Hochschule für Technik und Wirtschaft Dresden

Fakultät Informatik/Mathematik

Dokumentation

Forschungsseminar Sensornetze

Projekt: Regelverarbeitung und Web-Plattform

Enrico Uhlig
Robert Krampe

Wintersemester 2012/2013
03.03.2013

Inhalt

1 Einführung

1.1 Motivation

Dieses Forschungsseminar hat den Aufbau einer drahtlosen sensorbasierten Hausautomatisierung zum Ziel. Die Betonung liegt - im Hinblick auf dieses Teilprojekt - auf “Automatisierung”. Mit einem Sensornetz soll eine weitgehende Erleichterung im Alltag erreicht werden. Für diese Zielstellung und zur sinnvollen Verwendung eines umfangreichen Sensornetzes ist eine Regelverarbeitung von grundlegender Bedeutung. Sensoren und Aktoren müssen interagieren können. Neben diesem Trivialfall des einfach An- und Ausschaltens einer Lampe, sollen mithilfe eines Sensornetzes auch weiterführende Aufgaben automatisiert werden. Angefangen von der zeitlichen Steuerung der Heizungsanlage bis hin zu einer intelligenten Steuerung der Alarmanlage, sofern sich kein Bewohner mehr in Haus oder Wohnung befindet.

1.2 Ziel

Folgende Anforderungen soll ein Regelverarbeitungssystem mindestens erfüllen:

- Steuerung von Aktoren über Sensoren (Schalter oder Schwellwerte)
- Festlegen von Bedingungen bei deren Eintreten vordefinierte Aktionen ausgeführt werden
- zeitgesteuertes Auslösen von Aktionen

Die Bearbeitung des Regelwerks soll für erfahrene Benutzer schnell und einfach möglich sein, aber auch unerfahrenen Benutzern alle Möglichkeiten über eine grafische Benutzeroberfläche zur Verfügung stellen.

2 Durchführung

2.1 Verwandte Arbeiten

Um die am besten für das Projekt geeignete Umsetzung eines Regelwerks zu finden, wurde im Vorfeld umfangreiche Recherchearbeit geleistet. Dabei spielten die folgenden gängigen Hausautomatisierungssysteme eine Rolle: FHEM, Homeputer und IP-Symcom.

2.1.1 FHEM

FHEM ist ein sehr verbreitetes Heimautomatisierungssystem welches zur freien Verfügung unter GPL veröffentlicht wurde. [1] [2]

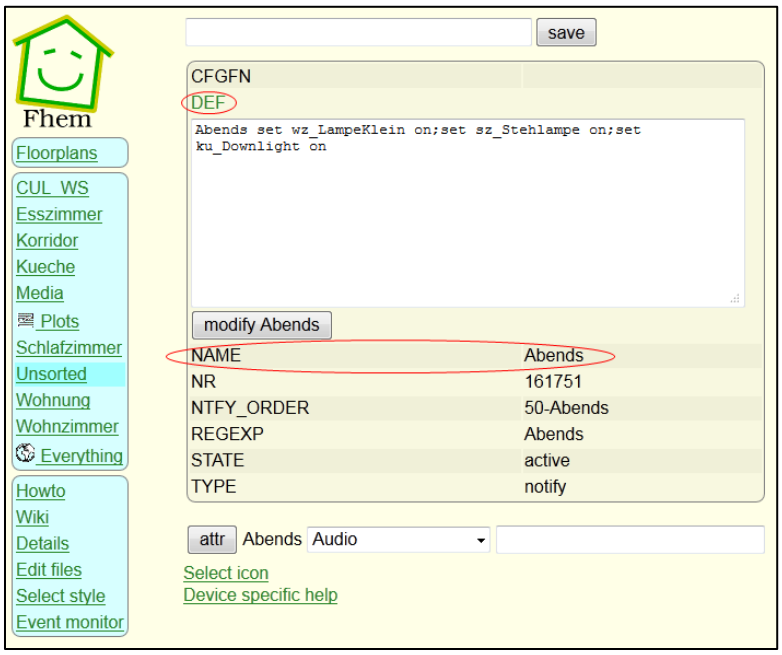

Abbildung 2.1: Screenshot der FHEM-Weboberfläche [2]

FHEM ist in der Programmiersprache Perl geschrieben. Deshalb gibt es auch die Möglichkeit, Perlcode aus FHEM heraus auszuführen, wobei Perlkommandos immer in geschweifte Klammern gesetzt werden. Der vermutlich am häufigsten verwendete Befehl ist der If-Befehl, der sich zur Formulierung von Regeln nach folgendem Muster eignet:

```
{ if (Bedingung) {Befehl 1} else {Befehl 2} }
```

Aus Perl heraus lassen sich auf diese Weise FHEM-Befehle absetzen. Ein Nachteil ist, dass die für den Endanwender vergleichsweise komplizierte Perlsyntax beherrscht werden muss und die Regeln per Hand als Befehle in der Kommandozeile (am besten mit Hilfe eines geeigneten Editors) oder alternativ direkt in der Konfigurationsdatei „fhem.cfg“ zu formulieren sind. Das folgende Beispiel definiert eine Regel, die einen Befehl ausführen soll, sobald der nach dem Schlüsselwort „notify“ angegebene Sensor (Schalter1) einen Funkbefehl übermittelt hat:

```
define Schalter1Notify notify Schalter1
{ if („%“ eq „off“) { fhem(„set wz_Media off“) } }
```

Der Wert, der von dem abgefragten Sensor gesendet wurde, wird hierbei automatisch in der Variablen „%“ gespeichert, die vor der Anweisung abgefragt wird. Wenn sie also den Wert „off“ besitzt, wird in FHEM der Befehl „set wz_Media off“ ausgeführt, der den angegebenen Aktor innerhalb des Netzes ansteuert und ausschalten soll. Um den „notify“-Befehl erstellen zu können, wird in der Kommandozeile zuerst ein „notify“ mit leerem Perlcodeblock erzeugt und dann in die Detailansicht gewechselt, um den Perlcode einzufügen und abzuspeichern.

2.1.2 Homeputer

Die Software Homeputer [3] ermöglicht die Erstellung von Regeln und Zeitabhängigkeiten über sogenannte Makros. Diese können sowohl manuell über eine spezielle Programmiersprache mit Befehlen in deutscher Sprache als auch menügeführt erstellt werden.

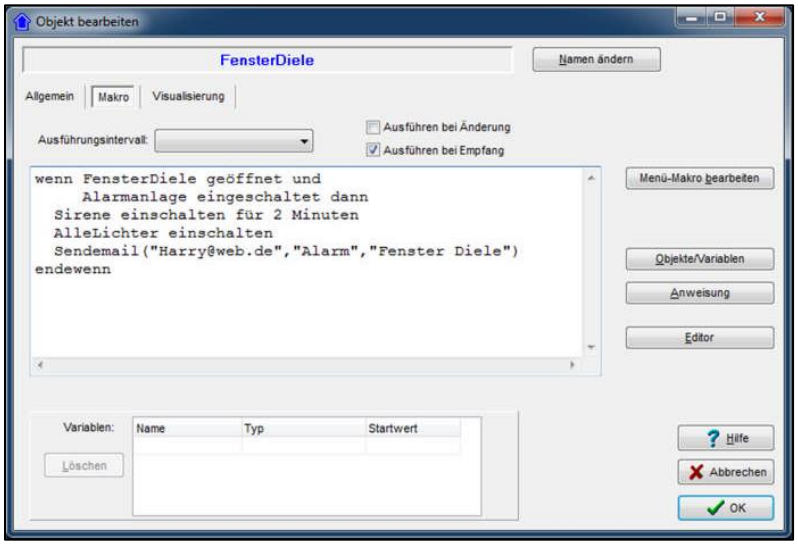

Abbildung 2.2: Erstellung von Makros bei Homeputer [4]

Das obige Beispiel beinhaltet unter anderem einfache „wenn …, dann …“-Anweisungen: Wenn bei eingeschalteter Alarmanlage das Fenster in der Diele geöffnet wird, soll für die Dauer von zwei Minuten die Sirene eingeschaltet werden; außerdem sollen sämtliche Lichter angeschaltet und der Hausherr über den vermeintlichen Einbruch per E-Mail benachrichtigt werden.

2.1.3 IP-Symcom

Bei IP-Symcom [5] handelt es sich um eine kommerzielle Software auf PHP-Basis. Es besteht die Möglichkeit zur menügeführten Skriptprogrammierung, d. h., der Nutzer ist in der Lage, verschiedene PHP-Befehle mit Hilfe eines Editors zu einem Skript zusammenzufügen, das sich manuell oder ereignisgesteuert ausführen läßt.

Die zur Definition eines bestimmten Ereignisses benötigten Variablen, Auslöser, Optionen usw. können über eine graphische Oberfläche gepflegt werden (siehe Abbildung 2.3), ohne das hierfür noch tiefere Programmierkenntnisse notwendig sind. Ausgehend von dieser Regelbasis werden dann die im Voraus erstellten Skripte entweder beim Eintreten der Bedingungen oder auch zyklisch zu bestimmten Zeitpunkten gestartet.

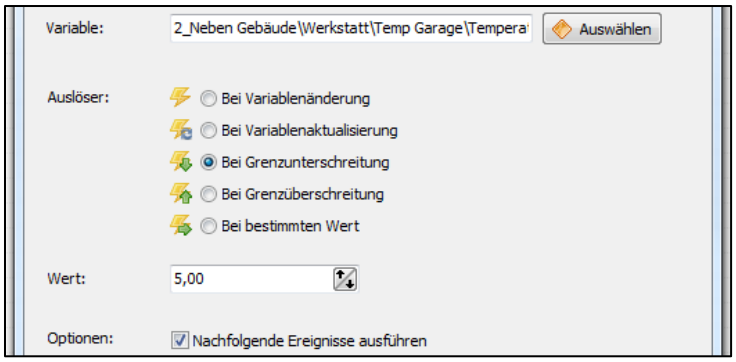

Abbildung 2.3: Erstellung eines Ereignisses bei iP-Symcon [6]

Auf Abbildung 2.3 wird beispielsweise ein Ereignis definiert, welches eine Heizung einschaltet, sobald die gemessene Temperatur unter einen bestimmten Schwellenwert gefallen ist. In der ersten Zeile wird als Variable der Name des Quellsensors aufgeführt, der in diesem Falle den jeweils aktuellen Temperaturwert bereitstellt. Als Auslöser wird anschließend der Punkt Grenzunterschreitung ausgewählt und der entsprechende Wert (in dieser Beispielanwendung 5 Grad) eingegeben. Über einen weiteren Menüpunkt kann noch festgelegt werden, ob nach der erstmaligen Erfüllung der Bedingung das Ereignis weiterhin ausgeführt werden soll, was in unserem Falle nicht sinnvoll wäre, da es ausreichend ist, die Heizung ein einziges Mal einzuschalten.

2.2 Fragestellungen

2.2.1 Format und Schnittstellen

Auf die Frage nach dem Speicherformat konnten in Gruppendiskussionen 2 favorisierte Möglichkeiten herausgearbeitet werden: SQL-Datenbank oder XML-Datei. Nach Abwägung der Vor- und Nachteile, fiel die Entscheidung zugunsten einer XML-Datei aus. Diese bietet bietet einerseits den Vorteil der einfachen Editierbarkeit und benötigt andererseits keine zusätzlichen Dienste oder Programme wie zum Beispiel einen SQL-Server. Darüber hinaus existiert für XML-Dateien eine Vielzahl von Schnittstellen um lesend oder schreibend auf den Inhalt zugreifen zu können.

Der Aufbau dieser XML-Datei ist in einer Document Type Definition (DTD) festgehalten:

```
<!ELEMENT rules (rule*)>
  <!ELEMENT rule (conditions, actions, alternatives?)>
    <!ELEMENT conditions (condition+)>
      <!ELEMENT condition (#PCDATA)>
        <!ATTLIST condition operator ( lt | eq | gt | le | ge | ne )
         #IMPLIED value CDATA #IMPLIED >
    <!ELEMENT actions (action+)>
      <!ELEMENT action (#PCDATA)>
        <!ATTLIST action value CDATA #IMPLIED >
    <!ELEMENT alternatives (action+)>
```

Mit diesem Aufbau können bereits fast alle Anforderungen erfüllt werden. Eine Sonderstellung nehmen hierbei noch zeitgesteuerte Aktionen ein. Diese konnten aufgrund von Zeitmangel nicht mehr weiter berücksichtigt werden. Möglich wäre hierfür ein spezieller Sensor, welcher einen Zeitstempel zurück gibt und somit wie ein einfacher Sensor behandelt werden könnte. Ein anderer Ansatz wäre die Erweiterung der Server-Funktionalität um solche zeitgesteuerten Ereignisse zu verarbeiten.

2.2.2 Verwaltung der Regeln durch Anwender

Bei dieser Lösung über eine XML-Datei können erfahrene Anwender mithilfe eines gewöhnlichen Text-Editors das Regelwerk einsehen und bearbeiten. Für unerfahrene Anwender kann eine einfache Benutzeroberfläche erstellt werden, über die sämtliche Regeln verwaltet werden können.

2.2.3 Verwaltung der Regeln durch den Server

Die erstellte XML-Datei muss nach einer Änderung neu in den Server eingelesen werden. Dies kann entweder zyklisch oder manuell geschehen. Beim Einlesen der XML-Datei wird diese durch einen Parser interpretiert und in eine programminterne Struktur übersetzt. Meldet nun ein Sensor einen veränderten Sensorwert (Schalter wurde von “aus” auf “an” geschalten, Temperatur ist von 21,0 °C auf 20,5 °C gefallen, etc.) so wird diese Veränderung durch den Server empfangen. Daraufhin werden vorhandene Regeln für diesen Sensor durchsucht und gegebenenfalls die entsprechenden Aktionen ausgeführt.

2.3 Implementierung

Die bereits erwähnte DTD ermöglicht beliebig viele Regeln in einer XML-Datei zu speichern. Jede Regel enthält mindestens eine Bedingung und eine Aktion. Eine Bedingung besteht aus einer Sensor-ID, einem Vergleichsoperator und einem Vergleichswert. Wird mehr als eine Bedingung angegeben, so werden diese logisch Und-verknüpft. Eine Oder-Verknüpfung ist bisher noch nicht vorgesehen, lässt sich aber einfach über mehrere Regeln realisieren. Als Aktion können ebenfalls mehrere Anweisungen angegeben werden. Eine Anweisung besteht aus einer Aktor-ID und einem Wert, auf welchen der Aktor gesetzt werden soll. So können beispielsweise durch Betätigung eines Schalters mehrere Lichtquellen oder andere Aktoren geschalten werden. Optional kann zu jeder Regel noch eine Alternative hinzugefügt werden, welche ebenfalls mehrere Anweisungen beinhalten kann.

Da der Heimautomatisierungs-Server in Python realisiert wurde, sollte auch das Modul zum Einlesen der XML-Datei in Python umgesetzt werden. Hierfür wurde ein Parser entwickelt, um die Regeln aus dem XML-Format in eine Programmstruktur zu übersetzen. Diese Struktur bleibt während der Laufzeit des Servers im Speicher erhalten und kann so, bei Eintreffen eines neuen Sensorwertes, schnell und komfortabel abgefragt werden. Trifft eine Regel zu, wird die angegebene Aktion bzw. Alternative ausgeführt. Hierfür wird ein CoAP-Paket erzeugt, welches den gewünschten Wert an den Ziel-Aktor übermittelt. Bei mehreren angegebenen Anweisungen wird für jede Anweisung ein CoAP-Paket erzeugt.

3 Resultate

3.1 Vergleich

Insbesondere gegenüber der komplizierten Regelerstellung von FHEM versucht die hier implementierte Lösung ein auf das Wesentliche reduziertes Konzept mit einfacher Bedienung zu verknüpfen, indem sie die menügeführte Regelerstellung bevorzugt, wie sie u. a. auch bei IP-Symcom verwendet wird. Auf eine umständliche Formulierung von Quelltext kann daher bewusst verzichtet werden.

Die leicht zu verwaltende Speicherung der Regeln an zentraler Stelle ist ein weiteres Merkmal dieses Projekts. Im Gegensatz dazu verfolgt das Hausautomatisierungssystem Hexabus [7] einen dezentralen Regelansatz, bei dem das Regelwerk über die einzelnen Knoten verteilt ist. Dies ermöglicht den Einsatz in autonomen Systemen und erhöht die Toleranz gegenüber Ausfällen von einzelnen Steuerungsknoten. Ein wesentlicher Nachteil ist jedoch der zusätzlich benötigte Kommunikationsaufwand für den Broadcast, während bei einer zentralisierten Verwaltung der Regeln weniger Kommunikation im Netz zu erwarten ist und die Unterstützung einfacher Clientanwendungen durchaus ausreicht.

Im Hinblick auf den Funktionsumfang ist das vorliegende Projekt den kommerziellen Anbietern noch unterlegen, da vor allem zeitabhängige Funktionen nicht zur Verfügung stehen. Im Gegensatz dazu ermöglicht es IP-Symcom, Ereignisse zu bestimmten Zeitpunkten sowie in bestimmten Abständen, z. B. alle 7 Tage, auszulösen.

3.2 Ausblick

Weitergehende Anforderungen an das bestehende Regelverarbeitungssystem können im Zuge einer Fortführung des Projekts berücksichtigt werden. Beispielsweise könnte das Regelwerk durch das Einführen von Zeitstempeln erkennen, ob ein Knoten über eine längere Zeit nicht gesendet hat und möglicherweise ausgefallen ist. Als formelles Modell bietet sich dabei ein Zustandsgraph an, wie er von dem bereits erwähnten System Hexabus verwendet wird.

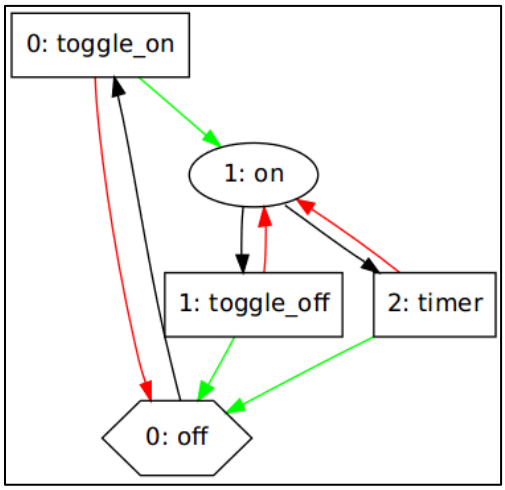

Abbildung 3.1: Zustandsgraph eines Schalters mit Zeitsteuerung [8]

Wie die Abbildung 3.1 verdeutlichen soll, benutzt es einen Automaten, der aus Zuständen und Übergängen (Transitionen) besteht, um den aktuellen Zustand eines Knotens zu modellieren. Jede Transition hat einen erforderlichen Ausgangszustand und verweist auf die Bedingungen, die erfüllt

sein müssen, um einen Zustandswechsel herbeizuführen. Wenn die Transition schließlich feuert, wird die gewünschte Aktion ausgeführt und der Automat geht je nach Erfolg oder Misserfolg in einen entsprechenden Zustand über, wodurch Fehlübertragungen oder dergleichen erkannt werden können. Die Zustandsautomaten werden dabei automatisch mit Hilfe eines Compilers erzeugt. Auf diese Weise ist es auch möglich Bedingungen zu implementieren, die eine Aktion auslösen, wenn der Automat zu lange in einem bestimmten Zustand geblieben ist. Dies kann die Verlässlichkeit des Systems erhöhen, indem zum Beispiel bei unerwarteten Ausfällen eines Sensors der Automat in den Ausgangszustand zurückversetzt wird.

4 Bedienkonzepte

Der Aufbau des gesamten Sensornetzes ist sehr modular. Die einzelnen Komponenten funktionieren für sich genommen in der Regel autark. Dadurch sind unterschiedlichste Arten des Zugriffs denkbar. Für den Betrieb des Netzes selbst ist weder ein Webserver noch eine Oberfläche zur Regelverwaltung notwendig. Auch die Regeln selbst sind für die reine Funktion des Netzes nicht notwendig. Wie eingangs erwähnt, würde ein solches Netz ohne jegliche Regeln jedoch nur wenig Sinn machen. Zur Steuerung und Visualisierung wurde im Rahmen dieses Projektes eine kleine Web-Oberfläche erstellt. Über diese ist es möglich, die angeschlossenen Sensoren zu überwachen, sowie Aktoren zu schalten. Die Verwaltung der Regeln kann ebenfalls in diese Oberfläche integriert werden. Abbildung xx zeigt, wie der Prototyp des Sensornetzes aus Sicht der Datenkommunikation derzeit aufgebaut ist.

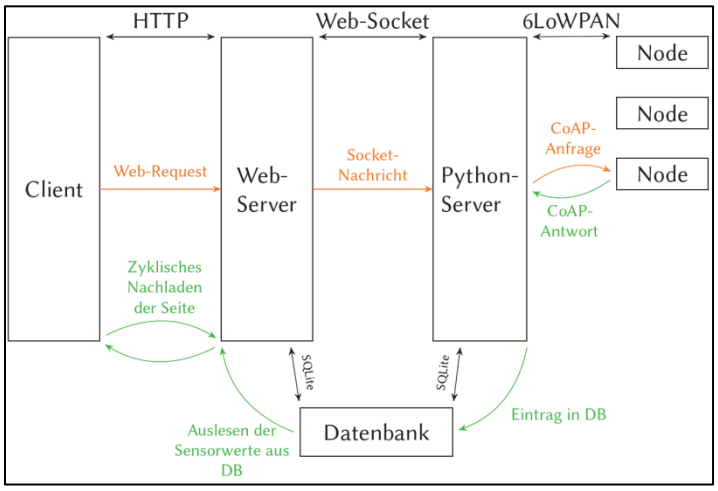

Abbildung 4.1: Datenkommunikation für Web-Oberfläche

Die einzelnen Nodes (Sensoren und Aktoren angeschlossen an einzelne Microcontroller) werden durch den Python-Server überwacht. Sämtliche Sensor-Werte werden in eine SQLite-Datenbank gespeichert. Durch dieses Vorgehen kann eine Web-Oberfläche auf Anfrage schnell aufgebaut werden, ohne dass zuvor sämtliche Sensorwerte abgefragt werden müssen. Zusätzlich lassen sich so Statistiken erstellen, indem alte Sensorwerte für eine gewisse Zeit aufbewahrt werden.

Durch die modulare Struktur ist dies aber nur eine Möglichkeit von vielen. Es wäre ebenso denkbar, statt einer Web-Oberfläche eine native Anwendung zur Steuerung bzw. Visualisierung einzusetzen. Voraussetzung ist lediglich die Möglichkeit auf die SQLite-Datenbank zugreifen zu können, sowie über einen Web-Socket mit dem Heimautomatisierungs-Server zu kommunizieren.

Zusammenfassend lässt sich somit sagen, dass der modulare Aufbau der wohl größte Vorteil des Systems ist. Durch die Schaffung von möglichst einfachen und abstrakten Schnittstellen (CoAP, XML, Web-Socket), kann das gesamte System individuell an verschiedenste Bedürfnisse angepasst werden.

Quellenangaben

[1] „FHEM - Description“ [Online]. Verfügbar über: http://fhem.de/fhem.html. [Zugriff am 01.03.2013].

[2] U. Maaß, „Heimautomatisierung mit FHEM“ 2012. [Online]. Verfügbar über: http://fhem.de/Heimautomatisierung-mit-fhem.pdf. [Zugriff am 01.03.2013].

[3] „homeputer CL Studio 4.0 für HomeMatic“ [Online]. Verfügbar über: http://www.contronics.de/shop/Zentralen-und-Software/Homeputer-CL-Studio-Software-fuer-HomeMatic. [Zugriff am 02.03.2013].

[4] contronics GmbH, „homeputer CL Software für die Hausautomation“ [Online]. Verfügbar über: http://www.contronics.de/download/homeputerCLBeschreibung.pdf. [Zugriff am 02.03.2013].

[5] iP-Symcon, [Online]. Verfügbar über: http://www.ip-symcon.de/. [Zugriff am 02.03.2013].

[6] iP-Symcon, „Dokumentation“ [Online]. Verfügbar über: http://www.ip-symcon.de/service/dokumentation/konzepte/ereignisse/. [Zugriff am 02.03.2013].

[7] „github - HexaBus“ [Online]. Verfügbar über: https://github.com/mysmartgrid/hexabus/wiki. [Zugriff am 27.02.2013].

[8] HexaBus, „Wiki“ [Online]. Verfügbar über: https://github.com/mysmartgrid/hexabus/wiki/Hexabus-Assembler. [Zugriff am 02.03.2013].