Hochschule für Technik
und Wirtschaft Berlin
University of Applied Sciences

Analyse, Planung und Implementierung einer internetfähigen Steuerung

Masterarbeit

von

Hans Maier

Matrikelnummer: s1234567

Fachbereich 1 – Energie und Information –
der Hochschule für Technik und Wirtschaft Berlin

zur Erlangung des akademischen Grades
Master of Engineering (M. Eng.)
im Studiengang
Informations- und Kommunikationstechnik

Tag der Abgabe: 07.04.2022

Erstgutachten:	Prof. Dr. Thomas Scheffler
Zweitgutachten:	Maxi Mustermann

Inhaltsverzeichnis

Kurzfassung

Diese Arbeit beschreibt die Erstellung einer internetfähigen Steuerung für elektrische Verbraucher. Anforderungen an die Steuerung werden nach dem Kano-Modell definiert. Über eine Nutzwertanalyse werden vorhandene Techniken und Standards bewertet. Exemplarisch wird die Lösung mit dem größten Nutzwert implementiert.

Ein Zigbit-Modul, bestehend aus einem AVR Mikrocontroller und einem IEEE 802.15.4 Funkchip, bildet die Basis für die Hardware. Zusammen mit einem selbst dimensioniertem Kondensatornetzteil wird das Modul in einem Steckdosengehäuse verbaut.

Um zukunftssicher zu sein, wird das Protokoll IPv6 eingesetzt. Die Adaptionsschicht übernimmt das Protokoll 6LoWPAN. Das verwendete Betriebssystem Contiki besitzt eine fertige Webserver-Applikation, die für die eigenen Zwecke angepasst wird. Das Protokoll IEC 60870-5-104 wird neu implementiert. Es basiert auf dem TCP/IP-Modell und wird vor allem im Umfeld von Energieleitsystemen eingesetzt. Es eignet sich besonders für einen automatisierten Zugriff.

Über eine öffentliche Adresse des IPv6-Tunnelbrokers SixXS ist die Steuerung weltweit erreichbar und der elektrische Verbraucher kann über einen Webbrowser oder von einem Energieleitsystem ein- und ausgeschaltet werden.

Die Anforderungen nach dem Kano-Modell wurden nahezu vollständig erfüllt. Die Implementierung eines Webservers und einer IEC 60870-5-104 Applikation ist mit den gegebenen limitierten Ressourcen möglich. Anwendungsmöglichkeiten für die Steuerung liegen im Bereich eHome und Smart Grid.

Abstract

This Master Thesis describes the implementation of a solution to control and monitor electric consumers via the Internet. Needs of this solution are defined by use of the Kano model. Existing technologies and standards are benchmarked by means of a cost-utility analysis. The solution that scores the highest value of benefit will be implemented typically.

A Zigbit Module forms the basis of the hardware. It bundles an AVR microcontroller and an IEEE 802.15.4 transceiver. Together with a self-dimensioned capacitive power supply it is mounted in a socket housing.

To be future-proof, the IPv6 protocol is used. The 6LoWPAN protocol handles the adaptation layer. Contiki is used as operating system. It is delivered with a ready-to-use web server application which is customized for the own purposes. The IEC 60870-5-104 protocol is implemented from scratch. It is based on TCP/IP and is used in the field of energy management systems. It is particularly suitable for automated access.

Via a public address given by IPv6 tunnel broker SixXS the solution is accessible worldwide. The electric consumer can be switched on and off by the means of a web browser or an energy management system.

The needs according to the Kano model are almost completeley achieved. It is possible to implement a solution consisting of a web server and IEC 60870-5-104 application in ressource constraint environments. Possible applications for such a solution are in the field of home automation and smart grid.

Kapitel 1: Einleitung

1.1 Aufgabenbeschreibung

Im Zuge dieser Masterarbeit soll eine Lösung erarbeitet und entwickelt werden, mit der ein beliebiger elektrischer Verbraucher über einen Fernzugriff gesteuert werden kann. Eine im Handel erhältliche Steckdose soll so modifiziert werden, dass die Stromzufuhr des Verbrauchers über das Internet ein- und ausschaltbar ist. Für den Zugriff auf die Steckdose soll keine zusätzliche Verkabelung notwendig sein. Das vorhandene Steuergerät der Steckdose soll durch eine selbst entwickelte Mikrocontrollerschaltung ersetzt werden. Der Mikrocontroller soll eine Applikation bereitstellen, damit menschliche Benutzer möglichst einfach auf die Steckdose zugreifen können.

Es sollen verschiedene existierende Technologien, die für diese Anwendung in Frage kommen, recherchiert und bewertet werden. Eine Lösung soll beispielhaft implementiert und nach einem Funktionstest kritisch begutachtet werden. Dadurch soll gezeigt werden, in wie weit es möglich ist sehr kleine Geräte mit besonders limitierten Ressourcen an ein lokales Netzwerk und an das Internet anzubinden.

1.2 Herangehensweise

In Kapitel 2 werden zuerst zentrale Begriffe, wie Smart Objects und Internet-of-Things, erläutert. Verschiedenen Technologien und Standards, die im Zusammenhang mit der Aufgabenstellung verwendet werden, werden vorgestellt. Diese Technologien werden eingeteilt in Übertragungstechniken und Kommunikationstechnologien. Eine Auswahl an Applikationen wird erläutert, die für den Zugriff auf die zu implementierende Lösung verwendet werden können. Betriebssysteme und Softwareumgebungen werden vorgestellt, die für die Bearbeitung der Aufgabenstellung in Frage kommen. Im Abschluss werden allgemein technische und nicht-technische Herausforderungen behandelt.

Im Kapitel 3 wird anhand einer Nutzwertanalyse die Komplettlösung ermittelt, die für die Aufgabenstellung am geeignetsten ist. Dazu werden zuerst, unabhängig von bestimmten Technologien, Anforderungen beschrieben, die an eine implementierte Lösung gestellt werden. Die Erfüllung dieser Anforderungen wird am Ende dieser Arbeit geprüft. Am Ende wird die Entscheidung hergeleitet, welche Lösung beispielhaft implementiert wird.

Die Beschreibung der eigentlichen Implementierung findet sich in Kapitel 4.

Nach Abschluss der Implementierung wurden Funktionstests der implementierten Applikationen durchgeführt. Der Testaufbau und die Funktionstests werden in Kapitel 5 behandelt.

Zum Ende der Arbeit wird die implementierte Lösung kritisch begutachtet. In Kapitel 6 wird geprüft, welche vorher beschriebenen Anforderungen erfüllt wurden. Der Nutzwert der implementierten Lösung wird ermittelt und mit dem Ergebnis der vorherigen Nutzwertanalyse verglichen. Verschiedene Anwendungsmöglichkeiten für die Implementierung und ähnliche Lösungsansätze werden behandelt. Ein Resümee der Arbeit wird gezogen.

Kapitel 2: Grundlagen

In diesem Kapitel wird nach anfänglichen Begriffserklärungen der schematische Hardwareaufbau eines Smart Objects erläutert. Es wird ein Überblick über verschiedene Technologien, die im Zusammenhang mit der Aufgabenstellung verwendet werden können, gegeben. Diese Technologien werden hier kurz angesprochen und einsortiert. Technische Details werden, soweit notwendig, in späteren Kapiteln erörtert.

Um darzustellen, wie ein Benutzer auf die zu implementierende Lösung zugreifen kann, werden verschiedene Applikationen vorgestellt. Außerdem werden verschiedene Betriebssysteme vorgestellt, die für eine Softwareimplementierung in Frage kommen.

Zum Abschluss dieses Kapitels werden technische und nicht-technische Herausforderungen angesprochen. [1]

2.1 Begriffserklärungen

Um die später vorgestellten Übertragungstechniken und Kommunikationstechnologien besser einordnen zu können, wird zu Beginn auf das OSI-7-Schichtenmodell eingegangen.

Außerdem werden weitere Begriffe erläutert, die helfen sollen, die zu implementierende Lösung einzugruppieren. Teilweise handelt es sich um englische Begriffe, weil diese auch in der deutschen Literatur gebräuchlicher sind.

2.1.1 OSI-7-Schichtenmodell

Das OSI-7-Schichtenmodell wird als bekannt vorausgesetzt. Weil es aber verschiedene Interpretationen und Namensgebungen gibt, werden in der Tabelle 2.1 Begriffe zugeordnet, wie sie in dieser Arbeit verwendet werden. Es werden die deutschen und englischen Begriffe für die sieben Schichten aufgelistet. Zusätzlich wird die Einordnung nach dem TCP/IP-Modell [RFC1122] gezeigt und es werden für jede Schicht Beispielprotokolle genannt.

Das OSI-7-Schichtenmodell wird in dieser Arbeit verwendet, um Übertragungstechniken und Kommunikationstechnologien einordnen zu können.

[1] Dieses Kapitel behandelt verschiedene Technologien, die von Smart Objects benutzt werden könnten. Zuerst werden allgemein die Anforderungen und Herausforderungen hervorgehoben und später die verschiedenen Technologien unter diesem Aspekt erläutert. Die Einteilung der Technologien ist an das TCP/IP-Modell angelehnt.

Tabelle 2.1: OSI-7-Schichtenmodell

Schicht			Beispielprotokoll	TCP/IP-Modell
7	Anwendungsschicht	Application Layer	HTTP, FTP, SNMP	Application Layer
6	Darstellungsschicht	Presentation Layer		
5	Sitzungsschicht	Session Layer		
4	Transportschicht	Transport Layer	TCP, UDP	Transport Layer
3	Vermittlungsschicht	Networking Layer	IPv4, IPv6, ICMP	Internet Layer
2	Sicherungsschicht	Data Link Layer	Ethernet, WLAN, ISDN	Link Layer
1	Physikalische Schicht	Physical Layer		

2.1.2 Eingebettete Systeme

Eingebettete Systeme – oft wird auch der englische Begriff embedded systems verwendet – sind kleine Computer, die Teilaufgaben in einem größeren technischen Kontext übernehmen. Sie bestehen häufig aus Mikrocontrollern oder digitalen Signalprozessoren und sind dafür ausgelegt eine bestimmte Funktion, oder auch mehrere bestimmte Funktionen, zu übernehmen. Im Gegensatz zu herkömmlichen Computern ist es nicht möglich, die Funktion eines eingebetteten Systems zu ändern [Heath 2003].

2.1.3 Drahtlose Sensornetze

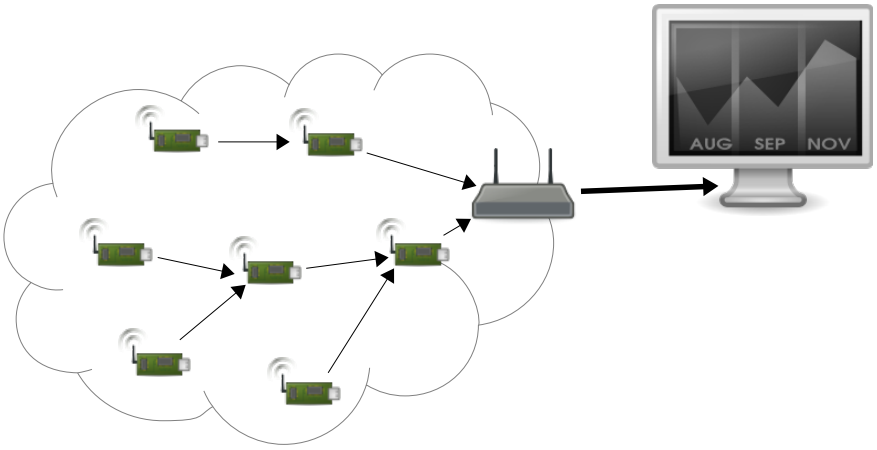

Abbildung 2.1: Aufbau eines drahtlosen Sensornetzwerkes

Bei drahtlosen Sensornetzen, international auch Wireless Sensor Networks (WSN) genannt, handelt es sich um Netzwerke aus kleinen Sensoren, die ihre Messergebnisse zu einer zentralen Station senden. Die Sensoren helfen dabei einander, die Informationen weiterzureichen (siehe Abbildung 2.1). So ein drahtloses Sensornetz kann zum Beispiel in einem Gebäude Temperatur- und Luftfeuchtigkeitswerte an eine zentrale Station liefern, die dann die Klimaanlage entsprechend ansteuert. Eine komplette Verkabelung jedes einzelnen Sensors

kann dabei je nach Anzahl sehr aufwendig und kostspielig sein. Wenn die Sensoren dazu noch mobil sein sollen, empfiehlt sich eine drahtlose Kommunikation.

Ein Sensor in einem drahtlosen Sensornetz ist ausgestattet mit einer Kommunikationseinheit, die sich selbstständig innerhalb des Netzes konfiguriert und darüber die eingelesenen Messgrößen zu einer zentralen Stelle transportiert.

2.1.4 Smart Objects

Diese Definition von Smart Objects beruht auf Vasseur und Dunkels [VD10]. Dabei handelt es sich um kleine Objekte, die mit folgenden Einheiten ausgestattet sind:

- eine Form von Sensor und/oder Aktor
- ein kleiner Mikrocontroller
- eine Kommunikationseinheit
- eine Energieversorgung

Ein Sensor und/oder Aktor wird benötigt, um mit der Umwelt interagieren zu können. Ein Sensor kann bestimmte Messgrößen erfassen, die dann verarbeitet werden können. Ein Aktor ist das Gegenstück zu einem Sensor. Über ihn kann aktiv die Umwelt beeinflusst werden. Die Kommunikationseinheit befähigt das Smart Object über ein Netzwerk kommunizieren zu können. Es ist dabei möglich mit anderen Smart Objects zu kommunizieren als auch mit weit entfernten Geräten, im Falle des Internets weltweit. Ein kleiner Mikrocontroller übernimmt die Informationsverarbeitung. Dieser nimmt Daten von Sensoren entgegen bzw. steuert den Aktor. Die Kommunikation wird ebenso von ihm geregelt, von einem Sensor gemessene Werte können über das Netzwerk weitergegeben werden. Der Mikrocontroller kann als Kern des Smart Object angesehen werden. Die Energieversorgung ist notwendig, um alle Einheiten ausreichend mit elektrischer Energie zu versorgen.

Alle diese technischen Eigenschaften machen ein Objekt aber noch nicht zu einem Smart Object. Erst die Anwendung und sein Verhalten machen aus einem Objekt mit den obigen Eigenschaften ein Smart Object. Allerdings ist es sehr schwierig, dieses Verhalten zu definieren. Die Anwendungen sind sehr unterschiedlich und es ist völlig unbekannt, wie Smart Objects in der Zukunft eingesetzt werden. Gemeinsam ist allen Anwendungen allerdings, dass Smart Objects mit der physikalischen Umwelt interagieren und über ein Netzwerk kommunizieren. Dieses Verhalten und die obigen technischen Eigenschaften machen ein Smart Object aus.

Von der Kommunikationstechnologie her ähnelt ein Smart Object einem Sensor in einem Sensornetz. Allerdings ist im Gegensatz zu einem Sensornetz ein Netz aus Smart Objects nicht allein auf die Datenübermittlung fokussiert. Bei einem Sensornetz ist der Datenfluss immer vom Sensor ins Netzwerk, wohingegen der Datenfluss bei einem Smart Object bidirektional ist. Der Fokus liegt dabei auf eine Vielzahl von Funktionen insbesondere der

Steuerung und Überwachung [VD10, Seite 12]. Es ist also die Anwendung, die ein Gerät zu einem Smart Object macht. Aber Entwicklungen, hauptsächlich in der Energieversorgung und Kommunikationstechnik, kommen beiden Forschungsgebieten zugute.

2.1.5 Internet-of-Things

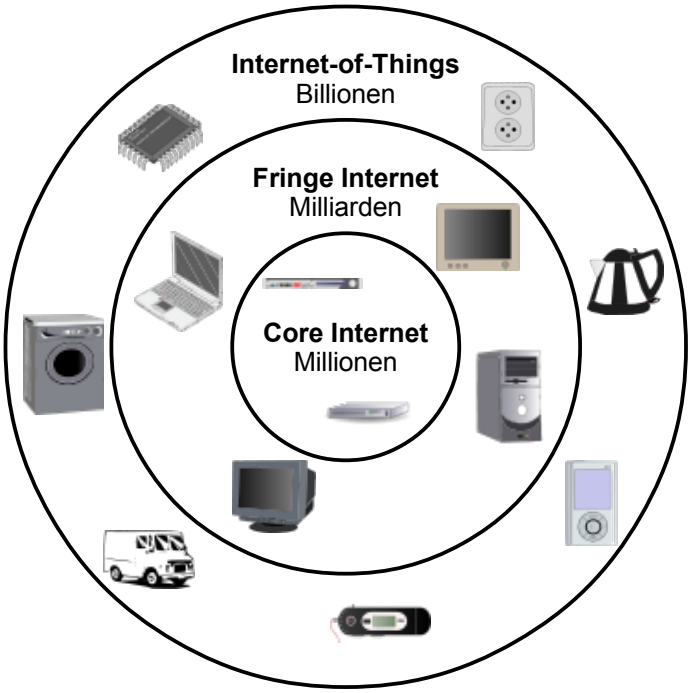

Abbildung 2.2: Die Internet-of-Things Vision [SB09, Figure 1.2]

Das Internet-of-Things, auch Internet der Dinge genannt, ist eine Vision, wie sich das Internet in Zukunft entwickeln kann. Es soll aus der Vernetzung vieler Kleinstgeräte – auch oder vielleicht hauptsächlich Smart Objects – bestehen. Shelby und Bormann [SB09] betrachten das Internet-of-Things als eine weitere Schale des Internets (Abbildung 2.2), die gerade anfängt sich heraus zu bilden. Der Kern des Internets (Core Internet) besteht heute aus vielen Routern und Servern, die zusammen Millionen von Teilnehmern ausmachen. Um diesen Kern herum liegt das sogenannte Fringe Internet, was beispielsweise aus privaten Rechnern oder Laptops besteht, aber auch aus lokalen Netzwerken, die zum Internet verbunden werden. Die Teilnehmer des Fringe Internet gehen in die Milliarden. Um dieses Fringe Internet herum beginnt sich eine weitere Schale zu bilden, das sogenannte Internet-of-Things. Auf lange Sicht kann hier mit Billionen von Teilnehmern gerechnet werden. Hauptsächlich bestehen diese Teilnehmer aus IP-fähigen eingebetteten Systemen (embedded systems). Eine so große Zahl an Teilnehmern kann nur mithilfe des Internet Protokolls der Version 6 ans Internet angeschlossen werden, da das bisherige Internet Protokoll der Version 4 keine freien Adressräume mehr zur Verfügung hat.

2.2 Theoretische Grundlagen

...

2.3 Kommunikationstechnologien

...

2.4 Betriebssysteme für SmartObjects

...

2.5 Zusammenfassung/Bewertung

...

Kapitel 3: Konzeption / Architektur

Da die verschiedenen Technologien aufeinander aufbauen, manchmal zueinander kompatibel sind, manchmal inkompatibel sind, werden nachfolgend mögliche Komplettlösungen hergeleitet. Sie werden zuerst kurz vorgestellt, miteinander verglichen und bewertet. Um einen systematischen Entscheidungsprozess zu gewährleisten, werden die möglichen Komplettlösungen mithilfe der Nutzwertanalyse verglichen und bewertet. Der Vorteil der Nutzwertanalyse ist es, dass die Bewertung nachvollziehbar und überprüfbar ist. Nach der Auswertung wird die Lösung ausgesucht, die implementiert werden soll.

3.1 Entwurf des eigenen Lösungskonzepts

- Darstellung des eigenen Ansatzes, ggf. Abgrenzung von anderen Arbeiten
- Vor-/Nachteile
- ...

3.2 Auswahl/Entwurf der Hardware

...

3.3 Auswahl/Entwurf der Software

3.3.1 Entwurf von Lösungsmodulen

...

Kapitel 4: Implementierung

4.1 Aufbau der Implementierung

Die Implementierung der Lösung erfolgte in zwei Teilen. Es wurde die Hardware erstellt und zeitlich parallel dazu wurde die Software entwickelt. Beide Teile konnten relativ unabhängig voneinander implementiert werden. Die Schnittstelle von Hardware und Software ist das Zigbit-Modul ATZB-24-A2. Auf diesem Modul wird die fertige Software einprogrammiert. Die Hardware versorgt das Modul mit Energie und schaltet nach Softwarevorgaben die Steckdose ein und aus.

4.2 Contiki-Verzeichnisstruktur

Die Contiki-Verzeichnisstruktur besteht aus verschiedenen Unterverzeichnissen, die verschiedene Bedeutungen haben.

Codeauszug 4.1: Contiki-Verzeichnisstruktur mit ausgewählten Unterverzeichnissen

```
/apps
    /ftp
    /ping6
    /telnet
    /telnetd
    /twitter
    /webserver
    /webserver-nano
    ...
/core
    /lib
    /net
    /sys
    ...
/cpu
    /arm
    /avr
    ...
/doc
/examples
```

```
21     /webserver-ipv6-raven
22     ...
23  /platform
24     /avr-raven
25     /avr-zigbit
26     ...
27  /tools
```

Das Verzeichnis /core beinhaltet das Kernstück von Contiki. Hier wird zum Beispiel das System im Unterverzeichnis /sys definiert. Das beinhaltet das Prozesshandling und Protothreads sowie verschiedene Timer. Ebenfalls befindet sich hier im Unterverzeichnis /net der Netzwerk-Stack aufbauend auf uIP. Zum Netzwerk-Stack gehört auch IEEE 802.15.4 und 6LoWPAN. Im Unterverzeichnis /lib stehen verschiedene libraries zur Verfügung.

In den beiden Verzeichnissen /cpu und /platform ist die unterschiedliche Hardware beschrieben, die von Contiki unterstützt wird. Ein Programm, das auf einem Zigbit-Modul vom Hersteller Atmel geladen werden soll, verwendet die Verzeichnisse /platform/avr-zigbit und /cpu/avr.

Das Verzeichnis /examples beinhaltet Beispielprojekte. Hier gibt es ein Projekt webserver-ipv6-raven. Anhand dieses Beispiels wird deutlich, was notwendig ist, um eine Webserver-Applikation für das Ravenboard zu kompilieren.

Im Hauptverzeichnis gibt es eine Datei Makefile.include. Diese Datei ist Teil des Contiki-Makefile-Systems. Sie wird innerhalb eines Contiki-Projektes aufgerufen und bindet, abhängig von der Konfiguration des Projektes, die richtigen Dateien des Contiki-Systems ein.

4.2.1 Übersetzung eines Contiki-Programms

Ein Programm wird in Contiki mithilfe des Befehls „make“ übersetzt. Am Beispiel des Projekts webserver-ipv6-raven soll exemplarisch gezeigt werden, wie das Makefile System funktioniert. Durch den Befehl „make“ wird die Datei Makefile im gleichen Verzeichnis abgearbeitet.

Codeauszug 4.2: Auszug aus examples/webserver-ipv6-raven/Makefile

```
1 ifndef TARGET
2   TARGET=avr-raven
3   MCU=atmega1284p
4 endif
5 all:
6   ${MAKE} -f Makefile.webserver TARGET=$(TARGET) NOAVRSIZE=1 webserver6.elf
```

Hier wird das TARGET, also die Plattform, und die MCU, die Microcontroller Unit, gesetzt und dann die Datei Makefile.webserver aufgerufen. Dabei wird der Parameter NOAVRSIZE

gesetzt, um beim Übersetzen eine zusätzliche Ausgabe zur Speicherbelegung (avr-size) zu unterdrücken. Das Programm wird als Datei webserver6.elf erstellt.

Codeauszug 4.3: Auszug aus examples/webserver-ipv6-raven/Makefile.webserver

```
1 all: webserver6
2 APPS=raven-webserver raven-lcd-interface
3 UIP_CONF_IPV6=1
4 CONTIKI = ../..
5 include $(CONTIKI)/Makefile.include
```

In der Datei Makefile.webserver werden die Applikationen raven-webserver und raven-lcd-interface über die Variable APPS eingebunden. Die Compiler Variable `UIP_CONF_IPV6` wird gesetzt, um IPv6 zu aktivieren, weiterhin wird das Haupt-Makefile `Makefile.include` eingebunden.

Codeauszug 4.4: Auszug aus examples/webserver-ipv6-raven/webserver6.c

```
1 #include "webserver-nogui.h"
2 /*---------------------------------------------------------------------*/
3 AUTOSTART_PROCESSES(&webserver_nogui_process);
4 /*---------------------------------------------------------------------*/
```

In der Datei webserver6.c wird ausgewählt, welche Contiki-Prozesse automatisch gestartet werden sollen. In unserem Beispiel ist das der Prozess „webserver_nogui_process“. Dieser Prozess ist Teil der Applikation raven-webserver.

4.2.2 Programmieren eines Mikrocontrollers

Durch Übersetzung des Contiki-Beispielprojektes webserver-ipv6-raven wird eine Datei „webserver6.elf“ im ELF Format erzeugt. Hiervon wird eine Kopie „webserver6-avr-raven.elf“ erstellt. Diese Datei enthält Informationen darüber, was in den Flash und in den EEPROM des AVR Mikrocontrollers geladen werden muss. Ebenfalls beinhaltet es Informationen über das Setzen der Fuse Bits. Fuse Bits sind Einstellungen des Mikrocontrollers, die nicht von der Software geändert werden können. Sie schalten gewisse Funktionen, zum Beispiel woher die Taktfrequenz bezogen wird, ein oder aus.

Codeauszug 4.5: Auszug aus examples/webserver-ipv6-raven/Makefile

```
1 TARGET=avr-raven
2 MCU=atmega1284p
3 OUTFILE=webserver6-$(TARGET)
4 avr-objcopy -O ihex -R .eeprom -R .fuse -R .signature \
5           $(OUTFILE).elf $(OUTFILE).hex
6 avr-size -C --mcu=$(MCU) $(OUTFILE).elf
```

Durch zusätzliche Befehle im Makefile wird eine Datei „webserver6-avr-raven.hex“ erzeugt. Diese Datei enthält den Inhalt, der in den Flash geschrieben werden soll, im Intel-HEX-Format. Mit dem „avr-size“-Befehl wird die Anzahl der Bytes angezeigt, die im Flash, im RAM und im EEPROM benötigt werden. Dies kann mit dem zur Verfügung stehenden Speicher verglichen werden.

Codeauszug 4.6: Ausgabe vom Befehl „avr-size“

```
AVR Memory Usage
----------------
Device: atmega1284p

Program: 70874 bytes (54.1% Full)
(.text + .data + .bootloader)

Data: 13013 bytes (79.4% Full)
(.data + .bss + .noinit)

EEPROM: 63 bytes (1.5% Full)
(.eeprom)
```

Durch Hinzufügen eines zusätzlichen Befehls ins Makefile kann eine Datei „webserver6-avr-raven_eeprom.hex“ erzeugt werden. Diese Datei enthält dann den Inhalt im Intel-HEX-Format, der in den EEPROM geschrieben werden soll. Mit den beiden Dateien im Intel-HEX-Format ist es möglich, den Mikrocontroller mit einem Programmiergerät zu beschreiben, das kein ELF Format lesen kann.

Codeauszug 4.7: Befehl um eeprom.hex zu erzeugen

```
avr-objcopy -O ihex -j .eeprom \
            -set-section-flags=.eeprom="alloc,load" --change-section-lma \
            .eeprom=0 $(OUTFILE).elf $(OUTFILE)_eeprom.hex
```

Mittels des Programms Atmel Studio 6 werden wahlweise die erzeugten Dateien im Intel-HEX-Format oder die erzeugte Datei im ELF Format in den Flash und in den EEPROM des Mikrocontrollers programmiert. Zu Beginn wurde das Beispielprojekt webserver-ipv6-raven auf den Mikrocontroller AT-Mega1284P eines Atmel Ravenboard programmiert. Dabei wurde die Programmierschnittstelle ISP und das Programmiergerät Atmel STK500 verwendet.

Später bei der entwickelten Hardware wurde die Programmierschnittstelle JTAG und das Programmiergerät Atmel JTAGICE3 verwendet.

Kapitel 5: Tests

In diesem Kapitel wird die erstellte Lösung getestet und die Entwicklung validiert. Dazu ist die Testumgebung zu beschreiben.

Ziel ist der glaubhafte Nachweis der Funktionsfähigkeit, bzw. die quantitative Bewertung der erstellten Lösung.

Ein wichtiges Kriterium ist die Nachvollziehbarkeit der Tests. Das heißt, die Tests müssen so beschrieben sein, dass der Leser der Arbeit die Ergebnisse eigenständig wiederholen und validieren kann.

5.1 Herleitung von Test-Cases

Herleitung der Testmethodik.

5.1.1 Funktionale Tests

...

5.1.2 Leistungstests/Performancetests

...

5.2 Bewertung der Ergebnisse

...

Kapitel 6: Fazit

In diesem Kapitel wird die erstellte Lösung begutachtet. Es werden Anwendungsmöglichkeiten, allgemein von Smart Objects und speziell für die Implementierung, behandelt. Mögliche Erweiterungen werden vorgeschlagen und ein Fazit der Arbeit wird gezogen.

6.1 Analyse der Implementierung

Die erstellte Lösung wird kritisch begutachtet. Dafür werden zuerst Kostenbetrachtungen bezogen auf die Hardware und auf die Software durchgeführt. Die Ergebnisse fließen in die abschließende Bewertung mit ein.

6.1.1 Kostenbetrachtung

Um in der Bewertung der Implementierung die Kosten einschätzen zu können, sollen hier zum einem die Hardwarekosten vorgestellt werden. Dazu sollen die einzelnen Materialkosten der verwendeten Bauteile zusammengefasst werden. Um die gesamten Hardwarekosten ermitteln zu können, müssen auch Entwicklungskosten und Fertigungskosten berücksichtigt werden. Fertigungskosten beinhalten dabei die eigentlichen Arbeitsstunden, die für die Fertigung aufgebracht werden müssen, als auch die Investitionskosten für den Fertigungsarbeitsplatz. Weil das sehr abhängig von der Art der Fertigung ist, sei es eine Einzelstückfertigung oder eine Serienfertigung, werden diese Kosten hier nicht näher betrachtet. Die Hardwarekosten werden rein durch die Materialkosten repräsentiert.

Zum anderen sollen die Softwarekosten abgeschätzt werden. Dies wird durch die Ermittlung der Entwicklungskosten für die Contiki-Applikation iec104 erreicht. Anhand eines angenommenen Stundensatzes für einen Softwareentwickler und den aufgewendeten Stunden werden die Kosten für die Entwicklung berechnet.

Materialkosten

In Tabelle 6.1 sind die Bauteile aufgelistet, die zum Erstellen der Steckdose verwendet wurden. Sie benennt das Bauteil und zeigt die Anzahl an, die verbaut wurden. Die Bauteile wurden – bis auf die Tchibo Steckdose – vom Handelsunternehmen Mouser Electronics bezogen. In der dritten Spalte ist die entsprechende Artikelnummer aufgelistet. Die vierte

Tabelle 6.1: Materialkosten Steckdose

#	**Bauteil**	**Artikelnummer**	**Einzel-preis in €**	**Massen-preis in €**
Zigbit-Platine				
1	ATZB-24-A2	556-ATZB-24-A2	25,43	14,42
1	Widerstand 100kOhm	71-CMF60100K00FKEB	0,206	0,099
1	Widerstand 1kOhm	71-CMF551K0000FHEK	0,116	0,005
1	Kondensator 3,3uF	667-EEU-HD1H3R3	0,165	0,104
1	Spannungsregler LP2950CZ-3.0	926-2950CZ-3.0/NOPB	0,676	0,27
1	Steckverbinder FFC 6 Pin	538-52271-0679	1,30	0,583
2	Steckverbinder FFC 18 Pin	538-52271-1879	1,71	0,94
Steckdosen-Platine				
1	Kondensator X2 275V	80-R46KN368050M2M	0,66	0,263
1	Widerstand 560Ohm, 5Watt	594-AC05W560R0J	0,38	0,198
2	Widerstand 1MOhm	594-MRS251M1%TR	0,074	0,038
2	Gleichrichterdiode	512-1N4004	0,076	0,025
2	Zener-Diode 24V	512-1N4749ATR	0,203	0,036
1	Kondensator 470uF	667-EEU-FR1E471YB	0,248	0,152
1	NPN-Transistor BC547B	512-BC547B	0,186	0,05
1	Widerstand 220kOhm	271-220K-RC	0,125	0,012
Steckdosen-Gehäuse				
1	Tchibo Digitale Zeitschaltuhr	4 043002 669758	5,99	5,99
Gesamtpreis			**38,61**	**24,27**

Spalte zeigt den Preis bei dem Bezug von nur einem Bauteil. Bei einer Massenfertigung der Steckdose werden andere Preise angeboten. Der Stückpreis bei einer Bestellung von 1000 Fertigungssätzen ist in der letzten Spalte aufgelistet. Die Preise wurden am 06.05.2013 abgefragt und sind inklusive Mehrwertsteuer angegeben.

Die Gesamtmaterialkosten der verwendeten Bauteile betragen 38,61€. Nicht berücksichtigt sind Verbrauchsmaterialien. Dazu zählen die Leiterplatte für die Zigbit-Platine, die aus einer vorhandenen Lochrasterplatine ausgesägt wurde, Leitungen für die Verkabelung und Lötzinn. Diese Kosten sollen hier vernachlässigt werden. Bei einer Kalkulation für eine Fertigung in einem Unternehmen müssen diese als Materialgemeinkostenzuschlag zu den Materialkosten hinzugefügt werden. Zusammen mit den Fertigungskosten, also die Arbeitskosten, die zum Fertigen der Steckdose notwendig sind, und den Verwaltungs- und Vertriebsgemeinkostenzuschlag ergeben sich die Selbstkosten je Stück.

Entwicklungskosten iec104

Um den Aufwand von einer Contiki-Applikation abzuschätzen, sollen beispielhaft die Entwicklungskosten der implementierten Applikation iec104 ermittelt werden. Zum Ermitteln

der Kosten, muss der Stundensatz des Entwicklers mit der Zeit in Stunden multipliziert werden, die dafür notwendig ist, eine solche Applikation zu programmieren. Dazu gehört eine Designphase, die eigentliche Programmierung und eventuelle Fehlerkorrekturen. Weil die Kosten proportional zur Zeit sind, soll nur diese betrachtet werden. Es wird geschätzt, dass für die Entwicklung der Applikation iec104 etwa drei bis vier Mannwochen benötigt wurden. Die Schätzung wird dadurch erschwert, dass nicht kontinuierlich an der Entwicklung gearbeitet wurde. Die Arbeit daran wurde öfter durch andere notwendige Arbeiten unterbrochen.

Deswegen soll noch eine quantitative Bewertung anhand der Anzahl geschriebener Zeilen Quellcode LOC (Lines Of Code) stattfinden. In der Praxis ist solch eine Bewertung, also der Rückschluss von den LOC auf den Aufwand, allerdings problematisch. Die Zeit, die für die Designphase verwendet wird, wird nicht berücksichtigt. Eine überlegte und optimierte Programmierung benötigt tendenziell weniger LOC. Die Qualität der Programmierung wird auch nicht bewertet. Verschiedene Programmiersprachen und individuelle Programmierstile wirken sich ebenso auf die LOC aus.

Trotzdem gibt es Faustformeln, die einen Zusammenhang zwischen der Anzahl geschriebener Zeilen Quellcode LOC und der eingesetzten Entwicklungszeit sehen. Nach [Ludewig und Lichter 2010, Seite 305] wird in einem Softwareprojekt nach n Stunden Aufwand – wobei hier der Gesamtaufwand gemeint ist, nicht die reine Programmierungszeit – ein System mit zwei mal n LOC erzeugt. Ein studentisches Projekt, das nicht kommerziell vertrieben werden soll, kann in der gleichen Zeit in etwa die dreifache Anzahl an LOC liefern. Dort können demnach mit jeder Stunde Aufwand in etwa sechs LOC entwickelt werden.

Um nach dieser Faustformel herauszufinden, wie viele Stunden in die Entwicklung der Contiki-Applikiation iec104 eingeflossen sind, werden die Anzahl an Zeilen Quellcode ermittelt. Die Contiki-Applikation iec104 besteht aus 14 Dateien:

- 1 Makefile
- 7 Header-Dateien
- 6 C-Dateien

Insgesamt enthalten diese Dateien 1011 Zeilen. Wenn die 211 Leerzeilen und 112 Kommentarzeilen abgezogen werden, bleiben 688 Zeilen Quellcode. Nach der Faustformel oben ergibt sich eine Entwicklungszeit von etwa 115 Stunden. Bei der Annahme von einem Stundensatz von 100€ ergeben sich Entwicklungskosten von 11.500€. Bei einer Wochenarbeitszeit von 40 Stunden, ergeben sich ungefähr 3 Mannwochen, die notwendig sind, um die Contiki-Applikation iec104 zu entwickeln. Das deckt sich in etwa mit den Erfahrungen aus dieser Arbeit.

6.2 Anwendungsmöglichkeiten

Dieses Kapitel stellt zuerst allgemein verschiedene Anwendungsbereiche von Smart Objects vor. Darauffolgend wird erörtert, in welchen Anwendungsbereichen die erstellte Lösung eingesetzt werden könnte. Anhand der Erfahrung, die bei der Implementierung gemacht wurden, werden mögliche Erweiterungen behandelt.

6.2.1 Anwendungsbereiche von Smart Objects

Smart Objects können in einer Vielzahl von Anwendungsbereichen eingesetzt werden, da sie sehr flexibel in ihrer Beschaffenheit sind. Die Programmierung kann individuell angepasst werden. Durch die Verwendung der richtigen Sensoren und Aktoren sind sie für verschiedenste Einsatzbereiche verwendbar. Nachfolgend sollen folgende fünf mögliche Anwendungsbereiche kurz beschrieben werden:

- eHome-Bereich
- Gebäudeautomation
- Industrieautomatisierung
- Logistik
- Smart Grid

Der eHome-Bereich ist ein Bereich in dem sich mehrere proprietäre Lösungen verbreitet haben, die oft nicht untereinander kompatibel sind. Dies kann auch ein Mitgrund dafür sein, dass sich die Heimautomatisierung bisher nicht so entwickelt hat wie vor etwa 7-9 Jahren prognostiziert [VD10, S. 360]. Wichtig ist auch, gerade im eHome-Bereich, eine einfache Installation der Geräte. Nur wenn ein Endnutzer ohne spezielles Expertenwissen Geräte in Betrieb nehmen und verwenden kann, wird sich eine Lösung durchsetzen können. Mögliche Einsatzgebiete von Smart Objects im eHome-Bereich sind zum Beispiel Steuerungen von Licht, der Heizung, von Fenster, von Rollläden und von Türschlössern.

Der eHome-Bereich wird für private Wohnhäuser eingesetzt. Im Gegensatz dazu zielt die Gebäudeautomation eher auf den professionellen Einsatz in großen Gebäuden meist im Zusammenspiel mit einem visualisierten Gebäudemanagementsystem. Vermesan und Friess [Vermesan und Friess 2011, Seite 304ff] beschreiben unter dem Titel „Smart IPv6 Building“ verschiedene Forschungsprojekte, die die Verwendung von IPv6 in der Gebäudeautomation untersuchen. Unter anderem wird auch das Hobnet-Projekt erwähnt, das spezielle Anwendungsfälle von Smart Objects in der Gebäudeautomation untersucht. Hier kommt auch 6LoWPAN zum Einsatz. Allgemein sind die Ziele einer Gebäudeautomation Energieeinsparungen, Sicherheit und Komfortgewinn.

Der Bereich Smart Grid wird einer der größten Anwendungsbereiche für Smart Objects werden [Hersent et al. 2012, Seite 271ff]. Seit mehreren Jahren geht der Trend in der Stromerzeugung immer mehr in Richtung erneuerbaren Energiequellen wie Windkraftanlagen oder Photovoltaikanlagen. Diese sind aber im Gegensatz zu klassischen fossilen Kraftwerken dezentral aufgestellt. Das führt zu einem erheblichen Umbruch in der Stromnetzführung. Die Energie wird nicht allein an wenigen zentralen Örtlichkeiten durch große Kraftwerke, sondern auch dezentral durch kleine teils privat teils kommerziell geführten Energiequellen erzeugt. Das erschwert die Kraftwerksregelung und die Leitung des Lastflusses. Um trotzdem eine gute Netzstabilität zu gewährleisten, muss das Stromnetz intelligenter und vielfältiger überwacht und gesteuert werden.

Eine Maßnahme dabei ist das Smart Metering. Stromverbrauchszähler liefern über eine Kommunikationsschnittstelle den aktuellen Energieverbrauch an das Energieversorgungsunternehmen. Solche Art intelligente Zähler existieren schon länger für große Energieverbraucher wie Produktionsbetriebe, seit einigen Jahren werden Smart Meter auch für Privathaushalte angeboten.

6.2.2 Anwendungsmöglichkeiten für die Implementierung

Das Protokoll IEC104 ist ein Fernwirkprotokoll aus dem Bereich der Energieautomatisierung. Der Bereich Smart Grid ist also ein klassischer Anwendungsfall für dieses Protokoll. An ein zentrales Leitsystem werden Informationen übertragen und Steuerbefehle entgegen genommen. Allerdings wird der Zugriff auf eine Steuerung von einem elektrischen Verbraucher im Privathaushalt für ein Energieversorgungsunternehmen nicht von Bedeutung sein. Ein möglicher Anwendungsfall liegt eher in einem Smart Meter. Hier könnten mittels IEC104 Verbrauchsstände übertragen werden. Zusätzlich wäre in Absprache mit dem Privathaushalt eine Steuerung möglich. Genaue Anwendungsfälle müssen noch erprobt werden. Denkbar wäre eine Notabschaltung einer lokalen Photovoltaikanlage, wenn mehr Energie ins Stromnetz eingespeist wird als dort verbraucht wird bzw. in andere Stromnetzregionen abgeführt werden kann. Ebenfalls denkbar ist eine gezielte Steuerung von größeren Energieverbrauchern, bei denen der Einsatz zeitlich flexibel ist (Demand Side Management). Beispiele dafür sind die Waschmaschine oder die Batterie eines Elektrofahrzeugs. Bei einer großer Netzauslastung können diese steuerbaren Verbraucher zurückgefahren werden. Das Energieversorgungsunternehmen muss dann weniger Reserven für Spitzenlast vorhalten. So hilft eine intelligente Netzführung dabei, Kosten in der Energieerzeugung zu reduzieren.

6.2.3 Erweiterungsvorschläge für die Implementierung

Die erstellte Lösung ist ein Prototyp, der beispielhaft implementiert worden ist. Während der Arbeit sind mehrere Ideen entstanden, wie die Implementierung verbessert oder erweitert werden kann. Zuerst werden Vorschläge für die IEC104-Slave-Applikation aufgelistet.

- Informationsmeldungen mit Zeitstempel
  Die Statusänderung der Steckdose erfolgt aktuell über den Datentyp M_SP_NA_1 (Type Ident 1): Einzelbitmeldung ohne Zeitstempel. Eine Verbesserung wäre die Verwendung des Datentyps M_SP_TB_1 (Type Ident 30): Einzelbitmeldung mit dem Zeitstempel CP56Time2a. Hier wird zusätzlich ein Zeitstempel mit Jahr, Monat, Tag, Stunde, Minute, Sekunde und Millisekunde übertragen. So ist der exakte Zeitpunkt der Statusänderung bekannt. Voraussetzung dafür ist aber, dass die Steuerung mit der aktuellen Zeit synchronisiert ist. Über IEC104 existiert dafür eine Möglichkeit mit dem Datentyp C_CS_NA_1 (Type Ident 103): Zeitsynchronisationsbefehl. Ein Zeitstempel wird vom IEC104-Master zum IEC104-Slave gesendet und dort übernommen. Weil die Übertragungszeit dabei aber nicht berücksichtigt wird, ist die Verwendung vom IEC-Standard nur bedingt empfohlen. In der Praxis wird oft das Protokoll NTP (Network Time Protocol) verwendet. Eine Implementierung von NTP im Betriebssystem Contiki existiert bereits [Web:Contiki-syslog].

- interne Statusinformationen des Contiki Betriebssystems
  Das Modul iec104-para kann mit zusätzlichen Informationsobjekten erweitert werden. Zusätzliche interne Statusinformationen wie die Anzahl der laufenden Prozesse, TCP/IP-Verbindungen oder die Betriebszeit könnten übertragen werden. Für Messwerte gibt es bereits den Datentyp M_ME_NA_1 (Type Ident 9): Messwert, normalisiert und ohne Zeitstempel. Weitere Datentypen können implementiert werden.

Eine weitere Verbesserungsmöglichkeit betrifft nicht die Implementierung selbst sondern dem Testaufbau. Als 6LoWPAN-Router wurde ein handelsüblicher PC mit einer Linux-Installation verwendet. Als 6LoWPAN-Schnittstelle wurde ein USB-Stick RZUSBSTICK von Atmel verwendet, der auf dem PC eine Ethernet-Schnittstelle simuliert. Unter anderem hat das den Nachteil, dass auf dem PC der 6LoWPAN-Netzwerkverkehr nicht beobachtet werden kann, weil nur der simulierte Ethernet-Verkehr sichtbar ist. Es wird aber aktuell daran gearbeitet, 6LoWPAN direkt im Linux-Kernel zu implementieren [Ott 2012]. Eine eingeschränkte Variante wird schon seit der Kernel Version 3.2.46 unterstützt und laufend erweitert. Als Funkchips werden der MRF24J40 von Microchip und der AT86RF230 von Atmel, der auch im Zigbit-Modul verbaut ist, unterstützt. Ott stellt eine Hardware vor bestehend aus einem BeagleBone und einem MRF24J40MA Funkchip. Damit kann 6LoWPAN nativ unter Linux verwendet werden ohne zusätzliche Hardware und ohne ein zusätzliches Betriebssystem. Es ist allerdings nicht bekannt, ob es Interoperabilitätsprobleme zwischen der 6LoWPAN-Implementierung im Linux-Kernel und im Contiki-Betriebssystem gibt.

6.3 Zusammenfassung

Es konnte gezeigt werden, dass eine internetfähige Steuerung mithilfe eines 8-bit Mikrocontrollers implementiert werden kann. Das Betriebssystem Contiki stellt dazu den notwendigen TCP/IP-Stack und andere Werkzeuge und Hilfsmittel zur Verfügung. Eine Webserver-Anwendung und eine Vielzahl anderer Anwendungen sind Teil des Betriebssystems. Mit der

Contiki-Applikation iec104 wurde gezeigt, dass die Implementierung des Protokolls IEC104 in einer limitierten Umgebung durchaus möglich ist. Auch die Verwendung von IEC104 im Zusammenspiel mit IPv6 hat funktioniert. Bisher war keine Implementierung von IEC104 über IPv6 bekannt. Generell zeigt die Contiki-Applikation iec104, dass die Implementierung neuer internetfähiger Anwendungen möglich und nicht aufwendiger als bei anderen Betriebssystemen ist. Die geringen Ressourcen müssen allerdings bei der Programmierung berücksichtigt werden.

Bei der Implementierung ist der 8KByte große RAM-Speicher die markanteste Ressourcengrenze. Die Auslastung beträgt 88,9%. Das Hinzufügen von zusätzlichen Contiki-Applikationen oder das Erweitern der Applikation iec104 ist deswegen nicht ohne Weiteres möglich. Durch den Einsatz von anderer Hardware mit einem größeren RAM-Speicher kann dieses Problem umgangen werden. So steht bei dem AVR Ravenboard mit 16KByte doppelt so viel RAM-Speicher zur Verfügung. Der Hersteller Redwire bietet mit dem Econotag ein fertiges Modul mit USB-Schnittstelle an. Verwendet wird der Freescale MC13224V, der einen 32-bit ARM7 Mikrocontroller mit einer IEEE-802.15.4-Schnittstelle kombiniert. Er verfügt über 128KByte Flash- und 96KByte RAM-Speicher. Das Betriebssystem Contiki unterstützt diese Modul über die Platform redbee-econotag. Beide, das AVR Ravenboard und das Econotag, sind aber für den Einbau in die verwendete Steckdose zu groß.

Als Kommunikationstechnologie wurde 6LoWPAN verwendet. Es scheint ideal für den Einsatz bei ressourcenbeschränkten Geräten zu sein. Anfang dieses Jahres hat die ZigBee Alliance die Spezifikation ZigBee IP veröffentlicht [ZigBee-IP]. Damit werden unter der Verwendung von 6LoWPAN vermaschte drahtlose IPv6-Netzwerke unterstützt. Dies ist ein Beispiel dafür, dass die Verwendung von 6LoWPAN in vielen Bereichen voranschreitet. Eine weitere Entwicklung, die die Verbreitung von 6LoWPAN unterstützt, ist die Spezifizierung des Protokolls RPL [RFC6550]. RPL ist ein Routing-Protokoll, das für verlustbehaftete Sensornetze entwickelt worden ist. Die speziellen Anforderungen konnten von existierenden Routing-Protokolle wie OSPF oder RIP nicht erfüllt werden. Die Implementierung von RPL im Betriebssystem Contiki wird ContikiRPL genannt [Tsiftes et al. 2010].

Sehr wichtig für die weitere Verbreitung von 6LoWPAN ist die Interoperabilität von verschiedenen Implementierungen. Ko u. a. [Ko11] haben zwei unabhängige Implementierungen, Contiki und TinyOS, zusammen getestet. Die Interoperabilität zwischen beiden war gegeben, allerdings hatten kleine Unterschiede im jeweiligen Protokollstack einen Einfluss auf die Gesamtsystemleistung. Neben der Interoperabilität ist die Leistungsfähigkeit bei dem Zusammenspiel verschiedener Implementierungen von Bedeutung.

Als weitere Schwierigkeit kommt hinzu, dass viele Implementierungen den Funkempfänger so oft wie möglich ausschalten. Im Betriebssystem Contiki steht diese Funktionalität unter dem Namen ContikiMAC zur Verfügung. So kann der Energieverbrauch um bis zu 80% reduziert werden [Dunkels 2011]. Das gesteuerte Ausschalten des Funkempfängers hat allerdings einen markanten Einfluss auf das Kommunikationsverhalten im Netzwerk. Weil keine Spezifikationen oder Standards für diese Funktionalität existieren, verhalten sich Implementierungen sehr verschieden. Dunkels, Eriksson und Tsiftes [Dunkels et al. 2011]

beschreiben, dass das Zusammenspiel von Kommunikationstechnologie und Ausschaltverhalten des Funkempfängers ein wichtiges Forschungsgebiet für das Internet-of-Things ist.

Anhang A: Der Blindtext

Weit hinten, hinter den Wortbergen, fern der Laender Vokalien und Konsonantien leben die Blindtexte. Abgeschieden wohnen Sie in Buchstabhausen an der Kueste des Semantik, eines grossen Sprachozeans. Ein kleines Baechlein namens Duden fliesst durch ihren Ort und versorgt sie mit den noetigen Regelialien. Es ist ein paradiesmatisches Land, in dem einem gebratene Satzteile in den Mund fliegen. Nicht einmal von der allmaechtigen Interpunktion werden die Blindtexte beherrscht - ein geradezu unorthographisches Leben.

Eines Tages aber beschloss eine kleine Zeile Blindtext, ihr Name war Lorem Ipsum, hinaus zu gehen in die weite Grammatik. Der grosse Oxmox riet ihr davon ab, da es dort wimmele von boesen Kommata, wilden Fragezeichen und hinterhaeltigen Semikoli, doch das Blindtextchen liess sich nicht beirren. Es packte seine sieben Versalien, schob sich sein Initial in den Guertel und machte sich auf den Weg. Als es die ersten Huegel des Kursivgebirges erklommen hatte, warf es einen letzten Blick zurueck auf die Skyline seiner Heimatstadt Buchstabhausen, die Headline von Alphabetdorf und die Subline seiner eigenen Strasse, der Zeilengasse. Wehmuetig lief ihm eine rethorische Frage ueber die Wange, dann setzte es seinen Weg fort. Unterwegs traf es eine Copy. Die Copy warnte das Blindtextchen, da, wo sie herkaeme waere sie zigmal umgeschrieben worden und alles, was von ihrem Ursprung noch uebrig waere, sei das Wort und und das Blindtextchen solle umkehren und wieder in sein eigenes, sicheres Land zurueckkehren. Doch alles Gutzureden konnte es nicht ueberzeugen und so dauerte es nicht lange, bis ihm ein paar heimtueckische Werbetexter auflauerten, es mit Longe und Parole betrunken machten und es dann in ihre Agentur schleppten, wo sie es fuer ihre Projekte wieder und wieder missbrauchten. Und wenn es nicht umgeschrieben wurde, dann benutzen Sie es immernoch.

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltextverzeichnis

Stichwortverzeichnis

Literaturverzeichnis

[Dunkels 2011] Adam Dunkels. *The ContikiMAC Radio Duty Cycling Protocol*. Techn. Ber. T2011:13. Swedish Institute of Computer Science, Dez. 2011. URL: `http://www.sics.se/~adam/dunkels11contikimac.pdf`.

[Dunkels et al. 2011] Adam Dunkels, Joakim Eriksson und Nicolas Tsiftes. *Low-power Interoperability for the IPv6-based Internet of Things*. Mai 2011. URL: `http://www.sics.se/~adam/dunkels11adhoc.pdf`.

[Heath 2003] Steve Heath. *Embedded Systems Design*. 2. Aufl. Elsevier Ltd., 2003, S. I–XX, 1–430. ISBN: 978-0750655460.

[Hersent et al. 2012] Olivier Hersent, David Boswarthick und Omar Elloumi. *The Internet of Things: Key Applications and Protocols*. 6. Aufl. John Wiley und Sons Ltd, 2012. ISBN: 978-1-1199943-5-0.

[Ko11] JeongGil Ko u. a. „Beyond Interoperability: Pushing the Performance of Sensornet IP Stacks“. In: *Proceedings of the ACM Conference on Networked Embedded Sensor Systems, ACM SenSys 2011*. Seattle, WA, USA, Nov. 2011. URL: `http://www.sics.se/~adam/ko11beyond.pdf`.

[Ludewig und Lichter 2010] Jochen Ludewig und Horst Lichter. *Software Engineering - Grundlagen, Menschen, Prozesse, Techniken*. Deutsch. 2. Aufl. dpunkt.verlag, 2010, S. I–XXI, 1–665. ISBN: 978-3-89864-662-8.

[Ott 2012] Alan Ott. „Wireless Networking with IEEE 802.15.4 and 6LoWPAN“. In: *Embedded Linux Conference Europe 2012 (ELC2012)*. Nov. 2012. URL: `http://www.signal11.us/oss/elce2012/` (besucht am 10. 06. 2013).

[RFC1122] R. Braden. *Requirements for Internet Hosts — Communication Layers*. RFC 1122. Internet Engineering Task Force, Okt. 1989. URL: `http://tools.ietf.org/html/rfc1122`.

[RFC6550] T. Winter and P. Thubert and A. Brandt and J. Hui and R. Kelsey and P. Levis and K. Pister and R. Struik and J. Vasseur and R. Alexander. *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*. RFC 6550. Internet Engineering Task Force, März 2012. URL: `http://tools.ietf.org/html/rfc6550`.

[SB09] Zach Shelby und Carsten Bormann. *6LoWPAN: the wireless embedded internet*. John Wiley und Sons Ltd, 2009, S. I–XX, 1–223. ISBN: 978-0-470-74799-5.

[Tsiftes et al. 2010] Nicolas Tsiftes, Joakim Eriksson und Adam Dunkels. „Poster Abstract: Low-Power Wireless IPv6 Routing with ContikiRPL“. In: *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN 2010)*. Stockholm, Sweden, Apr. 2010. URL: `http://www.sics.se/~adam/tsiftes10rpl.pdf`.

[VD10] Jean-Philippe Vasseur und Adam Dunkels. *Interconnecting Smart Objects with IP - The Next Internet*. Morgan Kaufmann, 2010. ISBN: 978-0123751652. URL: `http://TheNextInternet.org/`.

[Vermesan und Friess 2011] Dr. Ovidiu Vermesan und Dr. Peter Friess. *The Internet of Things - Global Technological and Societal Trends*. 1. Aufl. River Publishers, 2011. ISBN: 978-87-92329-73-8.

[Web:Contiki-syslog] Anuj Sehgal. *contiki-ntp-syslog*. URL: `http://cnds.eecs.jacobs-university.de/software/contiki-syslog/` (besucht am 10. 06. 2013).

[ZigBee-IP] ZigBee Alliance. *ZigBee IP Specification*. Feb. 2013. URL: `http://www.zigbee.org/Specifications/ZigBeeIP/Download.aspx` (besucht am 13. 04. 2013).

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Berlin, den 07.04.2022

Hans Maier