Duale Hochschule Baden-Württemberg
Mannheim

Studiengang Wirtschaftsinformatik

Studienrichtung Data Science

Seminararbeit zum Thema:

Smart Home - Erste Schritte zu skalierenden KI-gesteuerten Automatisierungen

Verfasser (Matrikelnummer)	Peter Behrens (8747156)
	Stephan Lenert (9347857)
	Nina Mergelsberg (6843257)
	Tom Müller (7434133)
Kurs	WWI 18 DSB
Semester der Veranstaltung	5. Semester
Veranstaltung	Integrationsseminar
Modul	Integrationsseminar zu ausgewählten Aspekten der Wirtschaftsinformatik
Dozent der Veranstaltung	Prof. Dr. Bernhard Drabant
Betreuende Dozentin	Viviane Schmidt
Abgabedatum	25.01.2021

Metadokument

Das Projekt zur Veranstaltung *Integrationsseminar* mit der vorliegenden Seminararbeit mit dem Titel „Smart Home - Erste Schritte zu skalierenden KI-gesteuerten Automatisierungen“ wurde in folgenden Teilen von den Studenten bearbeitet:

Peter Behrens (8747156)	- Erarbeitung der Kapitel 3 & 5 - Erstellung der Präsentation
Stephan Lenert (9347857)	- Implementierung des DIY-Kits - Erarbeitung des Kapitels 6
Nina Mergelsberg (6843257)	- Erarbeitung der Kapitel 1,2 & 7 - Layout der Seminararbeit
Tom Müller (7434133)	- Bearbeitung des Kapitels 4 - Erstellung der Präsentation

Inhaltsverzeichnis

Abkürzungsverzeichnis

CPU	central processing unit
DIY	Do it yourself
HDFS	Hadoop Distributed File System
HTML	Hypertext Markup Language
IoT	Internet of Things
IT	Informationstechnologie
JDBC	Java Database Connectivity
JVM	Java Virtual Machine
KI	Künstliche Intelligenz
KNX	Konnex
LED	Lichtemittierende Diode
MQTT	Message Queuing Telemetry Transport
openHAB	open Home Automation Bus
OSGi	Open Services Gateway initiative
QoS	Quality of Service
REST-API	Representational State Transfer - Application Programming Interface
RFID	Radio Frequency Identification
UI	User Interface

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die jüngsten Fortschritte und Weiterentwicklungen in der Technologie führen zu einem allgemeinen Trend, ein Teil des menschlichen Lebens zu werden und das Leben der Benutzer von Grund auf zu verändern. Das Internet der Dinge, was die Vernetzung von Alltagsgegenständen über das Internet darstellt, wird derzeit immer populärer und verbreitet sich zunehmend. Es bietet die Möglichkeit, den Anwender in vielen Bereichen des Alltags zu unterstützen und den Lebensmittelpunkt von vielen Menschen – das eigene Zuhause – effizienter zu gestalten. Dabei steigen die Ansprüche der Verbraucher kontinuierlich. Neben einer erhöhten Wohnqualität sind auch die Maßstäbe für Sicherheit, Komfort und Energiemanagement neu gesetzt worden. Eine Reaktion darauf ist die Entwicklung von Smart Home Systemen.

Um den Ansprüchen der Menschen gerecht zu werden, ist eine individuelle Gestaltung des Smart Home Systems zu ermöglichen. Jeder Anwender hat andere Ansprüche, was er unter einem smarten, vernetzten Zuhause versteht. Deswegen ist es unabdingbar, dass die aktuellen Innovationen so konzipiert werden, dass sie an die individuellen Bedürfnisse am besten angepasst werden können.

Das Ziel der vorliegenden Arbeit ist es, ein System zu entwickeln auf dem eine skalierte KI-gesteuerte Hausautomatisierung aufbaut werden kann. Es soll demzufolge ein „Do it yourself“-Kit entwickelt werden, das dem Benutzer einen Einstieg in die selbst erstellte Smart Home Automatisierung ermöglicht, die er nach Belieben erweitern kann. Die Ergebnisse der Testimplementierung haben gezeigt, dass es problemlos für die Smart-Home-Automation eingesetzt und individuell durch weitere Komponenten erweitert werden kann.

Im Anschluss an die Einleitung werden in Kapitel 2 und 3 die theoretischen Grundlagen beschrieben, um ein Verständnis für die Problematik aufzubauen. Dabei werden die Begriffe Internet der Dinge und Smart Home erläutert und anschließend ein Überblick über die Softwarelösung „open Home Automation Bus“ (openHAB). Das sich anschließende Kapitel 4 beschreibt die technische Erläuterung von IoT Geräten. Der theoretische Teil schließt mit der Beschreibung des Message Queuing Telemetry Transport (MQTT)-Protokolls, auf dem die Maschine zu Maschine Kommunikation dieses Projekts aufbaut.

Basierend auf diesen theoretischen Aspekten der Arbeit wird die Implementierung des DIY-Kits vorgenommen, dessen praktische Umsetzung, mit allen Erklärungen zur Erstellung der Bestandteile, in einer Präsentation zur Verfügung gestellt wird.

Auf dieser praktischen Implementierung aufbauend, wird in Kapitel 6 eine Zukunftsvision beschrieben, welche eine voll funktionierende und selbstlernende Haushalts-KI umfasst. Außerdem wird in diesem Kapitel erläutert wie zwei entsprechende Geschäftsmodelle aussehen könnten.

Abschließend werden die Erkenntnisse dieser Seminararbeit kritisch reflektiert, die wesentlichen Ergebnisse aufgegriffen und ein zukunftsweisender Ausblick gegeben.

2 Theoretische Grundlagen

In diesem Kapitel wird auf die theoretischen Grundlagen eingegangen, die zum Verständnis späterer Ausführungen notwendig sind. Diese beinhalten Theoriewissen über die Grundlagen des Terminus Internet der Dinge und Smart Home, damit sie im Kontext dieser Arbeit richtig eingeordnet werden können.

2.1 Das Internet der Dinge

> "If you think that the internet has changed your life, think again. The Internet of Things is about to change it all over again!" - Brendan O'Brien, 2014

Doch was steckt hinter dem Begriff „Internet of Things"? Erstmals erwähnt wurde der Begriff von Ashton (2009), der im Jahr 1999 im Zuge seiner Präsentation zu Radio Frequency Identification (RFID) bei Procter & Gamble vom Internet of Things sprach. Im deutschen Sprachgebrauch, vor allem in der Industrie und der Wirtschaft, ist der Begriff Internet der Dinge (engl. Internet of Things, kurz IoT) gebräuchlich und „beschreibt die Verbindung eindeutig identifizierbarer physischer ‚Dinge' bzw. Objekte mit dem Internet oder einer anderen vergleichbaren virtuellen Struktur […]" (Grohmann et al., 2017, S.5). Durch diese einzigartige Verbindung sind die „Dinge" und Objekte – wie z.B. RFID-Tags, Sensoren, Aktoren, intelligente Geräte wie beispielsweise Smartphones, Heizungsanlagen etc. – in der Lage, miteinander zu kommunizieren und zu interagieren, um gemeinsame Ziele erreichen oder auftretende Probleme bewältigen zu können.

Folglich stehen beim Internet der Dinge die menschlichen Interaktionen mit den Geräten im Vordergrund, wobei die Geräte mit dem Benutzer interagieren und diesen beim Eintreten von bestimmten Situationen informieren oder sogar warnen können. Dabei steuert und überwacht der Benutzer die Geräte aus der Ferne (Luber, 2017).

Durch die Vernetzung sämtlicher Gegenstände und die damit einhergehende Steigerung der Anzahl der angeschlossenen IoT-Geräte wird es nach der Schätzung und Prognose von IHS Markit bis 2025 mehr als 75,44 Milliarden Geräte geben, die mit dem Internet verbunden sein werden. Das Fünffache im Vergleich zu vor zehn Jahren.

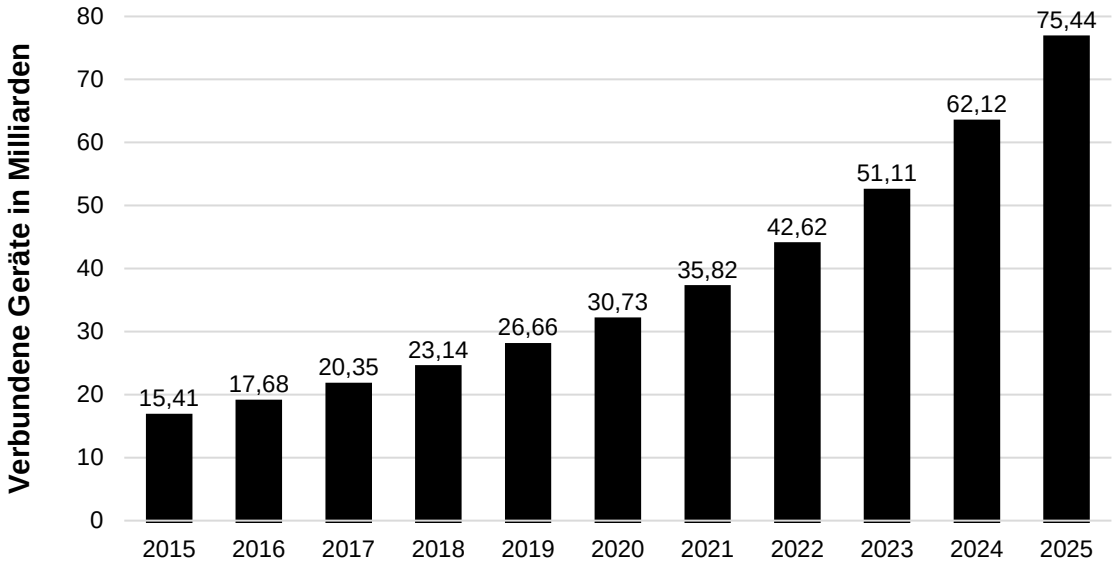

Abbildung 1: Anzahl der vernetzten Geräte im Internet der Dinge (IoT) weltweit in den Jahren 2015 bis 2025 (in Milliarden) (Quelle: Lucero, 2016)

2.2 Was ist Smart Home

Ein Anwendungsgebiet des Internet der Dinge stellt der Gebäudebereich dar, der in den letzten Jahren vielen Fortschritten und Weiterentwicklungen unterzogen ist. Hierbei wird das Internet der Dinge dafür benutzt, intelligente Wohnhäuser zu schaffen, die durch eine Gebäudeautomation realisiert werden. Smart Home bezeichnet dabei im Wesentlichen den Einsatz von technischen Systemen, automatisierten Abläufen und vernetzten, ferngesteuerten Geräten in Wohnungen und Häusern.

Strese et al. (2010) definierte Smart Home wie folgt:

„Das Smart Home ist ein privat genutztes Heim (zum Beispiel Eigenheim, Mietwohnung), in dem die zahlreichen Geräte der Hausautomation (wie Heizung, Beleuchtung, Belüftung), Haushaltstechnik (wie zum Beispiel Kühlschrank, Waschmaschine), Konsumelektronik und Kommunikationseinrichtungen zu intelligenten Gegenständen werden, die sich an den Bedürfnissen der Bewohner orientieren. Durch Vernetzung dieser Gegenstände untereinander können neue Assistenzfunktionen und Dienste zum Nutzen des Bewohners bereitgestellt werden und einen Mehrwert generieren, der über den einzelnen Nutzen der im Haus vorhandenen Anwendungen hinausgeht.“ (Strese et al., 2010, S. 8).

In vielen wissenschaftlichen Fachartikeln und Bücher werden oftmals unterschiedliche Begrifflichkeiten synonym zu Smart Home verwendet. Darunter zum Beispiel Intelligentes Wohnen, Smart House, Connected Home, Home of the Future, Smart Living, Aware Home oder auch Smart Environment (Strese et al., 2010, S. 8).

2.3 Historische Entwicklung

Bereits in den 1930er Jahren wurden erstmals die Idee eines Hauses vorgestellt/dargestellt, dass automatisch und selbstständig Funktionen übernehmen und agieren kann. Der Autor George H. Bucher beschrieb in seinem Artikel „Das elektronische Haus“, der im „Popular Mechanics Magazine“ veröffentlicht wurde, ein vernetztes Haus, in dem sich Türen automatisch öffnen und sich die Beleuchtung entsprechend den Bedürfnissen der Bewohner verändern würde (Völkel, 2020).

Die erste Umsetzung von vernetzten Gebäuden wurden schließlich ab den 1960er Jahren umgesetzt. Damals wurden erstmals Störmeldesysteme in größeren Gebäudeeinheiten über die konventionelle Elektroinstallation realisiert (Völkel, 2020). Im ersten Prototyp eines vernetzten Eigenheims kam zusätzlich noch die Kontrolle von Temperatur und Luftfeuchtigkeit sowie eine Begrenzung der TV-Zeit der Kinder hinzu. Ab diesem Zeitpunkt bekam die neue Technik einen Anschub. Zwar zunächst als Spielerei für Bastler und vermögende Personen etabliert, die sich mit der Automatisierung und Fernsteuerung einfacher Schaltvorgänge und smarten Sicherheits- und Feuerschutz-Maßnahmen beschäftigten, verhalf die Entwicklung der Mikroelektronik ab den 19990er Jahren Smart Home zum Durchbruch (Gietz, 2020).

Ein herstellerunabhängiger Standard für Häuser, der offene KNX-Standard, wurde im Jahr 2002 verabschiedet. Dabei handelt es sich um eine gewerkeübergreifende Vernetzung von Heizung, Kühlung, Lüftung, Verschattung und Beleuchtung. Geräte verschiedener Anbieter können dadurch miteinander kommunizieren (Völkel 2016).

Die Entwicklung von drahtlosen Netzwerken und Smartphones machen eine aufwendige und teure Verkabelung überflüssig und ebnen den Weg für den herkömmlichen Einsatz der Smart Home Technologien. Bereits im Jahr 2015 sind mehr als eine Million Gebäude in Europa KNX-vernetzt. Der Markt wächst und hält mehr als 7000 Produkte für die Vernetzung von Gebäuden bereit. Von LED-Lampen, die sich per App steuern lassen, über smarte Lautsprecher mit intelligenter

Sprachsteuerung bis hin zur vernetzten Jalousien- und Heizungsanlage. Laut einer Bitkom Studie nutzen fast 4 von 10 Verbrauchern in Deutschland bereits Smart Home-Anwendungen und die Nutzung soll in den nächsten Jahren noch weiter ansteigen (Paulsen & Klöß, 2020). Die grenzenlosen technischen Möglichkeiten bieten mittlerweile auch die Gelegenheit, ältere Häuser zu einem Smart Home um- und aufzurüsten (Völkel, 2020).

2.4 Möglichkeiten und Funktionsweise eines Smart Homes

Das Hauptziel des Einsatzes von Smart Home Systemen ist die Verbesserung der Lebensqualität und des Komforts im Haus, aber bietet den Nutzern auch mehr Möglichkeiten, die Häuser sicherer und energieeffizienter zu machen.

Bequemlichkeit durch Interkonnektivität

Durch Gebäudeautomation wird den Nutzern der Fernzugriff auf Systeme wie Heizungs- und Kühlsysteme, Gegensprechanlagen, Musik- und Multimediageräte im ganzen Haus ermöglicht, die bequem von unterwegs gesteuert werden können und vereinfachen damit die alltäglichen Aufgaben.

Energieeffizienz

Eine effizientere Nutzung von Energie durch vernetzte, fernsteuerbare Geräte ist ebenfalls ein Vorteil, den die Gebäudeautomation mit sich bringt. Lichter können automatisch ausgeschaltet werden, wenn sich niemand in einem Raum aufhält und das Thermostat kann so eingestellt werden, dass die Innentemperatur tagsüber sinkt, bevor sie abends auf ein angenehmeres Niveau zurückkehrt, bevor die Bewohner nach Hause kommen. All diese automatisierten Aufgaben sowie moderne, energieeffiziente Geräte sparen Strom, Wasser und Erdgas und schonen so die natürlichen Ressourcen.

Sicherheit

Die Ausstattung der Häuser durch fortschrittliche Sicherheitssysteme mit Kameras, Bewegungssensoren und gegebenenfalls mit einer Verbindung zur örtlichen Polizeistation oder einem privaten Sicherheitsunternehmen bietet Smart Home eine erhöhte Sicherheit für die Hausbewohner.

3 OpenHAB - Open Home Automation Bus

In diesem Kapitel geht es um die von Kai Kreuzer, im Jahr 2010, initiierte Softwarelösung „open Home Automation Bus“ (openHAB). Diese in Java entwickelte Anwendung ermöglicht es dem Nutzer verschiedene Komponenten der Gebäudeautomatisierung auf einer Plattform zu kombinieren. Dabei spielt es keine Rolle von welchem Anbieter die Komponente stammt, da openHAB eine hersteller- und protokollneutrale Architektur zur Verfügung stellt. Seit 2013 wird openHAB als offizielles Eclipse-Projekt unter dem Namen Eclipse Smart Home betrieben. Die Eclipse Foundation ist eine gemeinnützige Gesellschaft mit der Aufgabe, die Eclipse-Open-Source-Gemeinschaft und ihre Projekte zu leiten. Deren Ziel ist es, im Rahmen unterschiedlicher IT-Projekte eine Plattform für die Entwicklung von Software bereitzustellen. Während Architektur und Funktionen überwiegend im Eclipse Smart Home-Projekt weiterentwickelt werden, nutzen Binding-Entwickler nach wie vor die openHAB-Website für ihre Arbeit. Das Entwicklungstool legt seinen Fokus auf einen hohen Benutzerkomfort beim Einrichten und Konfigurieren seines Projekts. Das merkt man auch beim Arbeiten mit der openHAB Software. Befehle und neue Elemente lassen sich intuitiv hinzufügen, wodurch auch der Einstieg für technisch nicht versierte Benutzer erleichtert wird. Ebenfalls ist die Plattform betriebssystemunabhängig und es besteht die Möglichkeit Bindings um zusätzliche Technologien/Protokolle erweitern zu können. Dies bietet dem Anwender einen großen Freiraum bei der Einrichtung seines Smart Homes. Die Steuerung erfolgt über ein User Interface.

3.1 Persistenz

Das Auslesen von Events und Senden von Befehlen stellt immer eine Momentaufnahme dar. Das bedeutet, dass im User Interface beobachtet werden kann, wenn eine Lampe aus- bzw. eingeschaltet wird oder ein Bewegungsmelder eine Aktion registriert. Um das Smart Home jedoch effektiv zu nutzen, müssen diese Daten gespeichert werden. Deswegen ist Persistenz ein wichtiger Bestandteil von Smart Home Systemen, da es das Speichern von Statusinformationen der einzelnen Items ermöglicht. Dadurch können Verläufe visualisiert, historische Daten abgefragt oder bei einem Systemneustart der Status wiederhergestellt werden (*Persistence* | *OpenHAB*, o. J.).

3.2 Architektur

Wie bereits angeschnitten zeichnet sich openHAB durch seine hohe Konnektivität, dass bedeutet einfache Integration von Komponenten verschiedene Hersteller, aus. Um die unterschiedlichen Protokolle miteinander kompatibel zu machen, wird ein Konzept benötigt: OpenHAB muss zum einen das verwendete Protokoll verstehen und zum anderen die Daten in eine interne, abstrakte Form übersetzen, sodass herstellerspezifische Details vor dem restlichen System verborgen bleiben. Die Unterstützung möglichst vieler verschiedener Hersteller und die damit einhergehende Vielfalt an Protokollen hat dazu geführt, dass openHAB sehr modular konzipiert wurde. Das hat den Vorteil, dass nicht benötigte Technologien keinen Speicher und keine CPU-Leistung beanspruchen. Die Basisinstallation kann aber jederzeit durch Add-ons erweitert werden. Diese modulare Architektur wurde technisch mithilfe von OSGi-Plattformen umgesetzt. Die Protokolle werden innerhalb von OSGi Service Bundles, in openHAB auch Bindings genannt, implementiert (Freier & Leutenegger, 2015, S. 17ff.). Die nachfolgende Abbildung zeigt einen Überblick der Architektur:

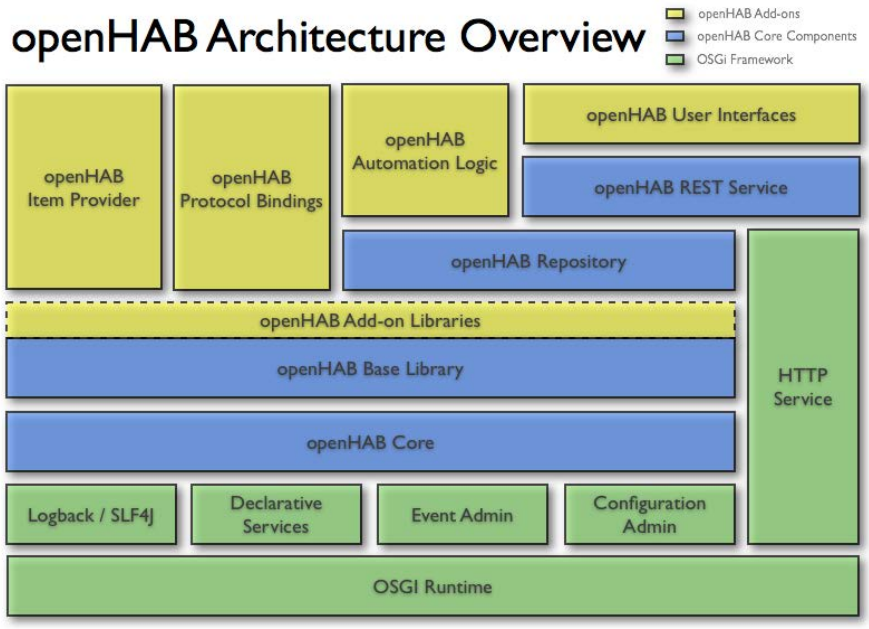

Abbildung 2: openHAB Architektur (Quelle: Kreuzer, 2014)

Das openHAB Projekt ist generell in zwei Kategorien zu unterteilen: die **openhab-runtime** und der **openhab-designer**. Während die openhab-runtime dem Serverprozess entspricht, der die Aktionen ausführt, sorgt der openhab-designer für

eine benutzerfreundliche Konfigurationsoberfläche auf Basis von Eclipse. Aufgrund fehlender Weiterentwicklung gilt der openhab-designer jedoch als veraltet und wurde von einer Erweiterung für Visual Studio Code abgelöst. Da erstere eine reine Java-Lösung ist, wird eine JVM (Java Virtual Machine) benötigt. Damit das Smart Home auch - während das System aktiv ist - erweiterbar bleibt, wird bei der Gebäudeautomation häufig OSGi eingesetzt. Diese Abkürzung steht für „Open Service Gateway initiative“ und stellt eine hardwareunabhängige dynamische Softwareplattform, auf Basis eines Komponentenmodells („Bundle“/„Service“) zur Verfügung.

Event Bus

Der für die interne Kommunikation verantwortliche Event-Bus stellt einen Basisservice von openHAB dar. Über ihn werden Events zwischen den verschiedenen OSGi Bundles versendet. Ein Event kann entweder ein Command, welcher eine Aktion ausführt oder ein Status-Update, welches Zustandsänderungen der Sensoren/Aktoren beinhaltet, sein. Alle Protokoll Bindings, die einen physischen Link zur Hardware darstellen, sollten über diesen Event Bus kommunizieren. Umgesetzt wurde der Kommunikationskanal mithilfe des OSGi Event Admins (*Event Bus* | *OpenHAB*, o. J.)

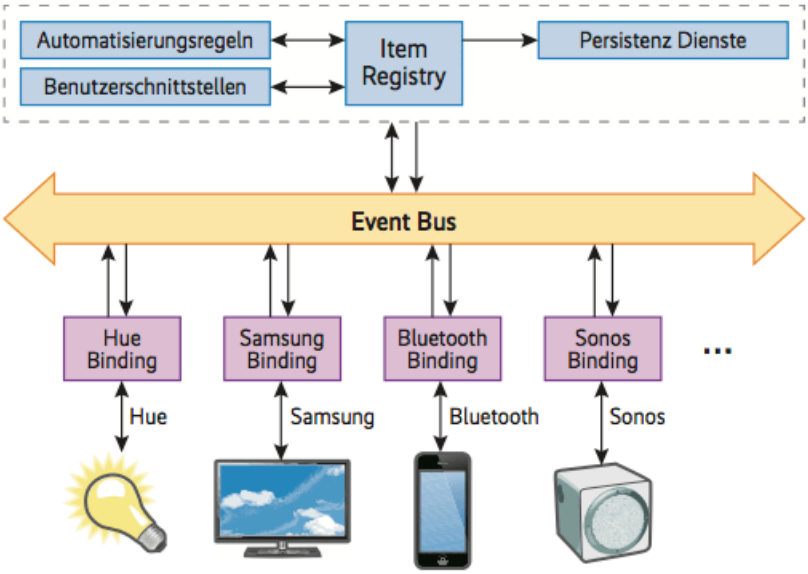

Abbildung 3: openHAB ermöglicht Integration verschiedener Technologien (Quelle: Eichstädt-Engelen et al., 2014)

Bindings

Bindings beschreiben die Verbindung zwischen openHAB und den externen Komponenten (engl. Things) und bilden somit die Grundlage der Konnektivität. Für die meisten Technologien sind bereits Bindings verfügbar, die einzeln heruntergeladen und dann als Add-on installiert werden können. Deren wesentliche Aufgabe besteht darin, sich mit dem externen Gerät zu verbinden, ständig auf den Event Bus zu horchen und die ausgetauschten Daten miteinander kompatibel zu machen. Amazon Alexa, Fritz!Box, Philips Hue, Samsung TV oder auch Twitter stehen beispielsweise bereits als offizielle Bindings zur Verfügung. Alle momentan verfügbaren Bindings sind in der openHAB Dokumentation oder auf dem zugehörigen GitHub nachzulesen (*Bindings | OpenHAB*, o. J.).

Things

Things stellen die physikalische Schicht der openHAB-Architektur dar. Aus Sicht der Konfiguration teilen die Things dem System mit, welche physischen Entitäten (Geräte, Webdienste, etc.) vom System verwaltet werden. Um ein Thing an openHAB anzubinden, muss zuerst das zugehörige Binding, dass den passenden Typ bereitstellt, identifiziert werden. Wenn das Smart Home Projekt also erstmals durch eine Philips Hue Lampe erweitert werden soll, muss zunächst das entsprechende Philips Hue Binding installiert werden. Jedes Thing bietet einen oder mehrere Channel, zur Verknüpfung mit Items und für den Zugriff auf die entsprechenden Funktionalitäten (*Things | OpenHAB*, o. J.).

Items

Um die verschiedenen Technologien auf einer Plattform nutzbar zu machen, bedarf es einer einheitlichen internen Repräsentation. Um die Integration umzusetzen, wurden Items als zentrale Entität im openHAB Domainmodell, eingeführt. Ein Item besteht aus:

- Typ*
- Name*
- Formatierung
- Icon

- Gruppe
- Bindingparameter

Die mit Sternchen markierten Eigenschaften Typ und Name sind obligatorisch, während die anderen Eigenschaften optionale Informationen beinhalten. Als Typ kann beispielsweise „Switch“, zum An- und Ausschalten von Geräten, oder „Dimmer“, zum Dimmen und Schalten von Lampen, verwendet werden (*Items* | *OpenHAB*, o. J.).

Item Repository

Das Item Repository verfolgt den aktuellen Status der Items und ist mit dem Event Bus verbunden. Es ist wichtig, dass der Zustand eines Items, auch über die Ausführungsdauer einer openHAB Instanz (bspw. Neustart) hinaus, abfragbar bleibt. Diese Aufgabe übernimmt das Item Repository, welches permanent den Event Bus auf Status-Updates abhört und die Änderungen persistiert. Das Item Repository verhindert, dass angebundene Dienste ihren Status selbst speichern müssen und stellt diese in einem zentralen Dienst zur Verfügung. Dafür gibt es verschiedene von openHAB unterstütze Services: InfluxDB , RRD4J, JDBC und mongoDB. Alle verfolgen den gleichen Zweck der Speicherung von Zeitdaten mit der zugehörigen ausgeführten Aktion.

Rules

Mithilfe von Rules können Automatisierungsregeln definiert werden. Dies geschieht, wenn gewisse Bedingungen, wie Zustandsänderungen von Items, Systemevents oder Events von selbst angelegten Timern, erfüllt werden. Beispiele für Automatisierungsregeln sind Einschalten von Licht bei Dunkelheit, pausieren der Heizung bei offenem Fenster-/Türkontakt oder Abstellen des Backofen nach Ablauf eines Timers (*Rules* | *OpenHAB*, o. J.).

Sitemaps

Eine Sitemap ist das generisch konfigurierbare User Interface (UI) von openHAB. Sitemaps werden in einer Xtext basierten DSL geschrieben und weisen eine baumartige Struktur von Widgets auf. Widgets, die den aktuellen Stand darstellende

Items oder auch Schalter und Grafiken sein können, definieren die verschiedenen Seiten und den Inhalt des UI. OpenHAB interpretiert das Konfigurationsfile und stellt die Sitemap über ein REST-API zur Verfügung. Es können unterschiedliche Sitemaps für mobile Geräte oder alternative Darstellungsoptionen definiert werden. Standardmäßig stehen bei openHAB drei verschiedene UI's zur Verfügung. Das Paper UI ist eine AngularJS basierte HTML5-Webanwendung zur Konfiguration und Verwaltung der openHAB Instanzen. Das bedeutet add-on Management, identifizieren und hinzufügen von Things im eigenen Netzwerk und das verbinden von Thing channels mit den entsprechenden Items. Darüber hinaus stehen dem Benutzer noch Classic UI und Basic UI zur Verfügung. Ersteres ist das originale Web-User Interface von openHAB 1 und damit auch die am weitesten verbreitete und gleichzeitig stabilste Benutzeroberfläche. Da Aussehen und User Experience nicht mehr den modernen Standards entsprechen, wird sich langfristig Basic UI als Nachfolger durchsetzen. Basic UI wurde so konzeptioniert, dass es mit verschiedenen Bildschirmgrößen verwendet werden kann und trotzdem ein stimmiges Design erhalten bleibt. Zusätzlich bietet es Ajax-Navigation und live Updates der vorhandenen Komponenten (*Sitemaps | OpenHAB*, o. J.).

4 ESP32 Mikrocontroller Modul

IoT Anwendungen, wie die hier vorgestellte Smart Home Automation, benötigen Geräte, welche die 6A (Anything, Anytime, Anyone, Anyplace, Any service and Any Network) erfüllen (Li et al., 2015).

Hierfür werden so genannte system-on-a-chip Mikrocontroller mit drahtlos Netzwerkfunktionen verwendet. Diese Mikrocontroller können Aktoren steuern und Sensoren auslesen und gleichzeitig über bspw. Wi-Fi mit einem Netzwerk verbunden sein (Maier et al., 2017).

4.1 Einführung

Ein Beispiel für solch einen kostengünstigen, kleinen und dennoch performanten Mikrocontroller ist der ESP32 des chinesischen Herstellers Espressif Systems. Dieser wird mit einer Antenne, Oszillator und Flash-Speicher in dem ESP32-WROOM-32 Modul kombiniert. Dieses Modul bietet ein hervorragendes Grundgerät für den Einsatz im Smart Home. Aufgrund dessen wird der ESP32-WROOM-32 bei vielen käuflich zu erwerbenden fertig IoT Geräten eingesetzt, wie auch bei dem für dieses Projekt eingesetzten Magic Home LED Controller.

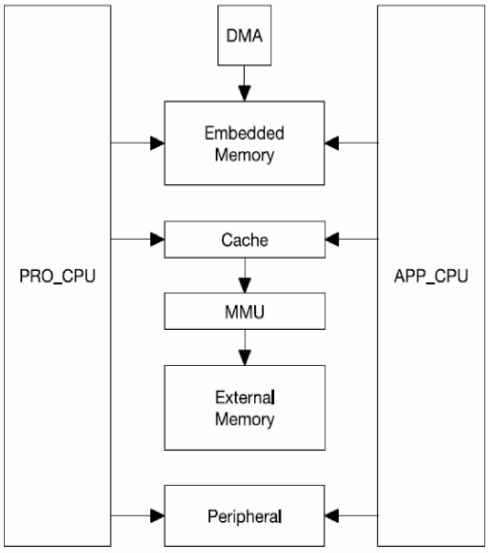

Abbildung 4: ESP System Struktur (Quelle: Maier et al., 2017, S. 143)

Chip (Module)	ESP32 (ESP-WROOM-32)
Details:	
CPU	Tensilica Xtensa LX6 32 bit Dual-Core at 160/240 MHz
SRAM	520 KB
FLASH	2MB (max. 64MB)
Voltage	2.2V bei 3.6V
Operating Current	80 mA Durchschnitt
Programmable	Free (C, C++, Lua, etc.)
Open Source	Ja
Konnektivität:	
Wi-Fi	802.11 b/g/n
Bluetooth®	4.2 BR/EDR + BLE
UART	3
I/O:	
GPIO	32
SPI	4
12C	2
PWM	8
ADC	18 (12-bit)
DAC	2 (8-bit)
Größe	25.5 x 18.0 x 2.8 mm
Preis	£8

Tabelle 1: Mikrocontroller für das IoT-Design (Quelle: Maier et al., 2017, S. 144)

4.2 Technische Details

Maier et al. (2017) deklarieren ein ESP32 als ein Dual-Core-System mit zwei Harvard Architecture Xtensa LX6 CPUs. Alle eingebetteten Speicher, externen Speicher und Peripheriegeräte befinden sich auf dem Datenbus und/oder dem Befehlsbus dieser CPUs. Der Mikrocontroller verfügt über zwei Kerne - PRO_CPU für das Protokoll und APP_CPU für die Anwendung, deren Aufgaben jedoch nicht festgelegt sind. Der

Adressraum für Daten- und Befehlsbus beträgt 4 GB und der periphere Adressraum 512 KB. Außerdem sind die eingebetteten Speicher 448KB ROM, 520KB SRAM und zwei 8KB RTC-Speicher. Der externe Speicher unterstützt bis zu viermal 16MB Flash (Maier et al., 2017).

Außerdem unterstützt der ESP32 folgende Protokolle: TCP/IP, 802.11 *b*/g/n Wi-Fi und Bluetooth v4.2 BR/EDR und BLE. Dies ermöglicht ein umfangreiches Spektrum an Einsatzfeldern und Integrationsmöglichkeiten (Carducci et al., 2019).

4.3 Arduino

Besonders ist auch noch für den ESP32, dass er Open Source ist und somit Freiraum für Bastler und Individualisierungen bietet. Aufgrund dessen wird der ESP32 bspw. auch von dem Arduino IDE unterstützt.

Arduino ist ein Hard- und Softwareunternehmen aus Italien, dessen Fokus darin liegt, Mikrocontroller Boards mit Sensoren und Aktoren herzustellen und diese über die Arduino Programmiersprache (basierend auf Wiring) im open-source Arduino IDE (basierend auf Processing) zu programmieren (Arduino, 2018).

Durch Erweiterungen lassen sich diverse Entwickler Boards einbinden, so auch das AZ-Delivery ESP32 Board. Die Einstellungen für das Arduino IDE lassen sich den Bedienungsanleitungen der Hersteller entnehmen.

Das Arduino IDE ermöglicht aufgrund seiner open-source auch die Entwicklung von Libraries. Ein Beispiel für solch eine Library ist die PubSubClient Library von knolleary auf GitHub. Diese Library bietet einen MQTT-Client für den Arduino, um publish und subscribe Nachrichten mit einem MQTT Broker auszutauschen (knolleary, 2020).

Aufgrund dessen können die durch den ESP32 ausgelesenen Sensordaten einfach an einen MQTT Broker veröffentlicht werden.

4.4 MQTT-Temp/Hum-Sensor

Für das DIY-Kit wurde ein DHT11 Temperatur- und Luftfeuchtigkeitssensor mit dem AZ-Delivery ESP-32 Developer Board verbunden und über Arduino so eingerichtet, dass dieses IoT-Gerät alle 10 Sekunden die Temperatur- und Luftfeuchtigkeitswerte in die entsprechenden Topics eines MQTT Brokers veröffentlicht.

Der Schaltplan dieses Gerätes ist in Abbildung 12 zu sehen.

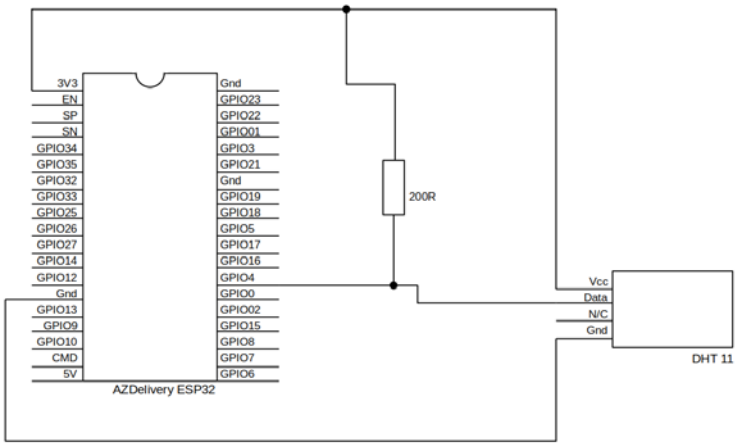

Abbildung 5: Schaltplan des DIY-Kits (Quelle: Eigene Darstellung)

Das auf den ESP32 gespielte sketch basiert auf dem Code von randomnerdtutorials.com. Es nutzt die Async MQTT Client Library für die Verwendung des MQTT Protokolls, hierfür wird außerdem die Async TCP library benötigt. Die Daten der DHT Sensoren können über die DHT library von Adafruit ausgelesen werden.

Neben den MQTT Methoden connectToMqtt, onMqttConnect, onMqttDisconect, onMqttPublish, deren Funktionalitäten durch die Namen leicht erkennbar sind, existieren des Weiteren die Methoden connectToWifi und WifiEvent, welche dem Aufbauen einer Wi-Fi Verbindung dienen. Dem entsprechend werden alle MQTT und Wi-Fi notwendigen Werte als Variable vorab definiert.

Die Anwendung selbst teilt sich in Setup und loop auf. Dies ist der Regelfall bei sketch Anwendungen für Mikrocontroller. Im Setup wird die Verbindung zu dem MQTT Broker hergestellt:

```
1 void setup() {
2   Serial.begin(115200);
3   Serial.println();
4
5   dht.begin();
6
```

```
  mqttReconnectTimer = xTimerCreate("mqttTimer", pdMS_TO_TICKS(2000),
pdFALSE, (void*)0,
reinterpret_cast<TimerCallbackFunction_t>(connectToMqtt));
  wifiReconnectTimer = xTimerCreate("wifiTimer", pdMS_TO_TICKS(2000),
pdFALSE, (void*)0,
reinterpret_cast<TimerCallbackFunction_t>(connectToWifi));

  WiFi.onEvent(WiFiEvent);

  mqttClient.onConnect(onMqttConnect);
  mqttClient.onDisconnect(onMqttDisconnect);
  mqttClient.onPublish(onMqttPublish);
  mqttClient.setServer(MQTT_HOST, MQTT_PORT);
  connectToWifi();
}
```

In der loop werden über den vorher definierten Port 4 des ESP32 über die DHT library die Werte für Temperatur und Luftfeuchtigkeit ausgelesen und anschließend in die entsprechenden MQTT Topics veröffentlicht:

```
#define MQTT_PUB_TEMP "esp32/dht/temperature"
#define MQTT_PUB_HUM  "esp32/dht/humidity"

void loop() {
  unsigned long currentMillis = millis();
  // Every X number of seconds (interval = 10 seconds)
  // it publishes a new MQTT message
  if (currentMillis - previousMillis >= interval) {
    // Save the last time a new reading was published
    previousMillis = currentMillis;
    // New DHT sensor readings
    hum = dht.readHumidity();
    // Read temperature as Celsius (the default)
    temp = dht.readTemperature();
    // Read temperature as Fahrenheit (isFahrenheit = true)
    //temp = dht.readTemperature(true);

    // Check if any reads failed and exit early (to try again).
    if (isnan(temp) || isnan(hum)) {
      Serial.println(F("Failed to read from DHT sensor!"));
      return;
    }

    // Publish an MQTT message on topic esp32/dht/temperature
    uint16_t packetIdPub1 = mqttClient.publish(MQTT_PUB_TEMP, 1, true,
String(temp).c_str());
    Serial.printf("Publishing on topic %s at QoS 1, packetId: %i",
MQTT_PUB_TEMP, packetIdPub1);
    Serial.printf("Message: %.2f \n", temp);

    // Publish an MQTT message on topic esp32/dht/humidity
    uint16_t packetIdPub2 = mqttClient.publish(MQTT_PUB_HUM, 1, true,
```

```
33 String(hum).c_str());
34     Serial.printf("Publishing on topic %s at QoS 1, packetId %i: ",
35 MQTT_PUB_HUM, packetIdPub2);
36     Serial.printf("Message: %.2f \n", hum);
37   }
38 }
```

(Code Beispiel des MQTT Broker)

5 Maschine zu Maschine Kommunikation (MQTT)

5.1 Definition und Funktionsweise von MQTT

In diesem Kapitel wird das MQTT-Protokoll vorgestellt. MQTT steht für **M**essage **Q**ueuing **T**elemetry **T**ransport. Es ist ein leichtgewichtiges Publish-and-Subscribe-System, mit dem man als Client Nachrichten veröffentlichen und empfangen kann. Es wird häufig als offenes Netzwerk zu Maschinenkommunikation verwendet (Schulz, 2017, S. 5).

5.2 Anwendungsbereich

MQTT wird unter anderem zur Anbindung von IoT-Devices an Backend-Plattformen wie ThingWorx verwendet. In unserem Projekt wurde OpenHAB genutzt. Auch im Bereich Connected Car findet MQTT Einsatz immer wieder. Mit MQTT kann man Befehle zur Steuerung von Ausgängen senden, Daten von Sensorknoten lesen und veröffentlichen und vieles mehr. Das Protokoll macht es daher wirklich einfach, (Schulz, 2017, S. 5)eine Kommunikation zwischen mehreren Geräten herzustellen. Wenn man Heimautomatisierungen mag und ein komplettes Smart Home aufbauen möchte, kommt man an MQTT kaum vorbei.

5.3 Funktionsweise

Das MQTT-Protokoll folgt den Regeln einer Publish-Subscribe-Kommunikation:

So kann eine einfache Kommunikation zwischen mehreren Geräten hergestellt werden. MQTT ist ein einfaches Messaging-Protokoll, das für eingeschränkte Nutzer auf einer geringen Bandbreite ausgelegt ist und ist somit perfekt für die Nutzung von IOT-Lösungen (Internet of Things).

Es gibt zwei verschiedene Teilnehmer:

Einen Broker als Nachrichtenverteiler und “n” Clients, wobei die Clients als Publisher und Subscriber nicht direkt miteinander kommunizieren, sondern Nachrichten über sogenannte Topics “publishen” (veröffentlichen) und “subscriben” (abonnieren) können (Schulz, 2017, S. 7ff.).

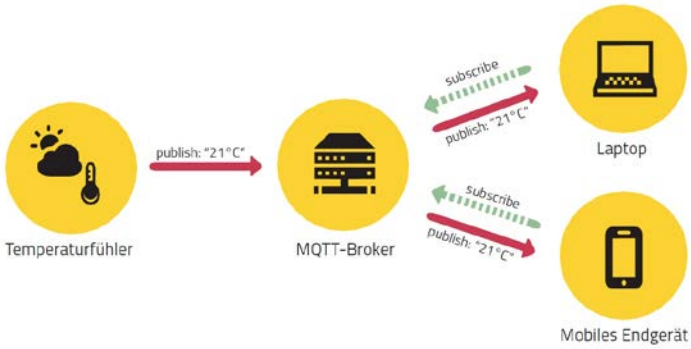

Abbildung 6: Publish/Subscribe-Architektur von MQTT (Quelle: : Raschbichler, 2017)

5.4 Basiskomponenten

In MQTT gibt es eine Hand voll wichtiger Komponenten, die im Folgenden kurz erläutert werden:

- Clients
- Publish/Subscribe
- Messages
- Topics
- Broker

Clients

Der Client ist, wie man sich ganz typisch vorstellt, der “Endnutzer” der Kommunikation und derjenige, der Nachrichten aktiv sendet. Jeder Client identifiziert sich durch eine Client ID, die auch seine Session komplett identifiziert, denn MQTT ist für die Client-Seite komplett zustandslos. Ein Client ist für die Nachrichtenkommunikation zu einem bestimmten Zeitpunkt wichtig (Biskoping, 2013, S. 21). Es gibt zwei verschiedene Kommunikationsorgane, wie im Folgenden erläutert:

Publish/Subscribe

In einem Publish-and-Subscribe-System kann ein Gerät eine Nachricht auf einem Topic veröffentlichen, oder es kann ein bestimmtes Topic abonnieren, um Nachrichten zu empfangen.

Beispiel:

Abbildung 7: Publish-and-Subscribe-System (Quelle: Sallat, 2018)

DEVICE 1 veröffentlicht eine Nachricht in einem Topic.

DEVICE 2 abonniert das gleiche Topic, in dem DEVICE 1 veröffentlicht.

Also empfängt DEVICE 2 die Nachricht (Sallat, 2018, S. 21).

Messages und QoS

Nachrichten sind die Informationen, die man zwischen Ihren Geräten austauschen möchten. Dabei ist es egal, ob es ein Befehl für eine Aktion oder ob es Daten sind.

Nachrichten werden mit einer "**Q**uality **o**f **S**ervice" verschickt, welche folgende drei Werte anbietet:

0: "Fire-and-forget" – Paket wird genau einmal verschickt. Kommt an, unter Umständen auch vielleicht mal nicht. Ist analog dem UDP-Protokoll und enorm schnell.

1: "Acknowledgement" – der Empfänger bestätigt dem Sender, das Paket erhalten zu haben. Es ist möglich, dass ein Paket mehrmals ankommt. Trotzdem ist diese Methode immer noch schnell.

2: "Synchronisiert" – das Paket erreicht garantiert das Ziel, und zwar garantiert nur einmal, jedoch erzeugt diese Variante etwas mehr Verkehr (Sallat, 2018, S. 34f.).

Topics

Ein weiteres wichtiges Konzept sind die Topics. Topics sind die Art und Weise, wie man Interesse an eingehenden Nachrichten anmelden oder angeben kann, wo man die Nachricht veröffentlichen möchte.

Topics werden mit Zeichenketten dargestellt, die durch einen Schrägstrich getrennt sind. Jeder Schrägstrich steht für eine Themenebene (Schulz, 2017, S. 11).

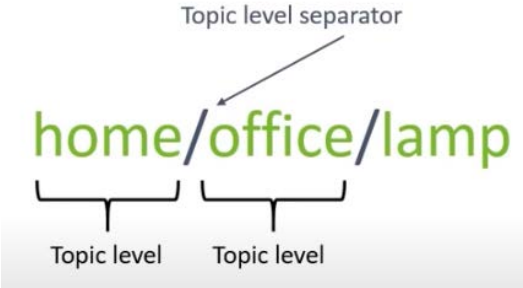

Abbildung 8: Darstellung von Topics (Quelle: Eigene Darstellung)

Beispiel:

Hier unser Beispiel, wie Sie eine Temperatur- und Luftfeuchtigkeitssensor in Ihrem Smart Home implementieren würden, der die Leuchtintensität ihrer LED-Leiste an die Temperatur anpassen kann:

1. Man benutzt einen Sensor, der die aktuellen Temperaturwerte als Nachrichten auf dem Topic „Zuhause/Wohnzimmer/Temperatur=Licht“ veröffentlicht.
2. Man benutzt ein Gerät, das die LED-Leiste in Ihrem Wohnzimmer steuert (dies kann ein ESP32, ESP8266 oder ein anderes Board sein). Der ESP32, der die LED-Leiste steuert, ist auf dem Topic abonniert: „Zuhause/Wohnzimmer/Temperatur=Licht“.
3. Wenn also eine neue Nachricht in diesem Topic veröffentlicht wird, empfängt der ESP32 die Nachricht und passt das Licht der LED-Leiste auf die entsprechende Temperatur an.

Dieses erste Gerät kann ein ESP32, ein ESP8266 oder eine Home-Automation-Controller-Plattform wie z.B. Node-RED, Home Assistant, Domoticz oder OpenHAB sein.

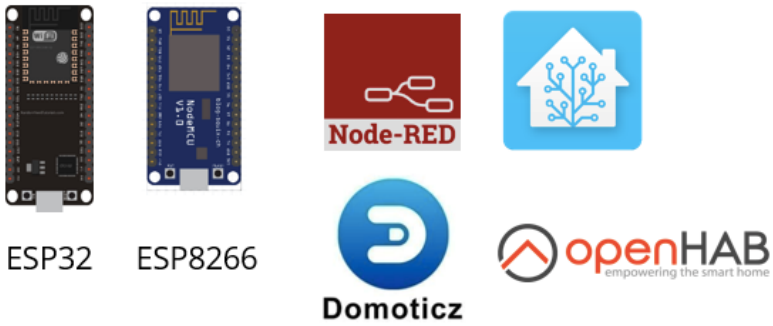

Abbildung 9: IoT – MQTT Komponenten

Broker

Der Broker ist in erster Linie dafür verantwortlich, alle Nachrichten zu empfangen, die Nachrichten zu filtern, zu entscheiden, wer daran interessiert ist und dann die Nachricht an alle abonnierten Clients zu veröffentlichen.

Es gibt verschiedene Broker, die man verwenden kann. In unserem Smart-Home-Projekt verwenden wir den Mosquitto-Broker, der auf dem Raspberry Pi installiert ist. Alternativ kann man auch einen Cloud-MQTT-Broker verwenden. Die Aufgabe des Brokers ist hier die Nachrichtenverwaltung und -verteilung (Sallat, 2018, S. 52f.).

Abbildung 10: Schematischer Aufbau MQTT Kommunikation (Quelle: Sallat, 2018)

6 Das Smart Home der Zukunft

Die Funktionalitäten rund um das eigene Smart Home System werden sich in den nächsten Jahren rasant entwickeln. Bis jetzt experimentieren die meisten privaten Anwender lediglich mit an das Internet angebundenen Steckdosen, Lampen, Kameras oder Thermostaten. Gesteuert werden diese in der Regel über eine App oder eine Sprachsteuerung. Wer sich besser auskennt, hat schon erste Automationen und Regeln für die Steuerung seines Smart Homes formuliert. Das ist eine Entwicklung, die wir in den nächsten Jahren verstärkt wahrnehmen werden. Denn der Wechsel vom manuellen, in der Wand verbauten Schalter, auf einen Knopfdruck in der App ist zwar praktisch, aber um ehrlich zu sein auch noch nicht wirklich smart. Es gibt jedoch auch schon erste Projekte von professionell geplanten Smart Homes. Je nachdem was der Interessent ausgeben möchte, ist dann auch die Steuerung der Verdunklung, ein Alarmsystem, vernetzte Haushaltsmaschinen oder der vollautomatische Rasenroboter ins intelligente Haus integrierbar.

6.1 Technischer Aufbau

Damit sich das Smart Home weitgehend selbst steuern kann, müssen verschiedene Regeln zur Automation festgelegt werden. Im Kapitel zur openHAB Software wurde bereits die Relevanz von Persistenz im Smart Home Kontext unterstrichen. In dem Zusammenhang ist auch der Begriff InfluxDB, als Datenbankservice zum Speichern der Zeitdaten mit den zugehörigen ausgeführten Befehlen, gefallen. Auch wir verwenden in unserem DIY-Kit InfluxDB zum Persistieren der Daten. Jedes openHAB besitzt also einen eigenen Datenbank-Service, welcher die Daten in einem Kafka-Cluster publisht. Das InfluxDB Image steht dem Nutzer als Docker Container zur Verfügung, sodass die Persistenz und Publishing Voraussetzungen einfach mit wenigen Klicks in das eigene Smart Home System implementiert werden können. Kafka wurde entwickelt, um Datenströme zu speichern und zu verarbeiten, und stellt gleichzeitig eine Schnittstelle zum Laden und Exportieren von Datenströmen zu Drittsystemen bereit. Kafka speichert die Daten in einem HDFS Data Lake und gibt sie an eine pyspark weiter. Anwendungen zum maschinellen Lernen finden in der pyspark Umgebung statt.

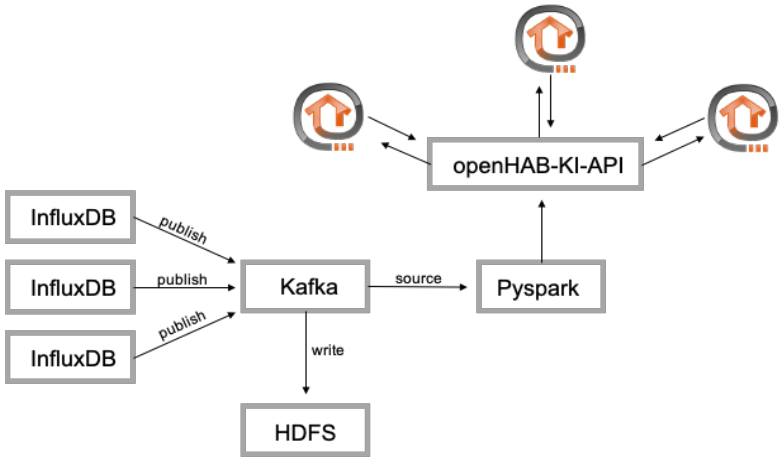

Abbildung 11: Kafka Architektur für ein zentralisiertes openHAB System (Quelle: Eigene Darstellung)

Die Abbildung zeigt eine Kafka Architektur für ein zentralisiertes openHAB System. Sie ist einem Kappa-Streaming Prozess nachempfunden. Die Daten aus den verschiedenen intelligenten Häusern werden zentral gespeichert und mithilfe von Machine Learning Algorithmen ausgewertet. Daraufhin wird ein Empfehlungsdienst (engl. Recommender System) basierend auf mehreren Modellen entwickelt. Das könnte mithilfe einer Support Vector Machine pro openHAB Gerät und einem auf Neuronalen Netzen basierendem generischen Empfehlungsdienst umgesetzt werden (Pfisterer, 2020, S. 288ff.).

6.2 Business Modell

Sobald Smart Home Systeme flächendeckender eingesetzt werden, ergeben sich neue Business Modelle für Anbieter. Wenn der Benutzer nicht über die fachlichen Kenntnisse verfügt, selber Automatisierungen zu schreiben respektive Sensoren oder andere Komponenten zu integrieren, so kann er auf ein „as a Service-Modell“ zurückgreifen. Dem Anwender wird eine openHAB-KI-API zur Verfügung gestellt, um die Daten zur Auswertung in die Cloud zu schicken. Zunächst kann aus einer Reihe von Automatisierungsmöglichkeiten gewählt werden, die durch intuitive Anpassungen den eigenen Anforderungen gerecht werden. In der Cloud läuft ein Machine Learning Algorithmus auf den Daten und registriert alle manuellen Anpassungen wie das Aktivieren der Kaffeemaschine, Anpassen der Heiztemperatur oder regulieren der Leuchtkraft. Bei jeder Aktion lernt das System also die persönlichen Präferenzen und

kann sich dadurch entsprechend des eigenen Tagesablaufs und der eigenen Vorlieben verbessern.

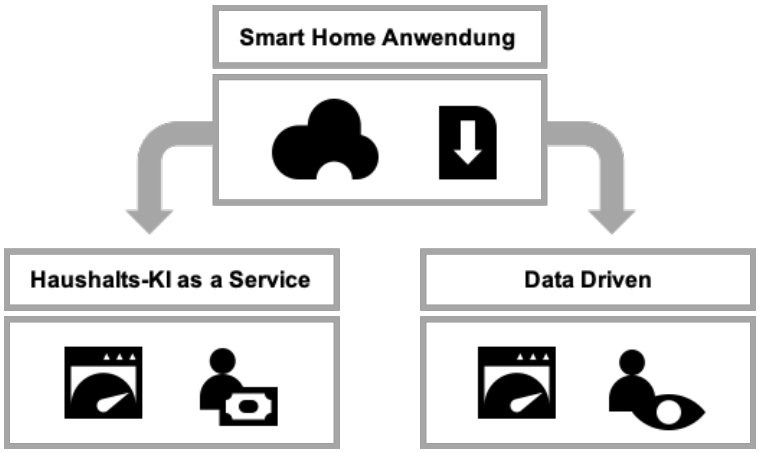

Abbildung 12: Geschäftsmodell (Quelle: Eigene Darstellung)

Um das Geschäftsmodell zu monetarisieren, kann der Benutzer zwischen einer entgeltpflichtigen Haushalts-KI oder einem Data-Driven Modell entscheiden. Ersteres würde dem Kunden einen monatlichen Betrag, für das Nutzen der Cloud und das Bereitstellen der Softwarelösung, kosten. Ein Daten getriebenes Modell könnte so aussehen, dass Kommunen oder Energieanbieter die anonymisierten Daten aus dem Smart Home gegen Entgelt zur Verfügung gestellt bekommen und diese für beispielsweise Statistiken zu Strom- oder Gasverbrauch nutzen können. Damit würde für den Endnutzer die Gebühr entfallen und er könnte die Softwarelösung samt cloudbasierter Optimierung kostenfrei nutzen (Gassmann et al., 2021, S. 224ff. & 337 ff.).

7 Schlussbetrachtung

Das Ziel der vorliegenden Arbeit ist es, ein System zu entwickeln, auf dem eine skalierende KI-gesteuerten Hausautomatisierung aufbauen kann. Hierfür wurde ein einfach einzurichtendes “Do-it-yourself"-Kit erstellt und dokumentiert. Dieses Kit beinhaltet mehrere Funktionen eines Smart-Homes wie bspw. ESP32 basierte MQTT Sensoren für Temperatur und Luftfeuchtigkeit oder auch die Ansteuerung von IoT Geräten über ein UI. Dieses DIY-Kit ermöglicht die Persistenz durch einen mit ein paar Klicks aufgesetzten Docker Container. Dieser Docker Container dient in unserer KI-Hausautomatisierung als Grundlage der Datensammlung und Bereitstellung für die Kappa Architektur.

Auf dieser praktischen Implementierung aufbauend bilden wir unsere Zukunftsvisionen, welche eine voll funktionierende und selbstlernende Haushalts-KI umfasst. Diese KI lernt auf den im DIY-Kit gelernten Daten in einer Kappa Architektur und trainiert mit jeder Änderung durch den Nutzer im Smart-Home. Für den Nutzer wird die Automatisierung durch die KI via REST-API angesprochen. Bei erfolgreicher Umsetzung wäre dies ein universelles, selbstlernendes System für die Hausautomatisierung.

Mit diesem Produkt sehen wir die Chance für zwei Business-Modelle. Einerseits ein As-a-Service Variante bei der der Nutzer eine monatliche Gebühr für die Nutzung der Haushalts-KI entrichtet und andererseits ein datengetriebenes Modell, wobei die Nutzung der Haushalts-KI kostenlos für den Nutzer ist, jedoch die gesammelten Daten ausgewährtet und an Dritte verkauft werden. Dritte könnten bei diesem Modell Kommunen oder Versorgungsunternehmen sein.

Zusammenfassend ist diese Arbeit praktisch ein erleichterter erster Schritt für den Leser ins eigene Smart-Home. Langfristig bietet das entwickelte DIY-Kit eben auch die Grundlage für eine skalierende Haushalts-KI wie in den entsprechenden Kapiteln dargelegt wurde.

8 Literaturverzeichnis

Biskoping, L. (2013). *Aufbau einer KommunikationsInfrastruktur auf Basis von MQTT.* Fachhochschule Südwestfalen. https://elib.dlr.de/86248/1/Bachelorarbeit_Biskoping_MQTT.pdf

Carducci, C. G. C., Monti, A., Schraven, M. H., Schumacher, M., & Mueller, D. (2019). Enabling ESP32-based IoT Applications in Building Automation Systems. In *2019 IEEE International Workshop on Metrology for Industry 4.0 and Internet of Things: Naples, Italy, June 4-6, 2019 : proceedings* (S. 306–311). IEEE.

Freier, D., & Leutenegger, M. (2015). *Internet of Things mit openHAB und Microsoft Azure* HSR Hochschule für Technik Rapperswil]. eprints.hsr.ch. https://eprints.hsr.ch/456/

Gassmann, O., Frankenberger, K., & Choudry, M. (2021). *Geschäftsmodelle entwickeln: 55 innovative Konzepte mit dem St. Galler Business Model Navigator* (3. überarbeitete und erweiterte Auflage).

Li, S., Xu, L. D., & Zhao, S. (2015). The internet of things: a survey. *Information Systems Frontiers*, *17* (2), 243–259. doi: 10.1007/s10796-014-9492-7

Maier, A., Sharp, A., & Vagapov, Y. (2017). Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things. In R. Picking, S. Cunningham, N. Houlden, D. Oram, V. Grout, & J. Mayers (Hrsg.), *2017 Internet technologies and applications (ITA): Proceedings of the seventh International Conference : Tuesday 12th-Friday 15th September 2017, Wrexham Glyndŵr University, Wales, UK* (S. 143–148). IEEE. doi: 10.1109/ITECHA.2017.8101926

Paulsen, N., & Klöß, S. (2020). Smart Home 2020: Jeder Zweite träumt vom komplett vernetzten Zuhause. *Bitkom e.V.,* 2020. https://www.bitkom.org/Presse/Presseinformation/Smart-Home-2020-Jeder-Zweite-traeumt-vom-komplett-vernetzten-Zuhause

Pfisterer, D. (2020). *Vorlesung Big Data: Kappa Architecture.*

Schulz, L. (2017). *Datenübertragung mittels MQTT.* Universität Ulm. http://dbis.eprints.uni-ulm.de/1570/1/document.pdf

Strese, H., Seidel, U., Knape, T., & Botthof, A. (2010). *Smart Home in Deutschland: Untersuchung im Rahmen der wissenschaftlichen Begleitung zum Programm Next Generation Media (NGM) des Bundesministeriums für Wirtschaft und Technologie.*

8.1 Internetdokument

Arduino. (2018). *What is Arduino?* Zugriff am 14.01.2021 unter: https://www.arduino.cc/en/Guide/Introduction

Bindings | *openHAB.* (o. J.). Zugriff am 11.01.2021 unter: https://www.openhab.org/docs/developer/bindings/

Brendan O'Brien. (2014). *The 'Internet of Things' Quotes to Consider*. Zugriff am 23.01.2021 unter: https://www.ariasystems.com/blog/internet-things-quotes-consider/

Eichstädt-Engelen, T., Kreuzer, & K. (2014). *Durchbruch — Offene Plattformen verhelfen Smart Home zum Erfolg*. Zugriff am 13.01.2021 unter: https://www.innoq.com/de/articles/2014/11/durchbruch/

Event Bus | *openHAB.* (o. J.). Zugriff am 11.01.2021 unter: https://www.openhab.org/docs/developer/utils/events.html

Gietz, J. (2020). *Die Geschichte des Smart Home von den 1920er Jahren bis heute*. Zugriff am 14.01.2021 unter: https://blog.1und1.de/2020/11/16/smart-home-geschichte/

Items | *openHAB.* (o. J.). Zugriff am 12.01.2021 unter: https://www.openhab.org/docs/configuration/items.html

Knolleary. (2020). *Arduino Client for MQTT.* GitHub repository. Zugriff am 14.01.2021 unter: https://github.com/knolleary/pubsubclient

Kreuzer, K. (2014). *OpenHAB Architecture (OpenHAB Developer)*. Zugriff am 12.01.2021 unter: https://raw.githubusercontent.com/wiki/openhab/openhab1-addons/images/architecture.png

Lucero, S. (2016). *IoT platforms: enabling the Internet of Things*. Zugriff am 14.01.2021 unter: https://cdn.ihs.com/www/pdf/enabling-IOT.pdf

Persistence | *openHAB.* (o. J.). Zugriff am 11.01.2021 unter: https://www.openhab.org/docs/configuration/persistence.html

Rules | *openHAB.* (o. J.). Zugriff am 13.01.2021 unter: https://www.openhab.org/docs/configuration/rules-dsl.html

Sallat, M. (2018). *Das Anwendungsprotokoll MQTT im Internet of Things*. Zugriff am 24.01.2021 unter: https://opus.hs-offenburg.de/frontdoor/deliver/index/docId/2771/file/THESIS_MARIO_SALLAT.pdf

Sitemaps | *openHAB.* (o. J.). Zugriff am 13.01.2021 unter: https://www.openhab.org/docs/configuration/sitemaps.html

Things | *openHAB.* (o. J.). Zugriff am 12.01.2021 unter: https://www.openhab.org/docs/concepts/things.html

Völkel, F. (2020, December 31). *Die Historie des Smart Home von 1963 - 2020: Meilensteine*. Zugriff am 31.12.2020 unter: https://www.smartest-home.com/smart_home_historie_1939_2019/

Ehrenwörtliche Erklärung

Wir versichern hiermit, dass wir die vorliegende schriftliche Ausarbeitung mit dem Thema „Smart Home - Erste Schritte zu skalierenden KI-gesteuerten Automatisierungen“ ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit wurde bisher weder gesamt noch in Teilen einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Mannheim, 24.01.2021	
Ort, Datum	Peter Behrens
Mannheim, 24.01.2021	
Ort, Datum	Stephan Lenert
Mannheim, 24.01.2021	
Ort, Datum	Nina Mergelsberg
Mannheim, 24.01.2021	
Ort, Datum	Tom Müller