

SQD+KD

31. Dezember 2017

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.

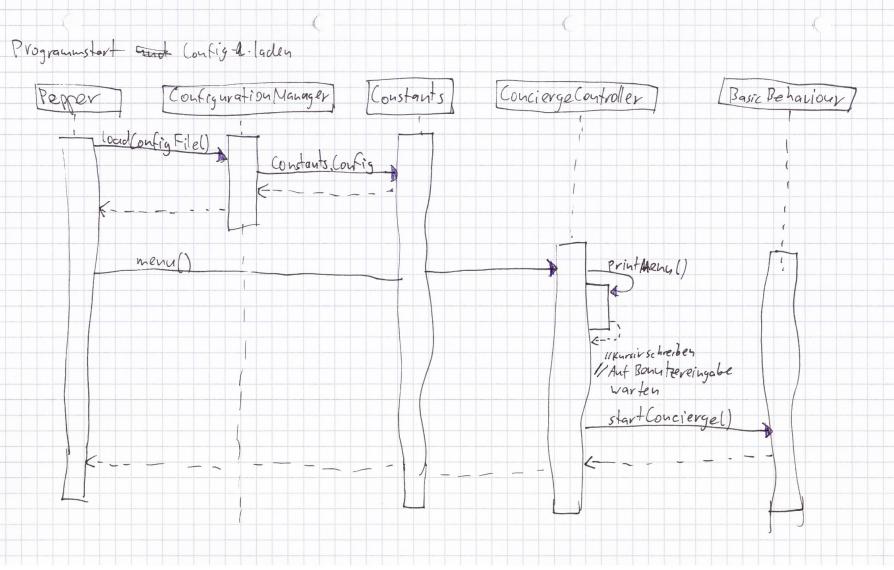


Abbildung 1: Sequenzdiagramm - Programmstart und laden der Konfigurationsdatei

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 ??) 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 ??). Kann keine Konfigurationsdatei im Verzeichnis gefunden werden, so werden die oben genannten Standardwerte verwendet.

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 (`enableWifi()`), sowie, falls schon eine Webseite oder ein Bild auf dem Tablet angezeigt wird, werden dies ausgeblendet (`hideWebView()`, `hideImage()`). Durch Aufruf der Methode `wakeUp()` wird der Roboter aus seiner Ruheposition in eine aufrechte Position versetzt. Danach werden noch entsprechende Event-Handler durch mehrfachen Aufruf der Methode `subscribeToEvent(eventName: String, callback: String, destination: → Object): Long` für nachfolgende Events, welche mithilfe der `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 veröffentlichte 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 veröffentlichten 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 Innenrollen und die beiden schaltbaren Türen.

3 Führung

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)` lädt die entsprechende Dialogdatei (`Welcome.top`). 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.

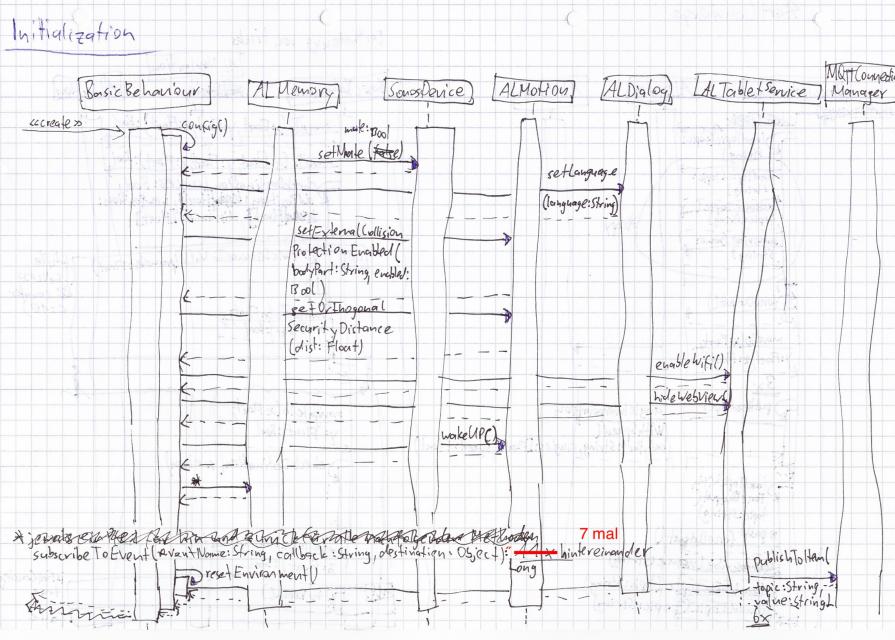


Abbildung 2: Sequenzdiagramm - Initialisierung mit Standardwerten

net 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 gesetzt, 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, 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, welche 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 1) gesteuert werden.

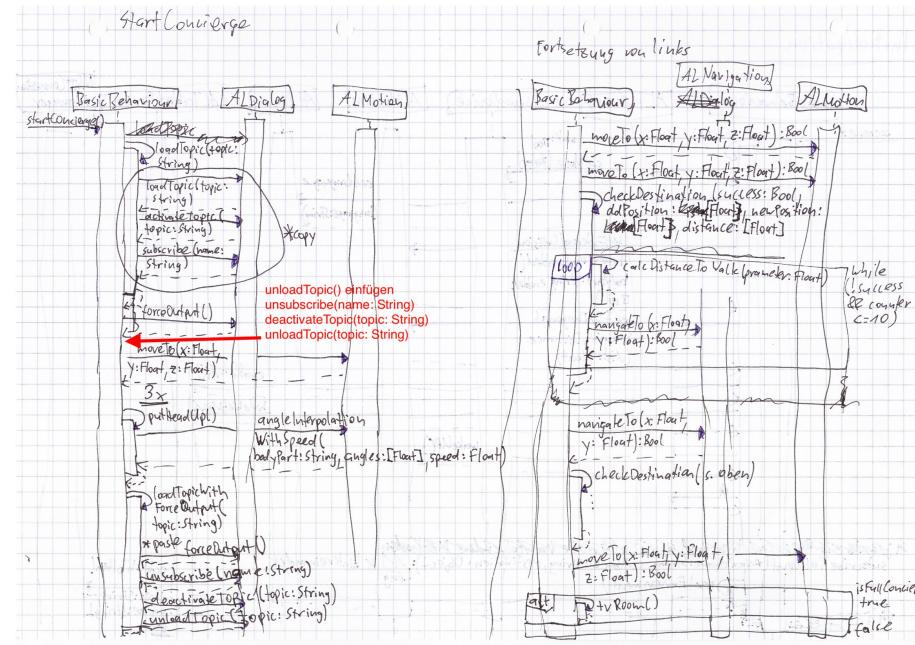


Abbildung 3: Sequenzdiagramm - Start der Führung

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`) und der Sprachausgabe des Roboters wird auf das Öffnen einer 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.

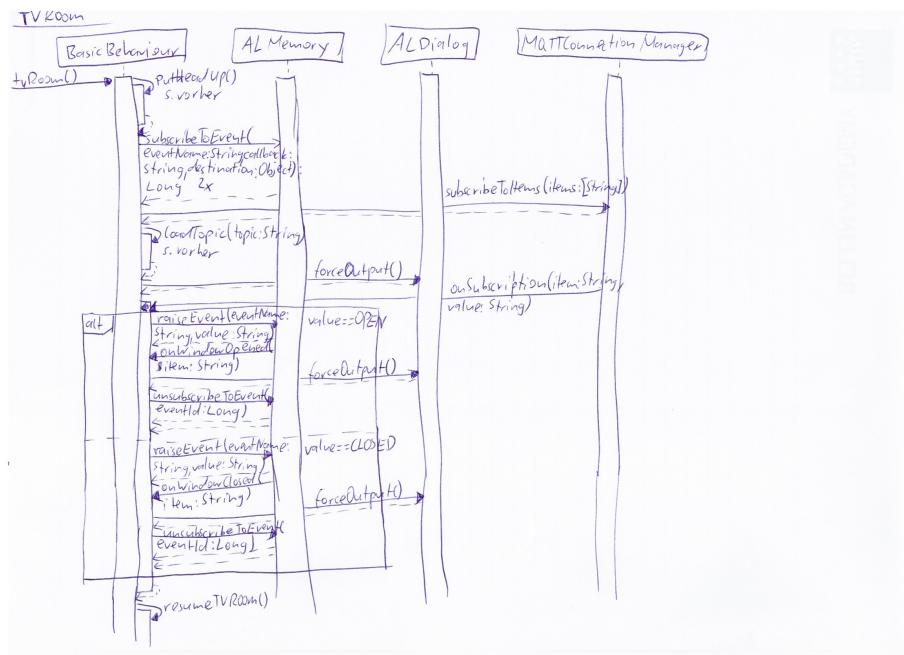


Abbildung 4: Sequenzdiagramm - Führung im TV Raum Teil 1

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 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 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` geladen, das darin enthaltene `proposal` wiedergegeben und das Topic wieder entladen. Danach erfolgt der Positionswechsel in das Ba-

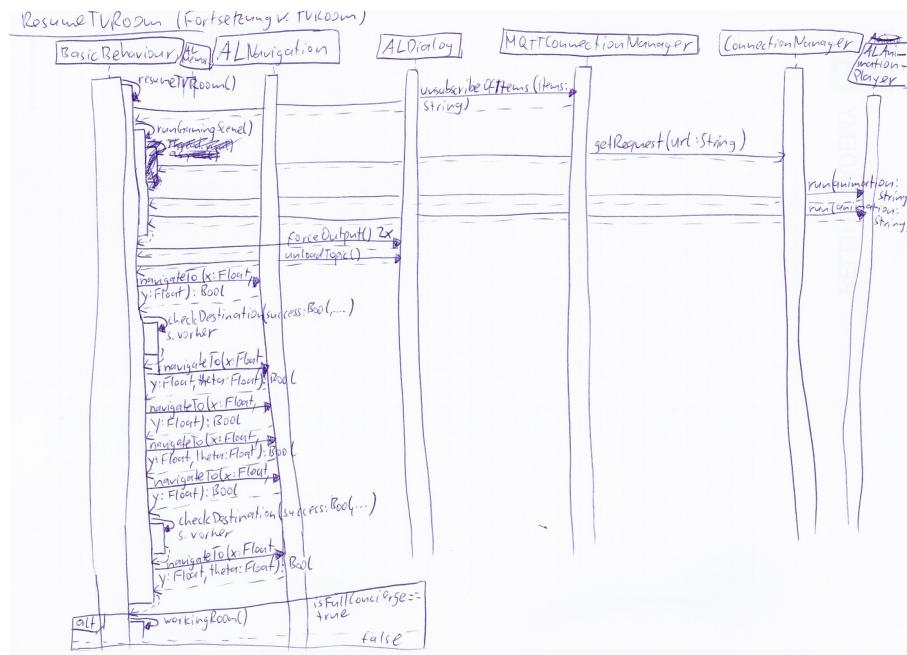


Abbildung 5: Sequenzdiagramm - Führung im TV Raum Teil 2

dezimmer. 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 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` geladen und die erste Sprachausgabe getätigt. Innerhalb des ersten `proposals` wird das Event `GetValue` (siehe Abschnitt 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

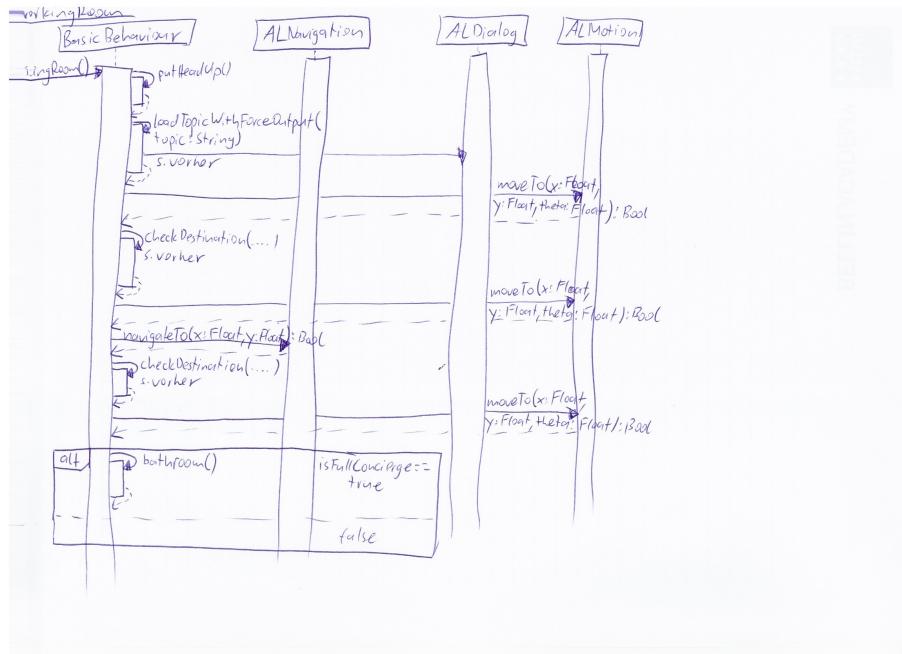


Abbildung 6: Sequenzdiagramm - Führung im Arbeitszimmer

werden, wie in Abbildung 7 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.

3.5 Küche

Der Beginn der Führung in der Küche ist weitgehend identisch mit dem des Badezimmers (siehe Abschnitt 3.4). Von der Positionierung des Kopfes des Roboters, über das Laden der Dialogdatei (`Kitchen.top`), 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 Reihe von `moveTo(x: Float, y: Float, theta: Float): → Bool`-Befehlen (siehe Abbildung 8) 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.

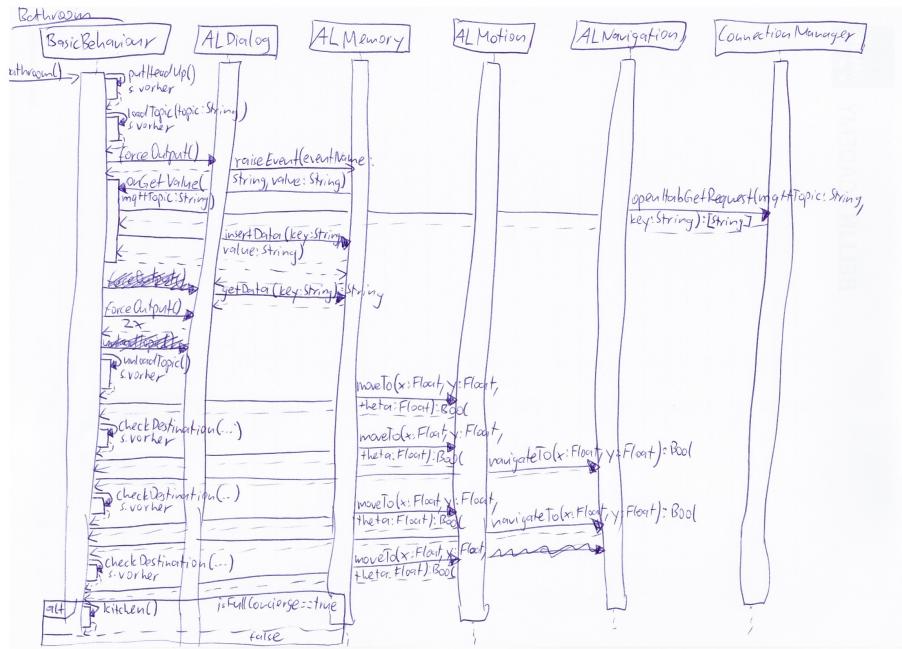


Abbildung 7: Sequenzdiagramm - Führung im Badezimmer

3.6 Abschlusszene

Nach der Positionierung des Kopfes des Roboters, dem Laden der Dialogdatei **Questions.top** 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 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 1.1) entnommen. Im Anschluss setzt der Roboter seine Sprachausgabe durch Laden der Dialogdatei **Goodby.top** fort. Die Abschlussshow wird durch aufrufen der Methode **lightShow()** durchgeführt. Der komplette Befehlsablauf der ist in Abbildung 9 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.

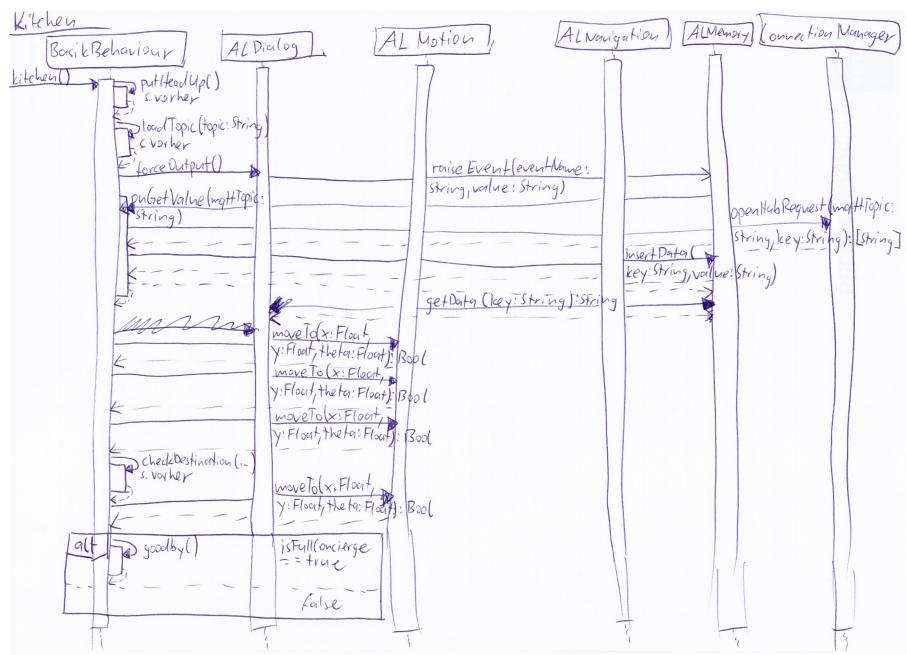


Abbildung 8: Sequenzdiagramm - Führung in der Küche

4 Klassendiagramme

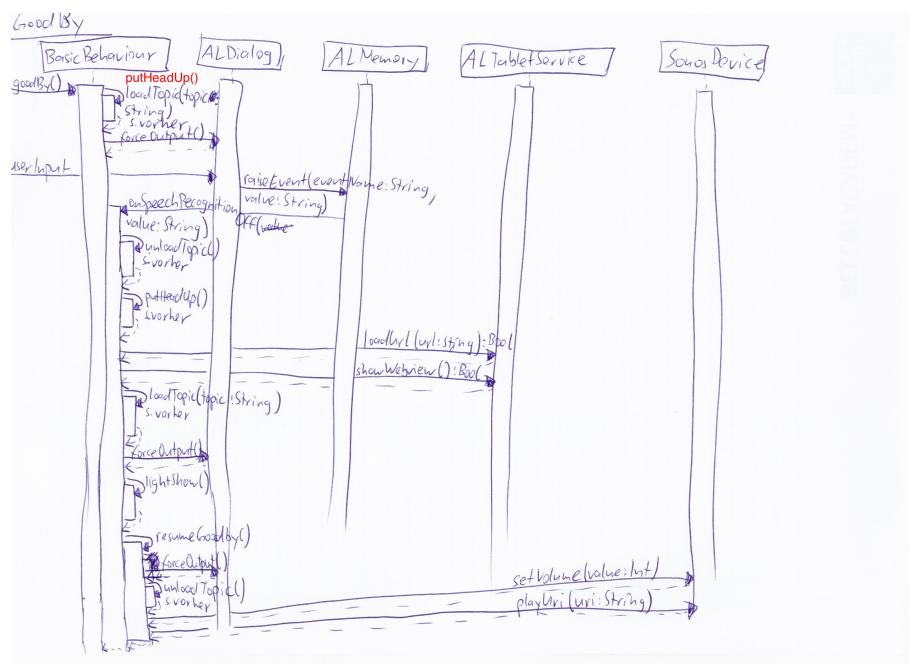


Abbildung 9: Sequenzdiagramm - Abschlusszene