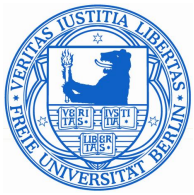


Projekt Telematik
Fachbereicht Technische Informatik, FU Berlin

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°C 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:
Im nächsten Kapitel wird beschrieben, was das Ziel des Projekts war und wie vorgegangen wurde. In Kapitel 3 wird die Architektur des Projekts erklärt und im besonderen auf die Modellierung eingegangen. In den nächsten 3 Kapiteln wird die Implementierung erklärt, wobei diese in Sensor-Knoten, Backend und Frontend eingeteilt wurden. Am Ende der Dokumentation wird darauf eingegangen, wie man das Projekt erweitern kann und was für Erfahrungen damit gemacht wurden.

2 Projektbeschreibung

2.1 Beschreibung

Im Rahmen dieses Projekts sollte ein System einwickelt werden, welches die Überwachung von Sensordaten im Haus ermöglicht. Die Sensoren werden im Haushalt verteilt, welche in periodischen Abständen die gemessenen Daten an den Access Point schicken. Um welche Art von Sensoren es sich handelt ist nicht relevant. Es können unterschiedliche Sensoren, welche unterschiedliche Daten messen, an das System angebunden werden.
Die gemessenen Daten werden beim Access Point in geeigneter Weise verarbeitet und werden über das Web zur Verfügung gestellt. Dabei werden die Daten in eine Datenbank abgespeichert und in Form von Liniendiagrammen und Tabellen für den Nutzer in einem Browser zur Verfügung gestellt. Der Benutzer hat so die Möglichkeit die Sensordaten und somit den Haushalt zu überwachen. Des weiteren sollte ein Warnsystem eingebaut werden, indem der Anwender gewarnt wird, wenn ein

Sensorwert einen vordefinierten Wert über- oder unterschreitet.

2.2 Szenarien

Für dieses System kann man sich eine Reihe von Szenarien überlegen, in der dieses System relevant für den Alltag ist. Diese sollen hier vorgestellt werden.

2.2.1 Szenario 1(Kühlschranküberwachung)

Es wird ein Sensorboard mit einem Temperatursensor und einen Taster an einer Kühlschranktür befestigt. Durch den Temperatursensor wird die Temperatur im Kühlschrank periodisch gemessen und an den Access Point über das WLAN-Modul geschickt. Durch den Taster am Sensorboard wird registriert, ob die Kühlschranktür geöffnet oder geschlossen ist. Auch diese Daten werden an den Access Point geschickt.
Über den Webbrowser kann der Nutzer sich die gespeicherten Daten der Sensoren ansehen und überwachen. Dabei werden die gemessenen Temperaturen in einem Liniendiagramm und die Werte des Tasters in einer Tabelle angezeigt. Da der Anwender im System eingestellt hat, dass er gewarnt werden möchte, wenn die Temperatur im Kühlschrank über 9°C beträgt, zeigt das System ein Dialogfenster an, sobald der Sensor einen Wert über 9°C misst und an das System schickt. Nun sieht der Benutzer in der Tabelle mit den Einträgen für den Taster (Event-Tabelle), dass die Kühlschranktür geöffnet ist und kann rechtzeitig reagieren.

2.2.2 Szenario 2(Zimmertemperaturüberwachung)

Es wird ein Sensorboard mit einem Temperatursensor und einem G-Sensor an das Schlafzimmer- oder Wohnzimmerfenster angebracht. Durch den Temperatursensor wird die aktuelle Temperatur und durch den G-Sensor die Neigung des Fensters gemessen und über das WLAN-Modul an den Access Point geschickt.
In dem Diagramm für die Temperatur kann der Anwender nun den Temperaturabfall beobachten, aber auch im Diagramm für den Neigungswert verfolgen, ob das Fenster angekippt ist oder nicht. Wird ein bestimmter Wert der Temperatur unterschritten, so wird dem Benutzer über eine Dialogbox Bescheid gegeben. Des weiteren kann der Benutzer auch definieren, dass eine Dialogbox angezeigt wird, wenn sich der Neigungswert grundlegend ändert (das Fenster also angekippt wird).
Wird das System weiter entwickelt, so kann man zum Beispiel über Fernzugriff die Heizung abstellen oder das Fenster schließen lassen, damit nicht unnötig Energie verbraucht wird.

2.2.3 Szenario 3(Waschmaschinenüberwachung)

An der Waschmaschine wird ein Sensorboard mit einem Feuchtigkeitssensor angebracht. Über ein Diagramm kann sich der Nutzer die Werte des Feuchtigkeitssensors

in einem Browser anzeigen lassen. Läuft die Waschmaschine über, so erhöht sich der Feuchtigkeitswert und es wird ab einen bestimmten Wert eine Dialogbox im Browser angezeigt, indem der Nutzer bei überhöhter Feuchtigkeit gewarnt wird.
Wird das System weiterentwickelt, so kann man sich vorstellen, dass der Nutzer per Fernzugriff die Waschmaschine ausstellen kann und sogar ein Nummer für den Klempner in der Nähe erhält.

2.3 Projektvorgehen

Für dieses Projekt wurde zum einen ein Sensorboard mit einem Temperatursensor, Feuchtigkeitssensor und G-Sensor zur Verfügung gestellt, welcher im Rahmen dieses Projekts programmiert werden muss. An dem Sensorknoten wurde ein WLAN-Modul abgebracht, welches konfiguriert werden musste.
Die Daten der Sensoren werden über das WLAN-Modul an den Access Point geschickt. Dieser Access Point wurde uns zur Verfügung gestellt. Dieser musste konfiguriert und programmiert werden, da auf diesen der Server läuft, der die Daten speichert. Diese Daten werden für das Frontend zur Verfügung gestellt, welches mit dem Google Webtoolkit programmiert wurde.

3 Architektur

Das Software-System besteht aus drei Teilen:

- Die Sensor-Knoten, welche Events detektieren und verschiedene Größen messen,
- das Backend mit Datenbank und Servlets zur Vorverarbeitung und Speicherung der Daten
- und dem Frontend, welches die Daten für den Benutzer aufbereitet und präsentiert.

Figur 1 verdeutlicht das Zusammenspiel dieser Komponenten. Die Sensoren versenden Nachrichten per HTTP-Request an ein Servlet, welches auf dem Access Point läuft. Dort werden die Daten vorverarbeitet und in der MYSQL-Datenbank gespeichert. Das Frontend, welches durch Aufruf der IP-Adresse des Access Point im Browser erreicht werden kann, greift ebenfalls auf die Daten zu und präsentiert diese mit Grafiken und Event-Listen.

Modellierung

Für die Modellierung der Daten wurden folgende Typen identifiziert:

- **Property**: Regelmäßig gemessener Wert, wie z.B. Temperatur oder Luftfeuchtigkeit

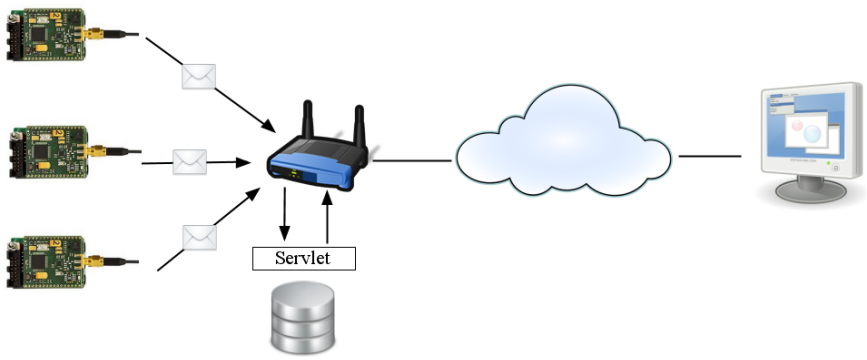


Abbildung 1: Architektur

- **Trigger**: Vom Benutzer festgelegtes Verhalten beim Erfüllen einer Bedingung
- **Event**: Spontanes Ereignis eines Sensors oder eines ausgelöster Trigger
- **Action**: Aktion die bei einem Event ausgeführt wird

Eine Property beschreibt einen regelmäßig von einem Sensor gemessenen Wert. Der bei diesem Projekt eingesetzte Sensor-Knoten 5 misst Temperatur, Luftfeuchtigkeit und Lage im Raum. Das System kann aber leicht auf weitere Properties, wie z.B. Rotation mit einem Gyroskop, oder Koordinaten eines GPS-Senders erweitert werden.

Ein Trigger definiert eine Regel, die auf Properties angewandt wird. Z.B. Überschreitung oder Unterschreitung eines Messwertes. Diese Trigger wurden in dem Software-System implementiert. Weitere Trigger könnten Vergleiche auf bestimmte Referenzwerte sein, Zeitabstand der Messwerte, Bereichs-Überprüfung, Auswertung mit komplizierteren Formeln, Anwendung von (Kalman-)Filtern und vieles mehr.

Ein Event beschreibt einen Zeitpunkt zu dem etwas Bestimmtes passiert ist. Zum Beispiel ein Knopfdruck am Sensor oder aber das Auslösen eines Triggers. Dabei kann ein Trigger auch viele verschiedene Ereignisse auslösen. Weitere Events, die in diesem Projekt aber nicht implementiert wurden, sind z.B. Zeit-Ticks, Events von externen Generatoren (SMS, Mail, ...) oder manuelles Erstellen eines Events durch einen Benutzer.

Eine Action besteht aus Programm-Code, welcher ausgeführt wird, sobald ein Event eintritt. Diese Actions können weitere Events generieren, den Benutzer benachrichtigen oder beliebige Aktionen durchführen. Dabei kann unterschieden werden, zwischen Code, welcher auf dem Server ausgeführt wird oder aber beim Client

im Browser. In diesem Projekt wurden zu Demonstrationszwecken zwei Aktionen implementiert: Zum einen eine serverseitige sysout-Benachrichtigung, zum anderen ein Dialogfenster, welches beim Client im Browser angezeigt wird.

4 Verwendetet Technologien

4.1 Linux Voyage [4]

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 ohne weitere Umstände MySQL und Tomcat installieren.

4.2 GWT [2]

Google Web Toolkit ist ein Toolkit zur Entwicklung und Optimierung von Browserbasierten Anwendungen [2]. Der Code wird dabei vom Entwickler in Java geschrieben und anschließend vom GWT-Compiler in Javascript-Code umgewandelt.
Die Vorteile bei GWT sind:

- Browser-Kompatibilität
- Testen in Java (Unit-Tests sind auch möglich)
- Entwickler benötigt keine Javascript Kenntnisse, da der Code in Java geschrieben wird
- viele Zusatz-Frameworks (z.B. Smart GWT)
- umfangreiche Dokumentation und Beispielanwendungen

Man muss allerdings erst den Aufbau von GWT verstehen, was am Anfang sehr aufwendig sein kann. So ist dieses Framework nicht unbedingt bei kleinen Anwendung zu empfehlen.
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 gegeben war.

Im Rahmen dieses Projektes wurde GWT als Plugin für Eclipse verwendet. Der Link für das Plugin ist der folgende: [3].

4.3 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: [5]) 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.

Möchte man Smart GWT nutzen, so muss die Jar-Datei (unter: [6]) heruntergeladen werden und in den „Build Path“ eingebunden werden.
In der Datei „*.gwt.xml“ [1] muss nun noch die folgende Zeile eingefügt werden:

```
<inherits name="com.smartgwt.SmartGwt"/>
```

4.4 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.

Um Google Charts nutzen zu können, muss man die Jar-Datei (unter: [1]) herunterladen und in das Projekt im „Build Path“ einbinden. In der Datei „*.gwt.xml“ muss nun die folgende Zeile eingefügt werden.

```
<inherits name='com.google.gwt.visualization.Visualization'/>
```

[1] steht für den Namen des Web Application Projekts

5 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.

5.1 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.

5.2 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.

5.3 Programmierung des Sensor-Knotens

Für die Programmierung des Sensor-Knotens wurde der Code Composer Essentials Professional (Version: CCE v3.1, Build: 3.2.4.3.8) 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 alloziert 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.

5.3.1 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.

5.3.2 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 5.1)
- Solange noch weitere Requests in der Warteschlange sind, werden diese ebenfalls entfernt und dann versendet

5.3.3 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.

6 Backend

Das Backend des Projekts nimmt die von Sensoren gemessenen Größen entgegen, verarbeitet diese Daten vor, speichert sie und gibt bei Bedarf Zugriff darauf. Es ist in zwei Teile aufgeteilt. Zuerst wird dabei auf die Daten-Verarbeitung eingegangen, welche in einem Java-Servlet auf dem Tomcat-Server läuft, und danach auf die Bereitstellung der Daten für das GWT-Frontend 7 mittels GWT-RPC.

6.1 Datenbank-Zugriff

Um bequemen Zugriff auf die Datenbank zu erhalten, benutzen wir spezielle Proxy-Klassen, welche die MYSQL-Datenbank in Klassen abstrahiert. Damit wir diese Klassen nicht selbst schreiben mussten (das dazugehörige Entwurfsmuster heißt DAO - DataAccessObjects in Kombination mit DTO - DataTransferObjects), haben wir einen DAO-Generator benutzt, welcher ein gegebenes SQL-Schema interpretiert und dann daraus Java-Klassen generiert. Der Generator, den wir benutzt haben, hieß FireStormDAO 4.0.1.

Wollen wir Code-Beispiele?

6.2 Daten-Verarbeitung

Die Daten-Verarbeitung besteht aus drei Dingen :

Bei der Auswertung der HTTP-GET-Requests wird zuerst der Typ der gesendeten Daten überprüft. Dieser Typ ist in dem Parameter *action* gespeichert. Anhand dieses Parameters wird dann entschieden, welche Methode zur Verarbeitung aufgerufen wird. Im Moment gibt es zwei zuständige Methoden: doKeyEvent und doProperty (Die Namensgebung orientiert sich an dem Servlet-Interface).
Die doKeyEvent-Methode speichert den Zeitpunkt des Events zusammen mit dem gedrückten Knopf (1: schwarz, 2: gelb, 3: beide) in der Datenbank ab. Danach wird die handleEvent-Methode zur Abarbeitung eventueller Aktionen bei einem Key-Event aufgerufen. (s.u.)
Die doProperty-Methode extrahiert die gemessenen Werte aus dem GET-Request. Im Moment sind diese Werte Temperatur, Feuchtigkeit und Lage im Raum (2 Werte). Zuerst müssen die Werte erneut gerundet werden, da die Floating-Point-Unit des Sensor-Knotens nicht besonders genau ist. Dann wird jeder einzelne Wert mit Zeitstempel und einer Referenz auf den sendenden Knoten in der Datenbank abgelegt.

Die Event-Erkennung wird direkt im Anschluss gestartet. Hierbei werden die Trigger für den entsprechenden Sensor-Knoten und die dazugehörige Property geladen und dann verglichen, ob die Property den definierten Schwellwert s des Triggers überschreitet. Abbildung 2 verdeutlicht unseren Mechanismus: Der im Trigger definierte Wert von 30°C (schwarze Linie) wurde überschritten. Bei der Ankunft des ersten gemessenen Wertes, der s übertrifft, wird vom Backend ein Start-Event generiert. Dieses bedeutet, dass der Schwellwert überschritten wurde. Damit ein Zittern um den Schwellwert herum ausgeschaltet werden kann, haben wir einen zweiten Schwellwert eingestellt (rote Linie). Dieser liegt um Δ verschoben in Richtung des normalen Wertebereiches, also bei einem Trigger für Wert-Überschreitung bei $s - \Delta$. Wird dieser zweite Schwellwert wieder unterschritten, so wird ein Stop-Event generiert.

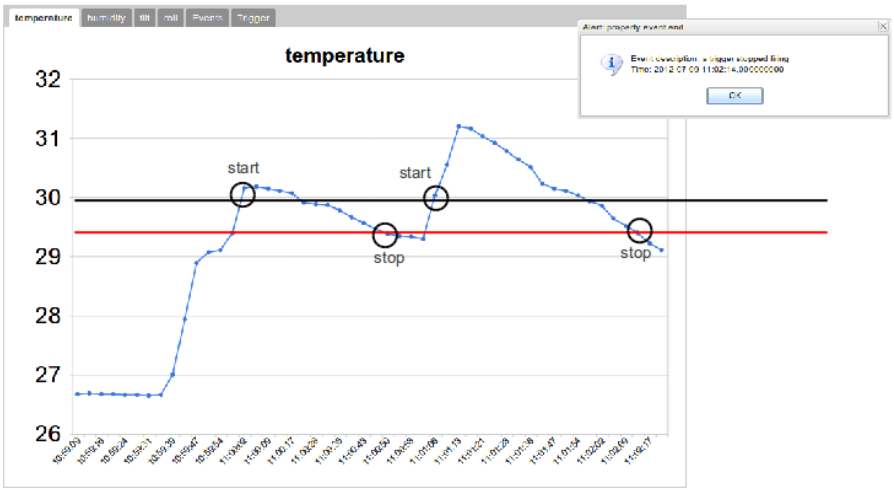
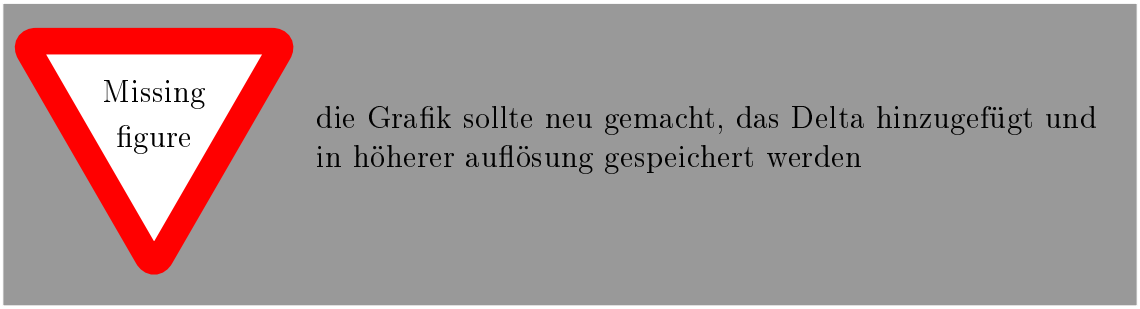


Abbildung 2: Event-Erkennung mit Thresholds



die Grafik sollte neu gemacht, das Delta hinzugefügt und in höherer auflösung gespeichert werden

Das Event-Handling wird nach jeder Event-Generation gestartet. Dabei wird überprüft, welche Actions für diesen Event-Typ registriert sind und führt diese aus. Es wird zwischen serverseitigen Aktionen und clientseitigen Aktionen unterschieden. Damit könnte das System mit anderen Komponenten gekoppelt werden, z.B. eine Benachrichtigung per SMS, E-Mail oder eben ein clientseitiger Alarm/Notifikation.

6.3 Daten-Bereitstellung

Die Daten-Bereitstellung für das Frontend geschieht mit GWT-RPC. Eine genauere Erklärung findet sich in Abschnitt 7.2. Die bereitgestellten Methoden dienen der Abfrage der Datenbank und lassen außerdem Manipulationen zu. Hier wird starker Gebrauch von den DataAccessObjects gemacht.

Beispiel-Code?

7 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 dem Nutzer zur Verfügung gestellt werden. Hierfür wurde eine Webseite gebaut, welche die gemessenen Daten in Diagrammen darstellen.
In diesem Kapitel wird dieses Frontend vorgestellt und beschrieben.

7.1 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.
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.

7.2 Projektaufbau

Da für dieses Projekt GWT genutzt wurde, war der Aufbau des Frontends zum Teil durch die Struktur von GWT vorgegeben.

Unter dem Ordner „src/de/berlin/fu/client“ befindet sich die Datei „TelproGWT.java“ in der man Hauptteil des Frontends findet. In der Datei „TelproGWT.gwt.xml“, welche man in dem Verzeichnis „src/de/berlin/fu/berlin“ findet, wird definiert welche onModuleLoad-Methode welcher Klasse (in unserem Fall TeproGWT) aufgerufen werden soll.
Eine wichtiges Interface in diesem Projekt ist das Service-Interface „MyServer.java“, welches sich in dem Ordner „src/de/berlin/fu/shared“ findet. Dieses gibt vor, welche Methoden von der Clientseite genutzt werden können um mit dem Backend zu kommunizieren. Damit nun die Clientseite die Methoden der Serverseite nutzen kann, muss ein sogenanntes Async-Interface erstellt werden. Dieses befindet sich auch in dem Ordner „src/de/berlin/fu/shared“ und heißt „MyServerAsync.java“. Es wird ein asynchrones Interface definiert, da man vermeiden möchte, dass die Anwendung beim Aufruf der Methoden der Serverseite einfriert und der Anwender warten muss. Google Webtoolkit gibt diese Art von Interface beim GWT-RPC immer vor. In der Klasse, welche die Benutzeroberfläche implementiert, wird dieses Async-Interface initialisiert. In den anderen Packages befindet sich die Implementation der Backendseite, welche in dem Kapitel 6 (Backend) näher erläutert wird.

7.3 Code

Die Klasse TelproGWT.java ist die Klasse, in der der Code der Clientseite steht. Jede Klasse in GWT, welche den Client-Code enthält, implementiert das Interface „EntryPoint“. Dabei wird von GWT vorgeschrieben, dass die Methode „onModuleLoad“ implementiert werden muss. Diese Methode wird aufrufen, sobald der Anwender die Webseite in dem Browser aufruft.
Wird die onModuleLoad-Methode aufgerufen, so holt sich die Clientseite als erstes alle Propertytypen und Eventtypen die in der Datenbank abgespeichert wurden. Diese Typen werden an mehreren Stelle im Code benötigt. Um nicht mehre Datenbankanfragen und somit die Anwendung langsamer zu machen, werden die Typen einmal abgefragt und in einer HashMap abgespeichert.
Mit den Propertytyen kann nun dynamisch die Ansicht für die Liniendiagramme aufgebaut werden. Pro Propertytyp wird ein Tab mit einem Liniendiagramm aufgebaut und in das Tab-Menü eingepflegt. Dieses Tab-Menü ermöglicht es, dass nur das Diagramm aktualisiert wird, welches gerade angezeigt wird. Da dieses alle 3 Sekunden aktualisiert wird, war dieser Schritt nötig, da dieses den Datenverkehr verringert und damit auch die Anwendung schneller macht.
Die Events werden in den 3 Sekunden immer mit aktualisiert und es ist egal, ob der Nutzer sich die Eventtabelle ansieht oder nicht. Löst ein Event nun eine Aktion aus (welche in dem Trigger definiert wurde), so soll der Anwender informiert werden (zum Beispiel über ein Dialogfenster), wenn dieser eintritt. So erhält man ein Warnsystem, welche wichtig für dieses System ist.
Durch GWT können nicht nur Oberflächenelemente eingefügt werden, sondern auch gestaltet werden. Zum Beispiel kann die Größe, Farbe, mit oder ohne Rahmen und

vieles mehr eingestellt werden. Wie die Oberfläche aussieht, kann dem nächsten Kapitel und den darin enthaltenden Bildern entnommen werden.

7.4 Screenshots

Um einen Überblick über das Interface zu bekommen, sollen hier einige Screenshots gezeigt werden.

Abbildung 3: Auswahl der Sensoren in einer Drop-down-Box.

Abbildung 4: Wurde ein Sensor ausgewählt, so wird diese Ansicht angezeigt. Oben rechts befinden sich die Sensorinformationen, welche man bearbeiten kann. In der Mitte findet man die Einstellungsmöglichkeiten für Ansicht der Diagramme. Zum einen kann die Anzahl der Punkte in dem Diagramm eingestellt werden, zum anderen wie viele Punkte übersprungen werden sollen („spreading factor“).

Abbildung 5: Das ist die Eventliste-Ansicht. Die Einstellungsmöglichkeiten für die Diagramme sind ausgestellt.

Abbildung 6: Diese Ansicht wird gezeigt, wenn der Nutzer auf Bearbeiten des Sensorknotens klickt. Der Ort des Sensors kann bearbeitet werden, die IP und die ID allerdings nicht.

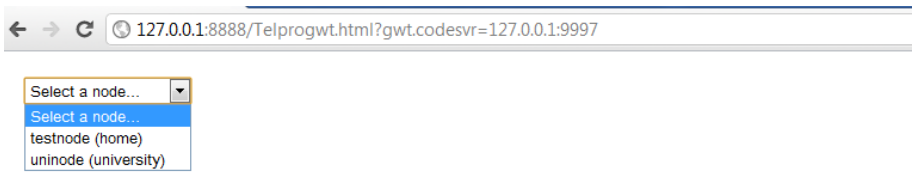


Abbildung 3: Screenshot

8 Zusammenfassung und Ausblick

Ziel dieses Projektes war es für uns, sich mit verschiedenen Technologien zu befassen und verschiedene Ebenen der Programmierung zu benutzen. Wir haben Konzepte

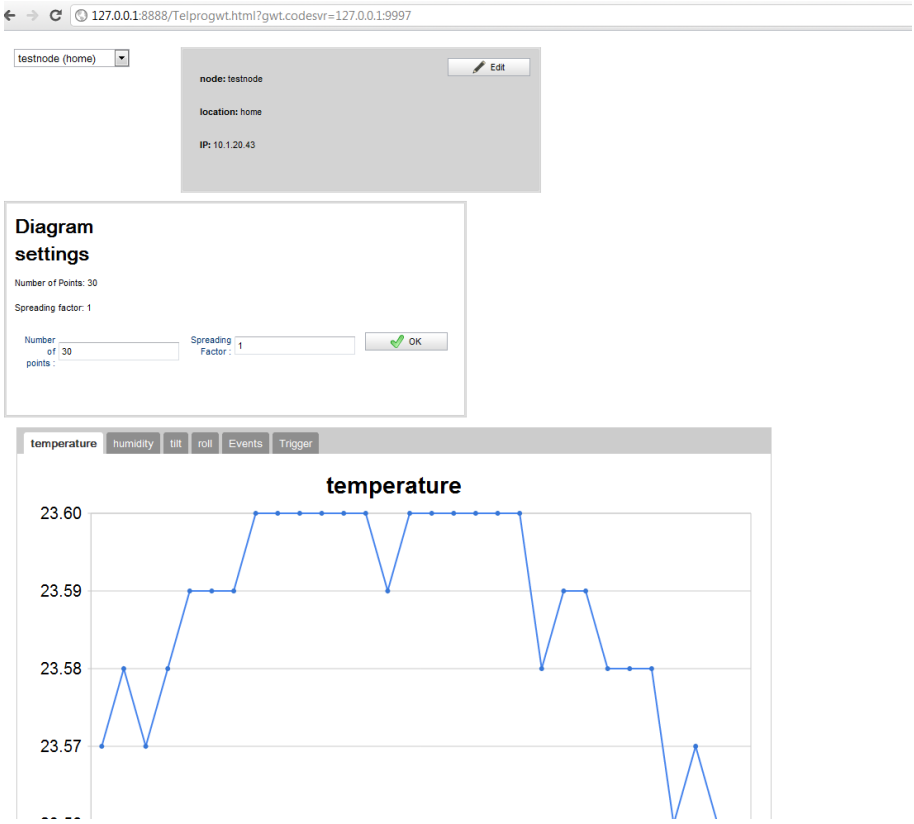


Abbildung 4: Screenshot

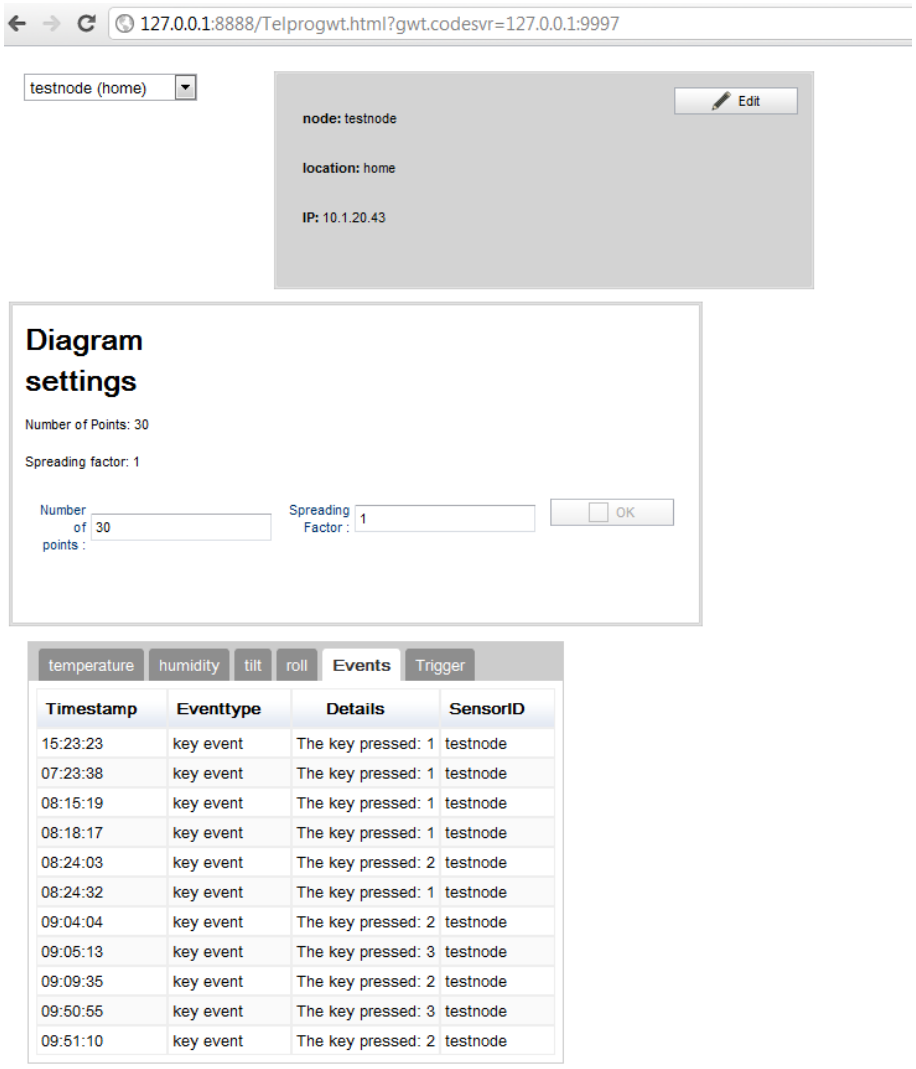


Abbildung 5: Screenshot

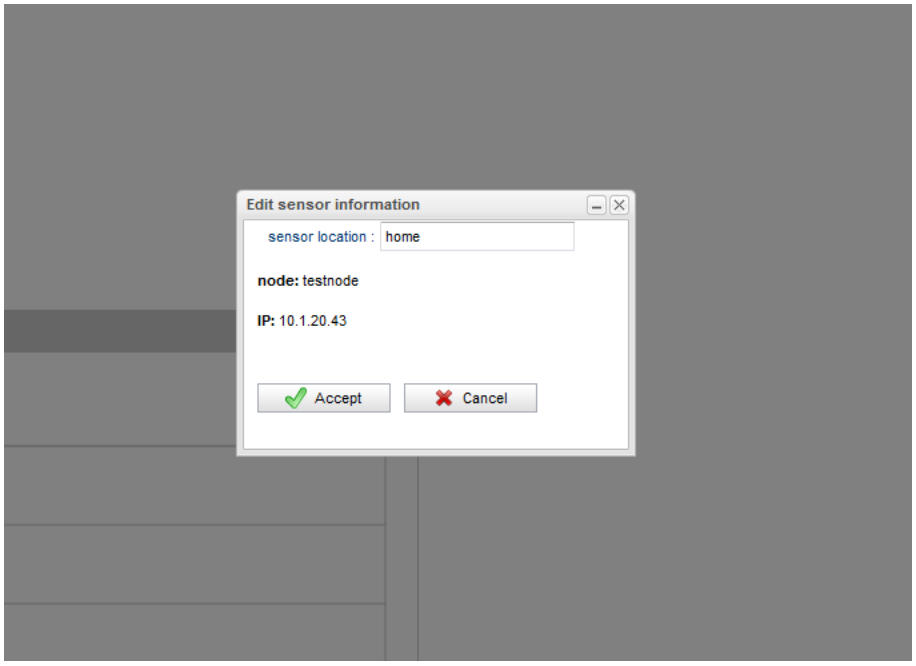


Abbildung 6: Screenshot

von administrativen Dingen, wie MYSQL-Datenbanken, Linux und Tomcat-Servern, über die hardwarenahe Programmierung und Arbeit mit externen Hardware-Modulen (WLAN) bis hin zu hochsprachigen Web-Plattformen, welche viele Dinge abstrahieren, aber auch neue Schwierigkeiten bereiten, erlernt und zum ersten Mal viele kleine Systeme zu einem großen Ganzen zusammengefügt. Der entstandene Prototyp ist so sicher nicht praktikabel, allerdings haben wir so weit, wie möglich, die Erweiterbarkeit in den Vordergrund gestellt, sodass es leicht zu einem simplen Home-Automatisation-Framework werden kann. Wir haben im Verlauf des Projekts viele Erfahrungen gesammelt, von denen wir einige hier beschreiben wollen.

8.1 Erfahrungen

insert stuff here

8.2 Erweiterungen

Abschließend wollen wir aufzeigen, welche Erweiterungen interessant für das System wären.

teilweise siehe architektur, sonst blablabla

9 Appendix

Literatur

[1] Google Charts. http://code.google.com/p/gwt-google-apis/downloads/list.

[2] Google Web Toolkit. https://developers.google.com/web-toolkit/.

[3] Google Web Toolkit Plugin. http://dl.google.com/eclipse/plugin/3.7.

[4] Linux Voyage. http://linux.voyage.hk/.

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

[6] Smart GWT Download link. http://code.google.com/p/smartgwt/downloads/list.