

Institut für Visualisierung und Interaktive Systeme
Universität Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Diplomarbeit Nr. 3496

Plattform für Sensoren und Aktoren in drahtlosen eingebetteten Geräten

Sven Plohmer

Studiengang: Softwaretechnik
Prüfer: Prof. Dr. Albrecht Schmidt
Betreuer: Thomas Kubitza M.Sc.

begonnen am: 28. Mai 2013
beendet am: 27. November 2013

CR-Klassifikation: C.2.4, C.3, C.5.3

Kurzfassung

Nur wieder ein weiteres Sensor Netzwerk? Diese Frage kann klar mit Nein beantwortet werden. Die hier vorgestellte Plattform ist mehr als nur ein Sensornetzwerk. Es ermöglicht es ein heterogenes Netzwerk von Sensoren und Aktoren zu verwalten. Dabei ist es möglich Mechanismen auf einfache Weise zu erstellen um Sensor Input mit Aktor Output zu verknüpfen. Dieses System soll es ermöglichen mittels unterschiedlichster Geräte interaktive Umgebungen zu gestalten. So können diese Mechanismen, hier auch Regeln genannt, steuern was passiert wenn bestimmte Geräte und deren Sensoren bestimmte Werte liefern. Man kann damit erreichen, dass durch diese Regeln dann die Aktoren von ganz anderen Geräten gesteuert werden können. Durch eine Abstraktion der Kommunikation und der Geräte Programmierung wird erreicht, dass auch Personen, welche keine Erfahrung mit der Mikrocontroller-Programmierung haben, diese Plattform effizient einsetzen können. Diese Arbeit stellt das Konzept zu dieser Plattform vor und zeigt die Details der Implementierung des Systems. Anschließend wird die Plattform durch eine Benutzerstudie evaluiert und am Schluss die Vor- und Nachteile des Systems diskutiert.

Inhaltsverzeichnis

1	Einleitung	9
1.1	Motivation und Problemstellung	9
1.2	allgemeiner Lösungsansatz	10
1.3	Gliederung	10
2	Grundlagen	13
2.1	Begriffe	13
2.1.1	Sensor	13
2.1.2	Aktor	13
2.1.3	Sensor Netzwerk	13
2.1.4	Rapid Prototyping	14
2.2	Related Work	14
2.2.1	Sensor Netzwerke	14
	Sense Web	14
	Web Mashups for Embedded Devices	15
	dynam	15
2.2.2	Rapid Prototyping	16
	Gadgeteer	16
2.2.3	visuelle Programmierung	17
	Scratch	17
2.2.4	Unterschiede zur eigenen Lösung	18
3	Konzept	19
3.1	Einführung	19
3.2	Szenario	19
3.3	Zusammenfassung des Konzepts	21
3.3.1	Server	23
3.3.2	Geräte	23
3.3.3	Regeln	24
3.3.4	Weboberfläche	25
	Geräte	26
	Regeln	28
	Sensordaten Monitor	30
3.4	Vorteile	31
4	Implementierung	35
4.1	Einführung	35

4.2	Eingesetzte Technologien	36
4.2.1	Raspberry Pi	36
4.2.2	Gadgeteer	37
4.2.3	Arduino	37
4.2.4	Node.js	37
4.2.5	JavaScript und CoffeeScript	38
4.2.6	JSON	38
4.2.7	xBee	38
4.3	Gesamtsystem	38
4.3.1	Architektur	38
4.3.2	Kommunikation und Schnittstellen	39
	xBee	40
	UDP	40
	Konfigurationen	41
4.4	Server	43
4.4.1	Architektur	43
4.4.2	Datenbank	45
4.4.3	Kommunikation	45
4.4.4	Regeln	45
4.4.5	Weboberfläche	46
4.4.6	Erweiterbarkeit	46
4.5	Geräte	46
4.5.1	Allgemein	46
	Architektur	46
4.5.2	Gadgeteer	47
	Architektur	47
	Verwendetes Mainboard	48
	Unterstützte Module	48
	Erweiterbarkeit	51
4.5.3	Arduino	52
	Architektur	52
	Verwendetes Mainboard	52
	Unterstützte Module	53
	Erweiterbarkeit	53
4.6	Weboberfläche	54
4.6.1	Geräte	54
4.6.2	Regeln	55
4.6.3	Aktoren steuern	56
4.7	Herausforderungen	56
5	Evaluation	59
5.1	qualitative Evaluation	59
5.2	quantitative Evaluation	64
5.2.1	Bandbreite	64
5.2.2	Latenz	64

5.3 Zusammenfassung	65
6 Zusammenfassung	67
6.1 Zusammenfassung	67
6.2 Diskussion	67
6.3 Future Work	68
Literaturverzeichnis	71

1 Einleitung

1.1 Motivation und Problemstellung

Die Menschen von heute sind heute immer mehr und mehr von Technik umgeben und nutzen diese auch im Alltag. So kann man unterwegs mit seinem Handy allerlei Multimedia genießen oder witzige Spiele mit dem Touchscreen spielen. Auch zu Hause ist es möglich den Fernseher nicht mehr mit der Fernbedienung zu steuern, sondern über Gesten. Es wird immer wichtiger wie man mit etwas interagieren kann. So findet sich heutzutage der Benutzer mit keinem Programm auf dem Computer ab, welches unlogisch aufgebaut ist und wo man sich durch unzählige Untermenüs klicken muss, um am Ende mit einer unverständlichen Fehlermeldung zu Enden. Auch wird es dem Benutzer immer wichtiger nicht nur Multimedia zu konsumieren sondern auch damit zu interagieren. Aus diesem Grund nehmen sogenannte Casual Games einen immer wichtigeren Stand ein.

Die meisten heutigen Museen haben das gleiche Problem: Sie bieten meisten nur starre Objekte an, mit denen man nicht interagieren kann, sondern nur die Bilder oder Skulpturen konsumieren kann. Wenn man heutzutage in eine Bildergalerie geht, kann man diese nur ansehen und die Informationen daneben lesen. Man kann nicht beeinflussen, welche Informationen angezeigt werden, weil einen vielleicht die Informationen über den Künstler mehr interessieren als die Geschichte des Bildes. Das Problem dabei ist, dass sich die heutige Jugend nicht mehr nur mit der Möglichkeit des Konsums zufrieden gibt, es muss etwas neues her.

Aus diesem Grund geht es in dieser Diplomarbeit um die Möglichkeit die Umgebung durch Sensoren und Aktoren interaktiv zu gestalten. In erster Linie ist diese Arbeit auf das Szenario für den Einsatz in Museen gedacht, kann aber natürlich auch auf viele andere Szenarien angewendet werden: für das Büro oder zu Hause. In diesem Umfeld kennen sich die Benutzer meist nicht mit Technik aus und aus diesem Grund muss dieses System es leisten, dass auch eine Person ohne Kenntnis von Programmierung, Sensoren und Aktoren in der Lage ist, dieses System zu benutzen. Deshalb muss die Konfiguration der Geräte und die Mechanismen zur Automatisierung so anlegbar sein, dass auch ein Nicht-Informatiker dieses System ohne größeren Aufwand benutzen kann. Es soll die Entwicklung erleichtern und nicht erschweren.

Das Ziel dieser Diplomarbeit ist es eine Plattform zu erschaffen, welches die Integration von Sensoren und Aktoren in die Umgebung unterstützt. Es soll eine Plattform konzipiert und erstellt werden, welche es ermöglicht die Umgebung intelligenter zu gestalten, ohne sich näher mit der Implementierung der einzelnen Geräte beschäftigen zu müssen. Um dies zu erreichen sollen verschiedenste Geräte in Verbindung mit Sensoren und Aktoren eingesetzt

werden. Das Ziel ist es, dass die Plattform eine heterogene Vielfalt von Geräten zulässt, ohne die Komplexität zu sehr zu steigern. So soll es möglich sein Geräte unterschiedlicher Hersteller und unterschiedlicher Architektur zu benutzen. Es sind dabei sowohl tragbare Mikrocontroller, mobile Geräte und auch kleine stationäre Computer denkbar. Um diese unterschiedlichen Geräte in das System integrieren zu können, ist es notwendig mehrere Kommunikationskanäle anzubieten. So sind Kommunikationsschnittstellen denkbar wie Wifi, xBee, Bluetooth oder andere drahtlose Kommunikationen. Weiterhin wird ein Szenario für einen möglichen Einsatzzweck des Systems im Umfeld des Museums und Büros vorgestellt, welches Beispiele für Anwendungsmöglichkeiten der Plattform geben sollen.

1.2 allgemeiner Lösungsansatz

Das System soll aus einem Server und mehreren Geräten bestehen. An die Geräte können dann Sensoren und Aktoren angeschlossen werden oder es sind schon Sensoren und Aktoren fest eingebaut wie bei einem Smartphone mit Display und Beschleunigungssensor. Die Geräte können dabei mit dem Server über verschiedenste Kommunikationsarten wie Wifi, xBee oder Bluetooth kommunizieren. Um die Reichweite des Servers für die Funkübertragung zu vergrößern, können Extender eingesetzt werden, welche die Reichweite der drahtlosen Kommunikation erweitern können. Diese stellen zusammen mit den Kommunikationsmöglichkeiten des Servers die Infrastruktur des Systems dar. Für jedes Gerät gibt es eine Repräsentation auf dem Server, welche durch den Benutzer mittels einer Weboberfläche und einem graphischen Assistenten erstellt wird. Wenn ein Gerät eingeschaltet wird, meldet es sich am Server an und übermittelt ihm seine eigenen Fähigkeiten. Dabei handelt es sich um Dinge wie was für Anschlussmöglichkeiten es besitzt oder welche Kommunikationsform verwendet wird. Anschließend sendet der Server dem Gerät seine hinterlegte Konfiguration und damit weiß das Gerät welche Sensoren, bzw. Aktoren, jeweils angeschlossen sind und kann diese initialisieren. Das Gerät sendet nun die Sensor Informationen an den Server, auf dem diese nun weiterverarbeitet werden. Der Benutzer kann nun Mechanismen, bzw. Regeln auf dem Server anlegen, welche es ermöglichen, dass ein oder mehrere Sensorwerte, vielleicht auch von unterschiedlichen Geräte, Aktoren auslösen können. Durch diesen Mechanismus wird die Interaktive Komponente im System dargestellt. Dabei soll es für den Benutzer möglich sein diese Mechanismen ohne große Kenntnisse der Programmierung zu erstellen. So ist es z.B. denkbar, dass es durch eine graphische Programmiersprache, welche einfach zu erlernen ist, geschieht. Der Grundgedanke hinter diesem System ist, die Programmierung so weit zu abstrahieren, dass es jeder Bedienen kann.

1.3 Gliederung

Diese Diplomarbeit ist so aufgebaut, dass im nachfolgenden Kapitel 2 die Grundlagen dieser Arbeit behandelt werden. Das bedeutet, dass wichtige Begriffe definiert werden und auch beleuchtet wird, welche verwandten Arbeiten es zu diesem Thema gibt. Im Kapitel 3 wird das Konzept der Plattform mit einem Szenario eingeleitet und anschließend das Konzept

selbst vorgestellt. Danach wird die Implementierung im Kapitel 4 beschrieben. Dies bezieht sich im wesentlichen auf den allgemeinen Aufbau der Plattform und der Beschreibung der einzelnen Komponenten. Im darauf folgenden Kapitel 5 wird sich der Evaluation des Systems gewidmet. Darin wird sowohl auf eine qualitative, als auch auf die quantitative Evaluation eingegangen. Das bedeutet, dass sowohl Usability als auch Performance des Systems untersucht wird. Das letzte Kapitel 6 beschäftigt sich mit einer Zusammenfassung der Ergebnisse, sowie der Diskussion und Bewertung dessen und einem Ausblick in die Zukunft dieses Systems.

2 Grundlagen

2.1 Begriffe

In diesem Abschnitt werden wichtige Begriffe definiert bzw. erklärt, welche für diese Arbeit wichtig sind.

2.1.1 Sensor

In [Wis13c] und [Dil13] wird ein Sensor definiert als ein System oder eine Komponente, welche es ermöglicht physikalische Größen, deren Änderungen oder einen chemischen Effekt in ein elektronisches Signal umzuwandeln. Dieses elektronische Signal kann dann durch einen Rechner weiterverarbeitet werden. Mögliche physikalische Größen, welche mittels Sensor ermittelt werden können, sind z.B. Druck, Beschleunigung, Lichtstärke, Temperatur oder Schall. Manche Sensoren benutzen dabei diese physikalischen Größen um auf andere Dinge zu schließen. So wird z.B. bei der Entfernungsmessung ein Ultraschall ausgesendet und dessen Laufzeit bis zur Reflektion gemessen um so auf die Entfernung zu schließen. Ein Sensor dient also als Eingabe in ein elektrisches System.

2.1.2 Aktor

In [Wis13a] wird ein Aktor als Gegenstück zum Sensor beschrieben. Entsprechend wandelt er elektrische Signale in andere Energieformen um. So z.B. Schall in Form von Audio, Temperatur in Form von Hitze oder Kälte oder einfach in Licht, z.B. als Bildschirmausgabe oder in Form einer einfachen LED. Auch ist es möglich die elektrischen Signale in Bewegung umzusetzen indem ein Motor angesteuert wird, welcher auf unterschiedliche elektronische Signale reagiert.

2.1.3 Sensor Netzwerk

In [Hua04] und [Wis13d] wird ein Sensor Netzwerk definiert als ein Netzwerk, welches aus mehreren Knoten besteht. Diese Knoten sind dabei mit Sensoren ausgerüstet und übertragen ihre Daten mittels eines Netzwerkes. Dabei kann dieses Netzwerk sowohl drahtlos, drahtgebunden als auch aus einer Kombination beider aufgebaut sein.

2.1.4 Rapid Prototyping

[Wis13b] nennt das Rapid Prototyping als eine Methode zur schnellen Herstellung eines Prototypen. Dies bezieht sich nicht nur auf mechanische bzw. konstruktive Modelle, wie beim 3D-Druck, sondern kann beispielsweise auch im Bereich der Software für die automatisierte Generierung von Programmcode aus vorhandenen Modellen eingesetzt werden.

2.2 Related Work

In diesem Abschnitt geht es um verwandte Arbeiten, welche in den Bereich dieser Arbeit fallen. Dieser Bereich ist nach Themengebieten sortiert, zu diesen Themengebieten werden jeweils unterschiedliche Arbeiten vorgestellt.

2.2.1 Sensor Netzwerke

Sense Web

Das von [GKN⁺07] vorgestellte Sense Web [Mic13b], wurde von Microsoft entwickelt. Dieses Sensor Netzwerk dient dazu große Daten von Sensor Informationen weltweit für Anwendungsentwickler durch ihr Framework zur Verfügung zu stellen. Dabei sollen Menschen weltweit ihre eigenen Sensordaten zur Verfügung stellen und ähnlich wie bei Wikipedia, wo viele Leute ihr Wissen teilen um zu einer großen Datenbank zu werden, eine große Datenquelle liefern, die von allen benutzt werden kann. Dabei ist es für den Anwender möglich seine Sensoren oder Sensornetzwerke wie z.B. Temperatursensoren am Haus oder der GPS Sensor beim Joggen in Verbindung mit einem Herzmonitor zu teilen. Dadurch das man dann viele Informationen zur Verfügung hat, kann man ein genaueres Temperaturbild eines Landes erstellen, was vielleicht bei der Wetteranalyse behilflich sein könnte. Das Ziel ist geteilte Sensoren verwenden zu können, welche über das gesamte Internet verteilt sind. Um die unterschiedlichen Plattformen mit verschiedenen Kommunikationsarten, Energie, Bandbreite, Leistungen zu verwenden, wird eine Abstraktionsschicht zwischen die Sensoren, bzw. Sensornetzwerke, und die Sense Web API dazwischengeschaltet. Dies geschieht durch sogenannte sensor gateways. Diese kümmern sich um die Kommunikation mit den Sensoren und senden diese Informationen an die standardisierte Web API. Um nun diese Informationen auf dem Server nutzen zu können, gibt es die Data transformers, welche die erhaltenen Sensor Informationen weiterverarbeiten. So ist es z.B. möglich aus einem Videostream die Personenanzahl pro Stunde zu berechnen oder andere Sensordaten zu fusionieren. Als letzte dem Endanwender sichtbare Schicht kommen die Applications. Diese stellen eine Endanwendung für Benutzer dar, welche die Informationen aufbereitet dem Nutzer zur Verfügung stellt. So wäre z.B. eine Karte möglich mit häufigen Joggingstrecken in Verbindung mit der Durchschnittlichen Herzrate, um so einen Überblick über den Schwierigkeitsgrad der Strecke zu bekommen. Das System muss viele Herausforderungen meistern: Es muss mit sehr großen örtlich und zeitlich abhängigen Daten hantieren, außerdem können

die Daten kontinuierlich in das System gelangen oder, wie im vorherigen Beispiel des Joggers, zu bestimmten Zeitpunkten. Außerdem können die Daten jegliche Dimension haben, sei es eine Scalar, ein Vektor, ein Bild oder ein ganzer Videostream. Diese großen Datenmenge können zu einem Problem der Bandbreite führen, welche irgendwann nicht mehr ausreichen kann, wenn nicht immer wieder nachgebessert wird. Insgesamt ist der Hintergedanke des Systemes, dass sich die Kosten des Systems durch die verteilten Sensoren, welche von freiwilligen Usern bereitgestellt werden und durch eine Vielzahl von möglichen Anwendungen amortisieren wird.

Web Mashups for Embedded Devices

In dem Artikel von [GT09] wird ein anderer Weg gegangen als beim Sense Web. Dieser Artikel handelt davon, dass jedes Gerät einen eigenen Webserver startet und damit eine REST Schnittstelle implementiert. Durch diese Schnittstelle soll es für andere Anwendungen möglich sein Sensor Daten abzufragen oder Aktoren auszulösen. Dies soll einfach über HTTP realisiert werden. Das erleichtert es für die meisten Anwendungen notwendige Daten einfach über das Internet abzufragen. Da das Internet heutzutage sowieso fast überall zur Verfügung steht, ist es eine einfache Methode Geräte in eine vorhandene Infrastruktur einzubinden. Für den Fall, dass die Geräte keinen eigenen Webserver und damit auch keine REST-Schnittstelle liefern können, sei es durch zu wenig Rechenleistung oder keiner Unterstützung einer Wifi Verbindung, so wird vorgeschlagen diese mittels eines Zwischengateways zu realisieren. Dieses Gateway übernimmt die Kommunikation mit einem oder mehreren Geräten ohne eigenen Webserver und kapselt die Informationen bei sich als REST-Schnittstelle. Durch diesen Zwischenschritt ist es möglich alle möglichen Arten von Geräten problemlos in das System zu integrieren. Da die REST-Schnittstelle über HTTP realisiert wird, bekommt man alle Funktionen von HTTP gleich mitgeliefert: So können die Daten durch Berechtigung vor fremden Personen geschützt werden, komprimiert werden oder verschlüsselt. Außerdem ist es nicht notwendig wie bei Webservices lange WSDL Dateien auszutauschen um miteinander zu interagieren. Die REST-Schnittstelle ist leichtgewichtig und leicht zu verwenden, hat aber auch Nachteile: Der größte Nachteil ist, dass es nicht so einfach ist für REST einen komplexen Service zu erstellen. Dies ist ein paradoxer Nachteil, da REST eigentlich durch seine Einfachheit heraussticht. Ein weiteres Problem ist, dass die Suche nach einzelnen Geräten nicht genau gleich ist wie die Suche nach einer Website mit Inhalt.

dinam

Eine weitere Methode für Sensor Netzwerke wird in [GBN10] vorgestellt. In diesem Artikel geht es darum, dass man den Entwicklungsprozess vereinfachen und damit schneller machen möchte. Die meisten Sensor Netzwerke werden top-down entwickelt. Bei diesem Vorgehen dauert es sehr lange, wenn man unterschiedlichste Hardware verwenden möchte. Man benötigt für jede einzelne Hardware-Architektur eine komplette Entwicklungsumgebung, Compiler, Linker und viele weitere Software. Das Einrichten solcher Software Umgebungen kann unter Umständen lange Zeit beanspruchen und ist Teil der Entwicklungszeit.

Außerdem wird unterschiedlichstes Wissen für verschiedene Hardware vorausgesetzt, da diese meist nach unterschiedlichen Paradigmen programmiert werden. Die Idee ist deshalb einen Bottom-Up Ansatz zu verfolgen, in dem die einzelnen Sensorknoten so konfiguriert werden, dass sie sie selbst alle wichtigen Dinge wie Entwicklungsumgebung, Programm, Interpreter und Debugger enthält. Zugang zu diesen Dingen soll über ein Weboberfläche realisiert werden, welche es ermöglicht auf alle Dinge zuzugreifen. So kann man sich einfach über das Web auf einem Gerät einloggen und den Quellcode des Sensorknotens durch eine integrierte Entwicklungsumgebung abändern. Außerdem befinden sich alle notwendigen Dateien ebenso auf dem Sensorknoten selbst wie etwaige Bibliotheken. Dadurch soll die Entwicklungszeiten von Sensor Netzwerken enorm verbessert werden, da die Entwickler nun nur eine Programmiersprache beherrschen müssen unabhängig von der eingesetzten Hardware. Es müssen nicht mühsam für jede Hardware die Entwicklungsumgebung aufgesetzt werden, sondern es wird vom Entwickler nur ein Webbrowser benötigt. Desweiteren wird Zeit gespart dadurch, dass der von einem Compiler generierte Binärcode auf die Hardware übertragen werden muss. Insgesamt bedeutet dieses System eine enorme Verbesserung des Entwicklungsprozesses von Sensor Netzwerken durch die Integration des Programmausführung und der Entwicklungsplattform auf einem Gerät wodurch eine zyklische Anpassung des Codes an die Hardware entfällt.

2.2.2 Rapid Prototyping

Gadgeteer

In dem von [VSH11] vorgestellten Artikel geht es um Prototyping mit Gadgeteer. Gadgeteer ist eine von Microsoft entwickelte Kombination von Hardware und Software. Es wird verwendet um schnell Prototypen zu bauen. Es verwendet das .NET Micro Framework von Microsoft und als Entwicklungsumgebung Visual Studio. Die Hardware wie auch die Software ist modularisiert, bzw. objektorientiert. Die Hardware besteht aus einem Mainboard und mehreren Modulen, welche Sensoren und Aktoren besitzen können. Diese Module können über einen standardisierten Port mit dem Mainboard verknüpft werden und über die Entwicklungsumgebung programmiert. Dabei geschieht der Zugriff auf die einzelnen Module über Objekte, welche die jeweiligen Funktionen zur Manipulation oder Sensorauswertung anbieten. Eine Neuheit dabei ist auch das man dabei einen Eventbasierten Ansatz verfolgt hat. So lässt sich z.B. bei einem Button Modul das Drücken und Loslassen als Events verarbeiten. Gadgeteer wurde dabei entwickelt um Rapid Prototyping zu betreiben oder Geräte mit nur sehr geringen Stückzahlen herzustellen. Dafür gibt es für jedes Modul auch CAD Daten, welche verwendet werden können um passende Gehäuse für die Geräte herzustellen.



Abbildung 2.1: Benutzeroberfläche von Scratch [RMMH⁺09]

2.2.3 visuelle Programmierung

Scratch

[RMMH⁺09] stellen in ihrem Artikel eine visuelle Programmiersprache für jederman vor. Diese Programmiersprache bzw. die Programmierumgebung wurde entwickelt um Menschen Programme entwickeln zu lassen, welche sich selbst nicht als Programmierer sehen. Es wurden schon viele verschiedene Programme in Scratch erstellt: Von Videospiele bis hin zu wissenschaftlichen Simulationen. Das Publikum welche Scratch verwendet ist sehr jung, aber auch Erwachsene verwenden Scratch. Der Hintergedanke von Scratch ist es für Menschen die zwar Digitale Medien konsumieren und auch damit interagieren, auch eigene Inhalte erstellen können. So kann jederman sein eigenes Spiel oder einfach eine Simulation erstellen. Die Abbildung 2.1 zeigt dabei die Benutzeroberfläche der Entwicklungsumgebung. Hier können einfache Anwendungen per Drag'n'Drop zusammengesetzt werden. So kann für einzelne Elemente bestimmte Events abgefragt werden, wie was passiert wenn auf ein Element geklickt wird oder was passiert bei einer Tastatureingabe. Dabei sind die einzelnen Elemente wie ein Puzzle aufgebaut, welche sich aber nur in einer sinnvollen Weise verbinden lassen. Zusätzlich sind die Code-Elemente über eine Farbe bestimmten semantischen Inhalten zugeordnet. Es ist eine einfach zu lernende Programmiersprache, welches es selbst Kinder ermöglicht eigene Programme zu entwickeln.

2.2.4 Unterschiede zur eigenen Lösung

Die größten Unterschiede zur eigenen Lösung stellen vor allem die Sensornetzwerke dar, es gibt bei allen die Möglichkeit die Sensordaten zu verwenden, aber es gibt keine Mechanismen wie man damit wiederum andere Geräte steuern kann. Den besten Ansatz liefert dynam mit seiner Entwicklungsumgebung auf den Geräten selbst, was in der eigenen Lösung für alle Geräte auf einem zentralisierten Server geschieht. Bei dem Web Meshup ist das Problem, dass es nicht möglich ist die Sensorinformationen vom Gerät verschickt werden, sondern es muss von der Anwendung abgefragt werden. Dadurch wird unter Umständen die Bandbreite bei zu häufigem pullen schnell zu klein. Als ein Beispiel für Rapid Prototyping wurde Gadgeteer vorgestellt, welches auf Grund seiner Flexibilität und seiner schnellen Entwickelbarkeit auch in der Implementierung des Konzeptes verwendet wurde. Bei der visuellen Programmierung vorgestellten Programmiersprache Scratch sollen vor allem die Einfachheit in die Zukunft der Plattform einfließen. Wenn selbst Kinder diese Programmiersprache verwenden können, welche vielleicht nicht unbedingt die Zielgruppe dieses System ist, dann ist es für viele Menschen welche mit Technik nicht viel am Hut haben auch zu bedienen. Für die visuelle Programmierung der Regeln dieses Systems würde sich aber eine erwachsenere Benutzeroberfläche anbieten, bei der man nicht unbedingt meint im Kindergarten zu sein.

3 Konzept

3.1 Einführung

In diesem Kapitel geht es um die Vorstellung des Konzepts zur Lösung der Problemstellung. Das Konzept ist dabei völlig losgelöst von jeglicher Software, bzw. Hardware. In diesem Teil sollen die Grundgedanken des Konzeptes vermittelt werden und wie die Plattform im Einsatz verwendet werden soll. Im ersten Teil wird das Konzept anhand von einem großen Szenario aus dem angestrebten Anwendungsgebiet des Museums illustriert. Anschließend wird das Konzept als Ganzes dargestellt und weitere Einzelheiten erläutert.

3.2 Szenario

In diesem Szenario geht es um Bob. Bob ist mitte dreißig und der Chef eines kleinen Museums in einer kleineren Stadt. Ihn hat schon immer gestört, dass die Ausstellungen in seinem Museum nur zum Ansehen sind und man nicht mit ihnen interagieren kann. Da er abends öfters im Internet unterwegs ist, um sich über die neuesten Exponate zu informieren, ist er zufällig auf eine Seite gestossen, welche eine Plattform anpreist. Diese Plattform soll es ermöglichen, ohne programmieren zu können, interaktive Umgebungen zu gestalten. Da der Druck für sein kleines Museum immer größer wird, weil immer weniger Menschen in dieses Museum kommen, da sie lieber zu Hause vor dem Computer sitzen und sich mit Sachen beschäftigen auf die sie durch Interaktion Einfluss nehmen können, entschliesst er sich eines der Starterpakete für interaktive Umgebungen zu bestellen. Ein paar Tage später trifft auch schon das Paket mit der Grundausstattung für die Interaktivität ein. Bob freut sich, dass alles in dem Paket enthalten ist und er nicht noch irgendwelche wichtigen Komponenten irgendwo anders bestellen muss. Er packt einen kleinen, handlichen Server aus, ein Tablet PC und eine Menge kleinerer Geräte in unterschiedlichen Formfaktoren und noch viele weitere Sensoren und Aktoren, welche mit dieser Plattform verwendbar sind. Da er aber der Chef ist und noch viele weitere wichtige Sachen zu erledigen hat, überträgt er die Aufgabe sein Museum interaktiv aufzurüsten an zwei seiner Mitarbeiter: Sandy und Andy.

Sandy und Andy sind die kreativen Köpfe des Museums, haben aber von Technik nicht viel Ahnung. Aber dennoch setzten die beiden sich an das Startpaket und bauen erste kleine Prototypen. Dazu schalten sie den Server ein, welcher sich auch gleich um die Infrastruktur für die Kommunikation kümmert. Anschließend nehmen sie das Tablet und können damit über eine Weboberfläche den Server und die Geräte konfigurieren. Sie schließen verschiedene Sensoren und Aktoren an unterschiedlichste Geräte an und konfigurieren diese auf dem

Server durch einfaches Drag'n'Drop von Sensoren, Aktoren für das jeweilige Gerät. Das funktioniert für die beiden so einfach, dass sie sich fragen für was man eigentlich Informatik studieren muss. Als die Geräte konfiguriert sind, fangen diese auch gleich an die Sensordaten an den Server zu schicken, wo diese leicht verständlich für die beiden angezeigt werden. Als nächster Schritt steht für die beiden an, sich Regeln zu überlegen, welche die Mechanismen zur Interaktivität darstellen. Die beiden überlegen sich nun was passieren soll wenn ein Sensor ausgelöst wird: Welche Aktoren sollen dann ausgelöst werden? Sie sind dabei überrascht wie einfach sich diese Regeln, auch sehr komplexe, erstellen lassen ohne programmieren zu können. Die beiden finden die Programmierung mittels graphischer Objekte oder den sehr einfachen Assistenten sehr schön. Da sie nun schon durch erste Tests gesehen haben, dass die neue Plattform einfach funktioniert, können die beiden sich daran machen, die ersten interaktiven Umgebungen zu realisieren.

Bevor die beiden aber das Büro verlassen um in die Ausstellungen zu gehen und diese interaktiv zu gestalten, schlägt Andy vor, dass man es doch auch im Büro benutzen könnte. Es nervt ihn einfach, wenn er mit jemanden im Büro sprechen will, aber der ist wieder einmal nicht da, da er sich einen Kaffee holt oder auf dem Klo ist. Aus diesem Grund beschließt Andy in jedem Büro ein kleines Display zu installieren, welches anzeigt, ob jemand in seinem Büro ist und wenn das Büro verlassen wurde, wann er das letzte mal da war. Sandy findet das eine tolle Idee und fängt an in ihrem Büro den ersten Bewegungsmelder zu installieren. Danach gehen sie durch alle Büros und installieren dort ebenso welche, inklusive je ein Display an der Wand. Die Geräte sind durch die gute Konfigurationsoberfläche schnell konfiguriert und auch die erforderlichen Regeln erstellen sich fast wie von selbst. Ab jetzt weiß jeder, ob die Person die er vielleicht sucht, in seinem Büro ist oder ob lieber am Kaffeeautomaten gesucht werden soll.

Als die beiden kreativen Köpfe des Museums dieses Problem gelöst haben, schlägt Sandy vor, das man den Besprechungsraum auch in dieses System integrieren kann. So kann jeder in seinem Büro sehen, ob der Besprechungsraum belegt ist oder ähnliches. Andy meint, dass man das auch anders lösen kann und nimmt eine kleine Box aus Pappe und baut eines der Geräte darin ein. Er schreibt auf verschiedene Seiten der Box: belegt, frei, reserviert und warten. Er konfiguriert das Gerät und ergänzt die Regeln so, dass nun wenn die Box so steht das z.B. belegt oben steht, es auf allen Displays angezeigt wird. Sandy will unbedingt wissen was es mit dem warten auf sich hat. Andy erklärt Sandy, das mit warten gemeint ist, dass der, welcher im Besprechungsraum eine Besprechung machen möchte, schon da ist und noch auf die Teilnehmer wartet. Andy grinst und erklärt Sandy, das wenn warten ausgewählt wird, in allen Büros das Display blinkt und eine Ton ausgegeben wird, um die Leute daran zu erinnern, das jetzt Besprechung ist und sie sich schnell auf den Weg machen sollen.

Jetzt machen die beiden sich auf in die Ausstellung, um ihre eigentliche Aufgabe zu lösen. Sie gehen durch die Ausstellung und machen sich Gedanken, was sie alles mit dem neuen System anstellen können. Andy findet die Armbänder ganz sinnvoll, wobei eines für jeden Besucher ist. Mit diesem Armband kann festgestellt werden, wer sich wann und wo befindet. Außerdem kann das Armband auf Bewegungen reagieren. Andy möchte dies mit einem Display an der Wand kombinieren, welches eigentlich immer nur statischen Inhalt anzeigt. Jetzt soll es für den Besucher möglich sein, durch herantreten an das Display, es zu steuern.

Dies soll mit Handbewegungen realisiert werden. Eine Handbewegung nach rechts und der nächste Inhalt wird angezeigt und eine nach links und der vorherige Inhalt kann angezeigt werden. So kann der Besucher selbst entscheiden was ihn interessiert und was nicht. Als sie alles installiert und getestet haben, fällt Sandy ein Bereich mit Sofas und einem Tisch in der Mitte auf. Sandy möchte das Besucher, die sich mit ihrem Armband auf ein Sofa setzen, erkannt werden und auf den Tisch dann Inhalte projiziert werden, welche zu den Bereichen passen, welche sie sich angesehen haben. Oder aber es sollen auf Grund der Positionsdaten Bereiche der Ausstellung dargestellt und hervorgehoben werden, welche der jeweilige Besucher noch nicht besucht hat. Nach kurzer Zeit haben sie auch diese Aufgabe gemeistert.

Sandy und Andy wundern sich wie viele Möglichkeiten diese Plattform bietet, bei trotzdem leichter Bedienung. Als die Sachen alle eingebaut sind, vergehen ein paar Wochen und es kommen immer mehr Besucher in das kleine Museum, da die neu entdeckte Interaktivität auch ein jüngeres Publikum anspricht. Darüber freut sich vor allem der Chef Bob und bestellt gleich neue Geräte, welche in seine Ausstellung integriert werden können. Sandy und Andy freuen sich, da sie in nächster Zeit noch viele Stunden mit dem Ausbau des Museums verbringen dürfen.

3.3 Zusammenfassung des Konzepts

Bei der durch dieses Konzept eingeführten Plattform geht es in erster Linie darum, einfach und ohne viel Aufwand interaktive Umgebungen zu gestalten, ohne sich Gedanken über die Programmierung machen zu müssen. Angelehnt an das Rapid Prototyping, soll es einfach und schnell zu ersten Ergebnissen führen. Prinzipiell soll das System als ganzes geliefert werden, so dass schon alles vorkonfiguriert ist und sowohl Server, die einzelnen Geräte und die gesamte Infrastruktur mitgeliefert werden. Dabei soll der Server den Datenverkehr der einzelnen Geräte verwalten, sowie auch gleichzeitig als zentraler Punkt der Infrastruktur dienen. Unter Infrastruktur wird verstanden, dass z.B. die Kommunikationsmittel unterstützt werden. Zum Beispiel bei WLAN als eine Kommunikationsart, fungiert der Server gleichzeitig als Router und dessen Reichweite kann durch Extender erweitert werden. Diese Extender sollen es für alle möglichen Arten von Kommunikationsarten geben: WLAN, xBee, Bluetooth, ... oder alles in einem Gerät.

Bei dem System besitzt jedes Gerät eine Firmware, welche es ihr ermöglicht den angeschlossenen Kommunikationskanal zu nutzen und sich beim Einschalten selbstständig am Server anzumelden und ihm dessen eigenen Fähigkeiten mitzuteilen. Fähigkeiten bedeutet in diesem Fall, dass das Gerät dem Server mitteilt um was für eine Art von Gerät es sich handelt, welche Kommunikationsart er verwendet (was wichtig ist für die Bandbreite) und welche Anschlussmöglichkeiten es für Sensoren und Aktoren besitzt. Es kann für jedes Gerät eine Firmware entwickelt werden um es in das System integrieren zu können. Ein Gerät dient dabei nur als Sensorquelle und/oder Aktuatorquelle. Das bedeutet, dass sich das Gerät nur um die Eingaben bzw. Ausgabe kümmern muss, da die meiste Logik auf dem Server abläuft. Dies hat den Vorteil, dass die Logik auf dem Server zentral verwaltet wird und

sich Änderungen sofort auswirken. Es gibt jedoch die Möglichkeit ein Teil der Logik auf die Geräte zu verlagern um so den Netzwerkverkehr zu reduzieren. So soll es möglich sein zu beeinflussen, wann die Sensordaten an den Server geschickt werden. Dabei soll es drei Möglichkeiten geben: 1. Die Sensorwerte sollen immer zu einem festen Intervall an den Server geschickt werden. Als Beispiel könnte ein Lichtsensor seine Werte immer alle 100 Millisekunden schicken. 2. Die Sensordaten werden nur an den Server verschickt, wenn sich der aktuelle Wert zu dem letzten Wert geändert hat. Bei einem Button der gedrückt wird, würden so nur 2 Werte geschickt werden: einmal beim Drücken und einmal beim Loslassen. 3. Es soll möglich sein bestimmte Wertebereiche anzugeben. Nur in diesem Bereich werden die Daten an den Server geschickt. Das könnte z.B. bei einem Entfernungssensor zum Einsatz kommen, in dem mich nur bestimmte Bereiche der Entfernung interessieren. Als letzter Punkt die Logik in die Geräte zu verlagern, soll es möglich sein verschiedene Filter für die Sensor Eingänge anzugeben. So wäre z.B. ein einfacher Mittelwertfilter oder auch ein Kalman-Filter denkbar. Alle diese Einstellungsmöglichkeiten stehen dann in der Konfigurationsdatei für die einzelnen Geräte und werden beim Start eines Gerätes vom Server an das jeweilige Gerät geschickt. Das Gerät kümmert sich dann um die Umsetzung dessen.

Der Server ist dazu da alle Kommunikationskanäle zu bündeln und die Kommunikation unter den Geräten zu ermöglichen. So ist es möglich, dass ein Gerät welches z.B. über Wifi angeschlossen ist, einen Aktor auf einem Gerät, welches mit xBee angeschlossen ist, zu aktivieren. Das bedeutet, dass es dadurch möglich ist ein komplett heterogenes Netzwerk an Geräten aufzubauen. Auf dem Server werden zudem zentralisiert die Konfigurationen der einzelnen Geräte erstellt und gespeichert. Die Konfiguration der einzelnen Geräte geschieht dabei durch einen Editor, welcher das jeweilige Gerät und die verfügbaren Ports darstellt, außerdem gibt es noch eine Liste mit Sensoren und Aktoren, welche durch Drag'n'Drop mit den einzelnen Ports verbunden werden können. Es können zusätzliche Einstellungen für die jeweiligen Module (Sensoren und Aktoren) gemacht werden. Darunter fällt z.B. die schon vorher erwähnten Filter oder die wie die Sensor Daten verschickt werden (wie vorher: festes Intervall, Event-basiert oder über Wertebereich). Außerdem kann noch ein Name für das Gerät und für einzelne Module zur besseren Übersichtlichkeit vergeben werden. Durch diese erstellte Konfiguration enthält der Server für jedes real vorhandene Gerät eine Repräsentation des Gerätes.

Um mit diesem System auch die interaktive Komponente abzubilden, ist es möglich auf dem Server sogenannte Regeln zu erstellen. Durch diese Regeln wird festgelegt was passiert, wenn bestimmte Sensorwerte von unterschiedlichsten Geräten vorliegen und was dann ausgelöst werden soll. Diese Regeln können von einfachen Wenn-dann-Regeln bis zu sehr komplexen Regeln mit mehreren Wenss und vielen abzufragenden Sensorquellen reichen. Außerdem kann in jeder Regel eine beliebige Anzahl an Aktoren geschaltet werden. Zusätzlich dazu werden die Regeln in einer Prioritätenliste eingeteilt, da manche Regeln wichtiger sind als andere und so keine Komplikationen auftreten wenn mehrere Regeln den gleichen Aktor schalten wollen, aber mit unterschiedlichen Parametern. Die Programmierung solcher Regeln soll über eine einfach zu bedienende, graphische Programmierung funktionieren.

Die Idee ist jede Menge unterschiedlichster Geräte zu unterstützen und es so ermöglichen, dass jeder die Hardware die er zu Hause hat auch für dieses System verwenden kann. Außerdem benötigt man für individuelle Lösungen auch zum Teil spezielle Geräte, z.B. wenn man ein tragbares Gerät benötigt weil der Sensor am Körper getragen werden soll. Aus diesem Grund soll dieses System Open Source gemacht werden, um bei gefallen seitens der Community, durch Mitarbeit immer mehr und mehr Geräte für dieses System verfügbar zu machen.

3.3.1 Server

Wie schon in dem einführenden Text erwähnt, gibt es einen zentralen Server, welcher alle Kommunikationsarten bündelt und für die Kommunikation zwischen den Geräten verantwortlich ist. Auf dem Server sind die jeweiligen Konfigurationen für die einzelnen Geräte, welche an dem System angemeldet sind, hinterlegt. Diese Konfiguration beinhaltet einige wichtige Informationen wie die Belegung der Ports an dem Gerät durch Sensoren und Aktoren, welche Kommunikationsart von dem Gerät verwendet wird und noch weitere Daten wie eine Benennung der einzelnen Geräte und der angeschlossenen Module. Diese Benennung muss dabei eindeutig sein, da man dadurch eine spätere Erstellung der Regeln vereinfachen kann. Auch diese Regeln werden zentralisiert von dem Server verwaltet. Diese Zentralisierung bedeutet für das System, dass es hinsichtlich der Skalierbarkeit, in diesem Falle die Erweiterung um sehr viele Geräte, ein Problem darstellen kann. Es gibt durch dieses zentralisierte System aber auch den Vorteil, dass die erstellten Regeln sofort mit Erstellung wirksam werden und Geräte mit unterschiedlicher Kommunikationsart miteinander kommunizieren können. Da es für einige Kommunikationsarten notwendig ist den einzelnen Geräten mitzuteilen unter welcher Adresse der Server verfügbar ist, ist es, abhängig von der jeweiligen Kommunikationsart, notwendig diese über eine Art Broadcasting den Geräten mitzuteilen. Durch den Server wird zusätzlich noch die Weboberfläche bereitgestellt um die Konfigurationen und Einstellungen des Systems vornehmen zu können.

3.3.2 Geräte

Die Geräte fungieren in diesem System grundsätzlich nur als Sensorquellen oder Aktorsensoren. Das bedeutet, dass die Geräte nur für die Eingabe bzw. Ausgabe von Daten zuständig sind. Man kann dabei durch die Konfiguration des Gerätes einzelne Bereiche beeinflussen. So kann man Einstellen, wann Sensordaten verschickt werden. Als Optionen dafür stehen verschiedenen Möglichkeiten zur Verfügung: Zum einen kann man Sensordaten verschicken zu einem bestimmten Intervall, also z.B. alle 100 Millisekunden oder aber Event-basiert. Event-basiert bedeutet in diesem Zusammenhang, dass sich die Sensorwerte verändern müssen, damit der Wert an den Server verschickt wird. Bei dem Beispiel eines Buttons als Sensorquelle würde bei dem Intervall in jedem angegebenen Zeitraum der Wert versendet, also auch mehrmals der gleiche Wert obwohl dieser sich nicht verändert. Dies hat den Nachteil, dass das Netzwerk über das die Daten versendet werden schnell überfüllt sein kann. In dem zweiten Fall würde hingegen nur einmal beim Drücken des Buttons und

beim Loslassen ein Wert an den Server übermittelt. Dadurch ist es möglich den Netzwerkverkehr zu begrenzen. Dies ist, wenn es durch das angeschlossene Modul bereitgestellt wird, die Standardeinstellung für die Art wann Sensordaten übermittelt werden. Als dritte Möglichkeit gibt es noch, dass man für einen Sensorwert bestimmte Bereiche angeben kann, in dem einen Werte interessieren. Dadurch kann man bestimmte Teile der Logik in das Gerät verlagern um z.B. durch einen Lagesensor nur benachrichtigt zu werden, wenn eine bestimmte Position erreicht wird. Da es bei analogem Input zu starken Schwankungen kommen kann, sollte es für erfahreneren Benutzer möglich sein Filter mit der Konfiguration mitzuschicken um diese Werte zu glätten. Dabei können verschiedene Filter auf den Geräten implementiert sein, welche dann durch die Konfiguration für bestimmte Module aktiviert werden. Damit man auf dem Server die Geräte nicht einzeln von Hand eingeben muss, was bei sehr vielen Geräten sehr mühsam ist, sendet das Gerät beim Starten seine Fähigkeiten an den Server. Dadurch weiß der Server dann schon um was für ein Gerät es sich handelt und welche Module angeschlossen werden können. Damit die Konfiguration die auf dem Server hinterlegt ist auf das Gerät kommt gibt es drei verschiedene Möglichkeiten. Die Konfiguration wird vom Server angefragt, wenn das Gerät gestartet bzw. eingeschaltet wird, wenn der Reset Button gedrückt wird am Gerät oder vom Benutzer über die Weboberfläche angestoßen. Die letzte Möglichkeit bringt den Vorteil, das der Benutzer des Systems nach dem Ändern der Konfiguration, welche mit keiner Änderung der Module zu tun hat, also im Falle das zusätzlich ein Filter verwendet werden soll oder ähnliches, der Benutzer nicht erst zu dem Gerät hingehen muss und es resetten, sondern bequem vom Computer aus die neue Konfiguration auf das Gerät schicken kann.

3.3.3 Regeln

Ein wichtiger Punkt des Systems ist die Erstellung von Regeln. Diese Regeln dienen dazu darzustellen, welche Aktionen ausgeführt werden sollen, wenn Sensordaten von einem Gerät aus den Server erreichen. Das Regelsystem ist als Programmiersprache aufgebaut damit man die Regeln komplex gestalten kann, wenn dies benötigt wird. Dabei besteht eine Regel aus einem Codeblock welcher auf Sensoren und Aktoren durch Variablen, bzw. Objekte zugreifen kann. Da nicht jeder Programmieren kann, gibt es unterschiedlichen Methoden diese zu erstellen, aber am Ende wird dies immer in Programmcode umgesetzt. Als einfachste Methode für die Erstellung von nicht zu komplexen Regeln gibt es einen Assistenten wo man für ein Gerät und dessen Sensorquelle einen Wertebereich oder einen einfachen Abgleich von Werten mit ist gleich, kleiner als, größer als und so weiter festlegen kann. Weiterhin kann man dann dort spezifizieren welche Geräte ihre Aktoren auslösen sollen und mit welchen Parametern. Das bedeutet, dass man als Input genau ein Sensor eines Gerätes überprüft wird und dann aber eine beliebige Anzahl an Aktoren ausgelöst werden können. Dies geschieht über einfaches Hinzufügen von Aktoren und deren Parametern wie z.B. das Licht an oder ein Text, Bild, Video welches auf einem Display angezeigt werden soll. Die zweite Möglichkeit ist für Programmierer interessant: Es wird für jede Regel ein Codeblock geschrieben in dem man auf die Geräte mittels Variablen zugreifen kann. Durch diese Variablen erhält man auch Zugriff auf ältere Sensorwerte, Durchschnittswerte, Minimum, Maximum und so weiter. Da ein Gerät einen eindeutigen Namen besitzt und auch jedes Modul einen eindeutigen Namen

besitzt, kann man so einfach mittels `Gerätenamen.Modulname.value()` auf den aktuellen Wert zugreifen. Wenn man aber das Minimum will kann man das `value()` durch `min()` ersetzen und bei den anderen Funktionen funktioniert das analog dazu. Bei den Aktoren funktioniert das im Prinzip auch gleich: durch `Gerätename.Modulname.LED.On()` kann z.B. eine LED eingeschalten werden und mit `Off()` wieder ausgeschalten werden. Die Funktionen die aufgerufen werden, können durch weitere Parameter weiter beeinflusst werden. So kann man mit `min(10)` das Minimum der letzten zehn Werte benutzen oder mit `LED.On(3000)` die LED für 3000 Millisenkunden einschalten. Dabei wird die Programmierung durch eine Entwicklungsumgebung im Browser unterstützt. So soll wie bei Visual Studio angezeigt werden wenn man eine Variable mit einem Punkt abschließt, was für Optionen man nun hat zum Programmieren. Also beispielsweise wenn man `Gerätename.` eingibt sollen dann die Module vorgeschlagen werden, welche das Gerät besitzt und wenn man sich für ein Modul entschieden hat mit `Gerätenamen.Modulname.` sollen die weiteren Optionen aufgelistet werden bis man am Ende angelangt ist. Weiter soll die Programmiersprache um häufige Funktionen erweitert sein bzw. erweitert werden können. So sollten beispielsweise mathematische oder Datums- Funktionen bereits vorhanden sein. Zusätzlich soll es dem Benutzer möglich sein solche Funktionen selbst zu ergänzen, damit man diese Funktionen in weiteren Regeln verwenden kann. Die letzte Möglichkeit der Regelerstellung baut auf der Programmiersprache auf und erlaubt es über eine graphische Programmiersprache die gleiche Komplexität des Programmcodes zu erzeugen wie mit der normalen Programmiersprache. Als letzte Erweiterung der Regeln soll es auch noch möglich sein virtuelle Sensoren zu verwenden. Diese Sensoren können wie eine Regel erstellt werden, am Ende kommt aber ein Ergebnis heraus, welches von anderen Regeln verwendet werden kann. Dieser virtuelle Sensor wird für die Sensorfusion verwendet. So könnte zum Beispiel der virtuelle Sensor anzeigen ob sich eine Person in einem Raum aufhält oder nicht. Dazu werden aber mehrere Sensoren verwendet um dieses Ergebnis zu berechnen. Da so eine Abfrage sehr komplex sein kann, ist es hiermit möglich das Ergebnis einfach in einer anderen Regel einfach als Sensor zu verwenden.

3.3.4 Weboberfläche

Die Weboberfläche dient dazu dem Benutzer die unterschiedlichen Möglichkeiten des Server zur Verfügung zu stellen. Hier kann der Benutzer die Geräte und Regeln verwalten (erstellen, bearbeiten und löschen). Weiterhin ist es möglich sich hereinkommende Sensordaten anzeigen zu lassen. Außerdem arbeitet das System mit PopUps um dem Benutzer der Weboberfläche anzuzeigen ob sich ein neues Gerät am System angemeldet hat oder ob ein bereits vorhandenes Gerät eingeschalten wurde und seine Konfiguration am Server abfragt. Dadurch erhält der Benutzer einen besseren Überblick über die Aktivitäten des Systems. Die einzelnen Bereiche für Geräte, Regeln und den Sensordaten Monitor werden nachfolgend weiter vertieft.

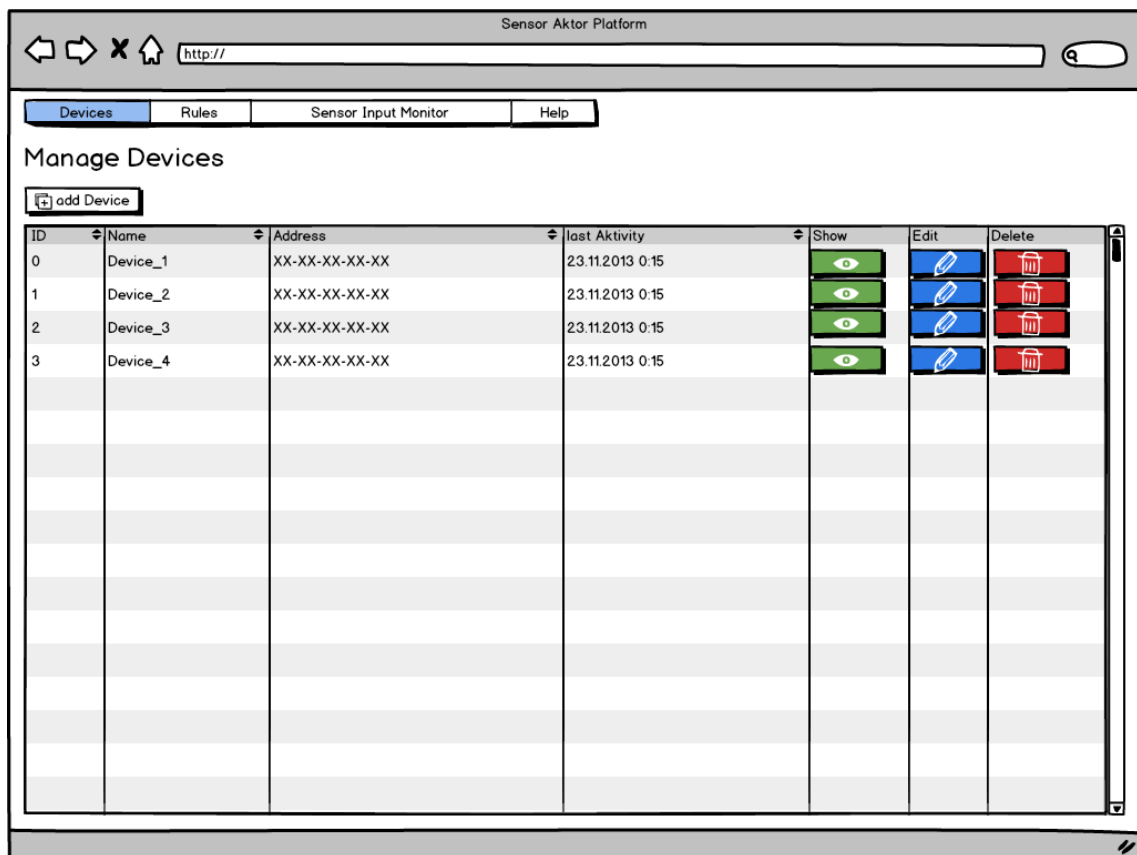


Abbildung 3.1: In dieser Seite werden die Geräte für die Plattform verwaltet. Man kann vorhandene Geräte ansehen, bearbeiten oder löschen. Zusätzlich können neue Geräte manuell hinzugefügt werden, was jedoch normalerweise nicht notwendig sein sollte, da sich Geräte beim Start automatisch am Server anmelden.

Geräte

Abbildung 3.1 zeigt die Übersicht für die Geräteverwaltung. Oben am Bildschirm sind die Links zu den verschiedenen Bereichen wie Geräte, Regeln und der Sensordaten Monitor. Darunter befindet sich ein Button um ein neues Gerät manuell hinzuzufügen. Dies ist im normalfall nicht notwendig, da sich ein eingeschaltetes Gerät automatisch am Server anmeldet. Durch klicken auf den Button kommt man auf einen Bildschirm, welcher der Abbildung 3.2 entspricht. Darunter befindet sich eine Tabelle mit allen auf dem Server vorhandenen Geräten. Ein Eintrag entspricht dabei einem Gerät und es wird der Gerätenname, die Adresse des Gerätes und wann die letzte Aktivität mit dem Server war, dargestellt. Außerdem kann man durch die drei Buttons am Ende des Eintrages sich die Konfiguration ansehen (show), bearbeiten (edit) und löschen (delete).

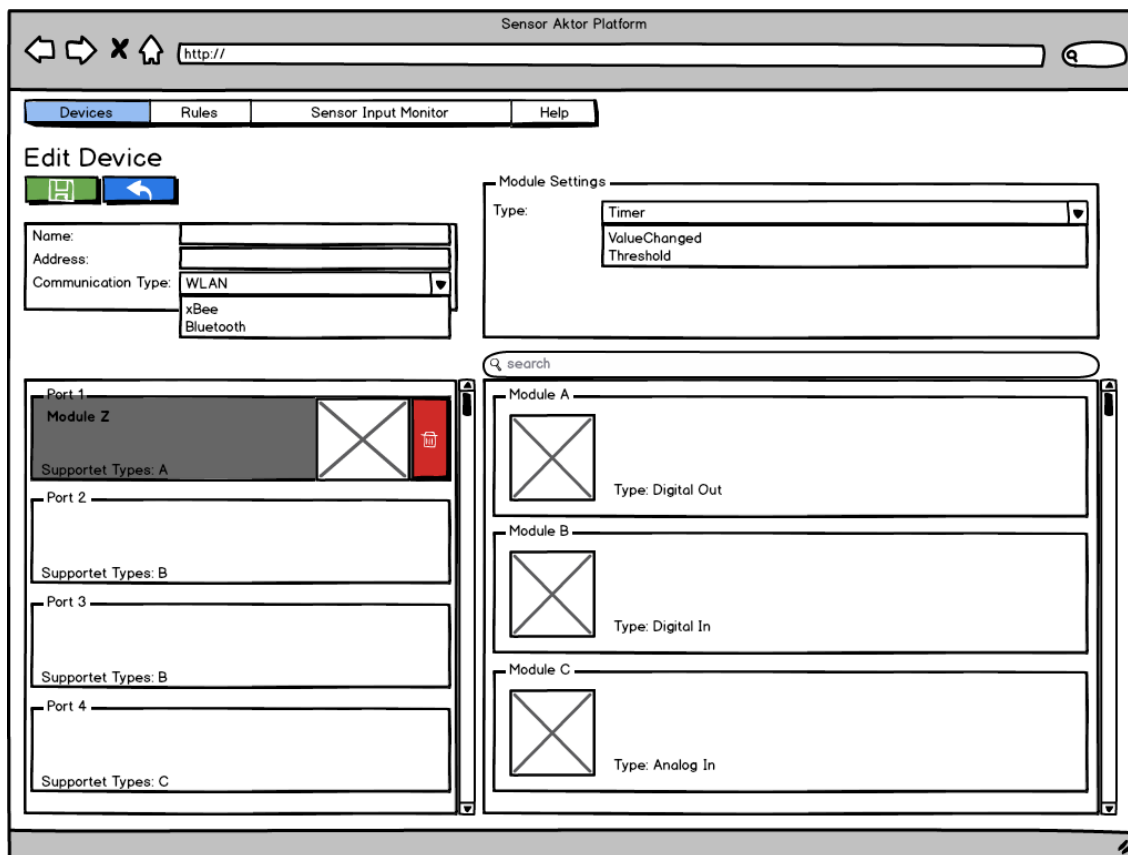


Abbildung 3.2: Bei dieser Seite handelt es sich um die Oberfläche zur Konfiguration eines Gerätes. Man kann hier Grundeinstellungen für jedes Gerät wie z.B. den Namen, die Adresse oder die Kommunikationsart einstellen. Zusätzlich lässt sich hier eine Zuweisung von Module (also Sensoren, Aktoren) zu den Ports der Geräte herstellen. Zusätzlich kann man die zugeteilten Module noch weiter konfigurieren.

In Abbildung 3.2 können die Konfigurationen für die Geräte erstellt bzw. geändert werden. Der Bildschirm ist in mehrere Bereiche aufgeteilt: Oben links kann man die Konfiguration speichern, bzw. ohne etwas zu verändern zur Übersicht der Geräte zurückkehren. Darunter befinden sich die allgemeinen Einstellungen des Gerätes wie der Name des Gerätes, die Adresse und die Kommunikationsart. Bei manueller Einstellung müssen diese Daten selbst eingegeben werden, wenn sich jedoch ein Gerät selbstständig am Server anmeldet, dann sind diese Daten schon vorhanden. Unter den allgemeinen Einstellungen sind die Ports des Gerätes dargestellt. Zusätzliche Informationen, wie z.B. welche Module angeschlossen werden können, werden angezeigt. Auf der Rechten Seite oben ist der Bereich für die Einstellungen der einzelnen Module vorhanden. Diese werden nur angezeigt, wenn ein Modul einem Port zugewiesen ist und man diesen Port markiert. Hier können Modulspezifische Einstellungen gemacht werden wie den Namen des Modules oder wann Daten von dem Modul versendet werden, wenn es sich um einen Sensor handelt. Direkt darunter

befindet sich eine Liste mit Modulen, welche mit dem aktuellen Gerät kompatibel sind. Diese Module können zusätzlich durch eine Suche gefunden werden. Man kann nun einfach per Drag'n'Drop die einzelnen Module auf die Ports ziehen. Wenn es ein Port ist, welcher nicht unterstützt wird, wird der aktuelle Port rot hinterlegt, ansonsten grün. Wenn ein Modul hinzugefügt wird, zeigt der Port eine Miniansicht des Modules und den Namen an. Man kann das Modul wieder durch den Löschen Button am rechten Rand entfernen. Zusätzlich kann man durch doppelklicken auf ein Modul aus der Liste der verfügbaren Module, dieses mit einem Auto Layouter mit einem passenden Port verbinden. Das hat den Vorteil, dass man nicht erst einen passenden Port suchen muss, ist jedoch wenig hilfreich wenn ein Gerät schon mit Modulen bestückt wurde und diese dann nicht übereinstimmen.

Regeln

Die Verwaltung der Regeln findet in Abbildung 3.3 statt. Oben sind drei Buttons für das erstellen neuer Regeln angebracht. Hier kann man entscheiden auf welche Weise neue Regeln erstellt werden sollen. Es gibt die Möglichkeit für einen Assistenten, welcher in Abbildung 3.4 dargestellt wird, eine reine Programmierung (Abb. 3.5) oder eine Programmierung mittels graphischen Elementen (Abb. 3.6). Darunter befindet sich eine Tabelle mit den einzelnen Regeln. Ganz vorne jeder Regeln ist eine Checkbox um die Regel zu aktivieren oder eben nicht. Jede Regel hat einen Namen um darzustellen was die Regel macht. So kann die Regel aussagen für was sie erstellt wurde, wie z.B. Displaysteuerung_Anwesenheit_Im_Buero. Dadurch ist die Verwaltung der Regeln deutlich einfacher. Die Regeln besitzen Prioritäten um zu vermeiden, dass zwei Regeln nicht einen Aktor mit unterschiedlichen Parametern ansteuern können und es damit zu inkonsistenten kommt. Hier ist die oberste Regel diejenige mit der höchsten Priorität. Neue Regeln werden am Ende eingefügt, können aber per Drag'n'Drop einfach an die gewünschte Stelle geschoben werden und somit kann die Priorität verändert werden.

Abbildung 3.4 zeigt den Assistenten für Benutzer welche nicht programmieren können. Das Prinzip dahinter ist eine einfache wenn dann Regelung. Ganz oben kann man die Regel entweder speichern oder verwerfen. Darunter kommt ein wenn (if). Hier kann man Sensoren hinzufügen und auf bestimmte Werte mit ist gleich, kleiner oder größer als prüfen. Wenn mehrere Sensoren hinzugefügt werden, müssen diese in Beziehung zueinander gebracht werden. Dies geschieht mit einem AND oder OR Operator. Dadurch kann man Regeln erstellen, welche komplexer sind als wenn nur ein Sensorwert ausgewertet wird. Gefolgt wird das ganze von einem dann (then). Hier lassen sich Aktoren hinzufügen und mit bestimmten Parametern ansteuern. Dies reicht von einer einfachen LED, welche ein und ausgeschaltet oder zum blinken in einer bestimmten Frequenz gebracht wird, bis zur Ansteuerung von Servos oder Ausgabe von Text oder Multimedia-Inhalten auf einem Display. Wenn die Regel gespeichert wird, ist sie sofort aktiv.

Die Programmierung in Abbildung 3.5 wird mit einem Editor realisiert, welcher Code Completion unterstützt. So kann während des Schreibens der Regel der Programmierer bei seiner Arbeit effizient unterstützt werden. Der Zugriff auf die Aktoren und Sensoren geschieht dabei über Objekte und deren Funktionen. So lassen sich Sensorwerte auslesen

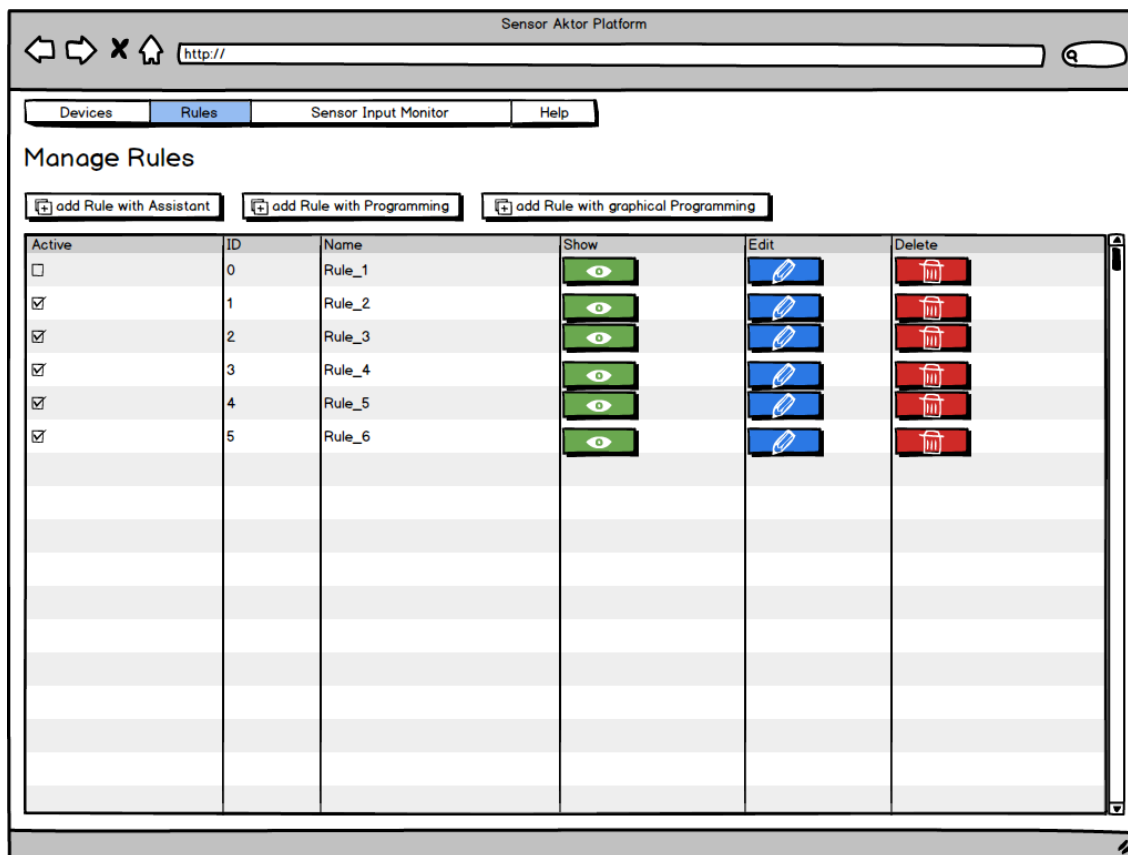


Abbildung 3.3: Regeln werden in dieser Seite verwaltet. Man kann vorhandene Regeln ansehen, bearbeiten oder löschen. Wenn man eine neue Regel erstellen will, kann man zwischen drei Methoden auswählen: der Erstellung mittels Assistent, Programmierung oder graphischer Programmierung. Jede vorhandene Regel hat am Anfang eine Checkbox, welche regelt ob die Regel aktiv ist. Außerdem besitzen die Regeln Prioritäten. Die oberste Regel hat dabei die höchste Priorität und dann immer weniger. Ändern der Prioritäten lässt sich durch verschieben der einzelnen Regeln an die gewünschte Stelle realisieren.

oder Aktoren steuern. Es ist so auch möglich Sensorwerte als Parameter für einen Aktor zu verwenden und z.B. den Sensorwert auf einem Display darstellen zu lassen.

Bei der graphischen Programmierung gibt es einige Vorteile gegenüber dem normalen Programmieren. Es ist vorallem für nicht Programmierer einfacher, da graphische Elemente leichter verstanden werden als reiner Text. Durch die Kombination der graphischen Elemente mit Text ist es möglich den Text jeweils für verschiedene Sprachen verfügbar zu machen. Außerdem gibt es beim Erstellen über die graphische Oberfläche keine Syntaxfehler, da nicht getippt wird, sondern die Elemente einfach per Drag'n'Drop hereingezogen werden oder über ein Kontext-Menü hinzugefügt werden. In Abbildung 3.6 ist dafür ein Beispiel gegeben. Hier können Elemente wie Geräte, Sensoren, Aktoren, Kontrollstrukturen und

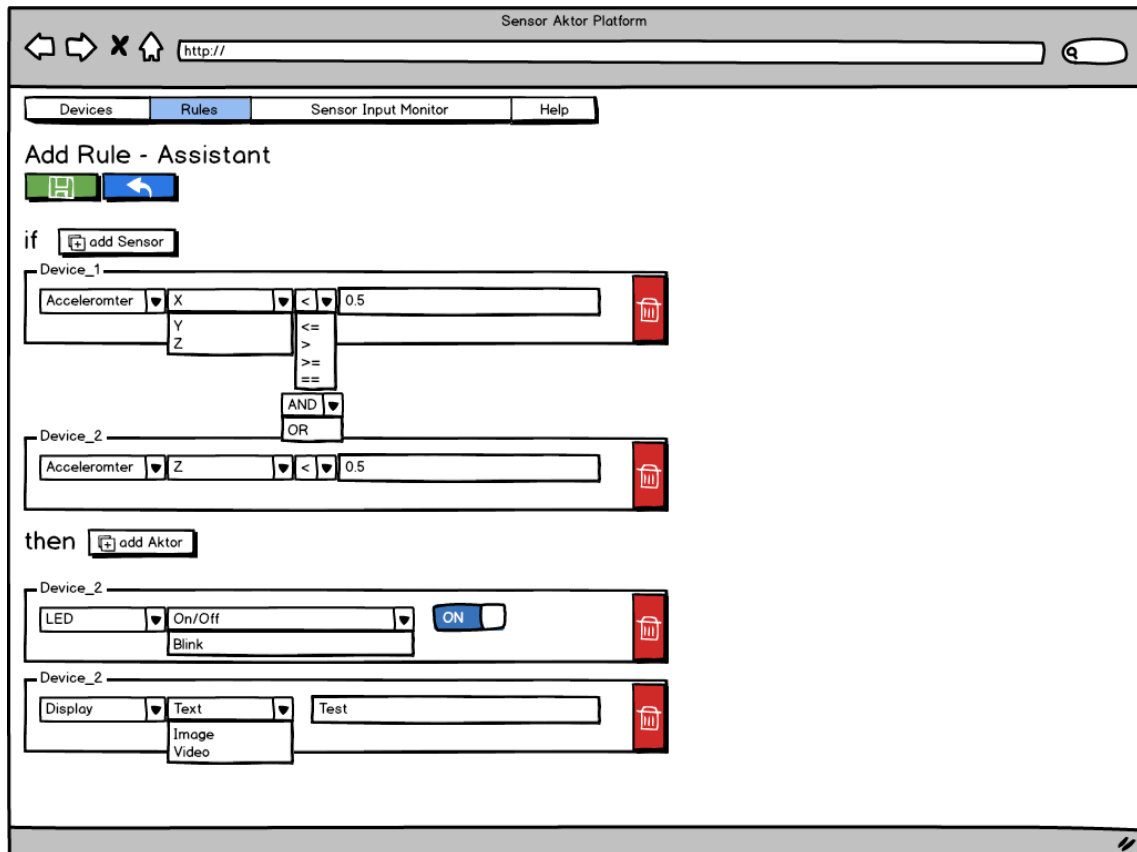


Abbildung 3.4: Diese Seite wird verwendet um mittels eines Assistenten neue Regeln anzulegen. Man kann am Anfang eine Reihe von Sensoren hinzufügen und jeden Sensor mittels einfachem ist gleich, kleiner gleich, usw. mit einem festen Wert verglichen. Zusätzlich können die Sensoren durch und bzw. oder miteinander verknüpft werden. Anschließend lassen sich Aktoren hinzufügen, welche mit bestimmten Parametern aufgerufen werden, wenn die Sensorwerte stimmen.

Funktionen über ein Kontext-Menü einfach hinzugefügt werden und dann miteinander verknüpft. Das soll es auch Nicht-Programmierern ermöglichen komplexere Regeln für das System zu erstellen.

Sensordaten Monitor

Auf der in Abbildung 3.7 angezeigten Seite werden ankommende Sensorwerte als Liste dargestellt. Damit das ganze Übersichtlich bleibt, kann man die Anzeige durch hinzufügen von Filtern einschränken. So lässt sich z.B. nach bestimmten Geräten Filtern oder nach bestimmten Modulen. Wenn man nur ein bestimmtes Modul eines Gerätes angezeigt bekommen möchte, dann kann man zwei Filter einstellen: eines für das Gerät und eines für die

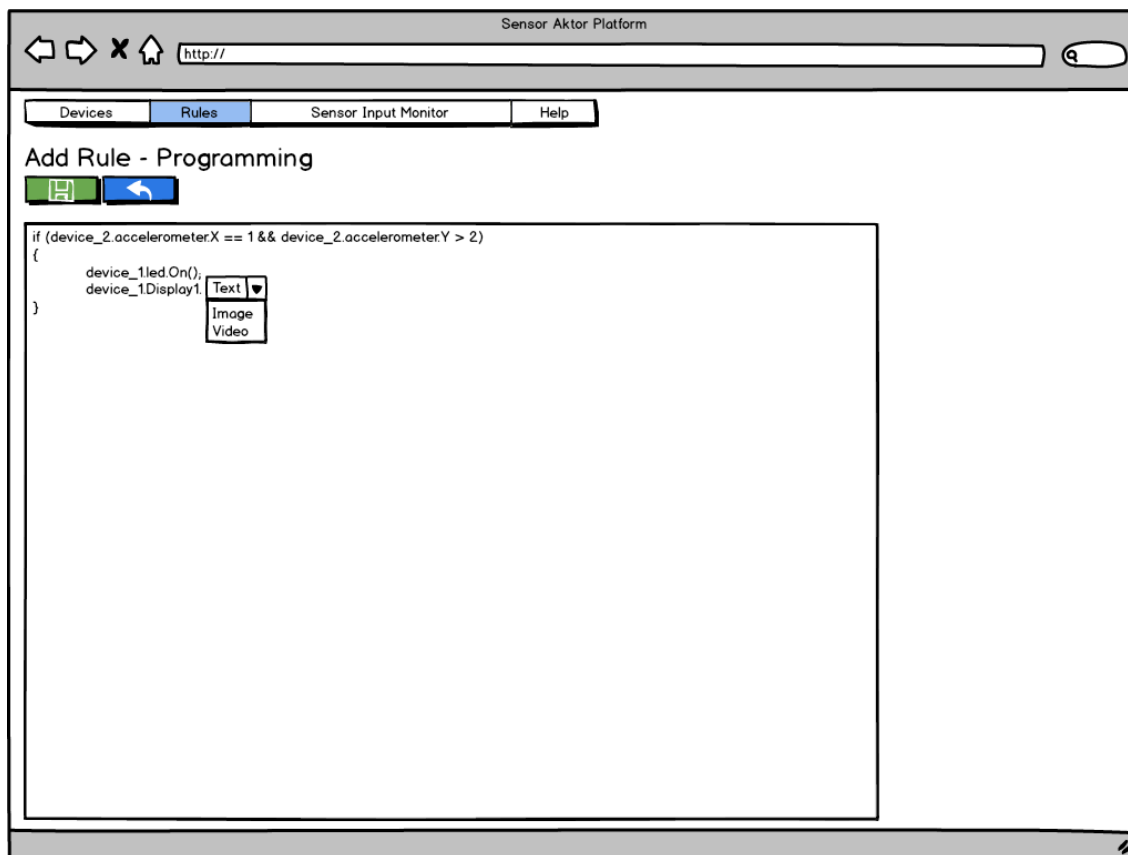


Abbildung 3.5: Es lassen sich Regeln durch einfache Programmierung erstellen. Die Programmierumgebung unterstützt dabei die Programmierung durch Code Completion. Der Zugriff auf die Aktoren und Sensoren geschieht dabei über Objekte und deren Funktionen. So lassen sich Sensorwerte auslesen oder Aktoren steuern.

Modulart. Damit wird nur ein bestimmtes Modul eines bestimmten Gerätes angezeigt. Da aber die Daten rasant zunehmen können, kann mit einem weiteren Filter geregelt werden, wieviele Werte jeweils auf der Seite angezeigt werden. Die Aktualisierung der Seite passiert dabei automatisch, so dass der Benutzer sich auf die Datenauswertung konzentrieren kann.

3.4 Vorteile

Die Vorteile des vorgestellten Konzeptes liegt in seiner Zentralisierung. Durch die zentralisierte Konfiguration der Geräte, muss ein Benutzer nicht zu einem Gerät hinlaufen wenn er nur kleine Änderungen macht, welches nichts mit der Änderung der realen Module zu tun hat. Die zentralisierte Programmierung der Regeln ermöglicht es für verschiedenste Geräte, welche eigentlich alle eine eigene Programmiersprache, Entwicklungsumgebung,

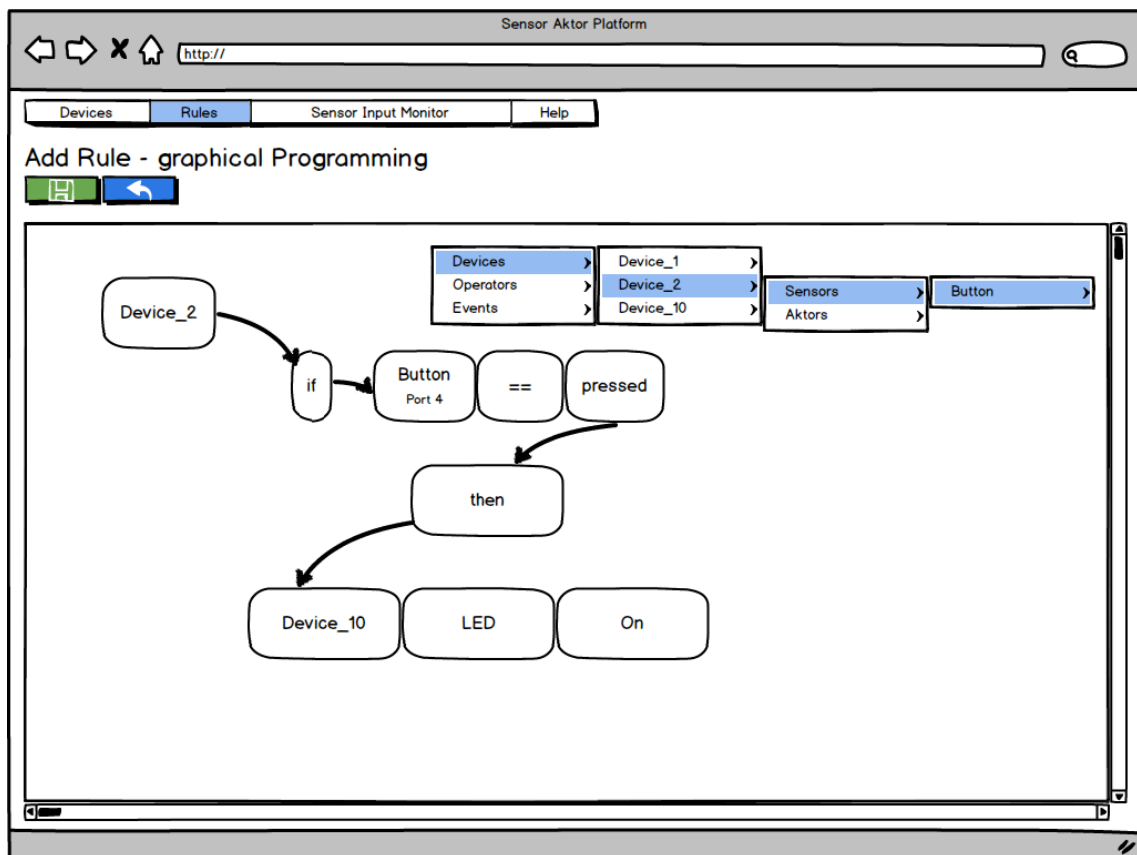
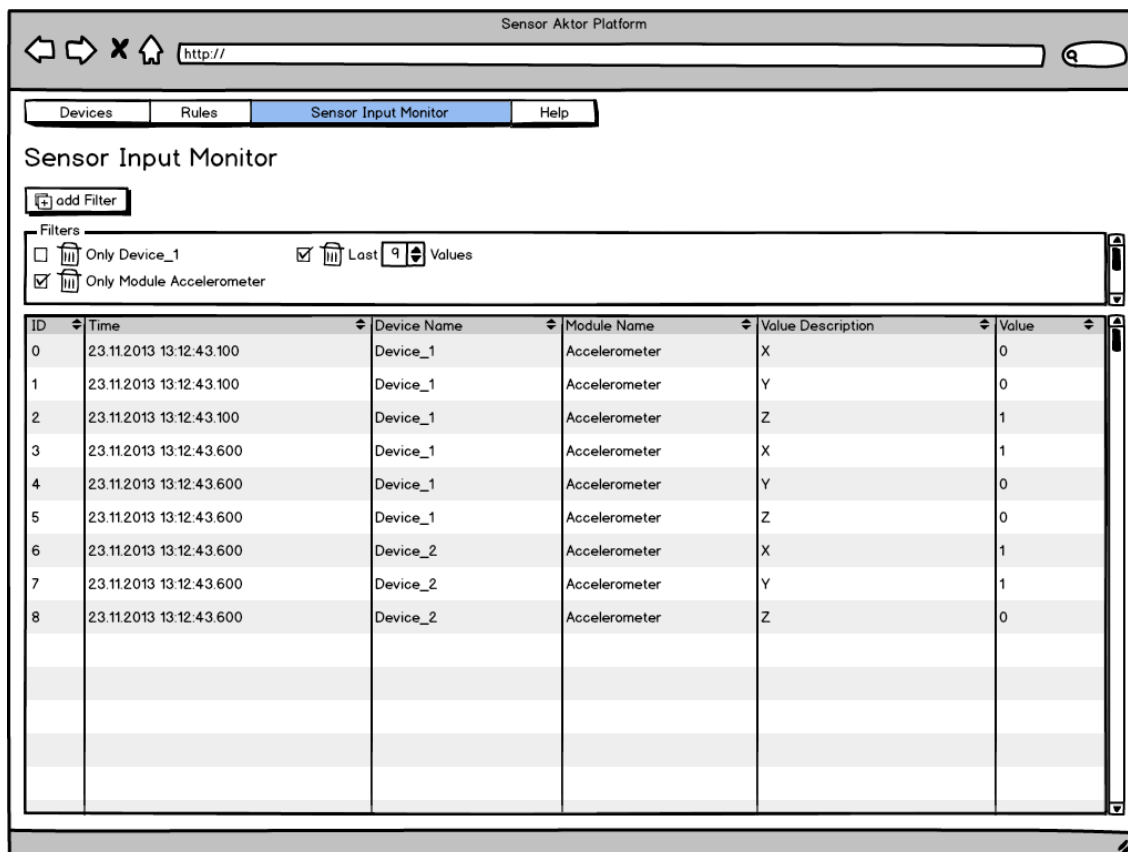


Abbildung 3.6: Diese Abbildung stellt die Seite dar, welche dazu benutzt wird durch graphische Programmierung Regeln zu erstellen. Man kann einzelne Elemente durch ein Kontext-Menü hinzufügen wie verschiedene Geräte und deren Sensoren bzw. Aktoren oder Kontrollstrukturen wie ein If oder weitere Funktionen wie Wait oder Repeat. Diese visuelle Ergänzung macht es auch nicht Programmierern einfacher komplexere Regeln als mit dem Assistenten zu erstellen.

Compiler, usw. besitzen, nur eine Entwicklungsumgebung und eine Programmiersprache zu verwenden. Das System lässt es dabei zu, dass das System auch von Benutzern verwendet werden kann, welche nicht Programmieren können. Dies wird durch den einfachen Assistenten oder die graphische Programmierung ermöglicht. Außerdem ist dieses System sehr einfach mit weiteren Gerätetypen erweiterbar. Dafür müssen die jeweiligen Gerätetypen nur mit einem Programm ausgestattet werden, welche es den jeweiligen Geräten ermöglicht mit dem Server zu kommunizieren. Die weitere Konfiguration der Module wird dann wieder über den Server abgewickelt.



ID	Time	Device Name	Module Name	Value Description	Value
0	23.11.2013 13:12:43.100	Device_1	Accelerometer	X	0
1	23.11.2013 13:12:43.100	Device_1	Accelerometer	Y	0
2	23.11.2013 13:12:43.100	Device_1	Accelerometer	Z	1
3	23.11.2013 13:12:43.600	Device_1	Accelerometer	X	1
4	23.11.2013 13:12:43.600	Device_1	Accelerometer	Y	0
5	23.11.2013 13:12:43.600	Device_1	Accelerometer	Z	0
6	23.11.2013 13:12:43.600	Device_2	Accelerometer	X	1
7	23.11.2013 13:12:43.600	Device_2	Accelerometer	Y	1
8	23.11.2013 13:12:43.600	Device_2	Accelerometer	Z	0

Abbildung 3.7: Der Sensordaten Monitor ist dafür da, bestimmte am Server ankommende Sensorwerte anzuzeigen. Dabei hat man die Möglichkeit durch hinzufügen von Filtern die Datenflut zu begrenzen und nur für einen selbst interessante Werte anzuzeigen. Dies kann einen Ersteller von Regeln unterstützen, wenn er nicht weiß welche Sensorwerte ein bestimmtes Modul in einer bestimmten Situation von sich gibt.

4 Implementierung

4.1 Einführung

Am Anfang der Implementation stand die Frage mit welchen Mitteln und Technologien das Konzept umgesetzt werden soll. Dabei wurden verschiedene Entscheidungen getroffen, welche teilweise vorgegeben waren und teilweise noch nicht zu Beginn feststanden. In diesem Abschnitt soll erläutert werden welche Technologien eingesetzt wurden und warum. Eine genauere Beschreibung der einzelnen Mittel findet im nachfolgenden Abschnitt 4.2 statt.

Um den Server zu implementieren wurde **Node.js** gewählt, da es für Node.js schon viele Module gibt, welche verschiedenste Bereiche abdecken und damit nicht alles von Grund auf neu gemacht werden muss. Außerdem ist Node.js sowohl auf vielen Betriebssystemen verfügbar und kann damit leicht auf unterschiedlichste Betriebssysteme portiert werden. Als Hardware für den Server wurde sich aus Kostengründen für einen **Raspberry Pi** entschieden. Auf diesem Gerät läuft sowohl Linux und Node.js ohne größere Probleme.

Als Kommunikationsart wurde sich auf **xBee** bzw. **Wifi** beschränkt. Damit soll gezeigt werden, dass sich auch Geräte mit verschiedenen Kommunikationsarten miteinander in das System integrieren lassen. Dabei wurden zwei unterschiedliche Arten verwendet: xBee hat eine geringere Bandbreite als Wifi und man muss sich bei der Implementierung um mehr selbst kümmern. xBee wird durch eine serielle Schnittstelle angesprochen und verbraucht aber dafür nicht soviel Energie wie das Wifi. Wifi auf der anderen Seite wird heutzutage überall verwendet, es kann größere Datenmengen wie Videos oder Sounds ohne Probleme von einem Punkt zum anderen übertragen. Die Konfiguration des Wifis ist bedeutend einfacher, den man muss nur SID, Passwort und Verschlüsselung auswählen. Im Gegensatz dazu muss das xBee Netz schon vorher durch eine serielle Schnittstelle konfiguriert werden. Für die Übertragung mittels Wifi wurde sich aus wegen der geringeren Latenz als TCP für UDP entschieden.

Für das Übertragungsformat zwischen den Geräten war es wichtig, dass es möglichst ein leichtgewichtiges Format ist, um die zu übertragenen Daten auf ein Minimum zu beschränken, aber dennoch genug Aussagekraft zu besitzen um alle nötigen Datenstrukturen darzustellen. Aus diesem Grund wurde als Format **JSON** gewählt. Es ist ausdrucksstark genug, um alle benötigten Informationen und deren Struktur abzuspeichern und der Aufbau ist deutlich kleiner als etwa bei XML oder SOAP.

Als Mainboard Arten wurde **Gadeteer** und **Arduino** verwendet. Hiermit soll auch wieder gezeigt werden, dass sich grundsätzlich unterschiedliche Typen in das System integrieren lassen. Gadeteer wird mit C# und damit einem objektorientierten Ansatz programmiert.

Es gibt vorgefertigte Module, welche an bestimmten Ports angeschlossen werden können. Es hat genug Rechenleistung, um auch ein WLAN-Modul anzusteuern. Auf der anderen Seite ist Arduino kleiner und kompakter, aber es hat deutlich weniger Rechenleistung und Arbeitsspeicher.

4.2 Eingesetzte Technologien

In diesem Abschnitt werden die einzelnen Technologien beschrieben, welche bei der konkreten Implementierung des vorangegangenen Konzeptes verwendet wurden.

4.2.1 Raspberry Pi



Abbildung 4.1: Raspberry Pi: kleiner kreditkartengroßer Rechner mit USB und HDMI Anschluss. Es können Linux und Node.js darauf installiert werden. [Wik13f]

In [Fou13] und [Wik13f] wird der Raspberry Pi (Abb. 4.1 als ein etwa kreditkartengroßer Rechner, welcher einen ARM Prozessor und ein halbes Gigabyte an Arbeitsspeicher besitzt, beschrieben. Außerdem bietet er 2 USB Schnittstellen und auch einen HDMI Anschluss. Er wurde von der Raspberry Pi Foundation entwickelt, zu dem Zweck die immer schlechter werdenden Kenntnisse von Schülern und Studenten in der Informatik zu verbessern. Sie identifizierten mehrere Probleme die zu diesem Umstand führten: Zum einen, dass immer weniger Schulen die Geldmittel für ausreichend Computersysteme haben und zum anderen wollen viele Eltern nicht, dass ihre Kinder am Familien PC herumexperimentieren und den Computer möglicherweise beschädigen könnten. So entstand die Idee für ein günstiges und kleines System für das es viele Einsatzmöglichkeiten gibt. Aufgrund der für den kleinen Rechner vielen Schnittstellen kann das System als Media Center im Wohnzimmer oder etwa als kleiner energiesparender Server eingesetzt werden. Es wurden auch schon viel ausgefallener Möglichkeiten gefunden den Raspberry Pi einzusetzen: z.B. als Steuerplatine für einen Quadrocopter. Aus diesem Grund ist das System durch seinen günstigen Preis,

seinen kleinen Formfaktor und den universellen Einsatzmöglichkeiten für die Plattform geeignet.

4.2.2 Gadgeteer

Gadgeteer [Mic13a] ist eine Kombination aus Software und Hardware um es zu ermöglichen eigene elektronische Geräte herzustellen ohne größere Kenntnis von Mikrocontrollern. Das ganze System ist als Open Source, bzw. Open Hardware konzipiert, was es jedem ermöglicht das System sowohl in der Software, als auch in der Hardware zu erweitern. Es gibt Mainboards und Module, welche Sensoren und/oder Aktoren sein können. Man kann nun diese Module mit Hilfe von Kabeln mit dem Mainboard verbinden. Die Programmierung findet dann durch das .Net Micro Framework statt, welches es erlaubt auch diese Mikrocontoller mit C# und damit einer objektorientierung zu programmieren. Das macht den Einstieg für Programmierer, welche sonst nur Desktop Software entwickeln sehr leicht. Durch das Framework wird auch viel Arbeit abgenommen, da viele Funktionen schon vorhanden sind. Das erlaubt es innerhalb kürzester Zeit Ergebnisse zu liefern. Gedacht ist Gadgeteer sowohl für die Ausbildung von Schülern und Studenten, das anspruchsvolle Hobbyprojekt zu Hause oder auch für Firmen, welche schnell Prototypen entwickeln müssen.

4.2.3 Arduino

Arduino wird in [Ard13] und [Wik13a] ähnlich wie Gadgeteer beschrieben. Es ist dafür Gedacht schnelle Ergebnisse zu liefern. Es wird häufig in der Kunst für interaktive Installationen eingesetzt. Aber auch viele andere Einsatzmöglichkeiten sind denkbar. Anders als bei Gadgeteer arbeiten man bei Arduino auf einer niedrigeren Abstraktionsstufe. Die Programmierung findet über die Arduino Programming Language (ein abgespecktes C, C++) statt. Wie bei der Mikrocontroller Programmierung üblich hat man eine Hauptschleife, welche immer wieder durchlaufen wird. Auch muss man mehr über Elektronik wissen, um Sensoren und Aktoren an ein Arduino Mainboard anzuschließen, da es keine Vorgefertigten Module gibt. Es werden grundsätzlich analoger Eingang, also das Messen von Spannungen, digitaler Input und Output und Puls Weiten Modulation unterstützt. Ferner können serielle Schnittstellen verwendet werden. Arduino Mainboards gibt es in vielen verschiedenen Modellen und Größen. Genauso wie Gadgeteer wird bei Arduino auf Open Source und Open Hardware gesetzt. Es ist also möglich die Software und Hardware Designs selbst zu verwenden. Es werden viele Funktionen für den Zugriff auf die Mikrocontroller schon in der Standard Software mitgeliefert, so z.B. Funktionen für den Zugriff auf EPROM.

4.2.4 Node.js

Node.js von [Joy13] ist eine serverseitige Plattform für die Entwicklung von schnellen und skalierbaren Netzwerkanwendungen. Es kann insbesondere für Webanwendungen eingesetzt werden. Was Node.js so schnell macht ist die ressourcensparende, Event-basierte Architektur,

welche das System leichtgewichtig und effizient macht. Es benutzt die JavaScript Runtime V8 welche von Google für den Chrome Browser entwickelt wurde. Node.js wurde speziell für Daten-intensive Real-Time Anwendungen im Netzwerk entwickelt, welche auch für verteilte Systeme eingesetzt werden kann.

4.2.5 JavaScript und CoffeeScript

Da für den Server Node.js gewählt wurde, ist man automatisch auch auf eine Programmiersprache festgelegt: JavaScript. Um die Entwicklung zu beschleunigen und den Quellcode übersichtlicher zu gestalten wurde CoffeeScript verwendet. CoffeeScript benutzt keine Klammern und vereinfacht die Programmierung erheblich. Es kann für Node.js eingesetzt werden, da CoffeeScript in JavaScript übersetzt werden kann. Dabei wird vor allem die Lesbarkeit durch Ersetzen der Klammern, durch Einrückung und weiterem syntaktischem Zucker erreicht.

[Scr13, Wik13b, Wik13c]

4.2.6 JSON

Die Java Script Object Notation werden Objekte dargestellt und damit so gut wie alle Informationen serialisiert werden. Es ist ein kompaktes Datenformat, welches sowohl von Menschen also auch von Maschinen lesbar ist. Ein JSON-Objekt stellt dabei ein gültiges JavaScript Objekt dar, aber dennoch ist JSON von der Programmiersprache unabhängig, da für fast alle Programmiersprachen entsprechenden JSON Parser zur Verfügung stehen. Es ist ein weit verbreitetes Datenformat, welches deutlich kompakter ist als etwa XML oder SOAP.

[Wik13d, JSO13a]

4.2.7 xBee

xBee ist ein Funkmodule, welches von Digi International [Inc13] entwickelt wird. Es gibt verschiedene Versionen, mit unterschiedlichen Reichweiten, etc. Für die Funkübertragung wird der Funkstandard ZigBee [JSO13b] verwendet. Das ganze System kann große Netzwerke aufbauen und jedes Modul ist sehr Energieeffizient.

4.3 Gesamtsystem

4.3.1 Architektur

Die Architektur des Gesamtsystems besteht aus den Hardwarekomponenten: Server, Geräte und Infrastruktur. In der Abbildung 4.2 ist dafür ein Beispielaufbau gegeben. Das weiße

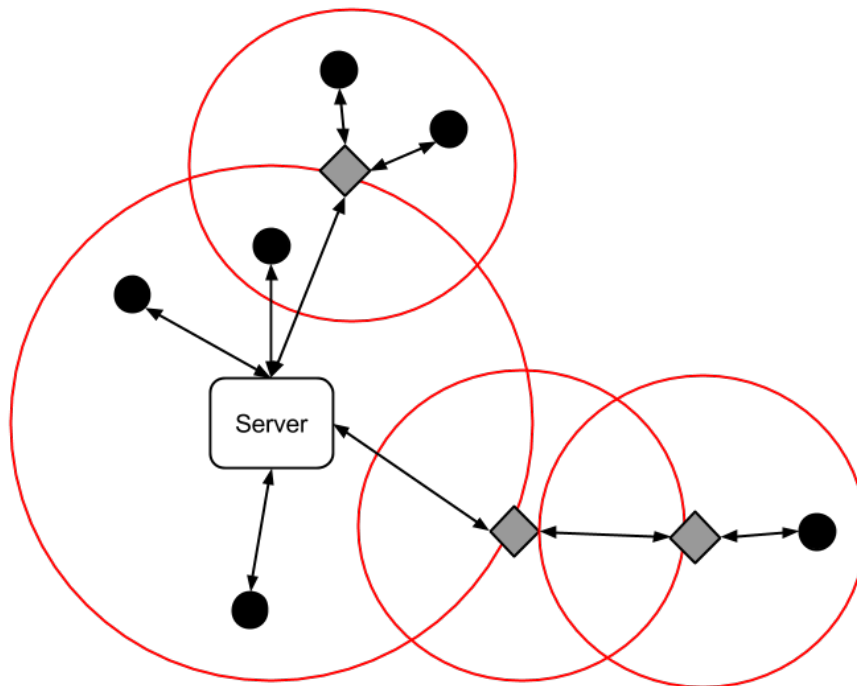


Abbildung 4.2: Allgemeine Architektur

Rechteck mit den abgerundeten Ecken ist der zentrale Server, die grauen Rauten stellen die Extender der Infrastruktur dar und die schwarzen Kreise stellen die Geräte unterschiedlichsten Typs dar. Der jeweilige rote Kreis um den Server und die Extender stellt die jeweilige Reichweite der drahtlosen Übertragung dar. Die Pfeile zwischen den Geräten und dem Server oder Extendern stellen die drahtlosen Verbindungen dar. Wenn ein Gerät außerhalb der Serverreichweite ist, aber innerhalb eines Extenders, so wird die drahtlose Kommunikation über den jeweiligen Extender an den Server weitergeleitet. Diese Extender stellen nur die Infrastruktur der jeweiligen Kommunikationsform dar und sind nur für Weiterleitungen und Vergrößerung des Netzwerkes zuständig.

4.3.2 Kommunikation und Schnittstellen

In der Abbildung 4.3 wird die Kommunikation zwischen den Geräten und dem Server als Sequenzdiagramm dargestellt. Dabei verwendet Arduino nur xBee und bei Gadgeteer ist es möglich entweder UDP, also Wifi oder Ethernet, oder xBee zu verwenden. Grundsätzlich gibt es vier Arten von Kommunikation: das Senden der eigenen Mainboard-Konfiguration, die Geräte-Konfiguration, Sensor Input und Aktor Output. Prinzipiell sendet jedes Gerät erst die eigene Mainboard-Konfiguration an den Server (Abb. 4.3 Punkt 1 und 5). In dieser Konfiguration steht um welche Art von Mainboard es sich handelt, welche Ports vorhanden sind, was diese können und um welche Kommunikationsart es sich handelt. Daraufhin antwortet der Server mit der Geräte-Konfiguration in der die angeschlossenen Module

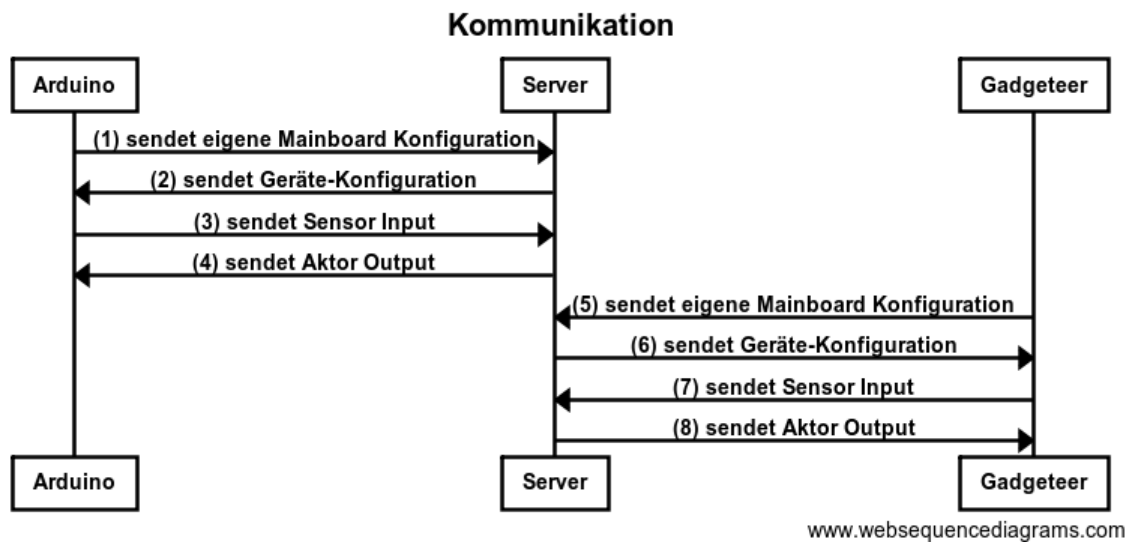


Abbildung 4.3: Kommunikation zwischen den beiden Geräten Arduino, Gadgeteer und dem Server

definiert sind (Abb. 4.3 Punkt 2 und 6). Wenn das Gerät dann die Module initialisiert hat, werden die Sensor Informationen zum Server geschickt (Abb. 4.3 Punkt 3 und 7). Der Server verarbeitet diese und löst gegebenenfalls die Aktoren an den Geräten aus (Abb. 4.3 Punkt 4 und 8). Zusätzlich dazu werden die Sensorwerte in der Datenbank gespeichert.

xBee

Um xBee zu verwenden, wurden für die Geräte jeweilige Bibliotheken verwendet, welche die xBee Modulsteuerung übernimmt. Da bei xBee es aber nicht möglich ist Nachrichten mit unbeschränkter Länge zu verschicken, wurde in die eigenen Nachrichten ein eigener Header eingebaut. Dieser Header besteht aus 3 Bytes, das erste stellt das Startbyte dar und wird gefolgt von zwei Bytes, welche die Länge der folgenden Nachricht definiert. Wenn xBee Nachrichten von einem Gerät zum Server geschickt werden oder umgekehrt, kommen die Nachrichten zerstückelt an und müssen beim Empfangen wieder zusammengesetzt werden. Dazu wird nach dem Startbyte gesucht und die Länge der Nachricht durch die zwei weiteren Bytes berechnet. Anschließend muss noch gewartet werden bis alle Bytes der Nachricht eingetroffen sind und die Gesamtnachricht zusammengesetzt werden kann.

UDP

Die Kommunikation mit UDP über Wifi oder Ethernet ist sehr einfach und unkompliziert, da man einfach eine Nachricht versendet und auf der anderen Seite empfangen kann. Um den Rest kümmert sich die Software, bzw. Hardware. Weiterhin hat es den Vorteil, das bei Wifi,

bzw. Ethernet eine weitaus höhere Bandbreite zur Verfügung steht als bei xBee. Dadurch und den Zugriff auf das Netzwerk können den Geräten Urls übergeben werden. Diese können dann von dem Geräten benutzt werden um Webseiten anzuzeigen oder Bilder, Videos usw.

Konfigurationen

In diesem Abschnitt werden Beispiele für die verschiedenen Kommunikationstypen vorgestellt welche das System hat. Die Konfigurationen sind alle im JSON Format, da dieses sehr leichtgewichtig und flexibel ist. Außerdem gibt es für sehr viele Programmiersprachen eine Bibliothek dafür.

Geräte-Konfiguration Dies zeigt die Konfiguration welche vom Server an die Geräte gesendet wird. Sie enthält vorallem die angeschlossenen Module und deren Eigenschaften. Das folgende Beispiel zeigt eine Konfiguration von einem Gadgeteer Mainboard (FEZ Spider) mit einem Button Modul angeschlossen an den Port 4. Die Konfiguration ist in zwei Teile aufgebaut: Erstens die Inforamtionen zum Mainboard wie Name, Version und eingesetzte Technologie. Dadurch kann auf dem Gerät verhindert werden, das die Konfiguration eigentlich für eine andere Hardware gedacht ist. Danach folgt eine Liste mit Modulen. Ein Modul besteht dabei aus einem Namen, der Technologie und den Ports an denen es angeschlossen werden kann. Dabei gibt es auch Module, welche mehrere Ports benötigen und gleichzeitig auch mehere verschiedene Ports unterstützen. In diesem Fall benötigt der Button nur einen Port und dieser Port muss vom Typ X oder Y sein. Wenn es ein weiteren Port benötigen würde, dann würde im ersten Array ein weiteres Array eingefügt werden. Dieses Array hätte dann wiederum die benötigte Portarten gespeichert. Danach folgen zwei weitere Abschnitte, einmal die Einstellungen für den Input und die Einstellungen für den Output. Da dieses Modul eine eingebaute LED besitzt, hat es beide Bereiche. Im Input Bereich sind die Einstellungen enthalten, z.B. wann ein Sensorwert versendet wird: entweder Timer mit einem festen Interval wie z.B. jede Sekunde, ValueChanged wenn sich ein Wert ändert oder Threshold mit einer Einschränkung wie kleiner gleich oder ähnlichem in Verbindung mit einem Wert. Desweiteren Enthält diese noch eine Beschreibung der Werte. Da es möglich ist, dass ein Modul mehrere verschiedene Werte senden kann, muss irgendwo definiert sein wie diese heißen. Am Beispiel eines Gadgeteer Beschleunigungssensors kann dieser die Beschleunigung der X, Y und Z Achse ausgeben. Im Bereich Output wird festgehalten welche Elemente angesteuert werden können und welche Werttypen es gibt. In dem Beispiel des Button Modules gibt es eine LED und diese kann ein- bzw. ausgeschalten werden (bool) oder mit einer Frequenz blinken (frequenz). Die Beispiel Konfiguration befindet sich in Listing 4.1.

Sensor Informationen Wenn ein Gerät Sensorwerte an den Server schickt, dann geschieht dies immer im gleichen Format. Zuerst wird die Nachrichtentyp als SensorInput festgelegt, dann wird die eindeutige Adresse des Gerätes angegeben und um welches Modultyp es sich handelt. Weiterhin wird zur entgültigen festlegung des Modules angegeben an welchen Ports

Listing 4.1 Beispiel einer Konfiguration fuer ein Gerat:

```
{
  "Mainboard": {
    "Name": "GHIElectronics.Gadgeteer.FEZSpider",
    "Version": "1.0",
    "Technology": "Microsoft Gadgeteer"
  },
  "Modules": [
    {
      "Name": "Gadgeteer.Modules.GHIElectronics.Button",
      "Technology": "Microsoft Gadgeteer",
      "PortType": [
        [
          [
            "X",
            "Y"
          ]
        ]
      ],
      "Input": {
        "Type": "ValueChanged",
        "ValueDescription": [
          "Button"
        ]
      },
      "Output": {
        "ValueDescription": "LED",
        "PossibleValueTypes": [
          "bool",
          "frequenz"
        ]
      },
      "Port": [
        "4"
      ]
    }
  ]
}
```

Listing 4.2 Beispiel einer Sensornachricht:

```
{
  "messageType": "SensorInput",
  "address": "00-23-A7-1D-59-C3",
  "moduleType": "Gadgeteer.Modules.GHIElectronics.Accelerometer",
  "portNumber": "4",
  "valueDescription": "Y",
  "valueType": "double",
  "value": 1.0
}
```

Listing 4.3 Beispiel einer Aktornachricht:

```
{
  "messageType": "AktorOutput",
  "address": "00-23-A7-1D-59-C3",
  "moduleType": "Gadgeteer.Modules.GHIElectronics.Button",
  "portNumber": "4"
  "valueDescription": "LED",
  "valueType": "frequenz",
  "value": 10.0
}
```

das Modul angeschlossen ist. Bei der Beschreibung des Wertes handelt es sich wie vorher schon erwähnt um ein Unterscheidungsmerkmal für verschiedene Werte, welche ein Modul senden kann. Im vorherigen Beispiel des Beschleunigungssensors könnte das entweder X, Y oder Z sein. Der Wertetyp beschreib ob es sich um eine Fließkommazahl handelt oder einen Text usw. Am Ende wird noch der eigentliche wichtige Wert mitgeschickt. Ein Beispiel für den Sensor Input als JSON befindet sich in Listing 4.2.

Aktor Ausgabe Wenn man einen Aktor an einem Gerät ansteuern will, sendet der Server eine Nachricht an das Gerät. Der Aufbau ist genau gleich wie beim Sensor Input: Am Anfang wird der Nachrichtentyp als AktorOutput definiert. Dann folgen wieder eindeutige Adresse des Gerätes, der Modultyp und die Ports an dem es angeschlossen ist. Die Wertebeschreibung stellt den Ausgabekanal dar. In dem konkreten Beispiel ist es eine LED, welche im Buttonmodul integriert ist. Dann folgt der Werttyp, in diesem Fall eine Frequenz mit der die LED blinken soll, und dann der Wert. Diese Nachricht würde an dem Gerät mit dem Button an Port 4 die LED mit 10 mal pro Sekunde blinken lassen. Dargestellt wird dieses Beispiel im Listing 4.3

Schnittstelle zur ConfApp Im Listing 4.4 ist die JSON Datei dargestellt, welche zwischen der Plattform und der ConfApp ausgetauscht werden. Diese enthält weit mehr Informationen als für die Initialisierung der Geräte notwendig ist. Deshalb wird aus dieser Konfiguration die Konfiguration aus Listing 4.1 erzeugt. Diese Konfiguration ist im Prinzip ähnlich wie die Geräte-Konfiguration aufgebaut, nur das sie mehr Informationen enthält und die Verbindung mit Modulen über Ports gegliedert ist und nicht über Module.

4.4 Server

4.4.1 Architektur

Die Hardware Komponente Server enthält mehrere logische Komponenten: Datenbank, Weboberfläche, Kommunikation, Regeln. In der Komponente Datenbank werden alle Daten persistent festgehalten wie z.B. Geräte Konfigurationen, Regeln, usw. Die Kommuni-

Listing 4.4 Beispiel der Konfiguration zwischen der ConfApp und der Plattform:

```
{
  "_description": "GHIElectronics.Gadgeteer.FEZSpider",
  "name": "GHI FEZ Spider Mainboard",
  "version": "1.0",
  "devType": "ghi_fez_spider_mainboard",
  "platformType": "WiFi/Ethernet",
  "platformTargetURL": "",
  "platformName": "",
  "platformMAC": "",
  "moduleTypes": [
    "Microsoft Gadgeteer"
  ],
  "imgUrl": "/db/Spider%20Mainboard_0.jpg",
  "devInfo": [],
  "details": [],
  "links": [],
  "ports__description": "bindTo should be the moduleobject",
  "ports_hardcoded": [
    {
      "id": "1",
      "state": "empty",
      "supportedPortClasses": [
        "D",
        "I"
      ],
      "bindTo": null
    },
    {
      "id": "2",
      "state": "empty",
      "supportedPortClasses": [
        "Z"
      ],
      "bindTo": null
    }
  ]
}
```

kationskomponente abstrahiert die verschiedenen Kommunikationsarten für die weitere Verarbeitung. Diese beinhaltet weitere logische Komponenten welche den einzelnen Kommunikationsarten entsprechen. In der Weboberfläche werden alle wichtigen Einstellungen und Konfigurationen der Geräte und Regeln auf dem Server gemacht. In der letzten Komponente werden die jeweiligen Regeln interpretiert und auch ausgeführt. Diese Komponente stellt die Interaktivität des Systems bereit. In der Abbildung 4.5 ist dieser Zusammenhang graphisch dargestellt. In der rechten Ecke ist die Komponente Weboberfläche (Web UI) in orange, die Komponenten Kommunikation (Communication) in rosa mit vier Beispielen für Kommunikationsarten: xBee, Ethernet, WLAN und Bluetooth dargestellt. In dem Teil der blau dargestellt ist, befindet sich als Datenbank Komponente die blauen Zylinder, welche jeweils

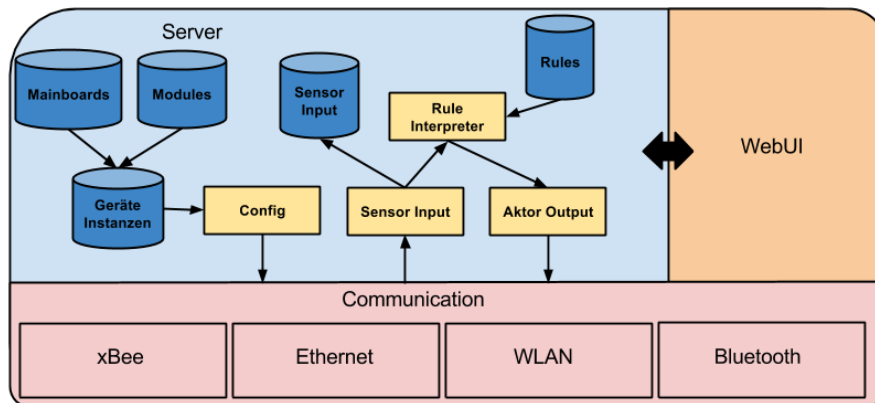


Abbildung 4.4: Server Architektur

eine Datenbanktabelle darstellen. Die Komponente Regeln wird durch das gelbe Rechteck Rule Interpreter dargestellt.

4.4.2 Datenbank

Die Datenbank ist für die persistente Sicherung der Daten verantwortlich. Dafür wird eine MySQL Datenbank verwendet mit Tabellen für die Konfigurationen, die Regeln und den Sensorinput.

4.4.3 Kommunikation

Die Kommunikationskomponente abstrahiert die Daten von dem verwendeten Kommunikationskanal. So muss für die weitere Verarbeitung im Server keine Rücksicht mehr auf den Kommunikationskanal gelegt werden. Dabei wird für jeden Kommunikationskanal eine eigene Komponente entwickelt, welche durch die Kommunikationskomponente verwendet wird. In diesen Komponenten werden Funktionen zum Empfangen von Daten, welche dann als Event in der Kommunikationskomponente landen, oder zum Senden von Daten bereitgestellt.

4.4.4 Regeln

Die Regeln werden mit JavaScript realisiert. Der Zugriff auf die einzelnen Werte der Sensoren oder Aktoren lassen sich über Objekte beeinflussen. Zur Sicherheit wird jede Regel in einer Sandbox ausgeführt, damit das System nicht gefährdet wird. Es können verschiedene JavaScript Funktionen und Objekte verwendet werden. So ist es möglich z.B. auf Datumsfunktionen zuzugreifen oder mit einem asynchronen Warte Befehl nach einer bestimmten Zeit weiteren Code auszuführen. Dadurch das mit JavaScript der komplette Umfang einer

Programmiersprache zur Verfügung steht, können auch sehr komplexe Regeln erstellt werden. Anders als im Konzept gibt es bei der gerade existierenden Implementierung keine Prioritäten der Regeln sondern sie werden in der Reihenfolge des Erstellens ausgeführt.

4.4.5 Weboberfläche

Die Weboberfläche wird in einem nachfolgenden Kapitel näher erörtert, da es die Schnittstelle zwischen Benutzer und System darstellt.

4.4.6 Erweiterbarkeit

Es gibt die Möglichkeit den Server um weitere Kommunikationsmethoden zu erweitern. Dazu enthält die Kommunikationskomponente weitere Komponenten welche den einzelnen Arten entspricht. Bis jetzt sind xBee und UDP vorhanden. Diese einzelnen Module bieten jeweils eine Senden Funktion an und schicken ein Event an die übergeordnete Kommunikationskomponente wenn Daten reinkommen. Die Kommunikationskomponente erhält dabei die vollständige Nachricht und von welcher Geräteadresse sie stammt. Sie ist nun für die Abstraktion vom jeweiligen Kommunikationskanal zuständig. Wenn die Daten weiter aufbereitet sind wird sie an die Hauptkomponente zur Regelverarbeitung weitergeschickt. Andersrum wird bei der Kommunikationsschicht die Senden Funktion aufgerufen, wird dort ermittelt um welche Kommunikationsart es sich handelt und an das jeweilige Modul weitergereicht. Durch dieses System kann man den Server leicht um weitere Kommunikationmittel erweitern.

4.5 Geräte

4.5.1 Allgemein

Architektur

Die Allgemeine Architektur eines Gerätes beinhaltet immer die zwei logischen Komponenten für die Kommunikation und für die Interpretation der Konfiguration. Die Kommunikationskomponente regelt dabei den Datenverkehr zwischen Gerät und Server, ist also für den Server Input an den Server und den Aktor Output am Gerät mitverantwortlich. Die Interpretationskomponente wird nur zum Beginn der Geräteinitialisierung benötigt. Wenn ein Gerät eingeschaltet wird, dann fragt es seine Konfiguration beim Server an und diese wird an das Gerät gesendet. Daraufhin wird diese Konfiguration vom Interpreter interpretiert und die entsprechenden Module, welche an das Gerät angeschlossen sind, werden initialisiert.

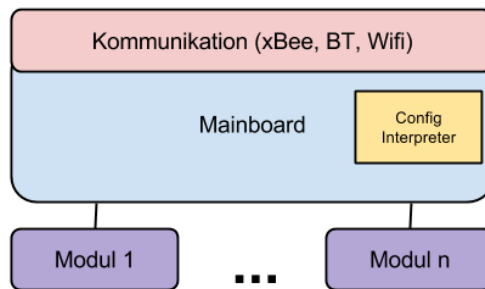


Abbildung 4.5: Diese Abbildung stellt die Architektur eines generischen Gerätes dar. Grundsätzlich besitzt jedes Gerät zwei wichtige Komponenten: die Kommunikationsschicht und den Konfigurations Interpreter. Über die Kommunikationsschicht werden alle Daten vom Server empfangen bzw. dort hin übertragen. Der Konfigurations Interpreter liest die Konfiguration ein und initialisiert die entsprechenden Module.

4.5.2 Gadgeteer

Gadgeteer wird objektorientiert mit C# entwickelt und unterstützt Programmierung und Debugging über das USB-Kabel. Durch das von Microsoft entwickelte .NET Micro Framework werden schon viele Funktionen mitgeliefert, was die Entwicklungszeit reduziert. Der Zugriff auf die einzelnen Module geschieht über die Manipulation von Objekten.

Architektur

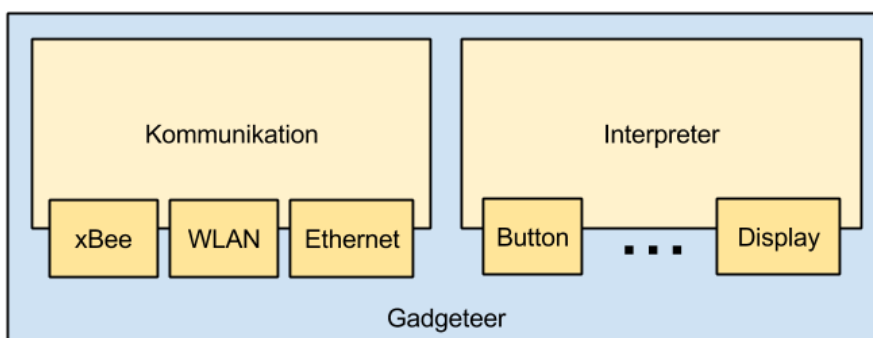


Abbildung 4.6: Architektur Gadgeteer

In Abbildung 4.6 ist die konkrete Architektur des Gadgeteer Mainboards dargestellt. Es besitzt die zwei Hauptkomponenten Kommunikation und Interpreter. Die Kommunikation ist wiederum unterteilt in xBee, WLAN und Ethernet, welches je nach gewünschtem Kommunikationskanal in der Software aktiviert werden muss. Der Interpreter interpretiert

die JSON Konfiguration und initialisiert die vorhandenen Module, welche jeweils als eigene Komponenten integriert sind.

Verwendetes Mainboard

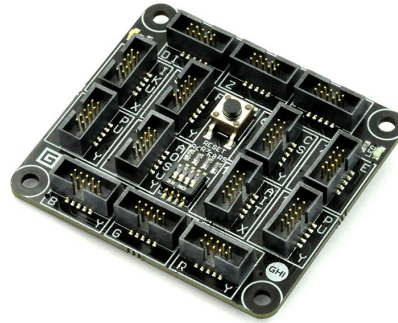


Abbildung 4.7: FEZ Gadgeteer Mainboard zum Anschluss von bis zu vierzehn Modulen.

Für die Implementierung der Gadgeteer Software wurde das FEZ Mainboard, welches in Abbildung 4.7 zu sehen ist, verwendet. Es besitzt vierzehn Ports an dem Module angeschlossen sind. Die Software ist so konfiguriert, dass sie immer von einem Strommodul an Port 1 ausgeht und einem weiteren Kommunikationsmodul, welches an einem passenden Port angeschlossen ist. Als Kommunikationsmodule sind implementiert worden: Wifi, Ethernet und xBee. Für jedes Kommunikationsmodul muss eine extra Software auf dem Mainboard geflasht werden.

Unterstützte Module

Da Gadgeteer Objektorientiert aufgebaut ist, werden die einzelnen Module als Objekte eingebunden. Da es aber kein generisches Modul gibt, mit dem man Zugriff auf die einzelnen Funktionen hat, muss jedes Modul einzeln implementiert werden. Nachfolgend werden alle Module vorgestellt, welche benutzt werden können.



Abbildung 4.8: Gadgeteer Button Modul mit eingebauter LED.

Button Der Button in Abbildung 4.8 besitzt eine eingebaute LED. Dieses Modul kann also sowohl als Input als auch als Output genutzt werden. Für den Input kann man das Modul so konfigurieren, dass es alle x Millisekunden den aktuellen Wert, 1 für gedrückt und 0 für nicht gedrückt, an den Server sendet oder Eventbasiert, durch drücken oder loslassen ausgelöst. Die LED leuchtet rot und kann entweder ein- und ausgeschaltet werden oder mit einer Frequenz zum blinken gebracht werden.

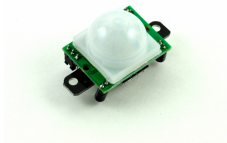


Abbildung 4.9: Gadgeteer PIR Modul als Bewegungssensor.

Bewegungssensor Der Bewegungssensor in Abb. 4.9 ist eine reine Sensorquelle. Es kann sowohl wie der Button über einen Timer die Werte an den Server senden, 1 für Bewegung und 0 für keine Bewegung, oder wieder Eventbasiert, wobei nur eine Nachricht gesendet wird, wenn Bewegung registriert wird.



Abbildung 4.10: Gadgeteer LED Modul.

LED Das Gadgeteer LED Modul (Abb. 4.10) kann wie die LED des Button-Modules verwendet werden: Es kann ein- bzw. ausgeschaltet werden oder zum blinken gebracht werden.

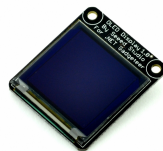


Abbildung 4.11: Gadgeteer OLED Modul zum Anzeigen von Text.

OLED Display Abbildung 4.11 zeigt ein Display welches Text darstellen kann. Dabei kann bei diesem Aktor entschieden werden ob sich der ganze Text ändert oder es können bestimmte Zeilen verändert werden.



Abbildung 4.12: Gadgeteer Joystick Modul: Es kann die X und Y Richtung detektiert werden und ein klicken durch den Joystick.

Joystick Der Joystick in Abbildung 4.12 kann die X und Y Richtung des Joysticks als Sensorwerte ausgeben und wie beim Buttonmodul ein klicken durch den Joystick-Button. Dabei gibt es drei verschiedene Arten die Sensorwerte zu verschicken: Wieder mit einem Timer, Eventbasiert, wenn der Button gedrückt oder losgelassen werden und durch einen Schwellwert. Dieser Schwellwert bestimmt ob die Daten an den Server geschickt werden sollen oder eben nicht.



Abbildung 4.13: Gadgeteer Potentiometer Modul.

Potentiometer Das Modul aus Abbildung 4.13 stellt ein Potentiometer dar, welches die aktuelle Stellung des Drehknopfes ausgibt. Dafür gibt es zwei Arten der Ausgabe: Einmal über einen Timer und einmal über einen Schwellenwert welcher über- bzw. unterschritten werden muss um Daten an den Server zu senden.

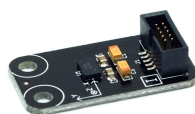


Abbildung 4.14: Das Gadgeteer Beschleunigungssensor Modul misst die Beschleunigung in alle 3 Achsen: X, Y und Z und gibt die Beschleunigung getrennt für die Werte aus.

Beschleunigungssensor Der Beschleunigungssensor (Abb. 4.14) ist eine reine Sensorquelle, sie gibt die jeweilige Beschleunigung nach den dreidimensionalen Achsen X, Y und Z getrennt aus. Die Art der Ausgabe geht entweder über einen Timer, welcher zum jeweiligen

Zeitpunkt die Daten an den Server sendet oder über einen Schwellenwert welcher erreicht werden muss aus.



Abbildung 4.15: Das Modul Lichtsensor des Typs Gadgeteer gibt die Lichtintensität aus.

Lichtsensor Das Modul Lichtsensor vom Typ Gadgeteer gibt die Lichtintensität, entweder durch einen Timer oder durch einen Schwellenwert getriggert aus. Der Sensor ist in Abbildung 4.15 dargestellt.



Abbildung 4.16: Extender Modul von Gadgeteer zum Anschluss von weiterer Elektronik durch 7 Anschlusspins und 3 Versorgungspins (3.3V, 5.0V und GND)

Extender Abbildung 4.16 zeigt ein Modul, welches es ermöglicht angeschlossene Elektronik entweder als Analog Input oder Digital Input einzulesen oder als Ausgabe für PWM oder Digitalen Output zu verwenden. Das Modul besitzt 7 Anschlussmöglichkeiten, welche getrennt konfiguriert werden können, einen 3.3V, einen 5.0V und einen GND Pin zur Versorgung der angeschlossenen Elektronik. Die Pins können ihre Werte, falls als Input konfiguriert wieder mit einem Timer oder durch einen Schwellwert gesteuert an den Server senden.

Erweiterbarkeit

Grundsätzlich muss für jedes Gadgeteer Mainboard eine eigene Firmware geschrieben werden, jedoch können viele Komponenten, welche schon vorhanden sind, genutzt werden. Dadurch lässt sich das bereits vorhandene Programm schnell an neue Gadgeteer Mainboards anpassen. Um die Gadgeteer Software mit weiteren Modulen auszustatten, ist es notwendig, aufgrund des objektorientierten Ansatzes, jedes neue Modul einzeln in den Konfigurationsinterpreter einzubinden.

4.5.3 Arduino

Arduino wird mit C wie ein normaler Mikrocontroller programmiert. Es wird durch Arduino schon einige Bibliotheken zur Verfügung gestellt, welche häufig benutzte Funktionen, wie der Zugriff auf EEPROM, Serielle Schnittstellen oder ähnliches, liefert. Da in diesem Fall das Arduino Fio Mainboard verwendet wurde, war es schwieriger über die Serielle Schnittstelle zu debuggen, da diese Schnittstelle auch für die Kommunikation mit dem xBee Modul verantwortlich ist.

Architektur

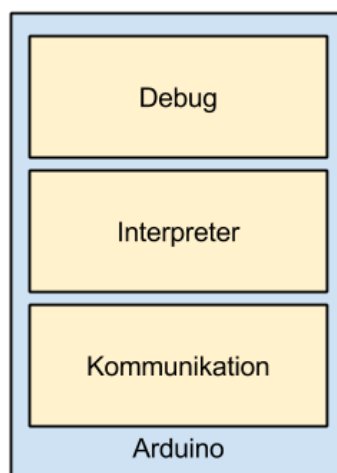


Abbildung 4.17: Architektur Arduino

In Abbildung 4.17 ist die konkrete Architektur des Arduino Mainboards dargestellt. Es besitzt drei wichtige Komponenten: den Interpreter für die vereinfachte Konfiguration für Arduino, die Kommunikation mit dem Server mittels xBee und die Debug Komponente, in der Funktionen sind, welche es erlauben Debug Informationen über xBee an den Server zu schicken, da die Serielle Schnittstelle für xBee verwendet wird.

Verwendetes Mainboard

Als Mainboard für Arduino wurde das Arduino Fio verwendet, da es tragbar ist durch seinen kleinen Formfaktor und auch schon einen Steckplatz für ein xBee-Modul besitzt. Außerdem kann daran ein Akku angeschlossen werden, welcher mit der USB-Buchse am Arduino direkt geladen werden kann. Der Fio verfügt über 8 Analog Port und 12 Digitale Ports wovon 6 als PWM Kanal verwendet werden können. Abbildung 4.18 zeigt dieses Mainboard.

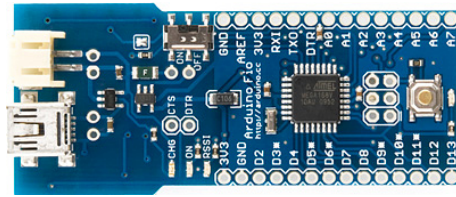


Abbildung 4.18: Arduino Fio Mainboard mit vorhandenem xBee Steckplatz und Ladeelektronik über USB.

Unterstützte Module

Da der Arduino grundsätzlich wie ein Mikrocontroller aufgebaut ist und auch so programmiert werden kann, kann man an die Ports verschiedene Arten von Sensoren und Aktoren verwenden. Diese sind sehr generisch und werden im weiteren näher erläutert.

Analog In Beim Analog In des Arduinos wird einfach eine Spannung gemessen. Dadurch lassen sich analoge Sensoren anschließen wie z.B. ein Lichtsensor oder ein Potentiometer. Da durch Rauschen die Spannung sehr stark Schwanken kann und es somit immer eine Veränderung des Wertes gibt, werden diese Daten nur über einen Timer

PWM Manche der digitalen Ports des Arduinos unterstützen die Ausgabe der Puls Weiten Modulation (PWM). Diese kann durch eine Funktion des Arduinos aufgerufen werden, welche es ermöglicht eine bestimmte Spannung an den Port anzulegen. Die Funktion wird als Ausgabe mit einem Wert zwischen 0 und 255 verwendet.

Digital In Beim Digital In wird z.B. bei einem Button als Sensor entschieden ob dieser gedrückt wird oder eben nicht. Es gibt als genau zwei Werte 0 und 1. Dabei ist es leicht möglich immer nur Werte auszugeben wenn sich diese ändern (Event basiert) oder mit einem Timer zu einem immer festen Intervall.

Digital Out Der digitale Output des Arduinos unterstützt es die Spannung ein- oder auszuschalten. Damit lässt sich z.B. eine LED steuern.

Erweiterbarkeit

Die Erweiterungsmöglichkeiten beim Arduino sind sehr generisch. So könnte man z.B. noch eine Serielle Schnittstelle anbieten oder I²C.

4.6 Weboberfläche

Die Weboberfläche wird zur Verwaltung der Geräte und Regeln verwendet. Zusätzlich dazu lassen sich manuell die Aktoren steuern und es können die Sensordaten angesehen werden. In dem nachfolgenden Bereich werden die einzelnen Seiten der Weboberfläche erläutert, welche das gesamte System konfiguriert und steuert.

4.6.1 Geräte

Dieser Abschnitt beschäftigt sich mit der Verwaltung der Geräte. Dies geht von der Erstellung einer Konfiguration bis zu zum editieren bzw. Löschen der Geräte.



Abbildung 4.19: Geräte verwalten

In der Abbildung 4.19 werden alle Geräte, welche dem Server bekannt sind angezeigt. Diese werden in einer Tabelle angezeigt, welche neben dem Namen des Gerätes auch die eindeutige Adresse und die verwendete Kommunikationsart anzeigt. Außerdem wird angezeigt wann die letzte Aktivität mit dem Server war. Ganz am Ende jeder Zeile finden sich die Buttons zum Anzeigen von weiteren Informationen, zum Bearbeiten und Löschen der Geräte. Wenn man auf Anzeigen (Show) oder Bearbeiten (Edit) klickt, kommt man zu einem Fenster, welches in Abbildung 4.20 dargestellt wird. Beim Anzeigen lassen sich lediglich die Einträge nicht verändern.

Einstellungen der einzelnen Geräte lassen sich in dem Fenster in Abbildung 4.20 machen. Hier kann man alle allgemeinen Informationen des Gerätes ändern, sowie die Konfigurationen manuell ändern. Wenn man die Konfiguration des Gerätes mit einem Assistenten bearbeiten bzw. erstellen möchte, gelangt man mittels des Buttons Create Config zur ConfApp, welche von [RR13] entwickelt wurde.

Die ConfApp (Abb. 4.21) lässt den Benutzer auf einfache Weise das Mainboard per Drag'n'Drop mit den vorhandenen Modulen bestücken. Es lassen sich durch klicken auf zugewiesene Module weitere Einstellungen machen wie die Art wann Sensorwerte gesendet werden. Zusätzlich dazu gibt es einen Auto Layouter, welcher durch doppelklicken auf ein Modul aus der Liste dieses auf einem freien Port plaziert. Anschließend wird die Konfigurationsdatei wieder auf den Server zurückgeleitet und dort verarbeitet.



Abbildung 4.20: Assistent zum manuellen erstellen eines Gerätes auf dem Server

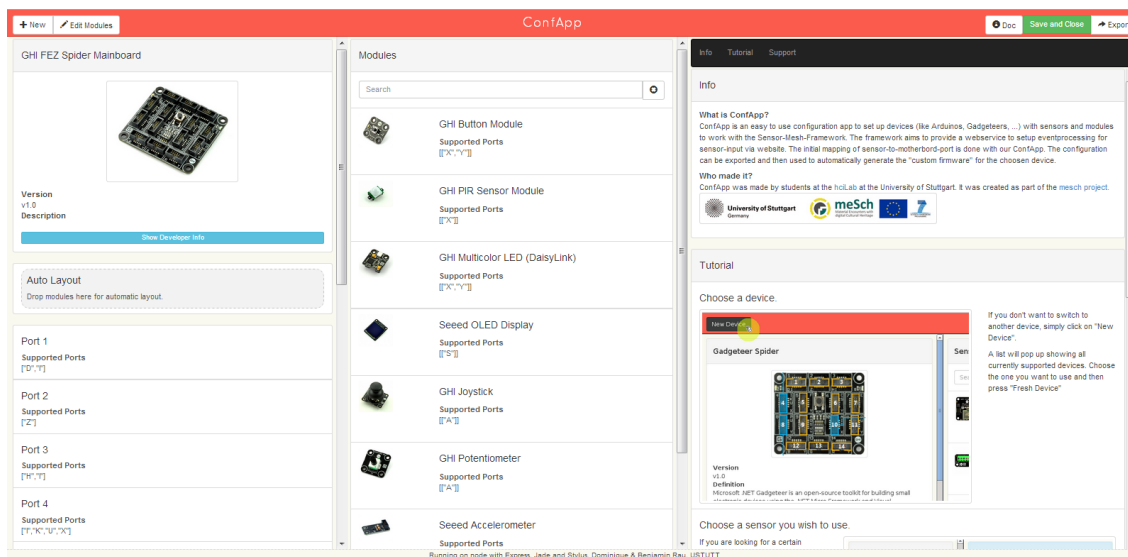


Abbildung 4.21: ConfApp zum einfachen erstellen einer Konfigurationsdatei für Mainboards mit verschiedenen Modulen

4.6.2 Regeln

Damit das System auch eine Interaktion bieten kann, muss vom Anwender Regeln erstellt werden, welche ausdrücken was passiert wenn Sensoren bestimmte Werte messen. Um möglichst auch unerfahrene Anwender zu unterstützen gibt es einen Assistenten (Abb. 4.22), welcher einfache Wenn-Dann-Regeln bietet. Um aber auch komplexere Regeln zu erstellen, gibt es weiterhin die Möglichkeit mittels Java Script und für eingeführten Variablen, welche den jeweiligen Sensoren und Aktoren entsprechen, komplexere Lösungen zu erstellen. Auch

4 Implementierung

Abbildung 4.22: Assistent zum unterstützen unerfahrener Anwender durch einfache Wenn-dann-Regeln.

für die Regeln gibt es eine verwaltende Oberfläche in denen die Regeln angezeigt, bearbeitet und gelöscht werden können.

4.6.3 Aktoren steuern

Außer der Möglichkeit Aktoren durch Regeln auslösen zu lassen, gibt es noch die Möglichkeit, die Aktoren manuell von der Weboberfläche aus (Abb. 4.23) zu steuern. Somit kann schnell und einfach z.B. eine Lampe ausgeschaltet werden, falls diese nicht mehr benötigt wird, ohne erst eine Regel dafür erstellen zu müssen.

4.7 Herausforderungen

Dadurch dass in dieses System verschiedenste Hardware integriert ist, angefangen vom Raspberry Pi als Server welches Node.js installiert ist und mittels Java Script bzw. Coffee Script programmiert wird, dem Gadgeteer, welches objektorientiert mit C# implementiert wird, bis zum Arduino. Dabei wird Arduino, wie bei vielen Mikrocontroller üblich, mit C programmiert. Dabei muss man sich selbst um die meisten Dinge kümmern wie das keine

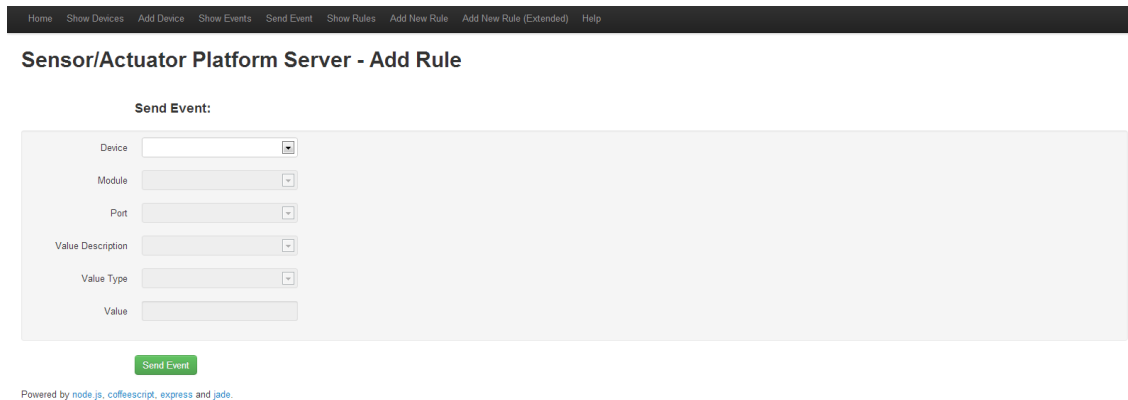


Abbildung 4.23: Events an Aktoren lassen sich mit Hilfe dieser Seite an Aktoren senden und somit Displays, Leds und weitere Aktoren auch über die Website steuern

Speicherlecks entstehen, da kein Garbage Collector vorhanden ist. Weiterhin sind verschiedene Kommunikationskanäle wie UDP über Wifi und xBee verwendet worden. Dies alles führt dazu, dass man sich in die verschiedensten Konzepte und Programmierparadigmen einarbeiten muss, es wird außerdem für jeden Bereich eine eigene Entwicklungsumgebung eingesetzt. Daraus folgt, dass es Schwierig ist Fehler zu finden, da z.B. beim Arduino die Debugging funktionen sehr eingeschränkt sind. Außerdem gibt es viele Kombinationsmöglichkeiten, welche aus den verschiedenen Geräten in Kombination mit den verschiedenen Kommunikationsarten resultieren.

5 Evaluation

5.1 qualitative Evaluation

In diesem Teil der Evaluation geht es um eine Studie um die Benutzbarkeit des Systems zu untersuchen. Dazu wurden 5 Probanden zu einer Studie eingeladen, die bestimmte Aufgaben lösen mussten. Einführend wurde der Zweck des Systems erklärt und die Funktionsweise wie z.B. die Weboberfläche und die Konfiguration der Geräte. Für die Studie sind zwei Gadgeteer Mainboards, jeweils mit WLAN Module und Strommodul, eingesetzt worden. Zusätzlich wurden die Gadgeteer Module Button, Bewegungsmelder (PIR Sensor), Joystick, RFID, Lichtsensor, Beschleunigungssensor, OLED Display zur Verfügung gestellt. Die beiden Mainboards wurden einmal mit blau und einmal mit rot markiert für die bessere Unterscheidbarkeit. Desweiteren wurde ein WLAN-Netz, ein Server und ein Rechner für den Zugang zur Weboberfläche bereitgestellt. Die Probanden mussten jeweils die Aufgaben lösen und anschließend den empfundenen Schwierigkeitsgrad auf einer Skala von 1 (leicht) bis 5 (schwer) bestimmen. Außerdem wurde jeweils die Zeit für die einzelnen Aufgaben gemessen.

Als erste Aufgabe sollten die Probanden das rote Gadgeteer Mainboard (mit schon WLAN und Strom Modul angeschlossen) nach einer vorhandenen Geräte-Konfiguration auf dem Server mit den entsprechenden Modulen ausstatten. Die Konfiguration bestand aus einem Beschleunigungssensor am Port 4, einem Button an Port 5, einem RFID Modul an Port 8, an Port 9 ein Oled Display und zu letzt an Port 10 ein Joystick.

Die zweite Aufgabe war es das blaue Gadgeteer Mainboard am Server anzumelden und den Namen in Blau zu ändern. Außerdem musste die vorhandene Bestückung mit Modulen am blauen Mainboard in der Weboberfläche als Konfiguration erstellt werden. Dazu war am blauen Mainboard ein Bewegungsmelder an Port 4, ein Button an Port 5 und ein Lichtsensor an Port 10 verbunden.

Die dritte Aufgabe war mit den zwei vorhandenen Geräten und deren Konfiguration die ersten Regeln anzulegen. Dazu sollte der vorhandene Assistent verwendet werden. Ziel war es hier, dass der Button des einen Modules die LED am Button des anderen Modules steuert und umgekehrt. Dazu mussten 4 Regeln angelegt werden um einzustellen was beim Drücken und was beim Loslassen des einzelnen Buttons geschieht.

Für die vierte Aufgabe wurde alles wieder auf Grundzustand gesetzt, d.h. alle Regeln und Konfigurationen wurden entfernt, aber die Geräte waren beim Server registriert. In dieser Aufgabe ging es darum das eine Gerät als Bewegungsmelder am anderen Gerät bei

Bewegung eine LED dauerhaft leuchten zu lassen und einen Alarmtext auf einem Display anzuzeigen.

In der fünften Aufgabe ging es darum eine Statusbox zu bauen. Ausgangslage war die selbe wie bei der Aufgabe zuvor. Diese Statusbox sollte drei Zustände haben: mitte, links, rechts. Dazu sollte an einer Box ein Gerät mit Beschleunigungssensor angebracht werden und für die Ausgabe des Status sollte an dem anderen Gerät ein Display sein. Wenn die Box normal da steht sollte dann auf dem Display mitteerscheinen und wenn es nach links oder rechts gedreht wird, den entsprechenden Status.

In der letzten Aufgabe sollte eine Erkennung von zwei RFID Chips vorgenommen werden, wobei wieder nur die zwei Geräte dem Server bekannt waren aber keine Konfigurationen oder Regeln vorhanden waren. Das Ergebnis sollte dann auf dem anderen Gerät auf einem Display erscheinen. Dazu wurden zwei RFID Chips mit einer Beschriftung versehen um diese unterscheiden zu können.

Die Tabelle 5.1 zeigt die persönlichen Daten der Probanden und welche Erfahrungen sie mit Technik bisher gemacht haben. Dabei wird deutlich das keiner der Probanden Erfahrungen mit Gadgeteer gemacht haben. Sie alle benutzen den Computer vorwiegend für normale Sachen wie Officeanwendunge, Computerspiele und so weiter. Drei der Probanden haben Erfahrung mit allgemeiner Programmierung wie Java oder eine ähnliche Programmiersprache und nur zwei mit Mikrocontrollern in Verbindung mit Sensoren und Aktoren. Insgesamt waren 3 von 5 Probanden in nicht technischen Berufen tätig, was der Zielgruppe entspricht.

In der Tabelle 5.2 sind die von den Probanden empfundenen Schwierigkeitsgrade der einzelnen Aufgaben von 1 wie leicht bis 5 (schwer) und die jeweils benötigte Zeit für die Lösung der einzelnen Aufgaben eingetragen. Im Durchschnitt brauchten sie für die Aufgabe 1 und 2 jeweils ca. 2 min., für Aufgabe 3 und 4 jeweils 4,5 min, für Aufgabe 5 ca. 8,5 min und für die letzte Aufgabe wieder ca. 4,5 min. Der durchschnittlich empfundene Schwierigkeitsgrad stieg von 1,2 bis Aufgabe 3 mit 1,8 Punkten an, die vierte Aufgabe wurde dann wieder mit 1,4 Punkten bewertet und die Aufgabe 6 war mit 1,6 Punkten im Schnitt auch in dieser Region. Der einzige Ausreiser war die Aufgabe 5, welche durchschnittlich mit einem Schwierigkeitsgrad von 2,8 beschrieben wurde, was aber aufgrund der Komplexität der Aufgabe zu erwarten war, da es notwendig war sich mit einem Sensor zu beschäftigen, welcher Sensorwerte für einen dreidimensionalen Raum ausgibt. Es sticht hierbei heraus, dass der Proband welcher Softwaretechnik studiert meist schneller als der Durchschnitt ist, obwohl er keinerlei Erfahrung mit Mikrocontrollern oder Sensoren hat. Insgesamt gab es keine großen Ausreiser und auch die Probanden mit keinen technischen Berufen kamen laut der vergebenen Punkte für den Schwierigkeitsgrad gut mit dem System zurecht.

In der Tabelle 5.3 wurden einige Aussagen zur Usability des Systems gemacht und die Probanden sollten diese mit Punkten von 1 bis 5 bewertet. Dabei stellt eine eins keine Zustimmung dar und 5 die volle Zustimmung. Die Aussagen sind aus vom System Usability Scale in der deutschen Übersetzung von [LS13] entnommen worden. Die Aussagen lauten wie folgt:

- Ich denke, dass ich dieses System gerne regelmäßig nutzen würde.

Proband	Alter	Beruf	Geschlecht	Erfahrung mit Technik
A	24	Rettungssanitäter	männlich	<input checked="" type="checkbox"/> Officeanwendungen <input checked="" type="checkbox"/> Computerspiele <input checked="" type="checkbox"/> Betriebssystem installieren <input checked="" type="checkbox"/> Programmierung <input type="checkbox"/> Gadgeteer <input checked="" type="checkbox"/> Mikrocontroller <input checked="" type="checkbox"/> Sensoren, Aktoren
B	25	Student Softwaretechnik	männlich	<input checked="" type="checkbox"/> Officeanwendungen <input checked="" type="checkbox"/> Computerspiele <input checked="" type="checkbox"/> Betriebssystem installieren <input checked="" type="checkbox"/> Programmierung <input type="checkbox"/> Gadgeteer <input type="checkbox"/> Mikrocontroller <input type="checkbox"/> Sensoren, Aktoren
C	25	Ingenieur	männlich	<input checked="" type="checkbox"/> Officeanwendungen <input checked="" type="checkbox"/> Computerspiele <input checked="" type="checkbox"/> Betriebssystem installieren <input checked="" type="checkbox"/> Programmierung <input type="checkbox"/> Gadgeteer <input checked="" type="checkbox"/> Mikrocontroller <input checked="" type="checkbox"/> Sensoren, Aktoren
D	25	Student Politik	männlich	<input checked="" type="checkbox"/> Officeanwendungen <input checked="" type="checkbox"/> Computerspiele <input checked="" type="checkbox"/> Betriebssystem installieren <input type="checkbox"/> Programmierung <input type="checkbox"/> Gadgeteer <input type="checkbox"/> Mikrocontroller <input type="checkbox"/> Sensoren, Aktoren
E	22	Student Wirtschaftsrecht	männlich	<input checked="" type="checkbox"/> Officeanwendungen <input checked="" type="checkbox"/> Computerspiele <input checked="" type="checkbox"/> Betriebssystem installieren <input type="checkbox"/> Programmierung <input type="checkbox"/> Gadgeteer <input type="checkbox"/> Mikrocontroller <input type="checkbox"/> Sensoren, Aktoren

Tabelle 5.1: Persönliche Daten der Probanden

5 Evaluation

Proband	Aufgabe 1	Aufgabe 2	Aufgabe 3	Aufgabe 4	Aufgabe 5	Aufgabe 6
A	1 (2:40min)	2 (2:30min)	2 (5:43min)	1 (5:16min)	3 (10:57min)	1 (5:47min)
B	2 (1:18min)	2 (2:13min)	1 (3:53min)	1 (4:45min)	2 (6:28min)	1 (3:17min)
C	1 (2:35min)	1 (2:13min)	2 (4:42min)	2 (4:13min)	3 (7:23min)	3 (5:42min)
D	1 (1:50min)	1 (1:52min)	2 (4:35min)	2 (5:13min)	3 (7:30min)	2 (4:22min)
E	1 (1:48min)	1 (1:55min)	2 (3:38min)	1 (4:07min)	3 (9:56min)	1 (4:01min)

Tabelle 5.2: Empfundener Schwierigkeitsgrad von 1 (leicht) bis 5 (schwer) und der Dauer für die Lösung der Aufgabe in den Klammern.

Proband	Frage 1	Frage 2	Frage 3	Frage 4	Frage 5	Frage 6	Frage 7	Frage 8	Frage 9	Frage 10
A	4	1	5	1	3	1	5	1	4	1
B	4	2	4	1	4	2	5	2	5	2
C	1	1	5	2	4	3	4	2	3	2
D	1	1	4	5	5	2	4	2	4	2
E	4	2	5	1	4	2	5	1	4	1

Tabelle 5.3: Fragen zur Usability, wobei 1 keine Zustimmung bedeutet und 5 volle Zustimmung.

- Ich fand das System unnötig komplex.
- Ich denke, das System war leicht zu benutzen.
- Ich denke, ich würde die Unterstützung einer fachkundigen Person benötigen, um das System benutzen zu können.
- Ich fand, die verschiedenen Funktionen des Systems waren gut integriert.
- Ich halte das System für zu inkonsistent.
- Ich glaube, dass die meisten Menschen sehr schnell lernen würden, mit dem System umzugehen.
- Ich fand das System sehr umständlich zu benutzen.
- Ich fühlte mich bei der Nutzung des Systems sehr sicher.
- Ich musste viele Dinge lernen, bevor ich mit dem System arbeiten konnte.

Dabei wurde die Usability des Systems insgesamt recht gut bewertet. Die einzelnen Probanden fanden sich nach einer Einweisung recht schnell im System zu recht. Bei den Fragen gab es lediglich zwei größere Ausreiser: Bei der ersten Frage, ob der Proband das System regelmäßig nutzen würde, antworteten zwei der Probanden mit keiner Zustimmung. Auf nachfragen meinten sie, dass sie so ein System einfach nicht brauchen würden. Desweiteren war ein Proband der Meinung, dass er die Unterstützung einer fachkundigen Person brauchen würde um das System nutzen zu können. Dabei ging es ihm vorwiegend um Unterstützung einer fachkundigen Person während der Anfangsphase beim benutzen des Systems.

Desweiteren wurde noch nach Verbesserungsvorschlägen von den Probanden gefragt. Bei den Verbesserungsvorschlägen ging es vor allem um die Verbesserung der Weboberfläche durch eine automatisierte Aktualisierung und einer besseren Integration der ConfApp in das System, da zum speichern zweimal bestätigt werden muss. Außerdem kam der Vorschlag auf den Assistenten zu verbessern, da dort zum erstellen einer Regel sehr viel geklickt werden muss. So sollte es auch möglich sein mehrere Aktoren in einer Regel zu steuern.

5.2 quantitative Evaluation

In diesem Abschnitt der Evaluation geht es um quantitative Zahlen. Hier sollen Überlegungen über die Latenz und die Anzahl möglicher Sensoren bei dem Wifi Modul oder dem xBee Modul gemacht werden.

5.2.1 Bandbreite

Das Gadgeteer Mainboard verwendet das Wifi Modul RS21, welches eine Datenrate von ca. 54 Mbit/s brutto hat. Dies entspricht ungefähr einer Datenrate von 22 Mbit/s. Bei einer Größe der Nachricht von ca. 250 Byte, was in etwa dem Durchschnitt einer Nachricht ist welche in diesem System verwendet wird, kann man theoretisch ca. 4200 Nachrichten pro Sekunde versenden. Wenn man Sensoren hat, welche so eingestellt sind, dass alle 100 Millisekunden der Sensorwert dem Server mitgeteilt wird, könnte man so ca. 420 Sensoren verwenden, bevor das WLAN aufgrund zu geringer Bandbreite zusammenbricht. Das verwendete xBee hat eine Datenübertragungsrate von ca. 250 kbit/s was 31250 Bytes pro Sekunde entspricht. Dabei können 125 Nachrichten mit einer Länge von 250 Byte pro Sekunde übertragen werden. Wenn man den gleichen Sensorintervall benutzt, würde das bedeuten, dass 12 solcher Sensoren gleichzeitig benutzt werden könnten. Das ist insgesamt ein sehr unzureichender Wert, aber nicht jedes Gerät schickt die Daten seiner Sensoren mit einer so hohen Frequenz, da dies meist nicht notwendig ist.

5.2.2 Latenz

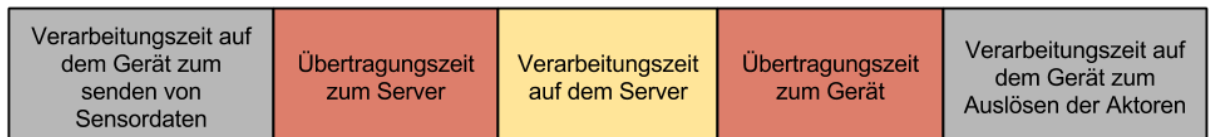


Abbildung 5.1: Die Latenz des Systems beschreibt die Zeit welche ein Gerät braucht um Informationen an den Server zu schicken und auf einem anderen Gerät einen Aktor anzusteuern. Die Latenz setzt sich dabei aus fünf Komponenten zusammen: Der Zeit welche das Gerät vom Auslesen der Sensordaten bis zum Versenden braucht, der Übertragungszeit zum Server, der Verarbeitungszeit im Server, die Übertragungszeit zurück zu ein Gerät und schließlich der Verarbeitungszeit auf dem Gerät bis zum Auslösen des Aktors.

Die Latenz des Systems stellt die Zeit dar, welche ein Sensor braucht um einen Aktor auszulösen. Dieses Zeit setzt sich aus fünf Teilen zusammen: Der Zeit welche das Gerät vom Auslesen der Sensordaten bis zum Versenden braucht, der Übertragungszeit zum Server, der Verarbeitungszeit im Server, die Übertragungszeit zurück zu einem Gerät und schließlich der Verarbeitungszeit auf dem Gerät bis zum Auslösen des Aktors. Es wurde die Latenz

berechnet für zwei Gadgeteer Module, welche beide einen Button besitzen und der Regel, das der eine Button die LED des anderen Buttons steuert. Beim Auslesen des Buttons bis zum Senden benötigte das Gadgeteer Mainboard ca. 140 ms, die Verarbeitung auf dem Server dauerte etwa 60 ms und die Verarbeitung auf dem Gadgeteer zum Auslösen der LED etwa 170 ms. Mit der Übertragungszeit, wobei mit den vorherigen Werten der Bandbreite gerechnet wird, des WLAN-Modules von 0,2 ms bzw. des xBee Modules mit ca. 8 ms, beträgt die Latzen in diesem Fall ca. 370 ms beim WLAN oder 386 ms bei xBee. Dabei fällt auf, dass die Übertragungszeit der drahtlosen Verbindung nur sehr wenig ins Gewicht fällt und es damit viel wichtiger ist, die Geräte in der Geschwindigkeit zu verbessern, da sie die meiste Zeit brauchen. Auch ist es wichtig den Server weiter zu optimieren, da bei mehr Regeln, Geräten und Sensorinformationen der Server das meiste zu tun hat, da er die zentrale Schnittstelle im System ist. Deswegen lohnt es sich später eine weitere Analyse des Servers zu machen um die Performance zu steigern.

5.3 Zusammenfassung

Insgesamt funktioniert das System und das dahinterliegende Konzept wurde schnell von den einzelnen Probanden der Usability Studie verstanden. Es gab dabei niemanden der Probleme mit dem Umgang der Plattform hatte. Es kam dabei aber auch raus, dass es notwendig ist die Oberfläche des Servers und damit die Schnittstelle zwischen System und Anwender noch benutzerfreundlicher zu gestalten. Dies reicht von der besseren Integration der ConfApp, damit nicht zweimal zum speichern bestätigt werden muss, der automatisierten Aktualisierung der Weborberfläche bis hin zur Verbesserung des Assistenten.

Auch bei der quantitativen Evaluation ist herausgekommen, dass es noch Verbesserungsbedarf hinsichtlich der Verarbeitungsgeschwindigkeit des Servers und der einzelnen Geräte gibt und das xBee nur beschränkt in einem großen System einsetzbar ist.

6 Zusammenfassung

6.1 Zusammenfassung

In dieser Diplomarbeit wurde ein Konzept für ein System entwickelt, welches es ermöglicht auf einfache Weise interaktive System und Umgebungen zu gestalten. Für den Benutzer gibt es mehrere Möglichkeiten der Erstellung von Mechanismen für die Interaktivität. Es gibt für Benutzer, welche nicht Programmieren können, einen einfachen Assistenten zum Erstellen. Desweiteren gibt es für erfahrene Anwender noch die Möglichkeit durch Programmierung mittels Programmiersprache in Textform oder mit visuellen Elementen komplexere Mechanismen zu entwickeln. Das System ist dabei zentral organisiert und so konzipiert, dass sich neue Geräte problemlos in das System integrieren lassen. Dies wird dadurch unterstützt, dass sich die Geräte von selbst am Server anmelden und ihre Daten mitteilt. Danach bekommen sie vom Server eine hinterlegte Konfiguration, welche die Zuordnung zwischen Sensoren, Aktoren und dem Gerät ermöglichen. Denn die Geräte sind so konzipiert, dass sie nicht wissen welche Sensoren bzw. Aktoren an ihnen angeschlossen sind, sondern nur die Konfiguration vom Server. Diese wird interpretiert und die einzelnen Ports initialisiert. Die Erstellung der Konfiguration für den Server geschieht dabei auf einfache Weise mit einem Assistenten, welcher per Drag'n'Drop die Zuordnung zwischen Ports und Modulen ermöglicht.

Teile dieses Konzeptes wurden implementiert um zu zeigen, dass sich verschiedenste Hardware in einem System integrieren lässt. Dazu wurden als Geräte Gadgeteer und Arduino Hardware verwendet, welche grundsätzliche, unterschiedliche Programmieransätze verfolgen. Auch wurde mit xBee und Wifi zwei unterschiedliche Kommunikationskanäle verwendet um die Interaktion der verschiedenen Arten der Kommunikation miteinander zu zeigen. Am Ende wurde eine kleine Studie gemacht, in der gezeigt wurde, dass das Prinzip des Konzeptes schnell verstanden wird. Aber auch dass es bei dem aktuellen System noch Verbesserungsbedarf, sowohl bei der Usability als auch in der Performance, gibt.

6.2 Diskussion

Ein großes Problem bei der zentralisierten Architektur ist die Skalierbarkeit des Systems mit sehr vielen Geräten. Dies kann irgendwann auf Grund der verwendeten Kommunikationsart zu Problemen führen, da die Übertragungskanäle verstopfen könnten. Dem kann entgegengesetzt werden, dass bei verwenden von mehreren unterschiedlichen Kommunikationskanälen, sich deren Übertragungsmöglichkeiten quasi addieren. Außerdem ist es möglich

mehrere Server mit separaten Kommunikationkanälen zu verwenden wenn das eine System nicht mehr ausreicht. Meistens ist es auch nicht notwendig, dass alle Geräte untereinander verbunden sind, weil sie vielleicht räumlich getrennt sind und eine Interaktion zwischen diesen Geräten keinen Nutzen bringen würde. Die Zentralisierung hat aber den Vorteil, dass dadurch Geräte, welche unterschiedliche Kommunikationen wie xBee oder Wifi benutzen, über den Server indirekt kommunizieren können.

6.3 Future Work

Am Anfang sollte der Focus der Weiterentwicklung auf der Verbesserung der Performance und der Usability liegen. Wenn diese Punkte abgehakt sind, können weitere Geräte zu dem System hinzugefügt werden wie z.B. ein Android Smartphone oder einen Webclient, welchen man auf vielen Rechnern benutzen könnte. Außerdem sollten weitere Module zur Gadgeteer Plattform hinzugefügt werden um einen größeren Umfang an Sensoren und Aktoren zu haben. Dadurch lassen sich dann auch immer neue Anwendungsmöglichkeiten für dieses System finden.

Damit man auf lange Sicht auch noch andere Geräte einbinden kann, müssen neue Kommunikationskanäle erschlossen werden. Man könnte dabei z.B. mit Bluetooth beginnen, welches in der neuen Generation sehr stromsparend ist. Denkbar wäre auch eine Anbindung an das UMTS-Netz um so auch Outdoor Anwendungen zu unterstützen.

Ein weiterer wichtiger Faktor ist die Integration von Medieninhalten für die Geräte, welche diese dann anzeigen oder abspielen können. Dies könnte mit Bildern beginnen und mit Videos und Sounds erweitert werden.

Langfristig sollte dieses System sich in eine Richtung bewegen, in der es dem unerfahrenen Anwender ermöglicht wird auch komplexere Regeln zu erstellen. Deshalb sollte sich ein Teil der Erweiterungen auf lange Sicht mit der Implementierung der schon im Konzept vorgestellten visuellen Programmierung beschäftigen. Diese sollte dann durch eine Benutzerstudie analysiert werden um die Vorteile dieses Teiles zu zeigen, bzw. ihn weiter auf die Bedürfnisse des Anwenders anzupassen.

Weiterhin sollte das Abstraktionslevel erhöht werden um das System immer weiter von der Low-Level-Programmierung zur semantischen Programmierung weiter zu entwickeln. So sollte es möglich sein ein Mapping für Sensorwerte anzulegen, welche diesen Werten Semantik vermitteln kann. Wenn z.B. ein Entfernungssensor Daten an den Server sendet, sind die Daten eine Gleitkommazahl, welche der gemessenen Spannung in Volt entspricht und haben keinerlei semantischen Wert. Wenn man nun diesem Sensor ein Mapping gibt, welches aussagt, welche Entfernung bei welcher Spannung vorliegt, dann kann man dieses semantische Wissen auch in den Regeln verwenden. Dadurch wird das System weiter in der Abstraktionsstufe gesteigert.

Ein weiteres Feature wäre, wenn man Handys in das System integrieren würde, so das auf dem jeweiligen Smartphone persönliche Regeln hinterlegt werden könnten, welche das System steuern. So könnte es möglich sein, dass eine Regel auf einem nahen Display

bestimmte Informationen oder Bilder anzeigt um die Umgebung an den Benutzer anzupassen. Oder es könnte in öffentlichen Gebäuden Displays geben, welche es ermöglichen, dass die Bilder oder die Musik der Besucher ausgestrahlt werden. So könnte jeder durch seine Individualisierung die Umgebung an seine Bedürfnisse anpassen.

Literaturverzeichnis

- [Ard13] Arduino. Arduino. <http://www.arduino.cc/>, 2013. (Zitiert auf Seite 37)
- [Dil13] P. D. R. Dillmann. Vorlesungsfolien, Robotik III, Sensoren in der Robotik. http://wwiain.ira.uka.de/Teaching/VorlesungRobotikIII/Pdf-Files/2SW_InterneSensoren.pdf, 2013. (Zitiert auf Seite 13)
- [Fou13] R. P. Foundation. Raspberry Pi. <http://www.raspberrypi.org>, 2013. (Zitiert auf Seite 36)
- [GBN10] D. Gordon, M. Beigl, M. A. Neumann. dynam: A wireless sensor network concept and platform for rapid development. In *Networked Sensing Systems (INSS), 2010 Seventh International Conference on*, S. 57–60. IEEE, 2010. (Zitiert auf Seite 15)
- [GKN⁺07] W. I. Grosky, A. Kansal, S. Nath, J. Liu, F. Zhao. Senseweb: An infrastructure for shared sensing. *Multimedia, IEEE*, 14(4):8–13, 2007. (Zitiert auf Seite 14)
- [GT09] D. Guinard, V. Trifa. Towards the web of things: Web mashups for embedded devices. In *Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences), Madrid, Spain*. 2009. (Zitiert auf Seite 15)
- [Hua04] W. Huang. Seminar Aggregierungsanfragen in Sensornetzwerken. http://www.ipd.uka.de/~ovid/Seminare/VIVSS04/Vortraege/OVID-VIVSS04_Aggregation-Sensornetzwerke_Vortrag.pdf, 2004. (Zitiert auf Seite 13)
- [Inc13] D. I. Inc. xBee. <http://www.digi.com/xbee/>, 2013. (Zitiert auf Seite 38)
- [Joy13] I. Joyent. Node.js. <http://nodejs.org/>, 2013. (Zitiert auf Seite 37)
- [JSO13a] JSON. Java Script Object Notation. <http://www.json.org/>, 2013. (Zitiert auf Seite 38)
- [JSO13b] JSON. ZigBee. <http://de.wikipedia.org/wiki/ZigBee>, 2013. (Zitiert auf Seite 38)
- [LS13] K. Lohmann, J. Schäffer. System Usability Scale (SUS) – An Improved German Translation of the Questionnaire. <http://minds.coremedia.com/2013/09/18/sus-scale-an-improved-german-translation-questionnaire/>, 2013. (Zitiert auf Seite 60)

- [Mic13a] Microsoft. Gadgeteer. <http://www.netmf.com/gadgeteer/>, 2013. (Zitiert auf Seite 37)
- [Mic13b] Microsoft. Sense Web. <http://www.sensormap.org>, 2013. (Zitiert auf Seite 14)
- [RMMH⁺09] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, et al. Scratch: programming for all. *Communications of the ACM*, 52(11):60–67, 2009. (Zitiert auf Seite 17)
- [RR13] D. Rau, B. Rau. ConfApp, 2013. (Zitiert auf Seite 54)
- [Scr13] C. Script. Coffee Script. www.coffeescript.org, 2013. (Zitiert auf Seite 38)
- [VSH11] N. Villar, J. Scott, S. Hodges. Prototyping with microsoft. net gadgeteer. In *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction*, S. 377–380. ACM, 2011. (Zitiert auf Seite 16)
- [Wik13a] Wikipedia. Arduino Plattform. <http://de.wikipedia.org/wiki/Arduino-Plattform>, 2013. (Zitiert auf Seite 37)
- [Wik13b] Wikipedia. Coffee Script. <http://de.wikipedia.org/wiki/CoffeeScript>, 2013. (Zitiert auf Seite 38)
- [Wik13c] Wikipedia. Java Script. <http://de.wikipedia.org/wiki/JavaScript>, 2013. (Zitiert auf Seite 38)
- [Wik13d] Wikipedia. Java Script Object Notation. http://de.wikipedia.org/wiki/JavaScript_Object_Notation, 2013. (Zitiert auf Seite 38)
- [Wik13e] Wikipedia. Node.js. <http://de.wikipedia.org/wiki/NodeJS>, 2013.
- [Wik13f] Wikipedia. Raspberry Pi. http://de.wikipedia.org/wiki/Raspberry_Pi, 2013. (Zitiert auf Seite 36)
- [Wik13g] Wikipedia. xBee. <http://en.wikipedia.org/wiki/XBee>, 2013.
- [Wis13a] I. Wissen. Aktor. <http://www.itwissen.info/definition/lexikon/Aktor-actuator.html>, 2013. (Zitiert auf Seite 13)
- [Wis13b] I. Wissen. Rapid Prototyping. <http://www.itwissen.info/definition/lexikon/Rapid-Prototyping-rapid-prototyping-RP.html>, 2013. (Zitiert auf Seite 14)
- [Wis13c] I. Wissen. Sensor. <http://www.itwissen.info/definition/lexikon/Sensor-sensor.html>, 2013. (Zitiert auf Seite 13)
- [Wis13d] I. Wissen. Sensor Netzwerk. <http://www.itwissen.info/definition/lexikon/Sensornetzwerk-sensor-network.html>, 2013. (Zitiert auf Seite 13)

Alle URLs wurden zuletzt am 26.11.2013 geprüft.

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Sven Plohmer)