Technische Hochschule Rosenheim

# Smart Home mit Open Home Automation Bus (OpenHAB)

Lukas Kiederle
Dominik Ampletzer
Daniel Böning
Fakultät für Informatik

WS 2019/20

# Inhaltsverzeichnis

# 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 weißt 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?<br>• Mindestens 1 Release im letzten Jahr<br>• Mindestens 3 aktive Contributer<br>• 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?<br>• Api-Dokumentation<br>• Getting Started Tutorial<br>• Aktuelle Dokumentation |

Tabelle 1: OpenHAB Projektbewertungskriterien

Der Online-Dienst **Open Hub**, ehemals **Ohloh** genannt, katalogisiert open-source Softwareprojekte. Für jedes Projekt werden Daten wie Name, 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 fließen auch Daten weiterer open-source Projekte ein, 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 durchschnittlichen Wert aller Java open-source Projekte, welche auf Open Hub registriert sind.[OPEf] Die hierbei angegeben 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?<br>• Mindestens 1 Release im letzten Jahr<br>• Mindestens 3 aktive Contributer<br>• 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?<br>• Api-Dokumentation<br>• Getting Started Tutorial<br>• 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 Komponenten von openHAB

In diesem Kapitel werden die grundlegenden Komponenten dargestellt, welche openHAB verwendet. Des Weiteren 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 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 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 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 [OPEd]:

- 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 tranformieren/modfizieren. 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 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. 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.

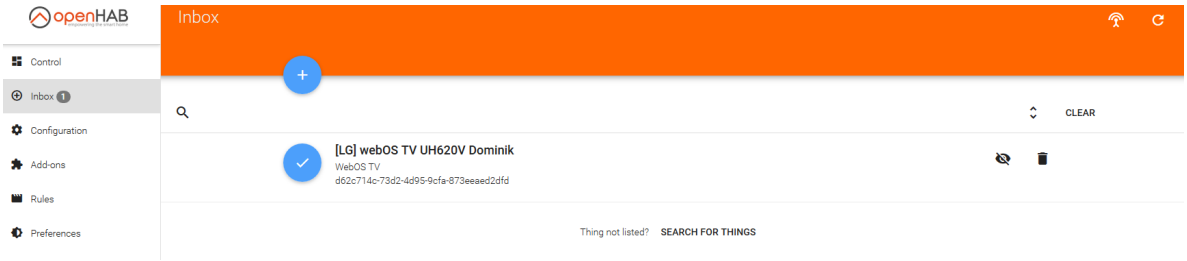


Abbildung 2: Thing per Web-Oberfläche

- 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 noch nachträglich geändert werden.

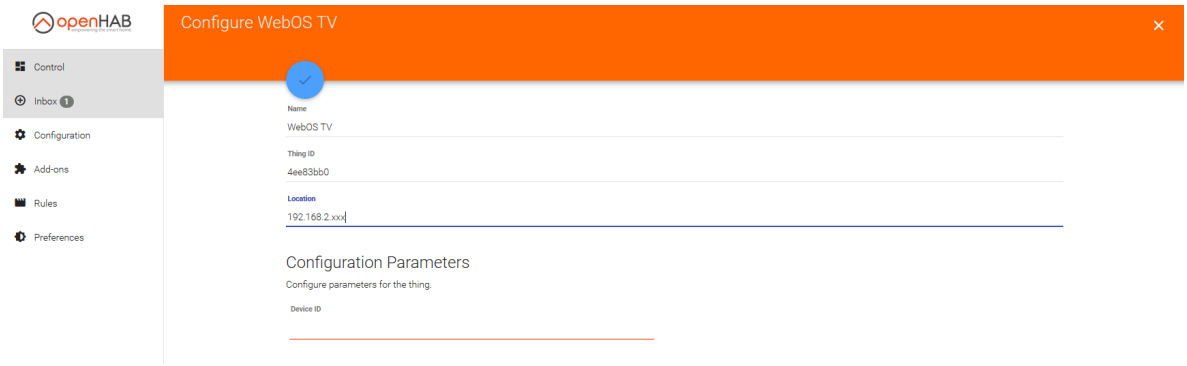


Abbildung 3: Thing per Web-Oberfläche Manuell

- 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.

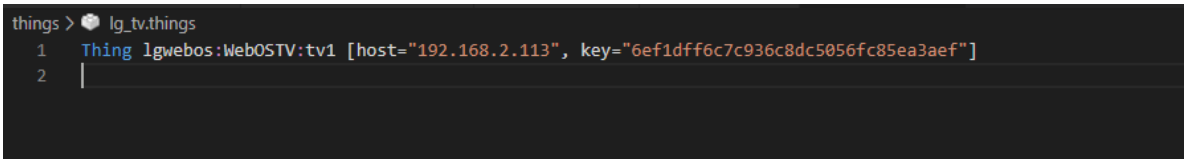


Abbildung 4: Thing per Code

## 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.

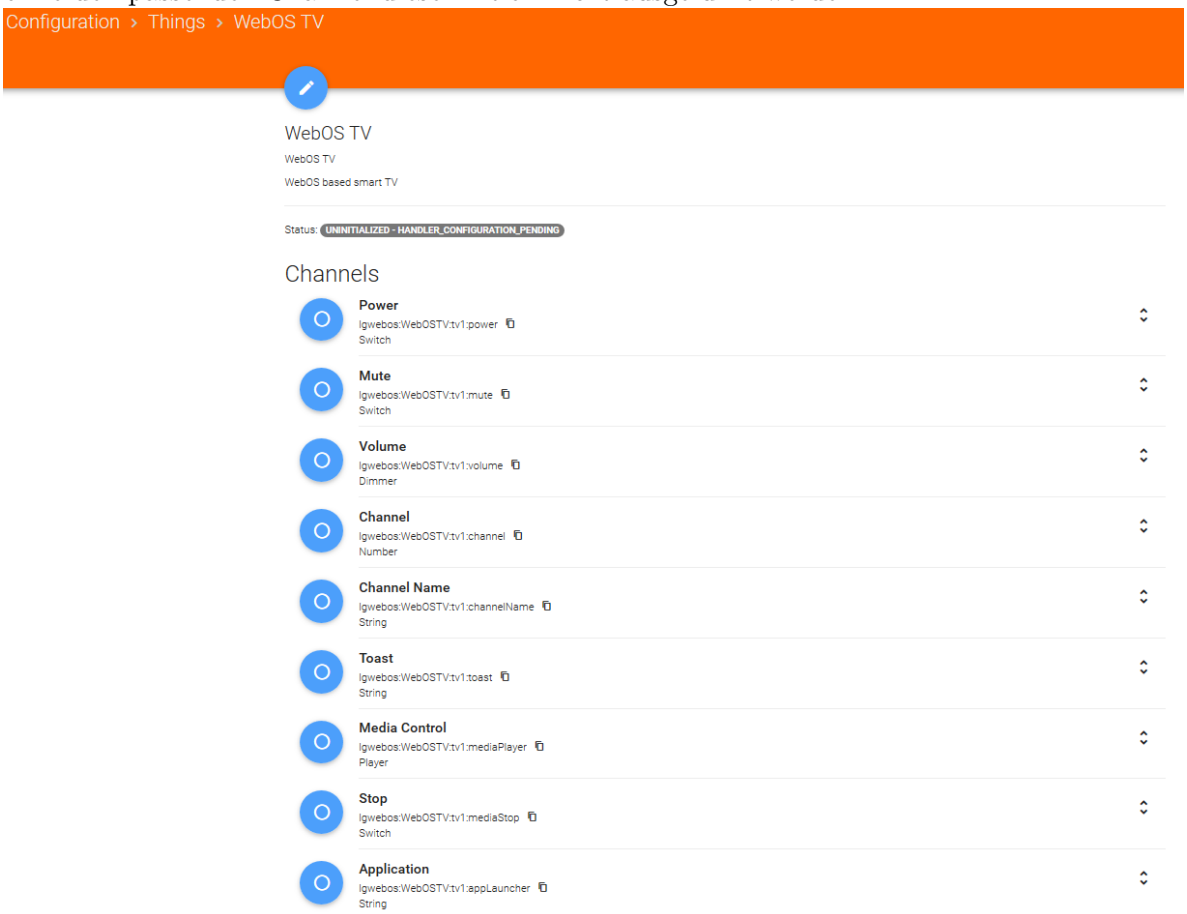


Abbildung 5: Channel Liste

## 4.5 Items

Der Begriff Item oder zu deutsch 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. 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 genutz werde 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**>]. 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 Homes. 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 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.

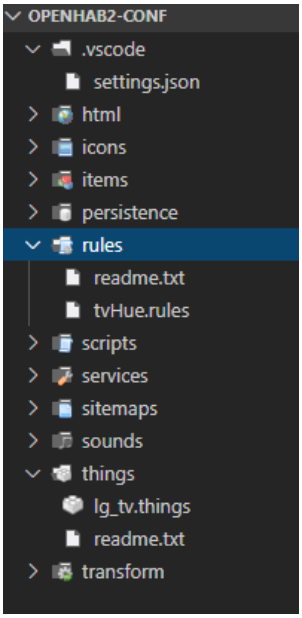


Abbildung 6: openHAB-conf Ordnerstruktur

## 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 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 [OPEc], 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.

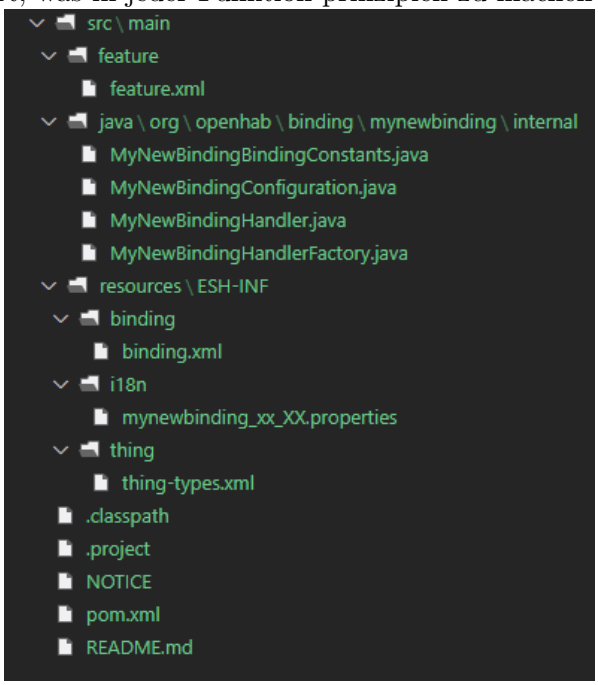


Abbildung 7: Skelett für Binding

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.

```
xxxBindingHandler.java
...
@Override
public void handleCommand(ChannelUID channelUID, Command command) {
if (CHANNEL_1.equals(channelUID.getId())) {
if (command instanceof RefreshType) {
// TODO: handle data refresh
}

// TODO: handle command

// Note: if communication with thing fails for some reason,
// indicate that by setting the status with detail information:
// updateStatus(ThingStatus.OFFLINE, ThingStatusDetail.
   COMMUNICATION_ERROR,
// "Could not control device at IP address x.x.x.x");
}
}

@Override
public void initialize() {
// logger.debug("Start initializing!");
config = getConfigAs(MyNewBindingConfiguration.class);

// TODO: Initialize the handler.
// The framework requires you to return from this method quickly.
   Also, before leaving this method a thing
// status from one of ONLINE, OFFLINE or UNKNOWN must be set. This
   might already be the real thing status in
...
```

Codebeispiel 2: Handler.java Ausschnitt

# 6 Verwendung von OpenHAB

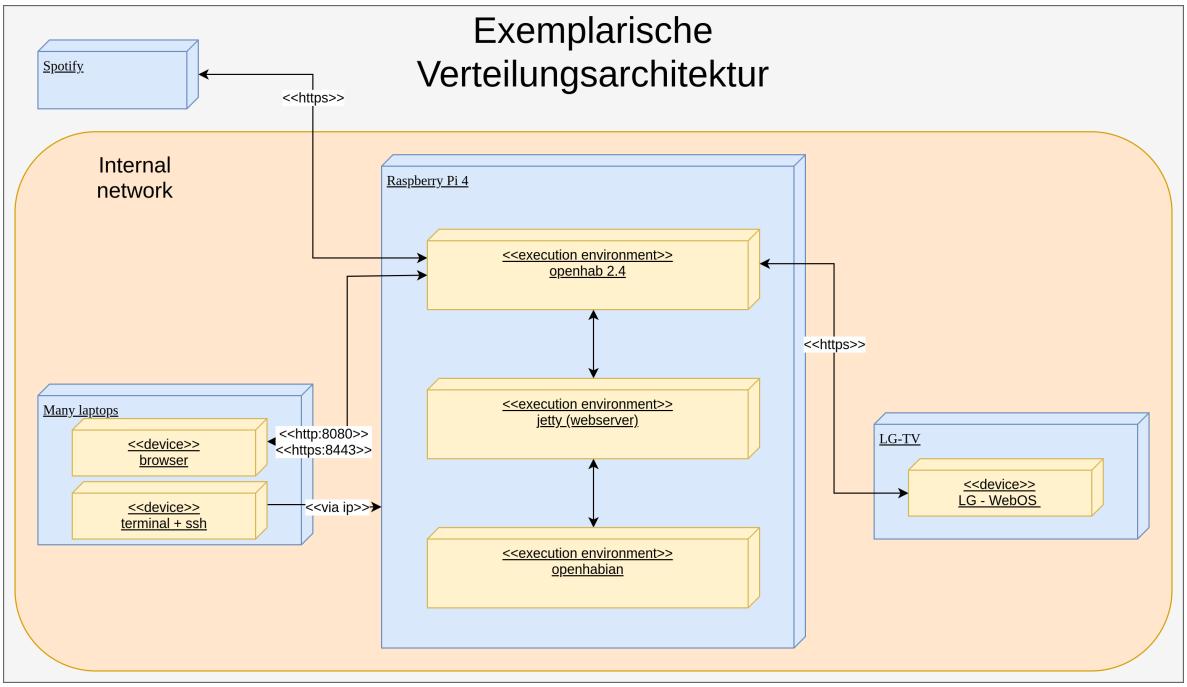


Abbildung 8: Übersicht einer exemplarischen Anwendung von OpenHAB

## 6.1 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.

### 6.1.1 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. Damit wir unser System später leicht bedienen können, wird ein 7 Zoll Touch-Monitor verwendet um ein HAB Panel anzuzeigen. Dank Touch-Funktionalität funktioniert er als Steuersystem und unser System kann jederzeit bedient werden.

### 6.1.2 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.

### 6.1.3 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.

### 6.1.4 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.
Den LG Smart TV integrieren wir mit dem LG WebOS Binding. Nach Installation wird der Fernseher von alleine erkannt wenn er im selben Netzwerk ist. Wir können dann Items erzeugen, um diesen an- und auszuschalten und die Lautstärke zu regeln.

Spotify integrieren wir über das Spotify Binding. Wir hinterlegen unsere ClientId und Secret Key, welche wir in der Spotify Developer Console angelegt haben. Das Binding erstellt uns ein Spotify Player Bridge Thing mit welchem wir nun unseren Account steuern können. Wir erzeugen Items, um den Songtitel auslesen zu können und zwei weitere um die Lautstärke und den Player zu kontrollieren. Der Spotify Musik Player wird dabei als Media Controler Item erzeugt und kann die Wiedergabe stoppen oder starten und Lieder wechseln.

Um unsere Wetterdaten abrufen zu können, brauchen wir das Weather Binding. Für dieses Binding ist eine manuelle Einrichtung auf dem Raspberry nötig. Per SSH muss die services/weather.cfg angepasst werden:

```
apikey.openWeatherMap=sdf7g69fdgdfg679dfg69sdgkj
location.home.provider=openWeatherMap
location.home.language=de
location.home.updateInterval=10
location.home.latitude=47.8011
location.home.longitude=13.0448
```

Codebeispiel 3: services/weather.cfg

Dabei wurde der API Key für openWeatherMap hinterlegt und eine Locations names „home“ angelegt. Für diese wurde auch die Latitude und Longitude hinterlegt, sodass unsere Wetterdaten für diesen Ort abgerufen werden können. Der updateInterval 10 gibt an, dass die Wetterdaten alle 10 Minuten erneut geladen werden. Nach diesen Einstellungen können wir Items erzeugen.

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

Codebeispiel 4: items/weather.items

Daraus erzeugt openHAB drei Items welche die Temperatur, die Niederschlagswahrscheinlichkeit und die Windgeschwindigkeit beinhalten. Wir verwenden bei unseren Items die weatherId „home“. Dadurch benutzt das Binding bei der Abfrage der Daten die zuvor erzeugten Ortsdaten.

### 6.1.5 Erstellung des HAB Panels

Ein HAB Panel kann auf der Web UI 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 noch einen Button geben, der die Helligkeit zwischen 100 Prozent oder 0 Prozent wechselt. Damit kann die Lampe ein und ausgeschaltet werden.

Für unser Panel greifen wir auf Standardkomponenten, Widget aus dem Community Store und auf selbst erstellte Widgets zurück. Eigene Widgets können als Angular Widget erstellt werden.

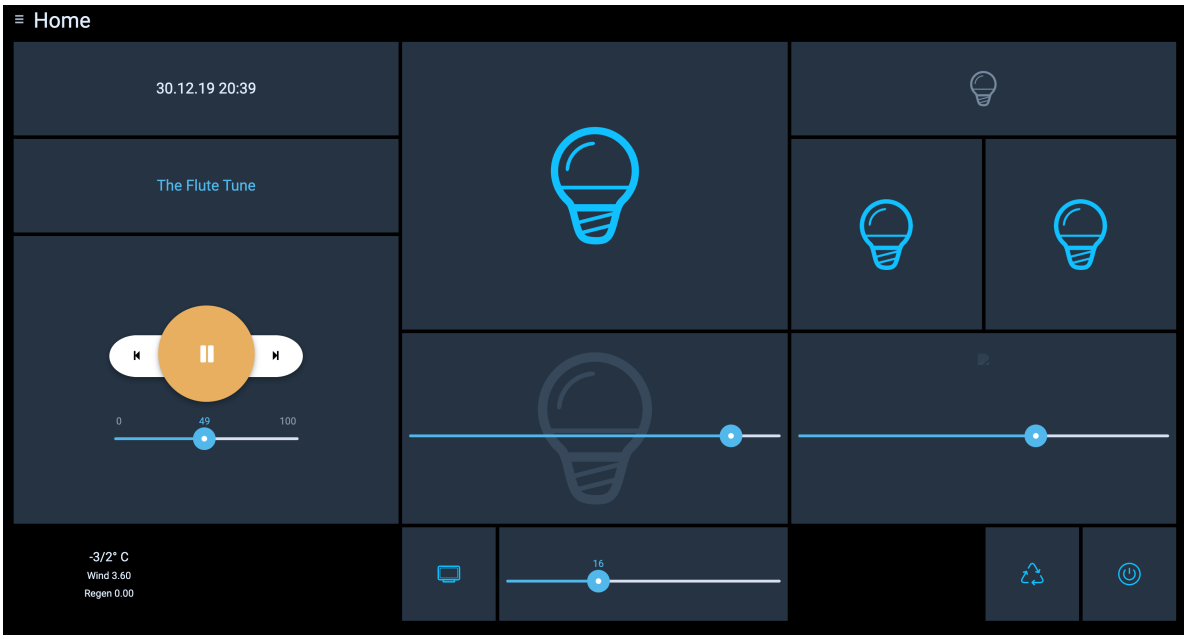


Abbildung 9: Erstelltes OpenHab Panel

In unserem Panel sehen wir zuerst den Namen des Dashboardes „Home“ und darunter die aktuelle Uhrzeit dargestellt. Letzteres ist ein Standardwidget von openHab. Darunter ist ein Label, welches den aktuellen Songtitel von Spotify anzeigt und ein vom Community Store eingebundener Music Controller, welcher Track und Lautstärke von Spotify steuern kann.

Ganz unten sehen wir ein selbst geschriebenes Widget, welches die Wetterdaten anzeigt.

In der rechten unteren Ecke befinden sich zwei Buttons, um den Raspberry neuzustarten oder auszuschalten. Diese können über das Exec Binding Befehle in der Kommandozeile ausführen.

Unten in der Mitte befindet sich die Steuerung des Fernsehers. Der Linke Button schaltet ihn an und aus, der Slide regelt die Lautstärke.

Alle restlichen Widgets steuern die Lampen. Das große Widget kann alle Lampen im Zimmer an und ausschalten. Die kleinen Widget präsentieren die einzelnen Lampen. Über die Slider lässt sich die Helligkeit und Weißfarbe aller Lampen einstellen.

### 6.1.6 Kontrollstation einrichten

Um unser System bequem steuern zu können, wollen wir eine Kontrollstation aus dem Raspberry pi selbst und einem 7 Zoll Bildschirm errichten. Wir wählen dafür einen Bildschirm, auf welchem sich der Raspberry befestigen lässt und direkt vom Raspberry Strom bezieht. Die Kontrollstation wird dann unser HAB Panel anzeigen und uns so Zugriff auf unser System ermöglichen.

Um auf dem Raspberry pi eine Benutzeroberfläche anzeigen zu können, installieren wir PIXEL. Dabei wird der Browser Chromium mit ausgeliefert. Wir müssen verschiedene Einstellungen vornehmen damit unser Raspberry pi ohne Login startet und direkt Chromium öffnet. Gleichzeitig soll der Mauszeiger ausgeblendet werden und unser Panel im Vollbildmodus anzeigen. Alle Einstellungen können nicht mit openHAB erledigt werden. Per SSH müssen diese direkt auf dem Raspberry pi eingestellt werden.

### 6.1.7 Externer Zugriff

Eine häufige Anforderung an Smart Home Systeme ist auch extern auf das System zugreifen zu können. So kann auch von unterwegs der Zustand der Geräte kontrolliert werden und diese auch gesteuert werden. Für openHAB gibt es die myOpenHab Cloud. Für diese installiert man zuerst das open Hab Cloud Connector Addon. Dieses erzeugt auf dem Pi ein Secret File mit einem Secret Key. Registriert man einen Account bei myOpenHab, muss man diesen Secret-Key und ein von openHab erzeugte UUID hinterlegen. Danach ermöglicht das Addon eine Verbindung nach außen. In myOpenHab können erstellte Panels angezeigt werden mit denen das Gerät wie gewohnt gesteuert werden kann. Das ganze System kann jetzt auch extern bearbeitet werden. MyOpenHab gibt dabei die Möglichkeit auf das System zuzugreifen als wäre man im selben Netzwerk. Es gibt für die myOpenHab Cloud auch 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 zugriffen werden und Items direkt gesteuert werden. Allerdings sollte das Panel auf die Größe des Smartphone Bildschirms angepasst werden.

## 6.2 Verteilungsarchitektur unseres Smart Home Systems

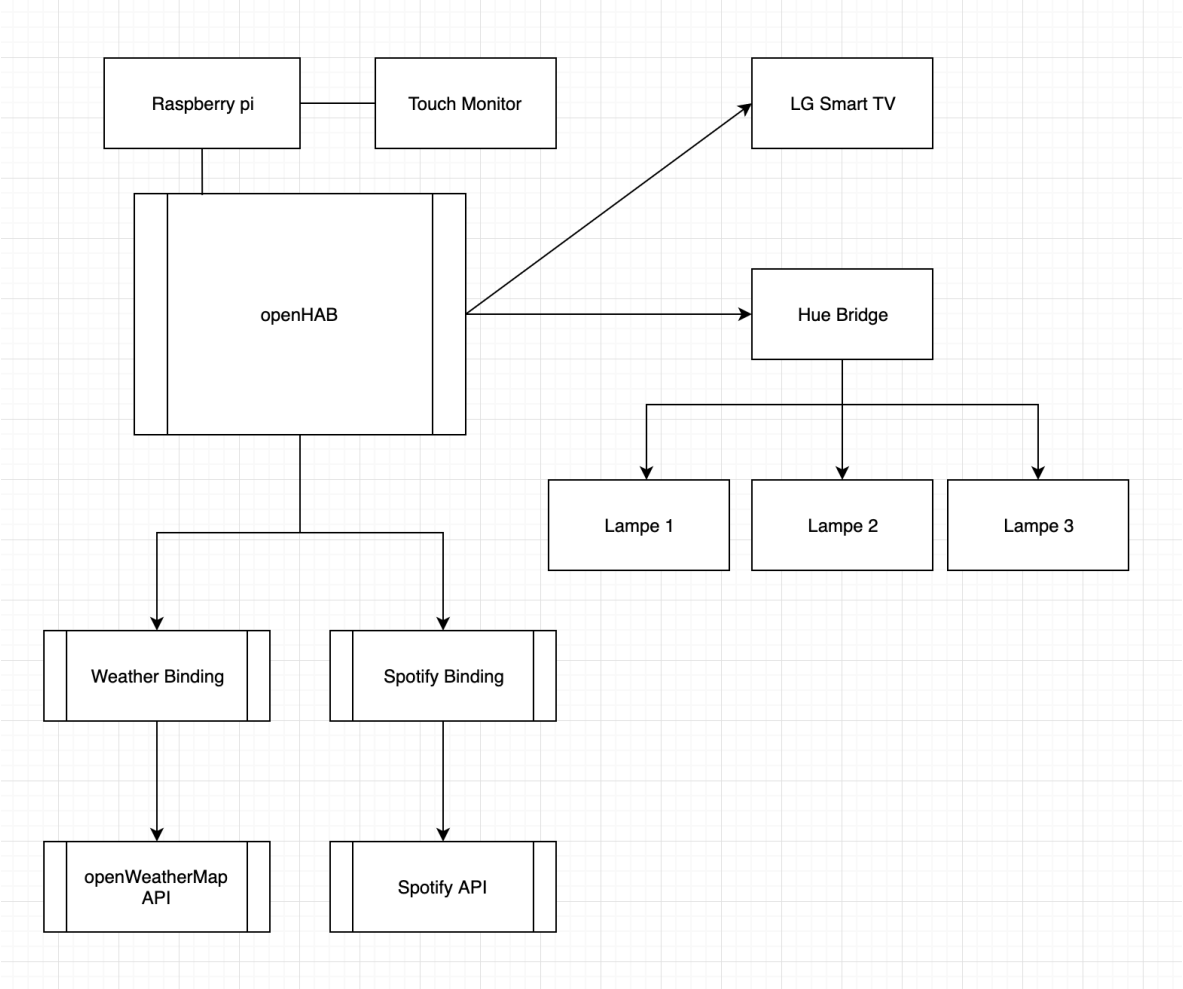


Abbildung 10: Verteilungsarchitektur des Smart Home Systems

Unsere Verteilungsarchitektur zeigt wie die verschiedenen Geräte und Services mit openHAB

verbunden sind. Diese Architektur stellt das Smart Home System für einen Raum da. doch ist dies beliebig erweiterbar ohne das die Architektur komplizierter wird. Der Vorteil von openHAB ist, dass sich hier beliebig viele weitere Geräte und Services anbinden lassen. Da alle Geräte und Services über Bindings integriert werden, ist diese Schicht vom User weg abstrahiert. Dies ermöglicht in einer Umgebung, in der es viele verschiedene Anbieter von Smart-Home-Fähigen Geräten und möglicher Services gibt, dem User eine reduzierte Komplexität darzubieten und ihn so stark zu unterstützen.

# 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 außerdem mögliche Sicherheitslücken und dazugehörige Gegenmaßnahmen.

Open Web Appliation 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 erratbaren 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 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 der Autor 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.[Kre] Folglich wird auf die sowohl internen als auch externen Sicherheits-Features von OpenHAB detaillierter eingegangen.

### 7.1 Interne vorhandene Funktionalitäten

OpenHAB bietet bereits ein paar 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.

Weiterhin warnt OpenHAB auf ihrer Website davor, Ports an der Firewall für OpenHAB zu öffnen. Es wird ausdrücklich gesagt, dass dies in keinem Fall benötigt wird.[OPEb]

### 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 aufbaut, wodurch OpenHAB durch die 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.

## 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 Fachbereich Iot mithalten zu können.[Ten] Das in diesem Bereich 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.

**TODO !!! Hier stehen eure Teile des Fazits !!! TODO**

Die Komponenten aus denen openHAB besteht, bzw. die Teile, welche einen Benutzer zur Verfügung stehen, sind klar Strukturiert und besitzen eine Abstraktionsstufe, welche 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 klare Möglichkeiten der Erweiterung. Dies wird im Falle von Bindings noch durch bereitsgestellten Skripte vereinfacht, was dazu führt, dass openHAB relative 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.
**TODO :::: Resümee:**

## 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 Open Source von der Community entwickelt wird kann sich an diesem Plan etwas ändern. Der Ausblick lässt aber vermuten, dass bereits jetzt ein gute 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


[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] Securing access to openHAB. https://www.openhab.org/docs/installation/security.html. Last visit: 29 Dez 2019.

[OPEc] Develope 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.openhub.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. 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. Last visit: 29 Dez 2019.