Bluetooth in Sensornetzwerken

Hagen Burchardt, Marc Haase, Frank Golatowski

Universität Rostock, Fachbereich Elektrotechnik und Informationstechnik
Institut f. Angewandte Mikroelektronik und Datentechnik
{marc.haase, hagen.burchardt, frank.golatowski}@etechnik.uni-rostock.de

Zusammenfassung: Dieser Artikel beschreibt die Entwicklung und Implementierung eines drahtlosen Sensornetzwerkes. Die Sensorknoten werden von einem Mikrocontroller gesteuert und kommunizieren mittels Bluetooth drahtlos über Peer-to-Peer Verbindungen miteinander. Für die Sensorknoten wurde ein minimaler Bluetooth-Stack für Mikrocontroller entwickelt. Über eine Gateway-Schnittstelle erfolgt der Übergang zum drahtgebundenen Netzwerk. Die Sensorknoten kommunizieren über ein ebenfalls entwickeltes leichtgewichtiges Protokoll mit dem Gateway. Dieses ist speziell für Sensornetzwerke mit heterogenen Sensorknoten geeignet, da es eine automatische Identifikation und Registrierung von Sensoren unterstützt.

Einleitung

Die Fortschritte auf den Gebieten der drahtlosen Kommunikation und der Mikroelektronik ermöglichen die Entwicklung von kostengünstigen, energiesparenden, kleinen Sensorknoten, die drahtlos über kurze Distanzen Daten austauschen können. Diese ermöglichen den Aufbau von verteilten Sensornetzwerken mit einer großen Anzahl von Sensorknoten. Der Zugriff auf diese Netze erfolgt meist über einen zentralen Knoten, der gleichzeitig die Schnittstelle zwischen drahtlosem Sensornetzwerk und einem drahtgebundenem Netz darstellt. An dieser Schnittstelle werden Access Points oder Gateways eingesetzt, die die von dem Sensornetzwerk erfassten Daten als Services auf einer höheren Abstraktionsebene dem drahtgebundenem Netzwerk anbieten. Der Gateway wird demzufolge auch als Gateway bezeichnet.

Die *Open Services Gateway Initative* OSGI [OSGI99] entwickelt eine allgemeine Spezifikation für Service-Gateways. Diese berücksichtigt im Ansatz diverse lokale Netzwerkarchitekturen und Breitbandnetzwerke, um eine Bereitstellung von Servicen zu unterstützen. Die Spezifikation soll praktisch alle verfügbaren Netzwerkstandards und Initiativen über Service-Gateways zusammenführen. Sie definiert Programmierschnittstellen-Standards (API) für Gateway-Plattformen, in denen Abhängigkeiten zwischen Servicen, Datenmanagement, Gerätemanagement, Klientenzugriff, Ressourcenmanagement und Sicherheit adressiert werden. Durch die Verwendung dieser API können Endanwender netzwerkbasierte Services von Serviceprovidern laden, während ein Gateway die Installation, die Versionskontrolle und die Konfiguration dieser Services kontrolliert.

SUN bietet mit der *Surrogated Host Architecture* [SUR01] eine auf Java basierende Schnittstelle für Service-Gateways, die als Surrogated Host bezeichnet werden. Die dieser Architektur zugrunde liegende JINI™ Technologie ermöglicht die Entwicklung von Systemen, die auf verteilte Komponenten über ein Netzwerk zugreifen. Der Surrogated Host ist dabei die Schnittstelle zwischen der Java basierten Infrastruktur und den ressourcenarmen Systemen, die von sich aus nicht in der Lage sind, Java zu unterstützen.

Zusammenfassend lässt sich sagen, dass die beiden vorgestellten Architekturen zwar ein leistungsfähiges Interface zum Zugriff auf Geräte über einen Service-Gateway bieten, jedoch für Sensornetzwerke ungeeignet sind, da es kein defacto Standard-Protokoll für Sensornetzwerke gibt, auf das diese Systeme aufgesetzt werden können. Diese Systeme sind für Enterprise-Lösungen meist java-zentriert ausgerichtet und damit für Sensorsysteme ungeeignet. Die Frage des Aufbaus des Sensornetzwerkes wird ebenfalls nicht durch die oben genannten Architekturen gelöst.

In diesem Artikel präsentieren wir eine Architektur, die die Funktionalität der oben genannten Technologien für ein Sensornetzwerk ermöglicht. Ein Gateway bildet die Schnittstelle zwischen Sensornetzwerk und drahtgebundenem Netzwerk. Diese Architektur bietet:

- die Selbstkonfiguration des Sensornetzwerkes durch automatischen Identifikation und Registrierung von Sensorknoten und
- den Zugriff auf Sensorknoten aus dem drahtgebundenen Netzwerk über eine API des Gateways.

Im Gegensatz zu den oben genannten Servicearchitekturen wird auf die Verwendung von Java und anderer resourcenbeanspruchender Toolkits verzichtet. Stattdessen wird versucht, trotz der begrenzten Ressourcen der Sensorknoten, eine dynamische, servicebasierte Interaktion mit den Sensorknoten zu gewährleisten.

Nachfolgend werden beginnend mit dem Aufbau eines Sensorknotens, der speziell für Sensorknoten angepasste Bluetooth Protokoll-Stack beschrieben und die Netzwerkarchitektur und die dafür entwickelten Protokolle vorgestellt. Anschließend folgt die Beschreibung der Referenzimplementierung des Gesamtsystems.

Aufbau des Sensorknotens

Ein Sensorknoten besteht aus Sensoren, Aktoren, einer Steuereinheit, einem I/O-Interface und einer Energiequelle. Abbildung 1 zeigt den Aufbau des Sensorknotens. Die Steuereinheit ist in Abhängigkeit von der Größe des Sensorknotens ein Mikrocontroller oder ein Embedded PC. Das I/O-Interface dient sowohl der Übertragung der Daten zum Host als auch dem Anschluss der Sensoren und Aktoren. Die drahtlose Kommunikation erfolgt durch ein an die serielle Schnittstelle angeschlossenes Bluetooth-Modul.

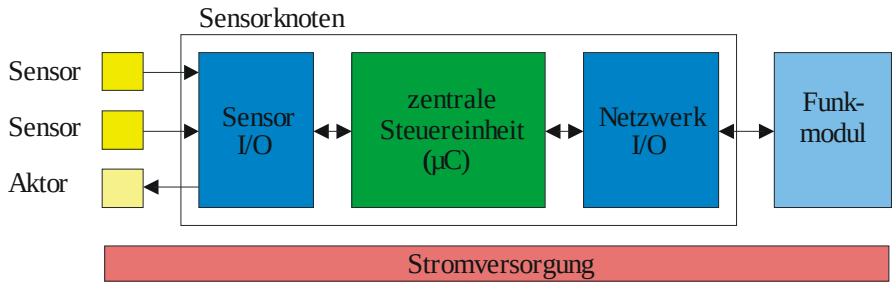

Abbildung 1: Aufbau der Sensorknotens

Bluetooth-Stack für Sensorknoten

Innerhalb des drahtlosen Sensornetzwerkes wird die Bluetooth-Technologie eingesetzt, da sie speziell für Systeme mit begrenzten Ressourcen in Bezug auf Rechenleistung und Energieversorgung entwickelt wurde. Die Steuerung des Bluetooth Interfaces erfolgt durch einen von uns speziell an die begrenzten Ressourcen des Sensorknotens angepassten Bluetooth Protokoll-Stack. Dieser enthält die Protokollschichten HCI und L2CAP und ist für Mikrocontroller ohne Betriebssystem geeignet.

Ausgehend von dem Open Source Linux Bluetooth-Stack BlueZ [Bl02] wurde ein für Mikrocontroller angepasster Bluetooth Stack entwickelt. Folgende Änderungen und Erweiterungen wurden am BlueZ-Stack vorgenommen. Das Scheduling des Betriebssystems wurde durch einen zyklischen Ablauf ersetzt. Das Interrupt Handling für die serielle Schnittstelle wurde implementiert und die Datenstrukturen des BlueZ-Stacks wurden an das Speichermodell des Mikrocontrollers angepasst. Zusätzlich wurde ein Timersystem hinzugefügt, das das Scheduling des nicht vorhandenen Betriebssystems ersetzt. Die auf Sockets basierte Programmierung des BlueZ-Stacks wurde aus programmiertechnischen Gründen beibehalten. Hierzu war die Anpassung der Socket-Implementierung an die begrenzten Ressourcen des Mikrocontrollers erforderlich.

Auf einem Motorola 68HC12 benötigt der modifizierte Bluetooth Stack, den wir µBlueZ nennen, 24 KB Programmspeicher und weitere 4 KB Datenspeicher inklusive der Datenpakete. Abbildung 2 zeigt den Aufbau des BlueZ-Stackes und im Vergleich dazu den µBlueZ. Daraus ist zu erkennen, dass die gesamte Funktionalität des Bluetooth-Stacks bei der Portierung beibehalten wurde und auf dem Mikrocontroller zur Verfügung steht.

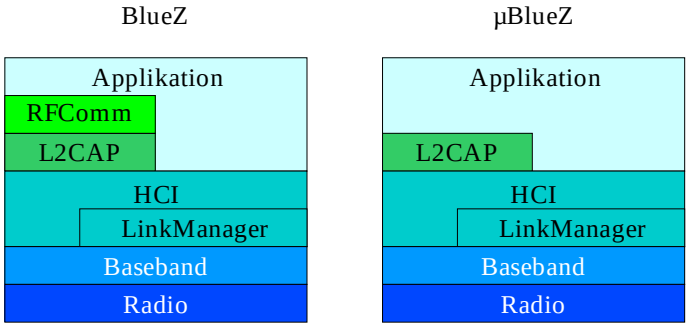

Abbildung 2: Aufbau von BlueZ und µBlueZ

Netzwerkarchitektur

Die dem Sensornetzwerksystem zu Grunde liegende Netzwerkarchitektur ist in Abbildung 3 dargestellt. Sie besteht aus dem drahtlosen Sensornetzwerk, dem Gateway und der drahtgebundenen Infrastruktur mit Client-Applikationen (Abbildung 3).

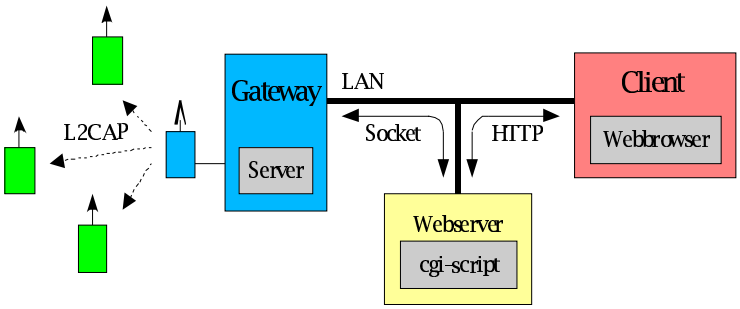

Abbildung 3: Architektur des Gesamtsystems, bestehend aus
Sensornetzwerk (links), Gateway (mitte) und drahtgebundener Infrastruktur (rechts)

Zwischen Webserver und Client werden HTML-Seiten übertragen. Das CGI-Script auf dem Webserver wird durch das Anfordern der Seite aufgerufen. Es baut eine Internet-Socket-Verbindung zwischen Webserver und Access Point auf. Über diese Verbindung wird dann der Befehl an den Access Point gesendet. Der Access Point verarbeitet den Befehl, verbindet sich in Abhängigkeit vom Kommando mit dem Sensorknoten und sendet die Antwort auf demselben Weg zurück zum CGI-Script. Im darauf folgenden Schritt wird die Webseite generiert und an den Webbrowser des Client gesendet.

Im Sensornetzwerk erfolgt die Datenübertragung auf der Ebene des Logical Link and Adaptation Protocols (L2CAP) der eingesetzten Bluetooth Technologie und in der drahtgebundenen Infrastruktur auf TCP/IP Ebene. Die Sensorknoten kommunizieren mit dem Gateway über Peer-to-Peer Verbindungen, die nur für die Dauer des Datenaustausches aktiv sind. Der Gateway muss sich in Reichweite der Sensorknoten befinden.

Der Gateway bildet die Schnittstelle zwischen Sensornetzwerk und drahtgebundenem Netzwerk. Jeder Sensorknoten registriert seine individuelle Konfiguration bei dem Gateway. Dieses tritt anschließend stellvertretend für alle Sensorknoten im drahtgebundenen Netzwerk auf und erfüllt folgende Funktionen:

- Verwaltung der Sensorknoten
- Übermittlung der Sensorliste an eine nachfragende Applikation
- Weiterleitung einer Client Anforderung an den adressierten Sensorknoten
- Übermittlung der Rückantwort an die Applikation
- Zwischenspeicherung von Abfragewerten und
- Timergesteuerte Abfrage von Sensorknoten.

Über eine definierte Gateway API lassen sich diese Funktionen aufrufen. Um die Anfragen vom Gateway an die Sensoren übermitteln zu können ist zusätzlich ein Sensornetzwerk-Prokoll notwendig, welches nachfolgend beschrieben wird.

Sensornetzwerk-Protokoll

Das entwickelte Sensornetzwerkprotokoll lässt sich in zwei Teilprotokolle unterteilen. Das erste Teilprotokoll steuert die Kommunikation zwischen Sensorknoten und Gateway und das zweite die Kommunikation zwischen Gateway und dem Client. Nachfolgend wird beginnend mit der Registrierung eines Sensorknotens beim Gateway die Funktionalität der Teilprotokolle beschrieben. Abbildung 4 zeigt den Protokollablauf.

Da der Gateway das gesamte Sensornetzwerk verwaltet, muss sich jeder neue Sensorknoten am Gateway anmelden. Dies erfolgt über ein leichtgewichtiges *Sensor Registration* Paket, das der Sensorknoten an den Gateway sendet (1). Es enthält die individuelle Konfiguration des Sensorknotens und die zur Steuerung des Knotens erforderlichen Parameter. Der Gateway nimmt anschließend den neuen Sensorknoten in die Liste der aktiven Sensorknoten auf und sendet eine Bestätigung an den Sensorknoten (2). Um nicht mehr im Netz erreichbare Knoten zu erkennen, wird die Anmeldeprozedur in periodischen Abständen wiederholt. Ein Client, der auf einen Sensorknoten zugreifen möchte, kontaktiert zuerst den Gateway (3) und erhält eine Liste der Sensoren mit ihren Parametern (4). Anschließend kann mit diesen ein Aufruf eines Sensorknotens erfolgen. Der Gateway empfängt das entsprechende *Request* Paket (5) und leitet es an den Sensor weiter (6). Der Sensorknoten führt das übermittelte Kommando aus und sendet das *Response* Paket an den Gateway (7) der die Antwort für den Client generiert (8).

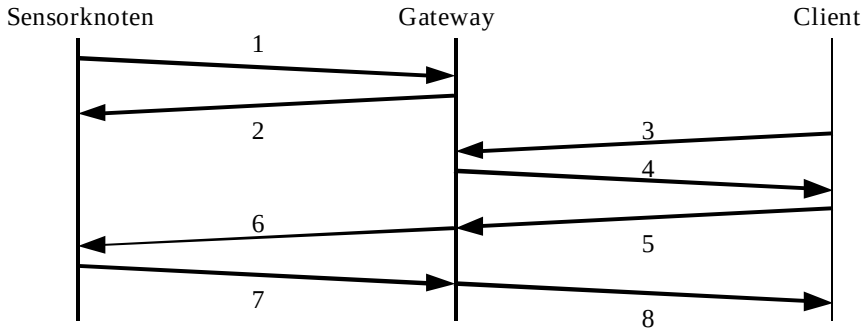

Abbildung 4: Ablaufdiagramm der Kommunikation zwischen Sensorknoten, Gateway und Client-Applikation

Da in Sensornetzwerken oft verschiedene Sensoren und Aktoren an die Knoten angeschlossen sind, ist eine hohe Flexibilität der Übergabe- und Rückgabeparameter notwendig. Das implementierte Sensornetzwerkprotokoll erlaubt es, für jeden Sensorknoten mehrere Kommandos mit mehreren Parametern sowie Rückgabewerten zu definieren. Es sind vier Datentypen definiert. Zahlenwerte werden in Byte, Word oder Longword übergeben. Für Zeichenketten steht ein null terminierter String zur Verfügung. Für jeden Parameter und Rückgabewert wird eine Beschriftung generiert, die später an den Client vom Gateway übergeben wird.

Abbildung 5 zeigt den Aufbau des Sensor Registration Paketes. Dieses besteht aus einer Liste von Templates. Jedes Template bildet ein Kommando ab und beginnt mit dem Kommandocode (CMD).

Hierfür ist ein Byte reserviert. Danach folgen die Kommandoparameter. Der Typ der Parameter wird durch ein Byte definiert (DCP). Ist der Datentyp eine Zahl, folgen ein Minimal- und Maximalwert. Jeder Parameter wird mit einem null-terminierten Beschriftungsstring abgeschlossen (Text). Die Rückgabewerte werden von den Parametern durch das ':'-Tag getrennt. Jeder Rückgabewert (DRP) beginnt wieder mit dem Typ gefolgt vom Beschriftungsstring. Ein ';'-Tag trennt die Kommandos voneinander.

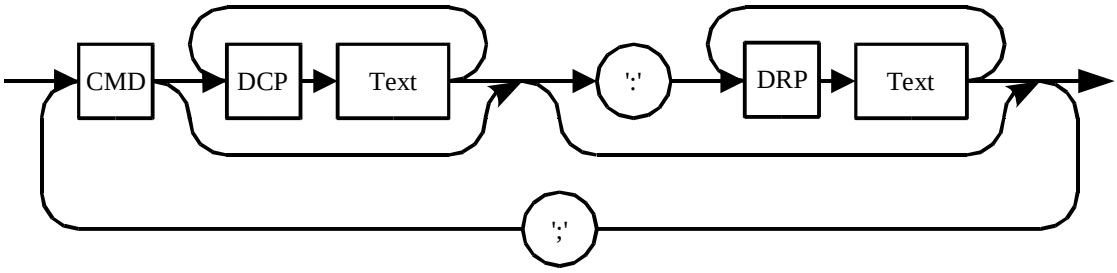

Abbildung 5: Aufbau des Service Registration Paketes

Bei der Anmeldung eines neuen Sensorknotens an den Gateway wird der zuvor beschriebene Templatestring übertragen. Dieser Templatestring kann von den Clients im drahtgebundenen Netzwerk abgerufen werden, um Informationen über einen Sensorknoten zu erhalten, und um Anfragen an den Sensor generieren zu können.

Kommandos an Sensorknoten werden anhand der vom Gateway übermittelten Templatestrings erstellt. Das erste Byte ist der Kommandocode. Danach folgen die Parameter je nach definiertem Typ (Abbildung 6). Nach dem Ausführen des Kommandos sendet der Sensorknoten eine Antwort an den Gateway. Diese besteht aus dem Response-Code und den Rückgabewerte (Abbildung 7).

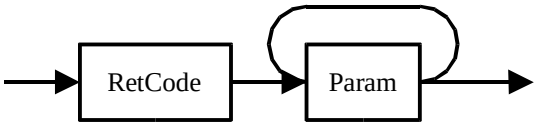

Abbildung 6: Anfrage an den Sensorknoten

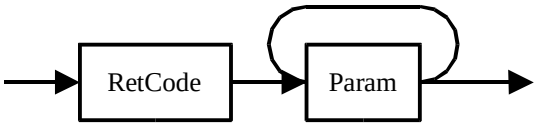

Abbildung 7: Antwort des Sensorknotens

Implementierung

Für die Evaluierung des Gesamtsystems ist eine Referenzimplementierung vorgenommen worden. Als Sensorknoten werden Mikrocontroller von Motorola (68HC12) [Mo02] und Mikroprozessoren (Sparc LEON) [Gai02] eingesetzt. Das Ericsson Bluetooth Modul ROK 101007 [Er02] wird für die drahtlose Kommunikation eingesetzt und ist an die serielle Schnittstelle des Sensorknotens angeschlossen. Als Gateway wird ein eingebettetes Linux System der Firma AXIS [Ax02] eingesetzt. An dieses ist ebenfalls ein Ericsson Bluetooth Modul angeschlossen und wird über den offiziellen BlueZ-Stack angesteuert. Dieses System ist gleichzeitig über eine Ethernet-Schnittstelle mit dem drahtgebundenen Netzwerk verbunden. Applikationsseitig kommt ein stationäres Linux System zum Einsatz, das neben dem Client auch den HTTP-Server enthält. Abbildung 8 zeigt die aus dem Template-Strings generierte Oberfläche des Clients.

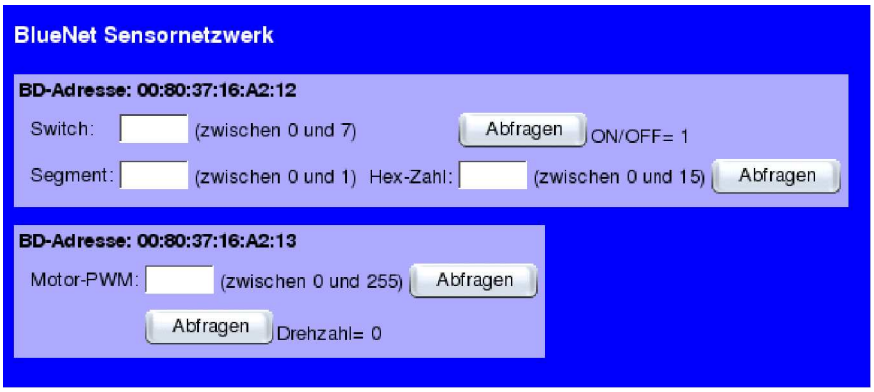

Abbildung 8: Aus den Template-Strings generierte Oberfläche

Zusammenfassung

Die in diesem Artikel vorgestellte Sensornetzwerkarchitektur ermöglicht die dynamische Integration von drahtlosen Sensorknoten, durch einen vom Sensorknoten selbst initiierten Registrierungsprozess. Der Datentransport innerhalb des drahtlosen Netzwerkes erfolgt auf dem L2CAP Layer von Bluetooth. Ein weiterer Vorteil der vorgestellten Architektur ist der Einsatz von Standardprotokollen zur Kommunikation mit den Sensorknoten, ohne eine javabasierte Servicearchitektur zu nutzen.

Ein weiteres Ergebnis der Arbeit ist der in der Referenzimplementierung entwickelte Bluetooth-Protokoll-Stack für Mikrocontroller. Dieser benötigt auf einem Motorola 68HC12 ungefähr 24 KB Programmspeicher und 4 KB Arbeitsspeicher und bietet die vollständige Funktionalität des HCI- und L2CAP-Layers.

Literaturverzeichnis

[Ax02] AXIS Developer Board LX, http://developer.axis.com/products/devboard/index.html

[Bl02] BlueZ, Official Linux Bluetooth protocol stack, http://bluez.sourceforge.net/

[Gai02] Gaisler Research, LEON P1754 Proessor, http://www.gaisler.com/

[Er02] Ericsson ROK101007 Bluetooth Modul http://www.ericsson.com/microe/products/bluetooth_solutions/mcm.shtml

[Mo02] Motorola Semiconductors, http://e-www.motorola.com/

[OSGI99] Open Service Garteway Architecture OSGI, http://www.osgi.org, 1999

[Sur01] The Jini™Technology Surrogate Architecture Overview, März 2000 http://developer.jini.org/exchange/projects/surrogate/overview.pdf

Kontakt:

Dipl.-Ing. Hagen Burchardt
Universität Rostock
Fachbereich Elektrotechnik und Informationstechnik
Institut f. Angewandte Mikroelektronik und Datentechnik
Richard-Wagner Str. 31
18119 Rostock-Warnemünde

Tel.: ++49 (0)381 498 – 3629
Fax: ++49 (0)381 498 – 3601
e-mail: hagen.burchardt@etechnik.uni-rostock.de