Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Projekt-Ausarbeitung

Arno Davids

Entwicklung eines Sensornetzes zur Gebäudeüberwachung im Rescue-Umfeld

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Arno Davids

Entwicklung eines Sensornetzes zur Gebäudeüberwachung im Rescue-Umfeld

Projekt-Ausarbeitung eingereicht im Rahmen des Masterstudiums
im Studiengang Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuende Prüfer :
Prof. Dr. rer. nat. Kai von Luck
Prof. Dr. rer. nat. Gunter Klemke

Abgegeben am 28. Februar 2007

Inhaltsverzeichnis

1 Einleitung

In dieser Ausarbeitung sollen die Arbeiten erläutert werden, die von mir innerhalb des Projekt-Blocks im Wintersemester 2006/2007 an der Hochschule für Angewandte Wissenschaften Hamburg (HAW) durchgeführt worden sind. Dabei ging es darum erste praktische Versuche, aufbauend auf dem theoretischen Wissen aus Anwendungen I und dem parallel zum Projekt-Block laufenden Anwendungen II, durchzuführen. Dies soll im Rahmen eines Rescue-Projekts geschehen, an dem auch noch meine Kommilitonen Steffen Hinck und Andreas Piening beteiligt sind. Die im Projekt-Block gesammelten Erfahrungen sollen genutzt werden, um darüber hinaus in der anschließenden Masterarbeit auf diesem Projekt aufbauenden, erweiterte Problemfelder zu untersuchen.

Das dem Rescue-Projekt zu Grunde liegende Szenario wurde von mir bereits ausführlich in der Seminar-Ausarbeitung beschrieben, ebenso wie die Motivation zu diesem Projekt (vgl. [Davids (2007a)]).

Der vom mir im HAW Rescue-Projekt übernommene Teil befasst sich mit dem Sammeln und Erfassen von Daten innerhalb eines Gebäudes. Dieses Daten sollen ausgewertet werden und die daraus gewonnenen Erkenntnisse an den Leitstand weitergeleitet, um dort eine bessere Koordination und Leitung eines Einsatzes zu ermöglichen und damit ein Feuer effektiver bekämpfen und die Risiken für die Feuerwehrmänner im Gebäude minimieren zu können. In Piening (2006), Piening (2007) Hinck (2006) und Hinck (2007) werden die weiteren Bereiche innerhalb des HAW Rescue-Projekts erläutert. Der Bereich des Leitstandes, in dem ein Rettungseinsatz koordiniert wird und die dort stattfindende Kollaboration der Einsatzleiter wird von Andreas Piening realisiert. Steffen Hinck befasst sich mit der Ausrüstung des einzelnen Feuerwehrmanns und der Erstellung von technischen Hilfsmitteln, die ihn bei der Arbeit unterstützen.

Meine Ziele im Projekt, die ich in der Anwendungen II Ausarbeitung dargestellt habe, sind dabei folgende:

> „Der von mir bearbeitete Teil des Projekts befasst sich erst einmal damit, Daten innerhalb eines Gebäudes zu erfassen und einen Austausch mit der Leitstelle zu ermöglichen. Es soll ein drahtloses Sensornetzwerk aufgebaut werden, um flächendeckend Temperaturwerte erfassen zu können. Dadurch sollen [. . .]

> Feuer automatisch erkannt und an die Leitstelle gemeldet werden und die Ausdehnung des Brandes, sowie die Temperaturwerte in den einzelnen Bereichen des Gebäudes erfasst und auf einem Gebäudeplan in der Leitstelle dargestellt werden.“ [Davids (2007b)]

Die Anforderungen, die an den Aufbau eines drahtlosen Sensornetzes zu stellen sind, wie beispielsweise die Toleranz gegenüber einem Knotenausfall, wurden ebenfalls bereits in [Davids (2007a)] erörtert. Daher soll in dieser Ausarbeitung in Kapitel 2 mit einem Vergleich der in Frage kommenden Sensorplattformen für den Aufbau des Sensornetzes begonnen werden.

2 Grundlagen

Für den Aufbau eines Sensornetzes werden als Hauptbestandteil Sensorknoten benötigt. Daher sollen zunächst einige Plattformen vorgestellt werden. In einem nächsten Schritt erläutere ich meine Entscheidung für einen Sensorknoten bezüglich der Realisierung des Sensornetzes im HAW Rescue-Projekt. Daran anschließend sollen die Softwaregrundlagen beschrieben werden, die nötig sind, diesen Sensorknoten zu programmieren.

2.1 Auswahl der Sensorplattform

Es gibt mehrere Sensorknoten, die zum Aufbau eines Sensornetzes geeignet sind. Es soll in diesem Kapitel eine Auswahl der Sensorplattformen, mit denen ich mich beschäftigt habe, vorgestellt werden und daran anschließend eine Entscheidung für eine der Plattformen getroffen werden.

2.1.1 Sensorplattformen

In diesem Abschnitt sollen die Scatterweb-Plattform, die BTnode, die Sun SPOT-Plattform und die Berkeley Motes beschrieben werden. Eine weitere interessante Plattform, die Intel iMote2, die allerdings erst seit Februar 2007 verfügbar ist, kann hier leider nicht mehr berücksichtigt werden, da die Entscheidung schon im Herbst 2006 getroffen werden musste, um rechtzeitig mit den ersten Versuchen beginnen zu können. Andere Sensorplattformen wie die μAMPS des MIT bleiben leider aus Zeitgründen unerwähnt.

2.1.1.1 Scatterweb

Die ScatterWeb-Plattform ist eine Entwicklung der FU Berlin ([ScatterWeb (2007)]). Es wurden verschiedene Knotentypen realisiert, wie Sensor- oder Gateway-Knoten für die Anbindung des Sensornetzes an ein PC oder ein LAN. Als Sensorknoten entstand u. a. die Scatterweb ESB, wobei ESB für „Embedded Sensor Board“ steht.

Die Scatterweb ESB nutzt als Microcontroller den MSP430 der Firma Texas Instruments. Auf dem Sensorknoten ist das Funkmodul RFM TR1001, dass eine Reichweite von 300m außerhalb und 100m im Gebäude haben soll, verbaut, um eine Kommunikation zu anderen Sensorknoten im 868MHz-Bereich mit bis zu 115.2kbps zu ermöglichen. Bereits integriert sind u. a. ein Temperatur-, Helligkeits-, Infrarot-, Bewegungs- und Erschütterungssensor, ebenso wie ein Mikrofon. Es sind noch weitere externe Module verfügbar, wie z. B. die Möglichkeit eine Stromversorgung über Solarzellen zu realisieren oder ein GSM-Modul zu betreiben. Außerdem besteht die Möglichkeit externe I/O-Module anzuschließen.

Die Scatterweb ESB wird mit 3 AAA Batterien betrieben und es ist laut [ScatterWeb (2007)] in eine Demo-Anwendung mit einem duty-cycle von 1%, in der die Sensoren genutzt und kommuniziert wird, eine Lebenszeit von 5 Jahren möglich. Sie hat einen sehr niedrigen Stromverbrauch von nur 8μA im Schlafzustand. Zur Programmierung besitzt der Sensorknoten eine serielle Schnittstelle. Es sind einige Beispiel-Applikationen für die Sensoren verfügbar und es besteht die Möglichkeit das TinyOS-Betriebssystem, auf das in Kapitel 2.2 näher eingegangen werden soll, auf der Scatterweb ESB einzusetzen.

2.1.1.2 BTnode

Die BTnode, an der ETH Zürich entwickelt, ist eine drahtlose Kommunikations- und Berechnungsplattform, die im Gegensatz zu vielen anderen Plattformen, ohne integrierte Sensoren auskommt ([BTnode (2007)] und [Beutel u. a. (2007)]). Eine weitere Besonderheit dieser Plattform ist es, dass sie mit zwei unterschiedlichen Systemen zur Kommunikation ausgestattet ist.

Die BTnode rev3 ist mit einem 8 MHz-Microcontroller Atmel ATmega 128L ausgestattet. Sie verfügt über 64+180kByte RAM und 128kByte Flashspeicher. Die BTnode ist mit Bluetooth-Funk der Klasse 2 und einem Chipcon CC1000 Funkmodul ausgestattet. Durch diese beiden Kommunikationswege soll eine Kompatibilität sowohl zu den älteren BTnodes, die nur ein Bluetooth-Modul besitzen, als auch zu einigen Berkeley-Motes, auf die in Abschnitt 2.1.1.4 näher eingegangen wird, sichergestellt werden. Es ist dabei möglich beide Systeme gleichzeitig zu betreiben, wodurch es jedoch zu einem erhöhten Energieverbrauch kommt. Dies ist allerdings als besonders kritisch anzusehen, da für die Funkkommunikation die meiste Energie benötigt wird. Die BTnode hat einen relativ hohen Stromverbrauch von 50μA im Schlafzustand, gegenüber der Scatterweb ESB oder der TelosB-Mote, die hier wesentlich geringere Werte aufweisen. Die BTnode besitzt, wie bereits erwähnt, keine eigenen Sensoren. Es besteht jedoch die Möglichkeit, über verschiedene Schnittstellen externe Sensoren anzuschließen.

Auf den BTnodes läuft sowohl das TinyOS Betriebssystem (siehe Kapitel 2.2) als auch eine angepasste Variante des Nut/OS[1], einem Open Source Realtime-Betriebssystem.

2.1.1.3 SunSpot

Von der Firma Sun Microsystems stammt die Sun SPOT Sensorplattform ([Sun (2007)]). Einem Sensorboard auf dem, im Gegensatz zu vielen anderen Plattformen, eine Java Virtual Machine (VM) läuft. Dabei handelt es sich um die Squawk VM, eine Java 2 Micro Edition (J2ME) VM. Es werden also Applikationen für diese Sensorplattform in Java entwickelt, mit der Möglichkeit die Sensoren, die I/O-Pins oder das Funkmodul mit Hilfe von Java Bibliotheken anzusprechen.

Die Sun SPOT Plattform besteht aus einem so genannten Main-Board, dass die Grundfunktionalität der Plattform realisiert. Es enthält eine mit 180MHz getaktete 32-Bit ARM920T CPU, die mit 512kByte RAM und 4MB Flashspeicher ausgerüstet ist. Als Kommunikationseinheit wird ein Chipcon CC2420-Modul verwendet, mit dem eine Kommunikation nach IEEE 802.15.4/ ZigBee mit 2,4GHz im ISM-Band möglich ist. Zum Main-Board wurde ein Sensor-Board entwickelt, dass mit einem Beschleunigungs-, Temperatur- und einem Helligkeitssensor ausgestattet ist und zusammen mit dem Main-Board in einem Gehäuse ausgeliefert wird. Weiterhin ist es möglich an die Sun SPOT externe I/O-Module anzuschließen.

Die Sun SPOT Plattform wird mit einem 750mAh Lithium-Ionen-Akku betrieben. Über die Lebensdauer sind aber leider keine Informationen verfügbar, obwohl dies hier von großen Interesse wäre, bei der wesentlich leistungsstärkeren Hardwareaustattung des Main-Boards gegenüber den anderen in diesem Kapitel vorgestellten Plattformen. Obgleich dieser Sensorknoten schön im Frühjahr 2006 angekündigt wurde, ist es gegenwärtig noch nicht möglich diese Plattform zu erwerben, sondern sie wird derzeit von Sun nur ausgewählten Projekten zu Testzwecken zur Verfügung gestellt.

2.1.1.4 Berkeley Motes

Bei den so genannten Berkeley Motes handelt es sich um mehrere Sensorplattformen, die an der University of California at Berkeley entstanden sind ([Hill u. a. (2000)], [Polastre u. a. (2004b)]) und von der Firma Crossbow Technology Inc. vertrieben werden. Zuvor wurden an der UC Berkeley bereits Erfahrungen in der Entwicklung von Sensorknoten beispielsweise im Smart Dust-Projekt, dass in meiner Ausarbeitung zu Anwendungen II vorgestellt wurde (vgl. [Davids (2007b)]), gesammelt. Dort ging es in erster Linie darum, neue Techniken zum

[1] egnite Software GmbH: Nut/OS – http://www.ethernut.de/en/software/index.html

Mote Type	*WeC*	*René*	*René 2*	*Dot*	*Mica*	*Mica2Dot*	*Mica 2*	*Telos*
Year	1998	1999	2000	2000	2001	2002	2002	2004
							
Microcontroller								
Type	AT90LS8535		ATmega163		ATmega128			TI MSP430
Program memory (KB)	8		16		128			60
RAM (KB)	0.5		1		4			2
Active Power (mW)	15		15		8		33	3
Sleep Power (μW)	45		45		75		75	6
Wakeup Time (μs)	1000		36		180		180	6
Nonvolatile storage								
Chip	24LC256				AT45DB041B			ST M24M01S
Connection type	I^2C				SPI			I^2C
Size (KB)	32				512			128
Communication								
Radio	TR1000				TR1000	CC1000		CC2420
Data rate (kbps)	10				40	38.4		250
Modulation type	OOK				ASK	FSK		O-QPSK
Receive Power (mW)	9				12	29		38
Transmit Power at 0dBm (mW)	36				36	42		35
Power Consumption								
Minimum Operation (V)	2.7		2.7		2.7			1.8
Total Active Power (mW)	24				27	44	89	41
Programming and Sensor Interface								
Expansion	none	51-pin	51-pin	none	51-pin	19-pin	51-pin	10-pin
Communication	IEEE 1284 (programming) and RS232 (requires additional hardware)							USB
Integrated Sensors	no	no	no	yes	no	no	no	yes

Abbildung 2.1: Übersicht der Berkeley Motes [Polastre u. a. (2004b)]

Bau von Sensorknoten zu erforschen und damit Knoten von einer Größe im Bereich weniger Kubikmillimeter zu entwickeln. Diesem Designziel wurde bei den Motes nicht verfolgt, sondern hier sollten mit Standardkomponenten (COTS = commercial off-the-shelf) und den damit verbundenen Vorteilen Sensorknoten erschaffen werden.

Auf den Motes sind ein Mikrocontroller, eine Funkkommunikationseinheit, ein Flash-Speicher und ein Interface, an das z. B. Sensoren angeschlossen werden können, als Hauptbestandteile enthalten. Es sind von 1998 bis 2004 verschiedene Sensorplattformen entstanden (siehe Abbildung 2.1, die im Laufe der Zeit immer weiter optimiert wurden. So wurden bei den Motes beispielsweise unterschiedliche Microcontroller eingesetzt, die in den später hergestellten Sensorknoten immer leistungsfähiger wurden. Dabei ging die Entwicklung vom Atmel AT90LS8535 über den Atmel ATmega163 und den ATmega128 bis zum Texas Intruments MSP430 bei der Telos-Mote.

Bei der Telos-Mote in der Revison B (als TelosB bezeichnet) handelt es sich um die neueste und am weitesten entwickelte Mote. Sie besitzt einen MSP430 Microcontroller, 10kByte RAM, 1MByte Flashspeicher und ist mit dem Chipcon CC2420-Modul, zur drahtlosen Kommunikation mit bis zu 250kbps nach dem IEEE 802.15.4/ ZigBee Standard, ausgerüstet. Es wird eine in die Platine integrierte Antenne zur Kommunikation genutzt, mit der laut Da-

tenblatt[2] eine Reichweite von bis zu 100m außerhalb und 30m innerhalb eines Gebäudes möglich ist. Zur Verbesserung der Reichweite ist es möglich, eine externe Antenne anzuschließen. Der eingesetzte Microcontroller, ein 16-Bit-Microcontroller mit von Neumann-Architektur, ist im Hinblick auf einen geringen Energieverbrauch optimiert. Es sind bereits Temperatur-, Feuchtigkeits- und ein Helligkeitssensoren in die Mote integriert, es können aber auch weitere Sensoren über eine Schnittstelle angeschlossen werden.

Die TelosB-Mote zeichnet sich durch besonders geringen Energieverbrauch in der Schlafphase, die die meiste Zeit ausmacht, und durch eine besonders kurze Aufwachzeit aus (sleep: 2.4μA; wakeup-time: max. 6μs) (vgl. [Polastre u. a. (2004b)]). So soll laut [Polastre u. a. (2004b)] in einer Beispielanwendung mit einem duty-cycle von <1% eine Lebensdauer von 945 Tagen möglich sein, erzielt mit 2 handelsüblichen AA Batterien.

Parallel zu den Motes wurde an der UC Berkeley das TinyOS Betriebssystem entwickelt, dass auch auf der TelosB zum Einsatz kommt. Auf TinyOS wir in Kapitel 2.2 näher eingegangen.

2.1.2 Entscheidung für eine Sensorplattform

Eine Entscheidung für eine der genannten Sensorplattformen, soll anhand einiger, meiner Meinung nach, wichtigen Kriterien getroffen werden. Dazu möchte ich die Sensoren anhand dieser Punkte bewerten, aber auch ihre spezifischen Vor- auch Nachteile in die Entscheidung mit einbeziehen.

Die wesentlichen Entscheidungskriterien waren dabei:

- Lange Lebensdauer
- Geringe Größe
- Funkkommunikation
- Zuverlässigkeit
- Verfügbarkeit
- Offene Standards
- Verbreitung/ Community

Wenn man die vier hier vorgestellten Sensorknoten anhand des ersten Gesichtspunkts der Liste beurteilt, dann sind besonders die Scatterweb ESB und die TelosB-Mote in Bezug auf die lange Lebensdauer und die Leistungsfähigkeit interessant, da sie einen besonders geringen Energieverbrauch haben.

[2] http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf

Die Größenunterschiede zwischen den unterschiedlichen Plattformen sind eher gering und damit ist dieser Punkt für eine Beurteilung nachlässiger zu bewerten. Der meiste Raum wird in der Regel durch die Batteriehalterung beansprucht.

Zur Kommunikation benutzen zwei Plattformen, TelosB und Sun SPOT, das Chipcon CC2420-Modul mit dem eine Kommunikation nach dem IEEE 802.15.4/ ZigBee Standard möglich ist. Damit kann prinzipiell eine Verbindung zu weiteren Geräten, die den Gleichen Protokoll-Stack implementieren, aufgebaut werden. Außerdem bieten diese Module die höchsten Datenraten mit 250kbps.

Die Zuverlässigkeit der Sensorknoten lässt sich ohne Praxistest schwer beurteilen, jedoch spricht es besonders für die Berkeley Motes, dass bereits mehrere Generationen an der UC Berkeley entwickelt wurden und daher meiner Meinung nach weniger Probleme in Bezug auf die Zuverlässigkeit und Stabilität zu erwarten sind.

Da die Sensorknoten im HAW Rescue-Projekt dazu eingesetzt werden sollen, Temperaturwerte zu erfassen, ist es vorteilhaft, wenn ein entsprechender Sensor bereits auf der Plattform integriert ist, auch wenn man einen Temperatursensor über entsprechende Schnittstellen anschließen könnte. Bereits einen eingebauten Temperatursensor besitzen die TelosB-Mote, die Scatterweb ESB und die Sun SPOT. Bei der Scatterweb ESB ist bemerkenswert, dass bereits viele Sensoren integriert sind und es außerdem interessante Erweiterungsmodule gibt.

Die Besonderheit der Sun SPOT, die Fähigkeit Java-Code auszuführen, ist sehr interessant. Jedoch ist dafür auch ein wesentlich leistungsstärkerer Microcontroller unerlässlich und es ist mehr Rechenzeit nötig, um eine Java Virtual Maschine auszuführen, womit wiederum ein höherer Energieverbrauch verbunden ist. Auch wenn die Plattform bereits verfügbar wäre, würde sie meiner Meinung nach deswegen nicht in Frage kommen. Außerdem wären wohl noch viele „Kinderkrankheiten“ in Bezug auf die Java Implementierung zu erwarten.

Die Software Umgebung für den Betrieb der Sensorknoten ist auch ein Entscheidungskriterium. Hier sollte am Besten ein Open Source Produkt zum Einsatz kommen, dass durch eine breite Community unterstützt wird, da dies die besten Eingriffsmöglichkeiten bietet. Hier bittet sich TinyOS an, dass auf vielen Sensorplattformen einsetzbar ist. So kann es, bis auf die Sun Plattform, mit allen anderen hier vorgestellten Sensorknoten genutzt werden.

Nachdem ich mich mit mehreren Plattformen beschäftigt habe und die Vor- und Nachteile der einzelnen Sensorknoten erläutert wurden, sind die, meiner Meinung nach, am Besten geeigneten Plattformen die TelosB-Mote und die Scatterweb ESB. Nach Abwägung aller Aspekte fiel die Wahl für die Sensorplattform für das HAW Rescue-Projekt auf die Berkeley Motes und dabei auf die TelosB-Mote. Sie erfüllt die meisten der genannten Kriterien und soll daher dazu genutzt werden ein Sensornetz aufzubauen.

2.2 TinyOS

TinyOS ist ein Open-Source Betriebssystem für Motes, das ebenfalls an der University of California at Berkeley entwickelt wurde ([Hill u. a. (2000)]). Es wurde ursprünglich für die Berkeley Motes entwickelt, ist aber inzwischen als öffentliches Sourceforge[3] Projekt, auf vielen weiteren Plattformen, zusätzlich zu den in dieser Arbeit vorgestellten, einsetzbar.

Da Sensorknoten nur sehr begrenzte Ressourcen aufweisen (Energie, Programmspeicher, Arbeitsspeicher, Prozessorleistung, etc.) war es ein Ziel ein Betriebssystem zu entwickeln, dass sehr sparsam mit den Ressourcen umgeht. TinyOS ist somit, im Gegensatz zu anderen bereits vorhandenen Betriebssystemen für Embedded-Systeme, speziell auf diese Anforderungen zugeschnitten. Besonders die Funkkommunikation war eines der wichtigsten Punkte bei der Entwicklung von TinyOS, da für senden, empfangen und lauschen auf dem Medium besonders viel Energie notwendig ist. Weiterhin wurde beim Design besonders darauf geachtet, einen hohen Grad an Nebenläufigkeit zu unterstützen und eine effektive Modularität zu ermöglichen. Den Entwicklern von TinyOS war außerdem wichtig, dass die Möglichkeit besteht, auch zukünftige Sensorknoten mit veränderter Hardware zu unterstützen (vgl. [Levis u. a. (2004)]).

Beim Design von TinyOS wurde eine Komponentenarchitektur gewählt, bei der sowohl die Anwendungen, als auch das Betriebssystem aus einzelnen miteinander verbundenen Komponenten besteht und nur die jeweils benötigten Teile gemeinsam kompiliert werden. Die Anwendungen für die Sensorknoten werden in nesC, eine an C angelehnte Sprache, die speziell für Sensorknoten entwickelt wurde, programmiert (siehe [Gay u. a. (2003a)], [Gay u. a. (2003b)] und [Gay u. a. (2005)]). Seit Anfang November 2006 ist TinyOS in der Version 2.0 verfügbar die in diesem Projekt auch eingesetzt werden soll.

Die Nutzung von TinyOS hat viele Vorteile bei der Anwendungsentwicklung. So kann z. B. einfach das Modul, dass für das Senden einer Nachricht zu benutzen ist, aufgerufen werden und der Nutzer braucht sich um das konkrete Senden der einzelnen Bytes mit Hilfe der Hardwarekomponente nicht zu kümmern. Damit wird beispielsweise die Nutzung eines bestimmten MAC-Protokolls komplett vor dem Anwender verborgen. Tatsächlich wird das B-MAC (auch Berkeley-MAC genannt) Protokoll genutzt.

B-MAC wurde von Woo und Culler an der UC Berkeley im Jahre 2001 entwickelt (vgl. [Woo und Culler (2001)] und [Polastre u. a. (2004a)]). Beim Design dieses Protokoll wurde davon ausgegangen, dass kurzfristig ein hohes Verkehrsaufkommen entstehen kann und dass die übermittelten Pakete relativ kurz sind (ca. 30 Bytes). B-MAC basiert auf dem CSMA-Prinzip, allerdings wird vor dem Lauschen auf dem Medium eine zufällige Zeitspanne im Schlafmodus gewartet. Durch die Wartephase sollen Kollisionen vermieden werden, da es häufig

[3] http://sourceforge.net/projects/tinyos/

vorkommt, dass mehrer Knoten dasselbe Ereignis zur selben Zeit erkennen. Für weitere Ausführungen zu diesem Punkt sei auch auf meine Anwendungen I-Ausarbeitung [Davids (2006)] verwiesen.

3 Realisierung

Für die Realisierung der Gebäudeüberwachung innerhalb des HAW Rescue-Projekts ist es zum einen nötig ein Sensornetz aufzubauen und die einzelnen Sensorknoten zu programmieren, zum anderen soll eine Anwendung auf einem PC entwickelt werden, die die Schnittstelle zum Sensornetz bildet und die Kommunikation zu den anderen Projektteilen des HAW Rescue-Projekts, wie z. B. zum Leitstand, der von Andreas Piening bearbeitet wird, ermöglicht.

Die Realisierung eines drahtlosen Ad-hoc Sensornetzes zur Datenerfassung innerhalb eines Gebäudes, soll mit der in Abschnitt 2.1.1.4 vorgestellten TelosB-Mote erfolgen, die wie in 2.1.2 erörtert, meiner Meinung nach am Besten geeignet ist für den Aufbau des Sensornetzes. Mit Hilfe des Sensornetzes können die Temperaturwerte innerhalb des Gebäudes durch die Sensorknoten überwacht werden. Mit diesen Daten soll ein Brand erkannt und dem Leitstand gemeldet werden. In der Leitstelle können die Temperaturwerte dann mit ihrem Standort visualisiert werden, um dort eine bessere Übersicht über das Geschehen zu erhalten.

3.1 Programmierung der Sensorknoten

Insgesamt standen für einen Testaufbau fünf TelosB-Motes zur Verfügung, davon musste allerdings eine als Basisstation für die anderen dienen und die Schnittstelle zum PC bilden. Somit konnten die restlichen vier Motes dazu genutzt werden die Temperaturwerte zu erfassen.

Zur Temperaturmessung musste eine Anwendung entwickelt werden, die den eingebauten Temperatursensor der Motes in einem Intervall abfragt und die ermittelten Werte an die Basisstation sendet. Hierzu war es notwendig eine entsprechende Anwendung, in der in Abschnitt 2.2 vorgestellte Programmiersprache nesC, zu schreiben. Im Abbildung 3.1 ist der Aufbau der Anwendung aus dem einzelnen TinyOS-Komponenten skizziert.

Für die Basisstation kann eine der TinyOS Beispielanwendungen genutzt werden. Die Applikation *BaseStation* dient als Schnittstelle zwischen PC und Sensornetz. Diese Anwendung leitet ein Packet, dass sie über das Funknetzwerk empfängt, an die serielle Schnittstelle

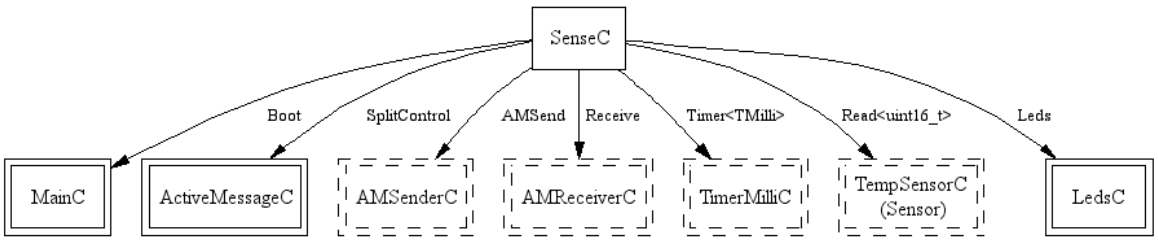

Abbildung 3.1: SenseC: TinyOS-Komponentenstruktur in der Temperaturmessanwendung

weiter. Ebenso werden Pakete, von der seriellen Schnittstelle an die Funkschnittstelle weitergeleitet. Damit kann mit dieser Anwendung die Schnittstelle Mote-zu-PC realisiert werden (Da die TelosB-Mote nur einen USB-Anschluss besitzt, kann mit Hilfe der Emulation einer seriellen Schnittstelle die Anwendung auch bei dieser Mote eingesetzt werden).

In Bezug auf die TinyOS Applikationsentwicklung lässt sich feststellen, dass durch die Tutorials und Beispielanwendungen eine Kommunikation zwischen den Motes und der Basisstation relativ schnell zu realisieren war. Leider erfordern aber Anwendungen, die über diese Beispiele hinausgehen, wesentlich mehr Einarbeitung in den TinyOS-Aufbau, mit einer teilweise unübersichtlichen Komponentenstrukur und Vererbungshierarchie. Hier wäre eine ausführlichere Dokumentation wünschenswert, die über die bereits genannten Tutorials hinausgeht.

3.2 Kommunikation zum Leitstand

Nachdem die Realisierung der Datenerfassung mit Hilfe des Sensornetzes im vorherigen Abschnitt beschrieben wurde, sollen diese Daten auch ausgewertet und genutzt werden. Da die Sensorwerte durch die BaseStation-Applikation bis an die serielle Schnittstelle des PC gelangen, ist daran anschließend eine Anwendung zu entwickeln, die die Werte dort abgreift. Danach sollen diese Werte lokal dargestellt und an den Leitstand weitergeleitet werden, an dem dann die eigentliche Auswertung vorgenommen werden soll und entsprechend reagiert werden kann. Der Leitstand im HAW Rescue-Projekt wird wie eingangs erwähnt, von Andreas Piening bearbeitet. Für weitere Details zur Realisierung sei auf seine Ausarbeitung zum Projekt verwiesen.

Die Kommunikation und der Datenaustausch zwischen den Teilbereichen der Projektteilnehmer soll mit Hilfe von Web Services durchgeführt werden. Im HAW Rescue-Projekt haben wir uns auf diesen Mechanismus geeinigt, da damit die größte Flexibilität erzielt werden kann. So ist eines der größten Vorteile von Web Services in Bezug auf das HAW Rescue-Projekt die

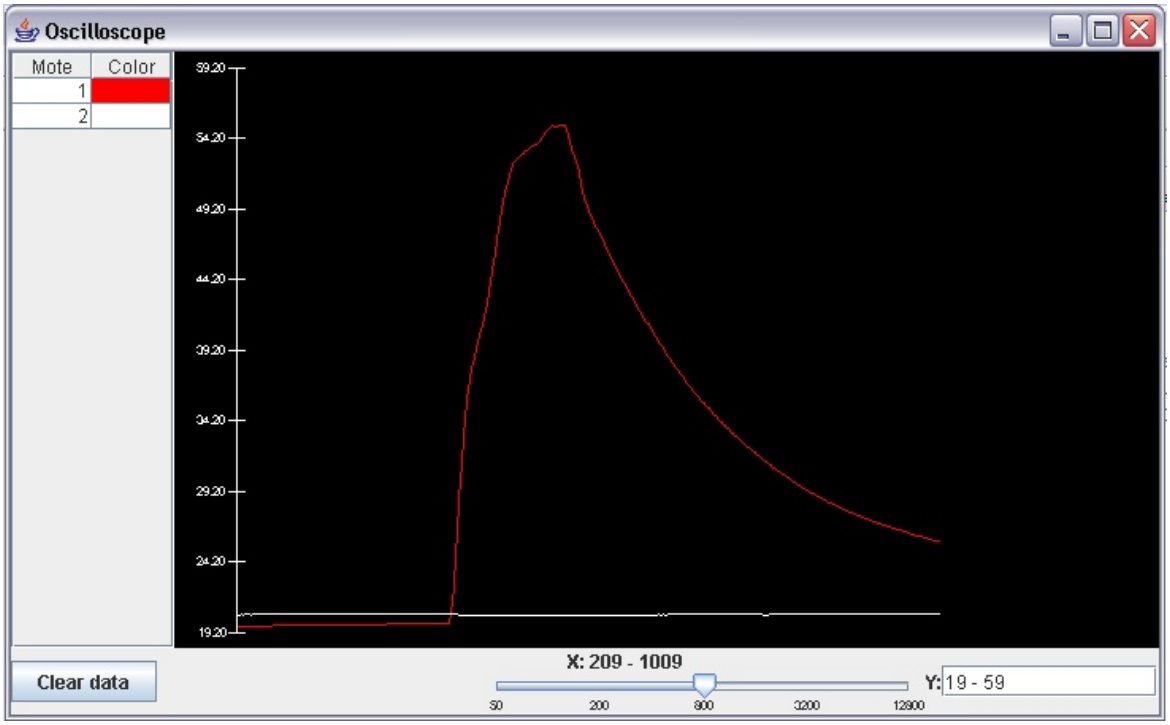

Abbildung 3.2: Darstellung des Temperaturverlaufs zweier Motes

Systemunabhängigkeit. Jeder Projektteilnehmer kann die für seine Ziele am Besten geeignete Programmiersprache und Systemumgebung einsetzen. Ein weiterer großer Vorteil ist die lose Koppelung. Die Kommunikation erfolgt über Nachrichtenaustausch mittels HTTP/XML. Web Service-Konsumenten oder -Anbieter brauchen sich um Implementierungsdetails nicht zu kümmern. Das hat zusätzlich den Vorteil, dass jeder im Projekt seine Anwendungen realisieren kann, ohne evtl. auf Teile eines anderen warten zu müssen. Weitere Vorteil von Web Services sind z. B. die Orts- und Protokolltransparenz.

Die lose Koppelung der Web Services bringt allerdings auch Nachteile mit sich. So ist in einem produktiven Einsatz des Rescue-Projekts beispielsweise sicherzustellen, dass keine Nachrichten, die zwischen Leitstand und dem Gebäudeüberwachungssystem ausgetauscht werden, verloren gehen. Auch Sicherheitsaspekte müssten in einem produktiven Einsatz unbedingt beachtet werden.

Für die Realisierung der lokalen Anwendung und die Kommunikation zum Leitstand sollte eine Java-Applikation entwickelt werden. Java ist deshalb meiner Meinung nach am Besten geeignet, da bereits einige Beispiele in dieser Programmiersprache im TinyOS-Paket vorhanden sind und deshalb aus Zeit- und Effizienzgründen Teile übernommen werden können.

Es wurde somit eine Anwendung geschrieben, die die reinen Byte-Werte der Mote-

Nachrichten interpretiert und darstellt. Dazu wurde eine Applikation erstellt, die die Sensordaten, mit der dazu gehörigen ID des Knotens, lokal in einer graphischen Darstellung visualisiert (siehe Abbildung 3.2). Ebenso wurde eine Struktur zu Datenhaltung implementiert. Hier soll später eine Datenbank angebunden werden, um darauf Auswertungen über die Daten zu ermöglichen. Die Sensorwerte können weiterhin durch Web Services vom Leitstand abgefragt werden. Dazu wurden Methoden entwickelt, die die IDs der aktiven Knoten bereitstellen, die die Möglichkeit bieten die aktuellen Temperaturwerte abzurufen und die die Positionen der Knoten im Gebäude liefern. Des Weiteren wurde ein Location-Server von mir entwickelt, über den die Abfrage und Zuordnung der Knotenposition der Sensorknoten innerhalb den einzelnen Räumen des Gebäudes möglich ist.

4 Fazit

Abschließend lässt sich feststellen, dass die für das Projekt geplanten Teile des HAW Rescue-Projekts realisiert werden konnten. Es wurde ein Sensornetz aufgebaut und es konnten damit die Temperaturen im einem Gebäude gemessen werden. Diese Daten wurden lokal in einer Anwendung ausgewertet und auch dem Leitstand zur Abfrage zur Verfügung gestellt. Im Leitstand, von Andreas Piening realisiert, konnten die Temperaturwerte mit ihrer zugehörigen Position angezeigt werden.

Aufbauend auf diesen Schritten kann damit begonnen werden die Ziele zu realisieren, die ich für die Masterarbeit definiert habe (vgl. [Davids (2007a)]). Beispielsweise kann versucht werden, die Position mobiler Knoten mit Hilfe der Nachrichtenempfangsstärke zu bestimmen.

4.1 Ausblick

Es ist erkennbar, dass ein drahtloses Sensornetz im HAW Rescue-Projekt viele Vorteile mit sich bringt und sehr gut für diesen Einsatzzweck geeignet ist, wie bereits in [Davids (2007a)] erläutert. Mit einem Sensornetz kann ein vielfältiges Einsatzspektrum abgedeckt werden, dass sich nicht nur auf die Gebäudeüberwachung erstreckt. So sind weitere denkbare Anwendungsmöglichkeiten:

- Umwelt- und Verhaltensforschung, wie Überwachung von seismische Aktivitäten oder Tierverhalten
- Gesundheitssektor, z. B. Monitoring von Patienten
- Infrastrukurüberwachung, wie Stabilitätskontrolle von Brücken oder Hochhäusern
- Der Militärbereich bietet vielfältige Anwendungsmöglichkeiten, beispielsweise Überwachung oder Objekt-Identifikation

Diese Auflistung zeigt einige der vielfältigen Einsatzgebiete eines Sensornetzes. Es wird deutlich, dass dieses Thema ein breites und interessantes Betätigungsfeld ist. Es werden in Zukunft, im Hinblick auf kleinere Strukturen, höhere Rechenleistung oder die Entwicklung neuer Materialien, sicherlich noch viele Fortschritte auf dem Gebiet der Sensornetze zu erwarten sein.

Literaturverzeichnis

[Beutel u. a. 2007] BEUTEL, J. ; BLUM, P. ; DYER, M. ; MOSER, C. ; YÜCE, M. ; STADELMANN, P.: *BTnode Programming - An Introduction to BTnut Applications*. Revision 1.5. ETH Zürich Computer Engineering and Networks Laboratory (Veranst.), 2007. – URL http://www.btnode.ethz.ch/pub/uploads/Documentation/btnodetutorial_1.5.pdf

[BTnode 2007] BTNODE: *ETH Zürich: BTnodes - A Distributed Environment for Prototyping Ad Hoc Networks*. 2007. – URL http://www.btnode.ethz.ch/. – Zugriffsdatum: 25.02.2007

[Davids 2006] DAVIDS, A.: *RESCUE: Sensorik zur Gebäudeüberwachung*. Hochschule für Angewandte Wissenschaften Hamburg. 2006. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2006/davids/abstract.pdf

[Davids 2007a] DAVIDS, A.: *Ad-hoc Sensornetzwerk zur Gebäudeüberwachung und Navigationsunterstützung*. Hochschule für Angewandte Wissenschaften Hamburg. 2007. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master06-07/davids/report.pdf

[Davids 2007b] DAVIDS, A.: *Vergleich von Projekten im Bereich Sensornetze unter Berücksichtigung des Rescue-Umfelds*. Hochschule für Angewandte Wissenschaften Hamburg. 2007. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master06-07-aw/davids/report.pdf

[Gay u. a. 2003a] GAY, D. ; LEVIS, P. ; BEHREN, R. von ; WELSH, M. ; BREWER, E. ; CULLER, D.: The nesC language: A holistic approach to networked embedded systems. In: *PLDI '03: Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation*. New York, NY, USA : ACM Press, 2003, S. 1–11

[Gay u. a. 2005] GAY, D. ; LEVIS, P. ; CULLER, D.: Software design patterns for TinyOS. In: *LCTES '05: Proceedings of the 2005 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems*. New York, NY, USA : ACM Press, 2005, S. 40–49

[Gay u. a. 2003b] GAY, D. ; LEVIS, P. ; CULLER, D. ; BREWER, E.: *nesC 1.1 Language Reference Manual*, 2003. – URL http://nescc.sourceforge.net/papers/nesc-ref.pdf

[Hill u. a. 2000] HILL, J. ; SZEWCZYK, R. ; WOO, A. ; HOLLAR, S. ; CULLER, D. ; PISTER, K.: System architecture directions for networked sensors. In: *ASPLOS-IX: Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*. New York, NY, USA : ACM Press, 2000, S. 93–104

[Hinck 2006] HINCK, S.: *RESCUE: Wearable Computer in Desaster-Szenarien*. Hochschule für Angewandte Wissenschaften Hamburg. 2006. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2006/hinck/abstract.pdf

[Hinck 2007] HINCK, S.: *Einsatz von Wearable Computing in Desaster Szenarien*. Hochschule für Angewandte Wissenschaften Hamburg. 2007. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master06-07/hinck/report.pdf

[Karl und Willig 2005] KARL, H. ; WILLIG, A.: *Protocols and Architectures for Wireless Sensor Networks*. John Wiley & Sons, 2005

[Levis u. a. 2004] LEVIS, P. ; MADDEN, S. ; GAY, D. ; POLASTRE, J. ; SZEWCZYK, R. ; WOO, A. ; BREWER, E. ; CULLER, D.: An adaptive energy-efficient MAC protocol for wireless sensor networks. In: *Proceedings of the First Symposium on Networked Systems Design and Implementation*, USENIX Association, 2004, S. 1–14

[Mattern und Römer 2003] MATTERN, F. ; RÖMER, K.: Drahtlose Sensornetze. In: *Informatik-Spektrum* Vol. 36 (2003), Nr. 3, S. 191–194

[Piening 2006] PIENING, A.: *RESCUE: Leitstand für Desaster-Szenarien*. Hochschule für Angewandte Wissenschaften Hamburg. 2006. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2006/piening/abstract.pdf

[Piening 2007] PIENING, A.: *RESCUE Leitstand für Desaster-Szenarien*. Hochschule für Angewandte Wissenschaften Hamburg. 2007. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master06-07/hinck/report.pdf

[Polastre u. a. 2004a] POLASTRE, J. ; HILL, J. ; CULLER, D.: Versatile low power media access for wireless sensor networks. In: *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA : ACM Press, 2004, S. 95–107

[Polastre u. a. 2004b] POLASTRE, J. ; SZEWCZYK, R. ; SHARP, C. ; CULLER, D.: *The Mote*

Revolution: Low Power Wireless Sensor Network Devices. Proceedings of Hot Chips 16: A Symposium on High Performance Chips. 2004

[ScatterWeb 2007] SCATTERWEB: *ScatterWeb*. 2007. – URL http://scatterweb.mi.fu-berlin.de/. – Zugriffsdatum: 24.02.2007

[Sun 2007] SUN: *Sun SPOT*. 2007. – URL http://www.sunspotworld.com/. – Zugriffsdatum: 25.02.2007

[Woo und Culler 2001] WOO, A. ; CULLER, D.: A transmission control scheme for media access in sensor networks. In: *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*. New York, NY, USA : ACM Press, 2001, S. 221–235

[Zhao und Guibas 2004] ZHAO, F. ; GUIBAS, L: *Wireless Sensor Networks: An Information Processing Approach*. Morgan Kaufmann, 2004