Webinterface für eingebettete Systeme in Dual-Server-Architektur

M. Handy, R. Rennert, D. Timmermann

Institut für Angewandte Mikroelektronik und Datentechnik
Universität Rostock, Richard-Wagner-Str. 31, 18119 Rostock
matthias.handy@etechnik.uni-rostock.de

Abstract. Eingebettete Systeme verfügen über begrenzte Ressourcen. Daher ist es schwierig, ergonomische und bedienerfreundliche Benutzerschnittstellen für derartige Systeme zu entwerfen. Bisher wurde versucht, mit den gegebenen Ressourcen auszukommen, was heute zu einer schlechten Benutzbarkeit bei vielen vorhandenen Systemen führt. Die Einführung von Embedded Webservern gestattet ein neues Architekturmodell für den Entwurf von Benutzerschnittstellen. Der hier vorgetragene Lösungsansatz umgeht die Ressourcenbeschränkungen, indem er hardwareunabhängige Teile der Benutzerschnittstelle auf einen Applikationsserver auslagert. Diese sogenannte Dual-Server-Architektur bietet eine flexible, leicht zu wartende Möglichkeit, komplexe Mensch-Maschine-Schnittstellen (MMS) für eingebettete Systeme zu entwerfen.

1 Einführung

Eingebettete Systeme haben den Siegeszug in unsere Alltagswelt angetreten. Eine Vielzahl von konventionellen Geräten in unserer Umgebung ist bereits heute mit derartigen Systemen ausgestattet. Die dadurch ausgelöste „Inflation der Funktionalitäten“ führt nicht zwangsläufig zu einer Erhöhung der Nutzbarkeit dieser Geräte. Gerade die fortschreitende Miniaturisierung erschwert die Aufgabe, ein technisches Gerät mit genügend Affordances – das sind die wahrgenommenen Eigenschaften eines Gerätes – auszustatten [1].

Als entscheidendes Evaluierungskriterium technischer Geräte löst die Bedienerfreundlichkeit den Funktionsumfang bereits heute in vielen Bereichen ab. Die Entwicklung einer Benutzerschnittstelle für eingebettete Systeme wird dann problematisch, wenn Aspekte wie Ergonomie und Bedienerfreundlichkeit in den Vordergrund rücken. Abhilfe dafür wurde durch die Entwicklung von Embedded PCs mit Ethernet Connectivity und darauf integrierten Webservern geschaffen. Der Einsatz von Webservern in eingebetteten Systemen hat zwei entscheidende Vorteile. Zum einen lassen sich derartige Systeme dadurch aus der Ferne steuern und überwachen, zum anderen wird Softwareentwicklern eine einfache und flexible Möglichkeit bereitgestellt, Benutzerschnittstellen zu entwickeln [2],[3]. Diese Interfaces werden in Markup-Sprachen, wie HTML oder XML, programmiert und um gerätespezifische Abschnitte erweitert, die den Zugriff auf die Hardware des eingebetteten Systems ermöglichen. Realisiert werden derartige Zugriffe beispielsweise durch Java-Applets oder CGI-Programme [4].

Der Einsatz von Webservern löst aber keinesfalls die Ressourcenbeschränkungen auf, denen eingebettete Systeme in Bezug auf Speichergröße und Rechengeschwindigkeit unterliegen. Zwar sind eingebettete Webserver in der Lage, zu jedem HTTP-Client (z.B. einem Browser) eine Verbindung herzustellen, jedoch können sie nur Dienste anbieten, deren Implementierung den Ressourcenbeschränkungen genügt. Daran ändert auch der Einsatz von Embedded Webservern wenig. Lediglich die Kommunikation mit dem eingebetteten System folgt nun dem Client-Server-Paradigma [5]. Allerdings können vorhandene Ressourcenbeschränkungen eingebetteter Systeme durch ein verteiltes Softwaresystem auf der Basis einer Dual-Server-Architektur umgangen werden. Die Idee dabei ist, nicht das komplette Webinterface auf dem Embedded Webserver zu speichern, sondern

hardwareunabhängige Programmteile auf einen zweiten Webserver (Applikationsserver) auszulagern, der beispielsweise bei einem Internet Service Provider (ISP) gehostet werden kann.

Dieser Beitrag wird zunächst das vorgeschlagene Architekturmodell allgemein erläutern und einsetzbare Komponenten vorschlagen. Anschließend wird ein Prototyp vorgestellt, der dieses Modell umsetzt. Dabei werden die verwendeten Hard- und Softwaremodule beschrieben sowie deren Schnittstellen erklärt.

2 Das Modell der Dual-Server-Architektur

Hauptzweck der Anordnung ist die Steuerung bzw. Regelung von Geräten, die über vorhandene Hardwareschnittstellen an die Embedded PCs angeschlossen sind. Anwendungsbereiche liegen beispielsweise in der Gebäude- oder Prozessautomatisierung. Das hier vorgestellte Modell besteht aus drei Komponenten:

Als Mensch-Maschine-Schnittstelle fungiert ein *Web-Browser*, der auf einem beliebigen PC laufen kann. Der eingesetzte Browser muss gängige Standards umsetzen, soll aber zur Darstellung des Webinterface keine Plug-Ins benötigen, um eine größtmögliche Kompatibilität zu erreichen. Die Umsetzung der Benutzerschnittstelle als Webinterface ermöglicht eine Steuerung der Embedded PCs von nahezu jeder Plattform aus. Das Webinterface kann auf jedem Betriebssystem laufen, für das HTML-konforme Browser angeboten werden.

Die zweite Komponente ist ein Webserver auf einem Embedded PC. Dabei bietet es sich an, die Möglichkeit zu berücksichtigen, mehrere Embedded PCs über eine zentrale Benutzerschnittstelle bedienbar zu machen. Deshalb kann die zweite Komponente auch als Aggregat mehrerer *Embedded Webserver* betrachtet werden.

Die Benutzerschnittstelle kann bei diesem Modell keine direkte Verbindung mit den Webservern auf den Embedded PCs herstellen, ohne vorher die dritte Komponente des Systems kontaktiert zu haben. Diese dritte Komponente ist ein leistungsfähiger *Applikationsserver*, der in der Lage ist, dynamische Webseiten zu erzeugen und dafür Verbindungen zu verschiedenen Datenbanken aufbauen kann. Dieser Applikationsserver arbeitet im Hintergrund und stellt der Benutzerschnittstelle diejenigen Daten zur Verfügung, die nicht direkt zur Steuerung von Geräten benötigt werden. Dazu gehören zum Beispiel die genaue Bezeichnung einzelner Geräte, eine leistungsfähige Hilfefunktion, die Benutzer des Systems bei ihren Bedienhandlungen unterstützt, oder eine Undo-Funktion. Abbildung 1 zeigt das beschriebene Architekturmodell:

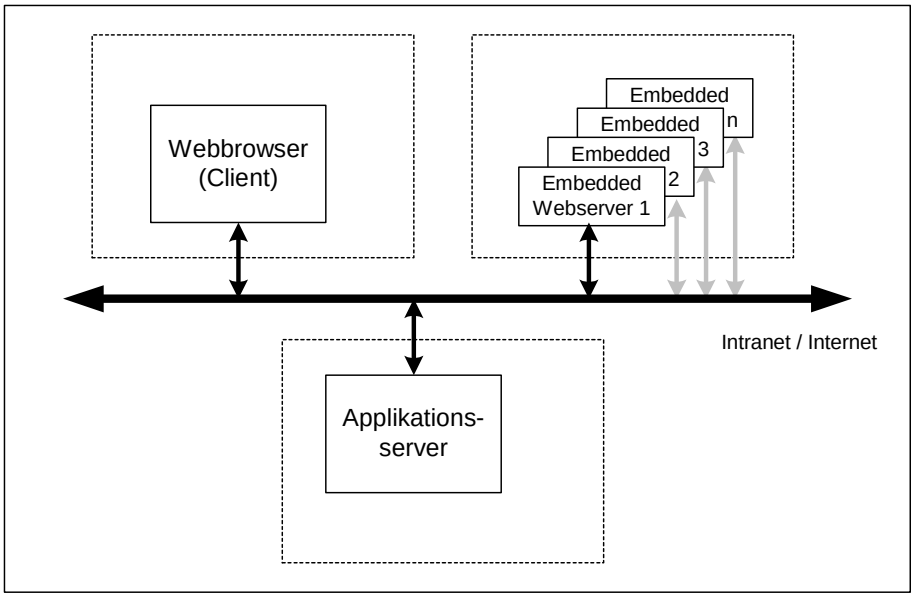

Abb. 1. Modell der Dual-Server-Architektur

Der Webbrowser übernimmt dabei die Rolle des Vermittlers und führt die Daten des Applikationsservers sowie die Prozessinformationen der Embedded PCs zusammen. Es spielt hier keine Rolle, in welcher Form die Daten auf dem Applikationsserver (bzw. in den angeschlossenen Datenbanken) oder den Embedded Webservern vorliegen. Der Datenstrom muss lediglich für die Darstellung im Browser aufbereitet werden, beispielsweise durch Transformation in HTML-Code oder durch Integration in Java-Applets.

3 Prototyp des Architekturmodells

3.1 Der Embedded Webserver

Als Embedded Server wird der Gipsy-Server eines Gesytec DIMM PC EC verwendet [6]. Die Netzwerkverbindung wird über einen Standard-Ethernet-Anschluss hergestellt. Des weiteren bietet der Embedded PC Anschlussmöglichkeiten für die serielle und parallele Schnittstelle, Keyboard und Speaker. Herz des DIMM PC EC ist ein mit 66 MHz getakteter 486er-Prozessor von AMD. Als Speicher stehen dem DIMM PC EC ein 4 MB großer Flash-Programmspeicher und ein 8 MB großer DRAM zur Verfügung.

Der Gipsy Server allein kann aber noch keine Verbindung zu der DIMM PC EC Hardware herstellen. Dafür wird außerdem der Gipsy 2000 Connection Server (G2CS.exe) benötigt, der ein einfaches Verfahren bietet, Webbrowser mit den Prozessinformationen des DIMM PC EC zu versorgen. G2CS.exe ermöglicht den Zugriff auf so genannte Datenpunkte (datapoints, DPs), die von Entwicklern für spezielle Anwendungen programmiert werden können. Dafür kommuniziert g2cs.exe mit einer Datei namens DP.DLL, in der alle Datenpunkte und deren Zugriffsmethoden gespeichert sind. Jeder Datenpunkt benötigt eine Input- und / oder eine Outputmethode. Ein beliebiger Browser, der via Internet eine bestimmte HTML-Seite vom Gipsy-Server anfordert kann eine Verbindung zum Connection Server aufbauen und gibt damit dem Nutzer die Möglichkeit, Datenpunkte abzufragen und zu verändern. Der Datenaustausch zwischen Browser und Gipsy-Server läuft hier über Java-Applets.

3.2 Der Applikationsserver

Die Anforderungen an den Applikationsserver gehen weit über das Bereitstellen von HTML-Dokumenten und darin eingebetteten Grafikdateien hinaus. Da dieser als leistungsfähige Unterstützung für den ressourcen- und funktionsarmen Embedded Webserver arbeiten soll, sind die Anforderungen an den Applikationsserver umso höher. Wichtigstes Kriterium ist dabei die Möglichkeit, dynamische Webseiten auf der Serverseite zu erzeugen. Außerdem sollte der Applikationsserver möglichst robust und sicher sein.

Der Prototyp nutzt für diese Aufgabe das Open-Source-Produkt Apache-Tomcat [7]. Tomcat ist ein kleiner kostenloser Webserver, der die offizielle Referenzimplementierung der Spezifikationen der Servlet 2.2-API und der JSP 1.1-API[1] darstellt [8]. Dabei kann Tomcat eigenständig als JSP- und HTTP-Server laufen oder als Erweiterung des Apache-Webserver installiert werden. Bei der Installation als Apache-Modul übernimmt Tomcat dann lediglich HTTP-Anforderungen, die dynamische JSP-Seiten liefern sollen. Apache bearbeitet dafür rein statische http-requests.

Ein großer Vorteil der JSP- und Servlet-Technologie ist die leichte Portierbarkeit von Webapplikationen auf andere Java-fähige Webserver. So kann eine Anwendung, die für Tomcat entwickelt wurde, auf einem IBM Websphere- oder einen BEA Weblogic-Server laufen, ohne den Code auch nur im geringsten ändern zu müssen.

[1] JSP: JavaServer Pages – Möglichkeit, statisches HTML mit dynamisch generierten Inhalten von Java-Servlets zu mischen.

3.3 Funktionsweise

Fordert ein beliebiger Browser die Startseite des Interfaces an, wird diese vom Applikationsserver geladen. Der HTML-Code, den der Applikationsserver an den Browser liefert, enthält mehrere Java-Applet-Aufrufe. Diese Applets werden jedoch nicht vom Applikationsserver, sondern vom Embedded Webserver geladen. Dadurch ergibt sich eine Architektur, in der der Client Verbindungen zu (mindestens) zwei Servern unterhält. Realisiert wird diese Technik durch das Attribut `codebase` im Applet-Aufruf der HTML-Seite (vergl. Listing 1). `Codebase` weist den Browser an, das im Attribut `code` angegebene Java-Applet von der angegebenen URL zu laden.

Ein sogenanntes Verbindungs-Applet, erzeugt, überwacht und beendet Verbindungen zwischen den Datenpunkten des Embedded PCs und dem Client, auf dem das Bedieninterface läuft. Das Verbindungs-Applet steuert aber nicht die an den DIMM PC angeschlossene Hardware. Diese Aufgabe übernehmen Steuerungs-Applets. Für jeden Datenpunkt muss ein Steuerungsapplet vom Embedded Webserver geladen werden. Abbildung 2 zeigt den Aufbau des Prototyps.

```
<applet code="c_applet" name="c<%=i%>" width="20"
height="20" align="absmiddle"
codebase="http://139.30.201.100">
    <param name="nv_id" value="<%= nv_id %>">
    <param name="img0" value="pics/bulb1.gif">
    <param name="img4" value="pics/bulb3.gif">
    <% if ( nv_id.startsWith("O") ) { %>
        <param name="option" value="send">
    <% }  %>
</applet>
```

Listing 1. Aufruf der Verbindungs-Applets im Webinterface

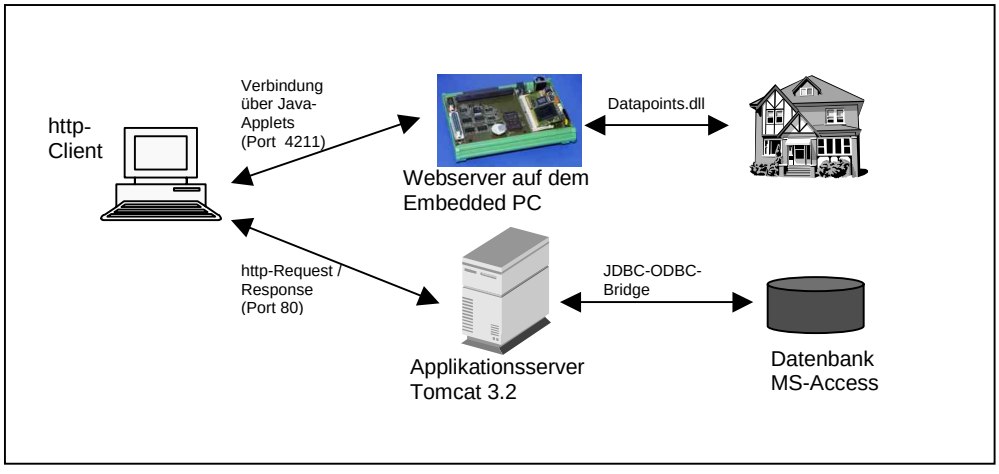

Abbildung 2. Aufbau des Prototyps der Dual-Server-Architektur

Der Prototyp speichert darstellungsbezogene Zusatzinformationen in einer MS-Access-Datenbank. Dafür kommuniziert der Applikationsserver mit der Datenbank über eine JDBC-ODBC-Bridge, die es Java-Anwendungen ermöglicht auf jede ODBC-Datenquelle zuzugreifen. Da jedoch Tomcat mit

der JSP-Technologie in der Lage ist über die JDBC-API[2] verschiedenste Datenbanken anzusprechen, können auch andere Datenbanksysteme verwendet werden.

Zu beachten ist weiterhin, dass Steuerungs- und Statusinformationen zwischen Browser und Embedded PC nicht über den Standard-HTTP-Port ausgetauscht werden sondern einen eigenen Port nutzen. Als Beispielanwendung läuft ein dynamisch generierbares Webinterface für die Gebäudeautomatisierung auf dem Prototyp. Damit können Anwender verteilte Sensoren und Aktoren über eine gemeinsame Benutzerschnittstelle bedienen (vergl. Abb. 3).

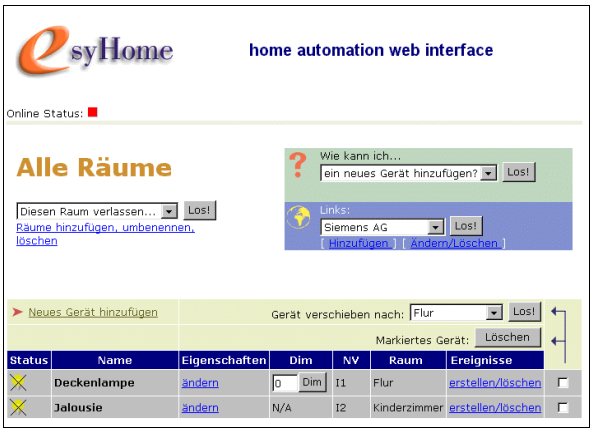

Abb. 3. *E-syHome* – Beispielanwendung der Dual-Server-Architektur

3.4 Der Informationsfluss

Betrachtet man das Modell nicht von der Architektur- sondern von der Datenflussseite, so kann man zwei separate Datenströme aus unterschiedlichen Quellen (den beiden Servern) beobachten. Diese Datenströme sind zunächst plattformabhängig und werden durch bestimmte Mechanismen als plattformunabhängige Informationen für den Browser aufbereitet.

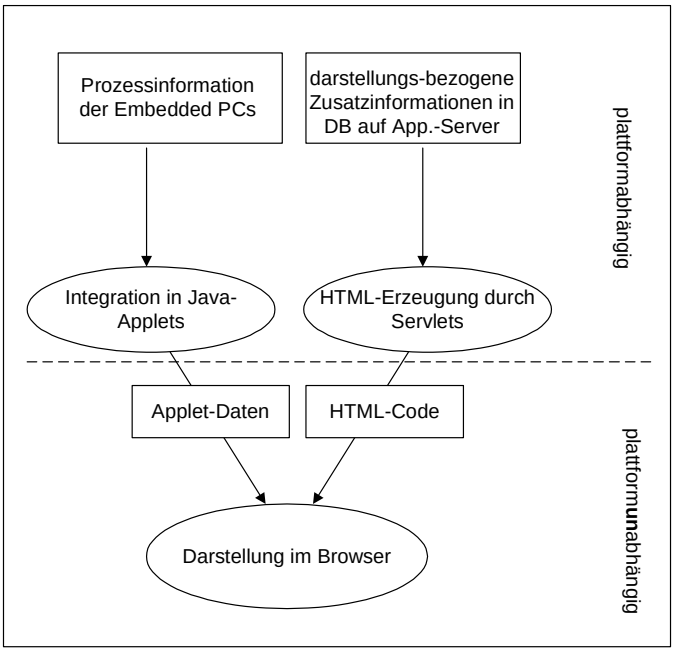

Abb. 4. Informationsfluss des Prototyps

[2] JDBC: Java DataBase Connectivity – Programmierschnittstelle für Datenbankapplikationen in Java

Auf der Embedded-PC-Seite geschieht die Transformation der plattformabhängigen Prozessinformationen der Datenpunkte durch Integration der Daten in Java-Applets. Der Applikationsserver wandelt darstellungsbezogene Zusatzinformationen, die in einer Datenbank gehalten werden, in HTML-Code um. Diese Umwandlung übernehmen Servlets und JavaBeans. JavaBeans verwalten die verschiedenen Verbindungen zur angeschlossenen Datenbank und führen SQL-Abfragen aus. Servlets bereiten die Abfragedaten auf und betten diese für die Darstellung im Browser in HTML-Code ein.

4 Vorteile der Dual-Server-Architektur

Eine Dual-Server-Architektur für Benutzerschnittstellen eingebetteter Systeme besitzt gegenüber der Single-Server-Lösung, bei der sämtliche Prozess- und Darstellungsinformationen auf dem Embedded Webserver gehalten werden, einige entscheidende Vorteile. Mit der Dual-Server-Architektur ist es möglich, mehrere Embedded PCs sowie daran angeschlossene Geräte über eine einheitliche Benutzerschnittstelle zentral bedienbar zu machen. Wird dem Gesamtsystem ein weiterer Embedded PC hinzugefügt, muss für diesen kein neues Interface entwickelt werden. Die bereits bestehende Benutzerschnittstelle wird lediglich um eine neue Komponente erweitert.

Ein weiterer Vorteil dieser Architektur ist die Möglichkeit einer bedienerfreundlichen und ergonomischen Gestaltung der Benutzerschnittstellen für Embedded PCs. Für die Entwicklung stehen sämtliche programmiertechnische Möglichkeiten des Applikationsservers zur Verfügung. Gleichzeitig vereinfacht sich die Wartung des Webinterface durch die Verfügbarkeit standardisierter Softwareschnittstellen des Applikationsservers.

Entscheidend ist außerdem die gute Portierbarkeit einer Benutzerschnittstelle in Dual-Server-Architektur. Es ist damit nicht mehr nötig, das komplette Webinterface auf ein anderes eingebettetes System zu übertragen und anzupassen, es müssen lediglich die Programmteile der Anwendung geändert werden, die Referenzen zum Embedded Webserver enthalten. Bei dem genannten Prototyp ist dies beispielsweise durch eine einfache Änderung des `codebase`-Attributs im Java-Applet-Aufruf möglich.

5 Zusammenfassung und Ausblick

Dieser Beitrag hat gezeigt, dass sich Bedienerfreundlichkeit und Komplexität einer Benutzerschnittstelle selbst bei begrenzten Ressourcen nicht ausschließen müssen. Die Weiterentwicklung der Speichertechnologien wird gewiss dazu führen, dass eingebettete Systeme mittelfristig über Ressourcen verfügen, die heute beispielsweise auf Desktop-PCs zur Verfügung stehen. Trotzdem wird es immer einen erheblichen Ressourcen-Unterschied zwischen eingebetteten und Standard-Desktop-Systemen geben. Das hier vorgestellte Modell der Dual-Server-Architektur zeigt, wie Ressourcenbeschränkungen bei Embedded PCs umgangen werden können. So kann der bereits erwähnte Nachteil der fortschreitenden Miniaturisierung ausgeglichen werden, indem reale Affordances, wie Schalter oder Regler, durch virtuelle Affordances ersetzt werden.

Referenzen

1. Gibson, J. J.: The ecological approach to visual perception, Houghton Mifflin, New York (1979)
2. Henderson, N.: Webserver in Embedded Applications, in: ESE Magazine, Nr. 5/2001, URL: http://www.esemagazine.co.uk
3. Jahnke, J.: Component Based Engineering of Distributed Embedded Systems, Embedded Systems Conference, San Francisco (2001)
4. Thomas, T.: Internet-Accessible Home Appliances – How to Make eToast, Embedded Systems Conference, Boston (2000)
5. Comer, D.: Computernetzwerke und Internets, Prentice Hall, München, London, New York (1998)
6. Gesytec: Embedded PC – Developer's Documentation, Aachen (1999)
7. The Jakarta Project – Jakarta Tomcat, URL: http://jakarta.apache.org/tomcat (2001)
8. Hall, M.: Core Servlets und JavaServer Pages, Markt+Technik Verlag, München (2001)