AG VERNETZTE SYSTEME
FACHBEREICH INFORMATIK
AN DER TECHNISCHEN UNIVERSITÄT
KAISERSLAUTERN

BACHELORARBEIT

ENTWICKLUNG EINES ENERGIESPARSAMEN SIGNALISIERUNGSSYSTEMS FÜR MOBILE SENSORKNOTEN

Christian Wolschke

30. September 2009

Entwicklung eines energiesparsamen Signalisierungssystems für mobile Sensorknoten

Bachelorarbeit

Arbeitsgruppe Vernetzte Systeme
Fachbereich Informatik
Technische Universität Kaiserslautern

Christian Wolschke

Tag der Ausgabe : 01. August 2009
Tag der Abgabe : 30. September 2009

Themensteller : Prof. Dr. Reinhard Gotzhein
weiterer Prüfer : Prof. Dr. Jens Schmitt
Betreuer : Marc Krämer

Ich erkläre hiermit, die vorliegende Bachelorarbeit selbständig verfasst zu haben. Die verwendeten Quellen und Hilfsmittel sind im Text kenntlich gemacht und im Literaturverzeichnis vollständig aufgeführt.

Kaiserslautern, 30. September 2009

(Christian Wolschke)

Kurzfassung

Netzwerke aus mobilen Sensorknoten erlangen immer größere Bedeutung. Sensorknoten werden häufig mit Batterien betrieben, daher ist ein geringer Energieverbrauch gewollt, um möglichst lange Batterielaufzeiten zu ermöglichen. Wir gehen davon aus, dass in einem Netzwerk mehrere Sensorknoten per Funk miteinander kommunizieren und die Sensorknoten zwischen den Funkkommunikationen in einem Schlafmodus verweilen können. Bei den von uns verwendeten Imote2-Sensorknoten, wird der Energiebedarf eines Sensorknotens wesentlich von dem integrierten Funkempfänger und von dem angenommenen Schlafmodus beeinflusst. Wartet der Imote2 auf eine Funkkommunikation, so verbraucht er relativ viel Energie, da der Funkempfänger einen hohen Verbrauch hat und nicht der tiefeste Schlafmodus gewählt werden kann. Es wurde daher ein Signalisierungssystem entwickelt, dass mit Hilfe eines zusätzlichen und deutlich energiesparsameren Funkempfängers Aufwecksignale empfangen kann. Dieser ermöglicht, dass der Imote2-Sensorknoten seinen integrierten Funkempfänger deaktivieren kann und währenddessen einen deutlich tieferen Schlafzustand annehmen kann. Das Signalisierungssystem ist in der Lage den Imote2 nur dann aufzuwecken, wenn der Imote2 benötigt wird. Es wurde untersucht, inwiefern das Signalisierungssystem zu einer Energieeinsparung führt.

Abstract

The importance of wireless sensor networks (WSN) is increasing. Sensor nodes are often run by batteries. This implies that a low energy consumption is wanted in order to achieve long battery runtimes. We suppose, that the sensor nodes communicate via wireless links and that the sensor nodes stay in a sleeping mode while no communication takes place. At the used Imote2 sensor nodes, energy consumption is mainly influenced by the integrated receiver and the adopted sleeping mode. If the Imote2 is waiting for a wireless communication, it consumes relatively much energy, as the receiver consumes much energy and as not the deepest sleeping mode could be adopted. Hence, a signaling system has been developed, which uses an additional low power receiver in order to receive wake up signals. This system allows the Imote2 to deactivate its integrated receiver while adopting a much deeper sleeping mode. By this, the signaling system is able to wake up the Imote2 only if the Imote2 is needed. It has been analysed, how the signaling system contributes to energy savings.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Mobile Sensorknoten besitzen vielfältige Einsatzfelder. Exemplarisch seien genannt: Die Verwendung in intelligenter Kleidung, um Personen überwachen zu können oder die Verwendung in intelligenten Büroumgebungen, um Licht und Temperatur an die Tageszeit und Anzahl der Personen im Raum anzupassen. Für weitere Einsatzmöglichkeiten siehe Haenggi [15].

Unter einem Sensorknoten wollen wir in diesem Zusammenhang, wie Haenggi [15], eine Einheit sehen, welche die Fähigkeiten hat an einer Funkkommunikation teilzunehmen, mit Sensoren Daten zu erheben und die diese Daten mit einer Recheneinheit verarbeiten kann. Ferner zeichnen sich Sensorknoten durch eine geringe Größe aus, verglichen mit herkömmlichen Personal Computern. In dieser Bachelorarbeit liegt die Betrachtung auf der Untermenge der mobilen Sensorknoten. Wir gehen dabei davon aus, dass die Sensorknoten mobil sind, damit man sie komfortabel an bestimmten Orten befestigen kann. An diesen Orten bleibt der Sensorknoten stationär und arbeitet im Batteriebetrieb. Er ist also nicht auf eine Stromversorgung durch das öffentliche Stromnetz angewiesen und kann auch in freier Natur eingesetzt werden.

Durch den Batteriebetrieb und die Einbettung in einen unter Umständen schwer zugänglichen Bereich ist ein einfaches Wechseln der Batterie meist nicht möglich. Daher werden Lebensdauern von mehreren Jahren, wenn nicht gar Jahrzehnten, als wünschenswert angesehen. Wenn man auch in Zukunft von nicht wesentlich leistungsstärkeren Batterien ausgeht [23], so besteht eine Verlängerung der Nutzungsdauer nur durch ein mögliches Aufladen der Batterie, wozu es aber einer zusätzlichen Hardware bedarf (beispielsweise Solarzellen) oder durch eine Einschränkung im Verbrauch. Die verwendeten Imote2-Sensorknoten besitzen dazu mehrere Schlafmodi, in denen sie weniger Energie verbrauchen als im gewöhnlichen Betrieb.

Da ein Sensorknoten allein nicht immer ausreicht, um eine komplexe Anwendung zu realisieren, liegt die Betrachtung daher auf Drahtlosen-Funk-Sensornetzwerken (Englisch: Wireless-Sensor-Networks, kurz: WSN). Wir gehen davon aus, dass bis zu 16 Imote2 in einem Netzwerk zusammenarbeiten. Die Imote2 arbeiten im Batteriebetrieb und können mit IEEE-802.15.4-Transceivern miteinander kommunizieren. Diese Transceiver benötigen relativ viel Energie, selbst dann wenn keine Daten ausgetauscht werden und nur der Funkkanal auf neue Daten abgehört wird. Der Prozessor des Imote2 muss dazu auch in Bereitschaft gehalten werden. Die Folgen

dieser Kommunikationsbereitschaft sind ein hoher Stromverbrauch und damit einhergehend eine geringe Batterielebensdauer.

In dieser Arbeit wird ein Signalisierungssystem entwickelt, in welchem der Imote2 nur dann kommunikationsbereit ist, wenn Daten für den Imote2 bereitliegen. Der Imote2-Prozessor kann, solange er nicht vom Signalisierungssystem geweckt wird, in der Zwischenzeit einen energiesparsameren Schlafzustand einnehmen, als er es in der Kommunikationsbereitschaft könnte. Mit diesem Signalisierungssystem soll dadurch eine Energieersparnis eintreten. Umgesetzt wurde das Signalisierungssystem durch eine zusätzliche Hardwarekomponente. Damit dieses System die Energieeinsparungen des Imote2 nicht durch den eigenen Energieverbrauch wieder zunichte macht ist es besonders wichtig, wenig Energie zu verbrauchen. Den Imote2-Sensorknoten, die schon über Funkmodule verfügen, wurde dazu noch ein zusätzlicher energiesparsamerer Receiver und ein zusätzlicher Transmitter zur Verfügung gestellt. Die Signalisierung erfolgt auf einem anderen Frequenzband als die eigentliche Kommunikation, so dass das Signalisierungssystem nicht mit der Funkkommunikation der Sensorknoten interferiert. Dies wird Band-Externe-Signalisierung genannt (Englisch: Out-Of-Band Signaling). Darüber hinaus wird ein Mikrocontroller zur Vorverarbeitung der Daten verwendet. Der verwendete Sensorknoten bietet zwar sehr viel Rechenleistung an, ist aber für die Verarbeitung der Funkkommunikation nicht so gut anpassbar, wie der Mikrocontroller.

Eine Veranschaulichung ist in Abbildung 1.1 zu sehen. Gezeigt sind zwei Imote2, an denen jeweils mittels eines I²C-Busses ein Signalisierungssystem angeschlossen ist. Mittels des Busses kann beispielsweise der linke Imote2 dem Mikrocontroller seinen Aufweckwunsch für den rechten Imote2 mitteilen. Der Mikrocontroller benutzt den an ihn angeschlossenen Transmitter zum Mitteilen des Übertragswunsches. Der Mikrocontroller des rechten Imote2 stellt mit seinem Receiver fest, dass der rechte Imote2 geweckt werden soll und sorgt daraufhin für das Aufwecksignal an dem rechten Imote2. Im folgenden können die beiden Einheiten über ihre IEEE-802.15.4-Transceivern kommunizieren und Daten austauschen. Eine Übertragung von größeren Datenmengen über das Signalisierungssystem ist ungeeignet, da die Übertragungsraten des Signalisierungssystems zu gering sind.

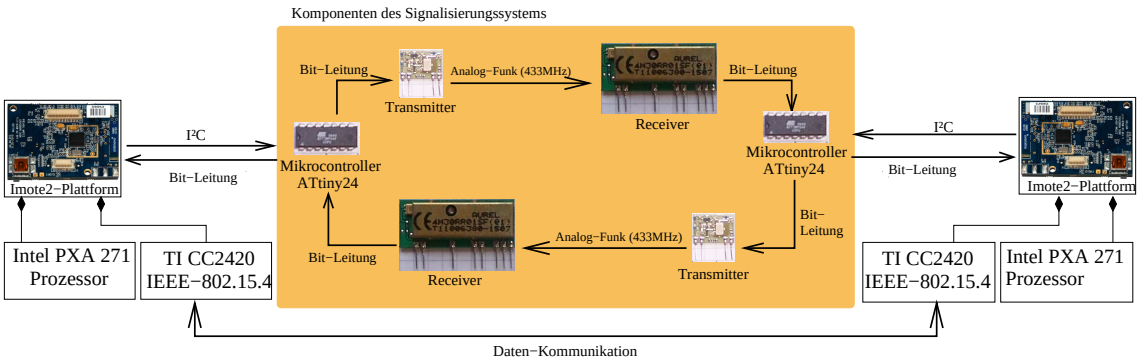


Abbildung 1.1: Bild des Signalisierungssystems.

Zunächst werden weitere mögliche Anwendungsszenarien in Kapitel 2 präsentiert. Insbesondere das Routing-Szenario wird hierbei näher betrachtet, um durch ein An-

wendungsbeispiel eine höhere Verständlichkeit zu erreichen. Auf den Einsatz ähnlicher Signalisierungssysteme wird in Kaptiel 3 eingegangen. In Kapitel 4 folgt eine Beschreibung der Hardware des Signalisierungsystems. Daraufhin werden in Kapitel 5 die Fähigkeiten und insbesondere auch die Schnittstellen des Signalisierungssystems im Detail vorgestellt. Anschließend werden in Kapitel 6 die Details des Entwurfs gezeigt. In Kapitel 7 wird eine Evaluation des Systems vorgenommen, in welcher der Stromverbrauch analysiert wird. Abschließend folgt eine Zusammenfassung und ein Ausblick auf mögliche Weiterentwicklungen.

Kapitel 2

Anwendungen des Signalisierungssystems

Das Signalisierungssystem ermöglicht das Aufwecken von mobilen Sensorknoten. Dieses Wecken ist dann besonders sinnvoll, wenn die Sensorknoten meistens schlafen und es nur wenige Anlässe gibt, zu denen die Sensorknoten gebraucht werden. Es sind viele Anwendungen denkbar, die diese Eigenarten ausnutzen. Es erfolgt eine exemplarische Betrachtung der Routing-Anwendung, an der sich die Erreichbarkeit von Knoten besonders gut studieren lässt. In diesem Kapitel wird zunächst die Routing-Anwendung detailiert analysiert, um anschließend weitere Alternativen zu skizzieren.

2.1 Anwendung zum Routing

In diesem Szenario gehen wir davon aus, dass ein Sensorknoten selten ein Ereignis wahrnimmt. Hat er jedoch ein Ereignis wahrgenommen und weiterverarbeitet, so soll er dies über andere Sensorknoten weiterleiten. Die anderen Sensorknoten, die nur Ereignisse weiterleiten, können zu Zeiten ohne Ereignis schlafengelegt werden. Durch das Signalisierungssystem können sie, sobald ein Ereignis weitergeleitet werden muss, vorher geweckt werden. Durch ein gezieltes Aufwecken bestimmter Sensorknoten lässt sich erreichen, dass nur Sensorknoten auf dem Routing-Pfad, der als bekannt angenommen wird, geweckt werden.

Es ist wünschenswert, wenn zu jeder möglichen IEEE-802.15.4-Übertragung eine Signalisierung möglich ist. Um dies zu erreichen, verwenden wir im Signalisierungssystem eine Antenne mit, verglichen mit der Antenne der Datenkommunikation, gleicher Empfangsempfindlichkeit. Im Sender hingegen wird eine stärkerer Sender in der Signalisierung eingesetzt als in der Datenkommunikation.

Ein Skizze für dieses Szenario ist in Abbildung 2.1 gegeben. In der Abbildung sind mehrere Knoten, denen zur Unterscheidung eine Zahl als Name gegeben ist. Die ungerichteten Kanten stellen dabei mögliche IEEE-802.15.4-Kommunikationswege dar, während die gerichteten Kanten mögliche Signalisierungswege darstellen. Wenn der Knoten 1 eine Nachricht an Knoten 4 senden will, so ist durch die höhere Sendeleistung des Signalisierungssystems sichergestellt, dass die benötigten Sensorknoten auf

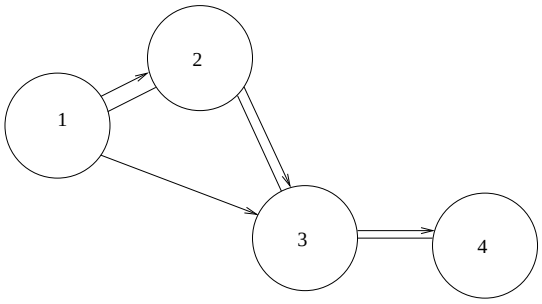


Abbildung 2.1: Beispiel für einen Routing-Pfad.

dem Pfad das Aufwecksignal empfangen. Unter Umständen könnte es auch möglich sein, dass ein Sensorknoten gleich mehrere weitere Sensorknoten weckt. In der Skizze ist diese Möglichkeit vorhanden, da eine gerichtete Kante zwischen Knoten 1 und 3 existiert.

2.2 Alternative Anwendungsszenarien

Alternative Anwendungen sind die Nutzung in *redundanten Systemen*, während *Wartungsarbeiten*, bei *Brandfällen* in Gebäuden, in *dynamischen Schließsystemen* und zur *Uhrensynchronisation*. Ihre Beschreibung erfolgt jeweils mit der Ausgangslage und mit den Verbesserungen, die das Signalisierungssystem hier bringt.

Redundante Systeme

Ausgangslage Zu einem Sensorknoten A ist ein gleicher Sensorknoten B redundant in der Nähe. A ist in der Lage, seine Batterieladung zu überwachen. B ist nur ein Ersatzsystem für A.

Verbesserung durch das Signalisierungssystem Das System erlaubt, dass A zunächst arbeitet und B in einem Schlafmodus ist. Sollte der Sensorknoten A feststellen, dass seine Batterie fast leer ist, so ist er noch in der Lage B aufzuwecken. B übernimmt daraufhin unterbrechungsfrei die Aufgaben von A.

Wartungsarbeiten

Ausgangslage An mehreren Orten sind Sensorknoten befestigt, die nur für Wartungsarbeiten gebraucht werden. Der Wartungsmitarbeiter kontrolliert mit einem Abfragegerät die Sensorwerte. Das Abfragegerät kann dabei beispielsweise ein Laptop oder ein Mobiltelefon sein.

Verbesserung durch das Signalisierungssystem Die Sensorknoten können für die meiste Zeit schlafengelegt werden. Erst wenn eine Wartungsanfrage kommt, können genau die Sensorknoten geweckt werden, die für die Anfrage die benötigten Daten erheben. Sensorknoten, die nichts mit der Anfrage zu tun haben, können weiter im Schlafmodus verweilen.

Rauminformation im Brandfall

Ausgangslage Eine durchgängige Videoüberwachung in einem Gebäude ist nicht immer gewünscht (vgl. [21]) und Informationen wie Luftfeuchtigkeit oder Temperatur sind häufig nicht nötig. Bei Brandfällen hingegen kann es wünschenswert sein, genau diese Informationen zu haben und dazu die notwendigen Sensorknoten zu aktivieren. Die Sensorknoten können einer Leitstelle im Brandfall wichtige Informationen liefern, unter anderem, in welchen Räumen nur Rauch und kein Feuer ist oder wo sich noch Menschen befinden.

Verbesserung durch das Signalisierungssystem Durch das System wird so ein Batteriebetrieb ermöglicht, wenn man davon ausgeht, dass die Knoten im Ruhezustand fast keine Energie benötigen. Gerade da im Brandfall die Stromversorgung eines Hauses zusammenbrechen könnte, ist es nicht sinnvoll sich auf sie zu verlassen.

Dynamisches Schließsystem

Ausgangslage Die Schlösser in einem Haus sollen, nicht wie gewöhnlich, aus mechanischen Schließanlagen bestehen, sondern aus elektronischen. Dazu gibt es viele Transponder, die jeweils per Funk ihre ID mitteilen können. Jedes Schloss hat eine Liste mit IDs. Betätigt ein Mensch seinen Transponder an einem Schloss, so kann das Schloss prüfen, ob die übermittelte ID in der Liste ist, und gegebenfalls die Türe öffnen. Während sich die Übergabe der ID mit einem energiesparsamen, aber sehr langsamen Protokoll realisieren lässt, ist die Programmierung des Schlosses über ein schnelleres Protokoll umgesetzt, da hierfür größere Datenmengen anfallen.

Verbesserung durch das Signalisierungssystem Das Signalisierungssystem erlaubt es, die Schlossprogrammiereinheit im gewöhnlichen Betrieb schlafen zu legen. Nur vor einer Programmierung wird ein Aufweckvorgang gestartet.

Uhrensynchronisation

Ausgangslage In einem bestimmten Sensorknotennetzwerk ist es wichtig, dass zu bestimmten Anlässen einige Sensorknoten ihre Uhren synchronisieren. Dabei wird die Annahme getroffen, dass die internen Uhren so genau sind, dass eine Sychonisation nur selten nötig ist.

Verbesserung durch das Signalisierungssystem Die schlafenden Sensorknoten können vor einer Uhrensynchronisation gezielt geweckt werden, um anschließend an der Synchronisation teilzunehmen.

Kapitel 3

Verwandte Arbeiten

In diesem Kapitel werden Signalisierungssysteme vorgestellt, bei denen ein Kommunikationsteil einen oder mehrere andere (Sensor-)Knoten aufweckt. Der Schwerpunkt liegt auf der Betrachtung energieeffizienter Lösungen. Da es sich in unserem Anwendungsfall nur um Single-Hop-Netzwerke handelt, wird hier keine Betrachtung von Multi-Hop-Netzwerken stattfinden. Häufig wird das Aufwachen im Zusammenhang eines gewissen Zeitplanes gesehen, bei dem dann natürlich auch die Uhrensynchronisation betrachtet wird. In dieser Bachelorarbeit wird davon ausgegangen, dass die Aufwachsignale zufällig eintreffen und somit das System triggern, daher findet keine Betrachtung von Scheduling-Verfahren und Uhrensynchronisation statt.

Ein guter Überblick über die verschiedenen Arten, wie ein Aufweckvorgang von mehreren Knoten bei gemeinsamen Mediumszugriff geschehen kann ist in [25] gegeben. Es sind Anwendungen von Time Division Multiple Access (TDMA), Code Division Multiple Access (CDMA) und Space Division Multiple Access (SDMA). Die TDMA-Methode besteht darin, den Sensorknoten Zeitslots zur Kommunikation zur Verfügung zu stellen. Außerhalb dieser Zeitslots kann der Sensorknoten in einen Schlafmodus gehen. Dadurch, dass ein Sensorknoten regelmäßig aufwachen muss, ist diese Lösung nur gut, wenn auch regelmäßig Daten vorliegen. Da in dieser Bachelorarbeit nur von seltenen und nicht planbaren Ereignissen ausgegangen wird, zu denen gesendet oder empfangen werden muss, ist TDMA als Lösung nicht verwendet worden. Die CDMA-Methode bedeutet auf das Szenario des Aufweckens verschiedener Sensorknoten angewandt, dass alle Knoten aufwachen, wenn ein Aufwecksignal kommt, und diese das Signal auswerten. Dadurch ist es möglich ohne viele Frequenzen eine große Anzahl an Sensorknoten gleichzeitig anzusprechen. Der Vorteil gegenüber TDMA-Verfahren ist, dass die Sensorknoten dadurch nicht regelmäßig aufwachen, sondern nur wenn ein Aufweckwunsch vorliegt. Diese Methode wurde auch in dieser Bachelorarbeit angewandt. Bei der SDMA-Methode wird eine neue Idee vorgestellt. Hierbei werden nicht nur die Sensornetze räumlich soweit getrennt, dass sie nicht interferieren, sondern das Prinzip einer rotierenden Antenne angewendet, die per Richtfunk Sensorknoten eines Winkelbereiches anfunkt. Eine räumliche Trennung von verschiedenen Imote2-Sensornetzen, von denen in der Bachelor-Arbeit ausgegangen wird, ist möglich. Auf die Verwendung einer rotierenden Antenne wurde verzichtet, da ihre Bewegung dem Ziel der Energieeinsparung widerspricht.

Ferner diskutiert Wendt [25] die Nutzung verschiedener Frequenzbänder, weil auf diesen auch energiesparsamerere Receiver zur Verfügung stehen. So verbraucht der vorgestellte Empfänger, ein Atmel ATA5282, nur ca. 3.5 μA bei 3.3 V. Dies ist $\frac{1}{5000}$ eines TI CC2420 ZigBee-Empfängers. Daher wurde in dieser Bachelorarbeit ein weiterer Empfänger gewählt, der diesen Effekt nutzen soll. Der ATA5282-Empfänger besitzt bereits Funktionen, wie Header-Erkennung und darauffolgend Aufwecken eines Sensorknotens. Der Unterschied zu der hier gezeigten Lösung ist, dass bei uns durch das Zwischenschalten eines Mikrocontrollers der Sensorknoten nur dann geweckt wird, wenn er auch geweckt werden soll. Der von uns verwendete Empfänger, ein RX-4M30RR01SF [5], ist zudem deutlich einfacher aufgebaut, so dass bei entsprechend großen Produktionsmengen ein geringer Preis erwartet werden kann.

Abseits von Sensorknoten weisen auch andere mobile Geräte, wie PDAs und Laptops, ähnliche Eigenschaften auf. So sind sie häufig im Batteriebetrieb und verfügen über Funkschnittstellen, wie WLAN oder Bluetooth. In den beiden folgenden Signalisierungssystemen werden Aufwecklösungen für WLAN-Knoten gezeigt.

Um einen Personal Digital Assistant (PDA), mit einem vergleichbaren großen Energieverbrauch, möglichst lange schlafen zu legen, entwickelte Shih [23] eine eigene Hardwarekomponente. Diese implementiert ein Signalisierungssystem, dass nicht das eingebaute WLAN-Modul benutzt, sondern einen energiesparsameren Transceiver. Dabei wird ein RFM ASH 915 MHz Receiver mit einem Mikrocontroller verbunden, der selbstständig den Kanal überwacht und bei Bedarf den PDA weckt. Der PDA selber kann zur Verwendung größerer Datenpakete auf ein WLAN-Modul (nach IEEE 802.11b) zurückgreifen. Die Idee dieses Systems ist dem in dieser Bachelorarbeit präsentiertem System ähnlich. Das WLAN-System entspricht in unserem Fall dem IEEE-802.15.4-Transceiver. In beiden Fällen soll dabei über energiesparsamere Band-Externe-Kommunikation das Hauptsystem bei Bedarf geweckt werden. Ein großer Unterschied zu der von uns entwickelten Lösung ist die Implementierung des Aufwachens des Mikrocontrollers. Shih implementierte einen Zeitplan, nach dem das Signalisierungssystem regelmäßig nach Aufwecksignalen frägt. Nach dem Senden eines Anfragesignals folgt eine Zeitspanne, in welcher Aufwecksignale empfangen werden können. Sind keine empfangen worden, so folgt eine Schlafphase. In unserer Lösung, ist das Signalisierungssystem schlafend, bis es auf dem Signalisierungsband ein Signal empfangen hat. Ferner sendet es nur Daten, wenn es andere Knoten aufwecken will.

Eine andere Variante einer eigenen Hardwarekomponente findet sich bei Mishra [20]. Dabei wird eine WLAN-Karte (mit IEEE 802.11b/g) geweckt. Das Wecken geschieht dadurch, dass ein energiesparsamerer IEEE-802.15.4-Receiver den Funkkanal abhört. Da WLAN und IEEE-802.15.4-Receiver beide das 2.4 Ghz-Band nutzen, ist es dem IEEE-802.15.4-Receiver möglich WLAN-Signale zu empfangen. Nimmt der IEEE-802.15.4-Receiver ein WLAN-Signal wahr, so ist er in der Lage die WLAN-Einheit aufzuwecken. Diese Variante nutzt, wie auch wir, einen energiesparsameren Receiver. Im Gegensatz zu unserer Lösung, bei der Interferenz durch eine Band-Externe-Kommunikation vermieden wird, ist hier Interferenz genutzt worden. Der Vorteil der Band-Externen-Kommunikation besteht darin, dass nicht nur das Anliegen eines Signals erkannt wird, sondern es auch noch ausgewertet werden kann. Somit ist ein gezieltes Aufwecken bei uns möglich. Ein weiterer Unterschied zu unserer Lösung

besteht in der Anwendung. Während bei Mishra [20] die Energieeinsparung bei der Funkkommmunikation von PCs, Laptops oder PDAs im Vordergrund steht, ist unser Schwerpunkt die Betrachtung kleiner, langlebiger Sensornetze. Diese unterscheiden sich durch geringere Datenmengen und höhere Anforderungen an die Laufzeit mit einer Batterie. So ist in [20] der IEEE-802.15.4-Transceiver schon ausreichend um Energie einzusparen. Was bei ihm aber das WLAN-System darstellt, ist in dieser Bachelorarbeit bereits der IEEE-802.15.4-Transceiver. Daher setzen wir in dieser Bachelorarbeit ein Receiver ein, der noch mehr an Energie einsparen kann.

In den oben dargestellten Arbeiten zeigte sich jeweils, dass energiesparsame Empfänger entscheidend sind. Die vorgestellten Empfänger waren dabei jeweils aktive Empfänger, dass heißt, sie verbrauchen auch in der Ruhe Strom. Eine andere Klasse an Empfängern im Bereich der Sensorknoten stellen passive Empfänger dar. Passive Empfänger nutzen dabei die Energie des empfangenen Signals um sich selbst mit Strom zu versorgen. Im Idealfall verbrauchen passive Empfänger keine zusätzlich Energie, so dass sie auch keine Batterie benötigen. In diesem Bereich hat sich die RFID-Technik etabliert. Die Problematik dabei ist, dass diese RFID-Empfangseinheiten ein starkes Sendesignal benötigen [14]. Da wir in dieser Arbeit davon ausgehen, dass die sendenden Einheiten selbst über geringe Batteriekapazitäten verfügen, ist ein starkes Sendesignal nicht gewünscht.

Um auch ein passives Empfangen mit geringen Sendeleistungen zu erreichen, ist es möglich, einen eigenen Empfänger zu bauen. Diesen Ansatz verfolgt Gu [14]. Dabei wird ein eigener Empfänger gebaut, der die Sendeenergie eines Signals in Kondensatoren ansammelt und, sobald genug Ladung gesammelt ist, eine logische-1 als Ausgangsspannung generiert. Der Erfolg des Verfahrens wird von Gu dadurch gezeigt, dass über 66% weniger Energie benötigt wird, als bei einem vergleichbaren Scheduling-Verfahren. Mit denen von den Autoren genannten Annahmen, lässt sich das System 178 Tage lang mit einer 1600 mAh-Batterie betreiben. Die Erfolge des passiven Systems sind zwar beachtlich, es gibt aber einige Gründe, dieses System nicht zu verwenden. Der erste Grund ist die hohe Latenz, sie beträgt nach [14] ca. 3 ms für 10 m, 20 ms für 20 m und 45 ms für 30 m Sendestrecke. Der zweite Grund ist, dass dieses System für einen Empfänger eine exklusive Frequenz benötigt. Die Ansteuerung mehrerer verschiedener Knoten ist nach Gu [14] nur mit einem Frequenzmultiplexverfahren (FDMA) möglich. Da wir in unserer Bachelorarbeit auf die Verwendung öffentlich zugänglicher Frequenzen angewiesen sind, konnte dieser Empfänger nicht verwendet werden.

Kapitel 4

Verwendete Hardware und Kommunikationstechniken

Für die Umsetzung der vorliegenden Arbeit wurde die im Folgenden beschriebene Hardware ausgewählt. Sie besteht aus den eingangs erwähnten Sensorknoten, die Imote2-Plattformen sind, den Sende- und Empfangseinheiten mit denen die Signalisierung erfolgt und Mikrocontrollern, welche die Verarbeitung der Funkdaten vornehmen. Zusätzlich wird auch das I^2C-Protokoll betrachtet, da über dieses der Mikrocontroller und der Imote2 kommunizieren. Alle Einheiten werden im Folgenden im Detail dargestellt. Besondere Beachtung findet dabei natürlich der Energiebedarf. Dieser wurde den jeweiligen Datenblättern entnommen, sofern nicht anders erwähnt. Sowohl Imote2 als auch das Signalisierungssystem werden mit der selben Batterie und somit der selben Spannungsquelle betrieben. Die Imote2-Plattform wird mit einer 4.5 V-Batterie versorgt. Um das Signalisierungssystem mit seinen benötigten 3 V zu versorgen, lässt man die entsprechende Spannungsdifferenz über einen Spannungswandler abfallen. Daher werden im Folgenden nur die Ströme, nicht aber die Spannungen betrachtet.

Abbildung 4.1: Imote2 mit Vorder- und Rückseite. Bildquelle: [12].

4.1 Die Imote2-Plattform

Die Imote2-Plattformen [10], siehe Abbildung 4.1, sind Sensorknoten, die für Anwendungen mit hohem Rechenbedarf konstruiert sind. Entwickelt werden diese Plattformen von Crossbow [11]. Die wichtigsten Einheiten dieser Plattform sind:

Intel PXA271 Prozessor Die Imote2-Plattform hat einen Intel PXA271 Prozessor mit dem sich auch anspruchsvollere Aufgaben, wie Bildverarbeitung lösen lassen. Für ein eingebettetes System besitzt der Prozessor mit 416 MHz maximaler Taktfrequenz, 32 MB Flash Speicher und 32 MB SDRAM über sehr üppige Ressourcen. Die Anwendungsentwicklung wird im *Intel PXA27x Processor Family Developers Manual* [18] und im *Update* [19] beschrieben. Hilfen zum Einschätzen des Energieverbrauchs sind vom Hersteller gegeben (vgl. [17], [16]), da diese aber nur eine grobe Orientierung bieten, wurde auf die Messungen von Gotzhein et al. [13] zurückgegriffen.

TI CC2420 IEEE-802.15.4-Funkmodul Zur Kommunikation per Funk steht ein IEEE-802.15.4-Funkmodul durch einen Texas Instrument CC2420 Transceiver bereit (Datenblatt: [24]) . Dieses bietet Übertragungsraten von 250 kbps. Die Antenne ist schon in den Imote2 integiert. Die Sendeleistung ist mit 0 dBm und die Empfangsempfindlichkeit mit −94 dBm gegeben. Damit können zwei Knoten bei Sichtverbindung bis zu 30 m entfernt liegen [10]. Dieses Funkmodul benötigt zum Senden ca. 12.95 mA und zum Empfangen 18.8 mA. Ist das Funkmodul abgeschaltet ergibt sich ein Verbrauch von 0.02 mA [13]. Das IEEE-802.15.4-Protokoll ist an sich schon auf den Sensorbereich angepasst, sofern man ZigBee darauf anwendet und ein Vergleich mit W-Lan oder Bluetooth stattfindet [8]. Im Sensorbereich genügen oft schon wenige Bit an Informationen aus. Für diesen Fall stellt ein PHY Layer Rahmen mit mindestens 6 Byte schon mal einen relativen großen Overhead dar [24].

I²C-Schnittstelle Über eine integrierte I²C-Schnittstelle können leicht Erweiterungen, wie in unserem Fall der Mikrocontroller, angeschlossen werden.

Da unsere Anwendungsszenarien lange inaktive Zeiten vorsehen, sind die Schlafmodi, ihre Aufweckmöglichkeit und der Energiebedarf im jeweiligen Modus besonders wichtig. Nach ihrem Energiebedarf geordnet sind die möglichen Modi in Tabelle 4.1 gezeigt.

4.2 Die Sende- und Empfangseinheiten

Nach einer Vorstellung der Sendeeinheit (TX-4MSIL) und der Empfangseinheit (RX-4M30RR01SF) wird eine Übersicht über das verwendete Frequenzband gegeben. Einige Eigenheiten der Sende- und Empfangseinheit, die Experimente ergeben haben, sind abschließend dokumentiert.

Modus	I[mA]	Aufweckmöglichkeiten	Kommentar
Deep Sleep Mode	1	Nur externe Events oder RTC	alle Daten gehen verloren
Sleep Mode	6	Nur externe Events oder RTC	Prozessorstatus geht verloren; Es sind noch einige Einstellungen, u.a. zum Halten des SRAMs, gegeben
Standy Mode	8	Nur externe Events oder RTC	
Deep-Idle Mode; 13 MHz;	27	alle möglichen Interrupts	nur CPU wird angehalten
Idle Mode; 104 MHz	39	alle möglichen Interrupts	nur CPU wird angehalten
Active Mode; 104 MHz	48	der Prozessor ist schon aktiv	-
Idle Mode; 416 MHz	107	alle möglichen Interrupts	nur CPU wird angehalten
Active Mode; 416 MHz	132	der Prozessor ist schon aktiv	-

Tabelle 4.1: Energieverbrauch des Prozessors des Imote2 in verschiedenen Modi. Anmerkung: RTC (=Real Time Clock) ist ein vom Benutzer einstellbarer Timer, mit einer Genauigkeit von einer Sekunde. Höhere Taktfrequenzen wurden nicht betrachtet, da diese für die Anwendung nicht nötig sind. Sofern nicht anders angegeben, bleibt der Prozessorstatus in den Modi erhalten.

4.2.1 Der TX-4MSIL und der RX-4M30RR01SF

Der Transmitter ist ein Aurel TX-4MSIL [6] und der Receiver ist ein Aurel RX-4M30RR01SF [5]. Sie sind eigenständige Bauteile, siehe Abbildung 4.2; dies ermöglicht einen Sensorknoten auch nur als Empfänger oder nur als Sender zu betreiben. Sie sind für einen geringen Energiebedarf, bei gleichzeitig geringen Übertragungsraten, ausgelegt. Sie nutzen, anders als der CC2420, das 433.92 MHz-Band um zu kommunizieren. Sowohl Sender als auch Empfänger benötigen zusätzlich noch Antennen, um ihre Daten zu übertragen. Für den Sender wird eine Marconi-Antenne verwendet. Beim Empfänger zeigte sich deutlich weniger Rauschen auf dem Kanal wenn man statt einer Marconi-Antenne eine empfohlene Antenne in Spulenform verwendet [7] (siehe Abbildungen 4.3, 4.4).

Der Transmitter sendet mit bis zu $4\,kbps$ mit einer An-Aus-Modulation (Englisch: On-Off-Keying OOK). Der Receiver empfängt auf der selben Frequenz Signale mit Maximalgeschwindigkeit von $3\,kbps$. Da die Sendestärke $-1\,dBm$ bis $2\,dBm$ beträgt und die Empfangsempfindlichkeit $-94\,dBm$ beträgt, ist die Reichweite im selben Bereich wie der CC2420. Somit gehen wir davon aus, dass ein Aufwecken genau dann möglich ist, wenn auch ein Kommunizieren über den CC2420 möglich ist.

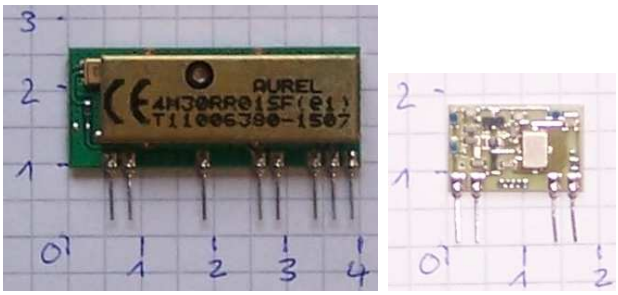


Abbildung 4.2: Links: Aurel RX-4M30RR01SF. Rechts: Aurel TX-4MSIL. Maßangaben jeweils in cm.

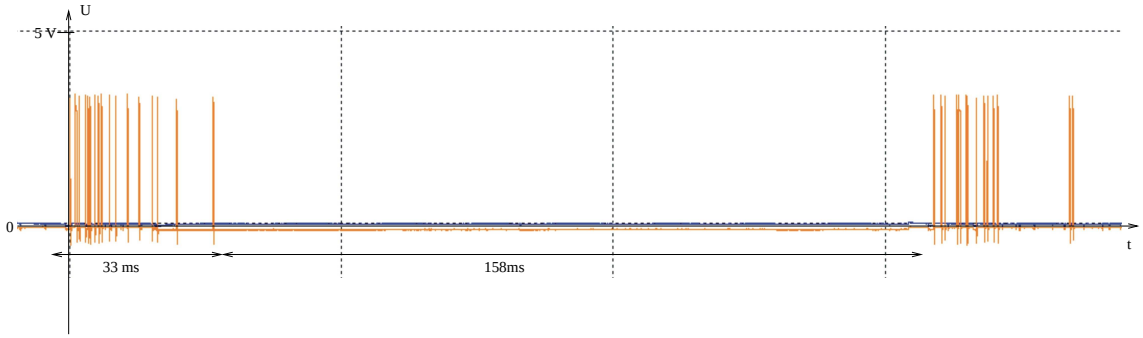


Abbildung 4.3: Messung eines unbelegten Funkkanals mit Marconi-Antenne.

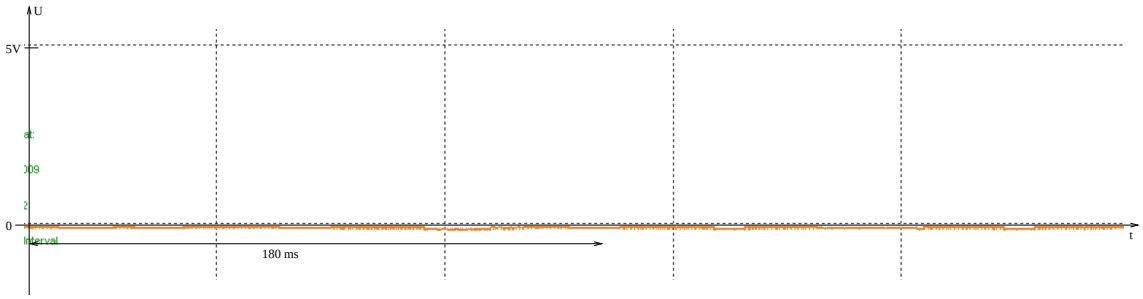


Abbildung 4.4: Messung eines unbelegten Funkkanals mit spulenförmiger Antenne.

	Energieverbrauch		
	[V]	[mA]	[mW]
Transmitter TX-4MSIL	3	6	18
Receiver RX-4M30RR01SF	3	0.07	0.21

Tabelle 4.2: Energieverbrauch der Schaltungen.

Der Energieverbrauch des TX-4MSIL und des RX-4M30RR01SF ist in Tabelle 4.2 gegeben.

4.2.2 Anmerkungen zum verwendeten Frequenzband

Das 433.92 MHz-Frequenzband ist zur "Nutzung durch die Allgemeinheit für nicht-öffentliche Funkanwendungen geringer Reichweite zugeteilt"[9]. Damit ist auf der Funkstrecke mit Kollisionen von Rahmen durch andere Teilnehmer zu rechnen. Beschränkungen hinsichtlich eines bestimmten Duty-Cycles, einer Listen-Before-Talk-Zeit oder eines Protokolles gibt es nicht.

"Die Reihenschaltung mehrerer Funkstrecken zum Zweck der Erhöhung der Reichweite ist nicht erlaubt"[9]. Damit kann man die Anwendung, dass ein routender Sensorknoten andere Sensorknoten aufweckt, damit diese Daten auf einem anderen Funkband weiterleitet, als juristisch fragwürdig betrachten.

4.2.3 Eigenarten des TX-4MSIL und des RX-4M30RR01SF

Leider zeigen sich einige unerwünschte Eigenschaften der Sende-/Empfangsstrecke. So ist auf Abbildung 4.5 zu sehen, dass eine Verzögerung von ca. $36\,\mu s$ bei einem $1 \rightarrow 0$ Wechsel und ca. $192\,\mu s$ bei einem $0 \rightarrow 1$ Wechsel vorliegt. Bei Frequenzen von unter $20\,Hz$ zeigte sich, dass einmal angelegte Einsen nicht gehalten werden, sondern dass die Spannung abnimmt. Zu sehen ist dies in der Messung in Abbildung 4.6. So ist nur von A bis B eine Eins am Ausgang des Empfängers, obwohl bis C eine Eins am Eingang des Senders anliegt. Ein weiterer Effekt ist, dass bei einem $1 \rightarrow 0$ Wechsel nach ca. $80\,ms$ ein unerwünschtes Rauschen auf dem Kanal liegt (zwischen D und E).

Folglich verzögert sich nach einem $0 \rightarrow 1$ Wechsel die nächste Abtastung, während bei einem $1 \rightarrow 0$ Wechsel die nächste Abtastung früher stattfindet.

4.3 Der Atmel AVR ATtiny24

Der Atmel AVR ATtiny24 ist ein low-power 8-Bit Mikrocontroller ([4]; siehe Abbildung 4.7). Er ist mit einer relativ geringen Rechenleistung (bis zu 8 MIPS[1]) und geringem Speicher ($2\,kB$ Flash; $128\,B$ SRAM) ausgestattet. Er ist für eine gute Interaktion mit seiner Umwelt ausgelegt, so besitzt er Analog-Digital-Konverter, zwei

[1] Million Instruction Per Second

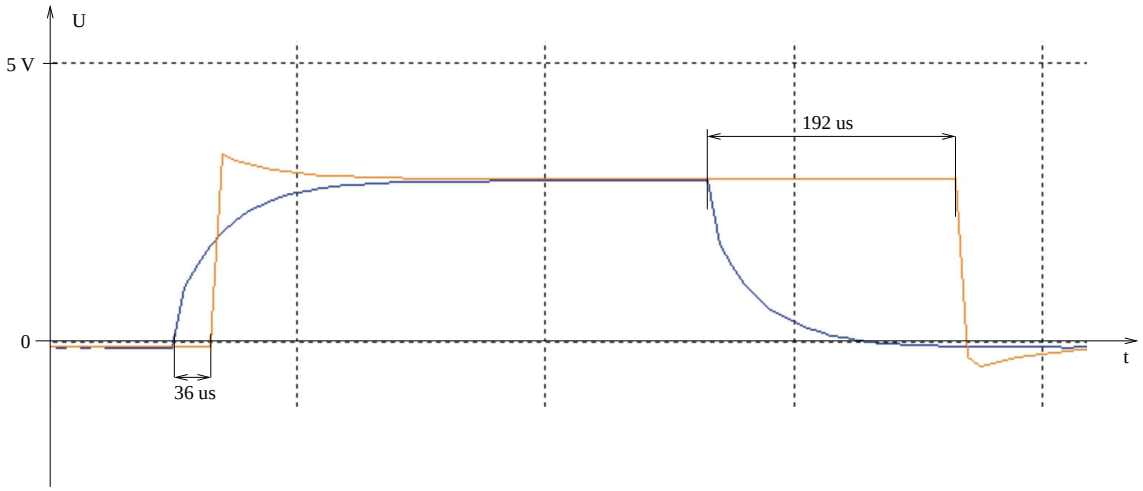


Abbildung 4.5: Verzögerungen bei Flankenwechsel. Blau: Messung an PIN_TX des sendenden Mikrocontrollers. Orange: Messung an PIN_RX des empfangenden Mikrocontrollers.

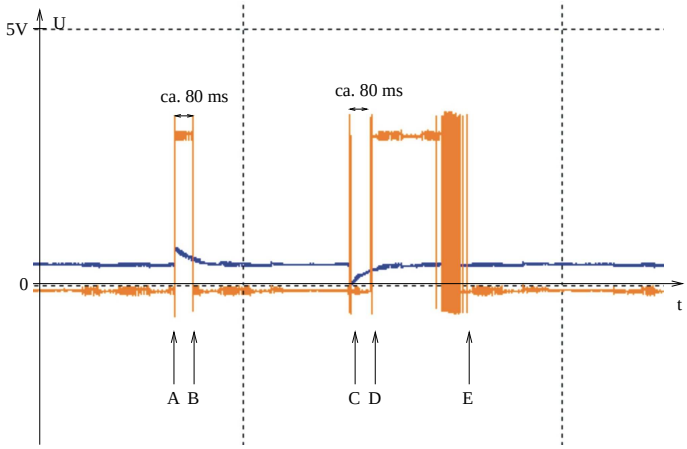


Abbildung 4.6: Messung an Pin *Test Point* und an Pin *Data Output* des Receivers [5]. Zwischen A und C war eine Eins zum Senden an Pin *Data In* des Transmitters [6] angelegt. Vor A und nach C war eine 0 an *Data In* angelegt.

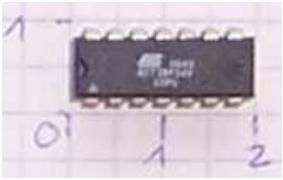


Abbildung 4.7: ATtiny24 mit 14-Pin Dual Inline Package. Maßangaben in cm.

Modus	Energieverbrauch			Aufweckmöglichkeit
	[V]	[mA]	[mW]	
Power down	3	0.0001	0.0003	I²C-Start; Pin Wechsel
Idle; 1MHz	3	0.07	1.8	Alle möglichen Interrupts
Idle; 8MHz	3	0.6	1.8	Alle möglichen Interrupts
Active; 1MHz	3	0.45	7.5	Mikrocontroller ist schon wach
Active; 8MHz	3	2.5	7.5	Mikrocontroller ist schon wach

Tabelle 4.3: Energieverbrauch und Aufweckmöglichkeiten des ATtiny 24 in verschiedenen Modi.

Timer und die Möglichkeit auf Flankenwechsel an Pins zu reagieren. Der ATtiny24 unterstützt hardwareseitig I²C. Andere I/O Pins können zu Debug-Zwecken genutzt werden.

Der ATtiny24 bietet zwei Energiesparmodi an: Den Idle-Mode und den Power-Down-Mode (vgl. Tabelle 4.3). Im Idle-Mode können die Timer und die I²C-Kommunikationseinheit arbeiten. Im Power-Down-Mode hingegen kann nur auf den Flankenwechsel eines Signals oder auf das Empfangen eines I²C-Startsignals hin der reguläre Betrieb wieder erreicht werden. Verwendet man zusätzlich während einer aktiven Phase oder im Idle-Modus noch den Timer0 und das I²C-Modul, so erhöht sich der Energiebedarf nach Tabelle 4.3 . Da sich die Prozente addieren, beträgt der Stromverbrauch, bei gleichzeitigem Betrieb von Idle-Mode (1MHz), Timer0 und I²C-Modul, $I = 0.07\,\text{mA} \cdot (1 + 0.104 + 0.061) = 0.07\,\text{mA} \cdot (1.165) = 0.082\,\text{mA}$.

4.4 Kurzvorstellung von I²C

I²C ist ein Kommunikationsverfahren, das auf einer Master-Slave-Kommunikation aufbaut. Die genaue Funktionsweise ist in [22] spezifiziert. Da I²C von Philipps eingeführt wurde, verwenden viele Hersteller, so auch der ATtiny24, andere Namen wie

Komponente	Erhöhung in % im aktiven Modus	Erhöhung in % im Idle-Mode
Timer0	2.3	10.4
I²C-Modul	1.4	6.1

Tabelle 4.4: Prozentuale Erhöhung des Energiebedarfs für zusätzliche Komponenten.

Two-Wire-Interface (TWI). Aus dem Datenblatt des ATtiny24 geht hervor, dass nur der Standard Mode mit 7-Bit-Adessierung unterstützt wird, sofern der ATtiny24 mit einer Mindesttaktfrequenz von 200 kHz betrieben wird [4]. Die Intel-Architektur des Imote2 beruft sich ausdrücklich auf den I²C-Standard von Philips Semiconductors ([18]). Nach der Klärung, auf welchen Standard sich die Vorstellung von I²C bezieht, wird nun das Protokoll vorgestellt.

Das Protokoll ist für das Signalisierungssystem geeignet, da es das Versenden kleiner Nachrichten ohne großen Overhead ermöglicht. Die Annahmen von I²C sind eine zu vernachlässigende Ausbreitungsverzögerung und ein unverfälschender Kommunikationskanal. Da I²C-Bus eine sehr geringe Länge von weniger als 30 cm hat, ist diese Annahme gerechtfertigt.

Bei I²C findet die Kommunikation über einen Bus statt, der zwei Leitungen hat. Die eine ist die Takt-Leitung (auch SCL benannt) und die andere ist die Daten-Leitung (auch SDA genannt). Beide Kanäle liegen an Pull-Up-Widerständen, so dass der Kanal über eine Wired-And-Struktur verfügt.

Eine logische-1 auf der Takt-Leitung bedeutet, dass gültige Daten anliegen. Entsprechend darf sich der Wert auf der Daten-Leitung in dieser Zeit nicht ändern. Ausnahmen von dieser Regel stellen die Start- und Stoppbedingung dar, die den Anfang beziehungsweise das Ende der I²C-Kommunikation signalisieren. Befindet sich auf der Takt-Leitung eine logische-0, so kann auf der Daten-Leitung der Wert für das nächste Bit eingestellt werden. Um die Kommunikation zu verzögern, kann ein Teilnehmer die Taktleitung auf logisch-0 halten. Dies wird als Zeit-Dehnung (Englisch: Clock-Stretching) bezeichnet. Die Taktleitung wird, sofern keine Zeit-Dehnung geschieht, vom Master gesteuert.

Da I²C ein Multi-Master-System ist, geschieht die Busarbitierung dadurch, dass ein Master versucht, während die Takt-Leitung auf logisch-0, ist sein Datum auf der Daten-Leitung aufzuprägen. Gelingt es ihm nicht, das gewünschte Datum auf den Kanal zu bringen bis die Takt-Leitung wieder auf logisch-1 ist, so hat der Master die Busarbitierung verloren. Dieser Verlust kann im Extremfall erst bemerkt werden, wenn das letzte Bit einer Bytefolge gesendet wird.

Die gesamte Kommunikation erfolgt Byteweise. Sind vom Master acht Bit gesendet, so lässt er die Daten-Leitung frei, betreibt die Takt-Leitung wie vorher weiter. Der Slave legt nach dem achten Bit eine 0 (für ACK) oder eine 1 (für NACK) auf den Kanal.

Leider ist die Spezifikation des ACK-Sendens unpräzise. Dazu wird das Beispiel einer gewöhnlichen Kommunikation betrachtet. Hat der Master ein Byte gesendet, wird die Takt-Leitung auf logisch-0 gesetzt, so ist es nun Aufgabe des Slave bis zum $0 \rightarrow 1$ Flankenwechsel der Taktleitung die Daten-Leitung auf logisch-0 zu ziehen. Während des Setzens kann der Slave Zeit-Dehnung betreiben. Hat der Slave sein ACK-Bit gesetzt, so kann der Slave die Takt-Leitung freigeben. Sobald der Master die Takt-Leitung freigegeben hat, so wird die Takt-Leitung auf logisch-1 gezogen. Bis der Master die Takt-Leitung wieder auf logisch-0 zieht, hält der Slave den ACK-Wert. Anschließend versucht der Master das erste Bit des nächsten Bytes zu setzen. Der Slave muss in der selben Zeit die Daten-Leitung wieder freigeben. Hält der Slave nun noch zulange seine 0 und der Master versucht eine 1 auf den Kanal zu bekommen, so

kann der Master fälschlicherweise die Annahme treffen, dass er die Busarbitierung gegen einen anderen Master verloren hat. Da in unserer Anwendung nur ein Master pro Bus vorhanden ist, kann die Erkennung einer verlorenen Busarbitierung beim Imote2 ausgeschaltet werden, um dieses Problem zu umgehen.

Kapitel 5

Schnittstellen des Signalisierungssystems

Wie die in Kapitel 4 beschriebenen Hardwarekomponenten miteinander kommunizieren, wird in diesem Kapitel erklärt. Zuerst wird ein Überblick über mögliche Kommunikationsabläufe gegeben, um anschließend die Schnittstellen präzise zu spezifizieren. Dabei werden die Rahmenformate der Kommunikation erläuterert.

5.1 Überblick über mögliche Kommunikationsabläufe

Als erstes werden die Anforderungen an das Signalisierungssystem gestellt. Die möglichen Nachrichten, die einzelne Komponenten miteinander austauschen können sind anschließend dokumentiert. Danach wird das Zusammenspiel aller Komponenten mit entsprechenden Sequenzdiagrammen illustriert. Abschließend werden noch weitere Anwendungen kurz diskutiert, die für die Zukunft denkbar sind.

5.1.1 Anforderungen

Der Imote2 kann (a) aufweckt werden von anderen Imote2 und (b) mehrere verschiedene Imote2 aufwecken. Um dies zu ermöglichen hat jeder Imote2 eine eigene ID, mit ID $\in \{0, 1, 2, \ldots, 15\}$. Für (b) wird gefordert, dass der Imote2 einen Vektor v ($v \in \{0, 1\}^{16}$) übergeben kann. Dabei bedeutet eine 1 an der Stelle i, dass der Imote mit ID $= i$ geweckt werden soll. Eine 0 bedeutet, dass der Imote wie bisher weiter operieren soll. Somit ist ein Aufwecken, nicht aber ein Schlafenlegen, möglich. Die Entscheidung wann Imote2 in einen Schlafmodus geht, entscheidet er entweder selbstständig oder es geschieht über die IEEE-802.15.4-Funkschnittstelle.

5.1.2 Nachrichten

Zunächst werden die Nachrichten, die der Imote2 und der Mikrocontroller austauschen spezifiziert. Daraufhin werden die Nachrichten, die über die 433MHz-Funkstrecke ausgetauscht werden können, spezifiziert.

5.1.2.1 Nachrichten zwischen Imote2 und Mikrocontroller

Tabelle 5.1 beschreibt die drei möglichen Nachrichten, die ein Imote2 senden oder empfangen kann.

Nachricht	RX	TX	Beschreibung
WakeUpImote2	Ja	Nein	Durch einen Flankenwechsel an einem Pin kann der Imote geweckt werden.
OwnId	Nein	Ja	Diese Nachricht gibt dem Mikrocontroller die ID des Imote2. Durch sie weiß der Mikrocontroller auf welche ID er zu reagieren hat, wenn er den Funkkanal abhört.
WakeUp	Nein	Ja	Hiermit wird der oben beschriebene Vektor übergeben.

Tabelle 5.1: Nachrichten des Imote2. Anmerkung: RX bedeutet, dass diese Nachricht vom Imote2 empfangen wird. TX bedeutet, dass der Imote2 diese Nachricht sendet.

5.1.2.2 Nachrichten auf der 433MHz Funkstrecke

Auf der 433MHz-Funkstrecke wird nur die Nachricht WakeUp gesendet. Sie enthält den Vektor v und geht an alle weiteren Signalisierungssysteme, die per Funk erreichbar sind.

5.1.3 Interaktionsabläufe

Zwei Exemplarische Kommunikationsabläufe sind in Abbildung 5.1 und 5.2 dargestellt.

In Abbildung 5.1 ist gezeigt, dass Imote2 von sich aus die OwnId-Nachricht verschickt. Es bedarf keiner Aufforderung des ATtiny24 dazu. Das ACK ist hierbei keine eigene Nachricht, sondern setzt sich aus den Empfangsbestätigungsbits zusammen, welche durch die verwendete I²C-Kommunikation kommen.

In Abbildung 5.2 ist ein Aufweckvorgang dargestellt. Zunächst sendet Alice die WakeUp-Nachricht an den Mikrocontroller A. Dieser bestätigt den Empfang mit den über das I²C vorgesehenen ACK Bits. Der Vektor v wird in A gepuffert, so dass die I²C Verbindung möglichst rasch beendet werden kann. Daraufhin sendet A einmal die WakeUp-Nachricht über die 433Mhz Funkstrecke. Ein mehrmaliges Senden der WakeUp-Nachricht infolge *einer* WakeUp-Nachricht ist nicht vorgesehen.

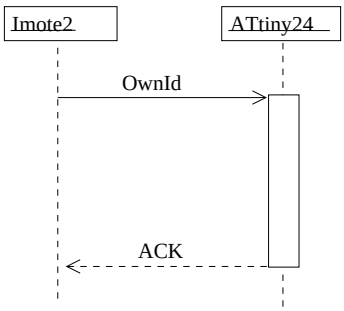


Abbildung 5.1: Sequenzdiagramm zur Mitteilung der Imote2-ID

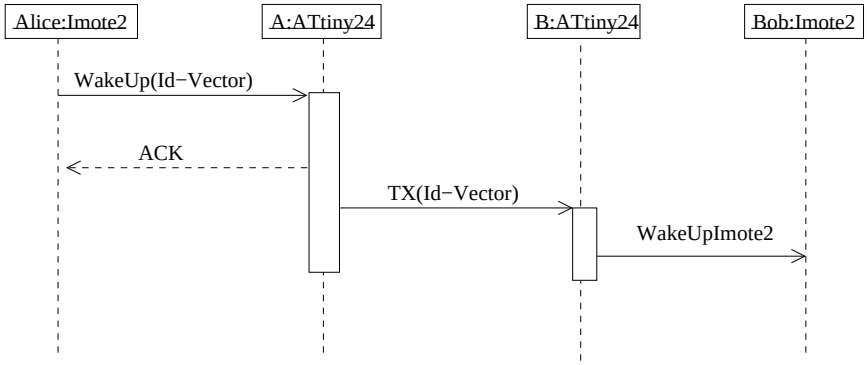


Abbildung 5.2: Sequenzdiagramm eines Aufweckvorganges

Wenn gewünscht wird, dass mehrmals eine WakeUp-Nachricht gesendet werden sollte, so muss der Imote2 mehrere WakeUp-Nachrichten nacheinander senden (siehe Abbildung 5.3). Dabei ist zu beachten, dass während einer Funkkommunikation ein Mikrocontroller keine I^2C-Kommunikation betreibt.

Es ist zu sehen, dass Nachrichten nicht quittiert werden. Da nachdem Senden der WakeUp-Nachricht eine IEEE-802.15.4-Kommunikation vorgesehen ist, kann über diese Kommunikation festgestellt werden, ob der Aufweckvorgang erfolgreich war oder nicht. Es ist möglich durch mehrfaches Senden von WakeUp eine ausreichend große Aufweckwahrscheinlichkeit herzustellen.

5.1.4 Weitere denkbare Anwendungen

Durch die Band-Externe-Kommunikation steht prinzipiell ein weiterer Kanal zur Verfügung, dieser könnte darüber hinaus genutzt werden, um auch andere Statusinformationen während einer Kommunikation auszutauschen. So könnte damit parallel zu einer ZigBee-Kommunikation angezeigt werden, dass der Empfänger gleich die Kommunikation abbricht, da seine Energie zu Ende ist.

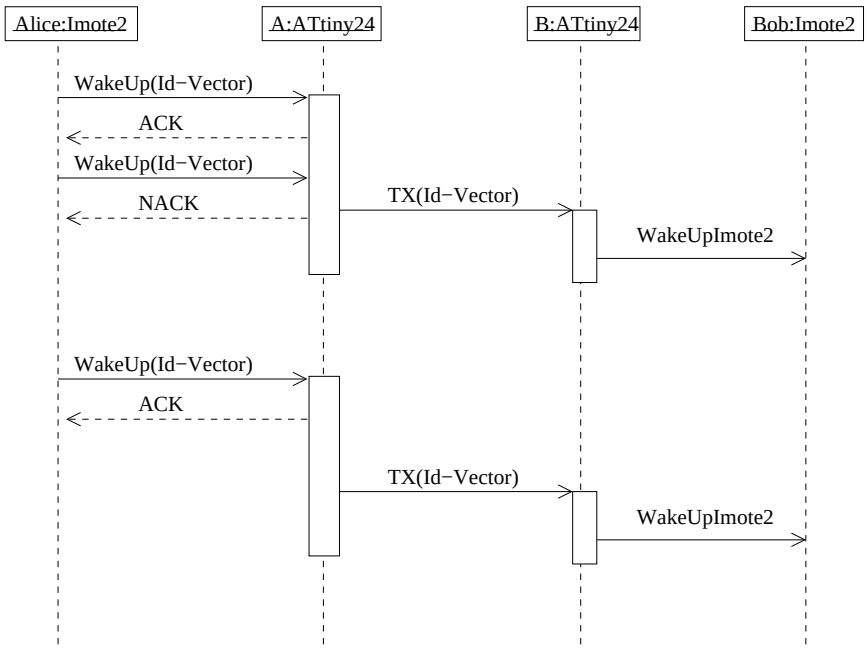


Abbildung 5.3: Sequenzdiagramm zum mehrmaligen Übertragen

5.2 Kommunikation des Imote2 mit dem ATtiny24

Die Kommunikation vom Imote2 zum ATtiny24 erfolgt mittels I²C. Nach einer kurzen Vorstellung des Protokolls, wird die entwickelte Anwendung gezeigt.

5.2.1 Anwendung des I²C-Protokolls

Es ist vorgesehen, dass der Imote2 als Master im I²C-Netz fungiert und der ATtiny24 als Slave. Ein Kommunikationsrahmen ist in Abbildung 5.4 gezeigt. Eine Nachricht besteht dabei immer aus vier Oktetts. Das erste Oktett ist, wie es der Standard [22] vorschreibt, die I²C-Adresse. Das zweite wird zum Kodieren der Nachricht verwendet. So ist eine 0 für die Nachricht OwnId und 1 für WakeUp vorgesehen.

Für eine OwnId-Nachricht bedeutet eine 1 im Feld IDi, dass der Imote2 die ID $= i$ hat. Die Anwendung kann somit für einen Imote2 nicht nur eine ID angeben, auf die der Imote2 geweckt werden soll, sondern mehrere. Gibt die Anwendung in keinem Feld eine 1 an, so wird der Imote2 niemals geweckt.

Entsprechend ist die Bedeutung der ID-Felder für Wake-Up-Nachrichten. Hier sind die ID-Felder die Implementierung des oben beschriebenen Vektors v.

Bei der Kommunikation wird davon ausgegangen, dass der Mikrocontroller nur in folgenden Fällen ein ACK setzt:

erstes Oktett: korrektes Erkennen seiner Slave-Adresse.

zweites Oktett: gültigen Application Code erkannt.

drittes und viertes Oktett: es wird immer ein ACK gesendet.

Sollte der Mikrocontroller kein ACK senden, so ist davon auszugehen, dass der Mikrocontroller mit einer vorhergehenden I²C-Übertragung beschäftigt ist.

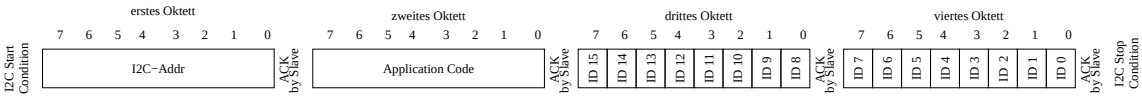


Abbildung 5.4: Die Anwendung des I²C-Protokolls

5.2.2 Vom ATtiny24 zum Imote2

Der Imote kann nicht per I²C aus dem tiefsten Schlafmodus geweckt werden, daher ist noch eine extra Signalisierungsleitung vom ATtiny24 zum Imote2 gelegt worden. Sobald sich der Wert auf dieser Leitung ändert, soll der Imote2 wach werden.

5.3 Kommunikationprotokoll der 433 MHz-Funkstrecke

Auf der 433 MHz-Funkstrecke soll nur die WakeUp-Nachricht weitergeleitet werden. Hierbei soll der Mikrocontroller beim Empfangen einer Folge von Flankenwechseln auf dem Funkkanal möglichst korrekt und schnell erkennen, wann eine Nachricht für ihn vorliegt und wann nicht. Es wird davon ausgegangen, dass die WakeUp-Nachricht nicht quittiert werden muss und insofern kann kein fehlerfreies Senden vom Aufweckenden festgestellt werden.

Es ist möglich, die Robustheit der Übertragung durch eine Vorwärtskorrektur (Forward Error Correction FEC) zu erhöhen. Geht man allerdings davon aus, dass Fehler nicht durch den Falschempfang eines einzelnen Bits, sondern durch Kollisionen stattfinden, so ist der Nutzen der Vorwärtskorrektur gering. Eine Vorwärtskorrektur bedeutet, dass sowohl eine Kodierung, als auch eine Dekodierung mit Fehlerkorrektur durchgeführt werden muss. Um das System aber möglichst einfach zu halten, wurde auf diese Möglichkeit verzichtet. Vorrangiges Ziel ist die Entwicklung eines lauffähigen Programmes gewesen, dass die Machbarkeit einer Aufwecklösung demonstriert.

Aus den oben bechriebenen Hardwareeigenschaften folgt, dass eine Mindestfrequenz von Flankenwechseln nötig ist. Daher wurde ein festes Bitstuffing eingefügt (siehe blau markierte Bits in Abbildung 5.5). Somit ist die Mindestfequenz für einen Bitwechsel mit $f = 10 \cdot 20\,\text{Hz} = 200\,\text{Hz}$ festgelegt.

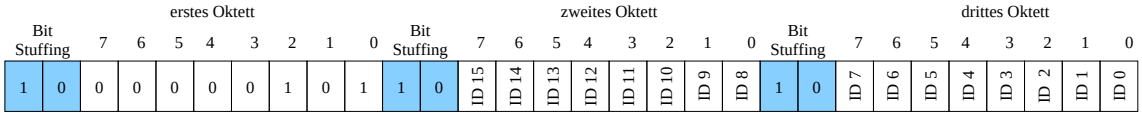


Abbildung 5.5: Die Rahmen der Funkübertragung

Ein Aufwecken des Imote2, obwohl kein Aufwecksignal vorliegt, sollte weitestgehend durch eine Präambel (siehe Abbildung 5.5) vermieden werden. Die Präambel besteht aus dem ersten Oktett und dem vorhergehendem Bitstuffing. Sie ist so aufgebaut, dass mit der führenden 1 zunächst ein Aufwachen ermöglicht wird. In den folgenden sechs Bitzeiten soll das Aufwachen des Mikrocontrollers erfolgen und an der Sequenz 101 soll erkannt werden, dass der Mikrocontroller angesprochen ist.

Kapitel 6

Energiesparsamer Entwurf

Mit der in Kapitel 4 besprochenen Hardware und den Anforderungen wurde ein modularer Entwurf erstellt. Dazu wurde auf der Imote Seite ein I^2C Treiber mit Programmen erstellt, um die Anwendung zu testen. Die Hauptarbeit lag allerdings im Programm des Mikrocontrollers, der durch die geschickte Verwendung von Interrupts möglichst wenig Energie verbrauchen soll.

6.1 Programme für die Imote2-Knoten

Als Programm für den Imote2 ist eine Anwendung des Treibers entwickelt worden. Die SDL-Spezifikation ist in Abbildung 6.1 zu sehen. Dabei wird zuerst die eigene ID an den Mikrocontroller übergeben und anschließend auf ein WakeUpNode-Signal eines anderen Prozesses gewartet, der die aufzuweckenden Knoten angibt. Das WakeUpNode-Signal stellt die Schnittstelle zu anderen Prozessen dar, die auf dem Imote2 laufen können.

6.2 Programm des ATtiny24

Im Folgenden wird der Aufbau des Programmes auf dem ATtiny24 gezeigt. Dabei werden zwei Abstraktionsebenen durchschritten. Zunächst wird das Programm mit Hilfe eines Zustandautomates modelliert. Das Modell wird im nächsten Schritt auf Hardware-Architektur abgebildet. Bei dieser Abbildung fließen auch Entscheidungen ein, wie die Benutzung bestimmter Interrupts und verschiedener Schlafmodi. Eine Anleitung zum Kompilieren, die Abbildung von logischen Pin-Bezeichnungen auf physische Pins findet sich im Anhang.

6.2.1 Modellierung

Die Modellierung des Programmes erfolgt mit einem Zustandsautomaten in 6.2.1.1. Dieser Zustandsautomat wurde in weitere Zustandsautomaten zerlegt. Dadurch erhält man eine hierarchische Struktur.

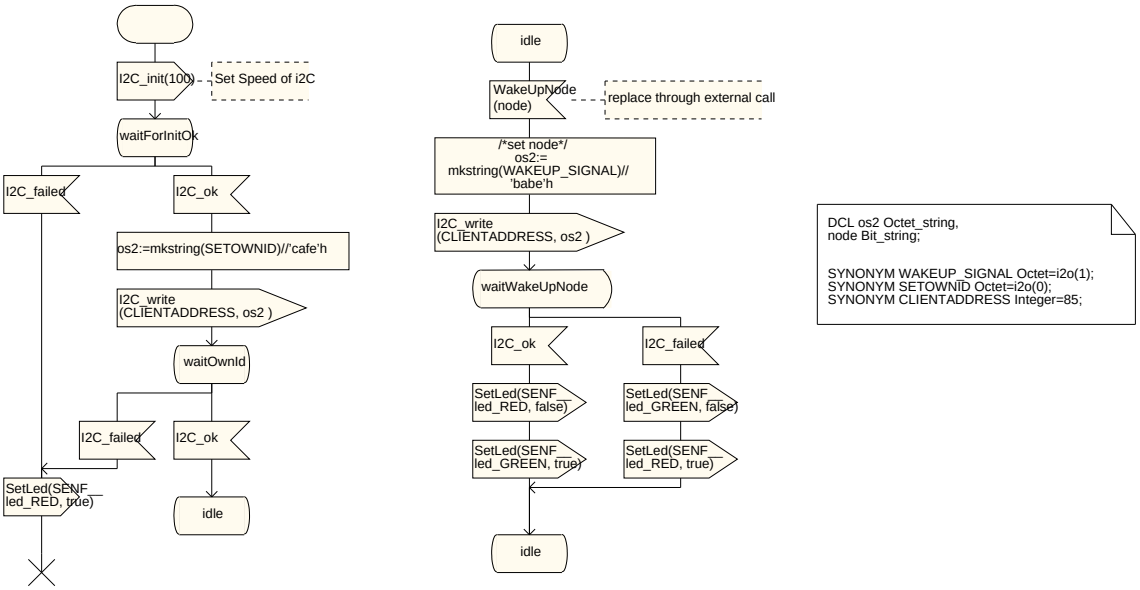


Abbildung 6.1: Sendender Imote2

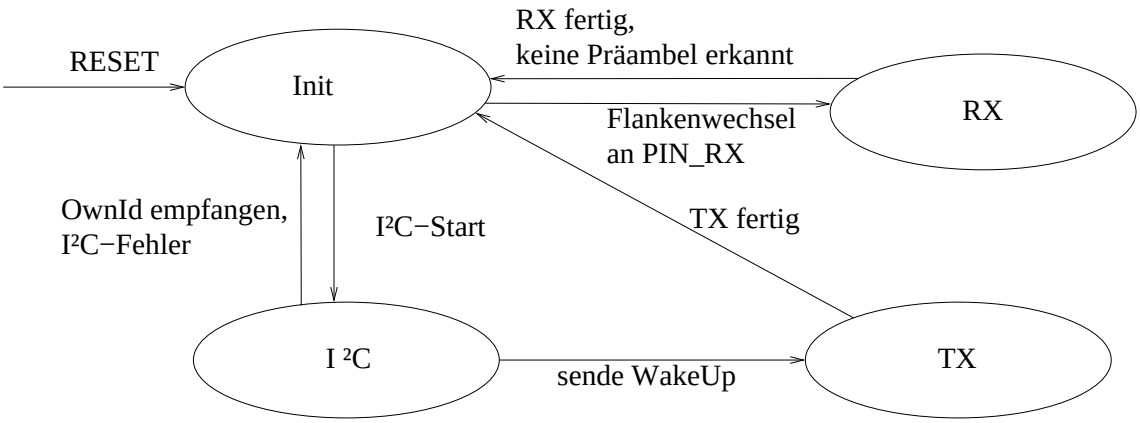


Abbildung 6.2: Zustandsautomat des ATtiny24-Programmes.

6.2.1.1 Zustandsautomat

Die Programm-Modellierung ist in Abbildung 6.2 und mit der Beschreibung 6.2.1.1 dokumentiert. An den vier Zuständen ist zu erkennen, dass jeweils nur eine Aufgabe durchgeführt wird. Eine Parallelisierung des Programm-Ablaufes ist nicht vorgesehen. Insbesondere das Testen des Programmes wird somit leichter möglich.

6.2.1.2 Algorithmen und verfeinerte Zustände

Durch hierarchische Zustandsautomaten wurde jeder Zustand des Zustandsautomaten (vgl. Abbildung 6.2.1.1) in weitere Unterzustände aufgeteilt. Diese Unterzustände sind auch im Programmkode so verwendet worden. Die Verfeinerung der einzelnen Zuständen ist in den Abbildungen 6.3, 6.4, 6.5 und 6.6 dargestellt. Eine Beschrei-

Zustand	Aufgabe	Zustandswechsel
Init	Init führt zunächst eine Initalisierung der Hardware aus. Dabei werden Pins als Ausgabe-Pins konfiguiert, die initiale Imote2-ID festgelegt und die I²C-Adresse zugewiesen. Anschließend wird auf einen Interrupt gewartet.	Wenn ein I²C-Startsignal oder ein Flankenwechsel am 433 MHz-Receiver festgestellt wird, so erfolgt entweder eine I²C-Kommunikation oder eine Überprüfung des 433 MHz-Funkkanals.
I2C	Dieser Zustand übernimmt die I²C-Kommunikation. Die I²C-Kommunikation besteht aus dem I²C-Startsignal und einer Folge von jeweils 9-Bit, die sich aus dem Empfang von 8 Bit und der anschließenden Versendung eines ACK bzw. NACK zusammensetzen. Da die I²C-Anwendung immer genau vier Byte verarbeitet, ist entsprechend genutzt worden.	Wurde eine OwnId-Nachricht empfangen, so ist die neue Imote2-ID gespeichert und es kann wieder der Zustand Init eingenommen werden. Bei WakeUp-Nachrichten wird der WakeUp-Vector v zwischengespeichert, die I²C-Kommunikation beendet und die Transmitter-Kommunikation im Zustand TX begonnen.
TX	Im Zustand TX werden die zu sendenden Bit an den TX-4MSIL übergeben. Dies geschieht dadurch, dass zu den richtigen Zeitpunkten, der richtige Flankenwechsel an einem bestimmten Pin (bezeichnet mit PIN_TX) vollzogen werden.	Ist die Nachricht gesendet wird der Zustand Init angenommen.
RX	In RX reagiert der ATtiny24 auf Pinwechsel an einem bestimmten Pin (bezeichnet mit PIN_RX). Ferner übernimmt dieser Zustand die Pufferung und Auswertung der empfangenen Bits. Zunächst wird dabei die mutmaßliche Präamble auf ihre Korrektheit überprüft. Eine nichtgültige Prämamble wird verworfen. Wurde eine gültige Präambel erkannt, wird der Kanal abgetastet bis die vollständige WakeUp-Nachricht empfangen ist. Anschließend wird, je nach WakeUp-Nachricht, der Imote2 geweckt.	Sollte erkannt werden, dass die empfangene WakeUp-Nachricht den eigenen Imote2 wecken sollte, so wird an der Bitleitung zum Imote2 ein Flankenwechsel durchgeführt. Anderenfalls geschieht kein Flankenwechsel. Nach Prüfung der WakeUp-Nachricht, wird wieder der Zustand Init betreten.

Tabelle 6.1: Zustände des ATtiny24-Programmes.

bung der verfeinerten Zustände und der verwendeten Algorithmen ist im Folgenden gegeben.

Verfeinerung von Init Wie in Abbildung 6.3 zu sehen, wurde der Zustand auf die Zustände STARTING und INACTIVE aufgeteilt. STARTING übernimmt die Hardwareinitalisierung, während in INACTIVE auf einen Interrupt gewartet wird.

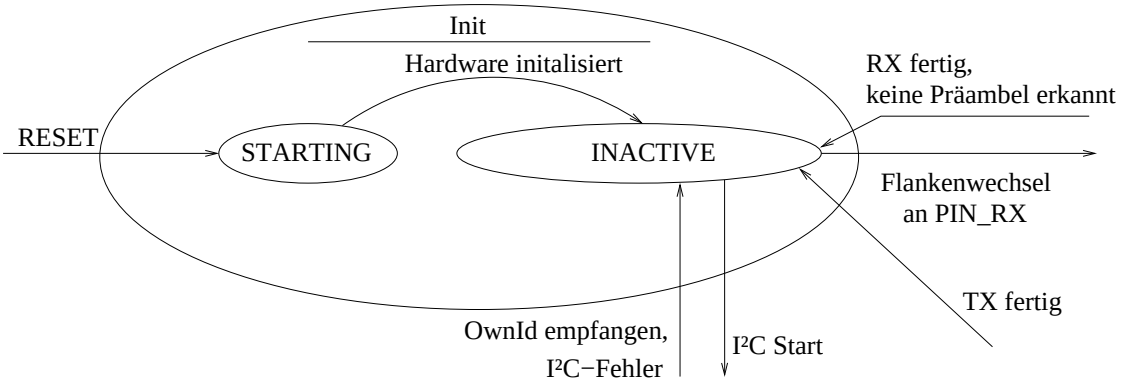


Abbildung 6.3: Verfeinerte Modellierung des Init-Zustands

Verfeinerung von I2C Der I2C-Zustand wurde in die Zustände I2C_SLAVE und I2C_SLAVE_SEND_ACK zerlegt (siehe Abbildung 6.4). Im ersten Zustand werden jeweils 8 empfangene Bits analysiert. Nach dem vierten Byte wird eine Marke gesetzt, mit der im I2C_SLAVE_SEND_ACK entschieden werden kann, ob der initiale Zustand angenommen wird oder ob eine WakeUp-Nachricht versendet werden soll. Im I2C_SLAVE_SEND_ACK-Zustand wird das in I2C_SLAVE berechnete ACK-Bit gesetzt. Am Ende der I²C-Nachricht, die exakt vier Byte lang ist, ist entweder die Imote2-Adresse aktualisiert worden oder ein Vektor v zum Aufwecken andere Sensorknoten zwischengespeichert worden.

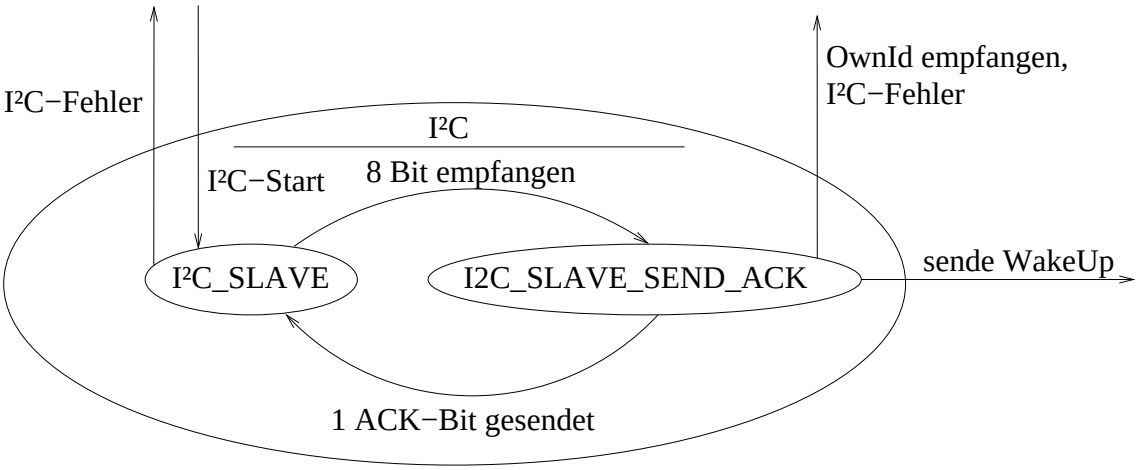


Abbildung 6.4: Verfeinerte Modellierung des I²C-Zustandes

Verfeinerung von TX Der TX-Zustand wird in die Zustände TX_SEND_PREAMBLE und TX_ACTIVE zerlegt (siehe Abbildung 6.5). In TX_SEND_PREAMBLE wird im Gegensatz zu TX_ACTIVE, ein Präambel-Byte und nicht der WakeUp-Vektor zum Senden gewählt. Nach der Wahl wird in beiden Zuständen in jeweils festen Zeitintervallen das nächste zum Senden vorgesehene Bit auf den 433 MHz-Funkkanal gelegt.

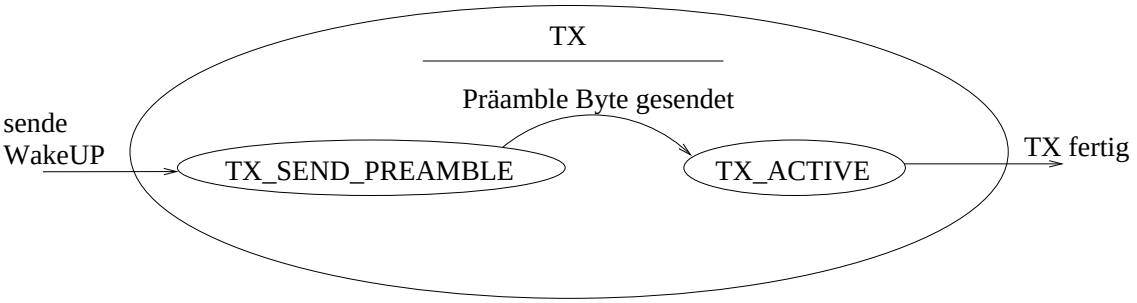


Abbildung 6.5: Verfeinerte Modellierung des TX-Zustandes

Verfeinerung von RX Der RX-Zustand wurde in gleich drei Zustände unterteilt siehe Abbildung 6.6. In RX_LISTEN_FOR_PREAMBLE wird auf ein Flankenwechsel nach folgendem Algorithmus eine Präamble erkannt:

- Warte eine feste Zeitspanne, bis die erste 1, welche vom Bitstuffing herrührt, vorbei ist.
- Zähle die Anzahl der Flankenwechsel von 1 nach 0 und von 0 nach 1.
- Wenn eine zu große Anzahl an Flankenwechsel in einer Bitzeit erkannt wurde, so verwerfe die Präambel.
- Wenn mindestens zwei Bitzeiten und weniger als neun Bitzeiten vergangen sind, so ist eine gültige Präambel gesendet worden.
- Nach neun Bitzeiten ohne gültige Präambel, ist der Empfangsvorgang abzubrechen.

In RX_HIGH und RX_LOW ist beim Empfangen einer WakeUp-Nachricht kodiert, ob eine logische-1 oder eine logische-0 zuletzt auf dem 433 MHz-Band lag. Da es unterschiedliche Verzögerungen bei Flankenwechsel gibt, wird nur wenn kein Flankenwechsel vorliegt gleichmäßig abgetastet. Eine Abtastung sollte in der Regel zur Mitte eines anliegenden Bits geschehen. Bei einem $0 \rightarrow 1$ Flankenwechsel wird die Abtastung um $192\,\mu s - 36\,\mu s = 156\,\mu s$ verzögert. Bei einem $1 \rightarrow 0$ Flankenwechsel wird die Abtastung um $156\,\mu s$ vorverschoben, beziehungsweise direkt ausgeführt.

6.2.2 Abbildung auf die Architektur des ATtiny24

Für die Implementierung des Entwurfs ist es unerlässlich die Hardware-Architektur des ATtiny24 zu betrachten. Da Interrupts maßgebend sind, für die Abbildung des

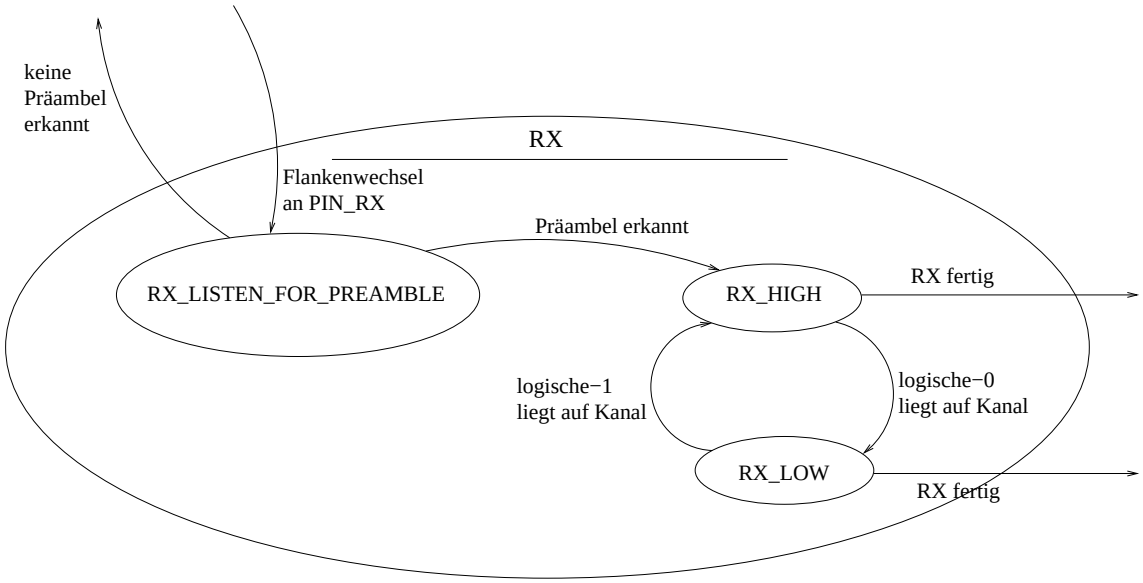


Abbildung 6.6: Verfeinerte Modellierung des RX-Zustandes

Entwurfs auf die Architektur wird in 6.2.2.1 zunächst ein Überblick über den Programmfluss gegeben. Daraufhin werden die Interrupts des Systems vorgestellt und analyisiert. Die Abbildung der Zustände auf bestimmte Interrupts wird anschließend vorgenommen. Da die Geschwindigkeitseinstellungen entscheidenden Einfluss auf den Energieverbrauch haben, sind diese auch dokumentiert.

6.2.2.1 Überblick über den Programmfluss

Im Programm des ATtiny24 befindet sich der Programmfluss meist in einer von zwei möglichen Positionen. Wie in Abbildung 6.7 zu sehen ist, wird zuerst ein Schlafmodus angenommen. Dieser ist vom jeweiligen Zustand des Programmes abhängig. Befindet sich der ATtiny24 in einem Schlafmodus, so wartet er bis ein Interrupt gestartet wurde. In der Behandlung des Interrupts können neue Zustände angenommen werden, Pins gesetzt werden und globale Variablen verändert werden. Ist der Interrupt abgearbeitet, so wird wieder ein Schlafmodus angenommen. Verschachtelte Interrupts sind nicht vorgesehen.

6.2.2.2 Interrupts des ATtiny24

Hier werden die verwendeten Interrupts vorgestellt. Neben ihrer Anwendung ist ferner dokumentiert, an welchem Pin zu Beginn des Interrupts ein Flankenwechsel stattfindet. Somit lässt sich das Programm in der Ausführung testen. Die Tranistionen, aus den oben genannten Zustandsautomaten, lassen sich teilweise auf das Auslösen von Interrupts abbilden. Wenn dies der Fall ist, so ist in Tabelle 6.2 diese Abbildung durch *eine kursive Schrift* gekennzeichnet. Die Bezeichung der I^2C-Interrupts als USI-Interrupts liegt an der Festlegung der Architektur des ATtiny24.

Name des Interrupts (Kürzel) und Flankenwechsel des entsprechenden Pins	Auslösen in der Anwendung	Wird geweckt aus dem Schlafmodus
Power-on Reset (*RESET*)	Dieser Interrupt wird von der Hardware ausgelöst, sobald der ATtiny24 mit Strom versorgt wird.	Vorher war der Mikrocontroller nicht aktiv, da kein Strom anlag.
Pin Change Interrupt Request 0 (PCINT0) PIN_CHECK_PCINT	Sobald ein *Flankenwechsel an PIN_RX* geschieht, löst dieser Interrupt aus.	Idle-Mode, Power-Down-Mode
Watchdog Time-out (WDT)	Acht Sekunden nachdem eine I²C-Kommunikation begonnen wurde, aber nicht erfolgreich zu Ende geführt wurde, wird dieser Interrupt ausgelöst. Dies ermöglicht bei einem falschen Erkennen einer I²C-Startbedingung, dass der ATtiny24 trotzdem weiterarbeiten kann. Dies ist eine mögliche Quelle für einen *I²C-Fehler*.	Idle-Mode, Power-Down-Mode
Timer/Counter0 Overflow (TIM0_OVF) PIN_CHECK_TIMER	Dieser Zeitzähler sorgt dafür, dass das richtige Bit zum richtigen Zeitpunkt entweder an den PIN_TX zum Senden gelegt wird, oder am PIN_RX gelesen wird. Der Interrupt wird ausgelöst, sobald der Zähler einen Überlauf hat. Um die Zeit zu steuern, bis dieser Interrupt kommt, wird das Zählregister TCNT0 manipuliert.	Idle-Mode
USI START (USI_STR) PIN_CHECK_ I2C_START_INT	Dieser Interrupt wird ausgeführt, sobald ein *I²C-Start* erkannt wurde.	Idle-Mode, Power-Down-Mode
USI Overflow (USI_OVF) PIN_CHECK_ I2C_OVF_INT	Dieser Interrupt wird ausgeführt, wenn der I²C-Zähler einen Überlauf hat. Dies passiert entweder nachdem *8 Bit empfangen* wurden oder wenn *ein ACK-Bit* gesendet wurde.	Idle-Mode
Sonstige Interrupts	werden nicht verwendet	-

Tabelle 6.2: Verwendete Interrupts und ihre Anwendung.

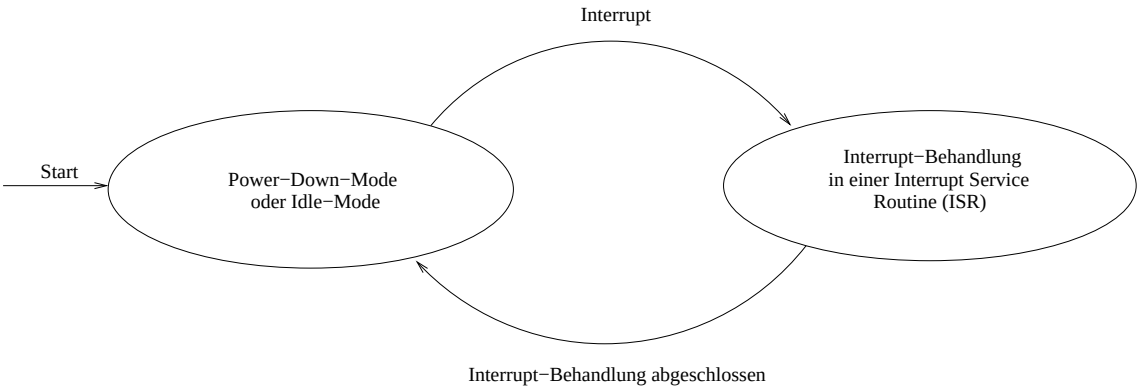


Abbildung 6.7: Programmfluss im ATtiny24-Programm.

Zustand	mögliche Interrupts	Schlafmodus
STARTING	keine	nur Aktiv
INACTIVE	PCINT0, USI_STR	Power-Down
RX_LISTEN_FOR_PREAMBLE	PCINT0, TIM0_OVF	Idle
RX_ACTIVE_HIGH	PCINT0, TIM0_OVF	Idle
RX_ACTIVE_LOW	PCINT0, TIM0_OVF	Idle
TX_SEND_PREAMBLE	TIM0_OVF	Idle
TX_ACTIVE	TIM0_OVF	Idle
I²C_SLAVE	USI_OVF	Idle
I²C_SLAVE_SEND_ACK	USI_OVF	Idle

Tabelle 6.3: Interrupts und Schlafmodus, die in den jeweiligen Zuständen angenommen werden.

6.2.2.3 Abbildung der Zustände auf Interrupts

Mit den oben genannten Interrupts ist noch nicht festgelegt, welche Interrupts in welchem Zustand ausgelöst werden können. Diese Festlegung wird in Tabelle 6.3 getroffen. In STARTING wird die Hardware initalisiert, währenddessen sollte noch kein Interrupt stattfinden. In INACTIVE befindet sich der ATtiny24 in seinem tiefsten Ruhezustand. Der PCINT0-Interrupt wird benutzt, um einen Flankenwechsel am PIN_RX zu registieren und zu verarbeitet. Der USI_STR-Interrupt meldet eine I²C-Startbedingung. In den RX-Zuständen wird während einer Bitzeit ein Flankenwechsel im Interrupt PCINT0 registiert. Erst im Interrupt TIM0_OVF werden die Flankenwechsel ausgewertet. Die TX-Zustände benötigen nur Timer-Interrupts, da sie keine Synchronisation mit anderen Komponenten haben. Für die I²C-Zustände wird der spezielle I²C-Zeitzähler verwendet, der bei einem Überlauf den Interrupt USI_OVF auslöst.

6.2.2.4 Geschwindigkeitskonfiguration des ATtiny24

Da die Taktfrequenz einen entscheidenden Einschluss auf den Energiebedarf hat, wird durch eine günstige Konfiguration möglichst viel Energie eingespart. Der ATtiny24 benutzt einen internen 8-MHz-Oszillator, der per Hardware-Einstellung (mit CKDIV8) auf ein Achtel heruntergetaktet wird. Ist der ATtiny24 aktiv, so wird er mit 1 MHz betrieben. Vor Idle-Phasen wird der Uhrenteiler (Englisch: Clock-Prescaler) auf 256 gesetzt, sofern ein TX- oder ein RX-Zustand vorliegt. Somit wird eine Taktfrequenz von $\frac{1\,\mathrm{MHz}}{256} = 3.9\,\mathrm{kHz}$ erreicht. In $\mathrm{I^2C}$-Zuständen wird der Uhrenteiler auf 4 gesetzt, da der Prozessor mit mindestens 200 kHz betrieben werden muss, um die $\mathrm{I^2C}$-Signale zu verarbeiten. Die Taktfrequenz wird damit auf $\frac{1\,\mathrm{MHz}}{4} = 250\,\mathrm{kHz}$ reduziert. Nach einem Idle-Modus wird der Uhrenteiler auf 1 gesetzt, so dass Aktiv-Phasen möglichst schnell wieder beendet sind. Im Power-Down-Schlafmodus, der nur im Zustand INACTIVE angenommen wird, ist der Stromverbrauch unabhängig von der Taktfrequenz.

Kapitel 7

Evaluation

Es stellt sich die Frage nach der Leistungsfähigkeit des Systems. Das Hauptaugenmerk liegt dabei auf der Reduzierung des Energieverbrauchs. Dazu werden zunächst Annahmen und Benennungen vorgenommen, um darauf aufbauend den Empfangs- und Sendeverbrauch zu berechnen.

7.1 Benennungen und Annahmen

Es werden drei unterschiedliche Modelle analysiert. Das erste Modell besteht aus der in dieser Bachelorarbeit vorgestellten Lösung. Das zweite Modell basiert auf der Annahme, dass der Mikrocontroller weggelassen würde, und der Imote2 die Band-Externe-Kommunikation überwacht. Im dritten Modell wird eine rein Band-Interne-Kommunikation über den CC2420 analyisiert, bei welcher der Imote2 die Kommunikation überwacht. Dafür ist es unerlässlich Annahmen zu treffen, so dass sich eine sinnvolle Vereinfachung der Rechnungen ergibt.

Die erste Annahme betrifft das Verhältnis von aktiven Prozessorphasen zu inaktiven Prozessorphasen während einer Funkübertragung. Da hierbei Zeitzähler (Englisch: Timer) laufen müssen, ist im ersten Modell der Idle-Mode des ATtiny24 aktiv. Im zweiten und dritten Modell jeweils der Deep-Idle-Mode des Imote2. Die Phasen, in denen der jeweilige Idle-Modus verlassen wird, werden als so kurz angenommen, dass sie vernachlässigbar sind.

Als zweites sind die Folgen von Flankenwechsel, die auf dem 433 MHz-Kanal liegen können zu unterscheiden. Sollte eine Folge eine WakeUp-Nachricht sein, so ergibt sich für die Band-Externe-Kommunikation eine Zeit von $t_{WakeUp} = \frac{3 \cdot (8+2) \cdot \mathrm{Bit}}{2\,\mathrm{kbps}} = 15\,\mathrm{ms}$. Die Häufigkeit einer WakeUp-Nachricht sei mit f_{WakeUp} bezeichnet. Für willkürliche Störungen auf dem Signalisierungskanal oder den Empfang von Rahmen anderer Sender, die nicht zum Signalisierungssystem gehören, lässt sich nur mit einem Parameter die Häufigkeit abschätzen, mit der diese Signale kommen. Diese Frequenz wird mit f_{waste} bezeichnet. Es wird angenommen, dass anhand der Präambel-Prüfung erkannt wird, dass das Signal nicht zum System gehört und entsprechend verworfen werden sollte. Daher dauert die Analyse dieses Signals $t_{waste} = \frac{1}{3} t_{WakeUp} = 5\,\mathrm{ms}$.

Um die Batterielebensdauer abzuschätzen, ist die vorhandene Batteriekapazität entscheidend. Gegeben ist auf der Imote2-Plattform eine Batteriekapazität von

1100 mAh. Da der Imote2 mindestens 1 mA im Deep Sleep Mode verbraucht und zusätzlich noch 0.02 mA für den CC2420-Transceiver, ergibt sich eine maximale Laufzeit von 45 Tagen.

7.2 Berechnung des Energieverbrauchs

In diesem Abschnitt werden die drei Modelle miteinander verglichen. Es erfolgt eine Unterscheidung in einen Ruhezustand, bei dem kein Empfang oder Senden stattfindet, in einen Empfangsvorgang und in einen Sendevorgang. Entscheidend ist, dass besonders im Ruhezustand wenig Energieverbraucht wird. Auch Empfangsvorgänge sollten sparsam sein, da von häufigerem Empfang ausgegangen wird. Für das Senden ist ein höherer Stromverbrauch akzeptabel, da nur von einem seltenen Senden ausgegangen wird. Eine Abschätzung der Häufigkeiten der verschiedenen Phasen erfolgt abschließend.

7.2.1 Ruhezustand

Im Ruhezustand liegt kein Flankenwechsel auf dem 433 MHz-Funkkanal an. Für das dritte Modell liegt entsprechend keine Signalisierung über das IEEE-802.15.4-Protokoll vor. Der Energieverbrauch für die einzelnen Modelle errechnet sich wie gefolgt:

erstes Modell

- Auf der Imote2-Plattform ist der Prozessor im Deep-Sleep-Mode und der CC2420 im Power-Down-Mode. Zusammen verbrauchen sie 1.02 mA.
- Der ATtiny ist im Power-Down-Mode und verbraucht 0.0001 mA.
- Der RX-4M30RR01SF verbraucht konstant 0.07 mA.

Der gesamte Verbrauch beträgt also $I_{rxSleep1} = 1.09\,\text{mA}$.

zweites Modell

- Auf der Imote2-Plattform ist der Prozessor im Deep-Idle-Mode und der CC2420 im Power-Down-Mode. Zusammen verbrauchen sie 27.02 mA. Ein energiesparender Schlafmodus ist nicht wählbar, da die Zeit, bis der Imote2 auf einen Flankenwechsel reagieren kann, zu groß ist. Außerdem würde das Aufwachen aus einem tieferen Schlafmodus wegen der langen Aufweckzeit relativ viel Energie kosten.
- Der RX-4M30RR01SF verbraucht konstant 0.07 mA.

Der gesamte Verbrauch beträgt also $I_{rxSleep2} = 27.09\,\text{mA}$. Man sieht, dass das Weglassen des Mikrocontrollers keine Ersparnis herbeiführt.

drittes Modell

- Auf der Imote2-Plattform ist der Prozessor nur im Deep-Idle-Mode und der CC2420 im Receive-Mode betreibbar. Zusammen verbrauchen sie $I_{rxSleep3} = 45.8\,\text{mA}$.

Der gesamte Verbrauch ist also ungefähr 40 mal höher als in den zuvorgezeigten Modellen.

7.2.2 Empfangsvorgänge

Bei einem Empfangsvorgang liegt eine Folge von Flankenwechsel auf dem 433 MHz-Funkkanal an. Für das dritte Modell liegt entsprechend ein Rahmen des IEEE-802.15.4-Protokoll vor. Der Energieverbrauch für die einzelnen Modelle errechnet sich wie gefolgt:

erstes Modell

- Auf der Imote2-Plattform ist der Prozessor im Deep-Sleep-Mode und der CC2420 im Power-Down-Mode. Zusammen verbrauchen sie 1.02 mA.
- Der ATtiny ist meistens im Idle-Mode und verbraucht 0.07 mA
- der RX-4M30RR01SF verbraucht konstant 0.07 mA

Der gesamte Verbrauch beträgt also $I_{rxIdle1} = 1.11\,\text{mA}$.

zweites Modell

- Auf der Imote2-Plattform ist der Prozessor meistens im Deep-Idle-Mode und der CC2420 im Power-Down-Mode. Zusammen verbrauchen sie 27.02 mA.
- Der RX-4M30RR01SF verbraucht konstant 0.07 mA

Der gesamte Verbrauch beträgt also $I_{rxIdle2} = 27.09\,\text{mA}$. Man sieht, dass das Weglassen des Mikrocontrollers zu einer drastischen Verschlechterung führt.

drittes Modell

- Auf der Imote2-Plattform ist der Prozessor nur im Deep-Idle-Mode und der CC2420 im Receive-Mode betreibbar. Zusammen verbrauchen sie $I_{rxIdle3} = 45.8\,\text{mA}$.

Der gesamte Verbrauch ist immernoch ca. 40 mal höher als im ersten Modell.

7.2.3 Sendevorgänge

Bei Sendevorgängen wird der Overhead, der durch die I²C-Kommunikation entsteht, vernachlässigt.

erstes Modell

- Auf der Imote2-Plattform ist der Prozessor im Deep-Sleep-Mode und der CC2420 im Power-Down-Mode. Zusammen verbrauchen sie 1.02 mA.
- Der ATtiny ist meistens im Idle-Mode und verbraucht 0.07 mA.
- Der TX-4MSIL verbraucht 6 mA.

Der gesamte Verbrauch beträgt also 7.09 mA für $t_{tx1} = t_1 = 15\,\mathrm{m}s$. Für $U = 4.5\,\mathrm{V}$ erhält man $W = t_{tx1} \cdot 7.09\,\mathrm{mA} \cdot U = 479\,\mu\mathrm{J}$.

zweites Modell

- Auf der Imote2-Plattform ist der Prozessor meistens im Deep-Idle-Mode und der CC2420 im Power-Down-Mode. Zusammen verbrauchen sie 27.02 mA.
- Der TX-4MSIL verbraucht 6 mA.

Der gesamte Verbrauch beträgt also 33.09 mA für $t_{tx2} = t_1 = 15\,\mathrm{m}s$. Für $U = 4.5\,\mathrm{V}$ erhält man $W = t_{tx2} \cdot 33.09\,\mathrm{mA} \cdot U = 2.23\,\mathrm{mJ}$. Man sieht, dass das Weglassen des Mikrocontrollers zu einer drastischen Verschlechterung führt.

drittes Modell

- Auf der Imote2-Plattform ist der Prozessor nur im Deep-Idle-Mode und der CC2420 im Transceive-Mode betreibbar. Zusammen verbrauchen sie 40 mA für $t_{tx3} = (16\,\mathrm{Bit} + 6\,\mathrm{Bit})/250kbps = 88\,\mu\mathrm{s}$. Dabei gehörten die 16 Bit zum zusendenden ID-Vektor und 6 Bit kommen durch den Overhead des IEEE-802.15.4-Protokolls.

Der Verbrauch ist hierbei mit $W = t_{tx3} \cdot 40\,\mathrm{mA} \cdot U = 15.8\,\mu\mathrm{J}$, wobei $U = 4.5\,\mathrm{V}$, gegeben. Die deutliche Energieeinsparung beim Senden über den CC2420 belegt, dass es nicht sinnvoll ist, weitergehende Signalisierung über das 433 MHz-Band zu leiten.

7.2.4 Batterielebenszeiten

Abschließend soll die Lebensdauer des Systems für die oben genannte Batteriekapazität berechnet werden. Da das Verhältnis $r_{WakeUpwase} = \frac{f_{WakeUp}}{f_{waste}}$ willkürlich ist, können, wie in Tabelle 7.1 zu sehen, verschiedene Werte zum Rechnen verwendet werden. Dabei ergit sich eine durchschnittliche Wachzeit mit $t_{wuw} = (1-r) \cdot t_{Waste} + r \cdot t_{WakeUp}$.

r	0	0.25	0.5	0.75	1
t_{wuw}/ms	5	6.25	7.5	8.75	15

Tabelle 7.1: Zeit, die das System braucht, um eine Nachricht zu verarbeiten in Abhängigkeit vom Verhältnis der Häufigkeit der Nachrichten des Signalisierungssystems zu der Häufigkeit der Nachrichten, die nicht zum Signalisierungssystem gehören.

Für die weitere Betrachtung wird angenommen, dass nur ein Empfangsvorgang stattfindet. Für t_{wuw} wird nur der Fall $t_{wuw} = 7.5\,\text{ms}$ betrachtet, da die anderen Werte ähnlich sind. In Tabelle 7.2 ist für verschiedene Frequenzen $f_{wuw} = f_{WakeUp} + f_{waste}$ die Kanalbelegung und die Batterielebenszeit in Tagen für die jeweiligen Modelle angegeben. Die Lebenszeit wurde berechnet mit $\frac{1100mAh \cdot \frac{1d}{24h}}{f_{wuw} \cdot t_{wuw} \cdot I_{rxIdleQ} + (1 - f_{wuw} \cdot t_{wuw}) \cdot I_{rxSleepQ}}$, wobei Q für das jeweilige Modell steht.

Während für das erste und zweite Modell unabhängig von der Kanalbelegungsdauer jeweils einen hohen Stromverbrauch aufweisen, und somit geringe Batterielebenszeiten, ist für die in dieser Arbeit entwickelte Lösung mit 42 Tagen eine deutliche Verbesserung erreicht worden. Zu beachten ist, dass die Rechnung nur den Empfangsverbrauch berücksichtigt. Dies ist gerechtfertigt, da wir davon ausgehen, dass Sendevorgänge, die eine Aufwecksignalisierung sind, sehr selten vorkommen.

f_{wuw}	$\frac{1}{15ms}$	$\frac{1}{50ms}$	$\frac{1}{100ms}$	$\frac{1}{1s}$	$\frac{1}{10s}$	$\frac{1}{100s}$	$\frac{1}{1000s}$
Kanalbelegung	0.5	0.15	0.075	$7.5 \cdot 10^{-3}$	$7.5 \cdot 10^{-4}$	$7.5 \cdot 10^{-5}$	$7.5 \cdot 10^{-6}$
Batterielaufzeit in Tagen für das ...							
... erste Modell	41.7	41.9	42.0	42.0	42.0	42.0	42.0
... zweite Modell	1.69	1.69	1.69	1.69	1.69	1.69	1.69
... dritte Modell	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Tabelle 7.2: Angabe der Batterielaufzeit in Tagen für die Modelle unter Berücksichtigung verschiedener Kanalbelegungen.

Kapitel 8

Zusammenfassung und Ausblick

Die Evaluation in Kapitel 7 des in dieser Bachelorarbeit neu entwickelten Systems zeigt, dass eine deutliche Energieeinsparung möglich ist. Die Vorteile einer langsamen und energiesparsamen Hardware wurden mit den Vorteilen einer leistungsfähigen und entsprechend energieverbrauchenden Hardware kombiniert.

Es zeigte sich, dass der Energieverbrauch beim Empfangen deutlich geringer ist, als bei einer Band-Internen-Kommunikation oder einer Band-Externen-Kommunikation ohne Mikrocontroller. Der Mikrocontroller ermöglicht das Filtern von unerwünschten Signalen. Durch die Eigenschaft verschiedene IDs anzusprechen, ist es möglich Sensorknoten gezielt aufzuwecken.

Der Mikrocontroller nutzt alle Schalfmodi aus, dadurch kann während des Sendens oder Empfangens nochmals viel an Energie eingespart werden, verglichen mit einem Busy-Wait. Durch die Pufferung der I^2C-Daten, muss der Imote beim Senden nicht auf den Mikrocontroller warten und kann selber wieder schneller in einen Schlafmodus übergehen.

Da es schwierig ist eingebettete Systeme zu testen, wird an entscheidenden Stellen der Wert eines Pins geändert, so dass sich Übergange nachvollziehen lassen. Der Aufruf von Interrupts kann so meistens nachvollzogen werden. Der Aufbau einer Hardware-In-The-Loop-Anlage, bei der jede Komponente, wie beispielsweise der ATtiny24, in einer Testumgebung einzeln getestet wird ist wünschswert. Da die Entwicklung einer solchen Testumgebung sehr aufwändig gewesen wäre, konnte sie leider nicht umgesetzt werden.

Durch die Verringerung des Energiebedarfs ist es möglich, den Sensorknoten bis zu 42 Tage ohne Batteriewechsel betreiben zu können. Gegenüber dem vorherigen Stand mit einem Tag ist dies ein Fortschritt. Denkt man an die Anwendung des Routing-Systems, so müsste eine Batterie ca. einmal im Monat gewechselt werden. Auch wenn mehrere Monate oder gar Jahre wünschenswert sind, so wurden akzeptable Batterielaufzeiten erreicht.

Die Integration von geeigneten PMIC (Power Management Integrated Circuit) ist sinnvoll. Insbesondere um die Spannungsversorgung des Receivers und Mikrocontrollers effektiver zu gestalten.

Zur Erhöhung der Lebendauer ist eine eigene Energiegewinnung, zum Beispiel mit Solarzellen, denkbar. Trotz Energiegewinnung ist es sinnvoll, möglichst energiespa-

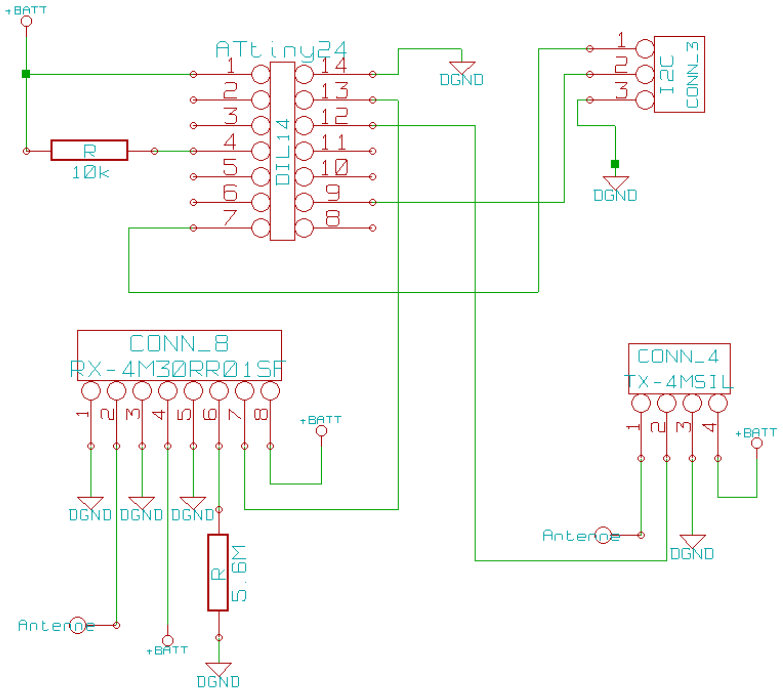
rend zu entwickeln. Zum Ersten verursachen Energiegewinnungsanlagen, wie beispielsweise Solarzellen, Kosten. Zum Zweiten benötigt die Energiegewinnung meist bestimmte Umweltsituationen, wie Sonne, Wind oder Bewegungen. Sollten diese für längere Zeit nicht oder nur unzureichend zur Verfügung stehen, so ist trotzdem eine möglichst lange Betriebsdauer gewünscht.

Weiterentwicklungen, wie zum Beispiel eine Erweiterung des Adressraumes auf mehr als 16 Knoten, sind denkbar. Auch deutlich weitergehende Arbeit, wie den Aufbau eines Simulators, der einem viele zeitaufwändige Tests auf der Hardware erspart, den Überlegungen zu intelligenten Aufwach-Algorithmen und der Einsatz des Systems in konkreten Anwendungsfällen, stehen noch aus. Dafür steht nun aber zumindest eine evaluierte Hardware-Plattform als Grundlage zur Verfügung, die weitere Entwicklungen ermöglicht.

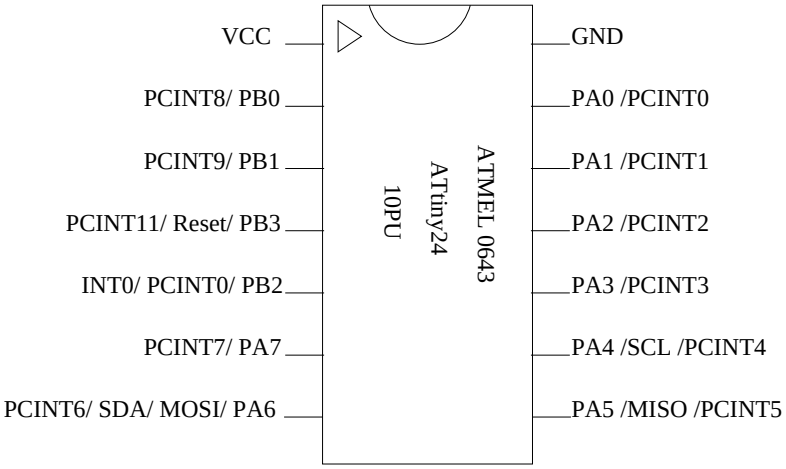
Anhang A

Elektrische Schaltungen und Pin Belegungen

A.1 Schaltplan der Anlage



A.2 Pin Belegung des Mikrocontrollers



Logischer Pin Name	Physischer Pin
PIN_RX	PA0
PIN_TX	PA1
PIN_CHECK_PCINT	PA2
PIN_CHECK_TIMER	PA3
PIN_SCL	PA4
PIN_CHECK_BEGIN	PA5
PIN_SDA	PA6
PIN_WAKE_UP_IMOTE	PA7
PIN_CHECK_HIGH	PB0
PIN_CHECK_I2C_OVF_INT	PB1
PIN_RX_SWITCH	PB2
PIN_CHECK_I2C_START_INT	PB0

Anhang B

Anmerkungen zur Entwicklung des ATtiny24

Für die praktische Weiterentwicklung des Programmes, ist es hilfreich weitere Details der Umsetzung zu kennen. Diese sind in diesem Abschnitt gezeigt und erläutert, sofern sie nicht schon in der eigenlichen Ausarbeitung dokumentiert sind.

B.1 Entwicklungswerkzeuge und -Hilfen

Die Entwicklung des Programmes des ATtiny24 benötigte einige Programme und Hilfen, die hier dokumentiert sind:

avr-gcc Dies ist ein C-Kompiler [1], der frei für Linux-Plattformen verfügbar ist. Mit ihm lassen sich Programme für unterschiedliche AVR-Plattformen kompilieren.

avr-libc Dies ist eine Funktionsbibliothek [1], die viele hardware-spezifische Funktionen zur Verfügung stellt.

avrdude Der avrdude [2] ermöglicht das Übertragen eines Programmes vom Personal Computer auf den Mikrocontroller.

Atmel Anwendungshinweise Neben dem Datenblatt des AVR ATtiny24 [4] stellt Atmel weitere Anwendungshinweise bereit [3]. So zum Beispiel auch das *AVR312: Using the USI module as a I2C slave*-Dokument, dass für die Benutzung der I²C-Komponente hilfreich ist. Da die Muster-Implementierung von I²C ohne Interrupts und nur mit busy-wait arbeitet, musste sie grundlegend überarbeitet werden.

B.2 Konfigurationsmöglichkeiten im Code

Einfache Änderungen im Code können in der Datei rtxConfig.c vorgenommen werden. Dabei kann die entsprechende I²C-Adresse und die Timer der 433 MHz-Kommunikation eingestellt werden.

B.3 Pin Belegung ändern

In der Datei rtxConfig.h befindet sich die Abbildung von logischen Pins auf reale Pins. Eine Umkonfiguration von PIN_CHECK_BEGIN, welches Inital auf PA5 gelegt ist, auf PB1 folgendermaßen geschehen:

```
//Vorher:
#define PIN_CHECK_BEGIN     PA5
#define PORT_CHECK_BEGIN &(PORTA)
#define DDR_CHECK_BEGIN    &(DDRA)
#define INPUT_CHECK_BEGIN &(PINA)

//Nachher:
#define PIN_CHECK_BEGIN     PB1
#define PORT_CHECK_BEGIN &(PORTB)
#define DDR_CHECK_BEGIN    &(DDRB)
#define INPUT_CHECK_BEGIN &(PINB)
```

Dananch sind keine weiteren Änderungen mehr nötig. Hinweis: Gewisse Pins können von der Hardware schon festgelegt worden sein, so beispielsweise die Pins der I^2C-Kommunikation.

B.4 Anleitung: Kompilieren eines Programmes für den ATtiny24

Zum Kompilieren wurde ein Makefile erstellt, dass den avr-gcc als Kompiler benutzt. Mit *make full* wird kompiliert. Wichtig sind die Kompileroptionen, so ist -Os benötigt. *-mint8* bedeutet, dass alle Operatoren auf 8-Bit-Basis laufen. Ansonsten hat man beispielsweise beim Oder-Operator || das Problem, dass vor der Andwendung eine unnötige Konvertierung der 8-Bit Variablen x und y geschieht. Im folgenden ist dies mit einem Kodebeispiel und die Umsetzung in Assemblerkode demonstriert. Zu sehen ist, dass der Umfang an Kode deutlich zunimmt.

```
//Beispielfunktion in der Programmiersprache C
uint8_t test(uint8_t x, uint8_t y){
return (x || y);
}

Übersetztes Assemblerprogramm, kompiliert mit Kompilieroption -mint8
0000021e <test>:
 21e:   90 e0           ldi     r25, 0x00       ; 0
 220:   88 23           and     r24, r24
```

```
 222:   09 f4           brne    .+2             ; 0x226 <test+0x8>
 224:   61 11           cpse    r22, r1
 226:   91 e0           ldi     r25, 0x01       ; 1
 228:   89 2f           mov     r24, r25
 22a:   08 95           ret

Übersetztes Assemblerprogramm, kompiliert ohne Kompileroption -mint8
00000250 <test>:
 250:   20 e0           ldi     r18, 0x00       ; 0
 252:   30 e0           ldi     r19, 0x00       ; 0
 254:   88 23           and     r24, r24
 256:   11 f4           brne    .+4             ; 0x25c <test+0xc>
 258:   66 23           and     r22, r22
 25a:   11 f0           breq    .+4             ; 0x260 <test+0x10>
 25c:   21 e0           ldi     r18, 0x01       ; 1
 25e:   30 e0           ldi     r19, 0x00       ; 0
 260:   93 2f           mov     r25, r19
 262:   82 2f           mov     r24, r18
 264:   08 95           ret
```

B.5 Anleitung: Übertragen eines Programmes auf den ATtiny24

Um das kompilierte Programm von einen gewöhnlichen Computer auf den Mikrocontroller zu bringen wird die Software *avrdude* verwendet. Der Befehl zum Übertragen eines Programmes lautet

```
avrdude -p t24 -c dapa -e -i 25 -U flash:w:main.hex
```

Literaturverzeichnis

[1] *AVR Libc Home Page*. http://www.nongnu.org/avr-libc/.

[2] *AVRDUDE - AVR Downloader/UploaDEr*. http://www.nongnu.org/avrdude/.

[3] Atmel: *Applications Notes*. http://www.atmel.com/dyn/products/app_notes.asp?family_id=607#Application_Example_and_Algorithms.

[4] Atmel: *ATMEL 8-bit AVR Microcontroller with 2/4/8K Bytes In-System Programmable Flash ATtiny24/44/84 Preliminary*, 2008. http://www.atmel.com/dyn/products/product_card.asp?part_id=3827.

[5] aurel: *Low Cost Micro Amps Receiver*. http://www.aurelwireless.com/wireless/Short_Form/650200737_sf.pdf.

[6] aurel: *TX-4MSIL 434 MHz OOK AM Transmitter with Low Consumption*, 2003. http://www.aurelwireless.com/wireless/Short_Form/650200737_sf.pdf.

[7] aurel: *Electrostatic features for RF receivers - Antenna isolation*, 2005. http://www.aurelwireless.com/wireless/Notes/isolation-antennas-application-notes.pdf.

[8] Bluetooth SIG, Inc.: *Technical Comparison*, 2009. Online: http://german.bluetooth.com/Bluetooth/Technology/Works/Compare/Technical/; Stand 7. September 2009.

[9] Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen: *Allgemeinzuteilung von Frequenzen für nichtöffentliche Funkanwendungen geringer Reichweite; Non-specific Short Range Devices (SRD)*. Vfg Nr. 30 / 2006; http://www.bundesnetzagentur.de/media/archive/6709.pdf.

[10] Crossbow: *Imote2 High-Performance Wireless Sensor Network Node*. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/Imote2_Datasheet.pdf.

[11] Crossbow: *Offical Website*. http://www.xbow.com/.

[12] Engel, M.: *Entwicklung eines flexiblen und robusten Bootloaders für die Imote2 Plattform Bachelorarbeit, AG Vernetzte Systeme, Fachbereich Informatik, Technische Universität Kaiserslautern*, Mai 2009.

[13] Gotzhein, R., M. Krämer, L. Litz und A. Chamaken: *Energy-aware System Design with SDL*. 2009.

[14] Gu, L. und J. A. Stankovic: *Radio-Triggered Wake-Up for Wireless Sensor Networks*. Real-Time Syst., 29(2-3):157–182, 2005.

[15] Haenggi, M.: *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*, Kap. Opportunities and challenges in wireless sensor networks. CRC Press, Florida, 2005.

[16] Intel: *Intel PXA270 Processor Electrical, Mechanical, and Thermal Specification*. `www.phytec.com/pdf/datasheets/PXA270_DS.pdf`.

[17] Intel: *Intel PXA27x Processor Family Electrical, Mechanical, and Thermal Specification*. `www.penguin.cz/~utx/zaurus/datasheets/CPU/28000304.pdf`.

[18] Intel: *Intel PXA27x Processor Family Developer's Manual*, Oktober 2004. `http://int.xscale-freak.com/XSDoc/PXA27X/2800002.pdf`.

[19] Intel: *Intel PXA27x Processor Family Specification Update*, Mai 2005. `http://int.xscale-freak.com/XSDoc/PXA27X/28007106.pdf`.

[20] Mishra, N., K. Chebrolu, B. Raman und A. Pathak: *Wake-on-WLAN*. In: *WWW '06: Proceedings of the 15th international conference on World Wide Web*, S. 761–769, New York, NY, USA, 2006. ACM.

[21] Orwell, G.: *1984*. Penguin Readers - Level 4, 1948.

[22] Philips Semiconductors: *THE I2C-BUS SPECIFICATION VERSION 2.1*, 2000. `http://www.nxp.com/acrobat_download/literature/9398/39340011.pdf`.

[23] Shih, E., P. Bahl und M. J. Sinclair: *Wake on wireless: an event driven energy saving strategy for battery operated devices*. In: *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, S. 160–171, New York, NY, USA, 2002. ACM.

[24] Texas Instruments: *CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver*, 2007.

[25] Wendt, T. M. und L. M. Reindl: *Wake-Up Methods to Extend Battery Life Time of Wireless Sensor Nodes*. I2MTC 2008 - IEEE International Instrumentation and Measurement Technology Conference Victoria, Vancouver Island, Canada, May 12-15, 2008, 2008.