innovations
for high
performance
microelectronics

Abschlußbericht für das Verbundprojekt FeuerWhere

Zuwendungsempfänger:	**Förderkennzeichen:**
IHP Microelectronics	**01BN0710F**

Vorhabensbezeichnung:
FeuerWhere - „TriCordermed: Eine Middleware-Plattform zur netzbasierten Überwachung von Vitalparametern“

Laufzeit des Vorhabens:
01.01.2008 bis 31.12.2010

Berichtszeitraum:
01.01.2008 bis 31.12.2010

Autor : Krzysztof Piotrowski, Anna Sojka, Peter Langendörfer

Version : 1.1
Datum : 28.06.2011

Contents

I. Kurze Darstellung

1 Aufgabenstellung

Ziel des Projektes FeuerWhere ist die Lokalisierung von Feuerwehreinsatzkräften innerhalb von Gebäuden unter Einsatzbedingungen. Dabei wird eine fortlaufende Verfolgung der Position mittels eines redundanten Systems angestrebt. Darüber hinaus sollen Vital-, Geräte- und Umweltdaten direkt an den Einsatzkräften oder in deren Umfeld gemessen und erfasst werden. Diese Daten sollen dann nach vorheriger Auswertung mit einem robusten Übertragungsverfahren zu einer zentralen Station außerhalb des Gebäudes transportiert werden. Dort sollen die Daten dem Einsatzleiter in grafischer Form präsentiert werden.
Das FeuerWhere-System dient zur Sicherheitsüberwachung von Feuerwehreinsatzkräften im Einsatz. Es soll bei der Brandbekämpfung und der technischen Hilfeleistung unterstützen. Dabei soll es insbesondere an unübersichtlichen Einsatzstellen, bei denen kein Blickkontakt zu den Einsatzkräften besteht, wie z.B. in stark verrauchten Gebäuden oder Anlagen, Verwendung finden.
Das IHP hat sich auf die Erfassung und auf die sichere und robuste Übertragung von Vital-, Geräte- und Umweltparametern konzentriert. Hierbei wurde eine neue Hardware- Plattform wie auch die Software dafür entwickelt. Eine Middleware-Schicht, die für das Speichern und Übertragen von Daten verantwortlich ist, wurde ebenfalls entwickelt und untersucht. Des Weiteren hat sich das IHP auf Sicherheitsmechanismen für die übertragenen Daten konzentriert. Alle entwickelten Komponenten wurden in den Demonstrator integriert.

2 Voraussetzungen

Das IHP hat im Rahmen des folgenden Projektes an körpernahen Sensor Netzwerken gearbeitet:

- Basuma (BMWA).

Middleware-Plattformen, insbesondere für kontextsensitive Dienste, wurden in den folgenden Projekten entwickelt:

- Mobile Internet Business (BMBF),
- Wireless Internet Zellular (BMBF),
- Wireless Internet Ad Hoc (BMBF).

Sicherheitsaspekte in kontextsensitiven Plattformen und Hardwarebeschleuniger für kryptographische Verfahren untersucht das IHP in den Projekten:

- Mobile Internet Business (BMBF),
- UbiSecSens (EU 6th Framework).

Sensorplattformen wurden in folgenden Projekten entwickelt:

- Trusted Sensor Node (BSI),
- Tandem (BMBF).

3 Planung und Ablauf des Vorhabens

Der zeitliche und inhaltliche Ablauf des Vorhabens verlief insgesamt im Rahmen der ursprünglichen Planungen. Eine größere Abweichung fand in der Realisierung der Hardware des Body Area Networks (BAN) statt. Im Gegensatz zur ursprünglichen Planung wurde nicht

unmittelbar nach Projektstart mit der Realisierung der Sensorboards begonnen. Dies war in der Tatsache begründet, dass die Art der zu erfassenden Daten zunächst im Rahmen der Spezifikationsphase mit den anderen Partnern, speziell mit der Feuerwehr Berlin, bestimmt werden musste. Stattdessen wurden insbesondere die Arbeiten im Bereich Middleware-Plattform TriCorder frühzeitiger begonnen. Die Herstellung von Sensorplatinen erfolgte später als geplant. Für diese Hardware mussten die Treiber implementiert werden.

4 Wissenschaftlicher und technischer Stand

Hardware
Für Forschung und vereinfachte Entwicklung und Debugging wurden einige aus fertigen Komponenten gebaute Sensorknoten vorgeschlagen. Ein Vergleich von den Plattformen wurde in [1] vorgestellt. In dem Artikel werden die populärsten Hardware-Lösungen verglichen, zum Beispiel TmoteSky von Sentilla [2] (jetzt erhältlich als TelosB [3] von MEMSIC [4]), MICA2 [5], MICAz [6], IRIS [7], und von Crossbow (jetzt MEMSIC). Die Sensorknoten können in Abhängigkeit der Prozessor-Architektur in drei Gruppen aufgeteilt werden: Entweder werden sie mit einem 8-bit Mikrokontroller (z.B. von der AVR Familie [8] von Atmel), einem 16-bit Mikrokontroller (z.B. MSP430 Familie [9, 10, 11] von Texas Instruments), oder mit einem 32-bit Mikrokontroller (basierend auf ARM [12]) ausgestattet. Die Architektur bestimmt die verfügbare rechenintensive Energie von der Plattform und spezifiziert auch den Energieverbrauch vom Prozessor. Das andere Kriterium zur Klassifizierung von Sensorknoten-Plattformen ist das verwendete Funkmodul. Alle obengenannten Plattformen benutzen die Radio Module von Chipcon (jetzt Texas Instruments): CC1000 [13] und seine zwei Nachfolger CC1101 [14] und CC2500 [15], CC2420 [16] – ein IEEE 802.15.4 und ZigBee konformer Transceiver und sein Nachfolger CC2520 [17].
Ein Nachteil von den allen auf dem Markt verfügbaren Hardware-Plattformen ist, dass sie jeweils nur ein Funkmodul verwenden. Damit werden eventuelle Untersuchungen im Bereich der Radioredundanz zur Erhöhung der Robustheit der Datenübertragung ausgeschlossen.

Betriebssysteme
Die Betriebssysteme für drahtlose Sensornetze sind eingeschränkt wegen der spezifischen Eigenschaften von drahtlosen Sensorknoten. Das Betriebssystem für solche Geräte muss folgende Aspekte berücksichtigen: die Begrenzung von Kodegröße, das Energie-, Speicher-, und Peripheriemanagement, die Nebenläufigkeit der Mechanismen, das Steuerprogramm, das Multitasking und die Robustheit.
TinyOS [18] ist ein ereignisorientierter Programmierrahmen für eingebettete Systeme. Es ist ein Set von unabhängigen Komponenten, die es ermöglichen ein anwenderspezifisches Betriebssystem für jede Applikation zu bauen. Die Komponenten wurden in nesC, einem Dialekt der Programmiersprache C mit sehr kleinem Speicherbedarf, implementiert. TinyOS .. bietet auch Unterstützung bezüglich möglicher Energieeinsparungen. Der Prozessor und alle Peripherien können in einen Schlafmodus versetzt werden wenn es keine Aufgaben zu realisieren gibt. Aufgrund von Optimierungen zwischen den Komponenten und der Komplett-Programm-Kompilation produziert TinyOS normalerweise kleineren und schnelleren Kode als pure-C-Ansätze.
Contiki [19] ist in der Programmiersprache C implementiert. Contiki bietet die Möglichkeit die Applikationen zur Laufzeit zu laden. Das ganze System muss also in dem Fall nicht als Einzelbinärdatei hochgeladen werden. Contiki ist ein Hybrid, d.h. eine Kombination aus einen ereignisorientierten Kernel als Basis des Systems und einer präemptiven multi-threading Bibliothek, die optional beigefügt werden kann. Das Contiki-System besteht aus dem Kern und den Prozessen. Der Kern wird kompiliert und auf dem Gerät vor dem Einsatz installiert. Ein

Prozess ist eine Applikation oder ein Service. Die generierten Binärdateien sind größer als vergleichbare TinyOS-Applikationen.
Reflex [20] ist ein Betriebssystem, welches die elementare Abstraktion für das event-flow Model implementiert. Die Implementierung erfolgt dabei in drei Schichten. Die niedrigste Schicht - der Kern - besteht aus den Interrupt-Händler-Routinen, den Energiemanagementroutinen und dem Steuerprogramm. Die mittlere Schicht bietet die event-flow Mechanismen, die durch die dritte Schicht verwendet werden. Die dritte Schicht besteht aus den Treibern. Reflex wurde in der Programmiersprache C++ implementiert. Die Applikation in Reflex wird aus Aktivitäten gebaut, die C++ Objekte sind und durch das Steuerprogramm gesteuert werden. In Reflex wurden verschiedene Steuerungsverfahren implementiert: first come - first serve scheduling, fixed priority scheduling, earliest deadline first scheduling und time triggered scheduling.

Verteilter gemeinsamer Speicher
TeenyLIME [21, 22, 23] Middleware ist eine Implementierung der LINDA Abstraktion und wurde für drahtlose Sensornetze entwickelt. TeenyLIME konzentriert sich auf folgende Herausforderungen:

- Örtliche Berechnungen - die Berechnungen werden so dicht bei der Messquelle durchgeführt wie möglich.
- Mehrere Tasks laufen parallel und benutzen die gemeinsamen Daten.
- Zustandsbehaftete Koordination - mittels gemeinsames Stück von Daten als Zustand
- Reaktive Interaktion – Reaktion auf externe Bedingungen.

In TeenyLIME besitzt jeder Sensorknoten einen Tuple Space, den er mit anderen Knoten, die in einer ein-hop Nachbarschaft liegen, teilt. TeenyLIME wurde in TinyOS implementiert. Die Middleware wurde entwickelt, um eine einfache Erweiterung und Anpassung von Middleware zu ermöglichen. Die lokalen und verteilten Prozesse und auch die Kommunikation sind komplett entkoppelt. Deshalb kann jeder Prozess ohne Einfluss auf andere Prozesse modifiziert werden.
MacroLab [24] ist ein datenzentrisches System für drahtlose Sensornetze. Die Applikation wird vom Entwickler als Einzelprogramm für das ganze Netzwerk geschrieben. Das System zerlegt das Programm in mehrere Mikroprogramme, mit denen die einzelnen Knoten programmiert werden. Die Daten in dem System werden als Vektor repräsentiert, wobei die erste Dimension das Set von Sensorknoten-Identitäten beinhaltet. Der Datensatz von jedem Knoten ist global für alle anderen Knoten in einem Netzwerk. Es gibt verschiedene Möglichkeiten um die im Vektor gespeicherten Daten über das Netzwerk zu verteilen. Der Vektor kann entweder auf einem zentralen Knoten gespeichert werden, oder jeder Knoten speichert nur eigene Daten aus dem Vektor, oder jeder Knoten speichert den ganzen Vektor. Die Vektoren können addiert und subtrahiert werden. Es können aber auch andere Vektoroperationen verwendet werden.
Unserer Meinung nach sind die beiden Ansätze ziemlich begrenzt. Die Abstraktionen haben eine sehr hohe Ebene erreicht, was die Anwendung komplexer macht und auch die Kontrolle über den tatsächlichen Ablauf von dem Engineer der Applikation verhindert.

5 Zusammenarbeit mit anderen Stellen

Das IHP hat im Rahmen des Projektes mit der Berliner Feuerwehr, der MSA Auer GmbH und der FU Berlin direkt zusammengearbeitet. Die Zusammenarbeit verlief erwartungsgemäß, produktiv und reibungsfrei. Die Berliner Feuerwehr hat als Experte sowohl bei der Systemspezifikation als auch bei dem Aufbau des Demonstrators wichtige Hinweise gegeben und die Anforderungen spezifiziert. Die robuste Übertragung von Messdaten zum Leitstand erfolgte dank der engen Zusammenarbeit zwischen IHP, MSA Auer und FU Berlin erfolgreich.

Die zahlreichen Projekttreffen und Statusmeetings haben zur schnellen Lösung entstandener Problemen geholfen.

II. Eingehende Darstellung

1 Verwendung der Zuwendung

Die Zuwendung wurde konform zur Antragstellung verwendet. Alle Arbeitspakete des IHP sowie der Unterauftrag der Firma Timekontor wurden in vollem Umfang bearbeitet. Alle anwendungsspezifischen Teillösungen wurden prototypisch umgesetzt. Dies gilt insbesondere auch für die Teilaufgaben, für die während des Projektverlaufes Änderungen in der Zielsetzung notwendig wurden. Für die Präsentation der Ergebnisse wurde ein FeuerWhere Demonstrator realisiert, in dem alle angedachten Dienste mit Hilfe der im Projekt entwickelten Soft- und Hardware demonstriert werden können.

2 Erzielte Ergebnisse

Die hier präsentierten Ergebnisse werden in die Arbeitspackete unterteilt.

2.1 AP1 Systemspezifikation

Zusammen mit den anderen Projektpartnern wurden die Aufgaben des Systems und die dazu notwendigen Eigenschaften definiert. Außerdem wurde die Architektur des gesamten Systems entwickelt und die Bedingungen der Zusammenarbeit der Komponenten definiert. Der Informationsfluss und die funktionelle Verkettung im System erfordern bestimmte Interaktionen zwischen den einzelnen Komponenten.

Abbildung 1 zeigt den Aufbau des Gesamtsystems. Aus unserer Sicht ist das Body Area Netzwerk (BAN) der Kern. In der Spezifikation wurden die BAN-Schnittstellen und der Datenaustausch definiert. Die Daten die im BAN generiert wurden und an die Leitstelle gesendet werden müssen, werden im Person Tracking and data Transport Network (PTTN) selbstständig zur Senke (Leitstelle) weitergeleitet. Diese Lösung separiert die Aufgaben. Einerseits kümmert sich das BAN nicht um die Probleme, die auf dem Weg zu Leitstelle passieren können, andererseits sind die BAN-Pakete für das PTTN nur Sätze von Bytes, die an eine bestimmte PTT Unit (PTTU) geliefert werden müssen. Die Lösung ermöglicht auch die Gewährleistung einer Ende-zu-Ende Sicherheit, die vom BAN bis zur Leitstelle reicht, z.B. durch eine Verschlüsselung von BAN-Paketen ohne Involvierung des PTTNs.

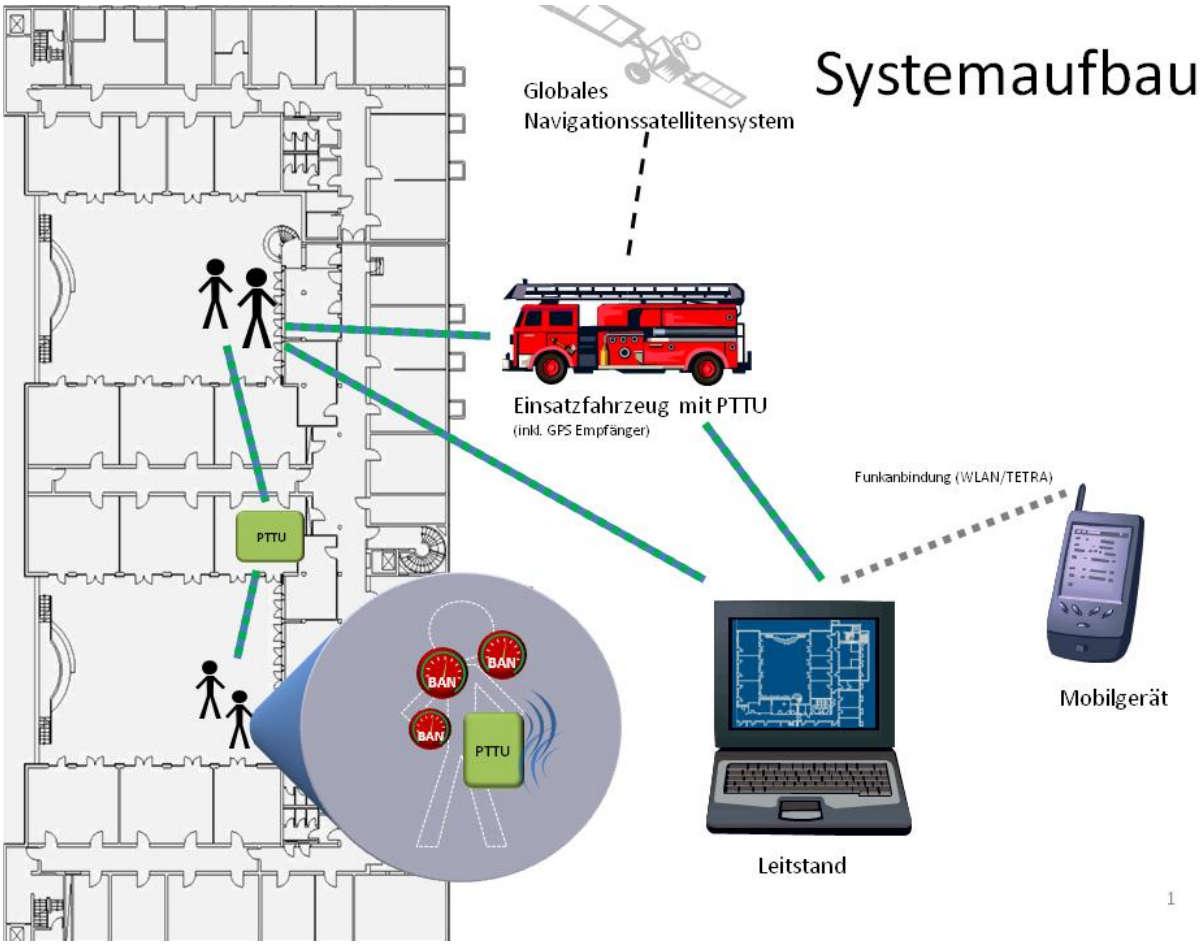

Abbildung 1: FeuerWhere Systemaufbau

Es wurden die folgenden Anforderungen bezüglich BAN formuliert:

1. Es muss drei Prioritäten für die Datenübertragung geben:
   - 1 = rot (Notfall),
   - 2 = gelb (Warnung),
   - 3 = grün (Status).

2. Für die Priorität rot soll die Übertragungsverzögerung maximal 3 Sekunden betragen.
3. Für die Priorität gelb soll die Übertragungsverzögerung maximal 10 Sekunden betragen.
4. Alle 60 Sekunden muss ein Keep-alive Signal mit Priorität grün übermittelt werden.
5. Das Verlassen eines Alarmzustandes (gelb/rot) muss am Leitstand signalisiert werden.
6. Ein Notfall tritt ein, wenn ein oder mehrere einstellbare Schwellwerte für ein gewisses Zeitintervall überschritten werden.
7. Falls eine Systemkomponente (PTTU oder BAN) ausfällt, muss eine Fehlermeldung mit gelber Priorität angezeigt werden.
8. Für die Datenübertragung sollen weltweite lizenzfreie Frequenzen verwendet werden.

9. Das Body-Area-Network ist ein drahtloses Sensornetz welches der Erfassung und Übertragung von Sensordaten aus der unmittelbaren Umgebung einer Einsatzkraft dient.
10. Bei der Realisierung ist eine andere Trägerfrequenz als die von den NanoLOC Transceivern der PTTU verwendete zu wählen.
11. Die Messdaten werden an die PTTU übertragen.
12. Die an die PTTU verschickten Daten enthalten eine eindeutige ID, die einen Rückschluss auf die Einsatzkraft zulässt, in deren Umgebung die Daten erhoben wurden.
13. Profile für allgemeine Schwellwerte der Sensorik müssen hinterlegt werden können. Die Überwachung von absoluten Werten und Gradienten muss pro Sensor möglich sein.
14. Die Schwellwerte müssen für eine Warnstufe und eine Alarmstufe hinterlegt werden können.
15. Folgende Sensoren sollen in das BAN eingebunden werden: Herzfrequenz, Sauerstoffsättigung in Blut, Körperkerntemperatur, Temperatur im Anzug, Beschleunigungssensor, Panikschalter, Temperatur und Luftfeuchtigkeit, Infrarotstrahlung, Pyrolyseprodukte, Explosive Atmosphäre, Druck Atemluftflasche bzw. Restdauer, Ladestand der Batterie für BAN und PTTU.
16. Die Ausgestaltung der einzelnen Sensoren soll so erfolgen, dass eine Behinderung der Einsatzkräfte im Einsatz ausgeschlossen ist. Dies heißt insbesondere, dass die Sensoren überwiegend berührungslos arbeiten sollen und in die Kleidung oder Geräte zu integrieren sind. Ein separates Anlegen von Sensorik ist zu zeitaufwändig und daher nicht möglich.
17. Die Aktivierung des gesamten Body-Area-Networks findet entweder automatisch oder durch maximal eine aktive Handlung der betreffenden Einsatzkraft statt.
18. Die Deaktivierung des gesamten Body-Area-Networks findet entweder automatisch oder durch maximal eine aktive Handlung der betreffenden Einsatzkraft statt.
19. Eine versehentliche Auslösung der Deaktivierung ist zu verhindern.
20. Die Kommunikation muss über den vollen Temperaturbereich sichergestellt werden, insbesondere wenn ein Kommunikationspartner sich im unteren Temperaturgrenzbereich befindet und der andere im oberen Grenzbereich. Das Gehäuse muss die zuverlässige Arbeit von einem Sensorknoten in einem Temperaturintervall von -30°C bis +70°C und 100% Luftfeuchtigkeit gewährleisten. Das Gehäuse muss zudem 10 Sekunden unter einer 800°C heißen Flamme überdauern. EMV-Grenzwerte sind durch Tests zu ermitteln. Eine möglichst gute EMV-Konformität ist anzustreben.
21. Als Gehäuseschutzklasse ist IP67 definiert.
22. Das Gewicht des Gesamtsystems (PTTU + BAN) inkl. Sensorik und Stromversorgung sollte 500 g nicht überschreiten.
23. Die Lebensdauer der Stromversorgung der in die Kleidung integrierten Sensoren beträgt bei Standby 2 bis 3 Monate. Die maximale Betriebsdauer im Einsatz muss mindestens 30 Stunden betragen.
24. Als Feedback an den Träger ist ein mindestens zweistufiges Signal vorzusehen. Eines davon ist ein Rückzugsignal.

25. Die Auslösung des Rückzugsignals kann dabei sowohl von der PTTU als auch vom BAN oder manuell vom Leitstand erfolgen. Manuelle Signale können sowohl individuell an einzelne Einsatzkräfte als auch an alle im Einsatz befindlichen Kräfte übermittelt werden.

2.2 AP2 Erforschung und Aufbau des physikalischen Grundnetzes

In dem System existieren zwei Netze: das PTTN- und das BAN-Netz. Beide Netze ergänzen sich dabei wie folgt: Ohne BAN würden nur unvollständige Informationen im PTTN-Netz zur Verfügung stehen. Ohne PTTN gibt es keine Möglichkeit für die im BAN gesammelten Informationen das Netz zu verlassen und zuverlässig an die Leitstelle übermittelt zu werden. Diese Aufteilung erlaubt einerseits einen Abbau der Komplexität in den einzelnen Netzen, anderseits aber erfordert sie eine saubere und klare Trennung der Funktionen und folglich eine Definition der passenden Schnittstellen.

2.3 AP3 körpernahes Funknetzwerk (TriCordermed-Plattform)

AP3.1 Aufbau geeigneter Sensorknoten aus „Components of the Shelf“

Die Herstellung der Prototyp-Platine des BAN-Knotens (Abbildung 2 und Abbildung 3) wurde etwas später als geplant erfolgreich abgeschlossen. Bevor die Hardware verfügbar war, haben wir bereits angefangen die plattformspezifischen Treiber für das ausgewählte Contiki-Betriebssystem zu implementieren. Da der Mikrokontroller erst seit einigen Monaten auf dem Markt war und die ganze Plattform eine neue Zusammenstellung von Komponenten war, gab es dafür noch keine Treiber und es musste fast alles neu programmiert werden. Abgesehen von den Treibern mussten noch zusätzliche Werkzeugapplikationen erstellt werden, um das Programmieren der Hardware unter Linux zu erlauben und zu vereinfachen.

Der Knoten verfügt über einen MSP430F5438-Mikrokontroller der Firma Texas Instruments. Mit 16 KB RAM und 256 KB Flash-Speicher eröffnet er viele Möglichkeiten. Bis zum Projektende wurde schon eine verbesserte Version entwickelt. Im Vergleich zur Anfangsversion wurden andere externe Flash-Chips ausgewählt. Die letzte Version des Knotens hat nun zwei mal 2 MB (und nicht 8 MB) Flash für das Speichern von Daten zur Verfügung. Dafür benötigen die beiden Flash-Chips eine niedrigere Spannung als die Standard Flash-Chips (sie arbeiten schon ab 2.3V statt 2.7V). Sonst sind die BAN-Knoten so aufgebaut wie geplant. Sie verfügen über drei Radio-Transceiver, zwei davon sind Pin und Befehl kompatible Transceiver CC1101 und CC2500, die die folgenden zwei ISM-Bänder unterstützen (868 MHz und 2,4 GHz). Die Kombination der beiden Chips erlaubt einen problemlosen Umstieg von einer Frequenz auf die andere und unterstützt so auch eine robuste Kommunikation auf beiden Frequenzen. Da das 2,4 GHz Band durch nanoLoc blockiert ist, haben wir die Mechanismen für Radio redundante Kommunikation nur separat untersucht. Für spätere Ansätze verfügt der BAN-Knoten auch über einen ZigBeeTM-kompatiblen Transceiver-Chip – CC2520. Damit lässt sich eine standardkonforme Kommunikation mit anderen ZigBeeTM-Geräten realisieren. Es wäre auch möglich die BAN-Netzwerke über den ZigBee-Kanal miteinander kommunizieren zu lassen, z.B. um einen robusten Kommunikationskanal, redundant zur PTTN, zur erstellen. Der CC2520-Chip kommuniziert aber auch im 2,4-GHz-Band und kann deswegen Störungen für den nanoLoc-

Chip im PTTN-Netz verursachen. Aus diesem Grund konnte dieser Ansatz in dem Standard-Ansatz für das Projekt nicht verwendet werden. Die Größe der Platine wurde an das verfügbare und für den Demonstrator vorgesehene Gehäuse von MSA Auer angepasst.

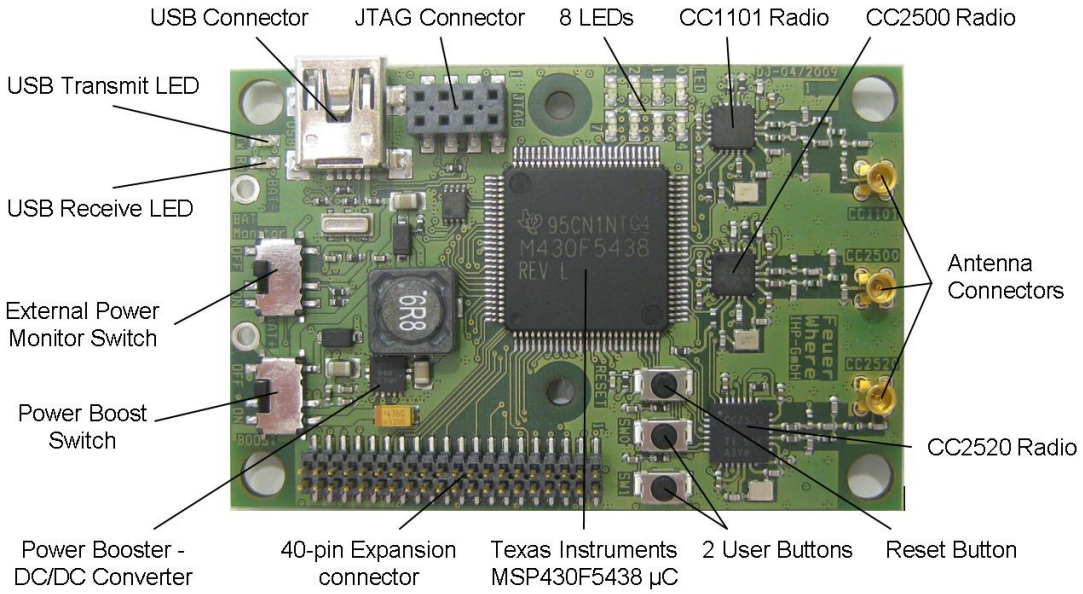

Abbildung 2: Die Vorderseite der Platine des BAN-Knotens

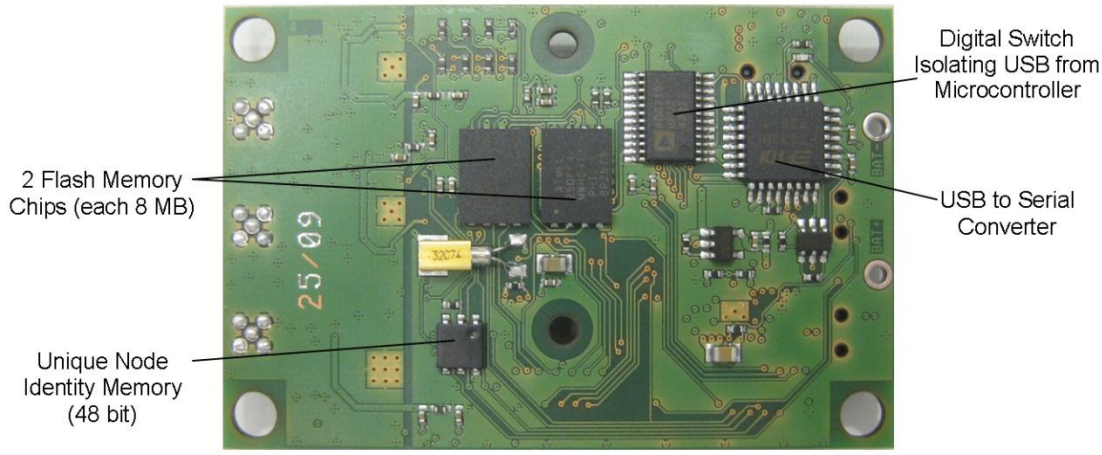

Abbildung 3: Die Rückseite der Platine des BAN-Knotens

Jeder Knoten verfügt über eine eindeutige 48-bit lange Identitätsnummer, die in einem Read Only Memory (ROM) Chip gespeichert ist. Zusätzlich verfügt der BAN-Knoten über mehrere Eingangs- und Ausgangsmöglichkeiten wie 8 LEDs, Analog/Digital-Wandler-Eingänge und digitale Schnittstellen (I2C, SPI, UART). Um die Ausnutzung der Batterien zu optimieren, ist der BAN-Knoten mit einem DC/DC-Spannungsregler ausgestattet. Dieser erlaubt es, den Knoten im Betrieb auch zu halten, wenn die Batteriespannung unter der normalerweise notwendigen Spannung ist, und ermöglicht so, die Lebensdauer des Knotens zu verlängern.

Es gibt zwei Möglichkeiten den Knoten zu programmieren. Aus der Sicht des Knotens kann man eine neue Version der Software entweder über die JTAG-Schnittstelle oder mit Hilfe des Boot Strap Loaders (BSL) hochladen. Aus Sicht des Anwenders und seiner Programmierungsumgebung schließen sich manche Möglichkeiten aus. Für die beiden Schnittstellen braucht man normalerweise ein Programmiergerät, das entweder an die USB- oder an die parallele (oder serielle) Schnittstelle des PCs angeschlossen wird. Für das Gerät braucht man auch einen Treiber für das PC-Betriebssystem, welches man gerade verwendet. Texas Instruments unterstützt die Microsoft-Betriebssysteme. Für Microsoft Windows gibt es also die Treiber für die neusten USB-Geräte und auch die neuste Tools-Kette, die die neusten Mikrokontroller aus der MSP430-Familie unterstützen. Die Entscheidung im FeuerWhere-Projekt Contiki für die Knoten zu verwenden, erzwingt in Linux als Betriebssystem auf dem Hostrechner.

Beim Linux-Betriebssystem mussten folgende Probleme gelöst werden: die Compilerunterstützung für den neuen Mikrokontroller und das Bereitstellen einer Möglichkeit, ein kompiliertes Programm auf den Knoten hochzuladen. Diese beiden Aufgaben haben wir erfolgreich abgeschlossen.

Um den kompilierten Code zu testen, brauchten wir eine Möglichkeit, den Code auf den Mikrokontroller hochzuladen. Beim Design der Knotenplatine haben wir uns entschieden, ein vereinfachtes BSL-Programmiergerät auf der Platine zu integrieren. Dies reduziert die Kosten bei der Softwareentwicklung und macht diese auch insgesamt einfacher. Der USB-zu-Serial-Konverter Chip erlaubt mittels des neu entwickelten BSL-Treibers die Programmierung des Knotens. Wir haben uns für BSL entschieden, weil das BSL-Protokoll im Vergleich zu JTAG einfacher ist. Es liefert zwar weniger Debug-Möglichkeiten, ist aber für das Hochladen eines Codes genau das Richtige.

Der frei verfügbare mspgcc ist eine Portierung vom gcc Kompilator/Linker für die MSP430 Mikrokontroller-Plattform. Dieser unterstützt allerdings nicht den MSP430F5438. Deswegen haben wir uns entschieden die erste Version der Software unter Windows zu entwickeln, wo die Code-Composer-Studio-Werkzeugkette auch fehlerfrei funktioniert. Um die Entwicklung unter Code Composer Studio zur ermöglichen, war es notwendig ein eigenes Betriebssystem zu implementieren, welches die Grunddienste wie z.B. Timer oder Task Scheduling liefert.

Es wurden einige Designfehler und optimierungsbedürftige Stellen in der ersten Version der Platine entdeckt. Als Beispiel wurde ein falscher Pin des Mikrokontrollers an die Signalleitung, die den BootStrap Loader (BSL) initialisieren soll, angeschlossen. Dies war bei den Tests des BSL äußerst zeitintensiv. Den Fehler kann man aber mit einer Überbrückung zwischen der falschen und der richtigen Leitung umgehen. Da der Fehler nur beim Programmieren eine Rolle spielt ist er nicht so kritisch, dennoch wurde er in der neuen Version der FeuerWhere-Knoten behoben.

AP3.2 BAN Vernetzung und Integration

Es ist sehr wichtig, dass die BAN-Knoten wissen, zu welchem BAN sie gehören und auch, dass die PTTU weiß mit welchem BAN sie arbeiten soll (die PTTU gehört somit eigentlich zum BAN).

Die zwei Probleme / Aufgaben beziehen sich auf die initiale Phase vor dem Feuerwehreinsatz. Es wurden Methoden zum BAN-Aufbau und Methoden zur BAN-PTTU-Verknüpfung vorgeschlagen und implementiert. Da Teile der Bekleidung wie Jacke, Hose und Helm personengebunden sind, muss die Anbindung von einzelnen BAN-Knoten zu einem spezifischen BAN nur während des Einfügens oder Austauschens von einzelnen Knoten erfolgen. Dagegen muss die Verknüpfung zwischen der PTTU-Einheit, die sich am Atemschutzgerät befinden soll, und dem BAN einer Person je Einsatz neu erfolgen und muss deswegen auch vor jedem Einsatz durchgeführt werden.

Die vorgeschlagenen Methoden zur BAN-PTTU-Verknüpfung nutzen RFID als das Medium für die Ermittlung der Identität der BAN-Knoten und der BAN. Ist die PTTU mit einem RFID-Lesegerät ausgestattet, kann sie die Identität des zugehörigen BANs einlesen.

Um die BAN-Knoten ihrem BAN-Netzwerk zuzuordnen, kann eine Hardware-Komponente (die z.B. zur PTTU baugleich ist) mittels RFID-Lesegerät die Identitäten von den einzelnen BAN-Knoten und der BAN-Identität einlesen und verbinden. Diese kombinierte Information kann das Lesegerät anschießend an jeden BAN-Knoten übermitteln. Die initiale BAN-Konfiguration kann somit erfolgen.

Basierend auf der BAN-Identität wird auch die Kanalnummer in dem BAN-Netzwerk konfiguriert. Dadurch werden die inter-BAN-Interferenzen reduziert und die theoretische Dichte (maximale Anzahl) der BAN-Netzwerke in dem System erhöht.

AP3.4 Managementfunktionalität und intelligente Auswertung der Vital-Parameter

In dem BAN-Netzwerk werden alle variablen oder konfigurierbaren Werte in einer Middleware, die die Abstraktion des gemeinsamen verteilten Speichers liefert, erfasst. Die Middleware setzt unterschiedliche Mechanismen ein um die Qualität des Speichers je nach Konfiguration zu gewährleisten. Die Konfiguration kann für einzelne Werte (Variablen) erfolgen.

Für die Applikation wurde ein Satz von Variablen definiert, die in jedem BAN auf den Knoten gespeichert werden. Der Satz von Variablen und deren Konfigurationen definieren das Verhalten des Kerns der Middleware. In den Variablen werden die Messwerte und auch die Konfigurationsparameter, wie Schwellwerte oder BAN-Identität, gespeichert. Dies ermöglicht z.B. eine erneute on-line Konfiguration von Schwellwerten für die verschiedenen Alarmstufen oder das Ändern von Sampling-Frequenzen.

Die Middleware ermöglicht es auch bestimmte Ereignisse zu detektieren. Dieser Mechanismus ist während der Übersetzung des Codes konfigurierbar. Logische Gleichungen, die auf den definierten Variablen basieren, werden dann definiert und schließlich zur Laufzeit ausgewertet, wenn sich die Werte der betroffenen Variablen ändern. Das Ergebnis der Auswertung ist der Zustand des Ereignisses.

Für die verwendeten Demonstratoren wurden einfache schwellwertbasierte Ereignis-Erkennungs-Mechanismen verwendet. Für jeden Parameter, der in dem System überwacht werden soll, wurden zwei Variablen für die Schwellwerte definiert. Dies bietet Flexibilität und

Konfigurierbarkeit. Komplexere Ereignisdefinitionen, die auf Schwellwerten und Gradienten basieren, wurden ebenfalls untersucht.

Die Variablen, die die Parameter speichern, werden explizit über alle Knoten im BAN verteilt. Dies ermöglicht allen Knoten die empfangenen Werte auszuwerten und demnach auch ihr Verhalten anzupassen, z.B. die Sampling-Frequenz erhöhen, um das Gesamtbild über die Situation zu verbessern.

AP3.5 Integration von Sicherheitsmechanismen in die Vitalsensoren

Es wurden Konzepte entwickelt, die den Einsatz von Sicherheitsmechanismen beinhalten, um den Zugang zu den Informationen aus dem BAN-Netzwerk für nicht autorisierte Personen zu verhindern. Dadurch wurden die Eigenschaften des Systems bezüglich Security und Privacy verstärkt.

Es wurden Algorithmen für leichtgewichtige kryptographische Operationen vorgeschlagen, die auf mathematische Richtigkeit geprüft wurden. Eine Reihe von Angriffssimulationen mit einer PC-Implementierung wurden durchgeführt. Eine Implementierung der Algorithmen für Sensorknoten wurde auf das TinyOS und das Contiki Betriebssysteme portiert.

AP3.6 BAN PTTU Gateway

Der Kern der Middleware ist in der Programmiersprache C implementiert. Zusammen mit einem Wrapper stellt die Middleware ein Softwaremodul für das Betriebssystem dar. Wir haben einen Wrapper für unser BANOS und auch für das FeuerWhere Betriebssystem, welches auf den PTTU-Knoten läuft, realisiert und getestet. Solange die Versionen auf unterschiedlichen Plattformen dieselbe Konfiguration teilen, können die heterogenen Systeme die definierten Variablen auch teilen.

Dadurch haben wir eine flexible Schnittstelle zwischen BAN und PTTN geschaffen. Alle Daten, die zwischen den beiden Subsystemen ausgetauscht werden, werden über die Middleware-Schicht übertragen.

2.4 AP6 Benutzerschnittstelle

AP6.2 Middleware

Wie bereits erläutert, stellt die Middleware die Abstraktion des gemeinsamen verteilten Speichers zur Verfügung. Zudem verfügt sie über einen Mechanismus zu Ereignisdetektion. Beide Funktionalitäten sind zur Kompileszeit konfigurierbar. Der Kern der Middleware ist Betriebssystem unabhängig. Es ist in der C Programmiersprache implementiert und braucht eine Adaptionsschicht (Wrapper) um in einem bestimmten Betriebssystem eingesetzt werden zu können. Diese Lösung fördert die Heterogenität und Flexibilität des Ansatzes.

Um eine reibungslose Zusammenarbeit aller Knoten, die ein Teil der Applikation sind, zu gewährleisten, muss eine Liste von Variablen, die durch die Middleware verwaltet werden, erstellt und allen Knoten zu Verfügung gestellt werden. Diese Liste stellt den Adressraum der verteilten Daten in einer Applikation dar. Zudem definiert sie die Schnittstelle zur Middleware. Der Zugang zu den gespeicherten Daten aus der Applikationssicht ist transparent.

Die Liste der Variablen kann entweder statisch definiert oder auch mit Hilfe von Tools an eine bestimmte Instanz des FeuerWhere-Systems angepasst werden.

Bei der FeuerWhere-Applikation beinhaltet diese Liste die Variablen für alle im BAN gemessenen Parameter, für die Zustände aller definierten Ereignisse und für die Werte aller Konfigurationsparameter.

Die Liste von verteilten Variablen definiert auch das gewünschte Verhalten der Middleware während der Bearbeitung der jeweiligen Variablen. Es wird festgelegt, ob die Werte einer Variablen repliziert werden und auch mit welcher Qualität dies erfolgen soll. Die Qualität bedeutet hier das Konsistenzmodell, welches eingehalten werden soll.

Wir haben unterschiedliche Anforderungen bezüglich der Konsistenz der Datenhaltung aller Arten der Variablen in der Applikation untersucht. Die, die am effektivsten zu implementieren waren, wurden dann auch umgesetzt. Bei der Implementierung war für uns sehr wichtig die Ebene der Implementierung zu definieren – die Logik kann zwischen Middleware und Applikation verteilt werden. Die Grundidee ist aber die Middleware möglichst allgemein zu realisieren, d.h. spezifische Lösungen sollen in der Applikationslogik implementiert werden, statt alle möglichen Mechanismen in die Middleware zu integrieren.

In der FeuerWhere-Anwendung gibt es drei Klassen von Variablen. Die erste Variablenklasse ist die der Messwerte. Diese Variablen müssen unter allen Knoten verteilt werden, um eine Auswertung der Situation auf diesen Messwert zu erlauben. Dies wird durch einen Broadcast jeden Wertes erreicht. Da die Variablen ziemlich oft geschrieben werden können, kann ein Verfahren, welches auf Bestätigungen basiert, das Netzwerk enorm belasten. Dies kann sich, begründet durch die Gefahr einer Kanalverstopfung, entgegen der eigentlichen Intension negativ auf die Robustheit auswirken. Aufgrund dessen, dass die Werte, die in die Variablen geschrieben werden, nur von kurzer Lebensdauer sind, ist es eher suboptimal zu viel Energie und Bandbreite für die einzelnen Werte zu investieren, solange diese nicht in dem als kritisch eingestuften Wertebereich liegen. Bestätigungen in der Applikation sind leicht umsetzbar. In diesem Fall bildet jeder Knoten im BAN eine Liste von Nachbarn und erwartet von diesen eine Bestätigung. Die Lösung erhöht die Robustheit, kann aber bei hohen Sampling-Frequenzen die Bandbreite schnell ausschöpfen.

Die zweite Klasse bilden die Variablen für Ereigniszustände. Diese werden generell nicht verteilt und jeder Knoten im BAN wertet die Ereignisgleichungen selbst und separat aus. Allerdings ist es in manchen Fällen von Vorteil die Zustandsänderungen in dem Netzwerk zu verteilen. Zudem speichern diese Variablen logische Werte. Basierend auf der Annahme, dass der normale (grüne) Zustand der Normalfall ist, geschieht dies jedoch relativ selten. Das Verteilen von Zustandsvariablen liefert redundante Informationen, die das Wissen eines Knotens ergänzen, falls der Knoten einen bestimmten Messwert nicht erhalten hat, welcher eine Änderung im Zustand verursacht. Wird die Verteilung der Messwertvariablen bestätigt, ist die Verteilung der Variablen für Ereigniszustände nicht mehr notwendig. Die Variablen, deren Wert von mehreren redundanten Messwerten abgeleitet wird, sind ebenfalls dieser Kategorie zuzuordnen.

Die dritte Klasse bilden die Konfigurationsvariablen. Da die Werte, die die Variablen speichern, eine kritische Rolle in der Funktion des BANs spielen, werden alle Schreibzugriffe auf die Variablen aus dieser Klasse bestätigt.

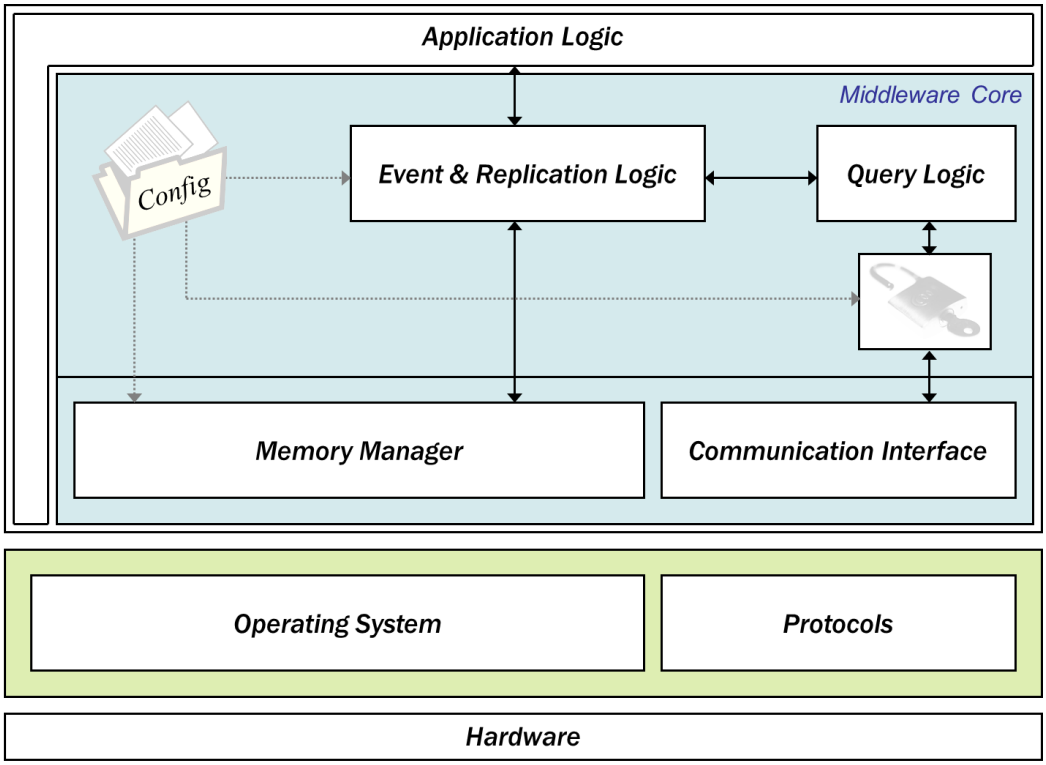

Abbildung 4: Die Architektur der Middleware

Abbildung 4 zeigt die Architektur der Middleware und der Software auf einem Knoten. Die Logik der Applikation (Application Logic) greift auf die Dienste der Middleware und auch auf die unteren Schichten - also auf das Betriebssystems (Operating System) und den Protokollstapel (Protocols) - zu. Diese beiden unteren Schichten greifen dagegen direkt auf die Hardware zu. Die Subkomponenten der Middleware werden durch die Konfiguration (Config) beeinflusst. Die Logik der Kommunikation (Communication Logic) passt die Kommunikation an bestimmte Eigenschaften der Plattform an. Die Anfrage-Logik (Query Logic) generiert und dekodiert die middlewarespezifischen Nachrichten. Der Speicher Manager (Memory Manager) verwaltet die lokal gespeicherten Informationen. Die Ereignis-und-Replikations-Logik (Event and Replication Logic) kümmert sich um die Ereignisdetektion und trifft Entscheidungen zur Verteilung der Variablen in dem Netzwerk.

Diese Architektur der Middleware wurde umgesetzt und auch erfolgreich in unterschiedlichen Konfigurationen getestet.

2.5 AP7 Demonstrator

Zusammen mit der Berliner Feuerwehr wurde die endgültige Konfiguration des Demonstrators definiert. Jedes BAN-Netzwerk besteht aus sechs BAN-Knoten (und einer PTTU). Die Knoten und die Stellen, an denen sich diese befinden, zeigen Abbildung 5 und Abbildung 6.

Abbildung 5: Sensorknoten auf dem Helm

Abbildung 6: Sensoren in der Jacke

Die BAN-Knoten wurden in die für den Demonstrator vorgesehenen MSA Auer-Gehäuse verpackt.

Um den BAN-Demonstrator zu realisieren, wurde eine BAN-GUI-Applikation (Abbildung 7) entwickelt. Die Applikation ermöglicht es, das PTTN Netzwerk durch einen einzelnen BAN-Knoten zu ersetzen, um die Werte des BANs direkt auf dem PC darzustellen. Damit können Ereignisse, wie z.B. das Verhalten des Mechanismus zur Berechnung des Wertes für redundanten Sensoren oder auch die Reaktionen auf kritische Situationen, überwacht werden. Diese Applikation haben wir als Demonstrator des BAN-Netzwerkes benutzt.

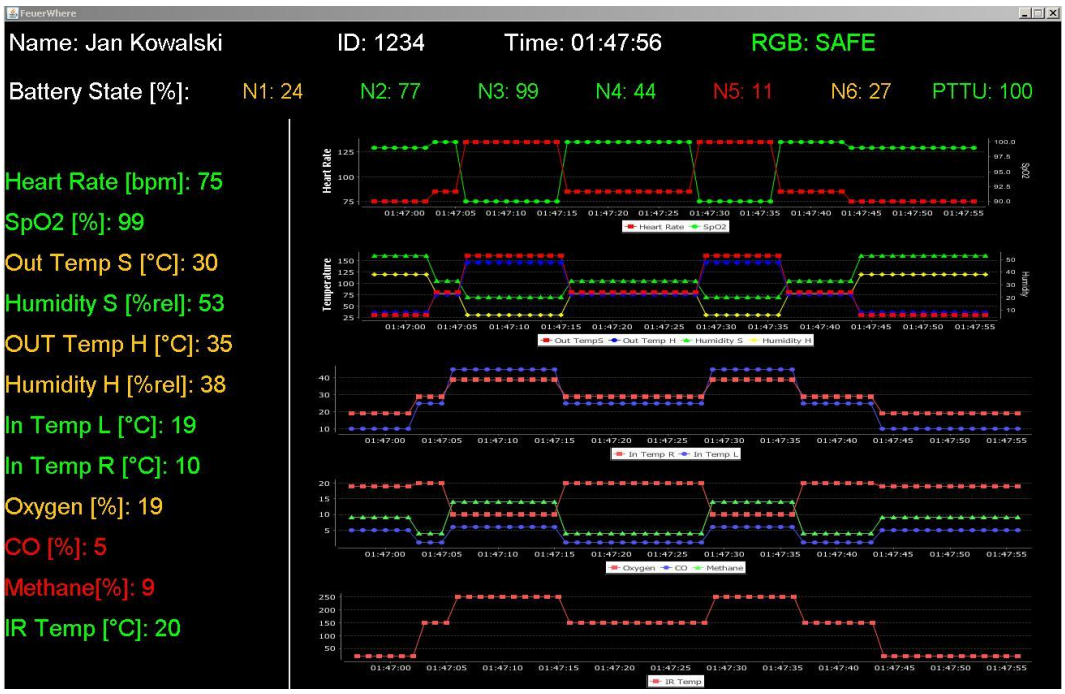

Abbildung 7: Die GUI des BAN Demonstrators

Der BAN-Demonstrator wurde auf folgenden Veranstaltungen vorgestellt:

- 2. Öffentliches Statusmeeting AVS, 11-12.02.2010, Berlin
- 5. Telematik-Konferenz, 17.02.2010, Uni Potsdam
- Jugend Messe, 11.03.2010, TU Cottbus
- Hannover Messe, 19-23.04.2010
- RFID und Mittelstand – Chancen und Potenziale, 27.05.2010, IHK Potsdam
- MobiCom 2010 Konferenz, 20-24.09.2010 in Chicago, USA

Die Zuschauer zeigten ein Interesse sowohl für die Hardware und Software als auch für das Grundkonzept.

Weitere Vorstellungen des BAN-Demonstrators erfolgten unter anderem beim Besuch von Ministerpräsident Matthias Platzeck (Abbildung 8) im IHP, in der Presse (Märkische Oderzeitung, Mosaik), in einer Fernsehsendung des rbb sowie in der Broschüre „Wir erforschen: Sicherheit“ von der Allianz der Wissenschaftsorganisationen.

Abbildung 8: Die Vorstellung des FeuerWhere-BAN-Demonstrators (von links: Ministerpräsident M. Platzek, M. Wilke – Frankfurts Oberbürgermeister, Prof. W. Mehr – Wissenschaftlich-Technischer Geschäftsführer des IHP und Prof. P. Langendörfer – der Leiter der Arbeitsgruppe Sensornetze).

3 Notwendigkeit und Angemessenheit der geleisteten Arbeit

Für die im Rahmen dieses Projektes untersuchten Fragestellungen existieren keine allgemein anerkannten oder gar standardisierten Lösungen. Vielmehr wird in einigen Bereichen noch sehr intensiv an ähnlichen bzw. gleichen Fragestellungen geforscht. Die Arbeiten waren also aus wissenschaftlich-technischer Sicht notwendig, um Aussagen zur Umsetzbarkeit der Basisidee des FeuerWhere-Ansatzes machen zu können. Zu allen Arbeitspaketen wurden intensive Untersuchungen sowohl theoretischer Natur, wie die Berechnungsart der Komplexität von Implementierungen kryptographischer Verfahren, als auch experimenteller Art durchgeführt.

4 Voraussichtlicher Nutzen

Eine Verwertung der FeuerWhere-Ergebnisse kann in unterschiedlichen Kontexten erfolgen. Die kommerzielle Vermarktung der Ergebnisse ist dabei eher langfristig ist zu sehen. Hier gibt es unterschiedliche Einsatzbereiche:

- Medizinische Fachberufe,
- Technische Fachkräfte,
- Bauhilfsarbeiter,
- Metallarbeiter,
- Allergiker.

Die Veranstaltung „Gründungsmanagement“ wurde gemeinsam mit dem Lehrstuhl für Planung und Innovationsmanagement des Instituts für Wirtschaftswissenschaften der BTU Cottbus durchgeführt. Hierbei erarbeiteten interdisziplinäre Gruppen von Studierenden aus wirtschaftswissenschaftlichen und technischen Studiengängen Businesspläne zu aktuellen Forschungsprojekten des IHP. Eine derartige Veranstaltung wurde im Jahr 2010 erstmalig in Deutschland durchgeführt und brachte zahlreiche gute Geschäftsideen hervor. Die zwölf Gruppen erarbeiteten unter anderem Geschäftsideen zum Thema körpernahe drahtlose Sensornetze, Multimediaanwendungen in Flugzeugen und sogar zum Einsatz von Sensornetzen zum Detektieren von Sonnenstürmen im Weltall.

Die FeuerWhere-Techniken könnten bei dem System zur Erkennung von gefährlichen Gasen und deren Konzentration bei Gasaustritt benutzt werden. Viele Berufsgruppen werden mit schädlichen Gasen oder Dämpfen konfrontiert, welche durch ihre transparente und meist geruchslose Eigenschaft nicht unmittelbar auszumachen sind. Eine Gruppe von Studenten hat diesbezüglich im bereits oben erwähnten Gründungsmanagement eine Produktidee – GasProtect – entwickelt, welche, basierend auf dem FeuerWhere-Knoten, die Erkennung von Gasen ermöglichen soll. Diese Gruppe ist dabei im Rahmen des „Lausitzer Existenzgründer“-Wettbewerbs bis unter die besten 6 Teams gekommen.

Eine zweite Produktidee – AllSenTec – ist ein Gerät, welches gefährliche Konzentrationen allergieauslösender Stoffe (wie Pollen) misst und den Nutzer davor warnt. Dabei ist das Gerät nur so groß wie eine Uhr und kann bequem am Handgelenk getragen werden, ohne dass es den Nutzer bei seinen alltäglichen Handlungen stört. Zur Zielgruppe gehören alle Personen, die unter den Symptomen einer durch Pollen, Ozon oder Luftpartikel induzierten Allergie leiden. Der Markt ist immens: 16 Millionen Menschen allein in Deutschland.

5 FE-Ergebnisse von dritter Seite

Es sind zum momentanen Zeitpunkt keine FE-Ergebnisse von dritter Seite bekannt, die die wesentlichen Projektziele beeinflussen.

6 Veröffentlichungen des Ergebnisses

Während der Projektlaufzeit sind die folgenden wissenschaftlichen Veröffentlichungen entstanden:

K. Piotrowski, P. Langendoerfer and St.Peter, tinyDSM: A Highly Reliable Cooperative Data Storage For Wireless Sensor Networks, The 2nd International Workshop on Distributed Collaborative Sensors Networks

K. Piotrowski, A. Sojka and P. Langendoerfer, Body Area Network for First Responders - a Case Study, BodyNets 2010

A. Sojka, K. Piotrowski and P. Langendoerfer, SHORT ECC a Lightweight Security Approach for Wireless Sensor Networks, SECRYPT 2010

K. Piotrowski, A. Sojka and P. Langendoerfer, Wireless Sensor Networks Can Save Lives – Benefits and Open Issues, Sensor + Test 2010

7 Referenzen

[1] M. Johnson, M. Healy, P. van de Ven, M.J. Hayes, J. Nelson, T. Newe, and E. Lewis. A comparative review of wireless sensor network mote technologies. In Sensors, 2009 IEEE, pages 1439-1442. IEEE.

[2] Sentilla Corp. Sentilla Corp. - Homepage, 2011. http://www.sentilla.com/.

[3] MEMSIC Inc. TELOSB Mote Platform, 2011. http://www.memsic.com/support/documentation/wirelesssensor-networks/category/7-datasheets.html?download=152:telosb.

[4] MEMSIC Corp. MEMSIC Corp. - Homepage, 2011. http://www.memsic.com.

[5] MEMSIC Inc. MICA2 Wireless Measurement System, 2011. http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=147:mica2.

[6] MEMSIC Inc. MICAz Wireless Measurement System, 2011. http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=148:micaz.

[7] MEMSIC Inc. IRIS Wireless Measurement System, 2011. http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=135:iris.

[8] Atmel Corp. Atmel AVR 8- and 32-bit, 2011. http://www.atmel.com/products/avr/.

[9] Texas Instruments Inc. MSP430x1xx Family User's Guide, 2006. http://www.ti.com/litv/pdf/slau049f.

[10] Texas Instruments Inc. MSP430x5xx/MSP430x6xx Family User's Guide, 2010. http://www.ti.com/litv/pdf/slau208h.

[11] Texas Instruments Inc. CC430F613x, CC430F612x, CC430F513x MSP430 SoC With RF Core, 2010. http://www.ti.com/lit/gpn/cc430f6137.

[12] ARM Ltd. ARM Ltd. - Homepage, 2011. http://www.arm.com/.

[13] Texas Instruments Inc. Single-Chip Very Low Power RF Transceiver, 2007. http://www.ti.com/lit/gpn/cc1000.

[14] Texas Instruments Inc. CC1101 Low-Power Sub-1 GHz RF Transceiver, 2010. http://www.ti.com/lit/gpn/cc1101.

[15] Texas Instruments Inc. Low-Cost Low-Power 2.4 GHz RF Transceiver, 2009. http://www.ti.com/lit/gpn/cc2500.

[16] Texas Instruments Inc. 2.4 GHz IEEE 802.15.4 / ZigBee-Ready RF Transceiver, 2007. http://www.ti.com/lit/gpn/cc2420.

[17] Texas Instruments Inc. 2.4 GHz IEEE 802.15.4/ZIGBEE RF TRANSCEIVER, 2007. http://www.ti.com/lit/gpn/cc2520.

[18] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, et al. Tinyos: An operating system for sensor networks. Ambient Intelligence, pages 115-148, 2005

[19] Adam Dunkels, Bj□orn Gr□onvall, and Thiemo Voigt. Contiki - a lightweight and exible operating system for tiny networked sensors. In Proceedings of the First IEEE Work-shop on Embedded Networked Sensors (Emnets-I), Tampa, Florida, USA, November 2004.

[20] Karsten Walther and Jorg Nolte. A exible scheduling framework for deeply embedded systems. In Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops - Volume 01, AINAW '07, pages 784-791, Washington, DC, USA, 2007. IEEE Computer Society.

[21] P. Costa, L. Mottola, A. Murphy, and G. Picco. Programming Wireless Sensor Networks with the Teeny Lime Middleware. Middleware 2007, pages 429-449, 2007.

[22] P. Costa, L. Mottola, A.L. Murphy, and G.P. Picco. TeenyLIME: transiently shared tuple space middleware for wireless sensor networks. In Proceedings of the international workshop on Middleware for sensor networks, pages 43-48. ACM, 2006.

[23] Paolo Costa, Luca Mottola, Amy L. Murphy, and Gian Pietro Picco. Developing sensor network applications using the teenylime middleware. Technical report, 2006.

[24] T.W. Hnat, T.I. Sookoor, P. Hooimeijer, W. Weimer, and K. Whitehouse. Macrolab: a vector-based macroprogramming framework for cyber-physical systems. In Proceedings of the 6th ACM conference on Embedded network sensor systems, pages 225-238. ACM, 2008.