Abschlussbericht für das Verbundprojekt Sens4U

Zuwendungsempfänger:	**Förderkennzeichen:**
IHP GmbH	**03WKP26A**

Vorhabensbezeichnung: Sens4U, Teilvorhaben: **IHP**

Laufzeit des Vorhabens:

01.04.2012 - 30.06.2014

Berichtszeitraum:

01.04.2012 - 30.06.2014

Autor : Krzysztof Piotrowski, Peter Langendörfer

Version : 1.0

Datum : 20.12.2014

Inhaltsverzeichnis

I. Kurze Darstellung 3
1 Aufgabenstellung 3
2 Voraussetzungen 4
3 Planung und Ablauf des Vorhabens 5
4 Wissenschaftlicher und technischer Stand 5
5 Zusammenarbeit mit anderen Stellen 9
II. Eingehende Darstellung 10
1 Verwendung der Zuwendung 10
2 Erzielte Ergebnisse 10
2.1 AP1 Spezifikation der Plattformarchitektur 10
2.2 AP3 Design / Realisierung Sensorknotenplattform 15
2.3 AP4 Netzwerk- und Systemmanagement 18
2.4 AP5 Power Management 21
2.5 AP6 Entwicklung Anwendungssoftware 21
2.6 AP7 Demonstrator 22
3 Notwendigkeit und Angemessenheit der geleisteten Arbeit 23
4 FE-Ergebnisse von dritter Seite 24
5 Veröffentlichungen des Ergebnisses 24
6 Referenzen 24

I. Kurze Darstellung

1 Aufgabenstellung

Das Hauptziel von Sens4U war die Definition und Realisierung einer modularen Sensornetzplattform für Umwelt Monitoring-Netze. Diese Plattform sollte sich von bisherigen Ansätzen vor allem dadurch abheben, dass sie es Nicht-Experten erlaubt, drahtlose Sensornetze, die aus heterogenen Komponenten bestehen können, zumindest im Groben zu entwerfen. Hierzu sollte ein halbautomatischer Entwurfsprozess entwickelt werden, der auf Basis eines beliebigen Anwendungsszenarios konkrete technische Anforderungen ableitet. Die Module, die in diesem Entwurfsprozeß Verwendung finden sollen und im Rahmen von Sens4U untersucht und prototypisch realisiert wurden, sind:

- Ein modularer drahtloser Sensorknoten für die Umweltüberwachung inklusive des Betriebssystems. Der Sensorknoten selbst kann mit einem Harvesting-Modul für die autonome Energieversorgung ausgerüstet werden bzw. besitzt Komponenten, um von einem externen Harvesting-Modul versorgt zu werden. Eine Repeaterfunktionalität sollte ebenfalls integriert sein.
- Externe Harvesting-Module sowie Harvesting-Komponenten für den Sensorknoten zur wahlweisen externen bzw. integrierten autonomen Stromversorgung von Sensorknoten und Repeater.
- Ein externer Repeater ausschließlich zur Vergrößerung der Reichweite mit externer bzw. interner autonomer Energieversorgung.
- Ein Gateway-Knoten zur Verbindung mit dem Internet.
- Nutzung von mobilen Geräten zur direkten Vor-Ort-Inspektion von Sensorknoten und Repeatern.
- Eine Programmierumgebung für Umweltfachleute zur einfachen Programmierung des Sensornetzes für Umweltmonitoring-Anwendungen.
- Ein Steuerprogramm zur Überwachung von Monitoring-Netzen.
- Ein Auswerteprogramm zur statistischen und grafischen Auswertung der Messergebnisse.
- Eine Testumgebung für Hardware- und Softwarekomponenten des drahtlosen Sensornetzwerkes.
- Ein Sicherheits- und Zuverlässigkeitskonzept zur Gewährleistung der erforderlichen Sicherheit für Menschen und Umwelt.

Zusätzlich sollte ein Expertensystem zur Spezifikation von Umwelt Monitoring Systemen realisiert werden, das den bereits erwähnten Entwurfsprozess unterstützt. So soll es möglich sein, schnell und mit wenig Aufwand Monitoring-Netze zu projektieren und auf der Basis der Plattformkomponenten zu realisieren.

In Folgenden werden die spezifischen Aufgaben und Ziele des IHP Teilvorhabens vorgestellt. Von den oben genannten Aussagen zu den Zielen werden nochmals die, die das IHP betreffen kursiv wieder gegeben und erläutert.

- *Ein modularer drahtloser Sensorknoten für die Umweltüberwachung inklusive des Betriebssystems. Der Sensorknoten selbst kann mit einem Harvesting-Modul für die autonome Energieversorgung ausgerüstet werden bzw. besitzt Komponenten, um von einem externen Harvesting-Modul versorgt zu werden. Eine Repeaterfunktionalität sollte ebenfalls integriert sein.*

Die Aufgabe des IHP war es Ansätze zur modularen Gestaltung der Sensorknoten zu untersuchen. Das Hauptaugenmerk des IHP sollte auf der Integration eines Adapters zum Anschluss von Energie Harvesting Modulen liegen. Im Bereich der Software sollte der Fokus bei der Auswahl von Sicherheitsmodulen und bei der Sicherstellung der Zuverlässigkeit der Systemlösung sein.

- *Eine Programmierumgebung für Umweltfachleute zur einfachen Programmierung des Sensornetzes für Umweltmonitoring-Anwendungen.*

Die Aufgabe bezieht sich auf die Definition einer abstrakten Beschreibung des Systemverhaltens die eine automatische Übersetzung in eine Konfiguration von Software Modulen, die das Verhalten abbilden ermöglicht. Das Systemverhalten beinhaltet das Auslesen von Sensorwerten, Bearbeitung von den Messwerten wie auch Reaktionen auf definierte Ereignisse.

- *Eine Testumgebung für Hardware- und Softwarekomponenten des drahtlosen Sensornetzwerkes.*

Das IHP sollte am Aufbau der Demonstratoren mitwirken, das heißt, das IHP sollte die Partner bei der Programmierung und bei dem Ausbringen der Sensorknoten unterstützen. Für die Ausbringung der Knoten wurde im Rahmen des Werkzeugansatzes eine Komponente entwickelt die es ermöglicht die Topologie des Netzwerkes zu planen und zu optimieren. Die Komponenten wurde beim Aufbau der Demonstratoren getestet. Das Ziel war es eine zuverlässige Kommunikationsarchitektur aufbauen zu können insbesondere, wenn Sensornetzwerk Nicht-Experten den Aufbau vornehmen.

2 Voraussetzungen

Das IHP hat im Rahmen mehrere Projekte an verschiedenen Aspekten vonr Sensornetzwerken gearbeitet.

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

- FeuerWhere (BMBF),

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

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

- TAMPRES (EU 7th Framework),
- WSAN4CIP (EU 7th Framework),
- UbiSecSens (EU 6th Framework),
- Mobile Internet Business (BMBF).

Sensorknotenplattformen wurden in folgenden Projekten entwickelt:

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

3 Planung und Ablauf des Vorhabens

Der zeitliche und inhaltliche Ablauf des Vorhabens entsprach weitgehend der ursprünglichen Planung. Eine größere Abweichung gab es bei der Realisierung und Integration der Hardware. Die Integration der einzelnen Komponenten in ein finales System war deutlich aufwändiger als ursprünglich geplant. Einige der bestellten Hardware Module wurden verspätet geliefert, weil die Lieferzeiträume für die Bauteile ungeplant und nichtplanbar lang waren.

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 (jetzt erhältlich als TelosB von MEMSIC[1]). Weitere Beispiele von Lösungen von MEMSIC sind die MICA und IRIS Plattformen. Die Sensorknoten können in Abhängigkeit von der Prozessor-Architektur in drei Gruppen aufgeteilt werden: Entweder werden sie mit einem 8-bit Mikrokontroller (z.B. aus der AVR Familie von Atmel[2]), einem 16-bit Mikrokontroller (z.B. aus der MSP430 Familie von Texas Instruments[3]), oder mit einem 32-bit Mikrokontroller (basierend auf ARM[4]) ausgestattet. Die Architektur bestimmt die verfügbare Rechenleistung der Plattform und beeinflusst auch den Energieverbrauch. Das andere Kriterium zur

[1] MEMSIC Corp., http://www.memsic.com
[2] Atmel Corp., http://www.atmel.com
[3] Texas Instruments Corp., http://www.ti.com
[4] ARM Corp., http://www.arm.com

Klassifizierung von Sensorknoten-Plattformen ist das verwendete Funkmodul. Alle obengenannten Plattformen benutzen Radio Module von Chipcon (jetzt Texas Instruments): CC1000 und seine Nachfolger CC1101 und CC2500 oder CC2420 – einen IEEE 802.15.4 und ZigBee konformen Transceiver und seinen Nachfolger den CC2520.

Ein Nachteil von allen auf dem Markt verfügbaren Hardware-Plattformen ist, dass sie eine statische Konfiguration haben, z.B. jeweils nur ein Funkmodul verwenden. Damit werden eventuelle notwendige Änderungen der Konfiguration um z.B. die Kommunikationseigenschaften zu beeinflussen stark eingeschränkt. In dem IHP Projekt TANDEM[5] "Ein extrem verbrauchsarmes, skalierbares, TANDEM-Prozessor-basiertes Funksystem für sensorische, aktuatorische und kennzeichnende Anwendungen" wurden grundlegende Fragen zum Aufbau effizienter und zuverlässiger Sensorknoten und Sensornetze bearbeitet. Im Rahmen dieses Projektes wurde ein erster Ansatz zum modularen Aufbau von Sensorknoten entwickelt. Abbildung 1 zeigt eine Beispielausprägung.

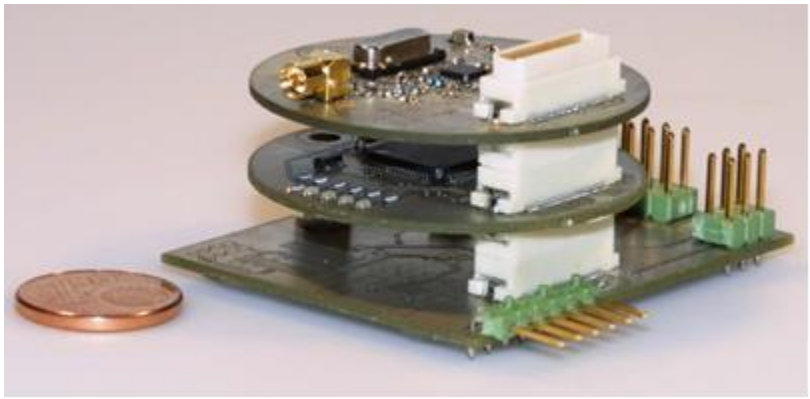

Abbildung 1: Tandem-Stack Beispiel

Betriebssysteme

Die Betriebssysteme für drahtlose Sensornetze sind wegen der spezifischen Eigenschaften von drahtlosen Sensorknoten eingeschränkt. Betriebssysteme für solche Geräte müssen folgende Aspekte berücksichtigen: die Begrenzung der Kodegröße, das Energie-, Speicher-, und Peripheriemanagement, die Nebenläufigkeit der Mechanismen, das Steuerprogramm, das Multitasking und die Robustheit.

TinyOS [2] 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 keine Aufgaben bearbeitet werden müssen. Aufgrund von Optimierungen zwischen den Komponenten und der Kompilierung in ein

[5] TANDEM Projekt, http://www.tandem-projekt.de

monolithisches System produziert TinyOS normalerweise kleineren und schnelleren Kode als pure-C-Ansätze.

Contiki [3] 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 monolithisches System hochgeladen werden. Contiki ist ein Hybrid, d.h. eine Kombination aus einem ereignisorientierten Kernel als Basis des Systems und einer präemptiven multi-threading Bibliothek, die optional eingefügt 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 [4] ist ein Betriebssystem, welches als elementare Abstraktion 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.

Middleware Lösungen für Sensornetze

TeenyLIME [5][6][7] 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 an der Messquelle durchgeführt wie möglich.
- Mehrere Tasks laufen parallel und benutzen gemeinsame Daten.
- Zustandsbehaftete Koordination - mittels gemeinsamen Teils der 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 [8] 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 – TeenyLIME und MacroLab- 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.

TinyDSM [9] ist eine Middleware für Sensornetzwerke die einen konfigurierbaren gemeinsamen Speicher für die Knoten im Netzwerk bietet. Die Middleware Schicht stellt den Anwendungen immer die gleiche API bereit und wird mit der Beschreibung der gemeinsam genutzten Daten konfiguriert. Der Speicher kann auf der Basis einzelner Variablen definiert werden. Die Beschreibung beinhaltet die Definition der Typen der individuellen Variablen sowie auch die gewünschte Behandlung der Daten. Dies ermöglicht auf einer Seite die Definition der Konsistenz des gemeinsamen Speichers und auf der anderen die Definition von Ereignissen, welche an die Anwendung gemeldet werden sollen.

Entwicklung und Testen von Sensornetzwerk Anwendungen

Da die Kosten und die Dauer für Sensornetzwerkentwicklungen hoch sind, ist die Entwicklung von Werkzeugen zur Vereinfachung des Entwicklungsprozesses ein Thema, das in der Forschung aktiv betrachtet wird. So wurden verschiedenste Programmierabstraktionen und grafische Oberflächen vorgeschlagen. Srijan [20] und Viptos [11] sind zum Beispiel grafische Frameworks für die Programmierung von Sensornetzen, die sich aber eher an Programmierer wenden. eBlocks [12] hingegen richtet sich auch an Endanwender, indem es Prozessflüsse wie in der Industrieautomatisierung visualisieren kann. Allerdings sind die funktionalen Basismethoden beschränkt und es unterstützt keine Zuverlässigkeits- oder Sicherheitsaspekte. Der Standardansatz der Programmierabstraktionen ist die Anwendung von einer mächtigen Middlewareschicht, die dem Programmierer Routinearbeiten abnimmt. TASK [10], Impala [16] und Macrolab [8] sind Beispiele für diese Technik, die aber nur selten das Potential von Sensornetzen ausschöpfen kann. Ein alternativer Ansatz sind komponentenbasierte Kompositionsansätze wie SNACK [14] und ConfigKIT [21], welche Anwendungen zusammenstellen ausgehend von den technisch formulierten Anforderungen. Das verspricht sehr hohe Leistung, da nur genau die Komponenten im System sind, die gebraucht werden. Das ist allerdings auch mit der Herausforderung verbunden, die korrekten Module auszuwählen und das resultierende System bewerten zu können. Ein weiteres Problem, das für alle Ansätze zutrifft, ist die Fragestellung, wie Endanwender ihre Anforderungen definieren können. In [23] und [22] wurden technische Sensornetzwerk-Taxonomien vorgeschlagen. Diese sind allerdings für Endanwender zu technisch. [19] untersuchte, wie die oft unscharfen Anforderungen der Anwender erfasst und zu technischen Anwendungsparametern abgeleitet werden können. Hierfür wurden ein kataloggesteuerter Dialogprozess und eine anschließende grafenbasierte

Erweiterung zu allen daraus ableitbaren Anforderungen vorgeschlagen. Diese wurde bisher allerdings nur theoretisch analysiert.

Testen wird als Prozess betrachtet, in dem gezeigt wird, dass das Produkt die spezifizierten Anforderungen und seinen Zweck erfüllt. Im Allgemeinen werden Ziele verfolgt, wie Aufdecken von Fehlern, Erzeugen von Vertrauen in das System, Liefern von Informationen über das Qualitätsniveau, Vorbeugen von Fehlern (in Zukunft). Aus diesen Zielen ergeben sich Schwerpunkte für bestimmte Tests, z.B. herstellerinterne Tests sollen viele Fehlerwirkungen zeigen, Abnahmetests weisen Funktionalität nach und Regressionstests prüfen speziell die Auswirkung von Änderungen auf das Gesamtsystem.

Die Überprüfung von drahtlosen Sensornetzwerken konzentriert sich meist auf Systemtests (Abnahmetests). Es werden Ausführungen des kompletten Systems, bestehend aus Hardware- und Softwarekomponenten, analysiert. Dabei stellt sich das Testen von drahtlosen Sensornetzwerken als eine komplexe Aufgabe dar, da drahtlose Sensornetzwerke als verteilte, nebenläufige Systeme über einen großen Zustandsraum verfügen. Derzeit stellen sich grundsätzliche Herausforderungen an das Testen von WSNs, wie z.B.

- Abbildung einer typischen WSN-Installation mit ähnlich hoher Anzahl an Sensoren und deren Vorortbedingungen
- Emulation von Umweltveränderungen und Stimulierung von Störungen durch die Umwelt
- Bereitstellung einer ausreichenden Isolierung zwischen der Testinfrastruktur und des zu testenden Systems.

Diese hohen Testanforderungen führen zu einem sehr knappen, kaum entwickelten Angebot an kommerziellen Lösungen für eine angemessene Testumgebung bzw. Testinfrastruktur. Die derzeitig verfügbaren Testtools wie z.B. TOSSIM [24], Motelab [17], the Deployment Support Network [15] oder OMNeT++ [18] sind zumeist universitäre Entwicklungen, die verschiedene Ansätze (Simulation eines Netzwerks, Applikation einer experimentellen Testumgebung) in unterschiedlichen Schichten bzw. Komponenten eines Netzwerkes realisieren. Beim Einsatz vorhandener Testtools ist darauf zu achten, dass nur die am meisten verwendeten HW- und SW-Komponenten für WSN unterstützt werden, bei Betriebssystemen z.B. TinyOS oder bei Mikrocontrollern z.B. Atmel AVR oder TI MSP430. Bei allen zurzeit zur Verfügung stehenden Testmitteln ist es erforderlich, dass die Tests für jede Plattform individuell entwickelt werden.

Eine Systematik des Testens, wie sie bereits in Industriezweigen wie z.B. der Luftfahrttechnik mit Standards DO178-B für Software [13] existiert, ist nicht zu identifizieren. Eine gute Grundlage bietet zurzeit nur das universitäre Projekt JAWS [15].

5 Zusammenarbeit mit anderen Stellen

Als Koordinator hat das IHP hat im Rahmen des Projektes mit allen Projektpartnern zusammengearbeitet. Die Zusammenarbeit verlief erwartungsgemäß, produktiv und reibungsfrei. Nur in der Anfangsphase musste zunächst mit einer gemeinsamen Sprache

gefunden und etabliert werden, was sich aus den Unterschieden im Grundwissen ergibt. Die zahlreichen Projekttreffen und Statusmeetings haben eine schnelle Lösung entstandener Probleme ermöglicht.

II. Eingehende Darstellung

1 Verwendung der Zuwendung

Die Zuwendung wurde konform zur Antragstellung verwendet. Alle Arbeitspakete des IHP 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, z.B. für die Aufgaben die ursprünglich von Ezent bearbeitet werden sollten. Für die Präsentation der Ergebnisse wurde ein Demonstrator realisiert, in dem alle angedachten Dienste mit Hilfe der im Projekt entwickelten Soft- und Hardware demonstriert werden können.

2 Erzielte Ergebnisse

Die Ergebnisse werden entsprechend der Arbeitspaketstruktur präsentiert.

2.1 AP1 Spezifikation der Plattformarchitektur

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.

AP1.1 Spezifikation der Architektur

In dem Sens4U Projekt sollte eine Methodik entwickelt werden mit der die Entwicklung von Anwendungen für (Umwelt) Monitoring beschleunigt und vereinfacht wird. Das Einsatz Szenario des Systems ist in der Abbildung 2 dargestellt.

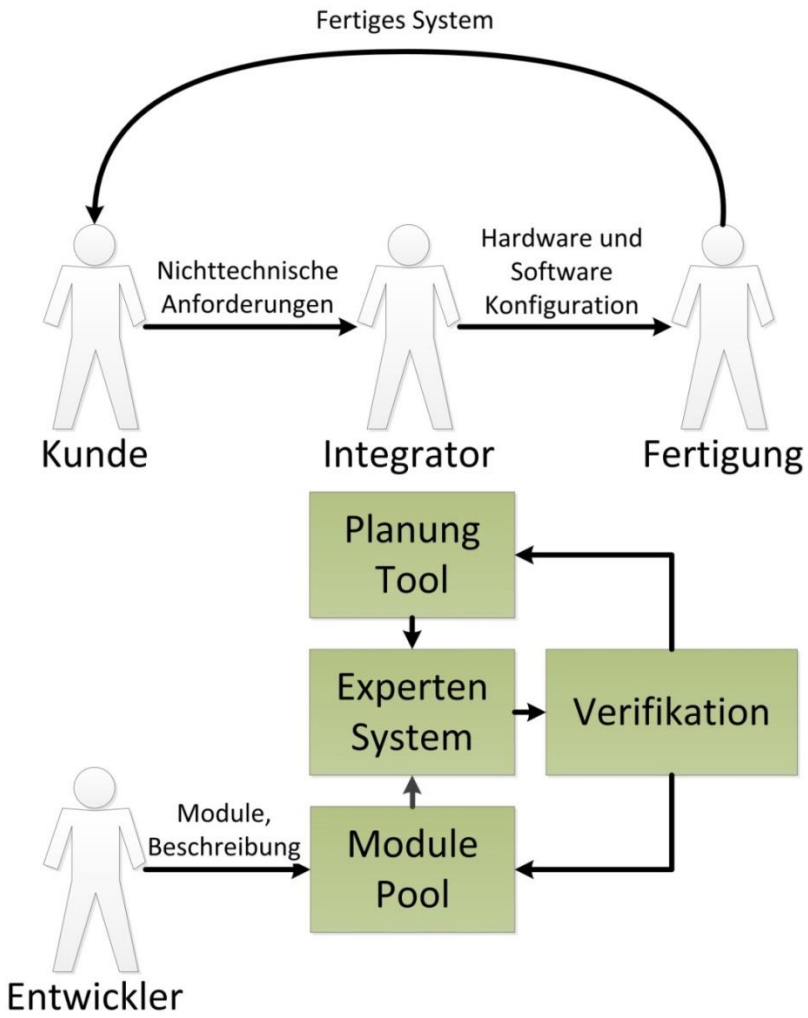

Abbildung 2: Entwicklungszyklus einer Sensornetzanwendung mit dem Sens4U Ansatz

In dem Einsatz Szenario gibt es vier Klassen von Beteiligten/Spielern. Der Kunde benötigt ein System für(Umwelt-) Monitoring. Der Kunde übergibt bzw. bespricht seine nichttechnischen Anforderungen an das System an einen (System) Integrator. Mit Hilfe des Planung Tool wandelt der (System) Integrator die nichttechnischen Anforderungen des Kunden in einen Satz technischer Anforderungen um. Diese werden dann mit Hilfe des Experten System in eine Reihe von Konfigurationen umgewandelt. Diese Konfigurationen beschreiben den Zusammenbau der Applikation/Anwendung und umfassen alle Schichten (Hardware und Software Knoten) und Gateway Konfiguration, wie auch GUI und Datenbank auf der PC Seite. Die Hardware und Software für Knoten und Gateway werden modular und applikationsspezifisch aufgebaut. Die durch die Entwickler implementierten Module bilden werden im Modul Pool zusammengefasst und verwaltet. Jedes vorhandene Software- und Hardware Modul wird durch einen Satz von Parametern beschrieben. Basierend auf den Anforderungen wählt das Experten System die am geeignetste Modul Kombination aus. Das Auswahlkriterium ist die Beschreibung des jeweiligen Modules. Außerdem, generiert das Experten System die Konfigurationen für die GUI und Datenbank auf dem PC. Die generierten Konfigurationen können verifiziert werden. Die erste Verifikationsphase erfolgt schon in dem

Experten System selbst.. Können die erwünschten Anforderungen nicht erfüllt werden, muss entweder der Modul Pool erweitert oder die Anforderungen geändert werden. In der on-line Phase, kann die Konfiguration bezüglich der Netzwerkverbindungen mit Hilfe des Deployment Tools und bezüglich Funktionalität/Daten mit Hilfe des Management Tools verifiziert werden. Werden mit einem der beiden Verifikation Tools Fehler entdeckt, können diese auch durch Änderungen der Module im Modul Pool oder durch Anpassung der Anforderungen behoben werden.

Mit der Konfiguration erstellt die Fertigung für den Kunden das fertige System (die Anwendung). Der Aufbau der Anwendung ist in Abbildung 3 dargestellt. Er umfasst alle Ebenen der Anwendung; das Netzwerk von Sensor Knoten, das Gateway wie auch die PC Domäne mit der GUI und Datenbank Anwendung. In der PC Domäne werden hier auch mobile Geräte berücksichtigt.

Abbildung 3: Der Aufbau der Sens4U (Umwelt-) Monitoring Anwendung

Die Hauptprobleme in der Methodik sind die Umwandlung von nichttechnischen Anforderungen in technische Anforderungen und die Spezifikation von Mechanismen für die Module Auswahl anhand dieser Anforderungen. Die Schwierigkeit des letzteren ist auch mit der Komplexität der Beschreibung wie auch mit Testbarkeit der Module verbunden. Die Beschreibung soll die Funktionalität des Modules wie auch seine Qualität umfassen. Idealerweise soll diese Beschreibung es aber auch erlauben die Module hinsichtlich ihrer Funktionalität und Qualität zu testen.

Die Hauptkomponenten des Sens4U Systems werden in Abbildung 4 dargestellt. Für jede Komponente werden die beteiligten Sens4U Partner genannt.

In der Abbildung erscheint auch die Simulationskomponente, die in der ursprünglichen Projektplanung nicht existierte. Wir wollten aber in dem Projekt zumindest die Einbindung einer derartigen Funktionalität untersuchen, da eine solche Komponente uns und später auch Anwendern zusätzliche Möglichkeiten zur Validierung des Sensornetzentwurfes liefert. Wir haben die Schnittstelle für diese Komponente identifiziert aber keinen Sensornetzwerk Simulator eingebunden bzw. entworfen, deswegen ist Komponente ausgegraut.

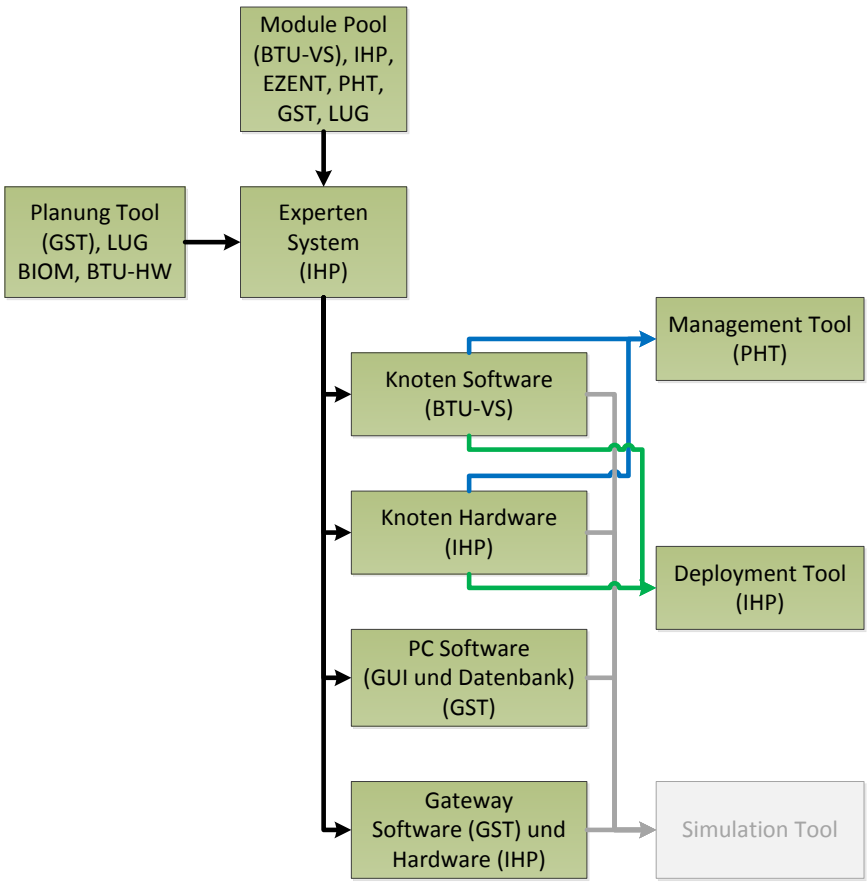

Abbildung 4: Die Komponenten des Sens4U Systems

AP1.2 Expertensystem

In dem Arbeitspaket haben wir die Konzepte für das Expertensystem und den Modul-Pool entwickelt und implementiert. Der Meilenstein M-I-1 wurde planmäßig erreicht.

Das Expertensystem kann auf Module aus der Datenbank (Modul-Pool) zugreifen und daraus alle Sensorknotenplattformen mit den entsprechenden Submodulen aufbauen. Ausgehend von den Anforderungen an den Sensorknoten wird die Konfiguration der Applikationsschicht erzeugt. Anhand dieser wird der restliche Sensorknoten mit den notwendigen Komponenten aufgebaut. Dabei wird anhand der gegebenen Schnittstellen überprüft, welche Module kompatibel sind. Inkompatible Module werden herausgefiltert.

Abbildung 5 stellt den Ablauf der Anfragebearbeitung und die Architektur des Expertensystems dar. Es besteht aus der Schicht, die die Anfragen bearbeitet und verifiziert und aus der Schicht die die Konfigurationen generiert. Jede Anfrage vom Planungstool. wird im JSON-Format[6] übertragen und wird mit einer ID identifiziert. Diese ID wird zurückgeschickt und wird zur Überprüfung des Anfragebearbeitungszustandes genutzt. Der Planungstool-Parser übersetzt

[6] JSON – JavaScript Object Notation, ist ein kompaktes Datenformat in einer einfach lesbaren Textform zum Zweck des Datenaustauschs zwischen Anwendungen. Ist eine Alternative zur XML und ist einfacher zum Parsen.

das JSON-Format der Anfrage ins GEXF-Format[7], welches das Sensornetzwerk als einen Graph beschreibt. Die Eigenschaften der und die Anforderungen an die Anwendung werden hier ebenfalls beschrieben. Der MTDL-Parser übersetzt die Beschreibung der Messaufgaben in eine Konfiguration der Anwendungsschicht für jeden Knoten. Diese Konfiguration definiert, welche Schnittstellen bei der Anwendungsschicht genutzt und geliefert werden und setzt damit die Anforderungen für die Auswahl der weiteren Module (Hardware und Software) für jeden Knoten.

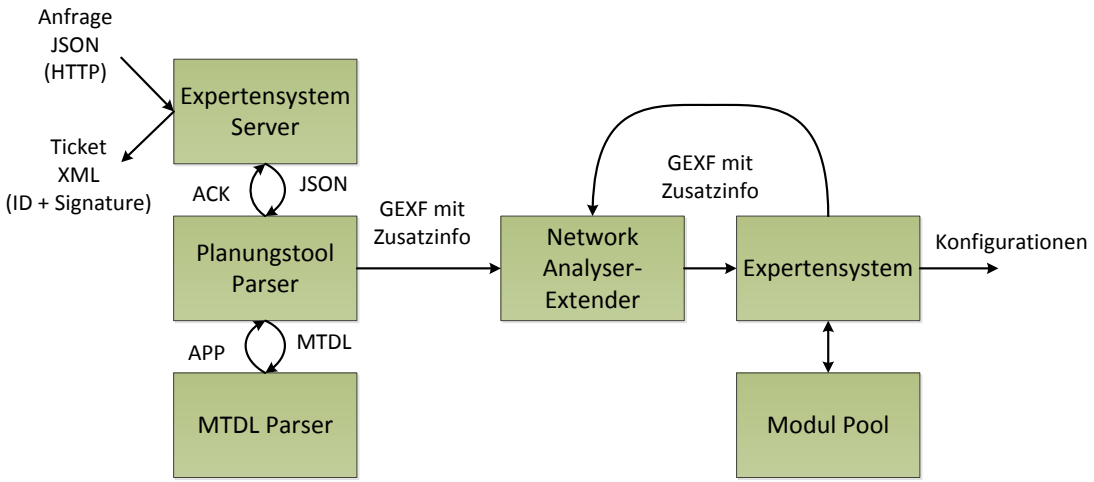

Abbildung 5: Die Architektur des Expertensystem Servers

Die fertige GEXF-Datei wird bearbeitet, um die Konfiguration(en) für die komplette Anwendung zu erzeugen. Im ersten Schritt wird diese an den Network-Analyser-Extender weitergeleitet. Das Werkzeug bearbeitet die Netzwerkeigenschaften des Sensornetzwerkes. Die JSON-Datei beschreibt die Positionen der Messstellen dabei werden aber die Eigenschaften der Kommunikation nicht berücksichtigt. Deswegen betrachtet der Network-Analyser-Extender die Messstellen und die Senke als ein Knotennetzwerk mit Sterntopologie und nutzt diese Information für eine weitere Analyse.

Nachdem die Netzwerktopologie und Netzwerkeigenschaften definiert sind, wird die veränderte GEXF-Datei in dem Expertensystem bearbeitet. Sollten die Anforderungen nicht erfüllbar sein, z.B. , steht im Modul-Pool kein Funkmodul zur Verfügung, dasr die Entfernung zwischen den Knoten im Netzwerk überbrücken kann, wird die GEXF-Datei mit einer entsprechender Meldung wieder an den Network-Analyser-Extender zur Bearbeitung geleitet. In diesem Fall können z. B. zusätzliche Repeater-Knoten in das Netzwerk eingefügt werden. Die Bearbeitung erfolgt in einer Schleife, bis die Konfiguration erfolgreich generiert ist oder bis festgestellt wird, dass eine Lösung nicht möglich ist.

[7] GEXF – Graph Exchange XML Format)ist eine Sprache zur Beschreibung komplexer Netzwerk Strukturen.

Das Expertensystem ist also verantwortlich für die Umsetzung der Anwendung auf der Basis der Beschreibung des Systemverhaltens. Die alle im Projekt definierten Software- und Hardware Module werden dabei berücksichtigt.

2.2 AP3 Design / Realisierung Sensorknotenplattform

In diesem Arbeitspacket haben wir uns mit der Entwicklung der Sensorknotenplattform wie auch mit der abstrakten Art der Beschreibung der einzelnen Module beschäftigt. Die Beschreibung der Module erlaubt es die Module in einer Konfiguration zu verbinden, die die Anwendung abbildet, d.h. die definierten Aufgaben und Anforderungen erfüllt.

AP3.1 Schnittstellenkonzeption und Umsetzung

Das Konzept für die Verwaltung von Software- und Hardwaremodulen wurde definiert. Dabei wurde darauf geachtet, dass das Konzept so abstrahiert wurde, dass Software- und Hardwaremodule gleichbehandelt werden können (sehe Abbildung 6).

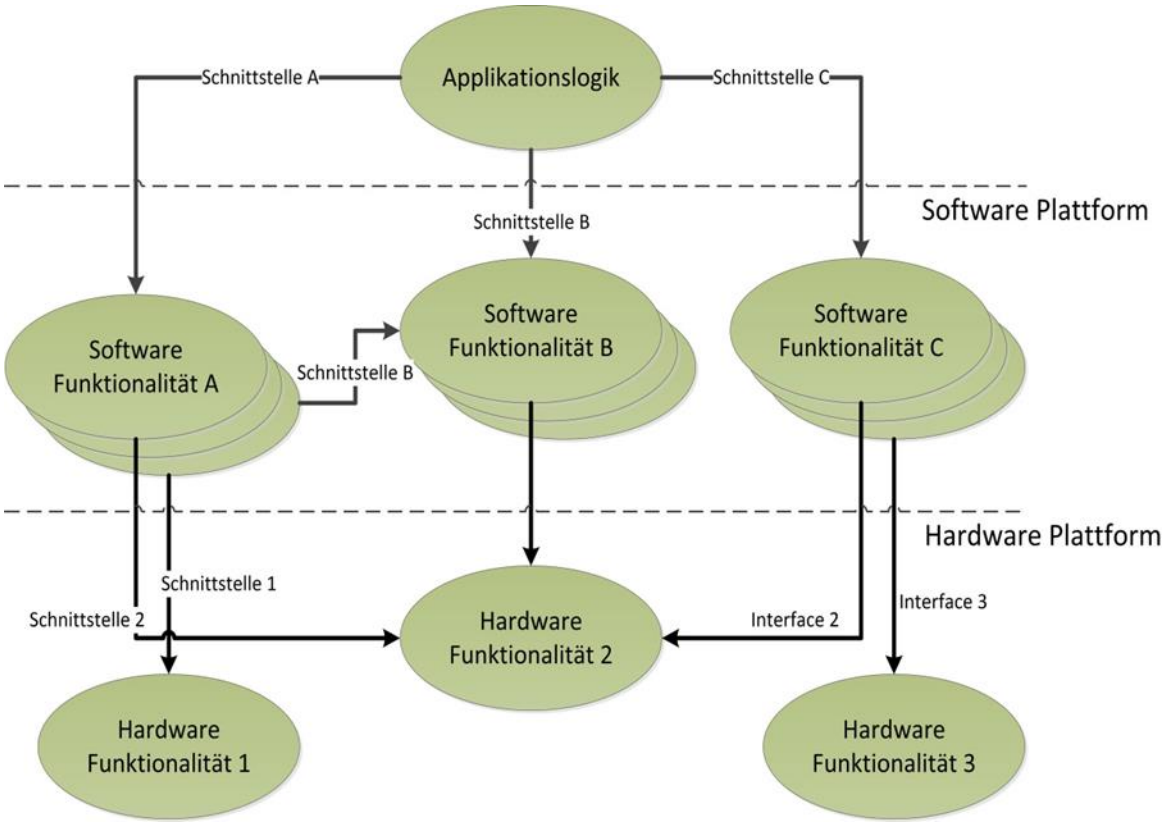

Abbildung 6: Das Konzept der Modularität

Die Umsetzung führte zu einer Datenbank (Module Pool, sehe Abbildung 7), die das Konzept komplett wiederspiegelt. In der Datenbank werden Module, deren Schnittstellen und Parameter wie auch die Beziehungen zwischen diesen gespeichert. Dabei entsprechen Module sowohl Hardware- als auch Softwarekomponenten. Schnittstellen sind die Komponenten, die ein Modul

zur Verfügung stellt und/oder benötigt, um korrekt arbeiten zu können. Parameter dienen dazu Module genauer zu beschreiben und Module mit gleicher Funktionalität unterscheidbar zu machen.

Weiterhin wurde eine Webapplikation entwickelt, die es ermöglicht auf die Moduldatenbank zugreifen zu können. Dabei besteht die Möglichkeit sich Datenbanktabellen anzeigen zu lassen und neue Module und Schnittstellen hinzuzufügen.

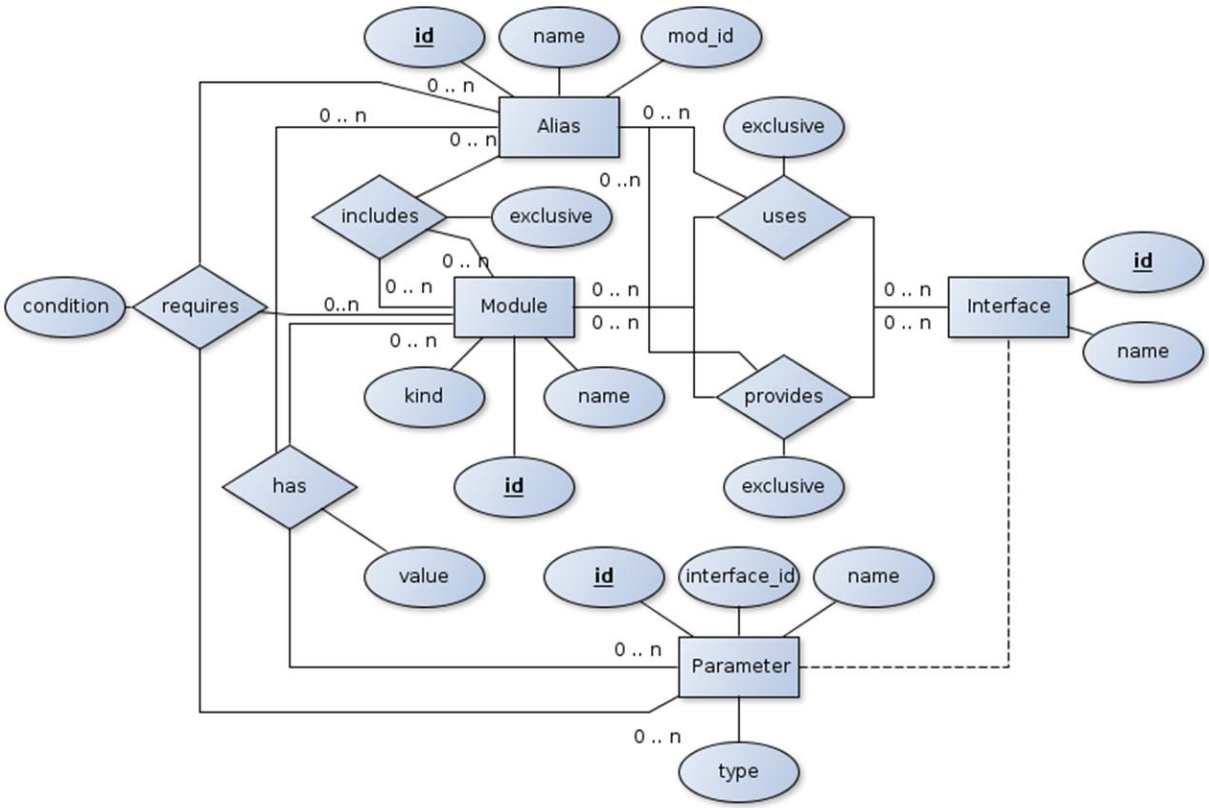

Abbildung 7: ER-Modell des Module Pools

Für die hardwaretechnische Umsetzung der Schnittstelle für verschiedene Sensoren (AP3.1.b) wurde eine Leiterplatine entworfen die es ermöglicht verschiedene Sensorarten (vor allem die die in den Demonstratoren verwendet werden) an die Test Sensor Knoten Plattform anzuschließen (sehe Abbildung 8).

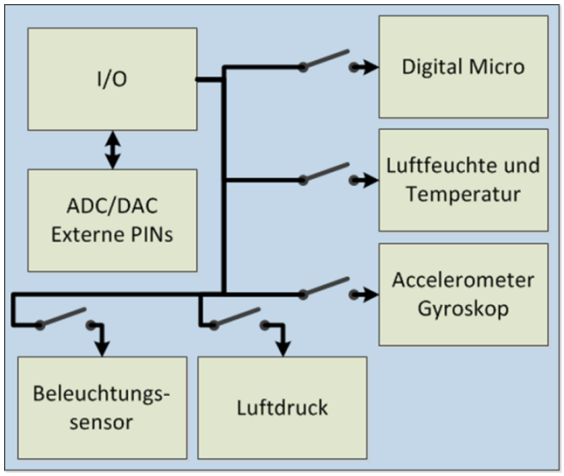

Abbildung 8: Die Sensorintegrationsplatine (Sensor Board)

AP3.2 Entwicklung Hardware für Sensorknotenplattform

In diesem Arbeitspacket haben wir uns mit der Entwicklung von Modulen der Sensorknotenplattform und mit deren Beschreibung für den Module Pool beschäftigt. Mit der Unterstützung von allen Partnern wurde eine Liste von Sensoren entworfen. Es wurden Adapterplatinen für den Anschluss dieser Sensoren an unsere Knoten entworfen.

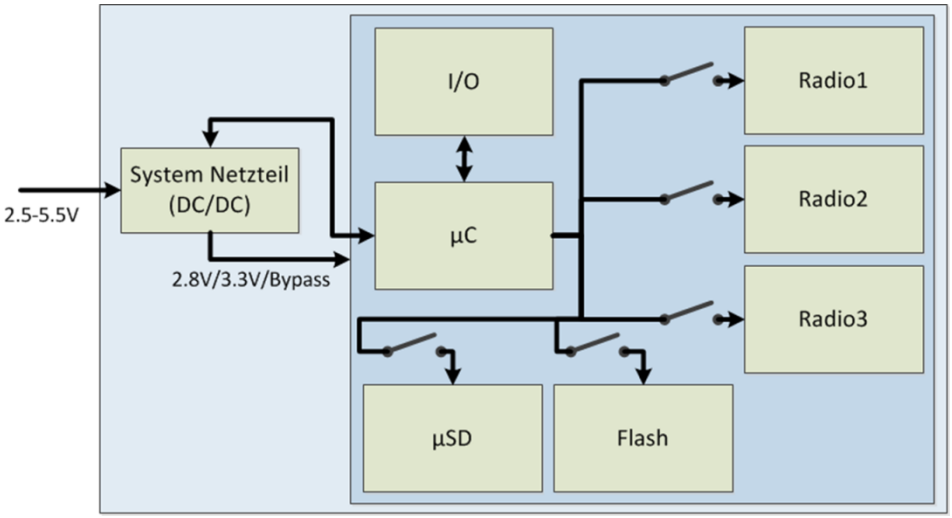

Abbildung 9: Die Architektur der Sensorknotenplattform

Abbildung 9 stellt die Architektur der neuentwickelten Sensorknotenplattform dar. Es ist eine Knotenbasisplatine mit einer Reihe von Konfigurationsmöglichkeiten die es erlaubt verschiedene Hardwarekonfigurationen zu erzeugen ohne dass es notwendig ist neue Hardware zu fertigen. Die Peripherien werden mit softwaregesteuerten Schaltern ein- oder ausgeschaltet. Die Flexibilität der Konfiguration ist auf die, auf der Platine verfügbaren Module begrenzt, aber diese Lösung ist zuverlässig und erlaubt auch bis zu einem gewissen Grad eine Re-Konfiguration, falls sich die Bedingungen zur Laufzeit ändern sollten. Damit ist die Plattform perfektes für einen Sens4U Proof-of-concept.

2.3 AP4 Netzwerk- und Systemmanagement

AP4.4 Implementierung Deploymentwerkzeug (DPT)

In diesem Arbeitspacket haben wir die Möglichkeiten für die Unterstützung der Nutzer während des Ausbringens des Sensornetzwerks untersucht. Das Ergebnis ist das Deployment Tool, welches die Kommunikationsbedingungen zu dem Zeitpunkt überwacht und eventuell Verbesserungen vorschlägt. Der Meilenstein M-I-3 wurde planmäßig erreicht.

Die Aufgabe wurde in drei Teilaufgaben untergeteilt. Diese entsprechen der Ebene der Bearbeitung der Daten, die die Kommunikationsbedingungen beschreiben.

In AP4.4.a haben wir uns mit dem Sammeln und Vorbearbeiten der Verbindungsdaten der einzelnen Knoten beschäftigt.

Für die Kommunikation zwischen dem Deployment Tool (DPT) und dem drahtlosen Sensornetz (WSN) wurde eine Kommunikationsschnittstelle auf der Basis von RESTful definiert. Diese webbasierte Schnittstelle stellt eine einheitliche Basis für den Zugriff sicher. Somit kann eine einfache Anbindung von mobilen Endgeräten sowie die Realisierung einer webbasierten Benutzeroberfläche gewährleistet werden. Das Protokoll, welches auf den Sensorknoten die benötigen Daten sammelt, wurde vollständig definiert und auf den Sensorknoten umgesetzt.

Um die Qualität der einzelnen Verbindungen bewerten zu können haben wir ein Verfahren entworfen, das auf dem F-LQE Verfahren basiert. Es berechnet Fuzzy-Logiken und liefert für jede Verbindung einen Qualitätsindikator (LQ).

In AP4.4.b haben wir Mechanismen entwickelt die diese Ergebnisse weiter bearbeiten. Für die Ableitung der Topologie-Hinweise wurde ein Algorithmus implementiert, welcher zunächst aus den gewonnenen Daten zur Verbindungsqualität einen gerichteten Graphen für das Ganze Sensornetzwerk generiert. Für die Gewichtung der Kanten wurden die LQ-Werte der einzelnen Verbindungen verwendet. Durch einen Minimum-Spanning-Tree-Algorithmus werden die günstigsten Verbindungen im gesamten Netz ermittelt. Die Analyse der gefilterten Verbindungen kann somit genutzt werden, um sehr schlechte Links und mögliche Alternativen zu ermitteln.

Zur besseren Ermittlung von Topologie-Hinweisen und um Verbesserungsvorschläge zu generieren ist es notwendig, Informationen aus dem Planungstool/Expertensystem zu kennen. Somit müssen die Konfigurationsdaten dem Deployment-Tool zur Verfügung stehen.

Der in AP1.2 vorgestellte Network-Analyser-Extender wurde teilweise auch unter AP4.4.b entwickelt. Die Ergebnisse der Analyse der Netzwerk Topologie in der Design Phase der Anwendung sind während des Ausbringesn vorteilhaft. In der Design Phase ist es möglich die Anzahl der notwendigen Repeater Knoten zu schätzen und damit die Kosten genauer zu planen. Danach wird das Deployment-Tool genutzt, um die Repeater Knoten richtig zu positionieren. Das Network-Analyser-Extender Werkzeug wird im Folgenden erläutert.

Die eigentliche Analyse des Netzwerks erfolgt durch eine Kaskade von Filtern, die GEXF/RuleML-Daten empfangen und aufbereitet oder angereichert wieder ausgeben.

Der erste Filter in der Kette ist die Distance-Analyse. Dieser Filter berechnet zwischen allen Messpunkten den Abstand und speichert diese in den Kanteninformationen. Die Ausgabe-GEXF/RuleML-Datei enthält also u. U. zusätzliche Kanten mit einem Entfernungsattribut.

Der zweite Filter ermittelt aus den Abständen die maximale Distanz zwischen zwei Messpunkten und legt diese in RuleML kodiert in der Ausgabedatei ab.

Der dritte Filter platziert die Datensenke im Messfeld. Ist ein Medienzugang (Ethernet, Stromquelle) vorhanden, wird die Senke dort platziert. Ansonsten wird der Mittelpunkt des umgrenzenden Polygons berechnet. Befindet sich dieser Punkt nicht innerhalb des Polygons, wird der Punkt im Polygon berechnet, der am dichtesten am Mittelpunkt liegt. Befindet sich der Punkt in einem verbotenen oder schwierigen Gebiet, wird der dichteste Punkt außerhalb des Gebietspolygons gewählt.

Die Datenratenanalyse ist im vierten Filter implementiert. Aus den Taskdefinitionen der einzelnen Messpunkte wird für jeden Messpunkt seine Ausgangsdatenrate ermittelt.

Filter 5 ermittelt aus diesen Werten die Summe und den Maximalwert und legt diese wie Filter 2 in seiner Ausgabedatei ab.

Die Topologieerstellung ist Aufgabe des sechstens Filters. Ohne Vorgaben platziert dieser Filter einen Sensor auf jeden Messpunkt und generiert eine Sterntopologie zwischen Datensenke und Sensorknoten.

Filter 7 führt eine Datenratenanalyse des Netzes durch. Ohne Vorgaben baut er ein zu den Datenraten und zur Topologie passendes Modell des Sensornetzes und berechnet anhand des Modells die entstehenden Datenraten entlang der Kanten und in den Knoten.

Schließlich haben wir in AP4.4.c eine GUI entwickelt, die dem Nutzer die Ergebnisse darstellt. Die Entwicklung der graphischen Benutzeroberfläche wurde sowohl für ein Android-System als auch für Standardbetriebssysteme, wie Windows und Linux, fertiggestellt. Somit ist es möglich, einen Standardrechner für die Analyse zu verwenden. Abbildung 10 und Abbildung 11 zeigen jeweils einen Screenshot der beiden Systeme. Bei der PC-Version ist es auch möglich, aus den Konfigurationsdaten des Expertensystems die Kartendaten und die geographische Lage der einzelnen Sensorknoten zu extrahieren.

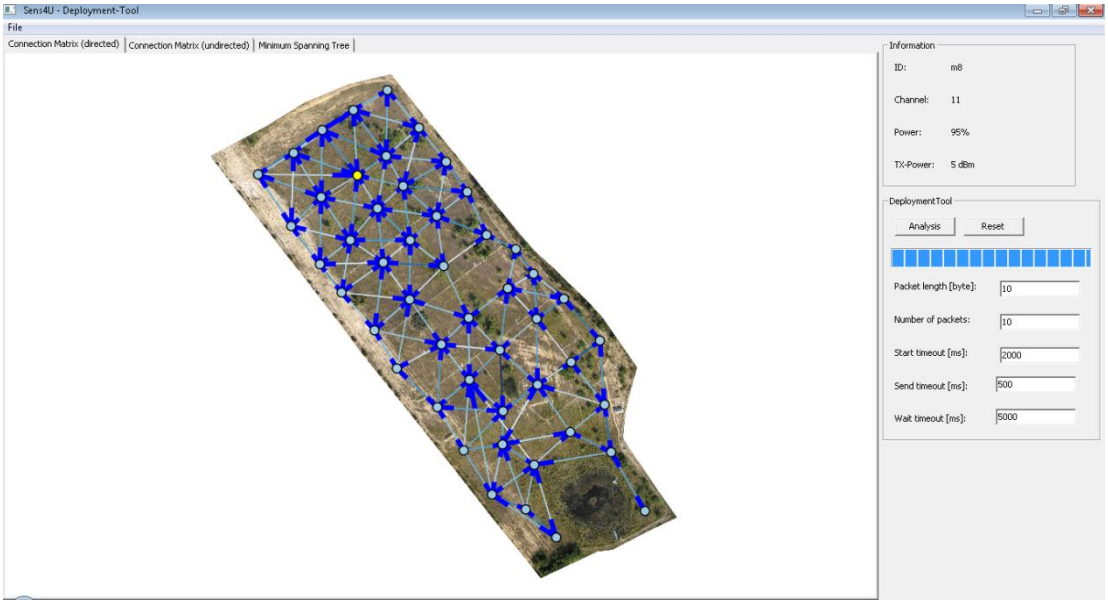

Abbildung 10: GUI des Deployment-Tools für Standardbetriebssysteme

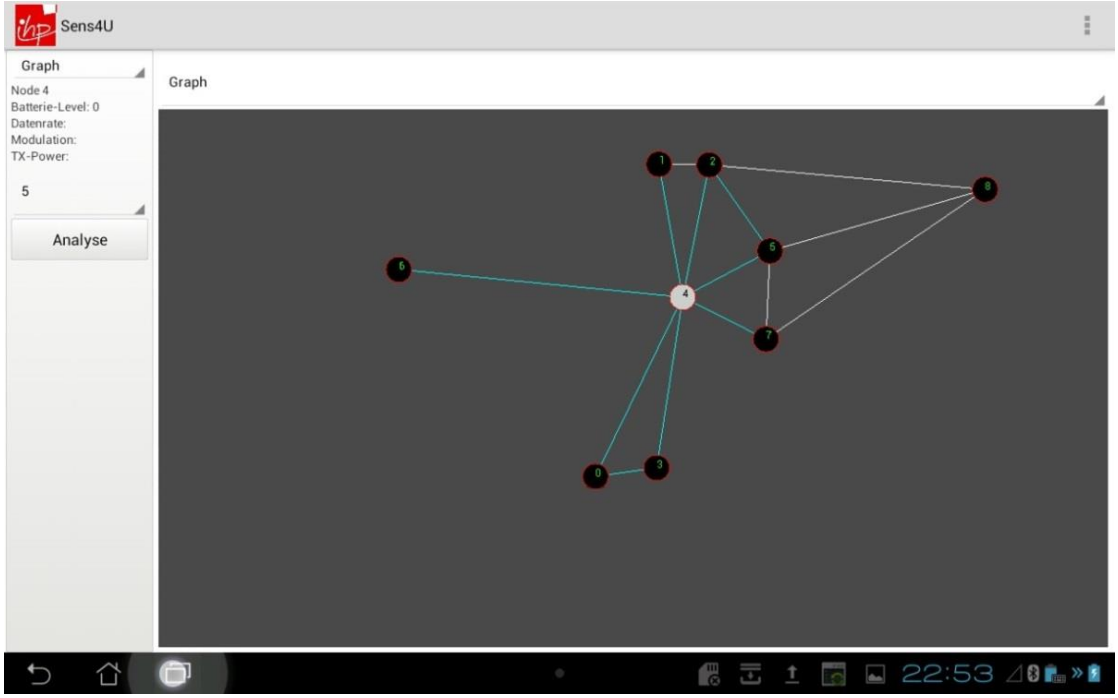

Abbildung 11: Graphische Oberfläche des Deployment-Tools für das Androidsystem

2.4 AP5 Power Management

AP5.4 Harvesting

In diesem Arbeitspacket haben wir die Möglichkeiten für Energieharvesting untersucht und die ausgewählten in Form eines Energieharvesting Integrationsmodules umgesetzt. Darüber hinaus, haben wir das Modul in das Gesamtsystem Integriert. Der Meilenstein M-I-2 wurde planmäßig erreicht.

Das Energieharvesting Integrationsmodul ist für die effiziente Anpassung der Spannungen aus unterschiedlichen Energiequellen wie auch für des Puffern der Energie zuständig. Abbildung 12 zeigt die Architektur des Modules (links) zusammen mit einem Sensorknoten der durch das Modul mit Energie versorgt wird.

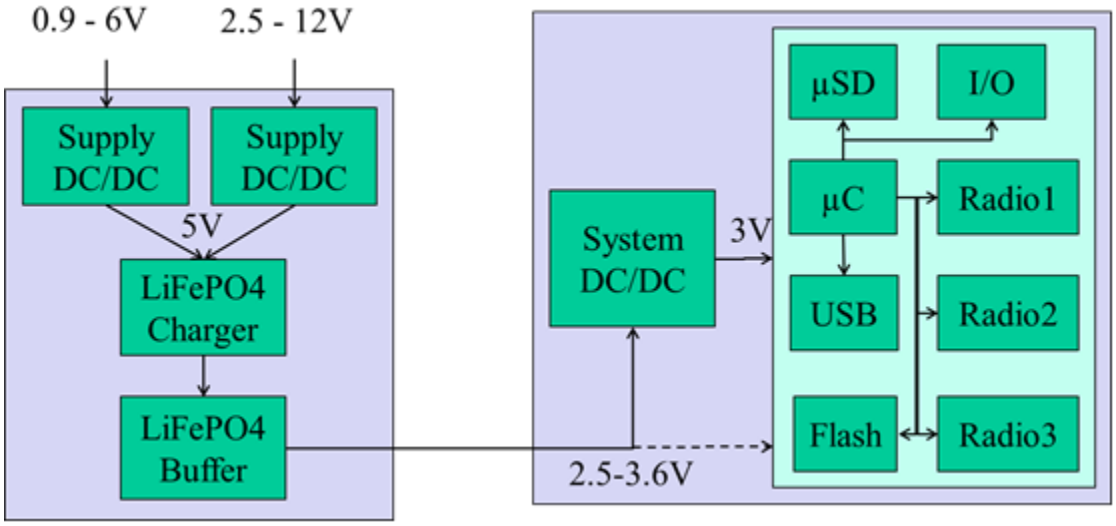

Abbildung 12: Energieharvester Modul und seine Verbindung mit dem Sensorknoten

2.5 AP6 Entwicklung Anwendungssoftware

AP6.1 Entwicklung der Datenbank

In Arbeitspaket AP6.1.e haben wir die Möglichkeitendie Knotenfunktionen in der Anwendung abzubilden untersucht. Diese Aufgabe ist stark mit AP6.2.d verbunden, da sie Mechanismen für die Konfiguration der Knoten aus der Anwendung berücksichtigt.

AP6.2 Entwicklung eines Anwendungsprogramms

In den Arbeitspaketen AP6.2.d und AP6.2.f haben wir untersucht, welche Möglichkeiten es gibt, Messszenarien zu definieren, aber auch zur Laufzeit zu konfigurieren. Wir haben verschiedene Ansätze für semi-automatische Softwareentwicklung untersucht, wie z. B. Java-basierte Knotensoftwareentwicklung oder Skriptsprachen. In unserem Ansatz haben wir uns für das Datenflusskonzept der BTU entschieden, das datenzentrisch und auch modulbasiert ist, was in

das Sens4U-Konzept sehr gut passt. Die Messaufgaben der Knoten werden mittels MTDL-Sprache in dem Planungs-Tool definiert und von dem Expertensystem in eine modulare Anwendungsschicht für jeden Knoten übersetzt.

2.6 AP7 Demonstrator

AP7.1 Planung und Aufbau der Demonstrator-1

In diesem Arbeitspaket haben wir den ersten Demonstrator spezifiziert. Wir haben uns entschieden die verfügbare Hardware Plattform zu nutzen – den IHP Knoten der in seiner ersten Version in dem FeuerWhere Projekt eingesetzt war. Wir haben untersucht wie gut sich diese Plattform für unsere Zwecke eignet. Wir haben uns auch entschieden eine abgespeckte Version der Sensor Integrationsplatine zu verwenden. Aus diesem Grund haben wir mit AP3.2 eher angefangen als ursprünglich geplant. Wir haben auch unterschiedliche Hardware Plattformen für das Gateway untersucht. Wir haben uns für die einfache und kostengünstige Raspberry PI Plattform entschieden. Aus Stabilitätsgründen haben wir aber diese später durch BeagleBoneBlack ersetzt.

AP7.2 Integration der Sens4U-Komponenten zum Demonstrator-2

Das Arbeitspaket beinhaltete alle Integrationsaufgaben des IHP. Hauptaufgabe war aber die Integration des Energieharvestermodules in den Demonstrator 2 was uns erfolgreich gelang. Abbildung 13 stellt den Aufbau der Plattform dar. Sie besteht aus dem Energieharvester Modul (unten), dem Sensorknoten Board und dem Sensorboard (oben).

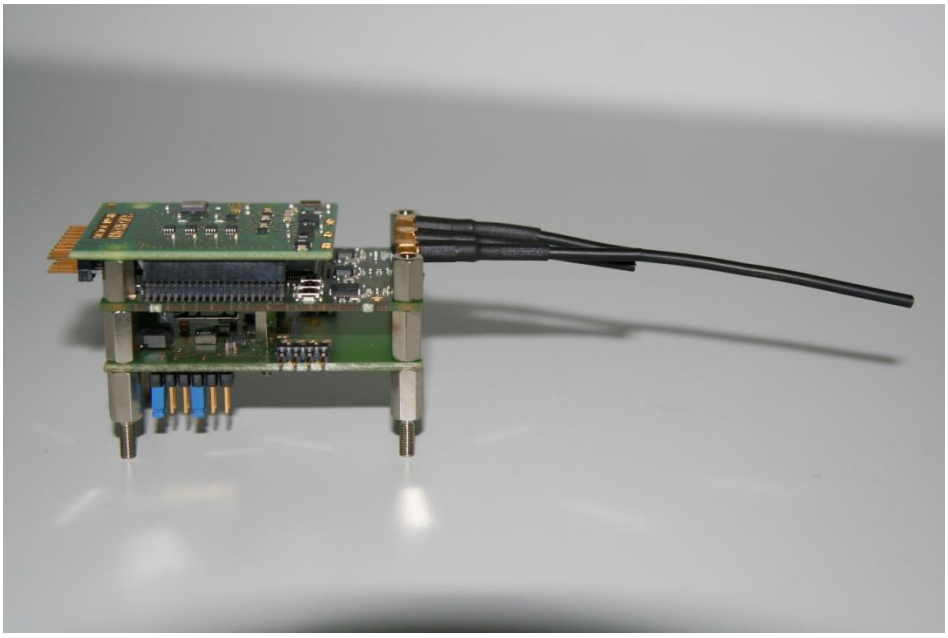

Abbildung 13: Der modularer Aufbau der Sensorknotenplattform

Die neuen Platinen sind mit der älteren Version des IHP Knoten pin-kompatibel. Diese konnten also einfach und eins-zu-eins ausgetauscht werden.

3 Notwendigkeit und Angemessenheit der geleisteten Arbeit

Für die im Rahmen dieses Projektes untersuchten Fragestellungen existieren keine anerkannten oder gar standardisierten Lösungen. Vielmehr wird in einigen Bereichen noch sehr intensiv an ähnlichen bzw. gleichartigen Fragestellungen geforscht. Die Arbeiten waren also aus wissenschaftlich-technischer Sicht notwendig, um Aussagen zur Umsetzbarkeit der Basisidee des Sens4U-Ansatzes machen zu können. Zu allen Arbeitspaketen wurden intensive Untersuchungen sowohl theoretischer Natur, wie die Berechnung der Komplexität von implementierten Verfahren, als auch experimenteller Art durchgeführt. Voraussichtlicher Nutzen

Eine Verwertung der Sens4U-Ergebnisse kann in unterschiedlichen Kontexten erfolgen. Die kommerzielle Vermarktung der Ergebnisse ist dabei eher langfristig zu sehen.

Wir haben mit anderen Partnern die Möglichkeit einer Vermarktung der Sens4U-Idee untersucht. Alle Partner sind an einer weiteren Kooperation nach dem Projektende interessiert. Die genaue Form der Kooperation ist noch offen. Die Möglichkeiten beinhalten z. B. eine Ausgründung mit der Sens4U-Kernkompetenz. Die Ergebnisse müssen aber noch weiterentwickelt werden, um die notwendige Marktreife zu erreichen. Das Konsortium hat gemeinsam die Ergebnisse des Projektes bei der Messe „Zukunftstechnologietage 2013“ am 14. und 15. November 2013 in Cottbus präsentiert.

Das IHP wird die im Projekt entwickelten Hardware- und Softwaremodule für weitere Forschungsaktivitäten verwenden. Die Sensorknoten und Energy-Harvester-Module werden auch später in einem Testbed verwendet, mit dem Lösungen für drahtlose Sensornetze getestet werden können. Das Testbed wird ebenfalls für Bildungszwecke für Studenten genutzt.

In Rahmen des ECOPOL Projekt haben wir auch untersucht welche Vermarktungsmöglichkeiten es für Sens4U gibt. Wir hatten uns an der **Ideenschmiede 2013** –ausgerichtet von der IHK Ostbrandenburg - teilgenommen.

Zusammen mit der Zukunftsagentur Brandenburg GmbH (ZAB) und dem Enterprise Europe Network Berlin-Brandenburg haben wir eine Kooperation unter Miteinbeziehung von Studenten aus dem Studienmodul „Management des internationalen Technologietransfers“ der Wirtschaftswissenschaftlichen Fakultät der Stiftung Europa-Universität Viadrina Frankfurt (Oder) über die **Erstellung und ggf. Vorbereitung für die Umsetzung einer Studie zur Vermarktung und Internationalisierung unserer Idee** vereinbart. Die Gruppe von Studenten hat die Möglichkeiten einer Vermarktung analysiert und einen initialen Businessplan erstellt.

Das IHP vertreten durch Herrn Dr.Ing. Krzysztof Piotrowski hat außerdem an einem Workshop in Rahmen des Projektes „Innovationen brauchen Mut“ teilgenommen. Bei dem Workshop wurden auch Wirtschaftliche und Marktorientierte Aspekte einer Vermarktung mit professionellen Beratern diskutiert und analysiert.

Die Markt- und Vermarktungsorientierte Aktivitäten waren für uns sehr interessant. Oft werden solche Aspekte bei der Forschung nicht berücksichtigt, was ein großer Fehler ist.

4 FE-Ergebnisse von dritter Seite

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

5 Veröffentlichungen des Ergebnisses

Während der Projektlaufzeit sind die folgenden wissenschaftlichen Veröffentlichungen entstanden:

K. Piotrowski, *Entwicklung von Sensornetzanwendungen zum Umweltmonitoring im Lego-Style,* Proc. SensorSysteme 2014 - 18th Leibniz Conference on Advanced Science, abstr. 14, (2014)

St. Lange, J. Lösche, K. Piotrowski, *Tool-Supported Requirements-Based Topology Design for Wireless Sensor Networks*, Proc. 3rd International Conference on Wireless Sensor Networks (WSN'14), 1049 (2014)

K. Piotrowski, J. Lösche, *Sens4U: A Modular Approach Towards the Ideal Sensor Node Software and Hardware*, Proc. 12. GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze", 33 (2013)

M. Brzozowski, P. Langendörfer, *Is Cross-Platform Protocol Stack Suitable for Sensor Networks? Empirical Evaluation*, Proc. 6th Join IFIP Wireless & Mobile Networking Conference (WMNC 2013), (2013)

M. Brzozowski, P. Langendörfer, *Overview and Benchmarks of Pragmatic Debugging Techniques for Wireless Sensor Networks*, Proc. of the 21st International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2013), (2013)

O. Maye, M. Maaser, *Comparing Java Virtual Machines for Sensor Nodes - First Glance: Takatuka and Darjeeling,* Proc. of the 8th International Conference on Grid and Pervasive Computing (GPC 2013) (2013)

K. Piotrowski, St. Peter, *Sens4U: Wireless Sensor Network Applications for Environment Monitoring Made Easy*, Proc. 4th International Workshop on Software Engineering for Sensor Network Applications (SESENA 2013), (2013)

6 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] 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

[3] 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.

[4] 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.

[5] P. Costa, L. Mottola, A. Murphy, and G. Picco. Programming Wireless Sensor Networks with the Teeny Lime Middleware. Middleware 2007, pages 429-449, 2007.
[6] 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.
[7] Paolo Costa, Luca Mottola, Amy L. Murphy, and Gian Pietro Picco. Developing sensor network applications using the teenylime middleware. Technical report, 2006.
[8] 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.
[9] 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
[10] P. Buonadonna, D. Gay, J. Hellerstein, W. Hong and S. Madden. TASK: Sensor Network in a Box. In *Proc. of the 2nd European Workshop on Sensor Networks (EWSN 2005)*, pages 133--144, Istanbul, Turkey, 2005.
[11] E. Cheong, E. A. Lee and Y. Zhao. Viptos: A Graphical Development and Simulation Environment for TinyOS-based Wireless Sensor Networks. In *Proceedings of the Third ACM Conference on Embedded Networked SensorySystems*, 2005.
[12] S. Cotterell, R. Mannion, F. Vahid and H. Hsieh. eBlocks: an enabling technology for basic sensor based systems. In *Proceedings of the 4th international symposium on Information processing in sensor networks*.
[13] RTCA Inc., SOFTWARE CONSIDERATIONS IN AIRBORNESYSTEMS AND EQUIPMENT CERTIFICATION.
[14] B. Greenstein, E. Kohler and D. Estrin. A sensor network application construction kit (SNACK). In *Proc. of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys)*, ACM, pages 69--80, 2004.
[15] Methodology for the development, test, deployment, and validation of wireless sensor networks, http://www.btnode.ethz.ch/Projects/Jaws
[16] T. Liu and M. Martonosi. Impala: a middleware system for managing autonomic, parallel sensor systems. In *Proceedings of the ninth ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 107--118, 2003.
[17] Experimental wireless sensor network testbed, http://motelab.eecs.harvard.edu/
[18] Discreteeventsimulationenvironment, http://www.omnetpp.org/
[19] F.J. Oppermann and S. Peter. Inferring Technical Constraints of a Wireless Sensor Network Application from End-User Requirements. In *2010 Sixth International Conference on Mobile Ad-hoc and Sensor Networks*, pages 169--175, 2010.
[20] A. Pathak, Q. Zhou and V. K. Prasanna. Srijan: A Graphical Toolkit for WSN Application Development. In *The 4th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS '08)*, 2008.
[21] S. Peter, K. Piotrowski and P. Langendörfer. In-Network-Aggregation as Case Study for a Support Tool Reducing the Complexity of Designing Secure Wireless Sensor Networks. In *Proc. of the Third IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*, 2008.
[22] K. Römer and F. Mattern. The Design Space of Wireless Sensor Networks. *IEEE Wireless Communications*, vol. 11(6), pages 54--61, December 2004.
[23] S. Tilak, N. B. Abu-Ghazaleh and W. Heinzelman. A taxonomy of wireless micro-sensor network models. *SIGMOBILE Mobile Computing and Communication Review*, vol. 6(2), pages 28--36, 2002.
[24] http://www.cs.berkeley.edu/~pal/research/tossim.html

III. Sachlich richtig

Der hier vorgelegte Abschlussbericht entspricht dem tatsächlichen Projektstand. Die dargestellten Ergebnisse werden hiermit bestätigt.

_____

Ort, Datum, Unterschrift