UPnP & Konnex

Diplomarbeit im Fach Mikroprozessortechnik

Roger Meier

UPnP & Konnex: Diplomarbeit im Fach Mikroprozessortechnik

Roger Meier

Möglichkeiten von UPnP zur Repräsentation von Geräten auf dem Feldbus Konnex

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Vorwort

In der Automation und wie auch in Haushalten werden immer häufiger Komponenten eingesetzt, die über eine Ethernet-Schnittstelle verfügen. Da die Netzwerkinfrastruktur immer wieder anders aussieht, ist eine manuelle Konfiguration der Netzwerkparameter fast immer erforderlich. Auf weitere Probleme stösst man, wenn eine Kommunikation zwischen den Geräten erforderlich ist. Die Geräte können einander fast nur durch spezielle properitäre Protokolle finden. Auch der Austausch von Daten ist meist properitär gelöst.

UPnP steht für Universal Plug and Play und spezifiziert eine Umgebung zur herstellerunabhängigen Ansteuerung von Geräten über ein IP basierendes Netzwerk. Dabei werden vor allem Standards des *World Wide Web Consortium[W3C]* oder Vorschläge der *Internet Engineering Task Force[IETF]* verwendet. Bereits jetzt sind einige Geräte, wie zum Beispiel Router oder Stereoanlagen, die UPnP unterstützen auf dem Markt.

In der Heimautomation ist der Feldbus Konnex(KNX, Nachfolger von EIB) der führende Standard in Europa. In der Diplomarbeit möchte ich die Möglichkeiten zur Koppelung von KNX-Geräten am Beispiel eines Heizungsreglers aufzeigen.

Auf der von der Firma *Siemens Building Technologies AG [SBT]* zu Verfügung gestellten Hard- und Software, welche das Betriebssystem GNU/Linux[1] verwendet und eine Schnittstelle auf den Konnex Feldbus bereit stellt. Wird ein UPnP-Gerät realisiert, welches Regler, die am Feldbus angeschlossen sind repräsentiert.

[1]GNU/Linux (häufig ausschliesslich Linux genannt) Betriebssysteme basieren auf Programmen die der *GNU Public Licence[GPL]* unterstehen und den Linux®[Linux]-Kernel verwenden.

Kapitel 1. UPnP

Universal Plug and Play (UPnP™) basiert auf einer Reihe von standardisierten Netzwerkprotokollen und Datenformaten. Es dient zur herstellerübergreifenden Ansteuerung von Geräten (Stereoanlagen, Router, Drucker, Haussteuerungen) über ein IP basierendes Netzwerk.

Das *UPnP Forum [UPnP-Forum]* ist ein Zusammenschluss führender Hersteller aus der Unterhaltungs- und Computerindustrie. Es wurde im Juni 1999 gegründet und zählt mittlererweile über 600 Mitglieder. Das UPnP Forum ist die zentrale Anlaufstelle für die Weiterentwicklung der UPnP Architektur und Spezifikationen.

Eine weitere wichtige Organisation ist die *UPnP Implementers Corporation [UPnP-IC]* , welche Geräte die dem Standard entsprechend zertifiziert und somit das führen des UPnP-Logos auf dem Produkt bewilligt.

Die UPnP Technologie zeichnet sich insbesondere durch die folgenden Hauptmerkmale aus:

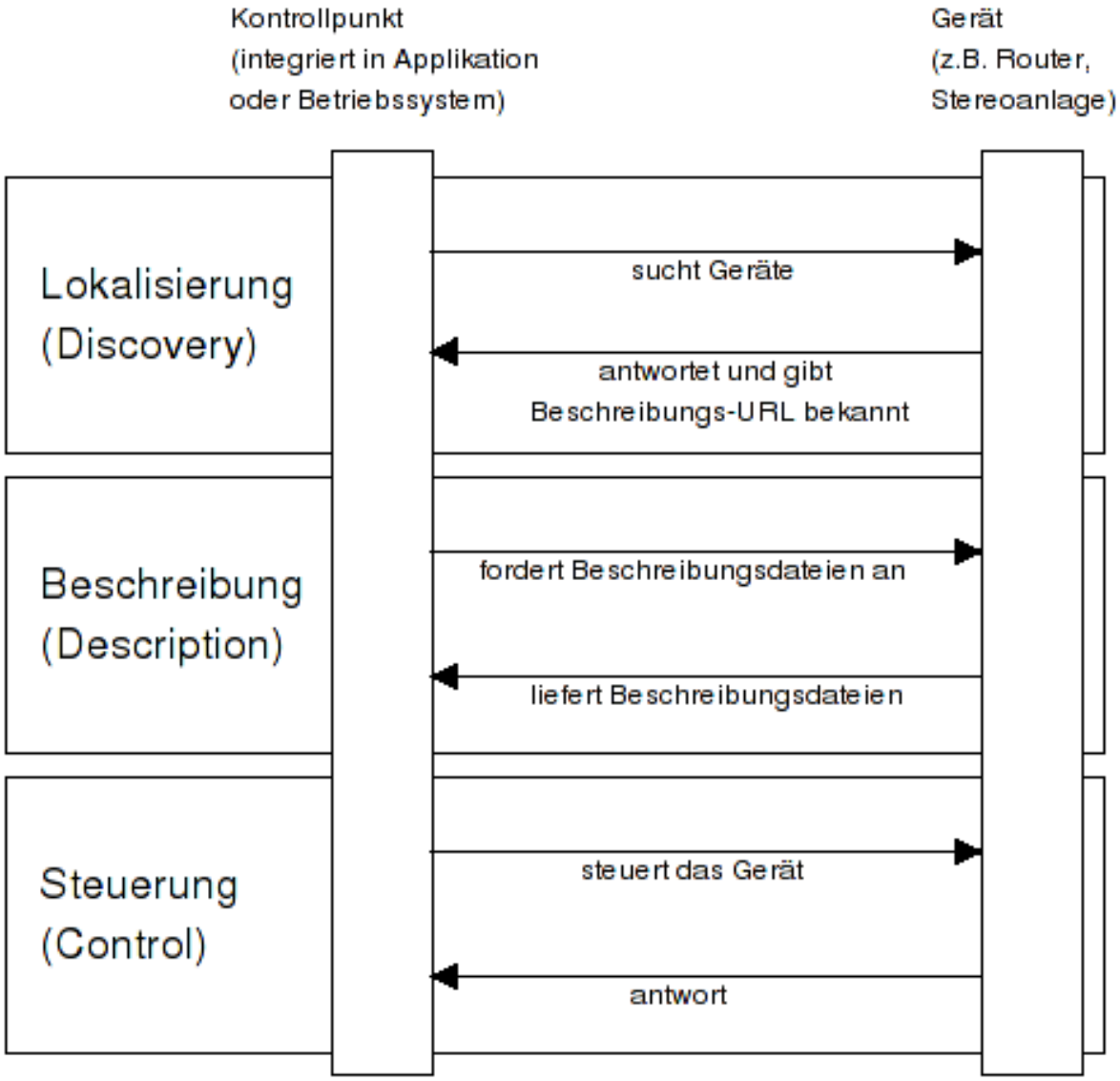
- Ein Kontrollpunkt (z.B. Handheld) kann die Geräte (z.B. Stereoanlage) ohne Interaktion des Benutzers finden.
- Alle physikalischen Medien, die IP-Kommunikation unterstützen können verwendet werden: z.B. Ethernet, Funk (Bluetooth, Wireless LAN), Firewire (IEEE 1394), Powerline
- Es werden standartisierte Technologien wie z.B. IP, UDP, Multicast, TCP, HTTP, XML und SOAP verwendet.
- Eine UPnP-Gerät oder -Kontrollpunkt kann auf jedem IP-fähigen Betriebssystem und mit beliebigen Programmiersprachen realisiert werden.
- Neben der Steuerung der Geräte durch konventionelle Programme oder via Browser, ist eine reine Maschinen-Maschinen-Kommunikation möglich.
- Es besteht die Möglichkeiten zur herstellerspezifischen Erweiterung.
- Unterstützung von Unicode (UTF-8 Kodierung [RFC2279]), so dass Zeichen aller Sprachen dargestellt werden können.

Die UPnP Architektur ist hauptsächlich für den Heimbereich und kleinere Firmennetzwerke gedacht. Es ist keine Konfiguration nötig und es wird keine IT-Infrastruktur wie DHCP oder DNS-Server vorausgesetzt.

Grundsätzlich wird bei UPnP zwischen Gerät(*Device*) und Kontrollpunkt(*Control Point*) unterschieden. Wobei die Kontrollpunkte meist in Desktop-, Web-Applikationen oder sogar Betriebssysteme wie Windows XP eingebunden sind.

Nach der IP-Adressierung folgt die Lokalisierung, Beschreibung und Steuerung.

Abbildung 1.1. UPnP: Lokalisierung, Beschreibung und Steuerung



1.1. Funktionsweise

Im folgenden werde ich die Funktionsweise des aktuellen Standards *UPnP Device Architecture Version 1.0 [UPnP-1.0]* erläutern.

Aus der Sicht der Netzwerkprotokolle wird UPnP wie folgt definiert.

Tabelle 1.1. Übersicht UPnP Protokoll-Stack

<table>
<tr><td colspan="5">UPnP Herstellerspezifische Erweiterungen</td></tr>
<tr><td colspan="5">UPnP Forum (Standards)</td></tr>
<tr><td colspan="5">UPnP Architektur</td></tr>
<tr><td>SSDP</td><td>GENA</td><td>SSDP</td><td>SOAP</td><td>GENA</td></tr>
<tr><td colspan="2">HTTP Multicast</td><td>HTTP Unicast</td><td colspan="2">HTTP</td></tr>
<tr><td colspan="3">UDP</td><td colspan="2">TCP</td></tr>
<tr><td colspan="5">IP</td></tr>
</table>

Die UPnP-Architektur lässt sich in die Phasen Adressierung, Lokalisierung, Beschreibung der Geräte, Steuerung, Ereignismeldungen und Präsentation aufteilen.

1.1.1. Adressierung

Da die Basis von UPnP ein IP-Netzwerk ist, muss ein Gerät oder Kontrollpunkt zuerst über eine gültige IP-Adresse verfügen. Dies kann nach dem UPnP-Standard einerseits via DHCP oder via Auto-IP erfolgen.

Das *Dynamic Host Configuration Protocol DHCP [RFC2131]* wird in den meisten modernen Netzwerken zur automatischen IP-Adressvergabe verwendet. Bei handelsüblichen Internet-Router für den Heimgebrauch gehört ein integrierter DHCP-Server heutzutage bereits zur Grundausstattung.

Falls kein DHCP-Server vorhanden ist, muss sich das Gerät selbst eine IP-Adresse vergeben. Diese Technik wurde ursprünglich von Microsoft unter dem Namen Auto-IP entwickelt und wird jetzt in der *Zeroconf Working Group [ZEROCONF]* der *Internet Engineering Task Force [IETF]* weiterentwickelt und spezifiziert.

Das Gerät muss eine IP-Adresse von dem von der *Internet Assigned Numbers Authority [IANA]* für Auto-IP reservierten IP-Adressbereich 169.254/16[2] auswählen und durch das *Adress Resolution Protocol ARP* prüfen, ob die Adresse schon vergeben ist. Dieser Vorgang muss solange wiederholt werden bis eine freie IP-Adresse gefunden wurde und auf der lokalen Netzwerkschnittstelle konfiguriert werden kann. Zudem muss das Gerät periodisch überprüfen ob nicht doch ein DHCP-Server verfügbar ist.

[2] die ersten und letzten 256 Adressen des Adressbereichs 169.254/16 sind reserviert und dürfen nicht benutzt werden

1.1.2. Lokalisierung

Die Lokalisierung der UPnP-Geräte erfolgt über das *Simple Service Discovery Protocol [SSDP]*, welches jedoch erst als Entwurf beim *Internet Engineering Task Force [IETF]* vorliegt. Obwohl der offizielle Standartisierungsprozess noch nicht abgeschlossen ist, kann durch die Präsenz aller namhaften Hersteller davon ausgegangen werden, dass sich das *Simple Service Discovery Protocol* in naher Zukunft als Standart durchsetzen wird.

Ein weiteres noch nicht standartisiertes Protokoll das bei *SSDP* eingesetzt wird ist *Multicast and Unicast UDP HTTP Messages [HTTP-UDP]* .

Nach der Lokalisierung ist die Adresse der Beschreibungsdatei verfügbar, über welche detaillierte Informationen über das Gerät verfügbar werden.

Jedes UPnP-Gerät das dem Netzwerk hinzugefügt wird, meldet sich durch einen UDP-Multicast auf der standartisierten Adresse 239.255.255.250 auf dem Port 1900 [3], mit der folgenden **ssdp:alive**-Meldung.

```
NOTIFY * HTTP/1.1
SERVER: Linux/2.4.24freakshow UPnP/1.0 CyberLink/1.3.1
CACHE-CONTROL: max-age=1800
LOCATION: http://10.0.0.99:4004/description.xml
NTS: ssdp:alive
NT: upnp:rootdevice
USN: uuid:c867-a0fb-7c19-e0a7::upnp:rootdevice
HOST: 239.255.255.250:1900
```

In der Zeile CACHE-CONTROL wird die Gültigkeitsdauer des Gerätes in Sekunden festgelegt. Jedes Gerät muss seine Anwesenheit im Netzwerk innerhalb dieser Gültigkeitsdauer erneut melden, sonst wird von den Kontrollpunkten angenommen das es vom Netzwerk entfernt wurde.

Beim expliziten Ausschalten eines Gerätes werden alle Geräte ebenfalls durch einen UDP-Multicast mit der **ssdp:byebye**-Meldung informiert.

```
NOTIFY * HTTP/1.1
NTS: ssdp:byebye
NT: upnp:rootdevice
USN: uuid:c867-a0fb-7c19-e0a7::upnp:rootdevice
HOST: 239.255.255.250:1900
```

Die alive und byebye Meldungen müssen für jedes Gerät(rootdevice), deren eingebetteten Geräte(embedded Devices) und jeden Dienst(Service) erfolgen.

Weiter kann auch explizit nach einem Gerät oder einem Dienst gesucht werden. Die Suche erfolgt ebenfalls durch einen UDP-Multicast.

```
M-SEARCH * HTTP/1.1
MX: 10
ST: upnp:rootdevice
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
```

Die Antwort auf die Suche ist die einzige Komponente des *Simple Service Discovery Protocol*, die einen UDP-Unicast verwendet.

[3] Die Vergabe von IP-Adressen und Portnummern erfolgt durch die *Internet Assigned Numbers Authority [IANA]* , siehe http://www.iana.org/assignments/multicast-addresses http://www.iana.org/assignments/port-numbers

```
HTTP/1.1 200  OK
ST:  upnp:rootdevice
EXT:
SERVER:  Linux/2.4.24freakshow UPnP/1.0 CyberLink/1.3.1
USN:  uuid:c867-a0fb-7c19-e0a7::upnp:rootdevice
CACHE-CONTROL:  max-age=1800
LOCATION:  http://10.0.0.99:4004/description.xml
Content-Length: 0
```

1.1.3. Beschreibung der Geräte

Nachdem ein Kontrollpunkt ein Gerät gefunden hat, holt er sich per HTTP über TCP/IP die Beschreibung des Gerätes von der URL, welche ihm bei der Lokalisierung mitgeteilt wurde. Diese stellt das Gerät in Form eines XML-Dokumentes zur Verfügung.

Die Beschreibung beinhaltet alle Informationen über den Hersteller, die Seriennummer, URL's für die Steuerung, Ereignisse und die Präsentation.

Die folgende Gerätebeschreibung repräsentiert ein BinaryLight.

```
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <device>
    <deviceType>urn:schemas-upnp-org:device:BinaryLight:1</deviceType>
    <presentationURL>/web</presentationURL>
    <friendlyName>lichtle</friendlyName>
    <manufacturer>Roger Meier</manufacturer>
    <manufacturerURL>http://www.bufferoverflow.ch</manufacturerURL>
    <modelDescription>Lichtschalter</modelDescription>
    <modelName>Lichtschalter Parallelport (Test Diplomarbeit)</modelName>
    <modelNumber>BL-1</modelNumber>
    <serialNumber>0000001</serialNumber>
    <UDN>uuid:b29a6ca5-ddaf-4284-9ede-ca6b27480bba</UDN>
    <iconList>
     <icon>
       <mimetype>image/png</mimetype>
       <width>48</width>
       <height>32</height>
       <depth>8</depth>
       <url>icon.png</url>
     </icon>
    </iconList>
    <serviceList>
      <service>
        <serviceType>urn:schemas-upnp-org:service:SwitchPower:1</serviceType>
        <serviceId>urn:upnp-org:serviceId:SwitchPower.0001</serviceId>
        <SCPDURL>SwitchPower/scpd.xml</SCPDURL>
        <controlURL>SwitchPower/control</controlURL>
        <eventSubURL>SwitchPower/event</eventSubURL>
      </service>
    </serviceList>
  </device>
</root>
```

Für jeden Service, den ein Gerät anbietet werden Kommandos und Aktionen, sowie Datentypen und Datenbereiche spezifiziert.

Die folgende Liste der standardisierten Datentypen für UPnP zeigt, dass sich für fast jede Applikation ein passender Datentyp definiert ist.

Tabelle 1.2. UPnP Datentypen

DatenTyp	Beschreibung
ui1	1 Byte Integer Wert ohne Vorzeichen
ui2	2 Byte Integer Wert ohne Vorzeichen
ui4	4 Byte Integer Wert ohne Vorzeichen
i1	1 Byte Integer Wert mit Vorzeichen
i2	2 Byte Integer Wert mit Vorzeichen
i4, int	4 Byte Integer Wert mit Vorzeichen
r4	4 Byte Fliesskommazahl
r8, number	8 Byte Fliesskommazahl
fixed14.4	gleich wie r8, aber mit einem Maximum von 14 Zeichen vor und 4 nach dem Komma
float	Fliesskommazahl mit Exponent
char	1 Unicode Zeichen
string	Unicode Zeichenkette beliebiger Länge
date	Monat, Tag, Jahr im ISO 8601 Format
dateTime	Datum und optional Zeit im ISO 8601 Format
dateTime.tz	Datum und optional Zeit im ISO 8601 Format mit Angabe der Zeitzone
time	Zeit im ISO 8601 Format
time.tz	Zeit im ISO 8601 Format mit Angabe der Zeitzone
boolean	boolean, 0 = false, 1 = true
bin.base64	Base64 codierte binäre Daten
bin.hex	Hexadezimale Zeichen als Octet
uri	*Uniform Resource Identifier*, z.B. http://www.gnu.org/ oder ftp://ftp.gnu.org
uuid	*Universally Unique ID*, spezifiziert als hexadezimale Octet

Hier die Beschreibungsdatei des Dienstes SwitchPower, welcher unter der SCPDURL zur Verfügung gestellt wird.

```
<?xml version="1.0" encoding="UTF-8"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <actionList>
    <action>
```

```
      <name>GetStatus</name>
      <argumentList>
        <argument>
          <name>ResultStatus</name>
          <direction>out</direction>
          <relatedStateVariable>Status</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>SetTarget</name>
      <argumentList>
        <argument>
          <name>newTargetValue</name>
          <direction>in</direction>
          <relatedStateVariable>Target</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
  </actionList>
  <serviceStateTable>
    <stateVariable sendEvents="yes">
      <name>Status</name>
      <dataType>boolean</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>Target</name>
      <dataType>boolean</dataType>
    </stateVariable>
  </serviceStateTable>
</scpd>
```

Die Beschreibung kann neben den Diensten die es anbietet auch alle eingebetteten Geräte mit deren Diensten beinhalten.

1.1.4. Steuerung

Anhand der Informationen die der Kontrollpunkt von der Beschreibung des Gerätes und der Dienste erhalten hat, kann er nun SOAP-Mitteilungen an die controlURL des Gerätes schicken um dieses zu steuern.

Das *Simple Object Access Protocol* [SOAP] wird vom *World Wide Web Consortium* [W3C] spezifiziert und weiterentwickelt. SOAP ist mittlererweile sehr stark verbreitet und wird häufig in Geschäftsapplikationen verwendet. Amazon bietet zum Beispiel SOAP-Schnittstellen an um auf den Warenkatalog zuzugreifen.

Die folgende SOAP-Nachricht zeigt die Steuerung(bzw. das Einschalten) des Dienstes SwitchPower des Gerätes BinaryLight. Dieser wird an die *controlSubURL* des UPnP-Gerätes, welche bei der Beschreibung mitgeteilt wurde via HTTP gesendet.

```
POST SwitchPower/control HTTP/1.1
CONTENT-TYPE: text/xml; charset="utf-8"
HOST: 10.0.0.99:4006
CONTENT-LENGTH: 280
SOAPACTION: "urn:schemas-upnp-org:service:SwitchPower:1#SetTarget"

<?xml version="1.0"?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" \
       s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <s:Body>
  <u:SetPower xmlns:u="urn:schemas-upnp-org:service:SwitchPower:1">
   <Power>1</Power>
  </u:SetPower>
 </s:Body>
</s:Envelope>
```

Die Antwort erfolgt ebenfalls als SOAP-Nachricht, wie das folgende Beispiel zeigt:

```
HTTP/1.1 200 OK
CONTENT-TYPE: text/xml; charset="utf-8"
SERVER: Linux/2.4.24freakshow UPnP/1.0 CyberLink/1.3.1
CONTENT-LENGTH: 298
EXT:
DATE: Sun, 09 May 2004 06:31:15 GMT

<?xml version="1.0"?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" \
       s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <s:Body>
  <u:SetPowerResponse xmlns:u="urn:schemas-upnp-org:service:SwitchPower:1">
   <Result>1</Result>
  </u:SetPowerResponse>
 </s:Body>
</s:Envelope>
```

Fehlermeldungen werden ebenfalls als SOAP-Nachrichten zurückgegeben, zudem wird auch die HTTP-Antwort mit dem Fehlercode 500(Internal Server Error) zurückgegeben.

1.1.5. Ereignismeldungen

Damit ein Gerät nicht dauernd über den Zustand eines Dienstes bzw. einer Statusvariable abgefragt werden muss, nutzt UPnP die XML-basierte *General Event Notification Architecture [GENA]*. Mit GENA können Kontrollpunkte Ereignisse auf Geräten abbonieren. Somit werden sie bei jeder Änderung einer Statusvariable automatisch informiert.

Die folgende SOAP-Nachricht an die *eventSubURL* des UPnP-Dienstes abonniert die Ereignisse bzw. Statusänderungen des Gerätes. Die folgenden Antworten und Statusänderungen werden an die angegebene CALLBACK-Adresse des Kontrollpunktes geschickt.

```
SUBSCRIBE SwitchPower/event HTTP/1.1
TIMEOUT: Second-300
HOST: 10.0.0.132:58435
CALLBACK: <http://10.0.0.132:9864/31bd3a40-020f-4b17-a9fe-be2ffc7545a0/SwitchPower.0001>
NT: upnp:event
```

Das UPnP-Gerät antwortet mit dem folgenden HTTP-Header der mit dem TIMEOUT-Feld die Gültigkeitsdauer bestätigt. Nach Ablauf der Gültigkeitsdauer muss der Kontrollpunkt die Ereignisse erneut mit einem SUBSCRIBE Befehl abonnieren.

```
HTTP/1.1 200  OK
TIMEOUT:  Second-300
SID:  uuid:31bd3a40-020f-4b17-a9fe-be2ffc7545a0-SwitchPower.0001-3
SERVER:  Windows NT/5.0, UPnP/1.0, Intel CLR SDK/1.0
Content-Length: 0
```

Das Gerät meldet nun bei jeder Änderung einer Statusvariablen den neun Wert:

```
NOTIFY /31bd3a40-020f-4b17-a9fe-be2ffc7545a0/SwitchPower.0001 HTTP/1.1
SEQ:  0
CONTENT-TYPE:  text/xml
HOST:  10.0.0.132:9864
SID:  uuid:31bd3a40-020f-4b17-a9fe-be2ffc7545a0-SwitchPower.0001-3
CONNECTION:  close
NT:  upnp:event
```

```
NTS:  upnp:propchange
Content-Length: 175
<?xml version="1.0" encoding="UTF-8"?>
<e:propertyset xmlns:e="urn:schemas-upnp-org:event-1-0">
   <e:property>
      <Status>0</Status>
   </e:property>
</e:propertyset>
```

Zudem wird die Sequenznummer(SEQ) im HTTP-Header mit jeder Meldung inkrementiert, damit der Kontrollpunkt die einzelnen Meldungen auseinanderhalten kann.

1.1.6. Präsentation (Presentation)

Die Präsentation ist eine Alternative zur Steuerung und den Ereignismeldungen. Über die presentationURL welche bei der Beschreibung (Description) bekanntgegeben wird, kann mittels Webbrowser auf das Gerät zugegriffen werden. Dies gibt dem Hersteller die Möglichkeit neben dem standardisierten Zugriff via UPnP eine alternative interaktive Benutzeroberfläche zur Verfügung zu stellen.

1.2. Geräte und Dienste

Auf Basis der UPnP Architektur können Dienste verschiedenster Art definiert werden. Es besteht ebenfalls die Möglichkeit eigene Dienste und Geräte zu definieren, dies zielt jedoch nicht gerade in die Richtung der Idee von UPnP.

1.2.1. standardisierte Geräte

Die Definitionen der Standards sind auf der Website des *UPnP Forum [UPnP-Forum]* verfügbar. Für jedes Gerät sind meist mehrere Dienste vorhanden, es müssen jedoch nur einzelne Dienste zwingend Implementiert sein.

Zurzeit (9. Mai 2004) sind die folgenden Standards veröffentlicht.

- **Internet Gateway Device (IGD) V1.0**

  Bereits sind über 60 Geräte (Router, ADSL Gateway), die den Standard *Internet Gateway Device* implementieren im Handel erhältlich und von der *UPnP Implementers Corporation [UPnP-IC]* zertifiziert worden.

  Ein grosser Vorteil des *Internet Gateway Device* gegenüber herkömmlichen Routern ist der Dienst *Layer3-Forwarding*. Er ermöglicht Applikationen wie zum Beispiel Instant-Messenger, Videokonferenz-Systemen oder Online-Spielen ohne Aktion des Benutzers ein sogenanntes Portforwarding einzurichten, damit die Applikation, obwohl sie sich auf einem Computer im privaten Netzwerk befindet, vom Internet aus erreichbar ist.

  Eine mögliche Anwendung in einem Gerät für die Hausautomation, wäre ein UPnP-Kontrollpunkt(Controlpint) zur Einrichtung einer Layer3-Forwarding Regel.

  „Die Website für die Bedienung des Gerätes via Internet könnte so komfortabel ein- und ausgeschaltet werden.“

  Dies würde allerdings die Verwendung einer fixen IP-Adresse oder eines ständig mit der neuen IP-Adresse aktualiserten Domainnamen erfordern.

  Ich könnte mir sehr gut vorstellen, dass ein weiterer Dienst des IGD's das Einrichten eines dynamischen DNS Update-Services (wie dies heute bei gewissen Routern bereits möglich ist) sein könnte.

- **Media Server V1.0 and Media Renderer V1.0**

  Der MediaServer kann Inhalte wie zum Beispiel Musik, Filme, und Fotos bereitstellen, die der MediaRenderer abspielt. Die Steuerung beider Geräte kann von jedem UPnP-Kontrollpunkt aus geschehen.

  Die ersten MediaServer wurden bereits zertifiziert. Zudem sind weitere Produkte auf dem Markt, die nicht zertifiziert sind. So der *Network Media Receiver* von Sony und *Streamium* von Philips.

Eine weitere herstellerübergreifende Organisation, welche in Ihren Definitionen ebenfalls auf diesen beiden UPnP-Geräten aufsetzt, ist die *Digital Home Working Group (http://www.dhwg.org/)*.

Bis heute wurden allerdings noch keine Mechanismen zum Schutz der Rechte der Medieninhalte spezifiziert.

Der MediaServer und MediaRenderer sind die Geräte, die nach dem Internet Gateway Device das grösste Marktpotenzial darstellen. Und es ist damit zu rechnen, dass noch etliche Produkte zur Wiedergabe bzw. Bereitstellung von Multimedia-Daten auf den Markt kommen werden.

- **Basic Device V1.0**

  Das Basic Device ist die Basis für eine herstellerspezifische Implementation eines UPnP-Gerätes. Es enthält keine Dienste, es können jedoch beliebige standardisierte oder auch selbst definierte Dienste implementiert werden. Es allerdings mindestens ein Standard-Dienst definiert sein, damit das Gerät nach der Zertifizierung durch die *UPnP Implementers Corporation [UPnP-IC]* das UPnP Logo tragen darf.

  Mit dem Basic Device ohne Dienste kann man trotzdem von den Lokalisierungs- und Beschreibungsdiensten der UPnP-Architektur profitieren, was für viele Anwendungen mit einem gut ausgebauten Webinterface ausreichen würde. Dies würde die Anforderungen bezüglich Flexibilität und Funktionsumfang der UPnP-Programmbibliothek bedeutend reduzieren, die UPnP-Architektur müsste lediglich bis zur Beschreibung des Gerätes verwendet werden. Die aufwändigeren Dienste (Steuerung, Ereignismeldungen) müssten nicht implementiert werden.

- **Device Security V1.0 and Security Console V 1.0**

  Die UPnP Device Security basiert auf den modernen Verschlüsselungstechnologien AES, SHA1 und RSA. Es ermöglicht das festlegen von Eigentümmern und Berechtigungen der UPnP-Geräte. Die *Device Security* ist eine wichtige Basis für den künftigen Erfolg von UPnP, nur mit gut ausgestatteten Sicherheitsmechanismen ist eine hohe Akzeptanz auf dem Markt möglich. Bis jetzt sind noch keine Geräte auf dem Markt, die Security Dienste implementiert haben, es gibt jedoch bereits einige Beispielimplementationen von Mitgliedern des UPnP-Forums.

- **Lighting Controls V1.0**

  Die *Lighting Controls* werden häufig in Demoprogrammen verwendet. Es werden Dienste zum Ein- uns Ausschalten und Dimmen von Lampen zur Verfügung gestellt.

Die folgenden Geräte und Dienste sind ebenfalls als Standards definiert, zum jetztigen Zeitpunkt sind aber noch keine Geräte auf dem Markt.

- **Printer Device and Print Basic Service V1.0**

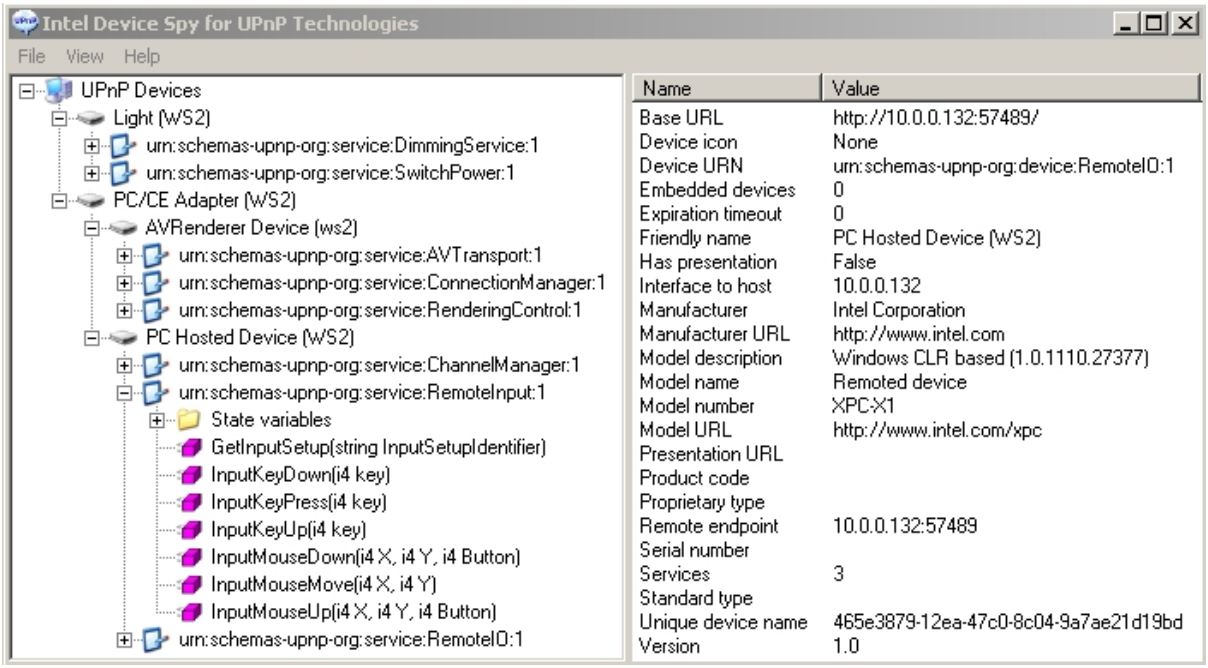
- **Scanner (External Activity V1.0, Feeder V1.0, Scan V1.0, Scanner V1.0)**
- **HVAC V1.0**
- **WLAN Access Point Device V1.0**

1.2.2. künftige Dienste

Ein vielversprechender Entwurf von Intel ist RemoteIO, welches mit Applikation wie X11, ICA oder Remote Desktop(RDP) vergleichbar ist.

Auf Basis des *Media Renderer* Standards wird eine komplette Bedienoberfläche in der entsprechenden Bildschirmgrösse einem PC, TV oder Handcomputer zur Verfügung gestellt. Die Eingabeparamter von Maus, Touchscreen und Tastatur werden an den RemoteIO-Server geschickt und dieser steuert dann die gewünschten Applikationen, wie zum Beispiel Stereoanlage, Klimaanlage und Licht.

Abbildung 1.2. Ansicht RemoteIO und Binary Light



Die Evaluationspacket *Intel® Remote I/O for UPnP Technologies[UPnP-INTEL]* zeigt die Möglichkeiten von RemoteIO sehr gut auf.

1.3. Alternativen zu UPnP

Multicast-DNS [ZEROCONF], welches bei *Apple* unter dem Namen *Rendevouz* bereits zum Einsatz kommt, ermöglicht ebenfalls auf Basis der IP-Kommunikation das Lokalisieren von Geräten und Diensten. Genaue Beschreibungen oder Ereignismeldungen wie bei UPnP werden jedoch nicht zur Verfügung gestellt, dies ist jeweils Sache der Applikationen wie zum Beispiel von *itunes* im Zusammenhang mit dem *Digital Audio Access Protocol (http://daap.sourceforge.net/)*.

Das *Service Location Protocol [RFC2608]*, welches von der *Internet Engineering Task Force [IETF]* als Standard herausgegeben wurde, beschränt sich ebenfalls auf das Lokalisieren von Diensten und wird hauptsächlich auf UNIX-Systemen verwendet.

Die *Jini Network Technology (http://www.sun.com/software/jini/)* wurde von Sun Microsystems entwickelt. Sie stellt ebenfalls Dienste zur Lokalisierung von Geräten und Applikationen zur Verfügung. Es basiert neben diversen Netzwerkprotokollen ausschliesslich auf der Java-Technologie und ist somit Platformunabhängig, jedoch nicht Sprachunabhängig.

Im Buch *Service and Device Discovery Richard2002* wird die Funtionalität von JINI, dem Service Location Protocol (SLP), UPnP und dem Bluetooth Service Discovery Protocol ausführlich beschrieben.

Es ist sehr schwierig klar zwischen den verschiedenen Technologien zu gewichten, da der Funktionsumfang nie deckungsgleich ist.

„Die hohe Anzahl der Hersteller, sowie der Einsatz von platformunabhängigen Technologien macht UPnP aus meiner Sicht zur zukunftsträchtigsten Technologie.“

1.4. Sicherheit

Durch die hohe Anzahl der eingesetzten Netzwerkprotokolle, wird die Angriffsfläche bei UPnP-Systeme im Vergleich zu herkömmlichen Systemen, die lediglich über ein Webinterface verfügen, um einiges grösser. Die UPnP-Technologie wird jedoch hauptsächlich in Heim-Netzwerken eingesetzt und somit müssen die Anforderungen an die Sicherheit, gegenüber Applikationen in öffentlichen Netzen, nicht so hoch eingestuft werden.

Trotzdem müssen entsprechende Massnahmen ergriffen werden um einer missbräuchliche Benutzung von UPnP-Geräten zuvor zukommen.

Mit dem standardisierten UPnP-Service *Security Console* werden eine Reihe von Sicherheitsmechanismen, die auf modernen Verschlüsselungsverfahren aufbauen, zur Verfügung gestellt. So können Eigentümer und berechtigte Benutzer einem Gerät zugewiesen werden. So dass andere Benutzer im Netzwerk keine Möglichkeiten haben darauf zuzugreifen. Geräte die auf diesem Dienst aufbauen sind jedoch noch nicht verfügbar. Es ist jedoch damit zu rechnen, dass in den nächsten Jahren Geräte auf den Markt kommen, die diesen Dienst unterstützen. Die Akzeptanz von UPnP-Geräten ohne Sicherheitsmechanismen wird aus meiner Sicht in Zukunft eher abnehmen.

In den Anfängen von UPnP, im Jahre 2001, hat das *FBI* gemäss der Seite UnPlug n' Pray (http://grc.com/UnPnP/UnPnP.htm) die Empfehlung herausgegeben die UPnP-Funktionalität (insbesondere den SSDP-Dienst) auf Windows XP Systemen aus Sicherheitsgründen zu deaktivieren. Diese Empfehlung wurde aber wieder zurückgenommen, nachdem Microsoft die schwerwiegende Sicherheitslücke im SSDP-Dienst behoben hatte(*Microsoft Security Bulletin MS01-054*).

Einer der Hauptvorteile des *Internet Gateway Devices*, das Layer3-Forwarding, stellt zugleich ein Sicherheitsrisiko dar. So könnte zum Beispiel ein UPnP-Kontrollpunkt in ein Virus verpackt werden und ohne Wissen des Nutzers einen Port auf dem Computer vom Internet aus zugänglich machen.

„Der Kompromiss zwischen Benutzerfreundlichkeit und Sicherheit wird leider immer eine Gratwanderung sein.“

1.5. Programmbibliotheken

Durch die Komplexität der Funktionsweise von UPnP, ist für die Realisation eines UPnP-Devices beziehungsweise UPnP-Controlpoints der Einsatz einer geeigneten Programmbibliothek die alle benötigten Funktionen der UPnP Architektur zur Verfügung stellt von grossem Vorteil.

Da die Implementation und Tests der einzelnen Funktionen von UPnP sehr aufwändig sind, lässt sich ohne den Einsatz eines geeigneten Protokoll-Stacks[4] keine UPnP-Komponente in vernünftigem Zeitrahmen realisieren.

Die folgenden Kriterien sind für die Wahl eines geeigneten Protokoll-Stacks von Bedeutung:

- **Hardwareressourcen**

  Bei Geräten wie zum Beispiel Router oder Stereoanlagen, die in sehr hohen Stückzahlen produziert werden, sind die Ressourcen meist sehr beschränkt und die Anforderungen an den Speicherplatzbedarf an den Stack relativ hoch(meist nur einige Megabytes). Die Hardwarekosten wirken sich viel entscheidender auf die Kostenrechnung aus, so dass die höheren Entwicklungsaufwände nicht besonders relevant sind.

  Bei regulären PC's oder Geräten, die in kleinen Stückzahlen realisiert werden, kann durch den Einsatz eines umfangreichen Stacks die Entwicklungszeit entscheidend reduziert werden. Hier wirken die Entwicklungskosten gegenüber den Hardwarekosten stärker auf die Gesamtkosten des Projekts.

- **Hardwareplatform**

  Beim Einsatz von C oder C++ kommt die Bytereihenfolge des Prozessors zum Tragen.

  > Daten die aus mehreren Bytes bestehen(z.B. 0xD7A2B1), werden bei *big endian* Systemen (z.B. PowerPC) in der korrekten Reihenfolge (D7 A2 B1) und bei *little endian* Systemen(z.B. Intel-x86) in umgekehrter Reihenfolge(B1 A2 D7) im Speicher abgelegt. Netzwerkprotokolle wie zum Beispiel IP, TCP, UDP verwenden ebenfalls das *big endian* Format.

  Implizite Bit- und Byte-Operationen müssen im Quellcode des Stacks jeweils separat für *little endian* und *big endian* realisiert werden. Bei der Kompilierung der Applikation bzw. des Stacks muss dem Kompiler mit einem Parameter mitgeteilt werden welche Bytereihenfolge verwendet werden muss.

  Da bei Java eine Abstraktionsschicht zwischen Programmcode und Betriebssystem/Prozessor vorhanden ist, wird die Problematik der Bytereihenfolge in der *Java Virtual Machine* gelöst und ist somit für den Programmcode des Stack nicht relevant.

[4] Bei Programmbibliotheken die Funktionalitäten von Protokollen zur Verfügung stellen, wird oft auch von einem Protokoll-Stack bzw. Stack gesprochen.

Der Energiebedarf der verschieden Prozessoren und Chipsets kann ebenfalls von Bedeutung sein, so stellen Mikrokontrollersysteme neben PC's und embedded Platformen eine interessante Alternative dar.

Dallas-Maxim's *Networkprocessor* [TINI] verfügt über eine integrierte Java Virtual Machine und diverse Schnittstellen wie zum Beispiel Ethernet, CAN, I^2C oder 1-Wire®. Es lassen sich bis zu 16MB Speicher addressieren. Eine Speisung via Ethernet(PoE, *Power over Ethernet*) wäre durch den geringen Energieverbrauch des *Networkprocessor*'s ebenfalls denkbar.

- **Betriebsystem**

  Für UNIX Betriebssysteme wie den BSD-Derivaten(openbsd, freebsd, netbsd, Mac OS X), Solaris, AIX oder GNU/Linux existiert der POSIX-Standard[5], welcher eine einheitliche Schnittstelle(insbesondere für die Sprachen C und C++) zum Betriebssystem definiert. Die meisten UNIX-Derivate sind auf sehr vielen Hardwareplatformen lauffähig und zeichnen sich durch eine hohe Stabilität aus. Zudem sind Software und Programmbibliotheken bei BSD-Derivaten und GNU/Linux-Systemen inklusive Quelltext frei verfügbar.

  Die Windows Betriebssysteme sind im Heim- und Firmeneinsatz als Arbeitsstationen sehr stark verbreitet. In kompakten Geräten wird von einzelnen Herstellern ebenfalls Windows CE eingesetzt.

  Neben den erwähnten Betriebssystemen, komen noch diverse Echtzeitbetriebssysteme für ein UPnP-Gerät in Frage.

- **Programmiersprache**

  Für komplexe Applikationen ist der Einsatz einer *objektorientierten Programmiersprache* wie *C++*, *Java* oder *C#* meist von grossem Vorteil. Die Software lässt sich gegenüber nicht-objektorientierten Sprachen viel besser strukturieren und bei Wiederverwendbarkeit von teilen der Applikation ist ebenfalls höher. In der Regel ist bei objektorientierten Sprachen jedoch mit einem gewissen *overhead* zu rechnen, was sich beim Speicherbedarf bemerkbar macht.

  Für einfachere Applikationen und in Bereichen wo der Speicherbedarf klein gehalten werden muss, hat die Programmiersprache *C* gegenüber den anderen Sprachen klare Vorteile.

  Ein weiterer wichtiger Punkt, insbesondere für platformunabhängige Programme, ist die Standardkonformität. Der C Standard des *American National Standards Institute [ANSI]* , häufig auch ANSI-C genannt, ist einer der wichtigsten Standards der von den meisten Compilern unterstützt wird. Er wurde auch von der *International Organization for Standardization[ISO]* übernommen (ANSI/ISO 9899-1989) und wird häufig auch als C89 bezeichnet. Als Weiterentwicklung des Standards folgen ISO/IEC 9899-1990(C90) und ISO/IEC 9899-1999(C90). Für C++ wurde der Standard ISO/IEC 14882-1998 veröffentlicht.

  Durch die Unabhängikeit vom Betreibssystem ist *Java* auf jeder Platform lauffä-

[5] POSIX wurde und wird vom *Portable Application Standards Committee [PASC]* der IEEE entwickelt. Die beiden Standards IEEE1003.1 und IEEE1003.2 sind heute auch in der globalen Industrienorm ISO/IEC 9945 festgelegt.

hig, für die es eine *Java Virtual Machine* gibt. Hier kann die Version der Java Virtual Machine jedoch von Bedeutung sein, da es Inkompatibilitäten zwischen den verschiedenen Versionen gibt.

- **Kosten**

  Neben kommerziellen UPnP-Stacks gibt es zur Zeit auch zwei frei verfügbare Stacks. Bei der Entwicklung eines komerziellen Produktes kann es je nach Applikation von Vorteil sein, einen UPnP-Stack einzukaufen, da die Weiterentwicklung des Stacks und der technische Support die Entwicklungszeit und somit die *Time to Market* massgebend verkürzen kann.

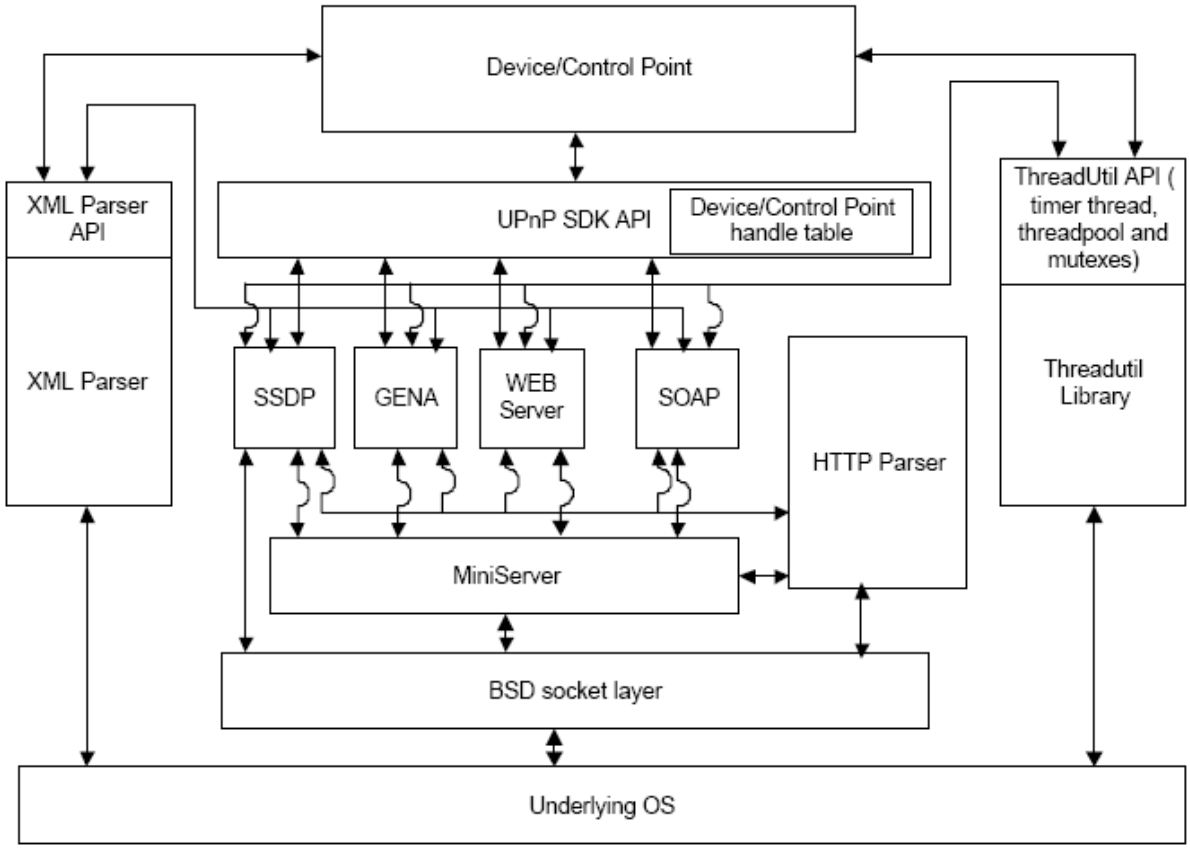
Ein weiteres wichtiges Entscheidungskriterium für den Einsatz eines geeigneten Stacks ist das vorhandene Fachwissen in Bezug auf Netzwerktechnologien, Hardwareplatformen, Betreibssystemen und Programmiersprachen. Im folgenden werde ich einige UPnP-Stacks kurz vorstellen.

1.5.1. Linux SDK for UPnP Devices

Im Jahre 2000 hat Intel die erste Version des *Linux SDK for UPnP Devices [LIBUPNP]* veröffentlicht und unter der BSD-Lizenz veröffentlicht. Der letzte Release wurde am 13 Februar 2003 veröffentlicht und steht inklusive Quelltext zur Verfügung.

Im Buch *UPnP Design by Example [JeronimoWeast2003]* wird die Funktionsweise und der Aufbau des Intel Linux SDK's ausführlich erläutert und der Einsatz anhand eines Toasters demonstriert.

Abbildung 1.3. Linux SDK Architektur



Das Linux SDK besteht aus einem XML-Parser, einer Thread-Programmbibiothek und der UPnP-Programmbibiothek welche insgesamt zirka 170kB benötigen.

Der folgende Ausgabe der Abhängigkeiten der gemeinsamen Programmbibliotheken, zeigt die wenigen Abhängikeiten des Linux SDK's.

```
roger@ws1:~/libupnp-1.2.1/upnp/bin$ ldd *.so
libixml.so:
 libc.so.6 => /lib/tls/libc.so.6 (0x4001f000)
 /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x80000000)
libthreadutil.so:
 libpthread.so.0 => /lib/tls/libpthread.so.0 (0x4001b000)
 libc.so.6 => /lib/tls/libc.so.6 (0x4002b000)
 /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x80000000)
libupnp.so:
```

```
libthreadutil.so => /usr/lib/libthreadutil.so (0x40033000)
libixml.so => /usr/lib/libixml.so (0x40039000)
libc.so.6 => /lib/tls/libc.so.6 (0x40042000)
libpthread.so.0 => /lib/tls/libpthread.so.0 (0x4017d000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x80000000)
```

Die 1,5 MB für die *GNU C Library [GLIBC]* Programmbibliotheken(libc, ld-linux und libpthread) gehören auf den meisten GNU/Linux Systemen zur Grundausstattung.

Inwiefern der Einsatz der Speicher- und Ressourcenoptimierten *dietlibc* (http://www.fefe.de/dietlibc/) oder *uClibc* (http://www.uclibc.org/) als Ersatz für die *GNU C Library* hier möglich ist habe ich nicht weiter untersucht. Dies wäre jedoch eine weiteres Möglichkeit Speicherplatz einzusparen.

Applikationen für ein einfaches Gerät oder einen Kontrollpunkt kommen auf zirka 40kB zu stehen.

Das Linux SDK ist, ohne Anpassungen, auch auf *big endian*-Systemen (z.B. PowerPC) lauffähig und somit sehr gut für den Einsatz auf allen Hardwareplatformen geeignet.

Es lassen sich einzelne Funktionen wie zum Beispiel GENA(Ereignissubsystem) deaktivieren. Die Anzahl der Threads sind ebenfalls einstellbar. Dies ermöglicht das Linux SDK den eigenen Bedürfnissen und der Rechenleistung der Hardware optimal anzupassen.

Durch den geringen Platzbedarf und die wenigen Abhängigkeiten ist das Linux SDK von Intel sehr gut für Systeme geeignet, die mit wenig Speicher zur Verfügung haben.

Netgear (http://www.netgear.com/) setzt im Produkt DG834G(54 Mbps Wireless ADSL Firewall Router) ebenfalls das Linux SDK ein, was die Stabilität und die Einsatzmöglichkeit auf embedded GNU/Linux Systemen bestätigt. Netgear setzt die oben erwähnte *uClibc* ein. Auf ihrer Webseite Open Source Code for Programmers (http://kbserver.netgear.com/kb_web_files/n101238.asp) stellt Netgear den Quellcode der verwendeten Programme und Bibliotheken inklusive der Änderungen, die vorgenommen wurden, zum Download bereit.

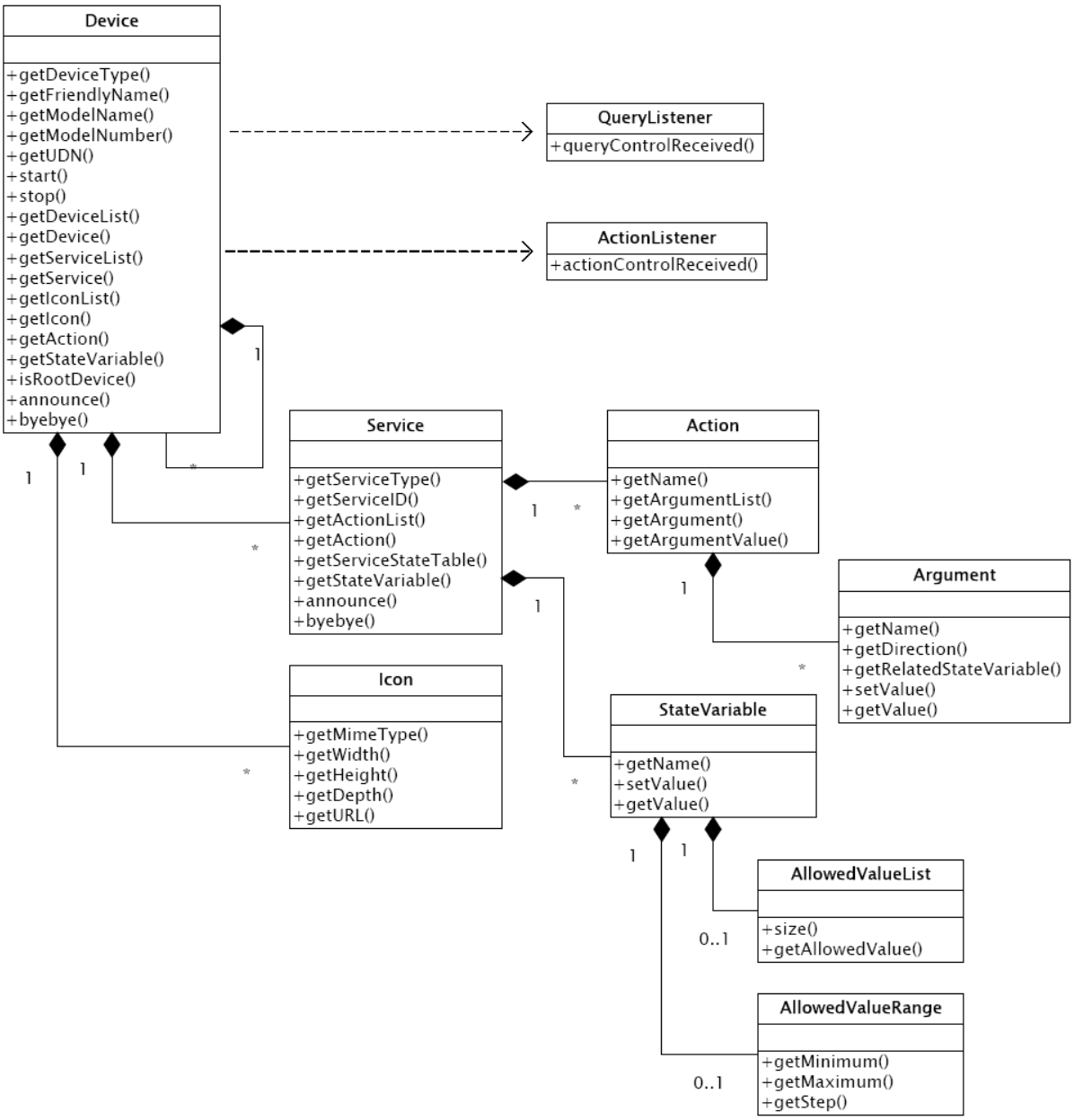
Im Linksys (http://www.linksys.com/) WMA11B (Wireless Digital Media Adapter) wird ebenfalls das Linux SDK eingesetzt.

1.5.2. CyberLink

Der *CyberLink UPnP-Stack [CYBERGARAGE]* wurde von Satoshi Konno aus Tokyo entwickelt und steht in den Programmiersprachen C++(Platformen Windows, Mac OS X, UNIX) und Java inklusive Quelltext zur Verfügung.

Durch den Einsatz der Objektorientierten Sprachen C++ und Java ist der CyberLink-Stacks sehr sauber und übersichtlich strukturiert.

Abbildung 1.4. CyberLink UPnP-Device UML-Diagramm



Für beide Sprachen stehen Klassen zur Erstellung eines Gerätes und eines Kontrollpunktes zu Verfügung.

Die Methoden und Klassen wurden beim Java- und C++-SDK gleich benannt. Ebenso wird auf beiden Stacks der Xerces XML-Parser in der entsprechenden Sprache eingesetzt.

Der hohe Abstraktionsgrad erspart einem viel Arbeit, so werden zum Beispiel die ssdp-Meldungen beim Starten des Gerätes automatisch ausgelöst. Das Verwalten der abonnierten Ereignisse und auch das versenden der Ereignisse (nach Änderung einer Statusvariablen) wird ebenfalls vom Stack realisiert.

1.5.2.1. Java

Mit dem Java-SDK wird eine umfangreiche Beispielapplikation zur Verfügung

gestellt, die einen sehr guten Überblick über die Möglichkeiten gibt.

Leider ist mein Versuch, eine einfache UPnP-Applikation auf Basis des Java-SDK's auf dem *Networkprocessor[TINI]*, in Betrieb zu nehmen, aufgrund der beschränkten Speicherkapazitäten fehlgeschlagen.

Auf der Beispielapplikation des Java-SDK's 1.3 hatte ich mit **ethereal (http://www.ethereal.com/)** eine falsche Berechnung der UDP-Checksum bei UDP-Multicasts bemerkt, diese habe ich Satoshi Konno gemolden und in der Version 1.4 des Stacks, hat er diesen Fehler bereits behoben.

1.5.2.2. C++

Durch den Einsatz des umfangreichen Xerces XML-Parsers und C++ steigen die Abhängigkeiten und der Platzbedarf gegenüber dem Linux SDK von Intel stark an, wie die Ausgabe von ldd zeigt.

```
roger@ws1:~/CyberLink/sample/device$ ldd device
 libxerces-c.so.23 => /usr/lib/libxerces-c.so.23 (0x4002d000)
 libpthread.so.0 => /lib/tls/libpthread.so.0 (0x40314000)
 libstdc++.so.5 => /usr/lib/libstdc++.so.5 (0x40324000)
 libm.so.6 => /lib/tls/libm.so.6 (0x403dd000)
 libgcc_s.so.1 => /lib/libgcc_s.so.1 (0x40400000)
 libc.so.6 => /lib/tls/libc.so.6 (0x40408000)
 libicuuc.so.21 => /usr/lib/libicuuc.so.21 (0x40543000)
 libicudata.so.21 => /usr/lib/libicudata.so.21 (0x405af000)
 /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Somit kommt ein *Basic Device* ohne Debug-Symbole auf 472 kB zu stehen. Der Xerces XML-Parser benötigt jedoch alleine zirka 3 MB. Falls an anderen Stellen der Applikation ebenfalls XML zum Einsatz kommt, lohnt es sich aber trotzdem den funktionsreichen und stabilen Xerces XML-Parser für die ganze Applikation einzusetzen.

Wegen der Grösse des XML-Parsers habe ich bei Satoshi Konno nachgefragt, ob auch platzsparendere Parser unterstützt werden. Er sagt, dass er in Zukunft neben Xerces auch expat (http://expat.sourceforge.net/)(zirka 130kB) und libxml (http://www.xmlsoft.org/)(zirka 980kB) unterstützen will. Somit wird dem Cyberlink C++-Stack für eine erfolgreiche Zukunft auf embedded GNU/Linux-Systemen nichts mehr im Wege stehen.

1.5.3. Intel

Intel bietet auf seiner Website[UPnP-INTEL] eine Reihe Tools und Informationen zur UPnP Technologie zur Verfügung.

Die *Intel® Tools for UPnP Technologies* beinhalten neben Demoapplikationen auch einen Universellen Kontrollpunkt und Programme zur Überprüfung der Standardkonformität und zur Fehleranalyse an UPnP-Geräten.

Die *Intel® Authoring Tools for UPnP Technologies* bieten mit dem *Device Builder* die Möglichkeit fertige UPnP-Programme, -Kontrollpunkte und -Bibliotheken zu generieren. Dies ist in der Sprache *C* für die Platformen PocketPC, Windows und POSIX(UNIX, GNU/Linux) möglich. Zudem kann auch in *Java* und *C#* für .NET generiert werden.

Da nur nach dem POSIX-Standard[PASC] programmiert wurde sind die Abhängigkeiten zu anderen Programmbibliotheken auf GNU/Linux Systemen minimal:

```
roger@ws1:~$ ldd BinaryLight/BinaryLight
 libpthread.so.0 => /lib/tls/libpthread.so.0 (0x4002d000)
 libc.so.6 => /lib/tls/libc.so.6 (0x4003c000)
 /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Die Programm eines einfachen *Binary Light*'s, welches das Ein- und Ausschalten einer LED am Parallelport ermöglicht benötigt lediglich 55kB.

Die Verwendung des POSIX-Stacks auf der PowerPC Platform ist ohne Anpassungen leider nicht möglich. Da mehrere Byteoperationen im Stack vorkommen, kommt die Bytereihenfolge, welche ich in der Einleitung zu den Programmbibliotheken angesprochen habe, zum tragen. So wird zum Beispiel beim SSDP-Multicast die IP-Adresse in umgekehrter Reihenfolge veröffentlicht und somit ist das Herunterladen der Beschreibung durch die Kontrollpunkte verunmöglicht.

Leider ist mein Versuch, eine einfache UPnP-Applikation, auf Basis des generierten Java-UPnP-Gerätes mit den Intel Tools, auf dem *Networkprocessor[TINI]*, in Betrieb zu nehmen, aufgrund der beschränkten Speicherkapazitäten fehlgeschlagen.

Der Vorteil der Intel Tools besteht darin, das sich der Aufwand zur Erstellung eines UPnP-Geräts oder Kontrollpunkts verkürzt und sich dadurch die *Time to Market* entscheidend Verkürzen kann. Durch die Erstellung der Programme für eine vordefinierte Anwendung wird der Speicherbedarf zudem auf ein Minimum reduziert. Je dynamischer die Applikation jedoch sein soll(zum Beispiel nicht immer die selben Dienste oder Geräte), umso weniger eignen sich die generierten Intel-Stacks.

1.5.4. Microsoft

Als einer der massgebenden Entwickler der UPnP-Architektur hat Microsoft wichtige Überzeugungsarbeit für den Erfolg von UPnP geleistet. Mit *Windows Millenium* brachten sie das erste Produkt auf den Markt, da UPnP unterstützt.

Heute sind Programmbibliotheken zur Erstellung eines UPnP-Kontrollpunktes auf den Betriebsystemen *Windows Millemium*, *Windows XP* und *Windows CE .NET* vorhanden. Programmbibliotheken zur Entwicklung von UPnP-Kontrollpunkten stehen mit Ausnahme von Windows Millenium ebenfalls bereit.

Es werden die Programmiersprachen Microsoft *Visual Basic* und *C++* unterstützt.

Zudem lassen sich Kontrollpunkte in *Visual Basic Scripting(VBScript)*, welches sich einfach in HTML integrieren lässt, relativ einfach realisieren. Allerdings mit dem Nachteil, dass die Webseiten mit *VBScript* nur auf den erwähnten Microsoft Betriebssystemen lauffähig sind.

1.5.5. Allegro

Die *Allegro Software Development Corporation (http://www.allegrosoft.com/)* ist spezialisiert auf die Entwicklung von embedded Software wie XML-Parser, Webserver oder UPnP-Stacks. Hersteller wie 3com, Cisco Systems, D-Link, Nortel und Xer-

ox setzen in ihren Produkten Softwarekomponenten von Allegro ein.

Die Produkte von Allegro sind in ANSI-C geschrieben und sind ohne Anpassungen direkt auf den Echtzeitbetriebs-Systemen(RTOS, Real Time Operating System) *ATI Nucleus*, *Express Logic ThreadX* und *Win River VxWorks* einsetzen.

Mit dem Produkt *RomPlug Basic* (SSDP, Lokalisierung von UPnP Geräten) kommen 5kB zum RomPager Webserver, der je nach option zwischen 10kB und 35kB benötigt, hinzu. Somit lassen sich bereits die Basisfunktionen Lokalisierung und Beschreibung realisieren.

RomPlug Advanced beinhaltet einen Webserver, Webclient und RomXML, es benötigt zirka 100kB. So lassen sich auch komplexere UPnP-Geräte mit Diensten und Ereignissteuerung(GENA) realisieren.

Die Preise der Programmbibliotheken hängen vom Betriebssystem, der Applikation und der Stückzahlen des Gerätes ab.

Für komerzielle Produkte ist der Einsatz eines Allegro Produktes sehr gut denkbar, da der Speicherbedarf gering ist und eine professionelle Firma dahintersteht, die auch Support anbietet. So müssen nur wenig Ressourcen in die UPnP-Technologie investiert werden und man kann sich stärker auf das zu entwickelnde Produkt konzentrieren, was schliesslich zu einer schnelleren Lancierung des Produktes auf dem Markt führt.

Kapitel 2. Konnex (KNX)

Die *Konnex Association [KONNEX]* ist aus einem Zusammenschluss des Batibus Club International (BCI), der European Installation Bus Association (EIBA) und der European Home Systems Association (EHSA) entstanden. Sie hat die Feldbusse BatiBUS, EIB und EHS in einer neuen Spezifikation dem Konnex Standard (Kurzbezeichnung KNX) vereint. Der Standard unterstützt den Trend hin zum "intelligenten Haus", in dem die verschiedenen Gewerke aus HLK-, Licht- und Sicherheitstechnik auf einem gemeinsamen Kommunikationsnetzwerk integriert sind.

Der Konnex Standard basiert auf folgenden Prinzipien:

- Zusammenarbeit von Produkten verschiedener Hersteller im gleichen Netzwerk(Interworking)
- Für Konnex zertifizierte Produkte erfüllen die Konnex-Standards
- Konnex ist rückwärtskompatibel zum EIB, wobei KNX/EIB-Geräte miteinander nur im S-Mode kommunizieren

Am 29. März 2004 wurden vom *Europäisches Komitee für elektrotechnische Normung [CENELEC]* Teile der Konnex-Spezifikation in die EN50090 Standard Serie (*Home & Building Electronic Systems HBES*) aufgenommen. Dies ist ein weiterer, strategisch wichtiger, Schritt für die *Konnex Association*. Dadurch wurde die Basis geschaffen, das sich Konnex in der Gebäudeautomation(mindestens in Europa) als Standard langfristig durchsetzen kann.

2.1. Funktionsweise

Die KNX Netzwerk stellt eine leistungsfähige, verteilte Applikation dar. Die einzelnen Geräte stellen standardisierte Funktionsblöcke und Datenpunkte zur Verfügung. Diese werden durch einen der folgenden drei Betriebsmodi miteinander verbunden.

- **S-Mode (System-Mode)**

  Beim System-Mode können alle Elemente frei parametriert und Aktionen auf Gruppen- oder Einzeladressen zugewiesen werden. Dies erfolgt in der Regel mit der PC-basierenden *EIB-Tool-Software(ETS)*.

  KNX Installationen werden meist von spezialisierten Unternehmen geplant, parametriert und in Betrieb genommen.

- **E-Mode (Easy-Mode)**

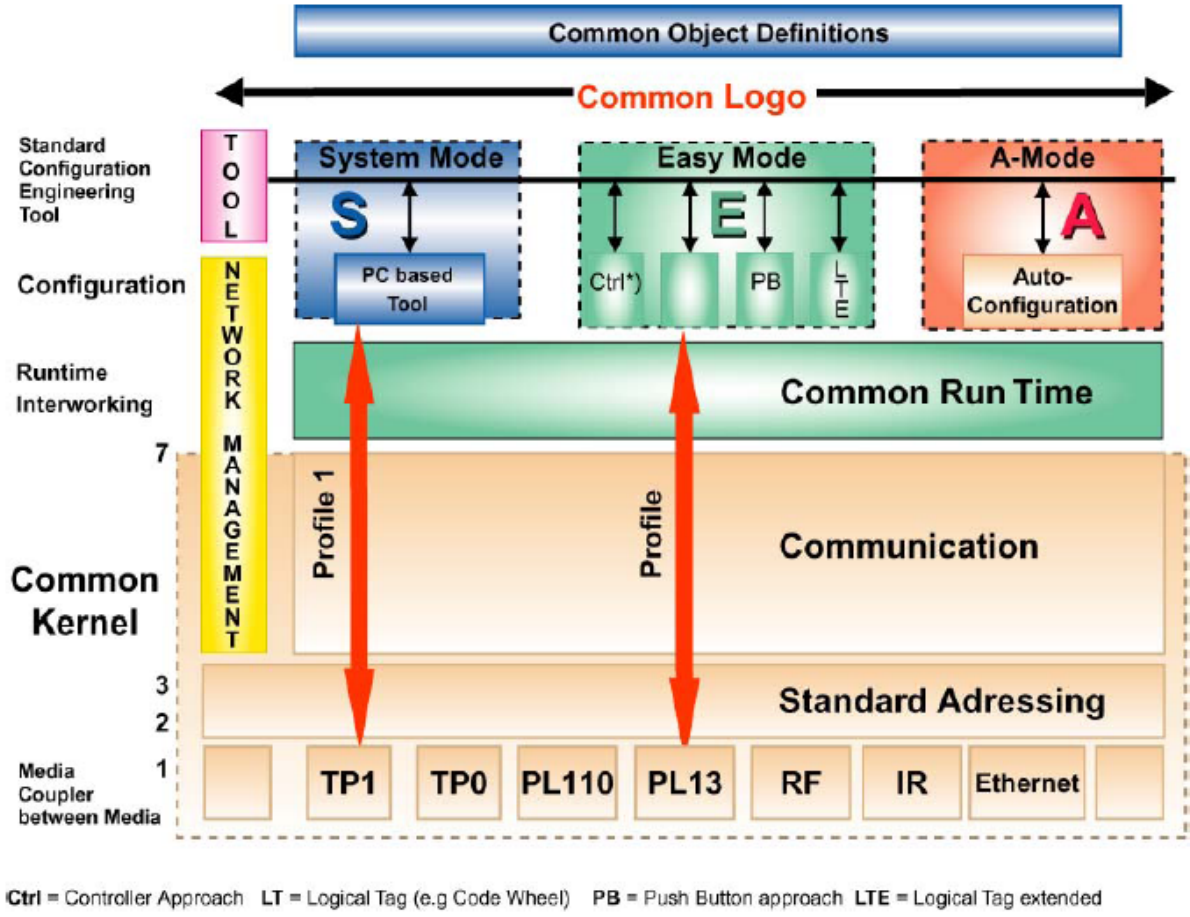
  Der Easy-Mode erlaubt einfachstes Konfigurieren ohne Tool-Software durch direkte Manipulationen an den Geräten. Somit kann die Konfiguration der KNX-Geräte teilweise direkt durch den Endbenutzer ausgeführt werden.

- **A-Mode (Automatic-Mode)**

  Im Automatic-Mode lassen sich die Geräte ohne Konfiguration direkt verwenden, sozusagen *Plug and Play*.

Die folgende Grafik aus der KNX Spezifikation zeigt den Aufbau der KNX Architektur.

Abbildung 2.1. KNX Architektur



Durch die Kombination der drei Feldbusse bietet KNX eine Reihe von physikalischen Median an. Neben den drahtgebundenen Leitungen sind heute bereits erste Produkte auf dem Markt, die mit Funk kommunizieren.

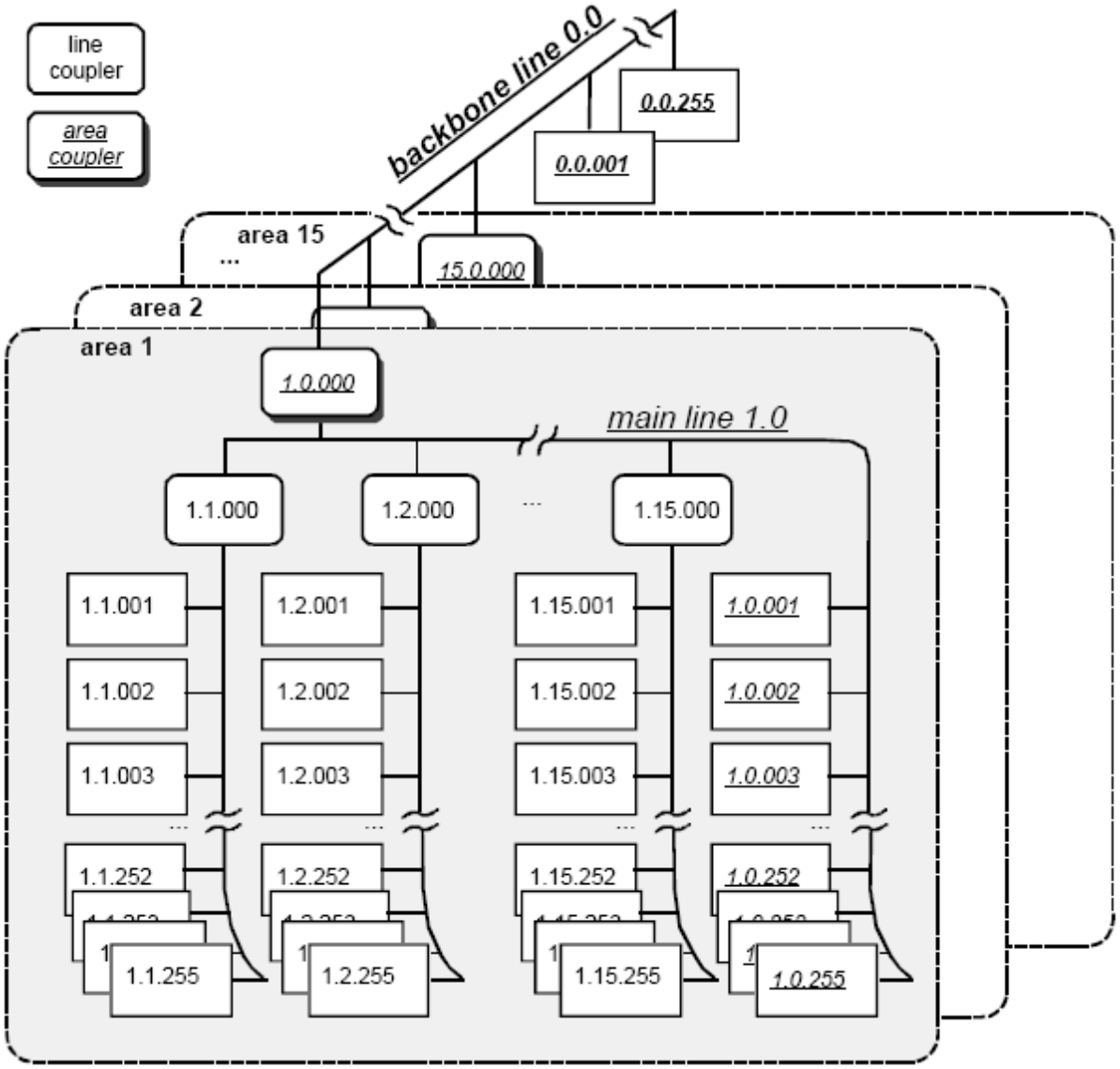
Tabelle 2.1. Physikalische Layer von KNX

Bezeichnung	Medium	Datendurchsatz	Bechreibung
TP0	geschirmte Busleitungen mit 2 verdrillten Adern	4.8 kbit/s	Vom BatiBUS stammendes Schnittstelle, Daten- und Strom übertragung über

Bezeichnung	Medium	Datendurchsatz	Bechreibung
			die selbe Leitung
TP1	geschirmte Busleitungen mit 2 verdrillten Adern	9.6 kbit/s	Standardschnittstelle von EIB, Daten- und Stromübertragung über die selbe Leitung
PL110	Stromnetz	1.2kbit/s	Modulation auf 110kHz
PL132	Stromnetz	2.4 kbit/s	Modulation auf 132kHz
RF	Funk(870MHz)	19.2 kbit/s	erste Geräte sind auf dem Markt
IR	Infrarot		zum heutigen Zeitpunkt noch nicht spezifiziert
Ethernet	Ethernet	10 Mbit/s, 100 Mbit/s	Mit EIBNet besteht die Möglichkeit KNX-Netzwerke über ein Ethernet miteinander zu Verbinden, sowie das Parametrieren von KNX-Geräten.

Die 16-bit Adressierung erlaubt es 65'536 KNX-Geräte in einem Netzwerk zu verwenden. Die logische Aufteilung erlaubt 254 Geräte auf einer Linie, 15 Linien in einer area und 15 area's in einem Netzwerk.

Abbildung 2.2. KNX Topologie



Für die Datenpunkte sind die folgenden Basis Datentypen definiert worden.

Tabelle 2.2. KNX Basis Datentypen

Datentyp	Beschreibung
Boolean	boolean, 0 = false, 1 = true
1-Bit Controlled	1 Kontrollbit mit 1-bit für den Modus (z.B. on/off)
3-Bit Controlled	1 Kontrollbit mit 3-bit für den Modus
Status with Mode	5 Kontrollbit mit 3-bit für den Modus
8-Bit Unsiged Value	1 Byte Integer Wert ohne Vorzeichen
2-Octet Unsigned Value	2 Byte Integer Wert ohne Vorzeichen
4-Octet Unsigned Value	4 Byte Integer Wert ohne Vorzeichen

Datentyp	Beschreibung
8-Bit Signed Value	1 Byte Integer Wert mit Vorzeichen
2-Octet Signed Value	2 Byte Integer Wert mit Vorzeichen
4-Octet Signed Value	4 Byte Integer Wert mit Vorzeichen
2-Octet Float Value	2 Byte Fliesskommazahl mit 2 Kommastellen
4-Octet Float Value	Fliesskommazahl im IEEE 754 Format
Time	Wochentag(0-7), Stunden, Minuten, Sekunden
Date	Tag, Monat, Jahr(0-99)
DateTime	Datum, Zeit, mit zusätzlichen Bits zur Angabe der Qualität
Access	4 Byte Datentyp für eine Zugangsauthentifizierung
Character Set	1 Zeichen des ASCII oder ISO8859-1 Zeichsatzes
String	14 Byte String ASCII oder ISO8859-1

Zudem sind einige weiterer anwendungsspezifische Datentypen definiert. Herstellerspezifische Datentypen sind ebenfalls möglich, dies muss aber mit einer Inkompatibilität für den entsprechenden Parameter bezahlt werden.

dataframe

Es sind Punkt zu Punkt (Unicast) und Gruppenaddressierung (Multicast) möglich.

2.2. Geräte

Es ist eine riesige Menge an Geräten von verschiedenen Herstellern verfügbar.

„KNX ist ein Paradebeispiel für herstellerübergreifende Standards!“

Durch die konsequente Weiterentwicklung des KNX Standards(z.B. Funk) kommen laufen neue Produkte auf den Markt.

Kapitel 3. Prototyp

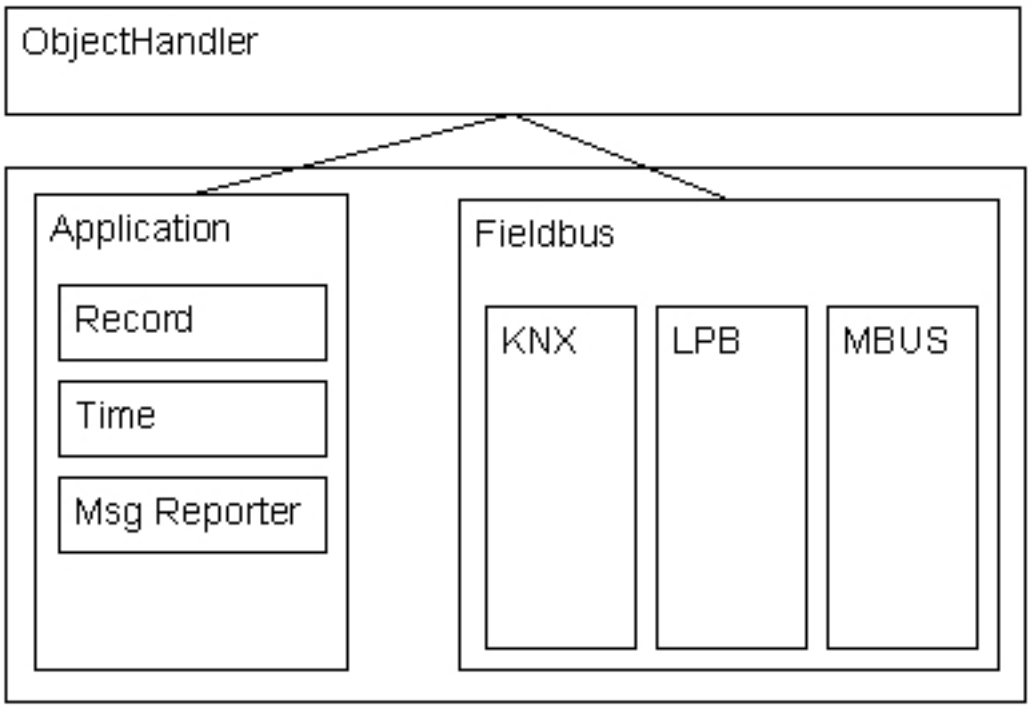
Mein Arbeitgeber, die Siemens Building Technologies, hat mir ein PowerPC mit 16MB Flash-ROM und 32MB RAM zur Verfügung gestellt.

Basierend auf dem Betriebssystem GNU/Linux ist eine TP1-Schnittstelle auf den Konnex-Feldbus vorhanden.

Die bestehende Applikation ist hauptsächlich in C++ geschrieben und verwendet an mehreren Stellen eine Abstraktionsschichten auf das Betriebssystem.

Über den so genannten Objecthandler kann auf den gewünschten Feldbus zugegriffen werden. Er ermöglicht den einheitlichen Zugriff auf die drei Feldbussysteme KNX, LPB und MBUS.

Abbildung 3.1. vorhandene Softwarearchitektur



Durch die Softwarearchitektur bzw. die Abstraktion der drei Feldbusse KNX, LPB und MBUS sind die Prozesssignale auf der Applikationsseite momentan noch nicht verfügbar. Dies aus dem einfachen Grund, dass in der Applikation, wo diese Hard- und Software eingesetzt wird, die Prozesssignale auf der Applikationsseite nicht benötigt werden und so im Feldbussubsystem verarbeitet werden.

Die Parameter der KNX Funktionsblöcke lassen sich über den Objecthandler mit der Busadresse(z.B. 1.2.250), dem Objektindex des Funktionsblockes, und der Property-ID des Datenpunktes ansprechen.

Als Feldbusgeräte steht mir ein Heizungsregler vom Typ RMU760 und ein Universal-

regler RMU 710 zur Verfügung. Diese werden im Prototyp als UPnP-Geräte repräsentiert.

Zu Beginn der Arbeit wollte ich neben den Reglern auch noch einen einfachen KNX-Lichtschalter implementieren. Da dieser über Gruppenadressen im S-Mode ein- und ausgeschaltet wird, habe ich durch die Applikationsarchitektur keine Möglichkeit, diese KNX-Telegramme in die UPnP-Applikation zu bekommen bzw. anzusteuern.

3.1. Konzept UPnP<=>KNX

Durch die bestehende Applikationsarchitektur bedingt habe ich keine Möglichkeit, Ereignisse auf dem KNX-Netzwerk direkt in die UPnP-Applikation zu leiten und als UPnP-Ereignisse weiterzuversenden. So müsste ich die Parameterwerte der KNX-Geräte in einem bestimmten Intervall lesen und so bei Änderungen künstlich ein UPnP-Ereigniss erzeugen. Da dies der Idee der Ereignisse grundsätzlich widerspricht, werde ich dies nicht implementieren.

Alle Werte, die via UPnP gesteuert oder gelesen werden sind einzelne Datenpunkt-Parameter der KNX-Geräte.

Der Prototyp repräsentiert ein standardisiertes UPnP HVAC_System. Dieses kann mehrere HVAC_ZoneThermostat beinhalten, welche jeweils einen Heizungsregler repräsentieren.

```
<?xml version="1.0" encoding="utf-8"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
 <specVersion>
  <major>1</major>
  <minor>0</minor>
 </specVersion>
 <device>
  <deviceType>urn:schemas-upnp-org:device:HVAC_System:1</deviceType>
  <friendlyName>HVAC System</friendlyName>
  <manufacturer>Roger Meier</manufacturer>
  <manufacturerURL>http://www.bufferoverflow.ch</manufacturerURL>
  <modelDescription>Diplomarbeit 2004 UPnP-KNX</modelDescription>
  <modelName>Prototyp Diplomarbeit</modelName>
  <modelNumber>1</modelNumber>
  <serialNumber>0000001</serialNumber>
  <presentationURL>/presentation.html</presentationURL>
  <UDN>uuid:2ea6c3ac-b14e-413c-a784-16c7dd1ea492</UDN>
  <serviceList>
   <service>
    <serviceType>urn:schemas-upnp-org:service:HVAC_UserOperatingMode:1</serviceType>
    <serviceId>urn:upnp-org:serviceId:SystemUserMode</serviceId>
    <SCPDURL>SystemUserMode/scpd.xml</SCPDURL>
    <controlURL>SystemUserMode/control</controlURL>
    <eventSubURL>SystemUserMode/event</eventSubURL>
   </service>
  </serviceList>
  <deviceList>
    <!-- mehrere HVAC_ZoneThermostat Geräte -->
  </deviceList>
 </device>
</root>
```

Der Dienst HVAC_UserOperatingMode muss im HVAC_System gemäss Standard ebenfalls vorhanden sein. Um generelle Einstellungen für alle Zonen festzulegen.

Für jeden Heizungsregler kann nun eine Gerätebeschreibung erstellt und in der deviceList des HVAC_System integriert werden. Da die Funktionalität der Heizungsregler

sehr unterschiedlich ist müssen die Dienste für jedes Modell separat definiert und implementiert werden.

```
<device>
 <deviceType>urn:schemas-upnp-org:device:HVAC_ZoneThermostat:1</deviceType>
 <presentationURL>/presentation.html</presentationURL>
 <friendlyName>Heizung Wohnzimmer</friendlyName>
 <manufacturer>Siemens Building Technologies, HVAC Products</manufacturer>
 <manufacturerURL>http://www.sbt.siemens.com/hvp/</manufacturerURL>
 <modelDescription>Modularer Heizungsregler</modelDescription>
 <modelName>RMH760</modelName>
 <modelNumber>1_1</modelNumber>
 <modelURL>http://www.landisstaefa.com/prd/d/prd_sta_hva_syn_700.asp</modelURL>
 <serialNumber>0000001</serialNumber>
 <UDN>uuid:knx_02.01.250_rmh760</UDN>
 <serviceList>
  <service>
   <serviceType>urn:schemas-upnp-org:service:HVAC_UserOperatingMode:1</serviceType>
   <serviceId>urn:upnp-org:serviceId:ZoneUserMode</serviceId>
   <SCPDURL>knx/rmh760/ZoneUserMode/scpd.xml</SCPDURL>
   <controlURL>knx/rmh760/ZoneUserMode/control</controlURL>
   <eventSubURL>knx/rmh760/ZoneUserMode/event</eventSubURL>
  </service>
  <service>
   <serviceType>urn:schemas-upnp-org:service:TemperatureSetpoint:1</serviceType>
   <serviceId>urn:upnp-org:serviceId:HeatingSetpoint</serviceId>
   <SCPDURL>knx/rmh760/HeatingSetpoint/scpd.xml</SCPDURL>
   <controlURL>knx/rmh760/HeatingSetpoint/control</controlURL>
   <eventSubURL>knx/rmh760/HeatingSetpoint/event</eventSubURL>
  </service>
  <service>
   <serviceType>urn:schemas-upnp-org:service:TemperatureSensor:1</serviceType>
   <serviceId>urn:upnp-org:serviceId:ZoneTemperature</serviceId>
   <SCPDURL>knx/rmh760/ZoneTemperature/scpd.xml</SCPDURL>
   <controlURL>knx/rmh760/ZoneTemperature/control</controlURL>
   <eventSubURL>knx/rmh760/ZoneTemperature/event</eventSubURL>
  </service>
</serviceList>
</device>
```

Die UDN der eingebetteten Thermostat Zonen repräsentiert die physikalische KNX-Adresse des Reglers und den Typ des Gerätes. Somit kann die Applikation zur Laufzeit auf das entsprechende Gerät zugreifen.

Somit muss lediglich die Beschreibungsdatei angepasst werden um weitere Geräte via UPnP darzustellen. Diese könnte nach einem Geräte Suchlauf automatisch erstellt werden. Die beiden Felder serialNumber und friendlyName könnten ebenfalls via KNX aus den Geräten gelesen und entsprechend gesetzt werden.

Der Dienst APCI_PHYSADDR_SERNO_READ auf dem Konnex-Netzwerk würde zudem die Möglichkeit bieten, anhand der Seriennummer des Gerätes die KNX-Adresse ausfindig zu machen. Dadurch müsste die Adresse in der Beschreibungsdatei nicht mehr angegeben werden und könnte automatisch durch die Applikation ermittelt werden.

3.1.1. Raumbetriebsart

Der UPnP Dienst HVAC_UserOperatingMode entspricht der Raumbetreibsart der beiden Regler RMH760 und RMU710.

Die Einstellungen der Raumbetriebsart haben beim RMH760 den Objektindex 41. Die Vorgabe der Betriebsart wird über das Property 238 eingestellt, dies wird über

die UPnP Funktionen GetModeTarget und SetModeTarget eingestellt. Die aktuelle Betreibsart entspricht nicht immer der Vorgabe, wenn Zeitschaltprogramme aktiv sind haben diese Vorrang. Über das Property 51 kann die aktuelle Betreibsart gelesen werden, sie entspricht der UPnP Funktion GetModeStatus. Beide Property's sind vom Datentyp DPT_HVACMode_Z bzw. eVpbN8Z8.

Auf den Reglern können die Werte Auto, Komfort, Prekomfort(Standby), Economy und Schutzbetrieb eingestellt werden.

Gemäss UPnP Standard sind die Betreibsarten Off, HeatOn oder CoolOn und ein Herstellerspezifischer Modus zwingend zu implementieren. Zudem sind die folgenden Modi als optional definiert: AutoChangeOver, AuxHeatOn, EconomyHeatOn, EmergencyHeatOn, AuxCoolOn, EconomyCoolOn, BuildingProtection, EnergySavingsHeating, EnergySavingsCooling.

Moderne Regler, wie zum Beispiel der RMH760 lassen sich allerdings gar nicht ausschalten, es ist lediglich ein Schutzbetrieb möglich. Da das Ausschalten unter Umständen den Heizkessel in Mitleidenschaft ziehen könnte.

Tabelle 3.1. Mapping Raumbetriebsart RMH760

Regler Mode, Enumerationswert	**UPnP Mode**
Auto, 0	AutoChangeOver
Komfort, 1	HeatOn
Prekomfort(Standby), 2	StandbyHeatOn (herstellerspezifischer)
Economy, 3	EconomyHeatOn
Schutzbetrieb(BldgProtect), 4	BuildingProtection,Off

Der UPnP-Mode Off lässt sich als Vorgabe mittels SetModeTarget einstellen, die Funktionen GetModeTarget und GetModeStatus geben jedoch immer den Wert BuildingProtection zurück.

Tabelle 3.2. Mapping Raumbetriebsart RMU710

Regler Mode, Enumerationswert	**UPnP Mode**
Auto, 0	
Komfort, 1	
Prekomfort(Standby), 2	
Economy, 3	
Schutzbetrieb(BldgProtect), 4	Off

3.1.2. Raumtemperatur

Die Raumtemperatur kann beim RMU760 über den Objektindex 24 und das Property 254 gelesen werden.

3.1.3. Raumtemperatur Sollwert

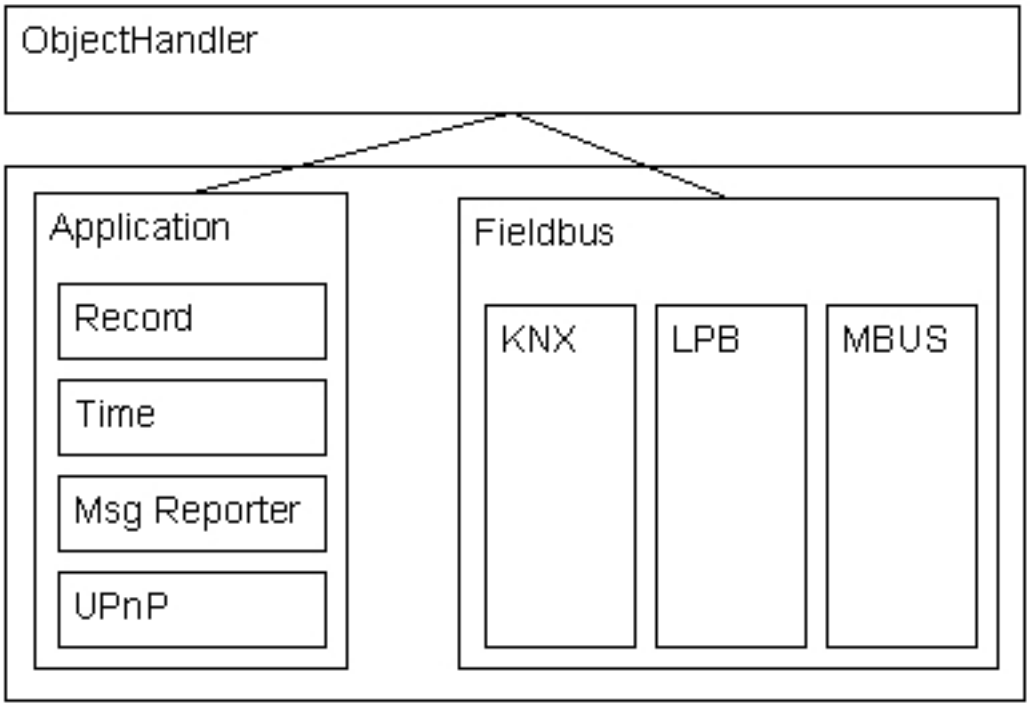
Der UPnP Dienst TemperaturSetpoint ist für die Einstellung der Solltemperatur vorgesehen. Bei den Reglern können allerdings nur Solltemperaturwerte für die verschiedenen Betriebsarten eingestellt werden.

So stellt sich die Frage welche Solltemperatur via UPnP eingestellt werden soll. Die Solltemperatur des aktuellen Betreibsmodus oder des Betreibsmodus der vorgegeben ist eingestellt werden?

3.2. Implementation

Das UPnP-Gerät wird im Applikationteil der Softwarearchitektur angesiedelt und implementiert. Dies würde auch eine Erweiterung der UPnP-Funktionalität für die beiden anderen Feldbusse LPB und MBUS ermöglichen.

Abbildung 3.2. Integration in die bestehende Softwarearchitektur



Die einzelnen Applikationen und Subsysteme werden alle zuerst über eine Singelton-Klasse instanziert und anschliessend in der notwendigen Reihenfolge gestartet.

3.2.1. Entwicklungsumgebung

Die komplette Entwicklungsumgebung basiert auf frei erhältlicher Software. Auf dem Betriebsystem Debian GNU/Linux wird das *Embedded Linux Development Kit (ELDK)* der Firma DENX Software Engineering (http://www.denx.de/) eingesetzt. Dieses stellt den *GNU C Compiler 3.2.2* mit allen benötigten Programmen(Linker, Binutils) für die PowerPC Platform bereit, somit können auf der Intel-Platform Programme für den PowerPC erstellt werden.

Nachdem der Target die IP-Adresse mittels DHCP-Server erhalten hat, wird der Linux-Kernel via tftp geladen und gestartet. Anschliessend wird das Filesystem via Network File System(NFS) vom Entwicklungshost eingehängt. Somit muss das Programm nicht jedesmal in den Flashspeicher des Targets programmiert werden, um es zu Testen.

Als Entwicklungsumgebung wird Kdevelop3 (http://www.kdevelop.org/) einge-

setzt. Mit dem GNU Debugserver wird eine IP-Verbindung zum Target hergestellt und so kann der Programmcode auf dem Target schritt für schritt ausgeführt werden.

3.2.2. UPnP Programmbibliothek

Durch die geringen Ressourcen der Hardware und dem Einsatz des Betriebssystemes GNU/Linux kommt bei den UPnP-Programmbibliotheken nur das *Linux SDK* oder ein mit dem *Intel Device Builder* generierte Programmbibliothek in Frage.

Die generierten Programmbibliotheken von Intel sind auf der PowerPC-Platform ohne Anpassungen nicht lauffähig. Zudem sind die UPnP Gerätebeschreibungen im generierten Stack fix und lassen sich nicht extern in einer Datei ablegen. Dies wäre zum Beispiel für das setzen der Seriennummer in der Gerätebeschreibung von Vorteil.

Das Linux SDK ist vom Ressourcenbedarf her ein wenig anspruchsvoller. Die einzelnen Teile der Programmbibliotheken lassen sich jedoch deaktivieren und die maximale Anzahl Threads ist ebenfalls einstellbar. Der erfolgreiche Einsatz des Linux SDK in komerziellen Produkten zeigt, dass die Stabilität und Zuverlässigkeit der Programmbibliothek auf embedded Systemen gewährleistet ist. Weiter können die Gerätebeschreibungen aus einem File geladen oder in der Applikation dynamisch erstellt werden.

Aus diesen Gründen habe ich mich für das Linux SDK entschieden, da dies mehr Flexibilität bietet und zudem auch in anderen Produkten bereits eingesetzt wird.

Das Erstellen der Programmbibliothek für eine andere Hardwareplatform wird vom Linux SDK direkt unterstützt. So genügen die folgenden Befehle um die Bibliotheken für die PowerPC-Hardware zu erstellen:

```
export CROSS=ppc_8xx
export TARGET=ppc_8xx
make CLIENT=0
```

Mit dem Parameter CLIENT=0 wird die Funktionalität zur Erstellung eines Kontrollpunktes deaktiviert, womit sich weiterer Speicherplatz einsparen lässt.

Die erstellten Bibliotheken sind nach erfolgreicher Kompilierung im Verzeichnis `libupnp-1.2.1/upnp/bin/ppc_8xx/` verfügbar. Diese werden ins Verzeichnis `/usr/lib/` auf der Zielplatform und der Crosscompile-Umgebung kopiert. Die Gesamtgrösse der drei Bibliotheken libixml.so, libthreadutil.so und libupnp.so beläuft sich nun auf rund 188 kB.

Da die bestehende Applikation in ANSI-C++ geschrieben ist, und den Compilerswitch -ansi verwendet, musste ich in allen Headerfiles des Linux SDK die Kommentare ANSI konform umschreiben. Nur so liess sich die Programmbibliothek in der bestehenden Applikation einsetzen.

Schliesslich musste ich feststellen, dass sich meine Testapplikation auf der Zielhardware nicht gleich verhaltet, wie auf der Intel-Platform. Der Kontrollpunkt konnte nicht alle Beschreibungsdateien meines Gerätes laden. Zuerst habe ich das Problem bei

der geringeren Rechenleistung vermutet, da die Anfragen für die Beschreibungsdateien sehr kurz aufeinander folgen und das Gerät diese eventuell nicht genügend schnell liefern kann. So habe ich die Anzahl Threads und den Cache des Webservers in der Programmbibliothek erhöht. Dies hat jedoch auch nicht zum gewünschten Resultat geführt. Schliesslich konnte ich durch das Entfernen von Diensten in der Beschreibungsdatei feststellen, dass die URL's der Bechreibungsdateien(SCPDURL) zu lange waren und der integrierte Webserver des Gerätes die Anfragen des Kontrollpunktes nicht verarbeiten konnte. Den Ursprung dieses Problems habe ich nicht weiter untersucht.

3.2.3. IP-Adressierung

Die IP-Adressierung erfolgt über einen DHCP-Server oder eine fest konfigurierte IP-Adresse. Da heute eine Router mit integriertem DHCP-Server sozusagen Standard ist, verzichte ich auf die Unterstützung von AUTO-IP.

Auf der Netzwerkkarte muss Multicast explizit aktiviert werden.

```
root@nscu:~#ifconfig eth0 allmulti multicast
```

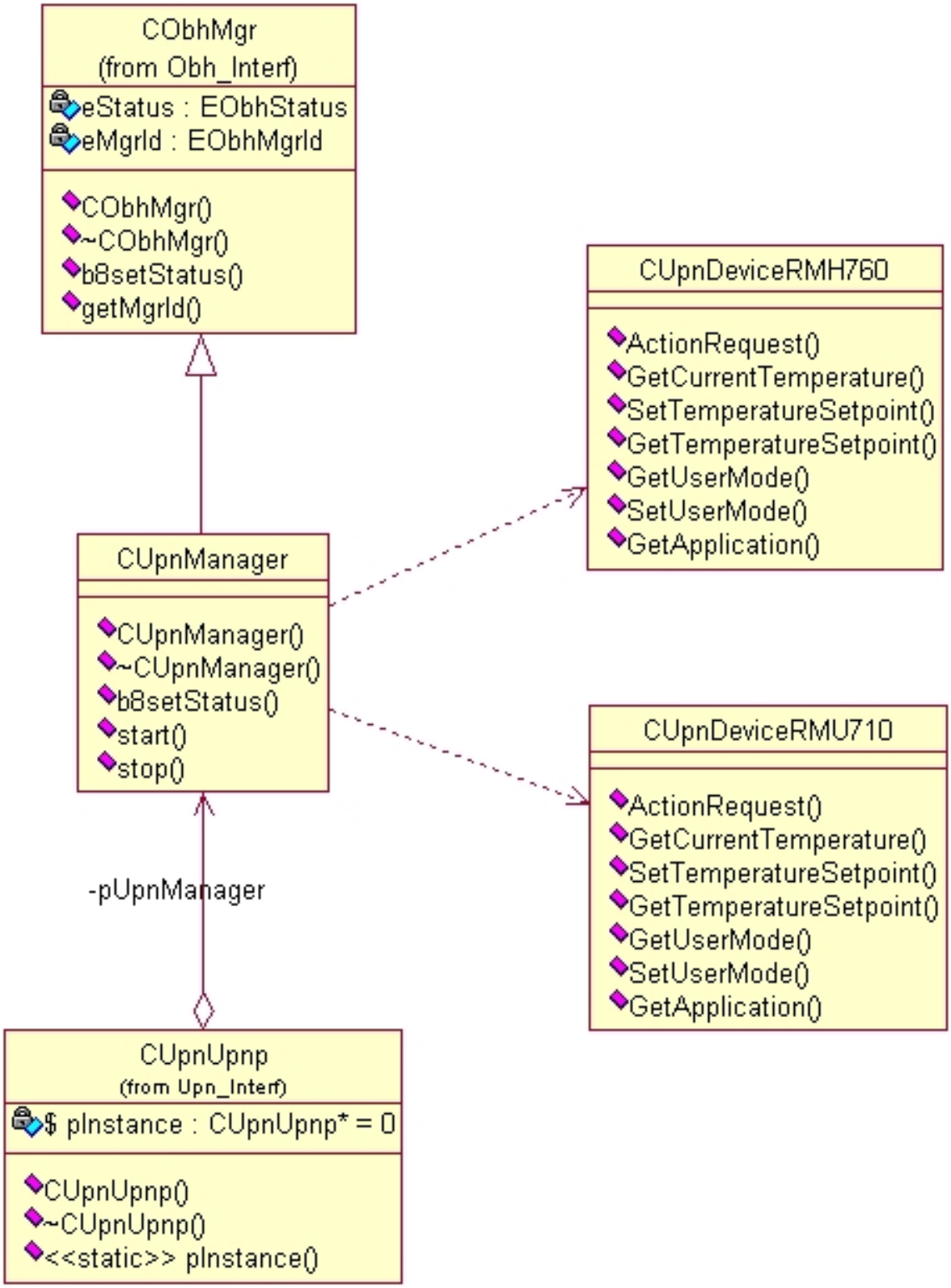
Der folgende Eintrag in der Routing-Tabelle muss ebenfalls eingerichtet werden, damit Multcast-Anfragen empfangen und gesendet werden können.

```
root@nscu:~#route add -net 239.0.0.0 netmask 255.0.0.0 eth0
```

3.2.4. Applikation

Die von der Singelton-Klasse CUpnUpnp instanzierte Klasse CUpnManager ist von der ObjecthandlerManager Klasse abgeleitet. Sie übernimmt die Funktionen zum Starten und Stoppen des UPnP-Gerätes.

Abbildung 3.3. Aufbau UPnP Applikation



3.2.5. Test

Anhang A. Hilfsmittel

A.1. Software

Betriebssystem

freie Software

XML-Werkzeuge

- ant

  ein sogenanntes Build-Tool ähnlich wie make, die

- xmllint

  Validieren von XML-Dateien

•
•

http://www.ethereal.com/

http://docbook.sourceforge.net/

A.1.1. Prototyp

A.1.2. Workstation

A.2. Hardware

Anhang B. Ablaufplan

Tabelle B.1. Ablaufplan

Phase	Thema	Termin	Ziele/Resultate
1	**Analyse**		
1.1	Einarbeiten in die Thematik von UPnP	mitte Mai	Erarbeitung des benötigten Grundwissens, Dokumentation der Funktionsweise und Aufzeigen der Möglichkeiten von UPnP
1.2	Einarbeiten in die Thematik von Konnex	ende Mai	Erarbeitung des benötigten Grundwissens
1.3	Vergleich der UPnP-Programbibliotheken, Implementation eines UPnP-Devices zur Analyse der UPnP-Programbibliotheken	ende Mai	Grundlage für die Wahl der Programmbibliothek
Milestone	Präsentation Problemanalyse	22. Mai 2004	
2	**Lösungskonzept ausarbeiten**		
2.1	Möglichkeiten der eingesetzten Hard- und Software prüfen	mitte Juni	Hardware, Software und Konnex-Programmbibliothek kennenlernen
2.2	Wahl der Konnex Feldbusgeräte	mitte Juni	Basis zur Wahl der Dienste die implementiert werden sollen
2.3	Entscheid welche Dienste im Prototypen implementiert werden	mitte Juni	Grundlage für Softwaredesign
2.4	Entscheid UPnP-Programmbibliothek	ende Juni	Grundlage für die Realisierung
Milestone	Präsentation Lösungskonzept	19. Juni 2004	
3	**Konzept UPnP <=> Konnex**		
3.1	Ausarbeiten Konzept	ende Juli	Wie können Felbuskomponenten auf UPnP-Dienste abgebildet werden.
4	**Realisierung**		
4.1	Softwaredesign	ende Juli	
4.2	Implementation	ende Juli	
4.3	Test	mitte August	

Phase	Thema	Termin	Ziele/Resultate
Milesto-ne	Abgabe Diplomarbeit	28.August 2004	
	Vorbereitung Taxation		
Milesto-ne	Taxation	September	

Bibliographie

Organisationen

[GNU] *GNU Operating System - Free Software Foundation (http://www.gnu.org)*.

[IETF] *The Internet Engineering Task Force (http://www.ietf.org/)*.

[ISO] *International Organization for Standardization (http://www.iso.org)*.

[ANSI] *American National Standards Institute (http://www.ansi.org)*.

[PASC] *Portable Application Standards Committee der IEEE (http://www.pasc.org)*.

[W3C] *World Wide Web Consortium (http://www.w3c.org/)*.

[IANA] *The Internet Assigned Numbers Authority (http://www.iana.org/)*.

[SBT] *Siemens Building Technologies (http://www.sbt.siemens.com/home.asp)*.

[Linux] *Linux Kernel (http://www.kernel.org)*.

[GPL] *GNU Public License (http://www.gnu.org/gpl)*.

[GLIBC] *GNU C Library (http://www.gnu.org/software/libc/libc.html)*.

[TINI] *TINI - Tiny InterNet Interfaces (http://www.maxim-ic.com/TINI)*.

UPnP

[UPnP-Forum] *UPnP™ Forum (http://www.upnp.org/)*.

[UPnP-IC] *UPnP™ Implementers Corporation (http://www.upnp-ic.org/)*.

[UPnP-1.0] *UPnP™ Device Architecture 1.0 (http://www.upnp.org/download/UPnPDA10_20000613.htm)*.

[LIBUPNP] *Open Source Linux SDK for UPnP Devices 1.2.1 (libupnp) (http://upnp.sourceforge.net/)*.

[CYBERGARAGE] *CyberLink UPnP™ Stack for C++ and Java (http://www.cybergarage.org/)*. Satoshi Konno.

[UPnP-INTEL] *Intel® software for UPnP™ technology (http://www.intel.com/technology/UPnP/)*.

[JeronimoWeast2003] *UpnP Design by Example: A Software Developer's Guide to Universal Plug and Play with CDROM*. Michael Jeronimo und Jack Weast.

Copyright © 2003. 0971786119. Intel Press.

[Richard2002] *Service and Device Discovery*. Golden G. Richard. Copyright © 2002. 0071379592. McGraw-Hill Professional Publishing.

IETF Standards

[RFC2131] *Dynamic Host Configuration Protocol (http://www.ietf.org/rfc/rfc2131.txt)*.

[RFC2608] *Service Location Protocol, Version 2 (http://www.ietf.org/rfc/rfc2608.txt)*.

[RFC2279] *UTF-8, a transformation format of ISO 10646 (http://www.ietf.org/rfc/rfc2279.txt)*.

IETF Entwürfe

[HTTP-UDP] *Multicast and Unicast UDP HTTP Messages (http://www.upnp.org/download/draft-goland-http-udp-04.txt)*.

[SSDP] *Simple Service Discovery Protocol/1.0 (http://www.upnp.org/download/draft_cai_ssdp_v1_03.txt)*.

[GENA] *General Event Notification Architecture (http://www.upnp.org/download/draft-cohen-gena-client-01.txt)*.

[ZEROCONF] *Zero Configuration Networking (Zeroconf) (http://www.zeroconf.org/)*.

W3C Standards

[SOAP] *Simple Object Access Protocol (SOAP) 1.1 (http://www.w3.org/TR/2000/NOTE-SOAP-20000508/)*.

[XML] *Extensible Markup Language (XML) 1.0 (Third Edition) (http://www.w3.org/TR/2004/REC-xml-20040204/)*.

Konnex

[KONNEX] *Konnex Association (http://www.konnex.org/)*. Konnex Association Bessenveldstraat, 5 1831 DIEGEM BELGIUM.

[CENELEC] *European Committee for Electrotechnical Standardization (http://www.cenelec.org/)*.

[EIBASWISS] *EIBA Swiss (http://www.eibaswiss.ch/)*. EIBA Swiss Postfach 7190 8023 Zürich .

[EIBGrundlagen] *Handbuch Gebäudesystemtechnik*. Grundlagen. 4. überarbeitete Auflage. Copyright © 1997. Zentralverband Elektrotechnik- und Elektronikin-

dustrie(ZVEI). Zentralverband der Deutschen Elektrohandwerke(ZVEH).

[EIBAnwendungen] *Handbuch EIB Haus- und Gebäudesystemtechnik*. Anwendungen. 2. Auflage. Copyright © 2002. Zentralverband Elektrotechnik- und Elektronikindustrie(ZVEI). Zentralverband der Deutschen Elektrohandwerke(ZVEH).

Stichwortverzeichnis

DocBook

Dieses Dokument wurde in DocBook, einem XML-basierter Standart der OASIS Organisation geschrieben. Durch die strikte Trennung von Inhalt und Formatierung ähnlich wie bei LaTeX ist das Quelldokument Mediumneutral. Das Erstellen eines Dokumentes in DocBook ermöglicht ein das Erstellen weiterer Formate wie HTML, PDF, Postscript, Microsoft HTMLHelp, UNIX man pages, JavaHelp, TeX, LaTeX, TeXinfo, RTF.

Weitere Informationen zu DocBook

- OASIS Organisation (http://www.oasis-open.org/) OASIS Organisation
- DocBook: The Definitive Guide (http://www.docbook.org/tdg/en/html/) (auch in Buchform erhältlich)
- Dokumentationen mit DocBook-XML (http://www.goshaky.com/docbook-tutorial/) von Lars Trieloff
- Using DocBook at CERN (http://xml.web.cern.ch/XML/goossens/dbatcern/index.html)
- DocBook XSL: The Complete Guide (http://www.sagehill.net/docbookxsl/)
- Docbook Frequently Asked Questions (http://www.dpawson.co.uk/docbook/index.html)

Abbildung 10. Ablauf der Publikation mit docbook