Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Alexander Pautz

Kabellose, stromsparende Sensornetzwerke im Bereich Ambient Intelligence

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Alexander Pautz

Kabellose, stromsparende Sensornetzwerke im Bereich Ambient Intelligence

Masterarbeitarbeit eingereicht im Rahmen der Masterarbeitprüfung
im Studiengang Master of Science Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Ing. Korf
Zweitgutachter : Prof. Dr. rer. nat. Klemke

Abgegeben am 8. März 2012

Alexander Pautz

Thema der Masterarbeit

Kabellose, stromsparende Sensornetzwerke im Bereich Ambient Intelligence

Stichworte

Funk, stromsparend, kabellos, Sensornetzwerk, WSN, Mesh, ZigBee, Ambient Intelligence, Living Place Hamburg

Kurzzusammenfassung

Eine intelligente Wohnung wird mit den nötigen Sensoren ausgestattet, damit die Computer Steueraufgaben erfolgreich umsetzen können. Hierfür wird ein kabelloses stromsparendes Sensornetzwerk entwickelt, welches sich leicht in eine bestehende Wohnung integrieren lässt. Durch die Verwendung von Funk werden der Installationsaufwand und die damit verbundenen Kosten niedrig gehalten. Der Einsatz stromsparender Technologien ermöglicht lange Laufzeiten und einen geringen Wartungsaufwand.

Alexander Pautz

Title of the paper

Design und implementation of a microprocessor interface for ARM processors including an exemplary application

Keywords

radio, low-current, wireless, Sensornetwork, WSN, Mesh, ZigBee, Ambient Intelligence, Living Place Hamburg

Abstract

A smart home will be equipped with the necessary sensors, so the computer can successfully implement control functions. For this purpose a wireless low current sensor network is developed, which can be easily integrated into an existing home. Through the use of wireless technology, long lead times and low maintenance requirements can be realised.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1. Einleitung

1.1. Motivation

Die Entwicklung von Computern schreitet unaufhörlich voran. So erinnerten uns noch vor einigen Jahren PCs an die nächsten Termine. Heute wird diese Aufgabe zunehmend von Smartphones erledigt. Doch wie sieht die Zukunft aus? Wie weit kann die Integration von Computern gehen? Wie gut können Computer ”erahnen”, was wir als nächstes tun und wie können sie uns dabei am besten unterstützen? Diese Fragen sollen in dem Forschungsbereich ”Ambient Intelligence” geklärt werden.

Ein Bereich von ”Ambient Intelligence” ist dabei die Erforschung einer intelligenten Wohnung. Der Aufgabenbereich beginnt bei der Steuerung des Lichts und Klimas entsprechend den Vorlieben der Bewohner. Weiterhin können Aufgaben, welche wir täglich in gleicher Weise erledigen, von der Wohnung selbst übernommen werden. Beispiele dafür sind Kaffeekochen und Lüften nach dem Aufstehen, Heizen bevor wir von der Arbeit kommen oder Einschalten des Fernsehers, wenn wir uns um 20 Uhr in den Fernsehsessel setzen.

Die Integration der Intelligenz in heutige Wohnungen ist ein wichtiger Aspekt, wenn die Technologie in den nächsten ein bis zwei Jahrzehnten unser Leben verändern wird. Diese Technologie wird aber nur zum Einsatz kommen, wenn Faktoren wie Installationsaufwand, Installations- und laufende Kosten in einem bezahlbaren Rahmen bleiben.

1.2. Ziel dieser Arbeit

Das Ziel dieser Arbeit ist es eine intelligente Wohnung mit den nötigen Sensoren auszustatten, damit die Computer die Steueraufgaben erfolgreich umsetzen können. Hierfür wird ein kabelloses stromsparendes Sensornetzwerk entwickelt, welches sich leicht in eine bestehende Wohnung integrieren lässt. Durch die Verwendung von Funk werden Installationsaufwand und damit verbundene Kosten niedrig gehalten. Durch Einsatz stromsparender Technologien werden die laufenden Kosten niedrig gehalten und die Umweltbelastungen minimiert.

1.3. Rahmen der Arbeit

Die Arbeit wird im Umfeld der Ambient Intelligence Forschung geschrieben. Der Living Place Hamburg an der HAW Hamburg ist so ein Labor für Ambient Intelligence. Der Kern des Labors ist eine etwa 130 m^2 große und voll funktionsfähige Wohnung. Im Rahmen des Living Place arbeiten zahlreiche Studenten und erforschen Möglichkeiten, wie Computer uns im Alltag unterstützen können. Das Sensornetzwerk dieser Masterarbeit wird im Living Place eingesetzt und stellt alle erfassten Messgrößen aus der Wohnung anderen Projekte zur Verfügung. Das folgende Foto 1.1[1] zeigt einen Teil der Wohnung.

Abbildung 1.1.: Blick in den Living Place Hamburg

[1] Quelle: http://livingplace.informatik.haw-hamburg.de/blog/?p=439 - Abruf: 02.03.12

2. Anforderungen

Bevor mit der Entwicklung begonnen werden kann, müssen zunächst die Anforderungen an das Sensornetzwerk aufgestellt werden. Diese Anforderungen werden im folgenden beschrieben.

2.1. Bereitstellung von Messwerten

Das oberste Ziel ist die Erfassung und Bereitstellung von Messwerten für andere Projekte. Die Messwerte sind über die Blackboardarchitekture zur weiteren Verwendung bereitzustellen. Im internen Wiki des Living Place Hamburg sind die Nachrichtenformate anderer Projekte für die Kommunikation über die Blackboardarchitekture definiert. Die Nachrichten des Sensornetzwerks sollen sich an diesen bereits existieren Nachrichten orientieren. Insbesondere soll für die Kommunikation das JSON-Format genutzt werden.

Aktuell wird die Blackboardarchitekture durch einen *Active MQ Server* und der dokumentenorientierten Datenbank *Mongo DB* bereit gestellt (sieh hierzu Otto und Voskuhl [2010] und Otto und Voskuhl [2011]). In einer anderen Masterarbeit beschäftigt sich ein Kommilitone derzeit mit der Modellierung der Wohnung in *CIM* (siehe Johannsen [2010]). Sollte dieses Modell den Active MQ ablösen, so ist es erforderlich, dass das Sensornetzwerk auch mit der neuen Infrastruktur zusammen arbeiten kann. Eine leichte Anpassbarkeit der Kommunikationsschnittstelle ist vorzusehen.

2.2. Zuverlässigkeit

Verrichtet das Sensornetzwerk seine Aufgabe nur unzuverlässig, kann sich kein anderes Projekt auf den regelmäßigen Erhalt von Messwerten verlassen. Auch wenn kaum eine Software fehlerfrei ist, so sind Fehler durch geeignete Tests zu minimieren. Auf der anderen Seite kann es durch äußere Einflüsse zu Störungen in kabelloser und kabelgebundener Kommunikation kommen. Das Sensornetzwerk muss in der Lage sein, mit diesen Störungen umzugehen. Im Speziellen muss das Netzwerk bei Verbindungsabbrüchen selbst in der Lage sein, die Kommunikation erneut aufzunehmen.

2.3. Erweiterbarkeit

Alle Softwarekomponenten sowie die Hardware der Sensorknoten müssen so ausgelegt sein, dass neue Sensoren eingebunden werden können. Es soll möglich sein, sowohl zusätzliche Sensoren an einen bestehenden Sensorknoten anzuschließen als auch neue Sensorknoten in das bestehende Netzwerk integrieren zu können.

2.4. Unauffälligkeit

Die Sensorknoten sollen möglichst klein sein. So fallen sie weniger auf und können leichter hinter Blenden oder durch andere Objekte verborgen montiert werden. Es wird die Größe einer handelsüblichen Zigarettenschachtel angestrebt. Zum Vergleich wird eine Pall Mall ”menthol blast” Zigarettenschachtel mit 19 Zigaretten verwendet. Es wurden die Abmessungen von 86 x 55 x 23 mm ermittelt.

Für den Fall, dass ein Sensorknoten an einer Stelle nicht unauffällig genug angebracht werden kann oder die Anbringung des Knotens aus bautechnischen Gründen nicht dort möglich ist, wo ein Sensorwert aufgenommen werden muss, sollen die eigentlichen Sensoren bis zu einem halben Meter weit abgesetzt werden können. Die Sensoren und der Sensorknoten können dann mit einem dünnen unauffälligem Kabel miteinander verbunden werden. Ein Knoten ließe sich damit in oder hinter einem Schrank einbauen, während der Sensor selbst außerhalb befestigt ist.

2.5. Kabellos

Verlegte Kabel können ebenso störend wirken wie die Sensorknoten selbst. Damit die Menge an Kabeln möglichst gering gehalten wird, soll eine kabellose Kommunikation verwendet werden. Ebenso sollen möglichst viele Sensorknoten ohne eine kabelgebundene Stromversorgung auskommen.

2.6. Geringer Installationsaufwand

Im Rahmen eines Projektes (siehe [Pautz und Johannsen, 2010, S.1]) wurde die Aufgabe gestellt, ein Produkt aus der Industrie in den Bereich Ambient Intelligence zu integrieren. Dies erfolgte beispielhaft an Fenstermotoren und Heizungsventilen der D+H Mechatronic AG. Es wurde ein auf Echtzeitlinux basierendes Gateway entwickelt. Andere Projekte können nun über den Active MQ darauf zugreifen und die Fenster sowie Heizungen steuern bzw. deren Statusinformationen auslesen.

Abbildung 2.1.: Oberlicht mit eingebautem Motor

Das oben stehende Foto (Abbildung 2.1) zeigt einen an einem Oberlicht verbauten Fenstermotor. Von diesem Motor und jedem der anderen neun führt ein Datenkabel zum entwickeltem Gateway. Insgesamt brachte diese Installation einen hohen Installationaufwand mit sich, wobei alleine das Verlegen und Anschließen der Kabel über eine Woche in Anspruch nahm.

Die Zeit der Installation soll nun verringert werden. Durch die Verwendung einer kabellosen Kommunikation und einer autarken Stromversorgung soll der Installationsaufwand gering gehalten werden. Nur Sensorknoten, welche für die Aufrechterhaltung der Kommunikationsinfrastruktur benötigt werden, sollen mit Strom versorgt werden.

2.7. Geringer Wartungsaufwand

Eine Verringerung des Installationsaufwandes soll nicht durch einen höheren Wartungsaufwand erkauft werden. So sollen Sensorknoten, welche Batterien zur Stromversorgung verwenden nicht nur auf Grund von verbrauchten Batterien gewartet werden müssen. Ein mit drei Sensoren ausgestatteter Sensorknoten muss mindestens alle 5 Minuten alle Sensoren abfragen können und soll dabei eine Laufzeit von mindestens einem Jahr haben. Der Stromverbrauch der Sensoren muss zwingend berücksichtigt werden. Die verwendeten Batterien sollen in jedem Elektrofachmarkt erhältlich sein. Für Größe und Anzahl der Batterien soll es keine Einschränkungen geben, lediglich die maximale Baugröße des Sensorknoten von einer Zigarettenschachtel soll nicht überschritten werden. Unter Berücksichtigung der Baugröße und der Verfügbarkeit in Elektrofachmärkten kommen so zum Beispiel maximal vier Micro oder drei Mignon Batterien in Frage. Dabei können die Batterietechnologien Alkaline oder Lithium verwendet werden.

2.8. Strom- und Spannungsversorgung der Sensoren

Es soll eine möglichst große Bandbreite an Sensoren unterstützt werden. Damit zwei Schaltkreise zusammen geschaltet werden können, müssen sie mit dem gleichen Spannungspegel arbeiten. Moderne ICs arbeiten mit einer IO-Spannung von 3,3 Volt, wobei auch noch sehr viele ICs mit 5 Volt arbeiten. Damit eine breite Palette von Sensoren genutzt werden kann, muss die IO-Spannung der Sensorknoten entweder 3,3 Volt oder 5 Volt betragen.
Die Sensoren müssen vom Sensorknoten mit Strom versorgt werden. Die Spannung soll dabei genauso hoch sein wie die IO-Spannung. Ein Sensorknoten muss einem Sensor mindest für den Zeitraum der Messung einen Strom von 20 mA zur Verfügung stellen können.

2.9. Unterstützte Schnittstellen

Um eine Kommunikation zwischen Sensor und Sensorknoten zu ermöglichen, muss neben der Spannung auch das gleiche Kommunikationsprotokoll verwendet werden. Dafür muss die Hardware der Sensorknoten die wichtigsten Kommunikationsschnittstellen unterstützten, die im Bereich der Inter-IC-Kommunikation Verwendung finden. Die wichtigsten Schnittstellentypen sind:

- diskrete IOs (GPIO)
- AD-Wandler
- I^2C
- SPI

Es ist gut, wenn die serielle Schnittstelle unterstützt wird, doch wird diese meist erst bei komplexeren Bausteinen als Sensoren für die Ansteuerung verwendet.

3. Analyse vergleichbarer Arbeiten

Die folgenden Projekte zeigen beispielhaft den Einsatz von Mesh Netzwerken zur Erfassung von physikalischen Messgrößen. Neben der reinen Auflistung dieser Beispiele werden diese auf nützliche Hinweise für die eigene Entwicklung analysiert.

3.1. Erzmine

Mit Hilfe eines kabellosen Sensornetzwerks sollen die Prozesse einer Mine in Skellefteå (Nord Schweden) optimiert werden (siehe Niels Aakvaag [2005]). Für eine Machbarkeitsstudie hat die ABB Corporate Research ein experimentelles ZigBee basiertes Sensornetzwerk in der Mine installiert. Dieses Netzwerk überwacht die Pumpen der Erzmühlen. Aus den Messwerten können Informationen über mögliche Optimierungen der Pumpen gewonnen werden.

Das aufgebaute Netzwerk besteht aus sieben Knoten, wobei einige der Knoten nicht in der Funkreichweite des zentralen Knotens sind und Routerknoten verwendet werden müssen, um die räumliche Ausdehnung des Netzwerks zu ermöglichen[1]. Bei dem Experiment stand ein Aspekt im Vordergrund - alle Teilnehmer müssen ohne eine externe Stromversorgung auskommen.

3.1.1. Zeitsynchronisierte Verbindungen

Wie in ZigBee-Teilnehmerklassen beschrieben, dürfen ZigBee-Router während des Betriebes nicht in Energiesparmodi wechseln, da sie jeder Zeit bereit sein müssen Pakete von anderen Teilnehmern weiterzuleiten. Ohne den Wechsel in einen Energiesparmodus ist jedoch der Stromverbrauch der Router zu hoch und sie können nicht ohne externe Stromversorgung betrieben werden.

Im aufgebauten Testnetzwerk wird oberhalb des ZigBee-Stacks eine Zeitsynchronisation eingeführt. Zusätzlich wird sichergestellt, dass alle Netzwerkteilnehmer ihre Messwerte zur gleichen Zeit übertragen. Durch diese beiden Maßnahmen können die Router in den Sendepausen deaktiviert werden und die Verwendung von Batterien als Stromversorgung wird wieder möglich.

Die Lösung reduziert den Stromverbrauch erheblich, ist jedoch nicht perfekt. Diese Lösung kann nur dann eingesetzt werden, wenn alle Teilnehmer ihre Messdaten im

[1] Siehe Kap. 4.2.3 für genauere Informationen über den Aufbau eines ZigBee-Netzwerks.

gleichen Intervall übermitteln. Zu dem erfolgt die Zeitsynchronisation oberhalb des ZigBee-Stacks. Die Nachricht zur Zeitsynchronisation muss somit durch den gesamten Stack bis zur Anwendung durchgereicht werden. Es entsteht ein größeres Delay und ein größerer Jitter, als wenn die Nachricht im Media Access Control Layer (MAC-Layer) verarbeitet wird. Bei ZigBee handelt es sich um ein Mesh Netzwerk. Die Synchronisationsnachricht muss alle Teilnehmer erreichen. Mit jedem Hop im Netzwerk steigen Delay und Jitter. Bei dem aufgebauten Netzwerk wurde vor allem versucht, das entstehende Delay zu kompensieren. So wurde zum Beispiel für die Durchreichung der Nachricht vom Stack eine Zeit von zwei Millisekunden veranschlagt. Dennnoch müssen die einzelnen Knoten, 200 ms bevor die Synchronisationsnachricht erwartet wird, aufwachen [Niels Aakvaag, 2005, s. 6].

3.2. Teeplantage

Der Bericht ”WSN Design and Implementation in a Tea Plantation for Drought Monitoring” von der ”South China Agricultural University” beschreibt den Aufbau eines kabellosen Sensornetzwerks zur Überwachung einer Teeplantage Sun u. a. [2010]. Die Teeplantage wird dabei auf Umweltbedingungen wie Temperatur, Boden- und Luftfeuchtigkeit überwacht.

Die Netzwerkknoten sind mit dem Betriebssystem TinyOS ausgestattet, welches eine 6lowWPAN (IPv6 over Low Power Wireless Personal Area Network) Implementation bietet und auch für den Aufbau des Netzwerks dieser Masterarbeit in Frage kam (siehe 4.1 bzw. [Pautz, 2011b, S.14ff]). Das Netzwerk der Teeplantage besteht aus 25 Knoten, von denen 11 zur Erfassung der Messwerte verwendet werden. Ein Knoten dient als Gateway und die übrigen werden zum Routen der Datenpakete benötigt.

3.2.1. Knotenverteilung und Datenverluste

Die Knoten wurden zunächst zufällig auf der Teeplantage verteilt. Die Entfernung der Knoten betrug dabei zwischen 20 und 80 Meter. Ab 50 Meter Entfernung zwischen zwei Knoten kam keine Funkverbindung mehr zustande. Bei der Übertragung der Messwerte im Intervall von 10 Sekunden kam es an einem Knoten fast zu 38 % Paketverlust. Dieser Knoten hatte 3 Nachbarn, von denen einer 20, einer 30 und einer 36 Meter weit entfernt war. An einem weiterem Knoten, dessen längste Funkstrecke 30 Meter betrug, kam es zu über 18 % Paketverlust. Durchschnittlich betrug die Paketverlustrate knapp 24 %.

In einem zweiten Schritt wurden die Knoten gleichmäßig im Abstand von 30 Metern verteilt, so dass ein rautenförmiges Muster entstand. Bei diesem Aufbau kam es bei einem Sendeintervall von 10 Sekunden zu maximal 8 und durchschnittlich 5 % Paketverlust.

Obwohl die Strecken zwischen den Knoten bei der unsymmetrischen Verteilung kaum größer und zum Teil sogar kürzer waren, kam es zu deutlich mehr Paketverlusten als bei der symmetrischen Verteilung. Auch wenn es sich in der Wohnung um weit kürzere Strecken handelt als auf der Teeplantage, sollte dieser Aspekt berücksichtigt werden und zumindest die Routerknoten des ZigBee-Netzwerks sind gleichmäßig zu verteilen.

3.3. Patientenüberwachung

In der Diplomarbeit von Marcel Noe am Karlsruhe Institut of Technology Noe [2010] geht es um die "Überwachung von Patienten bei einem Massenanfall von Verletzten". Da bei solchen Ereignissen nicht genug Personal für die Überwachung aller Patienten zur Verfügung steht, muss diese Überwachung durch technische Hilfsmittel unterstützt werden. Wenn die Erstversorgung der Patienten in Feldlagern in der Nähe des Unfallortes geschehen muss, fehlt es zudem an Infrastruktur.

Bei den Patienten wird die Atmung und der Puls von Sensoren überwacht. Die Sensoren übermitteln ihre Daten per Funk. Bei der Funkübertragung kommt ZigBee zum Einsatz, welches durch die Bildung eines Meshnetzwerks die nötige Infrastruktur selbst mitbringt.

Obwohl es bei der gesamten Diplomarbeit genauso um die Übermittlung von Sensorwerten geht und teilweise sogar die gleiche Hardware genutzt wird, kann von dieser Diplomarbeit nichts übernommen werden. Die ZigBee-Module der Firma Atmel werden mit Hilfe von AT-Kommandos gesteuert, wie sie auch beim Handy üblich sind. Die Software der Sensorknoten wird in einem ARM7 Controller ausgeführt. In der vorliegenden Masterarbeit befindet sich die Software der Sensorknoten in den Funkmodulen selbst. Durch den Wegfall des zweiten Controllers können die Sensorknoten platzsparender und energieeffizienter aufgebaut werden. Es muss sogar angemerkt werden, das die in der Diplomarbeit entwickelten Platinen nicht der Applikation Node von MeshNetics folgen und eventuell nicht die maximale Funkreichweite erzielen (vgl. [Noe, 2010, S. 80] und [MeshNetics, 2007, S. 16]).

4. Übertragungsprotokoll ZigBee

Die Wahl des Funkprotokolls wurde bereits in Projekt 2 vorgenommen (siehe [**?**, PJ2:2011]. In Projekt 2 wurden vor allem technische Aspekte betrachtet, welche zur Wahl von ZigBee gegenüber 6loWPAN führten. Hier soll noch einmal auf ZigBee eingegangen werden. Es steht dabei eine kurze Einführung in ZigBee und die aktuelle Verwendung im Vordergrund.

4.1. Wahl des Übertragungsstandards

Ziel von Projekt 2 (Pautz [2011b]) war die Wahl eines kabellosen Übertragungsstandards für das Sensornetzwerk dieser Masterarbeit. Zum Einsatz sollte ein herstellerunabhängiger stromsparender Funkstandard kommen. Für die genauere Analyse wurden die zwei auf IEEE 802.15.4 aufbauenden Funkstandards ZigBee und IPv6 over Low Power Wireless Personal Area Network (kurz 6loWPAN) betrachtet. Beide Funktstandards wurden anhand verschiedener Kriterien analysiert.

Übertragungsrate und Frequenz

Bei der Analyse war die Datenübertragungsrate der wichtigste Punkt. Wenn diese zu gering ist, kommt es zu Verzögerungen oder gar Datenverlust. Für ein Netzwerk mit bis zu 300 Sensoren wurde eine minimale Datenrate von 50 kBit abgeschätzt. Da beide Funkstandards auf den unteren Schichten IEEE 802.15.4 nutzen, haben beide die gleiche Datenrate. IEEE 802.15.4 bietet dabei 250 kBits auf jedem seiner 27 verfügbaren Kanäle. Die Kanalwahl wird jedoch etwas eingeschränkt, da in Deutschland nicht alle Frequenzen nutzbar sind. So dürfen 11 Kanäle im Bereich von 902 bis 928 MHz nicht verwendet werden. Für den einzigen Kanal im Bereich von 868 MHz gilt eine maximale Sendedauer von 1 %. So bleiben nur 11 Kanäle im Bereich von 2,4 GHz, welche für diese Datenübertragung verwendet werden können.

Vergleich von 6lowWPAN und ZigBee

6loWPAN Implementationen sind von TinyOS und Contiki quelloffen verfügbar. Beide Implementationen bieten eine gewisse Auswahl an Prozessoren und RF-Chips unterschiedlicher Hersteller an (z. B. ATMega128 von Atmel, MSP430 von Texas Instruments und Cortex M3). Von jedem Hersteller werden jedoch nur wenige

Prozessoren unterstützt. Ist ein Mikrocontroller mit spezieller Peripherie erforderlich, muss eventuell das gesamte Betriebssystem portiert werden. Die Lösungen sind Open Source, eine Portierung ist somit möglich. Hilfe kann hier aber nur aus Foren der Entwickler erwartet werden. TinyOS benutzt einen C-Dialekt, welcher das Programmieren vereinfachen soll, aber zunächst von jedem Benutzer von TinyOS erlernt werden muss.

Die ZigBee-Stacks sind zum Beispiel von Atmel und Texas Instruments verfügbar. Sie sind aber nur auf den Produkten des jeweiligen Hersteller verwendbar. Ein Wechsel des Herstellers bedeutet in jedem Fall den Wechsel des Stack. Beide Stacks laufen jedoch auf nahezu allen Prozessoren der Hersteller, welche genügend Speicherplatz und Rechenleistung bieten. Support kann hier von den Herstellern selbst sowie aus Internetforen erwartet werden.

Der Vorteil liegt bei ZigBee, auch wenn dies die Bindung an einen Hersteller bedeutet. Besserer Support und eine große Auswahl an Prozessoren unterschiedlicher Leistungsklassen innerhalb einer Familie sprechen für einen ZigBee Stack.

Aufbau eines Testnetzwerks

Nach der Entscheidung ZigBee zu verwenden, erfolgte eine Einarbeitung in den ZigBee Softwarestack ”BitCloud” von Atmel (Atmel Corporation [2011e]). Für den Aufbau eines Testnetzwerks wurden ATmega128RFA1 Prozessoren auf einem Evaluationsboard von sparkfun[1] verwendet. Mit diesem Testnetzwerk wurden Tests zu Funkreichweite, Pingzeiten und Übertragungsrate durchgeführt. Die Reichweite beläuft sich dabei auf über 13 Meter, die durchschnittlichen Pingzeiten auf 22,4 ms und die Datenrate abzüglich des Protokolloverheads auf 27 kbit/s. Die kompletten Ergebnisse können in [Pautz, 2011b, S. 20 ff] nachgelesen werden.

4.2. Funktionsweise von ZigBee

ZigBee ist ein auf dem IEEE 802.15.4 Protokoll aufsetzender Funkstandard für Wireless Personal Area Networks (WPAN). Durch die Verwendung von IEEE 802.15.4 stehen insgesamt 27 Kanäle, von denen 17 in Europa verwendet werden dürfen, zur Verfügung. Es stehen 16 Kanäle im Bereich von 2,4 GHz und ein Kanal im Bereich von 868 MHz zur Verfügung. Die Brutto-Datenrate liegt bei bis zu 250 kbit/s.

Mit ZigBee lassen sich Mesh-Netzwerke aufbauen. Somit müssen zwei Funkteilnehmer nicht in direkter Funkreichweite sein, um miteinander zu kommunizieren. Sie müssen lediglich im selben Netzwerk angemeldet sein. Auf dem gleichem Kanal können mehrere unabhängige ZigBee-Netzwerke nebeneinander existieren. Es besteht die Möglichkeit der verschlüsselten Datenübertragung, welche ähnlich der WEP-Verschlüsselung für WLANs arbeitet.

[1] www.sparkfun.com

Durch die selbstständige Verwaltung des Netzwerk als Mesh erfüllt ZigBee bereits die Anforderung "Erweiterbarkeit", in dem jederzeit neue Teilnehmer hinzugefügt oder entfernt werden können.

4.2.1. ZigBee Adressen

Jeder ZigBee Knoten besitzt eine 64 Bit lange Adresse. Diese Adresse muss auch über die Grenzen eines Netzwerks hinweg eindeutig sein. Nachdem ein Teilnehmer einem Netzwerk beigetreten ist, wird eine 16 Bit Adresse vergeben. Diese 16 Bit Adresse kann statisch oder dynamisch vergeben werden. Für die Kommunikation innerhalb des Netzwerks werden nur die kürzeren 16 Bit Adressen verwendet.

4.2.2. ZigBee Datenübertragung

Der Datenaustausch in einem ZigBee-Netzwerk erfolgt paketorientiert. Ein Paket bietet dabei eine Kapazität für bis zu 128 Byte Nutzdaten. Fehlerprüfung und automatische Neuübertragung sind bereits integriert. Zudem bietet ZigBee virtuelle Übertragungskanäle an. Diese *Endpunkt* genannten Kanäle ähneln den Ports bei TCP/IP.

4.2.3. ZigBee-Teilnehmerklassen

ZigBee definiert drei verschiedene Teilnehmerklassen.

Coordinator Der wichtigste Teilnehmer, welcher in jedem ZigBee-Netzwerk vorhanden sein muss, ist der *Coordinator*. Der Coordinator übernimmt Aufgaben wie Adressvergabe und Adressverwaltung im Meshnetzwerk. Er besitzt immer die 16 Bit Adresse 0x0000. Fällt der Coordinator aus, bricht das gesamte Netzwerk zusammen. In jedem ZigBee-Netzwerk stellt der Coordinator einen Single Point of Failure dar. Eine Möglichkeit diesen Fehler zu verringern besteht darin, einen anderen Router so zu konfigurieren, dass er bei einem Ausfall des Coordinators dessen Aufgabe übernimmt. Abgesehen von den Verwaltungsaufgaben verhält sich der Coodinator wie ein gewöhnlicher Routerknoten.

Router Die nächste Klasse von ZigBee-Teilnehmern sind *Router*. Da Pakete im Mesh nur von Routern und dem Coordinator weitergeleitet werden können, werden Router immer dann benötigt, wenn die Ausdehnung des Netzwerks größer ist, als die Funkreichweite des Coordinators ist.

End-Device Die letzte Klasse von ZigBee-Teilnehmern sind *End-Devices*. End-Devices können keine Datenpakete für andere Teilnehmer weiterleiten. Sie wählen beim Eintritt in das Netzwerk einen Routerknoten in ihrer Funkreichweite zu ihrem Parentnode (engl. für Elternknoten) und stehen nur mit diesem einem Router in direktem Kontakt. Alle Datentransfers zu und von einem End-Device erfolgen über sein Parentnode. Da End-Devices nicht für die Aufrechterhaltung der Netzwerkinfrastruktur benötigt werden, sind sie die einzigen Netzwerkteilnehmer, die während des Betriebes auf Standby gehen können. Netzwerkteilnehmer mit begrenzter Energiekapazität sollten daher nach Möglichkeit als End-Device eingesetzt werden.

4.2.4. Beispiel eines Meshs

Die folgende Grafik 4.1 veranschaulicht die eben erklärten ZigBee-Teilnehmer. Der Coordinator ist dunkelrot. Alle Routerknoten sind orange und die End-Devices blau. Wie erwähnt, können End-Devices nicht direkt miteinander kommunizieren. Obwohl die Teilnehmer A und B, sowie C und D in Funkreichweite sind, müssen A und B über den türkis markierten Pfad und C und D über den gelb markierten Pfad kommunizieren. Obwohl Knoten R in Funkreichweite von Knoten A ist, erfolgt die Kommunikation zusätzlich über Knoten S. S wurde beim Netzwerkbeitritt von Knoten A zu seinem Parentnode gewählt.

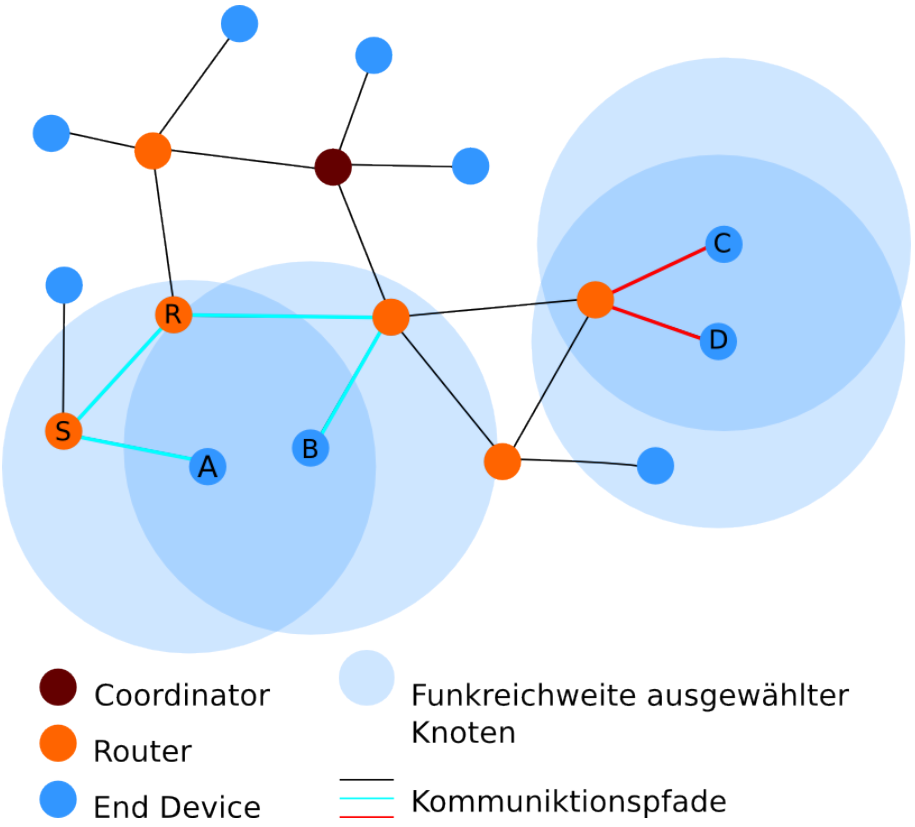

Abbildung 4.1.: Beispielhaftes ZigBee Mesh-Netzwerk

4.3. Marktplatzierung

Neben ZigBee und 6loWPAN gibt es im Bereich der Wireless Sensor Networks (kurz WSN) vor allem proprietäre Lösungen. Zum Teil nutzen diese ebenfalls wie ZigBee und 6loWPAN den IEEE 802.15.4 Übertragungsstandard. Durch einen geringeren Funktionsumfang und eine bessere Anpassung an das Endprodukt benötigen proprietäre Lösungen oftmals weniger Energie. Als Beispiel seien hier Produkte von EnOcean genannt, welche für die Übermittlung von Tastendrücken Energie direkt aus dem Tastendruck gewinnen[2].

Anfangs wurde Heim-, Gebäude- und industrielle Automatisierung von der ZigBee-Alliance als Einsatzgebiet anvisiert. Sieben Jahre nach dem ersten ZigBee Standard hat sich gezeigt, dass ZigBee nicht die hohen Echtzeitanforderungen von vielen industriellen Anwendungen erfüllen kann. Dort kommen nach wie vor proprietäre Lösungen zum Einsatz. In den anderen Bereichen wird ZigBee erfolgreich eingesetzt, wie dies auch anhand der folgenden Projekte gezeigt wird (vgl. [Sikora, 2009, S. 16]).

[2] Siehe http://www.enocean.com/de/ - Letzter URL Abruf: 30.01.2012

5. Design

In diesem Kapitel werden die wichtigsten Komponenten des Netzwerks beschrieben. Ebenso werden alle Entscheidungen getroffen, welche sowohl Auswirkung auf die Hard- und Softwareentwicklung haben. Entscheidungen, welche entweder nur die Hard- oder nur die Softwareentwicklung betreffen, werden in dem jeweiligen Kapitel beschrieben.

5.1. Struktur des Netzwerks

Der Aufbau des Netzwerks wird vom Erfassen eines Messwerts am Sensor bis zur Verfügungstellung des Messwerts erklärt. Es werden die wichtigsten Komponenten erklärt. Einige Rahmenbedingungen werden durch die Infrastruktur in der Wohnung aber auch durch die Wahl von ZigBee als Übertragungsstandard vorgegeben.

Die folgende Grafik 5.1 zeigt den schematischen Aufbau des Sensornetzwerks. Alle Teile der Grafik finden sich in den folgenden Erklärungen wieder.

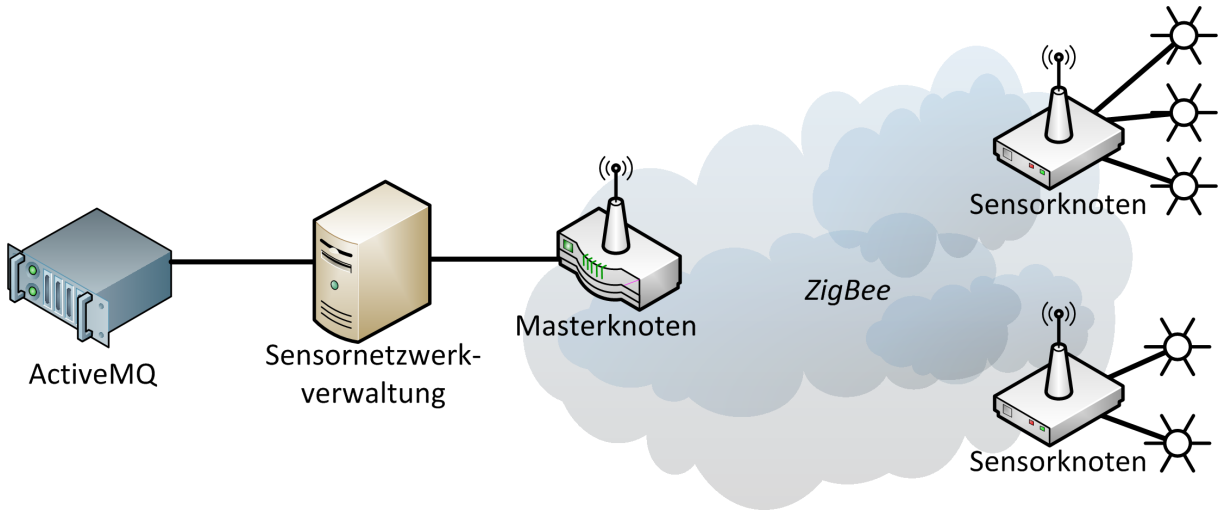

Abbildung 5.1.: Schematischer Aufbau des Sensornetzwerks

5.1.1. Sensor

Ein Sensor dient der Erfassung von physikalischen Werten aus der Wohnung. Beispiele dafür sind Temperatur, Luftfeuchtigkeit, Helligkeit sowie laufendes Wasser, geöffnete Türen und Fenster. Jeder Sensor benötigt eine bestimmte Spannung und einen Strom zum Arbeiten. Außerdem besitzt er ein bestimmtes Protokoll, wie die gemessenen Werte ausgelesen werden können. In den unteren Übertragungsschichten werden dafür meist I^2C, SPI oder eine analoge Spannung verwendet. Die Protokolle der höheren Schichten sind meist proprietär. Die Sensoren werden nach Bedarf ausgewählt und als fertige Bausteine eingekauft.

5.1.2. Sensorknoten

An einem Sensorknoten sind ein oder mehrere Sensoren angeschlossen. Jeder Sensorknoten fragt die an ihm angeschlossenen Sensoren in regelmäßigen Abständen ab. Wie oft ein Sensor abgefragt wird, richtet sich nach der Art des Sensors und dem Bedarf an aktuellen Messwerten. Zum Beispiel ist die Helligkeitserfassung in der Wohnung zur Regelung von Lampen weniger wichtig, wenn der Bewohner nicht zu Hause ist. Ist der eingelesene Messwert in einem geeigneten Format, wird der Messwert über das ZigBee-Netzwerk versendet. Gibt ein Sensor seine Messwerte in einem ungeeigneten Format aus, so rechnet der Sensorknoten den Messwert in ein geeignetes Format um und versendet ihn erst anschließend.

Die meisten Sensorknoten werden als ZigBee End-Devices eingesetzt. Diese wachen nur zum Abfragen der Sensoren und Versenden der Ergebnisse auf. Den Rest der Zeit befinden sie sich in einem Stromsparmodus. Diese Sensorknoten werden gemäß der Anforderung ”Kabellos” aufgebaut und über Batterien mit Strom versorgt.

5.1.2.1. Vergrößerung der Netzwerkausdehnung

Einige Sensorknoten übernehmen im ZigBee-Netzwerk die Funktion eines ZigBee-Routers. So kann die Ausdehnung des Netzwerks größer sein, als die Funkreichweite der Sensorknoten. Für alle Sensorknoten, welche als Router arbeiten, gilt, dass sie die ganze Zeit in Betrieb sein müssen und nicht in stromsparende Schlafmodi wechseln dürfen. Diese Sensorknoten müssen auf Grund des erhöhten Stromverbrauchs extern mit Strom versorgt werden.

Eine Lösung wie bereits am Beispiel der Erzmine (siehe Kap. 3.1) vorgestellt, ist nicht möglich. Es soll eine Vielzahl unterschiedlicher Sensoren integriert werden. Dabei werden die Messwerte der Sensoren in sehr unterschiedlichen Intervallen abgefragt, so dass eine Grundvorraussetzung für die Implementation von schlafenden ZigBee-Routern nicht mehr gegeben ist.

5.1.3. Masterknoten

Der Masterknoten bildet die Schnittstelle zwischen ZigBee-Netzwerk und PC. Da ohne den Masterknoten keine Datenübertragung vom ZigBee-Netzwerk zum PC möglich ist, muss er immer vorhanden sein. Aus diesem Grund übernimmt der Masterknoten die Funktionalität des ZigBee-Coordinators.

5.1.4. Sensornetzwerkverwaltung

Die Hauptaufgabe der PC-Software ist es, die Messwerte aus dem ZigBee-Netzwerk entgegen zu nehmen und über den Active MQ Server zur Verfügung zu stellen. Anwender bzw. Software, welche die Messwerte weiterverarbeiten, sind im Allgemeinen nicht an der Struktur des Sensornetzwerks interessiert. Ein aussagekräftiger Name bzw. eine gute Beschreibung eines Sensors sind an dieser Stelle nützlicher, als zum Beispiel eine ID. Die PC-Software sollte die Abstraktion von IDs auf Namen und Beschreibungen übernehmen. Für eine spätere Auswertung sollen die Messwerte auf dem PC gespeichert werden.

5.2. Soft- und Hardwareplattform

Durch die frühzeitige Entscheidung für eine Soft- und Hardwareplattform kann die Entwicklung beider Teile gleichzeitig ablaufen, ohne dass eine spätere Anpassung von Soft- oder Hardware nötig ist, weil Entscheidungen in einem der beiden Bereiche in dem anderem nicht umgesetzt werden können.

5.2.1. Verfügbare Komponenten

Die wichtigsten ICs des Sensorknotens sind ein Mikrocontroller und ein IEEE 802.15.4 kompatibler Tranceiver. Bei der Auswahl dieser Bausteine sind einige Dinge zu beachten.

Für die Kombination von beiden Bausteinen muss ein ZigBee-Stack verfügbar sein. Die Portierung eines ZigBee-Stacks von einem Mikrocontroller zu einem anderen würde viel Zeit in Anspruch nehmen und ist teilweise gar nicht möglich, da Hersteller oftmals nur compilierte Bibliotheken zur Verfügung stellen. Der Mikrocontroller muss genügend Speicherplatz für den Stack und die eigene Anwendung zur Verfügung stellen.

Die Rechenleistung des verwendeten Mikrocontrollers muss nicht sehr hoch sein. In Projekt 2 wurde ein 8 Bit Mikrocontroller mit 8 MHz verwendet. Dieser Controller arbeitet einen großen Teil der Befehl in einem Takt ab, womit im Idealfall bis zu 8 Millionen Befehle pro Sekunde ausgeführt werden können (vgl. [Atmel Corporation, 2011b, S. 1]). Diese Rechenleistung ist ausreichend, um den Messwert eines Sensors einzulesen, den Messwert zu versenden und zu berechnen, wann der nächste Sensor eingelesen werden

muss. Lediglich Floating Point Operationen sind zu vermeiden, da dieser Mikrocontroller keine Floating Point Unit besitzt.

Beide ICs dürfen nur wenig Strom verbrauchen, um die Anforderung des geringen Wartungsaufwandes zu erfüllen.

Bei der Implementation der Software ist eine hohe Verbreitung der Mikrocontrollerarchitektur von Vorteil. Je höher die Verbreitung eines Controllers, desto größer ist die Community, die bei Problemen während der Entwicklung helfen kann.

Bei der Recherche zu aktuellen ZigBee-Implementationen kristallisieren sich schnell zwei Hersteller heraus, welche weit verbreitete ZigBee-Stacks und die dazu passende Hardware anbieten.

5.2.1.1. Z-Stack von Texas Instruments

Texas Instrument bietet einen ZigBee Pro Stack namens ”Z-Stack” in Kombination mit seinen MSP430 Prozessoren und dem CC2520 RF-Tranceiver an[1].

Die MSP430 sind 16 Bit Prozessoren mit einem sehr geringen Stromverbrauch. Sie benötigen je nach Modell unter 2 mW (600μA @ 3,3 Volt) pro MHz. Es sind Modelle mit bis zu 128 kB Flash und 8 kB RAM verfügbar.

Der CC2520 ist ein IEEE 802.15.4 kompatibler Tranceiver Baustein. Er benötigt beim Senden mit 0 dBm knapp 86 mW (26 mA @ 3,3 Volt) und zum Empfangen ca. 61 mW (18.5 mA @ 3,3 Volt).

Beide zusammen benötigen während des Betriebs maximal 88 mW. Werden alle Verluste und sonstige Verbraucher außer Acht gelassen, könnte mit diesem Controller und Tranceiver und einer Batterie mit 3000 mWh[2] über 34 h am Stück gesendet werden.

5.2.1.2. BitCloud von Atmel

Atmel bietet einen ZigBee Pro Stack namen ”BitCloud” an (siehe Atmel Corporation [2011e]). Es werden eine Reihe von Atmel AVR 8 Bit Prozessoren und einige ARM7 und ARM Cortex Prozessoren von Atmel unterstützt. Derzeit werden drei IEEE 802.15.4 kompatible Tranceiver angeboten[3]. Als Besonderheit gibt es einen Chip, bei dem ein AVR und Tranceiver kombiniert sind. Dieser Chip wurde bereits in Projekt 2 [Pautz, 2011b, S. 19] verwendet. Von der Firma MeshNetics gibt es Module mit dem Namen ZigBit (Datenblatt siehe MeshNetics [2007]), welche einen Tranceiver, einen AVR und eine Antenne vereinen.

[1] http://focus.ti.com/mcu/docs/mcuorphan.tsp?contentId=24576&DCMP=MSP430Wireless&HQS=Other+EM+430wireless Abruf: 04.11.11

[2] Dies entspricht in etwa einer Alkaline Mignon Batterie bei geringer Entladung (siehe [Energizer Holdings Inc., 2008, S. 1])

[3] http://www.atmel.com/dyn/products/devices.asp?category_id=163&family_id=676&subfamily_id=1953 Abruf 04.11.11

Die AVR Prozessoren sind besonders in Deutschland weit verbreitet. Sie benötigen zum Abarbeiten der meisten Befehle nur einen Takt. Ihre Stromaufnahme liegt je nach Modell bei etwa 2,6 mW (800μA @ 3,3 Volt). Es stehen Modelle mit bis 256 kB Flash und 16 kB RAM zur Verfügung.

Die Stromaufnahme der Tranceiver liegt bei 41,3 mW (12,5 mA@ 3,3 Volt) beim Empfangen und beim Senden mit 0 dBm bei ca. 33 mW (10 mA@3,3 Volt).

Beide zusammen benötigen während des Beriebes maximal 43,9 mW. Werden alle Verluste und sonstige Verbraucher außer acht gelassen, könnte mit diesem Controller und Tranceiver und einer Batterie mit 3000 mWh[4] über 68 h am Stück empfangen werden.

Die ARM Prozessoren werden in Verbindung mit dem ZigBee Stack kaum verwendet. Dies kann am höherem Stromverbrauch mit knapp 1 mA (@1,8Volt) pro MHz und den zwei benötigten Spannungen für Peripherie und CPU-Kern liegen. Auch für diese Masterarbeit werden die ARM Prozessoren nicht weiter betrachtet.

Nicht zuletzt wurde bereits in Projekt 2 der BitCloud-Stack verwendet. Bei Verwendung von BitCloud entfällt die Einarbeitungszeit in einen neuen Stack, ebenso kann ein Teil des Quellcodes aus dem Projekt wiederverwendet werden.

5.2.1.3. Entscheidung für BitCloud

In dieser Masterarbeit wird mit BitCloud von Atmel gearbeitet. Bereits in Projekt 2 wurden mit diesem Stack erste Erfahrungen gesammelt. Es wurden keine negativen Eigenschaften gefunden, welche die Verwendung des Stacks für die Masterarbeit ausschließen. Die AVR Prozessoren benötigen geringfügig mehr Strom als die MSP430 Prozessoren, doch wird dieser Nachteil durch den geringeren Stromverbrauch der Tranceiver wieder ausgeglichen.

Es gibt von Atmel zwei IEEE 802.15.4 Tranceiver für 2,4 GHz. Außerdem gibt es den ATMega128RFA1 Prozessor mit integriertem Tranceiver. Alle drei Bausteine sind nur in einem Gehäuse mit einem sehr kleinen Pad-Abstand von 0,5 mm (Pad-Mitte zu Pad-Mitte) verfügbar. Bei allen drei Bausteinen stehen die Pads nicht hervor, sondern schließen mit dem Rand des Gehäuses ab. Zudem haben alle Bausteine auf der Unterseite eine große Massefläche, welche mit der Leiterplatte verbunden werden muss. Aus diesen drei Gründen sind die Bausteine nicht von Hand lötbar, sondern müssen per Reflow Verfahren verlötet werden. Das folgende Foto 5.2 zeigt den ATMega128RFA1 Chip von der Unterseite. Die helle Fläche in der Mitte ist die oben erwähnte Massefläche, welche mit der Platine verbunden werden muss. Die silbernen Kontakte am Rand des ICs sind lediglich 0,3 mm breit und sind in einem Abstand von 0,2 mm angeordnet.

[4] Dies entspricht in etwa einer Alkaline Mignon Batterie bei geringer Entladung (siehe [Energizer Holdings Inc., 2008, S. 1])

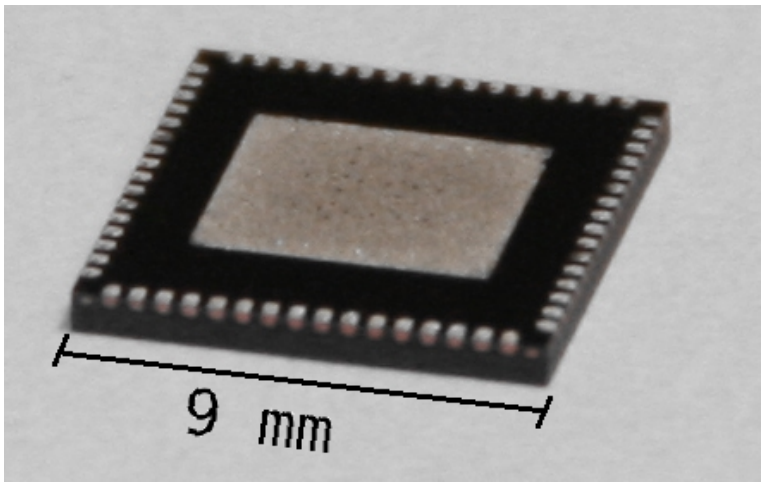

Abbildung 5.2.: Ansicht eines ATmega128RFA1 von unten

Der ATMega128RFA1 wäre sehr nützlich, da durch die Single-Chip-Lösung der Platzbedarf am geringsten ist. Ein Reflowofen steht zur Verfügung. Ob die kleinen Strukturen verarbeitet werden können, ist unbekannt. Alternativ gibt es von Atmel fertige Module mit dem Namen ZigBit. Diese Module enthalten einen ATMega1281 Prozessor, ein AT86RF230 Tranceiver und zwei Antennen. Diese Module können von Hand verarbeitet werden. Sie sollen als Alternative verwendet werden, wenn die Verarbeitung des ATMega128RFA1 nicht möglich ist. Der Wechsel zwischen den Modulen und der Single-Chip Lösung erfordert keine Softwareanpassung. Der AVR-Kern des ATMega128RFA1 basiert auf einem ATMega1281, wie er in dem Modul verbaut ist (vgl. [Atmel Corporation, 2011a, S. 3]). Die Ansteuerung des RF-Tranceivers übernimmt der BitCloud-Stack. Dies ist für den Entwickler transparent.

Sowohl der ATMega128RFA1, als auch die ZigBit Module können bei 3,3 Volt verwendet werden. Die Anforderung an die IO-Spannung (siehe ”Strom- und Spannungsversorgung der Sensoren”) kann somit erfüllt werden. Ebenso bieten beide Lösungen die wichtigsten Schnittstellen wie GPIO, I^2C, SPI, ADC und UART bereits ohne zusätzliche ICs an. Somit ist auch die Anforderung ”Unterstützte Schnittstellen” erfüllt.

5.2.2. Kommunikation zwischen ZigBee und PC

Es gibt derzeit drei weit verbreitete und geeignete Standards für die Datenübertragung zwischen einem Gerät und einem PC. Da die Art der Schnittstelle sowohl die Software- als auch Hardwareentwicklung beeinflusst, muss diese Entscheidung sehr früh im Entwicklungsprozess gefällt werden.

Serielle Schnittstelle Die älteste und einfachste Schnittstelle ist die serielle Schnittstelle RS232. Sowohl auf dem PC als auch auf dem Mikrocontroller ist die Verwendung sehr einfach. Auf keiner Seite muss ein Treiber entwickelt werden. Lediglich eine Pegelanpassung vom Mikrocontroller (3,3 Volt) an den PC (-12 bis +12 Volt) ist nötig. Ein Nachteil bei der seriellen Schnittstelle ist, dass die Übertragung nicht gesichert

ist. Kommt es zu einem Fehler, kann dieser durch die Verwendung von Paritybits zwar erkannt werden, das Verhalten bei einem Fehler muss der Benutzer der Schnittstelle jedoch selbst implementieren.

Die serielle Schnittstelle ist auf modernen PCs häufig nicht mehr vorhanden, so dass ein zusätzlicher Adapter von RS232 auf USB nötig ist. Der Treiber des USB-RS232-Adapters richtet auf dem PC eine virtuelle serielle Schnittstelle ein. Diese kann dann genauso wie jede physikalische serielle Schnittstelle auf einem PC angeprochen werden.

IP basierte Lösung Die flexibelste Schnittstelle ist TCP/IP bzw. UDP/IP über Ethernet. Durch die Verwendung von bis zu 100 Meter langen Kabeln und Ethernet-Switches kann der verarbeitende PC weit entfernt sein. Theoretisch ist sogar eine Steuerung über das Internet möglich. TCP/IP bietet eine gesicherte Übertragung mit automatischer Sendewiederholung. UDP/IP ist insgesamt einfach aufgebaut und bietet dies nicht.

Damit der AVR über das Ethernet kommunizieren kann, benötigt er eine Ethernetschnittstelle. Da weder der ATMega128RFA1 noch das ZigBit Modil über ein Ethernetinterface verfügen, ist ein zusätzlicher Ethernetcontroller wie zum Beispiel ein ENC28J60[5] nötig. Die Verwendung von Ethernet ist auf der Seite des PCs sehr einfach. Auf der Seite des AVR wird eine IP-Implementation benötigt, welche den Ethernetcontroller ansteuern kann.

USB USB-Geräte sind die dritte und aufwendigste mögliche Schnittstelle. Hierfür müssen auf beiden seiten Treiber entwickelt werden. Genauso wie TCP/IP bietet USB eine gesicherte Datenübertragung. Der ATMega128RFA1 und das ZigBit Modul verfügen über kein USB-Interface, so dass ein externer Baustein verwendet werden muss.

Entscheidung Es wird die Entscheidung getroffen, eine möglichst einfache Implementation zu erreichen. Dafür wird ein Zwischenweg aus USB und RS232 gegangen. Der bereits erwähnte Adapter von RS232 auf USB wird direkt auf die Platine des Masterknotens integriert. Somit kann der Masterknoten an jeden modernen PC angeschlossen werden und die einfache Kommunikation auf Basis von RS232 bleibt erhalten. Das serielle Protokoll wird durch die USB-Verbindung getunnelt. Bei einem Übertragungsfehler übernimmt USB automatisch die Neuübertragung. Die Übertragung ist somit gesichert und eine Datenwiederholung bei Paketverlust muss nicht mehr auf der Benutzerebene erfolgen.

[5] Produktdetails unter: http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en022889 Abruf 03.12.11

5.2.3. Stromversorgung der Sensorknoten

Die Art der Stromversorgung kann einen Einfluss auf die Software haben. Ein Sensorknoten, welcher als ZigBee-Router arbeitet, darf nicht in Energiesparmodi wechseln. Durch eine externe Spannungsversorgung steht ihm jedoch jederzeit genügend Energie zur Verfügung. Ein Sensorknoten ohne externe Energieversorgung muss möglichst oft einen energiesparenden Modus einnehmen.

Da die Sensorknoten zum Teil kabellos betrieben werden, benötigen sie eine eigene Stromversorgung. In den Anforderungen werden nur Batterien als Energiequelle angegeben. Im Folgenden soll dargelegt werden, warum keine andere Energiequelle wie Akkus oder Energy Harvesting Technologien in Betracht kommen. Solarzellen als Energy Harvesting Technologie können nicht eingesetzt werden, da die Module verdeckt angebracht werden sollen und demnach kein Licht auf das Solarpanel fallen würde (siehe Unauffälligkeit).

5.2.3.1. Akkumulatoren

Akkumulatoren oder kurz Akkus kommen als Energiequelle nicht Betracht. Akkus haben eine geringere Kapazität als vergleichbare Primärzellen[6]. Zu dem haben sie eine sehr hohe Selbstentladung. Im Handbuch über Nickel Metal Hydride Akkumulatoren (kurz NiMH Akkus) von Energizer steht dazu "NiMH batteries will typically retain appproximately 50% to 80% of their capacity after 6 months of storrage."[Energizer Holdings Inc., 2010, S. 12]. Umgekehrt bedeutet dies, dass sich ein NiMH Akku innerhalb eines Jahres durch seine Selbstentladung entladen kann. Auch bei auf Lithium basierenden Akkus ist die Selbstentladung signifikant vorhanden und verhindert die Verwendung in Schaltungen mit Laufzeiten von mehreren Monaten.

5.2.3.2. Energy harvesting

Energy harvesting (engl. für Energieernte) bezeichnet die Energiegewinnung aus der Umgebung. Mögliche Quellen für eine Energiegewinnung bieten zum Beispiel Licht, Luftströme, Bewegungen, Temperaturdifferenzen und Funkwellen. Eine Vielzahl an aktuellen Berichten zeigt, dass auf diesem Gebiet aktiv geforscht wird. Mit dem erfolgreichem Einsatz von Energy harvesting würde der Batteriewechsel in vielen Gegenständen des Alltags der Vergangenheit angehören. Fernbedienungen von Fernsehgeräten und Radios, Außensensoren von Wetterstationen, aber auch neu hinzukommendene Geräte aus dem Bereich des Ambient Intelligence könnten ohne Batterien auskommen.

Bisher ist die Technologie jedoch praktisch kaum einsetzbar. Forscher aus Frankreich (siehe Lhermet u. a. [2008]) haben eine Schaltung entwickelt, die aus Funkwellen der Umgebung und Temperaturunterschieden einen Akku laden können. Der gesamte Aufbau ist 30 mm^2 groß, hinzu kommt eine Antenne, welche zum Einfangen der

[6] nicht aufladbare Batterien

Funkwellen dient. Der verwendete Akku hat abhängig vom Ladestand eine Spannung zwischen 1,6 und 2,8 Volt. Er besitzt eine Kapazität von 30 μAh/cm^2. Der Stromverbrauch der Schaltung liegt bei 70 μA. In dem Aufbau der Forscher wurde der Akku unter Laborbedingungen mit 27 μA geladen. Dies macht eine maximale entnehmbare Ladung von 75,6 μW (2,8 V * 27 μA).

Eine andere Forschungsgruppe am Forschungsinstitute IMEC aus Eindhoven, Niederlande hat Energy harvesting mit einem Wireless Body Area Network (WBAN) kombiniert (Huang u. a. [2010b]). Ein solches WBAN ist für die Erfassung von Bewegungen und Vitalfunktionen eines menschlichen Körpers gedacht. Die Einsatzgebiete erstrecken sich von medizinischen Geräten, über Sportgeräte bis hin zu Steuerungen neuer Spielkonsolen. Im ersten Schritt haben die Forscher den Stromverbrauch von existierenden industriellen WSN-Funklösungen untersucht. In ihrem Beispielszenario entfallen fast drei Viertel des Stromverbrauchs auf Funkkommunikation. Insgesamt verbraucht die stromsparendeste untersuchte industrielle Lösung 1,3 mW. Durch die Entwicklung eines Ultra Low Power Radio Tranceivers mit 14 Metern Reichweite ist es ihnen unter optimalen Bedingungen und der Integrierung aller externen Komponenten in einen einzelnen ASIC gelungen, den Stromverbrauch gegenüber bisherigen industriellen Lösungen auf 360 μW (28%) zu senken.

Beide Forschungsarbeiten zeigen, dass es möglich ist, der Umwelt Energie zu entnehmen bzw. Schaltungen zu entwerfen, die mit sehr geringen Energiemengen auskommen. Doch es ist auch zu sehen, dass die entnehmbare Energie im Verhältnis zum Stromverbrauch aktueller Schaltungen sehr gering ist. Die erste Energy harvesting Schaltung müsste um ein Vielfaches größer aufgebaut werden, um den Stromverbrauch der zweiten Schaltung zu decken. Wenn der Stromverbrauch von integrierten Schaltkreisen weiter sinkt und die Effizienz bei Energy harvesting steigt, könnte es in einigen Jahren möglich sein, Energy harvesting Produktiv zu verwenden.

6. Hardwareentwicklung

Die Hardwareentwicklung beginnt mit dem Erstellen der Kriterien und der anschließenden Auswahl der wichtigsten Hardwarekomponenten.

6.1. Masterknoten

Der als Gateway arbeitende Masterknoten übernimmt die Kommunikation zwischen PC und ZigBee-Netzwerk (siehe 5.1). Da er immer vorhanden sein muss, wird dieser Masterknoten neben seiner Funktionalität als Gateway die Funktion als Coordinator übernehmen.

Der Masterknoten soll als Test dienen, ob die kleinen Strukturen des ATMega128RFA1 zuverlässig verarbeitet werden können. Können die kleinen Strukturen nicht verarbeitet werden, so sollen die Sensorknoten mittels der ZigBit-Module erstellt werden. Einer der Sensorknoten wird dann so erweitert, dass er die Funktion des Masterknoten übernehmen kann.

6.1.1. Hardwaredesign

In diesem Kapitel werden die Vorgaben beschrieben, welche zum Erstellen der Platine eingehalten werden müssen. Der daraus entstandene Schaltplan, das Layout und eine Bauteilliste können im Anhang eingesehen werden.

Bei dem Masterknoten benötigen folgende Bauteile eine besondere Aufmerksamkeit:

- ATMega128RFA1
- Antenne
- USB-Seriell Wandler
- Spannungsversorgung

Bei allen weiteren Bauteile handelt es sich um Kondensatoren, Widerstände und LEDs, welche keine besondere Beschaltung benötigen.

Mikrocontroller Der wichtigste Baustein des Masterknotens ist der ATMega128RFA1 Prozessor. Für die Verwendung im ZigBit-Netzwerk muss dieser mit einem 16 MHz Quarz ausgestattet werden [Atmel Corporation, 2011a, S. 81]. Für die Verwendung von BitCloud wird ein zusätzlicher Taktgeber in Form eines 32,768 KHz Quarz (siehe [Atmel Corporation, 2011d, S. 7]) benötigt. Beide Quarze werden entsprechend der Beschreibung der Applikation Note beschaltet [Atmel Corporation, 2011d, S. 12ff].

Antenne Zum Senden und Empfangen wird eine 2,4 GHz Antenne benötigt. Diese muss möglichst klein gehalten werden. Stabantennen, wie sie bei WLAN-Routern verwendet werden, scheiden auf Grund der Größe aus. Zur Verfügung stehen PCB-Antennen, welche direkt durch Leiterbahnen auf der Platine erzeugt werden und Chipantennen, welche wie andere SMD-Bauteile auf die Platine gelötet werden. PCB-Antennen müssen mit den Eigenschaften der Platine abgestimmt werden. Chipantennen sind resistenter gegen äußere Einflüsse, weshalb diese bevorzugt werden. Da die Chipantenne eine unsymmetrische Antenne ist und der AVR einen symmetrischen Antennenausgang besitzt, muss eine Anpassung vorgenommen werden. Für diese Anpassung wird ein Balun verwendet, welches auf die Frequenz und die Impedanz von Antenne und AVR abgestimmt ist. Der Leiterbahn von der Antenne zum Balun muss eine besondere Aufmerksamkeit gewidmet werden, da es sich mit 2,4 GHz um einen Hochfrequenzpfad handelt. Die Leiterbahn muss die gleiche Impedanz wie die Antenne aufweisen. Die Antennenimpedanz beträgt 50 Ohm. Die Impedanz einer Leiterbahn ist vor allem von der Dielektrizitätskonstante des Platinenmaterial, der Stärke der Platine, der Höhe der Kupferauflage und der Breite der Leiterbahn abhängig. Die Dielektrizitätskonstante gibt dabei an, wie gut sich elektromagnetische Felder in dem Material ausbreiten können.

Mit dem Programm AppCAD 3.0.2 von Agilent Technologies wurden die entsprechenden Werte ermittelt[1]. Die Grafik 6.1 zeigt die errechneten Werte. Die Frequenz wurde mit 2,45 GHz angegeben. Als Basismaterial wird ein Gemisch aus Epoxydharz und Glasgewebe verwendet, welches auch als FR-4 bezeichnet wird. Damit die Leiterbahn nicht zu breit wird (Buchstabe W in der Grafik), wird für die Platine des Masterknotens eine Stärke von 1,0 mm (Buchstabe H in der Grafik) anstelle der weit verbreiteten 1,6 mm verwendet. Die Stärke der Kupferauflage beträgt 35 μm (Buchstabe T in der Grafik). Das errechnete Ergebnis von 49,97 Ohm (Buchstabe Z0) liegt sehr Nahe an der Vorgabe von 50 Ohm.

[1] Webseite: http://www.hp.woodshot.com/ Abruf: 08.10.2011

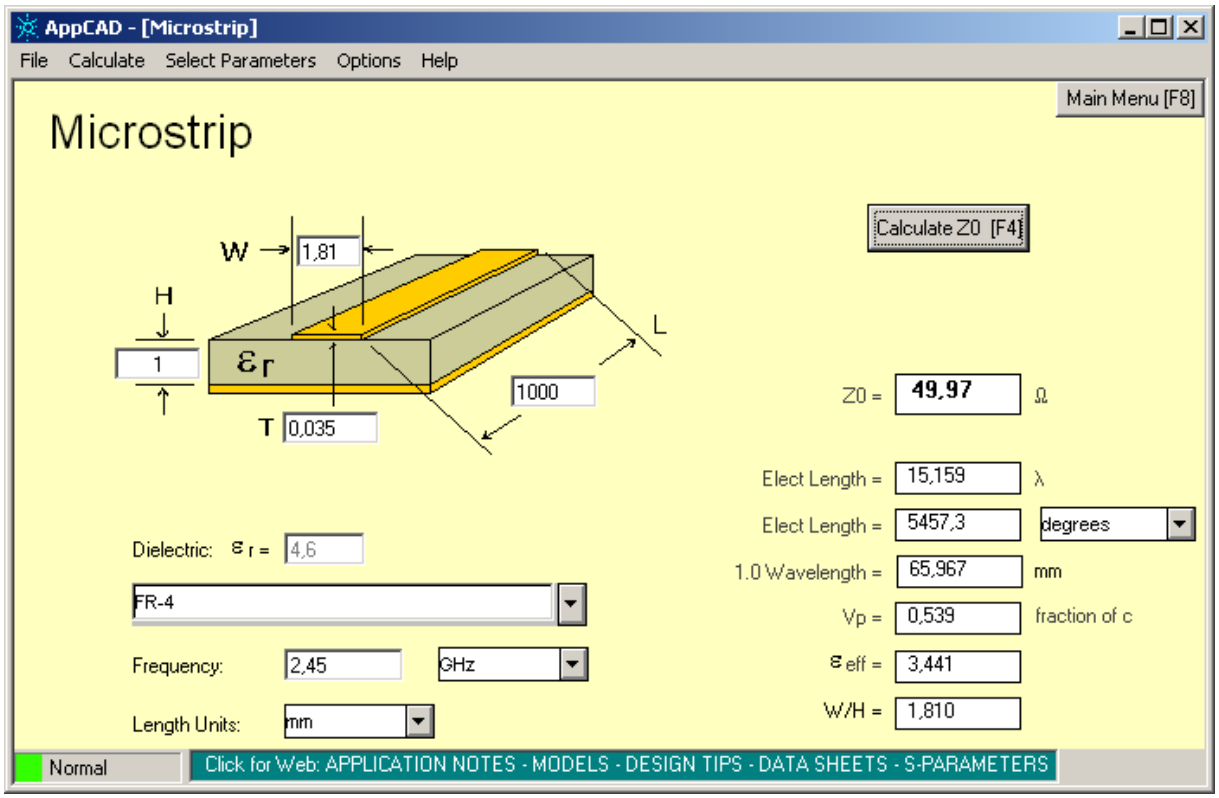

Abbildung 6.1.: Berechnung der Antennenleiterbahn

Da die Antenne unsymmetrisch ist, benötigt sie einen Gegenpol. Dieser Gegenpol wird durch die Massefläche der Platine bereitgestellt. Die Unterseite der Platine bildet dabei eine möglichst durchgehende Massefläche. Leiterbahnen werden, nur wenn nicht anders möglich dort verlegt und dann so kurz wie möglich gehalten. Für eine gute Abschirmung gegen äußere Einstrahlungen und gegen Abstrahlung nach außen wird die gesamte Platine von Durchkontaktierungen, sogenannten Vias, umrandet. Ebenso werden die untere und obere Massefläche in regelmäßigen Abständen mit Vias verbunden. Die Empfehlungen entstammen der Application Note [Atmel Corporation, 2011d, S. 12ff].

USB-Seriell-Wandler Für die Kommunikation zwischen AVR und PC wird ein USB zu RS232 Konverter eingesetzt. Hier wird ein FT232RL[2] eingesetzt. Dieser Chip benötigt, abgesehen von einigen Kondensatoren, keine weitere Beschaltung. An dafür vorgesehen Anschlüsse des ICs werden zwei LEDs angeschlossen, welche bei Sende- bzw. Empfangsaktivitäten blinken. Diese LEDs bieten eine gute Möglichkeit, Fehler bei der Kommunikation zu erkennen.

Spannungsregler Für die Spannungsversorgung wird ein LF33C (Datenblatt siehe STMicroelectronics [2010]) 3,3 Volt Spannungsregler verwendet. Bei diesem Spannungsregler muss darauf geachtet werden, dass sich am Eingang und Ausgang Kondensatoren

[2]Datenblatt und weitere Informationen: http://www.ftdichip.com/Products/ICs/FT232R.htm Abruf: 08.10.2011

mit genügend großer Kapazität befinden. Der Ausgangskondensator darf dabei keinen zu kleinen Innenwiderstand haben, da er sonst zu schwingen anfängt. Elektrolytkondensatoren sind für diese Aufgabe gut geeignet.

Sowohl der Schaltplan als auch das Layout befinden sich im Anhang B.1.

6.1.2. Fertiggestellte Platine

Die entworfene Platine wurde von einem Leiterplattenhersteller gefertigt. Die Bauteile wurden dann an der Hochschule mittels Reflowverfahren selbst verlötet. Das Foto 6.2 zeigt eine der drei gefertigten Platinen. Das weiße stabförmige Bauteil ist die Antenne. Der rechte Chip ist der ATMega128RFA1. Der Linke Chip ist der FT232RL, welcher sich direkt neben dem USB-Stecker befindet. Neben dem Spannungsregler im oberen Teil und der Außenbeschaltung der ICs sind die zahlreichen Vias zu sehen, welche die Platine umranden und in regelmäßigen Abständen auf der Platine vorhanden sind.

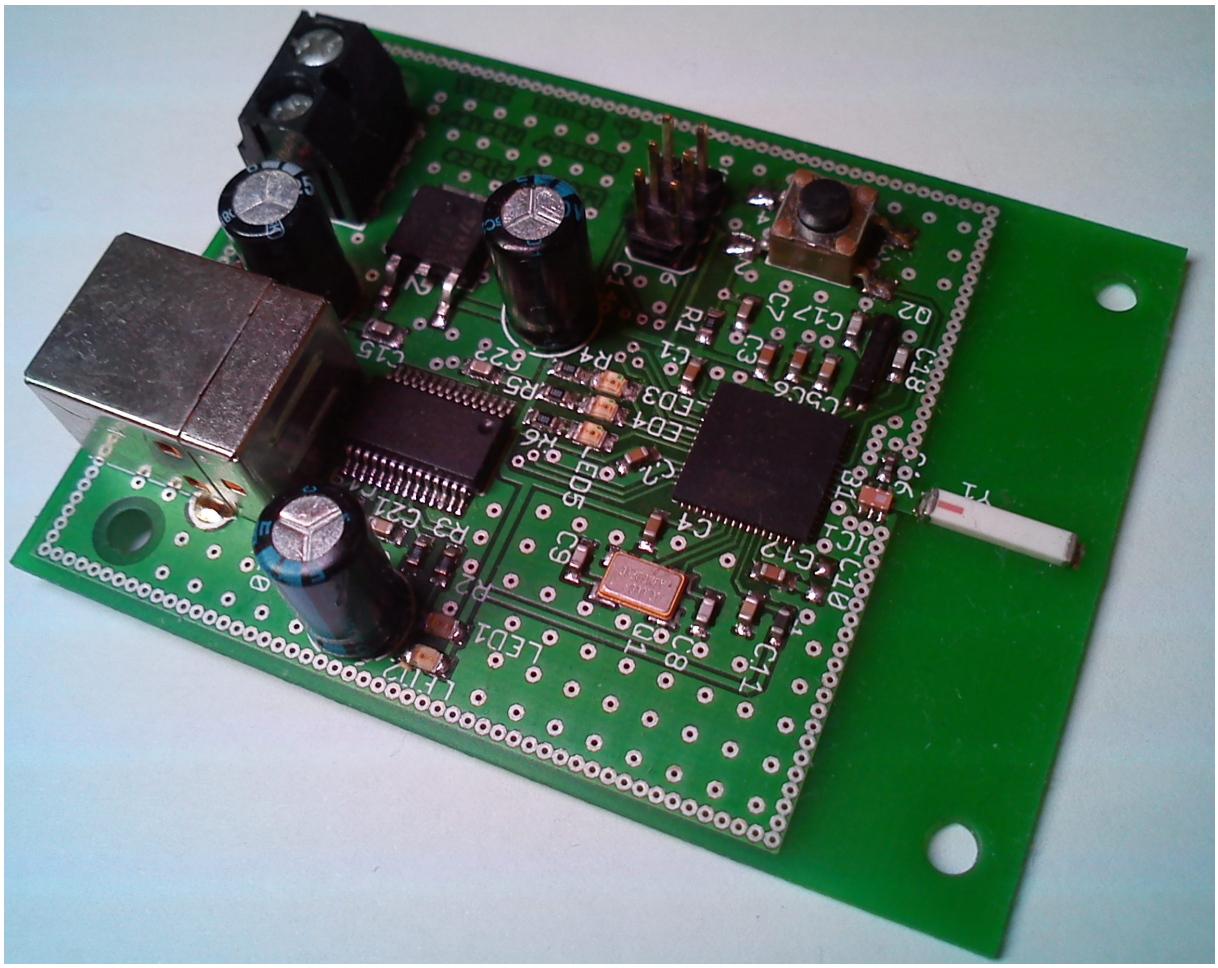

Abbildung 6.2.: Fertiggestellter Masterknoten

Bei der dritten Platine hat sich die Pastenschablone bereits mit Lot so stark zugesetzt, dass es während des Backens der Platine zu Kurzschlüssen kam. Die Abbildung 6.3 zeigt links den AVR und rechts den FT232RL. Bei beiden ICs sind einige Pins kurzgeschlossen. Die Kurzschlüsse ließen sich mit einem Lötkolben entfernen.

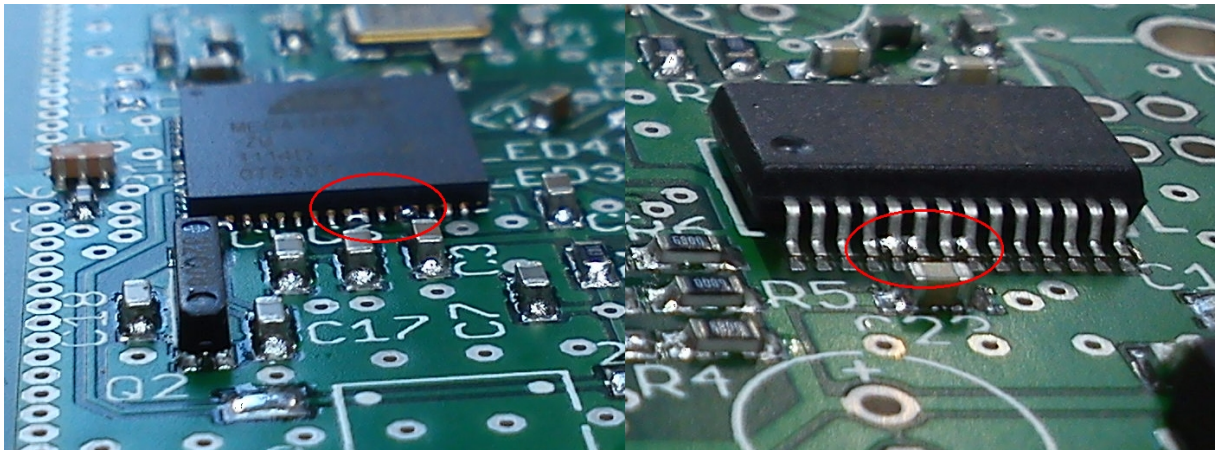

Abbildung 6.3.: Kurzschlüsse bei der Fertigung

Da die Kurzschlüsse bereits bei der dritten Platine aufgetreten sind und von den Sensorknoten 20 Stück gefertigt werden sollen, werden für die Sensorknoten die ZigBit-Module verwendet. Die Pastenschablone hat sich zu schnell zugesetzt und die kleinen Öffnungen lassen sich nur schwer reinigen, so dass sehr viele Kurzschlüsse während der Fertigung zu erwarten wären.

6.2. Sensorknoten

Auch für die Sensorknoten gibt es einige Punkte, welche beim Platinendesign beachtet werden müssen. Der wichtigste Aspekt ist die Spannungsversorgung. Möglichkeiten zur effizienten Spannungsversorgung werden folgend diskutiert. Anschließend wird auf die weiteren zu beachtenden Aspekte beim Layout der Platine eingegangen.

6.2.1. Effiziente Spannungsversorgung

Wie in Kapitel 5.2.3 gezeigt, kommen für alle Sensorknoten ohne externe Energiequelle nur Batterien in Frage. Im folgenden werden Möglichkeiten der effizienten Nutzung der Batterien diskutiert. Zunächst werden die Kriterien aufgestellt und anschließend werden die unterschiedlichen Möglichkeiten aufgezeigt.

Im einzelnen werden folgende Methoden der Spannungsversorgung analysiert:

- Direkte Versorgung mit Batterien
- Spannungsregler
- Spannungswandler

6.2.1.1. Kriterien einer effizienten Spannungsversorgung

Die Wahl der passenden Spannungsversorgung findet anhand der folgenden Kriterien statt. Die folgende nach Priorität sortierte Liste enthält die Kriterien, welche bei der Wahl der Spannungsversorgung helfen sollen:

- Einhaltung der Spannung und Lieferung des Strom
- Effizienz
- Kosten der Lösung

Einhaltung der Spannung und Lieferung des Strom Laut Anforderung "Strom- und Spannungsversorgung der Sensoren" müssen die Sensoren mit 3,3 oder 5 Volt versorgt werden. Da das ZigBit-Modul mit 1,8 bis 3,6 Volt versorgt werden kann (siehe [MeshNetics, 2007, S. 5], bleibt als Schnittmenge die Versorgung mit 3,3 Volt. Nahezu alle Schaltungen, welche mit 3,3 Volt arbeiten, können in einem Bereich von 2,7 bis 3,6 Volt betrieben werden. Eine Spannung in diesem Bereich ist somit zulässig.

Die Stromversorgung der Sensorknoten muss mindestens 40 mA bereitstellen können. Dabei entfallen knapp 20 mA auf das ZigBit-Modul (siehe [MeshNetics, 2007, S. 5]) und 20 mA müssen für die Sensoren bereitgestellt werden können (siehe Anforderungen 2.8). Dieses Kriterium muss von der zu wählenden Spannungsversorgung erfüllt werden, da die Sensorknoten sonst nicht korrekt arbeiten. Weil der benötigte Strom mit 40 mA jedoch recht gering ist, wird dieses Kriterium von jeder hier vorgestellten Lösung erfüllt.

Effizienz Abgesehen davon, dass der benötigte Strom zur Verfügung gestellt wird, soll die Ausnutzung der Batterien möglichst effizient erfolgen. Diese Effizienz lässt sich in zwei Punkte unterteilen. Zum Ersten sollten nur leere Batterien gewechselt werden müssen. Je mehr Restkapazität in den Batterien verbleibt, desto ineffizienter ist die Lösung. Zum Zweiten sollte der Eigenverbrauch der Lösung sehr gering sein. Da der endgültige Stromverbrauch noch nicht feststeht, wird im Folgenden angegeben, wie viel Prozent der zugeführten Energie von der Schaltung zur Spannungsversorgung selbst verbraucht wird.

Kosten der Lösung Es handelt sich bei dem Sensornetzwerk um keine Serienproduktion. Die Kosten stehen nicht im Vordergrund. Sie werden erwähnt, aber für die Entscheidungsfindung nur herangezogen, wenn zwei gleichwertige oder nahezu gleichwertige Lösungen mit einem großen Preisunterschied vorhanden sind.

6.2.1.2. Direkte Versorgung mit Batterien

Am einfachsten ist es, die Batterien direkt mit der Schaltung zu verbinden. Zwei in Reihe geschaltete 1,5 Volt Zellen ergeben im vollen Zustand 3 Volt. Batterien halten die Spannung jedoch nicht bis zum Ende. Im Laufe der Zeit fällt die Spannung. Alkaline Batterien gelten bei 0,8 bis 0,9 Volt als Entladen (siehe zum Beispiel Energizer AA Zellen [Energizer Holdings Inc., 2008, S.1]). Zwei in Reihe geschaltete Alkaline Batterien haben gegen Ende ihrer Lebenszeit somit noch 1,6 bis 1,8 Volt. Diese Spannung ist zu niedrig um die Anforderung 2.8 ”Strom- und Spannungsversorgung der Sensoren” zu erfüllen. Die Batterien nur bis zu einer Restspannung von 2,7 Volt zu verwenden ist indiskutabel, Bei dieser Spannung sind die Batterien noch nahezu voll. Die direkte Verwendung von Batterien ist nicht möglich.

6.2.1.3. Spannngsregler

Die zweite Möglichkeit ist die Verwendung von sogenannten Spannungsreglern. Sie regeln eine höhere Eingangsspannung auf eine niedrigere Ausgangsspannung herunter. Die Eingangsspannung eines Spannungsreglers darf sich im Laufe der Zeit verändern. Die Ausgangsspannung ist fest vom IC vorgegeben oder wird durch eine externe Beschaltung des Bauteils definiert. Ein großer Vorteil von Spannungsreglern ist es, dass sie keine elektromagnetischen Störungen an die Umwelt abgeben. Ebenso haben sie eine sehr einfache Beschaltung. Die meisten Spannungsregler benötigen für einen stabilen Betrieb nur zwei Kondensatoren. Bei Serienfertigungen sind Spannungsregler auf Grund der geringen Kosten zwischen 20 Cent und 2 Euro sehr interessant. Abbildung 6.4 zeigt die Beschaltung eines nicht näher definierten Spannungsreglers, dessen Spannung fix ist und nicht durch die äußere Beschaltung festgelegt wird.

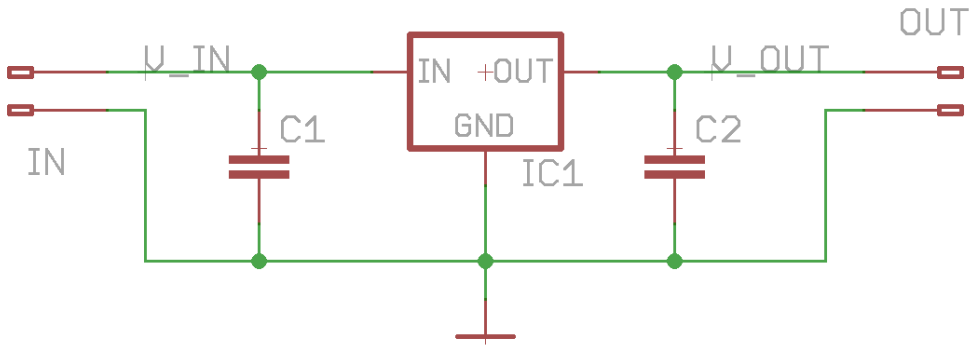

Abbildung 6.4.: Typische Beschaltung eines Spannungsreglers

Spannungsregler haben abhängig von der Spannungsdifferenz zwischen Ein- und Ausgangsspannung, sowie dem entnommenem Strom der Schaltung eine Verlustleistung. Diese Verlustleistung wird in Wärme umgewandelt. Die Effizienz eines Spannungsregler hängt somit von der Differenz zwischen Ein- und Ausgangsspannung ab. Die Eingangsspannung muss immer etwas höher sein als die angestrebte Ausgangsspannung. Diese Spannung wird auch ”Dropout Voltage” (engl. Abfallspannung) genannt. Ist die Spannungsdifferenz nicht mehr gegeben, sinkt die Ausgangsspannung. Moderne Spannungsregler erreichen abhängig von der Stromaufnahme der angeschlossenen Schaltung eine Dropout Voltage von 0,1 bis 0,3 Volt.

Wie jeder integrierte Schaltkreis haben Spannungsregler auch einen Eigenstromverbrauch. Bei einem maximalen Ausgangsstrom von einigen 10 mA liegt der Eigenstromverbrauch meist im Bereich von 10 bis 100 μA. Laut Datenblatt kann der Stromverbrauch des Zigbit-Moduls mit Hilfe von Schlafmodi auf bis zu 6 μA gesenkt werden. Stellt sich der Spannungsregler als effizienteste Möglichkeit der Spannungsregulierung heraus, so muss dieser mit Sorgfalt gewählt werden, um während der Schlafenszeiten nicht mehr Strom zu verbrauchen, als der Rest der Schaltung.

Ein Beispiel soll die Effizienz eines Spannungsreglers zeigen. Gegeben sei ein Spannungsregler, mit einer fixen Ausgangsspannung von 3,3 Volt und einer Dropout Voltage von 0,2 Volt. Als Spannungsquelle dienen vier in Reihe geschaltete 1,5 Volt Batterien. Die an den Spannungsregler angeschlossene Schaltung entnimmt einen Strom von 40 mA. Zu beachten ist, dass die Spannung der Batterien gegen Ende der Batterielebenszeit auf 3,2 Volt absinkt. Der Schaltung stehen somit nur noch 3,0 Volt zur Verfügung. Dies ist dennoch ausreichend (siehe Anforderung 6.2.1.1).

Spannung bei vollen Batterien:

$$1,5Volt * 4 = 6,0Volt$$

Spannung bei leeren Batterien:

$$0,8Volt * 4 = 3,2Volt$$

Die Spannungsdifferenz zwischen Ein- und Ausgangsspannung bei vollen Batterien beträgt:

$$6,0Volt - 3,3Volt = 2,7Volt$$

Bei leeren Batterien beträgt die Spannungsdifferenz:

$0,2Volt$ (Die gewünschte Ausgangsspannung ist niedriger als die Eingangsspannung.)

Im Spannungsregler wird eine Energie gleich der Spannungsdifferenz mal dem entnommenem Strom in Wärme umgewandelt.

Die Verlustleistung bei vollen Batterien beträgt:

$$2,7Volt * 0,04A = 0,108W$$

Die prozentualen Verluste bei vollen Batterien betragen:

$$(0,108W/(6,0Volt * 0,04A)) * 100\% = 45,0\%$$

Die Verlustleistung bei leeren Batterien beträgt:

$$0,2Volt * 0,04A = 0,008W$$

Die prozentualen Verluste bei leeren Batterien betragen:

$$(0,008W/(3,2Volt * 0,04A)) * 100\% = 6,3\%$$

Die Effizienz der Schaltung steigt somit im Laufe der Zeit. Die Verlustleistung geht von 108 mW auf 8 mW zurück. Die Entladekurve von Alkaline Batterien ist weitgehend linear (vgl. [Energizer Holdings Inc., 2008, S.2]). So kann die durchschnittliche Effizienz als Mittelwert zwischen dem Start- und Endwert berechnet werden.

$$100\% - ((45,0\% + 6,3\%)/2) = 74,4\%.$$

Für eine größtmögliche Effizienz sollten vier in Reihe geschaltete Batterien zum Einsatz kommen, falls sich der Spannungsregler als die effektivste Spannungsregulierungsmethode herausstellt.

6.2.1.4. Spannngswandler

Spannungswandler können je nach Modell eine Eingangsspannung in eine niedrigere aber auch eine höhere Ausgangsspannung umwandeln. Dies wird durch die Verwendung einer Spule erreicht, welche in schneller Folge ein- bzw. ausgeschaltet wird. Durch die Umwandlung von elektrischer in magnetische Energie und zurück in elektrische Energie steigt bzw. sinkt die Ausgangsspannung abhängig von der Beschaltung. Das Prinzip ähnelt einem Trafo, jedoch kommt ein Spannungswandler mit nur einer Spule aus. Auch wenn die Wandlung der Spannung nicht verlustfrei abläuft, so ist die Effizienz grade bei großen Differenzen zwischen Ein- und Ausgangsspannung erheblich größer. Die Effizienz eines Spannungswandlers liegt bei 70 bis über 90 %. Schaltregler kosten zwischen 1,50 und 10,00 Euro. Hinzu kommen die Kosten für Spule und Diode mit bis zu 1,00 Euro.

Die Spule wird mit einer Frequenz von einigen kHz bis mehreren MHz ein und ausgeschaltet. Durch die schnellen Schaltvorgänge in der Spule erzeugen Spannungswandler elektromagnetische Impulse, welche bei schlechtem Layout andere Bauteile auf der Platine oder andere Schaltungen im Umfeld negativ beeinflussen. Im Falle der Sensorknoten könnte zum Beispiel die Funk-Reichweite sinken. Die Abstrahlung muss durch einen kompakten Aufbau und die Umsetzung der Layoutvorgaben aus dem Datenblatt gering gehalten werden.

Bei modernen Spannungswandlern kann die Ausgangsspannung oftmals genauso hoch sein wie die Eingangsspannung. Für den Fall, dass Ein- und Ausgangsspannung gleich hoch sind, wird die Spule dauerhaft eingeschaltet. Eine mit Gleichspannung betriebene Spule beeinflusst diese Spannung nicht. Ein Step-Up Wandler für 3 Volt Ausgangsspannung könnte somit mit 3 Volt Eingangsspannung versorgt werden. Erst wenn die Spannung sinkt, fängt er an, die Spannung hoch zu transformieren. Ein Step-Down Wandler hört auf, die Spannung abzusenken, wenn die Eingangsspannung auf die Ausgangsspannung abgesunken ist.

Abbildung 6.5 zeigt die Standardbeschaltung eines Step-Down Wandlers. Der mit FB bezeichnete Eingang ist ein Feedback (engl. für Rückmeldung), über den der Wandler die Ausgangspannung messen und mit Hilfe der Spulenein- und Spulenausschaltzeit regulieren kann.

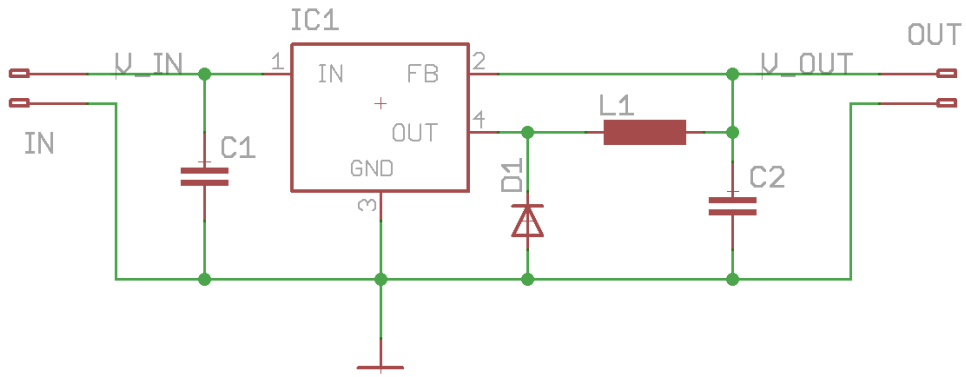

Abbildung 6.5.: Typische Beschaltung eines Step-Down Wandlers

Abbildung 6.6 zeigt die Standardbeschaltung eines Step-Up Wandlers. Auch hier gibt es eine Rückmeldung über ”FB”, über welche der Wandler die Ausgangsspannung messen und mit Hilfe der Spulenein- und Spulenausschaltzeit regulieren kann.

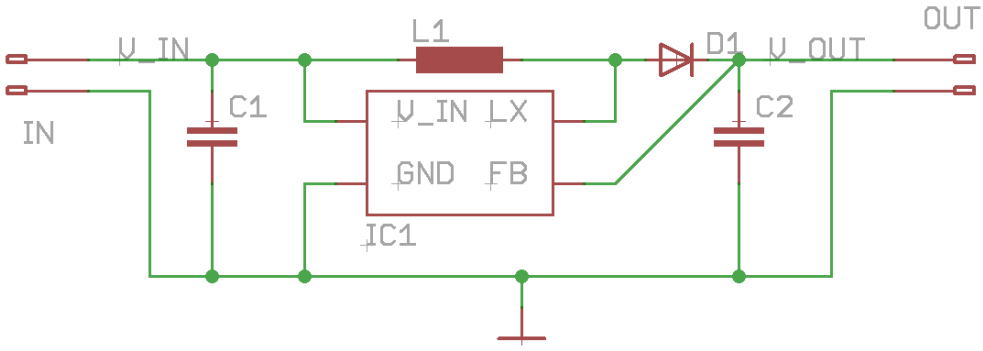

Abbildung 6.6.: Typische Beschaltung eines Step-Up Wandlers

Die Effizienz eines Spannungswandlers hängt von vielen Faktoren ab. Grundsätzlich steigt die Effektivität je größer die Eingangsspannung ist. Die richtige Dimensionierung der Spule, der Diode und des Ausgangskondensators haben einen weiteren wichtigen Einfluss auf die Effektivität. Wie die Bauteile zu dimensionieren sind, ist in den entsprechenden Datenblättern beschrieben.

Bei der Recherche nach einem geeignetem Spannungswandler stellte sich heraus, dass Step-Up Wandler für 3,3 Volt Ausgangsspannung eine Effektivität von über 90 % erreichen können. Step-Down Wandler erreichen bei 3,3 Volt meist nur 75 % Effektivität. Bei der Verwendung von Step-Down Wandlern müssen mindestens vier, besser noch fünf 1,5 Volt Zellen in Reihe geschaltet werden. Bei Step-Up Wandlern dürfen höchsten zwei Batterien in Reihe geschaltet werden. Es können aber auch vier Batterien verwendet werden, wenn je zwei in Reihe und zu den anderen beiden parallel geschaltet werden. Die Gesamtkapazität der Batterien bei einem Step-Up und einem Step-Down Wandler kann somit gleich hoch gewählt werden.

Der Eigenstromverbrauch von Spannungswandlern, welche für die Sensorknoten in Frage kommen, liegen bei 5 bis 100 μA.

6.2.1.5. Entscheidung für Step-Up Wandler

Von den analysierten Methoden zur Spannungsversorgung stellen Step-Up Wandler die effizienteste Lösung dar. Der komplexere Aufbau und der höhere Preis sind für den Aufbau dieses Sensornetzwerks zweitrangig.

Nach der Analyse zahlreicher Datenblätter von über 20 Step-Up Wandlern fiel die Entscheidung auf den TPS61097 (siehe Texas Instruments Incorporated [2009]). Dieser Spannungswandler hat eine konstante Ausgangsspannung von 3,3 Volt. Laut Datenblatt liefert er bei 1,6 Volt Eingangsspannung[3] noch bis zu 70 mA Strom bei einer Effizienz von über 85%. Je voller die Batterien sind, desto größer wird die Effizienz und der entnehmbare Strom. Die geforderten 40 mA können somit in jedem Fall bereit gestellt werden. Sein Eigenstromverbrauch liegt bei lediglich 5 μA. Abhängig vom Ladestand der Batterien und des entnommenen Stroms wird eine Effizienz von ca. 85 bis 94 %[4] erreicht. Diese Effizienz liegt damit immer über der durchschnittlichen Effizienz von Spannungsreglern.

Es gibt noch effizientere Step-Up Wandler, deren Kosten-Nutzen Verhältnis jedoch nicht mehr im Gleichgewicht erscheint. Die Preise steigen teilweise um über das Vierfache und die Effizienz steigt um durchschnittliche 1 bis 2 %.

Die Step-Up Wandler werden mit zwei in Reihe geschalteten Batterien des Typs Mignon versorgt. So steht eine möglichst große Kapazität der Batterien zur Verfügung, ohne dass die maximale angestrebte Baugröße überschritten wird (vgl. Anforderung Unauffälligkeit).

6.2.2. Spannungsversorgung von ZigBee-Routern

Einige ZigBee-Teilnehmer müssen als Router arbeiten. Diese ZigBee-Router müssen extern mit Energie versorgt werden, da sie nicht in Stromsparmodi wechseln können und somit zu viel Strom für einen Batteriebetrieb verbrauchen.

Bei der Installation kann viel Zeit gespart werden, wenn keine neuen Kabel für die Stromversorgung der Sensorknoten verlegt werden müssen. Die in Projekt 1 verbauten Fenstermotoren benötigen zusammen etwa 6 Ampere. Verbaut wurde jedoch ein 10 Ampere Netzteil. Es steht somit genug Strom für andere Anwendungen zur Verfügung.

Die Spannung der Fenstermotoren beträgt 24 Volt. Bei diesem großen Spannungsunterschied zwischen zur Verfügung stehender Spannung (24 Volt) und benötigter Spannung der Sensoren (3,3 Volt) kommen Spannungsregler nicht in Betracht. Die Effizienz eines Spannungsreglers liegt unter 14 %:

$3,3Volt/24Volt * 100\% = 13,75\%$

[3] Spannung zweier verbrauchter in Reihe geschalteter Batterien.
[4] Entnommen aus Figure 2 [Texas Instruments Incorporated, 2009, S. 9]

Bei einer Stromaufnahme von lediglich 40 mA würden über 800 mW in Wärme umgewandelt werden:

$(24Volt - 3,3Volt) * 0,04A = 0,828Watt$

Ein Temperaturmessung in der Nähe dieses Spannungsreglers würde immer verfälscht werden. Zudem ist diese Methode keineswegs ”stromsparend” wie es in der Überschrift dieser Masterarbeit verlangt wird.

Für die Versorgung der ZigBee-Router wird eine kleine Zusatzplatine gefertigt, auf der sich ein Step-Down Wandler befindet, welcher eine Spannung von 24 Volt auf 3,3 Volt herab regelt. Zudem wird die Platine der Sensorknoten so entwickelt, dass anstelle des der Step-Up Wandler auf der Platine eine Drahtbrücke verlötet werden kann, wodurch extern eine Spannung von 3,3 Volt zugeführt werden kann. Als Step-Down Wandler wird ein LM2574-3,3 von National Semiconductors verwendet (Datenblatt siehe Texas Instruments [2011]). Die Platine des Step-Down Wandler wird entsprechend der Applikation Notes des Datenblattes gefertigt.

Der Schaltplan und das Layout der Platine mit dem Step-Down Wandler befinden sich im Anhang B.3. Es sei hier darauf hingewiesen, dass für den Step-Down Wandler eine maximale Eingangsspannung von 28 Volt zulässig ist. Die Erklärung befindet sich ebenfalls im Anhang B.3.

6.2.3. Hardwaredesign

Für einen funktionierenden Sensorknoten werden der Prozessor samt RF-Chip und Antenne sowie eine Spannungsversorgung benötigt. Die Sensoren sollen individuell hinzugefügt werden können. Da es unzählige Gehäusetypen für die einzelnen Sensoren gibt, ist das Erstellen von Platzhaltern auf der Platine des Sensorknoten nicht möglich. Zudem sollen die Sensoren absetzbar sein. Somit werden lediglich die notwendigen Schnittstellen nach außen geführt.

Da es bei der Fertigung des Masterknotens bereits bei der dritten Platine zu mehreren Kurzschlüssen kam (siehe Kap. 6.1.2) und von den Sensorknoten eine größere Anzahl gefertig werden soll, werden die kleinen Strukturen des ATMega128RFA1 vermieden und es wird auf das fertige ZigBit-Modul zurück gegriffen. Dieses Modul verfügt über einen Prozessor, einen RF-Chip und zwei Antennen, welche abhängig vom Empfang automatisch umgeschaltet werden. Das ZigBee-Modul besitzt alle geforderten Schnittstellen (siehe Anforderung Unterstützte Schnittstellen). Die optional geforderte serielle Schnittstelle ist vorhanden. Die maximale Geschwindigkeit beträgt dabei bis zu 76,8 KBaud. Das Modul benötigt lediglich zwei Kondensatoren zur Stabilisierung der Spannung.

6.2.3.1. Layoutvorgaben der Platine

Das verwendete ZigBit-Modul stellt einige Bedingung an die zu entwickelnde Platine, die ähnlich den Vorgaben des Masterknotens sind. Das ZigBit-Modul muss mittig sitzen. Wie auch beim Masterknoten dient die Massefläche als Gegenpol der Antenne und wird möglichst großflächig ausgeführt. Die Platine ist mit Vias zu umranden, um ungewollte Ab- und Einstrahlungen zu unterbinden. Im Bereich der Antenne sind metallische Gegenständen zu vermeiden (siehe Datenblatt: [MeshNetics, 2007, S. 16]).

Weitere Layoutvorgaben entstehen durch möglichst kompakten Aufbau der Sensorknoten. Unterhalb der Platinen wird ein Bereich für einen Batteriehalter frei gelassen. Im Bereich des Batteriehalter dürfen keine Through Hole Devices (kurz THT, engl. für Durchsteckbauteile) verwendet werden. Da der Batteriehalter sonst nicht befestigt werden kann.

Um eine möglichst geringe Bauhöhe zu erreichen, werden die Anschlüsse für die Sensoren auf der Unterseite der Platine neben dem Batteriehalter platziert.

Sowohl der Schaltplan als auch das Layout befinden sich im Anhang B.2.

6.2.4. Fertiggestellte Platine

Wie bereits der Masterknoten, wurde auch die Platine der Sensorknoten und des Step-Down Wandlers von einem Leiterplattenhersteller gefertig. Die Bauteile wurden anschließend von Hand verlötet. Durch die Verlötung von Hand war es möglich, nur einzelne Funktionsgruppen zu bestücken. So wurden zunächst einige Platinen nur mit dem Step-Up Wandler bestückt und dessen Funktionalität getrennt von der restlichen Schaltung getestet.

6.2.4.1. Aufgebauter Sensorknoten

Das Foto 6.7 zeigt einen voll bestückten Sensorknoten mit darunter befestigtem Batteriehalter. Der Batteriehalter wurde mittels wieder ablösbarem zweiseitigem Klebeband fixiert. Im vorderen Bereich befindet sich der Step-Up Wandler samt Spule und der benötigten Kondensatoren. In der Mitte der Platine befindet sich das ZigBit-Modul, mit Außenbeschaltung und zwei LEDs. Im hinteren Bereich befindet sich ein Resettaster, um das Modul neu zu starten. Die Anschlüsse für die Sensorknoten befinden sich auf der Unterseite neben dem Batteriehalter.

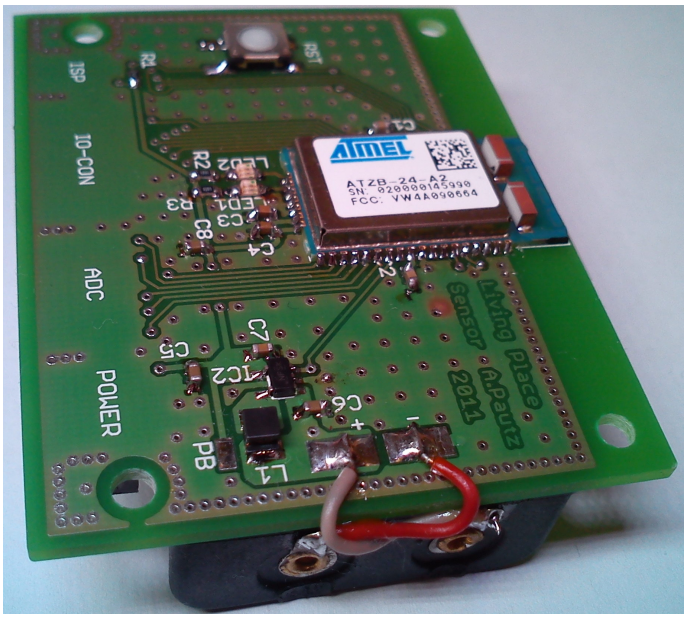

Abbildung 6.7.: Fertiggestellter Sensorknoten

Das fertige Modul inklusive Batteriehalter und eingelegter Batterien hat die Abmessungen 60 x 48 x 20 mm. Ein passendes Gehäuse, welches genug Platz für drei bis fünf Sensoren bietet, kann in allen Dimensionen kleiner bleiben als die angestrebte Größe der Zigarettenschachtel (86 x 55 x 23 mm siehe Anforderung Unauffälligkeit).

6.2.4.2. Aufgebauter Step-Down Wandler

Platinen die als ZigBee-Router arbeiten sollen, können nicht über Batterien versorgt werden. Bei diesen Platinen wurde der Step-Up Wandler nicht bestückt. Für eine stabile Spannungsversorgung wurde hierfür der Step-Down Wandler auf einer zusätzlichen Platine verarbeitet, welcher eine Spannung zwischen 4 und 24 Volt auf 3,3 Volt wandelt. Dieser Wandler ist auf dem Foto 6.8 zu sehen. Der Step-Down Wandler kann genauso wie der Batteriehalter auf der Rückseite der Platine oder über die zwei Bohrlöcher im Gehäuse des Sensorknoten befestigt werden.

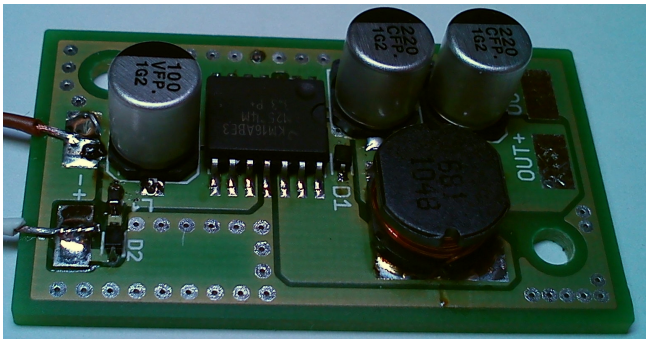

Abbildung 6.8.: Fertiggestellter Step-Down Wandler

7. Softwareentwicklung

In diesem Kapitel wird die Softwareentwicklung des Sensornetzwerks beschrieben. Dabei werden zunächst die Kommunikationsschnittstellen zwischen den einzelnen Komponenten festgelegt und anschließend die Software einzelnen Komponenten erläutert.

7.1. Kommunikationsschnittstellen

Es gibt eine Reihe von Komponenten, welche miteinander Kommunizieren müssen. So kommunizieren die Sensoren mit dem Masterknoten, der Masterknoten mit der PC-Software und die PC-Software über das Blackboard mit der Software aus anderen Projekten. Dabei hat jede Schnittstelle andere Anforderungen, welche erfüllt werden müssen.

7.1.1. Kommunikation im ZigBee-Netzwerk

Die Kommunikation innerhalb des ZigBee-Netzwerk sollte die paketorientierte Übertragungsmethode von ZigBee berücksichtigen. Durch das Erstellen mehrerer Endpunkte können verschiedene Daten unterschieden werden. So kann ein Kanal alleine für den Empfang von Messwerten verwendet werden. Ein anderer Kanal kann für die Übertragung von Fehlermeldungen vorgesehen werden.

Da sowohl der Masterknoten als auch die Sensorknoten nur eine eingeschränkte Rechenkapazität haben, sollte die Datenübertragung ein systemnahes Protokoll verwenden (siehe Kap. 5.2.1). Bei einer Recherche nach einem offenem, verbreitetem systemnahem Protokoll wurde kein geeignetes gefunden. Die ZigBee Alliance selbst bietet einige vordefinierte Endpunkte. Dadurch sollen unabhängige Anwendungen miteinander kommunizieren können, doch sind diese Definitionen nicht öffentlich verfügbar, sondern nur gegen Bezahlung erhältlich. Aus diesem Grund wurde ein eigenes Protokoll definiert, welches für die Kommunikation im Sensornetzwerk verwendet wird.

Das selbst definierte Protokoll verwendet für jeden Nachrichtentyp einen eigenen Endpunkt. Insgesamt werden drei unterschiedliche Endpunkte für die Übertragung von Messwerten, Sensorbeschreibungen und Fehler- bzw. Debugmeldungen verwendet. Um die Nachrichten kurz und so den Overhead klein zu halten, werden möglichst nur IDs verwendet. Anstelle eines Sensorennamens wird so bei einem Messwert nur eine Sensor ID verwendet.

Die genaue Umsetzung des Protokolls ist im Anhang D beschrieben.

7.1.2. Kommunikation zwischen Masterknoten und PC-Software

In Kapitel 5.2.2 wurde entschieden, dass die Kommunikation über eine virtuelle serielle Schnittstelle erfolgt. Die Kommunikation auf der seriellen Schnittstelle erfolgt byteorientiert. Mittels eines geeigneten Protokolls muss diese so erweitert werden, dass mehrere Bytes zu logischen Paketen zusammengefasst werden können. Einen Standard hierfür gibt es nicht, jedoch gibt es unterschiedliche Ansätze wie dies erfolgen kann. Als Vereinfachung kommt in diesem Fall hinzu, dass keine Sendewiederholung implementiert werden muss, da dies bereits durch die USB-Verbindung erfolgt.

7.1.2.1. ASCII Datenübertragung

Bei einer zeichenbasierten Datenübertragung werden bestimmte Symbole als Steuerzeichen benutzt. Ein Beispiel hierfür ist das NMEA 0183 Protokoll[1]. Dieses in der Schifffahrt und bei GPS-Geräten verwendete Protokoll nutzt das Dollarzeichen als Datensatzbeginn, das Komma zur Trennung zwischen verschiedenen Werten und den Zeilenumbruch für das Ende einer Nachricht.

Der Vorteil diese Datenübertragung liegt in der meist guten Lesbarkeit für den Menschen. Der Nachteil einer solchen Übertragung ist, dass Steuerzeichen in den Daten der Nachricht selbst nicht vorkommen dürfen. Da Zahlen als lesbare Zeichenketten übertragen werden, steigt zudem der Rechenaufwand für die Prozessoren, was gerade bei dem Masterknoten vermieden werden sollte.

7.1.2.2. Binäre Datenübertragung

Bei der binären Datenübertragung stehen alle Zeichen für die Übertragung zur Verfügung. Damit die Paketgrenzen dennoch erkannt werden, gibt es verschiedene Möglichkeiten. Eine Möglichkeit ist es, zu Beginn oder Ende einer Nachricht absichtlich einen Fehler auf dem Bus zu erzeugen. So könnte zum Beispiel eine serielle Datenübertragung neun statt 8 Datenbits enthalten. Eine andere Möglichkeit ist es, ein Zeichen weiterhin als Steuerzeichen zu definieren. Kommt dieses Zeichen in der Nachricht vor, wird es dort maskiert. Meist reicht es jedoch, eine feste Präambel von einigen Zeichen zu verwenden, welche nicht oder nur sehr unwahrscheinlich in den Daten vorkommen kann. Über die Verwendung einer Längenangabe des Pakets und einer CRC kann dann überprüft werden, ob die Daten richtig interpretiert wurden.

[1] Siehe zum Beispiel das NMEA Reference Manual eines Fasttrax IT300 GPS Empfängers: http://www.fastraxgps.com/showfile.cfm?guid=9de68fec-1d95-4fd3-8809-f068c9aaf220 - Abruf: 07.02.2012

7.1.2.3. Entscheidung für binäre Datenübertragung

Für die Kommunikation zwischen Masterknoten und PC-Software wird eine binäre Datenübertragung verwendet. Dies bedeutet für den Masterknoten weniger Rechenaufwand als die Verwendung einer ASCII Datenübertragung. Das Protokoll besitzt eine feste Präambel, eine Längenangabe und eine CRC-Prüfung. Die Längenangabe und die Päambel sind ausreichend, um eine Verwechslung der Präambel mit den Daten selbst auszuschließen.

Die genaue Umsetzung des Protokolls ist im Anhang E beschrieben.

7.1.3. Kommunikation auf dem PC

Die PC-Software kommuniziert mit der Software aus anderen Projekten über das Blackboard, den Active MQ Server. Über die Topics des Active MQ werden Nachrichten im menschenlesbaren JSON Format ausgetauscht. Für unterschiedliche Nachrichtentypen werden unterschiedliche Topics eröffnet. Weiterhin wird ein Nachrichtenaustausch ohne Active MQ vorgesehen, um so eine Ablösung durch eine CIM Implementation zu ermöglichen (siehe Anforderung 2.1). Für den Fall, dass der Active MQ Server nicht mehr verwendet wird, werden Nachrichten im JSON-Format über TCP/IP Sockets versendet und empfangen.

Bei der Kommunikation auf diesem Level kommt es mehr auf eine einfache Verständlichkeit der Nachrichten an. So sollen zum Beispiel Sensornamen und keine Sensor IDs verwendet werden.

Die genaue Umsetzung des Protokolls ist im Anhang F beschrieben.

7.2. Software der Sensorknoten

Die Aufgabe der Sensorknoten besteht darin Messwerte aus den angeschlossenen Sensoren auszulesen und sie über das ZigBee-Netzwerk zu versenden. In den Pausen zwischen den Messwerterfassungen haben Sensorknoten, die als End-Devices konfiguriert sind, nichts zu tun und verbringen die Zeit in einem Stromsparmodus, in dem nur noch der Timer zum Aufwecken des Knotens aktiv ist. ZigBee-Router bleiben in der Zeit aktiv und übermitteln Messdaten anderer Sensorknoten. Das Routing wird dabei vom BitCloud-Stack übernommen, so dass dies keiner besonderen Aufmerksamkeit bei der Entwicklung bedarf. Aus Sicht der zu programmierenden Software gibt es somit nur einen Unterschied zwischen End-Device und Router. Aus diesem Grund wird für alle Sensorknoten die gleiche Software entwickelt.

7.2.1. Zustände der Sensorknoten

Der folgende Zustandsautomat (Abb. 7.1) zeigt den Programmablauf der Sensorknoten. Der zentrale Zustand ist ”Calc Sleep”. In ihm werden die Abfrageintervalle aller angeschlossenen Sensoren verwaltet. Im Zustand ”Sleep” gehen End Devices in einen Stromsparmodus. Router bleiben in diesem Zustand aktiv und warten die Zeit bis zur Abfrage des nächsten Sensors in diesem Zustand ab.

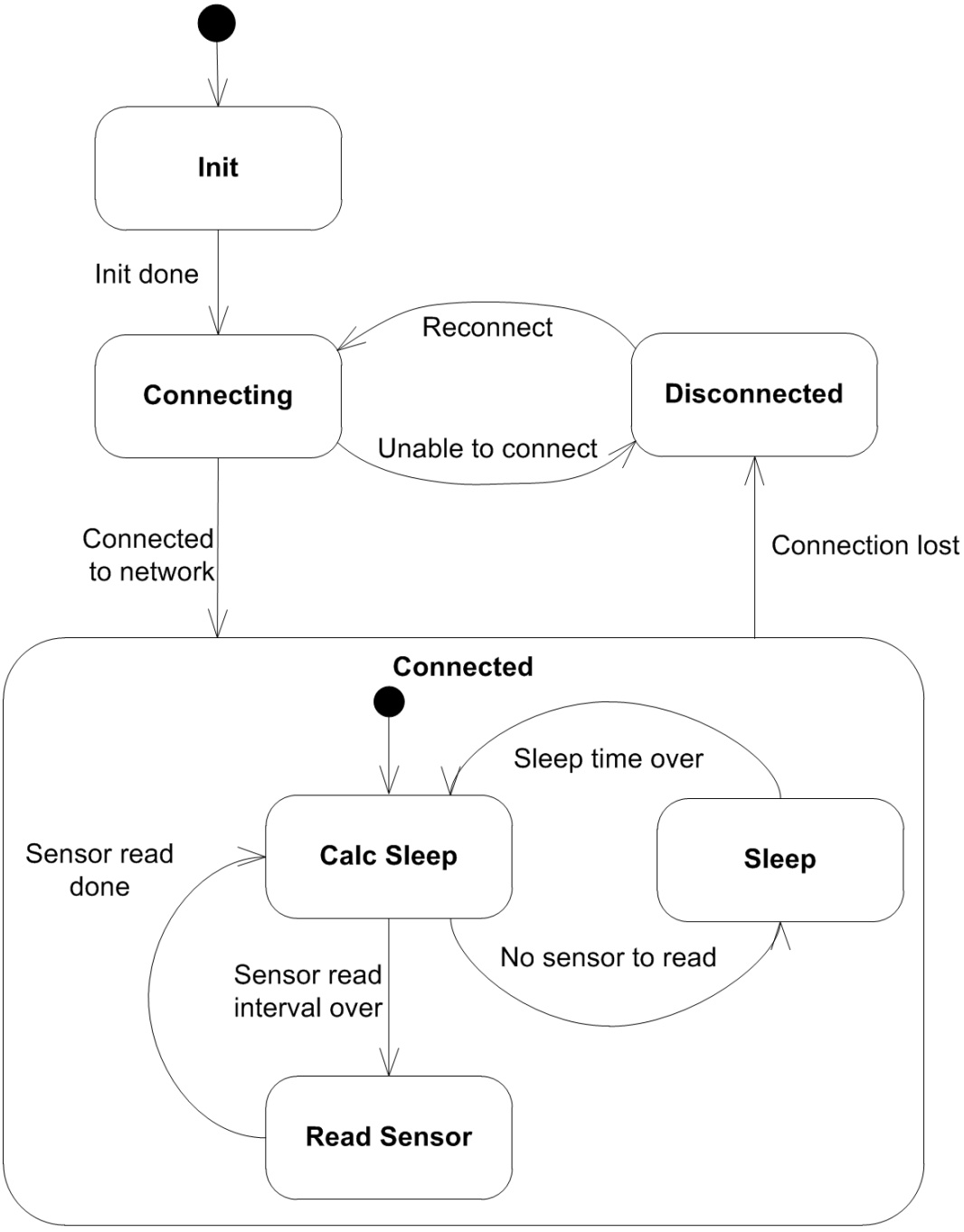

Abbildung 7.1.: Zustandsautomat der Sensorknoten

Nicht abgebildet ist, dass jeder Sensorknoten in der Lage ist Nachrichten zu empfangen. Über diese Nachrichten ist es möglich, den Abfrageintervall eines Sensors zu ändern. So kann der Abfrageintervall an die aktuellen Bedürfnisse angepasst werden. Schlafende Sensorknoten empfangen die Nachricht dabei erst beim nächsten Aufwachen. Die Zwischenpufferung der Nachrichten erfolgt in den Routerknoten des ZigBee-Netzwerks.

7.2.2. Behandlung des Unterschieds zwischen End-Device und Router

Ob der Sensorknoten in den Pausen aktiv bleibt oder nicht, wird über eine Präprozessoranweisung während des Kompilierens entschieden. Der folgende Codeausschnitt entstammt der Hauptschleife der Sensorknoten. Der Ausschnitt zeigt, wie End-Device über die Funktion ZDO_SleepReq, welche vom BitCloud-Stack bereit gestellt wird, in den Stromsparmodus versetzt werden. Das Define ”STACK_TYPE_ENDDEVICE” wird während der Kompilierung nur für End-Devices definiert.

```
1       switch(main_state)
2       {
3       //...
4       //Abarbeitung anderer Zustaende
5       //...
6       case STATE_ENTER_POWER_DOWN:
7   #ifdef STACK_TYPE_ENDDEVICE
8           //Nur End-Devices koennen schlafen
9           sleep_req.ZDO_SleepConf = ZDO_SleepConf;
10          ZDO_SleepReq(&sleep_req);
11  #endif
12          break;
13      }
```

Abbildung 7.2.: Unterschiedliche Behandlung von Routern und End-Devices im Schlafzustand

7.2.3. Softwarebasis aus Projekt 2

Da wie in PJ2 der BitCloud-Stack für die Kommunikation im ZigBee-Netzwerk verwendet wird, kann die Software aus PJ2 als Basis für diese Masterarbeit verwendet werden (siehe [Pautz, 2011b, S. 19ff]).

7.2.4. Treiber für Sensoren

Zum Zeitpunkt der Entwicklung ist es nicht möglich vorauszusagen, welche Sensoren jemals im Sensornetzwerk verwendet werden. Um die spätere Entwicklung zu vereinfachen, werden zumindest Treiber für die gesamte OnChip Peripherie zur Verfügung gestellt. Dies senkt den Aufwand einer späteren Erweiterung mit neuen Sensoren enorm, da sofort die Treiber der OnChip Peripherie genutzt werden können und nicht erst die Register zur Programmierung der Peripherie gesucht werden müssen.

7.2.4.1. Entwicklung eigener Treiber

Obwohl der ZigBee-Stack über eine Reihe von Treibern für die OnChip Peripherie verfügt, werden diese Treiber nicht genutzt. Der BitCloud-Stack nutzt das Event Driven Programming Model sehr stark (siehe [Atmel Corporation, 2011c, S.11ff]). Hierbei werden die aufgerufenen Funktionen nicht sofort ausgeführt, sondern kehren sofort zum Aufrufer zurück. Erst wenn der Stack selbst aufgerufen wird, werden diese Funktionen abgearbeitet. Die erfolgte Abarbeitung wird der Benutzersoftware anschließend über Callback Funktionen mitgeteilt. Dieser Mechanismus spart meist Rechenzeit, erhöht jedoch die Komplexität sehr stark, weshalb selbst entwickelte Treiber verwendet werden. Auch wenn durch die selbst entwickelten Treiber Rechenzeit verloren geht, ist die Abfrage eines Sensors in lediglich einer Funktion möglich.

Abfrage eines Sensorwerts mittels der BitCloud Treiber Abbildung 7.3 zeigt, wie eine Sensorabfrage mit dem Event Driven Programming Model aussehen würde. Die Sensorabfragefunktion beauftragt den Stack Daten aus dem Sensor zu lesen. Der Stack speichert die Anfrage ab und kehrt sofort zurück. Die Sensorabfragefunktion hat nach dem Absetzen der Anfrage nichts mehr zu tun und übergibt die Kontrolle dem Stack. Während der Stack seine Funktionen abarbeitet, führt er auch die von der Sensorabfragefunktion angeforderte Abfrage durch. Nachdem die Daten von den Sensoren eingelesen wurden, übergibt der Stack die Daten den Datenverarbeitungsfunktion.

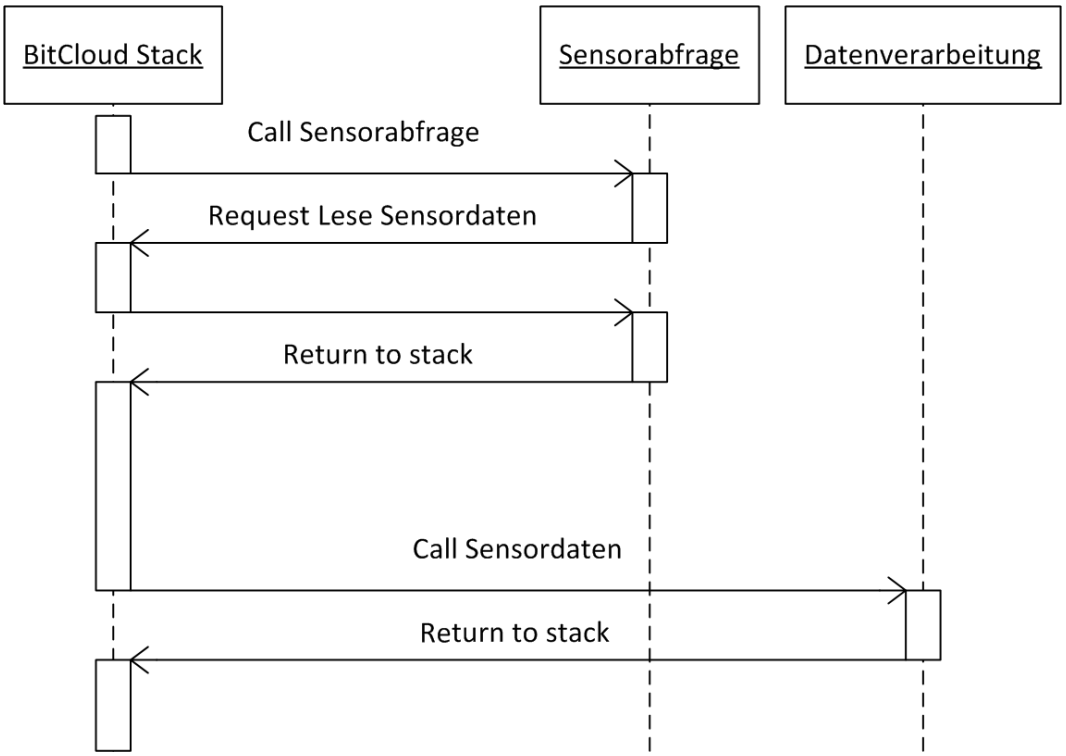

Abbildung 7.3.: Abfrage der Sensoren mit Treibern des BitCloudstack

In dem obigem Beispiel wird davon ausgegangen, dass der Sensor direkt abgefragt werden kann. Sensoren mit einem I^2C, SPI, UART oder One Wire Interface sind meist komplexer. Sie bieten oftmals verschiedene Optionen, wie eine einstellbare Genauigkeit oder eine Umschaltung des Wertebereichs an. Diesen Sensoren muss vor der eigentlichen Abfrage mitgeteilt werden, dass der Messwert ausgelesen werden soll und keine Einstellung ausgelesen oder geändert werden soll. Für jeden zusätzlichen Schritt bei der Kommunikation mit den Sensoren muss eine weitere Funktion geschrieben werden, welche wie die ”Sensorabfrage” Funktion ein Request an den Stack mit einer Lese- oder Schreiboperation absetzt.

Abfrage eines Sensorwerts mittels eigenen Treibern Durch die Verwendung von selbst entwickelten Treibern vereinfacht sich die Abfrage von Messwerten aus den Sensoren. Wie in der folgenden Grafik 7.4 zu sehen ist, blockiert die Funktion ”IO-Treiber” die ”Sensorabfrage” so lange, bis die Sensorwerte ausgelesen sind. Hierdurch geht zwar Rechenzeit verloren, jedoch ist der Aufbau der Funktion einfacher. Die gesamte Abfrage eines Sensors kann in einer einzelnen Funktion erfolgen. Auch wenn mehrere Kommunikationsschritte zwischen Sensor und Sensorknoten nötig sind, erhöht sich die Komplexität durch zusätzliche Funktionen nicht.

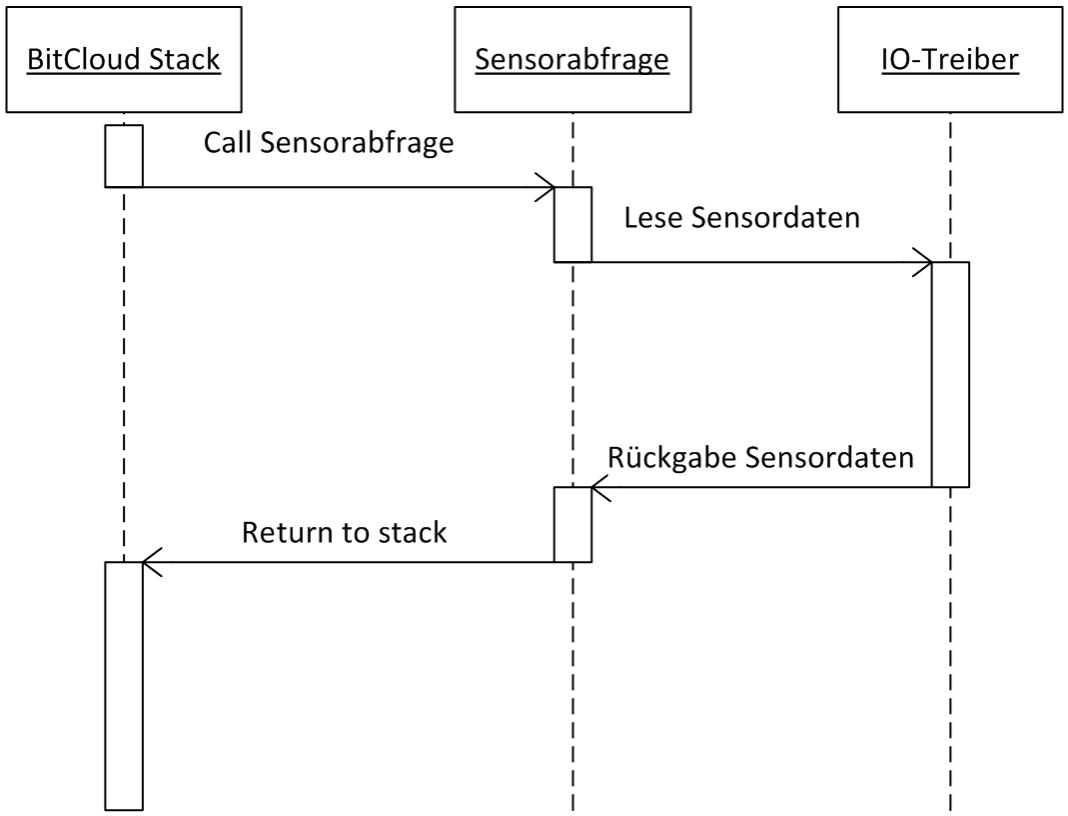

Abbildung 7.4.: Abfrage der Sensoren mit selbst entwickelten Treibern

Zur Verfügung gestellte Treiber Der ATMega1281 besitzt GPIOs, SPI, I^2C, USART und ADC als Schnittstellen (vgl. [Atmel Corporation, 2011b, S. 1]), für welche alle Treiber geschrieben werden. Zusätzlich wird ein Treiber für den One-Wire Bus zur Verfügung gestellt. Dieser Bus wird über die GPIOs des ATMega1281 realisiert. Die Anforderung "Unterstützte Schnittstellen" wird mit diesen Treibern erfüllt. Zusätzlich wird mit dem UART-Treiber die optionale Anforderung für eine serielle Schnittstelle erfüllt. Ebenso wird zusätzlich zu den geforderten Schnittstellen der One-Wire Bus unterstützt.

Wie die Treiber zu verwenden sind, ist mit Hilfe einer Doxygen Dokumentation beschrieben. Diese Beschreibung befindet sich auf der beigelegten DVD (siehe hierfür Anhang G)

7.3. Software des Masterknotens

Der Masterknoten arbeitet als Gateway zwischen dem ZigBee-Netzwerk und der PC-Software. Da er immer vorhanden sein muss, übernimmt er zudem die Aufgabe des Coordinators. Genauso wie die Sensorknoten verwendet der Masterknoten den BitCloud-Stack.

Anders als die Sensoren nutzt der Masterknoten die vom BitCloud-Stack zur Verfügung gestellten Treiber. Durch das Event Driven Programming Model kann der Masterknoten

so auf Nachrichten vom ZigBee-Netzwerk, als auch auf Nachrichten vom PC gleichzeitig lauschen, ohne Multithreading verwenden zu müssen.

Der Masterknoten muss mit dem ZigBee-Netzwerk und der seriellen Schnittstelle zwei sehr unterschiedliche Kommunikationsschnittstellen verbinden. Damit es beim Übergang von ein Netzwerk in ein anderes zu keinem Datenverlust kommt, wird ein Fifo zur Pufferung von Nachrichten eingesetzt.

7.4. PC-Software

In diesem Abschnitt werden zunächst die Aufgaben der PC-Software erläutert. Anschließend wird darauf eingegangen, wie die Umsetzung erfolgt.

7.4.1. Aufgaben der PC-Software

Die Sensorknotenverwaltung hat mehrere Aufgaben. Eine Aufgabe ist das Entgegennehmen der Messwerte vom Masterknoten. Aus den empfangen Messwerten muss die Sensor ID ermittelt werden, welche die PC-Software in einen Sensornamen umwandelt. Der aufbereitete Messwert wird dann über den Active MQ Server an die Software anderer Projekte übermittelt. Zusätzlich wird jeder Messwert gespeichert. So kann er zu einem späterem Zeitpunkt abgefragt werden. Dadurch ist es möglich, später Statistiken oder Simulationen mit Daten aus der Vergangenheit zu erstellen.

Das Sensornetzwerk besitzt keine Information darüber, welche Sensorwerte mit welcher Aktualität vorliegen müssen. Die Steuerung des Abfrageintervalls obliegt somit anderen Projekten, welche die Messwerte verarbeiten. Wie oben erwähnt, werden Sensoren im Sensornetzwerk mit IDs und auf der PC-Seite mit Namen angesprochen. Externe Software besitzt keine Informationen darüber, welche Sensor-IDs zu welchen Sensornamen gehören. Diese Verbindung kann nur die PC-Software des Sensornetzwerks herstellen. Sie muss die Konfigurationsanfragen entgegennehmen und über den Masterknoten an die Sensorknoten weiterleiten.

Die PC-Software muss somit drei unterschiedliche Eingaben verarbeiten. Dazu gehört der Empfang aller Daten aus dem Sensornetzwerk, Anfragen zu Messwerten aus der Vergangenheit, welche aus der internen Datenspeicherung geladen werden und Konfigurationsanfragen für Sensoren. Jeder Dateneingabe steht eine Datenausgabe gegenüber. Es sind somit auch drei verschiedene Datenausgaben zu bedienen. Der wichtigste Ausgabekanal ist die Ausgabe von Messwerten. Auf der PC-Seite gibt es einen zweiten Ausgabekanal für Statusmeldungen. Dieser wird auch zur Ausgabe von Fehlermeldungen verwendet. Der letzte Ausgabekanal geht über den Masterknoten in das Sensornetzwerk. Über diesen Kanal wird der Abfrageintervall der Sensoren konfiguriert. In der folgenden Grafik 7.5) zeigt die drei Kommunikationskanäle.

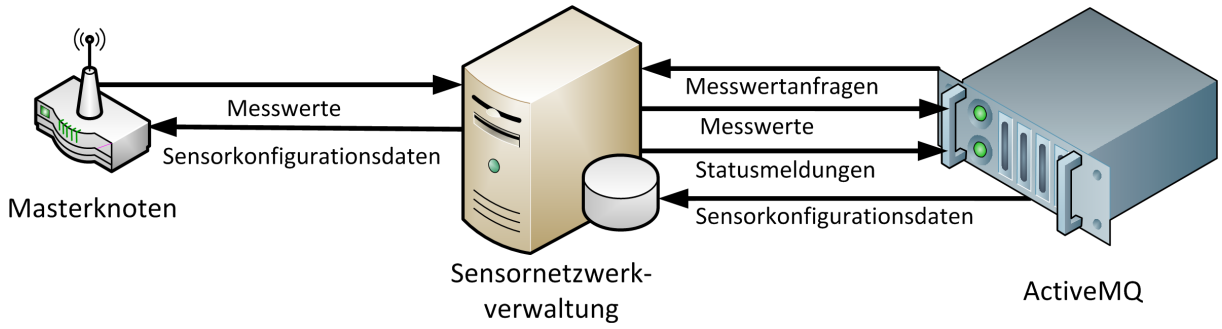

Abbildung 7.5.: Schnittstellen der Sensorknotenverwaltung

7.4.2. Funktionaler Aufbau

Um eine möglichst hohe Flexibilität zu erzielen wurde entschieden, den Kern der PC-Software sowohl ohne direkte Anbindung an den Masterknoten als auch den Active MQ Server zu entwickeln. Die Kommunikation zwischen PC-Software und Active MQ und zwischen PC-Software und Masterknoten übernehmen zusätzliche Programme. Diese lassen sich austauschen, ohne dass die Kernsoftware selbst angefasst werden muss. So wird eine spätere Anpassung vom Active MQ Server auf eine CIM basierte Lösung ermöglicht (siehe Anforderung ”Bereitstellung von Messwerten”). Die Kommunikation zwischen den einzelnen Programmen erfolgt mittels JSON über TCP/IP. Eine Dokumentation der Nachrichten befindet sich im Anhang F.

Die folgende Grafik 7.6 zeigt die Verfeinerung der zuvor gezeigten Grafik 7.5. Durch eine entsprechende Programmierung ist es möglich, dass auf jedem Kommunikationskanal mehrere Verbindungen aufgebaut werden. So können mehrere Sensornetzwerke parallel verwaltet oder parallele Verbindungen zu CIM und Active MQ unterhalten werden (siehe Anforderung ”Bereitstellung von Messwerten”).

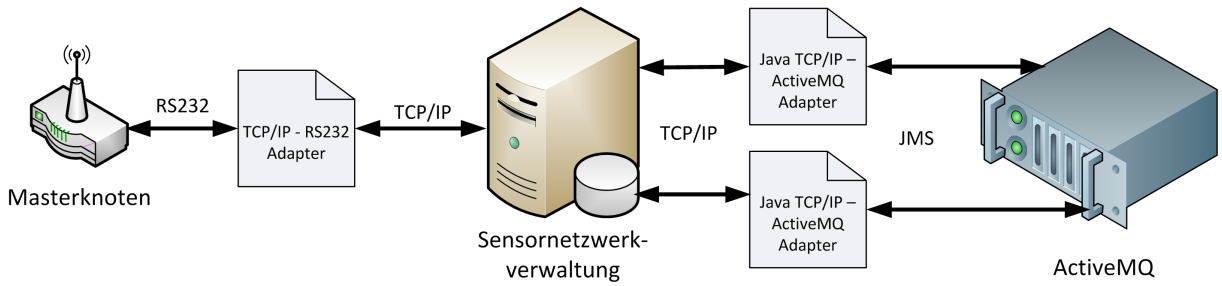

Abbildung 7.6.: Sensorknotenverwaltung mit erhöhter Flexibilität

Im ersten Moment scheint die Entwicklung zusätzlicher Komponenten mehr Zeit in Anspruch zu nehmen. Dies ist jedoch nicht der Fall, da auf bereits existierende Software bzw. Softwarekomponenten zurückgegriffen werden kann. So wird die Kernsoftware zudem schlanker und kann besser getestet werden. Für die Kommunikation zwischen PC und Masterknoten kann Software aus einem früherem Projekt verwendet werden (siehe

hierzu Pautz und Johannsen [2010]). Es müssen lediglich die Nachrichten angepasst werden. Für die Kommunikation zum Active MQ Server werden fertige Java Klassen anderer Studenten verwendet (siehe Otto und Voskuhl [2011]). Diese Komponenten müssten sonst in C++ selbst neu implementiert werden.

Die PC-Software wird in mehrere Programme unterteilt. So wird der Aufbau sehr flexibel und kann leicht angepasst werden.

Übersicht über Komponenten der Kernsoftware Wie in der folgenden Abbildung 7.7 zu sehen ist, besteht der Kern der PC-Software im Wesentlichen aus 4 Komponenten. Die *Datenbankkomponente* übernimmt die Speicherung aller Daten. Der *In Server* nimmt neue Messwerte aus dem Sensornetzwerk entgegen. Über den *CFG Server* können Informationen über Sensoren abgefragt und die Schlafenszeit der Sensoren eingestellt werden. Zusätzlich werden auf diesem Kanal Statusinformationen aus dem Sensornetzwerk auf diesen Ausgang weitergeleitet. Aus dem *Out Server* werden Messwerte ausgegeben. Auch können früher aufgenommene Messwerte auf diesem Kanal abgefragt werden. Die Anfragen werden dann zur Datenbank durchgereicht, welche die Antwort an den Out Server zurückliefert. Die Out und CFG Server Schnittstellen sind über ein Java Programm mit dem Active MQ Server verbunden. Die *Netzwerkverwaltung* übernimmt den Auf- und Abbau von TCP/IP Verbindungen.

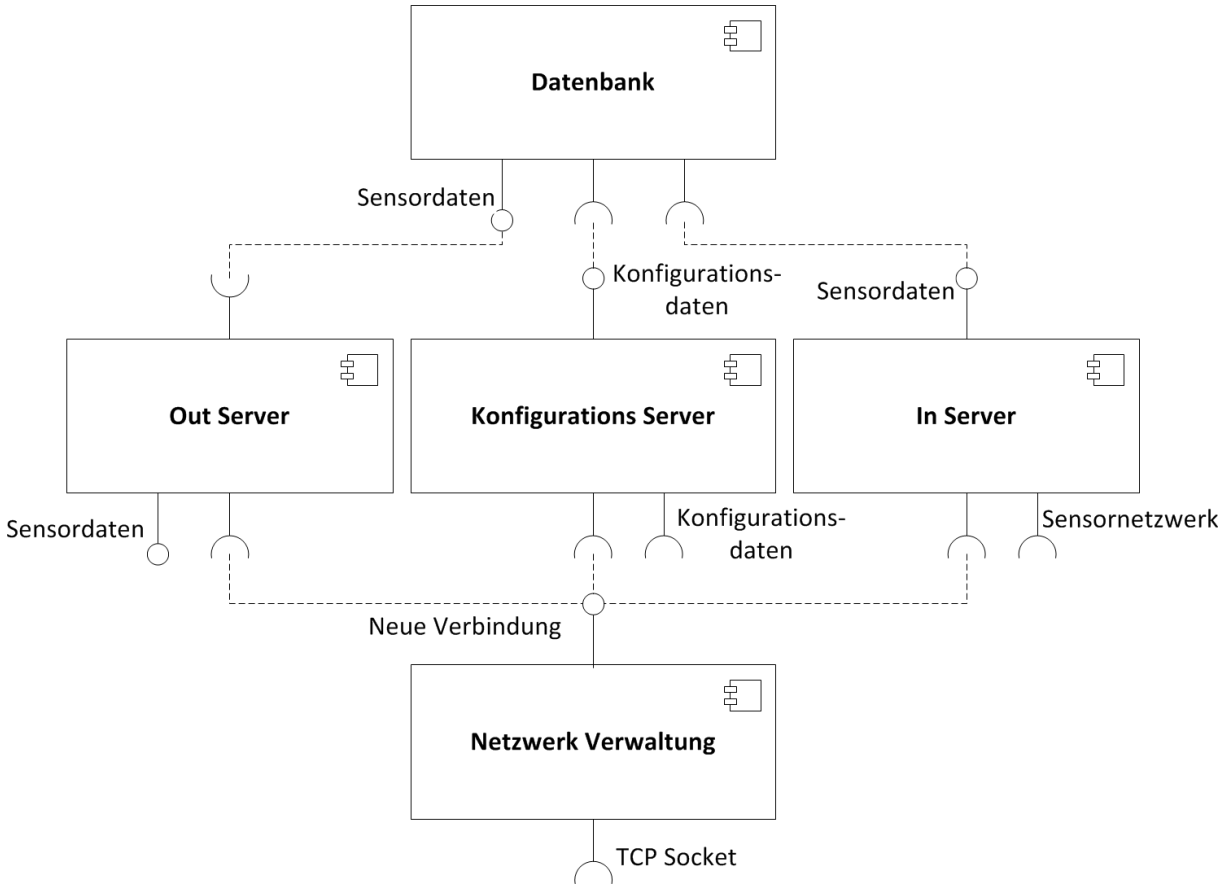

Abbildung 7.7.: Komponenten der PC-Software

Multithreading Die Software nutzt Multithreading soweit wie möglich. Für jede neue Verbindung wird ein eigener Thread verwendet. Jedes Modul, wie zum Beispiel die Ansteuerung der Datenbank, erhält ebenfalls einen eigenen Thread. Der Multiprozessorentwicklung der letzten Jahre wird so Rechnung getragen. Zur Vermeidung von Deadlocks und dem Einbau von Wartezyklen durch Semaphoren und Mutexen erfolgt die gesamte threadübergreifende Kommunikation per Messagepassing. Die Kommunikation läuft dabei immer asynchron ab. Es gibt keinen Thread, der aktiv auf Nachrichten von einem oder mehreren anderen Threads wartet.

Datenspeicherung Zur Speicherung der Messwerte empfiehlt sich der Einsatz einer Datenbank. Die dokumentenorientierte Datenbank Mongo DB, welche im Living Place zur Speicherung aller über den Active MQ versendeten Nachrichten genutzt wird, eignet sich nicht (siehe [Otto und Voskuhl, 2011, S. 8ff]). Durch den dokumentenorientierten Aufbau ist die Suche innerhalb von gespeicherten Datensätzen nicht praktikabel. Relationale Datenbanken ermöglichen eine schnellere Suche nach einzelnen Werten in einem Datensatz.

Auf die Datenbank selbst muss nur das Sensornetzwerk zugreifen können. Abfragen geschehen dann über das Sensornetzwerk. Um den Installationsaufwand gering zu halten, wird die serverlose Datenbank SQLite verwendet. SQLite muss nicht installiert werden, auch ist die Datensicherung sehr leicht, da lediglich eine Datei gesichert werden muss[2].

[2] Ein Einstieg in die Funktionsweise gibt die Dokumentation von SQLite: http://sqlite.org/docs.html - Abruf: 07.02.2012

8. Test

In diesem Kapitel wird beschrieben, wie einzelne Hard- und Softwarekomponenten während der Entwicklung und am Ende das gesamts System getestet wurden.

8.1. Funktionstest der Hardware

Jedes Hardwaremodul muss vor der Inbetriebnahme getestet werden. Die Tests werden in zwei Kategorien unterteilt. Die erste Kategorie sind *Funktionstests*. Diese Test diese Tests müssen mit jeder Platine durchgeführt werden. Mit diesen Tests wird sichergestellt, dass keine Fehler während des Aufbaus entstanden sind. Mögliche Ursachen hierfür sind beim Löten entstandene Kurzschlüsse, schlecht gelötete Bauteile oder auch defekte Bauteile.

Die zweite Art von Tests soll sicherstellen, dass während des Schaltplan- und Leiterplattenentwurfs keine Fehler entstanden sind. Diese Tests sind so genannte *Bring-Up Tests*. Ein Beispiel ist die falsche Platzierung der Antenne, welche die Funkreichweite beeinträchtigen könnte. Da diese Fehler bei allen oder keiner Platine eines Typs auftreten, muss bei diesen Tests nur jeweils eine Platine getestet werden. Zur Sicherheit werden bei diesen Tests zwei Platinen untersucht.

Es wird zunächst die korrekte Funktionalität der Masterknoten getestet. Die durchgeführten Tests der Sensorknoten und deren Ergebnisse werden anschließend beschrieben.

Jede Platine wird vor Inbetriebnahme optisch auf Kurzschlüsse und falsch eingesetzte Bauteile überprüft. Dieser Test gilt als obligatorisch und wird nicht weiter beschrieben. So kam es beim Bestücken eines Masterknotens zu mehreren Kurzschlüssen, welche direkt nach dem Reflowvorgang entfernt wurden (siehe Kap. 6.1.2).

8.1.1. Test der Masterknoten

Die Masterknoten besitzen eine Reihe von zu testenden Komponenten. Getestet werden muss die korrekte Funktion des Spannungsregler (siehe Kap. 6.1.1)s, des Mikrocontrollers (siehe Kap. 6.1.1), die Funktionalität des USB-Seriell-Wandlers (siehe Kap. 6.1.1) und die Funktion der *LEDs* (angeschlossen am Mikrocontroller und USB-Seriell-Wandler). Zuletzt muss die Funktion der Antenne und deren Reichweite überprüft werden. Der Bring-Up Test zur Ermittlung der Funkreichweite wird zusammen mit den Sensorknoten durchgeführt. Der Test und dessen Ergebnisse befinden sich in Kapitel 8.1.3.

8.1.1.1. Spannungsversorgung

Die Anpassung der Eingangspannung auf 3,3 Volt, welcher der ATMega128RFA1 benötigt, wird auf dem Masterknoten von einem LF33C Spannungsregler übernommen. Laut Datenblatt (siehe [STMicroelectronics, 2010, S. 17]) darf der Spannungsregler bei 50 mA und 25° eine Abweichung von 66 mV haben.

Ausgangsspannung Die Ausgangsspannung soll 3,300±0,066 Volt betragen. In einem Funktionstest muss diese Spannung bei jedem Knoten gemessen werden. Bei einem nicht entdecktem Kurzschluss zwischen Masse und geregelter Spannung könnte sich der Spannungsregler überhitzen und beschädigt werden.

Die Spannung wurde mit einem Multimeter[1] gemessen. Bei den drei Masterknoten wurden die Spannungen 3,297 Volt, 3,298 Volt und 3,312 Volt gemessen. Diese Abweichungen liegen unterhalb der Messgenauigkeit des Multimeters und innerhalb der maximalen Abweichung. Alle Spannungsregler der Masterknoten arbeiten korrekt.

Stabilität der Ausgangsspannung Es kann passieren, dass die Ausgangsspannung des Spannungsreglers schwingt. Dies passiert vor allem dann, wenn der Kondensator am Ausgang des Spannungsreglers einen zu kleinen Innenwiderstand besitzt. Da die Größe des Innenwiderstands bauartbedingt ist, tritt der Effekt bei keinem oder allen Platinen auf. Dieser Bring-Up Test wird somit nur bei zwei Masterknoten durchgeführt. Bei dieser Messung kommt es nicht auf die exakte Spannung an, sondern viel mehr, ob diese sich im Verlauf der Zeit ändert. Für die Messung wird ein Oszilloskop verwendet, welches eine mögliche Spannungsänderung vom Nanosekunden- bis Sekundenbereich aufzeigen kann.

Die folgende Abbildung 8.1 zeigt die gemessene Spannung an einem Masterknoten. Die Spannung ist stabil und ändert sich im Verlauf der Zeit nicht. Die Grafik zeigt die Messung mit einer horizontalen Auflösung von 10 ms pro Unterteilung. Insgesamt wurden an zwei Masterknoten Messungen in einem Bereich zwischen 20 ns und 1 s pro Unterteilung durchgeführt. Bei allen Messungen blieb die Spannung stabil. Der Spannungsregler arbeitet somit korrekt und der Ausgangskondensator wurde richtig dimensioniert.

[1] Multimeter: VC160, maximale Abweichung der Messung: 0,8% + 1dgt. Dies entspricht maximal 28 mV Abweichung bei 3,300 Volt.

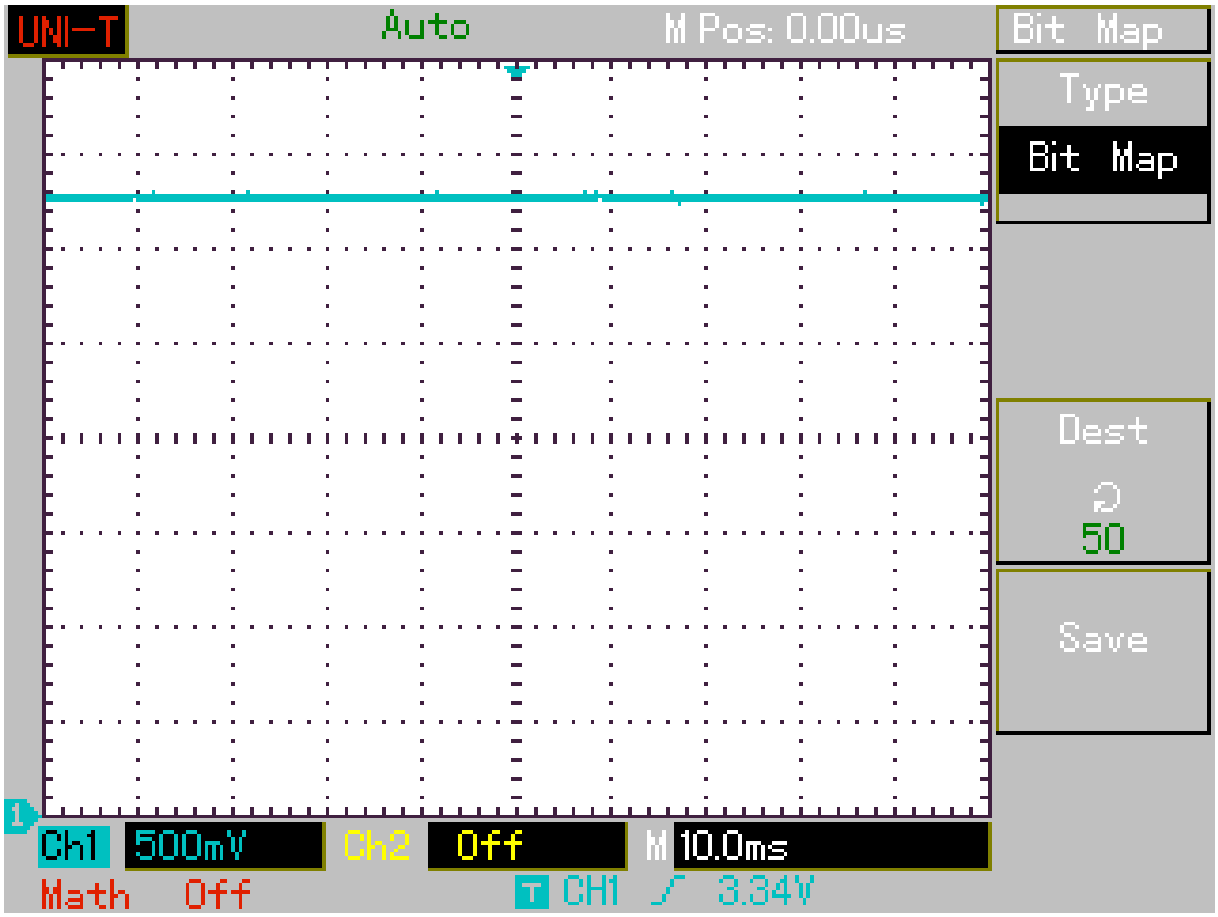

Abbildung 8.1.: Messung der Spannungsversorgung des Masterknotens

8.1.1.2. Test der verbauten Komponenten

Mit Hilfe einer Testsoftware können der Mikrocontroller, der USB-Seriell-Wandler und die LEDs getestet werden. Da es sich bei allen drei Komponenten um Bauteile handelt, welche bei falscher Handhabung während des Lötvorgangs leicht zerstört werden können, ist dies ein Funktionstest und muss mit allen Masterknoten durchgeführt werden.

Testsoftware Bei der Testsoftware handelt es sich um die normale Software des Masterknotens, lediglich vor Beginn des eigentlichen Programms werden einmalig die drei LEDs vom Masterknoten für eine Sekunde aktiviert und danach wieder deaktiviert. Anschließend erfolgt der eigentliche Start der Software. Die Software des Masterknotens kommuniziert bereits während der Initialisierung über den USB-Seriell-Wandler mit dem PC, was den USB-Seriell-Wandler und die LEDs am USB-Seriell-Wandler, welche bei aktiver Kommunikation blinken, testet.

Bei allen Masterknoten leuchteten die drei an den Mikrocontroller angeschlossenen LEDs und alle gaben beim Start die Meldung *Network start successful* über die PC-Software aus, während die LEDs an USB-Seriell-Wandler blinkten.

8.1.2. Sensorknoten

Die Sensorknoten besitzen genauso wie die Masterknoten eine Reihe von Komponenten, welche in Funktionstests getestet werden müssen. So verfügen die zur Batterieversorgung bestimmten Sensorknoten über einen *Step-Up Wandler* (siehe Kap. 6.2.1.5). Für alle als Router arbeitenden Sensorknoten existieren zusätzliche *Step-Down Wandler Platinen* (siehe Kap. 6.2.2). Beide müssen unabhängig voneinander überprüft werden.

In einem Bring-Up Tests wird zusätzlich analysiert, was geschiet, wenn beide Spannungswandler durch einen Fehler beim Aufbau hintereinander geschaltet werden.

Nachdem die korrekte Funktion der Spannungsversorgung überprüft ist, können die restlichen Komponenten wie das *ZigBit-Modul* (siehe Kap. 6.2.3) und die zwei an das Modul angeschlossenen *LEDs* mit einem Funktionstest überprüft werden. Das ZigBit-Modul selbst beherbergt einen Mikrocontroller, einen RF-Chip und zwei Antennen. Bei jedem Mikrocontroller und bei jedem RF-Chip muss überprüft werden, ob diese während des Lötvorgangs beschädigt wurden.

Die *Funkreichweite* wird durch einen Bring-Up Test in Kapitel 8.1.3 ermittelt.

8.1.2.1. Testen des Step-Up Wandlers

Alle mit Batterien versorgten Sensorknoten besitzen einen Step-Up Wandler (siehe Kap. 6.2.1.5), dessen Funktion getestet werden muss. Hierbei werden zwei Testfälle unterschieden. Zum einen muss während eines Funktionstest untersucht werden, ob jeder einzelne Step-Up Wandler funktioniert. Zum anderem ist zu untersuchen, ob der Spannungswandler in der Lage ist, die Ausgangsspannung unter allen Betriebsbedingungen einzuhalten. Dies ist abhängig von der korrekten Wahl der Bauteile für die Außenbeschaltung. Da alle Spannungswandler gleich beschaltet sind, wird hierfür ein Bring-Up Test an zwei Sensorknoten durchgeführt.

Alle Tests bezüglich der Step-Up Wandler werden durchgeführt, bevor die restlichen Bauteile auf der Platine bestückt wurden. So kann sichergestellt werden, dass ein falsch arbeitender Step-Up Wandler kein ZigBit-Modul beschädigt.

Einhalten der Ausgangsspannung Zunächst muss überprüft werden, ob der Spannungswandler eine Ausgangsspannung von 3,3 Volt erzeugt. Mit Hilfe eines regelbaren Netzteils wird eine Spannung von 2,0 Volt angelegt. Dieser Funktionstest wird mit jedem Sensorknoten, der über einen Step-Up Wandler verfügt durchgeführt. Dies zeigt, ob der Step-Up Wandler grundsätzlich korrekt arbeitet. Als Last werden ein bzw. zwei 150 Ohm Widerstände angeschlossen. Ein 150 Ohm Widerstand besitzt bei 3,3 Volt einen Stromverbrauch von 22 mA. Zwei parallel angeschlossene 150 Ohm Widerstände haben 44 mA Stromverbrauch. 22 mA entsprechen in etwa dem Stromverbrauch des ZigBit-Modul und 2 aktiven Low-Current LEDs. 44 mA liegen etwas über der Anforderung, zusätzlich zu dem ZigBit-Modul noch einen Sensor mit 20 mA Strom beliefern zu können (siehe Anforderung 2.8 sowie Kap. 6.2.1.1). Damit die ZigBit-Module während des

Betriebes nicht beschädigt werden und korrekt arbeiten können, muss eine Spannung im Bereich von 2,7 bis 3,6 Volt eingehalten werden.

Stromverbrauch des Widerstandes: $U = R * I \Rightarrow \frac{U}{R} = I \Rightarrow \frac{3{,}3V}{150Ohm} = \underline{\underline{0{,}02A}}$

Alle mit dem Multimeter[2] gemessenen Ausgangsspannungen der Step-Up Wandler lagen im Bereich zwischen 3,290 und 3,328 Volt. Die Abweichung zu 3,300 Volt beträgt maximal 0,029 Volt und somit weniger als 1%. Diese Genauigkeit ist mehr als ausreichend. An den einzelnen Knoten wurde beim Wechsel von 22 auf 44 mA eine Spannungsabsenkung um 1 bis 2 mV festgestellt. Dieser Wert ist sehr gering. Bei Laständerung könnte eine Spannungsänderung von 100 mV akzeptiert werden, so lange der Bereich von 2,7 bis 3,6 Volt nicht verlassen wird.

Verhältnis zwischen Aus- und Eingangsspannung Der Spannungswandler muss die Ausgangsspannung über die gesamte Lebensdauer der Batterien halten können. Die Spannung muss dabei sowohl bei aktivem Sensorknoten, als auch bei einem schlafendem Sensorknoten stabil bleiben. In diesem Abschnitt wird an zwei Sensorknoten ein Bring-Up Test durchgeführt. Es wird dabei untersucht, ob die 3,3 Volt Ausgangsspannung unter Last und im Leerlauf konstant gehalten werden können. Die Eingangsspannung wird im Bereich von 1,4 bis 3,5 Volt in 0,1 Volt Schritten erhöht. Die Eingangsspannung wird auf bis zu 3,5 Volt angehoben, damit erkennbar ist, wie sich die Step-Up Wandler bei einer erhöhten Eingangsspannung verhalten. So kann es vorkommen, dass ein Sensorknoten mit eigenem Step-Up Wandler extern mit Strom versorgt wird und hierfür einer Step-Down Wandler (siehe Kap. 6.2.2) mit einer Ausgangsspannung von 3,3 Volt verwendet wird. Die Spannung wird im Leerlauf und mit einem 150 Ohm Widerstand am Ausgang des Spannungswandlers gemessen.

Im folgenden Diagramm 8.2 sind die Eingangs- und Ausgangspannung des Step-Up Wandlers gegenüber gestellt. Zur besseren Übersicht befinden sich nur die Messwerte eines Knotens in der Grafik. Die Messwerte beider gemessenen Knoten können der Tabelle A.1 im Anhang A.1 entnommen werden.

[2]Multimeter: VC160, maximale Abweichung der Messung: 0,8% + 1dgt. Dies entspricht maximal 28 mV Abweichung bei 3,300 Volt.

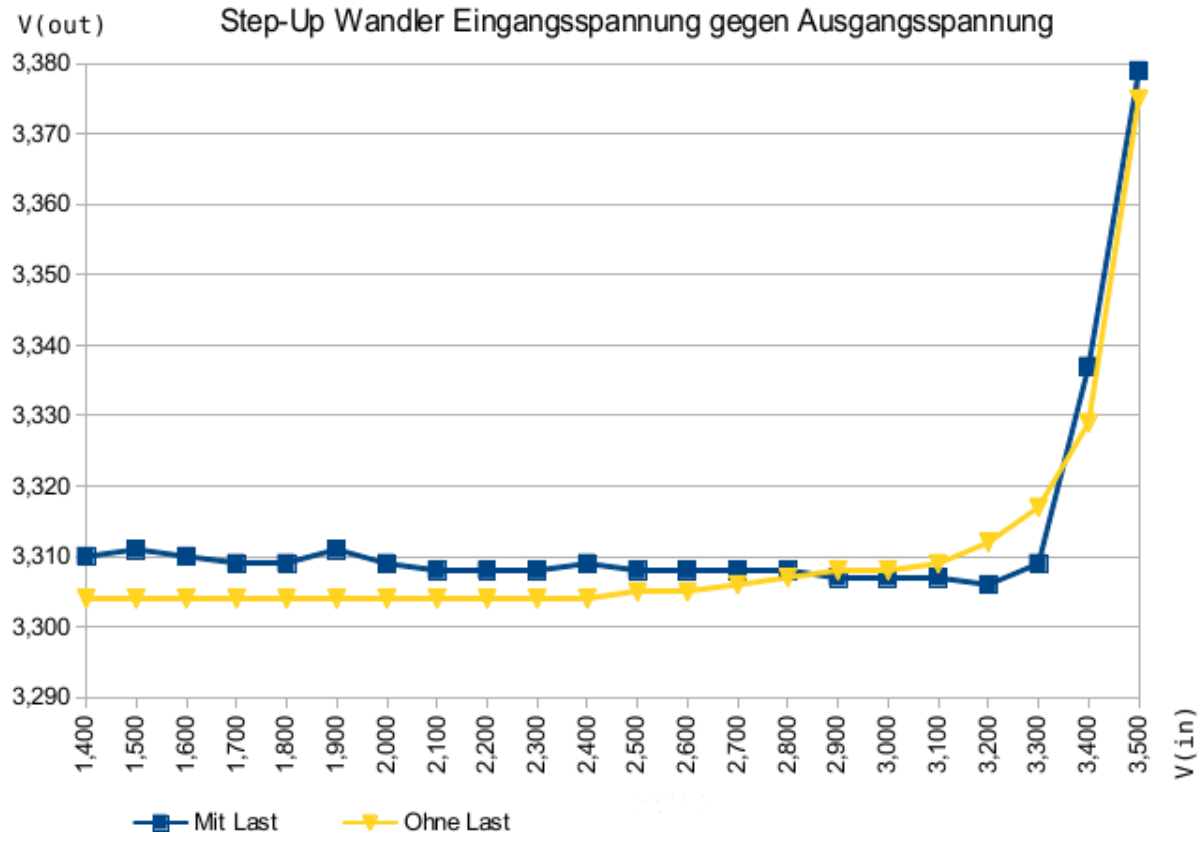

Abbildung 8.2.: Step-Up Wandler Eingangsspannung gegen Ausgangsspannung

Die Ausgangsspannung bleibt unabhängig von der Belastung im Bereich von 1,4 bis 3,1 Volt Eingangsspannung nahezu konstant. Die Differenz zwischen dem höchsten Wert unter Last und dem niedrigstem Wert ohne Last liegt bei gerade einmal 7 Millivolt oder umgerechnet 2,1 Promille und weit unter der Messgenauigkeit des eingesetzten Multimeters[3]. Der Step-Up Wandler hält die Ausgangsspannung über den gesamten Betriebsbereich der Batterien.

Im Bereich von 3,2 bis 3,5 Volt steigt die Ausgangsspannung langsam an. Dass die Ausgangsspannung bei 3,5 Volt Eingangsspannung nur 3,4 Volt erreicht liegt an Verlusten, welche im Step-Up Wandler auftreten. Die Ausgangsspannung bleibt unter 3,6 Volt, womit das ZigBit-Modul keinen Schaden nimmt (vgl. [MeshNetics, 2007, S.5]). Sensorknoten mit Step-Up Wandler werden somit korrekt arbeiten, wenn sie extern mit Strom aus einem Step-Down Wandler versorgt werden. Es müssen keine Vorkehrungen gegen einen falschen Aufbau vorgenommen werden.

Stabilität der Ausgangsspannung Genauso wie bei Spannungsreglern kann es bei Spannungswandlern zur Schwingung der Ausgangsspannung kommen, wenn die Bauteile zur Beschaltung des Step-Up Wandlers falsch dimensioniert sind. Im Folgenden wird die Stabilität der Ausgangsspannung überprüft. Der Test wird mit einem 150 Ohm

[3]Multimeter: VC160, maximale Abweichung der Messung: 0,8% + 1dgt. Die entspricht maximal 28 mV Abweichung bei 3,300 Volt.

Widerstand als Last und ohne Last durchgeführt. Die Eingangspannung wird im Bereich von 1,4 bis 3,5 Volt variiert.

Das folgende Bildschirmfoto 8.3 vom Oziloskop zeigt die Ausgangsspannung des Step-Up Wandlers bei 2,0 Volt Eingangsspannung und mit 150 Ohm Last. Wie auf der Aufnahme zu sehen ist, schwingt die Spannung nicht. Abgesehen von der Erhöhung der Ausgangsspannung ab 3,2 Volt Eingangsspannung blieb die Spannung während des gesamten Tests stabil. Die Spannung schwingt nicht.

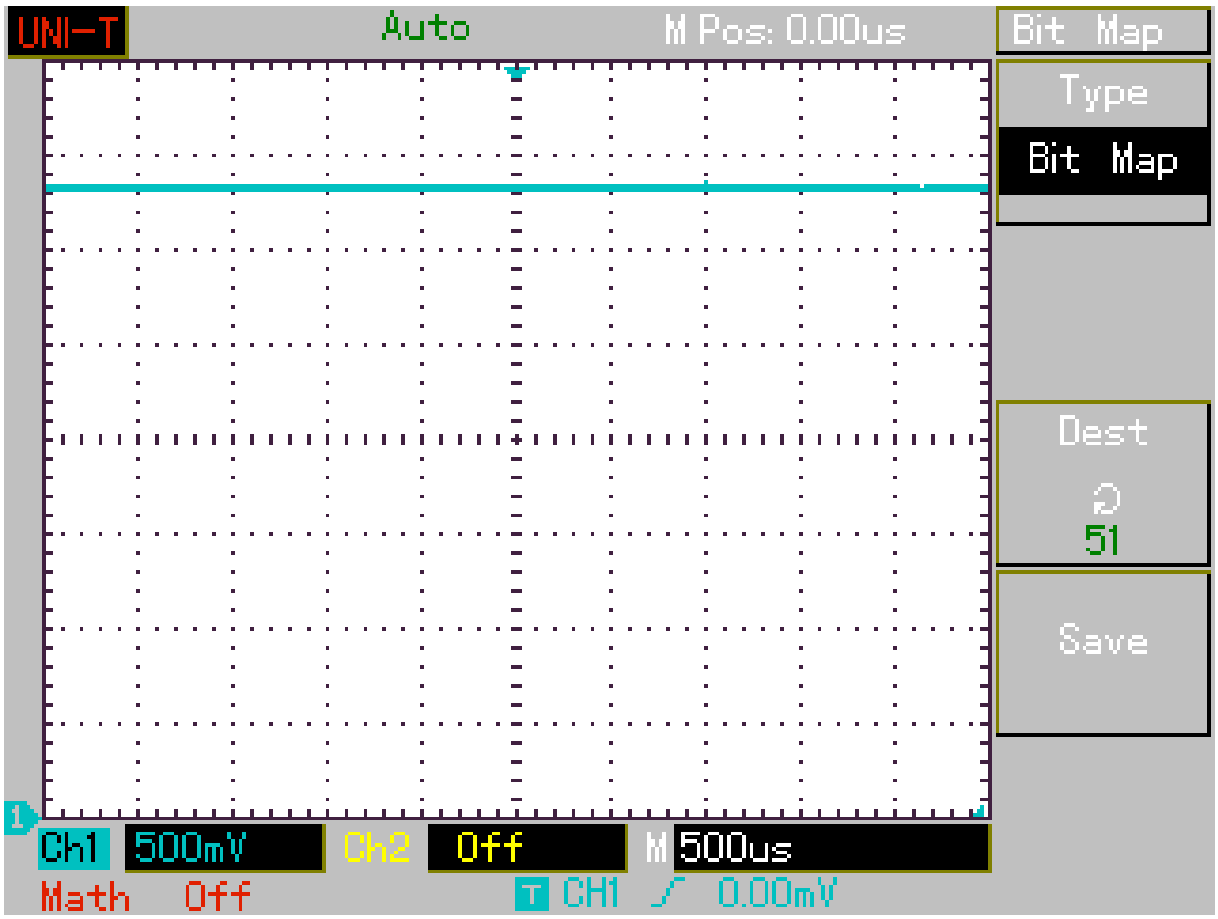

Abbildung 8.3.: Messung der Spannungsversorgung des Sensorknoten

Tests erfolgreich bestanden Alle Tests ergaben, dass die Step-Up Wandler unter den gestellten Bedingungen eine Spannung von 3,3 Volt erzeugen. Die Spannung ist sowohl unter Last, als auch ohne Last stabil und schwingt nicht. Die Step-Up Wandler können ohne Einschränkungen für die Versorgung der Sensorknoten verwendet werden. Zusätzlich müssen keine Vorkehrungen gegen einen falschen Aufbau getroffen werden, wenn die Step-Down Wandler an Sensorknoten mit Step-Up Wandler angeschlossen werden müssen.

8.1.2.2. Testen des Step-Down Wandlers

Für die Sensorknoten wurde eine zusätzliche Platine entworfen (siehe Kap. 6.2.2), die mit bis zu 28 Volt arbeitet und die Eingangsspannung auf 3,3 Volt wandelt. Die Tests dazu werden analog zu den Tests des Step-Up Wandlers (siehe Kap. 8.1.2.1) durchgeführt, nur dass die Eingangsspannung entsprechend höher gewählt wird. Die einzelnen Ergebnisse können im Anhang A.2 eingesehen werden.

Die Tests ergaben, dass die Step-Down Wandler im Bereich von 9 bis 28 Volt eine Spannung von 3,3 Volt erzeugen. Auch wenn die Spannung unter Last etwas stärker als bei den Step-Up Wandlern absinkt, ist die Spannung stabil und schwingt nicht. Die Step-Down Wandler können ohne Einschränkungen für die Versorgung der Sensorknoten verwendet werden.

8.1.2.3. Funktionsüberprüfung der ZigBit-Module

Nachdem sichergestellt wurde, dass die Spannungsversorgung korrekt arbeitet, wurden ZigBit-Module verlötet und werden nun einem Funktionstest unterzogen. Für den Test wird die reguläre Sensorknotensoftware in die Module geladen. Die Software wird als End-Device konfiguriert und die Batterieüberwachung wird aktiviert. Weiterhin wird für jeden IO-Pin der Sensortyp ”Schalter” konfiguriert.

Beim Start der Software leuchtet die Debug LED so lange, bis eine Verbindung zum Masterknoten hergestellt wurde. Somit sind nach dem Start bereits der Prozessor, der RF-Chip und eine der LEDs getestet. Die zweite LED wird getestet, indem die Eingangsspannung unterhalb von 1,6 Volt abgesenkt wird. Dadurch versendet die Batterieüberwachung eine Fehlermeldung, welche zum Batteriewechsel auffordert und die Fehler LED-leuchtet für 10 Sekunden. Anschließend werden an allen IOs 0 Volt bzw. 3,3 Volt angelegt. Die IOs werden als Sensortyp ”Schalter” abgefragt. Der Sensorknoten übermittelt sein Werte mit Hilfe des Masterknotens an den PC, an welchem die Ergebnisse abgelesen werden können.

Der Test ergab, dass alle Sensorknoten erfolgreich aufgebaut wurden. Lediglich bei einem Sensorknoten musste ein Kurzschluss behoben werden, der vor dem Test nicht erkannt wurde.

8.1.3. Ermittlung der maximalen Funkreichweite

Nachdem sowohl Master- und Sensorknoten aufgebaut sind und erfolgreich getestet wurden, wird nun die maximale Funkreichweite ermittelt. Die Funkreichweite ist vor allem vom Platinenlayout und den Umgebungsbedingungen abhängig. Dieser Test wird als Bring-Up test durchgeführt. Für den Test wird ein Sensorknoten und ein Masterknoten verwendet.

Wie bereits in Kapitel 7.2 erwähnt, baut die Software der Sensorknoten auf der Software aus Projekt 2 auf. Wenn die Funktionalität im Sensornetzwerk nicht mehr verwendet

wird, verfügen die Sensorknoten noch immer über eine Nachricht, über die ein Ping versendet werden kann. Dies wird für diesen Test genutzt. Der Test erfolgt genauso wie die Messung der Pingzeiten in Projekt 2:
”Für diesen Test werden zwei Module im Abstand von 20 cm nebeneinander platziert. Abbildung 8.4 stellt den Aufbau schematisch dar. Der Coordinator versendet ein Paket mit maximaler Größe und wartet auf eine Antwort (ebenfalls maximale Paketgröße). Direkt vor dem Versenden schaltet der Coordinator eine LED ein, bei Empfang der Antwort schaltet er sie wieder ab. Ein parallel zur LED angeschlossenes Oszilloskop wird zur Zeitmessung verwendet. Es werden insgesamt 100 Messungen durchgeführt. Auf Grund technischer Einschränkungen durch das Oszilloskop erfolgen die Messungen mit einer Genauigkeit von 0,2 ms. Es sei darauf hingewiesen, dass die Übertragung der Datenpakete gesichert erfolgt. Es werden also nicht nur die zwei Datenpakete sondern zusätzlich Acknowledge Frames versendet.” (Zitat: [Pautz, 2011b, S.21]).

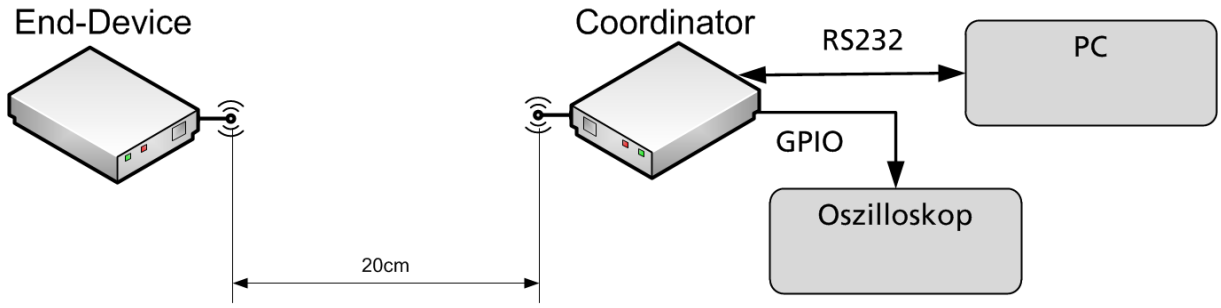

Abbildung 8.4.: Schematischer Aufbau für die Messung der Pingzeiten

Als Abwandlung zu dem Test aus Projekt 2, wird die Entfernung zwischen den Sensorknoten so lange erhöht, bis die Pingzeiten ansteigen oder es gar zu Ausfällen in der Datenübertragung kommt. Ansteigende Pingzeiten sind ein Indiz für eine schlechter werdende Verbindungsqualität. Für diesen Test wird die Software des Sensorknotens so modifiziert, dass dieser in den Messpausen keinen Stromsparmodus einnimmt, da dieser sonst erst beim nächsten Aufwachen antworten könnte. Die Sendeleistung der teilnehmenden Knoten beträgt bei diesem Test 3 db. Der Test wird im Flur eines Gebäudes durchgeführt. Zwischen den beiden Sensorknoten besteht Sichtkontakt.

Bei 28 Metern liegt die durchschnittliche Pingzeit bei 21,8 ms. 66 % der Messungen liegen in einem Bereich von 0,6 ms um den Durchschnitt. Bei Entfernungen unter 28 Metern singt die Pingzeit nicht weiter ab. Diese Messung deckt sich in etwa mit den Messergebnissen aus Projekt 2. Dort betrug die durchschnittliche Pingzeit bei 0,20 und 13,5 Metern Entfernung jeweils 22,4 ms (vgl. [Pautz, 2011b, S. 22]). Bei Entfernung zwischen 29 und 32 Metern steigt die durchschnittliche Pingzeit langsam an. Bei 32 Metern hat der Wert 25,8 ms erreicht. 66 % der Messungen liegen dabei in einem Bereich von ca. 3,4 ms um den Durchschnitt. Gerade die größere Streuung um den Durchschnitt zeigt, dass es hier zu einer erhöhten Anzahl an Sendewiederholung gekommen ist und die Verbindung bereits geschwächt ist. Trotz der automatischen Übertragungswiederholung von ZigBee gab es bei 33 Metern einen Verlust von 3 Paketen bei 100 Messungen. Bei 35 Metern konnte der Sensorknoten sich nicht mehr

zuverlässig beim Masterknoten anmelden. Es wurden keine Messungen durchgeführt. Die ermittelten Pingzeiten der jeweils 100 Messungen für die Entfernungen von 28, 32 und 33 Metern befindet sich im Anhang A.3.

Für das Living Place Hamburg (siehe Kap. 1.3) und die meisten heute existierenden Wohnungen ist die Funkreichweite von 28 Metern ausreichend. Vor allem da ZigBee ein Meshnetzwerk ist und die überbrückbare Reichweite durch Router erheblich vergrößert werden kann.

8.1.4. Messen der Stromaufnahme

Nachdem die Sensormodule fertig aufgebaut sind, und deren Funktion erfolgreich getestet wurde, wird nun die Stromaufnahme der Sensorknoten ermittelt. Hierfür wird zunächst die zur Verfügung stehende Leistung errechnet und anschließend der tatsächliche Stromverbrauch während der Wach- und Ruhephasen in einem Bring-Up Test ermittelt. Danach wird die Dauer der Wachphasen gemessen, wodurch abschließend errechnet werden kann, wie oft die Sensoren abgefragt werden können.

8.1.4.1. Anforderung und verfügbare Leistung

Laut der Anforderung Geringer Wartungsaufwand sollen die Sensorknoten mindestens ein Jahr ohne Batteriewechsel arbeiten. Dabei müssen mindestens alle 5 Minuten drei Sensoren abgefragt werden. Da diese Anforderung nicht durch einen ein Jahr andauernden Test überprüft werden kann, wird die maximale Laufzeit rechnerisch ermittelt.

Für die Versorgung der Sensorknoten stehen zwei Mignon Batterien (siehe 6.2.1.5) bereit. Energizer gibt für seine Batterien bei 25 mA Entladestrom 2800 mAh Kapazität an (siehe [Energizer Holdings Inc., 2008, S. 1]. Dabei steigt die Kapazität, je geringer der Entladestrom ist. Den größten Teil der Zeit befinden sich die Sensorknoten im Schlafmodus, den Batterien werden dann nur wenige μA entnommen. Während der Wachphasen werden den Batterien für einige Zehntelmillisekunden maximal 40 mA entnommen (vgl. Kap. 6.2.1.1). Welche Kapazität die Batterien bei dieser wechselnden Belastung genau aufweist lässt sich aus dem Datenblatt der Batterien nicht ermitteln. Für die folgenden Berechnungen wird der Vergleichswert von 2800 mAh verwendet.

Für die Ermittlung der durchschnittlichen Batteriespannung wird der Durchschnitt aus der Entladeschlussspannung und der Spannung zweier voller Batterien ermittelt:
$(2 * 1,5Volt + 2 * 0,9Volt)/2 = 2,4Volt$

Bei 2800 mAh und einer durchschnittlichen Spannung 2,4 Volt besitzen die Batterien:
$2800mAh * 2,4V = 6720mWh$

Nachdem die Gesamtkapazität bekannt ist, kann nun berechnet werden, wieviel Strom die Sensorknoten pro Stunde verbrauchen dürfen. Dafür wird die geforderte Laufzeit von einem Jahr angesetzt:
$6720mWh/(365 * 24h) = 0.767mW$

Den Sensorknoten stehen ca. 0,77 mW pro Stunde zur Verfügung.

8.1.4.2. Stromverbrauch der Sensorknoten in der Schlafphase

Das Diagramm 8.5 zeigt die gemessene Ruhestromaufnahme (blaub) eines als End-Device konfigurierten Sensorknotens. Die Eingangsspannung wurde dabei von 1,6 auf 3,0 Volt angehoben. Ebenso ist in der Tabelle die Effizienz (orange) angegeben. Im Datenblatt ist die Effizienz des Spannungswandlers bei diesen niedrigen Strömen nicht angegeben.

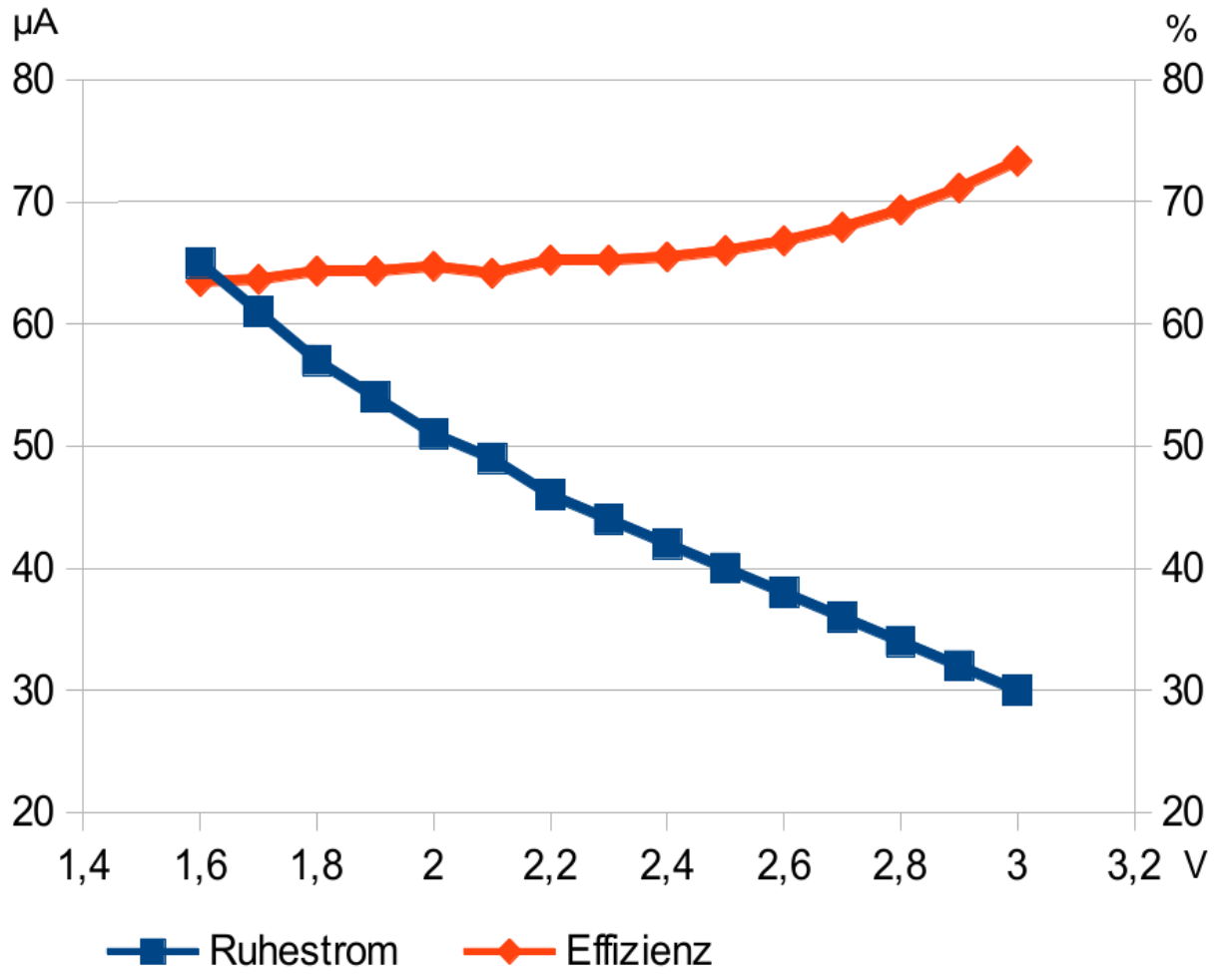

Abbildung 8.5.: Eingangsspannung gegen Ruhestromaufnahme

Durchschnittlich beträgt die Leistungsaufnahme 99,7 μW im Ruhemodus. Bei einem Jahr Batterielaufzeit und der verfügbaren Leistung von 770 μW pro Stunde verbraucht der Ruhestrom knapp 100 μW und es bleiben 670 μW pro Stunde für die Wachphasen, in denen die Sensoren abgefragt werden.

Die Effizienz wurde ermittelt, in dem die Ruhestromaufnahme bei 3,3 Volt eines Sensorknoten ohne Step-Up Wandler gemessen wurden. In diesem Fall wurden 20 μA Stromaufnahme gemessen. Danach wurde die Effizienz nach folgender Formel errechnet:
$[(3,3Volt * 20\mu A)/(Eingangsspannung * Ruhestrom)] * 100\% = Effizienz$

8.1.4.3. Stromverbrauch der Wachphasen

Für diesen Test wird ein Sensorknoten als Router konfiguriert und dessen Stromverbrauch ermittelt. Der Stromverbrauch eines End-Devices kann nicht direkt ermittelt werden, da die Wachphasen zu kurz sind, als dass sie von einem Amperemeter sicher erfasst werden können.

Der Routerknoten fragt während des Tests keine Sensoren ab. Die in Strom- und Spannungsversorgung der Sensoren geforderten 20 mA Stromaufnahme für einen Sensor werden durch einen 150 Ohm Widerstand simuliert. An diesem fallen bei 3,3 Volt 22 mA ab.

Das Diagramm 8.6 zeigt die gemessene Stromaufnahme (blau) des Routerknotens. Die Eingangsspannung wurde dabei wieder von 1,6 auf 3,0 Volt angehoben. In der Tabelle ebenfalls wieder die Effizient (orange) angegeben. Die Effizienz wurde wie zuvor ermittelt. Dabei liegt die Stromaufnahme ohne Step-Up Wandler bei 43,0 mA.

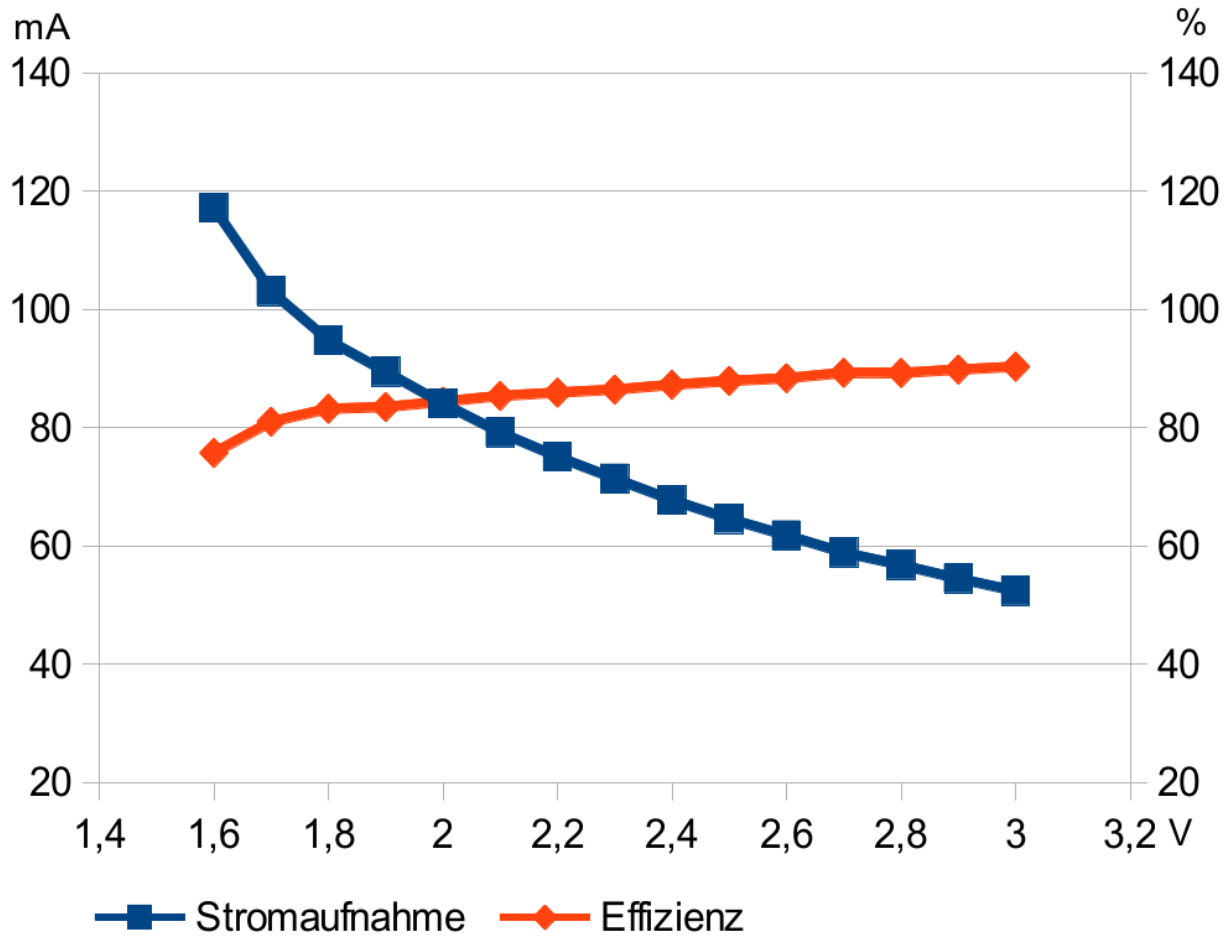

Abbildung 8.6.: Eingangsspannung gegen Stromaufnahme während der Wachphase

Die genauen Messwerte können in Tabelle A.7 in Anhang A.4 eingesehen werden. Durchschnittlich beträgt die aufgenommene Leistung 165,7 mW.

8.1.4.4. Dauer der Wachphasen

Um zu bestimmen, wie viel Strom die Sensorknoten zum Auslesen benötigen, muss nicht nur die Stromstärke, sondern auch die nötige Zeit zum Auslesen ermittelt werden.

Diese Zeit kann jedoch nicht über interne Funktionen des ZigBit-Moduls ermittelt werden. Nachdem die CPU aus dem Schlafmodus erwacht, meldet der BitCloud-Stack sich zunächst im ZigBee-Netzwerk, bevor die erste vom Nutzer zugreifbare Funktion aufgerufen wird.

Die Zeit, die ein Sensorknoten benötigt, um einen Sensor auszulesen wird von außerhalb ermittelt. Dazu wird ein 5 Ohm Widerstand zwischen der Spannungsversorgung und dem Sensorknoten eingesetzt. Wacht der Sensorknoten auf, benötigt er mehr Strom, dieser Strom fließt auch über den Widerstand, an dem gemäß des Ohmschen Gesetzes eine höhere Spannung abfällt, je höher der Stromfluss ist. Dieser Spannungsabfall wird mit einem Oszilloskop gemessen. Die Grafik 8.7 zeigt eine Skizze des Aufbaus.

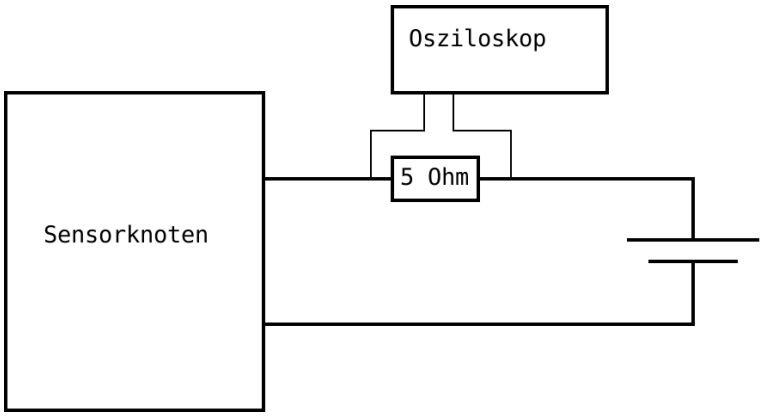

Abbildung 8.7.: Aufbau: Messung der Dauer einer Wachphase

Bei diesem Test wird ein DS18S20 Temperatur Sensor ausgelesen (siehe Maxim [2010] und Anhang G). Von allen implementierten Sensoren dauert das Auslesen dieses Sensors am längsten.

Die Aufnahme des Oszilloskops 8.8 zeigt für 40,4 ms eine Erhöhung des Spannungsabfalls am Widerstand.

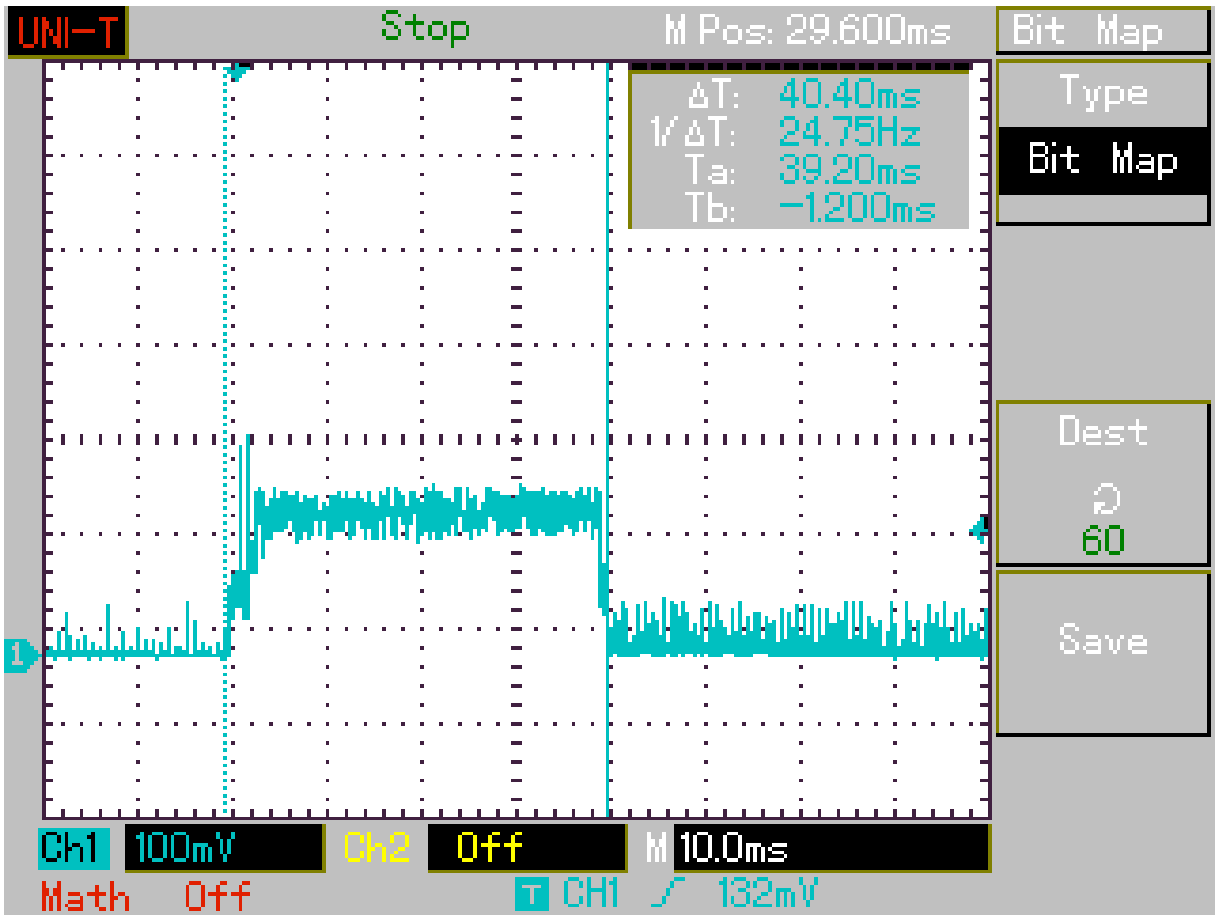

Abbildung 8.8.: Messung der Dauer einer Wachphase

8.1.4.5. Berechnung der maximalen Anzahl an Messungen

Es wurden alle Parameter, welche zur Abschätzung der maximalen Anzahl an Messungen pro Stunde erforderlich sind, gemessen bzw. errechnet.

Die Messung der Ruhestromaufnahme ergab knapp 100 μW. Bei den 770 μW, die pro Stunde zur Verfügung stehen, bleiben 670 μW für die Wachphasen. Während der Wachphasen werden 165,7 mW Leistung benötigt. Eine Wachphase dauert dabei höchstens 40,4 ms. Daraus ergibt sich die Leistungsaufnahme pro Wachphase wie folgt:

$165,7mW * 40,4ms = 165,7mW * 0,0404s = 165,7mW * 1,1222 * 10^{-5}h = 1,86\mu Wh$

Dem gegenüber stehen 670 μWh zur Verfügung. Die Anzahl der möglichen Messungen pro Stunde ergibt sich wie folgt:

$670\mu W/1,86\mu Wh = 360 * \frac{(}{1}]h/h$

In der Anforderung ”Geringer Wartungsaufwand” ist gefordert, dass drei Sensoren alle fünf Minuten abgefragt werden:

$3 * (60/5) = 36$

Die Sensorknoten können also mit 360 Messungen zehn mal so häufig abgefragt werden, wie gefordert. Zudem wurde die gesamte Messung sehr pessimistisch betrachtet. Es wurde mit einer Stromaufnahme von 20 mA für die Sensoren gerechnet. Dies ist in den Anforderungen festgelegt, die meisten Sensoren benötigen jedoch wesentlich weniger Strom. Der DS18S20 zum Beispiel benötigt maximal 1,5 mA (siehe [Maxim, 2010, S. 20]). Zudem wurde der Sensor gemessen, dessen Einlesevorgang am längsten dauert. In der Realität kann so nahezu die doppelte Anzahl an Messungen pro Stunde vorgenommen werden. Werden an Stelle von Alkaline Lithium Batterien verwendet, würde die zur Verfügung stehende Leistung weiter steigen, wodurch noch mehr Messungen durchgeführt werden können.

8.2. Softwaretest

Die Softwaretests sind in drei Kategorien eingeteilt. Bereits während der Entwicklung werden sogenannte *Modultests* durchgeführt. Nach erfolgreichem Abschluss der Modultests folgt der *Integrationstest* und zum Abschluss wird das Gesamtsystem einem *Systemtest* unterzogen.

8.2.1. Modultest

Beim Modultest werden einzelne abgeschlossene Module bzw. Komponenten getestet. Die kleinste abgeschlossene Einheit, welche einzeln getestet werden kann, ist eine Funktion. Es können auch mehrere Funktionen zusammengefasst und als Klasse (C++ / Java) oder Modul (C) getestet werden.

Module, welche keine Funktionalität der Zielhardware erfordern, werden auf dem PC getestet. So wurde ein Teil der Software für den Masterknoten und die ZigBit-Module auf dem PC getestet. Dies erspart das Einspielen der Software auf die Controller und zu dem können die Ergebnisse auf dem PC besser dargestellt werden.

8.2.1.1. Beispiel eines Modultests

Anhand eines Beispiel wird gezeigt, wie alle Module getestet wurden.

Zu testen ist die Funktion *debug_assemble_msg*, der Sensorknoten[4]. Ziel dieser Funktion ist es, eine Fehler- oder Debugnachricht entgegen zu nehmen und sie so aufzubereiten, dass sie über das ZigBit-Netzwerk gesendet werden und mit der PC-Software für den Benutzer sichtbar ausgegeben werden kann.

Die Funktion muss die Art der Nachricht über einen ”D:” für Debug oder ”E:” für Error (engl. für Fehler) kennzeichnen. Zusätzlich muss die Funktion sicherstellen, dass die Länge der Nachricht die Länge eines ZigBit-Telegramms nicht überschreitet. Die Vorbedingungen sind, dass keiner der Eingabeparamter ’NULL’ ist und die Eingabezeichenkette null terminierend ist.

Der folgende Quelltext 8.9 zeigt die zu testende Funktion *debug_assemble_msg* und eine Funktion, welche zwei Tests durchführt. Die maximale Länge des Ergebnisses wurde für eine bessere Übersicht in diesem Beispiel absichtlich reduziert.

```
1  #define MAX_LEN 12
2  #define DEBUG_MSG 0
3  #define DEBUG_ERR 1
4
5  /*!
6      \brief Setzt eine Nachricht so zusammen, dass sie ueber das
           Netzwerk versendet werden kann.
7      \details Die Dokumentation wurde zur besseren Uebersicht an dieser
           Stelle gekuerzt.
8  */
9
10 static uint8_t debug_assemble_msg(uint8_t * result,
11             uint8_t result_len, uint8_t * msg,
12             uint8_t level)
13 {
14     //Schwere der Nachricht
15     if(level == DEBUG_ERR)
16         result[0] = 'E';
17     else
18         result[0] = 'D';
19
20     result[1] = ':';
```

[4]Die Funktion befindet sich in der Datei ”debug.c”. Die Datei kann auf der beigelegten DVD eingesehen werden. Siehe dazu G

```
21      result[2] = '\0';
22      //Nachricht anhaengen
23      strncat((char*)result, (char*) msg, result_len);
24      //Laenge der Nachricht beschraenken
25      result[result_len -1] = '\0';
26
27      //Laenge als Ergebnis zurueckliefern
28      return strlen((char*)result);
29  }
30
31  void test()
32  {
33      uint8_t result_len;
34      uint8_t msg[50] = "Fehler!";
35      uint8_t msg2[50] = "Debugmeldung!";
36      uint8_t result[MAX_LEN];
37
38      //Test 1
39      result_len = debug_assemble_msg(result, MAX_LEN, msg, DEBUG_ERR);
40      printf("RESULT1: \%d : \%s\n", result_len, result);
41
42      //Test 2
43      result_len = debug_assemble_msg(result, MAX_LEN, msg2, DEBUG_MSG);
44      printf("RESULT2: \%d : \%s\n", result_len, result);
45
46  }
```

Abbildung 8.9.: Modultest der Funktion debug_assemble_msg

Es wird erwartet, dass beim ersten Test die Nachricht komplett ausgegeben wird. Die Nachricht und das davor zustellende ”E:” sind nur neun Zeichen plus Zeichenende lang. Die zweite Nachricht überschreitet bereits ohne den zusätzlichen Nachrichtenkopf die maximale zulässige Länge von 12 Zeichen. Diese Nachricht muss von der Funktion abgeschnitten werden.

Die Tests liefern die folgende Ausgabe 8.10 als Ergebnis. Wie zu sehen ist, werden die Tests korrekt erfüllt. Die zu lange zweite Nachricht wird abgeschnitten.

```
1  RESULT1: 9 : E:Fehler!
2  RESULT12 11 : D:Debugmeld
```

Abbildung 8.10.: Modultest Ausgabe

Da diese Funktion neben den Eingabeparametern keinem Einfluss von außen unterliegt und nicht erwartet wird, dass diese Funktion geändert werden muss, wird der Test nicht automatisiert und es wird kein Regressionstests vorgenommen.

8.2.2. Integrationstest

Beim Integrationstest wird die Interaktion bzw. Kommunikation verschiedener Komponenten untereinander auf Korrektheit überprüft. Der Bereich von Komponenten erstreckt sich von kleinen Einheiten wie Klassen (C++ / Java) oder Modulen (C) bis hin zu ganzen Programmen.

8.2.2.1. Testen der threadübergreifenden Kommunikation

Die PC-Software des Sensornetzwerks nutzt für die Kommunikation zwischen den einzelnen Threads ausschließlich Message Passing (siehe Kap. 7.4.2). Die Nachrichten zwischen den Threads müssen während der Entwicklung auf Korrektheit überprüft werden. Die Threads werden dafür mit simulierten Daten versorgt und erzeugen daraus Nachrichten, welche sie an andere Threads versenden. Die Threads, welche die Nachrichten empfangen, berechnen daraufhin Ergebnisse, senden diese zurück oder geben sie selbst aus. Die zurückgesandten bzw. ausgegebenen Ergebnisse werden dann mit den erwarteten Ergebnissen verglichen. Genau wie bei dem Modultest ist es hier wichtig alle Grenzfälle abzudecken.

Während der Entwicklung wurden alle Nachrichten zwischen den Threads auf diese Art und Weise getestet.

8.2.2.2. Interaktion zwischen Programmen

Bei der Interaktion zwischen Programmen wird ebenso wie bei der threadübergreifenden Kommunikation ein Programm mit Simulationsdaten gespeist und die Ergebnisse mit den zurückgelieferten ausgegebenen Daten verglichen.

PC-Software Im Falle der PC-Software welche untereinander ausschließlich über JSON-Nachrichten über TCP/IP kommuniziert, bietet es sich an eine Telnetverbindung zu starten, über die simulierte Daten an die PC-Software gesendet werden. Die Ausgaben, welche die PC-Software anschließend an Telnet zurücksendet, können dann mit den erwarteten Werten verglichen werden. Die gesamte Kommunikation der gesamten PC-Software wurde über ein Telnetähnliches Programm getestet.

Sensornetzwerk Das Sensornetzwerk bietet in diesem Bereich nur wenige Möglichkeiten die Interaktion zwischen den einzelnen Knoten zu testen. Durch die fertige Bibliothek (den BitCloudstack; siehe Kap. 5.2.1.2) kann in die Interaktion zwischen End-Devices und Routern nicht eingegriffen werden. Es gibt dort nur die Möglichkeit die Interaktion zwischen Sensorknoten und PC zu testen. Der Masterknoten wurde dafür mit einem Terminalprogramm verbunden. Die Nachrichten, welche die Sensorknoten an den Masterknoten gesendet haben, wurden dann mit dem definierten Protokoll abgeglichen (siehe

Kap. 7.1.2 bzw. im Anhang E für das Protokoll). Ebenso wurden mit Hilfe des Terminalprogramms Nachrichten an den Masterknoten gesendet, welcher die Nachricht dann an den entsprechenden Sensorknoten weitergeleitet hat.

Die folgende Ausgabe 8.11 zeigt ein Beispiel bestehend aus drei Nachrichten, welche von einem Sensorknoten an den Masterknoten gesendet wurden. Der Masterknoten hat diese dann am PC ausgegeben. Der Screenshot des Terminalprogramm wurde für eine bessere Verständlichkeit um Anmerkungen erweitert. In der ersten Nachricht meldet ein Sensorknoten seinen Beitritt im Netzwerk an. In der zweiten Nachricht, teilt der Sensorknoten mit, dass ein Temperatursensor angeschlossen ist. Die dritte Nachricht enthält den ersten Messwert des angeschlossenen Sensoren.

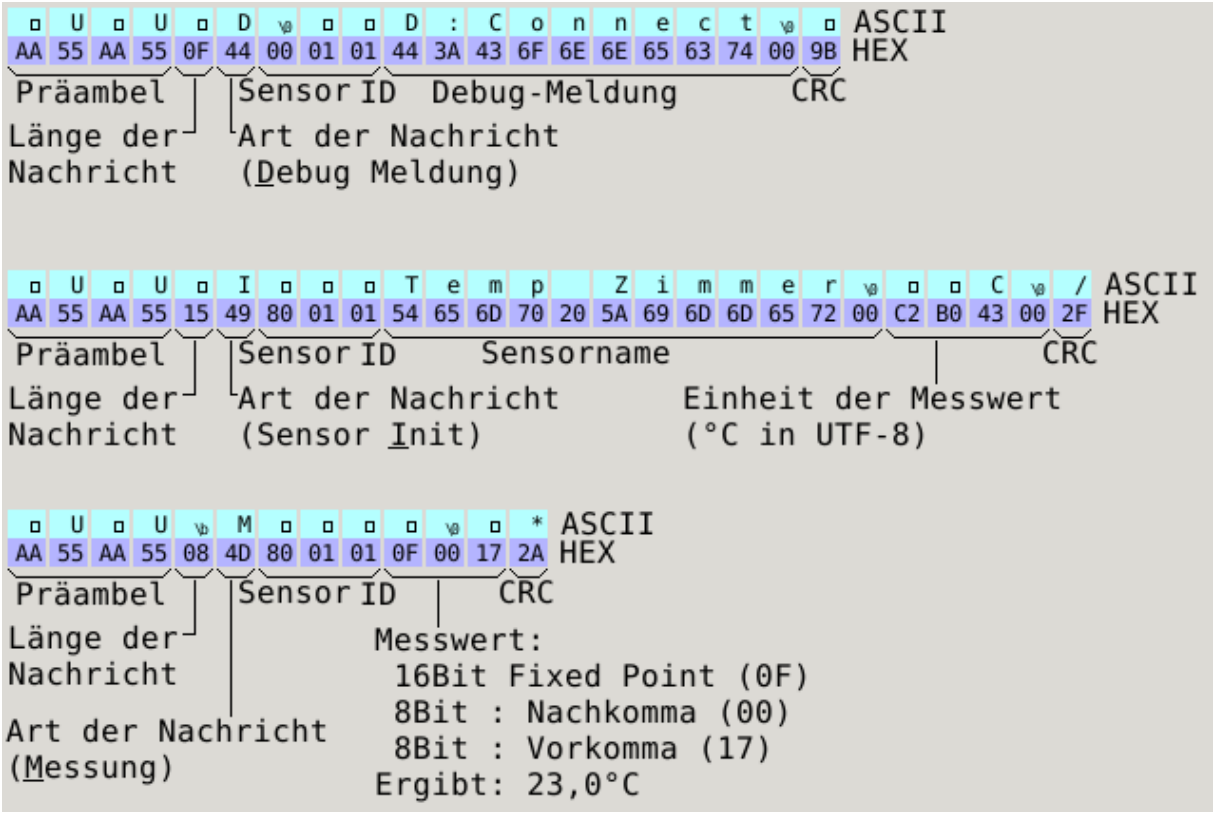

Abbildung 8.11.: Am Terminal ausgegebene Nachrichten

Der Sensorknoten besitzt die Adresse 0x0101. Diese Adresse wird immer in der Sensor ID übertragen. Die ID des Sensors selbst ist 0x80. Für ein komplette Zuordnung wird die Adresse vom Sensorknoten und die ID vom Sensor zu 0x010180 zusammengefasst Alle Daten werden Im Little Endian Format übertragen, dadurch sind die Byte von Zahlen verdreht, Texte können jedoch direkt gelesen werden.

Auf diese Weise konnte die Korrektheit aller Nachricht, welche zwischen PC und Sensornetzwerk ausgetauscht wurden, überprüft werden.

8.2.3. Systemtest

Beim Systemtest wird das gesamte System getestet. Es wird untersucht, ob das System als gesamtes zuverlässig arbeitet. Dieser Tests wird möglichst nahe an der Realität durchgeführt. Dieser Test wird als Blackboxtest durchgeführt. Es wird nicht die Funktionalität der einzelnen Softwarekomponenten analysiert, sonder die Funktion als ganzes. Es wird überprüft, ob bestimmte Eingaben die erwarteten Ausgaben produzieren.

Bevor das Sensornetzwerk im Living Place verbaut wird, wird ein letzter Test durchgeführt. Für diesen Systemtest wird ein Sensornetzwerk aufgebaut und dessen Ergebnisse analysiert. Dabei sollen folgende Dinge analysiert werden:

- Übermittlung der Messwerte im konfigurierten Intervall über den gesamten Testzeitraum
- Anzahl der Verbindungsabrüche im ZigBee-Netzwerk und auf dem PC
- Funktioniert die Verbindungswiederherstellung automatisch
- Gibt es einen Totalausfall durch fehlerhafte Hard- oder Software
- Speicherverbrauch der PC-Software während des Tests
- Plausiblität der Messwerte
- Korrekte Speicherung der Messwerte in der Datenbank

8.2.3.1. Aufbau

Die gesamte zum Sensornetzwerk gehörende PC-Software wird auf einem PC gestartet. Der Masterknoten wird an diesen PC angeschlossen.

Master- und Sensorknoten Es werden drei Sensorknoten aufgebaut, von denen einer als Router konfiguriert ist[5]. Die anderen beiden Sensorknoten werden als End-Devices konfiguriert. Ein End-Device und der Router werden in Funkreichweite des Masterknoten platziert. Der dritte Sensorknoten wird außerhalb der Reichweite des Masterknoten. Dieser muss seine Verbindung zum Netzwerk über den Router herstellen. Beide End-Devices besitzen nur Batterien als Spannungsversorgung. Der Router und der Masterknoten werden extern mit Spannung versorgt.

[5] Siehe Kap. 4.2.3 für eine Funktionsbeschreibung von Routern und End-Devices.

Verwendete Sensoren An beiden End-Devices wird je ein Schalter und ein DS18S20 Temperatursensor[6] angebracht. Die Schalter werden an einer Tür bzw. einem Fenster angebracht und alle fünf Sekunden abgefragt. Die Temperatursensoren werden alle 30 Sekunden abgefragt. Zusätzlich wird die Batteriespannung der zwei End-Devices überwacht. Das Abfrageintervall dafür liegt bei 10 Minuten. Die Batterieabfrage ist so konfiguriert, dass erst bei einer Spannung unter 2,5 Volt Informationen über den Ladestand der Batterien übermittelt werden. Da volle Batterien eingesetzt werden, darf diese Nachricht nie auftreten. Am Routerknoten werden keine physikalischen Sensoren angebracht. Der Routerknoten wird mit drei Pseudosensoren[7] konfiguriert. Jeder dieser drei Pseudosensoren produziert alle fünf Sekunden eine Textnachricht, welche wie jeder andere Messwert vom Sensornetzwerk weiter verarbeitet wird.

PC-Software Auf einem PC wird die gesamte entwickelte Software, sowie ein Active MQ Server und eine Mongo Datenbank gestartet (siehe Kap. 2.1). Zusätzlich werden drei Telnet ähnliche Programme gestartet, die die gesamten Ausgaben der PC-Software an jeder Schnittstelle aufzeichnen (siehe Kap. 7.4).

8.2.3.2. Durchführung

Nachdem alle Komponenten gestartet wurden und sichergestellt ist, dass alle Sensorknoten mit dem Netzwerk verbunden sind, gilt der Test als gestartet. Der Test ist auf eine Laufzeit von vier Tagen angesetzt. Der für den Test verwendete PC wird in diesem Zeitraum für keine anderen Aufgaben verwendet.

Beim Start belegt die PC-Software insgesamt 176.644 KByte Speicher. Die Arbeitsspeicherauslastung des gesamten PCs liegt bei 987 MB.

Die Spannungen der für die Sensorknoten verwendeten Batterien liegen beim Start bei jeweils 3,15 Volt[8].

Während des Tests werden die Zeitpunkte, zu denen das Fenster, bzw. die Tür geöffnet bzw. geschlossen werden, notiert. So können nach Testende die aufgezeichneten Daten mit den notierten Zeiten verglichen werden.

[6] Dieser Temperatursensor bietet eine Genauigkeit von 0,5 Grad. Er wird über einen One-Wire Bus abgefragt. Genauere Information sind unter http://www.maxim-ic.com/datasheet/index.mvp/id/2815 verfügbar. Abruf: 14.02.12

[7] Die Sensorabfrage read_hello_world arbeitet genauso wie jede andere Sensorabfragefunktion. Sie liefert jedoch immer ”hello world” und die Sensor-Id zurück. Eine genauere Beschreibung ist in der Softwaredokumentation vorhanden; siehe Anhang G

[8] Die Messung erfolgt über jeweils zwei in Reihe geschaltete Batterien

8.2.3.3. Testende

Kurz vor Ende des Tests wird überprüft, ob noch von allen Sensoren Messwerte empfangen werden oder ob es zum Ausfall eines der Sensoren gekommen ist. Von allen Sensoren außer der Batterieüberwachung werden Messwert empfangen. Der Test wird nach vier Tagen und ca. 10 Sekunden beendet.

Die Speicherauslastung des Programms liegt am Ende bei 828.521 KByte. Die Arbeitsspeicherauslastung des gesamten PCs nun bei 995 MB.

Die Spannung der Batterien liegt am Ende bei jeweils 3,14 Volt.

8.2.3.4. Auswertung

Nach Abschluss des Tests erfolgt nun die Auswertung. Analysiert werden die zu Beginn aufgestellten Kriterien (siehe Systemtest). Die Auswertung erfolgt ausschließlich auf Basis des Inhalts der Datenbank der PC-Software[9] und der aufgezeichneten Daten aus den Telnet ähnlichen Programmen.

Übermittlung der Messwerte im konfigurierten Intervall Zunächst wird überprüft, ob eine oder mehrere Meldungen über den Ladestand der Batterien eingetroffen sind. Die Batterien waren bei Testbeginn voll und wurden im Test nur sehr gering entladen. Da erst bei unter 2,5 Volt eine Meldung über den Zustand der Batterien erfolgt, dürfen keine Meldung bezüglichen des Batterieladestand eingetroffen sein. Ein Abfrage der Datenbank und eine Suche innerhalb der gespeicherten Datensätze ergab 0 Treffer. Die Batterieüberwachung hat korrekt funktioniert.

Ob alle Messwerte im konfiguriertem Intervall übermittelt wurden, wird statistisch anhand der Anzahl der übertragenen Nachrichten errechnet. Bei dem Test kamen insgesamt 9 Sensoren zum Einsatz. Fünf der Sensoren haben alle 5 Sekunden eine Messung[10] ausgegeben. Die beiden Temperatursensoren gaben ihre Messungen alle 30 Sekunden aus. Die Batterieüberwachungen haben keine Meldungen ausgegeben.

Es folgt eine Rechnung, wie viele Messungen erfolgt sein müssten:
Berechnung der Sekunden von vier Tagen: $4 * 24 * 60 * 60 = 345600$
Berechnung der Anzahl Messungen bei 5 Sekunden Abfrageintervall: $345600/5 = 69120$
Berechnung der Anzahl Messungen bei 30 Sekunden Abfrageintervall: $345600/30 = 11520$
Gesamte Anzahl erwarteter Messungen: $69120 * 5 + 11520 * 2 = 368640$

In der Datenbank sowie den aufgezeichneten Daten befinden sich 368651 Messungen. Dass 11 Messungen mehr vorhanden sind, liegt daran, dass der Test etwa zehn Sekunden Minuten länger als zwei Tage lief. Der Abfrageintervall der Sensoren wurde bei allen Sensoren eingehalten. Wäre dies nicht der Fall, würde eine große Differenz

[9] Siehe Kap. 7.4.2 für mehr Information über die Datenbank
[10] Die fünf Sensoren bestehen aus den drei Pseudosensoren des Routers, des Tür- und des Fensterkontaktes.

zwischen erwarteten und vorhandenen Messwerten vorhanden sein. Die Anzahl der Messungen sind in der Datenbank und den aufgezeichneten Daten gleich. Weder in der Datenbank noch auf der Datenausgabe Schnittstelle sind Messdaten verloren gegangen.

Verbindungsabbrüche Die aufgezeichneten Daten zeigen keine Verbindungsabbrüche. Da weder auf dem PC, noch auf dem Master- bzw. Sensorknoten Verbindungsabbrüche vorhanden waren und am Ende noch alle Sensoren Messwerte ablieferten, kann ausgeschlossen werden, dass Software in irgendeinem der Komponenten abgestürzt ist.

Speicherverbrauch Die Speicherauslastung des Programms steigt von anfänglich ca. 177 MB auf ca. 829 MB. Dies sieht nach einem enormen Speicherleck aus. Die gesamte Speicherauslastung des PCs stieg dem gegenüber von 987 MB auf nur 995 MB. Da während des Tests keine Anwendungen gestartet bzw. beendet wurden, kann eine der beiden Angaben nicht korrekt sein. Die Lösung liegt in der Art, wie Linux den Arbeitsspeicher verwaltet. Jedes Mal, wenn eine Anwendung Speicher anfordert und wieder freigibt, verbleibt der Speicher so lange bei der Anwendung, bis der Arbeitsspeicher im System knapp wird[11].

Aufgrund der Art, wie Linux den Speicher verwaltet, ist davon auszugehen, dass die Anwendung keine Speicherlecks enthält. Die PC-Software fordert nur bei neuen Netzwerkverbindungen zusätzlichen Speicher auf dem Heap an. Ansonsten verwendet das Programm nur Speicher auf dem Stack, welcher beim Verlassen der Funktionen wieder freigegeben wird. Die Freigabe des Speichers beim Schließen einer Verbindung wurde bereits mit einem Modultest überprüft.

Plausiblität der Messwerte Alle 368651 Messungen einzeln zu überprüfen ist nicht möglich. Dies kann nur stichprobenartig vorgenommen werden. Hierfür werden die aufgezeichneten Messwerte und die Messwerte aus der Datenbank hergenommen.

Die beiden Temperatursensoren zum Beispiel zeigten tagsüber eine Zimmertemperatur zwischen 22 und 23°C. Nachdem die Heizung abgeschaltet wurde, sank die Temperatur in der Nacht langsam auf bis zu 15,5°C ab. Am nächsten Tag wurde die Heizung wieder aufgedreht und die Temperatur stieg innerhalb einer halben Stunde auf 20,5 °C. Die Zeiten, in denen die Temperatur absank und wieder anstieg konnte zudem erfolgreich mit der Konfiguration der automatischen Heizungsanlage abgeglichen werden.

Die notierten Zeiten, in denen die Tür bzw. das Fenster geöffnet waren, werden stichprobenartig mit den aufgezeichneten Daten verglichen. Die notierten Angaben stimmen dabei mit den aufgezeichneten Daten überein.

Es gibt keine Hinweise drauf, dass Messwerte falsch erfasst wurden. Alle überprüften Messwerte sind plausibel.

[11] Weitere Informationen dazu können unter http://wiki.debianforum.de/Linux_Speichermanagement eingesehen werden. Abruf: 14.02.12

Korrekte Speicherung der Messwerte in der Datenbank Dass die Messwerte in der Datenbank gespeichert sind, wurde bereits überprüft (siehe ”Übermittlung der Messwerte im konfigurierten Intervall” und ”Plausiblität der Messwerte”). Die Datenbank arbeitet korrekt.

Integrationstest erfolgreich abgeschlossen Alle Analysen des Systemtests fielen positiv aus. Es sind keine Fehler oder Probleme aufgetreten. Dies ist vor allem den voran gegangenen Hardware-, Modul- und Integrationstests zu verdanken. Das Sensornetzwerk hat sich Zuverlässig erwiesen (siehe Anforderung Zuverlässigkeit) und ist bereit für den Einsatz im Living Place.

9. Installation des Sensornetzwerks

Die Entwicklung des Sensornetzwerks und alle Tests sind abgeschlossen. Als letzter Schritt erfolgt der Einbau der ersten Sensorknoten in den Living Place Hamburg.

Mit dem Einbau der ersten Sensorknoten soll eine gute Funkabdeckung des Living Place erreicht werden. Dafür werden die meisten Sensorknoten als Router konfiguriert. Spätere Erweiterungen können auf dieser Grundinstallation aufbauen. Diese später hinzugefügten Sensorknoten können dann als End-Devices verwendet und ohne Verkabelung eingebaut werden. Damit wird die Anforderung ”Geringer Installationsaufwand” für alle zukünftigen Sensorknoten erfüllt.

9.1. Montage der Sensorknoten

Die Platinen mit den ZigBit-Modulen werden in Gehäuse montiert. Sofern der Sensorknoten als Router eingesetzt wird, wird auf der Rückseite der Platine ein Step-Down Wandler montiert. Bei batteriebetriebenen Sensoren wird auf der Rückseite ein Batteriehalter montiert.

Verwendete Sensoren In jedes der fünf Zimmer wird ein Sensorknoten mit Temperatur- und Helligkeitssensoren installiert. Zusätzlich werden vier Fenster der Wohnung überwacht, ob diese geöffnet oder geschlossen sind. Bei den Fenstern kann es passieren, dass sie im geöffnetem Zustand mit den automatischen Rollläden kollidieren. Durch die Überwachung der Fenster soll dies in Zukunft verhindert werden.

Montage im Gehäuse Für die Temperatur- und Helligkeitssensoren werden Öffnungen in das Gehäuse gebohrt. So können die Sensoren im Gehäuse angebracht werden und besitzen eine direkte Verbindung nach außen. Das Foto 9.1 zeigt einen der Sensorknoten samt auf der Rückseite befestigtem Batteriehalter im Gehäuse. Oben links ist ein Fotowiderstand zur Messung der Helligkeit. Darunter befindet sich ein DS18S20 Temperatursensor.

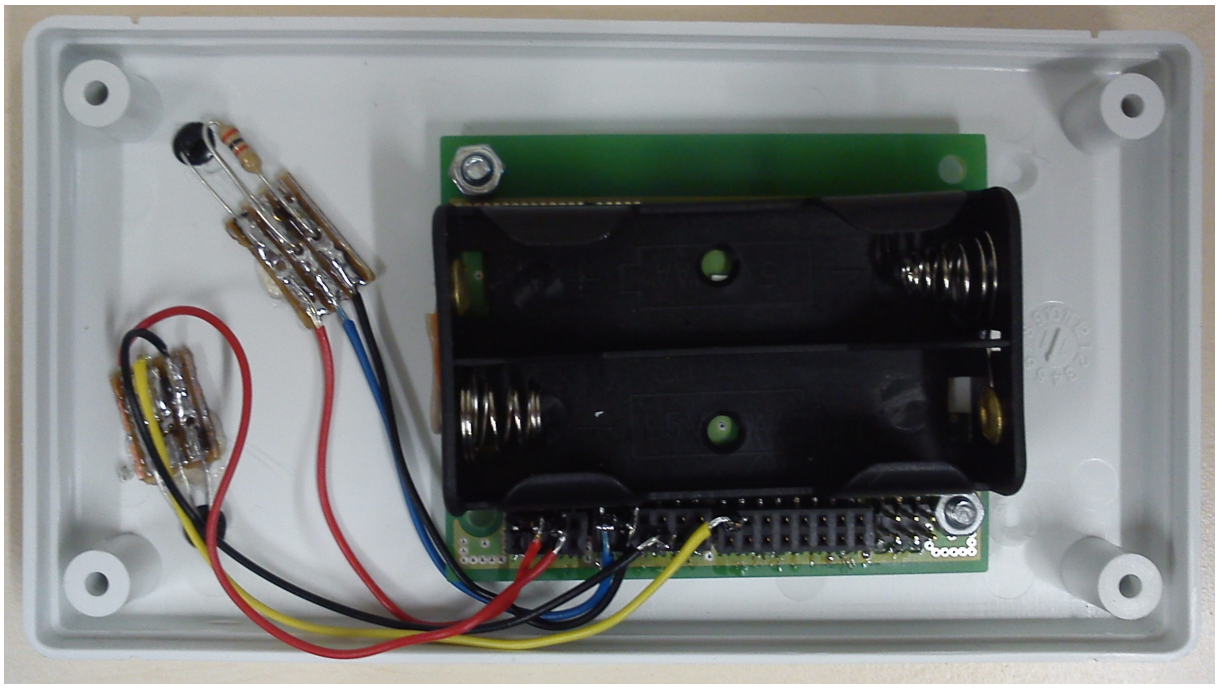

Abbildung 9.1.: Im Gehäuse montierter Sensorknoten

Das Gehäuse ist größer als die in Anforderung ”Unauffälligkeit” angestrebte Größe einer Zigarettenschachtel. Wie auf dem Foto zu sehen ist, ist in dem Gehäuse sehr viel Platz. Die Montage der Platine und der Sensoren in einem Gehäuse, dessen Abmessungen kleiner als eine Zigarettenschachtel sind, ist möglich. Doch alle gefundenen Gehäuse, deren Abmessungen kleiner oder gleich der einer Zigarettenschachtel sind, besitzen keine Innenhöhe von 20 mm[1]. Um die Anforderung der Baugröße zu erfüllen, müsste ein Gehäuse hergestellt werden, was im Rahmen dieser Masterarbeit nicht möglich ist.

Das Foto 9.2 zeigt die Vorderseite des Gehäuse.

[1] Vergleich die benötigte Höhe mit Kap. 6.2.4.1

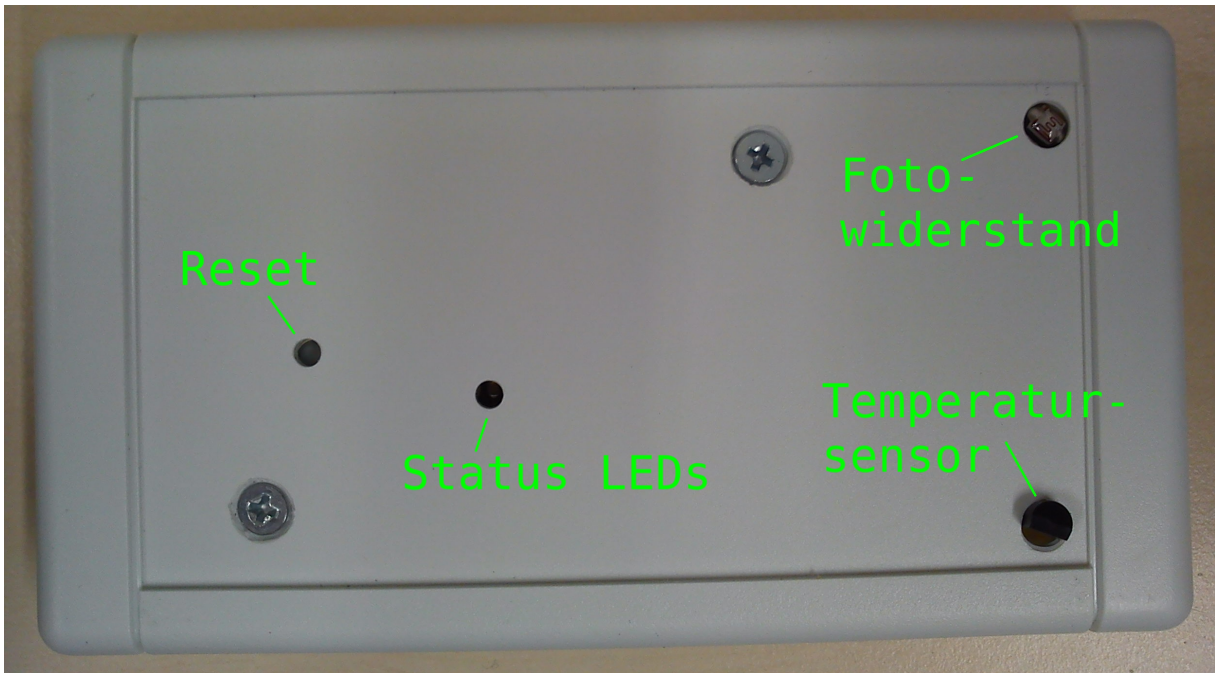

Abbildung 9.2.: Vorderseite des Gehäuse

9.2. Montage im Living Place

Die zusammengebauten Sensorknoten sind nun im Living Place zu montieren. Alle bis auf einen Sensorknoten werden als Router konfiguriert und extern mit Strom versorgt. Ein Sensorknoten wird als End-Device konfiguriert und dient dem Beweis, dass die Sensorknoten ohne externe Stromversorgung zuverlässig arbeiten.

Der folgende Plan 9.3 zeigt den Living Place. Der Montageort der einzelnen Sensoren ist grün markiert. Die Verteilung der IDs ist ZigBee 16Bit Adresse der ZigBee-Knoten (siehe 4.2.1) und weitgehend zufällig. Es wurde nur festgelegt dass alle Router eine ID im Bereich von 0x0001 bis 0x00FF haben. Die IDs der End-Device beginnt bei 0x0100. Der Coordinator muss die ID 0x0000 haben (vgl. Kap. 4.2.3). Auf den Aufbau des Netzwerks hat die räumliche Anordnung der Knoten IDs keinen Einfluss. Weitere Informationen zur Adressvergabe befinden sich in der Softwaredokumentation der Master- und Sensorknoten (siehe Anhang G).

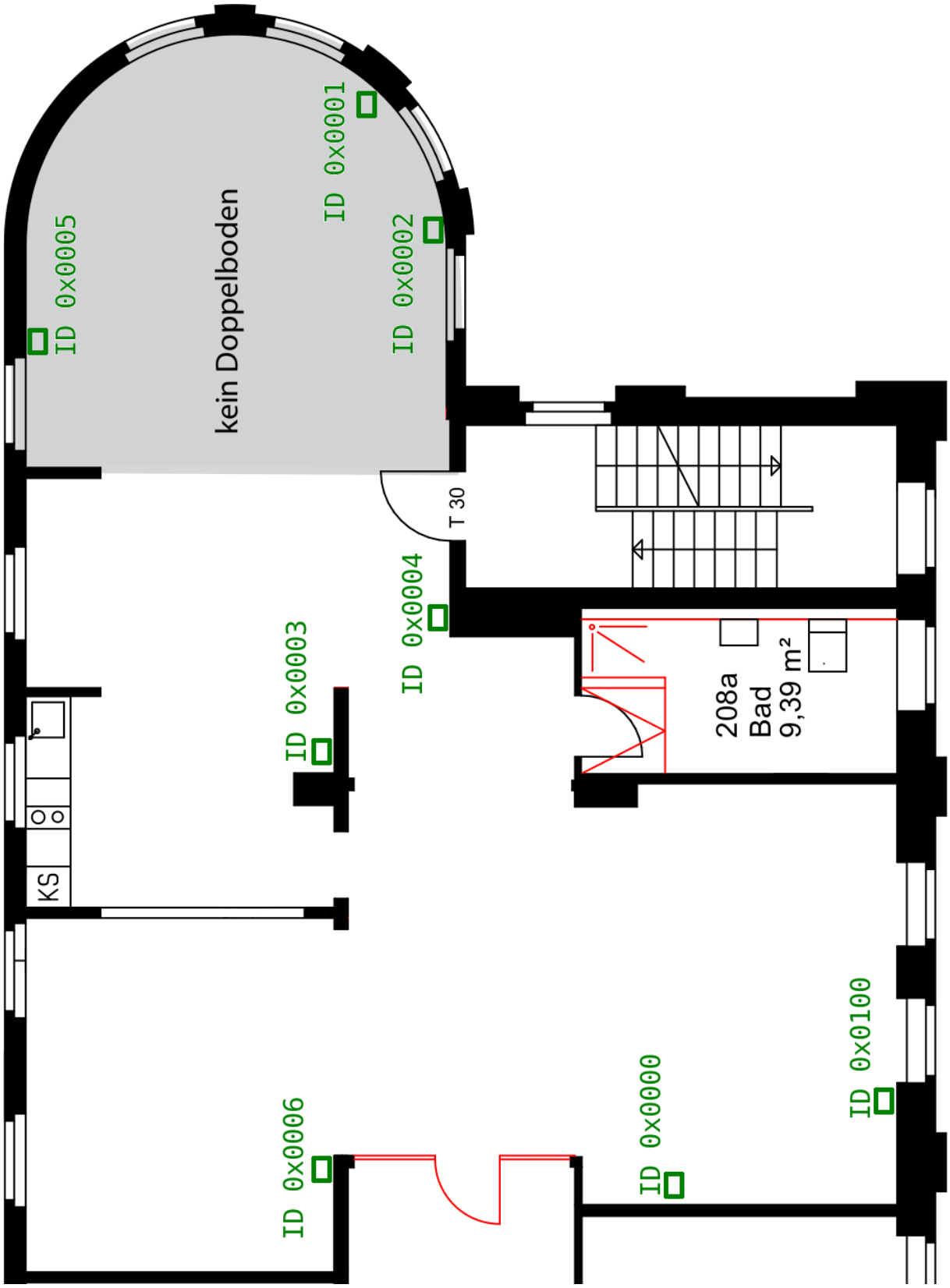

Abbildung 9.3.: Montageort der Sensoren

Die Tabelle 9.1 schlüsselt die Konfiguration der einzelnen Sensorknoten auf. Dabei steht ein C für Coordinator, ein R für Router und E für End-Device. Die Namen der Sensoren orientieren sich am bereits existierenden Namensschema des Living Place[2].

ID	Typ	Ort	Sensoren	Sensornamen
0x0000	C	Schlafzimmer	keine	-
0x0001	R	Lounge	3 Fensterkontakte	winBigLounge1, winBigLounge-Gross2, winBigLounge3
0x0002	R	Lounge	1 Fensterkontakt	winBigLounge4
0x0003	R	Küche	Temperatur und Licht	tempKitchen0, lightKitchen0
0x0004	R	Flur	Temperatur und Licht	tempCorridor0, lightCorridor0
0x0005	R	Lounge	Temperatur und Licht	tempLounge0, lightLounge0
0x0006	R	Esszimmer	Temperatur und Licht	tempDining0, lightDining0
0x0100	E	Schlafzimmer	Temperatur, Licht und Batterie	tempSleep0, lightSleep0, batterySleep0

Tabelle 9.1.: In der Wohnung installierte Sensorknoten

Der Masterknoten (Coordinator) ist über die USB-Schnittstelle (siehe Kap. 5.2.2) an einen PC im Kontrollzentrum des Living Place angeschlossen.

Das Foto 9.4 zeigt den Sensorknoten 0x0002 in der Wohnung. Obwohl der Sensorknoten aus diesem Blinkwinkel nicht vom Vorhang verdeckt wird, ist er kaum zu erkennen. Alle anderen Sensorknoten wurden ebenso unauffällig montiert. Der Sensorknoten erfasst den Zustand des rechten Fensters. Der Schaltkontakt befindet sich dafür in der rechten unteren Ecke des Fensters. Das Kabel vom Schaltkontakt zum Sensorknoten ist über drei Meter lang und wird im bereits vorhandenen Kabelschacht bzw. durch die Fensterbank verdeckt zum Sensorknoten geführt. Eine unauffällige Montage und die geforderte Absetzbarkeit der Sensoren, wie sie in der Anforderung ”Unauffälligkeit” festgelegt ist, wurde erfüllt.

[2]Das Namensschema kann im Artikel ”Object-IDs” des Living Place internem Wiki eingesehen werden.

Abbildung 9.4.: In der Wohnung montierter Sensorknoten

9.3. Installation der Software

Die Sensornetzwerkverwaltungssoftware inklusive der Active-MQ Adapter (siehe Kap. 7.4.2) stellt keine besondere Anforderung an den PC und läuft auf einer virtuellen Maschine. Lediglich die Schnittstelle zwischen dem PC und dem ZigBee-Netzwerk muss auf dem PC im Kontrollzentrum laufen, an dem der Masterknoten angeschlossen ist.

10. Zusammenfassung

In dieser Arbeit wurde ein kabelloses stromsparendes Sensornetzwerk für die Wohnung der Zukunft entwickelt und installiert. Dafür wurden zunächst die Anforderungen an das zu entwickelnde Sensornetzwerk erstellt. Weiterhin wurden in Kapitel 3 Arbeiten aus dem Bereich der kabellosen Sensornetzwerke analysiert, bevor die Entwicklung eines eigenen Sensornetzwerks begonnen wurde.

Die ersten Recherchen für die Entwicklung eines kabellosen Sensornetzwerks begannen bereits vor Beginn dieser Arbeit. So wurde ZigBee als Funkprotokoll für das Sensornetzwerk in einem früherem Projekt ausgewählt (siehe Pautz [2011b] und Kap. 4). In Kapitel 5 wurden weitere Festlegungen zur allgemeinen Struktur des Netzwerks und der grundsätzliche Aufbau der Soft- und Hardwareplattform vorgenommen. Auf Basis dieser Entscheidungen wurde begonnen die Soft- und Hardware getrennt voneinander zu entwickeln (siehe Kap. 6 und 7).

In umfangreichen Bring-Up und Funktionstests (siehe Kap. 8.1) wurde das Sensornetzwerk ausführlich auf zuverlässige Funktion analysiert. Dabei wurde unter anderem eine maximale Laufzeit von einem Jahr bei mindestens 360 Messungen pro Stunde für die Sensorknoten und eine Funkreichweite zwischen zwei Sensorknoten von über 30 Metern innerhalb eines Gebäudes ermittelt (siehe Kap. 8.1.4 und 8.1.3).

Die korrekte Funktion der Software wurde mit Modul-, Integrations- und einem vier Tage dauernden Systemtest überprüft (siehe 8.2).

Nach Abschluss aller Tests wurden 8 Funkknoten und die Software im Living Place Hamburg, einem Labor für die Wohnung der Zukunft installiert (siehe Kap. 9). Diese ersten Sensorknoten übermitteln Temperatur und Lichtverhältnisse im Living Place sowie den Status von 4 Fenstern, welche in geöffnetem Zustand mit den automatischen Rollläden kollidieren können.

Alle gestellten Anforderungen wurden erfolgreich umgesetzt. Lediglich die maximale Baugröße der Sensorknoten wurde nicht erfüllt. Grundsätzlich könnte auch diese Anforderung erfüllt werden. Hierfür müssten Gehäuse angefertigt werden, welche eine Bauhöhe von 20 mm bei gleichzeig geringen Abmessungen in Breite und Länge besitzen (siehe Kap. 9.1).

10.1. Ausblick

Die ersten Sensorknoten sind installiert und stellen ihre Messwerte im Living Place Hamburg bereit. Diese ersten Sensorknoten bilden das Grundgerüst eines noch ausbaufähigen Sensornetzwerks. Das Sensornetzwerk ist so aufgebaut, dass weitere Sensoren jederzeit hinzugefügt werden können und weitere Informationen aus der Wohnung oder der Umgebung erfasst werden können.

Das Vorhandensein der Sensoren alleine stellt noch keine intelligente Wohnung dar. Anderen Projekten obliegt nun die Auswertung der Messwerte und eine Steuerung der Wohnung nach den Wünschen der Bewohner. Welche Aufgaben letztendlich von Computern übernommen werden sollen und wie stark wir von der Technik beeinflusst werden wollen, muss dabei die Zukunft zeigen. Technisch ist bereits heute vieles möglich.

Literaturverzeichnis

[Atmel Corporation 2011a] Atmel Corporation: *ATmega128RFA1*. 8266C-MCU Wireless-08/11. Atmel Corporation 2325 Orchard Parkway San Jose, CA 95131 USA: , 2011. – URL http://www.atmel.com/dyn/resources/prod_documents/doc8266.pdf. – Zugriffsdatum: 13.07.2011

[Atmel Corporation 2011b] Atmel Corporation: *ATmega640/1280/1281/2560/2561*. 2549N–AVR–05/11. Atmel Corporation 2325 Orchard Parkway San Jose, CA 95131 USA: , 2011. – URL http://atmel.com/dyn/resources/prod_documents/doc2549.pdf. – Zugriffsdatum: 15.07.2011

[Atmel Corporation 2011c] Atmel Corporation: *Atmel AVR2050: Atmel BitCloud - Developer Guide*. 8199G–MCU Wireless–11/11. Atmel Corporation 2325 Orchard Parkway San Jose, CA 95131 USA: , 2011. – URL http://www.atmel.com/tools/BITCLOUD-ZIGBEEPRO.aspx. – Zugriffsdatum: 03.01.2012

[Atmel Corporation 2011d] Atmel Corporation: *AVR2044: RCB128RFA1 - Hardware User Manual*. Rev. 8339A-AVR-02/11. Atmel Corporation 2325 Orchard Parkway San Jose, CA 95131 USA: , 2011. – URL http://atmel.com/dyn/resources/prod_documents/doc8339.pdf. – Zugriffsdatum: 31.10.2011

[Atmel Corporation 2011e] Atmel Corporation: *BitCloud - ZigBee PRO*. 2011. – URL http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4495. – Zugriffsdatum: 10.06.2011

[Beyer u. a. 2003] Beyer, Stefan ; Mayes, Ken ; Warboys, Brian: Dynamic Configuration of Embedded Operating Systems / University of Manchester. University of Manchester, September 2003. – Forschungsbericht. – URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.6.2314&rep=rep1&type=pdf. – Zugriffsdatum: 2011.09.12

[Bremer u. a. 2004] Bremer, Roger ; Chavers, Tracey ; Yu, Zhongmin: Power supply and ground design for WiFi transceiver. In: *rfdesign* (2004), November, S. 16–22

[Energizer Holdings Inc. 2008] Energizer Holdings Inc.: *Energizer EN91*, 2008. – URL http://data.energizer.com/SearchResult.aspx. – Zugriffsdatum: 31.10.2011

[Energizer Holdings Inc. 2010] Energizer Holdings Inc.: *Nickel Metal Hydride (NiMH) Hanndbook and Application Manual*. NiMH01.11, 2010. – URL http://data.energizer.com/SearchResult.aspx. – Zugriffsdatum: 12.12.2011

[Farahani 2008] Farahani, Shahin: *ZigBee wireless networks and transceivers*. Elsevier Ldt., 2008. – ISBN 978-0-7506-8393-7

[Gregor 2009] Gregor, S.: Tangible Computing revisited: Anfassbare Computer in Intelligenten Umgebungen / HAW Hamburg. HAW Hamburg, 2009. – Forschungsbericht. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/papers/MMWismar2009.pdf

[Gungor und Hancke 2009] Gungor, Vehbi C. ; Hancke, Gerhard P.: Industrial Wireless Sensor Networks: Challenges, Design Principles, and Technical Approaches. In: IEEE (Hrsg.): *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS* Bd. 56 IEEE (Veranst.), October 2009, S. 4258 – 4265

[Hantsche 2009] Hantsche, Bernd: Der Wettstreit um drahtlose Sensor-Netzwerke. In: *Elektronik* (2009), September, Nr. Sonderausgabe, S. 20–23

[Huang u. a. 2010a] Huang, Jun ; Xing, Guoliang ; Zhou, Gang ; Zhou, Ruogu: Beyond Co-existence: Exploiting WiFi White Space for ZigBee Performance Assurance / Michigan State University. 2010. – Forschungsbericht

[Huang u. a. 2010b] Huang, Li ; Pop, Valer ; Francisco, Ruben de ; Vullers, Ruud ; Dolmans, Guido ; Groot, Harmke de: Ultra Low Power Wireless and Energy Harvesting Technologies – An Ideal Combination / Holst Centre / imec. September 2010. – Forschungsbericht

[Institute of Electrical and Electronics Engineers, Inc. 2006] Institute of Electrical and Electronics Engineers, Inc. (Veranst.): *IEEE Std 802.15.4-2006*. September 2006. – URL http://standards.ieee.org/about/get/802/802.15.html. – Zugriffsdatum: 06.06.2011

[Johannsen 2010] Johannsen, Benedikt: Generische Modelbasierte Kommunikationsinfrastruktur / HAW Hamburg. HAW Hamburg, December 2010. – Forschungsbericht. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master10-11-seminar/johannsen/bericht.pdf. Master Seminar

[Kopják und Kovács 2011] Kopják, József ; Kovács, Dr. J.: Event-driven control program models running on embedded systems. In: *International Symposium on Applied Computational Intelligence and Informatics* IEEE (Veranst.), May 2011

[Lemme 2009] Lemme, Helmuth: Drahtlos in die Zukunft. In: *Elektronik* (2009), September, Nr. Sonderausgabe, S. 28–32

[Lhermet u. a. 2008] Lhermet, Hélène ; Condemine, Cyril ; Plissonnier, Marc ; Salot, Raphaël ; Audebert, Patrick ; Rosset, Marion: Efficient Power Management Circuit: From Thermal Energy Harvesting to Above-IC Microbattery Energy Storage. In: *IEEE JOURNAL OF SOLID-STATE CIRCUITS* Bd. 43 IEEE (Veranst.), January 2008, S. 246 – 255

[Maxim 2010] Maxim: *DS18S20 High-Precision 1-Wire Digital Thermometer*. 19-5474; Rev 8/10. Maxim Integrated Products, 120 San Gabriel Drive, Sunnyvale , CA: , 2010. – URL http://www.maxim-ic.com/datasheet/index.mvp/id/2815. – Zugriffsdatum: 14.02.12

[MeshNetics 2007] MeshNetics: *ZigBitTM OEM Modules ZDM-A1281-**. DOC. M-251 01 V.1.9. MeshNetics 9 Dmitrovskoye Shosse Moscow 127434, Russia: , 2007. – URL http://www.mikrocontroller.net/topic/84931. – Zugriffsdatum: 15.07.2011. – Ältere Version des Datenblatt, enthält mehr Information als neuere Versionen

[Niels Aakvaag 2005] Niels Aakvaag, Gilles T.: Timing and Power Issues in Wireless Sensor Networks - an Industrial Test Case. In: *Parallel Processing Workshops* IEEE (Veranst.), July 2005

[nmea.de 2009] nmea.de: *NMEA-0183 Daten*. 2009. – URL http://www.kowoma.de/gps/zusatzerklaerungen/NMEA.htm. – Zugriffsdatum: 16.01.2012

[Noe 2010] Noe, Marcel: Entwurf und Implementierung eines kabellosen Sensornetzes zur Überwachung von Patienten bei einem Massenanfall von Verletzten / Karlsruhe Institute of Technology. Karlsruhe Institute of Technology, November 2010. – Forschungsbericht. – URL http://www.marcel-noe.de/publications/Diplomarbeit.pdf. – Zugriffsdatum: 2011.09.16

[Otto und Voskuhl 2010] Otto, Kjell ; Voskuhl, Sören: Entwicklung einer Architektur für den Living Place Hamburg / HAW Hamburg. HAW Hamburg, August 2010. – Forschungsbericht. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2010-proj1/otto_voskuhl.pdf. Projekt 1 Bericht

[Otto und Voskuhl 2011] Otto, Kjell ; Voskuhl, Sören: Weiterentwicklung der Architektur des Living Place Hamburg / HAW Hamburg. HAW Hamburg, February 2011. – Forschungsbericht. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master10-11-proj2/otto-voskuhl.pdf. Projekt 2 Bericht

[Pasha u. a. 2010] Pasha, Muhammad A. ; Derrien, Steven ; Sentieys, Olivier: A Complete Design-Flow for the Generation of Ultra Low-Power WSN Node Architectures Based on Micro-Tasking / University of Rennes. April 2010. – Forschungsbericht

[Pautz 2010a] Pautz, Alexander: Analyse von Feldbussystemen in Hinblick auf Ambient Intelligence / HAW Hamburg. HAW Hamburg, January 2010. – Forschungsbericht. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master09-10-aw1/pautz/bericht.pdf. Anwendungen 1 Ausarbeitung

[Pautz 2010b] Pautz, Alexander: Vermittlungsinfrastrukturen in komplexen Netzwerken / HAW Hamburg. HAW Hamburg, August 2010. – Forschungsbericht. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2010-aw2/pautz/bericht.pdf. Anwendungen 2 Ausarbeitung

[Pautz 2011a] Pautz, Alexander: Kabelloses Sensornetzwerk im Living Place Hamburg / HAW Hamburg. HAW Hamburg, February 2011. – Forschungsbericht. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master10-11-seminar/pautz/bericht.pdf. Masterseminar Ausarbeitung

[Pautz 2011b] Pautz, Alexander: Wahl eines Funkstandards für ein stromsparendes Sensornetzwerk / HAW Hamburg. HAW Hamburg, July 2011. – Forschungsbericht. Projekt 2 Ausarbeitung

[Pautz und Johannsen 2010] Pautz, Alexander ; Johannsen, Benedikt: *Einbindung eines proprietären Bussystems in eine komplexe Kommunikationsinfrastruktur*. 2010. – unterliegt NDA

[Sikora 2009] Sikora, Axel: ZigBee im Tal der Erleuchtung. In: *Elektronik* (2009), September, Nr. Sonderausgabe, S. 16–19

[STMicroelectronics 2010] STMicroelectronics: *Very low drop voltage regulators with inhibit*. REV 24. STMicro 39, Chemin du Champ des FillesPlan-Les-Ouates Geneva CH1228, France: , 2010. – URL http://www.st.com/internet/analog/product/63606.jsp. – Zugriffsdatum: 22.09.2011

[Sun u. a. 2010] Sun, Daozong ; Jiang, Sheng ; Wang, Weixing ; Tang, Jingchi: WSN Design and Implementation in a Tea Plantation for Drought Monitoring. In: *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery* IEEE (Veranst.), September 2010

[Texas Instruments 2011] Texas Instruments: *LM2574,LM2574HV*. DS011394. Texas Instruments, Post Office Box 655303, Dallas, Texas 75265, USA: , 2011. – URL https://www.national.com/mpf/LM/LM2574.html#Overview. – Zugriffsdatum: 30.01.2012

[Texas Instruments Incorporated 2009] Texas Instruments Incorporated: *LOW INPUT VOLTAGE SYNCHRONOUS BOOST CONVERTER WITH LOW QUIESCENT CURRENT*. JULY 2009, 2009. – URL http://www.ti.com/product/tps61097-33. – Zugriffsdatum: 14.12.2011

[The Apache Software Foundation 2010] The Apache Software Foundation: *Apache ActiveMQ*. 2010. – URL http://activemq.apache.org/. – Zugriffsdatum: 02.08.2010

[Wang u. a. 2011] Wang, W. S. ; O'Keeffe, R. ; Wang, N. ; Hayes, M. ; O'Flynn, B. ; O'Mathuna, C.: Practical Wireless Sensor Networks Power Consumption Metrics for Building Energy Management Applications / Tyndall National Institute. Tyndall National Institute, July 2011. – Forschungsbericht. – URL http://zuse.ucc.ie/forumbau2011/papers/19.pdf. – Zugriffsdatum: 2011.09.02

[ZigBee Alliance 2008] ZigBee Alliance (Veranst.): *ZigBee Specification*. r18. January 2008. – Die Spezifikation kann über www.ZigBee.org angefordert werden

A. Messergebnisse

Hier sind Messergebnisse aus den durchgeführten Tests. Die Bedeutung der Messergebnisse kann dem jeweils angegebenem Kapitel entnommen werden.

A.1. Verhältnis zwischen Aus- und Eingangsspannung

Die Tabelle A.1 zeigt die Messergebnisse aus Kapitel 8.1.2.1. Vin ist die Eingangsspannung. K1 und K2 sind die beiden getesteten Knoten. ”o. L.” ist ohne Last und ”m. L. ist mit einem 150 Ohm Widerstand als Last.

Vin	K1, m. L.	K2, m. L.	K1, o. L.	K2, o. L.
1,400	3,310	3,320	3,304	3,315
1,500	3,311	3,321	3,304	3,315
1,600	3,310	3,321	3,304	3,315
1,700	3,309	3,320	3,304	3,315
1,800	3,309	3,321	3,304	3,315
1,900	3,311	3,321	3,304	3,315
2,000	3,309	3,321	3,304	3,315
2,100	3,308	3,320	3,304	3,315
2,200	3,308	3,320	3,304	3,316
2,300	3,308	3,321	3,304	3,316
2,400	3,309	3,321	3,304	3,316
2,500	3,308	3,320	3,305	3,317
2,600	3,308	3,320	3,305	3,317
2,700	3,308	3,320	3,306	3,317
2,800	3,308	3,319	3,307	3,317
2,900	3,307	3,319	3,308	3,318
3,000	3,307	3,318	3,308	3,319
3,100	3,307	3,319	3,309	3,320
3,200	3,306	3,319	3,312	3,323
3,300	3,309	3,319	3,317	3,327
3,400	3,337	3,329	3,329	3,336
3,500	3,379	3,384	3,375	3,381

Tabelle A.1.: Step-Up Wandler Eingangsspannung gegen Ausgangsspannung

A.2. Testen des Step-Down Wandlers

Alle Sensorknoten, welche als Router arbeiten, müssen extern mit Strom versorgt werden. Dazu wurde ein Platine entwickelt, welche einen Step-Down Wandler trägt. Mit Hilfe dieses Step-Down Wandlers können die Routerknoten dann an die 24 Volt Stromversorgung der Fenstermotoren im Living Place angeschlossen werden (siehe 6.2.2). Genau wie beim Step-Up Wandler muss nun dessen Funktion getestet werden (siehe Testen des Step-Up Wandlers). Es werden die gleichen Tests durchgeführt. Der einzige Unterschied besteht in der gewählten Eingangsspannung von 3 bis 28 Volt.

Da der Step-Down Wandler sich auf einer separaten Platine befindet, können die Tests ohne das ZigBit-Modul durchgeführt werden. Bei einer falschen Ausgangsspannung werden so keine Bauteile beschädigt.

Einhalten der Ausgangsspannung Wie auch beim Step-Up Wandler wird zunächst überprüft, ob die Ausgangsspannung eingehalten wird. Hierfür wird eine Spannung von 24 Volt an den Eingang der Step-Down Wandlers angeschlossen. Der Ausgang wird dabei wieder mit 22 und 44 mA belastet (siehe auch Kap. 8.1.2.1).

Alle mit dem Multimeter[1] gemessenen Ausgangsspannungen der Step-Down Wandler lagen im Bereich zwischen 3,267 und 3,304 Volt. Die Abweichung zu 3,300 Volt beträgt maximal 0,033 Volt und 1%. Diese Genauigkeit ist ausreichend. An den einzelnen Knoten wurde beim Wechsel von 22 auf 44 mA eine Spannungsabsenkung um 7 bis 10 mV festgestellt. Dieser Wert ist gering. Bei Laständerung könnte eine Spannungsänderung von 100 mV akzeptiert werden, solange der Bereich von 2,7 bis 3,6 Volt nicht verlassen wird.

Verhältnis zwischen Aus- und Eingangsspannung Falls eine andere Versorgungsspannung als 24 Volt verwendet wird, muss die Ausgangsspannung dennoch stabil bleiben. Dies wird im Folgendem anhand zweier Spannungswandler getestet. Genau wie beim Step-Up Wandler wird die Spannung dabei mit und ohne Last getestet. In diesem Abschnitt wird die Untersuchung des vorherigen Abschnittes an zwei Spannungswandlern fortgeführt. Es wird dabei untersucht, ob die 3,3 Volt unter Last und im Leerlauf konstant gehalten werden können. Die Eingangsspannung wird im Bereich von 3,0 bis 28,0 Volt in 1,0 Volt Schritten erhöht. Die Eingangsspannung wird auf bis zu 3,0 Volt abgesenkt, um das Verhalten der Step-Down Wandler im Randbereich zu erkennen. Die Spannung darf nicht über 28 Volt angehoben werden, da eine korrekte Funktion sonst nicht mehr gewährleistet werden kann (siehe Anhang B.3). Die Spannung wird im Leerlauf und mit einem 150 Ohm Widerstand am Ausgang des Spannungwandlers gemessen.

Im folgenden Diagramm A.1 sind die Eingangs- und Ausgangspannung des Step-Down Wandlers gegenübergestellt. Zur besseren Übersicht befinden sich nur die Messwerte

[1] Multimeter: VC160, maximale Abweichung der Messung: 0,8% + 1dgt. Entspricht maximal 28 mV Abweichung bei 3,300 Volt

eines Wandlers in der Grafik. Die Messwerte beider gemessenen Knoten können der Tabelle A.2 unterhalb der Grafik entnommen werden. Zudem wurden die Messergebnisse unterhalb von 6 Volt Eingangsspannung nicht aufgeführt. In diesem Bereich liegt die Ausgangsspannung bei nahezu 0 Volt.

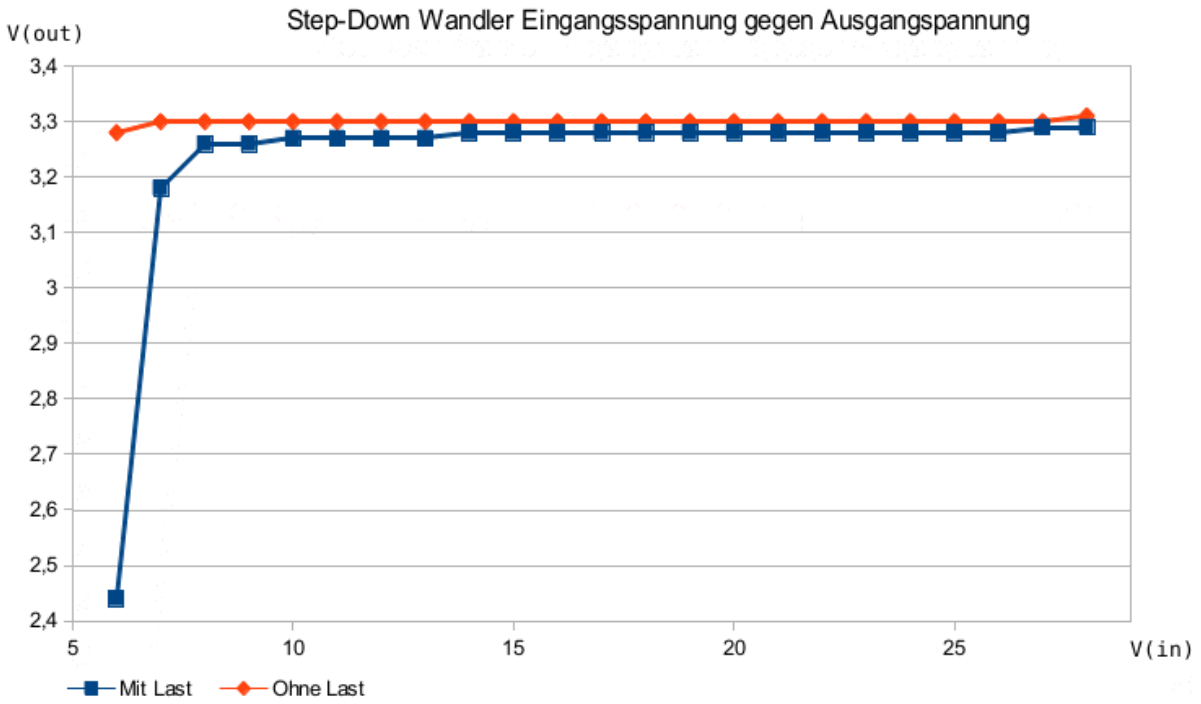

Abbildung A.1.: Step-Down Wandler Eingangsspannung gegen Ausgangsspannung

Die Tabelle A.2 enthält die Messergebnisse der 2 Step-Down Wandler des obigen Tests. W1 und W2 sind die beiden getesteten Step-Down Wandler. ”o. L.” ist ohne Last und ”m. L. ist mit einem 150 Ohm Widerstand als Last.

Vin	W1, m. L.	W2, m. L.	W1, o. L.	W2, o. L.
3	0,00	0,00	0,02	0,02
4	0,00	0,00	0,31	0,31
5	0,27	0,28	2,61	2,62
6	2,44	2,46	3,28	3,28
7	3,18	3,20	3,30	3,30
8	3,26	3,27	3,30	3,30
9	3,27	3,27	3,30	3,30
10	3,27	3,27	3,30	3,30
11	3,27	3,27	3,30	3,30
12	3,27	3,28	3,30	3,30
13	3,27	3,28	3,30	3,30
14	3,28	3,28	3,30	3,30
15	3,28	3,28	3,30	3,30
16	3,28	3,28	3,30	3,30
17	3,28	3,28	3,30	3,30
18	3,28	3,28	3,30	3,30
19	3,28	3,28	3,30	3,30
20	3,28	3,28	3,30	3,30
21	3,28	3,28	3,30	3,30
22	3,28	3,28	3,30	3,30
23	3,28	3,28	3,30	3,30
24	3,28	3,28	3,30	3,31
25	3,28	3,29	3,30	3,31
26	3,28	3,29	3,30	3,31
27	3,28	3,29	3,30	3,31
28	3,29	3,29	3,31	3,31

Tabelle A.2.: Step-Down Wandler Eingangsspannung gegen Ausgangsspannung

Eine stabile Ausgangsspannung steht erst ab ca. 9 Volt Eingangsspannung bereit. Im Bereich von 9 Volt bis 28 Volt steigt die Ausgangsspannung nur noch um 20 mV. Weiterhin ist zu beobachten, dass die Ausgangsspannung unter Last immer 20 mV geringer ausfällt. Dies stellt für den Betrieb der Sensorknoten jedoch kein Problem dar, vor allem da die als Router arbeitenden Sensorknoten nie in einen Stromsparmodus wechseln.

Die Step-Down Wandler haben etwas größere Toleranzen als die Step-Up Wandler. Dennoch arbeiten die Step-Down Wandler innerhalb akzeptabler Toleranzen und können für die Sensorknoten verwendet werden,

Stabilität der Ausgangsspannung Dieser Test wird ebenso ausgeführt wie der Test ”Stabilität der Ausgangsspannung” des Step-Up Wandlers. Lediglich die Eingangsspannung wird im Bereich von 9 bis 28 Volt variiert.

Das folgende Bildschirmfoto A.2 vom Oszilloskop zeigt die Ausgangsspannung des Step-Down Wandlers bei 24 Volt Eingangsspannung und mit 150 Ohm Last. Wie auf der Aufnahme zu sehen ist, schwingt die Spannung nicht.

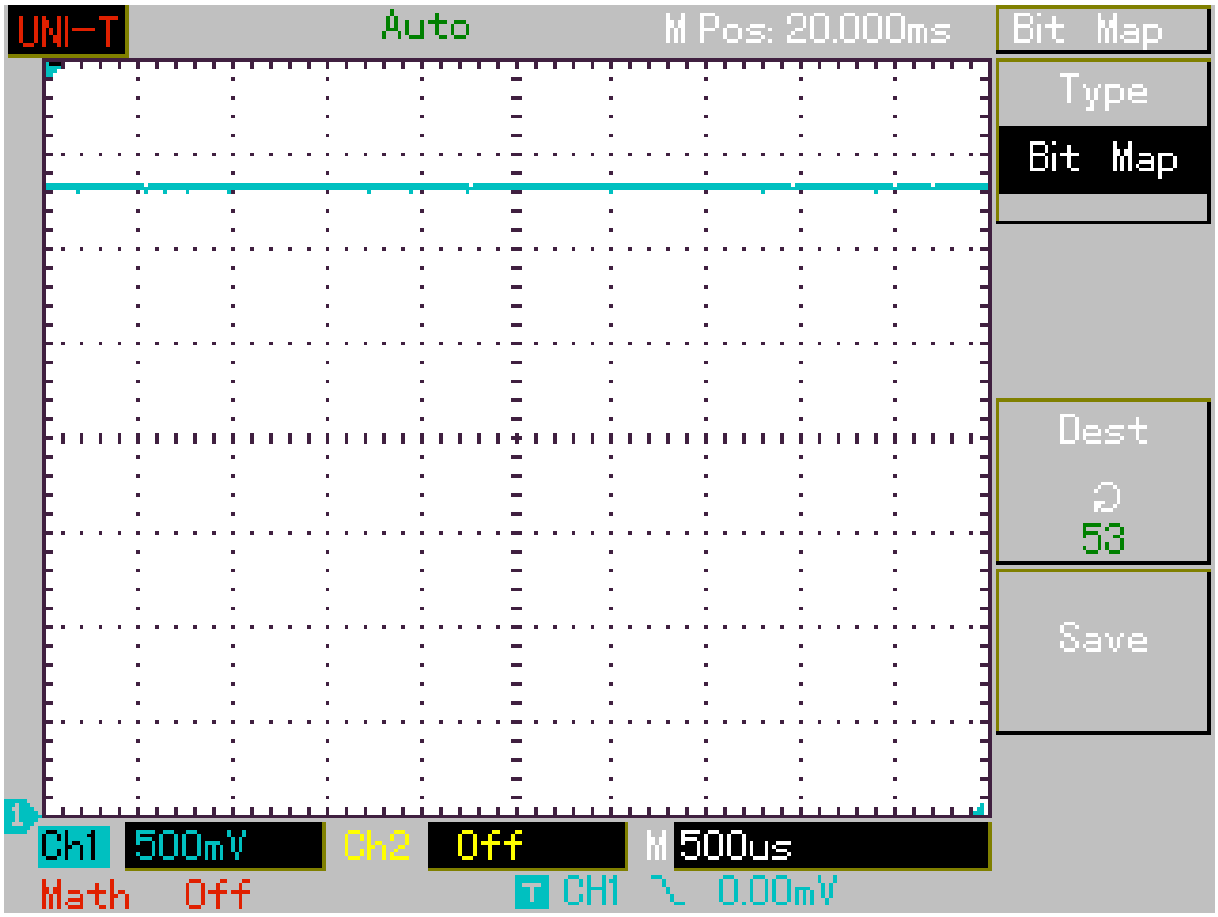

Abbildung A.2.: Messung der Spannungsversorgung des Step-Down Wandlers

Tests erfolgreich bestanden Die Tests ergaben, dass die Step-Down Wandler im Bereich von 9 bis 28 Volt eine Spannung von 3,3 Volt erzeugen. Auch wenn die Spannung unter Last etwas stärker als bei den Step-Up Wandlern absinkt, ist die Spannung stabil und schwingt nicht. Die Step-Down Wandler können ohne Einschränkungen für die Versorgung der Sensorknoten verwendet werden.

A.3. Ermittlung der maximalen Funkreichweite

Hier befinden sich die Messergebnisse bei der Ermittlung der maximalen Funkreichweiten. Die Testbedingungen werden in Kapitel 8.1.3 erklärt.

Tabelle A.3 zeigt wie häufig einzelne Pingzeiten gemessen wurden. Insgesamt wurden 100 Pings versendet und empfangen. Die Entfernung zwischen dem Master- und Sensorknoten betrug 28 m. Die durchschnittliche Pingzeit betrug 21,8 ms und 66 % der Messungen befanden sich innerhalb von 0,6 ms um den Durchschnitt.

Zeit in ms	Häufigkeit
18,2	1
19,4	1
20,2	1
20,6	5
21,0	3
21,4	5
21,6	4
21,8	2
22,0	7
22,2	11
22,4	20
22,6	13
22,8	6
23,0	4
23,2	2
23,6	9
24,6	1
25,2	2
26,0	2
27,0	1

Tabelle A.3.: Auflistung aller Pingzeiten bei 28 Metern

Tabelle A.4 zeigt wie häufig einzelne Pingzeiten gemessen wurden. Insgesamt wurden 100 Pings versendet und empfangen. Die Entfernung zwischen dem Master- und Sensorknoten betrug 32 m. Die durchschnittliche Pingzeit betrug 25,8 ms und 66 % der Messungen befanden sich innerhalb von 3,4 ms um den Durchschnitt.

Zeit in ms	Häufigkeit
18,8	1
20,6	1
21,4	2
21,6	1
21,8	4
22,0	6
22,2	5
22,4	2
22,6	1
22,8	2
23,4	1
23,6	2
24,4	2
24,6	3
24,8	1
25,2	4
25,4	4
25,6	3
26,0	2
26,2	3
26,4	2
26,8	7
27,0	5
27,4	3
27,6	5
27,8	4
28,0	1
28,2	4
28,4	6
28,8	1
29,0	3
29,2	6
29,4	4
29,6	1

Tabelle A.4.: Auflistung aller Pingzeiten bei 32 Metern

Tabelle A.5 zeigt wie häufig einzelne Pingzeiten gemessen wurden. Insgesamt wurden 100 Pings versendet und empfangen. Die Entfernung zwischen dem Master- und Sensorknoten betrug 33 m. Während des Tests gingen drei Pakete verloren.

Zeit in ms	Häufigkeit
19,0	1
20,2	1
21,4	1
21,6	2
21,8	5
22,0	5
22,2	4
22,4	3
23,8	1
24,4	1
24,6	3
24,8	2
25,0	1
25,2	2
25,4	4
25,6	3
26,2	2
26,4	3
26,8	7
27,0	4
27,2	1
27,4	4
27,6	4
27,8	3
28,0	2
28,2	4
28,4	6
28,6	1
29,0	5
29,2	4
29,4	4
29,6	1
ohne Antwort	3

Tabelle A.5.: Auflistung aller Pingzeiten bei 33 Metern

A.4. Messen der Stromaufnahme

Tabelle A.6 zeigt die Leistungsaufnahme eines Sensorknotens im Stromsparmodus in Abhängigkeit von der Eingangsspannung.

Volt (V)	Stromaufnahme (μA)	Effizienz (%)	Verbrauch (mW)
1,6	65	63,5	104
1,7	61	63,6	103,7
1,8	57	64,3	102,6
1,9	54	64,3	102,6
2	51	64,7	102
2,1	49	64,1	102,9
2,2	46	65,2	101,2
2,3	44	65,2	101,2
2,4	42	65,5	100,8
2,5	40	66,0	100
2,6	38	66,8	98,8
2,7	36	67,9	97,2
2,8	34	69,3	95,2
2,9	32	71,1	92,8
3	30	73,3	90
Durchschnitt	45,3	66,3	99,7

Tabelle A.6.: Messung des Leistungsaufnahme eines schlafenden Sensorknotens

Tabelle A.7 zeigt die Leistungsaufnahme eines aktiven Sensorknoten in Abhängigkeit von der Eingangsspannung.

Volt (V)	Stromaufnahme (mA)	Effizienz (%)	Verbrauch (μW)
1,6	117,2	75,7	187,5
1,7	103,1	81,0	175,3
1,8	94,8	83,2	170,6
1,9	89,5	83,5	170,1
2	84,1	84,4	168,2
2,1	79,2	85,4	166,3
2,2	75,1	85,9	165,2
2,3	71,4	86,5	164,2
2,4	67,8	87,3	162,7
2,5	64,6	87,9	161,5
2,6	61,8	88,4	160,7
2,7	58,9	89,3	159,0
2,8	56,8	89,3	159,0
2,9	54,5	89,8	158,1
3	52,4	90,3	157,2
Durchschnitt	75,4	85,9	165,7

Tabelle A.7.: Messung des Leistungsaufnahme eines aktiven Sensorknotens

B. Schaltpläne und Layouts

In diesem Anhang befinden sich alle Schaltpläne und Layouts zur schnellen Ansicht, welche für diese Masterarbeit erstellt wurden. Die Schaltpläne befinden sich nochmals auf der beigelegten DVD (siehe Anhang G).

B.1. Masterknoten

Die Abbildung B.1 zeigt den Schaltplan des Masterknotens. Im oberen Bereich ist der ATMega128RFA1 samt Außenbeschaltung zu sehen. Darunter ist der FT232RL mit der USB-Buchse abgebildet. Rechts unten ist das Anpassnetzwerk (Balun) und Antenne. Ganz unten ist der Spannungsregler sowie die benötigten Kondensatoren.

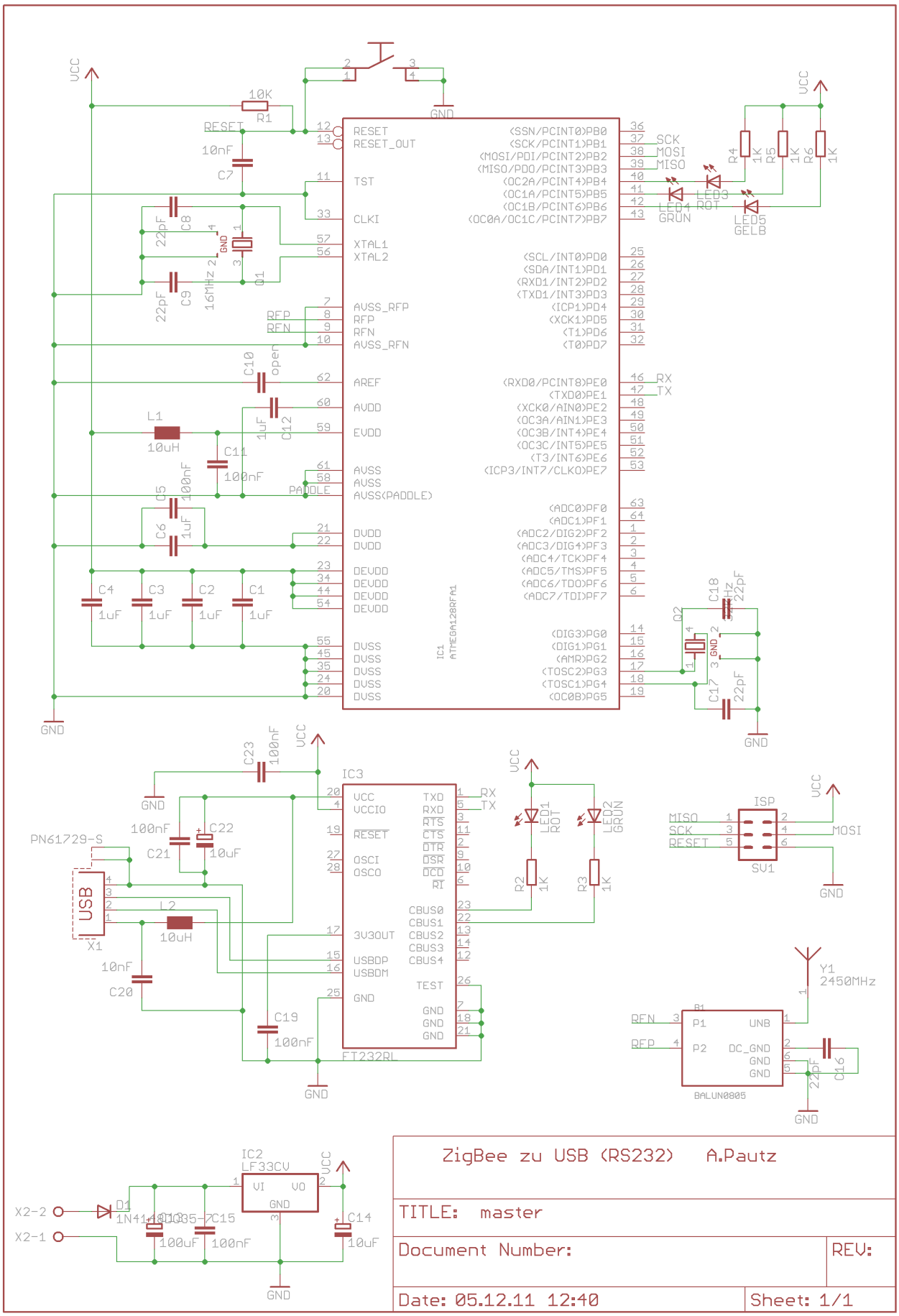

Abbildung B.1.: Schaltplan des Masterknotens

Die Abbildung B.2 zeigt das Layout des Masterknotens. Links ist die Antenne, gefolgt vom ATMega128RFA1. Oberhalb der Schrift ist der Spannungsregler und darüber. Oben rechts sitzt die USB-Buchse sowie der FT232RL. Die Platine ist 66,5 x 50 mm groß.

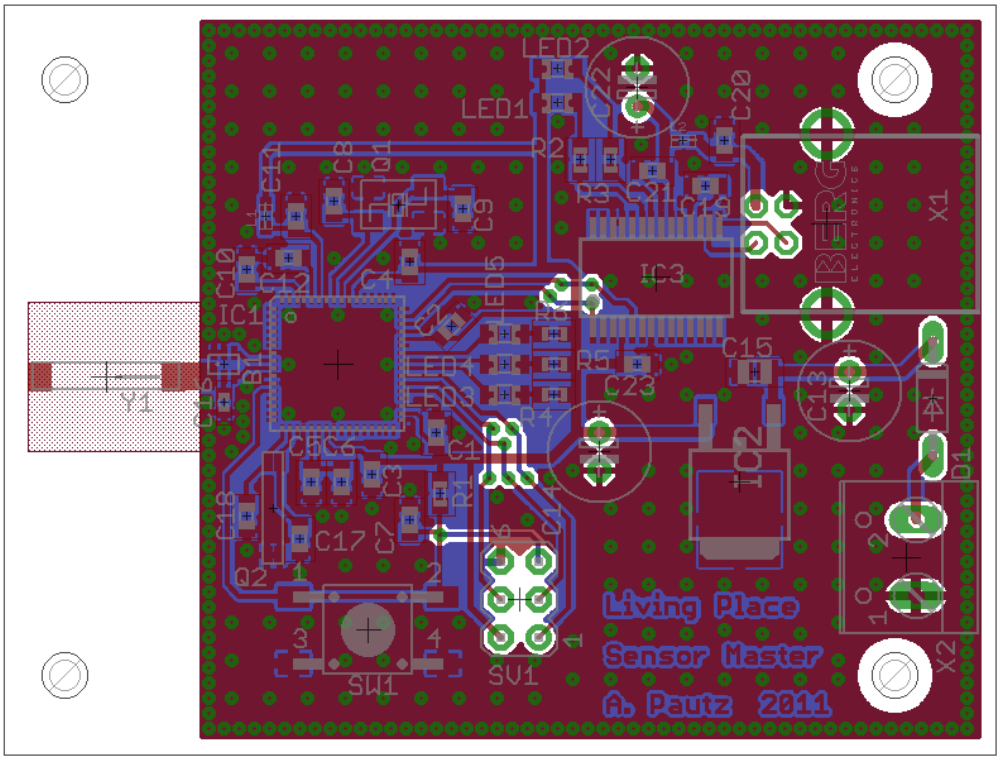

Abbildung B.2.: Layout des Masterknotens (200%)

B.2. Sensorknoten

Die Abbildung B.3 zeigt den Schaltplan des Sensorknotens. Im oberen Bereich ist das ZigBit-Modul, sowie die nötigen Kondensatoren und die Pinheader, über die die Sensoren angeschlossen werden. Im unterem Bereich ist der Step-Up Wandler samt Spule und Kondensatoren.

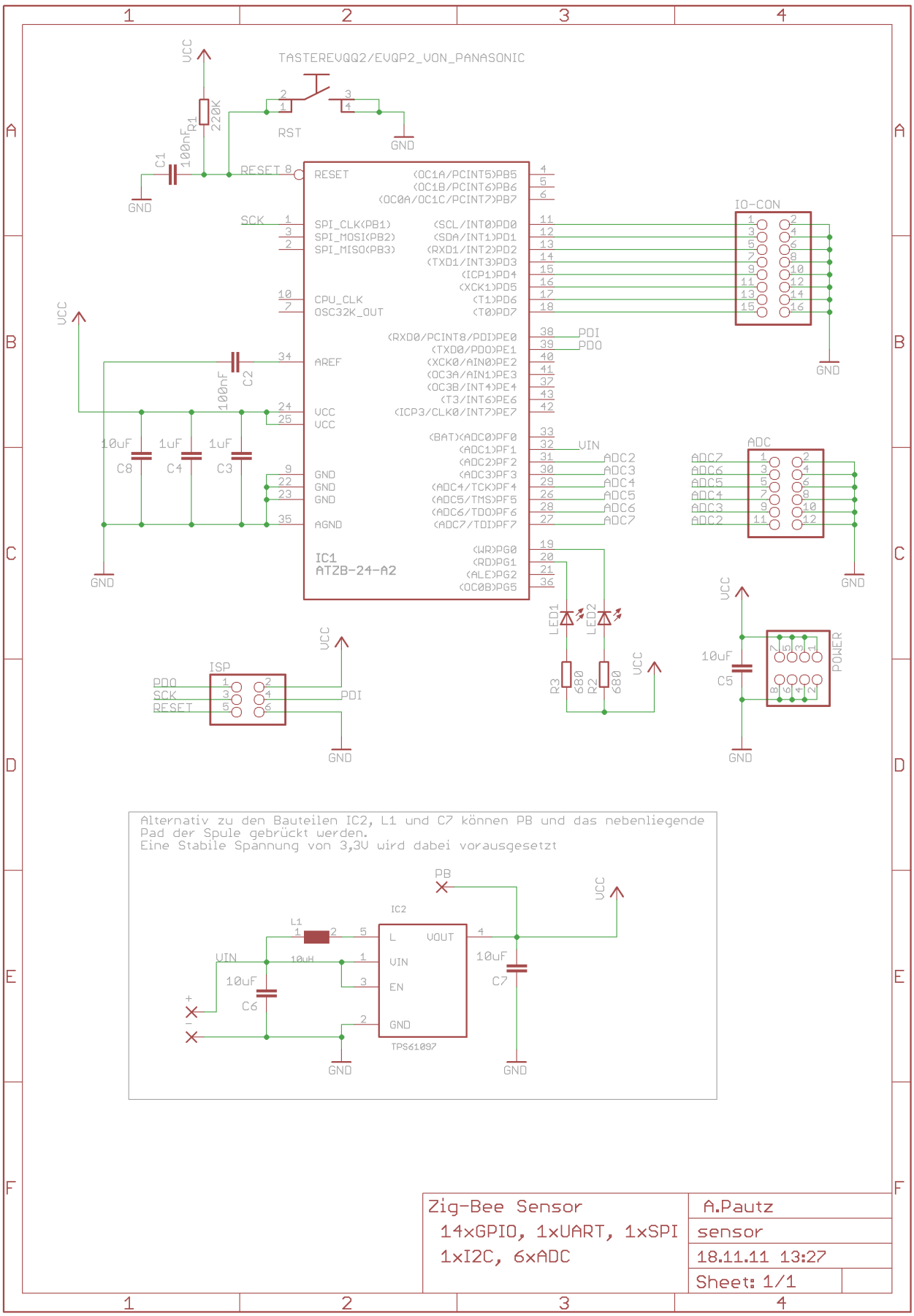

Abbildung B.3.: Schaltplan des Sensorknotens

Die Abbildung B.4 zeigt das Layout des Sensorknotens. Das ZigBit-Modul sitzt zentral. Der Spannungswandler ist oben angeordnet. Die Bauteile sind möglichst kompakt um den Spannungswandler angeordnet, so dass die Strompfade kurz sind und keine elektromagnetischen Abstrahlungen entstehen. Zusätzlich befindet sich unterhalb der Spule keine Massefläche, um eine Einkopplung der durch die Spule erzeugten elektromagnetischen Emissionen auf die Massefläche zu verhindern. Alle Pinheader befinden sich rechts auf der Unterseite der Platine. Auf der Unterseite der Platine ist ausreichend Platz für die Befestigung eines Batteriehalters für zwei Mignon Batterien. Die Platine ist 48,5 x 60 mm groß.

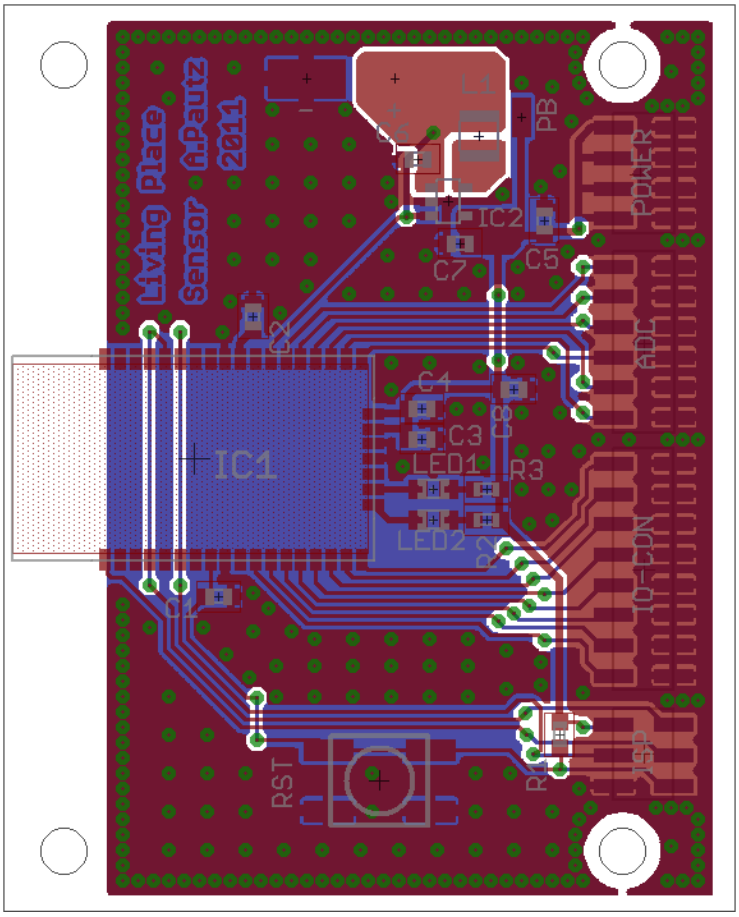

Abbildung B.4.: Layout des Sensorknotens (200%)

B.3. Step-Down Wandler

Die Abbildung B.5 zeigt den Schaltplan des Step-Down Wandlers. Anstelle eines großen Ausgangskondensators wurden zwei kleinere Ausgangskondensatoren eingesetzt. So konnte die Gesamthöhe der Platine niedrig gehalten werden. Um die Baugröße des Eingangskondensators gering zu halten, wurde ein Kondensator mit maximal 35 Volt Betriebsspannung gewählt. Dieses Bauteil besitzt die geringste Spannungsfestigkeit

auf der Eingangsseite und bestimmt somit die maximale Betriebsspannung. Die vom Hersteller angegebenen 35 Volt stellen den zulässigen Maximalwert dar. Dieser darf während des Betriebs nie überschritten werden. Da es während des Betriebs zu Spannungsspitzen kommen kann, sollte die Betriebsspannung unterhalb der 35 Volt bleiben. Eine Betriebspannung von *28 Volt* bietet genügend Sicherheit.

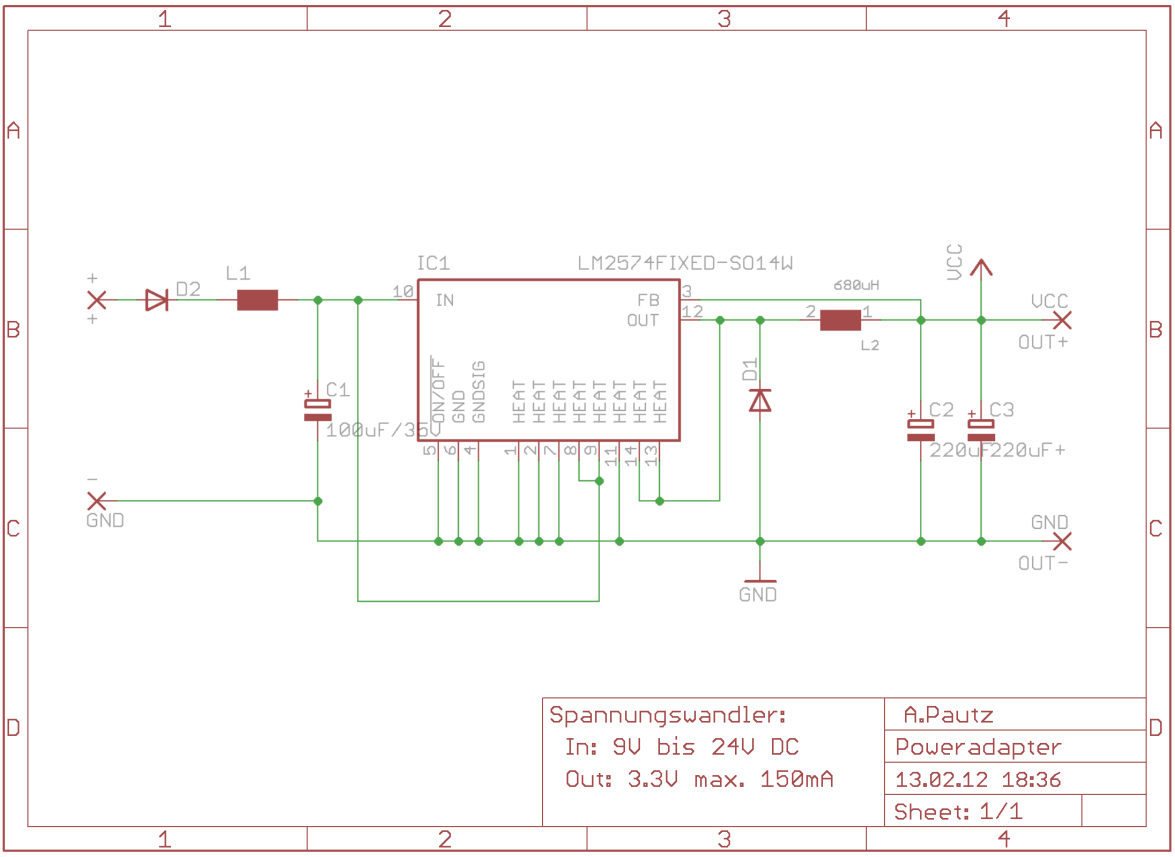

Abbildung B.5.: Schaltplan des Step-Down Wandlers

Die Abbildung B.4 zeigt das Layout des Step-Down Wandlers. Bei dieser Platine wurde auf die Verwendung von sogenannten Thermals[1] verzichtet, so ist eine bessere Verbindung zwischen den Bauteilen und der Platine gewährleistet. Wie beim Step-Up Wandler auf der Sensorknotenplatine wurde unterhalb des Bereichs der Spule die Massefläche ausgespart. Die Platine ist ca. 43,5 x 25,5 mm groß. Sie kann anstelle eines Batteriehalters unterhalb des Sensorknotens montiert werden.

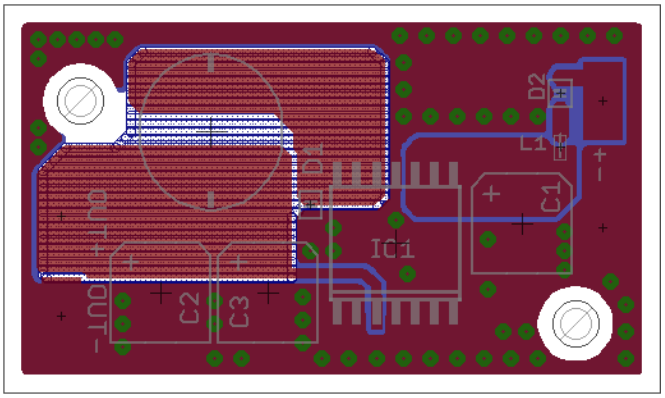

Abbildung B.6.: Layout des Step-Down Wandlers (200%)

[1] Thermals sind Aussparungen um die Pads von Bauteilen auf der Platine. Dadurch wird die Wärmeabfuhr verringert und die Lötbarkeit verbessert.

C. Verwendete Software

In diesem Kapitel wird die gesamte verwendete Software und deren Version angegeben. Bei einer Weiterentwicklung des Sensornetzwerks sollten wieder diese Softwareversionen eingesetzt werden.

Die gesamte Entwicklung wurde unter *Debian GNU/Linux testing (wheezy)* durchgeführt.

Entwicklung der PC-Software

Die PC-Software besteht aus drei verschiedenen Programmen (siehe Kap. 7.4.2). Aufgrund der bereits vorhandenen Software aus früheren Projekten und Software von Kommilitonen wurden unterschiedliche Programmiersprachen eingesetzt.

PC C Entwicklungsumgebung Zur Entwicklung der in C geschriebenen PC-Software wurde *Eclipse 3.5.2 mit CDT Plugin* und der *GNU GCC Compiler 4.6.2* eingesetzt. Abgesehen von der C-Standardbibliothek wurden keine weiteren Bibliotheken eingesetzt.

PC C++ Entwicklungsumgebung Den Kern der PC-Software bildet ein in C++ geschriebenes Programm. Es wurde *Eclipse 3.5.2 mit CDT und QT Plugin* und der *GNU G++ Compiler 4.6.2* verwendet. Zusätzlich zu den Standardbibliotheken wurden die QT Bibliotheken in der Version 4.7.4 eingesetzt.

PC JAVA Entwicklungsumgebung Java wurde ebenso wie die beiden anderen PC-Programme in *Eclipse 3.5.2* entwickelt. Für die Compilierung und Ausführung kam *Sun-JAVA 1.6.0_26* zum Einsatz.

Entwicklung der Software der Mikrocontroller

Die Softwareentwicklung für die Master- und Sensorknoten erfolgte mit *Geany 0.21*. Es wurde der *avr-gcc Version 4.5.3* Compiler eingesetzt. Neben der Standardbibliothek wurde *BitCloud 1.13.0* von Atmel eingesetzt. Zur Programmierung der Microcontroller wurde *avrdude 5.11.1* eingesetzt.

Entwurf der Platinen

Die Erstellung der Schaltpläne und Layouts der Platine erfolgt mit Hilfe der Freeware Version von *Eagle 5.12.0* der Firma Cadsoft[1]. Diese Freeware Version beschränkt die maximale Größe eines PCB (Printed Circuit Board) auf 80 x 100 mm und auf maximal zweilagige Platinen. Zudem darf die Freeware Version nicht kommerziell genutzt werden.

[1] URL: www.cadsoft.de Abruf: 05.12.2011

D. Nachrichten im ZigBee-Netzwerk

In diesem Anhang wird der Aufbau der Nachrichten, welche im ZigBee-Netzwerk versendet werden, beschrieben.

D.1. Rahmen der Nachricht vom Sensorknoten an den PC

Jede Nachricht, welche von einem Sensorknoten an den Masterknoten versendet wird, besitzt den selben Rahmen. Die Nachrichten wurden dabei so aufgebaut, dass sie direkt vom Masterknoten weiter an den PC übertragen werden können. So ist die Knoten ID immer in der Nachricht enthalten, obwohl die Absenderadresse durch das ZigBee-Netzwerk immer mit übermittelt wird.

Länge In diesem Byte wird die *Länge* der gesamten Nachricht ohne diese Längenangabe gespeichert.

Befehl Dieses Byte gibt an, um was für eine Nachricht es sich handelt.

Sensor ID Das folgende Byte gibt die *Sensor ID* an. Im Falle, dass die Nachricht keinem Sensor zugeordnet werden kann, ist diese ID 0x00.

Knoten ID Die zwei nächsten Bytes geben die *Knoten ID* an, von wem die Nachricht stammt. Es wird dabei zuerst das LSByte und dann das MSByte übertragen.

Daten Die nachfolgenden Bytes sind abhängig vom Befehl und werden bei jeder einzelnen Nachricht erklärt.

Die folgende Tabelle D.1 zeigt den Rahmen der Nachricht.

Byte	Inhalt	Beschreibung
0	Länge	Anzahl aller folgenden Bytes.
1	Befehl	Art der Nachricht, die übertragen wird.
2	Sensor ID	Sensor ID, von dem die Nachricht ist
3	Knoten ID LSB	Unteren 8 Bit der ID des Absenders
4	Knoten ID MSB	Oberen 8 Bit der ID des Absenders
4	Länge	Anzahl aller folgenden Bytes
...	Daten	Siehe Beschreibung der einzelnen Befehle

Tabelle D.1.: ZigBee Nachrichtenrahmen

D.2. Nachrichten von den Sensorknoten

Debug und Fehlermeldungen

Die Sensorknoten sind in der Lage Fehler- und Debugmeldungen über den Masterknoten an den PC zu senden. Diese Meldungen werden dann über den Active MQ von der PC-Software ausgegeben.

Das Befehlsbyte der Nachricht lautet *Ascii D* bzw. *Hex 0x44*. Die Länge einer Debug bzw. Fehlermeldung ist variabel. Bei einer Debug bzw. Fehlermeldung ist die Sensor ID immer 0. Die gesamte Nachricht ist als Text zu interpretieren. Das erste Datenbyte ist ein ”D” für Debug oder ”F” für Fehler. Darauf folgt ein ”:”. Die restlichen Datenbytes enthalten die eigentliche Debug- bzw. Fehlermeldung. Das Zeichen vor der CRC ist immer 0x00 und dient als Abschluss der Nachricht. So kann einfacher mit Standard String Funktionen auf diesen Text zugegriffen werden.

Byte	Inhalt	Beschreibung
0	Länge	Anzahl aller folgenden Bytes
1	”D”	Debug
2	0x00	Nachricht kommt von keinem speziellem Sensor
3	Knoten ID LSB	ID des Knotens, von dem die Nachricht ist
4	Knoten ID MSB	ID des Knotens, von dem die Nachricht ist
5	D oder F	D für Debug, F für Fehler
6	:	Fixer Wert
...	Nachricht	Text der Nachricht
N	0x00	Zeichenkettenabschluss

Tabelle D.2.: Debug bzw. Fehlermeldung

Initialisierung von Sensoren

Jeder Sensorknoten gibt beim Start die an ihm angeschlossenen und konfigurierten Sensoren aus. Dabei wird der Sensorname und die Messeinheit des Sensors angegeben.

Das Befehlsbyte der Nachricht lautet *Ascii I* bzw. *Hex 0x49* für ”Initialisierung”. Die Länge der Nachricht ist variabel. Sowohl der Sensorname, als auch die Messeinheit sind null terminierende Zeichenketten. Beide stehen in der Nachricht hintereinander. Die Messeinheit beginnt direkt nach der Termination des Sensornamens. Die Messeinheit darf leer sein. In dem Fall muss jedoch mindestens eine 0x00 für die Terminierung der Zeichenkette vorhanden sein.

Byte	Inhalt	Beschreibung
0	Länge	Anzahl aller folgenden Bytes
1	”I”	Initialisierung
2	Sensor ID	ID des Sensors, dessen Name übermittelt wird
3	Knoten ID LSB	ID des Knotens, von dem die Nachricht ist
4	Knoten ID MSB	ID des Knotens, von dem die Nachricht ist
...	Name	Name des Sensors
M	0x00	Abschluss des Namens
...	Einheit	Messgröße, des Sensors
N	0x00	Zeichenkettenabschluss

Tabelle D.3.: Initialisierung von Sensoren

Ausgabe von Messwerten

Jedes mal, wenn ein Sensor ausgelesen wird, wird ein Messwert zur weiteren Verarbeitung ausgegeben.

Das Befehlsbyte der Nachricht lautet *Ascii M* bzw. *Hex 0x4D* für ”Messwert”. Die Länge der Nachricht hängt von der Art des Messwerts ab und ist variabel. Das erste Datenbyte gibt an, in welchem Format ein Messwert ausgegeben wird. Vorzugsweise werden die Messwerte in Form von Integer oder Fixed Point Zahlen ausgegeben. dies erspart dem Sensorknoten die Verarbeitung von Floating Point. Dennoch ist die Ausgabe von Floating Point Zahlen und sogar Zeichenketten möglich. Welches Format wie interpretiert werden muss, würde den Rahmen dieser Beschreibung überschreiten und ist in der Softwaredokumentation der Sensorknoten beschrieben (siehe Anhang G).

Byte	Inhalt	Beschreibung
0	Länge	Anzahl aller folgenden Bytes
1	”M”	Messwert
2	Sensor ID	ID des Sensors, dessen Name übermittelt wird
3	Knoten ID LSB	ID des Knotens, von dem die Nachricht ist
4	Knoten ID MSB	ID des Knotens, von dem die Nachricht ist
5	Format	Format der Messwertausgabe (siehe Doxygen Doku der Sensorknoten)
...	Messergebnis	Messwertausgabe (siehe Doxygen Doku der Sensorknoten)

Tabelle D.4.: Messwertausgabe

D.3. Nachrichten an die Sensorknoten

Es folgen die Nachrichten, welche vom Masterknoten an die Sensorknoten versandt werden.

Konfiguration der Abfrageintervalle

Über die PC-Software ist es möglich, den Abfrageintervall jedes einzelnen Sensors zu konfigurieren. Der Masterknoten empfängt diese Nachricht und versendet sie auf Endpunkt *103* an die Sensorknoten. Byte 0 der Nachricht ist immer ein ”S”. Dies dient der eindeutigen Identifizierung der Nachricht. So können bei einer Erweiterung der Funktionalität über diesen Endpunkt auch andere Nachrichten versendet werden. Byte 1 (LSByte) und 2 (MSByte) werden zu einem 16 Bit unsigned Integer zusammengesetzt und geben die neue Schlafenszeit in Sekunden an. Das 3. Byte enthält die Sensor ID, für den der neue Abfrageintervall eingestellt werden soll.

Byte	Inhalt	Beschreibung
0	”S”	Identifizierung der Nachricht
1	LSByte Sekunden	Unteren 8 Bit des neuen Abfrageintervall in Sekunden
2	MSByte Sekunden	Oberen 8 Bit des neuen Abfrageintervall in Sekunden
3	Sensor ID	ID des Sensors, dessen Abfrageintervall geändert werden soll

Tabelle D.5.: Konfiguration des Abfrageintervall

E. Kommunikationsprotokoll zwischen Masterknoten und PC

In diesem Anhang wird der Aufbau der Nachrichten zwischen Masterknoten und PC beschrieben.

E.1. Konfiguration der Schnittstelle

Die Kommunikation zwischen Masterknoten und PC erfolgt über eine virtuelle serielle Schnittstelle. Die Geschwindigkeit ist auf 38400 Baud festgelegt. Jedes Byte einer Nachricht besteht aus einem Start-, acht Daten- und einem Stopbit. Es wird kein Parity und keine Flusskontrolle verwendet.

E.2. Rahmen jeder Nachricht

Jede Nachricht besitzt den selben Rahmen.

Präambel Jede Nachricht wird von einer vier Byte langen *Präambel* eingeleitet. Die Präambel besteht aus den Bytes 0xAA, 0x55, 0xAA und 0x55.

Länge Direkt nach der Präambel folgt ein Byte mit der *Länge* der Nachricht. Diese Längenangabe umfasst alle folgenden Bytes inklusive der CRC der Nachricht. Eine Nachricht kann somit maximal 255 Byte plus 4 Byte Päambel lang sein.

Befehl Der Längenangabe folgt ein *Befehl*sbyte. Dieses Byte gibt an, um was für eine Nachricht es sicht handelt.

Sensor ID Das folgende Byte gibt die *Sensor ID* an. Im Falle, dass die Nachricht keinem Sensor zugeordnet werden kann, ist diese ID 0x00.

Knoten ID Die zwei nächsten Bytes geben die *Knoten ID* an. Es wird dabei zuerst das LSByte und dann das MSByte übertragen. Der Masterknoten besitzt immer die ID 0x0000.

Daten Die nachfolgenden Bytes sind abhängig vom Befehl und werden bei jeder einzelnen Nachricht erklärt.

CRC Zur Sicherheit wird jede Nachricht mit einer *CRC* beendet. Die CRC wird berechnet, in dem alle Bytes zwischen der Länge und der CRC mit XOR zusammengefasst werden und abschließend noch einmal mit 0xFF verknüpft werden.

Die folgende Tabelle E.1 zeigt den Rahmen der Nachricht.

Byte	Inhalt	Beschreibung
0	0xAA	Teil der Präambel
1	0x55	Teil der Präambel
2	0xAA	Teil der Präambel
3	0x55	Teil der Präambel
4	Länge	Anzahl aller folgenden Bytes
5	Befehl	Definiert den Inhalt der Nachricht
6	Sensor ID	ID des Sensors, von dem / für den die Nachricht ist
7	Knoten ID LSB	ID des Knotens, von dem / für den die Nachricht ist
8	Knoten ID MSB	ID des Knotens, von dem / für den die Nachricht ist
...	Daten	Siehe Beschreibung der einzelnen Befehle
N	CRC	XOR verknüpfte CRC über alle Daten

Tabelle E.1.: UART Nachrichtenrahmen

E.3. Beschreibung der einzelnen Nachrichten

Im Folgenden werden alle Nachrichten, welche zwischen dem PC und dem Masterknoten ausgetauscht werden aufgelistet.

Debug und Fehlermeldungen

Der Master- und die Sensorknoten sind in der Lage Fehler- und Debugmeldungen an den PC zu senden. Diese Meldungen werden dann über den Active MQ von der PC Software ausgegeben.

Das Befehlsbyte der Nachricht lautet *Ascii D* bzw. *Hex 0x44*. Die Länge einer Debug- bzw. Fehlermeldung ist variabel. Bei einer Debug- bzw. Fehlermeldung ist die Sensor ID immer 0. Die gesamte Nachricht ist als Text zu interpretieren. Das erste Datenbyte ist ein ”D” für Debug oder ”F” für Fehler. Darauf folgt ein ”:”. Die restlichen Datenbytes enthalten

die eigentliche Debug- bzw. Fehlermeldung. Das Zeichen vor der CRC ist immer 0x00 und dient als Abschluss der Nachricht. So kann einfacher mit Standard String Funktionen auf diesen Text zugegriffen werden.

Byte	Inhalt	Beschreibung
0	0xAA	Teil der Präambel
1	0x55	Teil der Präambel
2	0xAA	Teil der Präambel
3	0x55	Teil der Präambel
4	Länge	Anzahl aller folgenden Bytes
5	”D”	Debug
6	0x00	Nachricht kommt von keinem speziellem Sensor
7	Knoten ID LSB	ID des Knotens, von dem die Nachricht ist
8	Knoten ID MSB	ID des Knotens, von dem die Nachricht ist
9	D oder F	D für Debug, F für Fehler
10	:	Fixer Wert
...	Nachricht	Text der Nachricht
N-1	0x00	Zeichenkettenabschluss
N	CRC	XOR verknüpfte CRC über alle Daten

Tabelle E.2.: Debug bzw. Fehlermeldung

Initialisierung von Sensoren

Jeder Sensorknoten gibt beim Start die an ihm angeschlossenen und konfigurierten Sensoren aus. Dabei wird der Sensorname und die Messeinheit des Sensors angegeben.

Das Befehlsbyte der Nachricht lautet *Ascii I* bzw. *Hex 0x49* für ”Initialisierung”. Die Länge der Nachricht ist variabel. Sowohl der Sensorname, als auch die Messeinheit sind null terminierende Zeichenketten. Beide stehen in der Nachricht hintereinander. Die Messeinheit beginnt direkt nach der Termination des Sensornamens. Die Messeinheit darf leer sein. In dem Fall muss jedoch mindestens eine 0x00 für die Terminierung der Zeichenkette vorhanden sein.

Byte	Inhalt	Beschreibung
0	0xAA	Teil der Präambel
1	0x55	Teil der Präambel
2	0xAA	Teil der Präambel
3	0x55	Teil der Präambel
4	Länge	Anzahl aller folgenden Bytes
5	”I”	Initialisierung
6	Sensor ID	ID des Sensors, dessen Name übermittelt wird
7	Knoten ID LSB	ID des Knotens, von dem die Nachricht ist
8	Knoten ID MSB	ID des Knotens, von dem die Nachricht ist
...	Name	Name des Sensors
M	0x00	Abschluss des Namens
...	Einheit	Messgröße, des Sensors
N-1	0x00	Zeichenkettenabschluss
N	CRC	XOR verknüpfte CRC über alle Daten

Tabelle E.3.: Initialisierung von Sensoren

Ausgabe von Messwerten

Jedes mal wenn ein Sensor ausgelesen wird, wird ein Messwert zur weiteren Verarbeitung ausgegeben.

Das Befehlsbyte der Nachricht lautet *Ascii M* bzw. *Hex 0x4D* für ”Messwert”. Die Länge der Nachricht hängt von der Art des Messwerts ab und ist variabel. Das erste Datenbyte gibt an, in welchem Format ein Messwert ausgegeben wird. Vorzugsweise werden die Messwerte in Form von Integer oder Fixed Point Zahlen ausgegeben, dies erspart dem Sensorknoten die Verarbeitung von Floating Point. Dennoch ist die Ausgabe von Floating Point Zahlen und sogar Zeichenketten möglich. Welches Format wie interpretiert werden muss, würde den Rahmen dieser Beschreibung überschreiten und ist in der Softwaredokumentation der Sensorknoten beschrieben (siehe Anhang G).

Byte	Inhalt	Beschreibung
0	0xAA	Teil der Präambel
1	0x55	Teil der Präambel
2	0xAA	Teil der Präambel
3	0x55	Teil der Präambel
4	Länge	Anzahl aller folgenden Bytes
5	”M”	Messwert
6	Sensor ID	ID des Sensors, dessen Name übermittelt wird
7	Knoten ID LSB	ID des Knotens, von dem die Nachricht ist
8	Knoten ID MSB	ID des Knotens, von dem die Nachricht ist
10	Format	Format der Messwertausgabe (siehe Doxygen Doku der Sensorknoten)
...	Messergebnis	Messwertausgabe (siehe Doxygen Doku der Sensorknoten)
N	CRC	XOR verknüpfte CRC über alle Daten

Tabelle E.4.: Messwertausgabe

Konfiguration des Sensorabfrageintervalls

Mit dieser Nachricht kann der Abfrageintervall von Sensoren eingestellt werden.

Das Befehlsbyte der Nachricht lautet *Ascii S* bzw. *Hex 0x53* für ”Schlafen”. Die Länge der Nachricht ist immer 7. Es gibt zwei Datenbytes, welche angeben, wie lang die Pause zwischen zwei Abfragen ist. Die Zeitangabe ist in Sekunden zu machen

Byte	Inhalt	Beschreibung
0	0xAA	Teil der Präambel
1	0x55	Teil der Präambel
2	0xAA	Teil der Präambel
3	0x55	Teil der Präambel
4	7	Anzahl aller folgenden Bytes
5	”S”	Schlafenszeit
6	Sensor ID	ID des Sensors, dessen Abfragezyklus konfiguriert wird
7	Knoten ID LSB	ID des Knotens, für den die Nachricht ist
8	Knoten ID MSB	ID des Knotens, für den die Nachricht ist
10	LSByte	LSByte des Abfrageintervall (Sekunden)
11	MSByte	MSByte des Abfrageintervall (Sekunden)
N	CRC	XOR verknüpfte CRC über alle Daten

Tabelle E.5.: Konfiguration des Sensorabfrageintervall

F. JSON-Nachrichten

In diesem Anhang wird der Aufbau der JSON-Nachrichten, welche zwischen den PC-Programmen des Sensornetzwerk und über den Active MQ versendet werden.

Zunächst wird der kurz der allgemeine Aufbau einer JSON-Nachricht erklärt. Anschließend werden die Nachrichten erklärt. Eine Erklärung der Schnittstellen findet sich in Kapitel 7.4.2.

F.1. Aufbau von JSON-Nachrichten

JSON ist ein Textbasiertes Format für den Datenaustausch. Jede JSON-Nachricht wird von einer öffnenden geschweiften Klammer eingeleitet und durch eine schließende geschweifte Klammer beendet. Innerhalb dieser Klammern befinden sich die zu übermittelnden Informationen in Form von ein oder mehreren Key-Value Paaren. Einzelne Key-Value Paare einer Nachricht werden durch ein Komma voneinander getrennt. Der Schlüssel und der Wert sind durch einen Doppelpunkt getrennt. Der Name des Schlüssel ist immer in Anführungszeichen eingefasst. Beim Wert werden nur Zeichenketten in Anführungszeichen eingefasst. Zahlen und Boolean werden nicht besonders eingefasst.

Im folgenden Listing F.1 ist eine JSON-Nachricht abgebildet.

```
1 {
2     ''key1'' : ''value'',
3     ''key2'' : 123,
4     ''key3'' : false
5 }
```

Abbildung F.1.: Beispiel einer JSON-Nachricht

Das Sensornetzwerk kann die Werte vom Typ String, Integer, Double und Boolean verarbeiten. Arrays und geschachtelte JSON-Nachrichten werden nicht verarbeitet. Intern werden JSON-Nachrichten in einem Hash gespeichert. Dies ermöglicht eine schnelle Verarbeitung, führt jedoch dazu, dass die Reihenfolge der Key-Value Paare bei der Ausgabe nicht bestimmt werden kann.

Eine ausführliche Beschreibung des JSON-Format ist unter http://www.json.org verfügbar (Abruf 24.02.12).

F.2. Nachrichten über die In-Server Schnittstelle

Die folgenden Nachrichten werden über die In-Server Schnittstelle ausgetauscht. Im Allgemeinen sind dies Nachrichten zwischen dem TCP/IP-RS232 Adapter und der Sensornetzwerkverwaltung (siehe Kap. 7.4.2).

Messwerteingabe

Mit der folgenden Nachricht werden neue Messwerte zur weiteren Verarbeitung an die Sensornetzwerkverwaltung übertragen.

Key	Value	Beschreibung
”cmd”	”measurement”	Gibt die Art des Befehls an
”sensorid”	Integer	ID des Sensors, dessen Messwert übermittelt wird
”value”	Messwert	Messwert als Double, Integer, String oder Boolean
”valuet”	”i”, ”b”, ”s”	Gibt das Format des Messwerts an

Tabelle F.1.: JSON - Messwerteingabe

Debugmeldungen

Mit der folgenden Nachricht werden Debug- und Fehlermeldungen der Sensorknoten an die Sensornetzwerkverwaltung übertragen.

Key	Value	Beschreibung
”cmd”	”message”	Gibt die Art des Befehls an
”sensorid”	Integer	ID des Sensors, dessen Meldung übermittelt wird
”msg”	Meldung	Text der Meldung
”type”	”debug” oder ”error”	Art der Meldung; Fehler oder Debug

Tabelle F.2.: JSON - Debug- und Fehlermeldungen

Initialisierungsnachricht

Mit der folgenden Nachricht werden Informationen über neue Sensoren übertragen. Diese Nachricht gibt jeder Sensorknoten für jeden Sensor beim Start aus.

Key	Value	Beschreibung
”cmd”	”new sensor”	Gibt die Art des Befehls an
”sensorid”	Integer	ID des Sensors, der neu Initialisiert wird
”name”	Name	Name des Sensors
”unit”	Einheit	Einheit, in dem die Messwerte in Zukunft übertragen werden

Tabelle F.3.: JOSN - Initialisierungsnachricht

Schlafzeitkonfiguration

Mit der folgenden Nachricht wird der Abfragezyklus der Sensoren eingestellt. Diese Nachricht wird von der Sensornetzwerkverwaltung an die Sensorknoten übertragen.

Key	Value	Beschreibung
”cmd”	”sleep”	Gibt die Art des Befehls an
”sensorid”	Integer	ID des Sensors, dessen Abfragezyklus eingestellt werden soll
”value”	Zeit	Zeit in Sekunden

Tabelle F.4.: JSON - Schlafzeitkonfiguration

F.3. Nachrichten über der CFG Schnittstelle

Über diese Schnittstelle gibt die Sensornetzwerkverwaltung Statusmeldungen aus. Zudem kann über diese Schnittstelle der Abfragezyklus der Sensoren konfiguriert werden. Sie ist über den Active MQ Server verfügbar.

Schlafzeitkonfiguration

Mit dieser Nachricht kann andere Software den Abfragezyklus der Sensoren einstellen. Diese Nachricht wird dann von der Sensornetzwerkverwaltung an die Sensorknoten übertragen.

Key	Value	Beschreibung
”cmd”	”set sleep”	Gibt die Art des Befehls an
”id”	Eindeutige ID	Identifiziert die Nachricht, so kann der Absender die Antwort identifizieren
”name”	Sensorname	Name des angesprochenen Sensors
”time”	Abfrageintervall	Neuer Abfrageintervall in Sekunden

Tabelle F.5.: Active MQ - Schlafzeitkonfiguration

Lösche Sensor

Diese Nachricht löscht einen Sensor aus dem System inklusive aller bisher aufgenommenen Messwerte. Ist der Sensor noch vorhanden, produzieren alle neuen Messwerte dieses Sensors Fehlermeldungen. Der Sensor wird automatisch wieder ins System aufgenommen, wenn der Sensorknoten, an dem der Sensor angeschlossen ist, neu gestartet wird. Bei dieser Funktion wird zur Sicherheit der Sensorname und die intern verwendete ID abgefragt. Die Sensor ID kann über den Befehl ”get sensors” erfahren werden (siehe Nachricht F.3).

Key	Value	Beschreibung
”cmd”	”del sensor”	Gibt die Art des Befehls an
”id”	Eindeutige ID	Identifiziert die Nachricht, so kann der Absender die Antwort identifizieren
”name”	Sensorname	Name des zu löschenden Sensors
”sensorid”	Sensor ID	ID des zu löschenden Sensors

Tabelle F.6.: Active MQ - Lösche Sensor

Abfrage aller Sensoren

Mit dieser Nachricht kann erfragt werden, welche Sensoren vorhanden sind. Die Antwort auf diese Nachricht ist unter ”Vorhandene Sensoren” erklärt.

Key	Value	Beschreibung
”cmd”	”get sensors”	Gibt den Befehls an
”id”	Eindeutige ID	Identifiziert die Nachricht, so kann der Absender die Antwort identifizieren

Tabelle F.7.: Active MQ - Abfrage aller Sensoren

Vorhandene Sensoren

Diese Nachricht wird von der Sensorverwaltungssoftware als Antwort gesendet, wenn sie mit der Nachricht ”Abfrage aller Sensoren” abgefragt wurden. Für jeden Sensor wird eine Nachricht versendet.

Key	Value	Beschreibung
”sensorid”	Sensor ID	ID des aktuell ausgegebenen Sensors
”name”	Sensorname	Name des aktuell ausgegebenen Sensors
”unit”	Messgröße	Einheit, in der der Sensor seine Messwerte ausgibt; dieser Wert kann leer sein
”desc”	Beschreibung	Beschreibung des Sensors; dieser Wert kann leer sein
”answer”	”get sensors”	Fix; gibt die Art der Nachricht an
”id”	ID der Anfrage	Dieser Wert ist gleich der bei der Anfrage gesendeten id

Tabelle F.8.: Active MQ - Vorhandene Sensoren

Sensorbeschreibung

Mit dieser Nachricht kann für jeden Sensor eine Beschreibung gespeichert werden. Ist bereits eine Beschreibung vorhanden, wird die alte Beschreibung durch die neue ersetzt.

Key	Value	Beschreibung
”cmd”	”Set desc”	Gibt den Befehl an
”name”	Sensorname	Name des Sensors, dessen Beschreibung geändert werden soll
”description”	Beschreibung	Neue Beschreibung des Sensors
”id”	Eindeutige ID	Identifiziert die Nachricht, so kann der Absender die Antwort identifizieren

Tabelle F.9.: Active MQ - Sensorbeschreibung

Ausgabe von Meldungen

Mit dieser Nachricht werden alle Fehler-, Debug- und sonstige Statusmeldungen vom Sensornetzwerk ausgegeben.

Key	Value	Beschreibung
”type”	Art der Nachricht	Gibt die Art der Nachricht an. Siehe unten
”modul”	Modul	Gibt an, aus welcher Komponente der Software die Nachricht stammt
”msg”	Nachricht	Text der Nachricht
”id”	Eindeutige ID	Ist leer, wenn die Nachricht vom Sensornetzwerk selbst erzeugt wurde oder die selbe ID der Anfrage, auf die geantwortet wird.
”unixdate”	Datum und Uhrzeit	Datum und Uhrzeit im Unixformat. Sekunden seit dem 1. Januar 1970
”date”	Datum und Uhrzeit	Datum und Uhrzeit als Zeichenkette

Tabelle F.10.: Active MQ - Ausgabe von Meldungen

Nachrichtentyp

Der Nachrichtentyp gibt an, um was für eine Nachricht es sich handelt. Es sind folgende Nachrichtentypen definiert.

Value	Beschreibung
”fatal”	Es ist ein schwerwiegender Fehler aufgetreten. Das Sensornetzwerk beendet sich nach dem Absetzen dieser Nachricht. Derzeit wird die Nachricht nur gesendet, wenn nicht mehr genügend Arbeitsspeicher zur Verfügung steht.
”error”	Es ist ein Fehler aufgetreten. Das Sensornetzwerk kann das Ereignis, dass diesen Fehler ausgelöst hat, nicht weiter verarbeiten. Die weitere Arbeit des Sensornetzwerks ist jedoch nicht beeinträchtigt.
”info”	Statusinformation. Wird zum Beispiel ausgegeben, wenn ein neuer Sensor ins Netzwerk aufgenommen wurde.
”debug”	Debugmeldungen, die der Kontrolle der Programmierung dienen. Die Ausgabe dieser Nachrichten kann beim Übersetzen des Programms abgeschaltet werden.

Tabelle F.11.: Active MQ - Meldungsausgabe Nachrichtentyp

F.4. Nachrichten über die Out-Server Schnittstelle

Über diese Schnittstelle gibt die Sensornetzwerkverwaltung alle erfassten Messwerte aus. Zusätzlich können über diese Schnittstelle Messwerte aus der Vergangenheit abgerufen werden. Diese Schnittstelle ist über den Active MQ Server verfügbar.

Messwertausgabe

Diese Nachricht gibt einen Messwert eines Sensors aus.

Key	Value	Beschreibung
”name”	Sensorname	Name des Sensors, dessen Messwert ausgegeben wird
”unit”	Messgröße	Einheit, in der der Messwert ausgegeben wird; kann leer sein
”valueb”	Messwert	Repräsentation des Messwerts als Boolean
”valuei”	Messwert	Repräsentation des Messwerts als Integer oder Double
”values”	Messwert	Repräsentation des Messwerts als String
”valuet”	”i”, ”b”, ”s”	Format, in dem der Messwert am besten zu interpretieren ist (siehe unten)
”unixdate”	Datum und Uhrzeit	Datum und Uhrzeit der Messung im Unixformat; diese Zeit kann in der Vergangenheit liegen
”date”	Datum und Uhrzeit	Datum und Uhrzeit als Zeichenkette
”id”	”” oder ID der Anfrage	Wenn es ein gerade aufgenommener Messwert ist, ist die ID leer. Wurde der Messwert durch eine Anfrage abgefragt, enthält das Feld die ID der Anfrage

Tabelle F.12.: Active MQ - Messwertausgabe

Interpretation des Messwerts

Da unterschiedliche Sensortypen ihre Werte am besten auf unterschiedliche Art und Weise ausgeben, wird der Sensorwert dreimal mit unterschiedlichen Datentypen ausgegeben. Valuet gibt dabei an, wie der Wert am besten ausgewertet wird. Die beiden anderen Felder enthalten möglichst gut abgebildete Werte.

Typ	valuet	valuei	valueb	values
String	”s”	Länge des Strings	false, wenn der String leer ist; true, wenn der String nicht leer ist	Die entsprechende Zeichenkette
Integer	”i”	Der Wert	false, wenn der Wert gleich 0 ist; true, wenn der Wert ungleich 0 ist	Den Wert ausgeschrieben als Zahl z.B.: ”125”
Boolean	”b”	0, wenn false; 1, wenn true	false bzw. true	”true” oder ”false” ausgeschrieben

Tabelle F.13.: Active MQ - Messwertausgabe - Interpretation des Messwerts

Abfrage von Messwerten aus der Vergangenheit

Mit der folgenden Nachricht können früher aufgenommene Messwerte abgefragt werden.

Key	Value	Beschreibung
”name”	Sensorname	Name des Sensors, dessen Messwert abgefragt wird
”date”	Datum und Uhrzeit	Datum und Uhrzeit zu dem der Messwert aufgenommen wurde. Der Wert muss als Unixzeit übergeben werden
”id”	Eindeutige ID	Diese ID wird bei der Antwort wieder mitgesendet; so ist eine Zuordnung zwischen Abfrage und Ausgabe möglich

Tabelle F.14.: Active MQ - Messwertabfrage

Wenn für das Datum und die Uhrzeit keine Messung vorhanden ist, wird die Messung ausgegeben, die als nächstes nach der angegebenen Uhrzeit erfasst wurde. Um zum Beispiel alle Messwerte eines Sensors abzufragen, kann bei der ersten Abfrage eine 0 im ”date”-Feld übergeben werden. Dann wird der erste erfasste Messwert dieses Sensors ausgegeben. Die nächste Anfrage kann anschließend mit der zurückgelieferten Zeit plus eine Sekunde erfolgen. So können alle Messwerte eines Sensors abgefragt werden.

G. DVD Inhalt

Dieser Masterarbeit ist eine DVD beigelegt, auf der sich Quelltexte, die Schaltpläne und eine Reihe weiterer Dokumente befinden. Dieser Anhang listet alle auf DVD vorhandenen Dokumente auf.

Software

Die gesamte während dieser Arbeit entwickelte Software befindet sich auf der beigelegten DVD. Sie ist im Ordner ”Software” zu finden.

Die folgende Tabelle G.1 enthält eine Auflistung aller Softwareprojekte mit.

Ordner	Beschreibung
workspace/mqconnector	Verbindet das Sensornetzwerk mit dem Active MQ
workspace/Sensornetzwerk	Kernteil der PC Software
workspace/SerialJson	Verbindet den Masterknoten mit dem Sensornetzwerk
BitCloud_MEGARF_1_11_0	Software des Masterknotens
BitCloud_ZIGBIT_1_13_0	Software der Sensorknoten

Tabelle G.1.: Liste der entwickelten Software

Eine Liste der verwendeten Entwicklungswerkzeuge ist in Anhang C zu finden.

Dokumentation der Software

Die Dokumentation aller Softwarekomponenten wurde mit Doxygen[1] erstellt. Die gesamte Softwaredokumentation befindet sich in Form von HTML und PDF-Dokumenten auf der DVD im Unterordner ”Softwaredokumentation”.

[1] Informationen zu Doxygen befinden sich unter: www.doxygen.org - Abruf 28.02.2012

Schaltpläne und Layouts

Für den Master- und die Sensorknoten, sowie den Step-Down Wandler wurden Schaltpläne gezeichnet. Auf der Basis dieser Schaltpläne wurden Layouts erstellt. Sowohl die Schaltpläne, als auch die Layouts sind auf der DVD im Ordner ”eagle” zu finden.

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §22(4) bzw. §24(4) ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 8. März 2012
Ort, Datum

Unterschrift