Einsatz **von OSGi auf Hardwarelimitierten Modulen**

Pavel Haiduk, Peter Heusinger, Michael Wagner

Fraunhofer Institut Integrierte Schaltungen
Projektgruppe Netzzugangstechnik, Nordostpark 93
90411 Nürnberg, Germany
(hdk,hsr,wgm)@iis.fhg.de
http://www.iis.fhg.de/ec/nat/index_d.html

Abstract. Der Beitrag beschäftigt sich mit internetbasierten Services auf embedded Systemen. Zuerst wird ausführlich auf die OSGi-Technologie eingegangen. Es werden alle notwendigen Komponenten vorgestellt. Dabei wird auf einige Dienstleistungen näher eingegangen und auch die entsprechenden gerätetechnischen Voraussetzungen für diese Services genannt. Dies bezieht sich insbesondere auf die Themen Netzzugang und die Verbindung mit haus- bzw. fabrikinternen Bussystemen wie Feldbusse, Gebäudebusse, Multimediakommunikation aber auch die einfache Kommunikation zwischen einzelnen Geräten. Resultat dieser Betrachtungen wird eine geeignete Architektur des Dienstleistungsgateways sein, die mit Kommunikationsschnittstellen zum Internet und zu unterschiedlichsten Anwendungen ausgestattet ist. Zum Abschluß werden drei Applikationen vorgestellt, die auf einem solchen Gateway einzeln oder miteinander laufen können.

1. Einführung

Das Internet wird in den nächsten Jahren nicht nur als Informationsbeschaffungsmedium in die einzelnen Haushalte dringen, sondern als das globale Vernetzungsmedium dem Verbraucher verstärkt die unterschiedlichsten Dienstleistungen anbieten. Als Beispiele solcher Dienstleistungen seien hier Telemedizin, Hausbelieferungsdienste (Home Replenishment), Facility-Management, Sicherheitsüberwachung des Eigenheims aber auch Freizeit- und Urlaubsplanung genannt. Allen diesen potentiellen Dienstleistungen ist gemeinsam, dass ein Kunde sie zu beliebigen Zeitpunkten abonnieren aber auch jederzeit wieder abbestellen kann.

Keiner der genannten Services muss zwangsläufig auf einem PC laufen, in einigen Situationen ist dies sogar kontraproduktiv. Das Gateway im Haus ist die zentrale Dienstleistungszentrale, die viele unterschiedliche internetbasierte Dienstleistungen zur Verfügung stellen kann. Der Einsatz als Consumer Produkt verbietet die Verwendung eines PCs für diese Aufgabe. Insbesondere bei älteren Leuten würde ein Gerät auf erhebliche Akzeptanzprobleme stoßen. Aber auch Kostengründe verbieten häufig den Einsatz des PCs bei vielen potentiellen Dienstleistungen. Um eine Gateway-Plattform möglichst universell einzusetzen, bedarf es eines Standards, der es ermöglicht, Softwarekomponenten der Dienstleistung auf diese Plattform zu laden, zu verwalten und in Betrieb zu nehmen. Diese Aufgabe übernimmt OSGi.

2. OSGi-Grundlagen

OSGi steht als Abkürzung für Open Services Gateway Initiative und wurde im März 1999 gegründet. Zu den Gründungsmitgliedern gehören große amerikanische Firmen wie SUN Microsystems, IBM, Oracle und Echelon. Im Rahmen der Aktivitäten ist ein auf Java basierender Standard festgelegt worden, der die nächste Generation von Internet Services im Heim, Auto, kleinen Büros und Fabriken ermöglichen soll. Basierend auf unterschiedlichsten vernetzten Hardware Plattformen wie Settop-Boxen, Gateways, Kablemodems aber auch Telekommunikationssystemen erlaubt OSGi das Management von hardwareunabhängigen Services. Als Basis stellt das sogenannte OSGi Framework eine Umgebung zur Verfügung, auf der online ladbare Services ausgeführt werden. Dabei ist ein Service die kleinste atomare Einheit und wird mit eventuell weiteren Services zu sogenannten Bundles zusammengefasst. Eine spezielle Registrierungsdatenbank ermöglicht den Export und Import von Services zwischen verschiedenen Bundles. Benötigt ein Service die Unterstützung durch Services in anderen Bundles, werden diese online nachgeladen. Spezielle Sicherheitsmechanismen gestatten dem Administrator die vollständige Kontrolle über die Plattform. Die Java Umgebung ermöglicht die einmalige Entwicklung eines Services für die unterschiedlichsten Geräte mit verschiedensten Prozessoren.

Fig. 1 zeigt den Aufbau einer OSGi-Implementierung als Schichtenmodell. Aufbauend auf beliebiger Hardware arbeitet ein Echtzeitbetriebssystem, das den gleichzeitigen Betrieb mehrere Softwareprozesse gestattet. Zu den Eigenschaften der Hardware gehört ein 32 Bit Prozessor für den leistungsfähige Betrieb einer Java Virtual Machine. Programm- und Datenspeicher müssen auf die eigentliche Aufgabe angepasst sein. Das Java Runtime Environment sollte aus einer Java 2 Virtual Machine und den dazugehörigen Java Applikationsinterfaces bestehen. Aufbauend auf dieser Umgebung kann das in Java geschriebene OSGi-Framework seine Aufgabe erfüllen. Die installierten Bundles dienen als Container für die jeweiligen Services, die entweder aus Java-Code aber auch aus nativen Code bestehen. Im Vergleich zu puren Java-Services entsteht durch die Verwendung von nativen Code das Problem, dass der in den Bundles gespeicherte Code für alle potentiell verwendeten Prozessoren und Betriebssystemen in compilierter Form vorliegen muss. Nur so kann gewährleistet werden, dass die Services des entsprechenden Bundles auf jeder

Zielplattform ausgeführt werden können. Ist diese Bedingung nicht erfüllt, verweigert das OSGI-Framework die Installation und den Betrieb des jeweiligen Bundles.

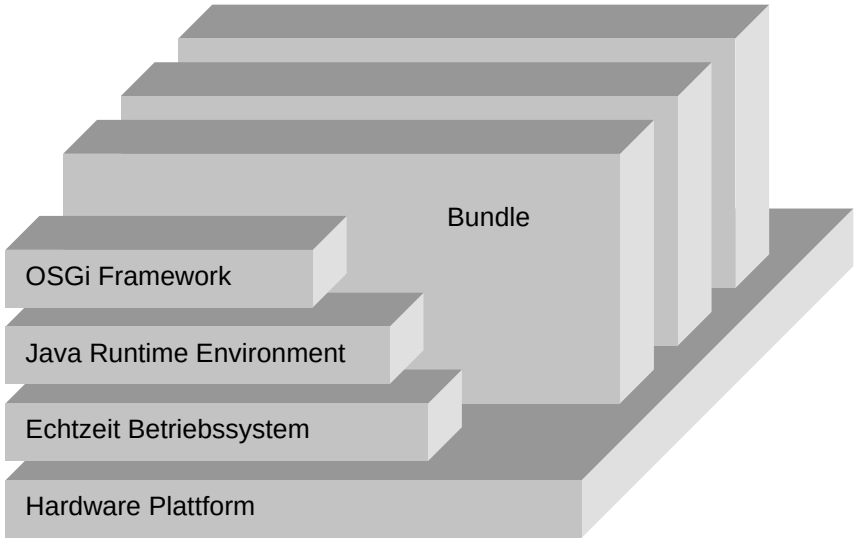


Fig. 1. Schichtenmodell einer OSGi - Implementierung

3. OSGi detailliert

Neben dem Framework beinhaltet eine OSGi Implementierung noch eine weitere Schlüsselkomponente, den sogenannten Device Access. Er ist für die Schnittstelle zwischen Gateway und den damit verbundenen Applikationen verantwortlich. Ein grundlegendes Verständnis von OSGi ist durch näheren Einblick in die Funktionsweise von Framework und Device Access möglich.

3.1 OSGI-Framework

Das Framework stellt die grundlegenden Funktionen von OSGi zur Verfügung. Zu diesen Aufgaben zählt der sichere Download von erweiterbaren Services, die in Form von Bundles zusammengefasst sind. Das Framework organisiert den Installations- bzw. Deinstallationsvorgang, beachtet Abhängigkeiten zwischen den Services, sorgt für dynamisches Nachladen von noch fehlenden Softwaremodulen. Seine Funktionalität ist insbesondere für den Betrieb in speicherlimitierten Baugruppen ausgelegt.

Die Kommunikation mit dem OSGi-Framework und den eigentlichen Services bestehen aus den drei Hauptkomponenten.

- Bundle
- BundleContext
- Service

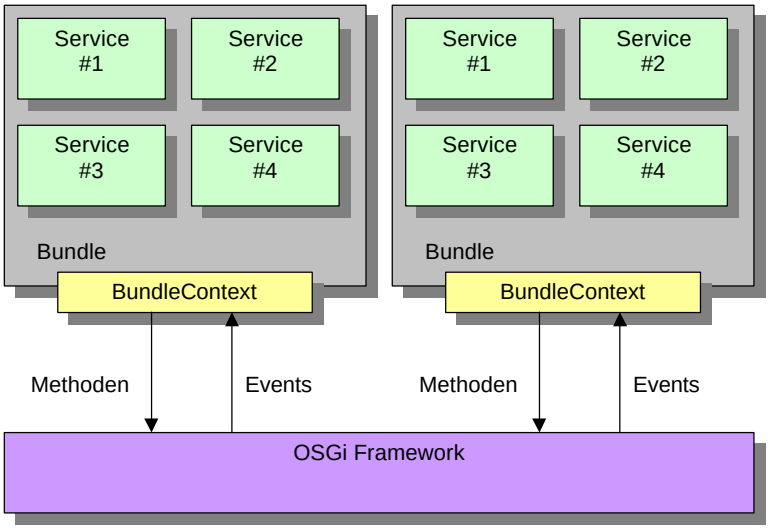


Fig. 2. Informationsaustausch zwischen Bundles und dem Framework

3.2 Bundle

Ein Bundle beinhaltet alle Informationen, die zum Betrieb eines speziellen Service notwendig sind. Dazu zählen Java-Klassen für das OSGi-Managment des jeweiligen Services, aber auch die eigentliche Service-Funktionalität selbst. Weiterhin können Ressourcen wie HTML-Daten, Bilder aber auch native Bibliotheken in einem Bundle zusammengefasst sein. Der eigentliche Bundle-Container sorgt für den Transport über das Kommunikationsnetzwerk und die Installation. Er besteht aus einer Java Archiv Datei, JAR-Archiv genannt. Das Format dieses Archivs ist mit dem bekannten WINZIP-Format auf dem PC identisch. Eine spezielle Textdatei mit dem Namen Manifest beschreibt, was in einem Bundle-Container aktuell zusammengefasst ist. Wie alle bereits genannten Ressourcen ist auch diese Datei in dem dazugehörigen Archiv abgelegt. Jedes Bundle kann einen oder mehrere Services enthalten, die im Rahmen der Installation vom Framework gestartet, aktualisiert und gestoppt werden können. Abhängigkeiten der Services mit anderen Bundles sind in

einer eigenen Sektion in der Manifest-Datei dokumentiert. Im Rahmen der Installation eines Services erzeugt das Framework für jeden Bundle ein eigenes Bundle-Objekt, das als Verwaltungsobjekt alle relevanten Daten innerhalb des Frameworks hält. Dabei legt die OSGi-Spezifikation die Interfaces dieses Bundle-Objekts als Java-Interfaces fest.

3.3 Bundle Context

Beziehungen zwischen dem Framework und dem jeweils installierten Bundle werden über einen sogenannten Bundle Context geregelt. Er ist das eigentlich ausführende Installationsmanagment Programm des Bundles. Auf ihn verweist die Manifest-Datei. Der Bundle Context ist als Java Interface definiert und stellt verschiedene Funktionen zur Verfügung, um auf das Framework oder Bundle-Objekte zugreifen zu können. Der Programmierer eines Bundles definiert eine Klasse, die das Bundle Context Interface realisiert. Sobald das Framework den Bundle installiert, erstellt es eine Instanz dieser Klasse und ruft dann den Bundle Context auf, der ab diesem Zeitpunkt für die Installation der einzelnen Services verantwortlich ist.

Dazu stehen ihm verschiedene Funktionen zur Verfügung

- An- und Abmelden beim Framework, um mit System Events versorgt zu werden
- Registrierung von Services aus dem Bundle in der Framework Registry, die nach diesem Vorgang innerhalb des Framworks verfügbar sind.
- Abfrage von Informationen zu einem bestimmten Service
- Belegung und Freigabe von Services
- Installation weiterer Bundles
- Abfrage aller installierten Bundles
- Erzeugung von Dateien im Dateisystem des Gateways

3.4 Service

Services arbeiten in der OSGi Umgebung als Java-Objekte mit festgelegten Interfaces. Bei kundenspezifischen Services gibt der Standard keine genaue Struktur vor, wie diese Schnittstellen auszusehen haben. Neben den kundenspezifischen Realisierungen existieren allerdings innerhalb der OSGi Spezifikation eine Vielzahl verschiedener allgemein verwendbarer Services, die für interoperable Zusammenarbeiten eingesetzt werden können. Weitere Spezifikationen werden in naher Zukunft folgen. Das Kapitel "5 Spezielle OSGi Services" weiter unten in diesem Proposal gibt einen ersten Einblick in diese vorgegeben Services.

Damit ein Service Objekt frei verfügbar eingesetzt werden kann, muss das übergeordnete Bundle es registrieren. Beziehungen zwischen Bundles organisiert dann das Framework und sorgt so z.B. für das korrekte Aufräumen, wenn Services

deinstalliert werden. Damit der richtige Service für eine spezielle Aufgabe gefunden werden kann, existiert ein spezieller Query-Mechanismus. Die einfachste Methode, den richtigen Service für eine Aufgabe zu finden ist die Suche über seinen Objektnamen. Zusätzlich existieren verschiedene Eigenschaften, die das Service-Objekt näher beschreiben z.B. Herstellerangaben, Versionsnummer, etc.. Über spezielle Events benachrichtigt das Framework alle installierten Bundles, wenn ein neuer Service installiert oder ein bestehender Service deinstalliert worden ist.

4. Device Access

Auf einem Gateway müssen Services und angeschlossene Geräte in geeigneter Form miteinander interagieren. Das Aktivieren von Services und Geräten stellt einen dynamischen Vorgang dar, der sich zu jedem beliebigen Zeitpunkt durch externe Einflüsse ändern kann. So können z.B. Geräte über Netzwerkschnittstellen wie USB oder FireWire während des Betriebs angeschlossen oder entfernt werden. Einige Netzwerkprotokolle unterstützen dieses sogenannte Hot Pluggin und fragen bei Bedarf über spezielle Lookup-Dienste dynamisch die Eigenschaften der neu erkannten Geräte ab. Eine Aufgabe des Gateways ist nun in ähnlich dynamischer Form, einen geeigneten Gerätetreiber zu finden, zu laden und mit passenden Services zu verbinden.

Dabei müssen folgende Vorgaben eingehalten werden.

- OSGi muss auf limitierten Embedded Geräten lauffähig sein
- Remote Administration ist zu unterstützen
- Herstellerunabhängigkeit muss gewährleistet sein
- Dauerbetrieb ohne Unterbrechungen muss möglich sein, auch bei Geräteinstallationen
- Dynamischer Update zu jedem beliebigen Zeitpunkt ohne Neustart des Systems

Für die korrekte Installation von Geräten ist in der OSGi Spezifikation der Device Manager vorgesehen. Er muss dafür Sorge tragen, dass zum jeweiligen Device auch der richtige Treiber aktiviert wird. Dafür gibt es einen speziellen Match-Algorithmus, der die Wahl des richtigen Treibers sicherstellen soll. Für die Verbindung zu Services gibt es eine große Anzahl verschiedener Treiber, die unterscheiden müssen, ob eine physikalische Ressource des Gateways wie z.B. Analog Digital Umsetzer und parallele Schnittstelle anzusteuern ist, oder ob der Treiber ein Gerät repräsentiert, das an einem Netzwerk hängt und über dieses geeignet angesteuert werden muss. Zusätzlich sind auch die Zugriffsebenen zu unterscheiden. Ein Gerät an einem USB-Anschluss kann physikalisch über Rohdaten angesteuert werden oder als dediziertes Gerät z.B. als Maus.

5 Spezielle OSGi Services

Neben den Services, die individuell für ein Gateway entwickelt und darauf installiert werden können, gibt es eine Menge spezieller Systemservices, die der OSGi-Standard exakt festlegt. Im folgenden sollen einige dieser Services kurz benannt und vorgestellt werden.

5.1 Package Administrations Service

Dieser Service organisiert die Abhängigkeiten zwischen installierten Paketen. Verwenden Services eines Paketes Ressourcen eines anderen Paketes, untersucht dieser Service nach der Deinstallation eines Paketes, ob auch die zu diesem in Abhängigkeit stehenden Pakete deaktiviert und aus dem System entfernt werden können.

5.2 Permission Admin Service

Jedes installierte Bundle kann eine unterschiedliche Anzahl von Zugriffsrechten besitzen, wie z.B. den Zugriff auf eine Datei in einem FLASH-Speicherbereich.. Die Aufgabe dieses Service ist auf Anfrage die Überprüfung dieser Zugriffsberechtigungen vorzunehmen und das resultierende Ergebnis zurückzugeben.

5.3 Service Tracker

Durch die Flexibilität des Frameworks, zu jedem beliebigen Zeitpunkt Bundles installieren, aktivieren, deaktivieren oder deinstallieren zu können, entstehen unter ungünstigen Umständen durch mögliche Abhängigkeiten zwischen den einzelnen Paketen schwer beobachtbare Ausnahmesituationen. Dies gefährdet den stabilen Betrieb des Gesamtsystems. Der Service Tracker stellt Funktionen zur Verfügung, die eine Registrierung von Services beobachtet und damit im Fehlerfall die Rekonstruktion der Registrierungsreihenfolge ermöglicht.

5.4 LOG Service

Der LOG Service stellt allen Services die Möglichkeit zur Verfügung, beliebige Informationen während des Betriebs in lesbarer Form als Nachricht abzuspeichern oder zu einem späteren Zeitpunkt wieder auszulesen.

5.5 HTTP Service

Die OSGi Umgebung stellt den Benutzern normalerweise den Zugriff auf Services im Internet oder anderen Netzwerken zur Verfügung. Dies ermöglicht mit WEB-Browsern Informationen von Services zu erhalten bzw. zu Services zu senden. Eine Realisierung erfolgt durch einen HTTP-Server, der eine serverseitige Ausführung von Java-Servlets unterstützt. Weiterhin können Ressourcen wie HTML-Dateien, Bilder registriert und unter der entsprechenden URL-Adresse verfügbar gemacht werden.

5.6 Device Access - Automatische Erkennung neuer Geräte im Netz

Der Device Access Manager erkennt neue Geräte im Netz und kann für diese dann die erfordeliche Treiber bereitstellen. [1]

5.7 Preferences - Dienst für Benutzereinstellungen

Viele Bundles benötigen eine Möglichkeit, Daten dauerhaft zu speichern. D.h. die Daten müssen ein Stoppen und Starten des Bundle, des Framework und der OSGi-Umgebung überleben. Benutzerspezifische Einstellungen sowie systemspezifische Daten werden auf diese Art dem Bundle zur Verfügung gestellt. [1]

[1] OSGi Service-Platform Release 2

6 Implementierung

Der Betrieb einer OSGi-Umgebung kann auf unterschiedlichen Hardwareplattformen erfolgen. Zum Beispiel bietet sich eine PC/104-Umgebung unter Embedded Linux mit einer geeigneten Java Virtual Machine an. Für viele Bereiche ist aber der Einsatz einer Embedded PC-Lösung zu aufwendig. Speziell im medizinischen Sektor müssen die Kosten in einen sehr engen Rahmen gehalten werden. Dies ist nur durch die Verwendung einer kostenoptimierter Lösung möglich.

Neben dem Einsatz einer geeigneten Prozessorarchitektur spielt insbesondere die Wahl der Schnittstellen und des richtigen Speichers eine entscheidende Rolle. Als Schnittstellen gewährleistet eine größere Anzahl von RS232 Schnittstellen, eine Infrarot-Schnittstelle, ein USB-Master aber auch aktive Ein- und Ausgänge wie digitale IO, Analog Digital und Digital Analog Umsetzer den flexiblen Einsatz des Gerätes für die verschiedensten Services. Die Kommunikation über das Internet sollte entweder über Ethernet oder Wählverbindung mit Modem möglich sein.

Der notwendige Speicher hängt sehr stark von der Anzahl der Services und der individuellen Implementation des OSGi-Standards ab. Aber bereits mit 2 MByte FLASH-Speicher und 1 MByte Datenspeicher kann ein abgespecktes OSGi-Gateway arbeiten

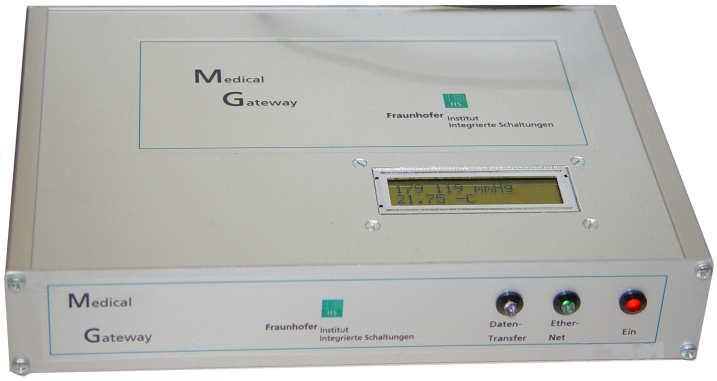


Fig. 3. Prototyp eines Gateway

Auf der Softwareseite muss Java 1.1 oder Java 1.2 unterstützt werden. Die OSGi Spezifikation selbst geht von Java 1.2 aus. Für OSGi existieren inzwischen einige Referenzimplementationen wie z.B. den Java Embedded Server von SUN Microelectronics oder ein Produkt der Firma IBM. Vom Betriebssystem sollte ein Dateisystem unterstützt werden, damit die online ladbaren JAR-Archive abgespeichert werden können. Ein Echtzeitbetriebssystem erleichtert den leistungsfähigen parallelen Betrieb von mehreren Services. Vom Platzbedarf ist zu beachten, dass parallel zu den Archivdateien die Java-Objekte ausgepackt und ausgeführt werden müssen, was den notwendigen Speicherplatz erhöht.

Ein weiterer Speicheraspekt betrifft die benötigten Java-APIs, die im System vorgehalten werden müssen. Folgende APIs sind für die Realisierung der OSGi-Funktionalität notwendig.

- Java.util.jar,
  da die Bundles aus den Java Archiven ausgepackt werden müssen
- java.util.Dictionary, java.util.EventObject, java.util.EventListener
- Java.io für den online Download von Bundles über IO-Streams
- java.lang für die Java Grundtypen einschließlich dem Java-Typ object
- java.security für den Securitymanager
- java.net für den Zugriff auf das Netzwerk über TCP/IP und HTTP.

Neben den Standard APIs können auch plattformspezifische APIs notwendig werden, die den Zugriff auf die Gatewayhardware kapseln. Dies ist insbesondere wichtig für die Erstellung geeigneter Gerätetreiber

7. Applikationen

Ein universelles Service-Gateway im Eigenheim macht nur dann Sinn, wenn auch entsprechende verschiedene Services dafür zur Verfügung gestellt werden können. Nachfolgend werden drei unterschiedliche Internetbasierte Dienstleistungen vorgestellt, die parallel auf einem Gateway arbeiten.

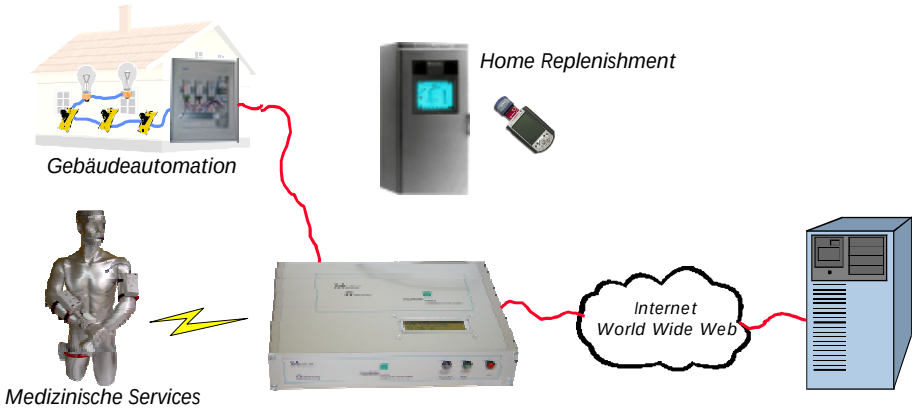


Fig. 4. Verschiedene Services im Betrieb

7.1 Home and Office Replenishment

Heimzulieferdienste können heute einen großen Teil der Zeit einsparen helfen, die für den Einkauf von Standardlebensmitteln benötigt wird. Replenishment stellt einen Spezialfall des Onlineeinkaufs dar. Beim Replenishment gibt der Kunde vor, welche Artikel er in seinem Lager oder Kühlschrank verfügbar haben möchte. Dazu definiert er für jeden Artikel eine Minimal- und eine Maximalmenge. Der Verbrauch wird über einem Scanner von einem Palm oder WindowsCE Gerät erfasst und drahtlos z.B. per DECT oder Bluetooth an das Gateway gesendet. Dort befindet sich in einer lokalen Datenbank der aktuelle Lagerbestand. Der Heimzulieferer überprüft in regelmäßigen Zeitabständen den Lagerbestand und organisiert bei Bedarf die Nachlieferung von Artikeln, bevor diese zur Neige gehen.

7.2 Gebäudeautomation

Die amerikanische Firma Echelon ist sehr aktiv bei der OSGi-Standardisierung. Mit ihrem LonWorks® Bussystem sind sie auch in Deutschland in den verschiedensten Bürogebäuden vertreten und schaffen damit die Grundlage für ein kostensparendes Facility Managment oder Energieeinsparungen durch geeignetes

Energiemanagement. Im Rahmen des OSGi Device Access sind im Netzwerk verteilte digitale oder analoge I/O-Komponenten im Gateway durch Devices repräsentiert. Auf Basis dieser Devices können spezielle Services den aktuellen Zustand von Eingängen abfragen oder nach Bedarf Aktoren individuell schalten. In dem Gateway der Projektgruppe Netzzugangstechnik ist in Form eines Device Treiber die direkte Kommunikation zwischen einem LonWorks® Neuronchip und dem Gateway-Prozessor realisiert. Über diese Kommunikationsbeziehung können Netzwerkvariablen auf dem Neuronchip und damit auf dem Netzwerk verändert oder ausgelesen werden. Über explizite Messages ist der Zugriff auf beliebige Geräte im Netzwerk möglich. Auf Basis dieses Treibers kann z.B. ein HTTP2LonWorks Gateway Service realisiert werden. Er gestattet einen direkten Zugriff über das Internet auf die Installationstechnik eines Gebäudes mit HTML-Formularen oder Java-Applets zu nehmen. Eine objektorientierte Programmierung des LonWorks-Netzes auf dem Gateway in Java unterstützt die schnelle Entwicklung von zuverlässigen und robusten Automationslösungen

7.3 Medizintechnik

Eine Herausforderung des Gesundheitswesen wird es sein, die kostenaufwendigen Aufenthalte von Patienten in den Kliniken weiter zu reduzieren, ohne dass die Qualität der Gesundheitsversorgung darunter weiter leidet. Dazu ist im Bereich der Telemedizin die gleichzeitige Erfassung verschiedener Vitalparameter notwendig. Die meisten Medizingeräte besitzen heute nur eine serielle Schnittstelle aber keinen Netzwerkanschluss, um ihre Daten an einen Rechner zu übermitteln. Ein Gateway kann nun dazu dienen, über die zahlreichen seriellen Schnittstellen die individuellen Diagnosegeräte anzuschließen, die Monitoringdaten abzufragen und entsprechend aufzubereiten. Für jedes angeschlossene Gerät wird ein individueller Treiber geladen, der die Kommunikation managt und so verteilte medizinische Applikationen ermöglicht. Die erfassten und aufbereiteten Daten werden über das Internet mit einer elektronischen Patientenakte abgeglichen, die neben den medizinischen Daten des Patienten in Zukunft auch dessen Vitalparameter aufnehmen kann. Ein spezieller Service im Gateway überprüft das Überschreiten oder Unterschreiten von einstellbaren Schwellwerten und meldet lebensbedrohliche Zustände zuverlässig weiter. Durch die Verfügbarkeit der Daten über das Internet können bei einem Notfall sowohl das nächstgelegene Krankenhaus, der Krankentransportdienst aber auch der zuständige Hausarzt schnell erste Informationen abfragen. Kombiniert mit einem hochintegrierten drahtlosen körpernahen Kommunikationssystem sind die Sensoren für die wichtigsten Vitalparameter ohne störende Beeinträchtigung des Patienten als Pflaster am Körper zu tragen und erfassen so über einen längeren Zeitraum alle notwendigen Informationen.

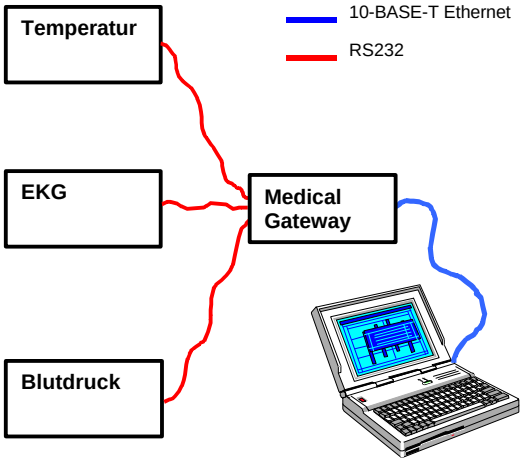


Fig. 5. Medizinisches Gateway als Dienst