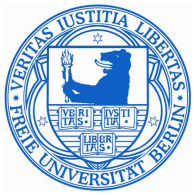

Projekt Telematik
Fachbereicht Technische Informatik, FU Berlin

Implementation of a wifi based home monitoring system

Anne Haase, Dominik Weidemann

Projekt-Dokumentation

Zusammenfassung

Hier ein kleine Zusammenfassung

Inhaltsverzeichnis

1 Einleitung

Sensoren in dem Haushalt sind für den Nutzer sehr praktisch und erhalten eine immer größer werdende Bedeutung. So hat fast jeder Haushalt mindestens einen Sensor für die Innen- oder Außentemperatur.
Es gibt eine große Auswahl von Sensoren, die im Haushalt behilflich sein können. So könnte ein mobiler Temperatursensor an mehreren Stellen eingesetzt werden, um zum Beispiel rechtzeitig den Temperaturanstieg im Kühlschrank zu erkennen oder den zu starken Temperaturabfall in der Sauna. Mit einem Vorwarnsystem kann der Nutzer nun alarmiert werden und rechtzeitig reagieren.
Weiterhin wäre es aber auch interessant, wenn der Nutzer selbst prüfen könnte, ob die gemessenen Daten sich noch so verhalten, wie sie sollten. Ob zum Beispiel die Temperatur im Kühlschrank immer noch konstant 7 Grad beträgt, oder abnimmt bzw. zunimmt.
Im Rahmen dieses Projektes wurde ein System entwickelt, dass gemessene Daten von Sensoren sammelt, welche im Haus verteilt sein können, und diese in Form von Liniendiagrammen auf einer Webseite anzeigt. Des weiteren wird der Nutzer auf dieser Webseite gewarnt, wenn definierte Bedingungen eintreffen (z.B. die Temperatur sinkt unter 40°C in der Sauna). Dieses Dokument stellt die Idee bis zur Implementierung vor. Dabei wird wie folgt vorgegangen: [TODO]

2 Verwendetet Technologien

2.1 Linux Voyage [2]

Als Betriebssystem für unseren Access Point haben wir uns für Voyage entschieden. Linux Voyage ist eine abgespeckte Version von Linux Debian, die für Access Points, Asterisk/VoIP Gateway und sonstigen Geräten geschrieben wurde. Da wir auf dem Access Point eine Flashkarte von 8 GB zur Verfügung hatten, war sich dieses Betriebssystem besonders gut geeignet, da es sehr klein ist (128MB). Des weiteren sind alle wichtigen Treiber schon enthalten, welche wir für die WLAN Karte lediglich aktivieren mussten.
Ein weiterer Vorteil ist, dass weitere Software wie gewohnt über die Konsole installiert werden kann. So konnten wir recht einfach MySQL und Tomcat installieren.

2.2 GWT [1]

Google Web Toolkit ist eine Entwicklungsumgebung zur Entwicklung von Webseiten. Der Client-Code wird in Java geschrieben und das GWT-Framework generiert daraus dann web-freundliches Javascript. So kann man möglichst einfach komplexe Webseiten entwickeln und hat eine einfach Anbindung an das Backend gegeben.
Die Vorteile bei GWT sind:

- Browser-Kompatibilität
- Besseres debuggen, da sich Javacode besser debuggen lässt
- einfacheres entwickeln, wenn man nicht Javascript kann
- viele Zusatz-Frameworks (z.B. Smart GWT)

Der Nachteil an GWT ist, dass es relativ komplex ist und es muss sich erst eingearbeitet werden um zu verstehen, an welchen Stellen im Code Änderungen vorgenommen werden müssen. Dafür gibt es allerdings eine Menge Tutorien, die weiter helfen.
GWT war für unser Projekt sehr hilfreich, da die Webseite abhängig von dem Backend aufgebaut wurde und durch GWT-RPC eine einfache Anbindung an das Backend (unsere Datenbank) gegeben war.

Im Rahmen dieses Projektes wurde GWT als Plugin für Eclipse verwendet. Dies lässt sich sehr leicht installieren... [TODO]

2.3 Smart GWT

Smart GWT ist ein Framework, welches, basierend auf GWT, eine Vielzahl von Erweiterungen der Oberfläche anbietet. Da es viele Widgets anbietet, die GWT nicht besitzt, wurde es für dieses Projekt verwendet.
Dabei bietet Smart GWT über die Showcases (hier ein Link: [3]) viele Codeschnipsel, die die Entwicklung einfach gestalten. Des weiteren lässt sich Smart GWT sehr leicht in den Code einbinden. Der Nutzer lädt lediglich Jars herunter, die eingebunden werden müssen.
Der Nachteil von Smart GWT ist allerdings, dass man die Komponenten nicht mit Komponenten von GWT kombinieren sollte (also zum Beispiel sollte man keine Smart GWT Buttons in Layouts von GWT involvieren), da dies zu unerwarteten Effekten führen kann. [TODO: installation]

2.4 Charts von Google

Google Visualization bietet viele Möglichkeiten, Charts und Tabellen in GWT zu erstellen. Dabei werden die Charts in Javascript geschrieben. Allerdings bietet Google auch ein Jar für GWT-Dateien an, so das an dieser Stelle der Programmierer Java-Code schreiben kann.
Um ein Chart zu erstellen, muss der Nutzer lediglich eine Datentabelle (DataTable) von Google erstellen und dabei die Formate der Eingabedaten festlegen. Diese Daten werden dem gewünschten Chart mitgegeben und dieser wird auf der Oberfläche dargestellt. Alles weitere wird mitgeliefert wie zum Beispiel die „Clickevents“, oder die dynamische Größenanpassung der Charts an die Menge der Daten.
Aber auch hier ist der Nachteil, dass man Charts von Google nicht einfach mit Smart GWT Elementen kombinieren kann, dafür aber mit GWT Elementen.

3 Architektur

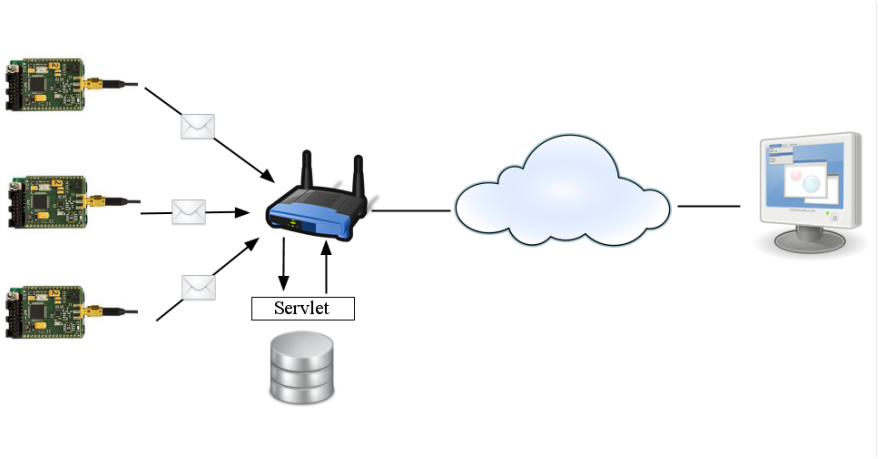

Abbildung 1: Architektur

4 Sensor-Knoten

Der verwendete Sensor-Knoten basiert auf dem MSB430H-Board. Es enthält mehrere Sensoren. Diese sind der SHT11-Sensor , welcher die Temperatur und Luftfeuchtigkeit misst und der MMA 7260Q G-Sensor, mit dem die relative Lage des Sensor-Knotens im Raum festgestellt werden kann. Der Sensor-Knoten wurde mit dem

WLAN-Modul RN-134 der Firma Roving Networks über die UART-Schnittstelle fest verlötet. Die Kommunikation zwischen dem programmierbaren Board und dem WLAN-Modul erfolgt also über die serielle Schnittstelle. An dieser Schnittstelle wurde ein Kipp-Schalter und ein serielles Kabel angebracht, welche es ermöglichen, den Daten-Austausch zwischen Board und Modul zu verfolgen. Damit fiel das Debuggen der Kommunikation leichter.

4.1 HTTP-Kommunikation

Bei der Erprobung des WLAN-Moduls über eine serielle Schnittstelle haben wir verschiedene Übertragungsverfahren getestet:

- Das Modul bietet die Möglichkeit, alle x Sekunden einen UDP-Broadcast zu senden, welcher die auf dem Modul gesetzte Zeit beinhaltet und alle gemessenen Sensor-Werte. Dies ist für das Projekt nicht praktikabel, da die Sensoren nicht direkt an das WLAN-Modul angeschlossen sind, sondern an dem Sensor-Knoten. Deswegen ist auch der UDP-Unicast unbrauchbar.
- Das Modul bietet die Möglichkeit, eine TCP Auto-Connection zu nutzen, also automatisch alle x Sekunden aufzuwachen und eine TCP-Verbindung zu einem vordefinierten Host aufzurufen. Da das WLAN-Modul aber keine Werte vom Sensor-Knoten pollen kann, ist diese Methode ebenfalls unbrauchbar.
- Eine weitere Methode, die aber ebenfalls wegen des oben beschriebenen Grundes fehlschlägt ist das automatische Senden von Sensor-Daten an einen HTTP-Server.

Nach ausgiebigen Tests haben wir uns entschieden, die Daten über eine TCP-Verbindung zu verschicken. Diese wird von dem Sensor-Knoten geöffnet. Danach wird ein GET-Befehl des HTTP-Protokolls abgesetzt. Dabei wird angegeben, dass die Verbindung nicht geschlossen werden soll. Dies ermöglicht es (HTTP 1.1), mehrere Datensätze hintereinander mit Hilfe des GET-Befehls versendet, ohne die TCP-Verbindung jedes Mal neu initiieren zu müssen. Da die Verbindung nach ca. 5 Minuten vom Server geschlossen würde, muss sie in regelmäßigen Abständen vom Client geschlossen und direkt danach neu aufgebaut werden.
Das WLAN-Modul verschickt einen festgelegten String (Default: „*HELLO* “), welcher nicht leer sein darf, beim erfolgreichen Aufbauen der TCP-Verbindung. Das hat uns zuerst Probleme bereitet hat, da der Tomcat-Server unbekannte Befehls-Präfices nicht ignoriert. Die Lösung ist das Benutzen von „GET“ als Communication-String. Da dieser aber nur beim Starten der Verbindung gesendet wird, muss jeder weitere

GET-Befehl vollständig übertragen werden. Dies ist eine unschöne Lösung, allerdings lies sich das Problem nicht anders beheben, da das WLAN-Modul selbst nicht programmiert werden kann.

4.2 Konfiguration des WLAN-Moduls

Bei jedem Start des Sensor-Knotens muss das WLAN-Modul in einen wohldefinierten Zustand gebracht werden. Da beim Booten des Knotens nur schwer verifiziert werden kann, ob alle Einstellungen korrekt vorgenommen sind, haben wir uns dafür entschieden, dass das WLAN-Modul bei jedem Start auf die Fabrikeinstellungen zurück gesetzt und dann neu konfiguriert wird. Dies ist nötig, da beim Programmieren der Sensor-Knoten zwar neu gestartet wurde beim Debuggen, die Stromversorgung des WLAN-Moduls aber nicht unterbrochen wurde und es so in einem beliebigen Zustand sein konnte. Das Zurücksetzen hat ebenfalls den Vorteil, dass das Modul nicht vor dem ersten Betrieb explizit initiiert werden muss, allerdings dauert jeder Boot-Vorgang etwas länger. Insgesamt jedoch werden bereits ca. 10 Sekunden nach dem Einschalten die ersten Sensor-Werte gesendet.

4.3 Programmierung des Sensor-Knotens

Für die Programmierung des Sensor-Knotens wurde der Code Composer [TODO: INSERT REAL NAME HERE] verwendet. Die Programmiersprache ist ein C-Dialekt, welcher fast alle Funktionen des Standards bereit stellt.
Das Programm des Sensor-Knotens besteht aus zwei Teilen: Zuerst wird eine Initalisierungs-Phase durchlaufen in der Interrupts angeschaltet werden, nötiger Speicher allociiert und das WLAN-Modul konfiguriert wird. Danach beginnt eine Endlos-Schleife, die das „Betriebssystem“ darstellt. In dieser Schleife werden die Sensor-Werte ermittelt und dann per WLAN verschickt.

4.3.1 Initialisierung

Bei der Initialisierung werden folgende Schritte chronologisch ausgeführt:

- Initialisierung der Port Register
- Aktivierung der quarzstabilen XT2 Taktquelle (7.3728MHz)
- Initialisierung der UART-RS232 Schnittstelle mit 9.6kBit/s
- Einschalten der Interrupts für die beiden Taster, Freigeben der Interrupts und Aktivierung der Interrupt-Routinen

- Zurücksetzen des WLAN-Moduls in den Auslieferungszustand
- Setzen von WLAN-SSID, Passwort und anderen Parametern
- Verbindungsaufbau mit dem Access-Point und einrichten der TCP-Verbindung zum Tomcat-Server

Dann tritt das System in die zweite Phase ein.

4.3.2 Betriebssystem

Das „Betriebsystem“ besteht aus einer Endlos-Schleife, in der folgende Aktionen ausgeführt werden:

- Überprüfung, ob bereits 20 Abfragen gesendet wurden; ggf. Trennung und erneuter Aufbau der TCP-Verbindung (der Server würde die Verbindung nach ca. 5 Minuten trennen)
- Auslesen der Sensoren, Aufbereitung der gelesenen Werte und Speicherung des Requests in einer Warteschlange, welche alle zu sendenden GET-Anfragen enthält
- Falls die Verbindung nicht gerade neu geöffnet wurde, wird „GET “ dem Request voran gestellt (Grund siehe Abschnitt 4.1)
- Solange noch weitere Requests in der Warteschlange sind, werden diese ebenfalls entfernt und dann versendet

4.3.3 Tastendruck

Die Erkennung eines Tastendrucks geschieht in dem in der Initialisierung definierten Interrupt. Dort wird direkt beim Interrupt und 0.5 Sekunden nach dem Auslösen gemessen, welcher Taster gedrückt wurde und dann beide Messungen logisch verodert. Damit können wir mit einem Interrupt ebenfalls feststellen, falls beide Taster gleichzeitig gedrückt wurden. Weiterhin kann damit ausgeschlossen werden, dass die analog gemessene Größe am Anfang noch etwas flackert und eventuell beim ersten Auslesen den falschen Wert liefert. Danach wird ein GET-Request für das Taster-Event erstellt und in die Warteschlange eingereiht. Beim nächsten Durchlauf der Betriebssystem-Schleife wird dieser Request dann ebenfalls mit verschickt. Die Auflösung der Tasten-Events ist damit gleich einem Schleifendurchlauf, der bei durchschnittlich 3 Sekunden, maximal bei 7 Sekunden (Trennen und neuer Aufbau der Verbindung) liegt.

5 Backend

5.1 Daten-Verarbeitung

5.2 Daten-Bereitstellung

6 Frontend

Die Sensoren, die im Haushalt verteilt sind, schicken die gemessenen Daten an den Access Point. Nun müssen diese Daten in geeigneter Weise aufbereitet werden und den Nutzer zur Verfügung gestellt werden. Hierfür wurde eine Webseite gebaut, die es den Nutzer ermöglicht, die gemessenen Daten in Diagrammen darstellen zu lassen. In diesem Kapitel wird dieses Frontend vorgestellt und beschrieben.

6.1 Anforderungen

An die Webseite, die es den Nutzer ermöglicht, die Daten der Sensoren zu überwachen, gibt es eine Reihe von Anforderungen.
Eine wichtige Anforderung ist die benutzerfreundliche Oberfläche, die in sich einfach gehalten ist und es den Anwender ermöglicht, sich schnell einzuarbeiten. Aus diesen Grund wurde die Oberfläche sehr schlicht gehalten und auf das Wesentliche reduziert.
Über eine Drop-down-Box kann der Nutzer zwischen den verschiedenen Sensoren auswählen und sich Daten anzeigen lassen. Dabei werden unterschiedliche Diagramme für die unterschiedlichen gemessenen Daten angezeigt. In unseren Fall hatten wir ein Diagramm für die Temperatur, Feuchtigkeit und 2 Diagramme für die Neigung des Sensors. Da es aber möglich sein soll, andere Sensoren an das System anzuschließen, welche andere Daten (Propertries) messen, passt sich die Oberfläche mit den Diagrammen an die jeweiligen Daten an. Es werden in die Diagramme angezeigt und mit Daten gefüllt, welche vom Sensor gemessen werden.
Nun kann der Nutzer die gemessene Werte überwachen. Dabei hat dieser aber nur ein Diagramm gleichzeitig im Blick (da pro Diagramm ein Tab in einem Tab-Menü zur Verfügung steht). Es kann passieren, dass der Anwender so zum Beispiel nicht beobachten kann, dass die Temperatur im Kühlschrank stetig zunimmt, weil dieser das Diagramm mit der Neigung des Sensors geöffnet hat. Damit der Nutzer gewarnt wird, wurden Trigger in unser System integriert. Dabei wird in serverseitiger und clientseitiger Trigger unterschieden. Wird eine Aktion auf der Clientseite ausgelöst, so wird eine Dialogbox auf der Webseite angezeigt. Wird eine serverseitige Aktion ausgelöst, so wird eine System-Ausgabe auf der Konsole ausgegeben.

Eine weitere Anforderung an das System war es, dass der Standort eines Sensors bearbeitet werden kann und dieser dann in die Datenbank abgespeichert wird. Es kann passieren, dass der mobile Sensorknoten an einen anderen Ort platziert wird. Bei der Anzeige der Sensordaten kann der Benutzer auf „Edit“ klicken und den Ort des Sensors bearbeiten. Die ID und die IP sollen nicht geändert werden.
Alle diese Anforderungen sind für die leichte Bedienbarkeit der Oberfläche und wurde in diesem System umgesetzt.

7 Projekt zum Laufen bringen

8 Ausblick

9 Appendix

Literatur

[1] Google web toolkit. https://developers.google.com/web-toolkit/.

[2] Linux voyage. http://linux.voyage.hk/.

[3] Smart GWT. http://www.smartclient.com/smartgwt/showcase/.