Projektdokumentation

Michael Kaufmann

OGEMA Smart Home

Projektidee / Projektziele

In vielen Bereichen, z.B. bei der Lagerung bestimmter Güter, sind konstante Luftfeuchtigkeits- und Temperaturwerte zwingend nötig um Beschädigungen an den gelagerten Waren zu verhindern. Es soll eine Möglichkeit geschaffen werden dauerhaft Luftfeuchtigkeit sowie Raumtemperatur zu messen und zu protokollieren. Außerdem soll eine Alarmfunktion zur Verfügung stehen, die bei zu großen Abweichungen der Werte auf irgend eine Art einen Verantwortlichen informiert. Die Messdaten sollen über die Arduino-Plattform bereitgestellt werden und grafisch aufbereitet werden. Dabei wird auf dem OGEMA Framework aufgebaut.
Das Projekt wurde als 1 Mann Projekt durchgeführt, wobei die elektronischen Schaltungen für die Messung durch weitere Personen realisert wurde.

Verwendete Technologien

Ogema steht für Open Gateway Energy Management Alliance und basiert auf der OSGi-Service Plattform. Es handelt sich dabei um ein Bundlesystem, bei dem zur Laufzeit Bundles hinzugefügt oder entfernt bzw. gestartet oder gestoppt werden können. Sowohl OSGi als auch OGEMA basieren auf Java.
OGEMA wird von einer Allianz mehrerer Firmen entwickelt, wobei der Hauptteil der Entwicklung derzeit durch das Fraunhofer IWES in Kassel durchgeführt wird.
Mit Ogema soll ein Gateway geschaffen werden um auf verschiedene elektrische Endgeräte zugreifen zu können. Ziel des Frameworks ist es die Möglichkeit zu schaffen alle Smart Grid fähigen Geräte in eine Schnittstelle für den Endbenutzer mit einzubinden und außerdem die Erweiterbarkeit z.B. durch neue Geräte oder Funktionalitäten in einer für den Benutzer einfachen Form sicher zu stellen. OGEMA wurde in erster Linie ausgewählt um später eine Möglichkeit zu haben dynamisch beliebig viele weitere Messstationen hinzufügen zu können.
Sämtliche Dokumentation für OGEMA bezieht sich auf die Entwicklungsumgebung Eclipse. Aus diesem Grund wurde Eclipse als Entwicklungsumgebung für dieses Projekt ausgewählt. Durch die Wahl des OGEMA Frameworks ist ebenso Java als Programmiersprache vorgegeben. Innerhalb von OGEMA werden Java sowie Java Server Pages verwendet. Einige Teile sind auch in reinem HTML, so dass für das Projekt ebenso geringe Kenntnisse von HTML nötig waren.
Die einzelnen Bundles sowie das OGEMA Framework werden über das Versionskontrollsystem Subversion (SVN) zur Verfügung gestellt.

Auf der Hardwareseite sollte eine möglichst kostengünstige Lösung gefunden werden um die Messdaten bereitstellen zu können. Hier bietet sich der Arduino Prozessor an, der über einen Sensor die Messdaten geliefert bekommt um diese dann an einen angeschlossenen PC oder an ein anderes Gerät zu übertragen von dem die Daten dann an einen Server weitergeleitet werden. Die Weiterleitung an den PC soll nach Möglichkeit über das HTTP-Protokoll funktionieren. Zunächst werden die Prozessoren jedoch über den USB-Port direkt an einem PC oder Notebook angeschlossenen.
Als Datenbank war bereits H2 im OGEMA Framework vorhanden. Dabei handelt es sich um eine sehr schlanke Open Source Datenbank. Um nicht verschiedene Datenbanken einsetzten zu müssen wird diese Datenbank auch für alle anderen Funktionalitäten die ein abspeichern oder laden von Daten erfordern eingesetzt.
Zur Visualisierung der Daten gab es im OGEMA Framework noch keine eindeutige Lösung. In anderen Erweiterungen wird teilweise Jframe als Basis für grafische Anzeigen wie Charts verwendet, dies ist jedoch im Moment nicht Bestandteil des OGEMA Frameworks. Für dieses Projekt wurde schließlich mit Flot eine Javascript Bibliothek für Chartdarstellungen ausgewählt. Im Gegensatz zu JFrame ist die Einarbeitungszeit in Flot kürzer somit können innerhalb kurzer Zeit ansprechende Lösungen realisiert werden.
Durch die Entscheidung das OGEMA Framework einzusetzen wurden also einige weitere Komponenten automatisch mit ausgewählt.

Projektdurchführung (Probleme, etc)

Bei der Durchführung des Projekts sind mehrere Probleme aufgetreten. Die Einarbeitung in das OGEMA Framework hat mehr Zeit in Anspruch genommen als zuvor angenommen worden war. Außerdem gab es während der Entwicklung in seltenen Fällen auch noch Änderungen an einzelnen Modulen des Frameworks. Nicht alle Module in OGEMA integrierten Module sind bisher komplett funktionstüchtig, so dass diese nicht alle als Lehrbeispiele dienen konnten. Leider traten auch beim starten des Frameworks bzw. beim einbinden der einzelnen Bundles ab und zu nicht nachvollziehbare Probleme auf. Meistens lag dies vermutlich an einer Fehlbedienung, da aber beim starten der einzelnen Bundles kein komplettes Logging erfolgte blieb dies in den meisten Fällen eine Vermutung. Das Problem ist aufgetreten, da in Eclipse beim Build nur die Fehlermeldungen des jeweils letzten gestarteten Bundles angezeigt werden und von später gestarteten Bundles direkt wieder überschrieben werden. Bei jedem Build werden jedoch mehrere Bundles gestartet, zumal es auch Abhängikeiten unter den Bundles gibt, so dass es auch nicht möglich ist nur ein einziges Bundle zu starten um die Fehlermeldungen komplett zu bekommen. Der Versuch die Fehlermeldungen in einer Datei zu protokollieren wurde zwar vom

Fraunhofer IWES unternommen, ist jedoch daran gescheitert dass es ab und zu zu Fehlern beim Dateizugriff gekommen ist da die Datei offenbar ab und zu noch vom letzten Bundle das eine Fehlermeldung protokolliert hat gelockt war, während durch das nächste gestartete Bundle versucht wurde wieder auf die Datei zuzugreifen. Auch das Verhalten auf verschiedenen Betriebssystemen war nicht immer identisch, obwohl dies eigentlich keinen Unterschied machen sollte. Die Entwicklung hat deshalb hauptsächlich unter Windows Vista stattgefunden.
Ein weiteres bisher ungelöstes Problem tritt beim Anschluss des Sensors an einen PC über die USB-Schnittstelle auf. Hier wird vom System bei jedem einstecken des Kabels ein anderer COM-Port zugewiesen über den die Messdaten dann ausgelesen werden können. Dies macht ein automatisches auslesen allerdings schwierig, da spätestens nach jedem Rechnerneustart der COM Port manuell neu konfiguriert werden müsste.
Das auslesen der Daten am COM-Port ist jedoch vom Betriebssystem abhängig. Während für Linux unter Java mit der Java COM Api eine Möglichkeit existiert auf USB Ports zuzugreifen wird im Moment keine offizielle API zum Zugriff auf USB Ports unter Windows bereitgestellt. Mit jUSB gibt es aber zumindest eine Möglichkeit durch einen Drittanbieter auf USB-Ports über Java zuzugreifen. Es ist aber nicht zwingend nötig diese Daten per Java abzufragen. Eine Alternativlösung besteht darin die Daten über Python auszulesen und anschließend über HTTP weiter zu senden. Vom Arduino Prozessor werden Luftfeuchtigkeit sowie Raumtemperatur hintereinander als im ASCII-FORMAT codierte Strings übertragen. Python stellt mit dem Serial Modul eine Möglichkeit zur Verfügung den USB Port unter Windows abzufragen. Es bleibt lediglich das Problem dass der COM Port manuell angegeben werden muss.
Außerdem ist es im Moment nur möglich die Daten etwa alle 5 Sekunden abzufragen. Bei einer häufigeren Abfrage würde der Prozessor überhitzen. Dies könnte zum Beispiel durch höherwertige Hardware gelöst werden wobei ein Zeitintervall von 5 Sekunden für die meisten Anwendungen auch ausreichend ist.
Da es für die grafische Darstellung der Daten bisher keine Funktion in OGEMA gegeben hat wurde als Alternativlösung eine Javascript-Lösung implementiert. Bis diese Lösung gefunden worden ist hat es mehrere vergebliche Versuche gegeben die Daten sauber darzustellen. Die Daten mit reinen HTML Funktionen inklusive Javascript anzuzeigen war zwar möglich, allerdings fehlten hierzu die Möglichkeiten Charts zu erzeugen, so dass die Darstellung weniger Benutzerfreundlich gewesen wäre. Auch der Verlauf der Temperatur- / Luftfeuchtigkeit ist in einem Chart am besten zu erkennen.
Für die Protokollierung der Daten wurde eine eigene Datenbanktabelle erstellt in die die Daten eingetragen werden. Für diese Protokollierung solltte bei Fortführung des Projektes noch eine Verdichtung realisiert werden, da es auf Dauer nur bedingt Sinn macht die Daten alle 5 Sekunden in eine Datenbank

einzutragen. Läuft diese Lösung einige Wochen würde die Datenbank immer weiter wachsen und die Abfrage der Daten würde auf Dauer aufgrund der Datenmenge langsamer werden. Getestet werden konnte dies allerdings nicht, da die Lösung bisher nicht im Dauerbetrieb erprobt worden ist und derartige Probleme erst zu einem späten Zeitpunkt auftreten würden. Die Protokollierung der Daten in die Datenbank lässt sich auch abschalten.
Auch eigene Fehler konnten zu Problemen führen, z.B. wenn für bestimmte Aktionen wie das starten des Webservers auf Port 80 Administratorrechte notwendig und Eclipse eben nicht mit diesen gestartet wurde oder für ein anderes Programm bereits ein Webserver auf demselben Port gestartet war und durch das Framework somit kein Start eines Webservers möglich war.

Beispiel aus dem Projekt: Versenden von Emails:

```
public class SendMail {

	String receiver = "", subject = "", message = "";

	public void sendMail()
	throws MessagingException {
		Properties prop = new Properties();
		prop.put("mail.smtp.host", "smtp.hdm-stuttgart.de");
		prop.setProperty("mail.smtp.ssl.trust", "smtp.hdm-stuttgart.de");
		prop.put("mail.smtp.auth", "true");
		prop.put("mail.smtp.port", "25");
		prop.put("mail.smtp.starttls.enable","true");
		Session session = Session.getDefaultInstance(prop,
				new javax.mail.Authenticator() {

			protected PasswordAuthentication getPasswordAuthentication() {
			return new PasswordAuthentication("mk146", "XXX");
			}
			});
		Message msg = new MimeMessage(session);
		InternetAddress senderaddress = new InternetAddress("mk146@hdm-
stuttgart.de");
		msg.setFrom(senderaddress);
		InternetAddress receiverAddress = new InternetAddress(receiver);
		msg.setRecipient( Message.RecipientType.TO, receiverAddress);
		msg.setSubject(subject);
		msg.setContent(message, "text/plain");
		Transport.send(msg);
}
```

In diesem Beispiel wird eine EMail über den Email Account der HdM von Java aus versendet. Entsprechend der Einstellungen der HdM wird die EMail über den Port 25 gesendet und SSL bzw. TLS verwendet. Dazu müssen auch Benutzername sowie Passwort eingegeben werden. Die eigentliche Nachricht wird in diesemm Fall als reiner Text gesendet, der Text sowie Empfänger und Sender werden außerhalb dieser Methode gesetzt. In einem möglichen Produktivbetrieb sollte für diese Funktionalitäten eine eigene Email-Adresse zur Verfügung stehen von der dann Warn-Mails

gesendet werden können.
Zum versenden von Emails muss die JavaMail API eingebunden werden. Diese API steht unter einer Open Source Lizenz zur Verfügung.

Eigener Lernerfolg

Da in diesem Projekt mehrere verschiedene Technologien verwendet worden sind konnte ein Lernerfolg in unterschiedlichen Bereichen erzielt werden.
Zu diesen Erfolgen gehören ein Einblick in die Konzepte der OSGi-Service Plattform sowie des damit verbunden Bundle-Konzeptes.
Dazu gehören auch einige Bereiche der Entwicklungsumgebung Eclipse bei der Entwicklung im Bereich von Packages sowie dem Plugin Management, sowie generell das Debugging mit Eclipse.
Die Struktur in einem Projekt dieser Größenordnung, an dem mehrere Personen beteiligt sind und das bereits einen größeren Umfang an Quellcode besitzt, zu sehen war lehrreich.
Auch die bereits in Java realisierte Umsetzung anderer Module zu sehen trug erheblich zu meinem persönlichen Lernerfolg bei.
Einen erheblichen Lernerfolg in diesem Projekt ist durch die Programmierung in Java sowie Java Server Pages in einem realen Projektumfeld eingetreten.
Eine weitere wichtige Lehre aus dem Projekt ist es das es sich in vielen Fällen lohnt auf bereits existierende Bibliotheken zurück zu greifen und damit möglichst wenig 'from scratch' zu entwickeln wie in diesem Fall bei der Grafikbibliothek.
Neben diesen positiven Aspekte konnte ich aus dem Projekt auch Lehren ziehen was besser nicht gemacht werden sollte. So sollte man während der Entwicklung nicht auf verschiedenen Rechner und unterschiedlichen Betriebssystemen entwickeln, treten hier Unterschiede auf sollten die Funktionen zunächst auf einem Betriebssystem vollständig entwickelt werden und anschließend gegebenenfalls eine Portierung vorgenommen werden. Wenn für das Projekt nötig können auch Zwischenlösungen oder noch nicht zu 100% perfekte Lösungen zur weiteren Entwicklung genutzt werden. Außerdem wäre das definieren von ausreichenden Testcases zwingend notwendig.

Weitere Lehren aus dem Projekt / Fazit

Es stellt sich die Frage ob es die richtige Entscheidung war das OGEMA Framework für dieses Projekt einzusetzen. Für die in diesem Projekt gestellten Aufgaben und Ziele ist dieses Framework sicherlich überdimensioniert und der Aufwand für die Einarbeitung in das Framework rechnet sich kaum. Dennoch ist ein Einsatz von OGEMA aufgrund der daraus resultierenden Lernerfolge in einem Studienprojekt vertretbar und bei der Realisierung weiterer Punkte

wie sie in der Zukunftsperspektive beschrieben sind unter Umständen auch sinnvoll.

Zukunftsperspektive

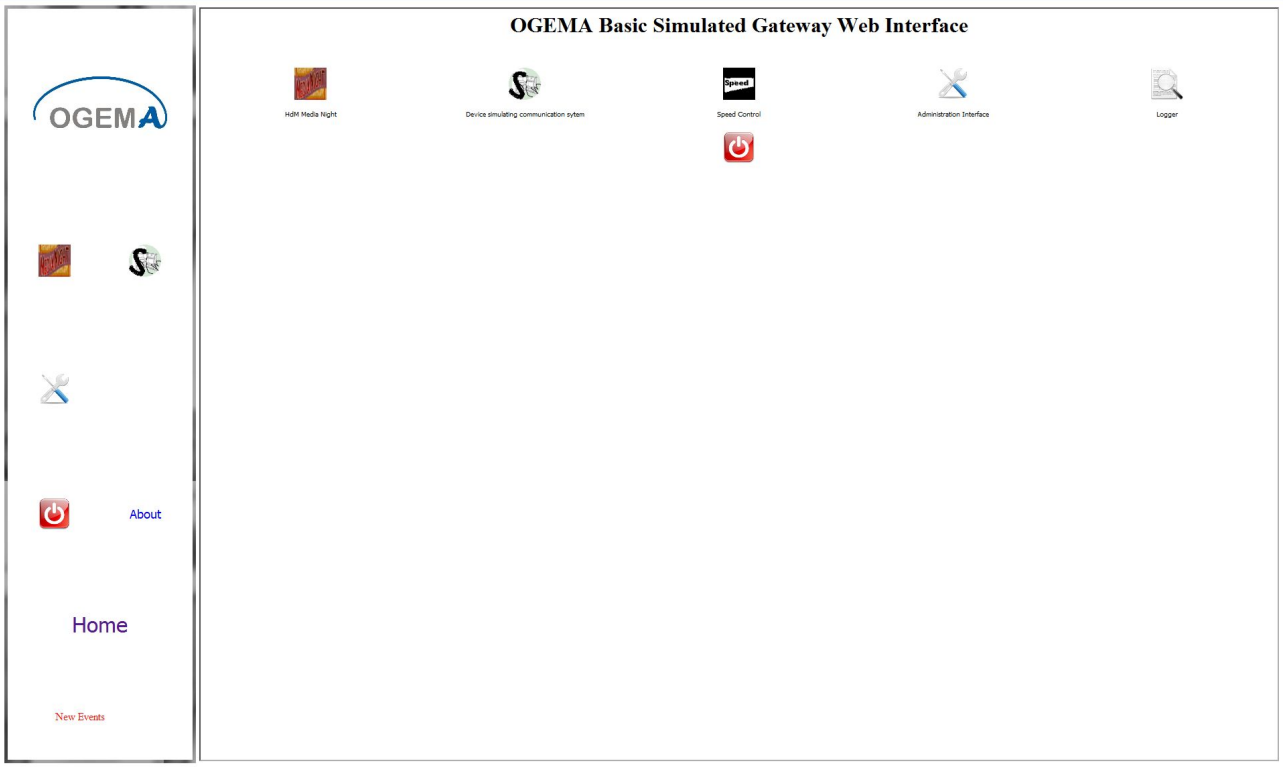
Es müssen noch weitere Funktionen implementiert werden um einstellen zu können wann Warnemails verschickt werden sollen.
Die Lösung soll noch weiter ausgebaut werden. Ob sie innerhalb der HdM im Bereich der Papierlagerung für die Druckmaschinen eingesetzt werden kann ist noch abschließend zu prüfen. Dazu sind noch weitere Dinge notwendig, wie das bereitstellen eines Servers auf dem die OGEMA-Umgebung dauerhaft läuft, sowie das erstellen weiterer Sensoren zur Messung der Daten, da ein einzelner Messpunkt in diesem Fall nicht ausreichend wäre. Außerdem muss geklärt werden wie die Datenübertragung von den Messgeräten zum Server in einem produktiven Einsatz möglich ist.
Darüber hinaus sind viele weitere Einsatzmöglichkeiten denkbar. So wäre es noch eine Option die Überwachung mittels Smart Phones durchzuführen. Wie das im Einzelfall umgesetzt werden könnte und ob für jedes Betriebssystem (Android, iOS, Windows Phone, …) eine eigene App nötig müsste ebenfalls erst noch überprüft werden.
Es besteht auch die Option die Funktionen aus dem OGEMA Framework herauszulösen um so eine schlankere Lösung zu haben. In diesem Fall muss ein eigener Webserver eingerichtet werden, der ja ansonsten bereits durch das Framework zur Verfügung steht, sowie Methoden geschrieben werden um mehr als eine Messstation zu unterstützen.

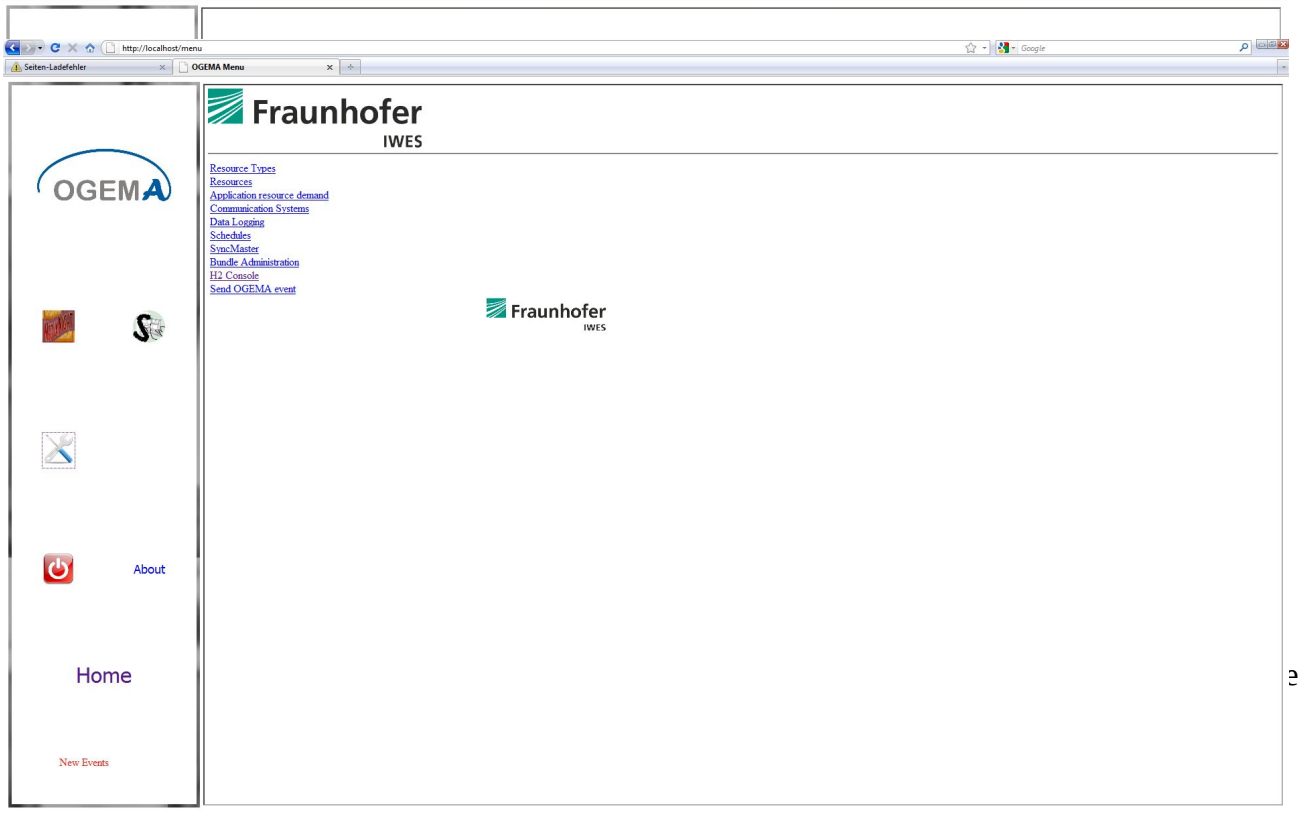
Theoretisch denkbar wäre es auch das Projekt so zu erweitern, dass die Temperatur automatisch auf einem bestimmten Niveau gehalten wird. Dies wäre möglich indem z.B. eine Heizungssteuerung angesprochen wird die entsprechend der gemessenen Temperatur ein- und ausgeschaltet wird. Ähnliche Prozeduren wären sicher auch für die Luftfeuchtigkeit denkbar. Diese Art der Erweiterung würde auch den Einsatz des OGEMA-Frameworks sinnvoller erscheinen lassen.

Eine weitere Möglichkeit besteht darin auch andere Daten zu messen. Hier sind der Fantasie fast keine Grenzen gesetzt, sofern die Daten auf irgend einem Weg elektronisch übermittelt werden können.

Anhang: Screenshots

Überblick über die generelle OGEMA User-Schnittstelle / Anzeige von Grid der Messfunktion


OGEMA Basic Simulated Gateway Web Interface
HdM Media Night
Device simulating communication sytem
Speed Control
Administration Interface
Logger
OGEMA
About
Home
New Events


http://localhost/menu
Seiten-Ladefehler
OGEMA Menu
Fraunhofer
IWES
Resource Types
Resources
Application resource demand
Communication Systems
Data Logging
Schedules
SyncMaster
Bundle Administration
H2 Console
Send OGEMA event
Fraunhofer
IWES
OGEMA
About
Home
New Events