

ASP 5 Projekt



von

Renate Eder
Wolfgang Hafenscher
Gerald Madlmayr
Daniela Zimmermann

Coach: DI Robert Kolmhofer

Projektbericht zu ASP5
am Fachhochschul-Studiengang
„Medientechnik und –design“
in Hagenberg

im WS 2002/03

1 Inhaltsverzeichnis

1	Inhaltsverzeichnis	2
2	Projektbeschreibung	4
3	Projektspezifikation	5
3.1	Schnittstellendefinition	6
3.1.1	Schnittstelle Server / InstaBus-Handling	6
3.1.2	Schnittstelle Client / Server	6
3.1.3	Befehlsstring – Definition	6
3.1.4	XML-Strukturerweiterung	7
3.1.5	Grafische Darstellung der Baumstruktur	7
4	Umsetzung EIB – goes mobile	8
4.1	GUI – EIB goes mobile Client Applikation	8
4.1.1	Was ist nun J2ME?	8
4.1.2	Testumgebung	9
4.1.3	Struktur der Applikation.....	9
4.1.4	Funktionalität der Client Applikation	10
4.1.5	Übertragung des Clients auf das mobile Endgerät.....	13
4.2	SocketConnection - Datenübertragung zwischen Client und Server	14
4.3	EIB – goes Mobile Serverstruktur	15
4.3.1	ETS2 Software	15
4.3.2	VisEditor	15
4.3.3	Eiblet Engine Trialkit	15

4.4	EIB-Goes-Mobile Server Framework.....	16
4.5	Visualisierung von ETS2, EIB, Falcon, Eiblet Engine und Server	16
4.6	Installationsanleitung für den Server:	17
5	Geräteinformationen	18
5.1	Benötigte Informationen am Server für Steuerung der Geräte am EIB	18
5.2	Benötigte Informationen am Client	19
5.3	Benötigte Informationen für Modis	20
6	Software	22
7	Hardware.....	22

2 Projektbeschreibung

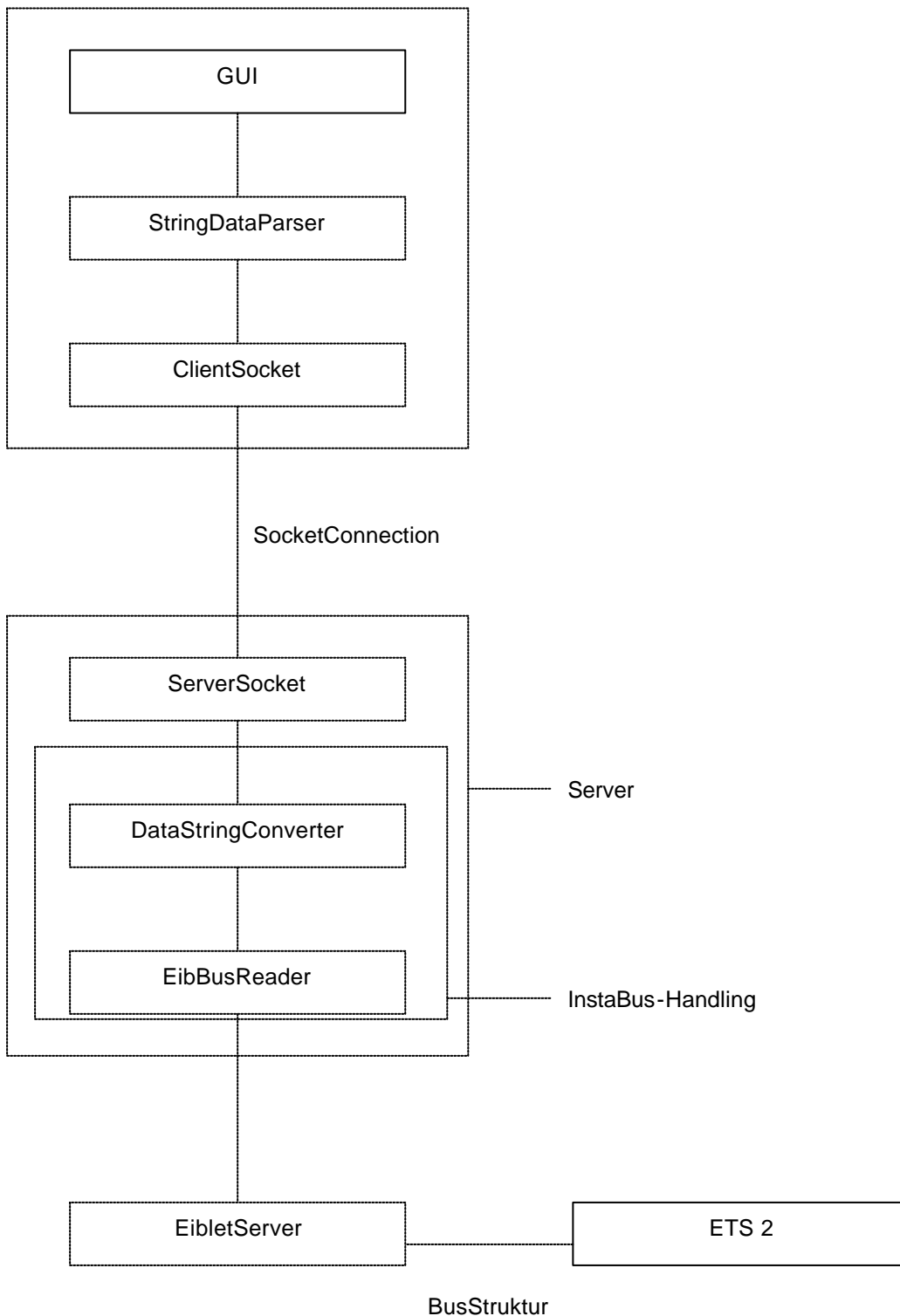
Das ASP5 Projekt „EIB goes mobile“ beschäftigt sich mit der Ansteuerung eines EIB-Systems (EIB = european instabus installation) über ein mobiles Endgerät. Mit dieser Applikation soll die Verwaltung von instabusgesteuerten Geräten (z.B. Lichtschalter) mittels mobilen Endgeräten ermöglicht werden.

Ziel dieses ASP5 Projekts ist die Entwicklung einer Applikation unter Java Micro Edition (J2ME). Diese Applikation soll mit einem Emulator getestet werden können. Auf diesem MicroEmulator soll es möglich sein, die vorhandene Instabus-Installation und die dort integrierten Geräten anzusteuern. Dies sind primär die Schritte und Aufgaben der ersten Phase unseres Projekts.

Anschließend soll in der zweiten Phase die Umsetzung für ein mobiles Eingabegerät erfolgen. Als Eingabegerät verwenden wir in unserem Fall ein MobileJava fähiges Handy.

3 Projektspezifikation

Nach grundsätzlichen Überlegungen und Diskussionen sowie einigen Abänderungen sind die grundlegende Struktur unseres Projekts sowie die Spezifikation unserer Schnittstellen entstanden.



3.1 Schnittstellendefinition

3.1.1 Schnittstelle Server / InstaBus-Handling

Der Server besitzt eine Instanz InstaBus und kommuniziert über die Schnittstelle execute(). Es wird ein Befehlsstring übergeben und die Methode liefert einen String als Rückgabewert.

3.1.2 Schnittstelle Client / Server

Zwischen Client und Server werden nur Strings geparsed. Der Client schickt einen Befehlsstring und erhält entweder die komplette XML-Baumstruktur (beim Login) oder nur die upgedateten kritischen Devices.

3.1.3 Befehlsstring – Definition

Der Befehl baut sich aus folgenden Teilstrings auf (genauere Angaben siehe Kapitel 5 „Geräteinformationen“):

user:password:command:object:value*

Die einzelnen Elemente werden durch ":" getrennt und der komplette String endet mit "*". Die Elemente "status" und "value" sind optional.

user: Username des Clients
password: Passwort des Clients
command: get oder set
value: "on" / "off" / percent / "up" / "down"

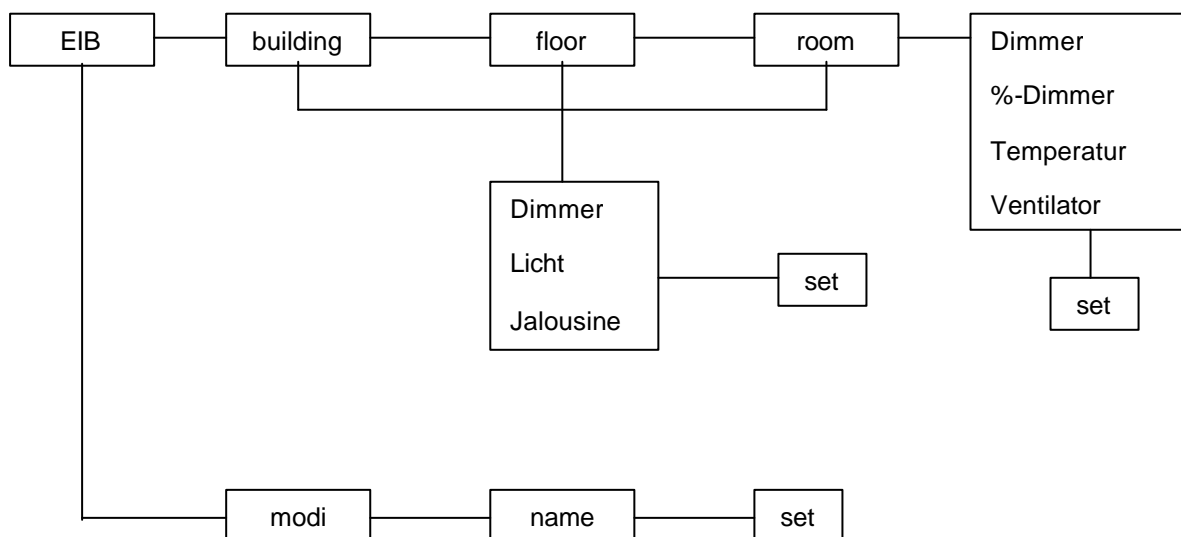
Command:	Rückgabe:
user:password:set:id:value	XML-Struktur mit Devices

3.1.4 XML-Strukturerweiterung

Zusätzlich zur festgelegten XML-Baumstruktur gibt es für die Möglichkeit unabhängig von Gebäude / Stockwerk / Raum Modi zu setzen (z.B. alle angeschlossenen Geräte werden auf den Nachmodus gesetzt). Bei einem Modus werden für unterschiedliche Geräte unterschiedliche Zustände definiert. Diese können einfach über einen einzelnen Befehl aktiviert werden und alle Geräte werden auf einmal gesetzt. Ein Modus kann nur aktiviert (,on') werden aber nicht abgestellt werden, da die Ausgangseinstellungen der Geräte nicht gespeichert werden.

3.1.5 Grafische Darstellung der Baumstruktur

Dieses Kapitel beinhaltet die grafische Darstellung der Baumstruktur. Diese Baumstruktur ist sowohl für den Client als auch für den Server wichtig. Bezogen auf diese Baumstruktur setzen die Geräte die Informationen zusammen (commands), die sie erhalten oder setzen möchten (auch das Setzen der Modi wird hier berücksichtigt). Wie die Information zusammengesetzt wird, steht im Kapitel 5 „Geräteinformationen“ noch genauer beschrieben.



4 Umsetzung EIB – goes mobile

4.1 GUI – EIB goes mobile Client Applikation

Die Programmierung des GUI für den Client erfolgte mittels Java 2 Micro Edition (J2ME). Dafür ist von Java bereits ein Emulator verfügbar (J2ME Wireless Toolkit 1.0.4_01). Damit war für uns ein relativ einfacher Testbetrieb möglich. Ein weiterer Vorteil dieses Emulators ist, dass er herstellerunabhängig funktioniert (z.B. im Gegensatz zum Siemens C55 Emulator).

4.1.1 Was ist nun J2ME?

Hinter J2ME (auch „mobile Java“ genannt) verbirgt sich der kleine Bruder von J2EE (Java 2 Enterprise Edition) bzw. J2SE (Java 2 Standard Edition).

Diese Version von Java 2 ist speziell für Endgeräte mit geringen Speichermöglichkeiten gedacht und stellt nur eine exquisite Auswahl an Basisklassen zur Verfügung. Das heißt, dass J2ME die spezielle Eigenschaften und Leistungswerte mobiler Endgeräte kompensiert.

Die Methoden und die Strukturen, wie man mobile Java Anwendungen implementiert, unterscheiden sich zu einem gewissen Grad von den herkömmlichen Standard-Java-Anwendungen. Java 2 Micro Edition (J2ME) ermöglicht jedoch Java-Entwicklern relativ einfach die Implementierung von Applikationen für mobile Geräte.

Alle J2ME Technologien basieren auf Konfigurationen und Profilen. Eine Konfiguration definiert die Mindestanforderung an Klassenbibliotheken die einer Gruppe von Geräten zur Verfügung stehen. Ein Profil definiert die APIs die einer bestimmten Familie von Geräten zur Verfügung stehen. Das Profil eines Mobiltelefons ist zum Beispiel unterschiedlich zu einem Profil für einen PDA, jedoch beide Geräte nutzen die gleiche Konfiguration.

Das MIDP (Mobile Information Device Profile) stellt Java APIs unter der Verwendung der CLDC (Connected Limited Device Configuration - der Konfiguration für kleine, mobile Endgeräte - wie zum Beispiel Mobiltelefone) zur Verfügung.

Ähnlich zu Applets definiert das MIDP so genannte MIDlets. Diese werden von der MIDP Implementierung verwaltet und bilden den Einstiegspunkt für die Programmierung. Im Unterschied zu "normalen" Java Applikationen ist es nicht möglich, ein Programm mit einer main Routine zu starten. Die Applikation wird vielmehr vom MIDP Framework verwaltet.

Beim Start eines MIDlets wird vom Framework die Methode `startApp()` aufgerufen, in der die Initialisierung der Applikation durchgeführt wird. Das Framework kann ein MIDlet durch Aufruf von `pauseApp()` temporär stoppen. Schließlich kann das MIDlet durch den Aufruf der `destroyApp()` Methode vom Framework zerstört werden.

4.1.2 Testumgebung

Für das Testen unserer Desktop Applikation verwendeten wir J2ME Wireless Toolkit (Version 1.0.4_01 for Windows). Mit Hilfe dieses frei verfügbaren Emulators war eine reibungslose Entwicklung des GUI gegeben.

Nach Fertigstellung der Client Applikation testeten wir unser GUI auch auf firmenspezifischen Emulatoren (Siemens M50 Emulator, Siemens C55 Emulator, Siemens S55 Emulator).



J2ME Wireless Toolkit



Siemens C55 Emulator

4.1.3 Struktur der Applikation

Die Struktur der J2ME folgt der grafischen Darstellung der Baustuktur (siehe Kapitel 3.1.5. Grafische Darstellung der Baumstruktur).

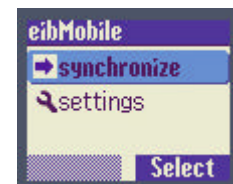
4.1.4 Funktionalität der Client Applikation

Die J2ME Client Applikation wird dynamisch aufgebaut. Dazu wird vom Server ein XML String an den Client geschickt. Dieser baut anschließend eine Instanz von der Klasse Instabus auf. Von den einzelnen Java Klassen werden die angeschlossenen Geräte von der Instanz ausgelesen und grafisch dargestellt.

Ausgehend vom Midlet (EibMobile.java) kommt man über die Copyright Anzeige (alert message) zum Startscreen, wo dem User folgende 2 Möglichkeiten zur Auswahl stehen:

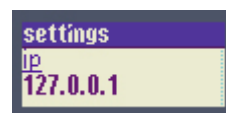
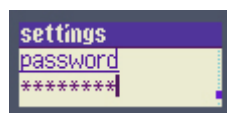
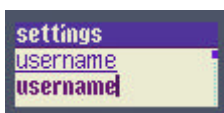


Copyright Anzeige



Startscreen

- (a) Synchronize: Start der Applikation mittels Defaulteinstellungen (bereits am Handy gespeichert).
- (b) Settings: Änderung der Einstellungen bezüglich User, Passwort, Host und Port.



Die Defaulteinstellungen bzw. die modifizierten Daten werden von der Applikation lokal gespeichert. Diese lokale Speicherung von Username, Passwort, IP-Adresse und Port musste von uns implementiert werden.

Das Konzept von J2ME sieht vor, dass Mobile Java keinerlei File-System Operationen erlaubt. Zur permanenten Datensicherung wurde aber das Konzept vom "Record Store" eingeführt. Der "Record Store" ist ein Bereich im Speicher des mobilen Endgerätes welcher von jeder j2me-Applikation benutzt werden kann.

Die Einschränkung - welche natürlich auch erhöhte Sicherheit bedeutet - ist, dass die Applikation nur "Record Sets" welche von ihr selbst generiert wurden auslesen, verändern und löschen kann. Ein solches "Record Set" kann man sich wie ein String Array bzw. einen String Vector vorstellen.

Für unsere Applikation ist nur ein "Record Set" notwendig in welchem die Daten Username, Passwort, IP-Adresse und Port gespeichert werden.

An dieser Stelle sei angemerkt, dass Siemens dem J2ME-Programmierer eine eigene API zur Verfügung stellt, welche weit mehr Funktionen zur Verfügung stellt als es J2ME an sich erlaubt. Beispiele sind der Zugriff auf das File System, Adressbuch, SMS-Speicher, Vibra-Funktion des Akkus, Sound-Funktionen, Beleuchtung erlaubt. Da unser Client dafür gedacht ist auf allen Geräten welche die Spezifikationen MIDP 1.0 und CLDC 1.0 erfüllen zu laufen haben wir diese API von vornherein nicht in Betracht gezogen.

Nachdem User, Passwort, Host und Port feststehen wird nun die eigentliche Applikation zur Steuerung des European Instabus gestartet.



Vom Objekt Instabus werden alle am Instabus angemeldeten Gebäude dynamisch ausgelesen. Zusätzlich findet man ein Modi-Objekt das dem User erlaubt vordefinierte Einstellungen zu setzen. Diese sind gebäudeunabhängig. Am Instabus ist deren Reichweite und Auswirkung definiert (z.B. Nachtmodus dimmt alle Lichter aller Gebäude.) Selbstverständlich kann die Funktionalität eines Modus direkt am Instabus frei eingestellt und jederzeit verändert werden.

Was die Gebäudeliste betrifft kann nun ausgewählt werden, in welchem Gebäude man Aktionen ausführen will.

Durch die Selektion wird eine wiederum dynamische Liste generiert, die alle Etagen des ausgewählten Hauses beinhaltet.



Etagen



Räume

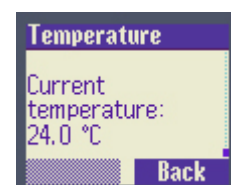


Geräte

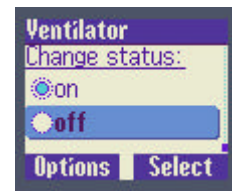
Durch die Auswahl einer Etage erreicht man alle darin liegenden Räume. Für jeden Raum kann der User die sich darin befindenden Geräte, die am Instabus angeschlossen sind, abfragen und deren Zustand verändern. Für die grafische Darstellung der einzelnen Geräte wird in der Klasse DeviceList abhängig vom Typ der einzelnen Geräte die passende Klasse aufgerufen (z.B. wenn es sich um ein Typ 1 Gerät handelt (= Schalter), wird die Klasse Switches aufgerufen usw.).

Funktionalitäten der einzelnen Geräteklassen

Temperature: Die Klasse Temperature ermöglicht die Anzeige der aktuellen Temperatur, die vom Objekt Instabus ausgelesen wird.



Switches: Diese Klasse wird für jede Art von Schalter aufgerufen (z.B. Ventilator, Licht). Damit ist es möglich, das Gerät entweder auf ‚on‘ oder ‚off‘ zu schalten. Zu Beginn wird der aktuelle Status des Schalters angezeigt (d.h. ob z.B. der Ventilator gerate ein- oder ausgeschalten ist).



Dimmer: In der Klasse Dimmer kann das Licht jeweils in Zehnerschritten gedimmt werden. Auch hier wird zu Beginn der aktuelle Status angezeigt.



Globale Einstellungen

Neben den einzelnen Geräten kann man auch globale Einstellungen treffen. So ist es z.B. möglich für ein gesamtes Gebäude die Lichtverhältnisse zu regeln. Das Menu enthält einen Punkt namens Globals, der diese Funktionalität für Gebäude, Stockwerke und Räume steuert.



Der command String

Für das Setzen einer Aktion muss ein command String zusammengesetzt werden. Dazu wird zuerst die Navigation des Benutzers mitgespeichert, in welchem Gebäude, Stockwerk bzw. Raum er sich genau befindet. Nach der endgültigen Auswahl einer Aktion in den Klassen (Ventilator, Dimmer, Temperatur oder Modus) wird der command String zu einem XML String umgewandelt und an den Server geschickt, der diese Information auf den Instabus umsetzt.

4.1.5 Übertragung des Clients auf das mobile Endgerät

Es gibt verschiedene Wege unseren Client auf das mobile Endgerät zu transferieren.

Je nach Eigenheiten des uns zur Verfügung stehenden Endgerätes testeten wir verschiedene Übertragungswege:

Siemens MT50: Um auf das Siemens MT50 per Windows Explorer zugreifen zu können ist die Installation der Siemens Data Exchange Software notwendig. Weiters muss das Gerät per MT50 Java Activator und Noname Datenkabel "behandelt" werden. Der Java Activator verändert Einträge im Speicher des Gerätes so, dass das Gerät per original Siemens Datenkabel an den COM-Port des PCs angeschlossen und via Data Exchange Software angesprochen werden kann. Das Noname-Kabel war deshalb notwendig, da zum Einspielen des Java Activators das Gerät ausgeschaltet sein und das Gerät vom Datenkabel mit 4.5 V Spannung versorgt werden muss. Die Stromversorgung wird von original Datenkabeln nicht unterstützt, sind aber dennoch ca. viermal so teuer wie Noname Datenkabel.

Siemens C55: Da dieses Modell erst während der Durchführung unseres Projektes erschienen ist waren keine Noname Datenkabel am Markt erhältlich um den Java Activator einzuspielen. Da dieses Gerät zusätzlich weder Bluetooth- noch Infrarot-Übertragung unterstützt war die einzige Möglichkeit unseren Client auf das Gerät zu Übertragen eine Übertragung via Internet. Damit dies Fehlerfrei möglich war, war es notwendig, dass am Server die MIME-Typen für jad- und jar-Dateien eingetragen sind.

Nokia 7650: In dieses Gerät wurde der Client ausschließlich per IRDA-Übertragung mittels Nokia PC Data Suite eingespielt.

4.2 SocketConnection - Datenübertragung zwischen Client und Server

Im MIDP 1.0 ist eigentlich nur ein Protokoll, nämlich "http" spezifiziert. Allerdings ist in der Praxis auch "socket" implementiert. Alle von uns verwendeten Emulatoren und Toolkits, sowie die von uns verwendeten Endgeräte beherrschten "socket".

Wir entschieden uns, die Verbindung zwischen Client und Server mittels einer Socket-Connection zu realisieren. Mit einer http-Connection hätten wir die im Protokoll definierten Routinen nachprogrammieren und für unsere Anforderung adaptieren müssen.

Die Socket-Connection war deshalb nicht so aufwendig zu implementieren. Diese Art der Verbindung hat den Vorteil, dass wir praktisch unser eigenes Protokoll erstellten, da die Socket-Connection lediglich eine byte-Stream-Übertragung darstellt welche über eine herkömmliche URL hergestellt wird und keine vordefinierten Routinen spezifiziert. Unsere Applikationen (Server und Client) wandeln den übertragenen Byte-Stream direkt in Characters um und interpretieren die so entstandene Zeichenkette. Der Aufbau einer solchen Befehls-Zeichenkette ist im Kapitel 5 "Geräteinformationen" zu finden.

Erfolgreich getestet wurde die Datenübertragung mit dem Wireless Toolkit von Sun, den C55/M(T)50 Emulatoren von Siemens und am Siemens C55 bzw. Siemens MT50.

Ein Problem ist, dass anscheinend nicht jeder Netzanbieter diese Art der Übertragung unterstützt. Unsere Ressourcen waren auf zwei österreichische Netzanbieter beschränkt und es hat sich herausgestellt, dass Connect Austria (ONE) die Datenübertragung über Socket unterstützt und T-mobile (ehemals Max Mobil) nicht.

4.3 EIB – goes Mobile Serverstruktur

4.3.1 ETS2 Software

Die ETS Software dient der fundamentalen Organisation des Bussystems. Die wichtigsten Eigenschaften, die den Geräten zugewiesen werden sind die GroupAddress und der Typ des Gerätes (Schalter, Scaling-Device etc). Wichtig für unsere Zwecke sind nur die Device Typen, da wir unsere Visualisierung nur aufgrund der physischen Anordnung (Gebäude, Stock, Raum) durchführen. Die logische Busstruktur anhand der GroupAddresses ist für uns nicht relevant. Die Location-Informationen (Gebäude-, Stockwerk- und Raumname), die in ETS2 eingegeben werden, werden aber nicht im Bussystem gespeichert, sondern dienen nur zur Orientierung im System.

4.3.2 VisEditor

Mit Hilfe des VisEditors kann (falls es eine bestehende EIB Installation gibt) eine Visualisierung gestaltet werden. Diese Visualisierung wird vom Trialkit verarbeitet und online präsentiert. Der VisEditor dient zur Gestaltung des Raumes (Interfaces). Hier werden die Location-Informationen erneut definiert (relevant für Anzeige in unserem GUI). Außerdem werden die Piktogramme und Name für die einzelnen Geräte definiert. Die Informationen werden in einem XML-File abgelegt, das nachträglich noch manipuliert werden kann. Wir verändern die Felder „Short-Name“ zur Speicherung der Global-Information und die „Description“ für die Priority Information.

4.3.3 Eiblet Engine Trialkit

Die Eiblet Engine stellt bereits ein Server-Framework für EIB dar. Es kann die Visualisierungsdaten vom VisEditor verarbeiten und auf einer Weboberfläche darstellen, da die Eiblet Engine über ein eigenes Server Modul verfügt, das auch HTML-Code produzieren kann. Die Anzeige erfolgt als Java Applet.

Wir bedienen uns der Eiblet Engine, um die Visualisierungsdaten zu bekommen und die Geräte anzusprechen. Es kann hier bereits gesagt werden, dass das Auslesen der Geräteinformationen, die nur im XML File stehen auch ohne angeschlossenes Bussystem funktionieren. Beim Senden von Commands an die einzelnen Geräte kommt es aber dann zu Fehlermeldungen.

4.4 EIB-Goes-Mobile Server Framework

Unsere Serversoftware greift über entsprechende Parameter auf die Informationen der Eiblet Engine zu und stellt dies als XML Daten für den Client zur Verfügung. Die physische Busstruktur kann aber nicht aus dem Bus ausgelesen werden, sondern wird von der Eiblet Engine als XML File vorbereitet. Diese Informationen werden dann über die Eiblet Engine an den EIB 4 Mobile Server übergeben, der diese Daten mit den Geräte zuständen kombiniert, um jedem Geräte auch einen Zustand zuweisen zu können.

4.5 Visualisierung von ETS2, EIB, Falcon, Eiblet Engine und Server

EIB 4 Mobile Server:

- Lesen der Topologie-, Status- und Typdefinition der Geräte
- Aufbereitung der Daten als XML Datenformat für Mobile GUI
- Bereitstellen der Daten auf Port 2401;
- Verarbeiten eingehender Commands der mobilen Applikation durch Kommunikation mit Eiblet Engine.

Eiblet Engine:

- Server Framework für Bus und Agent API. (Port 5555)
User: jnetsystems / Pwd: Eiblet
- Verarbeitet Visualisierung von VisEditor für Webapplikation in ein Applet.
- Senden Steuercommands an Falcon; Liest Statusinformation von Falcon
- Schaltet sich nach 30 min ab, da Falcon nur Demo ist.

VisEditor:

- Einlesen der Geräte vom Testaufbau über Framework der Eiblet Engine (Server muss laufen)
- Gestaltung einer Oberfläche mit den Geräten, die die Eiblet Engine zu Java Applet verarbeitet.
- Definition der Topologie der Geräte.
- Speichern der Daten in XML Form

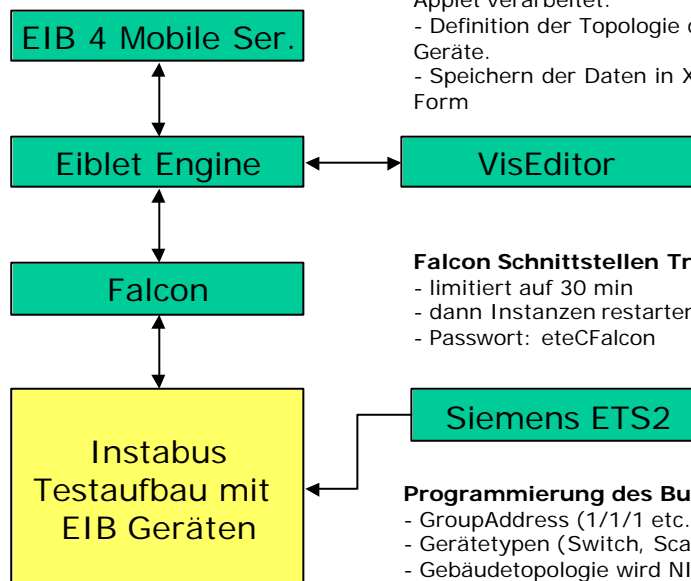
Falcon Schnittstellen Treiber:

- limitiert auf 30 min
- dann Instanzen restarten
- Passwort: eteCFalcon

Siemens ETS2

Programmierung des Busses:

- GroupAddress (1/1/1 etc.)
- Gerätetypen (Switch, Scaling etc)
- Gebäudetopologie wird NICHT übertragen.



4.6 Installationsanleitung für den Server:

- Falcon Treiber (FalconDev.exe) von der CD Installieren (Installationskennwort: eteCFalcon)
- NativeFalconImplDemo.dll ins jdk\bin Verzeichnis kopieren
- eibletutils.jar, trialeibletengine.jar (Eiblet Engine), nfbc.jar (Falcon) ins JRE Verzeichnis des JDK kopieren.
- Eiblet Trial Engine starten. (Achtung – application schließt sich nach 30 min wieder selbständig)
- im Webbrowser localhost:5555/ abfragen und die Test-Installation anklicken (user: eiblet – pwd: jnetsystems)
- wenn das geht die Serversoftware starten (maximal eine Instanz) läuft auf Port 2811 und verarbeitet die Befehle die reinkommen.

5 Geräteinformationen

Jeder Block unserer Spezifikation benötigt gewisse Geräteinformationen. Welche Informationen benötigt werden und in welcher Form diese Informationen aufbereitet werden müssen, wird in den folgenden Unterpunkten erklärt:

5.1 Benötigte Informationen am Server für Steuerung der Geräte am EIB

Folgende Informationen werden am EibletServer benötigt, damit die Geräte am Instabus gesteuert werden können.

Building	Floor	Room	Device	Name	Type	Status	Readonly	Area	Line
1	0	0	0	Fh	0	0	True	0	0
1	1	0	0	Eg	0	0	True	0	0
1	1	1	0	DiplLab	0	0	True	0	0
1	1	1	1	Dimmer	6	0.0	False	1	1
1	1	1	2	Temperatur	5	54.72	True	1	1
1	1	1	3	Ventilator	1	False	False	1	1
1	1	1	4	RS232	1	False	True	1	1
1	1	1	5	schalten_B	1	False	True	1	1

Device	URI	Priority	Global
0	NULL	0	0
0	NULL	0	0
0	NULL	0	0
2	eiblet://ee/LAYER_EIB_GROUP_OBJECT/testBus/Dimmer	2	3
1	eiblet://ee/LAYER_EIB_GROUP_OBJECT/testBus/Temperatur	0	0
3	eiblet://ee/LAYER_EIB_GROUP_OBJECT/testBus/Ventilator	0	4
4	eiblet://ee/LAYER_EIB_GROUP_OBJECT/testBus/s232	2	0
5	eiblet://ee/LAYER_EIB_GROUP_OBJECT/testBus/schalten_B	2	2

Diese Informationen kommen aus dem XML file im Conf-Verzeichnis der Eiblet Engine.

Priority: = description im XML File

Global: = short_name im XML File

Der RS232 Schnittstelle wird nicht das Visualisierungstool eingebunden. Deshalb erhält es keine Location-Informationen und somit wird es auch nicht zum Client übertragen. Alle ‚hidden‘ Devices, die vom Benutzer nicht angesteuert werden sollen/müssen/dürfen, werden einfach nicht im Visualisierungstool eingefügt – somit wird es nicht in das XML für die Eiblet-Engine geschrieben und die Übertragung an den Client entfällt.

5.2 Benötigte Informationen am Client

Auf der Clientseite werden ebenfalls Informationen über die Geräte, die am EIB hängen, benötigt. Aus diesen Informationen wird das GUI aufgebaut.

Building	Floor	Room	Device	Name	Type	Status	Readonly	Priority	Global
1	0	0	0	Fh	0	0	True	0	0
1	1	0	0	Eg	0	0	True	0	0
1	1	1	0	DiplLab	0	0	True	0	0
1	1	1	1	Dimmer	6	0.0	False	2	1
1	1	1	2	Temperatur	5	54.72	True	0	0
1	1	1	3	Ventilator	1	False	False	0	2
1	1	1	5	schalten_B	1	False	True	2	2

Bei Building, Floor, Room, Device und Priority sowie bei Type handelt es sich jeweils um Integer. Der Status und der Name werden als String übergeben. Readonly kennzeichnet Geräte, die nur gelesen werden können.

Das bedeutet nun, dass die **Commands** für das Schalten der Geräte folgendermaßen aussehen:

Command	Aktion
Gerald:kennwort:set:1.1.1.5.off*	Abschalten des Switches „schalter_b“
Wolfgang:passwd:set:1.1.1.1:10.0*	Einstellen des Dimmers auf 10.0%
Daniela:pwd:get:critical:*	Liefert Array mit Informationen über kritische Geräte
Renate:kennung:get:all:*	Liefert Array erneut mit gesamter Information

5.3 Benötigte Informationen für Modi

Hier handelt es sich um notwendige Informationen für Modi am Server und am Client. Ein Modus dient dazu, eine ganze EIB-Installation auf einen gewissen Zustand zu setzen. Kritische Geräten werden dabei auch berücksichtigt. Für die Modi wird jedem Geräte ein Status zugeordnet, welchen es bei welchem Modus einnimmt. Die Organisation dieser Daten muss in einer Datenbank vorgenommen werden, da jedes Gerät für jeden Modus individuell geschaltet werden kann. Die Modi sind für gewisse Bereiche nicht einzeln definierbar, sondern umfassen immer den gesamten Bus.

Modus: betrifft grundsätzlich fixe Auswahl von Geräten des gesamten Busses, die auf einen fest definierten Zustand gesetzt werden. (z.B. Nachmodus).

Building	Floor	Room	Device	Modusname	Type	Status	Readonly	Priority	Global
0	0	0	1	Nachtmodus	0	0	True	0	0
0	0	0	2	Tagmodus	0	0	True	0	0
0	0	0	3	Alle Lichter 100 %	0	0	True	0	0
0	0	0	4	Brandfall	0	0	True	0	0
0	0	0	5	All off	0	0	True	0	0

Hier ist es auch noch wichtig, dass man jedem Gerät einen Modus zuweist:

Gerät/Modus	Nacht	Tag	Alle Lichter 100 %	Brandfall	All off
Dimmer	10.0	0.0	100.0	100.0	0.0
Ventilator	False	True	False	false	False
schalten_B	False	True	False	false	false

Die Commands für das Schalten der Modi sehen dann folgendermaßen aus:

Command	Aktion
Geradl:kennwort:set:0.0.0.1:on*	Aktivieren des Nachtmodus
Wolfgang:passwd:set:0.0.0.2:on*	Aktivieren des Tagmodus

Bei jedem Modus werden nur die Werte für „Device“ gesetzt. Alle anderen Felder für die ID sind immer 0.

Weiters besteht die Möglichkeit bestimmte Locations (Room, Floor, Building) bestimmte ‚Globals‘ zuzuweisen. Hier kann eine Geräteklasse (Global Type) auf einmal geschaltet werden. Es gelten folgende Regeln:

Pro Gerät ist eine Zuordnung zu einer Geräteklasse möglich (= „Short Name“ im XML File). Die Zugeordneten Geräte müssen vom selben Device Typ sein (nur Schalter, nur Dimmer, nur Scale-Devices).

Die Datenübertragung sieht wie folgt aus:

Building	Floor	Room	Device	Modusname	Type	Status	Readonly	Priority	Global
0	0	1	0	Dimmen	6	0	True	0	0
0	0	2	0	Schalter	1	0	True	OnS	offS
0	0	3	0	Lampe	1		Ture	onL	offL

Alle Informationen der GroupAdress sind „0“ bis auf die Room ID. In diesem Feld wird die ID des „Globals“ gespeichert. Anhand des Devices Types kann definiert werden, welche Art des Commands benötigt wird. (True – False; Float-Value etc.)

Die Commands für das Schalten der Global sehen dann folgendermaßen aus:

Command	Aktion
Gerald:kennwort:set:1.1.2.0:80*	Global „Dimmen“ aufrufen und auf 80 % setzen im ersten Gebäude, ersten Stock und zweiten Raum.
Wolfgang:passwd:set:2.1.0.0:onL*	Global „Schalter“ aufrufen und alle Schalter im Gebäude 2, Stock 1 einschalten

6 Software

Nachfolgend ist eine Auflistung der von uns im Projekt verwendeten Software zu finden.

Entwicklungsumgebung

- Betriebssysteme (Windows XP, Windows 98)
- JDK 1.3.1_01
- J2ME Wireless Toolkit 1.0.4_01
- Siemens M50 Emulator (Version vom 4. September 2002, 16:14:00)
- Siemens C55 Emulator (Version vom 19. November 2002, 09:08:52)
- Siemens S55 Emulator (Version vom 13. Dezember 2002, 08:42:27)

Mobile Endgeräte

- MT50 Java Activator
- Siemens Data Exchange Software (Version 2.67)
- Nokia PC Data Suite for 7650 (Version 1.1.0)
- MT50 Firmware Version 10 und Version 17
- C55 Firmware Version 14
- Nokia 7650 Firmware Version 3.16

7 Hardware

Nachfolgend ist eine Auflistung der von uns im Projekt verwendeten Hardware zu finden.

Entwicklungsumgebung

- Desktop PCs
- Notebooks
- Instabus Beispielinstallation

Mobile Endgeräte

- 1 x Siemens C55 + Siemens Datenkabel
- 2 x Siemens MT50 + Siemens Datenkabel und Noname Datenkabel
- 1 x Nokia 7650