HOCHSCHULE FURTWANGEN UNIVERSITY | HFU

Semesterprojekt

im Studiengang

Mobile Systeme Master

Entwicklung eines Concierge-Behaviours für den humanoiden Roboter Pepper

Referent : Prof. Dr. Elmar Cochlovius

Vorgelegt am : 25.01.2018

Vorgelegt von : Caroline Fichtner
Erik Meiß
Lothar Mödl
Julian Keller

Inhaltsverzeichnis

Inhaltsverzeichnis I
Abbildungsverzeichnis III
Tabellenverzeichnis V
1 Einleitung 1
1.1 Motivation 1
1.2 Zielsetzung 2
1.3 Ziele der Arbeit 2
2 Allgemeine Grundlagen 3
2.1 Projektorganisation 3
3 Spezifikation 7
3.1 Funktionsmodell 7
3.1.1 Empfangsbereich 7
3.1.2 TV-Raum 7
3.1.3 Arbeitszimmer 8
3.1.4 Badezimmer 9
3.1.5 Küche 9
3.1.6 Verabschiedung 9
4 Konstruktion 11
4.1 Funktionalitäten 11
4.1.1 Anwendungsstart 11
4.1.2 Initialisierung der Klassen mit Standardwerten 13
4.1.3 Führung 15
4.2 Klassendiagramme 25
4.2.1 Übersicht Klassendiagramm 25
4.2.2 Übersicht der Methoden der Klassen 26
5 Anwendungshinweise 27
5.1 Rahmenbedingungen 27
5.1.1 Stühle 27
5.1.2 Schiebetüren 27
5.1.3 Interaktion mit Fernseher 28
5.1.4 Interaktion mit Sonos 28

5.2 Positionen 29
5.2.1 Startpunkt 29
5.2.2 Wohnzimmer 29
5.2.3 Arbeitszimmer 30
5.2.4 Badezimmer 31
5.2.5 Küche 32
6 Modifikation und Inbetriebnahme der Anwendung 33
6.1 Inbetriebnahme 33
6.2 Anpassung und Bearbeitung des Quellcodes 33
6.3 Anpassung und Bearbeitung durch Konfigurations- und Dialogdateien . 33
6.4 Wichtige Befehle zur Verwendung innerhalb einer Dialogdatei 34
6.5 Besonderheiten bei der Programmierung 35
7 Ausblick 37
8 Fazit 39
Literaturverzeichnis 41
Eidesstattliche Erklärung 43
Anhänge 45
Anhang A Dateien 45
A.1 Deutsches Lexikon 45
A.2 Dialogdateien 54

Abbildungsverzeichnis

Abbildung 1: Projektplan 4
Abbildung 2: Projektstrukturplan 5
Abbildung 3: Use Case Flur 7
Abbildung 4: Use Case TV-Raum 8
Abbildung 5: Use Case Arbeitszimmer 8
Abbildung 6: Use Case Bad 9
Abbildung 7: Use Case Küche 10
Abbildung 8: Use Case Verabschiedung 10
Abbildung 9: Sequenzdiagramm - Programmstart und laden der Konfigurationsdatei 12
Abbildung 10: Sequenzdiagramm - Initialisierung mit Standardwerten 15
Abbildung 11: Sequenzdiagramm - Start der Führung 17
Abbildung 12: Sequenzdiagramm - Führung im TV Raum Teil 1 19
Abbildung 13: Sequenzdiagramm - Führung im TV Raum Teil 2 20
Abbildung 14: Sequenzdiagramm - Führung im Arbeitszimmer 21
Abbildung 15: Sequenzdiagramm - Führung im Badezimmer 22
Abbildung 16: Sequenzdiagramm - Führung in der Küche 23
Abbildung 17: Sequenzdiagramm - Abschlussszene 24
Abbildung 18: Klassendiagramm - Übersicht aller Klassen 25
Abbildung 19: Klassendiagramm - Klassen inklusive Membervariablen und Methoden 26
Abbildung 20: Startpunkt: Eingangsbereich 29
Abbildung 21: Position: Wohnzimmer 30
Abbildung 22: Position: Arbeitszimmer 31
Abbildung 23: Position: Badezimmer 31
Abbildung 24: Position: Küche 32

Tabellenverzeichnis

Tabelle 1: Rollenverteilung 3

1. Einleitung

Roboter im Allgemeinen bekommen mit fortschreitenden technischen Entwicklungen zunehmend mehr an Bedeutung und rücken immer mehr in den Fokus in einer Vielzahl von Bereichen, die verschiedener nicht sein könnten. Von Robotern, welche für die Produktion unterschiedlichster Produkte wie z. B. Autos eingesetzt werden, bis hin zu Haushaltsrobotern und noch weiter scheint es kaum denkbare Grenzen für deren Einsatz zu geben.
Aufgrund dieser Entwicklung hat die Fakultät Informatik der Hochschule Furtwangen zwei Roboter der Firma „Aldebaran“ der Baureihe „Pepper“ angeschafft, um verschiedenste Use-Cases mittels Semesterprojekten zu entwickeln und in diesem Zug auch praxisnah umzusetzen. Hierdurch sollen die Studierenden durch die praktische Arbeit mit den Robotern neue Fähigkeiten erarbeiten, wie diese eingesetzt werden können und welche Faktoren beachtet werden müssen, dass ein Projekt mit Robotern zu einem Erfolg wird.
Im Zuge des vorliegenden Projektes soll hierbei eine Führung durch das Smart Home Labor der Hochschule Furtwangen geplant und umgesetzt werden, bei dem der Roboter Pepper möglichst autonom im Raum navigieren soll, um die Besucher im Labor zu führen und die Einrichtungen zu erläutern, erklären und wenn möglich auch vorzuführen.

1.1. Motivation

Motiviert wird das Projekt durch den gegebenen Hintergrund, dass Roboter jeglicher Art und Herkunft zunehmend wahrnehmbar in den unterschiedlichsten Bereichen auftauchen. Waren es in der Vergangenheit eher nur Roboter die in großen Produktionsstädten, wie z. B. in der Automobilindustrie von sich reden haben lassen, so haben sich mittlerweile auch eine Vielzahl von Roboter den Weg in die Wohnzimmer der Menschen gebahnt.
Neben Vorreitern wie Google mit „Google Now“ oder mittlerweile dem „Google Assistent“, die schon früh die aktuellen Entwicklungen im Bereich Spracherkennung für die Nutzer nutzbar gemacht haben. Mittlerweile gibt es auch eine Vielzahl von Robotern

für z. B. Staubsaugen, Bodenwischen, Rasenmähen und noch viel mehr für den privaten Einsatz. Diese sind in der Regel auch alle untereinander vernetzt und können beispielsweise über eine zentrale Stelle verwaltet und angesteuert werden.
Aus dieser erkennbaren Tendenz heraus besteht nun die Motivation, über einen praxisnahen Bezug studierende zu dem Thema der Robotik zu sensibilisieren und mit praktischen Projekten den fachlichen Hintergrund zu stärken und schon früh auf die technischen Tendenzen hin vorzubereiten, um später im Beruf von den erlernten Fähigkeiten profitieren zu können.

1.2. Zielsetzung

Das Ziel des Projektes besteht darin, mit dem Roboter der Baureihe Pepper der Firma Aldebaran eine möglichst interaktive Führung durch das Smart Home Labor der Fakultät Informatik der Hochschule Furtwangen zu konzipieren und umzusetzen. Hierbei besteht die Anforderung, dass der Roboter autonom im Raum navigiert und die jeweiligen Räume mittels Sprachausgabe erläutert. Weitergehend soll es dem Roboter möglich sein, mit diversen Smart-Home-Geräten im Labor zu interagieren und diese auch an definierten Punkten während der Führung durch das Labor zu steuern. Dabei soll auch der Nutzer in die Führung durch Spracherkennung und mit sprachlichen Aufforderungen durch den Roboter mit eingebunden werden, um die Führung möglichst intuitiv zu erleben. Abschließend soll nach der Führung auf dem Tablet noch die Möglichkeit bestehen, die Führung mit Sternen von eins bis fünf zu bewerten.

1.3. Ziele der Arbeit

Ziel dieser Arbeit ist es, die Hintergründe des Projektes zu erläutern und zu beschreiben. Dabei sollen die Ideen zu Beginn der Entwicklung dargelegt und beschrieben werden wie im Laufe des Projektes vorgegangen wurde. Hierbei wird auch besonderes auf die endgültige Umsetzung eingegangen, um nachvollziehbar diese zu dokumentieren und anhand von Beispielen, Code und Grafiken verständlich zu untermauern. Dadurch soll sichergestellt sein, dass sich nachfolgende Projekte ohne Probleme in die bisherige Umsetzung einarbeiten können und das bisher umgesetzte in weiteren Projekten um neue Use-Cases erweitern zu können.

2. Allgemeine Grundlagen

Mit dem Beginn des Projektes, das sich auf keine Vorprojekte stützt, wird ein erstes Treffen mit dem Betreuer und Kunden des Projektes vereinbart. Ziel dieses Treffens ist die Spezifikation der Projektziele, die Definition eines Concierge-Verhaltens, sowie gewünschte Ergebnisse seitens des Betreuers. Zusätzlich finden interne Projekttreffen ohne Betreuer mindestens einmal die Woche über das gesamte Semester hinweg statt. Vordefinierte Projekttreffen seitens des Betreuers dienen der Gruppe als Möglichkeit, den aktuellen Stand zu demonstrieren und weiteres Feedback in den Projektablauf zu integrieren.

2.1. Projektorganisation

Um einen geregelten Ablauf in der Gruppe zu garantieren, wird zu Beginn eine Rollenverteilung vorgenommen. Die Zuteilung der Rollen, sowie Aufgabenbereiche sind fest definiert, dennoch werden freie Kapazitäten für offene Aufgaben und der Unterstützung anderer Gruppenmitglieder genutzt. Die Rollenverteilung setzt sich wie folgt zusammen:

Name	**Rollen**
Fichtner, Caroline	Dokumentation
Keller, Julian	Dokumentation
Meiß, Erik	Testing
Mödl, Lothar	Implementierung

Tabelle 1.: Rollenverteilung

Um das Projekt termingerecht ausliefern zu können, werden nach der Rollenverteilung sämtliche anfallende Aufgaben festgehalten und diese in einem Projektplan zeitlich terminiert. Einen Überblick lässt sich der Abbildung 1 entnehmen.

Grau unterlegte Felder kennzeichnen die Meilensteine im Projekt. Die jeweiligen Aufgaben eines Meilensteins werden nach Aufwand in Personentagen geschätzt. Die Tätigkeiten werden zur besseren Übersicht über abgeschlossene oder offene Aufgaben mit einem Status von null bis 100 in Prozent versehen. Des Weiteren lässt sich in

Aufgabe/Tätigkeit	Beginn [Datum]	Ende nach [AT]	Ende [Datum]	Status [0 bis 100]	Meilenstein [Datum]	Verantwortlich [Name]	Unterstützung [Name]
M1 Projektstart und -vorbereitung					**26.10.17**		
Einarbeitung, SDK, I DE	16.10.17	9	26.10.17	100%		Team	
Projektbesprechung Cochlovius	17.10.17	1	17.10.17	100%		Team	Cochlovius
Planung potentielle Funktionalitäten, Ablaufsszenarien	24.10.17	2	25.10.17	100%		Team	Schweizer
Smart Home Lab Einweisung	26.10.17	1	26.10.17	100%		Team	
M2 Basic Interaction					**01.11.17**		
Einarbeitung in die Basicfunctions Laufen, Sprechen, Gesichtserkennung	27.10.17	4	01.11.17	100%		Team	
M3 Tourplanung + Dummy					**16.11.17**		
Tourablauf des Peppers planen	02.11.17	1	02.11.17	100%		Team	
Beispieltexte für Pepper planen	03.11.17	5	09.11.17	100%		Meiß	
Storyboard Raumablauf erstellen	10.11.17	3	14.11.17	100%		Fichtner	
Erster Dummy mit Basicfunctions	02.11.17	10	15.11.17	100%		Mödl, Keller	
Projekttreffen mit H. Cochlovius	16.11.17	1	16.11.17	100%		Team	Cochlovius
M4 Sprachinteraktion und Gesten					**27.11.17**		
Planung verschiedener Unterhaltungen und zugehöriger Gesten	17.11.17	4	22.11.17	100%		Meiß	
Storyboard Unterhaltung erstellen	17.11.17	2	20.11.17	100%		Meiß	
Planung Interaktion mit Smart Home Lab	20.11.17	5	24.11.17	100%		Team	
Implementierung Sprachinteraktionen	19.11.17	6	27.11.17	100%		Mödl, Keller	
Implementierung Gesten	22.11.17	4	27.11.17	100%		Mödl, Keller	
M5 Implementierung der Concierge-Funktionen					**13.12.17**		
Implementierung des Raumablaufs von Pepper	28.11.17	6	05.12.17	100%		Mödl, Keller	
Anpassung der Texte	01.12.17	3	05.12.17	100%		Meiß	
Implementierung weiterer Concierge-Funktionen	06.12.17	6	13.12.17	100%		Mödl, Keller	
Projektbesprechung Cochlovius	14.12.17	1	14.12.17	100%		Team	Cochlovius
M6 Dokumentation					**10.01.18**		
Erstellung einer Dokumentation	16.11.17	30	27.12.17	100%		Fichtner, Keller	
M7 Bugfixing					**27.12.18**		
Ausführliches Testen der Funktionen	15.12.17	4	20.12.17	100%		Meiß	
Bugfixing	21.12.17	5	27.12.17	100%		Mödl, Keller	
M8 Projektende					**28.12.18**		
M9 Finale Projektbesprechung Cochlovius					**25.01.18**	Team	Cochlovius
M10 Projektvorstellung					**26.01.18**	Team	

Abbildung 1.: Projektplan

Abbildung 1 der grobe Aufgabenbereich der einzelnen Gruppenmitglieder ablesen. Eine weitere Möglichkeit zur Strukturierung des Projekts bietet der Projektstrukturplan, der in Abbildung 2 zu finden ist.

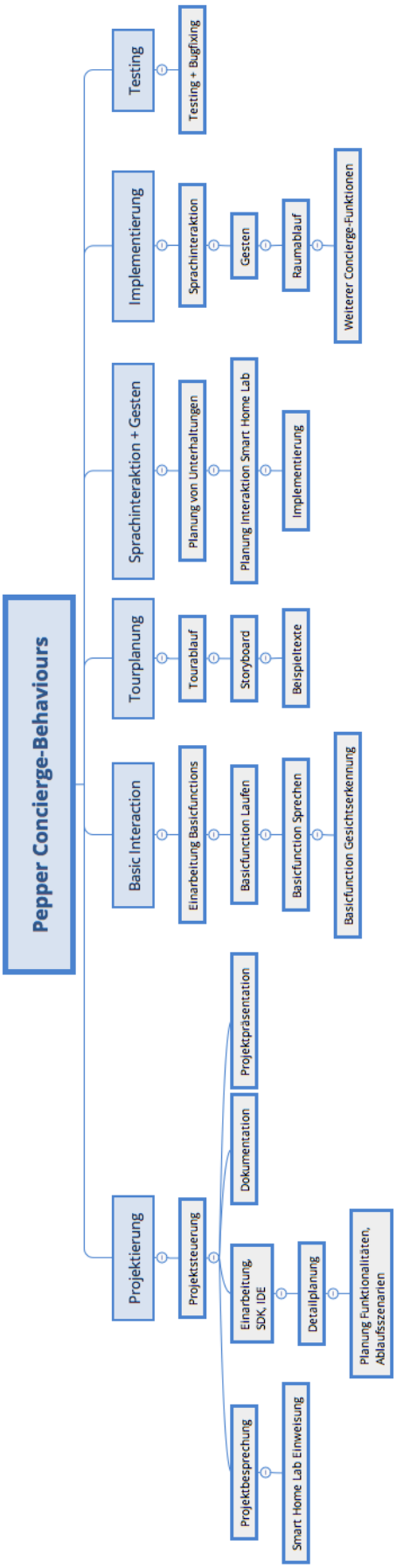

Abbildung 2.: Projektstrukturplan

3. Spezifikation

3.1. Funktionsmodell

3.1.1. Empfangsbereich

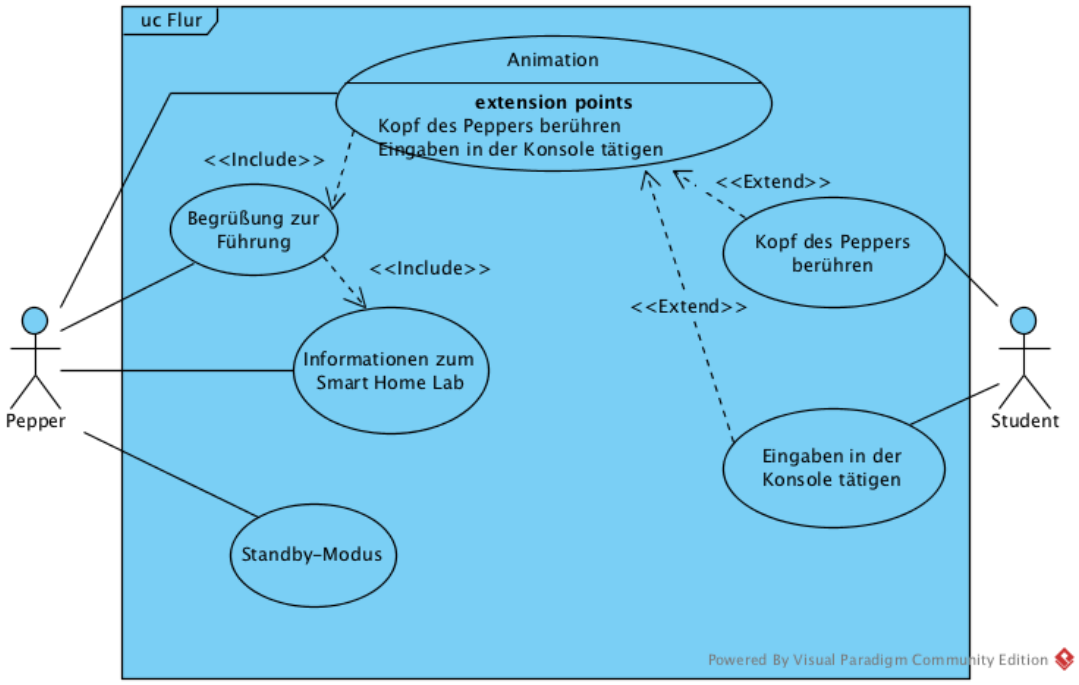

Abbildung 3.: Use Case Flur

Zu Beginn kann die Führung durch einen Betreuer über ein textuelles Menü, oder durch Berührung des Kopfes des Roboters gestartet werden. Nachfolgend bietet Pepper eine Animation dar und startet daraufhin die Führung mit einer Begrüßung der Teilnehmer und gibt erste Informationen zum Smart Home Labor.

3.1.2. TV-Raum

Im TV-Raum teilt der Roboter zunächst Informationen über den Raum per Sprachausgabe mit und fordert einen Teilnehmer der Führung auf, ein Fenster zu öffnen. Daraufhin reagiert er und erklärt die Funktionalität, bis wiederum ein Teilnehmer aufgefordert wird, das Fenster zu schließen, woraufhin Pepper erneut eine Sprachausgabe

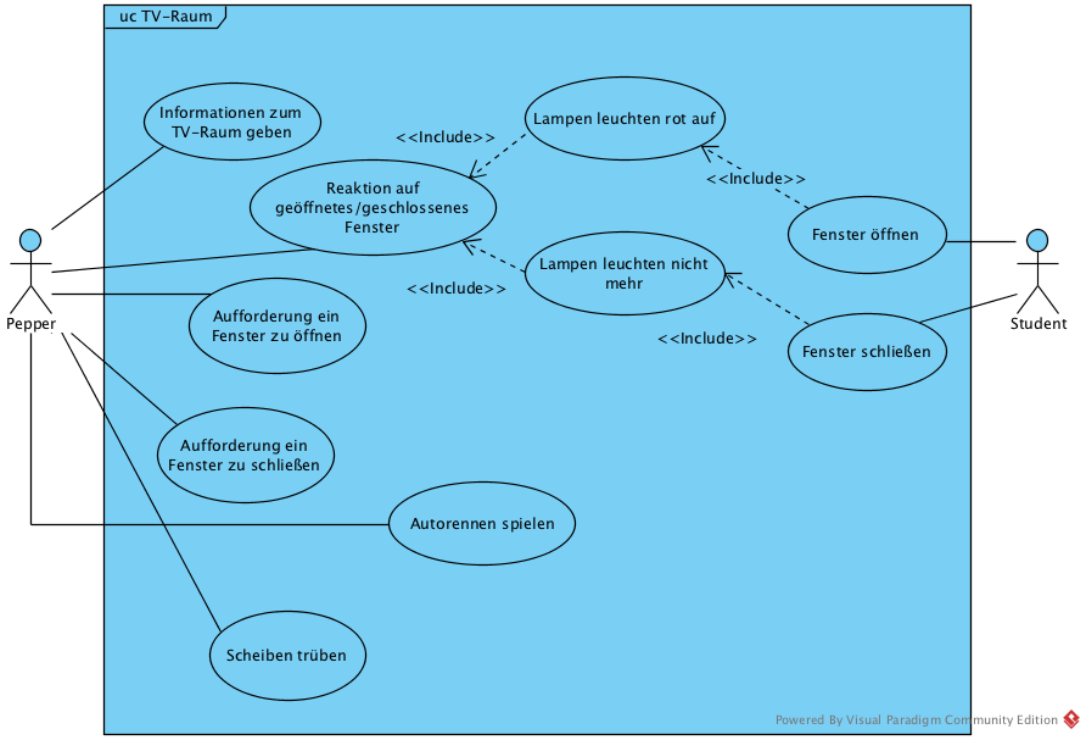

Abbildung 4.: Use Case TV-Raum

liefert. Danach folgt eine Interaktion mit dem Fernseher, den der Roboter nutzt, um eine Spieleszene, in Form eines Autorennens, zu präsentieren, welche er parallel dazu mit einer angepassten Animation unterstützt. Für eine Demonstration zum Schutz der Privatsphäre werden die Türen des Raumes milchig getrübt.

3.1.3. Arbeitszimmer

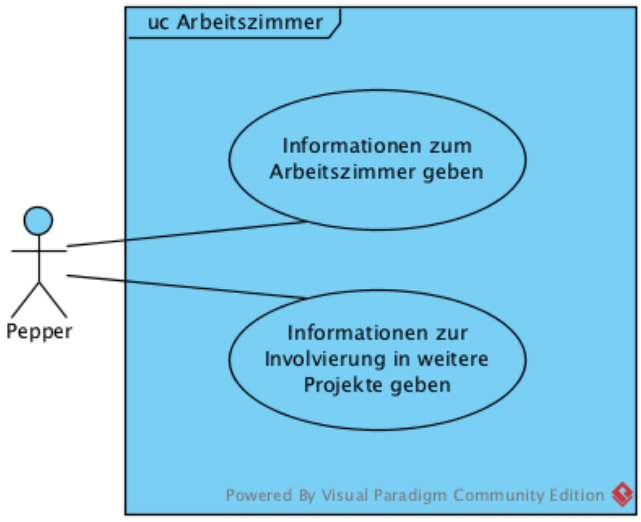

Abbildung 5.: Use Case Arbeitszimmer

Im Arbeitszimmer teilt der Roboter über eine Sprachausgabe die Bedeutung der Nao-

Marks an der Wand, sowie seine Involvierung in andere Projekte (Water Buddy+) in diesem Raum mit. Außerdem gibt er noch zusätzliche Informationen über ein weiteres Projekt, in welchem der Smart-Table Verwendung findet.

3.1.4. Badezimmer

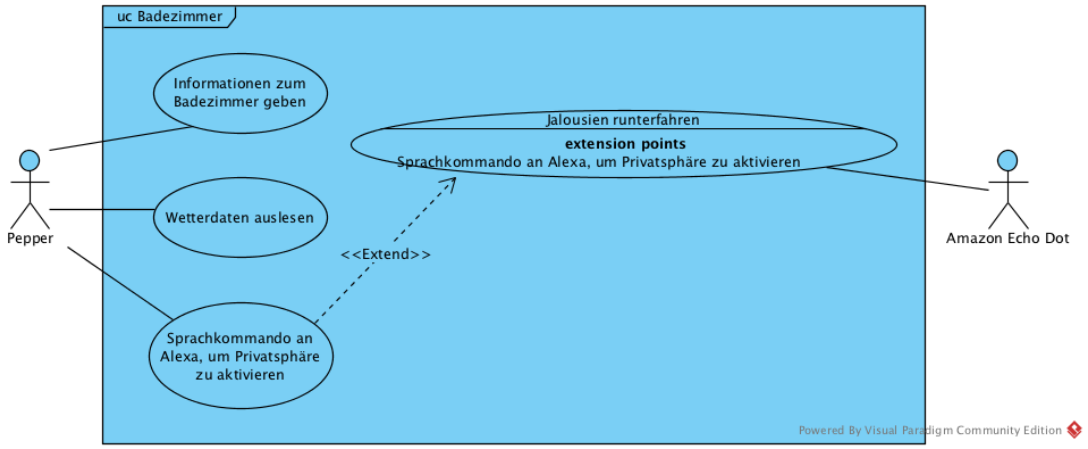

Abbildung 6.: Use Case Bad

Im Badezimmer teilt der Roboter ebenfalls per Sprachausgabe Informationen über die Geräte in diesem Raum, wie die Waschmaschine und den Trockner, mit. Zudem liest er noch die aktuelle Temperatur und den CO_2-Wert der im Bad montierten Wetterstation aus und teilt diese den Besuchern ebenfalls mit. Für eine Demonstration der Innenjalousien, welche an den Türen des Badezimmers und der Küche angebracht sind, kommuniziert Pepper verbal mit der Sprachsteuerung „Alexa“ in Verbindung mit einem „Amazon Echo Dot“ und fordert diese auf, den Privatsphäremodus zu aktivieren, woraufhin die Jalousien heruntergefahren werden.

3.1.5. Küche

Die Conciergefunktion des Roboters beinhaltet des Weiteren eine Interaktion mit der Küche. Dort wird neben der Sprachausgabe auch die aktuelle Temperatur des Kühlschranks, wiedergegeben.

3.1.6. Verabschiedung

Am Ende der Führung erfolgt eine ausführliche Verabschiedung der Teilnehmer durch den Roboter. Hierzu zählt eine sprachliche Interaktion mit Pepper durch die Teilnehmer, d.h. die Teilnehmer können dem Roboter Fragen stellen, seine Small-Talk-Fähigkeiten testen, sowie die Ansteuerung von Geräten im Smart Home Labor fordern.

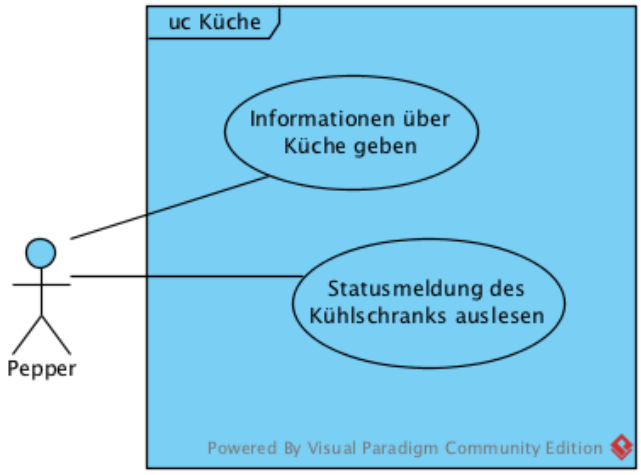

Abbildung 7.: Use Case Küche

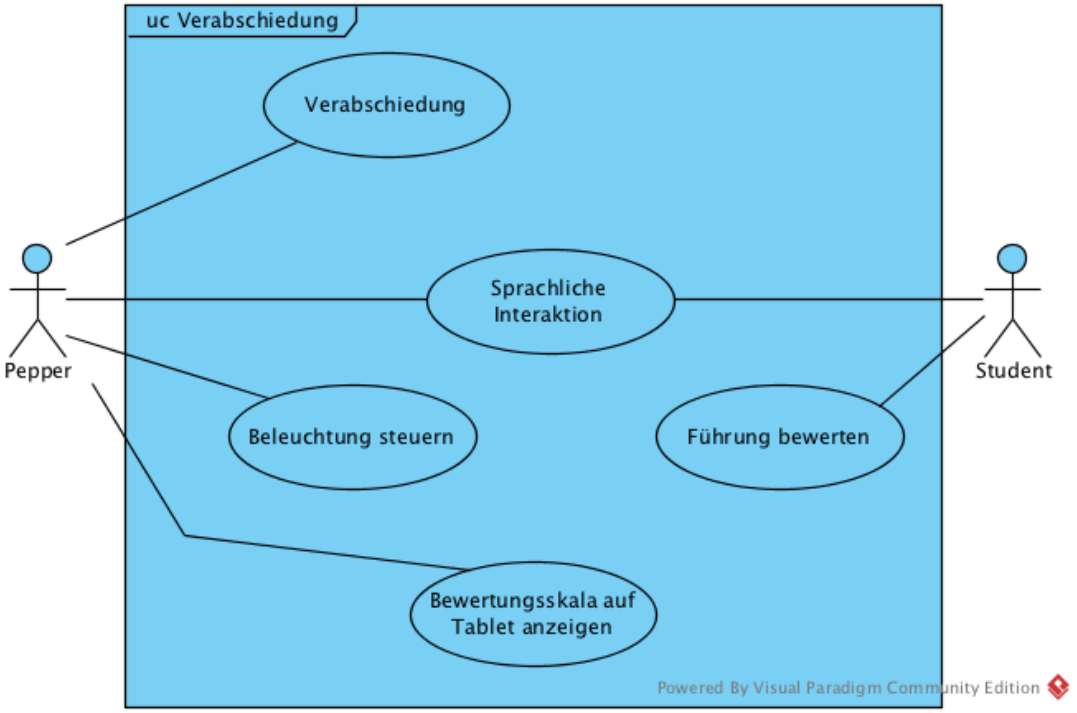

Abbildung 8.: Use Case Verabschiedung

Zudem demonstriert Pepper, wie er die Beleuchtung im gesamten Labor steuern kann, in Verbindung mit einer musikalischen Untermalung. Zum Schluss können die Besucher die Führung auf dem Tablet des Roboters noch bewerten.

4. Konstruktion

4.1. Funktionalitäten

4.1.1. Anwendungsstart

Beim Start der Anwendung wird als erstes die Konfigurationsdatei, durch Aufruf der Methode `loadConfigFile()` der Klasse `ConfigurationManager`, eingelesen. Hierbei wird eine Datei mit dem vordefinierten Namen `concierge.conf` verwendet, welche im gleichen Verzeichnis wie die Anwendung liegen muss. Die eingelesenen Werte werden hierbei durch die statische Klasse `Constants.Config` verwaltet. Im Anschluss wird eine `Application` erzeugt. Der `ConciergeController` startet durch Aufruf der Methode `menu()` die Ausgabe des Menüs auf der Konsole, sowie die Routine zur Eingabe eines Zahlenwertes, welcher jeweils einer eigenen Funktion entspricht, die im Folgenden aufgerufen wird. Alle Funktionalitäten werden von der Klasse `BasicBehaviour` verwaltet. Nachfolgende Möglichkeiten bietet das Menü und deckt die möglichen Eingabewerte von 0-10 ab:

- 0) Fortlaufende Tour ab ... starten: (true/false)
- 1) Komplette Tour starten
- 2) Begrüßung und allgemeine Informationen starten
- 3) TV Raum starten
- 4) Arbeitszimmer starten
- 5) Bad starten
- 6) Küche starten
- 7) Verabschiedung starten
- 8) Smart Home Labor auf Ausgangseinstellungen setzen
- 9) Reboot Pepper
- 10) Shutdown Pepper

Durch die Eingabe der Zahl 0 kann bestimmt werden, ob ab dem Start eines Raumes, wie zum Beispiel des TV Raumes, alle fortfolgenden Räume ebenfalls durchlaufen werden sollen (true), oder nur der jeweilige Raum (false). Alle weiteren Funktionalitäten sind selbsterklärend.

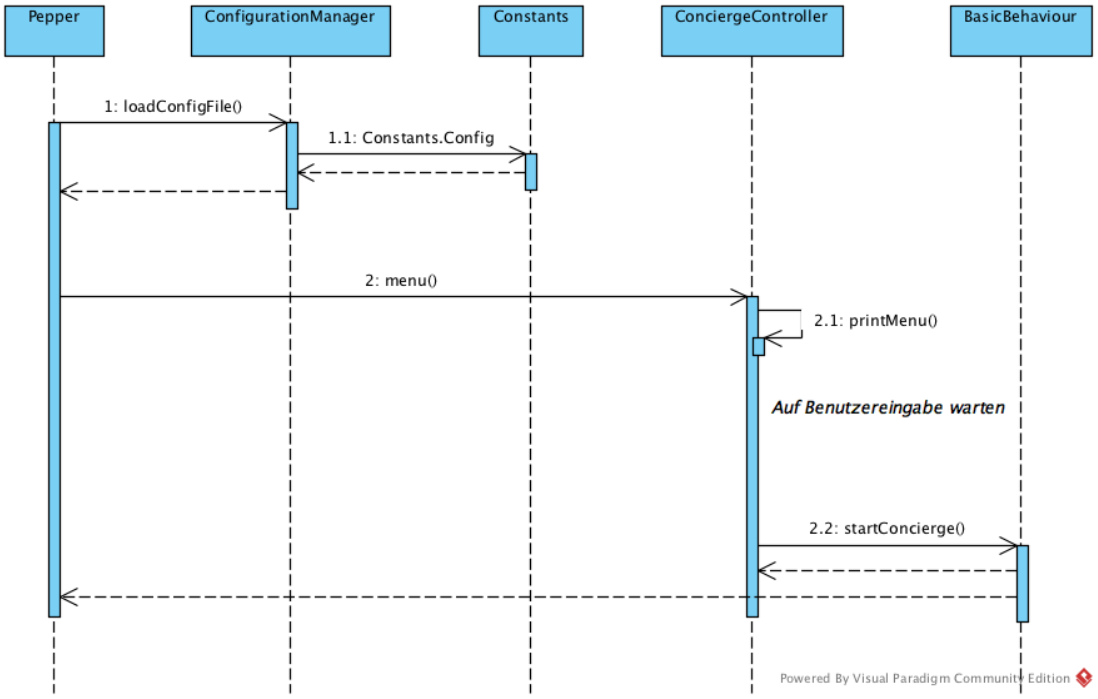

Abbildung 9.: Sequenzdiagramm - Programmstart und laden der Konfigurationsdatei

4.1.1.1. Aufbau der Konfigurationsdatei

Die Konfigurationsdatei ist im JSON-Format und beinhaltet folgende Attribute:

```
{
   "pepperIP": "192.168.0.71",
   "headless": false,
   "debug": false,
   "movieUrl": "http://192.168.0.65/video.php?path=videos/CUTRace_v4.mp4",
   "sonosIP": "192.168.0.30",
   "ratingUrl": "http://192.168.0.65/"
}
```

Der Wert des Schlüssels `pepperIP` gibt die IP-Adresse des jeweiligen Roboters an, welcher zum Ausführen der Anwendung verwendet werden soll. Der Schlüssel `headless` gibt an, ob der Rechner, auf welchem die Anwendung ausgeführt und damit der Roboter gesteuert wird, ein Display, sowie Eingabemöglichkeiten besitzt oder nicht. Wird der Wert auf `true` gesetzt, so wird davon ausgegangen, dass der Rechner kein Display besitzt und somit wird auch kein Menü auf der Konsole ausgegeben. Damit ist es auch nur möglich, die komplette Tour durch das Smart Home Labor durch berühren des Kopfes des Roboters zu starten. Bei aktiviertem `Debug`-Modus (`debug: true`) werden

entsprechende Debug-Meldungen ausgegeben und eine Log-Datei wird erstellt. Der Wert des Schlüssels `movieUrl` gibt an, welches Video während der Spieleszene im TV Raum (siehe Abschnitt 4.1.3.2) auf dem Fernseher wiedergegeben werden soll. `sonosIP` gibt die IP-Adresse der Sonos-Soundanlage an, auf welcher die Wiedergabe von Musik erfolgen soll. Der letzte Schlüssel `ratingUrl` gibt an, wo die Webseite für die Bewertung am Ende der Tour liegt, welche auf Peppers Tablet angezeigt wird (siehe Abschnitt 4.1.3.6). Kann keine Konfigurationsdatei im Verzeichnis gefunden werden, so werden die oben genannten Standardwerte verwendet.

4.1.2. Initialisierung der Klassen mit Standardwerten

Nach dem Laden der Konfigurationsdatei werden die Klassen, vor allem diese vom Hersteller „Aldebaran“ (alle Klassen, die mit AL beginnen), mit notwendigen Standardwerten initialisiert. Die Klasse `BasicBehaviour` ist für die Steuerung des kompletten Ablaufs des Roboters verantwortlich und enthält entsprechend auch die Objektreferenzen der benötigten Klassen. Die Klasse `SonosDevice` wird mit der in der Konfigurationsdatei hinterlegten IP-Adresse initialisiert und im Anschluss wird der Lautlosmodus des Geräts deaktiviert, um eine Audioausgabe zu ermöglichen. Die Klasse `ALDialog`, welche für die Steuerung der Sprachein- und -ausgabe des Roboters zuständig ist, wird nach der Initialisierung durch Aufruf der Methode `setLanguage(language:String)` auf die Sprache „German“ gesetzt, sodass der Roboter im Folgenden auf deutsch kommunizieren kann. Nach der Initialisierung der Klasse `ALMotion`, welche für die Steuerung von Bewegungsabläufen und Kollisionserkennung zuständig ist, wird die Kollisionserkennung des kompletten Roboters durch Aufruf der Methode `setExternalCollisionProtectionEnabled (bodyPart: String, enabled: Bool)` aktiviert. Ebenso wird die Sicherheitsdistanz, welche der Roboter zu vor ihm liegenden Objekten einhalten soll, auf 15 cm gesetzt (`setOrthogonalSecurityDistance(dist:Float)`). Um die spätere Nutzung des Tablets (`ALTabletService`) von Pepper zu ermöglichen, wird dessen WLAN aktiviert (`enable Wifi()`), sowie, falls schon eine Webseite oder ein Bild auf dem Tablet angezeigt wird, werden diese ausgeblendet (`hideWebview()`, `hideImage()`). Durch Aufruf der Methode `wakeUp()` wird der Roboter aus seiner Ruheposition in eine aufrechte Position versetzt. Danach werden noch entsprechen Event-Handler durch mehrfachen Aufruf der Methode `subscribeToEvent(eventName: String, callback: String, destination: Object): Long` für nachfolgende Events, welche mithilfe der Klasse `ALMemory` erzeugt werden können, registriert:

1. `subscribeToEvent("PlayMusic", "onPlayMusic::(s)", this)`

2. `subscribeToEvent("SubscribeMQTTTopic", "onSubscribeMQTTTopic::(s)", this)`

3. `subscribeToEvent("UnsubscribeMQTTTopic", "onUnsubscribeMQTTTopic::(s)", this)`
4. `subscribeToEvent("GetValue", "onGetValue::(s)", this)`
5. `subscribeToEvent("PublishMQTTMessage", "onPublishMQTTMessage::(s)", this)`
6. `subscribeToEvent("OpenUrl", "onOpenUrl::(s)", this)`
7. `subscribeToEvent("TakePicture", "onTakePicture::(s)", this)`

Alle Events, speziell diese zum Ausführen von Funktionalität, bieten sich an, um zum Beispiel von außerhalb über einen Dialog aufgerufen zu werden. Für einen internen Aufruf einer der Funktionalitäten durch Änderung des Quellcodes, muss der Umweg über die Events von `ALMemory` nicht erfolgen.

Bei Eintreten des ersten Events wird der Musiktitel, welcher durch die Url übergeben wird, auf der Sonos-Anlage abgespielt. Hierbei ist darauf zu achten, dass der Präfix `x-file-cifs://` für das Abspielen eines Titels, welcher bei einer der bei der Sonos-Anlage hinterlegten Quellen verfügbar ist, hinzugefügt wird. Der Präfix `x-rincon-mp3radio://` muss zum Abspielen eines Online-Musikstreams verwendet werden.
Durch Aufrufen des zweiten Events kann das übergebenen MQTT-Topic abonniert werden. Die durch das Abonnement erhaltenen Werte werden in `ALMemory` mit dem Namen des MQTT-Topics als Schlüssel abgelegt. Eine Ausnahme stellen die Werte beim Öffnen oder Schließen eines Fensters dar. Diese werden direkt verarbeitet und führen zu einer Sprachausgabe bzw. zum Ausführen eines Teils der Tour des TV Raums.
Das dritte Event beendet das Abonnement des übergebenen MQTT-Topics.
Das vierte Event ruft bei OpenHab hinterlegte Werte durch das übergebenen MQTT-Topic ab. Diese Werte werden in `ALMemory` mit dem Titel des MQTT-Topics als Schlüssel abgelegt. Dadurch kann der Abruf eines Wertes über den Dialog gesteuert und anschließend direkt im Dialog ausgegeben werden.
Das fünfte Event ermöglicht das veröffentlichen eines Wertes über MQTT. Hierfür muss das MQTT-Topic und der zu veröffentlichende Wert über ein Semikolon getrennt übergeben werden.
Das sechste Event ermöglicht das Laden einer Webseite auf Peppers Tablet. Hierfür muss die entsprechende Url übergeben werden.
Das siebte Event ermöglicht die Erstellung eines Bildes durch Pepper, welches im Anschluss auf seinem Tablet angezeigt wird.

Danach werden die für die Führung benötigten Geräte durch veröffentlichen der entsprechenden Werte in Verbindung mit den jeweiligen MQTT-Topics auf den Ausgangszustand gesetzt (`publishToItem(topic: String, value: String)`). Zu den Geräten zählen die vier Innenrollos und die beiden schaltbaren Türen.

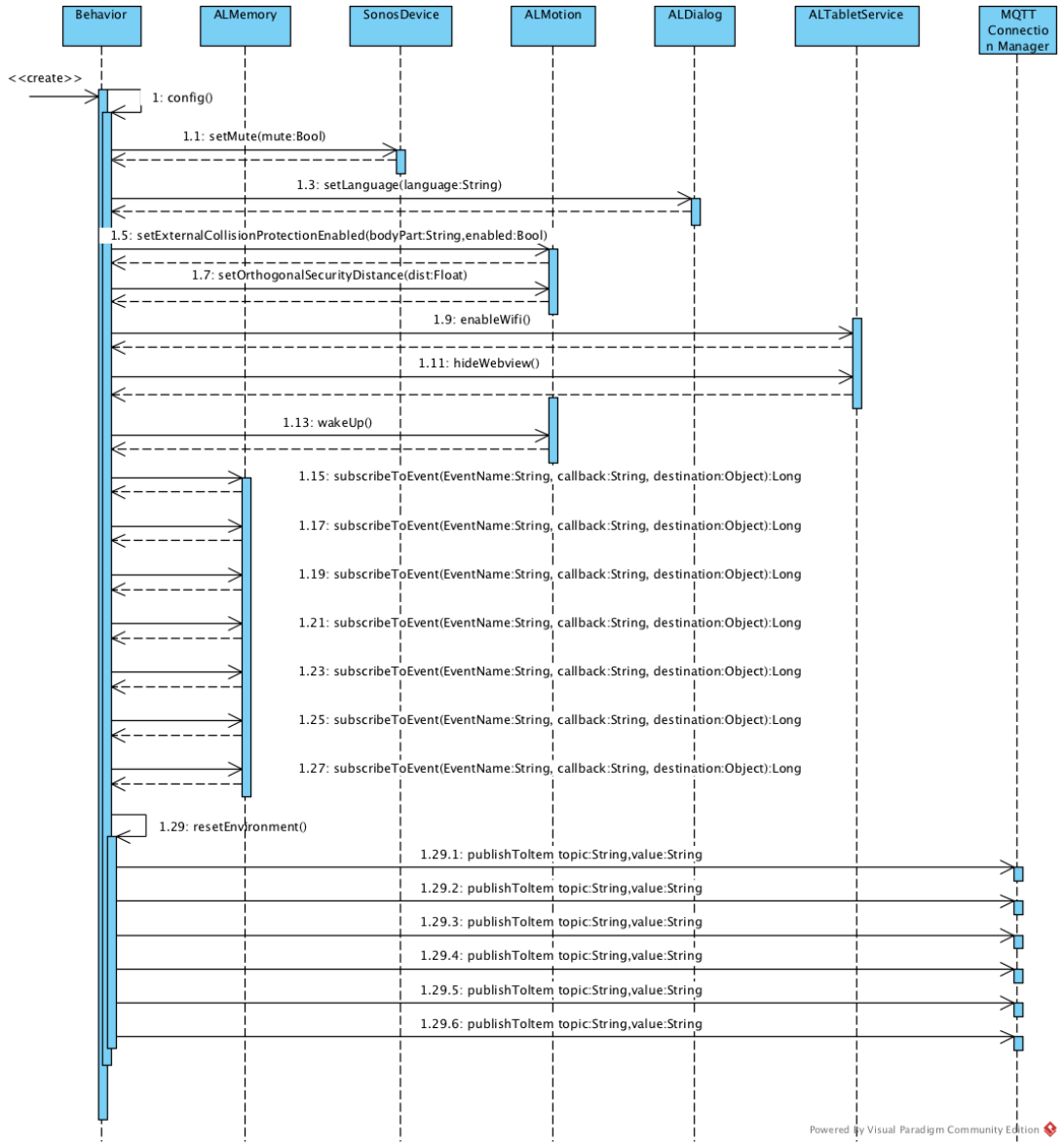

Abbildung 10.: Sequenzdiagramm - Initialisierung mit Standardwerten

4.1.3. Führung

4.1.3.1. Start der Führung

Die Führung beginnt mit einer Sprachausgabe. Hierfür wird die Klasse `ALDialog` benötigt. Beim Start und beim Beenden der Sprachausgabe ist sehr genau auf die Reihenfolge der Aufrufe zu achten, sonst kann möglicherweise die Sprachsteuerung nur durch einen Neustart des Roboters beendet und somit erneut verwendet werden! Zum Start der Sprachsteuerung mit einer Dialogdatei, muss die entsprechende Datei im angegebenen Verzeichnis auf dem Roboter liegen. Ein Remote-Verzeichnis wird nicht akzeptiert! Die Methode `loadTopic(topic:String)` läd die entsprechende Dialogdatei (`Welcome.top` siehe Anhang A.2). Die Methode `activateTopic(topic: String)` aktiviert

das angegebene Topic, sodass Sprachein- und -ausgaben durch dieses Topic abgearbeitet werden. Durch Aufruf von `subscribe(name:String)` wird die Sprachein- und -ausgabe auf dem Roboter aktiviert, sodass nun mit dem Roboter gesprochen werden kann. `forceOutput` ermöglicht die Ausgabe der nächsten Phrase, welche durch `proposal` in der Dialogdatei gekennzeichnet ist. Zum Beenden der Sprachsteuerung mit einer Dialogdatei muss zunächst durch `unsubscribe(name:String)` die Sprachsteuerung auf dem Roboter deaktiviert werden. Die Methode `deactivateTopic(topic:String)` deaktiviert das zuvor aktivierte Topic und mithilfe von `unloadTopic(topic:String)` wird die geladene Dialogdatei freigegeben, sodass mit Durchführung der oben genannten Befehle wieder eine neue Dialogdatei eingelesen werden kann.
Die Methode `moveTo(x:Float, y:Float, theta: Float): Bool` der Klasse `ALMotion` ermöglicht die Fortbewegung des Roboters in die x- und y-Richtung. Durch Angabe des Theta-Wertes wird eine Drehung des Roboters ermöglicht. Hier erfolgt zunächst eine 180°-Drehung, eine Fortbewegung in x-Richtung, sowie eine erneute 180°-Drehung. Durch Aufruf der Methode `putHeadUp()` unter der Verwendung der Methode `angleInterpolationWithSpeed(bodyPart: String, angles: [Float], speed: Float)` wird der Kopf des Roboters in eine senkrechte Position versetzt, sodass er auf die Personen, welche an der Führung teilnehmen, gerichtet ist. Die Methode `loadTopicWithForceOutput (topic:String)` kapselt die oben genannten Schritte zum laden und entladen einer Dialogdatei inklusive der Ausgabe des ersten `proposals` der jeweiligen Datei. Hier wird die Datei `General.top` geladen.
Im Anschluss wird die Tour in Richtung des TV Raumes fortgesetzt durch eine 90°-Drehung im Uhrzeigersinn, sowie einer Bewegung in x-Richtung. Bei wichtigen Strecken folgt ein Aufruf der Methode `checkDestination(success: Bool, oldPosition: [Float], newPosition: [Float], distance: [Float])`, welche bei nicht Ankunft am Ziel, aufgrund zum Beispiel von Hindernissen, die Strecke, die noch zurückzulegen ist bist zum Ziel, berechnet und diese unter der Verwendung der Methode `navigateTo(x: Float , y: Float): Bool` versucht zurückzulegen. Die Methode `navigateTo` ermöglicht bestmöglichst die Umfahrung von Hindernissen auf dem Weg. Da bei nicht Ankunft am Ziel ein Fehlerstatus im Hintergrund vorhanden ist, welcher über die weitere Fortbewegung des Roboters entscheidet und es eine unbestimmte Zeit benötigt bis dieser wieder zurückgesetzt ist, wird die Interpolation der Fortbewegung des Roboters bis zu zehn Mal versucht oder bis er am Ziel angekommen ist.
Am Ende eines jeden Raumes wird bestimmt, ob der nachfolgende Raum ebenfalls ausgeführt werden soll oder nicht. Dies kann durch das Menü (siehe Abschnitt 4.1.1) gesteuert werden.

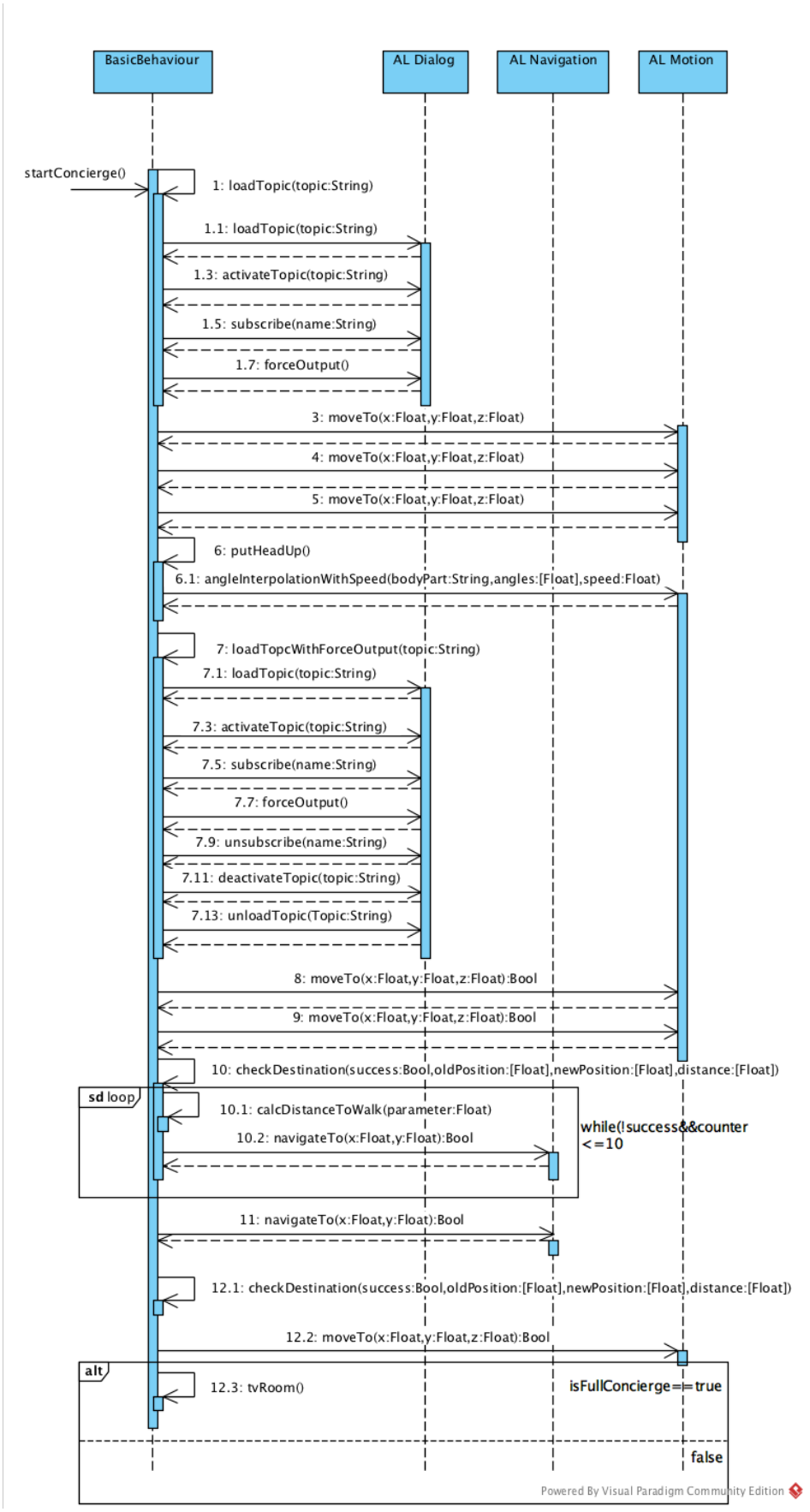

Abbildung 11.: Sequenzdiagramm - Start der Führung

4.1.3.2. TV Raum

Im TV Raum werden zu Beginn zwei Events bei der Klasse `ALMemory` registriert, welche ausgelöst werden, wenn ein Fenster geöffnet bzw. geschlossen wird. Ebenso werden die MQTT-Topics aller Fenster abonniert, sodass eine Reaktion auf das Öffnen oder Schließen eines der Fensters ausgeführt werden kann. Nach dem Laden der Dialogdatei (`TVRoom.top` siehe Anhang A.2) und der Sprachausgabe des Roboters wird auf das Öffnen eine Fensters gewartet. Sobald ein Fenster geöffnet wird, wird die Callback-Methode `onSubscription(item: String, value: String)` aufgerufen. Hier wird entschieden, ob ein Fenster geöffnet oder geschlossen wurde. Bei Öffnen eines Fensters wird das entsprechende Event der Klasse `ALMemory` ausgelöst, welches eine weitere Sprachausgabe des Roboters ausführt und das Abonnement dieses Events beendet. Nach Schließen eines Fensters wird ebenso das jeweilige Event ausgelöst, welches wiederum eine Sprachausgabe und das Beenden des Abonnements des Events zur Folge hat.

Im Anschluss wird die Führung des TV Raumes durch Aufruf der Methode `resumeTVRoom()` fortgeführt. Die Abonnements der MQTT-Topics aller Fenster werden ebenso beendet. Danach folgt die Spieleszene (`runGamingScene()`). Durch den asynchronen Aufruf der Methode `getRequest(url:String)` der Klasse `ConnectionManager` erfolgt das Abspielen eines Videos auf dem Fernseher. Hierbei wird die entsprechende Url, welche in der Konfigurationsdatei hinterlegt ist (siehe Abschnitt 4.1.1.1) verwendet. Parallel dazu wird noch mithilfe der Klasse `ALAnimationPlayer` eine Animation des Roboters abgespielt (`run(animation:String)`). Danach folgen weitere Sprachausgaben des Roboters, sowie das Beenden des Dialoges. Durch eine Reihe weiterer `navigateTo(x : Float, y: Float): Bool` Befehle erfolgt der Positionswechsel in das Arbeitszimmer. Die navigateTo-Befehle beinhalten zuerst eine Bewegung in x-Richtung, eine 90°-Drehung im Uhrzeigersinn, eine weitere Bewegung in x-Richtung, sowie eine weitere 90°-Drehung im Uhrzeigersinn und eine erneute Bewegung in x-Richtung. Bei Ankunft erfolgt noch eine 180°-Drehung, um sich dem Publikum wieder zuzuwenden. Dort angekommen wird abgeprüft, ob die Tour weiter fortgesetzt, oder beendet werden soll (siehe Abschnitt 4.1.1).

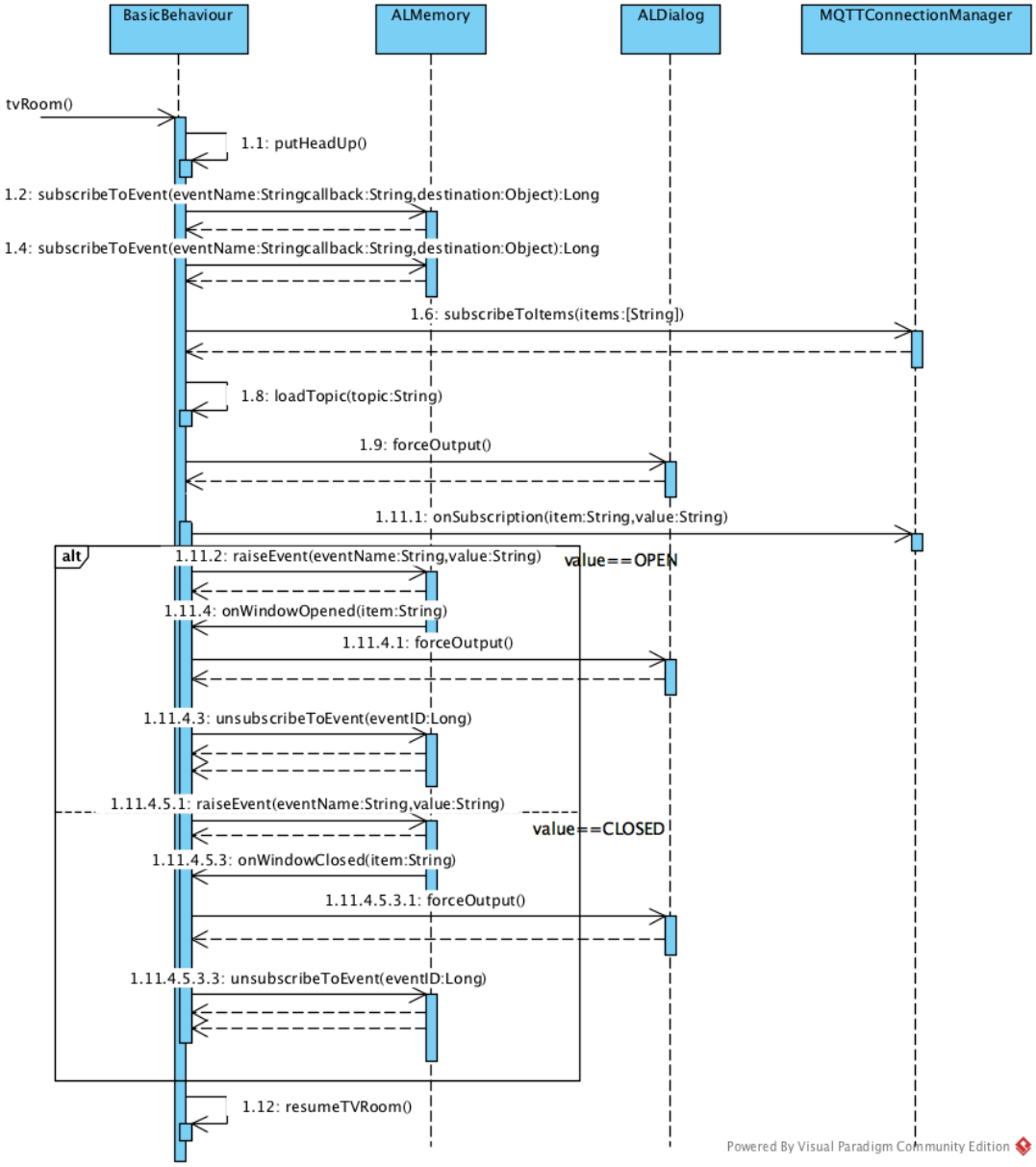

Abbildung 12.: Sequenzdiagramm - Führung im TV Raum Teil 1

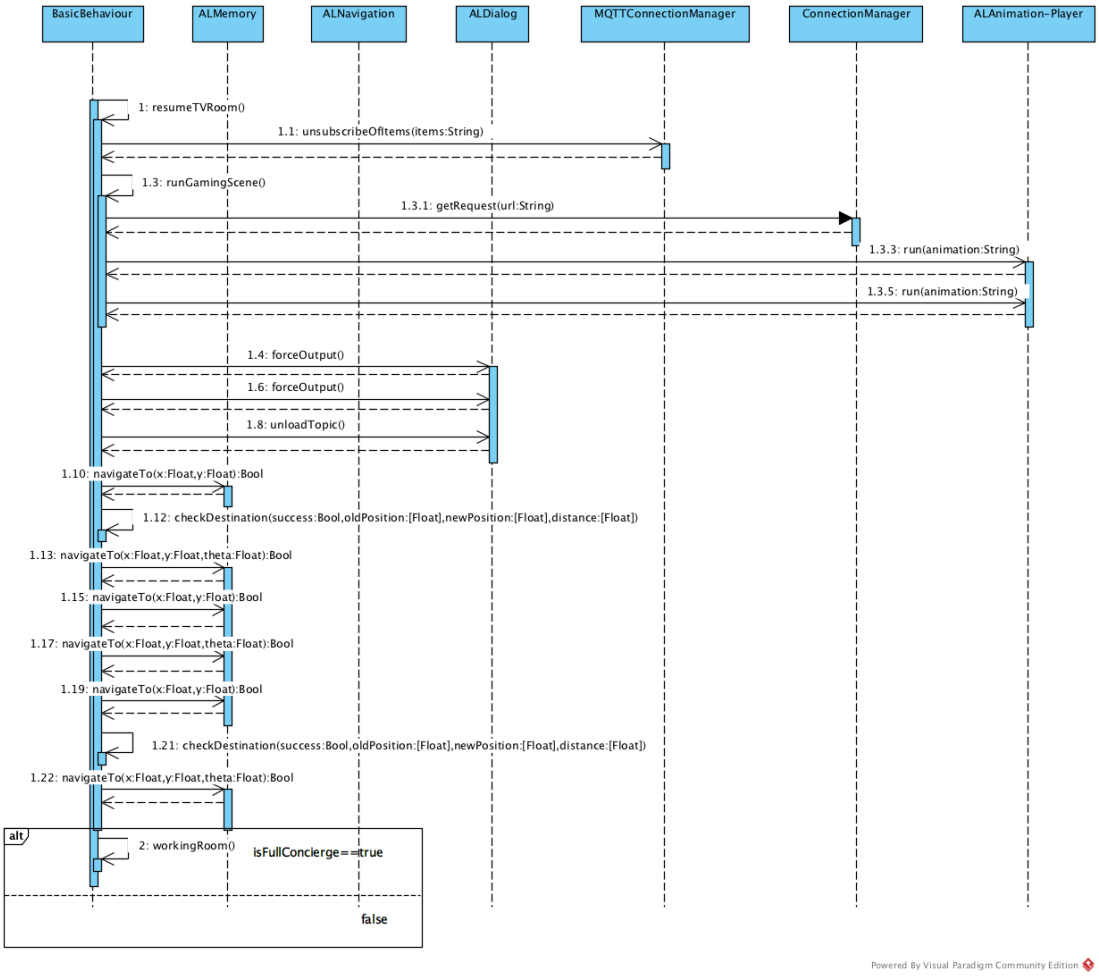

Abbildung 13.: Sequenzdiagramm - Führung im TV Raum Teil 2

4.1.3.3. Arbeitszimmer

Im Arbeitszimmer wird wie auch bei alle den anderen Räumen und Positionen zuerst Peppers Kopf in eine senkrechte Position versetzt, durch Aufrufen der Methode `putHeadUp()`. Im Anschluss wird die entsprechende Dialogdatei `WorkingRoom.top` (siehe Anhang A.2) geladen, das darin enthaltene `proposal` wiedergegeben und das Topic wieder entladen. Danach erfolgt der Positionswechsel in das Badezimmer. Hierfür wird zunächst eine Bewegung in x-Richtung mithilfe eines Aufrufes von `moveTo(x: Float, y : Float, theta: Float): Bool` getätigt, gefolgt von einer Überprüfung über die Ankunft am Ziel, sowie eine mögliche Interpolation an das geplante Ziel (`checkDestination (success: Bool, oldPosition: [Float], newPosition: [Float], distance: [Float])`). Nach einer 90°-Drehung im Uhrzeigersinn wird die Bewegung in x-Richtung fortgesetzt, um das Badezimmer zu betreten, ebenfalls gefolgt von einer Überprüfung über die Ankunft am Ziel. Ist der Roboter im Badezimmer angekommen, so tätigt er eine 180°-Drehung und wendet sich den Teilnehmern der Führung wieder zu. Wie auch

bei den vorherigen Räumen wird überprüft, ob der aktuelle Raum, in diesem Fall das Badezimmer ebenso durchgeführt werden soll oder nicht (siehe Abschnitt 4.1.1).

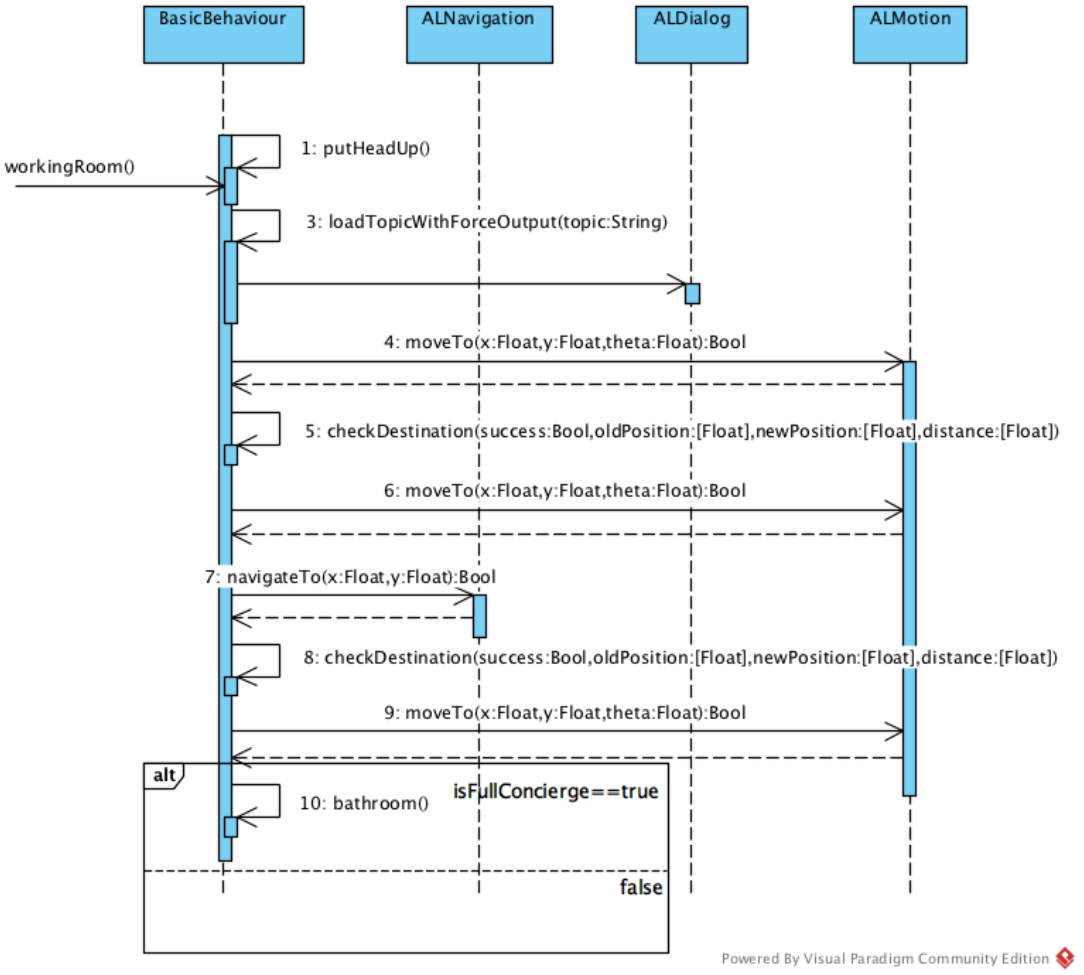

Abbildung 14.: Sequenzdiagramm - Führung im Arbeitszimmer

4.1.3.4. Badezimmer

Zu Beginn im Badezimmer wird der Kopf des Roboters in eine senkrechte Position versetzt. Danach wird die entsprechende Dialogdatei (`Bathroom.top` siehe Anhang A.2) geladen und die erste Sprachausgabe getätigt. Innerhalb des ersten `proposals` wird das Event `GetValue` (siehe Abschnitt 4.1.2) ausgelöst und somit der Temperaturwert von OpenHab (`openHabGetRequest(mqttTopic: String, key:String): [String]`) abgerufen und in `ALMemory` gespeichert. Innerhalb des gleichen `proposals` wird der abgerufene Wert aus dem Speicher geladen und vom Roboter dynamisch wiedergegeben. Im Anschluss folgen weitere zwei Sprachausgaben (`forceOutput()`), wobei die letzte eine Interaktion mit dem Sprachassistenten „Alexa“ darstellt. Nach dem entladen der Dialogdatei folgt die Navigation in die Küche. Hierfür werden, wie in Abbildung 15 zu sehen ist, einige Funktionsaufrufe von `moveTo(x: Float, y: Float, theta: Float)`

`: Bool`, `navigateTo(x: Float, y: Float)` und `checkDestination(success: Bool, oldPosition : [Float], newPosition: [Float], distance: [Float])` durchgeführt. Zuerst erfolgt eine Bewegung in x-Richtung, gefolgt von einer 90°-Drehung im Uhrzeigersinn, sowie einer Navigation in x-Richtung, ebenfalls gefolgt von einer weiteren 90°-Drehung im Uhrzeigersinn, einer weiteren Navigation in x-Richtung und bei Ankunft am Ziel eine 180°-Drehung, damit Pepper wieder in Richtung des Publikums schaut. Eine Überprüfung über die weitere Durchführung der Führung beendet den Abschnitt des Badezimmers.

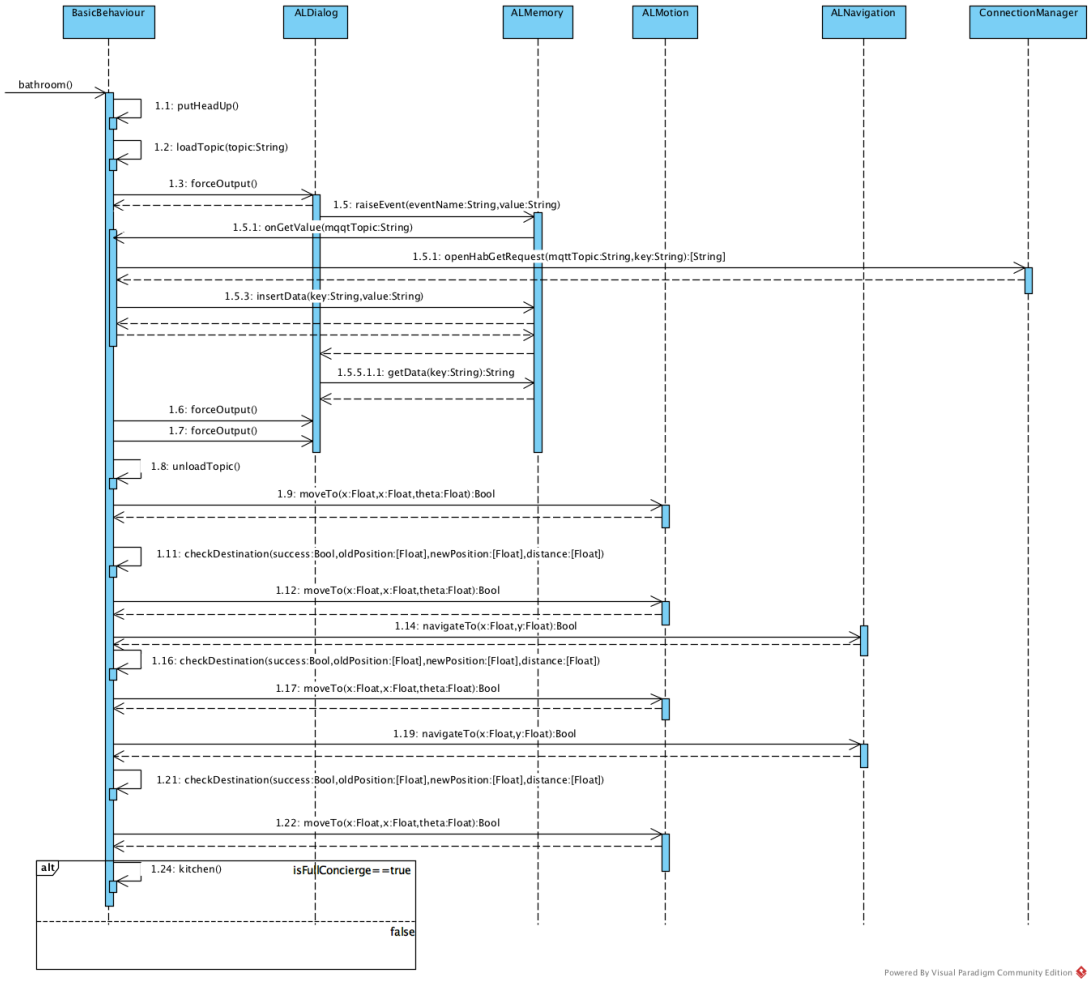

Abbildung 15.: Sequenzdiagramm - Führung im Badezimmer

4.1.3.5. Küche

Der Beginn der Führung in der Küche ist weitgehend identisch mit dem des Badezimmers (siehe Abschnitt 4.1.3.4). Von der Positionierung des Kopfes des Roboters, über das Laden der Dialogdatei (`Kitchen.top` siehe Anhang A.2), sowie das Abrufen der Kühlschranktemperatur über OpenHab, Speichern in `ALMemory` und dynamisches Abrufen des Wertes und einbauen in die Sprachausgabe ergibt sich kein relevanter Unterschied. Die Navigation aus der Küchen in den Empfangsbereich wird durch eine Rei-

he von `moveTo(x: Float, y: Float, theta: Float): Bool`-Befehlen (siehe Abbildung 16) realisiert. Die Navigation baut sich wie folgt auf: eine Bewegung in x-Richtung, gefolgt von einer 90°-Drehung im Uhrzeigersinn, einer weiteren Bewegung in x-Richtung und einer abschließenden 180°-Drehung. Zum Schluss wird wie nach jedem Raum die Fortführung der Tour überprüft und gegebenenfalls fortgesetzt.

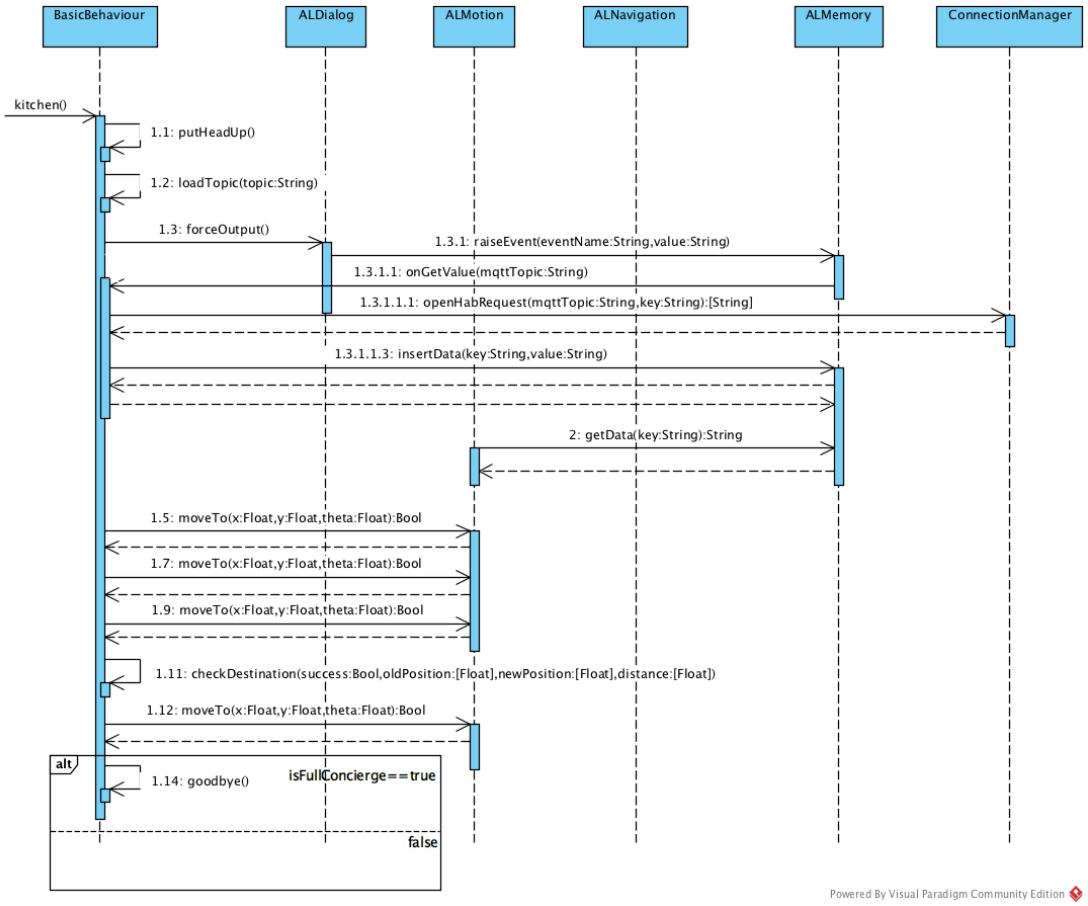

Abbildung 16.: Sequenzdiagramm - Führung in der Küche

4.1.3.6. Abschlussszene

Nach der Positionierung des Kopfes des Roboters, dem Laden der Dialogdatei (`Questions.top` siehe Anhang A.2) und einer Sprachausgabe des Roboters erfolgt die sprachlicher Interaktion in Verbindung mit Pepper. Hierbei können die Besucher Pepper einige Fragen stellen oder Befehle ausführen lassen. Hierzu zählen zum Beispiel Fragen, wie „Wie heißt du?“, „Wie alt bist du?“ oder „Wie viele Geräte gibt es hier im Smart Home Labor?“und Befehle wie „Schalte das Licht ein“ oder „Mach ein Bild“. Die vollständige Liste an möglichen Fragen und Befehlen kann der oben genannten Dialogdatei entnommen werden. Durch die Spracheingabe des Wortes „Ende“ und damit einhergehendes Auslösen des Events `SpeechRecognitionOff` (siehe Abschnitt 4.1.2) wird die Sprachsteuerung beendet und der Roboter setzt seine Tour fort. Zunächst

wird durch Aufruf der Methoden `loadUrl(url:String): Bool` und `showWebview(): Bool` der Klasse `ALTabletService` die Webseite für die Bewertung der Führung auf Peppers Tablet geladen und angezeigt. Die Url wird der Konfigurationsdatei (siehe Abschnitt 4.1.1.1) entnommen. Im Anschluss setzt der Roboter seine Sprachausgabe durch Laden der Dialogdatei `Goodby.top` (siehe Anhang A.2) fort. Die Abschlussshow wird durch Aufrufen der Methode `lightShow()` durchgeführt. Der komplette Befehlsablauf ist in Abbildung 17 aufgrund des Umfangs nicht aufgeführt. Hierbei handelt es sich um eine Menge an MQTT-Befehlen zur Steuerung der Rollläden, Lichter und Türen im Smart Home Labor mit musikalischer Untermalung. Nach der Abschlussshow folgt eine letzte Sprachausgabe des Roboters zur Beendigung der Tour und entladen der Dialogdatei.

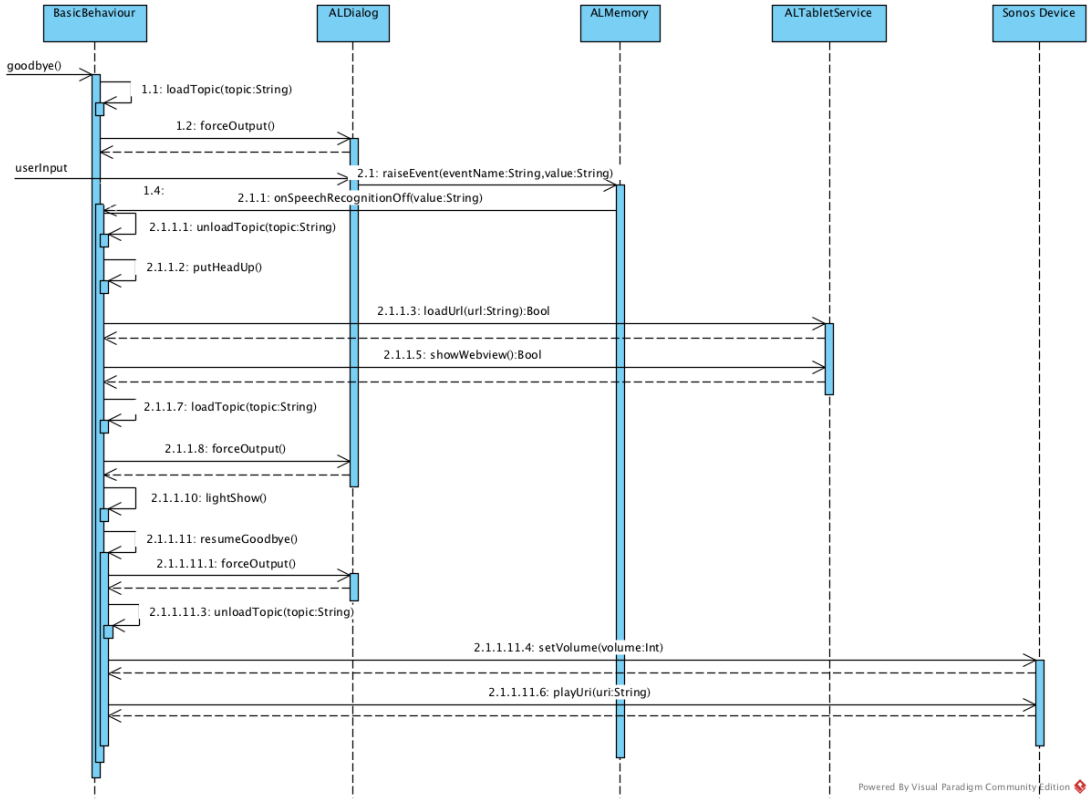

Abbildung 17.: Sequenzdiagramm - Abschlussszene

4.2. Klassendiagramme

4.2.1. Übersicht Klassendiagramm

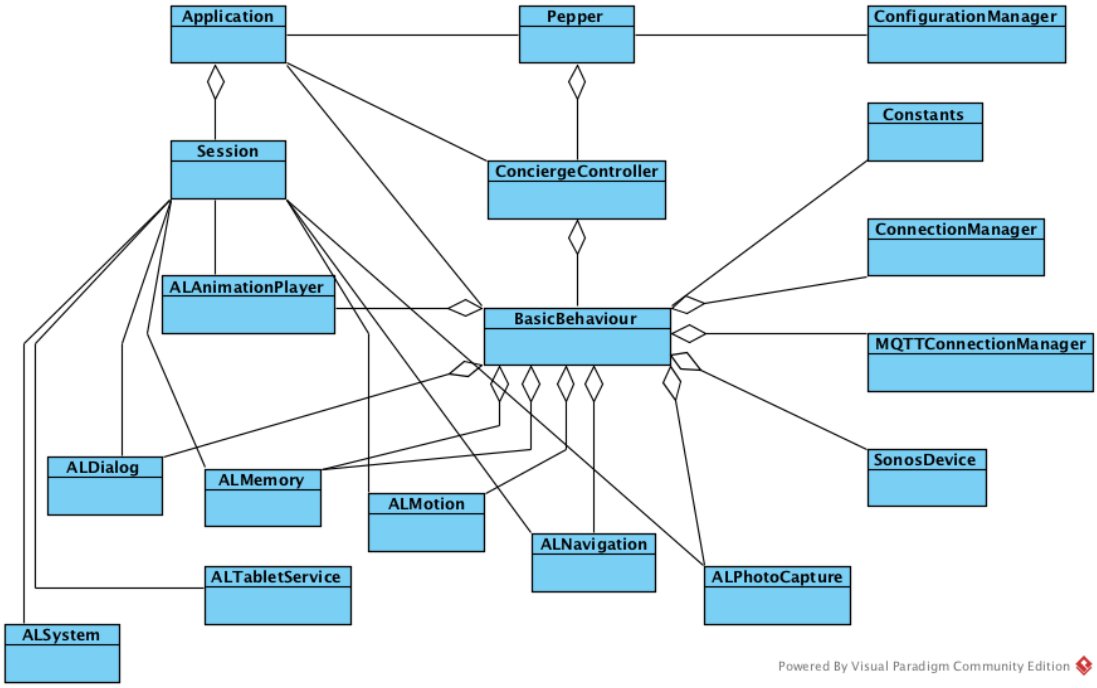

Abbildung 18.: Klassendiagramm - Übersicht aller Klassen

4.2.2. Übersicht der Methoden der Klassen

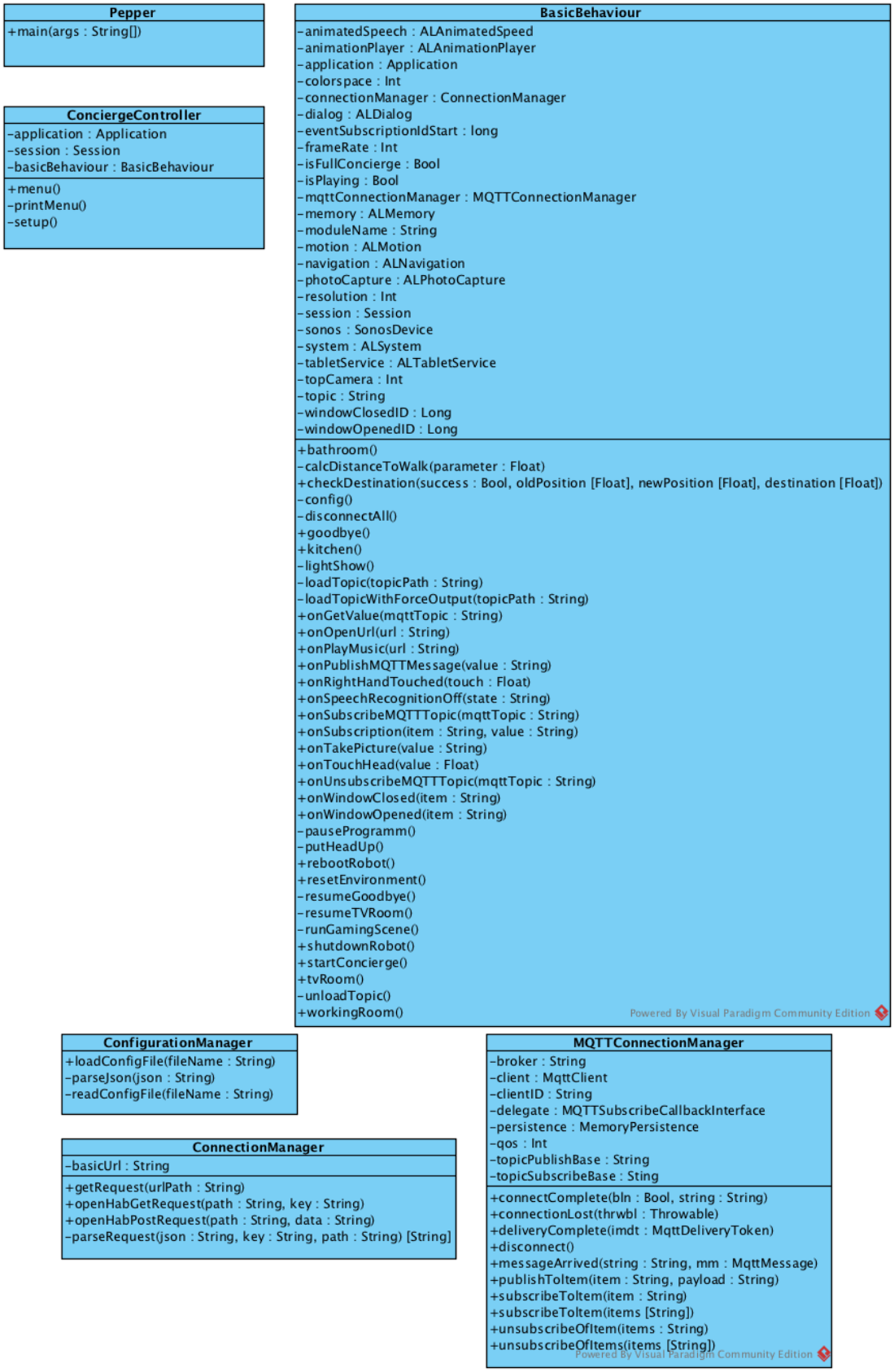

Abbildung 19.: Klassendiagramm - Klassen inklusive Membervariablen und Methoden

5. Anwendungshinweise

Damit der Roboter möglichst effizient und erfolgreich eine Führung durch das Smart Home Labor geben kann, müssen einige Rahmenbedingungen zu Beginn an erfüllt sein. Diese Bedingungen sind nötig, um etwaige Erkennungsschwierigkeiten, wie sie im Smart Home Labor auftreten können, zu kompensieren. Diese Problematik entsteht wesentlich durch auftretende Hindernisse, deren Erkennung und den nötigen Ausweichmanövern. Auch spielen Grenzwerte von Sensoren und tief im System verankerten Schwellwerte für Abstände, Kollisionserkennung und Erschütterungssensorik eine große Rolle. Diese sind dazu im Stande, den Programmablauf des Roboters kurzzeitig zu unterbrechen und beeinflussen dadurch eine flüssige Führung durch das Smart Home Labor.

5.1. Rahmenbedingungen

Bei der Entwicklung wurden daher für die genannten Fälle Rahmenbedingungen festgehalten die einen reibungslosen Ablauf garantieren sollen.

5.1.1. Stühle

Es sollte sichergestellt werden, dass alle Stühle unter den Tisch gestellt werden. Dies soll sicherstellen, dass Pepper den engen Weg zwischen Tisch und Fensterwand erfolgreich passieren kann. Sollten Stühle im weg stehen, können die dünnen Metallbeine von Pepper nur teilweise erkannt werden, wodurch die Wegfindung deutlich beeinträchtigt wird.

5.1.2. Schiebetüren

Die Schiebetüren von Wohnzimmer und Arbeitszimmer sollten sich in mittiger Position zwischen beiden Zimmern befinden. Für Bad und Küche hat sich das verschieben der Türen nach ganz rechts als optimal erwiesen. Das stellt einen möglichst großen Abstand zwischen Tür und Pepper her. Zusätzlich ermöglicht es einen besseren Blick

der Zuschauer, wenn Pepper mit genannten Räumen interagiert. Auch kann Pepper das Zimmer so besser befahren ohne Probleme mit Abständen zu bekommen.

5.1.3. Interaktion mit Fernseher

Pepper interagiert im Laufe seiner Führung mit dem Fernseher im Wohnzimmer. Dazu muss als Vorbedingung der Fernseher eingeschaltet und auf den HDMI-Kanal gestellt werden, an welchem der Videoausgang des Raspberry-Pis angeschlossen. Zusätzlich muss der Raspberry-Pi, der sich hinter dem Fernseher befindet, mit Strom versorgt werden.
Ist alles korrekt verkabelt, sollte der Boot-Verlauf des Pis auf dem Fernseher zu sehen sein. Der Pi ist so konfiguriert, dass nach kurzer Zeit das Bild auf schwarz gesetzt wird. Dies soll die Illusion erschaffen, dass der Fernseher ausgeschaltet ist und Pepper ihn durch seine Interaktion aktiviert. Optional lässt sich der Fernseher auch etwas in die Blickrichtung der Zuschauer drehen, damit verfolgt werden kann was passiert.

5.1.4. Interaktion mit Sonos

Pepper wird zum Ende seiner Führung eine Abschlusssequenz vorführen. Dazu muss das Sonos System aktiviert sein und bestenfalls, für einen optimalen Klang, die Intensität des Subwoofers im TV-Raum auf rund 70% eingestellt sein.

5.2. Positionen

Bei der Planung der Führung wurden initiale Positionen festgelegt, an denen Pepper im Optimalfall stehen soll. An jeder dieser Positionen findet eine Interaktion statt, sodass es ratsam ist, sollte Pepper die jeweilige Position nicht selbst finden, ihn an die jeweilige Position zu schieben, um einen weiteren planmäßigen Ablauf zu garantieren. Die Orte dafür sind in folgenden Abschnitten genauer ersichtlich und erklären vorliegende Bedingungen. Die genauen Standorte, sind mit Hilfslinien in grün, in jeder Abbildung ersichtlich und spiegeln die optimale Position wider.

5.2.1. Startpunkt

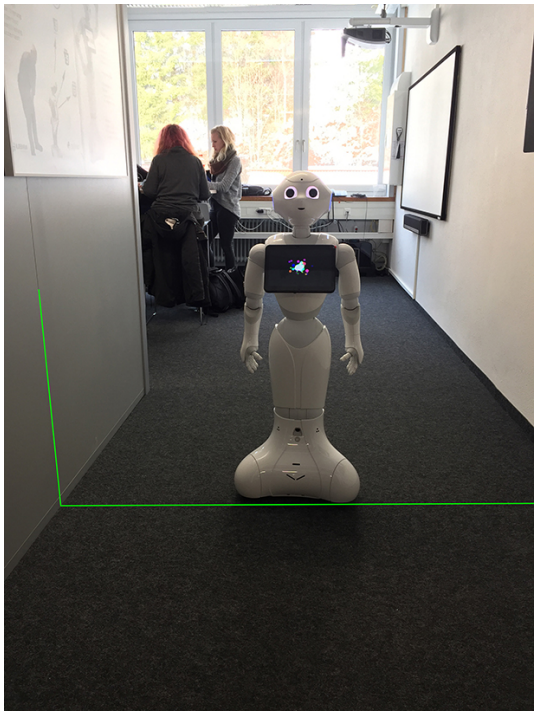

Abbildung 20.: Startpunkt: Eingangsbereich

Zu Beginn an sollte Pepper nahe der Tür positioniert werden. Hierbei sollte darauf geachtet werden das Pepper mittig zwischen den beiden Wänden steht und eine Linie mit der Nut der Holzwand bildet. Der Abstand zu beiden Seiten sollte hierbei rund 50 cm betragen, wie in Abbildung 20 zu sehen.

5.2.2. Wohnzimmer

Die optimale Position von Pepper befindet sich im Wohnzimmer auf Höhe des Bettes, mit einem Abstand von rund 25 cm zwischen Bett und Metallschrank, wie in Abbil-

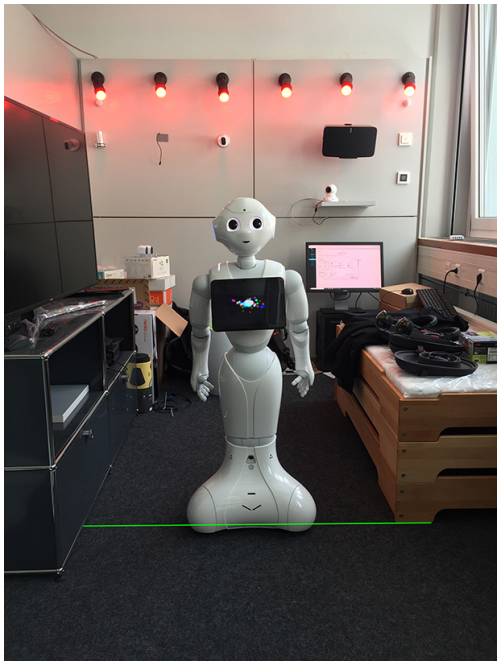

Abbildung 21.: Position: Wohnzimmer

dung 21 ersichtlich. Dies lässt Raum für Interaktion und lässt Pepper ohne größere Schwierigkeiten den Raum betreten, sowie verlassen.

5.2.3. Arbeitszimmer

Pepper erwartet seinen Standort im Arbeitszimmer etwa 1 m von der Türschwelle entfernt. Dies ist nicht ganz mittig aus Sicht des Eingangs, ist aber nötig um etwas Abstand zum Touch-Table zu gewähren. Abbildung 22 gibt einen Eindruck von den Abständen.

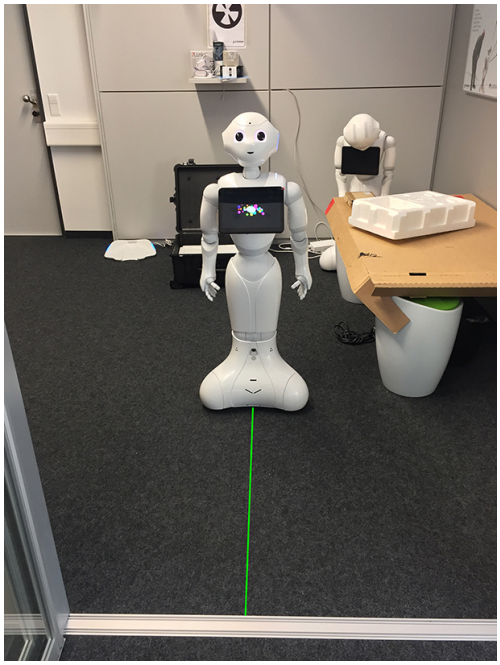

Abbildung 22.: Position: Arbeitszimmer

5.2.4. Badezimmer

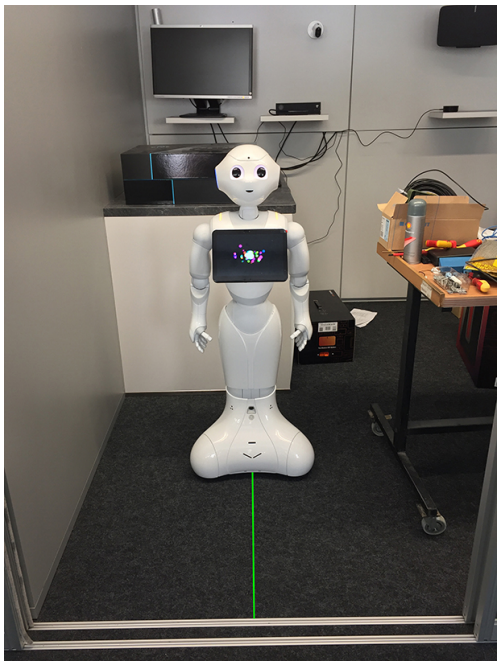

Abbildung 23.: Position: Badezimmer

Im Badezimmer befindet sich der vorgesehene Standort etwa 1 m im Raum, mittig der Tür-Führungsschiene. Wie in 5.1.2 erwähnt, sollten die Türen für Badezimmer und Küche, nach Möglichkeit ganz nach rechts geschoben sein.

5.2.5. Küche

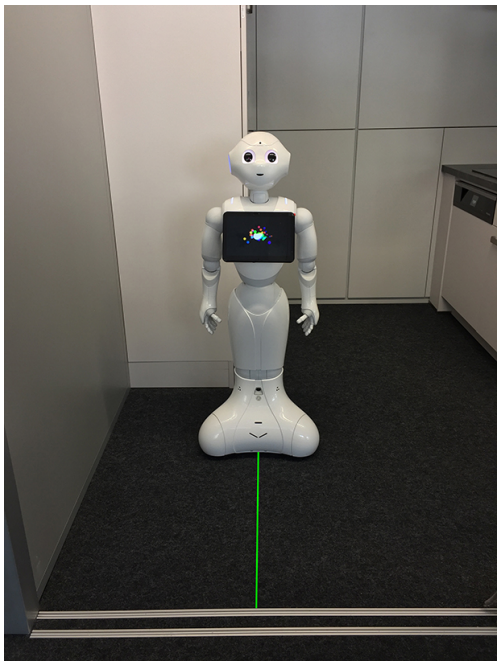

Abbildung 24.: Position: Küche

Für die Küche sind die gleichen Bedingungen wie für das Badezimmer (siehe Abschnitt 5.2.4) vorgesehen. Türen nach Möglichkeit komplett an die rechte Seite schieben, der optimale Standort ist somit 1 m innerhalb des Raumes, mittig der Tür-Führungsschiene.

6. Modifikation und Inbetriebnahme der Anwendung

6.1. Inbetriebnahme

Für die Verwendung der Anwendung müssen Java in der Version 8 oder neuer installiert sein und die „Pepper.jar“- und „concierge.conf“-Datei im selben Verzeichnis liegen. Mit dem Befehl „java -jar Pepper.jar“ kann die Anwendung gestartet werden. ARM-basierte Architekturen werden vom Hersteller Aldebaran mit deren SDKs nicht unterstützt, deshalb ist die Ausführung nur auf x86-Architekturen, aber mit jedem Betriebssystem (Windows, Mac, Linux) möglich. Unter Windows muss Java in der 32-Bit-Version installiert sein.

6.2. Anpassung und Bearbeitung des Quellcodes

Die komplette Anwendung ist in Java geschrieben und der Quellcode, sowie eine einsatzbereite .jar-Datei, welche die notwendigen Bibliotheken für jedes Betriebssystem (Windows, Mac, Linux) enthält, liegen der Dokumentation bei. Die Anwendung kann mit einer geeigneten IDE bearbeitet werden. Erstellt wurde sie mit Netbeans.

6.3. Anpassung und Bearbeitung durch Konfigurations- und Dialogdateien

Ebenso bestehen einige Möglichkeiten, die Anwendung durch die Konfigurationsdatei und die Dialogdateien zu modifizieren ohne die Anwendung neu kompilieren zu müssen. Die Modifikationen über die Konfigurationsdatei können in Kapitel 4.1.1.1 nachgelesen werden. Modifikationen über die Dialogdateien können folgende sein:

- Änderung der Texte, die von Pepper wiedergegeben werden, welche jeweils durch ein `proposal` widergespiegelt werden
- `proposals` können entfernt werden, zusätzlich hinzugefügte `proposals` werden bei der Wiedergabe nicht berücksichtigt

- Hinzufügen (oder entfernen) von Animationen, wie zum Beispiel `^start(dialog_move_head /animations/LookRight)`
- Events auslösen
  - Veröffentlichen eines MQTT-Werts auf einem MQTT-Topic, wie zum Beispiel eine Türe im TV-Raum undurchsichtig schalten: `^pCall(ALMemory.raiseEvent("PublishMQTTMessage", "HMScheibentransparenz1_1_State;OFF"))`
  - Abrufen eines Wertes von OpenHab und speichern in `ALMemory` wie zum Beispiel der Raumtemperatur: `^pCall(ALMemory.raiseEvent("GetValue", "NAInnen raumsensorBad_Temperature"))`
  - Abrufen eines Wertes von `ALMemory` wie zum Beispiel den zuvor abgerufenen Wert der Raumtemperatur: `^call(ALMemory.getData("NAInnenraumsensorBad_ Temperature"))`
  - Ein Bild von Pepper erstellen und auf seinem Tablet anzeigen lassen: `^pCall(ALMemory.raiseEvent("TakePicture", "test"))`
  - Ein Musikstück auf der Sonos-Anlage wiedergeben: `^pCall(ALMemory.raiseEvent ("PlayMusic","x-file-cifs://192.168.0.10/Medialib/Audio/path/to/file.mp3"))`

    Achtung: auf die richtigen Präfixe achten (siehe Kapitel 4.1.2)
  - Öffnen einer Webseite auf Peppers Tablet: `^pCall(ALMemory.raiseEvent(" OpenUrl", "http://192.168.0.11:8080/"))`
- Verwendung von dynamischen Variablen wie zum Beispiel `$Dialog/RobotName`. Weitere dynamische Variablen können der Dokumentation des Herstellers [alda] entnommen werden

6.4. Wichtige Befehle zur Verwendung innerhalb einer Dialogdatei

- Reaktion Peppers auf eine Nutzereingabe wie zum Beispiel:
  `u:(Wie alt bist du) 1 Jahr alt.`
  `u:(Wie alt bist du)` beschreibt die Spracheingabe, die der Nutzer tätigen muss. Alle nachfolgenden Sätze in der selben Zeile werden als Sprachausgabe von Pepper wiedergegeben
- Verwendung von dynamischen Variablen wie zum Beispiel: `$Dialog/Hour` gibt die aktuelle Stunde an

- Verwendung von Animationen durch folgende Befehle:
  `^start(path), ^wait(path), ^run(path).`
  `^start(path)` startet eine Animation. Die Spracheingabe wird ohne Unterbrechung fortgesetzt. `^wait(path)` wartet, bis eine zuvor gestartete Animation beendet ist. `^run(path)` unterbricht die aktuelle Sprachausgabe, spielt die Animation vollständig ab und setzt danach die Sprachausgabe wieder fort

- Ausführen von Aktionen/Aufrufen von Methoden:
  `^call(command), ^sCall(command), ^pCall(command).`
  Es können nur Naoqi-Methoden aufgerufen werden, keine eigen erstellten.
  `^call(command)` ruft einen Wert ab, wie zum Beispiel einen Wert von `ALMemory`. Die Evaluation der Aufrufe erfolgt zu Beginn eines Satzes. Gefolgt von `c1:(_*) $1` in einer neuen Zeile wird der abgerufene Wert dynamisch in die Sprachausgabe eingesetzt. `^sCall(command)` ruft eine Methode auf und verarbeitet das Ergebnis. Die Evaluation des Aufrufs erfolgt direkt an der jeweiligen Stelle im Satz und unterbricht die Sprachausgabe. `^pCall(command)` erlaubt den asynchronen Aufruf einer Methode. Zusätzliche Informationen können der Dokumentation von Aldebaran [aldb] entnommen werden.

- Mit `^break` kann die Reevaluation des nachfolgenden Satzes angestoßen werden. Dies ist notwendig, wenn das Kommando `^call(command)` verwendet wird und sich seit Beginn der Sprachausgabe die Werte, wie zum Beispiel die gespeicherten Werte in `ALMemory`, geändert haben

- Reaktion auf eine Berührung des Roboter:
  `u:([e:FrontTactilTouched]) Fass mich nicht an!`

- Verwendung von `concepts` aus der eigen erstellten `lexicon_ged.top`-Datei zur Kapselung von Synonymen eines Wortes oder von Phrasen:
  `u:(~Wie_heisst_du) Ich heiße $Dialog/RobotName`

- Verwendung von mehreren verschiedenen oder optionalen Nutzereingaben:
  `u:(~Kannst_du {einen} Kaffee [machen kochen]) Leider noch nicht.`
  Angaben in geschweiften Klammern sind optionale Eingaben, Angaben in eckigen Klammern stellen eine mögliche Auswahl von Eingaben dar.

6.5. Besonderheiten bei der Programmierung

- Der Aufruf der Demo-Webseite für Pepper (http://192.168.0.20/pr) funktioniert nur über http, https führt zu einem fehlerhaften Nachladen der JavaScript-Ressourcen

- Nach der Verwendung einer dynamischen Variable `$Dialog/RobotName` in einem Satz in einem Dialog muss ein Whitespace folgen
- Es existiert keine `lexicon_ged.top`-Datei zur Kapselung von Synonymen eines Wortes oder von Phrasen von Seiten des Herstellers, deshalb haben wir eine eigene Datei erstellt (siehe Anhang A.1).
- Dialogdateien müssen immer in einem Verzeichnis auf dem Roboter liegen! Für diese Anwendung wurde das Home-Verzeichnis (/home/nao) gewählt
- Nach dem Hochfahren des Roboters ist die direkte Ausgabe eines `proposals` nicht möglich. Es muss zuerst eine Nutzereingabe (reale Nutzereingabe oder programmatische Nutzereingabe (`dialog.forceInput("xxx")`) macht keinen Unterschied) erfolgen
- Die Speicherung von Nutzereingaben im Dialog und anschließende Verwendung ist aktuell nicht in deutscher Sprache möglich (nur Englisch und Japanisch)
- Bei Fortbewegung des Roboters durch `moveTo` oder `navigateTo` existiert ein lang anhaltender Fehlerstatus im Hintergrund, der dazu führt, dass schnell aufeinander folgende Befehle übersprungen werden
- Sonarsensoren können nicht aktiviert werden bzw. die Callback-Funktionen von `ALMemory` liefern kein Ergebnis
- Pepper und sein Tablet sind zwei komplett eigenständige Geräte, d.h. Pepper und das Tablet haben eigene WLAN-Module und müssen somit auch getrennt voneinander eingerichtet werden (nur einmal notwendig)
- Auf Peppers Tablet können entweder URLs aufgerufen werden, oder Dateien, die lokal in Peppers Dateisystem abgelegt sind (remote Verzeichnisse werden nicht akzeptiert), angezeigt werden. Bei der Anzeige von lokalen Dateien müssen diese in einem speziellen `PackageManager`-Verzeichnis (`/home/nao/.local/share/PackageManager/apps/<application directory>/html`) abgelegt sein. Für den Aufruf, bzw. der Anzeige der Dateien auf Peppers Tablet, muss dessen interne IP-Adresse inklusive dem Anwendungspfad verwendet werden (`http://198.18.0.1/apps/<application directory>/`).

7. Ausblick

Der humanoide Roboter Pepper bietet mit den ausgestatteten Sensoren einen zukunftssicheren Einsatzbereich in den von uns behandelten Concierge-Funktionen. Die Anwendungsbereiche und Orte sind vielseitig vorstellbar, Hotel, Verkaufsräume, Flughäfen oder in Museen. Überall dort in denen viele Menschen auf wenige Mitarbeiter treffen, macht ein Concierge-Service mittels Roboter Sinn. Pepper ist ein sympathischer Roboter der sich exzellent mit verbalen, aber auch mit non-verbalen Mitteln verständigen kann. Dies prädestiniert ihn für den Einsatz mit Menschen.

Speziell für den aktuellen Anwendungsfall, der Concierge-Funktion für das Smart Home Labor, können noch einige Erweiterungen und Anpassungen in zukünftigen Versionen erfolgen.

- Steuerung weiterer Smart Home-Geräte per Sprache
- Steuerung der Smart Home-Geräte per Tablet
- Steuerung von Gerätegruppen (z.B. alle Lichter im Badezimmer einschalten)
- Gesichts-/Personenerkennung bei Raumwechsel (sicherstellen, dass die Besucher genügend Abstand zu Pepper einhalten und erst danach wird die Navigation in den nächsten Raum gestartet)
- Speicherung und Auswertung der Bewertungen der Führung
- Auslagerung der Concierge-Anwendung auf einen Server
- Steuerung der Concierge-Funktionen per App
- Objekterkennung von Smart Home-Geräten (falls ein Besucher Fragen zu einem Gerät hat, aber die Bezeichnung des Geräts nicht weiß, kann Pepper das Gerät erkennen)
- Dynamische Beschreibung verschiedener Profile für unterschiedliche Einsatzgebiete des Roboters (z.B. Smart Home Lab, Museum, ...) per JSON- oder XML-Datei (**Hinweis:** sehr großer Arbeitsaufwand und sehr gute Programmierkenntnisse werden benötigt)

- Navigation verbessern durch Erstellen einer virtuellen Karte durch Pepper und anschließender Navigation innerhalb der Karte (**Hinweis:** Aldebaran arbeitet bereits an dieser Funktion, ist aber aktuell noch nicht verwendbar)
- Falls mehrere Anwendungen für verschiedene Funktionen (Concierge, Water Buddy, ...) für Pepper existieren, eine App zur Steuerung der Funktionalitäten entwickeln
- „Follow-Me"-Funktion, Pepper über die fest installierten Schalter im Smart Home Labor rufen, feststellen, woher der Aufruf kam und dann in den jeweiligen Raum fahren

8. Fazit

Nachdem die Entwicklung von Concierge-Funktionen mit Pepper stattgefunden hat, lässt sich abschließend sagen, dass es durchaus möglich ist, eine Führung durch das Smart Home Labor zu realisieren. Allerdings ist zum aktuellen Zeitpunkt der Entwicklung die Software und die SDKs von Seiten des Herstellers noch lange nicht ausgereift und es fühlt sich in der Verwendung mehr nach einer Beta-Variante an. Zudem ist die Entwicklung einer sinnvollen Anwendung für die Steuerung des Peppers mit der grafischen Obefläche, dem Choreograph, nicht möglich. Hier entstehen zu viele Abhängigkeiten, die vom Entwickler nicht beeinflusst werden können. Es ist also ratsam, eines der SDKs (Python, Java, C++,...) zu verwenden und die Funktionalitäten eigenständig zu entwickeln. Ebenso ist eine dynamische Einsetzung des Roboters an jedem beliebigen Ort zum aktuellen Zeitpunkt fast undenkbar, denn hierfür gibt es zu viele Abhängigkeiten von der Umwelt, seien es die Navigation im Raum oder die Interaktion mit zusätzlichen Geräten. Wird Seitens der Hersteller eine Funktionalität integriert, die es ermöglicht, von der aktuellen Umgebung des Roboters eine interaktive Karte für die Navigation zu erstellen, so ist ein sinnvoller Einsatz des Peppers durchaus denkbar. Mit der entwickelten Anwendung zur Umsetzung von Concierge-Funktionen im Smart Home Labor existiert eine sehr gute Grundlage, um darauf aufbauend weitere Funktionalitäten zu implementieren und die Dynamisierung zu erhöhen. Ebenso ist eine Steuerung der Infrastruktur des Smart Home Labors in Verbindung mit MQTT und OpenHAB nun sehr leicht und kann ebenso in andere Anwendungen integriert werden.

Literaturverzeichnis

[alda] *ALDialog API*. http://doc.aldebaran.com/2-5/naoqi/interaction/dialog/aldialog-api.html#event-list, Abruf: 01.01.2018

[aldb] *Qi-Chat Syntax*. http://doc.aldebaran.com/2-5/naoqi/interaction/dialog/dialog-syntax_full.html, Abruf: 01.01.2018

Eidesstattliche Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbständig verfasst und hierzu keine anderen als die angegebenen Hilfsmittel verwendet habe. Alle Stellen der Arbeit die wörtlich oder sinngemäß aus fremden Quellen entnommen wurden, sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt oder an anderer Stelle veröffentlicht.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben kann.

Furtwangen, 25.01.2018, Caroline Fichtner

Furtwangen, 25.01.2018, Erik Meiß

Furtwangen, 25.01.2018, Lothar Mödl

Furtwangen, 25.01.2018, Julian Keller

A Dateien

A.1. Deutsches Lexikon

```
topic: ~lexicon()
language: ged

#pronunciation:(I) [i]

dynamic:robotname
dynamic:all_descriptions
dynamic:all_languages
dynamic:myrobotname

dynamic:app-name
dynamic:app-dance
dynamic:app-game
dynamic:app-story

#================================================================
===#
##### Voice tweaking #####
#concept:(neutral) \style=neutral\ \rspd=100\ \vct=110\
#concept:(neutral_slow) \style=neutral\ \vct=105\ \rspd=100\
#concept:(joyful) \style=joyful\ \rspd=90\ \vct=95\
#concept:(didactic) \style=didactic\ \rspd=105\ \vct=110\
#================================================================
===#

#================================================================
===#
##### Verbs #####
#----------------------------------------------------------------#
concept:(möchte) ["[möchte will "würde [sehr liebend]"] gerne"]
#================================================================
===#

#================================================================
===#
##### 1rst person verbs #####
#----------------------------------------------------------------#
concept:(Ich_bin) ["Ich bin" "Mein Name ist"]
concept:(Ich_habe) ["Ich habe" "Ich hab"]
concept:(Ich_werde) ["Ich werde"]
concept:(Ich_wuerde) ["Ich würde {gerne}"]
concept:(Ich_bin_nicht) ["Ich bin nicht" "Bin nicht"]
concept:(Ich_habe_nicht) ["Ich habe nicht" "Ich hab nicht" "Das hab ich nicht" "Das
habe ich nicht"]
concept:(Ich_tue_nicht) ["Ich tue nicht" "Das machte ich nicht" "Das tue ich nicht"]
concept:(Ich_werde_nicht) ["Ich werde nicht" "Das werde ich nicht" "Ich möchte das
nicht"]
concept:(Ich_wuerde_nicht) ["Ich würde das nicht" "Würde das nicht"]
concept:(Ich_muss) ["Ich muss"]
concept:(Ich_will) ["Ich will" "Ich beabsichitge" "Ich wünsche"]
```

```
concept:(Ich_moechte_nicht) ["Ich möchte nicht" "Möchte [lieber] nicht"]
concept:(Ich_moechte) ["Ich möchte" "Ich möchte [sehr] gerne" "Liebend gerne"]
concept:(Wir_sind) ["Wir sind"]
concept:(Wir_sind_nicht) ["Wir sind nicht"]

#-----------------------------------------------------------------#
##### 2nd person verbs #####
#-----------------------------------------------------------------#
concept:(Du_bist) ["Du bist"]
concept:(Du_warst) ["Du warst"]
concept:(Du_hast) ["Du hast"]
concept:(Du_wirst) ["Du wirst"]
concept:(Du_bist_nicht) ["Du bist nicht" "Bist du nicht"]
concept:(Du_hast_nicht) ["Du hast nicht" "Das hast du nicht"]
concept:(Du_warst_nicht) ["Du warst das nicht" "Das warst du nicht"]
concept:(Du_wirst_nicht) ["Du wirst nicht" "Das wirst du nicht"]

#-----------------------------------------------------------------#
##### 3rd person verbs #####
#-----------------------------------------------------------------#
concept:(es) [es "[das] {Ding hier}"]
#concept:(es_ist) ["es ist" "das ist" "[es das]" "{"Für meine Begriffe" "Meiner
Auffassung nach" "Meines Erachtens nach" "Ohne Zweifel"}" ["ist es" "ist das"]]"
concept:(es_ist_nicht) ["Das ist es nicht" "Das war es nicht" "Das wars nicht"]
#=================================================================
===#

#=================================================================
===#
##### Other verbs #####
#-----------------------------------------------------------------#
concept:(vb_imp_sprechen) "sprich nicht {"mit mir"} {über}"
concept:(vb_imp_plaudern) "plauder nicht {"mit mir"} {über}"
concept:(vb_imp_sagen) "Sag mir nicht {darüber}"
concept:(vb_imp_diskutieren) "Diskutier das nicht mit mir"
concept:(vb_imp_lästern) "Läster nicht {"darüber"}"
concept:(verbs_imp_talk) [~vb_imp_sprechen ~vb_imp_plaudern ~vb_imp_sagen
~vb_imp_diskutieren ~vb_imp_lästern]
concept:(Erzähl_mir) ["Erzähl mir" "Erzähls mir"]
#=================================================================
===#

#=================================================================
===#
##### Adjectives and adverbs #####
#-----------------------------------------------------------------#
concept:(verlangen) [
   "[begierig sehnen] nach"
   "bessesen von"
   "fiebern nach"
```

```
   "brenne auf"
]
concept:(groß) [groß enorm bedeutsam bedeutend gewichtig]
concept:(klein) [klein winzig marginal gering kaum überschaubar]
concept:(Groesse) [~groß ~klein]
#----------------------------------------------------------------#
concept:(adverbs) [wirklich unbedingt völlig wirklich tierisch absolut perfekt ungern
zu besonders derart genauso bis sehr kaum ziemlich]
#================================================================
===#

#================================================================
===#
##### Vocabulary #####
#----------------------------------------------------------------#
concept:(apps_sing) [application app activity programm software]
concept:(apps_plur) [applications apps activities programe software]
concept:(Monate) [Januar Februar März April Mai Juni Juli August September
Oktober November Dezember]
concept:(über) [über]
#----------------------------------------------------------------#
concept:(Tage) [Montag Dienstag Mittwoch Donnerstag Freitag Samstag Sonntag]
#================================================================
===#

#================================================================
===#
##### Questions to the robot #####
#----------------------------------------------------------------#
concept:(Hast_du) ["Hast du"]
concept:(Magst_du) [
   "Magst du"
   "Was denkst du darüber"
   "Verstehst du das"
   "Wie gehts ["geht es"] dir"
]
concept:(Weisst_du) ["Weißt du [wer was] das ist" "Hast du [davon] gehört"]
concept:(Kannst_du) [
   "[Kannst würdest könntest möchtest] du {bitte}"
   "Denkst du du kannst das"
   "Bist du bereit"
   "Weißt du wie"
   "Was kannst du"
]
concept:(Bist_du) ["Bist du"]
concept:(Warst_du) ["Warst du"]
concept:(Wo_bist_du) ["Wo bist du" "Weiß du wo du bist"]
concept:(Wer_bist_du) ["Wer bist du" "Weiß du wer du bist" "Du bist wer"]
concept:(Was_ist) [
   "Erzähl mir" "Was ist das"
```

```
    "Was ist das"
    "Was sind"
    "Was war"
]
concept:(Wer_ist) [
    "{"Weißt du" "Kannst du mir sagen"} wer [das ist]"
    ["Weißt du"]
]
concept:(Wo_ist) ["Wo ist" "Wo kann ich das finden"]
concept:(Kann_ich) [
    "[Kann könnte] [ich]"
    "Würde es dir was ausmachen wenn ich"
    "Ist es möglich [dass "dass ich"]"
]
concept:(Wie_heisst_du) [
    "Wie heißt du"
    "wie [darf kann] ich dich nennen"
    "Erzähl mir wer du bist"
    "Wer bist du"
]
concept:(Wie_ist_der_hallo) [
    "Wie lautet der [Name Titel]"
    "Wie lautet [ihr sein der] [Name Titel]"
]
concept:(Ueber_was_geht_es) [
    "Worüber geht es"
    "Was [erzählst sagst] du"
    "Worum geht es {"in der [Diskussion Konversation]"}"
]
concept:(Wie_viel_ist) [
    "Wie viel ["kostet das" "ist das wert"]"
]
#-----------------------------------------------------------------#

#-----------------------------------------------------------------#
##### Orders to the robot #####
#-----------------------------------------------------------------#
concept:(helfen) [
    "{"Würdest" "Kannst du"} mir helfen {bitte}"
"Kannst du mir {einige} Informationen geben"
    "Ich brauche [Informationen Details Hilfe]"
]

concept:(stoppen) [
    stopp
    "hör auf"
    "["Kannst du" "Könntest du"] {bitte} ["leise sein" aufhören]"
    "fuck off"
    "sei leise"
]
```

```
concept:(stop_it) ~stoppen
concept:(wieder_sprechen) [
   "Sprich mit mir"
   "{"Du kannst"} wieder ["mit mir"] sprechen"
]

concept:(stop_application) [
   "[stopp schließ verlasse] [die das den] [Software App Applikation Anwendung
Programm Dialog]"
   "stoppe das"
   "Das reicht"
]
concept:(restart_application) [
   "[restart "starte wieder" reboot replay] [die das den] [Software App Applikation
Anwendung Programm Dialog]"
   "Geh wieder zum [Anfang Start]"
   "Zurück bitte"
]
concept:(Wiederholen) [
   "Wiederhole {mir} [deine die] {letzte} Frage"
   "Was war die letzte Frage"
   "Wiederhole ["den Satz" "deinen letzten Satz" "Was hast du eben gesagt"]
{wieder} {bitte}"
   "Sag es mir wieder"
   "Was [war ist] die Frage"
   "Was ["hast du" [gesagt gefragt]]"
   "Stell die Frage noch einmal"
   "Wiederhole [das bitte]"
   "[Sag Frage] {das} nochmal"
   "Kannst du [das die [Animation Bewegung Ding Sound Geräusch]]" nochmal
machen" {bitte}"
]
concept:(stop_talking) [~vb_imp_sprechen ~vb_imp_plaudern ~vb_imp_sagen
~vb_imp_diskutieren ~vb_imp_lästern "stop talking"]
concept:(lets_talk_about) [sprechen "sprich mit mir"]
concept:(come_to_me) [
   "[Komm lauf] [hier näher] ["zu mir"]"
]
concept:(show_me) [
"Zeig {ihm ihr mir uns ihnen denen}"
"Kannst dus {ihm ihr mir uns ihnen denen} zeigen"
"Kann ich das sehen"
]
#================================================================
===#

#================================================================
===#
##### Human inputs #####
#----------------------------------------------------------------#
```

```
concept:(hallo) ^rand[
   Hallo Hey Hi Morgen Mahlzeit
   "hey ihr"
   "{Guten} [Morgen Mittag Abend]"
]
concept:(thank_you) [
   "Vielen Dank"
   "Danke Dir"
]
concept:(i_dont_know) [
   "Ich weiß nichts {darüber über}"
   "Ich weiß gar nichts"
   "Was ist das"
   "Ich weiß [es] nicht"
   "Davon habe ich noch nie gehört"
]
concept:(i_didnt_understand) [
   "{Sorry} Ich habe dich nicht verstanden"
   "Das habe ich nicht verstanden"
"Was ist [das "das hier"]"
"Was meinst du [wolltest du sagen]"
   "Was versucht du zu sagen"
   "Das macht keinen Sinn"
]
concept:(you_didnt_understand) [
   "Du verstehst kein Wort von dem was ich sage"
"Du hast es nicht verstanden"
   "Du liegst falsch"
   "Da liegst du falsch"
]
concept:(im_not_sure) [
   "Ich bin mir da nicht sicher"
   "Weiß [ich] nicht"
   "Bin mir nicht sicher"
   "keine Idee"
]
concept:(its_not_true) [
   "Das stimmt [so] nicht"
   "Das ist nicht wahr"
]
concept:(its_a_stupid_question) [
   "Das ist {wirklich} eine {komplett wahrhaft} [dumme idiotische uninteressante]
Frage"
]
concept:(schaetzen) [
   rate schätze vermute rate
   "was denkst du"
   "was rätst du mir"
   "Was würdest du mir empfehlen"
   "Wie lautet dein Rat"
```

```
]
concept:(Ich_denke) [
    "Ich denke"
    "Ich schätze"
    "Ich bin mir sicher"
    "natürlich"
    "meine Antwort ist"
    "das ist"
    "das war"
    "es könnte sein"
    "ich würde [sagen schätzen antworten]"
]
concept:(ich_werde_versuchen) [
    "Lass es uns versuchen"
    "Das werde ich versuchen"
]
#-------------------------------------------------------------------#
concept:(Ja) ^rand[
    "{oh} [Ja Jep Jap Yeah] {Bitte "ich auch"}"
    "[ok okay] {sicher "warum nicht"}"
    gut
    richtig
    nagut
    "alles klar"
    super
    natürlich
    "sicher {"Das mache ich"}"
    "Los gehts"
    "Du liegst richtig"
    perfekt
    "warum nicht"
    absolut
    total
    definitiv
    "{Ja Jep Jap Yeah} Das wäre super"
    "Wenn du magst"
    "Kein Problem"
    "[super exzellente] Idee"
    exakt
]
#-------------------------------------------------------------------#
concept:(Nein) ^rand[
    "[Nein Nope Nah Nö Ne No "Nix {da}"] {Danke}"
    "no {fucking} way"
    "{Nein} nicht wirklich"
    "Das interessiert mich nicht"
    "{Nein} ["Das möchte ich nicht"]"
    "Ich bin mir da nicht sicher"
    "Das sehe ich nicht so"
    "Das ist falsch"
```

```
    "Nicht wirklich"
    "Auf keinen Fall"
]
#-----------------------------------------------------------------#
##### Human mood #####
#-----------------------------------------------------------------#
concept:(Mir_gehts_gut) [
    "Mir ["geht es" gehts] [sehr absolut auch] gut"
    "nicht schlecht"
    "Ich fühle mich fit"
    super
    perfekt
    läuft
    "{sehr} gut {danke}"
    "hey yeah Mir geht es gut"
    "geht gut"
    "auch gut"
]
concept:(Mir_gehts_schlecht) [
    "Mir geht es nicht so gut"
    "Ich bin nicht in der besten Verfassung"
    "Ging schon mal besser"
    "nicht {sehr so} gut"
    "passt schon"
    "Geht so"
]
#=================================================================
===#

#=================================================================
===#
##### Sonstiges #####
concept:(happy_human) [wundervoll exzellent super "das ist {wirklich} cool" "das ist
[super gut wundervoll awesome]" "Du bist {wirklich sehr} nett"]
#=================================================================
===#
concept:(what_were_you_saying) [
    "Was war [die deine] Frage"
    "Über was [redest plauderst sprichst] du"
]



#=================================================================
===#
##### DEPRECATED #####
#-----------------------------------------------------------------#
concept:(Raueme) [Küche Badezimmer Toilette Wohnzimmer Schlafzimmer Flur
Hausgang Kinderzimmer Fernsehrraum]
#-----------------------------------------------------------------#
```

```
concept:(myname)[
    "Mein Name ist"
    "Nenn mich"
    "Mich kennt man als"
]
```

A.2. Dialogdateien

Welcome.top

topic: ~welcome()
language: ged

u:(xxx) Herzlich Willkommen im Smart Home Labor der Hochschule Furtwangen.
proposal: Mein Name ist $Dialog/RobotName . Ich stehe euch für eine Führung zur freien Verfügung. Bitte folgt mir!

General.top

topic: ~general()
language: ged

proposal: Zuerst ein paar allgemeine Informationen zum Labor. Eingerichtet wurde es mit dem Ziel Smart Home Komponenten im Kontext einer Wohnung zu erforschen. Dabei wird ein besonderes Augenmerk auf die Interaktion der Geräte untereinander und die Entwicklung weitere Jus Cases sowie deren ausführliches testing gelegt. Es besteht die Möglichkeit Komponenten über das Netzwerk fern zu steuern und auch deren Status abzufragen. Auf einer Fläche von 8 mal 12 Metern sind hierzu vier Räume angelegt mit insgesamt weit über 100 Smart Home Geräten angefangen bei intelligenten Lampen bis hin zur Smart Waschmaschine. Weiter geht es nun in den ersten Raum, den TV-Raum. Bitte folgt mir auffällig.

TVRoom.top

topic: ~tvroom()
language: ged

proposal: Hier befinden wir uns nun im Fernsehzimmer. Puh... Ist euch auch so warm wie mir? ^start(dialog_move_head/animations/LookLeft) ^start(dialog_move_arms/animations/UpLArm) ^start(dialog_move_arms/animations/StretchLArm) Ich komme leider nicht an das Fenster heran. Könnte einer mir bitte helfen und ein Fenster öffnen?

proposal: ^run(animations/Stand/Waiting/MysticalPower_1) Ist das nicht wundervoll? Durch das Öffnen der Fenster werden automatisch hier im Raum die Lampen aktiviert. Sobald das Fenster wieder geschlossen wird, geht das Licht wieder aus. Könnte einer das Fenster bitte wieder schließen?

proposal: Sehr gut, das Fenster ist wieder geschlossen. Jetzt möchte ich aber noch in Ruhe eine Runde Nied for spied spielen.

proposal: Aaahhh... das war geil! Zum Schutz der Privatsfähre kann man auch die Scheiben undurchsichtig ^pCall(ALMemory.raiseEvent("PublishMQTTMessage", "HMScheibentransparenz1_1_State;OFF")) machen.
proposal: Jetzt sind wir aber fertig hier. Macht etwas Platz für mich, jetzt geht es weiter in das Arbeitszimmer.

WorkingRoom.top

topic: ~workingroom()
language: ged

proposal: Hier befinden wir uns nun im Arbeitszimmer. An den Wänden seht ihr sogenannte Nao Marks. Ich bin hier auch noch in ein anderes Projekt involviert, bei dem ich als Woter Baddy dafür sorge, dass vorwiegend ältere Menschen ausreichend Flüssigkeit zu sich nehmen. Im Zweifel bringe ich ihnen auch Wasser. Diese Marks ermöglichen es mir mit einfachen Mitteln mich im Raum zu orientieren und meine Position zu bestimmen. Dies ist vor allem zu Beginn von der Entwicklung von Programmen für mich von großem Nutzen. Zu meiner Linken seht ihr den Ideum Tatsch Teibl, der zum Arbeiten genutzt werden kann. Außerdem arbeitet ein anders Tiem damit, um pseudo 3D Hologramme darzustellen. Hinter den Wänden befindet sich die gesamte Technik für die Steuerung aller Geräte hier im Raum. Jetzt geht es aber weiter in den nächsten Raum. Bitte macht Platz und lasst mich durch.

Bathroom.top

topic: ~bathroom()
language: ged

proposal: ^pCall(ALMemory.raiseEvent("GetValue", "NAInnenraumsensorBad_Temperature")) ^pCall(ALMemory.raiseEvent("GetValue", "NAInnenraumsensorBad_CO2")) Das ist das Badezimmer. Leider kein stilles örtchen wie wahrscheinlich bei euch zu Hause, aber trotzdem ein toller Raum. Ja, ihr seht richtig, hier ist keine Toilette, aber als Elektro-Roboter brauche ich auch keinen Ölwechsel ^run(animations/Stand/Emotions/Positive/Laugh_1). Hier gibt es aber eine Menge anderer toller Geräte, die über das Netzwerk überwacht und gesteuert werden können, wie eine Waschmaschine und einen Trockner. Die Temperatur hier im Raum beträgt ^break ^call(ALMemory.getData("NAInnenraumsensorBad_Temperature")) Grad. Der CO 2 Wert beträgt ^break ^call(ALMemory.getData("NAInnenraumsensorBad_CO2")) ppm . Jetzt gehen wir auch schon in den letzten Raum, die Küche. Bitte macht den Weg für mich frei und folgt mir.
  c1:(_*) $1

proposal: Halt wartet! Wo ist eigentlich Siri? ^start(dialog_move_head/animations/LookRight) ^wait(dialog_move_head/animations/LookRight) ^start(dialog_move_head/animations/LookLeft) ^wait(dialog_move_head/animations/LookLeft) Ich muss aufpassen, dass mir Siri nicht über den Weg läuft. Die ist sauer, weil Alexa immer mit mir flirtet. ^start(animations/Stand/Waiting/Think_1) ^wait(animations/Stand/Waiting/Think_1) Alexa ^start(animations/Stand/Waiting/Think_1) ^wait(animations/Stand/Waiting/Think_1) aktiviere Privatsphäre. ^run(animations/Stand/Waiting/Think_1)

proposal: Ok gut, jetzt kann ich unbemerkt in die Küche gehen. Bitte macht den Weg für mich frei.

Kitchen.top

topic: ~kitchen()
language: ged

proposal: ^pCall(ALMemory.raiseEvent("GetValue", "MieleFridgeFreezer_CurrentTemperatureFridge")) Nun sind wir im wichtigsten Raum der Wohnung angekommen, der Küche. Hier gibt es informatikerfreundlichen starken Kaffee, einen Kühlschrank und auch einen intelligenten Herd. Ich könnte jetzt natürlich den Herd einschalten, aber ich will ja nicht das ganze Labor abfackeln, deshalb teile ich euch nur die Temperatur des Kühlschranks mit, welche aktuell ^break ^call(ALMemory.getData("MieleFridgeFreezer_CurrentTemperatureFridge")) Grad beträgt.
c1:(_*) $1

Goodby.top

topic: ~goodby()
language: ged

proposal: Ich möchte mich recht herzlich bei euch bedanken, dass ihr die Tour mit mir gemacht habt. Außerdem möchte ich noch meinen 4 Entwicklern, Caroline Fichtner, Eric Meiß, Julian Keller und natürlich Lothar Mödl danken, die mir das ganze Semester über so viele tolle Funktionen beigebracht haben. Wenn ihr nachher den Raum verlasst, würde ich mich freuen, wenn ihr die Tour noch auf ^start(animations/Stand/Gestures/ShowTablet_1) meinem Tablet bewerten würdet. Jetzt wünsche ich euch noch viel Spaß bei meiner Abschlussschow.
proposal: Ich hoffe, es hat euch gefallen und wir sehen uns bald wieder ^start(animations/Stand/Gestures/ShowTablet_1) und bewerten nicht vergessen. Bis bald ^run(animations/Stand/Gestures/Hey_1)

Questions.top

topic: ~questions()
language: ged

include: lexicon_ged.top

```
proposal: Nun sind wir schon fast am Ende angekommen. Habt ihr noch Fragen?
Solltet ihr keine Fragen mehr haben, dann sagt doch einer bitte Ende, damit ich
weiter machen kann.

u:([e:FrontTactilTouched e:MiddleTactilTouched e:RearTactilTouched]) Autsch, hör
auf mich anzufassen!

u:(~Wie_heisst_du) Ich heiße $Dialog/RobotName , aber da wir Freunde sind, darfst
du mich auch Peppi nennen.
#Und wie heißt du?    u1:(_*) ~hallo $1 . Freut mich dich kennenzulernen

u:(Peppi) Hallo I bims hier.

u:(["Was hast du {davor} gesagt" ~Wiederholen]) Ich sagte $Dialog/Answered

u:(e:Dialog/NotUnderstood) ^rand["Das habe ich jetzt nicht verstanden." "Was
laberscht du" ]

u:(["Welches Datum" "{Den} wie vielten"] haben wir {heute}) Heute ist der $Dialog/
DateCode

u:(["Wie spät ist es" "Wie viel Uhr ["ist es" "haben wir"]"]) Es ist jetzt $Dialog/Hour
Uhr $Dialog/Minute

u:({"[Schalte Schalt] das"} Licht ein) Ok, ich schalte das Licht ein
^pCall(ALMemory.raiseEvent("PublishMQTTMessage",
"Multimediawand_HUE6_Toggle;ON"))
^pCall(ALMemory.raiseEvent("PublishMQTTMessage",
"Multimediawand_HUE5_Toggle;ON"))
u:({"[Schalte Schalt] das"} Licht aus) Ok, ich schalte das Licht aus
^pCall(ALMemory.raiseEvent("PublishMQTTMessage",
"Multimediawand_HUE6_Toggle;OFF"))
^pCall(ALMemory.raiseEvent("PublishMQTTMessage",
"Multimediawand_HUE5_Toggle;OFF"))

u:(In welcher Position bist du) Meine Position ist ^call(ALRobotPosture.getPosture())
    c1:(_*) $1  # say any result from call
    c1:(crouch) Meine Position ist crouch.

u:(~hallo) ~hallo
u:(ende) ok ich schalte die sprachsteuerung aus
^pCall(ALMemory.raiseEvent("SpeechRecognitionOff", "OFF"))

u:([Erzähl Erzähle] {mir einen} Witz) Das Gras ist hoch, es ist kaum zu überblicken,
darin kann man gut... Na, weißt du's?
    u1:(~nein) Man bist du schlecht. Verstecken spielen natürlich
    u1:(ficken) Was bist du denn für ein Schwein. Verstecken spielen natürlich

u:(Wie alt bist du) Das weiß ich nicht genau, aber denke irgendwas zwischen 1 und
3 Jahre.
```

u:(Wie groß bist du) Ich bin kleiner als du, aber trotzdem ein großer Hitzkopf ^run(animations/Stand/Emotions/Positive/Laugh_1)

u:([~Bist_du "Ob du"] [dumm behindert] {[bist bischt] "hab ich dich gefragt"}) Ich bin eine Maschine, wenn dann bist du dumm, du Penner

u:(Wie ["geht es" "gehts"] dir) ^rand[~Mir_gehts_gut "So la la, ein paar Schaltkreise könnten mal wieder eine Lötkur gebrauchen"] und dir?
  u1:(~Mir_gehts_gut) Das freut mich aber
  u1:(~Mir_gehts_schlecht) Oh, warum das?
    u2:(*) Das tut mir leid, ich wünsche dir eine gute Besserung

u:(Magst du mich) Das weiß ich noch nicht, dafür müsste ich dich erstmal besser kennen lernen

u:(~Kannst_du [mich mir "die Tour" "die Führung" nochmal [machen herumführen zeigen]]) Natürlich kann ich das, aber die Zeit drängt und andere möchten auch noch meine Führung machen, deshalb würde ich das Ganze erstmal vertagen

u:(Wie schnell [bist kannst] du {fahren laufen}) Schneller als eine Schnecke, aber langsamer als du

u:(~Kannst_du [herkommen "zu mir kommen"]) Da muss ich erstmal meine Programmierer fragen. Er hat mir gesagt, ich soll nicht zu nahe an fremde Personen herantreten

u:(Wie ["gefällt es dir" "findest du es"] hier {so}) ^rand["Sehr gut, ich fühle mich hier wie Zuhause" "Oh nee, hier komm isch mir vor wie im Omma paradies"]

u:(~Kannst_du kochen) Nein, leider nicht, aber du könntest ja was für mich kochen. Meine Lieblingsspeise ist Ökostorm mit ner richtig guten Ladung ^startSound(Aldebaran/enu_ono_laugh_10)

u:(["~Bist_du {ein} [männlich Mann] oder {eine} [weiblich Frau]" Welches Geschlecht [~Bist_du "hast du"]]) Ich habe kein Geschlecht. Ich bin einfach ein liebenswerter Roboter

u:(~Kannst_du nicht) Alles, was mir mein Programmierer verboten hat, wie zum Beispiel putzen

u:(Wie viel ["hast du gekostet" "kostest du"]) ^startSound(Aldebaran/enu_ono_laugh_1) Das wüsstest du jetzt gerne, was

u:(Hast du Freund) Ja klar! Neben mir gibt es hier im Labor noch einen Pepper und ein paar kleine Nao Freunde hab ich auch noch

u:(~Kannst_du fliegen) Leider nur auf die Nase ^startSound(Aldebaran/enu_ono_laugh_2)

u:(~Kannst_du mir bei [den meinen] Hausaufgaben helfen) Könnte ich natürlich, aber ich glaube, das gefällt deinen Lehrern gar nicht

u:(["Wie findest du" ~Magst_du] die [HFU Hochschule]) Meeeeeeega geil

u:(Rauchst du) Nein, das wäre für meine Schaltkreise ziemlich ungesund

u:(Wie groß ist das {"Smart Home"} Labor) 8 mal 12 Meter

u:(Wie [werden "steuerst du"] die Geräte {hier gesteuert "hier gesteuert"}) Alle Geräte hier werden über das MQTT Protokoll mit Hilfe eines MQTT Brokers gesteuert. Über die kostenlose Plattform Open Hab kannst du als Mensch auch die Geräte ganz leicht steuern.

u:(Wie viele Geräte gibt es {hier "im Labor" "hier im Labor"}) Mehr als ein Dutzend Großgeräte und über 100 Kleingeräte

u:(~Kannst_du {einen} Kaffee [machen kochen]) Leider noch nicht. Die Kaffeemaschine will noch nicht mit mir reden, aber bald

u:(~Magst_du Kaffee) Isch ja ekelhaft, lass mich in Ruhe mit dem Zeug. Meine Entwickler lieben die braune Brühe aber

u:(Was [spielst zockst] du {[gerne "am liebsten"]}) G T A oder Nied for spied natürlich

u:([~Magst_du "liebst du"] mich) Wenn du lieb zu mir bist, bin ich auch lieb zu dir!

u:(* bims) I bims 1 neicer Roboter am bien

u:(Vong) ^rand["I bims 1 Roboter vong neisigkeit her sehr hoch" "Mobile Systeme ist 1 Studium vong Masterlevel her sehr hoch" "Wie macht das denn der Daimler vong Fahren her"
"Rosem simd rot Glühwein ist das was jedem gefällt i habe mir 1 Pizza bestellt, komm bitte her umd leg dich drauf, denn auf mein Pizza fehlt 1 Lauch"
"I treff mi heut mit X und Y. Halo i bims am absoluten Nullpunkt"
"Wow ab jedzd dürfem wir in mathe 1 Taschenrechmer benutzem. Jedzd wird Mathe so einfach. Halo i bims dumm wie brod"
"Wer leuft so spät durch Nacht umd singt? Es ist dem Studemt vong Rausch fast blimd. Er hält ihm sicher, hält ihm warm dem Döner in seim Arm."
"Rosem simd Rot, der Glühweim ist heif, zumge verbrannt, was für 1 leif"
"Lass die amderen sich verändern umd bleib so wie du bimst. Halo i bims die e funktiom beim ableitem"
]

u:(Was ~Kannst_du alles) Alles, was mir meine Entwickler beigebracht haben. Von sprechen über tanzen bis hin zum Steuern von Geräten im Smart Home

u:(~Kannst_du tanzen) Na klar, soll ich es dir zeigen?
u1:(~Ja) Ok, dann zeig ich es dir ^run(animations/Stand/Waiting/AirGuitar_1)
u1:(~Nein) Schade, aber ich zeig es dir trotzdem, einfach, weil ich es kann und du

```
nichts dagegen machen kannst ^run(animations/Stand/Waiting/AirGuitar_1)
^run(animations/Stand/Emotions/Positive/Laugh_2)

u:(Gefällt) Gefällt mir auch

u:({["Was ist" "Wie lautet"]} das Jugendwort des Jahres) I bims natürlich, weißt du
das etwa nicht?
   u1:(~Ja) Warum fragst du dann überhaupt, wenn du es sowieso weißt
   u1:(~Nein) Man man man, das geht ja mal gar nicht. Du lebst wohl noch im
achzehnten Jahrhundert

u:({Was ist} zwei plus zwei) Tu plus tu ist for, minus won, ist thrie, quick maths

u:(Spiel * bims [Lied Song]) Ok, ich spiele den I Bims Song
^pCall(ALMemory.raiseEvent("PlayMusic", "x-file-cifs://192.168.0.10/Medialib/Audio/
Pepper/Reece_MIKI-I_Bims.mp3"))

u:(Öffne Open Hab) Ok, ich lade Open Hab auf meinem Tablet
^pCall(ALMemory.raiseEvent("OpenUrl", "http://192.168.0.11:8080/"))

u:(["Wie viel Grad hat es {hier}" "Welche Temperatur ["hat es" "haben wir"] {hier}"])
^pCall(ALMemory.raiseEvent("GetValue", "NAInnenraumsensorBad_Temperature"))
Aktuell haben wir hier ^break
^call(ALMemory.getData("NAInnenraumsensorBad_Temperature")) Grad
   c1:(_*) $1

u:(Mach ein [Bild Selfie]) Ok, ich mach ein Bild von dir
^pCall(ALMemory.raiseEvent("TakePicture", "test"))
```