



# **Smart Home mit Open Home Automation Bus (OpenHAB)**

Lukas Kiederle  
Dominik Ampletzer  
Daniel Böning  
Fakultät für Informatik

WS 2019/20



# Inhaltsverzeichnis

<b>1</b>	<b>Motivation</b>	<b>4</b>
<b>2</b>	<b>Was ist openHAB?</b>	<b>4</b>
<b>3</b>	<b>Bewertung von openHAB</b>	<b>4</b>
3.1	Fazit der Bewertung . . . . .	6
3.2	Weiterführende Bewertungen . . . . .	6
<b>4</b>	<b>Komponenten von openHAB</b>	<b>7</b>
4.1	Add-Ons . . . . .	8
4.2	Bindings . . . . .	8
4.3	Things . . . . .	9
4.4	Channels . . . . .	10
4.5	Items . . . . .	11
4.6	Rules . . . . .	11
4.7	Sitemaps . . . . .	12
<b>5</b>	<b>Programmieren in und für openHAB</b>	<b>12</b>
5.1	Programmierung für den Nutzer . . . . .	12
5.2	Offen für andere Systeme . . . . .	13
5.3	Eigenes Binding . . . . .	13
<b>6</b>	<b>Verwendung von openHAB</b>	<b>15</b>
6.1	Beispiel Aufbau eines openHAB Smart-Home . . . . .	15
6.1.1	Hardware für openHAB . . . . .	15
6.1.2	Eingebundene Geräte . . . . .	15
6.1.3	Eingebundene Services . . . . .	16
6.1.4	Integration der Geräte und Services . . . . .	16
6.1.5	Erstellung des openHAB Panels . . . . .	17
6.1.6	Kontrollstation einrichten . . . . .	18
6.1.7	Externer Zugriff . . . . .	18
6.2	Verteilungsarchitektur des Smart Home Systems . . . . .	19
6.3	Integration der Big Player . . . . .	20
6.3.1	Amazon Alexa . . . . .	20
6.3.2	Google Home . . . . .	20
<b>7</b>	<b>Datenintegrität und Sicherheit</b>	<b>20</b>
7.1	Interne vorhandene Funktionalitäten . . . . .	21
7.2	Externe Erweiterungsmöglichkeiten . . . . .	22
<b>8</b>	<b>Fazit</b>	<b>23</b>
8.1	Zusammenfassung . . . . .	23
8.2	Schlussfolgerung . . . . .	23
8.3	Ausblick . . . . .	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-Automatisierungs-Software für das Smart-Home. 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 von 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"><li>• Mindestens 1 Release im letzten Jahr</li><li>• Mindestens 3 aktive Contributor</li><li>• Stetige Commit-Aktivität von mindestens 1x pro Monat</li></ul>
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"> <li>• Api-Dokumentation</li> <li>• Getting Started Tutorial</li> <li>• Aktuelle Dokumentation</li> </ul>
---------------	--

Tabelle 1: openHAB Projektbewertungskriterien

Der Online-Dienst *Open Hub*, ehemals *Ohloh* genannt, katalogisiert Open-Source Software-Projekte. Für jedes Projekt werden Daten wie Name, Beschreibung und Quellcode erfasst. Basierend auf diesen Daten erstellt Open Hub eine Statistik, die es ermöglicht Codeanalyse, Projektmitarbeiter, Aktivitäten und eine Übersicht zu erhalten. Dabei fließen auch Daten weiterer Open-Source Projekte ein, um aussagekräftige Statistiken zu erhalten und Aussagen treffen zu können.

In der Auswertung über openHAB ist zu finden, dass 2019 über 380 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 durchschnittlichen Wert aller Java Open-Source Projekte, welche auf Open Hub registriert sind.[OPEf] 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 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.[?]

### 3.1 Fazit der Bewertung

Kriterium	Beschreibung	Erfüllt?
Projekt Aktivität	Gibt es? <ul style="list-style-type: none"> <li>• Mindestens 1 Release im letzten Jahr</li> <li>• Mindestens 3 aktive Contributor</li> <li>• Stetige Commit-Aktivität von mindestens 1x pro Monat</li> </ul>	✓
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"> <li>• Api-Dokumentation</li> <li>• Getting Started Tutorial</li> <li>• Aktuelle Dokumentation</li> </ul>	X/✓

Tabelle 2: openHAB Projektbewertungskriterien Ergebnis

### 3.2 Weiterführende Bewertungen

	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

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. In 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 2017/2018 veröffentlicht wurde, stellt diese nicht mehr den aktuellen Stand dar. Seitdem wurde die neuere Version 2.5 entwickelt. Diese beinhaltet einige fundamentale Veränderungen. Dabei sind Anpassungen, um eine verbesserte Nutzung einhergegangen. Auf diese Weise können sowohl openHAB-Entwickler, als auch Entwickler, die openHAB als Basis verwenden, deutlich einfacher programmieren, laut Kai Kreuzer in dessen Online Blog Post über das openHAB 2.5 Release.[?]

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 Komponenten von openHAB

In diesem Kapitel werden die grundlegenden Komponenten dargestellt, welche openHAB verwendet. Des Weiteren wird auf die Beziehung der Komponenten untereinander eingegangen. Ein Kernaspekt welcher in diesem Kapitel zum Vorschein kommt ist, dass es in der Regel den "einen Weg" nicht gibt. openHAB bietet meist mehrere Wege ein Ziel zu erreichen, je nach Vorlieben des Nutzers. So ist es zum Beispiel 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 ein gewünschtes Ziel erreicht werden 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 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 nach [?]:

- 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 es sich um Services, welche genutzt werden, um Audioquellen abzuspielen oder Stimminteraktionen mit Nutzern zu ermöglichen.

## 4.2 Bindings

Bindings sind Komponenten, welche es ermöglicht Systeme oder Geräte mit openHAB zu integrieren. Mit einem Binding kann sowohl ein physisches Geräte wie z.B. ein 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 ein 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 [OPEe].

### 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 zu 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.

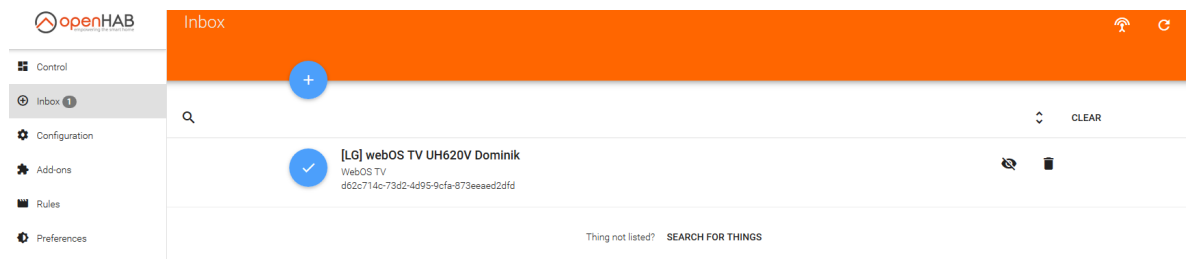


Abbildung 2: Thing per Web-Oberfläche

- 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 nachträglich geändert werden.

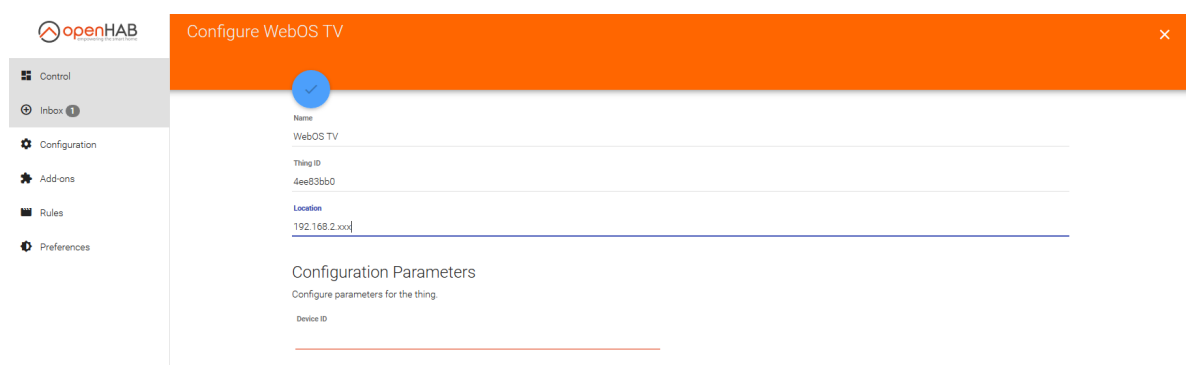


Abbildung 3: Thing per Web-Oberfläche Manuell

- Per Code. Ermöglicht und benötigte komplette manuelle Eingabe aller nötigen Informationen im Dateiordner *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. Eine gewünschte Aktion eines Things kann ohne den passenden Channel diese Aktion nicht ausgeführt werden.

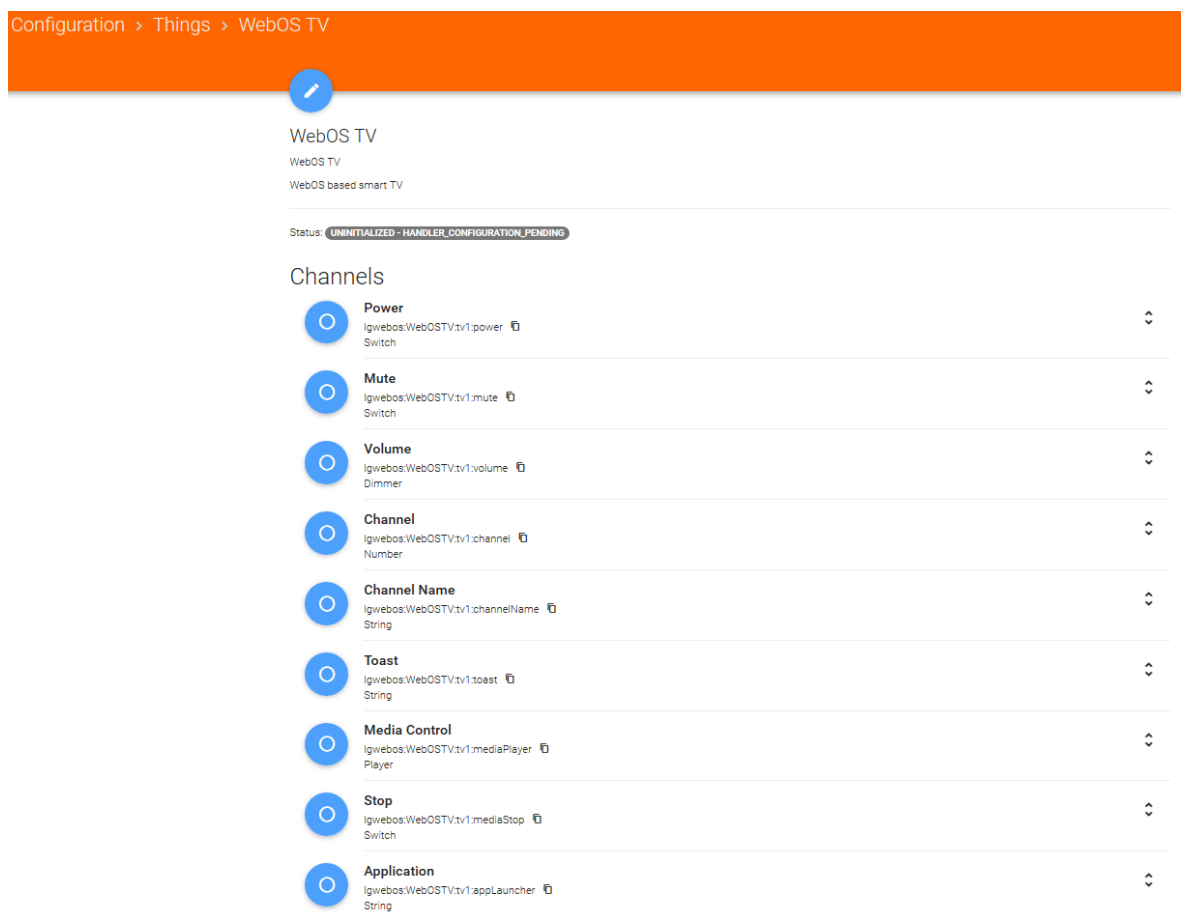


Abbildung 5: Channel Liste

## 4.5 Items

Der Begriff Item oder zu deutsch Gegenstand, 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. Bezeichnung, 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 sowohl 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 Homes 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 sowie 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 Programmierung 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>]. Wenn diese Bedingung erfüllt ist, wird im DANN-Pfad die Reaktion definiert. So können Item oder ganze Gruppen von Items angesprochen werden. Hierbei wird an 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: "
              + LGWebOSTVUH620VDominik_Volume.state.toString)
6 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 fast immer in openHAB ist es auch hier möglich per Code oder über vorinstallierten Editoren Sitemaps, Dashboards oder Panels zu erzeugen.

## 5 Programmieren in und für openHAB

openHAB sieht verschiedene Formen der Programmierung vor. Einerseits stößt der normale openHAB Nutzer schnell an den Punkt, an dem geringe Programmierungen vorzunehmen sind. Spätestens wenn komplexere Automatisierungen nötig sind. Andererseits soll es Fremdsystemen ermöglichen openHAB zu integrieren und die Community wird dazu angehalten, openHAB selbst weiterzuentwickeln als auch neue Add-Ons zu entwickeln. Wie openHAB diesen Anforderungen gerecht wird, ist Inhalt dieses Kapitels.

### 5.1 Programmierung für den Nutzer

Als Benutzer ist es möglich viele Funktionen von openHAB zu nutzen, ohne dabei programmieren zu müssen. Dennoch ist es, wie im Kapitel 4 beschrieben, 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 Dateiendungen in dem jeweiligen Ordner zu verwenden sind und wo Beispiele in der openHAB Dokumentation zu finden sind. Diese sehr einfache und klare Struktur ermöglicht es auch nicht Programmieren sich schnell zurechtzufinden und in Kombination mit den *readme.txt*-Dateien bzw. der Dokumentation in den richtigen Ordner lauffähigen Code zu schreiben.

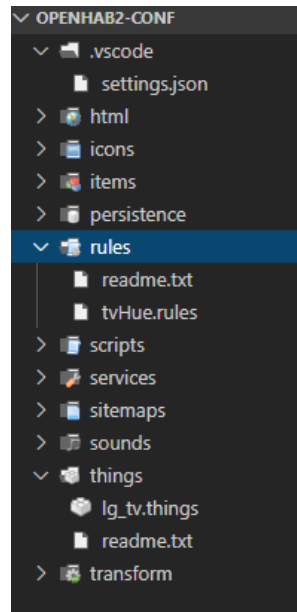


Abbildung 6: openHAB-conf Ordnerstruktur

## 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 erzielt werden, ohne über openHABs Web-Oberfläche zu navigieren.

## 5.3 Eigenes Binding

Sollte es bei openHAB kein passendes Binding für ein gewünschtes Gerät geben, steht der Erweiterung durch ein selbst entwickeltes Add-On nichts im Weg. Bereits im Kapitel 4.1 wurde darauf verwiesen, dass openHAB offen für Add-Ons ist. Da die Dokumentation für Add-Ons meisten nur mit einem *TODO* beschreiben ist, wird hier nur auf die Entwicklung von Bindings eingegangen.

Um ein Binding zu entwickeln, bietet openHAB ein 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 erfahren Entwickler Schreibarbeit abgenommen, sondern auch unerfahren Entwicklern innerhalb der Dateien nochmal erklärt, was in jeder Funktion prinzipiell zu machen bzw. zu ändern ist.

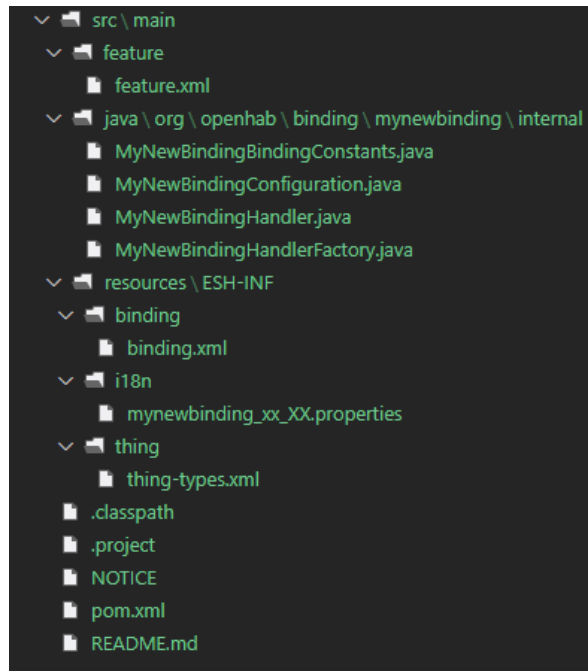


Abbildung 7: Skelett für Binding

Die Binding-Vorlage gibt u.a. vor, dass im XML-Format beschrieben werden muss zu welchem Gerät das Binding gehört und welche Things inklusive Channels erzeugt werden. In Java werden benötigte Binding-Handler und Konfigurationen geschrieben. Codebeispiel 2 kann entnommen werden, wie intensiv die Unterstützung durch die 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 Verwendung von openHAB

### 6.1 Beispiel Aufbau eines openHAB Smart-Home

Um mit openHAB Erfahrungen zu sammeln wurde für das Projekt ein 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.

#### 6.1.1 Hardware für openHAB

Für das Smart-Home-System wurde ein Raspberry Pi der vierten Generation mit 2gb Arbeitsspeicher gewählt. Der Raspberry ist mit Wlan und Bluetooth ausgestattet. Darauf wurde openHABian installiert, wobei es sich um ein an openHAB angepasstes Linux Buster Betriebssystem handelt. In openHABian ist die neuste Version von openHAB, nützliche System-Libraries und Einstellungen integriert, um openHAB best möglich zu unterstützen [openhabian]. Damit ist der Raspberry Pi bereits fertig eingerichtet und openHAB startet automatisch bei Systemstart des Raspberry. Die Web-Oberfläche wird im Netzwerk unter <https://openhab:8080> bereitgestellt. Dort können die meisten Einstellungen vorgenommen werden, um Geräte und Systeme einzubinden und eine Benutzeroberfläche zum Steuern des Smart-Home-System zu erstellen. Per SSH kann direkt auf den Raspberry zugegriffen werden, was für Anpassungen nötig sind, welche Raspberry bezogen sind und somit nicht über die openHAB Web-Oberfläche möglich sind. Um das Smart-Home-System leichter und jederzeit zu steuern, wird ein 7 Zoll Touch-Monitor verwendet welcher er openHAB-Panel anzuzeigt.

#### 6.1.2 Eingebundene Geräte

Für diese Arbeit wurden exemplarisch verschiedene Geräte gewählt, welche in einem Wohnzimmer üblich sein könnten.

- Hue Bridge
- Hue LED
- Osram Lightify LED

- LG Smart TV mit WebOS

Beide Lampensysteme arbeiten mit dem Zigbee Protokoll welches eine eigene Frequenz verwendet und somit viele verwalten kann [zigbee]. Dadurch kommuniziert das openHAB System z.B. nur mit der Hue-Bridge, wenn die Lampen angesprochen werden sollen. Dies passiert allerdings im Hintergrund, in der openHAB Web-Oberfläche stellt die Hue-Bridge jede Lampe als eigenes Item dar.

Bei den Lampensystemen lässt sich Helligkeit und Farbtemperatur anpassen. Der Smart TV lässt sich an-, ausschalten, Lautstärke und das Programm verändern.

### 6.1.3 Eingebundene Services

Das Smart-Home-System soll Musiksteuerung und aktuelle Wetterdaten integrieren und anzeigen. Dafür wird Spotify als Musikanbieter und openWeatherMap für Wetterdaten verwendet. Beide Dienste sind in einer kostenlosen Version nutzbar.

Spotify ermöglicht es über die Spotify-Developer-Webseite eine Web-Anwendung zu registrieren. Durch diese kann das Smart-Home-System autorisiert werden einen Spotify Account zu steuern. Das Smart-Home-System schickt dabei Befehle an Server von Spotify, welche wiederum Befehle an den registrierten Spotify Client weiterleiten [Spotify]. Es muss also nicht direkt Spotify auf dem Raspberry installiert werden. Das steigert die Performance und ist in der Handhabung angenehmer. Das Smart-Home-System soll die aktuelle Wiedergabe anzeigen, es ermöglichen Lautstärke zu verändern, den Track zu wechseln und die Wiedergabe zu starten sowie zu stoppen.

Um die Wetterdaten abzurufen ist ein Account bei openWeatherMap nötig. Für das Smart-Home-System wird die kostenlose Basisversion gewählt. Mit dem Account ist ein API-Key verknüpft, um das Smart-Home-System zu authentifizieren. Mit der Basisversion können aktuelle Wetterdaten verschiedener Standorte abrufen werden.

### 6.1.4 Integration der Geräte und Services

Die Integration der Geräte erfolgt über die Web-Oberfläche, welche openHAB bereitstellt. Für die Integration der Lampen müssen diese zuerst über die Hue-App mit der Hue-Bridge verbunden werden. Danach wird das Hue Binding installiert. Die IP Adresse der Bridge muss konfiguriert werden und das System erkennt dann die Hue-Bridge automatisch. Die Bridge kann nun als Thing in das System aufgenommen werden. Dadurch werden automatisch die Lampen als Things erstellt. Für drei Lampen werden je zwei Items angelegt, um Helligkeit und Lampenfarbe steuern zu können. Diese Items erlauben es, den aktuellen Wert auszulesen. Da die Osram Lampe das Lightify-Protokoll nutzt, welches die Hue-Bridge ebenfalls unterstützt, funktioniert die Einrichtung dort identisch.

Der LG Smart TV wird mit dem LG WebOS Binding integriert. Nach Installation wird der Smart TV von alleine erkannt, wenn er im selben Netzwerk ist. Es können dann Items erzeugt werden, um den Fernseher an- und auszuschalten und die Lautstärke zu regeln.

Spotify wird über das Spotify Binding integriert. Es muss ClientId und Secret Key hinterlegt werden, welche in der Spotify Developer Console angelegt wurden. Das Binding erstellt ein Spotify Player Bridge Thing mit welchem nun ein Spotify Account gesteuert werden kann. Es werden Items erzeugt, um den Songtitel auslesen zu können und zwei weitere um die Lautstärke und den Player zu steuern. Der Spotify Musik Player wird dabei als Media Controller Item erzeugt und kann die Wiedergabe stoppen oder starten und Lieder wechseln.



Um Wetterdaten abzurufen, wird das Weather Binding benötigt. Für dieses Binding ist eine manuelle Einrichtung per Code auf dem Raspberry nötig. Per SSH muss die services/weather.cfg angepasst werden.

```
1 apikey.openWeatherMap=sdf7g69fdgdfg679dfg69sdgkj
2 location.home.provider=openWeatherMap
3 location.home.language=de
4 location.home.updateInterval=10
5 location.home.latitude=47.8011
6 location.home.longitude=13.0448
```

Codebeispiel 3: services/weather.cfg

Wie in Codebeispiel 3 zu sehen ist, wurde der API Key für openWeatherMap hinterlegt und eine Locations names „home“ angelegt. Für diese wurde auch die Latitude und Longitude hinterlegt, sodass Wetterdaten für den Ort der Smart-Home-Systems abgerufen werden können. Der updateInterval 10 gibt an, dass die Wetterdaten alle 10 Minuten erneut geladen werden. Nach diesen Einstellungen können Items erzeugt werden.

```
1 Number    Temperature    "Temperature [%.2f °C]"
2 {weather="locationId=home, type=temperature, property=current"}
3 Number    Precip_Probability    "Precip probability [%d %%]"
4 {weather="locationId=home, type=precipitation, property=probability"}
5 Number    Wind_Speed     "Windspeed [%.2f km/h]"
6 {weather="locationId=home, type=wind, property=speed"}
```

Codebeispiel 4: items/weather.items

Die Item-Datei der Wetter Items aus Codebeispiel 4 erzeugt drei Items welche die Temperatur, die Niederschlagswahrscheinlichkeit und die Windgeschwindigkeit beinhalten. Es wird bei den Items die weatherId „home“ verwendet. Dadurch benutzt das Binding bei der Abfrage der Wetterdaten die zuvor erzeugten Ortsdaten.

### 6.1.5 Erstellung des openHAB Panels

Ein openHAB Panel kann auf der Web-Oberfläche aus verschiedenen Widgets und Items zusammengestellt werden. Dabei werden einzelne Widgets auf Dashboards angeordnet und dann mit verschiedenen Items verknüpft. Es kann z.B. ein Slider angelegt werden, welche die Helligkeit der Lampen regelt. Einzelne Items können dabei beliebig oft verwendet werden. So kann es neben dem Slider z.B. auch einen Knopf geben, der die Helligkeit zwischen 100 Prozent oder 0 Prozent regelt, wodurch ein- und ausgeschalten erzielt wird.

Für das Panel des Smart-Home-Systems werden Standardkomponenten, Widgets aus dem Community Store und auf selbst erstellte Widgets verwendet. Eigene Widgets können als Angular-Widget erstellt werden [widgets]. Abbildung 8 zeigt ein erstelltest Panel. Im Detail ist links Oben der Namen des Dashboards „Home“ und darunter die aktuelle Uhrzeit zu sehen. Letzteres ist ein Standard-Widget von openHAB. Darunter ist ein Label, welches den aktuellen Songtitel von Spotify anzeigt und ein vom Community Store eingebundener Music Controller, welcher Track sowie Lautstärke steuert.

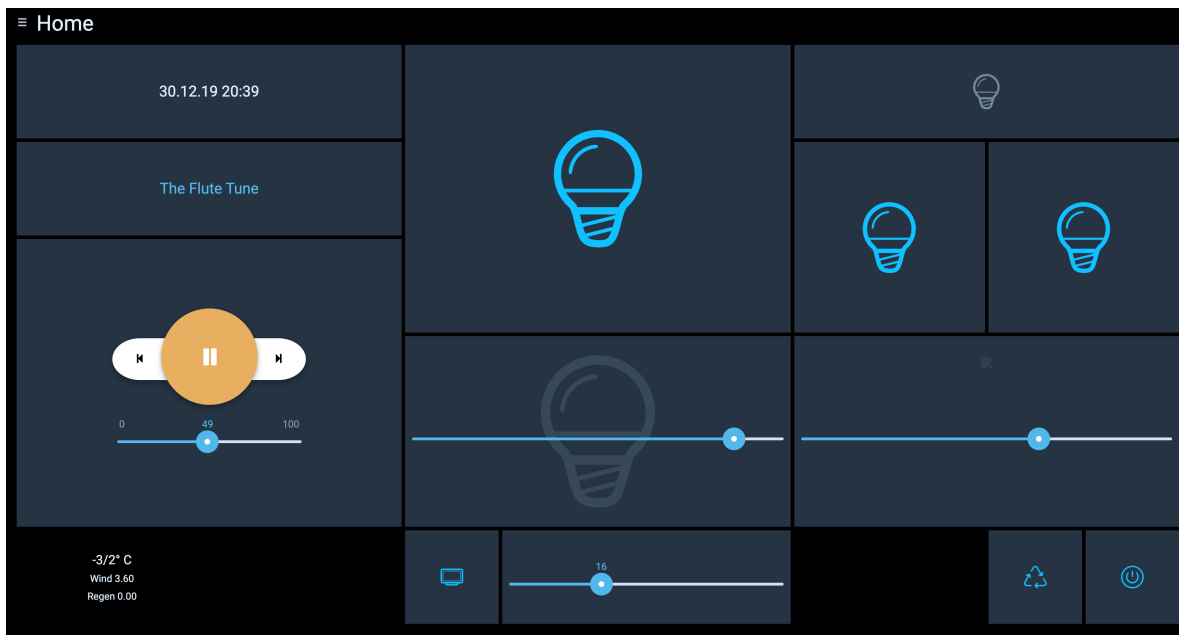


Abbildung 8: Erstelltes openHAB Panel

Unten Links befindet sich ein selbst geschriebenes Widget, welches Wetterdaten anzeigt. In der rechten unteren Ecke befinden sich zwei Schalter, welche den Raspberry neu starten und auszuschalten. Diese können über das Exec-Binding Befehle in der Kommandozeile ausführen. Unten in der Mitte befindet sich die Steuerung des Fernsehers. Der Linke Knopf schaltet ihn an und aus, der Slider regelt die Lautstärke. Die restlichen Widgets steuern die Lampen. Das große Widget oben mittig kann alle Lampen im Zimmer an- und ausschalten. Die kleinen Widget präsentieren die einzelnen Lampen. Über die verschiedenen Slider lassen sich Helligkeit und Weißfarbe der Lampen steuern.

#### 6.1.6 Kontrollstation einrichten

Um das Smart-Home-System zu steuern, wird eine Kontrollstation aus dem Raspberry Pi selbst und einem 7-Zoll Bildschirm errichtet. Es wird dafür ein Bildschirm gewählt, auf welchem sich der Raspberry befestigen lässt und direkt vom Raspberry Strom zu beziehen. Die Kontrollstation soll das erstellte openHAB Panel aus Abbildung 8 anzeigen und so Zugriff auf das System ermöglichen.

Um auf dem Raspberry Pi eine Benutzeroberfläche anzeigen zu können, muss die Bibliothek PIXEL installiert werden. Mit PIXEL wird der Browser Chromium ausgeliefert. Es müssen verschiedene Einstellungen vorgenommen werden, damit der Raspberry Pi ohne Login startet und direkt Chromium öffnet. Gleichzeitig soll der Mauszeiger ausgeblendet werden und das Panel im Vollbildmodus anzuzeigen. Die nötigen Einstellungen können nicht mit openHAB erledigt werden. Per SSH müssen diese direkt auf dem Raspberry Pi vorgenommen werden.

#### 6.1.7 Externer Zugriff

Eine häufige Anforderung an Smart-Home-Systeme ist Extern auf das Smart-Home-System zugreifen zu können um Zustand von Geräten zu kontrolliert und diese auch zusteuern. Hierfür bietet openHAB die myopenHAB Cloud. Um diese zu nutzen, muss das openHAB Cloud Connector Add-On installiert werden. Dieses erzeugt auf dem Raspberry ein Secret-Datei

mit einem Secret-Key. Dieser Key und eine von openHAB erzeugte UUID muss in einen myopenHAB-Account hinterlegt werden. Danach ermöglicht das Add-On eine Verbindung nach außen zur myopenHAB Cloud. In myopenHAB können erstellte Panels angezeigt werden mit denen das Gerät wie gewohnt gesteuert wird. Auf das Smart-Home-System kann nun von außen zugegriffen werden, wie als wäre der Benutzer im selben Netzwerk. Es gibt für die myopenHAB Cloud Smartphone Apps für Android und iOS. Nach der Anmeldung mit dem myopenHAB Account ist ein sichere Zugriff auf das System auch von unterwegs möglich. Dabei kann auf Panels zugegriffen werden und Items gesteuert werden. Allerdings sollte das Panel auf die Größe des Smartphone Bildschirms angepasst werden, oder eigene Panels für Mobile Geräte erstellt werden.

## 6.2 Verteilungsarchitektur des Smart Home Systems

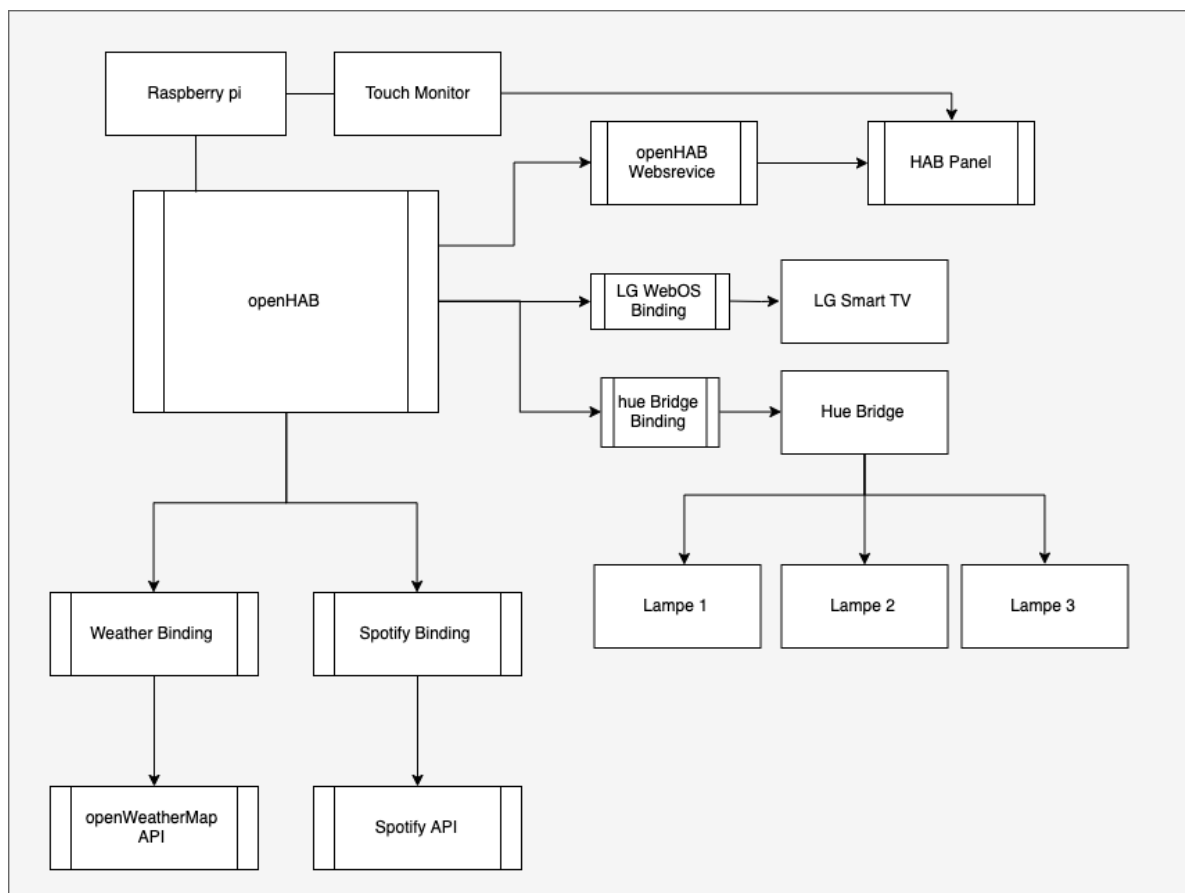


Abbildung 9: Verteilungsarchitektur des Smart Home Systems

Abbildung 9 stellt die Verteilungsarchitektur des erstellten Smart-Home-Systems dar. In der Darstellung sind Hardwarekomponenten durch einfache Rechtecke dargestellt und Softwarekomponenten durch Rechtecke mit zwei vertikalen Linien. Die Verteilungsarchitektur zeigt die verschiedenen Geräte und Services und wie sie mit openHAB verbunden sind. Dabei sind alle Services und Geräte über Bindings verbunden. Der Touch-Monitor ist direkt über HDMI mit dem Raspberry Pi verbunden und zeigt ein erstelltes openHAB Panel an, welches von der Web-Oberfläche des openHAB Systems bereitgestellt wird. Das Hue-Binding spricht die Hue-Bridge an, welche die Lampen steuert.

## 6.3 Integration der Big Player

Amazon Alexa und Google Home sind die bekanntesten Voice Services laut Google Trends. [GOOb] Damit ist es möglich über verbale Kommandos bestimmte Funktionalitäten anzusteuern. Beispielsweise das abspielen eines Radiosenders oder das An- und Ausschalten einer Lampe.

Sowohl der Google Home Assistent als auch Amazons Alexa sind in das Heimautomatisierungstool openHAB integrierbar. In den folgenden zwei Unterkapiteln wird dies genauer erläutert.[?]

### 6.3.1 Amazon Alexa

In openHAB kann ein zertifizierter Amazon Alexa Smart-Home-Skill verwendet werden. Damit ist es möglich eine Smart-Home durch verbale Kommandos zu verwenden. Es können dadurch Lichter, Schlösser, Thermostate, Sensoren und viele weitere Geräte angesteuert werden. Dies ist möglich durch Alexa oder den Echo Dot.[ALEa][ALEb] Um die Geräte von Amazon als Eingabe nutzen zu können, wird zusätzlich die Verbindung zur myopenHAB Cloud benötigt, da diese die Verarbeitung der Kommandos an Amazon weiterleitet.

Seit der Version 2.5 nutzt der Alexa Skill von openHAB die neueste Smart-Home-Skill-API V3. Dadurch können nun viele Smart-Home Geräte deutlich vielseitiger und präziser angesprochen werden. Außerdem unterstützt der Skill nun alle 15 Sprachen, in denen Alexa verwendet werden kann. Weiterhin ist Abwärtskompatibilität zu früheren Versionen gewährleistet.[?]

### 6.3.2 Google Home

Prinzipiell sind mithilfe von Google Home alle Optionen gegeben, die auch durch Amazons Alexa existieren. Somit können ebenfalls verschiedenste Geräte angesteuert werden. Für die Verwendung wird, wie beim Amazon Skill, eine Verbindung mit der myopenHAB Cloud benötigt.

Beim letzten Update von openHAB wurde die Anwendung von Google Home ebenfalls verbessert. Dabei ist sowohl die Codequalität als auch die Sicherheit der Integration gesteigert worden. Weiterhin ist es durch die neueste Version möglich, deutlich mehr Funktionen der Google Smart-Home-API zu nutzen. Dabei wurden neben neue Funktionen der Support für alle Sprachen eingebaut.[?][GOOa]

## 7 Datenintegrität und Sicherheit

Dieses Kapitel bietet einen Überblick über die Sicherheit im Internet-of-Things-Bereich (IoT). Anschließend wird auf die Sicherheit in openHAB selbst und dafür verwendbare Erweiterungen eingegangen. Dies beinhaltet mögliche Sicherheitslücken und dazugehörige Gegenmaßnahmen.

Open Web Application Security Projekt (OWASP), eine Community, die frei erhältliche Artikel zur Sicherheit im Internet bereitstellt, veröffentlichte 2019 eine Liste der Top Sicherheitsbedrohungen aus dem Jahr 2018 im IoT-Bereich.[OWA] Das dabei größte Sicherheitsrisiko besteht bei schwachen, einfach zu ratenden oder hart-kodierten Passwörtern. Dies wird von OWASP beschrieben mit „Use of easily bruteforced, publicly available, or unchangeable credentials, including backdoors in firmware or client software that grants unauthorized access to deployed

systems“.[OWA] Auf Platz 2 und 3 befinden sich unsichere Netzwerk Services und Ecosystem Interfaces. Damit sind indirekte Zugriffsmöglichkeiten durch ungesicherte Services gemeint, die entweder von innerhalb oder außerhalb des IoT-Netzwerks erreichbar sind. Sobald ein Angreifer diesen Service „übernommen“ hat, kann dieser problemlos Zugriff erlangen, wenn keine Sicherheitsvorkehrungen diesbezüglich getroffen wurden.

openHAB in der Version 2.4 wird in der Masterarbeit von Jesús Antonio Soto Velázquez über „Securing openHAB Smart Home through User Authentication and Authorization“, auf Sicherheit geprüft.[Vel] Dabei untersucht der Autor mithilfe von Wireshark, wie openHAB kommuniziert und kommt zu dem Resultat, dass es drei verschiedene Kommunikationsszenarien gibt, die sicherheitstechnische Relevanz haben. Alle drei Szenarien nutzen Bindings als Mittel der Kommunikation. Diese sind:

- intern: Thing - openHAB
- extern mit logischem Thing: openHAB-Remote Web Service
- extern mit physikalischem Thing: openHAB-Remote Web Service Thing

Interne Kommunikation zwischen Thing und openHAB-Instanz findet typischerweise über WLAN statt, welches mithilfe von AES verschlüsselt ist. Um in das System einzudringen, müsste ein Angreifer entweder AES entschlüsseln können oder das Passwort anderweitig herausfinden. AES gilt als nicht entschlüsselbar in einer annehmbaren Zeit. Daraus lässt sich schließen, dass die interne Kommunikation von openHAB und Things nur so sicher ist, wie das ganze WLAN-Netzwerk selbst.

Bei den zwei externen Punkten sind laut Jesús Antonio Soto Velázquez nur Lauschangriffe möglich. Normalerweise sollten externe Thing's mit openHAB und myopenHAB Cloud ausschließlich über das Hyper Transfer Protocol Secure (HTTPS) miteinander kommunizieren. Dabei hängt es von der jeweiligen Implementierung von Binding und Aufrufen für die Datenübermittlung ab, ob HTTPS oder dessen unverschlüsselte Form HTTP eingesetzt wird. Weiterhin weißt Jesús Antonio Soto Velázquez darauf hin, dass sobald ein Binding Daten unverschlüsselt über das Internet verschickt, andere Bindings davon nicht betroffen sind. Diese gelten weiterhin als sicher, solange der Angreifer nicht über andere Wege in das System eingedrungen ist.

Trotz der Untersuchung der nicht ganz aktuellen Version 2.4, ist der Inhalt dieser Masterarbeit aktuell, denn mit der Version 2.5 wurden kaum Updates bezüglich Sicherheit integriert.[?] Folglich wird auf die sowohl internen als auch externen Sicherheits-Features von openHAB detaillierter eingegangen.

## 7.1 Interne vorhandene Funktionalitäten

openHAB bietet bereits Sicherheitsvorkehrungen „out-of-box“ an. Grundsätzlich kann die openHAB-Instanz nur über Secure Shell (SSH) oder HTTP/HTTPS erreicht werden. Bei der Kommunikation über SSH wird, wie bereits erwähnt, das als sicher geltende AES verwendet. HTTPS hingegen nutzt Zertifikate und wird zur Herstellung von Integrität und Vertraulichkeit der Kommunikation zwischen Client und Server im Internet verwendet. Es gilt ebenfalls als sicher.

Beim ersten Aufsetzen von openHAB wird ein selbst-signiertes SSL-Zertifikat (256-bit ECC) erstellt und im Jetty (Webservice) Keystore gespeichert. Dadurch kann sichergestellt werden,

dass jede openHAB-Instanz eindeutig ist. Ein potenzieller Angreifer kann demzufolge keinen Evil-Twin (Böser Zwilling) aufbauen und ein openHAB System nachahmen.

Die Abbildung 10 zeigt eine Verteilungsarchitektur mit den verwendeten Protokolle einzelner openHAB-Komponenten aus Kapitel 6.

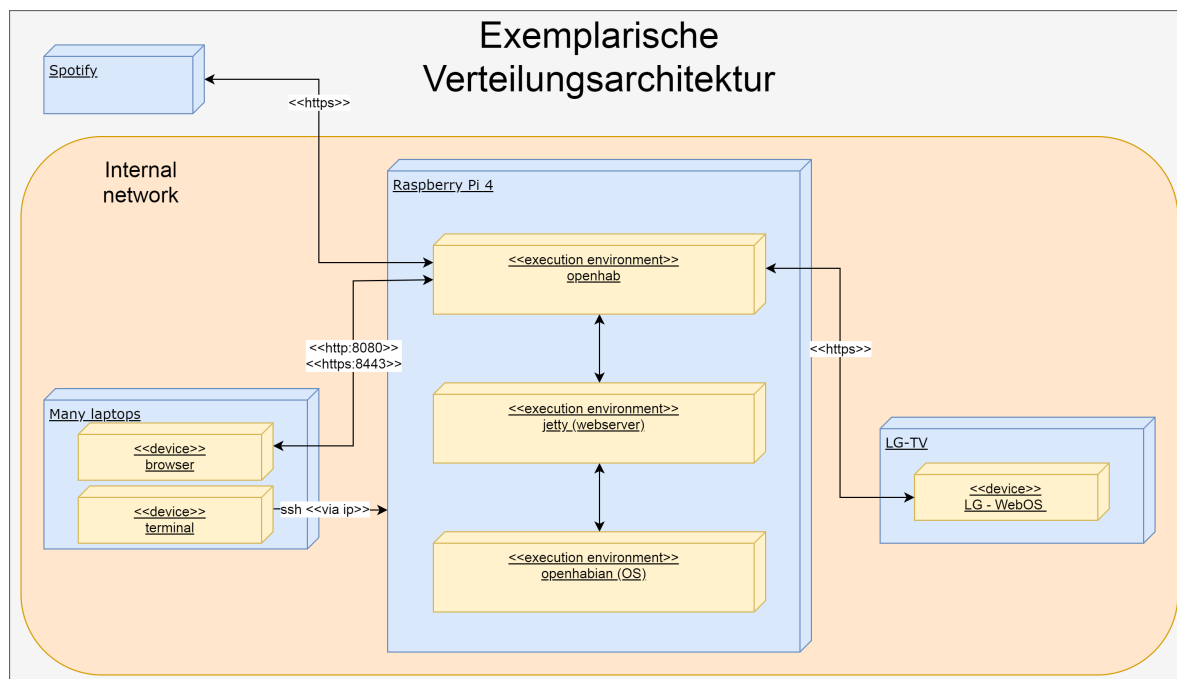


Abbildung 10: Übersicht einer exemplarischen Anwendung von openHAB

Die darin zu sehenden Komponenten sind durch einen blauen Hintergrund markiert. Die jeweils darauf laufende Software ist durch die Farbe Gelb dargestellt. Alles bis auf Spotify ist im internen Netzwerk abgebildet (orange eingegrenzten Bereich).

Die jeweiligen Komponenten kommunizieren mit der zentralen openHAB-Instanz, welche auf einem Raspberry Pi gehostet wird. Dabei sind die verwendeten Protokolle für die Informationsübertragung auf den gerichteten Pfeilen zu sehen.

## 7.2 Externe Erweiterungsmöglichkeiten

Ein weiterer Weg der Benutzung einer openHAB-Instanz wäre dies von außerhalb des Heimnetzwerks über beispielsweise das Smartphone zu erreichen. Hierfür werden die „Best Practices“ in der openHAB Dokumentation erläutert.

Dabei wird eine VPN Verbindung in das Heimnetzwerk als sicherster Weg bewertet. Den openHAB Cloud Service zu nutzen, ist eine weitere Möglichkeit. Dabei wird ebenfalls eine Tunnelverbindung aufgebaut, wodurch openHAB durch die myopenHAB Cloud angesteuert wird.

Schlussendlich kann openHAB hinter einem Reverse Proxy betrieben werden. Dieser leitet Client-Anfragen weiter an den gewünschten Server. Damit ist es möglich über den Proxy auf die im Heimnetzwerk eingesetzt openHAB-Instanz über eine bestimmte Domain zuzugreifen.

Weiterhin warnt openHAB auf ihrer Website davor, Ports an der Firewall für openHAB

zu öffnen. Es wird ausdrücklich darauf hingewiesen, dass dies in keinem Fall benötigt wird.[?]

## 8 Fazit

### 8.1 Zusammenfassung

Diese Arbeit analysiert das Open-Source Heimautomatisierungstool openHAB auf Markttauglichkeit und Benutzbarkeit. Zu Beginn wird dafür erklärt, was openHAB ist und wofür es verwendet werden soll. Anschließend folgt eine Bewertung des Projektes mithilfe von Open Hub und einer Facharbeit. Das daraus resultierende Ergebnis besagt, dass openHAB ein gutes Heimautomatisierungstool ist und auch im Vergleich zu seinesgleichen auf dem Markt etabliert ist.

Weiterhin werden die technischen Grundlagen, Programmier-Schnittstellen und mehrere Anwendungsbeispiele in den darauffolgenden Kapiteln erläutert.

Abschließend geht diese Arbeit auf Datenintegrität und Sicherheit ein.

### 8.2 Schlussfolgerung

Heimautomatisierungstools werden immer notwendiger, um mit dem wachsenden Markt im Bereich IoT mithalten zu können.[Ten] Das bekannte Open-Source Tool openHAB bietet die fundamentalen Funktionalitäten für Heimautomatisierung. Dies bekräftigt auch das Ergebnis dieser Arbeit. Dabei wurde festgestellt, dass openHAB aktiv von einer wachsenden Community weiterentwickelt wird und sowohl privat als auch kommerziell eingesetzt werden kann, da das Projekt unter einer EPL-2.0 Lizenz steht.

Die Komponenten aus denen openHAB besteht, bzw. die Komponenten, welche einem Benutzer zur Verfügung stehen, sind klar strukturiert und besitzen eine Abstraktionsstufe, die gut verständlich ist. Auch wenn die Namensgebung der Items nicht optimal ist, wird ersichtlich wie der Zusammenhang zwischen Bindings, Things, Channels und Items ist. openHAB bietet durch die Add-On Architektur einige Möglichkeiten der Erweiterung. Dies wird im Falle von Bindings noch durch die bereitgestellten Skelett-Skripte vereinfacht, was dazu führt, dass openHAB relativ einfach um neue Geräte erweitert werden kann.

Weiterhin wurde festgestellt, dass openHAB selbst sichere Verbindungen beim Austausch von Informationen über Geräte nutzt. Dabei wird für Zugriffe von innerhalb und außerhalb des Netzwerks auf die Technologien HTTPS und SSH gesetzt. Das einzige aktuelle Sicherheitsproblem ist, dass ältere integrierbare Bindings, welche meist durch Dritte geschrieben wurde, HTTP verwenden. Da sowohl HTTPS als auch SSH als sicher gelten, sind nur über die unsicheren HTTP Verbindungen Lauschangriffe möglich, sofern sich keine anderen Lücken im System befinden.

Die Autoren dieser Arbeit bewerten openHAB als gut gelungenes Heimautomatisierungstool. Nach einer gewissen Einarbeitungszeit ist das Zusammenbauen eines Smart-Home damit unkompliziert umsetzbar. Es erlaubt die Einbindung fast aller Geräten aus dem IoT-Bereich, erfordert jedoch an manchen Stellen Kenntnisse in der Softwareentwicklung.

### 8.3 Ausblick

Mit dem Release 2.5 Ende 2019 ist offiziell die Entwicklung an Version 3.0 gestartet. Version 2.5 wird allerdings weiter gepflegt werden. Für die Version 3 sind einige Breaking Changes geplant. So soll vor allem die verschiedenen UI's (Paper UI, Basic UI, und mehr) zusammengeführt

werden und das erstellen von Rules wird überarbeitet.[?] Da das Projekt von der Community entwickelt wird kann sich an diesem Plan etwas ändern. Der Ausblick lässt aber vermuten, dass bereits jetzt ein guter Schwerpunkt gelegt wurde und openHAB mit der nächsten Version einen weiteren Schritt nach vorne machen wird. Es steht derzeit noch kein Release Datum fest. Da wichtige Releases in der Vergangenheit oft am Ende des Jahres zur Weihnachtszeit veröffentlicht wurden, liegt es nahe, dass frühestens Ende 2020 damit zu rechnen ist.



## Literatur

- [openhabian] openhabian. <https://github.com/openhab/openhabian>. Last visit: 06 Jan 2020.
- [zigbee] zigbee. <https://zigbeealliance.org/why-zigbee/>. Last visit: 06 Jan 2020.
- [Spotify] spotify. <https://www.openhab.org/addons/bindings/spotify/>. Last visit: 06 Jan 2020.
- [widgets] widget. <https://www.openhab.org/docs/configuration/habpanel.html>. Last visit: 06 Jan 2020.
- [ALEa] Amazon Alexa Smart Home Skill. <https://www.openhab.org/docs/ecosystem/alexa/>. Last visit: 04 Jan 2020.
- [ALEb] Amazon Echo Control Binding. <https://www.openhab.org/addons/bindings/amazonechocontrol/>. Last visit: 04 Jan 2020.
- [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](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.
- [GOOa] Google Assistant Action. <https://www.openhab.org/docs/ecosystem/google-assistant/>. Last visit: 04 Jan 2020.
- [GOOb] Google Trends Vergleichen: amazon alexa, Google Home, HomePod, Microsoft Invoke. <https://trends.google.de/trends/explore?q=amazon%20alexa,%2Fg%2F11cn9xkmpf,%2Fg%2F11dfjbltv9,Microsoft%20Invoke>. Last visit: 04 Jan 2020.
- [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] Securing access to openHAB. <https://www.openhab.org/docs/installation/security.html>. Last visit: 29 Dez 2019.
- [OPEc] Develop a NEW binding. <https://www.openhab.org/docs/developer/#develop-a-new-binding>. Last visit: 22 Dez 2019.
- [OPEd] Developer Guide. <https://www.openhab.org/developer>. Last visit: 22 Dez 2019.
- [OPEe] openHAB - Home Page. <https://www.openhab.org>. Last visit: 20 Dez 2019.

- [OPEf] openHAB - empowering the smart home. <https://www.openhub.net/p/openhab>. Last visit: 23 Dez 2019.
- [OWA] OWASP Internet of Things Project. [https://www.owasp.org/index.php/OWASP\\_Internet\\_of\\_Things\\_Project](https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project). Last visit: 29 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.
- [Vel] J. A. S. Velázquez. Securing openHAB Smart Home through User Authentication and Authorization. [https://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=7&ved=2ahUKEwjq\\_c-k0s7mAhXOIVAKHS4dDYgQFjAGegQICBAC&url=https%3A%2F%2Fcomserv.cs.ut.ee%2Fhome%2Ffiles%2FSoto\\_computer\\_science\\_2018.pdf%3Fstudy%3DATILoputoo%26reference%3DB7FBB08D43717164067F1F9DBA92B98A65913AC9&usg=AOvVaw0631sYhHX3wKxW8kPwxpzE](https://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=7&ved=2ahUKEwjq_c-k0s7mAhXOIVAKHS4dDYgQFjAGegQICBAC&url=https%3A%2F%2Fcomserv.cs.ut.ee%2Fhome%2Ffiles%2FSoto_computer_science_2018.pdf%3Fstudy%3DATILoputoo%26reference%3DB7FBB08D43717164067F1F9DBA92B98A65913AC9&usg=AOvVaw0631sYhHX3wKxW8kPwxpzE). Last visit: 29 Dez 2019.