



Smart Home mit Open Home Automation Bus (OpenHAB)

Lukas Kiederle
Dominik Ampletzer
Daniel Böning
Fakultät für Informatik

WS 2019/20

Inhaltsverzeichnis

1	Motivation	4
2	Was ist OpenHAB?	4
3	Bewertung OpenHAB	4
3.1	Fazit der Bewertung	6
3.2	Weiterführende Bewertungen	6
4	openHAB aus technischer Sicht	7
4.1	Add-ons	8
4.2	Bindings	8
4.3	Things	9
4.4	Channels	10
4.5	Items	10
4.6	Rules	11
4.7	Sitemaps	12
5	Programmieren in und für openHAB	12
5.1	Programmierung für den Nutzer	12
5.2	offen für andere Systeme	13
5.3	eigene Binding	13
6	Datenintegrität und Sicherheit	15
6.1	Interne vorhandene Funktionalitäten	15
6.2	Externe Erweiterungsmöglichkeiten	16
7	Verwendung von OpenHAB	16
7.1	Integration der Big Player	16
7.2	Beispiel Aufbau eines OpenHAB Smart-Homes	17
7.2.1	Hardware für openHAB	17
7.2.2	Eingebundene Geräte	17
7.2.3	Eingebundene Services	18
7.2.4	Integration der Geräte und Services	18
7.3	Umgang mit OpenHAB	20
8	Fazit	21
8.1	Stärken	21
8.2	Schwächen	21
9	Outlook	21
10	Infos:	22
A	Erster Abschnitt des Anhangs	23

1 Motivation

Die Ausgaben für das Internet der Dinge (IoT) wird weltweit, laut Statista, bis zum Jahr 2022 auf 1000 Milliarden US-Dollar steigen. Im Vergleich zum Jahr 2019 bedeutet dies, eine Steigerung von über 40%.[Ten] Bei einem solch starken Trend in der Informatik-Branche sollten sowohl Studenten, als auch Professoren dessen Grundlagen kennen. Deshalb ist im Rahmen des Faches Softwarearchitektur an der technischen Hochschule Rosenheim diese Ausarbeitung geschrieben worden. Das Ziel ist es OpenHAB, ein Heimautomatisierungs-Tool, aus praktischer und technischer Sicht zu untersuchen. Außerdem wird dabei auf die Aspekte Markttauglichkeit und Benutzbarkeit in der Praxis eingegangen.

2 Was ist OpenHAB?

OpenHAB ist eine technologie-unabhängige Open-Source-Automatisierungssoftware für Smart-Homes. Das Projekt wurde von Kai Kreuzer 2010 erstmals initiiert und wird mittlerweile durch die Community weiterentwickelt. Die Software ist hauptsächlich in Java und der Auszeichnungssprache XML geschrieben. Seit dem 16. Dezember 2019 ist die Version 2.5 erhältlich.

Auf der offiziellen Website von OpenHAB <https://www.openhab.org/> sind drei klare Hauptziele definiert, die diese Software erreichen soll. Dabei ist ein Ziel die Plattformunabhängigkeit. Somit kann OpenHAB sowohl auf Linux, MacOS oder Windows betrieben werden. Auch das Hosten mit Docker oder einem Raspberry Pi wird unterstützt.

Weiterhin soll es durch die Plugin-Architektur möglich sein, fast jedes Gerät zu integrieren. Es werden über 200 Technologien und mehrere tausende verschiedene Geräte unterstützt.

Das dritte Ziel weist auf die vielen verschiedenen Automatisierungsmöglichkeiten hin, die OpenHAB zu bieten hat. Dabei werden Auslöser (in englisch: Trigger), Aktionen, Skripte und auch Voice-Kontrolle genannt.

3 Bewertung OpenHAB

In diesem Kapitel wird das open-source Projekt OpenHAB untersucht und bewertet. Es befindet sich auf Github unter <https://github.com/openhab/openhab-core>. Die hierfür verwendeten Kriterien sind orientiert an der QAware Open Source Quick-Check Liste. Diese beinhaltet:

Kriterium	Beschreibung
Projekt Aktivität	Gibt es? <ul style="list-style-type: none">• Mindestens 1 Release im letzten Jahr• Mindestens 3 aktive Contributor• Stetige Commit-Aktivität von mindestens 1x pro Monat
Reifegrad	Existiert eine stabile Version > 1.0?
Lizenz	Für welche Zwecke kann das Projekt genutzt werden?
Support	Existiert Issue-Tracking und ein Antwortzeit auf Tickets unter 24h?

Dokumentation	Existiert? <ul style="list-style-type: none"> • Api-Dokumentation • Getting Started Tutorial • Aktuelle Dokumentation
---------------	--

Tabelle 1: OpenHAB Projektbewertungskriterien

Der Online-Dienst **Open Hub**, ehemals **Ohloh** genannt, katalogisiert opensource Softwareprojekte. Dabei werden Daten wie Projektname, Beschreibung und Sourcecode erfasst. Basierend auf diesen Daten erstellt Open Hub eine Statistik, die es ermöglicht, Codeanalyse, Projektmitarbeiter, Aktivitäten und eine Übersicht zu erhalten. Dabei werden auch viele weitere open source Projekte miteinander verglichen um aussagekräftige Statistiken und Aussagen treffen zu können.

In der Auswertung über OpenHAB steht beispielsweise, dass im letzten Jahr 343 Entwickler aktiv an dem Projekt mitgearbeitet haben. Somit ist OpenHAB unter den top 2% der größten Projektteams auf Open Hub.

Des Weiteren ist ein stetiger Anstieg von Interesse erkennbar. Dies wird durch den Vergleich von Codebeiträgen des aktuellen Jahrs und des Vorjahres begründet.

Insgesamt haben über 1140 Entwickler bereits mehr als 20000 Beiträgen in Form von Programmcode zum Projekt beigetragen. Der Umfang des zu 98% in Java geschriebenen Codes beträgt mehr als 1.5 Millionen Zeilen. Davon wurden circa 31% dokumentiert. Dies entspricht dem durchschnittliche Wert aller Java open-source Projekte, welche auf Open Hub registriert sind.[OPEb] Die hierbei angegebenen Werte beziehen sich auf alle Projekte und Subprojekte von OpenHAB. Eine Übersicht hierfür ist unter <https://github.com/openhab> zu finden.

Wie bereits im Kapitel 2 erwähnt, ist OpenHAB aktuell in der Version 2.5 erhältlich. Daraus lässt sich auf einen stabilen Codestand schließen.

Des Weiteren ist das Projekt unter einer EPL-2.0 Lizenz veröffentlicht. Dies erlaubt sowohl die private als auch kommerzielle Nutzung. Hinzu kommt die Möglichkeit für Modifizierung und Weiterverbreitung.[LIC]

Des Weiteren wird die Qualität des Supports evaluiert. Hierfür stellt Github einen Issue-Tracker zur Verfügung. Beim Teilprojekt openhab-core wurden darüber über 1250 Issues verfasst, wovon bei circa 90% bereits die Bearbeitung abgeschlossen ist.[GIT] Dies lässt auf eine aktive Nutzung des Issue-Trackers schließen und bietet somit eine hilfreiche Support-Möglichkeit. Hierfür ist vor allem das Label **Bug** von Bedeutung. Eine Stichprobenartige Untersuchung von Issues dieses Types hat eine durchschnittliche Antwortzeit in unter 24h ergeben. Dies wurde anhand des ersten Kommentars gemessen. Zudem wurden bereits circa 83% der insgesamt 68 dokumentierten Bugs gelöst.

Abschließend wird der Zustand der Dokumentation untersucht. Das Getting-started Tutorial ist ausführlich und aktuell. Die darüber hinausgehende Basisdokumentation ist vollständig. Nur in Spezialfällen, wie beispielsweise der Implementierung von Services, ist die Dokumentation lückenhaft.[OPEa]

3.1 Fazit der Bewertung

Kriterium	Beschreibung	Erfüllt?
Projekt Aktivität	Gibt es? <ul style="list-style-type: none"> • Mindestens 1 Release im letzten Jahr • Mindestens 3 aktive Contributor • Stetige Commit-Aktivität von mindestens 1x pro Monat 	✓
Reifegrad	Existiert eine stabile Version > 1.0?	✓
Lizenz	Für welche Zwecke kann das Projekt genutzt werden?	✓
Support	Existiert Issue-Tracking und eine Antwortzeit auf Tickets unter 24h?	✓
Dokumentation	Existiert? <ul style="list-style-type: none"> • Api-Dokumentation • Getting Started Tutorial • Aktuelle Dokumentation 	X/✓

Tabelle 2: OpenHAB Projektbewertungskriterien Ergebnis

3.2 Weiterführende Bewertungen

OpenHAB wurde bereits in der Bachelorarbeit von Pirmin Gersbacher vom Jahr 2017/2018 anhand von Usecases untersucht und verglichen. Dabei kam er zu folgender Ergebnis.

	OpenHAB	ioBroker	Home Assistant	Node-RED
Installation	+	+	+	++
Oberfläche	+	O	O	++
Technologien	+	-	++	++
Einfachheit	O	+	++	-
Visualisierung	+	++	O	O
Erweiterbarkeit	++	++	++	++
Automation	++	++	O	O
Verbreitung	+	-	++	O

Abbildung 1: Vergleich OpenHAB und anderen Heimautomatisierungstools von 2017/2018
In der Tabelle 1 sind die unterschiedlichen Gebiete der Untersuchung auf der linken Seite zu

finden. In der horizontalen sind die einzelnen Heimautomatisierungs-Tools aufgelistet. In der jeweiligen Zelle, basierend auf Reihe und Spalte, sind die Bewertungen festgehalten. Dabei zeigt ein blau markiertes Feld, welches das beste Tool für ein Gebiet ist.

OpenHAB sticht dabei nicht sonderlich heraus. Allerdings ist es in keinem der aufgelisteten Kategorien negativ bewertet.

Der Autor schreibt in seiner Bachelorarbeit, dass er persönlich auf OpenHAB setzen würde. Dies liegt einerseits daran, dass die Entwickler von OpenHAB stark an Vereinfachung von komplizierteren Komponenten arbeitet. Andererseits sollen aber auch komplexere Automationen, durch die Nähe von Framework und Java, möglich sein.[Ger]

Dadurch, dass die Bachelorarbeit von P. Gersbacher vor über einem Jahr veröffentlicht wurde, stellt diese nicht mehr aktuellen Stand dar. Seitdem wurde die neuere Version 2.5 entwickelt. Diese beinhaltet einige fundamentale Veränderungen. Dabei sind Anpassungen, um eine leichtere Nutzung sicherzustellen. Auf diese Weise können sowohl OpenHAB-Entwickler, als auch Entwickler, die OpenHAB als Basis verwenden, deutlich einfacher programmieren, sagt Kai Kreuzer in seinem Online Blog Post über das OpenHAB 2.5 Release.[Kre]

In dieser Arbeit wird auf die speziellen Änderungen der Version 2.5 eingegangen. Dabei werden die Ergebnisse von P. Gersbacher als zusätzliche Quelle genutzt.

4 openHAB aus technischer Sicht

In diesem Kapitel werden die grundlegenden Komponenten, welche openHAB verwendet, dargestellt. Desweiteren wird auf die Beziehung der Komponenten untereinander eingegangen. Es wird ein Kernaspekt von openHAB vorgestellt. openHAB bietet in der Regel nicht den "einen Weg". Es bietet mehrere Wege ein Ziel zu erreichen, je nach Vorlieben des Nutzers. So ist es z.B. möglich ein Gerät über die Web-Oberfläche als auch über geschriebenen Code zu integrieren. Das gleiche ist auch bei Rules zu sehen, welche entweder über ein Add-on in der Web-Oberfläche definiert werden können oder über Code. Dieses Konzept ist ein Grundgedanke, der von openHAB verfolgt wird, welcher allerdings zu Verwirrung führen kann, da bei anfänglichen Umgang mit openHAB nicht unbedingt klar ist, wie man sein gewünschtes Ziel erreichen kann. In Tabelle 3 sind einige der Grundlegenden Komponenten aus openHAB zur schnellen Übersicht aufgeführt. Detailliert werden diese in den entsprechenden Unterkapiteln beschrieben.

Komponente	Beschreibung
Add-ons	Erweiterungen, welche die Funktionalitäten von openHAB erhöhen.
Bindings	openHAB-Komponenten, welche die Schnittstelle zu fremd Systemen darstellt.
Things	Repräsentation von physischen Geräten/Services in openHAB.
Items	Darstellung von Eigenschaften und Ressourcen von openHAB - Thing bezogen
Channels	Übertragungskanal zwischen „Items“ und „Things“.
Rules	Automatisierungsregeln, in Wenn-Dann-Struktur.
Sitemaps	individuelle Benutzeroberfläche, welche Informationen präsentiert und Interaktionen ermöglicht.

Tabelle 3: OpenHAB Komponenten

4.1 Add-ons

openHAB setzt verwendet das Konzept von Add-ons als Teil ihrer zusammensteckbaren Architektur (pluggable architecture). Dadurch dass vordefinierte Bereiche von openHAB erweitert werden können, wird openHAB der Anforderung gerecht “alles“ zu integrieren. Das Konzept der Add-ons ermöglicht es außerdem nur Benötigte Module zu installieren bzw. zu verwenden wodurch openHAB schlank, leicht und überschaubar bleibt. Die Bereiche in denen openHAB offen ist um erweitert zu werden sind:

- Bindings
- Automation Engine Modules
- Transformations / Profiles
- IO Services
- Persistence Services
- Audio & Voice

Um den Rahmen dieser Arbeit nicht zu übersteigen wird nur auf Bindings detaillierter eingegangen, siehe Kapitel 4.2. Bei den nicht näher Beschriebenen Add-ons handelt es bei Automation-Engine-Modules um Bedingungen oder Aktionen, welche für Rules oder Scripte verwendet werden können. Transformations / Profiles können genutzt werden um Werte welche durch Channels übertragen werden zu transformieren/modifizieren. IO Services ermöglichen es interne Schnittstellen von openHAB nach außen aufzumachen, dies geschieht z.B. bei der REST-API, dem HomeKit für Apple oder dem Hue Emulation Service für Philips. Persistence Services können genutzt werden um den Status von Items z.B. in Datenbanken abzuspeichern und wieder Abzurufen. Und bei Audio & Voice handelt sich um Services welche genutzt werden können um Audioquellen abzuspielen oder Stimminteraktionen mit Nutzern zu ermöglichen.

4.2 Bindings

Bindings sind die Komponenten, welche es ermöglicht Systeme oder Geräte mit openHAB zu integrieren. Mit einem Binding kann sowohl ein physisches Geräte wie z.B. eine LG Fernseher mit WebOS als auch eine Service z.B. Spotify angebunden werden. Die Bindings sind dabei meist soweit abstrahiert, dass nicht jedes einzelne Model bzw. Version eine eigenes Binding benötigt. Nach der Installation eines Bindings, was über die Web-Oberfläche oder per Code geschehen kann, ist openHAB in der Lage Geräte/Systeme im Netzwerk zu finden. Mittels Bindings können auch Geräte/Systeme integriert werden, welche wiederum die Steuerung für andere Geräte übernehmen z.B. Hue-Bridge oder Bluetooth. Diese Geräte/Systeme wiederum ermöglichen es i.d.R. nach dem Hinzufügen in openHAB, dass durch das Binding Kindkomponenten gefunden und ebenfalls zu openHAB hinzugefügt werden können. Dadurch ist es möglich, wie im Falle des Hue-Systems nicht nur die Hue-Bridge, sondern alle mit der Hue-Bridge verbundenen Lampen einzeln in openHAB zu integrieren und dadurch zu steuern. Dieses ansteuern von einzelnen Kindkomponenten im Falle des Hue-Systems, wäre durch ZigBee theoretisch möglich, würde aber einen sehr hohen und unnötigen Aufwand bedeuten. Laut openHAB gibt es aktuell über 300 Bindings welche es ermöglichen über 2000 Things anzusprechen.

4.3 Things

Things stellen die Repräsentation von physischen Geräten/Services innerhalb von openHAB dar. Sie bestehen neben verschiedenen Status und Konfigurationsinformationen aus Channels, welche für die Ansteuerung der Things benötigt werden und näher in Kapitel 4.4 beschrieben werden. Things können auf drei verschiedene Arten openHAB hinzugefügt werden. Diese sind:

- Per Web-Oberfläche automatisch: Nachdem das für das Thing benötigte Binding installiert wurde, erscheint das Thing in der Inbox, siehe Abbildung 2. Von dieser Inbox aus, kann das Thing mit wenigen Klicks openHAB hinzugefügt werden. Durch diese Methode werden alle nötigen Einstellungen für das Thing automatisch getroffen. Diese können nachträglich bearbeitet werden z.B. umbenannt werden.

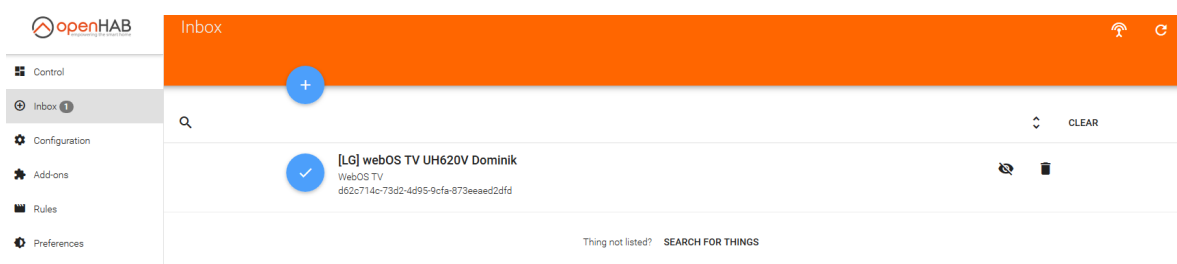


Abbildung 2: Thing per Web-Oberfläche

- Per Web-Oberfläche Manuell: Das manuelle Hinzufügen von Things ermöglicht volle Einstellungsfreiheit für Things. Dazu zählen ID, Bezeichnung, IP-Adresse und Konfigurationsparameter. Dies ist in Abbildung 3 zu sehen. Diese Einstellungsfreiheit setzt allerdings auch voraus, dass der Nutzer weiß, welche Informationen er in welches Feld eintragen will/muss. Genau wie beim automatischen Erkennen können die Einstellungen noch nachträglich geändert werden.

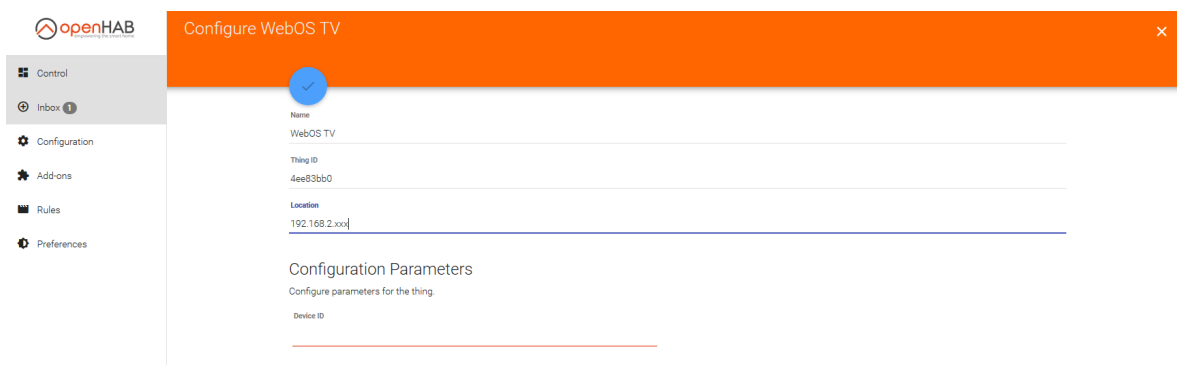


Abbildung 3: Thing per Web-Oberfläche Manuell

- Per Code. Ermöglicht und benötigte komplette manuelle Eingabe aller nötigen Informationen im Dateordner *openHAB-config/things/*. Diese sind wie im einfachen Beispiel Abbildung 4 Binding-Pfad, IP-Adresse, und ID. Um dies zu vereinfachen sind in der Dokumentation Beispiele für die verschiedenen Bindings mit zugehörigen Things zu finden.


```
things > lg_tv.things
1 Thing lgwebos:WebOSTV:tv1 [host="192.168.2.113", key="6ef1dff6c7c936c8dc5056fc85ea3aef"]
2 |
```

Abbildung 4: Thing per Code

4.4 Channels

Channels sind Kommunikationswege welche von Things angeboten werden und diese mit Items verbinden. Mit Hilfe der Channels werden Aktionen, welche Things auszuführen haben Parameter übergeben. So bietet der verwendete LG Fernseher wie in Abbildung 5 zu sehen eine Liste von Channels an, welche Daten für Aktionen übertragen um diese auszuführen. Das heißt, dass Channels sowohl die Kommunikationswege mit zugehörigen Parameter repräsentieren, als auch Aktionen, welche Things ausführen können. Ein gewünschte Aktion eines Things kann ohne den passenden Channel diese Aktion nicht ausgeführt werden.

Configuration > Things > WebOS TV



WebOS TV
WebOS TV
WebOS based smart TV

Status: **UNINITIALIZED - HANDLER_CONFIGURATION_PENDING**

Channels



















	Power lgwebos:WebOSTV:tv1.power Switch	
	Mute lgwebos:WebOSTV:tv1.mute Switch	
	Volume lgwebos:WebOSTV:tv1.volume Dimmer	
	Channel lgwebos:WebOSTV:tv1.channel Number	
	Channel Name lgwebos:WebOSTV:tv1.channelName String	
	Toast lgwebos:WebOSTV:tv1.toast String	
	Media Control lgwebos:WebOSTV:tv1.mediaPlayer Player	
	Stop lgwebos:WebOSTV:tv1.mediaStop Switch	
	Application lgwebos:WebOSTV:tv1.appLauncher String	

Abbildung 5: Channel Liste

4.5 Items

Der Begriff Item oder zu deutsch Ding, kann missverständlich aufgefasst werden. Bei einem Item handelt es sich um alle Informationen, welche es über eine Aktion gibt. So hat ein Item u.a. Be-

zeichnung, Kategorie, Typ, Status als auch Konfigurationen z.B. dass nur Werte von 0 - 50 übertragen werden. Diese Informationen werden benötigt um Gruppierungen zu ermöglichen, den Status eines Gerätes abzufragen (z.B. Lautstärke eines Fernsehers) und auch mit einem Thing zu kommunizieren bzw. Aktionen auszuführen. Der Typ eines Items gibt an, welche Basistypen von dem Item akzeptiert/konsumiert werden können. Solche Basistypen können einfache aus der Programmierung bekannte Typen wie String, Number, DateTime usw. sein, als auch komplexere openHAB spezifische Typen wie Location, Player oder Switch. Eine komplette Auflistung ist unter <https://www.openhab.org/docs/configuration/items.html#type> zu finden.

Items sind essentiell um Aktionen auf der Web-Oberfläche dazustellen. So können diese in Gruppen oder Einzeln dargestellt werden um als Statusanzeige und/oder als manuelle Steuerung des Smart Home zu agieren. Für diese Steuerung ist anzumerken, dass jedem Item nur ein Channel zugeordnet werden kann, aber jedem Channel mehrere Items. Die Kardinalität ist somit Item 1 -> n Channel.

Items können wie bei openHAB üblich ebenfalls über die Web-Oberfläche als auch im Code erzeugt werden. Da dieses Prinzip schon hinreichend erklärt wurde, wird mit Ausnahme der automatischen Namensgebung nicht näher darauf eingegangen. Diese Namensgebung ist in erster Linie frei, jedoch vergibt openHAB über die Web-Oberfläche als Item-Name eine Kombination aus Thing- und Channel-Namen wodurch sehr gut ersichtlich wird, wofür ein Item genutzt werden kann z.B. *LGWebOSTVUH620V_VolumeTV* oder *HueWhiteLamp1_Brightness*.

4.6 Rules

Automatisierung in openHAB findet durch Rules/Scripte statt. Diese Rules werden aktuell standardmäßig durch Programmieren erzeugt. Um Rules zu erzeugen, wird eine einfache WENN-DANN-Struktur verwendet. Diese Regeln können auf verschiedene Bedingungen "lauschen". So können Bedingungen (WENN) von Items, Member of (Gruppierungen), Time, System oder Things stammen. Genauer kann die Bedingung durch eine verschiedene Formen eingeschränkt werden wie z.B. received command [<command>], received update [<state>] oder changed [from <state>] [to <state>] usw. Wenn diese Bedingung erfüllt ist wird im DANN-Pfad die Reaktion definiert. So können Item oder ganze Gruppen von Items angesprochen werden. Hierbei wird ein/jedes Item Informationen an Channels weitergeben, welche eine Aktion bei dem dazu gehörigen Thing auslöst. Wie im Codebeispiel 1 zu sehen, gibt es die Möglichkeit innerhalb des DANN-Blocks weitere Strukturelemente einzufügen um aus mehreren Rules eine zu machen.

Alternativ zur Programmierung ist es mit einem experimentelles Add-on möglich, über die Web-Oberfläche Rules zu definiert. Allerdings ist hier anzumerken, dass komplexere Regeln schwierig bis gar nicht zu erzeugen sind. Dies liegt daran, dass die Eingabemöglichkeiten beschränkt sind und durch das Konvertieren in das JSON-Format Sonderzeichen wie >, <, =, ! nicht funktionieren. Des Weiteren ist das Einfügen von zusätzlichen Strukturelementen über das experimentelle Add-on nicht möglich.

```
1 rule "React on Volume (LGWebOSTVUH620VDominik_Volume) change"
2 when
3     Item LGWebOSTVUH620VDominik_Volume changed
4 then
5     logDebug("React some changes on Volume", "current Value: "
6         + LGWebOSTVUH620VDominik_Volume.state.toString())
7 if ( LGWebOSTVUH620VDominik_Volume.state >= 20 ) {
```

```

7      HueWhiteLamp2_Brightness.sendCommand(80)
8  }
9  else {
10     HueWhiteLamp2_Brightness.sendCommand(5)
11  }
12  end

```

Codebeispiel 1: Rule Beispiel

4.7 Sitemaps

Eine Sitemap dient als Übersicht, Gruppierung und Visualisierung des Smart Home. So kann durch die Sitemap das Smart Home z.B. in Räume, Stockwerke oder den Außenbereich unterteilt werden und diesen Bereichen Items in Einzelform oder Gruppen zugewiesen werden. Dadurch ist es möglich manuell Items zu betätigen oder eine Übersicht über einen Bereich zu erhalten. Es gibt neben Sitemaps noch die Möglichkeit verschiedene Dashboards und Panels zu erzeugen, welche ebenfalls genutzt werden können um individuelle Übersicht oder Steuerung zu erhalten. Wie im ganzen openHAB ist es auch hier möglich per Code oder openHAB vorinstallierten Editoren sich seine eigenen Sitemap, Dashboard oder Panel zu erzeugen

5 Programmieren in und für openHAB

openHAB sieht verschiedene Formen der Programmierung vor. Es wird Fremdsystemen die Möglichkeit gegeben openHAB zu integrieren. Die Community wird dazu angehalten, openHAB selbst weiterzuentwickeln als auch neue Add-ons zu entwickeln. Aber auch der normale openHAB Nutzer kommt schnell an den Punkt an dem er geringe Programmierungen vornehmen muss. Spätestens wenn er komplexere Automatisierungen vornehmen will.

5.1 Programmierung für den Nutzer

Als Nutzer ist es möglich viele Funktionen von openHAB zu Nutzen, ohne programmieren zu müssen. Dennoch ist es, wie im Kapitel 4 verwiesen, auch über Code machbar Bindings, Things usw. zu integrieren. Abbildung 6 kann entnommen werden, wie die Ordnerstruktur aussieht, welche dem Nutzer für die Programmierung unter dem Dateiordner *openHAB2-conf* zur Verfügung steht. Aus der Abbildung wird schnell klar, wo welche Komponente hinzugefügt werden können. In den entsprechenden *readme.txt*-Dateien wird nochmals darauf hingewiesen, was in den jeweiligen Ordner geschrieben werden muss. Da es sich bei allen *openhAB2-conf*-Dateien um Textdateien handelt wird in den *readme.txt*-Dateien darauf hingewiesen, welche Dateieindungen in dem jeweiligen Ordner zu verwenden sind und wo Beispiele in der openHAB

Dokumentation zu finden sind.

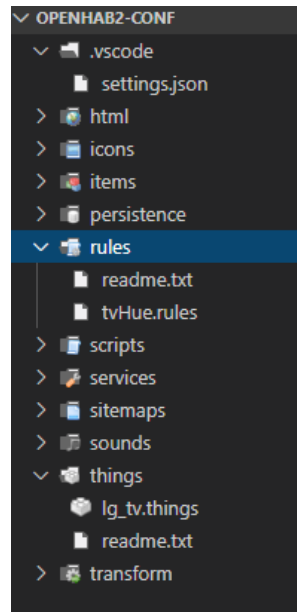


Abbildung 6: openHAB-conf Ordnerstruktur

5.2 offen für andere Systeme

openHAB bietet durch eine REST-API anderen Systemen die Möglichkeit, openHAB zu integrieren. Da diese REST-API die Funktionalität der Web-Oberfläche widerspiegelt, können andere Systemen die volle Funktionalität der openHAB Web-Oberfläche ebenfalls nutzen. Diese REST-API ermöglicht es auch für mobile Geräte openHAB Oberflächen zu entwickeln, welche bereits für Android, iOS und als Windows App angeboten werden.

Die REST-API ermöglicht es auch Entwicklungsumgebungen Informationen aus dem openHAB System zu erhalten und dadurch das Entwickeln für openHAB zu erleichtern. Visual Studio Code, Eclipse und IntelliJ stellen Add-Ons/Extensions zur Verfügung. Diese bieten neben farblich kenntlichem Code noch Things und Items an, welche über die REST-API erhalten wurden. Dadurch können Fehler beim Abtippen von Bezeichnungen oder IDs vermieden werden und bei der Entwicklung kann ein guter Überblick über die zu verwendenden Komponenten erhalten werden, ohne über openHABs Web-Oberfläche zu navigieren.

5.3 eigene Binding

Sollte es bei openHAB keine passendes Binding für ein gewünschtes Gerät geben, steht der Erweiterung durch ein selbst entwickeltes Add-on nicht im Weg. Bereits im Kapitel 4.1 wurde darauf verwiesen, dass openHAB offen für Add-ons ist. Da die Dokumentation für meisten Add-ons nur mit einem *TODO* beschreiben ist, wird hier nur auf das Binding eingegangen. Um ein Binding zu entwickeln bietet openHAB eine skeleton-script welches eine komplette Binding-Vorlage anlegt, siehe Abbildung 7. Diese Vorlage enthält alle Funktionen und Dateien, welche geändert werden müssen um ein funktionsfähiges Binding zu entwickeln. Dadurch wird nicht nur die von openHAB vorgegebene Struktur erzielt und dem erfahrenen Entwickler schreibarbeit abgenommen, sondern auch unerfahren Entwicklern innerhalb der Dateien nochmal erklärt,

was in jeder Funktion prinzipiell zu machen bzw. zu ändern ist.

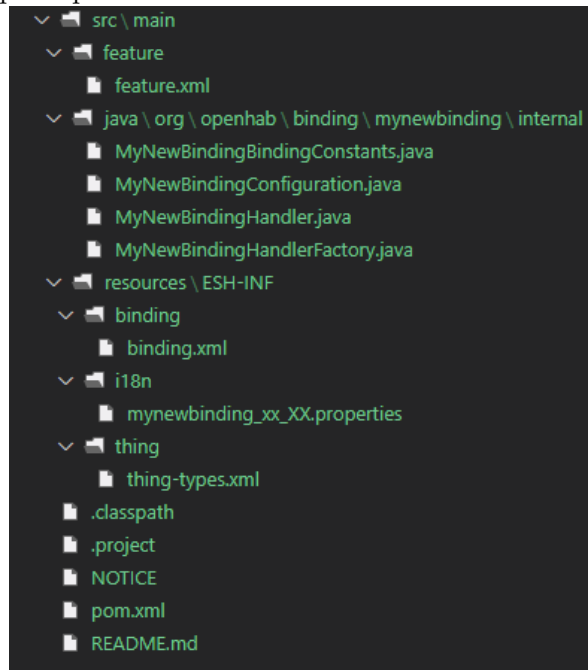


Abbildung 7: Skelett für Binding

Die Binding-Vorlage gibt u.a. vor, dass im XML-Format beschreiben werden muss, mit welchem Gerät das Binding stattzufinden hat und welche Things inklusive Channels erzeugt werden. In JAVA werden benötigte Binding-Handler und Konfigurationen geschrieben. Abbildung 2 kann entnommen werden, wie intensiv die Unterstützung durch Binding-Vorlage ist. So wird nicht nur mit einem Beispiel gezeigt, wie auf Informationen, welche durch einen Channel übertragen werden reagiert werden kann, sondern auch auf Besonderheiten hingewiesen und was in welcher Funktion zu entwickeln ist.

```
1 xxxBindingHandler.java
2 ...
3 @Override
4 public void handleCommand(ChannelUID channelUID, Command command) {
5     if (CHANNEL_1.equals(channelUID.getId())) {
6         if (command instanceof RefreshType) {
7             // TODO: handle data refresh
8         }
9
10        // TODO: handle command
11
12        // Note: if communication with thing fails for some reason,
13        // indicate that by setting the status with detail information:
14        // updateStatus(ThingStatus.OFFLINE, ThingStatusDetail.
15        //     COMMUNICATION_ERROR,
16        //     "Could not control device at IP address x.x.x.x");
17    }
18 }
```

```

19 @Override
20 public void initialize() {
21     // logger.debug("Start initializing!");
22     config = getConfigAs(MyNewBindingConfiguration.class);
23
24     // TODO: Initialize the handler.
25     // The framework requires you to return from this method quickly.
26     // Also, before leaving this method a thing
27     // status from one of ONLINE, OFFLINE or UNKNOWN must be set. This
28     // might already be the real thing status in
29     ...

```

Codebeispiel 2: Handler.java Ausschnitt

6 Datenintegrität und Sicherheit

- It is known that security breaches may have a significant economic impact on a firm, as described by Goel and Shawky [18]. Data loss or theft, tampering, and unauthorized operations are just some of the possible occurrences led by the lack of proper security mechanisms in place. In the case of openHAB, a smart home application, it is not quite quantifiable how expensive it results to have a security breach occur at any level. The consequences may go from mere user discomfort to identity theft, or worse.
- https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project
- https://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=7&ved=2ahUKEwj_q-k0s7mAhXOIVAKHS4dDYgQFjAGegQICBAC&url=https%3A%2F%2Fcomserv.cs.ut.ee%2Fhome%2Ffiles%2FSoto_computer_science_2018.pdf%3Fstudy%3DATILoputoo%26reference%3DB7FBB08D43717164067F1F9DBA92B98A65913AC9&usg=AOvVaw0631sYhHX3wKxW8kPwxpzE
- Weak, Guessable or Hardcoded Passwords: Easy to brute force (no captcha), unchangeable credentials
- Masterarbeit gefunden -> Status ist veraltet -> Checken, was sich seitdem bei OpenHAB getan hat -> keine wirklich relevanten security changes alexa und google home.
- 2.4 hat auch nur kleine security Changes für GPS Tracker Binding

6.1 Interne vorhandene Funktionalitäten

<https://www.openhab.org/docs/installation/security.html>

- Through the command line console, which is done through SSH and thus always authenticated and encrypted. You will find all details about this in the Console documentation.
- Through HTTP(S) over <https://<ip>:8443>
- SSL Certificates On the very first start, openHAB generates a personal (self-signed, 256-bit ECC) SSL certificate and stores it in the Jetty keystore (in \$USER_DATAetc/keystore).

This process makes sure that every installation has an individual certificate, so that nobody else can falsely mimic your server. Note that on slow hardware, this certificate generation can take up to several minutes, so be patient on a first start - it is all for your own security.

- Security Warning: It is vitally important that you **MUST NOT** directly expose your openHAB instance to the Internet (e.g. by opening a port in your firewall)!

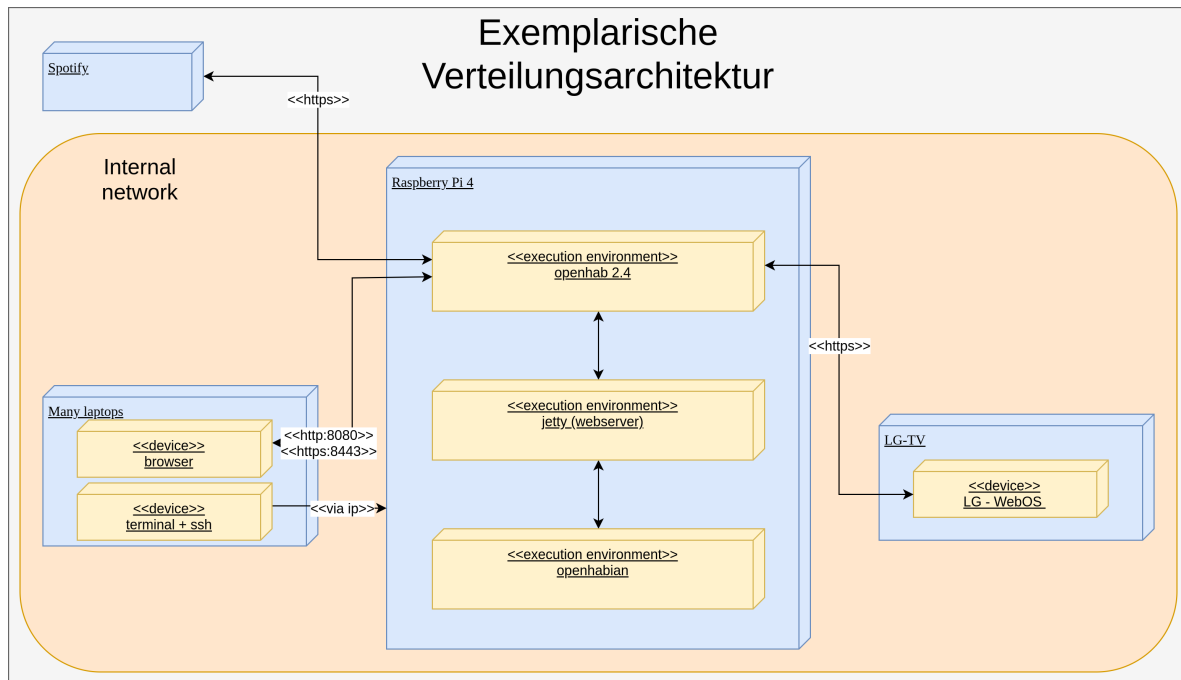


Abbildung 8: Übersicht einer exemplarischen Anwendung von OpenHAB

6.2 Externe Erweiterungsmöglichkeiten

- Options for Secure Remote Access
 - VPN: The most secure option is probably to create a VPN connection to your home network
 - myopenHAB Cloud Service with a tunnel that forwards all requests to the openHAB instance
 - Running openHAB Behind a Reverse Proxy: A reverse proxy simply directs client requests to the appropriate server. This means you can proxy connections to `http://mydomain_or_myip` to your openHAB runtime.

7 Verwendung von OpenHAB

7.1 Integration der Big Player

- Amazon Alexa und Echo Dot Integration möglich
- Changes die mit v2.5 released wurden miteinbauen in das Kapitel: <https://www.openhab.org/blog/2019-12-14-openhab-2-5-release.html>

- Alexa:
 - * This certified Amazon Smart Home Skill allows users to control their openHAB powered smart home with natural voice commands. Lights, locks, thermostats, AV devices, sensors and many other device types can be controlled through a user's Alexa powered device like the Echo or Dot.
 - * <https://www.openhab.org/docs/ecosystem/alexa/>
 - * <https://www.openhab.org/addons/bindings/amazonechocontrol/>
- Google Home
 - * Google Home Integration möglich
 - * With the Action you can voice control your openHAB items and it supports lights, plugs, switches and thermostats. The openHAB Action comes with multiple language support like English, German or French language.
 - * The openHAB Action links your openHAB setup through the myopenHAB.org cloud service to the Google Assistant platform
 - * openHAB Cloud Connector configured using myopenHAB.org . (Items DO NOT need to be exposed to and will not show up on myopenHAB.org , this is only needed for the IFTTT service!) Google account. Google Home or Google Home mini.

<https://www.openhab.org/docs/ecosystem/google-assistant/>

7.2 Beispiel Aufbau eines OpenHAB Smart-Homes

Um mit openHAB Erfahrungen zu sammeln wurde für das Projekt ein eigenes Smart-Home System erstellt. Dabei wurde darauf geachtet ein praktisches System zu erstellen, dass die üblichen Anforderungen an ein Smart-Home erfüllt und zudem eine gute Übersicht über die Grenzen und Möglichkeiten von openHAB geben kann.

7.2.1 Hardware für openHAB

Für unser System wählen wir einen Raspberry pi der vierten Generation mit 2gb Arbeitsspeicher. Der Raspberry ist mit Wlan und Bluetooth ausgestattet. Darauf wird openhabian installiert. Dies ist ein an openHAB angepasstes Linux Buster Betriebssystem. Mit openhabian kommt die neuste Version von openhab und nützliche Systemlibraries und Einstellungen um besser mit openHAB arbeiten zu können. Damit ist unser Raspberry pi bereits fertig eingerichtet und openHAB startet automatisch beim Systemstart. Die Weboberfläche wird im Netzwerk unter <https://openhab:8080> bereitgestellt. Dort können wir die meisten Einstellungen vornehmen um Geräte und Systeme einzubinden und eine UI zu Erstellen. Über SSH ist ein Zugriff direkt auf den Raspberry pi möglich. Ein SSH Zugriff ist für verschiedene Anpassungen nötig.

7.2.2 Eingebundene Geräte

In Openhab lassen sich eine vielzahl an Geräten und Services einbinden. Für unsere Arbeit wählen wir exemplarisch verschiedene Geräte aus die in einem üblichen Wohnzimmer nützlich sind.

- Hue Bridge

- Hue LED
- Osram Lightify LED
- LG Smart TV

Beide Lampen arbeiten mit dem Zigbee Protokoll welches eine eigene Frequenz verwendet und somit leichter viele Geräte verwalten kann. Dadurch kommuniziert unser System immer nur mit der Hue Bridge wenn wir die Lampen ansprechen wollen. Dies passiert allerdings im Hintergrund, In der openHAB Weboberfläche stellt uns die HUE Bridge jede Lampe aber als eigenes Item dar, wodurch wir scheinbar direkt mit ihnen kommunizieren können.

Bei beiden Lampen lässt sich Helligkeit und Farbtemperatur anpassen. Der Fernseher lässt sich an und ausschalten und man kann die Lautstärke und das Programm verändern.

7.2.3 Eingebundene Services

Wir wollen in unser System die Möglichkeit Musik zu steuern integrieren und aktuelle Wetterdaten anzeigen. Dazu wählen wir Spotify als Musikanbieter und openWeatherMap um die Wetterdaten abzurufen. Beide Dienste sind in einer kostenlosen Version nutzbar und daher als Beispiel nützlich.

Spotify ermöglicht es uns über die Developer Webseite von Spotify eine Webanwendung zu registrieren. Damit können wir unser System autorisieren unseren Spotify Account zu steuern. Unser System schickt dabei Befehle an die Server von Spotify, welche wiederum die Befehle an den aktuellen Spotify Client weiterleiten. Wir müssen also nicht direkt Spotify auf unserem Raspberry installieren. Das steigert die Performance und ist in der Handhabung angenehmer. Wir wollen in unserem System das aktuelle Lied anzeigen und es ermöglichen Lautstärke zu verändern, den Track zu wechseln und die Wiedergabe zu starten und stoppen.

Um die Wetterdaten abzurufen ist ein Account bei openWeatherMap nötig. Wir wählen die kostenlose Basisversion und erhalten einen API Key um unser System zu authentifizieren. Mit der Basisversion können wir die aktuellen Wetterdaten verschiedener Standorte abrufen.

7.2.4 Integration der Geräte und Services

Die Integration der Geräte erfolgt über die Paper UI des Webfrontend unseres openHAB Systems.

Für die Integration der Lampen müssen wir diese zuerst über die HUE App mit der Hue Bridge verbinden. Danach wählen wir das HUE Binding. Wir konfigurieren die IP Adresse der Bridge und unser System erkennt die hue Bridge automatisch. Die Bridge kann jetzt als Thing in unser System aufgenommen werden. Dadurch erhalten wir auch automatisch die Lampen als Things. Bei allen drei Lampen erstellen wir je zwei Items um Helligkeit und Lampenfarbe steuern zu können. Diese Items erlauben es uns auch den aktuellen Wert auszulesen. Da die Osram Lampe das Lightify protokoll nutzt, das die Hue Bridge ebenfalls unterstützt, funktioniert die Einrichtung dort identisch.

- Welche Geräte haben wir mit OpenHAB verbunden?
 - Spotify
 - * Lautstärkeregler
 - * Aktueller Song Display

- LG Smart TV
 - * Lautstärkeregler
 - * An- und ausschalten
 - * One-Way-Chat
- Lampen
- Wie haben wir die Geräte verbunden?
 - **Verschiedene Binding:**
 - Spotify Binding
 - LG Smart TV Binding
- On the server the configuration is stored somewhere in userdata (/var/lib/openhab2 for apt-get installs). In an upgrade the userdata folder is preserved when using apt-get.

Activitydiagram for a tv-lamp-rule

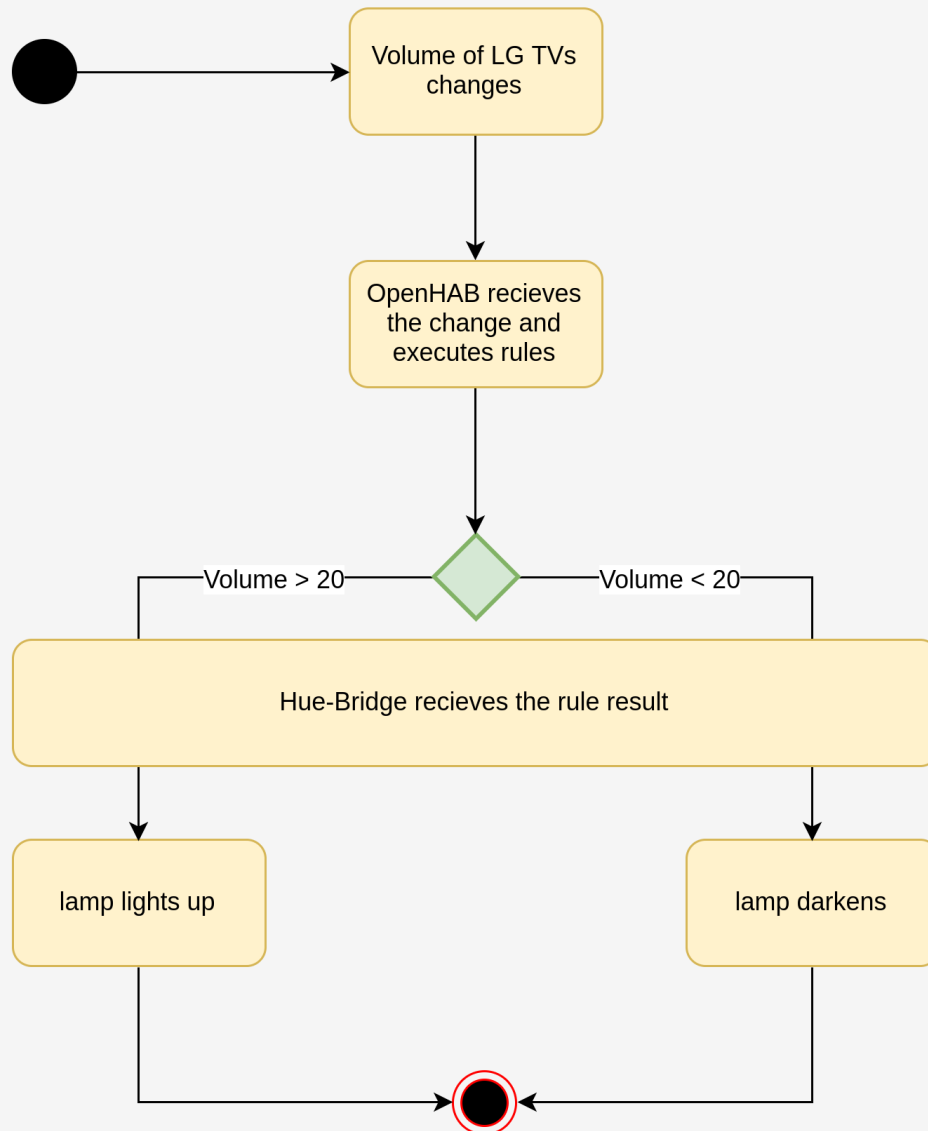


Abbildung 9: Aktivitätsdiagramm für eine Rule

7.3 Umgang mit OpenHAB

- Das meiste klickt mans ich zusammen: Bindings, Rules, Channels, Items, Things
- Implementierung von rules scheint idiotensicher, weil:
 - einfacher Syntax
 - Abhängigkeiten managed Openhab
- Bindings schreiben scheint eher schwieriger

8 Fazit

8.1 Stärken

Some of openHAB's strengths are:

Its ability to integrate a multitude of other devices and systems. openHAB includes other home automation systems, (smart) devices and other technologies into a single solution To provide a uniform user interface and a common approach to automation rules across the entire system, regardless of the number of manufacturers and sub-systems involved Giving you the most flexible tool available to make almost any home automation wish come true; if you can think it, odds are that you can implement it with openHAB.

8.2 Schwächen

Wollen wir das hier als SWOT Analyse aufziehen?

- Integration von USB-Geräten scheint eher kompliziert. Vor allem auf Raspberry Pi
- Serial Binding wird nicht angezeigt
 - Mikrofon an Raspberry Pi oder anderes Geräte verbinden
 - Input des Mikrofons über OpenHAB an ein Ausgabegerät, wie zum Beispiel eine Bluetooth Box, senden und abspielen
 - Raspberry hat da auch für große Probleme bei der Geräteerkennung gesorgt - USB gerät wurde nicht im devices Verzeichnis aufgeführt und somit konnte auch keine Verbindung mit OpenHAB aufgebaut werden
 - OpenHAB Serial Device Binding wurde auch nicht angezeigt, um Geräte darüber zu suchen

9 Outlook

Outlook With the 2.5 release build, our development master branch has now become 3.0.0. This means that there will most likely be no openHAB 2.6 runtime in future, while there will still be 2.x updates on the add-ons, though. The focus of the core maintainers will clearly be on openHAB 3 from now on, which will bring bigger changes that have been discussed since a while: The existing UIs will be replaced by a single one, completely implemented from scratch. The "next-generation" rule engine will become the default one, bringing powerful Python-scripting to all users. Many more changes are discussed that will bring you a fully new experience, while offering an upgrade path for all existing users - so stay tuned!

I would like to thank all our maintainers, contributors and users being such a fantastic community. It is awesome that we have reached another big milestone by shipping openHAB 2.5 and it has been a great journey so far - openHAB will celebrate its 10th anniversary next year! Please continue spreading the word and help growing the community.

Enjoy the upcoming festive season, play with the new openHAB release and share your experiences with us, your family and your friends!

10 Infos:

Ausgangslage Untersuchen Sie die Architektur und Features von OpenHAB und schreiben Sie ein Beispielanwendung. Mit myOpenHub existiert eine kostenlose Plattform die sie nutzen können.

Beantworten Sie dabei

- Aktueller Status des Projekts und
- Integration der Big Player wie Alexa und Google Home
- Welche Tools und Konzepte und APIs gibt es
- Welche Deployment Modi und Betriebsmodi existieren
- Untersuchen Sie auch Aspekte wie Datenintegriertät und Sicherheit

Unterlagen Linkes

- <https://www.myopenhab.org/>
- <https://www.openhab.org/>
- <https://jaxenter.de/openhab-2-4-78711>

A Erster Abschnitt des Anhangs

In diesem Anhang wird . . .

Literatur

- [Ger] P. Gersbacher. Untersuchung und Vergleich von Open Source Plattformen für das Smart Home . https://opus.hs-offenburg.de/frontdoor/deliver/index/docId/2805/file/Abschlussarbeit_P_Gersbacher_178004.pdf. Last visit: 21 Dez 2019.
- [GIT] openhab/openhab-core. <https://github.com/openhab/openhab-core/issues?utf8=>Last visit: 23 Dez 2019.
- [Kre] K. Kreuzer. openHAB 2.5 Release. <https://www.openhab.org/blog/2019-12-14-openhab-2-5-release.html>. Last visit: 24 Dez 2019.
- [LIC] Eclipse Public License - v 2.0. <https://www.eclipse.org/legal/epl-2.0/>. Last visit: 23 Dez 2019.
- [OPEa] Services. <https://www.openhab.org/docs/configuration/services.html>. Last visit: 23 Dez 2019.
- [OPEb] openHAB - empowering the smart home. <https://www.openhub.net/p/openhab>. Last visit: 23 Dez 2019.
- [Ten] F. Tenzer. Prognose zu den Ausgaben für das Internet der Dinge (IoT) weltweit in den Jahren 2018 bis 2022. <https://de.statista.com/statistik/daten/studie/537226/umfrage/prognose-zu-den-ausgaben-fuer-das-internet-der-dinge/>. Last visit: 23 Dez 2019.