T U M

I N S T I T U T F Ü R I N F O R M A T I K

Schlussbericht AmbiComp: Software Entwicklung mit universell kombinierbaren Ambient Intelligence Komponenten

Thomas Fuhrmann (Hrsg.)

TUM-I1022
Juni 10

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

TUM-INFO-06-I1022-0/1.-FI

Druck: Institut für Informatik der
Technischen Universität München

Thomas Fuhrmann (Hrsg.)

Institut für Informatik, Technische Universität München

Schlussbericht

AmbiComp

Software Entwicklung mit universell kombinierbaren Ambient Intelligence Komponenten

Juni 2010

Das diesem Bericht zugrunde liegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01ISF05 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.

AmbiComp war ein Verbundprojekt der Technischen Universität München, der Hochschule der Medien, des Fraunhofer Instituts für Experimentelles Software-Engineering, der Beecon GmbH und der Alcatel-Lucent Deutschland AG. Dieser Schlussbericht wurde auf Grundlage der Zwischenberichte und anderer Materialen der beteiligten Projektpartner von Dr. Thomas Fuhrmann zusammengestellt. Als Autoren haben die folgenden Personen mitgewirkt:

Technische Universität München

Dr. Thomas Fuhrmann (Projektkoordinator)
Björn Saballus
Johannes Eickhold

Hochschule der Medien, Stuttgart

Prof. Dr. Johannes Maucher
Jürgen Butz
Thomas Suchy

Fraunhofer IESE, Kaiserslautern

Daniel Schneider
Christian Peper

Beecon GmbH, München

Dr. Thomas Fuhrmann
Sven Schlender

Alcatel-Lucent Deutschland AG, Stuttgart

Harald Orlamünder
Arnold Eppler

Wir danken dem Bundesministerium für Bildung und Forschung für die großzügige Unterstützung des Projekts, sowie dem Deutschen Zentrum für Luft- und Raumfahrt, insbesondere Herrn Roland Mader für die kompetente Begleitung der Projektdurchführung.

AmbiComp Abschlussbericht
Teil 1 – Kurzdarstellung

1. Aufgabenstellung

Das vom Bundesministerium für Bildung und Forschung unter Nummer 01ISF05 geförderte Projekt „AmbiComp – Software Entwicklung mit universell kombinierbaren Ambient Intelligence Komponenten“ zielte darauf ab, das Gebiet des Software-Engineerings (SE) für verteilte Systeme mit anwendungsnaher Forschung im Bereich der Ambient Intelligence zu verbinden. Dementsprechend waren sowohl der allgemeine Erkenntnisgewinn als auch dessen Umsetzung zur Nutzbarmachung in der Software-Entwicklung Ziele des Projektes.

Im Einzelnen sollten folgende Ziele umgesetzt werden:

1. Ambient Intelligence sollte durch das Zusammenwirken vieler Einzelgeräte in einem System erzeugt und nicht im Vorhinein konstruiert werden.
2. Auf Grundlage des von Dr. Thomas Fuhrmann entwickelten selbstorganisierenden Wegewahlverfahrens (Scalable Source Routing, SSR) sollten Nachrichten in einer beliebigen Netzwerktopologie ohne zusätzliches Konfigurieren der Geräte zugestellt werden. Dieses Wegewahlverfahren sollte zum dynamischen Auffinden und Binden von Diensten verwendet werden. Hierzu sollten die Selbstorganisation und Ressourceneffizienz von SSR in den Bereich der verteilten Systeme übertragen werden. Auf SSR aufbauend sollte das Gesamtsystem inhärent Code-, Objekt- und Threadmigration zwischen verschiedenen Geräten ohne zentrale Instanz bereitstellen und so eine einfache Programmierung ermöglichen.
3. Um das Gesamtsystem für die Entwicklung und den Einsatz realer Anwendungen nutzbar zu machen, sollte ein „Hardwaresubstrat“, die so genannte „Ambient Intelligence Control Unit“ (AICU) entwickelt werden. Diese AICU sollte aus Hardware (Mikrocontroller plus externer Beschaltung) und der fest eingebrachten Software, der sogenannten „Firmware“, bestehen. Diese Firmware sollte als minimales Betriebssystem agieren, bestehend aus grundlegenden Hardware-Treibern, einer Java Ausführungsumgebung (Java Virtual Machine, JVM) und weiteren Anteilen wie z.B. den Kommunikationsprotokollen. In diese Firmware sollte SSR so integriert werden, dass sowohl lokal als auch auf entfernten Geräten auf Code, Objekte und Threads zugegriffen werden kann.
4. Die Hardware, die die Grundlage für die AICUs darstellt, sollte von der Beecon GmbH entwickelt werden. Dabei war ein Projektziel, eine vollständige Spezifikation der AICUs zu schaffen und am Ende des Projektes zu veröffentlichen. Die Einhaltung dieser Spezifikation sollte durch Konformitätstests zur Qualitätssicherung, während der Projektlaufzeit und darüber hinaus, gesichert werden. Ein Logo sollte Produkten Dritter bestätigen, dass ihre Geräte die Konformität erfüllen.
5. Um AICUs in realen Geräten Dritter zu nutzen, sollten Geräteschnittstellen zu den jeweiligen Sensoren und Aktoren geschaffen werden, die nach oben zur Anwendung hin möglichst plattformübergreifend sein sollten. Diese Schnittstellen sollten in konkreten Beispielen Anwendungen und Demonstratoren exemplarisch umgesetzt werden.
6. In Anlehnung an das Konzept der „klassischen“ Parallelrechnerprogrammierung sollten die Konzepte des Software-Engineerings für verteilte Systeme weiter entwickelt und in Fachveröffentlichungen publiziert werden. Hieraus sollte Schulungsmaterial geschafft werden, das es Software-Entwicklern ermöglicht, anhand von Beispielanwendungen eigene Systeme zu entwickeln.
7. Um Software-Entwicklern eine schnelle Produktivität zu ermöglichen, sollte auf Basis des Open Source Werkzeugs „Eclipse“ eine Entwicklungsumgebung geschaffen werden.

Diese sollte es ermöglichen, rasch Ambient Intelligence Anwendungen zu entwickeln und zu testen.

8. Das Zusammenspiel vieler Einzelgeräte mit eingebetteten AICUs sollte durch Benutzerschnittstellen auf Basis von Handys und „Persönlichen Digitalen Assistenten“ (PDAs) ergänzt werden.

2. Voraussetzungen, unter denen das Vorhaben durchgeführt wurde

Durch die lange Verzögerung bis zum Start des Projektes haben sich die Rahmenbedingungen für die Projektdurchführung im Vergleich zur ursprünglichen Planung grundlegend geändert.

Die ursprüngliche Projektskizze zu AmbiComp entstand Anfang 2005 aufgrund der gemeinsamen Vorarbeiten der Beecon GmbH, der NetAds Gbr und der Hochschule der Medien (HdM), dort maßgeblich von Ansgar Gerlicher unterstützt. Durch die lange Zeit bis zum Projektbeginn im Sommer bzw. Herbst 2006 hatten sich wesentlichen Änderungen in dieser Partnerkonstellation ergeben: Herr Gerlicher hatte die HdM verlassen. NetAds musste den Geschäftsbetrieb einstellen. Die dort maßgeblichen Personen, Herr Frech und Herr Iffland, standen dem Projekt anschließend nicht mehr zur Verfügung. Bei der Beecon GmbH ist der Mitarbeiter Till Harbaum ebenfalls noch vor dem Projektstart aus der Firma ausgeschieden. Herr Harbaum war der Mitarbeiter, der die Hardwareentwicklung bei der Beecon maßgeblich vorangetrieben hat und stark in die Entwicklung der Kommunikationsstacks involviert war.

Durch das Ausscheiden dieser Personen ist ein großer Teil der personellen Grundlage und damit des Know-Hows des Projektes weggebrochen. Dieses Know-How musste erst wieder neu aufgebaut werden. Hierfür mussten bei den Partnern neue Mitarbeiter eingestellt werden, die sich von Grund auf neu einarbeiten mussten.

NedAds wurde im AmbiComp Konsortium durch die Alcatel SEL AG ersetzt. Alcatel unterhielt zum Zeitpunkt des Projektstarts eine Abteilung „Zentrale Unternehmensentwicklung“, die neben dem planerischen Teil z.B. für Marktstudien, auch einen praktischen Teil enthielt, der Demonstratoren aufbaute. Zudem gab es mit dem sogenannten „3G Reality Center“ eine Einrichtung, in der externen Kunden und Partnern neue Entwicklungen gezeigt und sogar die Mitarbeit ermöglicht wurde. AmbiComp hätte von beiden Einrichtungen profitieren können. Leider haben sich innerhalb der Laufzeit die Randbedingungen bei Alcatel so verändert, dass die meisten der ursprünglichen Aufgaben nicht wahrgenommen werden konnten. Alcatel und Lucent fusionierten. Dabei entstand sehr viel Unsicherheit im Konzern, und die meisten mit AmbiComp befassten Personen verließen das Unternehmen. Durch die Auflösung der „Zentralen Unternehmensentwicklung“ und des „3G Reality Centers“ fielen Schlüsselbereiche weg, was besonders die Themen „Industriekontakte“, „Beispielkunden“ und „Beispielanwendungen“ negativ beeinflusste.

Zusätzlich wechselte die Arbeitsgruppe von Dr. Thomas Fuhrmann während der Projektlaufzeit von der Universität Karlsruhe an die TU München.

3. Planung und Ablauf des Vorhabens

Durch die lange Anlaufphase des Projektes haben sich alle Arbeitspakete nach hinten verschoben. Dies hat zu einer (kostenneutralen) Verlängerung des Projektes um ein halbes Jahr geführt. Weiterhin wurden einige Arbeitsteile in ihrer Komplexität unterschätzt, so dass viele Arbeiten nicht im ursprünglich geplanten Umfang realisiert werden konnten.

Im Einzelnen wurden folgende Arbeiten durchgeführt:

Die Beecon GmbH hat mit der Entwicklung mehrerer sogenannter Sandwichmodule (SMs) begonnen, die in beliebigen Kombinationen zu einer AICU zusammengesteckt werden können. Es wurden mehrere Generationen SMs in verschiedenen Ausprägungen entwickelt.

Teilweise waren diese SMs mit einem 8-Bit-Mikrocontroller ausgerüstet (intelligente SMs) und mit verschiedenen Kommunikationsschnittstellen versehen. Diese SMs bildeten die Grundlage für sämtliche weiteren Schritte bei der Software-Entwicklung. Für jedes dieser SMs wurde ein Datenblatt in englischer und deutscher Sprache erstellt, welche jeweils auf der AmbiComp Webseite zum Download bereit stehen.

Während der Projektlaufzeit wurden der Projektplan und die Arbeitspakete Aufgrund von geänderten Rahmenbedingungen wiederholt angepasst. Hierzu gehörte, dass während der Laufzeit festgestellt wurde, dass eine Hardware-Abstraktionsschicht für die verschiedenen Hardwareplattformen zur einfachen Anpassung der Java Laufzeitumgebung von Vorteil ist. Diese besondere Firmware, das so genannte BIOS, kapselt den Zugriff auf die Hardware und stellt diesen über eine einheitliche Schnittstelle für Anwendungen zur Verfügung. Das BIOS wurde in mehreren Iterationsstufen so angepasst, dass es für den Zugriff auf Ressourcen innerhalb eines AICUs aber über SM Grenzen hinweg, optimal zugeschnitten war. Hieraus entstand der so genannte *schnelle Speicherzugriff* (*Fast Memory Access*, FMA) der den Zugriff auf den gesamten Speicher innerhalb eines AICUs ermöglicht. Auf Basis dieses Zugriffverfahrens wurden in Zusammenarbeit der Beecon GmbH, der TU München und dem Fraunhofer IESE erste Ansätze eines allgemeinen *Object Distribution Mo*del (ODM) entwickelt, das auch über die Projektlaufzeit hinaus in den bereits genehmigten Nachfolgeprojekten „Sovereign“ und „J-Cell“ verwendet und weiter entwickelt wird.

Ein wesentlicher Bestandteil des Projektes war die Java-Laufzeitumgebung, die sogenannte *AmbiComp Virtuelle Maschine* (ACVM). Als Grundlage für die ACVM diente eine bei der Beecon bereits vorab entwickelte Java Virtuelle Maschine für eingebettete Systeme. Diese wurde im Laufe des Projektes weitgehend überarbeitet und an vielen Stellen aus Effizienzgründen komplett neu implementiert. Um die ACVM möglichst effizient auf den 8-Bit-Mikrocontrollern der intelligenten SMs auszuführen, wurde der herkömmliche Java-Bytecode von dem sogenannten Transcoder in ein effizienteres Zwischenformat übersetzt. Dadurch wurde es möglich, das ACVM-Speicherabbild so klein wie möglich zu halten.

Um die Entwicklung und das Debugging der ACVM zu vereinfachen sowie um die Plattformkonformität mit reproduzierbaren Tests sicherzustellen, wurde an der Uni Karlsruhe/TU München ein Linux-Emulator entwickelt. Hauptbestandteil dieses Emulators ist das Linux-Emulations-BIOS und eine Linux-ACVM, die aus dem Eclipse-Plugin heraus über den sogenannten *ACVM-Runner* gestartet werden konnte. Dieser ACVM-Runner ist ein Hilfsprogramm, das zusätzlich für die automatisierten Tests verwendet wird. Das Linux-Emulations-BIOS emuliert die reale Hardware und wird von der ACVM genauso verwendet wie das BIOS auf der echten AICU-Hardware. Dadurch entsteht der Vorteil, dass aufgrund der BIOS-Hardwareabstraktion der Quellcode für die ACVM derselbe ist, der auch für die reale Hardware kompiliert wird und dadurch nur einmal gepflegt werden muss. Dies erleichtert die Entwicklung und die Fehlerbehebung erheblich und macht den gesamten Entwicklungsprozess wesentlich weniger fehleranfällig.

Damit die ACVM ein Java-Programm ausführen konnte, musste dieses Programm zuerst die AmbiComp-Toolchain durchlaufen. Diese Toolchain besteht aus Java Compiler, Transcoder, ACVM und den zusätzlich benötigten Hilfswerkzeugen um z.B. die ACVM in den Speicher der SMs zu übertragen. Im ersten Schritt wurde hierzu der kompilierte Programmcode zusammen mit den benötigten Java-Bibliotheksfunktionen in das ACVM spezifische *Binary Large Object* (Blob) Zwischenformat transcodiert und statisch gebunden. Der daraus entstandene Blob konnte anschließend entweder von der ACVM unter Linux oder auf der realen Hardware ausgeführt werden. Um den Blob auf der Hardware auszuführen, musste dieser noch in den Speicher der Hardware übertragen („deployed“) werden. Anfangs geschah dies per serielle Schnittstelle, später per I2C Bus. Der hierfür benötigte Transcoder wurde an der Uni Karlsruhe/TU München entwickelt. Dort entstanden während der Projektlaufzeit auch die sogenannten *Compliance Tests*, welche die einzelnen Komponenten der Toolchain sowie deren Zusammenspiel kontinuierlich testeten und dadurch eine gleichbleibend hohe Code-qualität der AmbiComp-Software gewährleisteten.

An der Hochschule der Medien (HdM) wurde das AmbiComp Eclipse-Plugin entwickelt und fortlaufend verbessert, optimiert und während der gesamten Laufzeit erweitert und jeweils an die geänderten Gegebenheiten, wie die Einführung des BIOS, angepasst. Dazu wurde die AmbiComp Toolchain in dieses Plugin integriert bzw. daran angebunden. Es wurden verschiedene Sichten (Views) entwickelt, die es dem Entwickler ermöglichen, schnell AmbiComp Anwendungen zu entwickeln, diese im Linux-Emulator zu testen und anschließend auf die reale Hardware aufzuspielen (Deployment). Das Eclipse-Plugin steht auf der AmibComp Webseite zum Download zur Verfügung. Aus diesen Arbeiten entstand außerdem das Eclipse-Plugin Datenblatt welches, wie auch die anderen Datenblätter, zum Download auf der AmbiComp Webseite zur Verfügung steht.

Daneben war die HdM maßgeblich bei den Diskussionen zu den Systembibliotheken (Java-APIs) und speziell der Entwicklung der AmbiComp-API beteiligt. Hierauf aufbauend wurden an der HdM mehrere Beispieldemonstratoren entwickelt, z.B. der Java-Treiber für ein LCD Display oder die Client-Server Kommunikation über eine Bluetooth- bzw. Ethernet-Kommunikationsstrecke. Bei diesen Arbeiten lieferte die HdM immer wieder wertvolle Beiträge und Verbesserungsvorschläge zu den jeweiligen Entwicklungen bei der Beecon GmbH und der Uni Karlsruhe/TU München wie z.B. Bug Reports zu Fehlern in der ACVM oder dem BIOS.

Das Fraunhofer IESE konzentrierte sich auf die Untersuchung und Definition geeigneter Entwicklungsmethoden für AmbiComp Systeme. Ein wesentliches Charakteristikum solcher Systeme ist ihre Fähigkeit sich ad hoc zu vernetzen, um höherwertige Dienste anbieten zu können. Dazu müssen solche Systeme flexibel sein und adaptiv auf sich ändernde Anforderungen und sich ändernde Verfügbarkeit von Ressourcen, Geräten und Diensten reagieren können. Dies bedingt verschiedene Herausforderungen hinsichtlich geeigneter Entwicklungsansätze.

Als Lösungsansatz und Kernaspekt einer Entwicklungsmethode für ambient intelligente Systeme entwickelte das Fraunhofer IESE im Projektverlauf ein Komponentenmodell für Ad hoc Systeme. Dieses Komponentenmodell sollte den Entwurf dynamisch-adaptiver Systeme durch Definition geeigneter Architekturen, Schnittstellen und Infrastruktur-Komponenten unterstützen. Um das Modell als Projekt-Teilergebnis auch für andere Anwendungskontexte verfügbar zu machen, wurde dabei besonders auf die Unabhängigkeit von konkreten Technologien (wie etwa C oder Java) geachtet.

Neben dem Komponentenmodell hat das Fraunhofer IESE verschiedene Demonstratoren und Simulatoren zur Evaluation der Ergebnisse realisiert. Insbesondere der „Feuermelder-Demonstrator“ wurde über den gesamten Projektverlauf hinweg in verschiedenen Varianten umgesetzt – von rudimentären ersten Schritten bis hin zum umfassenden CeBIT Demonstrator. Dabei wurden in enger Zusammenarbeit mit den Partnern die jeweils aktuellen Bausteine aus dem Projekt verwendet. Neben der damit einhergehenden Erprobung der Bausteine (Hardware, ACVM, API, Tools) hat das Fraunhofer IESE auch direkte Unterstützung bei der Entwicklung der Bluetooth API geleistet.

Gegen Ende des Projektes hatte sich das Fraunhofer IESE vermehrt der Erstellung von Informations- und Lehrmaterial gewidmet. In Zusammenarbeit mit den Partnern wurden Materialien wie Plakate, Flyer, Präsentationen, Flash-Filme und Tutorials erstellt. Plattformen für die Materialien waren die Präsentation auf der CeBIT, die AmbiComp Homepage, und das gegen Projektende neu geschaffene Wiki für „externe Interessenten“.

Bei der Alcatel-Lucent Deutschland AG wurden in Hinblick auf die zukünftige Verwertung Überlegungen zu den Rollen in der Wertschöpfungskette angestellt und Diskussionen mit den Projektpartnern geführt. Dieser Schritt war wichtig, um einerseits ein gemeinsames Verständnis zu erzielen und um andererseits die verschiedenen Zielgruppen (Hersteller von HW und Basis-SW, Anwendungs-Entwickler, Vertriebsorganisationen, usw.) entsprechend adressieren zu können.

Schon frühzeitig im Entwicklungsprozess hat Alcatel-Lucent die ersten Ergebnisse ausführlichen Tests unterzogen, beispielsweise das Eclipse-Plugin, die jeweils aktuelle

Hardware inklusive ACVM und die jeweiligen Demonstratoren.

Im späteren Projektverlauf hat Alcatel-Lucent Teile der Dokumentation übernommen und die Datenblätter für die verschiedenen SMs erstellt. Des Weiteren hat Alcatel-Lucent Unterstützung im Bereich der Messevorbereitung geleistet.

4. wissenschaftlicher und technischer Stand, an den angeknüpft wurde

„Ambient Intelligence“ war zu Beginn des Projektes ein vergleichsweise junges Gebiet, das sich im Bereich der Grundlagenforschung bewegte. An diesem Umstand hat sich auch bis zum Ende der Projektlaufzeit wenig geändert. Es gibt mittlerweile einzelne Produkte wie z.B. die „Sun SPOTS“ der Firma SUN Microsystems Inc. (jetzt Oracle) oder die „Particles“ der Firma Particle Computer GmbH, einer SAP Tochter.

Bei den Sun SPOTS handelt es sich um kleine, in Java programmierbare, Sensorknoten auf Basis eines ARM Prozessors mit einigen Megabyte Speicher. Im Vergleich zu den AmbiComp AICUs sind die Sun SPOTS damit deutlich leistungsfähiger, aber auch größer und teurer.

Bei den Particles handelt es sich um kleine Sensorknoten, welche mit einem 8-Bit PIC Prozessor, einer properitären Funkschnittstelle und verschiedenen Sensoren ausgestattet sind und eine properitäre Virtuelle Maschine ausführen. Die Programmierung erfolgt in einer eigenen, zu VisualBasic ähnlichen Programmiersprache, *ParticleBasic*, welche von einem eigenen Compiler in einen speziellen Bytecode übersetzt wird.

Sowohl Sun SPOTS als auch Particles verfügen über keine umfangreiche und transparente Unterstützung einer verteilten Nutzung wie sie in AmbiComp geplant war. Beide Produkte erreichten während der Projektlaufzeit keine weite Marktdurchdringung und die Particle Computer GmbH ist mittlerweile nicht mehr am Markt aktiv.

Zu Beginn des Projektes gab es einige Komponenten, die Teil einer „Ambient Intelligence“ Umgebung hätten sein können. Hierzu gehörten beispielsweise das von Philips entwickelte „Showline Media Center“, welches von einer zentralen Komponente aus Geräten in der ganzen Wohnung ansteuerte, oder das Busch-Jäger „Powernet“, das über einen „European Installation Bus“ (EIB) verschiedene Komponenten der Gebäudeautomation im Haus steuerte. Beide Produkte wurden weiter entwickelt und können derzeit als kommerzielle Produkte im Handel erworben werden. Im Vergleich zu AmbiComp bieten sie aber keine dezentrale verteilte Haussteuerung oder Methoden zur selbstorganisierten Vernetzung.

Es gab verschiedene Technologien, die aber nicht mit den im AmbiComp Projekt zu entwickelnden Techniken zu vergleichen waren. Zu diesen gehörten „Web-Services“, die einen Client-Server-Ansatz verfolgten, „Universal Plug and Play“ (UPnP) zur automatischen Vernetzung verschiedener Geräte von unterschiedlichen Herstellern, sowie Sensornetze zur Datengewinnung. Zusätzlich existierte noch die „Open Services Gateway initiative“ (OSGi). OSGi spezifiziert eine hardware-unabhängige dynamische Softwareplattform, die es erleichtert, Anwendungen und ihre Dienste per Komponentenmodell („Bundle“ und „Service“) zu modularisieren und zu verwalten („Service Registry“).

Zu den vorangegangenen Arbeiten der Antragsteller gehörte das von der Beecon GmbH zusammen mit NetAds GbR entwickelte eingebettete System, das BTKit. Das BTKit ermöglichte die einfache Programmierung von Kommunikationsfunktionalitäten in eingebetteten Systemen auch durch weniger erfahrene Programmierer. Es wurde in einer großen Computerzeitschrift (verbreitete Auflage ca. 400.000 Exemplare) einem breiten Publikum vorgestellt.

[CT05a] *Dr. Till Harbaum. Funkzentrale – Heim-Elektronik per Funk fernbedienen (Teil 1). c't Magazin für Computertechnik. Heft 14, 2005, pp. 204ff.*

[CT05b] *Dr. Till Harbaum. Drahtlos-Dolmetscher – Heim-Elektronik per Funk fernbedienen (Teil 2). c't Magazin für Computertechnik. Heft 15, 2005, pp. 220ff.*

[CT05b] *Dr. Till Harbaum. Draht- und Funkbeziehungen – Heim-Elektronik per Funk fernbedienen (Teil 3). c't Magazin für Computertechnik. Heft 16, 2005, pp. 178ff.*

An der Uni Karlsruhe waren Konzepte und Ideen entwickelt worden, wie Java auf kleinen, eingebetteten Systemen, z.B. in der Lehre, eingesetzt werden kann. Diese Konzepte und Ideen wurden auf der Fachtagung „GI/ITG Fachgespräch Sensornetze Zürich“ einem Fachpublikum vorgestellt:

[GI05] *Thomas Fuhrmann, Till Harbaum. A Platform for Lab Exercises in Sensor Networks.Technischer Bericht – 4. GI/ITG Fachgespräch Sensornetze Zürich, Schweiz, 23-24 März, 2005, Seiten 29-32.*

Folgende Kernkompetenzen der Projektpartner bildeten die Grundlage dieses Projektes:

1. Universität Karlsruhe, später TU München – Die Arbeitsgruppe von Dr. Fuhrmann hat zuvor ein effizientes Routingverfahren entwickelt, das ohne zentralisierte Komponenten auskommt. Dieses Verfahren sollte die Grundlage der in diesem Projekt zu entwickelnden Programmierumgebung werden.

2. Beecon GmbH – Das Kernprodukt der Beecon GmbH war ein flexibler und äußerst kompakter Bluetooth-Protokollstack für 8-Bit-Mikrocontroller. Neben dem Bluetooth-Stack wurden zuvor auch verschiedene weitere Kommunikationsprotokolle für kleine Mikrocontroller implementiert, einschließlich Ethernet und TCP/IP.

   Aufbauend auf diesen Implementierungen hat die Beecon nach Kundenwunsch kleine eingebettete Hardware-Module entwickelt, die von den Kunden in ihre jeweiligen Produkte eingebracht werden konnten. Weiterhin hatte die Beecon eine sehr kompakte Java Ausführungsumgebung entwickelt, die gemeinsam mit den Kommunikationsprotokollen in einen kleinen Mikrocontroller geladen werden konnte.

3. Hochschule der Medien (HdM) Stuttgart – Im Labor für „Mobile Application Development“ wurden in Zusammenarbeit mit der Industrie Projekte im Bereich Softwareentwicklung für mobile und eingebettete Systeme unter Einsatz verschiedenster Betriebssystem-Plattformen umgesetzt, z.B. mobile Informationssysteme, Ferndiagnose- und Fernsteuerungssoftware, Management, Location-Based-Services, mobile Zahlungssysteme, mobile Haussteuerung und Hausüberwachung. Außerdem hatte die HdM bereits verschiedene Eclipse-Plugins für die Embedded Softwareentwicklung entwickelt.

4. Alcatel SEL AG, später Alcatel-Lucent Deutschland AG – Die Alcatel SEL AG hatte in den Jahren vor Projektbeginn ein attraktives Portfolio verschiedener drahtgebundener und drahtloser Netzzugangsprodukte entwickelt. Dabei kamen Technologien wie OSGi Dienste, Java Middleware und multimodale Kommunikationsschnittstellen zum Einsatz. Zusammen mit Beecon und NetAds, sowie in enger Anlehnung an die HdM, wurden verschiedene Projekte im Bereich Heimautomatisierung und mobile Informationsdienste durchgeführt.

5. Fraunhofer IESE – Das Fraunhofer IESE befasste sich seit fast 10 Jahren mit der Entwicklung und Einführung industrietauglicher Methoden und Techniken des Software-Engineerings (SE), besonders auf dem Gebiet der eingebetteten Software. Im Kontext des hier beantragten Projektes sollten die Erfahrungen und Ergebnisse aus den Bereichen der Produktlinien-Architekturen (PuLSE), der Architekturbewertung, der komponentenbasierten Entwicklung (KoBRA) und der Qualitätssicherung (Built-In Tests), sowie der zu Projektstart dann vorliegenden Ergebnisse des deutsch-ungarischen Projektes BelAmi („Bilateral German-Hungarian Collaboration on Ambient Intelligent Systems“) zum Thema Ambient Intelligence in AmbiComp eingebracht werden.

Eine besondere Grundlage für den neuartigen Ansatz, der in diesem Projekt verfolgt werden sollte, war das an der Universität Karlsruhe entwickelte so genannte Scalable Source Routing (SSR), das es ermöglicht, äußerst Ressourcen-schonend Nachrichten in einem unstrukturierten Netzwerk zu vermitteln. SSR ist selbstorganisierend und benötigt somit

keinerlei zentrale Strukturen oder Server.

Scalable Source Routing wurde in den folgenden Veröffentlichungen beschrieben:

[IFIP05] *Thomas Fuhrmann. A Self-Organizing Routing Scheme for Random Networks. Proceedings of the 4th IFIP-TC6 Networking Conference, LNCS 3462, Waterloo, Canada, May 2-6, 2005 pp. 1366-1370*

[ENS05] *Thomas Fuhrmann. The Use of Scalable Source Routing for Networked Sensors. Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors Sydney, Australia, May 30-31, 2005 pp. 163-165*

[SACN05] *Thomas Fuhrmann. Scalable Routing for Networked Sensors and Actuators. Proceedings of the Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, Santa Clara, California, 26-29 September 2005.*

5. Zusammenarbeit mit anderen Stellen

Ein Ziel von AmbiComp war es, möglichst frühzeitig mit assoziierten Partnern im Bereich der kleinen und mittelständischen Unternehmen in Kontakt zu treten, um sie zu überzeugen, die AmbiComp-Technologie für ihre Produkte einzusetzen.

Als möglicher assoziierter Partner intessierte sich Herr Rohr von der Firma Rohr Technische Informatik GmbH in Singen im April 2008 für die AmbiComp-Technologie. Es fand ein Treffen bei der HdM in Stuttgart statt, bei dem Ihm das AmbiComp-SM Konzept und die gesamte Entwicklungsumgebung vorgestellt wurde. Leider ergab sich jedoch aus diesem Kontakt keine weitere Zusammenarbeit, da die AmbiComp-Hardware zu dem Zeitpunkt noch nicht seinen Vorstellungen und Bedürfnissen entsprach.

Auf der CeBIT 2009 wurde das AmbiComp Projekt einer breiten Öffentlichkeit vorgestellt. Es gab mehrere Kontakte, sowohl aus der Industrie als auch von anderen Forschungseinrichtungen und Universitäten, die großes Interesse sowohl an der AmbiComp-Technologie an sich als auch an den innovativen Ideen des AmbiComp Projektes gezeigt haben. Allerdings ergab sich trotz intensiver Nachbereitung der entstandenen Kontakte im Anschluss an die CeBIT keine weitere Zusammenarbeit mit Dritten.

Über Alcatel-Lucent ergaben sich externe Kontakte aus einem verwandten EU-Forschungsprojekt „ESTIA“, in dessen Rahmen auch AmbiComp vorgestellt wurde, sowie mit den Bell-Labs, dem Forschungsbereich von Alcatel-Lucent, der evtl. die Ergebnisse in Demonstratoren verwenden kann. Die Vorstellung des Projektes AmbiComp und der aktuellen Datenblätter zu den einzelnen Hardware-Sandwichmodulen stieß bei den Bell-Labs auf großes Interesse. Dort wird derzeit geprüft, ob Teile aus diesem Projekt bei aktuellen Projekte wie “Smart Senior“ oder “Netcomp“ verwertet werden können. Die Mitarbeiter des Forschungszentrums wollen sich mit dem Projekt Koordinator Dr. Fuhrmann von der TU München in Verbindung setzen, um weitere detaillierte Informationen über die einzelnen Module und deren Funktionalität zu erfragen.

Nachdem Sven Schlender nach Ende der Projektlaufzeit aus der Beecon GmbH ausgeschieden ist, versucht er nun bei seiner neuen Firma, der ESA-Grimma GmbH, das AmbiComp BIOS als Low-Level Abstraktionsschicht einzuführen.

Durch das BMBF Nachfolgeprojekt „J-Cell“ ergab sich ein Kontakt zu der MicroDoc GmbH aus München, die in Betracht zieht, AmbiComp-Technologien einsetzen.

6. Zusammenfassung der Projektziele

Projektziel	Erreichte Lösung
1. Ambient Intelligence soll durch das Zusammenwirken vieler Einzelgeräte in einem System erzeugt und nicht im Vorhinein konstruiert werden.	Im Projekt wurde ein umfangreicher Baukasten von Sandwichmodulen entwickelt. Darauf setzt die AmbiComp Virtual Machine auf, die eine einfache Programmierung dieser Module in Java ermöglicht. Durch das AmbiComp BIOS können auch nativ kompilierte Anwendungen in das AmbiComp System mit eingebunden werden.
2. Auf Grundlage des Scalable Source Routing Protokolls sollen Nachrichten in einer beliebigen Netzwerktopologie ohne zusätzliches Konfigurieren der Geräte zugestellt werden, um so Ressourcen und Dienste dynamischen auffinden und binden zu können. So soll das Gesamtsystem Code-, Objekt- und Threadmigration zwischen verschiedenen Geräten ohne zentrale Instanz ermöglichen. Zusätzlich sollte SSR gegen Angriffe von bösartigen Teilnehmern und gegen Fehler bei der Datenübertragung als auch bei fehlerhafter Hard- und Software geschützt werden.	Das Object Distribution Model (ODM) beschreibt die einheitliche Vernetzbarkeit von AmbiComp-Systemen. Auf Basis dieses Models können – in zukünftigen Systemen – Anwendungen verteilt über mehrere Sandwichmodule hinweg ausgeführt werden. Das ODM nutzt den Ansatz von Singletons und Domänen um Ressourcen aufzufinden. Arbeiten zum IESE Komponentenmodell unterstützen die Kombinierbarkeit von Diensten. Das Domänenmodell bezieht außerdem Sicherheitsaspekte mit ein. Zum Zugriff auf entfernte Objekte wurde SSR untersucht und verfeinert. Daraus entstand ein GAO-Zugriffsprotokoll, das Ideen von SSR aufgreift, aber effizienter auf Objekte zugreifen kann als SSR selbst.

3. Ein Hardwaresubstrat, die so genannte *Ambient Intelligence Control Unit* (AICU) macht das Gesamtsystem für die rasche Prototypentwicklung und den Einsatz in realen Anwendungen nutzbar. Diese AICU besteht aus Hardware (Mikrocontroller plus externer Beschaltung) und der fest eingebrachten Software, der sogenannten „Firmware“, bestehen.

   Diese Firmware agiert als minimales Betriebssystem, bestehend aus grundlegenden Hardware-Treibern, einer Ausführungsumgebung (Java Virtual Machine, JVM) und weiteren Anteilen wie z.B. Kommunikations-protokollen.

   In diese Firmware sollte SSR so integriert werden, dass sowohl lokal als auch auf entfernten Geräten auf Code, Objekte und Threads zugegriffen werden kann.

   Die AICU wurde in Form von Stapeln aus Sandwichmodulen realisiert. So kann für jeden Anwendungszweck die passende Kombination aus Kommunikation und I/O zusammengesteckt werden.

   Auf den Sandwichmodulen steht das AmbiComp BIOS und die AmbiComp Virtual Machine (ACVM) zur Verfügung. Zusammen mit den in die ACVM ladbaren Systembibliotheken decken sie alle das gesamte Firmwarespektrum ab.

   Die Integration von SSR bzw. den aus der SSR-Idee weiterentwickelten Protokollen konnte im Rahmen der Projektlaufzeit *nicht* mehr abgeschlossen werden.

4. Zum Projektende wird eine vollständige Spezifikation der AICUs veröffentlicht. Die Einhaltung dieser Spezifikation soll durch Konformitätstests gesichert werden. Ein Logo soll Produkten Dritter bestätigen, dass ihre Geräte die Konformität erfüllen.

   Für alle Sandwichmodule wurden Datenblätter mit den Kenndaten der Module veröffentlicht. Das Object Distribution Model, das BIOS und die ACVM sind in technischen Berichten spezifiziert, die auf der AmbiComp Webseite verfügbar sind.

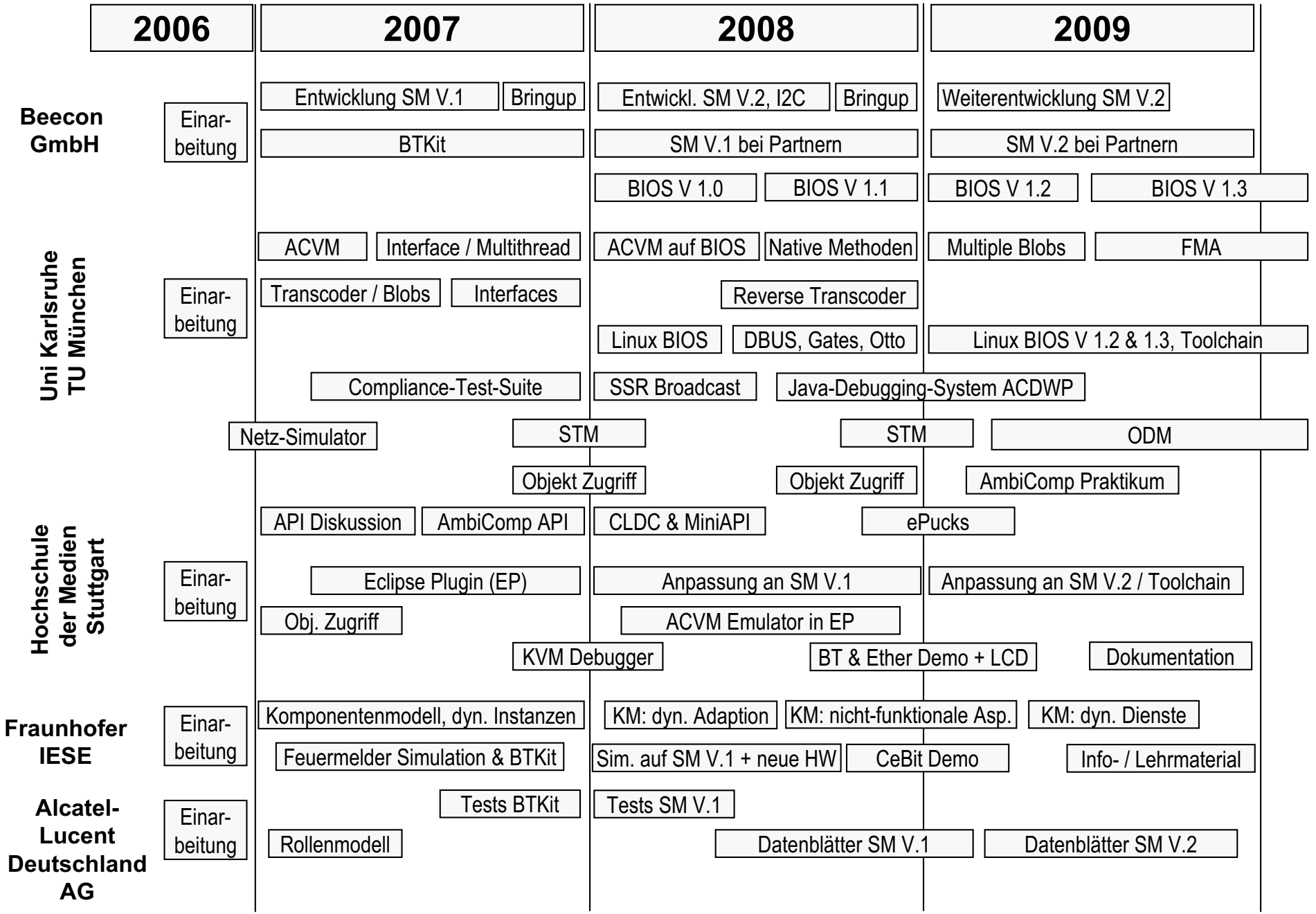
   Für das BIOS und die ACVM existiert jeweils ein großer Fundus mit Konformitätstests, den so genannten AppTests bzw. Compliance-Tests.

   Das AmbiComp-Logo ist ein eingetragenes Markenzeichen der Beecon GmbH.

5. Um AICUs in realen Geräten Dritter zu nutzen, sollten Geräteschnittstellen zu den jeweiligen Sensoren und Aktoren geschaffen werden, die nach oben zur Anwendung hin möglichst plattformübergreifend sein sollten. Diese Schnittstellen sollten in konkreten Beispielenanwendungen und Demonstratoren exemplarisch umgesetzt werden.

   Die AmbiComp Sandwichmodule unterstützen die Geräteschnittstellen nach dem Roboternetzstandard. Sie können über APIs auch Java heraus angesprochen werden. Dieser Einsatz wurde in Demonstratoren vorgeführt.

6. Anlehnend an Konzepte der klassischen Parallelrechnerprogrammierung sollen die Konzepte des Software-Engineerings für verteilte Systeme weiter entwickelt und in Fachveröffentlichungen publiziert werden. Hieraus sollte Schulungsmaterial geschafft werden, das es Software-Entwicklern ermöglicht, anhand von Beispielanwendungen eigene Systeme zu entwickeln.	Das Fraunhofer IESE hat sein Komponentenmodell in vielen Fachveröffentlichungen beschrieben. Das AmbComp Tutorial zeigt, wie mit dem AmbiComp-System Software entwickelt werden kann. Der praktische Umgang mit dem System wurde in einem Praktikum an der Uni Karlsruhe von Studenten erprobt.
7. Um Software-Entwicklern eine schnelle Produktivität zu ermöglichen, sollte auf Basis des Open Source Werkzeugs „Eclipse“ eine Entwicklungsumgebung geschaffen werden. Diese sollte es ermöglichen, rasch Ambient Intelligence Anwendungen zu entwickeln und zu testen.	Das AmbiComp Eclipse Plugin vereinigt alle Aspekte der Entwicklung von AmbiComp Anwendungen.
8. Das Zusammenspiel vieler Einzelgeräte mit eingebetteten AICUs sollte durch Benutzerschnittstellen auf Basis von Handys und „Persönlichen Digitalen Assistenten“ (PDAs) ergänzt werden.	In Beispielanwendungen hat die Hochschule der Medien die Verbindung von AmbiComp Sandwichmodulen und Smartphones demonstriert.


2006
2007
2008
2009
Beecon GmbH
Einar-beitung
Entwicklung SM V.1
Bringup
Entwickl. SM V.2, I2C
Bringup
Weiterentwicklung SM V.2
BTKit
SM V.1 bei Partnern
SM V.2 bei Partnern
BIOS V 1.0
BIOS V 1.1
BIOS V 1.2
BIOS V 1.3
Uni Karlsruhe TU München
ACVM
Interface / Multithread
ACVM auf BIOS
Native Methoden
Multiple Blobs
FMA
Einar-beitung
Transcoder / Blobs
Interfaces
Reverse Transcoder
Linux BIOS
DBUS, Gates, Otto
Linux BIOS V 1.2 & 1.3, Toolchain
Compliance-Test-Suite
SSR Broadcast
Java-Debugging-System ACDWP
Netz-Simulator
STM
STM
ODM
Objekt Zugriff
Objekt Zugriff
AmbiComp Praktikum
Hochschule der Medien Stuttgart
API Diskussion
AmbiComp API
CLDC & MiniAPI
ePucks
Einar-beitung
Eclipse Plugin (EP)
Anpassung an SM V.1
Anpassung an SM V.2 / Toolchain
Obj. Zugriff
ACVM Emulator in EP
KVM Debugger
BT & Ether Demo + LCD
Dokumentation
Fraunhofer IESE
Einar-beitung
Komponentenmodell, dyn. Instanzen
KM: dyn. Adaption
KM: nicht-funktionale Asp.
KM: dyn. Dienste
Feuermelder Simulation & BTKit
Sim. auf SM V.1 + neue HW
CeBit Demo
Info- / Lehrmaterial
Alcatel-Lucent Deutschland AG
Einar-beitung
Tests BTKit
Tests SM V.1
Rollenmodell
Datenblätter SM V.1
Datenblätter SM V.2

AmbiComp Abschlussbericht
Teil 2 – Eingehende Darstellung

1 Projektbeginn, Einstellungs- und Einarbeitungsphase

Das Projekt startete formal zum 1. Mai 2006. Bis die Bewilligungsbescheide zugestellt waren und die jeweiligen Prozesse in den Verwaltungen der beteiligten Projektpartner angestoßen waren, vergingen jedoch einige Wochen. Erst nachdem diese formalen Prozesse abgeschlossen waren, konnte bei den einzelnen Projektpartnern die Personalauswahl beginnen. Soweit intern kein geeignetes Personal zur Verfügung stand, konnten mit Stellenausschreibungen geeigneten Mitarbeitern gesucht werden. Dieser gesamte Prozess zog sich über mehrere Monate hin.

Hinweis: Die nachfolgende namentliche Nennung der Projektmitarbeiter, sowie deren Zuordnung zu einzelnen Arbeitsbereichen ist vertraulich zu behandeln!

In den ersten Monaten nach ihrer Einstellung arbeiteten sich die einzelnen Mitarbeiter in das AmbiComp Projekt und ihre jeweilige Aufgabenstellung ein. Parallel dazu begannen die Projektpartner mit den inhaltlichen Diskussionen und der Abstimmung untereinander. Ein Teil dieser Arbeiten war das gemeinsam entwickelte AmbiComp-Logo. Es wurde in enger Abstimmung mit den Projektpartnern durch die Studio 88 Werbe- und Trickfilm GmbH gestaltet und durch die Beecon GmbH beim Deutschen Patent- und Markenamt zusammen mit dem Namen „AmbiComp“ als Bildmarke und Wortmarke angemeldet.

Zusammen mit dem in kirschrot und grau gehaltenem Design bildet das Logo die Grundlage für die Projekt-Webseiten www.ambicomp.org. Dort findet sich eine Übersicht über das Projekt, die beteiligten Partner und eine Auswahl der im Projekt entstandenen Dokumente wie Datenblätter, Veröffentlichungen und Spezifikationen. Die Webseite richtet sich in erster Linie an interessierte Dritte. Während der Projektlaufzeit sollte diese Webseite nicht nur das Projekt

vorstellen, sondern beispielsweise auch zur Gewinnung assoziierter Partner dienen. Nach Abschluss des Projekts wurde die Webseite grundlegend überarbeitet. Sie dient nun hauptsächlich der Darstellung der Projektergebnisse.

Abbildung 1: Screenshot der AmbiComp Website

Zur Projektorganisation wurden die offiziellen Dokumente auf einem BSCW Server am Fraunhofer IESE abgelegt und mehrere Mailinglisten und später ein Wiki eingerichtet. Des Weiteren wurde das Projektplanungswerkzeug „JIRA“ eingesetzt.

2 Anlaufphase

Um den Partnern möglichst rasch Hardware zur Verfügung zu stellen, wurde bei der Beecon der in Vorarbeiten entstandene eingebettete Steuerknoten, das „BTKit“, nochmals in geringer Stückzahl gefertigt und an die Projektpartner ausgeliefert. Auf diese Weise konnte die Beecon den gesamten Herstellungsprozess der eingebetteten Hardware durchlaufen und erproben, angefangen mit der Einarbeitung in das Hardwaredesign, über die Auswahl und den Einkauf der benötigten Elektronikbauteile, bis hin zum eigentlichen Fertigungsprozess bei einem Dienstleister und der anschließenden Qualitätskontrolle. Parallel zu diesen Arbeiten wurden zusätzlich mögliche Übertragungstechniken und verschiedene Mikrocontrollerplattformen für die nächste Hardwaregeneration gesichtet.

Während bei der Beecon die erste Hardware gefertigt wurde, entwickelte die Uni Karlsruhe zusammen mit der HdM eine erste Anforderungsanalyse für das Eclipse-Plugin. Das Ergebnis der Analyse wurde ausführlich diskutiert und von der HdM als detaillierte Anforderungsspezifikation [HDM08] in einem Dokument zusammengefasst. Da die Anforderungen an die Entwicklungsumgebung wesentlich von dem zu entwickelnden System abhängen, musste zu Beginn dieser Studie das angestrebte AmbiComp System selbst in seinen grundlegenden Zielen, Anwendungsszenarien, Komponenten, Funktionen und Schnittstellen definiert werden. Dabei wurden die Grundzüge der zu entwickelnden AmbiComp Plattform definiert, die erforderlichen Schnittstellenkategorien festgelegt und die Integration in die zu entwickelnde Simulationsplattform skizziert. Dieses Dokument wurde während der Projektlaufzeit fortlaufend angepasst und erweitert.

[HDM08] Thomas Suchy, Jürgen Butz, Johannes Maucher, Jörg Rech. Anforderungsanalyse Eclipse-Plugin. Version 1.0.0, 15. August 2008

Nach dieser ersten Analyse wurde eine Kategorisierung der möglichen Entwicklergruppen von zukünftigen AmbiComp Systemen abgeleitet. Hierzu erfolgte eine Einteilung dieser Anforderungen in die drei Ebenen Anwendung, Software-System und Hardware.

Darauf aufbauend wurden weitere Fragestellungen, die sich in diesem Zusammenhang ergaben, formuliert und ihr Bezug zu verwandten Forschungsgebieten erschlossen. Zusammen mit dem AmbiComp Glossar entstand aus diesen Arbeiten ein Dokument [HDM07], das diese verschiedenen Aspekte beschreibt. Es verfolgt einen Top-Down-Ansatz: Der Begriff Ambient Intelligence wurde im Zusammenhang mit seinen Zielvorstellung und Visionen definiert. Daraus ergaben sich unterschiedliche funktionale und nichtfunktionale Anforderungen an eine Ambient Intelligence Umgebung. Die im Dokument einheitlich definierten Begriffe erleichterten die Kommunikation zwischen den Projektpartnern: Das Dokument diente als Grundlage für Diskussionen über die Projektziele, sowie die Einordnung der Arbeiten der einzelnen Projektpartner in die unterschiedlichen Ebenen von Ambient Intelligence Systemen im Allgemeinen und AmbiComp im Speziellen.

Dieses Dokument wurde während eines technischen Meetings in Karlsruhe von der HdM vorgestellt und eingehend mit den Projektpartnern diskutiert. Insbesondere wurde es einem Vorschlag des Fraunhofer IESE gegenübergestellt. Dabei verfolgt das Fraunhofer IESE Dokument [IESE06] den Bottom-Up-Ansatz: Anhand einer Übersicht zu Ambient Intelligence Projekten mit AmbiComp-Bezug und konkreter Beispielszenarien im Umfeld Ambient Intelligence werden darin Anforderungen generalisiert, um daraus die wesentlichen Herausforderungen hinsichtlich der Entwicklung solcher Systeme abzuleiten.

[HDM07] Thomas Suchy, Jürgen Butz, Johannes Maucher, Harald Orlamünder. Ambient Intelligence Definition. Version 0.6, 01 Februar 2007

[IESE06] Michalis Anastasopoulos, Frank Böhr, Thomas Patzke, Christian Peper, Daniel Schneider, Martin Sohn. Ambient Intelligence Systems: Scenarios and Challenges. IESE-Report No. 191.06/E, Version 1.0, 22 Dezember 2006

Beide Ansätze führten zu sehr ähnlichen Ergebnissen bzgl. Anforderungen an Ambient Intelligence Umbebungen. In Diskussionen mit den Projektpartnern konnte das AmbiComp-Projekt so in das allgemeine Umfeld Ambient Intelligence eingeordnet werden. Außerdem wurden die Anforderungen, bezogen auf die jeweiligen Aufgaben der Projektpartner, gewichtet.

Auf Basis dieser Diskussionen wurden bei Alcatel-Lucent erste interne Kontakte aufgenommen und die vorhandenen Möglichkeiten für spätere Tests und Demonstrationsmöglichkeiten ausgelotet. Ein Vortrag zum Thema „Ambient Intelligence aus Sicht der Industrie“ der im Rahmen der Eröffnung des “Assisted Living Labors” am Fraunhofer IESE in Kaiserslautern gehalten wurde, stellte diese Aspekte öffentlich vor.

Aufgrund der späteren Konzernumstrukturierung bei Alcatel-Lucent kam es leider nie dazu, diese damals angebahnten Möglichkeiten auch wirklich zu nutzen.

Ebenfalls in Hinblick auf die zukünftige Verwertung der Projektergebnisse wurden parallel dazu bei Alcatel-Lucent Überlegungen zu den Rollen der verschiedenen Marktteilnehmer entlang der Wertschöpfungskette angestellt. In Diskussionen mit den Projektpartnern wurden diese Rollen weiter heraus gearbeitet. Dieser Schritt war wichtig, um einerseits ein gemeinsames Verständnis über die Begrifflichkeiten zu erzielen, und um andererseits die verschiedenen Zielgruppen (Hersteller von Hardware und Basis-Software, Anwendungs-Entwickler, Vertriebsorganisationen, usw.) entsprechend zu adressieren zu können.

Zeitgleich zu diesen, auf die Verwertung zielenden Arbeiten wurde an der Uni Karlsruhe mit den ersten Implementierungen eines Netzwerksimulators für AmbiComp-Systeme begonnen. Dazu wurde auf Basis des OMNeT++-Netzwerksimulators ein allgemeines Wrapper-Konzept entwickelt, mit dem verschiedene Implementierungen einzelner Netzwerkaspekte in ein

gemeinsames Rahmenwerk eingebunden werden können. Ursprünglich sollte dieser Netzwerksimulator an das Eclipse-Plugin angebunden werden. Im weiteren Projektverlauf zeigte sich allerdings, dass eine solche Integration sehr aufwendig gewesen wäre, so dass die Arbeiten nicht mehr im Projektzeitraum abgeschlossen hätten werden können.

3 Arbeitsphase mit dem BTKit

Nachdem die Fertigung der BTKits bei der Beecon erfolgreich abgeschlossen wurde, konnten diese Module an die Projektpartner ausgeliefert werden. Die BTKits verfügten über einen 8-Bit-AVR-Mikrocontroller, der die AmbiComp Virtuelle Maschine (siehe unten) ausführen konnte, sowie ein Bluetooth-Funkmodul für die Kommunikation mit Peripheriegeräten. Um Zugriff auf das BTKit von einem PC aus zu bekommen, verfügte das BTKit außerdem über eine serielle Schnittstelle, über die das BTKit mittels eines speziellen Programmiersteckers an einen PC angeschlossen werden konnte. So konnten mit Hilfe des weit verbreiteten XModem-Datenübertragungsprotokoll Daten an einen PC geschickt bzw. von diesem empfangen werden. Empfangene Daten konnten dabei beispielsweise neue Java-Programme, die die ACVM ausführten sollte oder auch Updates für die Firmware selbst, d.h. in der Regel neue Versionen der ACVM, sein. Über die serielle Schnittstelle war es außerdem möglich, Debug-Information auszugeben und anzuzeigen.

Im Verlauf der weiteren Arbeiten zeigte sich allerdings, dass die physikalische Verbindung zwischen BTKit und dem Programmierstecker sehr sensibel auf mechanische Belastungen reagierte. Deshalb war das Aufspielen von neuer Software sowohl zeitaufwendig als auch sehr fehleranfällig.

Parallel zur BTKit Fertigung und den ersten Arbeiten mit dem BTkit wurde von der Beecon und der Uni Karlsruhe gemeinsam ein modulares Hardwarekonzept entwickelt, bei dem eine AICU als Sandwich aus so genannten Sandwichmodulen (SM) zusammengesetzt werden kann. Diese Sandwichmodule werden über einen Rückwandbus (engl. Backplane) miteinander verbunden.

Im ersten Halbjahr 2007 begann die Beecon dann mit der Entwicklung der ersten Sandwichmodule. Als erstes sollten SMs mit den Kommunikationstechnologien Bluetooth (BTSM) und Ethernet (EtherSM) umgesetzt werden. Parallel dazu wurde ein Input-Output-Sandwichmodul (IOSM) entwickelt, das den Betrieb verschiedener Sensoren und Aktoren an einer AICU ermöglichte.

Das IOSM sowie das BTSM wurden hierbei mit einem 8-Bit-AVR-Mikrocontroller bestückt, wodurch sie zu so genannten intelligenten SMs wurden. Das EtherSM hingegen verfügte über keinen eigenen Mikrocontroller, war also insofern passiv, d.h. auf die Steuerung durch ein intelligentes SM angewiesen.

Neben der Hardwareentwicklung befasste sich die Beecon ebenfalls mit der Entwicklung der Firmware. Diese Arbeit stellte sich als deutlich zeitaufwendiger heraus als ursprünglich erwartet. Die erweiterten Anforderungen an die Funktionalität der AmbiComp Virtuellen Maschine (ACVM) machten mehr Änderungen am bestehenden Code erforderlich als geplant. So wurde beispielsweise das Speichermanagement in großen Teilen umgearbeitet, um später den Zugriff auf entfernte Objekte zu vereinfachen. Außerdem musste der ACVM Code nicht nur für die jeweilige Hardware, d.h. BTKit und Sandwichmodule geschrieben werden, sondern ebenfalls für Linux entwickelt werden, da nur so das einfache Debugging unter Linux möglich war. Andernfalls hätte jeder einzelne Entwicklungsschritt einen aufwendigen Flashvorgang auf die Hardware erfordert.

Beim Einarbeiten dieser Änderungen in die bestehende Software-Basis, sowie durch die intensiven Tests der Universität Karlsruhe wurden außerdem viele kleine Implementierungsfehler (Bugs) gefunden. Diese zu beseitigen erforderte ebenfalls einigen zusätzlichen Zeitaufwand. Insgesamt entstanden dadurch erhebliche Zeitverzögerungen bei den einzelnen Arbeitspaketen.

Da die ACVM möglichst effizient und klein sein musste, um eine möglichst hohe Performance auf der Hardware zu erreichen, wurde sie so entwickelt, dass sie ein spezielles Binärformat ausführt, das vom herkömmlichen Java-Bytecode abweicht. Dieses Binärformat – das so genannte „Binary Large Object“ (Blob) Format – kann auf den AICUs effizienter gespeichert und ausgeführt werden als der ursprüngliche Java-Bytecode selbst.

Um ein herkömmliches Java-Programm auf der ACVM ausführen zu können, muss dieses zuerst in dieses Blob-Binärformat umgewandelt werden. Dazu dient der so genannte Transcoder. Dabei handelt es sich um ein Java-Programm das den Java-Bytecode in das Blob-Format der ACVM übersetzt, also transcodiert. Die Uni Karlsruhe begann zunächst mit der grundlegenden Überarbeitung des aus Vorarbeiten bereit vorhandenen, eigenständigen Transcoders. Er wurde dann später von der Hochschule der Medien in das Eclipse-Plugin integriert.

Bei den Arbeiten am Transcoder hat sich gezeigt, dass der nötige Aufwand für diese Arbeit im Projektantrag deutlich unterschätzt wurde. Beim Schreiben des Antrags wurde unterstellt, dass der Transcoder nur ein kleines Detail des Eclipse-Plugins darstellt, das vollständig aus dem Fundus der Vorarbeiten zum Projekt übernommen werden kann. Der Transcoder wurde deshalb im Projektantrag gar nicht als eigenes Arbeitspaket ausgewiesen. Beim Versuch, den vorhandenen Transcoder für AmbiComp zu übernehmen, hat sich aber gezeigt, dass dieser fast vollständig neu geschrieben werden musste.

Um den Transcoder und die ACVM jeweils einzeln und im Zusammenspiel miteinander zu testen, wurde an der Uni Karlsruhe eine so genannten *Compliance-Test-Suite* mit Hilfe des Continuous-Integration-Systems *Hudson* aufgesetzt. Dieses System verbindet den Transcoder und die Linux-ACVM mit einem großen Satz von verschiedenen, kleinen Java-Einzeltests die so konzipiert und implementiert wurden, dass sie die vielen verschiedenen Aspekte der ACVM kontinuierlich testen können. Ähnlich wie bei Unit-Tests wurde so die Funktionsfähigkeit der einzelnen Transcoder- und ACVM-Komponenten getestet. Außerdem wurde so das Gesamtsystem automatisch geprüft und etwaige Abweichungen von der Spezifikation direkt dem Entwickler gemeldet. Hierdurch konnte die reibungslose Integration der Projektbeiträge gewährleistet werden.

Bei diesen Compliance-Tests handelt es sich um eine Vorstufe des AmbiComp Plattform-Conformance-Tests. Der eigentliche, im Projektantrag beschriebene Plattform-Conformance-Test der Systeme von Drittanbietern zertifizieren sollte wurde innerhalb der Projektlaufzeit nicht mehr fertig gestellt, da dieser aufgrund des fehlenden Interesses möglicher assoziierter Partner mit einer niedrigeren Priorität eingestuft wurde. Für das AmbiComp Projekt erfüllt der vorhandene Compliance-Test die neuen Anforderungen an das AmbiComp-System besser, weil er vor allem die Kern-Komponente, die ACVM, gegenüber der für AmbiComp angepassten Java-Spezifikation testet. Aspekte des verteilten Betriebs können so hingegen nicht geprüft werden.

Der Erfolg des gewählten Compliance-Test Ansatzes zeigte sich während der gesamten Projektlaufzeit anhand einer durchgehend hohen Stabilität der AICU-Firmware und dem reibungslosen Zusammenspiel der einzelnen Komponenten der AmbiComp Toolchain, d.h. Java Compiler, Transcoder und ACVM. Über die gesamte Projektlaufzeit hinweg hat das Aufsetzen und Pflegen der Compliance-Test-Suite immer wieder erhebliche Personalressourcen an die Uni Karlsruhe bzw. TU München beansprucht. Am Ende der Projektlaufzeit standen dadurch ca. 200 Einzeltests zur Verfügung die jeweils für die unterschiedlichen APIs und verschiedenen Plattformen (Sandwichmodule und Emulator) ausgeführt werden können.

Im Rahmen der Compliance-Test-Suite wurde zwischen den Projektpartnern auch der Funktionsumfang der Basisbibliotheken diskutiert, die nativ auf einer AICU zur Verfügung stehen müssen, um den Zugriff auf die AICU und die daran angeschlossene Sensorik und Aktorik zu ermöglichen. Hierzu gehörte das Zusammentragen einer Übersicht über die verschiedenen APIs anderer Java-Systeme, um so einen für AmbiComp gangbaren Weg zu finden, der die größtmögliche Kompatibilität mit diesen anderen Systemen sicherstellt.

Diese Diskussion wurde auch im Rahmen der Kooperation zwischen Dr. Thomas Fuhrmann und Frau Prof. Ursula Damm (Bauhaus Universität Weimar) am Projekt *Fernfühler* geführt. Dabei handelte es sich um autonome, fahrbare Würfel die ihr „Verhalten“ dem der Menschen in ihrer Umgebung anpassen sollten. Für AmbiComp ergaben sich aus diesem Projekt wertvolle Anregungen zu den Anforderungen an Hardware und Firmware der AICUs, die die Grundlage für die weiteren Arbeiten im Projekt bildeten. So begannen die Projektpartner beispielsweise eine Diskussionen über mögliche Java Application Programming Interfaces (APIs). Leider hat sich nach der anfänglichen Kooperation keine weitere Zusammenarbeit mit Frau Prof. Damm ergeben, da die beiden Projekte, AmbiComp und Fernfühler, in zwei vollkommen unterschiedliche Richtungen weiterliefen: AmbiComp eher in technischer Hinsicht, das Fernfühlerprojekt eher in künstlerischer Richtung.

Parallel zu dieser Zusammenarbeit wurde bei der HdM die Entwicklung des Eclipse-Plugins begonnen. Hierzu wurden bei der HdM zunächst Dummy-Screenshots erstellt, um den Projektpartnern die Zielvorstellungen der HdM präsentieren zu können. Anschließend wurde mit der Umsetzung begonnen. Zunächst wurde ein Wizard implementiert, der dem Anwender Hilfestellung beim Anlegen eines AmbiComp Projekts in Eclipse leistet. Des Weiteren wurde begonnen, den Transcoder in das Eclipse-Plugin einzubauen. Um die Programmierung der Hardware zu erleichtern, wurde außerdem der Vorgang des Aufspielens von Java-Anwendung (Deployment) in das Eclipse-Plugin integriert. Dadurch war es möglich, aus dem Eclipse-Plugin heraus AmbiComp Anwendungen zu transcodieren und anschließend auf die Hardware (BTKit) zu übertragen. Ein Update-Mechanismus für die ACVM selbst stand zu diesem Zeitpunkt noch nicht zur Verfügung. Die ACVM musste noch manuell bei der Beecon oder an der Uni Karlsruhe aufgespielt werden.

Die auf dem BTKit laufende erste Version der ACVM konnte über die serielle Schnittstelle vom Eclipse-Plugin aus angesprochen werden, um beispielsweise Debug-Ausgaben auszugeben. Dadurch wurde die Benutzung der Hardware im Vergleich zu anderen eingebetteten Systemen deutlich vereinfacht. Allerdings waren noch nicht alle Schritte automatisiert. So war der Benutzer beispielsweise selbst dafür verantwortlich, den benötigten Treiber für die serielle Schnittstelle des Entwicklungsrechners installiert zu haben. Das Eclipse-Plugin überprüfte nur, ob auf die serielle Schnittstelle zugegriffen werden konnte und informierte den Nutzer, falls dies nicht der Fall war.

Da die Kommunikation über die serielle Schnittstelle und das XModem Protokoll, wie oben beschrieben, aufwendig und fehleranfällig war, wurde die serielle Schnittstelle im weiteren Verlauf des Projektes durch ein spezielles Protokoll auf Basis einer I2C Verbindung ersetzt.

Im Verlauf der weiteren Arbeiten wurde der Transcoder dann vollständig in das Eclipse-Plugin eingebunden. Dabei hatte der *AmbiCompTranscoder-Job* im Eclipse-Plugin die Aufgabe, alle zum AmbiComp Projekt gehörenden *.class Dateien innerhalb der Projektstruktur und der zum Projekt verlinkten weiteren Projekte, sowie der eingebundenen Bibliotheken, zu bündeln (siehe Abbildung 2). Die Visualisierung für den Aufruf des AmbiCompTranscoder-Jobs wurde an verschiedenen Stellen in der Eclipse Oberfläche realisiert.

Nach dem Transcodieren einer Anwendung wurde der Blob per serieller Schnittstelle auf die Hardware übertragen. Für die Übertragung von Blobs wurde im Eclipse-Plugin die Client-Seite des XModem-CRC16 Protokolls implementiert. Die Implementierung und Integration des XModem-Protokolls (nach dem Standard von Ward Christensen und der Protokollreferenz von Chuk Forsberg) im Eclipse-Plugin wurde über die beiden Komponenten „XModem“ und „AmbiCompDeploy-Job“ abgebildet.

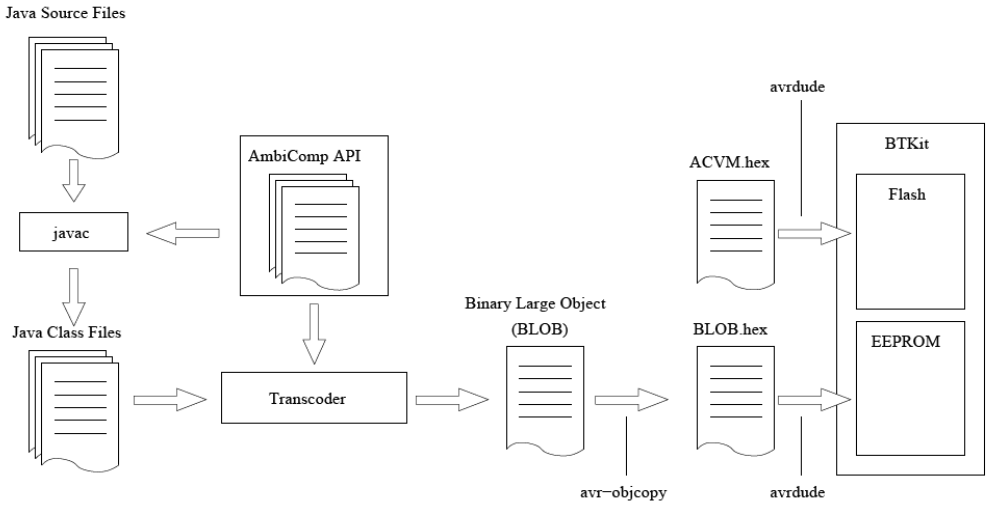


Abbildung 2: Transcodier- und Deployschritte

Bei der Alcatel-Lucent Deutschland AG wurde diese erste Version des Eclipse-Plugins installiert und getestet. Dabei wurden einige Bugs gefunden, die anschließend von der HdM beseitigt werden konnten. Derartige Tests fanden auch im weiteren Projektverlauf regelmäßig bei Alcatel-Lucent statt. Leider stand der hierfür zuständige Mitarbeiter aber ab Herbst 2008 nicht mehr zur Verfügung. Ab diesem Zeitpunkt mussten alle weiteren Tests von den anderen Projektpartnern mit übernommen werden, wodurch deren Arbeitsbelastung zusätzlich erhöht wurde.

Neben den eher praktischen Arbeiten wurden bei den einzelnen Projektpartnern zusätzlich die konzeptionellen Arbeiten vorangetrieben. Bei der HdM wurde im ersten Halbjahr 2007 ein Konzept zum transparenten lokalen und entfernten Objektzugriff, ähnlich des Java-RMI-Konzeptes, entwickelt und beispielhaft implementiert. Da noch keine AICUs zur Verfügung standen, wurde der Proof-Of-Concept in Form von MIDlets auf Basis von Mobiltelefonen implementiert. Darauf aufbauend wurden verschiedene Use-Cases erstellt, die die möglichen Tätigkeiten eines Anwenders darstellen. Diese Darstellung zeigte dem Anwender, welche Vorraussetzungen geschaffen werden müssen, damit diese Tätigkeiten umgesetzt werden können. Die Ergebnisse flossen in das Konzept und Design der Benutzerschnittstellen ein und zeigten auf, an welchen Stellen das existierende Zugriffskonzept für das BTKit erweitert und an die Bedürfnisse von AmbiComp angepasst werden musste. Parallel dazu wurde mit der Entwicklung einer Anwendung begonnen, die die erarbeiteten Konzepte im Rahmen eines Temperatursensor-Szenarios umsetzte. Dabei wurden die Funktionalitäten, der Aufbau und die Erweiterbarkeit der bis dato vorhandenen Hardware und Software des BTKit's ausgelotet und prototypisch umgesetzt. Diese Arbeiten wurden an der HdM auch im Rahmen einer Diplomarbeit unterstützt:

- Markus Heimann, „Entwicklung von Schnittstellen zur Konfiguration und Verwaltung eines Ambient Intelligence Systems“, Diplomarbeit, Hochschule der Medien, Oktober 2007

Das Fraunhofer IESE begann während dieser Zeit mit der Entwicklung ihres Komponentenmodells, das insbesondere die dynamische Instanziierung und das dynamische Management verteilter Anwendungen sowie die dynamische Integration neuer Komponenten berücksichtigte. Des Weiteren wurde der Aufbau eines ersten Simulators in Form eines *Feuermelder-Szenarios* und einer entsprechenden Simulations- und Konfigurationsumgebung vorangetrieben, mit der die in einem Gebäude verteilt angebrachten AmbiComp-Feuermelder zu einer Gruppe zusammengefasst werden sollten, siehe Abbildung 3. Zu diesem Simulator wurde eine semi-formale Spezifikation der Anforderungen erstellt und eine erste Realisierung

basierend auf handelsüblichen Feuermeldern und den BTKits entwickelt. Diese Simulations- und Konfigurationsumgebung wurde ständig erweitert und zusammen mit der Implementierung des Feuermelder-Szenarios entsprechend der im Projekt gewonnenen Ergebnisse hinsichtlich der Software-Engineering-Methodik laufend angepasst.

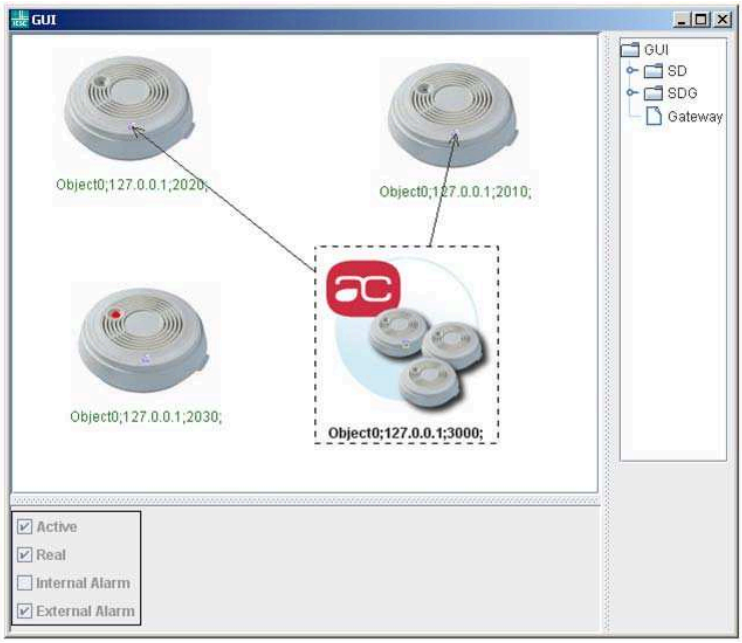


Abbildung 3: Rauchmelder-Simulation

Darüber hinaus wirkte das Fraunhofer IESE bei einem Workshop-Paper mit, in dem die Herausforderungen in der Entwicklung adaptiver service-basierter Ambient Intelligence Systeme beschrieben werden:

[FSOA07] Daniel Schneider, Christian Bunse und Klaus Schmid, „Towards Adaptive Service Engineering", Proceedings of the International Workshop on the Foundations of Service-Oriented Architecture (FSOA 2007), Juli 2007.

Zusätzlich wurden die Anfang 2007 begonnenen Arbeiten hinsichtlich der Software-Engineering-Methodik für AmbiComp am Fraunhofer IESE bis Ende 2007 maßgeblich vorangetrieben. Im Zuge dessen wurden das Komponentenmodell und die in Beziehung stehenden Konzepte konsolidiert und dokumentiert. Die daraus entstandenen Resultate wurden in Form eines weiteren Papers veröffentlicht:

[SEAMS08] C. Peper und D. Schneider: „Component Engineering for Adaptive Ad-hoc Systems", Proc. of the 30th Intl. Conf. on Software Engineering ICSE'08, SEAMS Workshop (Software Engineering for Adaptive and Self-Managing Systems), Leipzig, Mai 2008, Leipzig.

Neben den oben beschriebenen Aktivitäten wurde am Fraunhofer IESE die Interoperabilität zwischen AmbiComp und dem *BelAmI* („Bilateral German-Hungarian Collaboration on Ambient Intelligent Systems") Projekt betrachtet. Aus diesen Betrachtungen entstand ein Dokument, das zwei grundlegende Ansätze zur Integration skizziert. Außerdem entstand eine entsprechende erste Implementierung eines BelAmI-Service-Bundles, dem über eine XML Datei konfigurierbare BelAmi-Gateway. Basierend auf dem in einer Konfigurationsdatei geschriebenen *Service-Matching* bildet das Gateway Dienstaufrufe in beiderlei Richtungen ab. Dieses Gateway wurde für den ersten Feuermelder-Simulator gebaut und sollte eigentlich in das Assisted Living Lab (ALL) integriert werden. Diese Anbindung wurde aber während der

Projektlaufzeit nicht mehr realisiert.

Auch seitens der Uni Karlsruhe wurde zu den Konzepten zur verteilten Nutzung der AICUs beigetragen. Hier konzentrierten sich die Arbeiten hauptsächlich auf die Wechselwirkung mit dem zugrunde liegenden Netzwerkprotokoll. Dazu waren sowohl die Integration des *Scalable Source Routing* (SSR) Protokolls in die ACVM notwendig, als auch die Garantie der sicheren Nutzung dieser Netzwerkkommunikation durch verteilt operierenden AICUs. Wie ersteres in der ACVM umgesetzt werden könnte wurde an einem Beispiel in folgender Veröffentlichung untersucht:

[GI07] *Björn Saballus, Johannes Eickhold und Thomas Fuhrmann, „Towards a Distributed Java VM in Sensor Networks using Scalable Source Routing", 6. Fachgespräch Sensornetzwerke der GI/ITG Fachgruppe 'Kommunikation und Verteilte Systeme', Aachen, Germany, July 16-17, 2007*

Für letzteres war die Konzeption entsprechender Sicherheitsmechanismen erforderlich. Für beide Themenbereiche wurden erste Ansätze entwickelt. Die Grundlage für den sicheren Betrieb einer Anwendung über mehrere AICUs hinweg war ein besonderes Domänenkonzept: Personen und Organisationen besitzen AICUs (bzw. haben die Verfügungsgewalt darüber) wenn diese sich in ihren jeweiligen Domänen befinden. Darüber hinaus gibt es orts- und strukturinduzierte Domänen, die ein AICU-Netz logisch oder geographisch gliedern. Die AICUs können mehreren Domänen angehören und sind über kryptographisches Schlüsselmaterial gegeneinander gesichert. Diese Konzepte wurden später in Form eines Papers publiziert und einem größeren Fachpublikum vorgestellt:

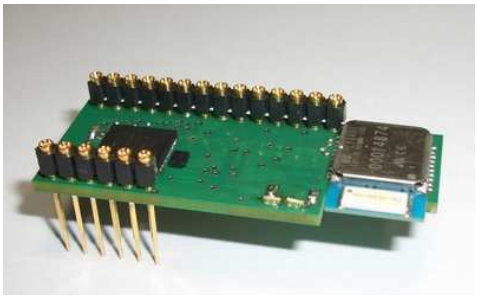
[SENS08] *Bjoern Saballus, Johannes Eickhold und Thomas Fuhrmann, „Global Accessible Objects (GAOs) in the AmbiComp Distributed Java Virtual Machine", Proceedings of the Second International Conference on Sensor Technologies and Applications (SENSORCOMM 2008), Cap Esterel, France, August 25-31, 2008*

4 Erste Sandwichmodulgeneration

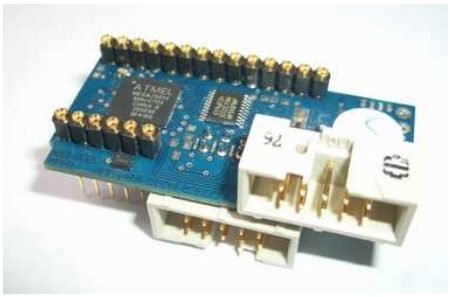
Parallel zu den oben beschriebenen Arbeiten der Projektpartner mit dem BTkit wurde bei der Beecon GmbH, in Zusammenarbeit mit der Uni Karlsruhe die erste Generation der AmbiComp-Sandwichmodule in Betrieb genommen. Diese Module konnten ab Anfang 2008 an die Projektpartner ausgeliefert werden. Zu dieser ersten Generation gehörten die folgenden Module:

- **Bluetooth-Sandwichmodul (BTSM):** Das BTSM stellt eine Bluetooth-Schnittstelle bereit. Dadurch ist eine Nahbereichskommunikation mit anderen Modulen und Geräten (Mobiltelefone, PDAs usw.) möglich. Das BTSM ist mit einem Atmel 8-Bit-AVR 2561 Mikrocontroller, einem HCI (Host Controller Interface) Bluetooth-Modul, und zusätzlich einem 128kB großen, externen SRAM Speicher bestückt.
- **Input-Output-Sandwichmodul (IOSM):** Das IOSM stellt eine Reihe allgemein nutzbarer, programmierbarer Analog- und Digital-Schnittstellen bereit. Es ist ebenfalls mit einem Atmel 8-Bit-AVR 2561 Mikrocontroller bestückt. Für die Input-/Output-Schnittstellen wurde ein Schnittstellenkonzept auf Basis der im Roboternetz-Projekt (siehe www.roboternetz.de) definierten Schnittstellen entwickelt.
- **Ethernet-Sandwichmodul (EtherSM):** Das SM stellt eine Ethernet-Schnittstelle sowie die Spannungsversorgung über Power-over-Ethernet (PoE) bereit. Dadurch ist sowohl eine Kommunikation mit anderen Modulen und Geräten (z.B. einem PC) möglich, als auch die Einbindung des AmbiComp-Systems in ein Netzwerk. Die erste EtherSM Generation wurde noch ohne Mikrocontroller gefertigt. Es konnte daher nur in Verbindung mit einem BTSM oder IOSM genutzt werden, auf dem der Ethernet-Kommunikationsprotokollstapel ausgeführt werden musste.

Zur Verbindung der Module diente der Rückwandbus. Bei der ersten Sandwichmodul-generation teilte er sich in zwei Steckerleiten auf, die entlang der langen Kanten der Platine verliefen. Über diese Leisten konnten mehrere Sandwichmodule zu einem AICU zusammengesteckt werden. Der Rückwandbus lieferte dabei die Spannungsversorgung für den gesamten AICU, einen JTAG Zugang, um den Mikrocontroller neu zu programmieren, die serielle UART Schnittstelle um nach außen (z.B. zum PC) kommunizieren zu können, sowie eine SPI Schnittstelle, um das EtherSM von einem anderen SM aus (BTSM oder IOSM) ansteuern zu können.



BTSM Rev. 0.4



IOSM Rev. 0.1

EtherSM Rev. 0.3

AICU aus BTSM und EtherSM

Abbildung 4: AmbiComp SMs, 1. Generation

Um weiterhin mit der ACVM auf den SMs kommunizieren zu können, wurde das XModem Protokoll des BTKits übernommen. Um auf die serielle Schnittstelle zugreifen zu können, musste das SM dazu in ein spezielles Programmierboard, das sogenannte *DevBoard* eingesteckt werden. Dieses DevBoard machte alle Schnittstellen des Rückwandbusses nach außen zugänglich. Es musste auch verwendet werden, um eine neue ACVM Firmware auf das SM aufspielen zu können. Dieser Prozess konnte allerdings nur bei der Beecon durchgeführt werden.

Die ACVM, die auf den intelligenten Sandwichmodulen BTSM und IOSM zum Einsatz kommen konnte, wurde um den Microblue-Kommunikationsprotokollstapel der Beecon GmbH erweitert. Als wichtigste Neuerung unterstützte die ACVM sowohl Multithreading als auch Java Interfaces. Aus Effizienzgründen wurde nur ein einfaches, aber für die geplanten Anwendungen völlig ausreichendes Prioritätsscheduling implementiert. Außerdem standen seit dieser Version die Java-Mechanismen zur Thread-Steuerung zur Verfügung, insbesondere der Monitor-Mechanismus zur Implementierung kritischer Abschnitte in nebenläufigem Java-Code. Damit ging die ACVM in ihrer Leistungsklasse (8-Bit-Mikrocontroller mit 128kB ROM und 8kB RAM) deutlich über alle konkurrierenden Implementierungen hinaus.

Die Anbindung des Microblue-Kommunikationsprotokollstapels an die ACVM wurde an der Uni Karlsruhe durch eine Studienarbeit unterstützt:

- Malte Cornils, „Integration von Bluetooth in die AmbiComp Plattform für eingebettete Systeme“, Studienarbeit, Uni Karlsruhe, Februar 2008

Neben diesen Arbeiten an der Stabilität der ACVM wurde im ersten Halbjahr 2008 bei der Uni Karlsruhe die Codequalität, Stabilität und Standardkonformität des Transcoders weiter verbessert. Diese Verbesserungen sorgten dafür, dass dieser stabil lief und dabei fast alle Spracheigenschaften von Java unterstützen konnte, insbesondere die oben erwähnten Java-Interfaces.

4.1 BIOS Version 1.0

Zusammen mit der neuen Hardware wurde Anfang 2008 das sogenannte BIOS Konzept eingeführt. Das BIOS war notwendig, da der Projektverlauf bis dahin gezeigt hat, dass der ursprüngliche Plan einer monolithischen Firmware zu einer nicht mehr effizient beherrschbaren Komplexität führte. Der Grund für die zunehmende Komplexität entstand durch die Vielzahl der Varianten, in denen die drei bisherigen SMs zusammengesteckt werden konnten. Für jede dieser Varianten musste eine eigene, spezielle ACVM-Version entwickelt bzw. gebaut werden.

In enger Zusammenarbeit zwischen der Beecon GmbH und der Uni Karlsruhe wurde daher die Trennung der Firmware in ACVM und BIOS beschlossen und die BIOS Schnittstelle spezifiziert. Im Vergleich zu einer alternativ möglichen Trennung durch Einführung eines Betriebssystems bot der BIOS-Ansatz den Vorteil, auf einfache Weise die spezifischen Eigenschaften der Hardware von denen der ACVM zu trennen, jedoch die enge Verzahnung von Betriebssystemfunktionalitäten mit der ACVM beizubehalten.

Das Design dieser ersten Version des BIOS wurde in einer Studienarbeit an der Uni Karlsruhe ausführlich beschrieben:

- Sven Schlender, „AICU BIOS, Design und Implementierung einer Hardware-Abstraktions-Schicht für AmbiComp Sensorknoten“, Studienarbeit, Uni Karlsruhe, März 2008

Durch die Trennung in ACVM und BIOS war es notwendig, die ACVM von der Hardware zu trennen und auf das BIOS zu portieren. Die ACVM wurde dadurch von der tatsächlichen Hardware unabhängig und somit Quellcodekompatibel über verschiedene Hardware-plattformen hinweg. Neben der Beecon arbeitete auch die Uni Karlsruhe an der Anpassung der ACVM, speziell in Hinblick auf die nativen Methoden. Diese nativen Methoden bildeten die Schnittstelle zwischen der realen Hardware und der Java-Anwendung bzw. dem Java *Application Programming Interface* (API) (siehe unten).

Auf der Java Seite wurden die nativen Methoden in der AmbiComp-API verankert. Wenn eine dieser Methoden in einem Java-Programm aufgerufen wurde, führte die ACVM die dazugehörende native Methode in der ACVM aus. Innerhalb dieser Methode wurden dann die entsprechenden BIOS Aufrufe ausgeführt.

Das Setzen eines Pins auf dem IOSM konnte so beispielsweise mit folgendem Methodenaufruf in Java verwirklicht werden:

```
Gpio.setPinValues(int Pin, int Port, bool Value);
```

Innerhalb der nativen Methode in der ACVM wurden die Parameter ausgelesen und die benötigten BIOS Funktionen aufgerufen, beispielsweise:

```
bios_u8GetResIO(OUT, Pin, Port);// Reservierungsfunktion
bios_WriteIO(OUT, Pin, Port);   // Schreibfunktion
```

Des Weiteren ermöglichte die Einführung des BIOS den Projektpartnern, selbstständig ACVM Updates aufzuspielen. Hierzu konnte entweder ein Kommandozeilenwerkzeug oder das Eclipse-Plugin verwendet werden. In beiden Fällen wurde die neue Firmware über das oben beschriebene XModem Protokoll an das SM übertragen, dort vom BIOS empfangen und in

den Speicher des SMs geschrieben.

Neben diesen Arbeiten wurden die jeweils erkannten Fehler in der aktuellen Hardware-Generation behoben und vielfältige Unterstützung bei den Projektpartnern mit den Prototypen und Demonstratoren geleistet (siehe unten).

Parallel zur BIOS Entwicklung bei der Beecon wurde an der Uni Karlsruhe und später an der TU München die Grundlage für einen AmbiComp-Emulator unter Linux implementiert. Dazu wurde das BIOS auf Linux portiert und die benötigten Anpassungen und Neu-implementierungen am neuen Linux-Emulation-BIOS vorgenommen. Dadurch war es möglich, dieses BIOS - in seiner jeweils aktuellen Version - unter Linux auszuführen. Es stellte damit eine wichtige Grundlage für den Emulator dar.

Durch das Linux-Emulation-BIOS war es möglich, dass sich die ACVM, die aus ein und demselben Programmcode kompiliert wurde, fast völlig identisch verhielt, egal ob sie auf einer AICU oder in einer Linux-Emulation ablief. Da später auch das Eclipse-Plugin diesen Emulator ansprechen konnte, konnten viele Entwicklungsaufgaben bei der Uni Karlsruhe sowie bei den Projektpartnern wesentlich effizienter unter Linux erfolgen, als dies auf der Hardware möglich gewesen wäre.

Um den Projektpartnern möglichst rasch einen stabilen Stand der aktuellen Entwicklungen zur Verfügung stellen zu können, wurden seit Anfang 2008 im Sechswochenzyklus neue Releases der AmbiComp-Software veröffentlicht. Diese Releases enthielten jeweils die folgenden AmbiComp-Software-Komponenten in ihrer jeweils aktuellsten Version: die verschiedenen Sandwichmodul-BIOSe, die ACVM, die Emulationsumgebung für Linux, die System-bibliotheken der APIs, sowie die benötigten Zusatztools und -skripte.

4.2 Application Programming Interfaces (APIs)

Um Java-Programme für die neue SM-Generation schreiben zu können, wurde eine Programmierschnittstelle (Application Programming Interface, API) benötigt. Diese API sollte auf allgemeinen Java Standards basieren, den sogenannten Java Specification Requests (JSR). Hierzu wurden verschiedene Programmierschnittstellen evaluiert und als Beispielfall die JSR82 (Bluetooth) Schnittstelle an der Uni Karlsruhe prototypisch für die ACVM implementiert. Diese Schnittstelle setzt auf einem Teil des Microblue Kommunikationsstack der Beecon GmbH auf und ermöglicht es Java Programmen auf einer Bluetooth-fähigen AICU mit anderen Bluetooth-fähigen Geräten zu kommunizieren, beispielsweise mit Handys oder PDAs. Die Java-Seite der JSR82 API, die die nativen Methoden der ACVM verwendet, wurde am Fraunhofer IESE entwickelt.

Des Weiteren wurde die von SUN Microsystems veröffentlichte *Connected Limited Device Configuration* (CLDC) als AmbiComp Programmierschnittstelle ausgewählt und an die Erfordernisse der ACVM angepasst. Die CLDC enthält Basisfunktionalitäten, die üblicherweise unter Java bereit stehen, beispielsweise Zeichenkettenoperationen, mathematische Funktionen und grundlegende Datenstrukturen wie z.B. Hash-Tabellen. Damit besitzt AmbiComp eine, anderen Projekten vergleichbare, Programmierschnittstelle. Zusätzlich zur CLDC wurde die so genannte *AmbiComp MiniAPI* eingeführt. Sie enthält nur die wichtigsten API Funktionen, die eine Java Anwendung unbedingt benötigt. Die MiniAPI erlaubt es, Java Programme auch ohne CLDC zu kompilieren, zu transcodieren und auf der ACVM auszuführen. So konnte eine im Vergleich zur CLDC extrem kleine Codegröße erzielt werden. Mit der MiniAPI programmierte Java-Anwendungen sind allerdings nicht mehr Quellcode-kompatibel zu anderen Java-Systemen.

Von der CLDC und der MiniAPI unabhängig sind die weiteren API Funktionen, mit denen die für AmbiComp spezifischen Funktionalitäten wie beispielsweise General Purpose Input Output (GPIO) angesprochen werden. Hierzu wurde bei der HdM die AmbiComp-spezifische API entwickelt. Diese AmbiComp-API enthält unter anderem alle Teile, die direkt einem SM zugeordnet werden können. So können hierüber z.B. komfortabel auf die einzelnen Ports und

Pins des IOSMs zugegriffen oder wie vom Entwickler gewohnt Netzwerk-Funktionalitäten des Ethernet-Moduls verwendet werden. Für die Einbindung der Ethernet-Schnittstelle (EtherSM) wurde eine rudimentäre UDP-Socket-Schnittstelle implementiert. Damit können AICUs über Ethernet mit anderen Internet-fähigen Geräten kommunizieren.

Bei der AmbiComp-API Entwicklung traten immer wieder grundlegende Fragen auf, deren Lösung konzeptionelle Arbeiten und Diskussionen mit den Projektpartnern, speziell der Uni Karlsruhe und der HdM, notwendig machten. So wurde zum Beispiel das Problem des konkurrierenden Zugriffs auf Ressourcen untersucht und es wurden entsprechende Mechanismen bei der Implementierung eingeführt, z.B. um einen Pin zu reservieren. Des Weiteren wurde ein Konzept erarbeitet, um auf Ereignisse an bestimmten Ressourcen reagieren zu können. Dieses Konzept entspricht dem bekannten Konzept der Listener, das in vielen Java-APIs verwendet wird. Über solche Listener ist es einem Entwickler z.B. möglich, auf steigende oder fallende Flanken an digitalen Pins oder auf das Erreichen eines bestimmten Schwellwertes bei analogen Pins zu reagieren.

Durch die im Projektverlauf nach und nach ergänzten Funktionen wurde die Komplexität der AmbiComp-API zunehmend größer. Damit wuchs auch die Codegröße der Anwendungen, die diese API verwendeten. Dadurch wurden selbst einfache Programme für den beschränkten Speichers auf dem IOSM zu groß. Dies führte dazu, dass diese AmbiComp-API der HdM leider für das Projekt ungeeignet wurde und im weiteren Projektverlauf nicht mehr verwendet werden konnte.

Trotzdem war diese Entwicklung hilfreich bei der Weiterentwicklung der nativen Methoden, indem sie dazu führte, dass verschiedene Möglichkeiten der Verbindung zwischen Java Anwendung und Hardware bzw. BIOS untersucht wurden. Die verschiedenen untersuchten Möglichkeiten reichten vom reinen Durchreichen der BIOS Funktionen bis hin zu einer vollständigen Verwaltung der GPIO Funktionalität innerhalb der ACVM. Um den Speicherbedarf der ACVM möglichst gering zu halten, wurde letztendlich der Ansatz gewählt, die BIOS Funktionalität so weit wie möglich direkt in der Java API abzubilden.

Nachdem sich die erste Version der AmbiComp-API als zu groß herausgestellt hatte, wurde die API für das IOSM und das EtherSM neu implementiert. Da, wie eben geschildert, im Wesentlichen die BIOS Schnittstelle in die Java-Ebene durchgereicht wurde, beanspruchte diese Implementierung nicht viel Zeit. Dementsprechend genauer muss allerdings ein Entwickler die zu übergebenden Parameter bzw. die Rückgabewerte kennen. Hierzu erstellte die HdM eine umfangreiche Dokumentation, speziell für diese hardwarenah zu nutzenden Methoden. Dies war insbesondere für die Methoden notwendig, die direkt die Input-Output Funktionalität der einzelnen SM ansprachen.

4.3 Eclipse-Plugin

Durch die erste Generation der neuen Sandwichmodule, durch die erweiterte ACVM und durch das neu eingeführte BIOS ergab sich bei der HdM die Notwendigkeit, das Eclipse-Plugin an die geänderten Bedingungen anzupassen. Diese Anpassungen betraf sowohl die Kommunikation mit der Hardware als auch die Programmierung von Anwendungen durch die Einführung der neuen APIs.

Um verschiedene Informationen aus dem BIOS und der ACVM darstellen zu können, wurde das Eclipse-Plugin entsprechend angepasst. Weiterhin wurden Anpassungen vorgenommen, die dazu dienten, genauere Angaben zur Analyse und Fehlerbehebung des Eclipse-Plugins selbst ausgeben zu lassen.

Um die Kommunikation zwischen Eclipse-Plugin (bzw. Entwicklungsrechner) und AICU auszugeben, wurde eine zusätzliche Sicht, die *AICU Monitor View* implementiert und in das Eclipse-Plugin integriert. Hier kann der Entwickler alle Ausgaben der ACVM und alle an die AICU gesendeten Befehle mitverfolgen. Über ein Menü kann zunächst die Verbindung ausgewählt werden, deren Kommunikationsfluss dargestellt werden soll. Die Ansichten

wurden im späteren Projektverlauf noch erweitert, um den noch freien Speicherplatz auf dem ausgewählten AICU je nach AmbiComp-Projekt anzuzeigen. So kann der Entwickler schon während der Programmierung abschätzen, ob seine Anwendung auf der angeschlossenen AICU genügend Speicherplatz findet.

Parallel dazu wurde ein so genannter *Simulation-Manager* entwickelt, der eine beliebige Anzahl von Linux-Emulator Instanzen erzeugen und verwalten kann. Er bildet die Grundlage für die Emulation eines Netzes aus AICUs.

Außerdem zog die Einführung der MiniAPI und CLDC-API einige Änderungen im Eclipse-Plugin nach sich. Unter Anderem kann der Entwickler in dieser neuen Version des Plugins beim Anlegen eines neuen Projektes im AmbiComp Project Wizard auswählen, welche API-Variante er nutzen möchte. Die richtigen Bibliotheken werden dann automatisch eingebunden und für den Compilier- und Transcodier-Vorgang genutzt. Weiterhin hat sich gezeigt, dass es oft sinnvoll ist, auch nach Anlegen des Projektes in die andere API-Variante zu wechseln. Dies wird über projektspezifische Einstellungen realisiert (Property Pages), in denen der Entwickler zwischen MiniAPI und CLDC-API hin- und herschalten kann.

4.4 Erste Demonstratoren

Auf Basis der CLDC und der MiniAPI, sowie unter Verwendung der neuen, schlanken AmbiComp API für den Hardwarezugriff, wurden einige erste Beispielanwendungen umgesetzt. Im Einzelnen handelte es sich zum Beispiel um einen an der HdM implementierten Demonstrator, bestehend aus einer Kommunikationsstrecke mit Bluetooth und Ethernet und einem angeschlossenen LC-Display. Die HdM entwickelt nicht nur das Demonstratorkonzept, sondern auch die benötigten Java-Treiber für das LC-Display und die Client- und Server-applikationen (siehe Abbildung 5). In diesem Zusammenhang wurden auch noch diverse weitere Anwendungen und Treiber für verschiedene Roboternetz Hardwarekomponenten implementiert, welche an die AICUs angeschlossen werden konnten.

Bei der HdM wurde als Ergänzung zu diesem Demonstrator eine AmbiComp Handy-anwendung entwickelt, welche mit AmbiComp Komponenten über Bluetooth kommunizieren kann. Das Handy führt dabei ein JavaME-Programm (Midlet) aus, das sich über das L2CAP-Protokoll per Bluetooth mit der AICU des oben beschriebenen Demonstrators verbinden und eingegebene Texte versenden kann. Diese werden dann über UDP auf die zweite AICU übertragen und dort auf dem LC-Display ausgegeben (siehe Abbildung 5).

Zusätzlich wurde an der HdM eine AmbiComp-Eclipse-Emulator Demonstration entwickelt, welche zeigte, wie emulierte AICUs in Eclipse verwendet werden können.

Das Fraunhofer IESE hat weiter an dem Feuermelder-Simulator, welcher die kooperativen Feuermelder visualisiert, gearbeitet. Dieser Simulator implementierte und simulierte einige der angedachten AmbiComp Kommunikationskonzepte auf einem handelsüblichen PC. Dabei wurde die bestehende Simulationsumgebung um die neuen AmbiComp AICUs der ersten Generation und reale Feuermelder erweitert. Um ein realistischeres eingebettetes System einsetzen zu können, wurden durch einen neu entwickelten Adapter zusätzliche Signale des handelsüblichen Rauchmelders zugänglich gemacht. Es konnten dadurch zusätzlich Status-informationen zum Zustand der Batterie und der Rauchkammer über das IOSM einer AICU ausgelesen, sowie Selbsttests initiiert werden. Ferner wurden am Fraunhofer IESE Programmierarbeiten zur Unterstützung bei der Implementierung der AmbiComp-API, sowie zur Bereitstellung von Scripts zur Generierung der API-Dokumentation beigetragen.

Bei der Entwicklung der verschiedenen Demonstratoren kam neben den AICUs auch das Eclipse-Plugin zum Einsatz, so dass hier eine stetige Überprüfung der verfügbaren Funktionalitäten durchgeführt und dabei gemachte Erfahrungen den Partnern direkt zurückgemeldet werden konnten. Dadurch wurden wertvolle Erfahrungen gewonnen und z.B. die Bedienweise des Eclipse-Plugins optimiert oder Fehler in der Implementierung der nativen Funktionen behoben. Durch die Demonstratoren als Ganzes konnten zusätzlich einige Fehler

im Zusammenspiel der Komponenten erkannt und in Zusammenarbeit mit den Projektpartnern behoben werden.

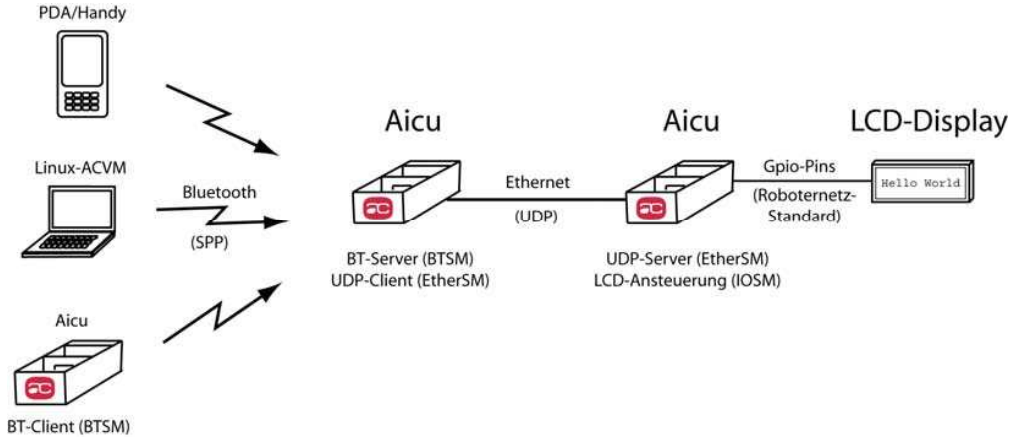


Abbildung 5: Demonstratoraufbau

Diese Demonstratoren zeigten aber bei weitem noch nicht das Potential von AmbiComp, sondern nur die einfache Programmierung der AICUs. Hierbei wurden jedoch wertvolle Erfahrungen gewonnen, was wiederum Auswirkungen die APIs, aber auch auf die grafische Oberfläche des Eclipse-Plugins hatte.

Im Rahmen eines Workshops des EU-Projektes „ESTIA“ wurde am 12.03.2008 das AmbiComp Projekt bei Alcatel-Lucent in Stuttgart vorgestellt. Bei dieser Gelegenheit hielt Herr Dr. Fuhrmann einen Vortrag mit dem Titel „AmbiComp - Ambient Computing for Sensor and Home Networks“. Mit den oben erwähnten Demonstratoren konnte bei diesem Workshop einem internationalen, fachkundigen Expertenkreis erste AmbiComp-Hardware und -Software vorgestellt werden.

Diese oben beschriebenen Beispielanwendungen wurden zusätzlich am 29.04.2008 einem potenziellen assoziierten Partner, der Firma Rohr Technische Informatik GmbH, vorgestellt. Leider entwickelte sich keine weitere Zusammenarbeit über diese Treffen hinaus, da der zu dem Zeitpunkt vorhandene AmbiComp Stand nicht ausreichend genau zu den Anforderungen der Firma Rohr passten.

4.5 Konzeptionelle Arbeiten

In Hinblick auf die zukünftige Verwertung von AmbiComp wurden von Alcatel-Lucent, in Zusammenarbeit mit den anderen Partnern, Überlegungen zu den veschiedenen Rollen in der Wertschöpfungskette angestellt. Dieser Schritt war wichtig, um einerseits ein gemeinsames Verständnis dieser Rollen zu erzielen, andererseits um die verschiedenen Zielgruppen, beispielsweise die Hersteller der Hardware und der Basis-Software, die Anwendungs-Entwickler, Vertriebsorganisationen, usw. entsprechend adressieren zu können. Durch diese Klarheit über die verschiedenen Rollen konnte dann bei Präsentationen vor Publikum zielgerichtet auf diese eingegangen werden.

Am Fraunhofer IESE wurde im ersten Halbjahr 2008 weiter am Komponentenmodell, dessen Instantiierung für den AmbiComp-Kontext und dem Feuermelder-Demonstrator gearbeitet.

Das im Projektverlauf entwickelte abstrakte Komponentenmodell sollte den Entwurf dynamisch-adaptiver Systeme durch Definition geeigneter Architekturen, Schnittstellen und Infrastruktur-Komponenten unterstützen. Um das Modell als Teilergebnis des Projekts auch für andere Anwendungskontexte verfügbar zu machen, wurde dabei besonders auf die

Unabhängigkeit von konkreten Technologien (wie etwa Java) geachtet. Des Weiteren wurden Arbeiten zur Unterstützung des Komponentenmodells durch das Fraunhofer IESE-Werkzeug ComposeR begonnen, das die Spezifikation und Analyse von komponenten-basierten Architekturen erlaubt. Darüber hinaus wurde das Modell selbst weiterentwickelt. Dazu gehören z.B. eine Charakterisierung des verwendeten Adaptivitätsbegriffs, die Einführung eines hierarchischen Modells zur Reflektion von Umweltaspekten, das benötigte Umgebungsinformationen so lokal wie möglich zu halten versucht und eine Lösung für den automatischen Start von Diensten. Um das Komponentenmodell für AmbiComp verwendbar zu machen, mussten seine Konzepte auf die von AmbiComp verwendeten Technologien abgebildet werden. Zu diesem Zweck wurde eine Reihe von typischen Anwendungsszenarien gesammelt, aus denen letztlich Anforderungen an die Plattform abgeleitet werden konnten. Des Weiteren wurden bereits einige der abstrakten Konzepte des Modells auf die AmbiComp-Welt, d.h. insbesondere auf die Verwendung von Java, abgebildet.

An der HdM wurden einige konzeptionelle Arbeiten durchgeführt, um das Debugging von Anwendungen auf der ACVM zu erleichtern. Unter anderem wurde ein Vergleich mit der „KVM Debugger Architecture“ für das Debuggen von Anwendungen auf der von Sun stammenden „Kilobyte Virtual Machine“ (KVM) durchgeführt. Die KVM ist die Sun VM für die „Java 2 Micro Edition, Connected Limited Device Configuration“ (J2ME CLDC) und wird z.B. auch auf Mobiltelefonen eingesetzt. Als Ergebnis dieser Arbeiten wurde ein Konzept vorgestellt, das beschreibt, wie ein Debugging Framework in AmbiComp aussehen könnte. Dieses nutzt einen Proxy, um die Funktionalität, die die kleine ACVM nicht leisten kann, bereitzustellen bzw. zu simulieren. Die so benannte „AmbiComp Debugger Architecture“ wurde zwischen den Projektpartnern diskutiert und führte unter anderem zu der Einsicht, dass in AmbiComp zwar ein Java-Debugger benötigt wird, dies aber zum damaligen Zeitpunkt erst einmal zurückgestellt werden musste, weil anderen Arbeiten für den Projekterfolg wichtiger erschienen. Zu einem späteren Zeitpunkt nahm die Uni Karlsruhe dieses Thema dann aber wieder auf und entwickelte das „AmbiComp Debug Wire Protocol“ (ACDWP) (siehe unten).

Zusätzlich wurde eine Zusammenarbeit zwischen der Uni Karlsruhe und der Uni Freiburg begonnen. Im Rahmen dieser Zusammenarbeit wurden neue Konzepte zur Synchronisierung und Konsistenzsicherung verteilt ablaufender Java Programme untersucht. Ziel war es, die Probleme bei der Umsetzung der in Java vorhandenen Locking- bzw. Monitor-Mechanismen zum wechselseitigen Ausschluss durch Verwendung eines neuartigen Transaktionskonzepts zu umgehen. Diese Arbeiten dauern auch über das Projektende hinaus an und werden derzeit im BMBF Projekt „J-Cell“ weiter verfolgt. Sie untersuchen die Konsistenz von Objekten bei verteilten mehrfädigen Anwendungen. Bei diesem neuen Paradigma des „Software Transactional Memory“ (STM) werden keine Sperren mehr auf Objekte gesetzt, vielmehr werden alle Objektzugriffe in Transaktionen gekapselt. In Erweiterung der von Dritten publizierten Verfahren wurde bei den Diskussionen in AmbiComp ein Ansatz verfolgt, der völlig dezentral arbeitet. Teile dieser Arbeit wurden in einer Diplomarbeit genauer untersucht, die sich mit derartigen Konsistenzverfahren auf Basis von Software Transactional Memory (STM) befasste:

- Clemens Koller, „Software Transactional Memory for the AmbiComp distributed Java Virtual Machine”, Diplomarbeit, Uni Karlsruhe, Februar 2009

Im August 2009 wurde eine ausführlichere Diskussion dieser Konzepte in einem technischen Bericht an der TUM veröffentlicht. Die Arbeiten wurden später dann auch auf der IPDPS’10 veröffentlicht:

[STM] *Annette Bieniusa and Thomas Fuhrmann, „A Description and Analysis of ’dstm’, a Fully Decentralized Software Transactional Memory Algorithm“,Technical Report TUM-I0920, August 2009*

[IPDPS10] *Annette Bieniusa and Thomas Fuhrmann, „Consistency in Hindsight, A Fully Decentralized STM Algorithm”, Proceedings of the 24th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2010), April 19-23, 2010, Atlanta, Georgia, USA*

Eine weitere Diplomarbeit untersuchte die Charakteristiken und den Zugriff auf verteilte Objekte, die sogenannten *Globally Accessible Objects* (GAOs). Dabei ging es vorrangig um die effiziente Speicherung und verschiedene Zugriffsmechanismen innerhalb der ACVM.

- Alexander Kiening, „Accessing Remote Objects in a Distributed Embedded Java VM“, Diplomarbeit, TU München, Mai 2008

Eine dritte Diplomarbeit untersuchte eine Modifikation des SSR-Protokolls, die trotz Objektmigrationen den Netzwerkverkehr zur Aufrechterhaltung der SSR-Konsistenz begrenzte. Kernidee dabei war, dass SSR im AmbiComp-Kontext zwischen zwei verschiedenen Graphen vermitteln muss: dem durch die Netzwerktopologie gegeben Netzwerkgraphen und dem durch die Referenzen zwischen den Objekten einer Anwendung gegebenen Objektgraphen. Kern der Arbeit war es, das Scalable Source Routing Protokoll so zu erweitern, dass die Erreichbarkeit eines GAOs sichergestellt ist, auch wenn es auf eine andere AICU verlagert wurde.

- Mathias Kellerer, „Extension of the Scalable Source Routing Protocol for Addressing and Migrating Global Accessible Objects“, Diplomarbeit, TU München, Dezember 2008

Daneben wurde das SSR Protokoll selbst ständigen Verbesserungen unterzogen. Durch Berücksichtigung der Broadcast-Eigenschaft wie sie in Funkmedien, beispielsweise dem IEEE 802.15.4 MAC Layer, vorhanden ist, wurde die Stabilität und Effizienz der Kommunikation verbessert. Diese Verbesserung wurde 2009 in einer Fachpublikation veröffentlicht:

[IWCMC09] Pengfei Di and Thomas Fuhrmann, „Using Link-Layer Broadcast to Improve Scalable Source Routing“, Proceedings of the International Conference on Wireless Communications and Mobile Computing (IWCMC'09), Leipzig, Germany, June 2009

5 Zweite Sandwichmodulgeneration

Nachdem die erste Generation Sandwichmodule Anfang 2008 an die Projektpartner ausgeliefert wurde, begannen bei der Beecon GmbH die Vorplanungen für die zweite AICU Generation auf Basis eines neuen Rückwandbusses. Als neues Grundprinzip wurde nun jedes Sandwichmodul mit einem Mikrocontroller ausgestattet, d.h. jedes Modul konnte aktiv an der Kommunikation innerhalb des Sandwichmodulstapels teilnehmen. Zuvor konnten nur intelligente SMs miteinander kommunizieren. Passive SMs mussten mit passenden intelligenten SMs kombiniert werden, was die Flexibilität der Kombinationen deutlich eingeschränkt hatte.

Abbildung 6: Neues Rückwandbuslayout

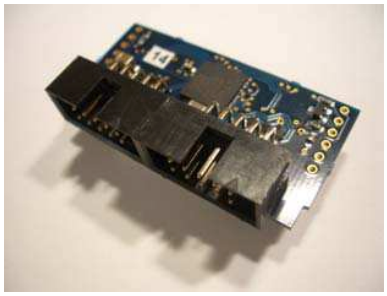
Die Kommunikation zwischen den Modulen innerhalb einer AICU wurde dafür mittels eines I2C Busses auf dem Rückwandbus realisiert. So wurde eine im Vergleich zu der bis dahin verwendeten seriellen UART Kommunikation deutlich höhere Übertragungsgeschwindigkeit (400kBit/s statt 115kBit/s) erzielt. Außerdem hatte diese Lösung den Vorteil eines erweiterten Adressierungsschemas (Punkt-zu-Punkt und Rundspruch auf dem Bus), sowie einer physisch gegebenen Quittierung der Datenrahmen.

Die zweite SM-Generation behielt den bisherigen Formfaktor bei, veränderte aber nicht nur die Signale des Rückwandbusses sondern auch seine physische Platzierung, um so die mechanische Steifigkeit des Modulstapels zu erhöhen und das Layout der Schaltung der einzelnen Module zu erleichtern, siehe Abbildung 6.

Insbesondere stand damit mehr Platz für Ein- und Ausgabeschnittstellen am vorderen Rand der Module zur Verfügung.

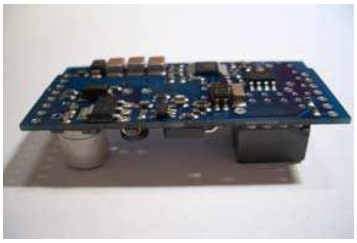
Nachdem diese Vorplanungen abgeschlossen waren, begann die Beecon mit der Entwicklung dieser zweiten Generation Sandwichmodule. Mitte 2008 standen die ersten Einzelstücke bei der Beecon und der Uni Karlsruhe zur Verfügung und konnten von diesen beiden Projektpartnern getestet werden. Anschließend, ab Ende 2008, wurden die neuen SMs dann auch an die anderen Projektpartner ausgeliefert.

BTSM Rev. 1.3, HCI & SPP



IOSM Rev. 1.2

EtherSM Rev. 1.0.2



PoESM Rev. 1.1

Abbildung 7: AmbiComp SMs, 2. Generation

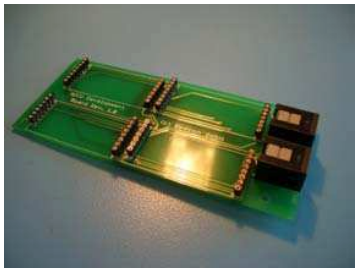
Die bestehenden Sandwichmodulen BTSM, EtherSM und IOSM wurden komplett überarbeitet und an das neue Rückwandbuslayout angepasst. Während dieses Re-Designs wurde das EtherSM um einen 512kB großen Zusatzspeicher erweitert und zusätzlich mit einem eigenen Prozessor bestückt.

Außerdem wurden die folgenden neuen SMs entwickelt:

- **Power-Over-Ethernet-Modul (PoESM):** Das PoESM stellt in Verbindung mit einem EtherSM eine weitere Stromversorgung über Power-over-Ethernet zur Verfügung.
- **Backplane Primary Supply Sandwichmodul (BPPRISM):** Das Backplane Primary Supply Sandwichmodul stellt die Stromversorgung für die anderen Sandwichmodule innerhalb eines AmbiComp-Systems bereit. Als Quelle kann dabei ein Netzteil mit Gleichstromausgang oder eine Batterie dienen.
- **Backplane-Sandwichmodul (BPSM):** Das BPSM stellt den AICU-Bus für Debugging-Zwecke auf einem RJ12-Steckverbinder zur Verfügung, um von außen mit dem AICU kommunizieren zu können.

- **BTSM SPP (BTSM_SPP) und BTSM Class 1 (BTSM_C1):** Zusätzlich zu dem BTSM, welches mit einem HCI Bluetooth-Modul bestückt ist, wurde ein neues BTSM entwickelt, welches mit einem SPP (Serial Port Profile) Bluetooth-Modul bestückt wurde. Dieses neue Modul stellt eine standardkonforme serielle Schnittstelle über Bluetooth zur Verfügung. Baulich ist kein Unterschied zwischen den beiden Bluetooth-Module zu erkennen, sie unterscheiden sich äußerlich nur anhand einer Markierung des Gehäuses. Des Weiteren wurde ein BTSM HCI mit einem Class-1-Bluetooth-Modul mit externer Antenne entwickelt. Dieses BTSM bietet eine Reichweite von bis zu 100m.
- **AkkuSM:** Das Ende 2009 als Prototyp fertig gestellte AkkuSM ermöglicht neben Steckernetzteil und Power-over-Ethernet eine dritte Art der Stromversorgung.

Des Weiteren wurden ein neues Entwicklungsboard (DevBoard) mit vier Steckplätzen für Sandwichmodule, sowie Anschlüsse für die Stromversorgung und den Zugriff auf den Rückwandbus entwickelt.



DevBoard Rev. 1.0

AkkuSM Rev. 1.0

Abbildung 8: AmbiComp-Hardware, 2. Generation

Abweichend von der ursprünglichen Vorhabensbeschreibung basieren alle diese Sandwichmodule auf derselben Mikrocontroller-Familie, dem Atmel AVR 8-Bit-Mikrocontroller. Die Arbeiten eines Studenten an der Portierung des BIOS auf den ARM-Mikrocontroller brachten leider keine Ergebnisse. Die Portierung wurde bis zum Ende des Projektes nicht mehr weiter verfolgt, da andere Aufgaben im Mittelpunkt standen und die Arbeitskraft der Projektmitarbeiter gebunden haben. Das gleiche gilt für die Integration von 802.15.4 („ZigBee“). Auch hier konnte das Thema nach dem Scheitern der entsprechenden studentischen Arbeit nicht von den Projektmitarbeitern aufgegriffen werden.

Neben der Entwicklung der neuen Sandwichmodulgeneration waren bei der Beecon auch immer wieder Arbeiten für die Projektpartner erforderlich, beispielsweise um sie bei der Nutzung der bisherigen ersten Hardwaregeneration zu unterstützten. Hierzu zählten unter anderem die zeitnahe Beseitigung von erkannten Fehlern, sowie die Anleitung zum allgemeinen Umgang mit den einzelnen Software- und Hardwarekomponenten. Hierdurch wurde die Dokumentation der einzelnen Teile signifikant verbessert und vereinfacht, so dass nun auch unerfahrene Benutzer schnell einen Zugang zu der Verwendung der AmbiComp-Technologie finden können.

Neben Software-Fehlern bei der Implementierung von ACVM und BIOS traten auch immer wieder Hardware-Fehler auf. Anders als die Software-Fehler wurden die Hardware-Fehler aber nicht durch Entwurfs- bzw. Implementierungsfehler verursacht, sondern waren Folge von mechanischer und elektrostatischer Einwirkung auf die AICUs. Derartige Fehler konnten durch umfangreiche Tests bei der Einführung der zweiten Generation der Sandwichmodule vermindert werden.

5.1 Hardware-BIOS Version 1.1

Einhergehend mit den Arbeiten an der zweiten SM Generation wurde bei der Beecon GmbH die BIOS Version 1.1 entwickelt. Diese Version unterstützte erstmals das neue InterSM-Backplane Kommunikationskonzept über den I2C-basierten Rückwandbus. Darüber hinaus enthielt sie zahlreiche neue Funktionen zur besseren Unterstützung der ACVM, beispielsweise Zeitgeber (Timer). Neue Low-Level-Treiber sprechen die verschiedenartige Kommunikations-hardware (Ethernet, Bluetooth, etc.) mit einem einheitlichen, Ereignis-basierten Stream-Konzept an. Eine ausführliche Dokumentation beschreibt alle Funktionen des BIOS im Bezug auf die verschiedenen Varianten der Sandwichmodule. Diese Dokumentation wurde den Projektpartnern über das interne Wiki zeitnah zur Verfügung gestellt.

Durch den Wechsel vom alten UART-Kommunikationskanal zum neuen, I2C-basierten Rückwandbus änderte sich der Zugriff auf die Sandwich-Module grundlegend. Um von einem Linux-Rechner aus auf ein SM zugreifen zu können, wurde von nun an der Schnittstellenadapter *Aardvark* der Firma TotalPhase verwendet. Er ermöglicht die Verbindung des I2C-basierten AICU Busses mit der USB Schnittstelle eines PCs, d.h. man kann damit ein Sandwichmodul an einen Linux bzw. Windows-Rechner anschließen. Für die Ansteuerung des Aardvark wurde in Zusammenarbeit zwischen der TU München und der Beecon GmbH das so genannte I2C-Access Programm implementiert, über das mit dem BIOS der Sandwichmodule kommuniziert werden kann.

Durch die neue I2C-Schnittstelle können nun alle Steuerkommandos an das BIOS und die ACVM, wie auch alle Ausgaben der ACVM inkl. Debug-Ausgaben, in einer einheitlichen Weise über den Rückwandbus der AICUs laufen. Zu diesem Zweck wurde ein Protokoll spezifiziert und implementiert, das erstens mit einem selbstorganisierten Adressvergabeverfahren alle SMs eindeutig adressierbar macht und zweitens die Kommunikation gegen Bitfehler im Kanal absichert. Zu diesen Protokollen wurde zusätzlich die Gegenstelle für den AmbiComp-Emulator implementiert.

Durch die neue BIOS Schnittstelle wurde das Aufspielen neuer Firmware wesentlich überarbeitet, so dass dabei weniger Fehler auftreten können. Außerdem wurden durch das neue BIOS 1.1. auch BIOS-Updates möglich. Grundlage dazu ist eine besondere Firmware, die das neue BIOS zunächst in den Anwendungsspeicher des SMs lädt und dann von dort aus das neue BIOS in den Flashspeicher schreibt. Durch diese Neuerung konnte im Regelfall die gesamte Firmware der AICUs (BIOS, ACVM) direkt im Zielgerät aktualisiert werden, ohne dass dabei Bedienfehler dauerhafte Schäden verursachen konnten. Dadurch war es den Projekt-partnern ab jetzt möglich, die gesamte Firmware selbstständig auszutauschen. Dies war den bis zu diesem Zeitpunkt nur für die ACVM der Fall.

Im Rahmen dieser Entwicklungen wurde auch das Portfolio der Testprogramme erweitert. Das sogenannte AppTest-Programm setzt auf dem BIOS auf und testet die grundlegenden BIOS Funktionalitäten sowie die darunterliegenden Hardwarefunktionen, wie z.B. die LEDs eines SMs. Dieser AppTest dient vorrangig zur Inbetriebnahme neuer SMs und zum Finden von Fehlern in der Hardware. Zu Beginn einer neuen BIOS Version wurde aber auch jeweils das BIOS mit diesem Programm getestet und auf Fehler überprüft. Neben den Compliance-Tests, die die ACVM, den Transcoder und die Systembibliotheken testen, vervollständigt der AppTest das Portfolio von Testprogrammen, indem es BIOS- und Hardware-Funktionen auf Einhaltung der Spezifikation testet.

5.2 Linux-Emulation-BIOS 1.1 und Linux-Toolchain

Parallel zu der Entwicklung des BIOS für den AVR-Mikrocontroller wurde an der TU München die Entwicklung des Linux-Emulation-BIOS für die Version 1.1 vorangetrieben. Dabei wurde die Funktionalität des Hardware-BIOS 1.1 in das Linux-Emulation-BIOS 1.1 überführt, das als Grundlage für den Linux-Emulator dient. Dieser Emulator und das BIOS sollten nicht nur die Funktionalität des Hardware-BIOS bereitstellen, sondern das Verhalten der AVR-Hardware so genau wie möglich emulieren.

Ein wesentlicher Bestandteil dabei war es, die gesamte Kommunikation (Rückwandbus, Ethernet, Bluetooth, etc.) unter Linux zu emulieren, so dass sich eine ACVM unter Linux genauso verhält, wie es auf der realen AICU-Hardware der Fall wäre. Hierzu wurden alle Linux-BIOS plus Linux-ACVM Entitäten an den DBUS (=Systembus) des Linux-Betriebssystem angeschlossen. Die einzelnen Kommunikationskanäle wurden über DBUS-Dienste verbunden und über die sogenannten „Gates“ (I2C-Gate, Bluetooth-Gate, Ethernet-Gate) zur Verfügung gestellt. So war es möglich, mehrere AICUs und/oder SMs unter Linux zu konfigurieren und zu emulieren. Als Schnittstelle für den einfachen Zugriff auf die unter Linux emulierten AICUs sowie auf die eventuell angeschlossene reale Hardware wurde in Zusammenarbeit der TU München und der HdM das so genannte Otto-Protokoll spezifiziert und anschließend an der TU München implementiert. Diese Implementierung enthält alle erforderlichen Konfigurationswerkzeuge und ermöglicht es zusätzlich, über die so genannte Monitor-Komponente („i2cmonitor“, später „acmonitor“), die Kommunikation auf sämtlichen DBUS-Kanälen für Debugging-Zwecke anzuzeigen und ggf. aufzuzeichnen.

5.3 ACVM-Firmware Anpassungen

Nachdem das neue BIOS 1.1 fertig gestellt war, begann die Uni Karlsruhe/TU München damit, sowohl die ACVM als auch die damit zusammenhängenden Kommunikationsbibliotheken (Ethernet- und Bluetooth-Library) auf das neue BIOS 1.1 umzustellen. Durch diesen Schritt wurde die ACVM-Firmware noch unabhängiger von der jeweiligen Sandwichmodul-Hardware als dies mit dem BIOS 1.0 der Fall war. Viele Kommunikationskonzepte wurden verallgemeinert und so deren Nutzung aus der ACVM heraus vereinfacht. Außerdem wurde der neue Rückwandbus („Backplane“) über die neue *Backplane-Library* voll in das Konzept der ACVM integriert.

Aus Sicht der Java-Programme, die auf einer AICU ablaufen, wurden alle diese Kommunikationskanäle einheitlich als so genannte *blockierende Ressourcen* ausgeführt, so dass in Zusammenarbeit mit dem ACVM-Scheduler Threads blockierend auf Eingaben, beispielsweise Ethernet-Frames, warten können. Pro Kommunikationskanal kann dadurch immer nur ein einziger Thread warten. Sollen in einem Javaprogramm zum Beispiel verschiedene Threads jeweils eine eigene UDP-Verbindung auf einem eigenen UDP-Ports aufbauen können, so muss das etwaige Aufspalten von ankommenden UDP-Packeten in diese verschiedene Kanäle (das Demultiplexing) innerhalb von Java, z.B. in einer eigenen API, erfolgen.

Durch dieses Design wurde eine saubere Trennung der in C++ geschriebenen Kommunikationsbibliotheken und der Java-Seite erreicht. Damit der Zugriff auf die verschiedenen, von den einzelnen SMs angebotenen Funktionalitäten aus Java heraus möglich ist, wurden weitere Gruppen von nativen Methoden, z.B. für die verschiedenen Streams (z.B. Ethernet, Backplane, Bluetooth HCI und SPP, Console) innerhalb der ACVM implementiert. Hier stehen mit Projektende alle erforderlichen Zugriffsfunktionen für die Kommunikations- und I/O-Schnittstellen zur Verfügung. Während sich die Beecon GmbH dabei vor allem auf die Kernfunktionalität der ACVM konzentriert hat, übernahm die TUM die Implementierung der nativen Methoden der ACVM und der Java Bibliotheken.

Als wesentliche Neuentwicklung im Kern der ACVM ist die Erweiterung auf die Ausführung mehrerer Blobs hinzugekommen. Der Transcoder kann nun Blobs erzeugen, die gegen andere Blobs binden, d.h. ein Blob muss nicht mehr den gesamten Code der Anwendung enthalten. Die ACVM lädt dann alle Blobs, die für die Anwendung erforderlich sind und bildet die darin enthaltenen Klassenreferenzen aufeinander ab. So können mehrere Anwendungen gleichzeitig in einer ACVM laufen.

Die durch die Arbeiten an der ACVM erkannten Wechselwirkungen mit dem BIOS wurden in Form einer Verfeinerung und Schärfung der BIOS-Schnittstelle umgesetzt.

5.4 AmbiComp Debug Wire Protocoll (ACDWP)

Um das Debugging von Java-Anwendungen auf der Java-Ebene zu ermöglichen, wurden die Vorarbeiten der HdM zum Debugging-Proxy an der Uni Karlsruhe wieder aufgenommen. Hierzu wurde an der Uni Karlsruhe/TU München zusammen mit der Beecon das *AmbiComp Debug Wire Protokoll* entwickelt. Es erlaubt dem Java-Debugger einer Java-Entwicklungsumgebung direkt auf den internen Zustand der ACVM zuzugreifen und diesen auch in gewissem Rahmen zu verändern. Hierbei läuft der Debugger beispielsweise auf einem PC und die ACVM auf einem Sandwichmodul.

Da die ACVM auf dem Sandwichmodul selbst nicht genügend Ressourcen zur Verfügung stellen kann, um ein vollständiges Debugging der Anwendung zu gewährleisten, wird auf dem PC, der den Debugger ausführt ein so genannter *Debug-Proxy* gestartet, der sich mit der ACVM auf dem SM verbindet.

Da einige Anfragen des Debuggers auf nicht veränderliche Werte der Anwendung, wie beispielsweise Zeilennummern im Quellcode oder Namen von Methoden, Felder und Variablen, zugreifen, können diese auch direkt vom Debug-Proxy beantwortet werden. Alle Anfragen, die der Proxy nicht selbst beantworten kann oder die die Programmausführung beeinflussen (z.B. das Setzen von Breakpoints), werden vom Proxy an die ACVM weitergeleitet.

Hierzu wurde an der TU München der Transcoder um die Funktionalität eines Reverse-Transcoders erweitert, um so den Debug-Proxy und damit das Debugging von AmbiComp-Anwendungen zu unterstützen.

Der Reverse-Transcoder erzeugte dafür zusätzlich zum normalen Blob den sogenannten Debug-Blob. Dieser enthält zusätzliche Debug-Informationen wie Klartextnamen oder die Zuordnung von Bytecodes zu den korrespondierenden Zeilennummern im Java-Quellcode. Dieser Debug-Blob wird vom Debug-Proxy geladen und anschließend verwendet, um die Anfragen des Debuggers zu beantworten.

Die Java-Seite inklusive Debug-Proxy wurden an der TU München in einer Studienarbeit bearbeitet, in deren Rahmen auch das ACDWP Protokoll mit spezifiziert wurde. Mit dem Ergebnis dieser Studienarbeit können in Zukunft Anwendungen, die auf der ACVM ausgeführt werden, direkt aus Eclipse heraus gesteuert und für das In-Circuit-Debugging betrachtet werden:

- Wolf-Dennis Pahl, „In-System-Debugging von Java-Programmen in der AmbiComp-Virtual-Machine“, Sudienarbeit, Uni Karlsruhe, Juli 2009

5.5 Eclipse-Plugin

Die Erweiterungen des Transcoders wurden wie schon zuvor von der HdM in das Eclipse-Plugin integriert. Außerdem wurden aufgrund der Umstellung der Kommunikation auf I2C inkl. der Einführung des Otto-Protokolls auf dem Rückwandbus der AICUs an der HdM erneut Änderungen im Emulator Manager und der Emulator View vorgenommen. Des Weiteren mussten die Sichten im Eclipse-Plugin daran angepasst werden, dass nun einzelne Sandwichmodule und nicht mehr nur ganze AICUs als kleinste Einheit ansprechbar sind.

Der Emulator Manager wurde so umstrukturiert, dass er es ermöglicht, emulierte AICUs zu konfigurieren, zu erstellen und dann zu starten. Dabei können die Linux-ACVMs so konfiguriert werden, dass ihre Fähigkeiten den unterschiedlichen Sandwichmodulfunktionalitäten (I/O-Sandwichmodul, Ethernet-Sandwichmodul, Bluetooth-Sandwichmodul) entsprechen. Die emulierten Sandwichmodule werden vom Linux-Emulator bereitgestellt, d.h. der Emulator muss auf einem Linux-Server laufen, zu dem sich das Eclipse-Plugin über eine Socket-Schnittstelle verbindet. Ein passendes Protokoll wurde von der HdM entworfen und auf beiden Seiten implementiert. So kann beispielsweise ein im Eclipse-Plugin erstellter Blob an den Emulator Manager übertragen werden, der dann wiederum über den ACVM-Runner eine

simulierte Linux-AICU mit dieser Blob-Datei startet (siehe auch Abbildung 9 und 15).

Für den Emulator Manager wurde ein eigenes Protokoll festgelegt, über das z.B. Emulator-ACVMs gestartet und gestoppt werden können. Die Einheiten AICU, Sandwichmodul und Linux-ACVM sind nur noch logische Einheiten und benötigen keine eigenen Backdoor-Kommandos mehr. Für jeden logischen Endpunkt wurde eine Handler-Klasse geschaffen, die auf einfache Weise über die Handler-Factory-Klasse instanziiert und dann für die Kommunikation mit dem Endpunkt genutzt werden kann. Alle Komponenten, die für die Kommunikation über diese Handler notwendig sind, wurden in eine eigene Bibliothek, die ACEPConnectionLibrary, ausgelagert und werden mit dem Eclipse-Plugin ausgeliefert.

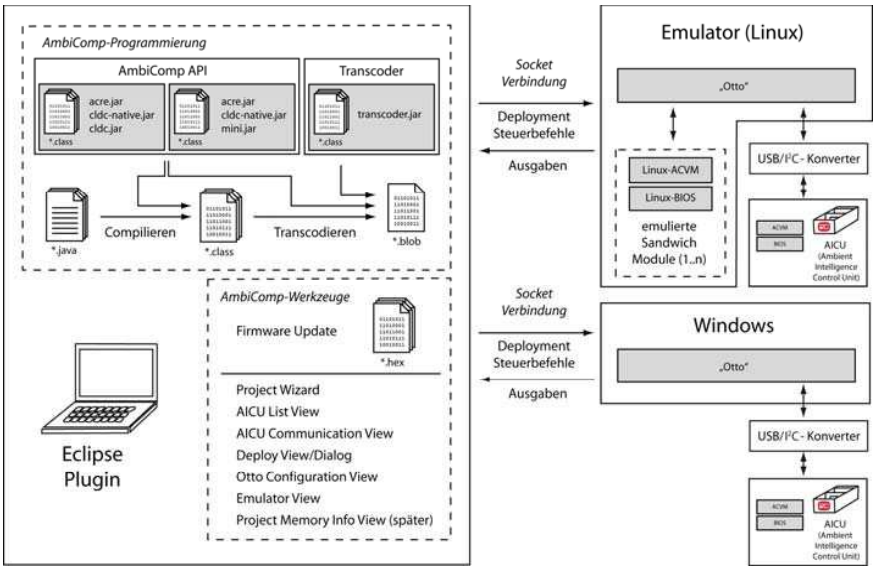


Abbildung 9: Übersicht Eclipse-Plugin Features

Da in der zweiten Hälfte des Jahres 2008 die neue Eclipse Version 3.4 herausgegeben wurde, musste das AmbiComp Eclipse-Plugin zusätzlich auf diese Version portiert werden, was wiederum einige Anpassungen im Quellcode notwendig machte.

Als Resultat dieser Arbeiten standen im Eclipse-Plugin zum Ende des Projektes die folgenden Views und Funktionalitäten zur Verfügung:

- **AmbiComp Project Wizard**: Mit dem AmbiComp Wizard kann der Entwickler in drei Schritten ein AmbiComp-Projekt anlegen (s. Abbildung 10). Bei der Erstellung werden je nach Auswahl die kleinere MiniAPI oder die größere, dafür komfortablere CLDC-API automatisch eingebunden. Außerdem wird die Version des Java-SDKs automatisch auf die zum Transcoder kompatible Version 1.4 gesetzt.

  Im letzten Schritt vor Anlegen des Projektes, wird der Entwickler noch gefragt, ob er in die AmbiComp-Perspektive wechseln will (sofern diese nicht bereits geöffnet ist). Bei Bestätigung werden dann alle notwendigen AmbiComp Views geöffnet und sinnvoll angeordnet. Auf die einzelnen Hilfsansichten (Views) wird weiter unten noch genauer eingegangen.

  Auch während der Programmierphase kann der Nutzer zwischen den beiden API-Varianten wechseln, um entweder die komfortablere CLDC-API nutzen zu können oder unter Verwendung der MiniAPI kleinere Dateigrößen zu erzielen. Diese

Umstellung kann über die Project Properties vorgenommen werden.

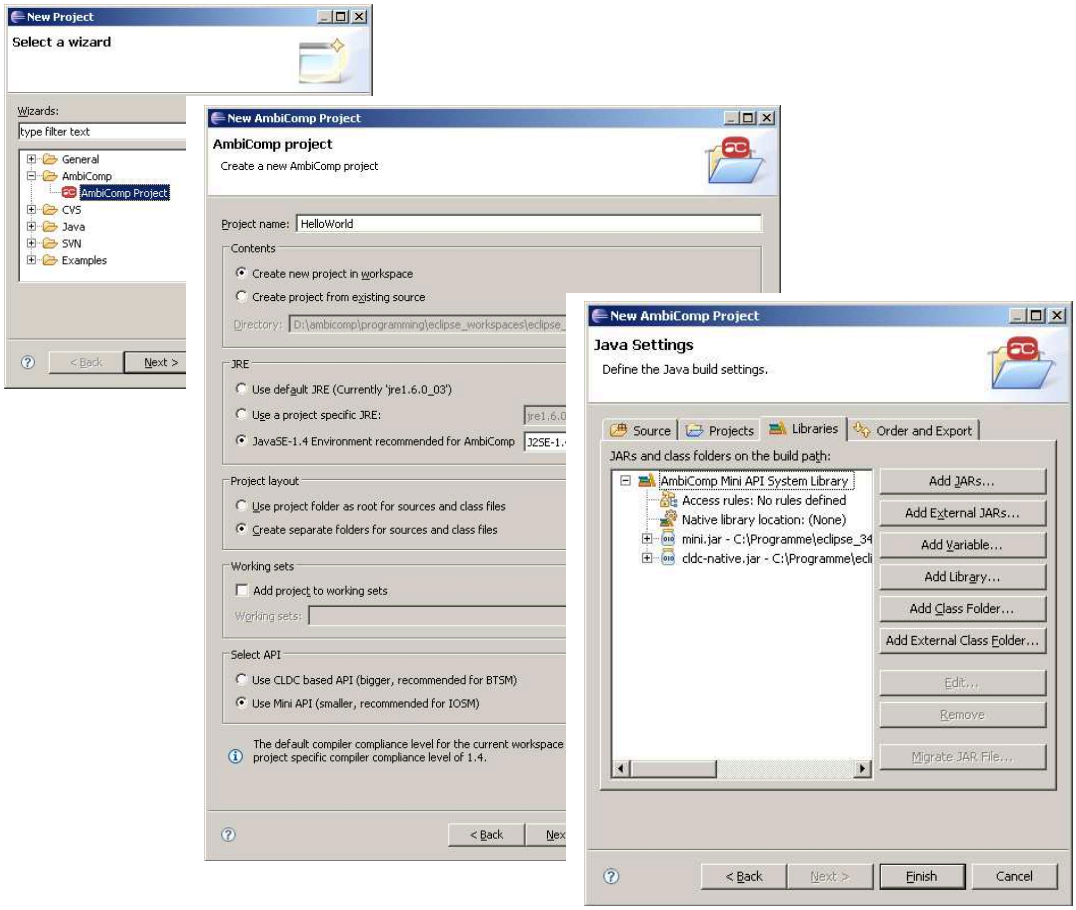


Abbildung 10: Anlegen eines AmbiComp-Projekts

Transcodiervorgang: Der Transcodiervorgang wurde so umgebaut, dass er dem Benutzer größtmögliche Freiheiten gibt. Nachdem das AmbiComp-Projekt angelegt wurde schreibt der Entwickler seinen Quellcode in Java (.java-Dateien). Dieser wird dann unter Verwendung der AmbiComp-spezifischen Bibliotheken (APIs) mit dem Eclipse-internen Compiler in .class-Dateien kompiliert. Anschliessend muss der Nutzer den Transcodiervorgang anstoßen. Um den Transcodiervorgang möglichst einfach und flexibel zu gestalten, erfolgt dies über ein Kontextmenü (siehe Abbildung 11). Wird der Projektordner ausgewählt, so wird das gesamte Projekt transcodiert. Werden einzelne Dateien ausgewählt, so wird dem Entwickler, je nach Anzahl der ausgewählten zu transcodierenden .java-Dateien, ermöglicht, eine einzige oder mehrere .blob-Dateien zu erstellen. Diese Flexibilität hat sich als sehr nützlich erwiesen, da so die unnötige Erstellung vieler einzelner Projekte vermieden werden kann.

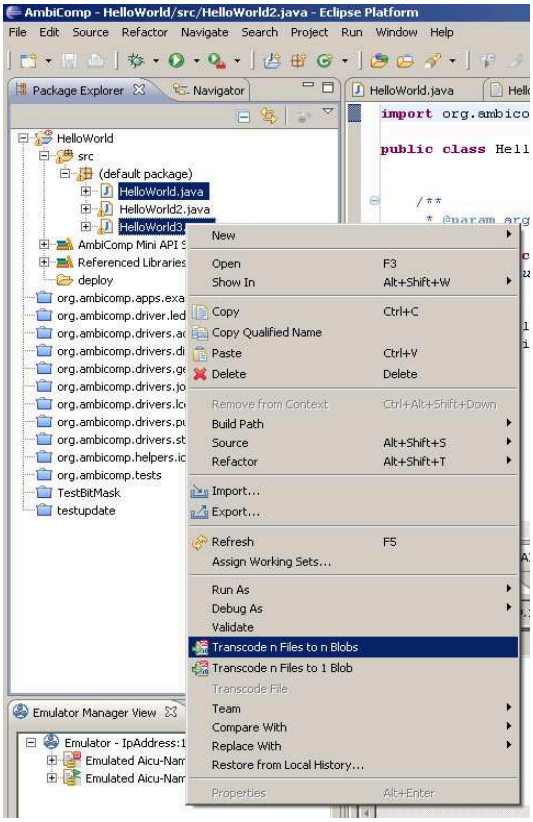


Abbildung 11: Kontextmenü Transcoding

- **Otto Configuration View**: Das Eclipse-Plugin kommuniziert mit den emulierten und realen AICUs über die „Otto“-Schnittstelle. Um die hierfür verwendete Socketschnittstellen komfortabel einrichten zu können, wurde eine Konfigurationsansicht implementiert. Hier können neue Konfigurationen angelegt und die Verbindung über IP und Port über einen Dialog eindeutig konfiguriert werden.
- **AICU List View**: Die AICU List View dient als zentrale Ansicht im Eclipse-Plugin und stellt alle an den Otto angeschlossenen AICUs, deren Sandwichmodule, sowie die einzelnen Endpunkte der Sandwichmodule dar. Dies sind zum einen reale AICUs, die über den I2C-USB-Konverter („Aardvark“ der Firma Totalphase) an den DBUS angeschlossen sind. Zum anderen alle Linux-ACVMs, die vom Emulator Manager gestartet wurden (Abbildung 12).

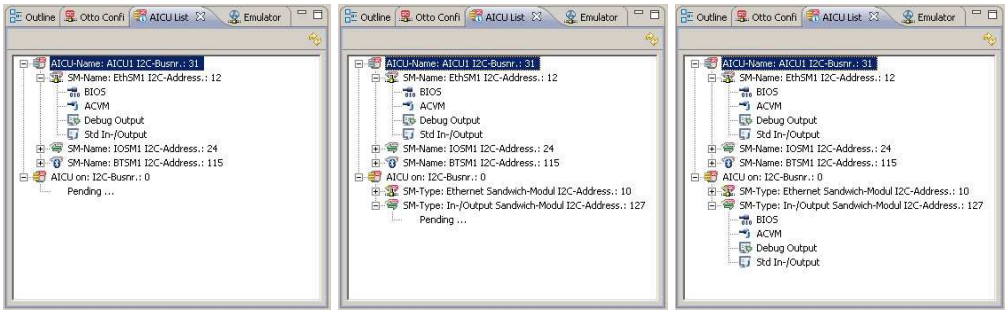


Abbildung 12: Initialisierung der Komponenten in der AICU List View

- **AICU Communication View**: Bei der Nutzung des Eclipse-Plugins zur Programmierung von Beispielanwendungen stellte sich heraus, dass in der Communication View oftmals dieselben Anzeigen bzw. Aktionen genutzt werden, diese aber nicht gleichzeitig zugänglich bzw. sichtbar waren. Daher wurde die AICU Communication View um einen neuen Reiter erweitert, der diese Anzeigen vereint und es dem Benutzer z.B. ermöglicht, im selben Reiter die ACVM und das BIOS zu steuern, die Ausgaben des Debug- und des Standard-Out-Endpunktes darzustellen, sowie die Kommunikation mit BIOS- und ACVM-Endpunkt zu verfolgen.

  Diese View löste die im Projektverlauf zunächst implementierte AICU Monitor View ab, in der noch der Kommunikationsfluss über die serielle Schnittstelle angezeigt wurde. Die AICU Communication View wurde gegenüber den ersten Versionen wesentlich erweitert, so dass nun manuell Kommandos abgesetzt werden können, beispielsweise beim BIOS oder dem ACVM-Endpunkt. Außerdem können Blob-Dateien sowie ACVMs direkt auf ein bestimmtes Sandwichmodul übertragen werden.

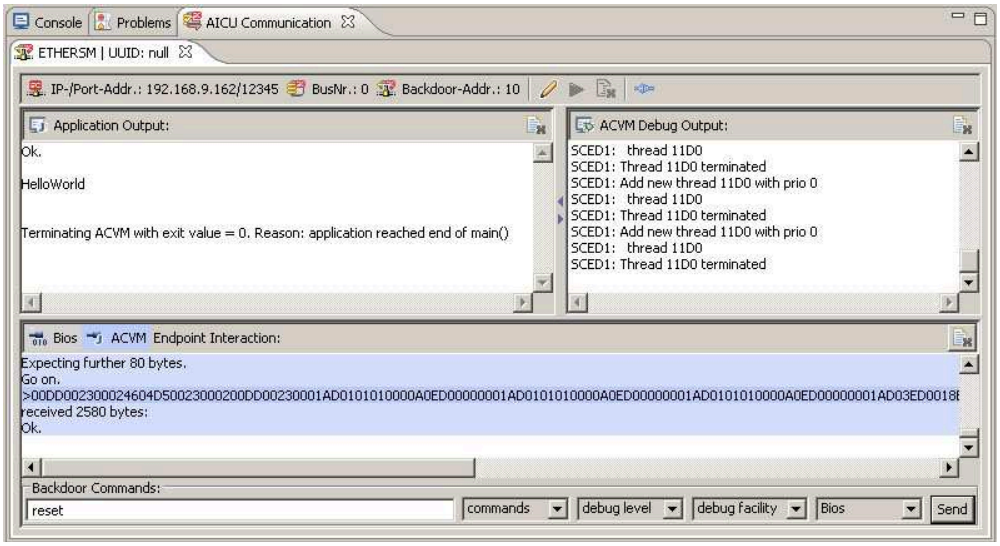


Abbildung 13: AICU Communication View

Wird ein einzelner Endpunkt ausgewählt, so öffnet sich ein Reiter, in dem nur die Kommunikation mit diesem Endpunkt darstellt wird. Wird ein Sandwichmodul ausgewählt, so öffnet sich ein Reiter, in dem er möglich ist, die ACVM/das BIOS zu steuern (siehe Abbildung 13, unterer Bereich), die Ausgaben des Debug- und des Standard-Out-Endpunktes darzustellen (siehe Abbildung 13, „Application Output und „ACVM Debug Output“), sowie die Kommunikation mit BIOS-/ACVM-Endpunkt zu verfolgen (siehe Abbildung 13, mittlerer Bereich „Endpoint Interaction“). Die Ausgaben der Interaktion mit den Endpunkten werden entsprechend Endpunkt und Art der Ausgabe farblich codiert.

- **Deploy Dialog/Deploy View**: Eine AmbiComp-Anwendung muss auf ein Endgerät (real oder emuliert) übertragen werden, damit sie dort ausgeführt werden kann. Hierzu kann der Entwickler in dieser View auswählen, auf welches Sandwichmodul das Blob-File übertragen und ob es dort persistent im Flash oder flüchtig im RAM gespeichert werden soll. Ein Blob-File kann somit problemlos in einem Arbeitsschritt auf mehrere Endgeräte übertragen werden.
- **Firmware Update**: Auch die Firmware kann über einen Dialog in einem Arbeitsschritt auf ein oder mehrere Sandwichmodule übertragen werden. Bei der Auswahl dieser Option öffnet sich ein Dialog, der im Wesentlichen gleich gestaltet ist wie der Deploy

Dialog für Blob-Files.

- **AmbiComp Preferences**: Für die bessere Nachvollziehbarkeit von Fehlern kann der Nutzer in den AmbiComp Preferences die Versionen der momentan eingebundenen Werkzeuge und Bibliotheken ablesen. Außerdem wurde, basierend auf dem frei verfügbaren Log4J-Framework, ein ErrorLogger implementiert, der Tracing- und Fehlermeldungen über die AmbiComp Console ausgibt und gleichzeitig eine Log-Datei schreibt. Diese Log-Datei ist speziell für das Debugging des Eclipse-Plugins notwendig und sinnvoll.
- **AmbiComp Perspective**: Da im Projektverlauf immer mehr AmbiComp-spezifische Views entwickelt wurden, wurden diese in einer gemeinsamen AmbiComp-Perspektive zusammengefasst. Wählt der Entwickler diese Perspektive aus, beispielsweise statt der Perspektive für reine Java-Programmierung oder für Debugging-Aufgaben, so werden alle für die AmbiComp Entwicklung notwendigen und sinnvollen Views dargestellt.

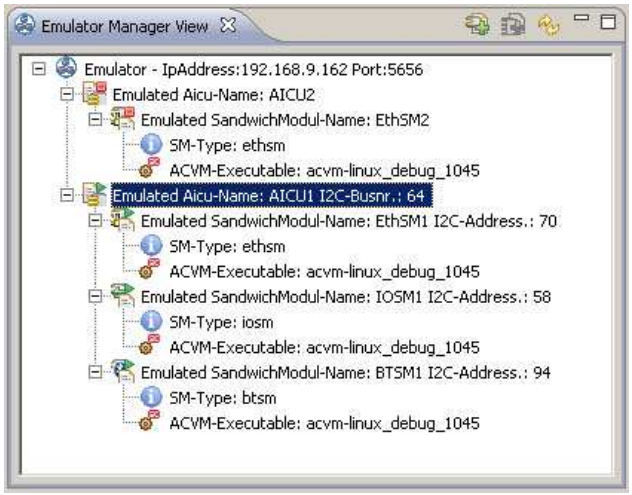


Abbildung 14: Emulator Manager View

- **Emulator Manager View und Emulator Connections Configuration View**: Der Emulator Manager kommuniziert über eine Socketverbindung mit dem Eclipse-Plugin. Mit Hilfe des entworfenen Protokolls kann der Emulator über eine grafische Oberfläche gesteuert werden: Zunächst legt der Benutzer die Konfigurationen von AICUs und Sandwichmodulen an. Bezeichnung, Typ und verwendete ACVM werden in der Emulator Manager View (Abbildung 14) entsprechend angezeigt. Die Linux-ACVMs werden dem Benutzer gegenüber logisch gruppiert, um den Eindruck einer AICU analog zu einer realen Einheit zu schaffen. Nachdem die Konfigurationen angelegt sind, kann eine AICU gestartet und gestoppt werden. Wenn eine Linux-ACVM nicht mehr läuft, wird dies automatisch vom Emulator Manager erkannt und das Eclipse-Plugin entsprechend benachrichtigt. Diese Änderung wird automatisch in der Emulator Manager View angezeigt. Die Konfiguration der Verbindung erfolgt gleich wie die Konfiguration der Verbindung zur Otto-Komponente (s.o.) über IP und Port in der so genannten Emulator Connections Configuration View.

  Durch die Umsetzung diverser Konzepte (z.B. Drag&Drop, adaptives Kontextmenü, Logische Gruppierung in Views, etc.) wurde versucht, das Plugin einfach nutzbar zu halten. Insbesondere Wert gelegt wurde auch auf die einheitliche Gestaltung der Icons bzgl. Farbe und AmbiComp-Logo, so dass der Nutzer oft schon anhand der Symbole Informationen dargeboten bekommt (so unterscheiden sich beispielsweise reale und

emulierte Sandwichmodule durch die Farbe, wohingegen IO-Module von Ethernet-Modulen über ein anderes Zusatzsymbol unterschieden werden können).

Zu Dokumentationszwecken wurden von der HdM diverse Grafiken erstellt. Die Visualisierung der teil recht komplexen Zusammenhänge diente als gute Grundlage für die Diskussionen mit den Projektpartnern. Die Grafiken konnten zu dem auch zur Außendarstellung eingesetzt werden, z. B. in den diversen Datenblättern, und wurden ebenfalls in diesem Dokument verwendet.

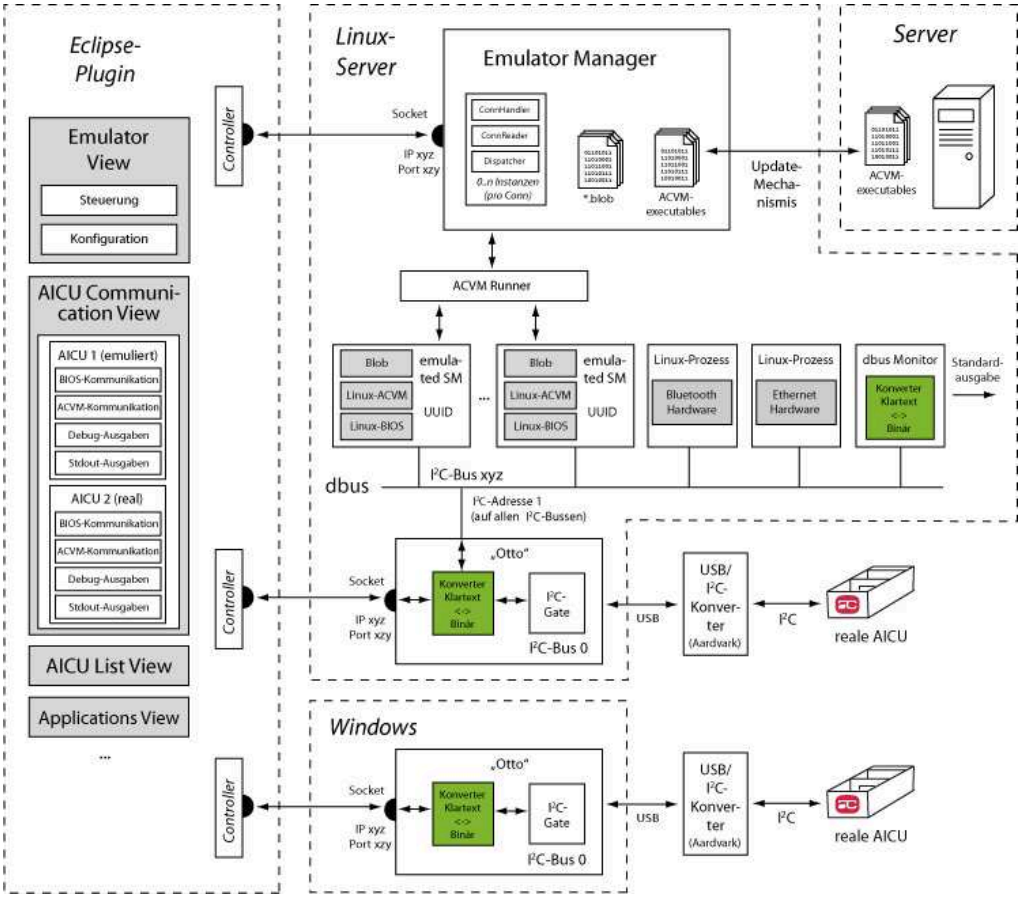


Abbildung 15: Übersicht Kommunikation und Schnittstellen

Zusätzlich wurden von der HdM einige Tests geschrieben, die die Funktionalität der Otto-Komponenten überprüften. Hierzu wurden alle Kommunikationskomponenten in die so genannte „ACEP Connection Library“ (AmbiComp Eclipse-Plugin Connection Library) ausgelagert. Diese Bibliothek wurde von der TU München bereitgestellt und wurde dort auch für die automatisierten Tests genutzt.

Um dem Benutzer den Fortschritt des Deploy- und Transcodierungsvorgang darzustellen, wurden Prozessbalken eingeführt. Damit beim Deployvorgang die grafische Oberfläche während des Übertragungsvorganges weiterhin auf Eingaben reagiert, wurden Deploy- und Transcodierungsvorgang in separate Threads ausgelagert und das Job-Framework von Eclipse genutzt.

5.6 Framework für graphische Oberflächen und Beispielanwendung

Darüber hinaus wurde an der HdM ein Framework entwickelt, mit dessen Hilfe es möglich ist, auf einfache Weise grafische Oberflächen für AmbiComp-Anwendungen zu entwickeln. Die AmbiComp APIs enthalten keine entsprechenden Bibliotheken, die dies unterstützen würden. Außerdem sind Speicherplatz und Prozessorgeschwindigkeit auf den Sandwichmodulen nicht ausreichend, um die Darstellung von Anwendungen auf jeder AICU selbst bewältigen zu können. Aus diesen Gründen musste ein Konzept gefunden werden, das eine Trennung der Model-, View- und Controller-Komponenten (MVC-Komponenten) erlaubt und zugleich auf vielen unterschiedlichen mobilen Geräten wie Laptop, PDA, oder Smartphone, ohne großen Installationsaufwand eingesetzt werden kann.

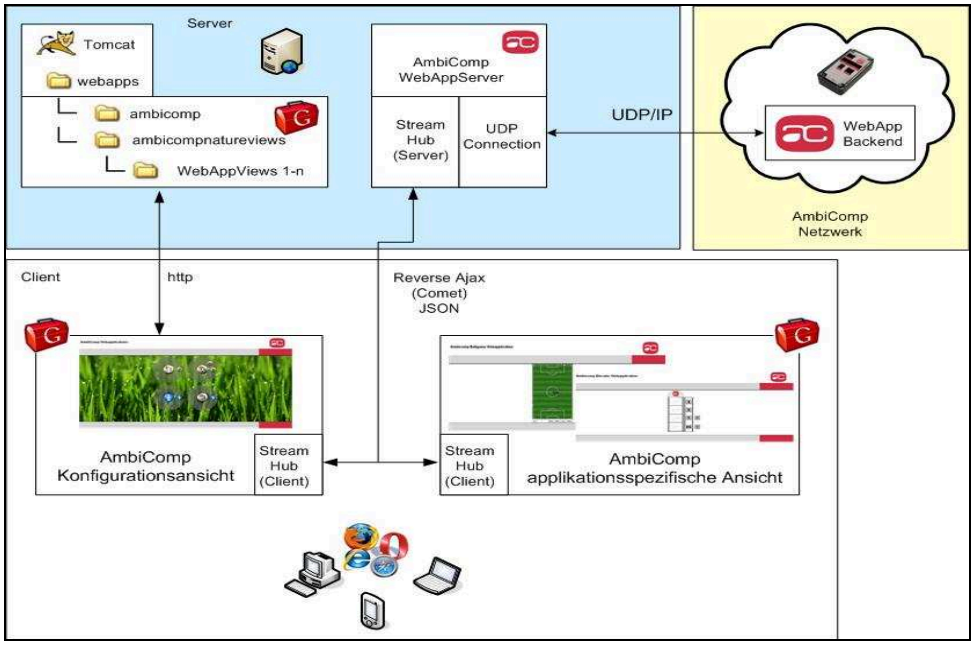


Abbildung 16: AmbiComp Webarchitektur

Das entwickelte Framework regelt dabei die Kommunikation der MVC-Komponenten über Events. Diese Events werden innerhalb eines AICU-Netzes über TCP/IP übertragen und münden in ein Gateway. Dieses wiederum überträgt die Events zum darstellenden Endgerät über HTTP. Da eine AICU zu wenige Ressourcen bietet, speichert das Gateway zusätzlich die HTML-Dateien und verarbeitet die HTTP-Anfragen und dient somit als zentrale Server-Komponente. Für die Darstellung auf dem Endgerät (View) wurde HTML ausgewählt, da diese Technologie auf fast allen Smartphones, PDAs und Laptops mit Hilfe eines Browsers genutzt werden kann.

Dieses Framework wurde an der HdM prototypisch umgesetzt und auf dem Projekttreffen am 13.07.2009 an Hand einer weiter unten beschriebenen Beispielanwendung präsentiert. Die Einsatzzwecke für visuelle Darstellung erstrecken sich über die Systemadministration (Fehlerbehandlung, Monitoring, Konfiguration) bis hin zur expliziten Steuerung von Aktorik, sowie die Aufbereitung von Sensordaten.

Diese Arbeiten wurden an der HdM teilweise im Rahmen einer Diplomarbeit unterstützt:

- Christian Schiepe, „Konzeption und Umsetzung eines Java-basierten Frameworks zur Entwicklung von visuellen Interaktionsmöglichkeiten von AmbiComp Anwendungen“, Diplomarbeit, Hochschule der Medien, Februar 2009

Für die Echzeitunterstützung beim Auslesen von Sensordaten wurde die *Reverse-Ajax-*

Technologie eingesetzt und mit verschiedenen Umsetzungen (Direct Web Remoting, StreamHub) getestet. Der Vorteil bei dieser Art der Anbindung des Browsers an den Server ist, dass der Server bei Bedarf anfallende Sensordaten direkt an eine Webinstanz übertragen kann, ohne dass diese Browserinstanz durch z.B. erneutes laden der Seite eingreifen muss. Für das Framework zum Abschluss des Projektes wurde *StreamHub* als Reverse-Ajax-Technologie für die Umsetzung herangezogen. Der Grund hierfür ist, dass sich diese Implementierung sehr gut in das Webframework des *Google Webtoolkit* (GWT) integrieren lässt.

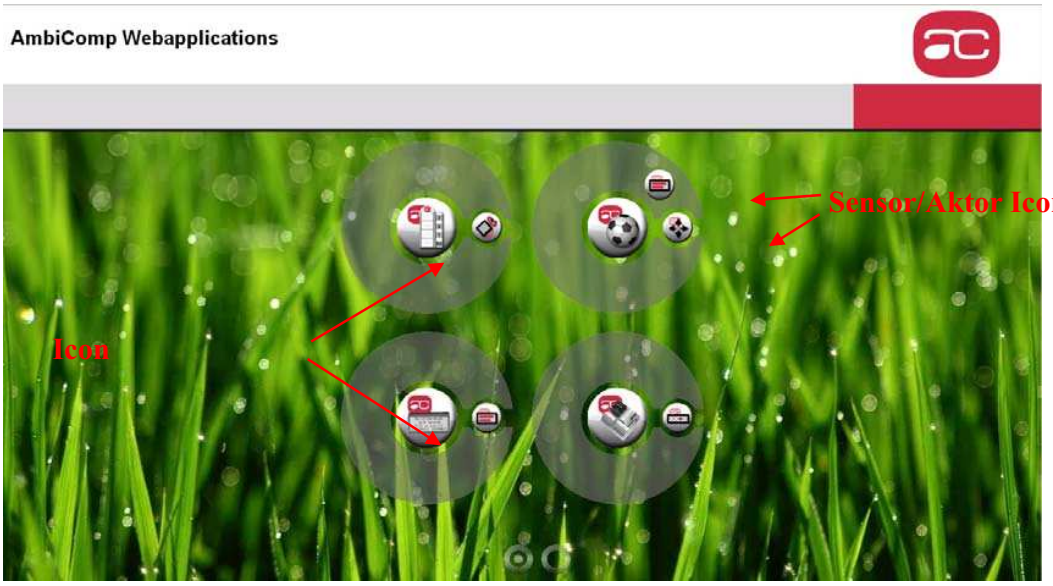


Abbildung 17: AmbiComp Konfigurationsansicht

Die GWT Komponente wiederum dient innerhalb des AmbiComp Frameworks als IDE welche den AmbiComp Entwickler die fehlende IDE Unterstützung bei der Erstellung von Frontend Webseiten unterstützt. Durch den Einsatz der beiden neuen Komponenten, GWT und StreamHub, ergibt sich die in Abbildung 16 dargestellte Architektur.

Der Server besteht dabei aus den beiden Komponenten Tomcat und AmbiComp WebAppServer. Die Tomcat Komponente ist für die Auslieferung der Webseiten an den Client zuständig. Die AmbiComp applikationsspezifischen Webanwendungen werden im Verzeichnis „ambicompnatureviews“ und die initiale „AmbiComp Konfigurationsansicht“-Webanwendung wird im Verzeichnis „ambicomp“ hinterlegt.

Die Komponente „AmbiComp WebAppServer“ ist das Bindeglied für die Kommunikation zwischen Client und „WebAppBackend“ zuständig. Die Kommunikation zum Client wird über Reverse Ajax (StreamHub) und zum „WebAppBackend“ über UDP/IP hergestellt. Die „AmbiComp WebAppServer“ Komponente verwaltet die angemeldeten Webanwendungs-instanzen und „WebAppBackend“ Instanzen, und ist somit in der Lage die aufkommenden Daten zu entsprechenden Enpunktinstanzen weiterzuleiten.

Dem Endanwender bieten sich zwei Arten von AmbiComp Webanwendungen. Zum einen die „AmbiComp Konfigurationsansicht-Webanwendung und zum anderen die „AmbiComp applikationsspezifischen Ansicht“-Webanwendungen.

Die „AmbiComp Konfigurationsansicht“-Webanwendung ist die zentrale Anlaufstelle. Hier werden alle vorhandenen AmbiComp applikationsspezifischen Webanwendungen angezeigt (s. Abbildung 17). Jeder Kreis repräsentiert eine „AmbiComp applikationsspezifischen

Ansicht“-Webanwendungen. Das große runde Icon ist stellvertretend für die Anwendung und die kleinen runden Icons in der Umlaufbahn repräsentieren die jeweiligen Aktoren oder Sensoren (Dienste) die für diese Anwendung benötigt werden.

Die WebAppBackend Komponente ist für die Bereitstellung der Dienste der lokalen Anwendung auf einer AICU zuständig. Die Anwendung kann z.B. den direkten Steuerungs- und Lesezugriff auf Aktorik und Sensorik bereitstellen wie es auch in den von der HDM entwickelten Demonstrationsapplikationen der Fall ist. Diese Komponente meldet sich beim Starten der AICU über UDP/IP am AmbiCompWebAppServer an und offeriert die unterstützten Dienste. Dies ist notwendig damit eine Zuordnung zu den jeweiligen Webanwendungen stattfinden kann.

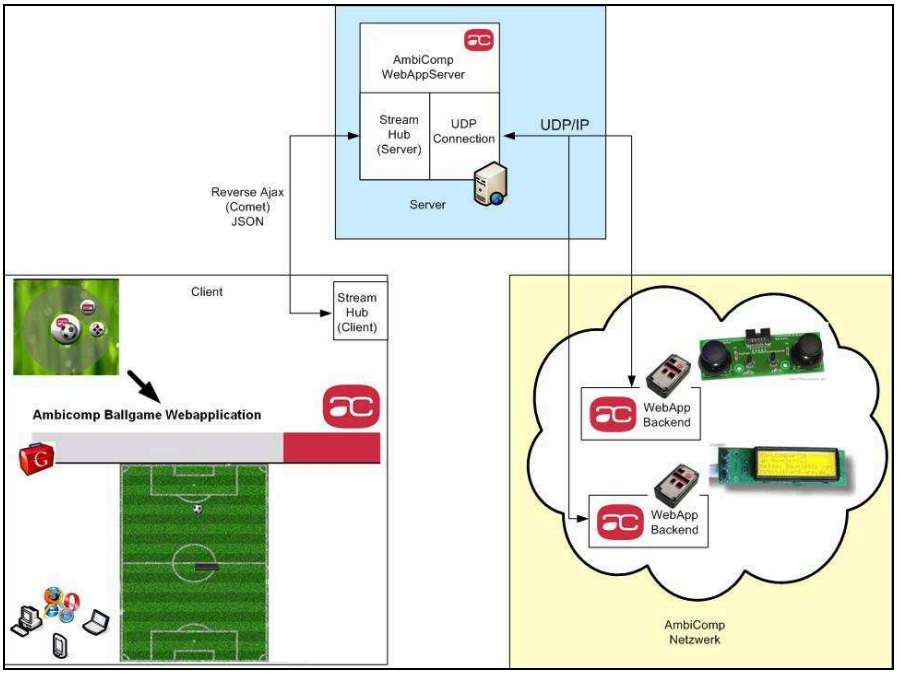


Abbildung 18: Aufbau Reverse-Ajax Beispielanwendung

Mit Hilfe des Frameworks kann der Nutzer über einen Browser auf Anwendungen zugreifen, die im AmbiComp-Netzwerk vorhanden sind. Wird eine AICU per Ethernet an das Netzwerk angeschlossen, so registriert sie sich an einem Gateway. So kann sie beispielsweise eine Motor-Anwendung anmelden.

Wird die AICU vom Netz getrennt, so erkennt das Gateway dies und die Änderung kann durch Einsatz der Reverse-Ajax-Technologie direkt angezeigt werden. Durch den Einsatz konventioneller Browser-Technologie ist die Interaktion mit AmbiComp-Anwendungen über das Framework nun auch auf mobilen Endgeräten möglich. Über den Browser kann z.B. der an ein I/O-Sandwichmodul angeschlossener Motor gesteuert werden.

Um in einem Beispiel zu zeigen, welche Möglichkeiten sich durch den Einsatz von Reverse-Ajax ergeben können, wurde als Prototyp die Beispielanwendung „Ballgame“ implementiert und wie erwähnt erfolgreich demonstriert. Der Aufbau dieses Szenarios bestand, wie in Abbildung 18 zu sehen, aus einem LCD-Display und einem Joypad die jeweils am IOSM einer AICU angeschlossen sind.

Die Aufgabe im Spiel war es, den stetig bewegten Ball mit der Spielerfigur, einem Balken, so abzulenken, dass dieser ins Tor geht. Die Anzeige des Spielstandes erfolgte über das LC-Display an der einen AICU und die Steuerung der Spielfigur über das an der anderen AICU

angeschlossene Joypad.

Die Reverse-Ajax Technologie kam hier bei der Steuerung der Spielerfigur, also des Balkens, zum Tragen. Hier wurden die abgetasteten Positionen des Joypads von der AICU per UDP/IP über den Server, welcher per Reverse-Ajax mit dem Client in Kontakt steht, an diesen übertragen. Der zeitliche Versatz mit dem die Pakete, welche Positionsinformationen beinhalten, am Client eintreffen war so gering, dass der Anwender fast keinen Unterschied zu einer herkömmlichen Joypadsteuerung wahrnahm.

5.7 BIOS Version 1.2 und Version 1.3

Neben der Weiterentwicklung und Erweiterung des Sandwichmodulportfolios bei der Beecon wurde auch das BIOS weiterentwickelt. So entstanden im Laufe des Jahres 2009 die BIOS Versionen 1.2 und 1.3. Während die Version 1.2 vor allem kleinere Verbesserungen an der BIOS Schnittstelle brachte, beispielsweise bei den Timern, führt die Version 1.3 eine wesentliche Änderung beim Speicherzugriff ein: Durch die so genannten *Fast Memory Access* (FMA) Funktionen können nun 32Bit-Worte auf beliebigen Sandwichmodulen innerhalb einer AICU gelesen und geschrieben werden. Zugriffe auf andere Sandwichmodule werden mittels synchroner I2C-Nachrichten für die Anwendung transparent über den Rückwandbus abgewickelt. Dadurch, dass der Speicherzugriff nun über das BIOS erfolgt, können ACVM und native Anwendungen einheitlich über den Speicher kommunizieren, sich beispielsweise Referenzen auf die im Speicher abgelegten Objekte übergeben. Dazu wurden in mehreren Diskussionen mit dem Fraunhofer IESE und der TU München verschiedene Anforderungen an ein allgemeines *Object Distribution Model* (ODM) erarbeitet.

Zusätzlich zu den verschiedenen Speicherarten bildet das BIOS nun auch den Zugriff auf die internen Zustände des Sandwichmoduls mittels FMA ab. So können Informationen über das jeweilige Sandwichmodul ausgelesen, die Kommunikationsschnittstellen (Ethernet, Bluetooth, etc.) konfiguriert und die I/O-Leitungen angesteuert werden. Dieses neue Konzept vereinfacht und verschlankt die BIOS-Schnittstelle deutlich und schafft gleichzeitig viele Möglichkeiten zur späteren Erweiterung des Sandwichmodulportfolios.

Die Weiterentwicklung des Hardware-BIOS machte es notwendig, auch das Linux-Emulation-BIOS für den Emulator in den Versionen 1.2 und 1.3 nachzuziehen. Diese Arbeiten wurden erneut an der TU München durchgeführt. Das Ergebnis wurde in einer Studienarbeit in Form einer Beschreibung der gesamten Linux-Emulationsumgebung im Allgemeinen und dem Linux-BIOS im Speziellen für die am Projektende vorliegende Linux-Toolchain mit BIOS 1.3 beschrieben:

- Thomas Kittel, „Design und Implementierung einer Hardwareabstraktionsschicht für die AmbiComp Virtual Machine und Entwicklung eines Frameworks zur Simulation von kompletten AICUs“, Software Entwicklungsprojekt, TU München, Februar 2010

5.8 CeBIT 2009

Der Stand der Arbeiten im Projekt wurde der interessierten Öffentlichkeit bei der CeBIT im März 2009 demonstriert. Neben den Demonstratoren, die in Folgenden noch genauer beschrieben werden, wurde zusätzlich Informationsmaterial wie die Datenblätter zu den verschiedenen Sandwichmodulen, eine AmbiComp CD und einige Faltblätter und Flyer zum Mitnehmen zur Verfügung gestellt (siehe unten, Kapitel 5.12).

Abbildung 18: AmbiComp auf der CeBIT

Auf der CeBIT gab es ca. 30 - 40 ernstzunehmende Kontakte mit Unternehmen und Institutionen, die sich eine Nutzung der AmbiComp-Technologie vorstellen konnten. In der Nachbearbeitung der Messe wurde die Liste dieser Interessenten auf ca. 10 Unternehmen reduziert. Diese potentiellen Partner wurden dann noch einmal genauer analysiert bzw. angesprochen. Trotz dieser intensiven Nachbearbeitung ergaben sich aber aus diesen Kontakten keine weiteren Zusammenarbeiten mit Dritten.

Abbildung 19: Gespräche der AmbiComp-Mitarbeiter am CeBIT-Stand

Im Vorfeld der CeBIT wurden durch den Einsatz der gesamten Werkzeugkette (Toolchain) noch einige Fehler der darunter liegenden Schichten, sowie im Eclipse-Plugin identifiziert und behoben, was mehr Zeit in Anspruch nahm als erwartet. Auch die Hardware von Drittanbietern musste zunächst evaluiert, beschafft und dann in die Demonstratoren integriert werden. Auf der CeBIT konnten dann einige Beispielanwendungen gezeigt werden, die auf den AICUs der zweiten Hardware Generation und dem BIOS 1.2 basierten:

1) Zur Demonstration der Feuermelder wurde ein Modellgebäude, der *AmbiComp Tower* gebaut. Je ein Feuermelder darin wurden an das IOSM je eines AICUs angeschlossen. Die AICUs waren im Turm verteilt und über ein Ethernet-Sandwichmodul kabelgebunden

mit einem PC verknüpft. Auf dem PC lief die Visualisierung der Zustände der verfügbaren Feuermelder. Die einzige funktionale Komponente konnten zu Gruppen zusammengeschaltet und über die GUI konfiguriert werden konnte. Zusätzlich gab es noch einen Feueralarm, der durch Aufnahme in eine Gruppe im Feuerfall mit aktiviert wurde. Einzelne Räume des AmbiComp Towers konnten auf Knopfdruck mit Rauch aus einer Nebelmaschine geflutet werden, wodurch der Alarm bei den Feuermeldern der zugehörigen Gruppe ausgelöst wurde.

2) Der Fahrstuhl im *AmbiComp Tower* wurde über einen Schrittmotor bewegt, der an ein IOSM angeschlossen war. Der Motor bewegte den Fahrstuhl auf Knopfdruck in die verschiedenen Etagen. Der Schrittmotor konnte durch Taster, sowie durch ein Joypad gesteuert werden.

3) An eine AICU konnten ein Joypad oder ein Abstandssensor angeschlossen werden und mit diesen die LEDs einer LED-Leiste oder ein LC-Display angesteuert werden.

Abbildung 20: Der AmbiComp Tower

5.9 Konzeptionelle Arbeiten

Am Fraunhofer IESE wurde anknüpfend an den vorhergehenden Arbeiten am Komponentenmodell an der Frage gearbeitet, wie nichtfunktionale Eigenschaften von Komponenten im Rahmen einer Dienstbeschreibung spezifiziert und dann zur Laufzeit im Sinne eines Dienstvertrages ausgehandelt werden können. Die Aushandlung wurde dabei durch die, gemäß einem entsprechenden Realisierungspattern über alle partizipierenden Komponenten verteilten, Kontrolleinheiten durchgeführt und betraf sowohl funktionale als auch nicht-funktionale Eigenschaften. Das Ziel der Aushandlung war Dienstverträge zu etablieren, die als „fair“ bezeichnet werden können. Diese Fairness ist jedoch, gerade im Hinblick auf nicht-funktionale Eigenschaften, nicht allgemein zu definieren.

Verschiedene Ideen zur Aushandlung von Dienstverträgen und Fairness wurden im Rahmen eines Papers beschrieben:

[SINTER09] C. Peper und D. Schneider: „On Runtime Service Quality Models in Adaptive Ad-hoc Systems“, SINTER '09: Proceedings of the 2009 ESEC/FSE workshop on Software integration and evolution @ runtime, Amsterdam, September 2009

Im Kern dieser Fragestellung lag das Problem der Abbildung *benötigter* und *zur Verfügung*

gestellter Diensteigenschaften. Aufbauend auf den vorangegangenen konzeptionellen Vorarbeiten zur Erstellung einer Komponenten-Architektur für Ad hoc Systeme wurden zentrale Teile dieses Ansatzes prototypisch realisiert. Im Rahmen mehrerer Projektarbeiten wurden dazu die grundlegenden Mechanismen der Kommunikation und des Service-Managements als Simulationsumgebung in Java implementiert und später auf die reale Hardware übertragen.

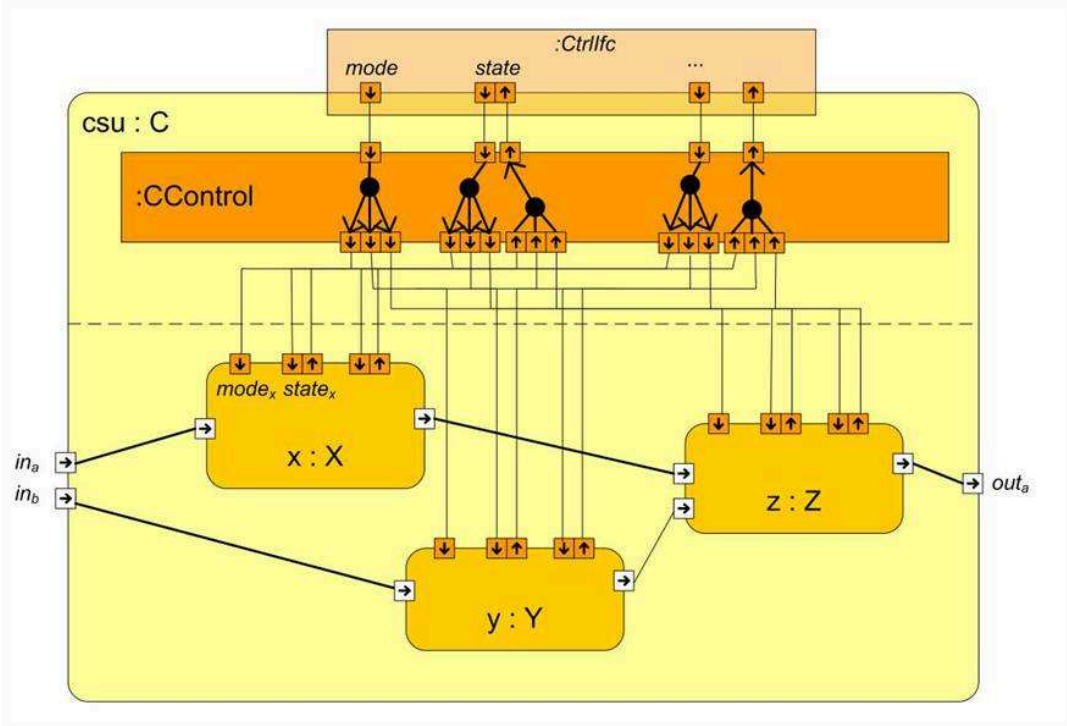


Abbildung 21: Realisierungspattern

An der Uni Karlruhe bzw. TU München wurden einige weitere forschungsnahe Arbeiten in enger Zusammenarbeit mit Studenten prototypisch umgesetzt. Zum einen wurden verschiedene Erweiterungen des SSR Protokolls implementiert. Hierzu gehörten vor allem die in einer Diplomarbeit ausgearbeiteten Erweiterungen bzgl. asymmetrischer Verbindungen. Derartige Verbindungen werden im Allgemeinen nicht betrachtet, obwohl sie bei Funkverbindungen sehr häufig auftreten:

- Pascal Birnstill, „Efficient Detection and Utilization of Asymmetric Links in Scalable Source Routing (SSR)“, Diplomarbeit, TU München, März 2009

Eine weitere Arbeit bereitete den bestehenden SSR Code für die Verwendung in AmbiComp vor:

- Fabian Knittel, „Scalable Source Routing in the AmbiComp Environment“, Studienarbeit, Uni Karlsruhe, April 2009

Außerdem geben die folgende zwei Veröffentlichungen einem Gesamtüberblick über das AmbiComp Projekt im Allgemeinen und der AmbiComp Virtuelle Machine im speziellen:

[KuVS08] *Johannes Eickhold, Björn Saballus und Thomas Fuhrmann. „A Distributed Virtual Machine for Ambient Computing”, Poster beim 7. GI/ITG KuVS Fachgespräch „Drahtlose Sensornetze“, 25./26. September 2008*

[AMI08] *Johannes Eickhold, Thomas Fuhrmann, Björn Saballus, Sven Schlender,*

und Thomas Suchy. „AmbiComp: A platform for distributed execution of Java programs on embedded systems by offering a single system image", Proceedings of the European Conference on Ambient Intelligence (AmI-Blocks '08 Workshop), Nürnberg, Germany, November 19, 2008

Bei der TU München waren darüber hinaus auch die Weiterentwicklungen bei den Globally Accessible Objects (GAOs) und beim Scalable Source Routing (SSR) Gegenstand zweier Fachveröffentlichungen:

[HPDC09] Björn Saballus und Thomas Fuhrmann „Maintaining Reference Graphs of Globally Accessible Objects in Fully Decentralized Distributed Systems", In Proceedings of the International ACM Symposium on High Performance Distributed Computing (HPDC 2009), Munich, Germany, June 2009

Sven Schlender von der Beecon hat zusätzlich einen AmbiComp-Vortrag auf dem Entwickerforum in Ludwigsburg gehalten:

[DEEF09] Sven Schlender, Thomas Suchy, Björn Saballus, Johannes Eickhold und Thomas Fuhrmann. AmbiComp: Eine Plattform zur einfachen Programmierung von verteilten eingebetteten Systemen. Tagungsband Design & Elektronik Entwicklerforum Embedded-System-Entwicklung, Ludwigsburg, 7-8 Oktober 2009. WEKA FACHMEDIEN GmbH, 2009

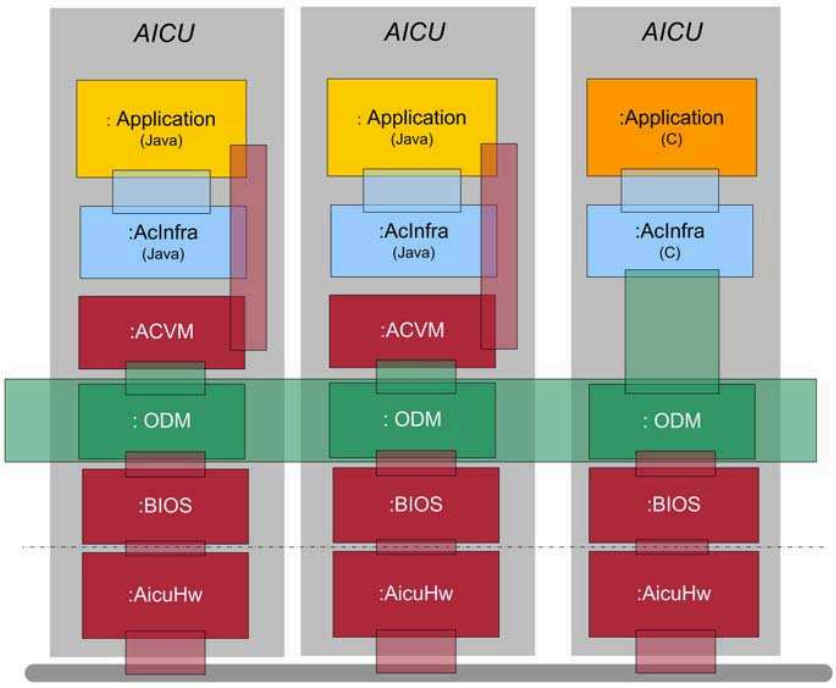


Abbildung 22: ACVM und native Anwendung mit ODM

Im Laufe des Jahres 2009 verstärkte sich die Erkenntnis, dass es wünschenswert wäre, hybride Systeme unterstützen zu können, d.h. Systeme aus mehreren interoperablen AICUs, deren Implementierung sowohl in Java (ACVM-basiert) oder nativ in der Programmiersprache C erfolgen kann. Dies hat den Vorteil, eine potentiell größere Anwenderschicht zu erreichen,

da auch „klassische“ C-Entwickler von den AmbiComp-Ansätzen zur Verteilung von Anwendungen direkt profitieren könnten (siehe Abbildung 22). Zu diesem Zweck wurde auf Grundlage verteilter Objekte eine erste gemeinsame Abstraktion entwickelt.

Gemeinsam mit der TU München, dem Fraunhofer IESE und der Beecon wurde hierfür ein neues Modell für den verteilten Objektzugriff diskutiert, das schon oben erwähnte *Object Distribution Model* (ODM). Es umfasst Ideen bekannter Ansätze wie OSGi und CAN Open, ist aber speziell auf die Anforderungen von AmbiComp zugeschnitten. Die Projektpartner sind überzeugt, mit dieser Diskussion und der entsprechenden Implementierung der Konzepte einen Mehrwert zu schaffen, der die von OSGi und anderen Systemen offen gelassenen Lücken schließen kann.

ODM steht dabei einmal für das *Object Distribution Model,* welches wie bereits erwähnt ein Modell für den verteilten Objektzugriff bereitstellt. Es steht gleichzeitig auch für eine konkrete Implementierung, die sogenannte *Object Distribution Management* (ODM) Schicht. Diese leichtgewichtige ODM-Schicht sollte am Fraunhofer IESE zunächst möglichst pragmatisch realisiert werden - insbesondere auch, um das bislang nur simulierte Komponentenmodell auf reale AmbiComp-Hardware bringen zu können (siehe Abbildung 23). Dies ist zwar im Projektverlauf nicht mehr vollständig gelungen, es konnten jedoch im Rahmen der Entwicklungstätigkeiten die Schnittstellen der ODM-Schicht zur Anwendung definiert und implementiert werden.

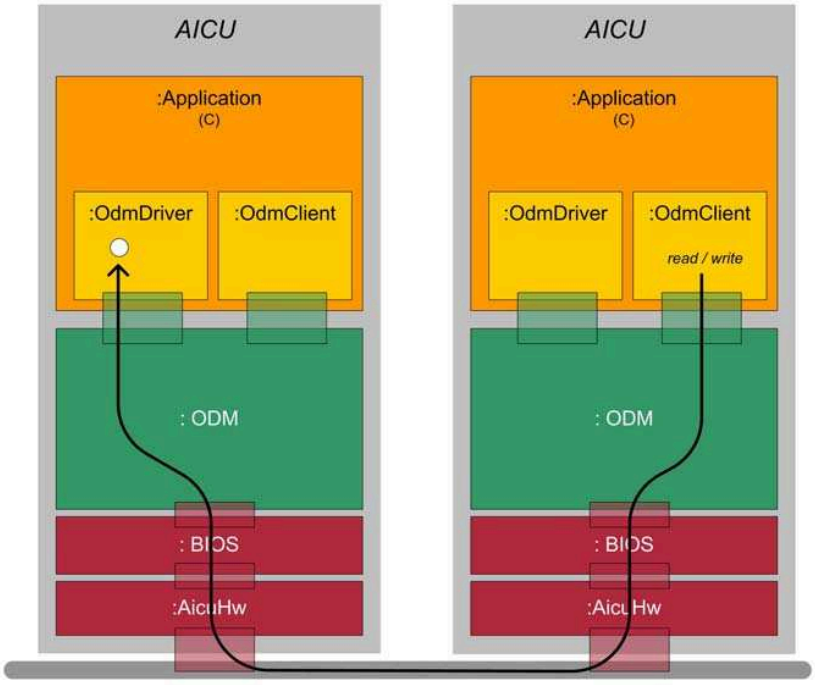


Abbildung 23: ODM Zugriff zwischen zwei AICUs

Über die ODM Schicht angebotene Objekte müssen mit einem einfachen Typ-Deskriptor versehen sein, der es erlaubt, die übertragenen Rohdaten eines Objekts auch ohne Vorwissen zu interpretieren.

Bei der TU München wurde zur Projektlaufzeit begonnen, eine formale Spezifikation des Object Distribution Models zu erstellen. Diese Arbeiten werden derzeit im „J-Cell“ und im „Sovereign“-Projekt weiter verfolgt.

5.10 AmbiComp-Praktikum

Neben der CeBIT 2009 fand eine weitere Demonstration der AmbiComp-Technologie im Rahmen des mehrwöchigen Praktikums „Eingebettete Systeme“ im Sommersemester 2009 an der Universität Karlsruhe statt. Dabei wurden Hardware, Software und Dokumentation von Studierenden einem Belastungstest unterzogen. So konnten die gesamten AmbiComp-Ergebnisse unter realistischen Bedingungen evaluiert werden: die AICU-Hardware, die Software rund um das Eclipse-Plugin, inklusive dem Emulator, sowie der Dokumentation und dem Tutorial.

Der Aufbau des Praktikums war so gestaltet, dass die Studenten langsam an die AmbiComp-Technologie herangeführt wurden. Hierzu wurden die Teilnehmer in Gruppen aufgeteilt, die jeweils vorgegebene Aufgaben lösen mussten.

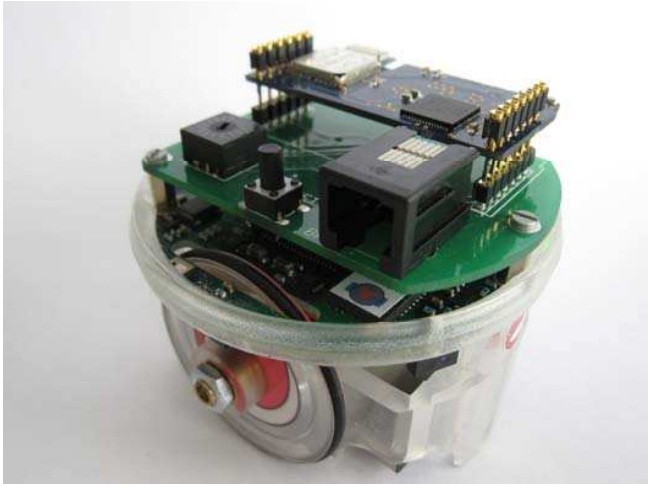


Abbildung 24: ePuck Roboter, mit AmbiComp-Hardware erweitert

Zu Anfang des Praktikums lernten die Teilnehmer den Umgang mit dem Eclipse-Plugin und der gesamten Toolchain. Die erste Aufgabe war es, ein „Hello World“ Programm mit dem Eclipse-Plugin zu erstellen, auf die Hardware zu übertragen und die Ausgaben im Eclipse-Plugin anzuzeigen.

In der zweiten Aufgabe musste ein bestehender, in Java geschriebener Treiber für ein LCD Display an vorgegebenen Stellen erweitert werden. Diese Aufgabe vermittelte den Umgang mit der API Dokumentation und der GPIO Funktionalität des IOSMs.

Bei der nächsten Aufgabe lernten die Studenten die Inter-SM Kommunikation per expliziter Backplanekommunikation zwischen zwei SMs innerhalb eines AICUs kennen. Von einem Joypad, welches an einem IOSM angeschlossen war, sollten die Werte des Joypads am IOSM ausgelesen werden. Diese Werte mussten dann über die Backplane an das BTSM des AICUs geschickt werden, wo sie von einem Programm empfangen und verarbeitet wurden. Das Resultat dieser Verarbeitung wurde über die Backplane zurück an das IOSM geschickt, und dort von einem LC-Display ausgegeben.

Anschließend an diese Aufgabe sollte als nächstes eine Bluetooth-Kommunikations-verbindung zwischen zwei BTSMs in zwei verschiedenen AICUs aufgebaut werden. Mittels dieser Verbindung sollten die Steuersignal eines Joypads, welches an der einen AICU angeschlossen war, von diesem AICU per Bluetooth an den anderen AICU geschickt werden. Bei diesem zweiten AICU sollten ankommenden Signale auf einem angeschlossenen LC-Display angezeigt werden.

Als Erweiterung der Bluetooth-Aufgabe wurden die ePuck Roboter (siehe Abbildung 24 und 26) eingesetzt. Bei diesen handelt es sich um kleine, fahrbare Roboter, welche um eine „AmbiComp-Huckepackplatine“ erweitert wurden, auf welche AmbiComp-SMs aufgesteckt werden konnten. Die Firmware auf dem ePuck wurde so modifiziert, dass auf diesen ein rudimentäres BIOS lief, welches per I2C Bus (Rückwandbus) mit dem aufgesteckten SM kommunizieren konnte. So war es möglich, mit AmbiComp-Technologie den ePuck zu steuern. Die Aufgabe war es, einen ePuck von einem AICU mit angeschlossenem Joypad aus per Bluetooth zu steuern.

Nachdem alle Gruppen diese vorhergehenden Aufgaben gelöst hatten, entwickelte jede Gruppe ein eigenes Projekt mit den bisher kennengelernten Technologien, welches sie implementieren und am Ende des Praktikums erfolgreich den anderen Studenten vorstellen mussten.

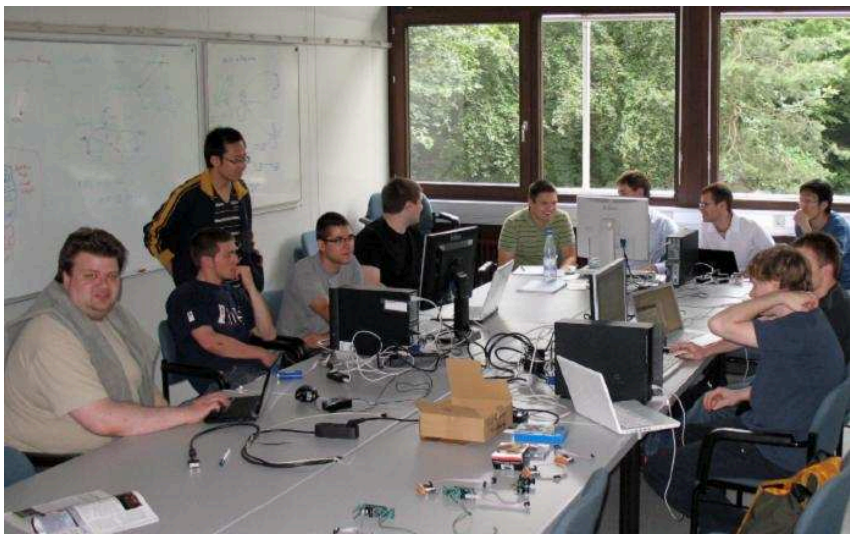


Abbildung 25: AmbiComp Praktikum an der Uni Karlsruhe

Dieses Praktikum wurde an der Universität Karlsruhe unter anderem im Rahmen einer Studienarbeit entwickelt:

- Martin Melullis, „Distributed Embedded Systems in Java - Design of a Lab Course“, Studienarbeit, Uni Karlsruhe, Dezember 2009

Für das Projekt erfüllte das Praktikum die folgenden wichtigen Funktionen:

- Praxistest aller einzelnen Komponenten, insbesondere der Sandwichmodule und der Toolchain, inkl. Eclipse-Plugin
- Test des reibungslosen Zusammenspiels aller Hardwarekomponenten, d.h. Sandwichmodule und externe Peripherie
- Einsicht in die Praxistauglichkeit der Projektergebnisse

Der realitätsnahe Praxistest durch die am Praktikum teilnehmenden Studenten unterstützte dabei die Schritt für Schritt voranschreitende Vervollständigung des AmbiComp-Systems.

5.11 Tag der offenen Tür

Die AmbiComp-Technologie wurde am 24.10.2009 beim Tag der offenen Tür an der TU München der interessierten Öffentlichkeit vorgestellt. Hierzu wurden die schon im Praktikum

verwendeten und um AmbiComp-Technologie erweiterten ePucks (Abbildung 24 und 26) und einige AICUs verwendet. Der demonstrierte Aufbau und die verwendete Software basierten dabei auf einer der Aufgaben aus dem AmbiComp-Praktikum: Die Besucher des Standes konnten über ein Joypad, welches an eine AICU angeschlossen war, die ePucks steuern. Damit dies möglich war, hat das BTSM in dem AICU zuerst selbsttätig eine Bluetooth-Verbindung zum ePuck aufgebaut. Anschließend wurden die Signale des Joypads vom IOSM über den Rückwandbus weiter an das BTSM gereicht, von wo aus die Daten dann über Bluetooth an den ePuck geschickt wurden.

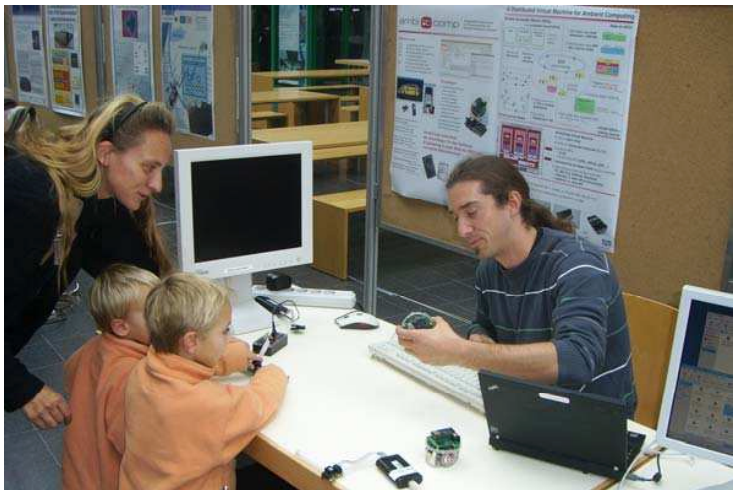


Abbildung 26: AmbiComp beim Tag der offenen Tür der TU München

Neben dieser praktischen Demonstration wurden auch einige Plakate ausgestellt, welche zum einen die AmbiComp-Vision darstellten und zum anderen einige wissenschaftliche Erkenntnisse, welche in diesem Projekt entwickelt wurden, zeigten.

Insgesamt stieß der AmbiComp Stand auf reges Interesse, sowohl bei interessierten Laien, als auch bei Studenten, welche sich über AmbiComp und die entwickelte, eingebettete Hardware informieren wollten.

5.12 Darstellung in der Öffentlichkeit

Neben den wissenschaftlichen Veröffentlichungen und den studentischen Arbeiten gab es einige Vorträge vor interessiertem Publikum.

Beim VDE/ITG Workshop „Neue Herausforderungen in der Netzsicherheit“ in Essen wurde AmbiComp als Teil eins Vortrages mit dem Titel „Identitätsmanagement als Vorraussetzung benutzerfreundlicher Kommunikationsanwendungen“ vorgestellt.

Beim Tag der Technologie am Fraunhofer IESE wurde im Jahr 2007 das AmbiComp-Projekt mit einem eigenen Stand, Postern und einigen Sandwichmodulen präsentiert.

Bei dem bereits erwähnte ESTIA Workshop in Stuttgart im Jahr 2008 wurde eine AmbiComp-Demonstration gezeigt und ein Vortrag mit dem Titel „AmbiComp – Ambient Computing for Sensor and Home Networks“ gehalten. Diesen Vortrag haben sich ca. 41 Teilnehmer aus 24 Organisationen, unter anderem Siemens, Thales, Bosch-Siemens-Hausgeräte und Gorenje (Hausgeräte), angehört.

Zusätzlich wurde 2008 bei der Summer School der TU Darmstadt auf Burg Ebernburg ein kurzer Vortrag zu AmbiComp gehalten und ein AmbiComp Poster sowie Hardware präsentiert.

Auf dem ECOMODIS Workshop „Efficient component-based Development of Dependable

Computing Systems“ im Jahr 2009 in Mannheim wurde AmbiComp als Teil eines Vortrages präsentiert. Bei diesem Vortrag waren ca. 40 Teilnehmer auf Forschung und Industrie anwesend, unter anderem ABB und Honda.

Während der Veranstaltung „10 Jahre Medieninformatik an der HdM Stuttgart“ im Jahr 2008 würde das AmbiComp-Projekt in einer Präsentation vorgestellt.

Auf dem „Eclipse Embedded Day 2009“ in Stuttgart wurde ein Vortrag mit dem Titel „Distributed Embedded Systems with AmbiComp“ gehalten. Die Folien sind im Netz verfügbar unter:

http://wiki.eclipse.org/images/4/40/AmbiComp.pdf

Des Weiteren erschien bei der HdM im ersten Halbjahr 2009 ein ausführlicher Artikel über das AmbiComp-Projekt auf der Homepage der Hochschule der Medien. Das Interview wurde mit Prof. Dr. Johannes Maucher, sowie den beiden Mitarbeitern Jürgen Butz und Thomas Suchy geführt. Es erschien am 19.05.2009 und konnte zum Zeitpunkt zu dem dieser Abschlussbericht geschrieben wurde, über folgenden Link aufgerufen werden:

http://www.hdm-stuttgart.de/view_news?ident=news20090511125037

5.13 Dokumentation, Lehrmaterial und Tutorials

Alle Entwicklungen im Projekt wurden 2009 vermehrt dokumentiert und im Projekt-Wiki den Projektpartnern zugänglich gemacht. Mit dem Begleitmaterial zum Praktikum stand damit die Grundlage für das AmbiComp-Tutorial zur Verfügung.

Am Fraunhofer IESE wurde für Lehrmaterial und Tutorials ein allgemeines Konzept entwickelt, an welchem sich die weiteren Arbeiten orientierten:

- **Info/Motivation**: Dieser Teil des Lehr- und Demomaterials hat den Zweck, das Projekt und dessen Ziele vorzustellen. In diesem Zusammenhang werden insbesondere die wesentlichen Ergebnisse und deren Nutzen für den Anwender dargestellt. Durch diesen Teil soll das Interesse geweckt und potentielle Kunden von der AmbiComp Vision überzeugt werden. Als Plattform sollen die Projekt-Homepage, Messepräsentationen und Handouts (z.B. CDs und Datenblätter) dienen.
- **Mini-Tutorial**: Dieser Teil hat das Ziel, den Einstieg in die AmbiComp Programmierung zu erleichtern. Dazu sollen die grundlegenden Bestandteile des AmbiComp Frameworks anhand eines einfachen Beispiels vermittelt werden. Die entsprechenden Beschreibungen und Anleitungen sollen dabei direkt in das Eclipse Hilfe-System integriert werden. Daraus resultierende Vorteile sind unter anderem die Verbreitung des Lehrmaterials als Bestandteil des Eclipse Plugins und die Möglichkeit der „Verlinkung“ des Beispiels mit Toolfunktionen. Neben der Integration in Eclipse können einige Bestandteile, wie zum Beispiel anleitende Filme, auch über die Website zugänglich gemacht werden. Hierzu wurde ein Mini-Tutotial in Eclipse integriert das als Anleitung für einen Entwickler bei der Erstellung einer ersten AmbiComp Applikation dienen kann. Besonders hilfreich ist dabei die Möglichkeit die im Tutorial beschriebenen Schritte teilautomatisiert durchzuführen. Der Nutzer kann zum Beispiel durch einen einzelnen Klick innerhalb des Tutorials die beschriebene Aktion automatisch ausführen lassen.
- U**mfassendes Fallbeispiel**: Dieser Teil schlägt die Brücke zwischen der „abstrakten“ Beschreibung der Vision und der Konzepte und dem konkreten Minibeispiel. Hier wird umfassend die Verwendung des AmbiComp Frameworks zur Erstellung eines „AmbiComp-Produkts“ beschrieben. Verfügbar gemacht werden soll dieser Teil als Online-Tutorial über die Projekt-Homepage oder CDs und gegebenenfalls auch als gedruckte Version.

Gemäß dieser dreigliedrigen Konzeption wurden vielfältige Materialien erstellt, welche im Folgenden kurz beschrieben werden.

Was die erste Kategorie „Information und Motivation“ angeht wurden am Fraunhofer IESE, insbesondere im Vorfeld der CeBIT, verschiedene Materialien erstellt. Hierzu gehören unter anderem Flyer mit einem groben Gesamtüberblick zum Projekt, Faltblätter mit Details zu den verschiedenen Hauptbestandteilen des Projektes, ein Präsentationsfilm mit Audiokommentar und eine AmbiComp CD mit ausführlichen Informationsmaterialien, wie zum Beispiel den Datenblättern der Alcatel-Lucent. Auch auf der Webseite wurden einige dieser Materialien untergebracht. So ist dort unter anderem ein Flash Film mit Audio zur Vorstellung des Projektes zu finden.

Gegen Projektende wurden vermehrt „Mini-Tutorials“, also Informationsmaterialien der zweiten Kategorie, erstellt. Themen waren unter Anderem das Aufsetzen einer AmbiComp Entwicklungsumgebung, das Erstellen einer ersten Applikation und die Nutzung grundlegender Funktionen der AmbiComp API. Plattform dieser Mini-Tutorials ist einerseits das Web, wo ein Wiki als technische Plattform ausgewählt wurde. Die meisten dieser Mini-Tutorials werden durch kurze „Screencasts“ (Flashfilme, die zeigen was in der beschriebenen Situation auf dem Monitor des PCs vor sich geht) ergänzt. Gleichsam wurden viele der Mini-Tutorials von der HdM auch direkt ins Eclipse Plugin integriert. Dies birgt den Vorteil, dass auch Offline gearbeitet werden kann und trotzdem die wesentlichen Informationen zur Einarbeitung in AmbiComp in einer Quelle vorliegen.

Zur dritten Kategorie, dem umfassenden Fallbeispiel, sollten die detaillierten Informationen des internen AmbiComp Wikis verwendet und in einer aufbereiteten Form in das externe Wiki transferiert werden. Insgesamt änderte sich die Zielsetzung dieses Konzeptionsteils etwas – weg vom konkreten Fallbeispiel und hin zu einer umfassenden, tiefgreifenden aber allgemeinen Beschreibung der AmbiComp Bestandteile. Erste Schritte dazu wurden gegen Ende des Projektes durchgeführt. Neben Informationen aus dem internen Wiki wurden auch die Datenblätter der Alcatel-Lucent in das Wiki überführt.

Um die Tutorials des Fraunhofer IESE in das Eclipse-Plugin zu integrieren, wurde das in Eclipse integriertes Hilfe-Framework verwendet. Dieses wird vom Eclipse Plugin genutzt, um eine Einführung in AmbiComp zu geben. Sie enthält allgemeine Informationen über AmbiComp, eine Einführung in die Nutzung des AmbiComp Eclipse-Plugins, einen Bereich, in dem die ersten Schritte erklärt werden, sowie einen Bereich mit einigen Beispielanwendungen mit dazugehörigem Quellcode. Die Hilfe wird als eigenes Plugin im AmbiComp-Paket mit dem Eclipse-Plugin ausgeliefert und bei Änderungen gemeinsam mit dem Eclipse-Plugin aktualisiert. Die Hilfe kann über das Menü in Eclipse angezeigt werden (Help -> Help Contents).

Ein weiteres hilfreiches Konzept in Eclipse sind die sogenannten Cheat Sheets. Hier wird der Nutzer interaktiv durch die ersten Schritte geleitet, wie z.B. das Anlegen eines AmbiComp-Projektes. Die HdM erstellte mehrere Cheat Sheets, um den Einstieg in die AmbiComp-Programmierung zu erleichtern.

Auch die Cheat Sheets werden, wie das Plugin, über den Eclipse-internen Update-Mechanismus ausgeliefert und automatisch aktualisiert. Die Cheat Sheets für AmbiComp werden in Eclipse angezeigt über das Menü (Help -> Cheat Sheets -> AmbiComp Development).

Bei der Alcatel-Lucent Deutschland AG wurden im Laufe des Jahres 2009 die bestehenden Datenblätter zum Teil überarbeitet und weiter Datenblätter erstellt. Insgesamt existieren damit am Ende des Projektes 14 Datenblätter in deutsch und englisch für die folgenden Komponenten des AmbiComp-Systems:

AmbiComp (Überblick) – 4 Seiten:

Allgemeine Beschreibung des Projektes, besonders des AmbiComp-Systems mit seinen verschiedenen Sandwichmodulen, den verschiedenen

Kommunikationsmöglichkeiten, den Entwicklungswerkzeugen sowie einigen Einsatzbeispielen.

AmbiComp Eclipse-Plugin– 3 Seiten:

Beschreibung der AmbiComp-Software-Entwicklungs-Plattform, insbesondere der verschiedenen AmbiComp-Sichten im Java-Werkzeug Eclipse.

Sandwich Modul – Allgemeine Informationen – 3 Seiten:

Dieses Datenblatt beschreibt die für alle Sandwich-Module gültigen Eigenschaften wie z.B. die mechanischen Abmessungen. Hierbei handelt es sich um Hardware-Module, die entsprechende Funktionen bereitstellen. Viele Sandwich-Module tragen einen Mikrocontroller, der mit einem BIOS ausgestattet ist und in den die notwendige Software geladen werden kann, in der Regel eine ACVM und entsprechende Applikationen. . Damit beschreibt dieses Datenblatt die Basis eines AmbiComp-Systems.

Bluetooth Sandwich Modul (BTSM, SPP & HCI) – 3 Seiten
Bluetooth Class 1 Sandwich-Modul – 3 Seiten:

Das Bluetooth Sandwich-Modul und das Bluetooth Class 1 Sandwich-Modul sind Komponenten des AmbiComp Projektes, die eine Bluetooth-Schnittstelle bereitstellen. Das BTSM SPP stellt eine standardkonforme serielle Schnittstelle über Bluetooth zur Verfügung. Das BTSM HCI stellt eine Schnittstelle zum HCI Interface bereit. Diese SMs ermöglichen eine Nahbereichskommunikation mit anderen Modulen und Geräten (Mobiltelefone, PDAs usw.). Das BTSM HCI ist in einer zweiten Ausführung, dem BTSM Class 1, mit einer höheren Sendeleistung verfügbar, welches für eine größere Reichweite geeignet ist.

Ethernet Sandwich Modul (EtherSM) – 3 Seiten:

Das Ethernet-Sandwichmodul stellt eine Ethernet-Schnittstelle bereit, durch die sowohl eine Kommunikation mit anderen Sandwichmodulen und Geräten von Drittanbietern möglich ist, als auch die Einbindung des AmbiComp-Systems in ein existierendes Netzwerk.

Eingabe-/Ausgabe Sandwich Modul (IOSM) – 3 Seiten:

Das Eingabe-/Ausgabe-Sandwichmodul ist eine Komponente des AmbiComp Projektes, die eine Reihe allgemein nutzbarer, programmierbarer Analog- und Digital-Schnittstellen nach dem Roboternetzstandard bereitstellt.

Peripheral Sandwich Modul (PeriSM) – 3 Seiten:

Das Peripheral-Sandwichmodul enthält eine Reihe unterstützender Funktionen, beispielsweise LEDs, Buzzer, Beschleunigungssensor, etc. Es dient in erster Linie zur raschen Demonstration von AmbiComp, beispielsweise im Rahmen von Praktika und Lehrveranstaltungen.

Backplane Primary Supply Sandwich Modul (BPPRISM) – 3 Seiten:

Das Backplane Primary Supply Sandwichmodul stellt die Stromversorgung für die anderen Sandwichmodule innerhalb eines AmbiComp-Systems bereit. Als Quelle kann dabei ein Netzteil mit Gleichstromausgang oder eine Batterie dienen. Zusätzlich stellt das Modul den AICU-Bus für Debugging-Zwecke auf einem RJ12-Steckverbinder zur Verfügung.

Backplane Secondary Supply Sandwich Modul (BPSECSM) – 2 Seiten:

Das Backplane Secondary Supply Sandwich-Modul bietet einem Sandwichstapel eine alternative Stromversorgung über ein Steckernetzteil. Anders als das BPPRISM arbeitet es nur zusammen mit einem primären Modul, beispielsweise einem Power-over-Ethernet-Sandwichmodul.

Power-over-Ethernet Sandwich Modul (PoESM) – 3 Seiten:

Das Power-over-Ethernet Sandwich-Modul erlaubt im Zusammenhang mit dem Ethernet Sandwichmodul die Stromversorgung eines AmbiComp-Systems über ein Ethernet-Kabel. Dabei sorgt das Ethernetmodul für eine Trennung zwischen den Ethernet-Daten und der Stromversorgung und stellt die Rohspannung von 48 V dem Power-over-Ethernet Sandwich-Modul zur Verfügung, das daraus die notwendigen Versorgungsspannungen von 5 V und 3,3 V erzeugt.

Accumulator Sandwich Modul (AccuSM) – 3 Seiten:

Das Akku Sandwichmodule versorgt einen Sandwichmodulstapel netzunabhängig mit Strom. Als Quelle dient ein auf dem Modul fest eingebauter Lithium-Polymer-Akkumulator. Zur Ladung des Akkumulators wird eine externe Spannung von 5 V benötigt. Ist neben dem AccuSM eine weitere Stromversorgung im System vorhanden, z.B. ein BPRISM, dann trennt eine Vorrangschaltung die Akkumulatorversorgung vom System. Alle Funktionen des AccuSM werden von einem Mikrocontroller gesteuert und überwacht.

HW-Entwicklungswerkzeuge (DevKit) – 3 Seiten:

Für Entwicklung und Fehlersuche stehen zwei passive Leiterplatten bereit.

Das so genannte *DevBoard* kann bis zu vier Sandwichmodule aufnehmen. Es sorgt für die notwendigen Verbindungen untereinander und erlaubt den Zugriff auf den AICU-Bus über zwei RJ12-Buchsen.

Das *AICU-Interface* ist ein einfacher Adapter für den AICU-Rückwandbus der zwischen dem im AmbiComp-System verwendeten RJ12-Steckverbinder und den 10-poligen Wannensteckern vermittelt, wie sie z.B. im I2C-USB-Adapter (Aardvark, Beagle) verwendet werden. Zusätzlich trägt das Modul einen Reset-Taster, mit dem das System neu gestartet werden kann.

AICU-Access Modul (AICUaccess) – 3 Seiten:

Das AICU-Access-Modul ist ein Schnittstellenwandler, der den AICU-Bus des AmbiComp-Systems auf eine serielle Schnittstelle umsetzt. Die Verbindung zum AmbiComp-System erfolgt über den dort definierten RJ12-Steckverbinder im TTL-Pegel, die serielle Schnittstelle steht an einem 9-poligen D-Sub-Steckverbinder zur Verfügung und erlaubt damit den direkten Anschluss eines PCs.

Das AICU-Access-Modul ist damit die kompakte Alternative zu der während des Projekts hauptsächlich verwendeten Kombination aus dem passiven AICU-Interface und einem externen I2C-USB-Adapter wie Aardvark oder Beagle.

6 Zusammenfassung des Projektverlaufs

Am Projektende stehen zwar die wichtigsten Komponenten des AmbiComp-Systems für die Entwicklung von Prototypen eingebetteter Systeme zur Verfügung; die ursprüngliche Planung, dieses System direkt in der Praxis zu erproben und eine Vielzahl von Beispielanwendungen auszuarbeiten, konnte aber nicht umgesetzt werden. Im letzten Projektjahr 2009 konzentrieren sich die Arbeiten daher auf den sauberen Abschluss des Projekts, insbesondere auf die Fertigstellung der zentralen AmbiComp-Entwicklungswerkzeuge, die Dokumentation der einzelnen Komponenten, sowie die Erstellung von Tutorials. Im Vergleich zur ursprünglichen Planung fehlt in dieser Aufzählung der AmbiComp-Technologien noch die transparente AICU-übergreifende Kommunikation. Diese wird aber derzeit in zwei Folgeprojekten, „J-Cell“ und „Sovereign“, weiter entwickelt und umgesetzt. In einem dritten Folgeprojekt, „SODA“ wird die AICU-Firmware hinsichtlich ihrer Echtzeitfähigkeit weiter ausgebaut.

Wir denken, dass durch diese Fokussierung auf eine funktionierende ACVM inkl. Toolchain dem ursprünglichen Projektziel am besten entsprochen wird: AmbiComp stellt ein

komponentenbasiertes System zur Verfügung mit dem auf einfache Weise Prototypen eingebetteter Systeme schnell miteinander vernetzt werden können, um Ambient Intelligence Systeme zu entwickeln.

Hierzu steht bei der Beecon eine breite Basis von verschiedenen Sandwichmodulen zur Verfügung, die in Kleinserie gefertigt wurden und zum Verkauf bereit liegen. Mit der ACVM wurde eine schlanke, einsatzfähige Laufzeitumgebung für diese eingebetteten Systeme geschaffen, deren Praxistauglichkeit mit Hilfe der Compliance Tests und in verschiedensten Beispielanwendungen gezeigt wurde. Der in das Eclipse-Plugin integrierte Transcoder kann in Java geschriebene Anwendung in das von der ACVM ausführbare Zwischenformat transcodieren und stellt zusätzlich die benötigten Informationen für das AmbiComp Debugger Framework zur Verfügung. Mit diesem ist es möglich, Java-Anwendungen nicht nur an einem Linux-PC zu debuggen, sondern sogar direkt während der Ausführung auf der eingebetteten Hardware.

Das Eclipse-Plugin ist für die Entwicklung von AmbiComp-Anwendungen einsatzfähig. Eventuell auftretende Fehler können mit Hilfe des ErrorLoggers nachvollzogen und behoben werden. Die Behebung eventuell auftretender Fehler wird durch Weitergabe des Quellcodes an die Stelle, die das Plugin nach Projektende weiter betreut, gewährleistet.

Die gesamte im Projekt entwickelte Software wurde unter einer dualen Lizenz, der GNU Public Licence (GPL) und der Mozilla Public Licence (MPL) veröffentlicht und steht damit Dritten zur Verwendung zur Verfügung. Nachdem Sven Schlender zum Projektende die Beecon GmbH verlassen hat, evaluiert er derzeit den möglichen Einsatz des AmbiComp-BIOS bei seinem neuen Arbeitgeber, der ESA Elektroschaltanlagen Grimma GmbH. Andere Ergebnisse des Projekts, vor allem die AmbiComp Virtual Machine inkl. Toolchain, sowie das AmbiComp Object Distribution Model werden in Projekten an der TU München weiter entwickelt.

Zum Ende der Projektlaufzeit haben sich leider die wirtschaftlichen Erfolgsaussichten weiter verschlechtert. Insbesondere blieb der kommerzielle Erfolg nach der CeBIT aus. Es gab zwar einiges Interesse an der AmbiComp-Plattform, das Konzept scheint aber mittlerweile für Unternehmen uninteressant geworden zu sein:

- Oft sind Unternehmen bereits an eine Hardware-Plattform gebunden und würden daher nicht unsere Sandwichmodule einsetzen.
- Für viele Anwendungen sind die Sandwichmodule noch deutlich zu groß.
- Für andere Anwendungen, wo die Abmaße der Steuergeräte keine solche Rolle spielen, werden eingebettete PCs mit Linux oder Windows bevorzugt.

Aus diesem Grund scheint der aussichtsreichste Markt zurzeit der Bereich von Forschung und Lehre zu sein, wo innovative Lösungen weiter reifen können. Hier sollen die erstellten Dokumentations- und Lehrmaterialien dazu dienen, die weitere Forschung an Ambient Intelligence Systemen zu unterstützen.

7 Arbeiten Dritter

Während die oben beschriebenen Arbeiten umgesetzt wurden, tauchten Mitte 2007 erste Ergebnisse Dritter in Form der „Sun SPOTS“ von der Firma Sun auf. Dabei handelt es sich um in Java programmierbare kleine Sensorknoten auf Basis eines ARM Prozessors mit einigen Megabyte Speicher. Im Vergleich zu den AmbiComp AICUs sind die Sun SPOTS damit deutlich leistungsfähiger, aber auch größer und teurer. Sie verfügen außerdem über keine umfangreiche und transparente Unterstützung einer verteilten Nutzung wie sie in AmbiComp geplant war. Eine Evaluation der Sun SPOTS an der HdM zeigte, dass das Produkt zwar schon recht weit entwickelt, jedoch bei weitem noch nicht ausgereift sind. Für den verteilten Betrieb wären ähnliche Probleme zu lösen wie im AmbiComp Projekt. Dies hat sich während der gesamten Projektlaufzeit und darüber hinaus bis heute nicht geändert.

Neben den Sun SPOTS wurde auch ein „Zigbee Development Kit“ von der Firma Meshnetics

erprobt. Diese Entwicklungsboards sind, wie die SMs, mit einem AVR 8-Bit-Mikrocontroller bestückt und schienen hierdurch besonders für die drahtlose Kommunikation in AmbiComp-Systemen geeignet. Aufgrund der Fokussierung auf die AmbiComp-Sandwichmodule konnten im weiteren Projektverlauf allerdings keine weitergehenden Tests mit diesen Boards durchgeführt werden.

Neben den Sun SPOTS wurden die „Particles“ Sensorknoten von der Firma Particle Computer GmbH, einem Spin-Off der Uni Karlsruhe, entwickelt. Diese Knoten konnten sich jedoch nicht am Markt durchsetzen und die Firma, die Eng mit SAP zusammen arbeitete, ist nicht mehr am Markt aktiv.

Im Laufe des Jahres 2008 tauchten neben den Sun SPOTS zwei weitere eingebettete Plattformen auf, die mit AmbiComp vergleichbar sind: die „Perk Kit“ der Firma Sentilla und die „BUGbase“ von der Firma Bug Labs. Die Firma Sentilla bot seit Mitte des Jahres ein speziell auf das Gebiet des Pervasive Computing ausgerichtetes Modul namens „JCreate“ an, das in Form des so genannten „Perk Kit“ als Evaluation Kit verfügbar war. JCreate kann als Nachfolger der aus dem Forschungsbereich Sensornetze bekannten „TMote Platform“ angesehen werden.

Eher an den Consumer Markt richtet sich das Angebot von Bug Labs, die ein modular erweiterbares Basisgerät - die so genannte BUGbase - anbieten. An dieses Basisgerät lassen sich größere Module für dedizierte Aufgaben anschließen. Der so individuell zusammengesteckte „Bug“ arbeitet mit einem ARM-Prozessor und lässt sich ausschließlich in Java programmieren.

In der Gesamtschau fiel auf, dass keines der vorgestellten Produkte Abstraktionsmechanismen für den Betrieb verteilter Anwendung anbietet. AmbiComp liefert hier entscheidende Impulse, indem es die Kommunikationsaspekte als inhärenten Bestandteil der Plattform betrachtet. So ist während der Projektlaufzeit und darüber hinaus bisher noch keines dieser Produkte an die Fähigkeiten der AmbiComp-Plattform herangekommen bzw. hat eine größere Marktdurchdringung erlangt.

Liste der Veröffentlichungen

1. Wissenschaftliche Veröffentlichungen

[MEIS06] *Thomas Fuhrmann. „Software Engineering for Ambient Intelligence Systems“. Proceedings of Workshop on Mobile and Embedded Interactive Systems (MEIS'06), Dresden, 6 Oktober 2006*

[IESE06] *Michalis Anastasopoulos, Frank Böhr, Thomas Patzke, Christian Peper, Daniel Schneider, Martin Sohn. „Ambient Intelligence Systems: Scenarios and Challenge“s. IESE-Report No. 191.06/E, Version 1.0, 22 Dezember 2006*

[FSOA07] *Daniel Schneider, Christian Bunse und Klaus Schmid, „Towards Adaptive Service Engineering”, Proceedings of the International Workshop on the Foundations of Service-Oriented Architecture (FSOA 2007), Juli 2007*

[GI07] *Björn Saballus, Johannes Eickhold und Thomas Fuhrmann, „Towards a Distributed Java VM in Sensor Networks using Scalable Source Routing”, 6. Fachgespraech Sensornetzwerke der GI/ITG Fachgruppe 'Kommunikation und Verteilte Systeme', Aachen, Germany, July 16-17, 2007*

[SEAMS08] *C. Peper und D. Schneider: „Component Engineering for Adaptive Ad-hoc Systems”, Proc. of the 30th Intl. Conf. on Software Engineering ICSE'08, SEAMS Workshop (Software Engineering for Adaptive and Self-Managing Systems), Leipzig, Mai 2008, Leipzig.*

[SENS08] *Bjoern Saballus, Johannes Eickhold und Thomas Fuhrmann, „Global Accessible Objects (GAOs) in the AmbiComp Distributed Java Virtual Machine”, Proceedings of the Second International Conference on Sensor Technologies and Applications (SENSORCOMM 2008), Cap Esterel, France, August 25-31, 2008*

[KuVS08] *Johannes Eickhold, Björn Saballus und Thomas Fuhrmann. „A Distributed Virtual Machine for Ambient Computing”, Poster beim 7. GI/ITG KuVS Fachgespräch „Drahtlose Sensornetze“, 25./26. September 2008*

[AMI08] *Johannes Eickhold, Thomas Fuhrmann, Björn Saballus, Sven Schlender, und Thomas Suchy. „AmbiComp: A platform for distributed execution of Java programs on embedded systems by offering a single system image”, Proceedings of the European Conference on Ambient Intelligence (AmI-Blocks '08 Workshop), Nürnberg, Germany, November 19, 2008*

[IWCMC09] *Pengfei Di and Thomas Fuhrmann, „Using Link-Layer Broadcast to Improve Scalable Source Routing“, Proceedings of the International Conference on Wireless Communications and Mobile Computing (IWCMC'09), Leipzig, Germany, June 2009*

[HPDC09] *Björn Saballus und Thomas Fuhrmann „Maintaining Reference Graphs of Globally Accessible Objects in Fully Decentralized Distributed Systems”, In Proceedings of the International ACM Symposium on High Performance Distributed Computing (HPDC 2009), Munich, Germany, June 2009*

[STM] *Annette Bieniusa and Thomas Fuhrmann, „A Description and Analysis of 'dstm', a Fully Decentralized Software Transactional Memory Algorithm“,Technical Report TUM-I0920, August 2009*

[SINTER09] *C. Peper und D. Schneider: „On Runtime Service Quality Models in*

Adaptive Ad-hoc Systems“, SINTER '09: Proceedings of the 2009 ESEC/FSE workshop on Software integration and evolution @ runtime, Amsterdam, September 2009

[DEEF09] Sven Schlender, Thomas Suchy, Björn Saballus, Johannes Eickhold und Thomas Fuhrmann. AmbiComp: Eine Plattform zur einfachen Programmierung von verteilten eingebetteten Systemen. Tagungsband Design & Elektronik Entwicklerforum Embedded-System-Entwicklung, Ludwigsburg, 7-8 Oktober 2009. WEKA FACHMEDIEN GmbH, 2009

[DASC09] Daniel Schneider und Mario Trapp, „Runtime Safety Models in Open Systems of Systems,” In Proceedings of the Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC), pp. 455-460, Chengdu, China, 2009

[IPDPS10] Annette Bieniusa and Thomas Fuhrmann, „Consistency in Hindsight, A Fully Decentralized STM Algorithm”, Proceedings of the 24th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2010), April 19-23, 2010, Atlanta, Georgia, USA

2. Studentische Arbeiten

- Markus Heimann, „Entwicklung von Schnittstellen zur Konfiguration und Verwaltung eines Ambient Intelligence Systems“, Diplomarbeit, Hochschule der Medien, Oktober 2007
- Malte Cornils, „Integration von Bluetooth in die AmbiComp Plattform für eingebettete Systeme“, Studienarbeit, Uni Karlsruhe, Februar 2008
- Sven Schlender, „AICU BIOS, Design und Implementierung einer Hardware-Abstraktions-Schicht für AmbiComp Sensorknoten“, Studienarbeit, Uni Karlsruhe, März 2008
- Alexander Kiening, „Accessing Remote Objects in a Distributed Embedded Java VM“, Diplomarbeit, TU München, Mai 2008
- Mathias Kellerer, „Extension of the Scalable Source Routing Protocol for Addressing and Migrating Global Accessible Objects“, Diplomarbeit, TU München, Dezember 2008
- Clemens Koller, „Software Transactional Memory for the AmbiComp distributed Java Virtual Machine”, Diplomarbeit, Uni Karlsruhe, Februar 2009
- Christian Schiepe, „Konzeption und Umsetzung eines Java-basierten Frameworks zur Entwicklung von visuellen Interaktionsmöglichkeiten von AmbiComp Anwendungen“, Diplomarbeit, Hochschule der Medien, Februar 2009
- Pascal Birnstill, „Efficient Detection and Utilization of Asymmetric Links in Scalable Source Routing (SSR)“, Diplomarbeit, Uni Karlsruhe, März 2009
- Fabian Knittel, „Scalable Source Routing in the AmbiComp Environment“, Studienarbeit, Uni Karlsruhe, April 2009
- Wolf-Dennis Pahl, „In-System-Debugging von Java-Programmen in der AmbiComp-Virtual-Machine“, Studienarbeit, Uni Karlsruhe, Juli 2009
- Martin Melullis, „Distributed Embedded Systems in Java - Design of a Lab Course“, Studienarbeit, Uni Karlsruhe, Dezember 2009
- Thomas Kittel, „Design und Implementierung einer Hardwareabstraktionsschicht für die AmbiComp Virtual Machine und Entwicklung eines Frameworks zur Simulation von kompletten AICUs“, Software Entwicklungsprojekt, TU München, Februar 2010

3. Webseite des AmbiComp Projektes

Spezifikationen

- The AmbiComp Virtual Machine
- The AmbiComp BIOS
- The AmbiComp Object Distribution Model

Datenblätter

- AmbiComp
- Sandwich Module (allgemeine Informationen)
- Bluetooth Sandwich Module
- Bluetooth Class 1 Sandwich Module
- Ethernet Sandwich Module
- Input/Output Sandwich Module
- Backplane Primary Supply Sandwich Module
- Backplane Secondary Supply Sandwich Module
- Power-over-Ethernet Sandwich Module
- Peripheral Sandwich Module
- Accu Sandwich Module
- AmbiComp Eclipse Plugin (Software)
- HW Development Tools (DevBoard, AICU-Interface)
- AICU-Access Module

Sonstiges

- Doxygen Dokumentation: BIOS (Source Code Dokumentation)
- Thomas Suchy, Jürgen Butz, Johannes Maucher, Harald Orlamünder „Ambient Intelligence Definition“, Version 0.6, 01 Februar 2007
- Thomas Suchy, Jürgen Butz, Johannes Maucher, Jörg Rech „Anforderungsanalyse Eclipse-Plugin“, Version 1.0.0, 15. August 2008