht angeboten, die diese Bitmanipulation durch Klartextbefehle ersetzt. Hier können aber auch Funktionen zusammengefasst worden sein, die zwar den allgemeinen Umgang mit dem Controller erleichtern, aber nicht im Sinne der Energieeffizienz optimieren. Daher ist es sinnvoll, bei Auffälligkeiten im Energiebedarf diese Routinen zu überprüfen und gegebenenfalls doch die bitbasierte systemnahe Programmierung umzusetzen.

**Layer 7: programming style** Der Aspekt des Programmierstils ist von dem der Algorithmik zu trennen. Hierzu gehören beispielsweise Aspekte wie Objektorientierung, Verwendung von auf- oder abzählenden Schleifen oder umfangreiche mathematische Operationen. Bei allen diesen Aspekten gilt in erster Näherung als Grundregel: alles, was die Lauf- und Rechenzeit reduziert, optimiert sehr wahrscheinlich auch aufgrund der geringeren Zyklenzahl den Energiebedarf. Natürlich muss auch hier berücksichtigt werden, dass bestimmte Aufgaben von spezialisierten Strukturen auf einem Controller - z. B. digitale Signalprozessoren (DSPs) übernommen werden können, deren Energieeffizienz aber in Frage gestellt werden darf.

**Layer 8: algorithm** Ein Algorithmus als Kernelement einer rechnerischen Problemlösung entscheidet üblicherweise deutlich über den Rechenaufwand und somit auch über den Energiebedarf. Als Beispiel sei nur die rekursive Programmierung erwähnt: Diese gilt oft als Königsdisziplin der Algorithmik, da ein Problem so beschrieben wird, dass es sich selbst lösen kann. Hieraus entsteht oft ein sehr kurzer, kompakter Code. Der permanente Selbstaufruf hat allerdings im Rechnerkern zur Folge, dass leistungsoptimierende Strukturen wie Pipelines noch unzureichend genutzt werden können. Heap- und Stackspeicher als schnelle Speicher zur Aufnahme von Systemzust änden bei Sprüngen können überlaufen und in langsamere RAM-Speicher verschoben werden. Folglich sind kurze, knappe und einfache Algorithmen in der Regel solchen vorzuziehen, die eine rechnerisch-mathematische Eleganz umzusetzen versuchen.

**Layer 9: problem** Der Aspekt der Problemformulierung ist elementar für die Systementscheidung. Es existieren beispielsweise derzeit keine praktischen Ansätze, um ein diagnostisches Dauer-Elektrokardiogramm über 24 Stunden mit drahtlosen Sensorsystemen übertragen zu können, ohne auf externe Energiezufuhr zurückgreifen zu müssen (Stichwort: *drahtlose EKG-Elektrode*). Hier sind ca. 4000 Samples pro Sekunde im Millivolt-Bereich mit hoher Auflösung erforderlich. Eine

Randbedingung, die der Leistungsfähigkeit heutiger batteriebetriebener Sensorsysteme widerspricht. Andere Anwendungen wären beispielsweise eine wartungsfreie Überwachung von Umweltparametern mit garantierter Datenverfügbarkeit über einen langen Zeitraum. Verfügbarkeitsgarantien widersprechen heute noch dem Ansatz von Niedrigenergiesystemen und Energy Harvesting, da genau im deren Gegensatz die Philosophie des Designs besteht: Im Zweifel das System schlafen legen, bei Datenaufkommen – gegebenenfalls etwas zu spät - aufwecken und im Zweifel bei Energiemangel solange warten, bis die Speicher wieder aufgeladen sind.

Anhand der vorhergehenden Überlegungen wird deutlich, dass Energiemanagement ein primäres Entwurfsziel sein muss, das anhand des zuvor vorgestellten Vorgehens planerisch in alle Stufen der Architektur und Phasen der Umsetzung integriert werden muss. Dabei kann in jeder Planungsstufe erneut die Entscheidung zwischen Algorithmus und Energieeffizienz erforderlich werden, die zulasten einer der beiden Seiten ausfällt. Im Zweifel kann unter bestimmten Voraussetzungen kein autonomer Sensorknoten entworfen werden oder der Algorithmus/das Anwendungsgebiet muss unter Abstrichen realisiert werden.

## 3 Architekturbasierte Modellierung eines Sensorknotens

Im folgenden soll kurz dargestellt werden, wie der in Kapitel 2.1 exemplarisch eingeführte Lichtsensor unter Berücksichtigung der Architektur implementiert werden könnte. Dabei wird zunächst die Verarbeitungseinheit von der Kommunikationseinheit getrennt betrachtet, um die Gesamtkomplexität zu reduzieren. Als zentrale Fragestellung wird zunächst *Layer 9 (problem)* betrachtet, ob die Anwendung eines autonomen Lichtsensors mit gegebenenfalls sogar sicherheitsrelevanten Schaltaufgaben einer Niedrigstenergieanwendung widerspricht. Dabei ist zu beantworten, ob für die Anwendung in ausreichend schneller Datenrate Helligkeitsinformation zur Verfügung gestellt werden können und ein Ausbleiben aufgrund durch Energiemangel in Maßen toleriert werden kann. Aufgrund einer erforderlichen Reaktionszeit von 500ms reichen Messungen im Intervall von 200ms, um selbst bei Verlust eines Datenwortes und ungünstiger Intervalllage zumindest einen Datensatz zu übertragen.Weiterhin kann bei Ausbleiben von Daten auf der Empfängerseite ein Fail-Safe-Modus mit dem Standardwert ”Licht einschalten“ implementiert werden. Auch die Kommunikationsreichweiten zwischen Sender und Empfänger sind mit wenigen Metern für übliche drahtlose Übertragungsverfahren problemlos umsetzbar. Daraus folgt, dass das Problem grundsätzlich für eine Niedrigenergieanwendung geeignet ist.

### 3.1 Sensorsystem

Die *funktionale Hardware (Layer 1)* kann so gestaltet werden, dass der relevante Helligkeitsbereich zwischen 1000lx (Licht an) und 6000lx (Licht aus) beispielsweise

durch Photowiderstände abgebildet werden kann, die jeweils zum Messzeitpunkt über einen digitalen Eingang aktiviert werden.

Die *operative und prozessorinterne Hardware (Layer 2 und 4)* benötigt daher nur einen AD-Wandler mit einer vergleichsweise geringen Auflösung - z. B. 8 Bit - und Präzision. Diese Aspekte sind deshalb eher unkritisch, da die Präzision wesentlich durch die Qualität und Auslegung der Peripherieelemente beeinflusst wird. Funktionale Randbeschaltungenfür den Prozessor sind nicht zu erwarten. Als Taktgeber kann eine interne Uhr mit geringer Präzision und Taktrate (wenige kHz) verwendet werden, da die Echtzeitansprüche der Anwendung niedrig sind und durch einen Fail-Safe-Modus abgefangen werden.

Aufgrund der Einfachheit der Anwendung sind auf den *programmierrelevanten Ebenen (Layer 5 bis 8)* keine besonderen Herausforderungen zu erwarten: die AD-Wandlerwerte können als ganzzahlige *counts* direkt über Integeroperationen auf einen Schaltwert umgerechnet werden. Der Schaltwert für dann zu einer binären Ausgabe des Schaltzustandes auf dem drahtlosen Übertragungskanal.

Als *Programmierstil (Layer 7)* sollte eine interruptbasierte Programmierung vorgesehen werden, um das System zwischen den Messungen in den tiefstmöglichen Sleepmodus zu versetzen.

### 3.2 Kommunikationssystem

Für die drahtlose Kommunikation wird hier BLE vorgegeben. Mit dem in dieser Applikation separaten Sendesystem wird auf Layer 3 über ein einfaches Protokoll mit langsamer, overhead-armer Kommunikation gesendet. Um eine Manipulation der Daten und somit ein fälschliches Ausschalten des Lichtes zu verhindern, sollte kein Broadcast-, sondern ein Peer-to-Peer-Protokoll mit Verschlüsselung eingesetzt werden. Hierdurch entstehen zwar negative Implikationen auf die Energiebilanz erforderliche Sicherheit kann aber hergestellt werden [Tow+14, Kapitel 3]. Oberhalb der Verschlüsselungsebene wird ein sehr einfaches Protokoll verwendet, in dem entweder nur der binäre Schaltzustand oder aber der Messwert selbst kommuniziert wird. Hierzu bietet BLE neben der Verwendung spezialisierter Profile auch die Möglichkeit der Definition eigener Profile, worauf in diesem Fall zurückgegriffen wird [Cyp16b, Kapitel 2.3 und 4.6], [Cyp16a, Kapitel 4].

## 4 Zusammenfassung

In diesem Beitrag wurde eine generische Architektur für den Entwurf drahtloser Sensorsysteme vorgestellt und in der Funktion der einzelnen Schichten analysiert.

Exemplarisch an dem Entwurfsprozess eines Helligkeitssensors wurde die Anwendung der Architektur vorgestellt. In weiteren Forschungsschritten wird diese Architektur inhaltlich weiter ausgearbeitet und mit Designrichtlinien sowie Entwurfsmustern untermauert.

## Literaturverzeichnis

[Cyp16a] Cypress. Application Note 91162: Creating a BLE Custom Profile. 2. Nov. 2016.

[Cyp16b] Cypress. Application Note 92584: Designing for Low Power and Estimating Battery Life for BLE Applications. 14. März 2016.

[DP10] Waltenegus Dargie und Christian Poellabauer. Fundamentals of wireless sensor networks: theory and practice. Wiley series on wireless communications and mobile computing. Chichester, West Sussex, U.K. ; Hoboken, NJ: Wiley, 2010. 311 S. Metaarchitektur für IoT und Sensornetzwerke 9

[Fel+13] S. Feldmann u. a. "Model-Driven Engineering and Semantic Technologies for the Design of Cyber-Physical Systems". In: IFAC Proceedings Volumes 46.7 (Mai 2013), S. 210–215.

[ISO94] ISO/IEC. ISO/IEC 7498-1:1994(E): Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model. Genf/CH: ISO/IEC, 15. Nov. 1994.

[KKP99] Joseph M. Kahn, Randy H. Katz und Kristofer SJ Pister. "Next century challenges: mobile networking for Smart Dust". In: Proceedings of the 5th annualACM/IEEE international conference on Mobile computing and networking. ACM, 1999, S. 271–278.

[MF10] Friedemann Mattern und Christian Floerkemeier. "From the Internet of Computers to the Internet of Things". In: From active data management to eventbased systems and more. Springer, 2010, S. 242–259.

[Pue15] Fernando Puente Leon. Messtechnik. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015.

[SSS13] A. Smirnov, K. Sandkuhl und N. Shilov. "Multilevel Self-Organisation and Context-Based Knowledge Fusion for Business Model Adaptability in Cyber-Physical Systems". In: IFAC Proceedings Volumes 46.9 (2013), S. 2045–2050.

[Tow+14] Kevin Townsend u. a. Getting started with Bluetooth low energy: tools and techniques for low-power networking. Revised First Edition. OCLC: ocn870896083. Sebastopol, CA: O'Reilly, 2014. 164 S.