Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung

Alexander M. Sowitzki

Umsetzung eines smarten Puppenhauses

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Alexander M. Sowitzki

Umsetzung eines smarten Puppenhauses

Ausarbeitung eingereicht im Rahmen der Arbeit zum Hauptprojekt

im Studiengang Master of Science Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Herr Prof. Dr. Kai von Luck

Eingereicht am: 31. Oktober 2018

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Problemstellung . . . 1
- 1.2 Motivation . . . 1
- 1.3 Abgrenzung . . . 2

2 Szenarien **3**
- 2.1 Aufstehen . . . 3
- 2.2 Bird Alert . . . 3
- 2.3 Ventilation . . . 4

3 Umsetzung **5**
- 3.1 Hardware . . . 5
  - 3.1.1 Grundriss . . . 5
  - 3.1.2 Rohbau . . . 6
  - 3.1.3 Baugruppen . . . 7
  - 3.1.4 Ausführungsplattform . . . 8
  - 3.1.5 Verdrahtung . . . 8
- 3.2 Software Infrastruktur . . . 9
  - 3.2.1 Bereitstellung . . . 9
  - 3.2.2 Cluster . . . 10
  - 3.2.3 Agenten . . . 11

4 Schluss **13**
- 4.1 Fazit . . . 13
- 4.2 Ausblick . . . 13

Glossar **14**

1 Einleitung

1.1 Problemstellung

Um Anforderungen für ein Cyber Physical System (CPS) zu erstellen, ist die Benennung von konkreten Szenarien und deren Simulation hilfreich. Wenn ein CPS ein komplexes verteiltes System wie ein Smart Home umfasst, ist eine Simulation im realen Raum nur umständlich möglich. Eine rein virtuelle Simulationslösung vereinfacht zwar den Versuchsaufbau, führt jedoch zusätzliche Komplexität, Fehleranfälligkeit und Implementierungsaufwand ein.

Aus diesem Grund soll ein Puppenhaus umgesetzt werden, mit welchem ein Smart Home simuliert wird. Anhand dieses Aufbaus sollen Anforderungen für eine Softwareabstraktionsschicht erstellt werden, welche die Steuerung eines verteilten Smart Homes übernimmt und dabei Schwerpunkte auf Erweiterbarkeit und Ausfallsicherheit legt.

1.2 Motivation

Für die Datenverarbeitung und -bündelung in einem Smart Environment stehen zahlreiche Programme (wie z.B. openHAB und Node-RED) zur Verfügung, welche über das Netzwerk Informationen von Internet of Things (IoT) Geräten entgegennehmen, sie nach einer vorgegebenen Logik verarbeiten und wieder über das Netzwerk weitere Geräte steuern. Für die Entwicklung der Software jener Geräte bieten diese Programme jedoch oft nur geringe Unterstützung.

Viele Sensoren und Aktoren erfordern direkte Interaktion mit Hardwareschnittstellen, also z.B. General-purpose input/output (GPIO), I2C, Serial und Serial Peripheral Interface (SPI). Der Linuxkernel bietet Treiber für jene Schnittstellen, die über sogenannte System Calls in der Programmiersprache C gesteuert werden. Es ist also eine hohe Hardwarenähe erforderlich, die jedoch zwecks Wartbarkeit nicht mit Programmkomplexität bezahlt werden darf.

Ein weiterer relevanter Punkt ist die Ausfallsicherheit. In einem Smart Home befinden sich eine Vielzahl von kritischen Systemen, durch deren Fehlfunktion ein Personen- oder Sachschaden entsteht oder die Wohnung unbewohnbar wird. Dies umfasst die Steuerung von Licht-,

Belüftungs-, Bewässerungs- und Brandmeldeanlagen, Heizungssysteme sowie Futterspendern für Tiere. Da die Entwicklung in der Heimautomatisierung zu einem großen Teil von der Maker Community getrieben wird, die die Automatisierung des eigenen Zuhauses in ihrer Freizeit vornimmt, ist der defensive Ansatz für diese Problematik, kritische Systeme nicht anzubinden. U.a. soll in diesem Projekt untersucht werden, wie trotzdem eine Anbindung dieser Systeme vorgenommen werden kann.

1.3 Abgrenzung

Wie bereits vorweggenommen, soll die zu entwickelnde Abstraktionsschicht nicht ausschließlich für die Datenaggregation in einem Smart Environment zuständig sein, sondern die Anbindung von Hardwarekomponenten vereinfachen. Hier existieren Projekte wie *Eclipse Smart Home* und *Mbed*, welche für die Implementierung verwendet werden können. Leider haben diese Projekte dem Ziel entgegenstehende Designphilosophien.

Eclipse Smart Home verwendet Java als Programmiersprache, welche sich durch eine hohe Hardwareabstraktion auszeichnet. Hardwareanbindungen müssen in C geschrieben werden und der Java Runtime verfügbar gemacht werden. Zudem sieht die Architektur vor, dass ein einzelner Dienst für die komplette Datenaggregation zuständig ist. Verteilte Datenverarbeitung durchzuführen ist nur mit der Entwicklung eigener Clienten oder durch den parallelen Betrieb von zwei getrennten Datenaggregatoren möglich.

Mbed ist, daher der Name, auf Embeddedsysteme ausgelegt, läuft aber auch auf einem vollwertigen Linux. Es ist zudem zur Anbindung an eine Cloudplattform vorgesehen und erfordert folglich eine ausfallsichere Internetverbindung, um zu funktionieren.

2 Szenarien

Mit dem Puppenhaus soll anhand von drei Szenarien überprüft werden, ob die zu entwickelnde Lösung einem realistischen Anwendungsfall genügt.

2.1 Aufstehen

Die Bewohnerin des Hauses wacht früh morgens auf und beginnt ihren Arbeitstag. Sobald sie sich im Bett aufsetzt, dimmt die Beleuchtung des Schlafzimmers und Badezimmers hoch, bis im Raum die Zielhelligkeit erreicht ist und die Tür zum Badezimmer öffnet sich. Die Bewohnerin begibt sich ins Badezimmer und beginnt die Morgenhygiene, während das System die Kaffeemaschine startet.

Sobald die Bewohnerin die Morgenhygiene abgeschlossen hat, begibt sie sich in die Küche, wobei sie andere Räume durchquert. Das Licht dimmt im aktuellen Raum voll und in anliegenden Räumen teilweise hoch. In weiter entfernten Räumen wird die Beleuchtung ausgeschaltet. Die Türen zu angrenzenden Räumen öffnen sich.

Nachdem die Bewohnerin gefrühstückt hat, begibt sie sich in den Eingangsraum. Die Tür schließt sich hinter ihr und das Garagentor öffnet sich, während die Bewohnerin sich für das Verlassen des Hauses vorbereitet. Ist sie fertig öffnet sich die Tür zur Garage schließt sich hinter ihr. Sie steigt in das Auto und fährt aus der Garage. Das System schließt daraufhin das Garagentor.

2.2 Bird Alert

Die Bewohnerin besitzt einen Nymphensittich, der sich frei im Haus bewegen kann. Dem Vogel ist der Zugang zum Arbeitszimmer, dem Atrium und dem Wohnzimmer gestattet. Fliegt er in Räume, die anliegend zu verbotenen Räumen liegen, so schließen sich die trennenden Türen. Sollte der Vogel trotzdem in verbotene Räume gelangen, so löst das System einen Alarm aus, den die Bewohnerin einsehen kann. Dabei wird unterschiedenen, ob der Vogel in einen Raum

gelangt ist, dessen Hygiene dadurch beeinträchtigt wird oder einen Raum, durch den der Vogel nach draußen gelangen kann. Erster Fall ist als Warnung zu betrachten, zweiter als kritischer Zustand.

2.3 Ventilation

Viele Neubauten sind Niedrigenergiehäuser, welche eine aktive Lüftung erfordern. In diesem Haus soll das System bestimmen können, in welchem Raum Luft ausgetauscht wird. Entschieden wird dies anhand von Luftqualitätssensoren, die in den Räumen angebracht sind. Hat ein Raum eine Temperatur, die außerhalb des Sollbereichs liegt oder ist die Luft verbraucht so wird die Belüftung angemessen verstärkt. Sollten die Messwerte stark außerhalb der Toleranzzone liegen, so ist der Benutzer anhand eines Alarms aufzufordern, den Raum zu verlassen.

Da diese Funktion kritisch ist und deren Ausfall zu Schäden an Wohnung und Menschen führen kann, muss sie gegen Ausfall abgesichert sein. Durch Redundanz soll es möglich sein, einen Teil des Systems außer Betrieb zu nehmen, ohne die Funktionsfähigkeit signifikant zu beeinträchtigen.

3 Umsetzung

In diesem Kapitel wird auf die Umsetzung des Puppenhauses eingegangen. Dies Umfasst den Hardware- und Softwareteil der Arbeit.

3.1 Hardware

Unter Hardware ist der Rohbau des Puppenhauses selbst, die verwendete Elektronik und die Ausführungsplattformen zusammengefasst.

3.1.1 Grundriss

Der Grundriss des Puppenhauses wurde von Grund auf konzipiert und orientiert sich an Anforderungen eines fiktiven Zweipersonenhaushalts mit Haustieren. Die in echten Häusern bestehenden architektonische Einschränkungen wurden aufgrund des Anwendungsszenarios nicht beachtet.

Der entworfene Grundriss (Siehe Abbildung 3.1) beinhaltet ein zentrales Atrium, welches an alle Wohnräume angrenzt und eine Topfpflanze enthält, die als Repräsentant für einen Baum fungiert. Die Wohnräume teilen sich auf in Schlaf-, Bade- und Wohnzimmer sowie Küche und Hobbyraum. Zusätzlich dazu befindet sich hinter Küche und Wohnzimmer ein Hauswirtschafts- und Eingangsraum sowie eine Garage.

Konzeptionell soll die Versorgung mit natürlichem Licht und Frischluft ausschließlich durch das Atrium statt finden. Die korrekte Ausleuchtung mit Tageslicht würde das Projekt an anderen Stellen unflexibel gestalten, weswegen dieser Aspekt hier vernachlässigt wird. Die Belüftung des Hauses soll aktiv in den Räumen Luft ableiten, sodass über eine Zuleitung im Atrium Luft nachströmen kann.

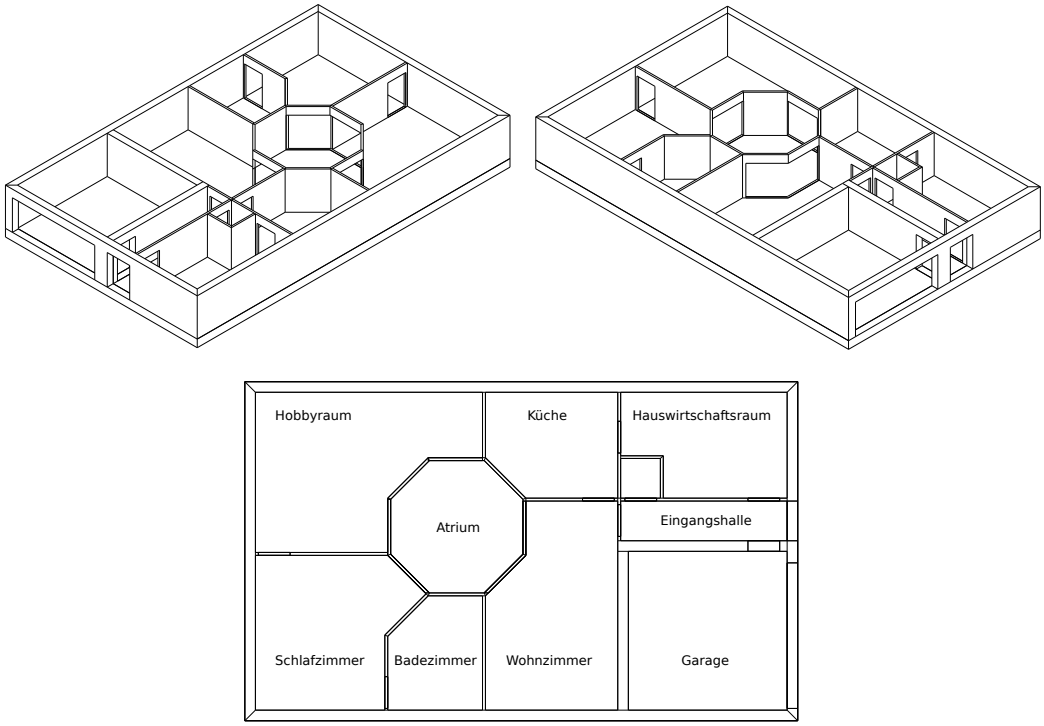

Abbildung 3.1: Grundriss des Puppenhauses

3.1.2 Rohbau

Für die Grundfläche des Puppenhauses wurde ein Seminartisch mit $80\,\mathrm{cm}$ Breite und $60\,\mathrm{cm}$ Tiefe verwendet. Durch die bereits vorhandenen Beine kann das Puppenhaus innerhalb des Raumes aufgestellt werden und ist dadurch von allen Seiten einfach zugänglich. Um den Zustand von Türen ausdrücken zu können, wurden in den Boden Löcher an Verbindungspunkten gesägt und mit einer Polysterolscheiben überdeckt, durch welche hindurch der Durchgang eingefärbt beleuchtet werden kann. Für die Wände werden Pressholzplatten aus Birke mit $10\,\mathrm{mm}$ Durchmesser verwendet. Die Aussparungen für die Türen wurden hier als kompletter Durchschnitt umgesetzt. Die Ummantlung des Atriums besteht aus zwei $2\,\mathrm{mm}$ Polysterolplatten mit einem Zwischenraum von $8\,\mathrm{mm}$.

Um Luftbewegung zu analysieren, wird die Wohnung mit einer Decke aus 2 mm Polysterol ausgestattet, welche bedarfsweise mit lichtundurchlässigem Stoff überdeckt werden kann. Die Kontaktfläche wird mit einer dünnen Schicht Montagekleber belegt, um Luftzüge zu vermeiden.

Das Computersystem wird offen auf der Unterseite des Bodens montiert. Mittels Bohrungen wird benötigte Verkabelung in die Räume eingeführt. In fünf der Räume wird zusätzlich eine Aussparung in die Wände gesägt, durch welche jeweils ein Lüfter Luft absaugen kann.

3.1.3 Baugruppen

Für die Funktionalität des Puppenhauses werden mehrere elektronische Baugruppen verwendet, die bis auf die SK6812 LEDs über I2C angesteuert werden. Jede Baugruppe verfügt über eine eigene Kontrollgruppe, welche Kommandos ausführen kann, die von einem Controller gesendet werden.

BME680 Ein Umweltsensor zum Erfassen von Temperatur, Feuchtigkeit, Qualität und Druck der Umgebungsluft. Der verwendete Chip ist selbstkalibrierend, weswegen Messwerte hinreichend präzise sind. Pro Raum im Puppenhaus wird ein Sensor verwendet, aus wessen Daten die benötigte Luftdurchwälzung pro Raum ermittelt wird.

SK6812 Dieses Bauteil beinhaltet eine RGBW LED, welche durch ein 1-Wire Protokoll in Reihe angesteuert werden kann. Dies erspart komplexe Verdrahtung Softwarebauten. Die Länge einer solchen Reihe ist in Theorie auf 1024 Einheiten begrenzt. Diese Pixel werden verwendet, um das Puppenhaus von innen zu beleuchten. Alle Pixel werden dabei in einer Reihe angesteuert. Während der separate weiße Farbkanal zur Erhellung genutzt werden kann, können die RGB Kanäle die Räume thematisch einfärben.

TSL2561 Ein Helligkeitssensor, der sich durch einen breiten Messbereich auszeichnet. Er wird verwendet um die aktuelle Helligkeit innerhalb und außerhalb des Hauses zu ermitteln, um die benötigte Helligkeit für die Raumbeleuchtung zu errechnen und diese entsprechend einzustellen.

PCA9685 Ein Pulsweitenmodulation (PWM) Treiber, welcher die Stromzufuhr zu den Lüftern reguliert. Die GPIOs von System-on-a-Chip (SoC) Platinen verfügen in der Regel über keine PWM Hardwareunterstützung, um Abnehmer mit reduzierter Leistung zu betreiben. Die Baugruppe kann Ausgänge mit maximal 1.6 kHz unabhängig vom SoC ansteuern. Zwischen

diesem Bauteil und den Lüftern befindet zudem eine H Bridge, welche das Steuersignal auf den Laststrom des Motors übersetzt.

Figurensockel Um die Szenarien zu erfüllen ist es nötig, festlegen zu können, in welchem Raum sich bestimmte Personen befinden. Die Personen sollen in diesem Projekt als Figuren im Puppenhaus dargestellt werden. Zur Umsetzung bieten sich verschiedene Möglichkeiten an. So könnte ein funkbasiertes Ortungssystem eingesetzt werden, welches jedoch, sowohl vom Arbeitsaufwand als auch von den Kosten her, den Umfang dieser Arbeit sprengen würde. Für dieses Projekt wird daher eine effizientere Lösung verwendet: Vierpolige Sockel werden in den Räumen des Puppenhauses im Boden angebracht. Jede Figur enthält an der Unterseite einen Stecker für jenen Sockel. An zwei Adern des Steckers werden je Masse und Logikspannung der Embeddedsysteme angelegt, die übrigen zwei Adern sind digitale Eingänge. Innerhalb der Stecker werden die Adern der Eingänge mit Masse oder Logikspannung fest verdrahtet. Durch das Einlesen der Eingänge der Sockel kann eindeutig ermittelt werden, welche der drei Figuren sich in einem Sockel befindet. Durch den Einsatz von Pullup Widerständen kann zudem ermittelt werden, ob kein Stecker verbunden ist.

3.1.4 Ausführungsplattform

Die virtuellen Komponenten des Aufbaus sollen verteilt auf mehreren SoCs laufen, um Ausfallsicherheit und angemessene Performance herzustellen. Dazu werden insgesamt 5 SoCs in einem Netzwerk zusammengeschaltet. Ein System wird einem Raum zugeordnet und unter diesem montiert. Es übernimmt dabei präferiert Aufgaben dieses Raumes, kann aber auch bei Ausfällen für weitere Räume einspringen. Dazu ist ein entsprechendes Design der Verdrahtung erforderlich.

Für dieses Projekt werden als SoCs Raspberry Pis der Version 2 verwendet, da diese die erforderlichen Anforderungen erfüllten und dem Preisrahmen entsprechen.

3.1.5 Verdrahtung

Sämtliche Komponenten des Puppenhauses benötigen eine Spannungsquelle von $5\,\mathrm{V}$. Diese wird von einem Step Down Converter mit $40\,\mathrm{A}$ Kapazität bereitgestellt.

Die Komponenten BME680, TSL2561 und PCA9685 werden über das Busprotokoll I2C angesprochen, welches ein sog. Multi-Master Setup, also mehrere aktive Kommunikationsteilnehmer, unterstützt. Dies bedeutet, dass mehrere Plattformen sich den Zugriff auf Baugruppen über eine

Leitung teilen können. Dies ermöglicht die direkte Verdrahtung von zwei Systemen mit den Baugruppen. Zusätzliche Elektronik würde hier gegen Hardwareversagen und Blockade der Baugruppe rüsten. Aufgrund der Adressbeschränkung der Baugruppen und zwecks Erhöhung der Ausfallsicherheit werden die Baugruppen auf mehrere Busse aufgeteilt, welche von je zwei Systemen angesprochen werden können.

Der SK6812 wird über ein properitäres Protokoll angesprochen, welches mit SPI übertragen wird. Um hier mehreren Systemen Zugriff auf die Baugruppe zu geben, muss das SPI System auf inaktiven Systemen deaktiviert werden, da die SPI Spezifikation keinen Multi-Master Betrieb vorsieht. [1]

3.2 Software Infrastruktur

Dieser Abschnitt handelt von der eigentlichen Softwareinfrastruktur, die für die Arbeit umgesetzt wurde. Dies Umfasst die Bereitstellung, den Aufbau des Clusters und die Verbindung von Agenten, sowie die eigentliche Systeminfrastruktur, die für den reibungslosen Softwarebetrieb aufgestellt wurde.

3.2.1 Bereitstellung

Die Anzahl der zu betreuenden Betriebssysteme ist mit fünf gering gehalten, nichtsdestotrotz ist der Aufwand für die Konfiguration und Wartung der Systeme signifikant. Softwareupdates müssen regelmäßig durchgeführt sowie eine Sammlung von Konfigurationsdateien und Verschlüsselungszertifikaten auf alle Systeme verteilt werden. Grundsätzlich lässt sich der Bereitstellungsprozess in drei Schritte unterteilen:

Provisioning Die Installation und Konfiguration eines Betriebssystems wird in der Provisioningphase übernommen. Dies bedeutet das Aufspielen eines Dateisystems und initiale Konfiguration eines Configuration Management Dienstes. Über diesen werden für das Zielsystem benötigte Bibliotheken installiert und die Hardwareschnittstellen vorbereitet. Im konkreten Anwendungsfall beinhaltet dies den Einsatz von Docker und die Zusammenführung der einzelnen Knoten in einen Schwarm. Hinzu kommt die Installation einer Python Laufzeitumgebung für die hardwarenahen Agenten.

Deployment Die Auslieferung der eigentlichen Nutzlast, also dem Quellcode der Agenten, muss ebenfalls konsistent über alle Systeme hinweg erfolgen. Zudem müssen die Prozesse der Agenten im Betriebssystem als Dienste konfiguriert und bei Quellcodeänderungen neu gestartet werden.

Für die Durchführung dieser Aufgaben wird Salt verwendet. Diese Software erfordert, dass der Zielzustand eines Systems definiert wird. Der Dienst selber kümmert sich um die Ermittlung und Umsetzung der nötigen Schritte, um das System in den Zielzustand zu überführen. Das Agentenframework kommuniziert mittels der Transportverschlüsselung Transport Layer Security (TLS) und authentifiziert sowohl Server als auch Clienten über Zertifikate. Dafür ist eine Certificate Authority (CA) nötig, welche den Teilnehmern dieses Netzwerkes Vertrauen ausspricht. Die Verwaltung dieser CA kann vollständig durch Salt übernommen werden. Entsprechend konfigurierte Knoten erhalten über einen sicheren Transportweg Zerifikate, die von den dort eingesetzten Agenten direkt verwendet werden können.

3.2.2 Cluster

Wie in Abschnitt 3.1.4 aufgezeigt werden fünf SoC zusammengeschlossen. Diese werden mit entsprechender Software zu einem Schwarm gebündelt. In vorhergegangen Arbeiten wurde Docker als angebrachte Virtualisierungslösung ermittelt [2][3]. Zur Clusterisierung könnte wie vorhergegangen Kubernetes verwendet werden. Für diesen Anwendungsfall ist jedoch eine höhere Ausfallsicherheit von Nöten, weswegen eine weniger überladene Lösung erforderlich ist. Deshalb wird für dieses Projekt Docker Swarm verwendet.

Der Schwarm besteht aus fünf Knoten, die allesamt als Manager fungieren, also in Kooperation die einzelnen Dienste koordinieren.

Vor dem Beginn der detailierten Projektumsetzung sind zwei Container einzurichten, die die Grundfunktionalität des Systems ermöglichen. Zum einen ist dies ein Messagebroker, über welchen alle Agenten miteinander kommunizieren. Dies kann als einzelner Container oder eine Sammlung von geclusterten Containern sein. Zum anderen ist ein Agent mit hoher Ausfallsicherheit nötig, der den Zustand des Gesamtsystems überwacht und anhand der Situation das System konfiguriert.

3.2.3 Agenten

Innerhalb des Multiagentensystems werden verschiedene Funktionsbereiche umgesetzt, welche zu einer umfangreichen Anzahl von Agenten führen. In diesem Dokument soll deshalb auf zwei Bereiche eingegangen werden: Die Ansteuerung der Beleuchtung und die Ventilationsregelung.

Die Beleuchtungsstimmung und -intensität der Räume des Puppenhauses wird unabhängig voneinander vorgenommen. Relevante Eingangsdaten dazu sind, welche Bewohner des Hauses sich im Raum befinden, welche Helligkeit dort herrscht und welches Farbthema gewünscht ist. Für eine Übersicht über beteiligte Agenten und die resultierenden Datenströme siehe Abbildung 3.2. In der Abbildung wird dabei, der Übersichtlichkeit halber, nur auf Agenten eingegangen, die für die Steuerung des Hobbyraums nötig sind.

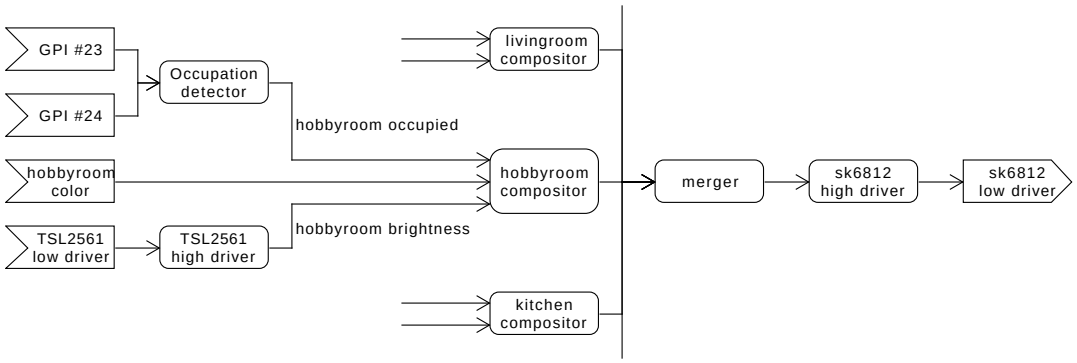

Abbildung 3.2: Verwendete Agenten zur Beleuchtungsregelung

Die Steuerung der Ventilationssysteme wird wie bei der Beleuchtung von mehreren Agenten in Kooperation erledigt. Siehe dazu Abbildung 3.3. Auch hier sind nur für den Hobbyraum relevante Agenten eingezeichnet.

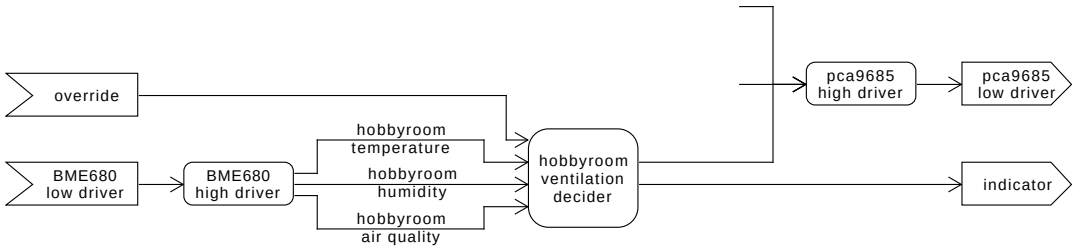

Abbildung 3.3: Verwendete Agenten zur Ventilationsregelung

4 Schluss

4.1 Fazit

Die in der Einleitung aufgestellten Szenarien können mit der erstellten Hardwarelösung umgesetzt werden. Selbst komplexere Beleuchtungsszenen sind mit dem verwendeten System umsetzbar und durch die Belüftungsanlage kann das komplexe Zusammenwirken verschiedener Komponenten in einem Echtzeitsystem getestet werden. Die Umsetzung eines vollständigen Cluster- und Deploymentsystems führt dazu, dass das Puppenhaus auch in Zukunft auf einfache Weise Updates erfahren kann.

4.2 Ausblick

Dank dieses Projektes kann die erwähnte Abstraktionsschicht in folgenden Arbeiten vollständiges implementiert und schon während der Entwicklung anhand dieser Testumgebung überprüft und angepasst werden. Auf Basis der aufgestellten Anwendungsszenarien kann zudem ermittelt werden, ob weitere Anpassungen nötig sind. Nach der Fertigstellung der Abstraktionsschicht kann das Puppenhaus für andere Projekte der HAW Hamburg weiterverwendet werden.

Glossar

Agent "Ein Agent ist ein Computersystem, das sich in einer bestimmten Umgebung befindet und welches fähig ist, eigenständige Aktionen in dieser Umgebung durchzuführen, um seine (vorgegebenen) Ziele zu erreichen."[4]. 9

C Eine ältere, weit verbreite prozedurale Programmiersprache. 2

CA Ein Stammzertifikat, welches bei Verschlüsselungsmechanismen anderen Zertifikaten Vertrauen zuschreibt. 9

CPS *Cyber Physical System*, die Kombination von Geräten mit Elektronische Datenverarbeitung (EDV) Komponenten und der Zusammenführung dieser in ein Netzwerk. 1

Eclipse Smart Home Ein Framework für die Anbindung von Smart Home Komponenten verschiedener Hersteller[5]. 2

GPIO Durch einen Prozessor auf einem SoC direkt ansprechbare Eingangs- und Ausgangsleitungen. 1, 7

I2C Ein Busprotokoll für die Ansteuerung von Teilkomponenten innerhalb einer Hardwarekomponente. 1, 6, 7

IoT *Internet of Things*, die Vernetzung von Geräten zum Datenaustausch. 1

Java Eine weit verbreitete objektorientierte Hochprogrammiersprache. 2

Linux Ein universell einsetzbares Betriebssystem mit freien Lizenzbestimmungen. 2

Mbed Ein Betriebssystem für eingebettete ARM-Prozessor Architekturen mit starkem Anwendungsbezug auf das IoT[6]. 2

. 7

Node-RED Ein webbasiertes Programm zur Verarbeitung von Daten. 1

openHAB Ein webbasiertes Programm zur Verarbeitung von Daten. 1

. 7

PWM Ein Verfahren mit welchem digitale Systeme analoge Signale emulieren können. 7

Salt Ein Configuration Management Tool[7]. 9

Serial Ein Kommunikationsprotokoll für den Austausch von Daten zwischen zwei Teilnehmern. 1

Smart Environment Eine Umgebung mit vernetzten automatisierbaren Geräten und Computern. 1, 2

Smart Home Ein Haushalt mit Smart Environment Fähigkeiten. 1

SoC Eine Integrierte Schaltung, die ein komplettes Computersystem und andere Schaltungen beinhaltet. 7, 9

SPI Ein Busprotokoll für die Ansteuerung von Teilkomponenten innerhalb einer Hardwarekomponente. 1, 8

Step Down Converter Ein Spannungswandler, der die Eingangsspannung um einen spezifischen Wert reduziert. 7

System Call Der Aufruf von Betriebssystemfunktionalitäten durch normale Programme mittels Funktionsaufruf. 1

TLS Eine Transportverschlüsselung für Netzwerke. 9

Literatur

[1] *KeyStone Architecture: Serial Peripheral Interface (SPI)*. Englisch. URL: http://www.ti.com/lit/ug/sprugp2a/sprugp2a.pdf (besucht am 18. 10. 2018).

[2] *Eine Ausführungsplattform für Cyber Physical Systems*. URL: https://dev.eqrx.net/gpj.pdf (besucht am 19. 09. 2017).

[3] *Dokumentation zu TTI*. URL: https://dev.eqrx.net/haw-tti/ (besucht am 19. 09. 2017).

[4] *VDE/VDI 2653. Agentensysteme in der Automatisierungstechnik*. Juni 2010.

[5] *A flexible Framework for the Smart Home*. Englisch. URL: https://www.eclipse.org/smarthome (besucht am 18. 10. 2018).

[6] *arm mbed*. Englisch. URL: https://www.mbed.com/en/ (besucht am 18. 10. 2018).

[7] *Intelligent, event-driven IT automation*. Englisch. URL: https://www.saltstack.com (besucht am 18. 10. 2018).