HAW HAMBURG

BACHELOR THESIS
Arthur Haarring

Ein Open-Source basiertes System zur Sprachinteraktion in Smart-Homes

FAKULTÄT TECHNIK UND INFORMATIK
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Arthur Haarring

Ein Open-Source basiertes System zur Sprachinteraktion in Smart-Homes

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Elektro- und Informationstechnik*
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Pawel Buczek
Zweitgutachter: Prof. Dr. Kai von Luck

Eingereicht am: 23. November 2022

Arthur Haarring

Thema der Arbeit

Ein Open-Source basiertes System zur Sprachinteraktion in Smart-Homes

Stichworte

Smart-Home, User Interface, Interpretieren gesprochener Sprache

Kurzzusammenfassung

Thema dieser Bachelorarbeit ist die Konzeption und Umsetzung eines Systems zur Sprachinteraktion in Smart-Homes, basierend auf Open-Source Projekten. Das System erlaubt die Steuerung verschiedener Devices mittels Sprachbefehlen, gibt akustisches Feedback über den Status eines Befehls und implementiert ein Wake Word, um versehentliche Spracheingaben zu vermeiden. Für die Speech-to-Text Umwandlung wird ein vortrainiertes Mozilla DeepSpeech Modell verwendet.

Arthur Haarring

Title of Thesis

An Open-Source based System for Speech Interaction in Smart-Homes

Keywords

Smart-Home, User Interface, Speech Recognition

Abstract

Subject of this thesis is the development of a system for speech interaction in smart-homes, based on open-source projects. The system allows for various devices to be controlled using voice commands, provides acoustic feedback on the status of a command and implements a wake word to avoid accidental inputs. A pre-trained Mozilla DeepSpeech model is used for the speech-to-text conversion.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

"The most profound technologies are those that disappear" schreibt Mark Weiser 1991 in seinem Artikel "The Computer for the 21st Century" [36] und prägt damit den Begriff des Ubiquitous Computing. Er argumentiert weiter, dass die Entwicklung von Computern zu einer unsichtbaren, allgegenwärtigen Technologie ein Umdenken der Mensch-Computer-Interaktion erfordert. Das Interface dieser Technologie sollte dabei nur dann unsere Aufmerksamkeit auf sich ziehen, wenn dies erforderlich ist. "Calm Technology" taufte Mark Weiser diese unaufdringliche Art des User Interface, welches seine User nicht mit Informationen überhäuft, sondern als Teil der Umgebung auf ihre Anwendung wartet [37].

Heutzutage ist Ubiquitous Computing insbesondere in den Bereichen Cyber Physical Systems und Internet-of-Things Normalität. Autos, Häuser, Wohnungen und viele weitere Bereiche des Alltags existieren in Form von Smart-Environments, in denen Computer unsichtbar aber allgegenwärtig den Aufenthalt ihrer User komfortabler gestalten. Natural User Interfaces erlauben dabei dem User, seine Umgebung mit natürlichen Interaktionen, wie z.B. Gesten, Sprache oder Blicken, zu manipulieren [12].

Besonders Sprache als Interaktion zwischen Mensch und Computer ist heutzutage verbreitet, sei es im Smartphone, im Laptop oder im Auto. Auch in Smart-Environments, besonders in Smart-Homes kommen Sprachassistenten zum Einsatz. Die meisten dieser Sprachassistenten arbeiten jedoch nicht lokal, sondern über die Server ihrer Anbieter, wobei der Weg und die Verwendung der Daten durch den User nicht immer nachvollzogen werden kann. Um diesem Umstand entgegenzutreten, wurden bereits zahlreiche Open-Source Projekte ins Leben gerufen, welche den lokalen, serverunabhängigen Einsatz von Sprachassistenten ermöglichen.

1.2 Ziel

Ziel dieser Arbeit ist die Umsetzung eines Systems zur Sprachinteraktion in einem Smart-Home. Das System soll lokal, also serverunabhängig arbeiten und auf Open-Source Bibliotheken aufbauen. Das Smart-Home soll mittels Sprachbefehlen angesteuert werden können. Projektumgebung ist das Living Place Labor an der HAW-Hamburg, welches eine passende Smart-Home Umgebung für Forschungszwecke bietet [1].

1.3 Aufbau der Arbeit

Diese Arbeit befasst sich in Kapitel 2.1 zunächst mit dem wissenschaftlichen Kontext von Smart-Homes sowie der Interaktion zwischen Mensch und Computer in diesen speziellen Umgebungen. Anschließend wird in Kapitel 2.2 mit einem fiktiven Szenario an das Thema der Sprachsteuerung in Smart-Homes herangeführt, bevor in Kapitel 2.3 die konkrete Laborumgebung dieser Arbeit vorgestellt wird. In Kapitel 2.4 und 2.5 folgen einige technische Grundlagen der automatischen Spracherkennung und die Vorstellung ausgewählter Open-Source Lösungen. Die in Kapitel 2 herausgearbeiteten Anforderungen werden in Kapitel 2.6 konkretisiert und erläutert.

Kapitel 3 ist aufgeteilt in Konzeption (3.1) und Umsetzung (3.3) des Systems und enthält in Kapitel 3.2 außerdem die Erläuterungen zur Wahl des Frameworks für die Speech-to-Text Umsetzung. Kapitel 4 befasst sich mit der Evaluation des entwickelten Systems im Hinblick auf die in Kapitel 2.6 definierten Anforderungen.

2 Analyse

Gegenstand des Analysekapitels ist zunächst die Einbettung der Arbeit in ihren übergeordneten fachlichen Kontext. Es folgt ein fiktives Szenario, welches an die Problemstellung der Arbeit heranführt und die Beschreibung des konkreten Umfeldes, in welchem die Arbeit durchgeführt wird. Der nachfolgende Abschnitt erläutert die grundlegenden Komponenten und Methoden, welche für die automatische Spracherkennung benötigt werden. Den Abschluss bilden die im Laufe der Analyse herausgearbeiteten Anforderungen an das Zielsystem definiert.

2.1 Cyber Physical Systems

Smart-Homes zählen zu einer speziellen Gruppe von Systemen, den Smart Environments. Allgemeiner betrachtet gehören Smart Environments zu den sogenannten Cyber Physical Systems (CPS), die sich dadurch auszeichnen, dass sie sich sowohl aus Softwarekomponenten als auch aus physikalischen, also sensorischen und aktorischen Komponenten zusammensetzen [15]. Durch das komplexe Zusammenspiel von eingebetteten Systemen, Anwendungssystemen und Infrastrukturen ermöglichen CPS die Verbindung von physikalischer und informationstechnischer Welt [13]. Dabei sind diese Systeme keineswegs abgeschlossene Einheiten, sondern bilden durch die Nutzerinteraktion offene, hochgradig vernetzte, soziotechnische Systeme [13]. Als solche bieten sie neue Funktionalitäten und Verbesserungen der Lebensqualität in vielen kritischen Bereichen, zum Beispiel bei Fahrzeugsteuerung, Verkehrsmanagement und Kommunikationsnetzen [13], aber auch in der Krankenpflege, Produktion sowie Energieversorgung und -verbrauch [15].

Ein wichtiger Teil von CPS ist die Interaktion des Systems mit seinen Usern. Forderungen an derart vernetzte Systeme beinhalten unter anderem eine "[...] transparente, für Nutzer verständliche und beherrschbare Mensch-Maschine-Interaktion, um Fehlbedienungen und fehlerhaften Einsatz der Systeme und damit eine Gefährdung der Umgebung und

der Beteiligten zu vermeiden [...]" (Geisberger & Broy, 2012, S.74f) . In wie weit diese und weitere Forderungen umzusetzen sind, hängt dabei wesentlich von Einsatzgebiet und Aufgabe des Systems ab. Sicherheitskritische Systeme wie Smart Grids müssen eine erweiterte Betriebssicherheit aufweisen [13] und besitzen dabei fundamental andere Rollenverteilungen und Fähigkeiten als Systeme des Ambient Assisted Living (AAL)[13].

2.1.1 Smart Environments

Ein Aspekt von Cyber Physical Systems sind Smart Environments [13]. Smart Environments sind Netzwerke von Geräten mit der Aufgabe, das Leben bzw. den Aufenthalt der Bewohner bzw. Nutzer eines Umfeldes komfortabler zu gestalten [12]. Cook & Das definieren Smart Environments wie folgt:

"We [..] define a smart environment as one that is able to acquire and apply knowledge about an environment and also to adapt to its inhabitants in order to improve their experience in that environment." (Cook & Das, 2005, S.3)

Eine zentrale Rolle spielt hier die Verbindung der Systemkomponenten zur physikalischen Welt. Dabei übernehmen die Komponenten in der Regel einzelne vordefinierte Aufgaben, wie z.B. Bewegungssensoren, die mit einem automatischen Türöffner gekoppelt sind [31]. Eine grundlegende Charakteristik von Smart Environments ist die Möglichkeit der Fernsteuerung seiner Komponenten, genannt Devices, wobei diese sowohl automatisch vom System, als auch durch Interaktion mit dem User erfolgen kann [12]. Damit ist dieser befreit von der Notwendigkeit der direkten physischen Interaktion mit dem zu steuernden Device. Er muss z.B. nicht mehr aufstehen und den Lichtschalter betätigen, sondern kann von der Couch aus mittels Fernbedienung das Licht an- oder ausschalten [12].

2.1.2 HCI in Smart Environments

Die Möglichkeiten der Human-Computer-Interaction (HCI) in Smart Environments stellt Mark Weiser mit dem Konzept des allgegenwärtigen, unsichtbaren Computers bereits 1991 in seinem Artikel "The computer for the 21st century" [36] vor. Der Aspekt der Unsichtbarkeit steht dabei im Konflikt mit der herkömmlichen Computerinteraktion über Ein- und Ausgabegeräte wie Tastatur, Maus und Bildschirm. In der Folge werden neue Arten der Interaktion zwischen Mensch und Computer entwickelt, um dem User die Kontrolle des Systems und den Abruf von Informationen zu ermöglichen [36]. Ishii und

Baudisch haben durch Experimente und Studien viele neue Möglichkeiten der Interaktion untersucht, unter anderem Interaktion durch Gesten, Fernbedienungen sowie Augmented und Virtual Reality [19, 10, 16]. In dieser Arbeit wird der Ansatz der Sprache als Interaktion zwischen Mensch und Computer im Umfeld von Smart Environments, genauer Smart-Homes näher untersucht.

HCI setzt sich aus drei grundlegenden Konzepten zusammen [30]:

- Menschen: einzelne oder mehrere User mit variierenden physischen bzw. mentalen Fähigkeiten, welche kooperativ oder kompetitiv interagieren.
- Computer: nicht nur PCs, sondern auch eingebettete Systeme aller Art.
- Interaktion: geschieht über Befehle oder Manipulation von virtuellen Objekten, kann aber auch natürliche Interaktionen wie Sprache, Gesten etc. beinhalten.

Um die Interaktion sinnvoll zu gestalten, sollte das System möglichst gut auf die Anforderungen und Aspekte der Interaktion und des Users abgestimmt werden.

HCI kann sowohl implizit als auch explizit erfolgen. Explizite HCI erfordert einen direkten Befehl oder Eingabe vom User, woraufhin der Computer die damit verbundene Interaktion ausführt [30]. Beispielsweise sagt Sal im Szenario (siehe Kap. 2.2) "Vorhänge auf [...]" und gibt dem Computer damit einen expliziten Befehl, welchen dieser ausführt. Implizite HCI (iHCI) benötigt keine direkte Eingabe des Users. Stattdessen interpretiert der Computer über sensorische Komponenten das Verhalten des Users und führt als Konsequenz bestimmter Verhaltensmuster definierte Aktionen aus [30]. Die automatische Öffnung von Türen, ausgelöst durch Bewegungsmelder ist ein Beispiel für iHCI [31].
Wie bereits erwähnt, ist die Steuerung mittels Sprachbefehlen der expliziten HCI zuzuordnen [30].

Bei der Kommunikation mit dem Smart-Home nimmt der User eine gewisse Erwartungshaltung ein, welche sich durch bestimmte Aspekte auszeichnet.

- Dem User ist bewusst, dass er mit einem technischen Artefakt interagiert.
- Der User ist mit dem System vertraut, es gibt keine "casual User".
- Der User sieht die Wohnung als Befehlsempfänger, nicht als Gesprächspartner.

Das unter diesen Aspekten zu entwerfende System muss demnach keine natürliche Sprache verstehen. Es muss lediglich auf bestimmte Befehle oder Schlüsselwörter reagieren, welche gegenüber dem User als bekannt vorausgesetzt werden.

Wake Word

Die fehlende Fähigkeit eines solchen Systems, die Eingaben semantisch zu interpretieren, führt zu einem Problem: es reagiert immer, auch wenn dies vom User gar nicht beabsichtigt ist. Dieses Problem lässt sich am einfachsten durch ein Wake Word lösen. Dabei handelt es sich um ein bestimmtes Wort, welches die Befehlsaufnahme des Systems aktiviert. Dieses kann ebenfalls als dem User bekannt vorausgesetzt werden und sollte dabei so gewählt werden, dass es nicht unbeabsichtigt ausgesprochen wird. Am bekanntesten dürften die Wake Words von Apple, "Hey Siri" und Amazon "Alexa" sein.

Round-Trip-Time

Als Round-Trip-Time (RTT) wird im Bereich HCI die Zeit zwischen der Aktion bzw. Befehlseingabe des Users und der für den User wahrnehmbaren Reaktion des Systems bezeichnet, beispielsweise die Zeit von der Eingabe des Befehls zum Öffnen eines Fensters bis zum tatsächlichen Öffnen des Fensters. Bei der Entwicklung von HCI-Systemen wird Wert darauf gelegt, die RTT gering zu halten, da eine lange Verzögerung die User-Experience negativ beeinflusst. Kohrs et. al definiert diese Grenze bei 200 ms [22].

2.2 Szenario

Als Einführung in Thematik und Problemstellung dieser Arbeit dient ein fiktives Szenario, angelehnt an Mark Weisers Vision in dem Artikel "The computer for the 21st century" [36]. Dabei soll ein Eindruck der zentralen Eigenschaften eines Systems zur Sprachinteraktion in Smart-Homes vermittelt werden. Hauptcharakter ist "Sal", welche in einem Smart-Home lebt, das sich mittels Sprachbefehlen steuern lässt.

Sal erwacht. Durch ihre Vorhänge scheinen bereits die ersten Sonnenstrahlen. Der Wecker zeigt 8 Uhr. "Mark," sagt sie, während sie die Beine aus dem Bett schwingt, "Vorhänge auf und lüften.". Sogleich schwingen die Vorhänge beiseite und mit einem leisen

Surren öffnen sich die Fenster. Sonnenlicht und die vertrauten Geräusche der morgendlichen Nachbarschaft erfüllen das Schlafzimmer. "Mark" ist der Name ihrer Wohnungs-KI, welche über Mikrofontechnologien auf die Erwähnung dieses Schlüsselwortes wartet. Sal geht in die Küche, wo sich auf ihren Befehl hin die Kaffeemaschine in Gang setzt. Während sie auf den Kaffee wartet, geht sie ins Badezimmer, welches über Nacht merklich abgekühlt ist. "Mark, stell die Heizung im Badezimmer auf 24 Grad." befiehlt sie ihrer Wohnung auf dem Weg zurück in die Küche, wo ihr die Kaffeemaschine piepsend die Fertigstellung ihrer allmorgendlichen Tätigkeit signalisiert. "Sal, die neue Zieltemperatur im Badezimmer wird in etwa 10 Minuten erreicht.", antwortet Mark.
Nach dem Frühstück geht sie im aufgewärmten Bad duschen und macht sich für die Arbeit fertig. Mit dem Befehl "Mark, ich gehe jetzt." werden die Lichter gelöscht, die Fenster geschlossen und die Heizung im Badezimmer geht zurück in den Energiesparmodus.

Nach einem langen Tag im Büro kommt Sal wieder nach Hause und setzt sich auf die Couch. "Mark, ich will die Tagesschau sehen.", sagt sie, woraufhin Mark den Fernseher anschaltet und den richtigen Sender einstellt. "Sal, die Tagesschau beginnt in 15 Minuten.". In der Zwischenzeit geht Sal in die Küche und setzt Nudeln auf. "Mark, stelle einen Timer auf 10 Minuten" - "Ein Timer wurde auf 10 Minuten gestellt und gestartet." antwortet Mark.
Als Sal später ins Bett geht, sagt sie "Mark, gute Nacht", woraufhin sich die Fenster schließen, die Heizung in den Nachtmodus schalten und die Vorhänge vorgezogen werden. Bevor sie einschläft hört Sal noch einmal Marks Stimme: "Gute Nacht Sal, ich werde dich morgen um 8 Uhr wecken."

Ein Szenario wie dieses ist schon lange keine Science Fiction mehr. Viele große Firmen der Tech-Industrie haben bereits vergleichbare Systeme auf den Markt gebracht, welche sich über Sprachassistenten von Google, Apple oder Amazon steuern lassen. Der Fokus dieser Arbeit liegt auf der serverunabhängigen, lokalen Umsetzung eines solchen Systems mittels Open-Source Bibliotheken und dessen Integration in eine Smart-Home Infrastruktur.

2.3 Laborumgebung "Living Place"

Das im Jahr 2009 gegründete Labor "Living Place" ist Teil des Forschungs- und Transferzentrums "Smart Systems" der Fakultät Technik und Informatik an der HAW Hamburg.

Es bietet einen realitätsnahen Kontext zur Untersuchung von Ubiquitous-Computing-Systemen und beinhaltet ein Smart-Home für die Durchführung von Experimenten unter realitätsnahen Bedingungen. Die Laborfläche beträgt 140 m^2 und ist in verschiedene Bereiche unterteilt, wie im Grundriss und auf den Fotos zu sehen (siehe Abb. 2.1 & 2.2). Das Smart-Home umfasst die Bereiche Wohnzimmer, Esszimmer, Schlafzimmer, Bad, Flur und Küche [1].

Das Wohnzimmer ist mit Sitzecke, Couchtisch und einem Fernseher eingerichtet. Die Küche umfasst eine freistehende Arbeitsplatte mit integriertem Ceranfeld und Backofen. Eine weitere Arbeitsplatte mit Spüle ist vor dem Fenster angebracht. Im Esszimmer steht ein Esstisch mit Stühlen, ein Barelement dient als Abgrenzung zur Küche. Im Schlafzimmer steht ein Doppelbett mit Nachttisch, das Badezimmer ist mit Smart-Mirror, einer Dusche und einer Toilette ausgestattet. Der Flur bildet den Bereich zwischen Küche, Schlaf- und Esszimmer.

Abbildung 2.1: Grundriss Living Place [1]

(a) Wohnzimmer

(b) Küche

(c) Schlafzimmer

(d) Esszimmer

(e) Badezimmer

Abbildung 2.2: Fotos vom Living Place (Foto: Manuel Grandt, 2022)

Durch Interaktion mit der Wohnung stehen dem User diverse Möglichkeiten zur Verfügung, die Umgebung nach seinen Wünschen anzupassen:

- Das Licht ist sowohl zentral für die ganze Wohnung als auch in den einzelnen Bereichen steuerbar. Die Steuerung für die einzelnen Bereiche erlaubt außerdem auch die Anpassung des Farbspektrums.
- Die Fenster jedes Bereiches können ferngesteuert gekippt werden.

- Alle Fenster sind mit Rollos ausgestattet, welche pro Bereich gemeinsam gesteuert werden können.
- Mit Ausnahme von Flur und Küche sind in allen Bereichen Heizkörper verbaut. Diese erlauben für jeden Bereich die Einstellung einer Zieltemperatur.
- Flur, Wohn- und Schlafzimmer sind mit Gardinen ausgestattet, welche sich ferngesteuert öffnen und schließen lassen.
- Die Stromzufuhr der Kaffeemaschine in der Küche lässt sich ferngesteuert ein- und ausschalten.

Smart-Home Infrastruktur

Grundlage der lokalen Smart-Home Infrastruktur ist das Message Queuing Telemetry Transport Protokoll (MQTT) [9], welches die Kommunikation zwischen den verschiedenen vernetzten Geräten und Smart Devices regelt. Die Steuerung der Systemkomponenten erfolgt durch gezielte Selektion und Manipulation. Dabei sind jedem Aktor bzw. Sensor des Systems ein oder mehrere Topics zugeteilt. Ein Aktor kann mittels JSON Befehlen an sein Topic gesteuert werden, Sensoren können ihren Status über eine JSON Payload an ihr Topic mitteilen. Die meisten steuerbaren Komponenten geben Rückmeldung über ihren aktuellen Status an die ihnen zugeteilten Topics zurück. Das Küchenfenster lässt sich beispielsweise durch eine JSON-Payload mit dem String "OPEN" und das Topic "LP/Kueche/Fenster/Set" öffnen und gibt seine aktuelle Position über das Topic "LP/Kueche/Fenster/NodeStatus" zurück.

2.4 Automatische Spracherkennung (ASR)

Wichtigster Bestandteil des zu entwerfenden Systems ist die automatische Spracherkennung (Automatic Speech Recognition, ASR). Diese hat die Aufgabe, aus Audiodaten Buchstaben und Wörter zu interpretieren. Diese Technologie ist mittlerweile im Alltag vieler Menschen integriert und erleichtert in Form von Sprachdialogsystemen in der Telekommunikation [21] oder durch Sprachassistenten wie Siri (Apple), Cortana (Microsoft) und Google Assistant die Kommunikation zwischen Mensch und Computer. Dieses Kapitel umfasst eine Einführung in die gängigsten technischen Herangehensweisen an das Problem der automatischen Umwandlung von Sprache zu Text: Pipeline ASR und End-to-End ASR.

2.4.1 Pipeline ASR

Unter dem Begriff "Pipeline ASR" werden die traditionellen algorithmischen und statistischen Herangehensweisen zusammengefasst, wobei stets mehrere Module mit speziellen Aufgaben hintereinander geschaltet werden, um aus den Audiodaten Schritt-für-Schritt lesbare Wörter zu Dekodieren [14]. Diese Systeme lassen sich in drei voneinander unabhängige Subsysteme aufteilen [35] (siehe Abb. 2.3):

- Acoustic Model (Feature Extraction):
  Aus den eingehenden Audiodaten werden bestimmte Sequenzen von Features extrahiert. Dies sind meist Phoneme (bzw. Subphoneme), welche die kleinsten bedeutungsunterscheidenden sprachlichen Einheiten bilden.
- Pronounciation Model (Acoustic Decoding):
  Die extrahierten Phoneme werden auf die jeweiligen Grapheme abgebildet, welche die kleinsten bedeutungsunterscheidenden Einheiten in einem Schriftsystem bilden.
- Language Model (Language and Phonetic Decoding):
  Aus den Graphemen werden mit Hilfe eines Sprachmodells lesbare Wörter und Sätze gebildet.

Jedes dieser Subsysteme bedient sich spezieller Algorithmen, von denen die beiden wichtigsten im Folgenden aufgeführt werden.

Hidden Markov Model (HMM)

Beim Hidden Markov Model handelt es sich um ein stochastisches Modell, welches erstmals Mitte der 1980er Jahre im Bereich ASR eingesetzt und bis in die frühen 2000er Jahre intensiv erforscht und weiterentwickelt wurde [35]. Vor dem Aufkommen von End-to-End ASR mittels Neural Networks und Deep Learning dominierten die auf HMM-Technologie basierenden Pipeline ASR Modelle den Markt [35]. Meist kommt HMM im Acoustic Model zum Einsatz, wird aber teilweise auch in den anderen beiden Subsystemen genutzt.

Dynamic Time Warping

Eine weitere erwähnenswerte Technik ist das Dynamic Time Warping (DTW), welches im Acoustic Model angewandt wird. DTW ist ein Algorithmus, der die Ähnlichkeit von Zeitreihendaten bestimmt [26], indem er den optimalen Pfad zwischen zwei Samples berechnet. Je kleiner der Pfad, desto ähnlicher sind die Daten. Ein Vorteil dieses Algorithmus ist, dass die beiden Samples nicht gleich lang sein müssen, um deren Ähnlichkeit zu bestimmen. So wird das Problem unterschiedlicher Sprechgeschwindigkeiten gelöst [29].

2.4.2 End-to-End ASR

Moderne ASR Systeme basieren meist auf Deep Learning Technologie und wandeln Audiodaten direkt in lesbaren Text um [38, 14]. Die akustischen, phonetischen und linguistischen Komponenten der traditionellen Herangehensweise sind in einem einzigen Neuronalen Netzwerk zusammengefasst [20, 14]. Dieses bekommt meist Audio-Spektrogramme als Input und generiert daraus eine Sequenz von Buchstabenwahrscheinlichkeiten [17]. Hannun et. al beschreibt diesen Vorgang wie folgt [17]:

Ein Set von Trainingsdaten X, bestehend aus einzelnen Sprachsamples x und den dazugehörigen Wörtern in Textform, genannt Labels y, sei wie folgt definiert:

$$X = \{(x^{(1)}; y^{(1)}), (x^{(2)}, y^{(2)}), ...\}$$

Jedes Sample $x^{(i)}$ der Länge $T^{(i)}$ besteht aus einer Reihe von Vektoren mit Audiofeatures $x_t^{(i)}, t = 1, ..., T^{(i)}$. Diese Features sind in der Regel Spektrogramme, sodass der einzelne Input in das System letztendlich $x_{t,p}^{(i)}$ ist, d.h. die Leistung der p-ten Frequenz im einzelnen Audioframe zur Zeit t. Aus diesem Input wird die Sequenz von Buchstabenwahrscheinlichkeiten y für das Sample x generiert, mit $\hat{y}_t = \mathbb{P}(c_t|x)$ und $c_t \in \{a, b, c, ..., z, space, apostrophe, blank\}$. c_t wird als Alphabet bezeichnet und bildet die Menge der Buchstaben und Zeichen.

Neben dem aktuellen Audiosample werden eine bestimmte Anzahl Frames rechts und links des aktuellen Frames zusätzlich als Input in das System gegeben, um einen Kontext zum aktuellen Frame zur Verfügung zu stellen.

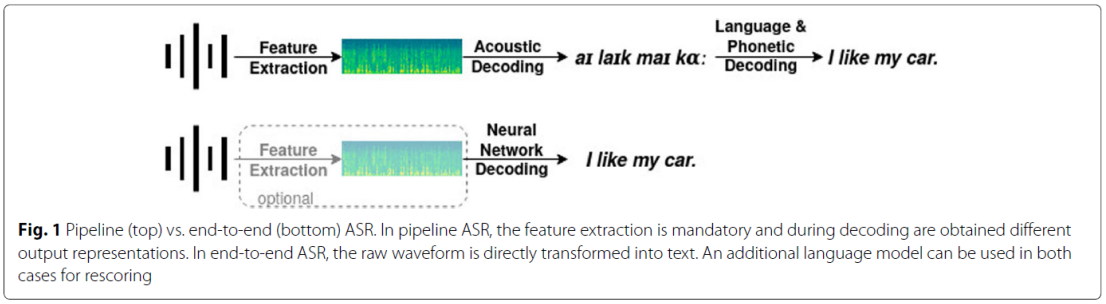


Fig. 1 Pipeline (top) vs. end-to-end (bottom) ASR. In pipeline ASR, the feature extraction is mandatory and during decoding are obtained different output representations. In end-to-end ASR, the raw waveform is directly transformed into text. An additional language model can be used in both cases for rescoring

Abbildung 2.3: Konzepte von Pipeline und End-to-End ASR [14]

Training von Sprachmodellen (Speech-to-Text Models)

Das Training von Neuronalen Netzen ist sehr rechenintensiv und benötigt tendenziell große Datenmengen. Neuere Sprachmodelle, wie z.B. NVIDIA QuartzNet, werden mit über zehntausend Stunden Sprachaufnahmen trainiert, ihre Genauigkeit wird mit Testdatensätzen von einigen hundert Stunden geprüft [17]. Dabei hängt die Qualität des erstellten Sprachmodelle ebenso maßgeblich von der Qualität der zur Verfügung gestellten Daten ab, wie von der Menge. Um höhere Genauigkeit bei lauten Umgebungsgeräuschen zu erzielen, werden Sprachaufnahmen sogar absichtlich verzerrt oder mit Rauschen hinterlegt [17]. Open-Source Projekte wie Mozilla Common Voice [8] oder Libri Speech [28] haben es sich zum Ziel gemacht, diese Daten zum Training von Sprachmodellen zu generieren und zur Verfügung zu stellen. Die Daten werden dabei von freiwilligen Sprecher:innen generiert.

Transfer Learning

Eine in diesem Kontext oft angewandte Methode ist das Transfer Learning. Oft ist die Menge an Trainingsdaten unzureichend, um ein effizientes Sprachmodell aufzubauen. Diese Datenknappheit wird umgangen, indem ein bereits existierendes, mit anderen Daten vortrainiertes Modell mit den neuen Trainingsdaten trainiert wird[25]. Diese Methode ist in konventionellen ASR Systemen weit verbreitet, da sie es unter anderem möglich macht, ein bereits existierendes Sprachmodell für eine neue Sprache mit weniger Daten umzutrainieren [20].

2.4.3 Leistung von Sprachmodellen

Hauptfaktor für die Quantifizierung der Leistung von Sprachmodellen ist die Word-Error-Rate (WER), also der Prozentsatz an falsch erkannten Wörtern. Die WER wird in Prozent angegeben und berechnet sich wie folgt:

$$WER = \frac{Anzahl\ der\ falsch\ transkribierten\ Wörter}{Gesamtanzahl\ der\ Wörter} \cdot 100\%$$

Ein weiterer Faktor ist der Ressourcenverbrauch der Modelle. Ausschlaggebend ist der Speicherbedarf des trainierten Modells und der Bedarf an Rechenleistung.

2.4.4 Voice Acitivity Detection (VAD)

Voice Activity Detection befasst sich mit der Detektion von Sprache in einem Audiosignal [33], [34] und ist ein wichtiger Teil vieler Anwendungen der Sprachverarbeitung [23]. Im Gebiet der Spracherkennung wird VAD hauptsächlich dazu genutzt, weitere Erkennungsprozesse auszulösen, wenn es die Anwesenheit von Sprache erkennt [34]. Dabei sollte das VAD-Modul möglichst schnell und effizient arbeiten, um die Rechenzeit des Speech-to-Text Moduls zu reduzieren [34].

2.5 ASR Frameworks

ASR Frameworks bieten Lösungen an, um eigene Speech-to-Text Modelle zu entwickeln, zu trainieren und anzuwenden. Dabei kommen je nach Framework verschiedene Algorithmen zum Einsatz, welche die Qualität des Modells beeinflussen. In diesem Kapitel wird eine Auswahl der gängigsten Open-Source Frameworks vorgestellt.

2.5.1 Kaldi

Kaldi ist ein 2011 entwickeltes Open-Source Toolkit für Forschung im Bereich Spracherkennung [32]. Es basiert auf dem Pipeline Modell (siehe Kap. 2.4.1), allerdings werden die Module teilweise auch mit Hilfe von Neuronalen Netzen umgesetzt. Kaldi stellt verschiedene Modelle für unterschiedliche Zwecke zur Verfügung. Diese sind in C++ geschrieben, mit dem Ziel den Code flexibel und einfach erweiterbar zu gestalten [32].

2.5.2 Mozilla DeepSpeech

Mozilla DeepSpeech ist eines der ersten Open-Source Projekte im Bereich der End-to-End Spracherkennung. Grundlage des Projektes ist das 2014 von Baidu Research veröffentlichte Paper 'DeepSpeech: Scaling up end-to-end speech recognition" [17]. Das System implementiert ein Recurrent Neural Network (RNN), welches durch ein optimiertes Trainingssystem auf mehreren GPUs trainiert werden kann. Zum Zeitpunkt der Veröffentlichung erreicht das System eine WER von 16 % und funktionierte in Umgebungen mit vielen Nebengeräuschen besser als kommerziell verwendete Systeme.

Das Open-Source Projekt wurde im Jahr 2017 von Mozilla ins Leben gerufen und bis 2021 weiterentwickelt. Zuletzt wurden WERs unter 10 % erreicht, außerdem wurden die Ressourcenanforderungen so weit verringert, dass DeepSpeech auch auf leistungsschwachen Systemen ohne Leistungseinbußen funktioniert. Aufgrund der langjährigen Entwicklung existiert eine große Fülle von vortrainierten Sprachmodellen und Support für das Projekt.

External Scorer

Eine Funktion von DeepSpeech ist die Möglichkeit zur Aktivierung eines zusätzlichen Sprachmodells, genannt "Scorer". Dieser kann auf die in der Anwendung benötigten Vokabeln trainiert werden und so durch Einschränkung des Wortschatzes die Fehlerrate senken. Grundlage des Scorers ist das KenLM Language Model Toolkit [18].

2.5.3 NVIDIA NeMo

NVIDIA NeMo ist ein Open-Source Toolkit für die Entwicklung von dialogorientierten KI Modellen. Es umfasst Kollektionen für ASR, Natural Language Processing (NLP) und Text-to-Speech (TTS) [2]. Die ASR Kollektion unterstützt eine Vielzahl von Sprachmodellen, welche je nach Art der Anwendung auf der Basis verschiedenartiger Neural Networks trainiert wurden. Besonders hervorzuheben ist die QuartzNet-Architektur [24], welche derzeit eine der niedrigste WERs aufweist (vgl. Abb. 2.4).

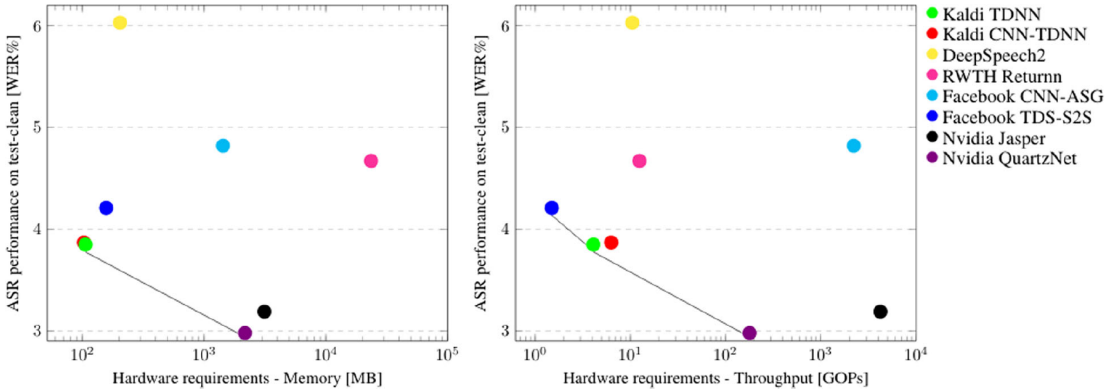


Abbildung 2.4: Vergleich einiger Open-Source ASR Modelle[14]

2.6 Anforderungen

Nachfolgend werden die in der Analyse herausgearbeiteten Anforderungen an das Zielsystem konkretisiert. Dabei soll das Zielsystem für die Verwendung in einem Einpersonenhaushalt ausgelegt werden. Es wird demnach davon ausgegangen, dass immer nur eine Person spricht und die Menge an Störgeräuschen gering ist.

2.6.1 Funktionale Anforderungen

Die funktionalen Anforderungen beschreiben die umzusetzenden Systemfunktionalitäten, sowie wichtige Eigenschaften, welche bei der Umsetzung zu beachten bzw. zu optimieren sind.

Steuerung der Devices

Das Zielsystem soll die verschiedenen Devices der in Kapitel 2.3 beschriebenen Smart-Home Umgebung ansteuern können. Dabei wird zwischen Devices mit direktem Feedback und Devices mit indirektem Feedback unterschieden, außerdem wird die Art der Steuerung angegeben, welche binär oder mehrwertig sein kann.

Devices mit direktem Feedback:

- Fenster, binäre Steuerung (auf, zu)
- zentrales Licht, binäre Steuerung (an, aus)
- Kaffeemaschine, binäre Steuerung (an, aus)
- Vorhänge, binäre Steuerung (auf, zu)
- Rollos, binäre Steuerung (hoch, runter)

Devices mit indirektem Feedback:

- Heizungen, mehrwertige Steuerung (an, aus, Temperatur)

Gibt es mehrere Devices desselben Typs (z.B. Fenster), sollen diese sowohl pro Raum as auch alle auf einmal steuerbar sein.

Sprecherunabhängigkeit

Das System sollte für unterschiedlichen Personen nutzbar sein, ohne dass Änderungen an der Umsetzung vorgenommen werden müssen.

User Feedback

Akustisches Feedback über den Status eines Befehls soll insbesondere dann erfolgen, wenn der Befehl nicht zu einer durch den User direkt erkennbaren Aktion führt. Darunter fallen Devices mit indirektem Feedback, beispielsweise die Heizung. Auch nicht erkannte oder ungültigen Befehle sollen dem User über akustisches Feedback kommuniziert werden.

Fehlerrate

Das Vermeiden von Fehlern und durch den User ungewolltes Verhalten soll bei der Umsetzung besonders berücksichtigt behandelt werden, da dies die User-Experience negativ beeinflusst. Dabei ist zwischen drei Szenarien zu unterscheiden.

Im ersten Fall hat das System ein oder mehrere Wörter nicht bzw. falsch verstanden, woraus ein ungültiger Befehl resultiert. Aus User-Sicht passiert in diesem Fall nichts, der gewünschte Befehl wird nicht ausgeführt. Um diese Art Fehler zu verhindern, soll bei der Umsetzung darauf geachtet werden, die WER (siehe Kap. 2.4.3) des Systems möglichst gering zu halten.

Im zweiten Fall führt das System aufgrund eines oder mehrerer falsch erkannter Wörter einen anderen Befehl aus, als vom User beabsichtigt. Ein solches Verhalten kann gravierendere Konsequenzen haben als im ersten Fall, da der User mit einer nicht beabsichtigten Handlung konfrontiert wird. Das Eintreten dieses Falles kann ebenfalls durch eine geringe WER vermieden werden. Zusätzlich soll darauf geachtet werden, dass die phonetischen Unterschiede zwischen den möglichen Spracheingaben ausreichend groß sind.

Im dritten Fall erkennt das System eine Eingabe, obwohl der User diese nicht beabsichtigt hat, beispielsweise wenn dieser gerade telefoniert. Die Wahrscheinlichkeit des Eintretens dieses Falles soll durch Umsetzung eines Wake Word verringert werden, welches die Spracheingabe des Systems aktiviert.

Latenz

Bei der Umsetzung ist darauf zu achten, die Latenz gering zu halten, um möglichst wenig Zeit zur Round-Trip-Time beizutragen.

2.6.2 Nicht-funktionale Anforderungen

Portierbarkeit

Um die Übertragbarkeit auf andere Umgebungen und Systeme zu ermöglichen, sollte das Zielsystem möglichst unabhängig von den verfügbaren Ressourcen arbeiten.

Erweiterbarkeit

Das Zielsystem sollte um weitere Räume, Devices und Befehle erweitert werden können.

3 Design

Das Designkapitel befasst sich zunächst mit der Konzipierung des Systems und seiner Komponenten, unter Berücksichtigung der Anforderungen und Erkenntnisse des Analysekapitels. Anschließend wird der Auswahlprozess eines passenden ASR Frameworks und die Umsetzung des Konzeptes erläutert.

3.1 Konzept

Um das System zu konzipieren, werden zunächst die einzelnen Systemkomponenten definiert und ihre jeweiligen Aufgaben beschrieben. Darauf aufbauend wird ein Systemablaufplan entworfen, welcher als Vorlage für die Programmstruktur dient und die Funkionen der einzelnen Komponenten vereint.

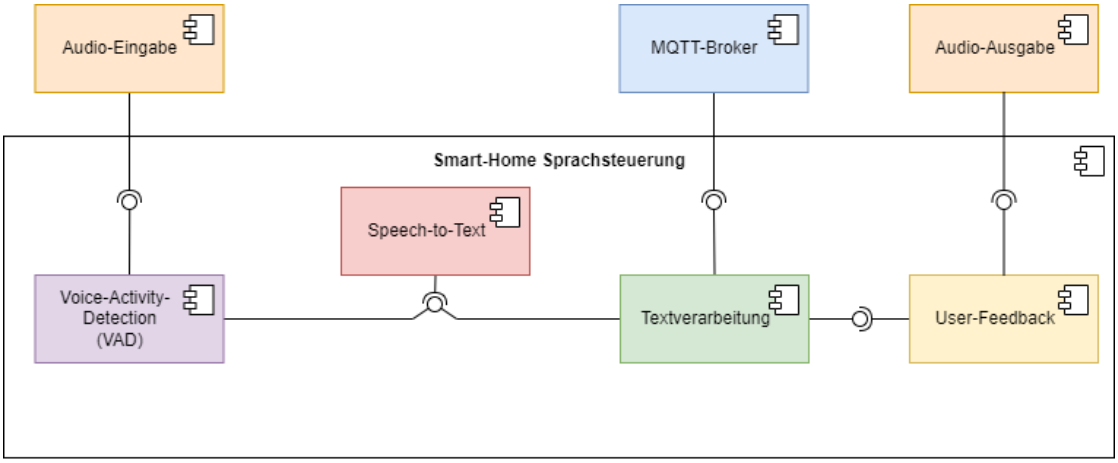


Abbildung 3.1: Komponentendiagramm

Das Komponentendiagramm in Abbildung 3.1 stellt die System-Komponenten mit ihren jeweiligen Interfaces dar, sowie die Schnittstellen des Systems nach außen.

3.1.1 Schnittstellen

Input - Audio-Eingabe

Um Daten aufzunehmen, wird ein Interface zum Standard-Audioeingabegerät des lokalen Systems benötigt, welches es möglich macht, in Echtzeit eingehende Audiodaten auszulesen und zu speichern.

Output - MQTT, Audio-Ausgabe

Um die erkannten Befehle weiterzuleiten, ist ein Interface zu einem MQTT Modul notwendig, welches wiederum eine Verbindung zum MQTT Broker der Smart-Home Infrastruktur herstellt und die Befehle an diesen sendet.

Um dem User akustische Rückmeldungen zu geben, ist außerdem ein Interface zum lokalen Standard-Audioausgabegerät erforderlich.

3.1.2 VAD Modul

Das VAD-Modul greift auf das Standard-Audioeingabegerät des lokalen Systems zu und nimmt eingehende Audiodaten in Chunks auf, nachfolgend als Audioframes bezeichnet. Die Audioframes werden auf die Anwesenheit von Sprache hin untersucht. Enthält ein Frame Sprache, wird dieses zum Audiobuffer des Speech-to-Text Moduls hinzugefügt und das nächste Frame wird aufgenommen. Enthält ein Frame keine Sprache, so wird dem Speech-to-Text Modul der Befehl zum Auslesen und Übersetzen des Audiobuffers gegeben.

3.1.3 Speech-to-Text Modul

Das Speech-to-Text Modul bekommt Audioframes als Input und wandelt diese in eine Zeichenkette um. Diese wird zwischengespeichert und der Audiobuffer wird nach der Umwandlung gelöscht.

3.1.4 Textverarbeitung

Die Befehle lassen sich aufteilen in ein Befehlsziel und eine Art der Manipulation. Beispielsweise ist das Ziel des Befehls "Licht an" das Device "Licht" und die Manipulation ist "an". Jede weitere Kategorisierung eines Befehls hängt dabei vom Aufbau des Smart Environment und der Art der Device-Steuerung ab.

Im Modul der Textverarbeitung wird ein Befehl zunächst in einzelne Stichwörter zerlegt, welche anschließend in Kategorien sortiert werden. Dadurch ergibt sich für jeden Befehl eine einzigartige Kombination von kategorisierten Stichwörtern, was es schließlich möglich macht, bestimmte Wortkombinationen auf bestimmte Befehle an das Smart Environment abzubilden. Basierend auf der Laborumgebung werden die in Tabelle 3.1 aufgeführten Kategorien und Stichwörter festgelegt.

Kategorie	Stichwörter
Raum	"alle", "Wohnzimmer", "Schlafzimmer", "Badezimmer" "Küche", "Esszimmer", "Flur"
Device	"Licht", "Fenster", "Heizung", "Vorhang", "Rollo" "Kaffeemaschine"
Manipulation	"an", "aus", "auf", "zu", "hoch", "runter"
Zahl	"acht", "neun", "zehn", "elf", ... , "fünfundzwanzig"

Tabelle 3.1: Stichwörter und Kategorien

Wake Word

Damit die Befehlsverarbeitung nicht unbeabsichtigt auslöst, wird ein Wake Word festgelegt, welches die Textverarbeitung für eine bestimmte Zeitspanne aktiviert. Damit hat der User nach Nennung des Wake Word Zeit, einen Befehl einzugeben. Um dem User die Bereitschaft des Systems zur Aufnahme von Befehlen mitzuteilen, wird bei Erkennung des Wake Word eine positive Rückmeldung gegeben.

Extraktion des Befehlsziels

Um das Ziel eines Befehls zu ermitteln, wird der Befehl zunächst nach der Kategorie "Raum" durchsucht. Dazu wird jedes aufgenommene Wort mit einer Liste vorhandener Räume verglichen, bis einer gefunden wurde. Um mehrere Devices unabhängig vom Raum zu steuern, wird das Wort "alle" ebenfalls dieser Kategorie zugeordnet. Anschließend wird der Befehl nach der Kategorie "Device" durchsucht. Mit den Kategorien Device und Raum lassen sich sämtliche Befehlsziele adressieren, wobei einige Befehle wie das Steuern des zentralen Lichts des Smart-Home auch ohne Nennung des Raumes auskommen.

Extraktion der Manipulation

Fast alle Devices der Umgebung sind in ihrer Steuerung binär, lassen sich also entweder ein- oder ausschalten, auf- oder zumachen. Wörter wie "ein" und "aus" werden demnach der Kategorie "Manipulation" zugeordnet. Mehrwertig steuerbare Devices erfordern neben dieser Kategorie außerdem die Kategorie "Zahl".

Textverarbeitung als Finite-State-Machine

Am einfachsten lässt sich die Funktionsweise der Textverarbeitung als Finite-State-Machine (FSM) darstellen. Abbildung 3.2 zeigt den Ausschnitt der FSM für die Schlafzimmersteuerung. Dabei wird von einer sinnvollen Eingabe durch den User ausgegangen, in der nach dem Wake Word genau ein Raum, ein Device und eine Manipulation genannt werden. Es handelt sich um eine vereinfachte Darstellung im Stil eines Zustandsautomaten, welcher keine Ausnahmen berücksichtigt und zur Veranschaulichung der Funktionalität dient. Die Endzustände des Automaten beschreiben den extrahierten Befehl. In der Umsetzung wird jedem dieser Zustände der jeweilige Befehl für die Smart-Home Infrastruktur zugeordnet. Der vollständige Zustandsautomat befindet sich im Anhang A.1.

Abbildung 3.2: Ausschnitt der FSM von der Schlafzimmersteuerung

3.1.5 User Feedback Konzept

Das User Interface des Systems ausschließlich akustisch konzipiert ist, müssen auch die Rückmeldungen des Systems an den User akustischer Natur sein. Diese Rückmeldungen sind vor allem dann notwendig, wenn der User die Auswirkungen seines Befehls nicht unmittelbar wahrnehmen kann. Es werden zwei verschiedene Arten der Rückmeldung definiert: positiv und negativ. Dabei sind diese durch eindeutige akustische Merkmale zu unterscheiden. Eine positive Rückmeldung besteht aus zwei Tönen mit aufsteigender Tonhöhe, eine negative aus zwei absteigenden Tönen. Folgende Rückmeldungen sind für die User Interaktion vorgesehen:

Ereignis	Art der Rückmeldung
Wake Word erkannt	positiv
Befehl erkannt und erfolgreich gesendet	positiv
Befehl erkannt aber nicht erfolgreich gesendet	negativ
Befehl nicht erkannt	negativ

Tabelle 3.2: Rückmeldungen an den User

3.1.6 MQTT Interface

Aus einem eingegebenen Sprachbefehl wird im Modul Textverarbeitung schließlich ein MQTT Befehl gebildet, welcher an die Smart-Home Infrastruktur gesendet werden muss. Dazu ist ein Interface zu einem MQTT Modul nötig, welches eine Verbindung zum MQTT Broker herstellt und aufrechterhält. Über dieses Interface wird das generierte Topic und die entsprechende Payload gesendet. Die Rückmeldung des Moduls über den Status des MQTT Befehls entscheidet abschließend, welche Art der Rückmeldung an den User ausgegeben wird.

3.1.7 Systemablauf

Abbildung 3.3 zeigt den in diesem Kapitel erörterten Ablauf des Systems mit seinen Komponenten und schafft eine Übersicht über die Funktionsweise des Systems.

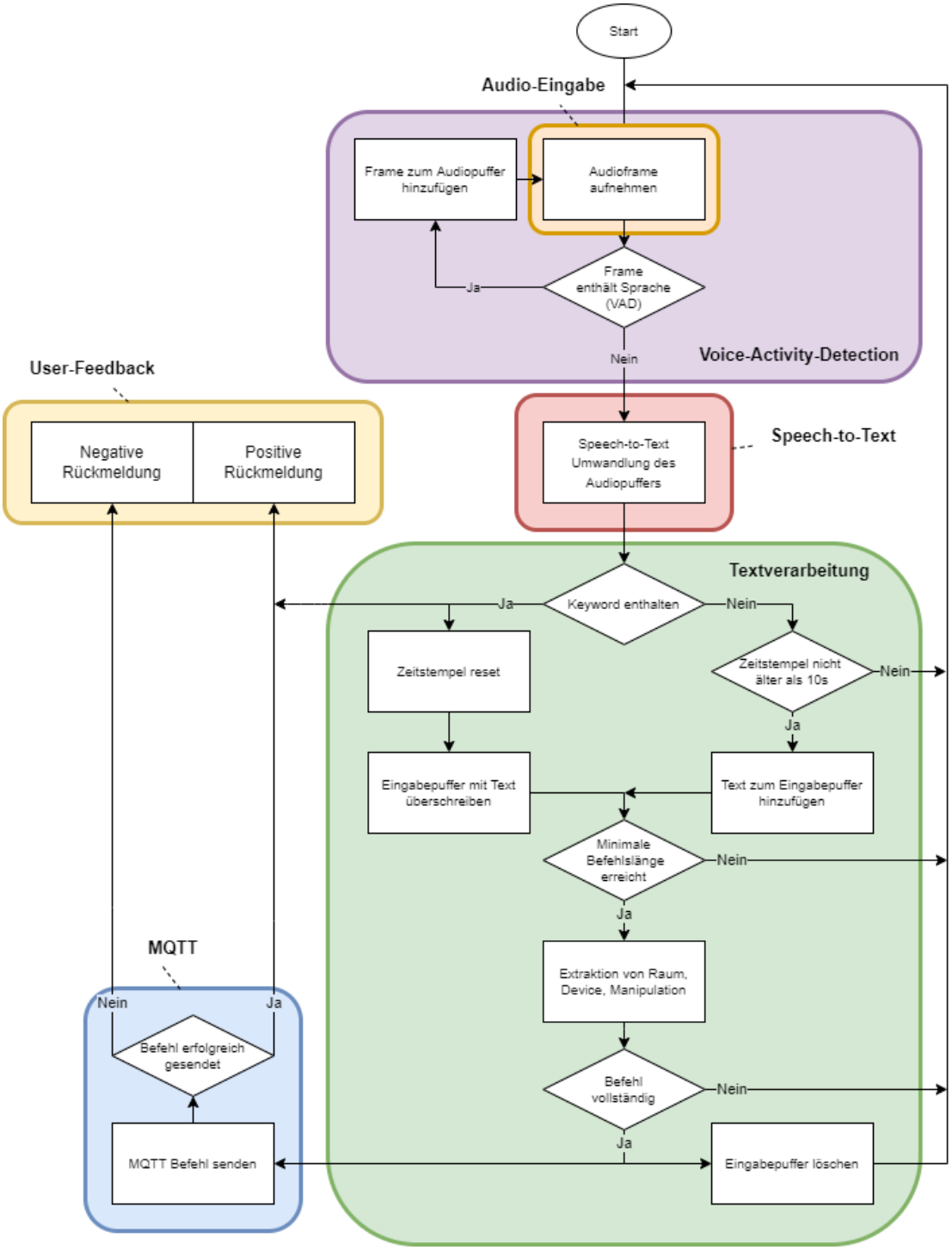


Abbildung 3.3: Systemablaufdiagramm

3.2 Auswahl des ASR Frameworks und des Sprachmodells

In diesem Kapitel wird die Wahl des vortrainierten Speech-to-Text Modells erläutert, basierend auf den in Kapitel 2.5 vorgestellten ASR Frameworks. Zunächst wird für jedes Framework ein vortrainiertes deutsches Modell ausgewählt, um diese anschießend zu vergleichen. Die ausgewählten Modelle sind in Tabelle 3.3 aufgeführt. Sowohl das Mozilla DeepSpeech Modell, als auch das NeMo QuartzNet Modell stammen vom "Sciberismo" Projekt der Universität Augsburg [11], das Kaldi Modell stammt aus dem Projekt von Milde & Köhn [27]. Bei allen Modellen handelt es sich um sogenannte "Speaker Independant Models", welche Sprecherunabhängig sind und von verschiedenen Personen verwendet werden können.

Modell	Training	WER
Kaldi	1000 h	14,29 %
DeepSpeech	1582 h	12,8 %
QuartzNet15x5	2370 h	6,6 %

Tabelle 3.3: Eckdaten der vortrainierten Speech-to-Text Models

In Tests stellten sich die meisten deutschen Kaldi-Modelle als ungeeignet für Echtzeit-ASR heraus, was meist an zu großen Verzögerungen lag. Das vorliegende Modell konnte zwar in Tests umgesetzt werden, allerdings liegt die Fehlerrate deutlich oberhalb der beiden anderen Modelle. Daher wird das Kaldi Modell für die kommenden Tests und die Umsetzung ausgeschlossen. Das DeepSpeech Modell besitzt zwar ebenfalls eine deutlich höhere WER als das QuartzNet Modell, diese kann jedoch durch Aktivieren eines external Scorer verringert werden. Daher werden im folgenden Kapitel diese beiden Modelle getestet und gegenübergestellt.

Anwendungstests

Um die Modelle unter gleichen Bedingungen zu testen, werden insgesamt 26 verschiedene Spracheingaben eingegeben, welche ausschließlich die in Tabelle 3.1 aufgelisteten Wörter und einige Füllwörter enthalten. Die Ausgaben beider Modelle sowie deren Rechenzeiten werden protokolliert. Für das DeepSpeech Modell wird zusätzlich ein external Scorer mit einem auf die Stichwörter in Tabelle 3.1 und einige Füllwörter eingeschränktem Wortschatz aktiviert (siehe Kap.3.3.3). Die Daten des Testsystems sind in Tabelle 3.4

aufgeführt. Als Audioeingabegerät wird ein *Google Pixel Buds A-Series* In-Ear-Headset verwendet.

Modell	Asus GU603HM-KR006T
CPU	Intel(R) Core(TM) i7-11800H @ 2.30GHz
RAM	16 GB
GPU	NVIDIA GeForce RTX 3060 Laptop GPU
OS	Ubuntu 20.04.5 LTS

Tabelle 3.4: Technische Daten des Testsystems

Die Tests werden in einem ruhigen, geschlossenen Raum durchgeführt. Für beide Module ist dasselbe VAD-Modul implementiert (siehe Kap.3.3.2). Die Testauswertungen sind in Abbildung 3.4 und Tabelle 3.5 dargestellt.

Auswertung der Anwendungstests

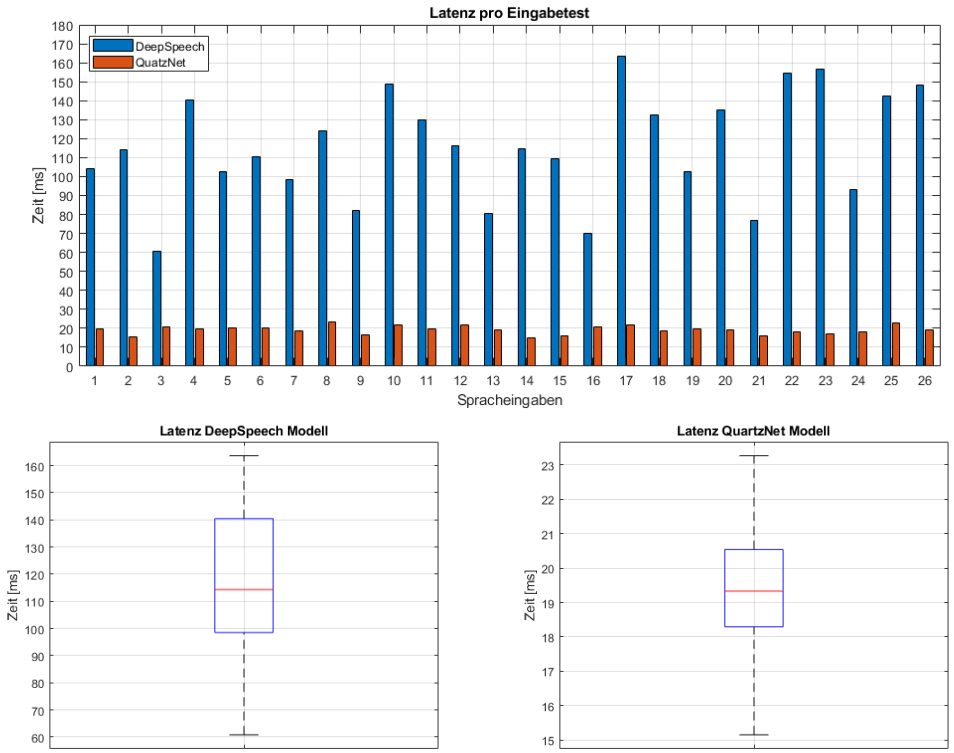


Abbildung 3.4: Ergebnisse der Anwendungstests, Latenzen

Modell	WER	Latenz-Mittelwert	Latenz-Standardabweichung
DeepSpeech	4,9 %	116 ms	28 ms
QuartzNet	11,2 %	19 ms	2 ms

Tabelle 3.5: Ergebnisse der Anwendungstests, statistische Kennwerte

In den Anwendungstests erzielt das DeepSpeech Modell bessere Leistungen als das QuartzNet Modell, mit einer um 6,3 % niedrigeren WER. Dies lässt sich auf die Aktivierung des external Scorer mit eingeschränktem Vokabular zurückführen. Das QuartzNet Modell, welches durch die GPU des Testsystems beschleunigt wird, erbringt die Ergebnisse der Speech-to-Text Umwandlung dafür im Mittel fast 100 ms schneller als das DeepSpeech Modell.

Aufgrund dieser Abschätzung des Anwendungsverhaltens der beiden Modelle wird Mozilla DeepSpeech für die Umsetzung ausgewählt. Eine zusätzliche Verzögerung ist bei einer Verbesserung der WER um 6,3 % vertretbar, solange sie niedriger als 200 ms ist [22]. Außerdem ist in Hinblick auf die Anforderungen eine geringe WER ist für dieses System vorzuziehen. DeepSpeech wird auch in Hinblick auf die Portierbarkeit des Systems ausgewählt, da es auf weniger Ressourcen angewiesen ist als QuartzNet.

3.3 Umsetzung

In diesem Kapitel wird die Implementierung des im vorhergehenden Kapitel konzipierten Systems beschrieben und dokumentiert. Auf die Umsetzung der Systemkomponenten wird in der Reihenfolge des Programmablaufes eingegangen, angefangen bei der Aufnahme der Audiodaten, bis hin zum Senden des Befehls an die Smart-Home Infrastruktur und die Rückmeldung an den User. Als Programmiersprache für die Umsetzung wird Python 3.8 gewählt, da alle betrachteten ASR Frameworks Python-Pakete bereitstellen.

3.3.1 Aufnahme der Audiodaten

Als Interface für die Aufnahme von Audiodaten wird die Bibliothek PyAudio verwendet [6]. PyAudio bietet eine Python-Anbindung für PortAudio [5], eine Cross-Platform Audio I/O Bibliothek. Mittels der PyAudio API wird auf die Audiodaten des Standard-Audioeingabegerätes des lokalen Systems zugegriffen.

3.3.2 Umsetzung der Voice Activity Detection

Das DeepSpeech Projekt bietet ein Beispiel für die Verwendung eines Speech-to-Text Models zusammen mit einem VAD-Modul, welches für diesen Anwendungsfall angepasst und erweitert wird. Grundlage des Moduls ist die Bibliothek "webrtcvad", welche ein Python Interface zum Voice Activity Detector des WebRTC Open-Source Projektes bietet [7].

Das VAD-Modul greift über PyAudio auf die Audiodaten des Eingabegerätes zu und prüft diese auf die Anwesenheit von Sprachaktivität. Erkennt das VAD-Modul diese, werden alle zusammenhängenden Audioframes, die Sprache enthalten, in den Buffer des Speech-to-Text Moduls geschrieben. Es folgt ein einzelnes Frame, welches keine Sprache enthält und damit das Ende der Spracheingabe markiert.

3.3.3 Implementierung Mozilla DeepSpeech

Um ein vortrainiertes DeepSpeech Sprachmodell zu implementieren, muss die "deepspeech" Python-Bibliothek installiert werden.

Bei Programmstart wird das Sprachmodell initialisiert, wobei zusätzlich der external Scorer aktiviert wird. Dann wird der Inputstream des Modells gestartet. Das VAD-Modul kann nun Audioframes in den Buffer des DeepSpeech Modells schreiben. Gibt das VAD-Modul ein Frame ohne Sprachaktivität aus, wird der Inputstream beendet, das Modell übersetzt den Inhalt des Buffers und gibt das Ergebnis als String aus. Der Inputstream wird neu gestartet und der Prozess beginnt von vorne.

Umsetzung External Scorer

Der external Scorer wird mit Hilfe des DeepSpeech Framework zunächst mit einem deutschen Textdatensatz der Universität Hamburg [27] und anschließend mit den Stichwörtern (3.1), dem Wake Word und einigen Füllwörtern trainiert (siehe Anhang A.2). Dazu dient das vom DeepSpeech Framework zur Verfügung gestellte Script "generate_scorer_package".

3.3.4 Umsetzung der Textverarbeitung

Dieses Unterkapitel beschreibt die Umsetzung des Moduls für die Textverarbeitung. Zunächst wird die Wahl des Wake Word zur Aktivierung der Spracheingabe erläutert, anschließend die Sprachbefehle zur Steuerung der einzelnen Devices.

Wake Word

Das Wake Word sollte möglichst kein Wort des allgemeinen Sprachgebrauchs sein, um versehentliche Spracheingaben zu vermeiden. Für diese Anwendung wurde der Name "Mark" gewählt.

Bei Erkennung des Wake Word wird ein Zeitstempel gesetzt und eine positive Rückmeldung an den User ausgegeben. Alle folgenden Spracheingaben, die das Wake Word nicht enthalten, werden nur verarbeitet, wenn der Zeitstempel nicht älter als 10 Sekunden ist. Bei allen folgenden Befehlsbeispielen wird davon ausgegangen, dass das Wake Word vorher genannt wurde und die Spracheingabe aktiv ist.

Nicht erkannte Befehle

Wird die Spracheingabe nicht vollständig oder falsch erkannt, sodass kein MQTT Befehl generiert werden kann, wird dem User eine negative Rückmeldung ausgegeben.

Steuerung der Heizung

Für die Steuerung der Heizungen muss das Device "Heizung" genannt werden zusammen mit einem beliebigen Raum, wobei mit dem Stichwort "alle" auch alle Heizungen gemeinsam angesprochen werden können. Außerdem muss entweder eine Zahl (Temperatur in Grad Celsius) genannt werden oder eine der Manipulationen "an" oder "aus". Letztere stellen eine für die jeweilige Heizung vordefinierte Temperatur ein.

Beispiel eines Sprachbefehls für die Einstellung einer Temperatur von 20 Grad im Wohnzimmer:
"Stell die Heizung im Wohnzimmer auf zwanzig Grad."

Aus diesem Satz werden folgende Wörter extrahiert und kategorisiert:

Wort	Kategorie
"Wohnzimmer"	Raum
"Heizung"	Device
"zwanzig"	Zahl

Daraus wird folgender MQTT Befehl abgeleitet:

Topic	Payload
lp/wohnzimmer/heizung/set	20

Steuerung der Fenster

Für die Steuerung der Fenster muss das Device "Fenster" genannt werden zusammen mit einem beliebigen mit Fenstern ausgestatteten Raum. Dabei können mit dem Stichwort "alle" auch alle Fenster gemeinsam angesprochen werden. Außerdem muss eine der Manipulationen "auf" oder "zu" genannt werden.

Beispiel eines Sprachbefehls für das Öffnen des Küchenfensters:
"Mach das Fenster in der Küche auf."

Aus diesem Satz werden folgende Wörter extrahiert und kategorisiert:

Wort	Kategorie
"Kueche"	Raum
"Fenster"	Device
"auf"	Manipulationsart

Daraus wird folgender MQTT Befehl abgeleitet:

Topic	Payload
lp/kueche/klappfenster/set	OPEN

Steuerung der Rollos

Für die Steuerung der Rollos muss das Device "Rollo" genannt werden zusammen mit einem beliebigen Raum, wobei mit dem Stichwort "alle" auch alle Rollos gemeinsam angesprochen werden. Außerdem muss eine der Manipulationen "auf", "zu", "hoch" oder "runter" genannt werden.

Beispiel eines Sprachbefehls für das Schließen des Rollos im Schlafzimmer:
"Mach das Rollo im Schlafzimmer runter."

Aus diesem Satz werden folgende Wörter extrahiert und kategorisiert:

Wort	Kategorie
"Schlafzimmer"	Raum
"Rollo"	Device
"runter"	Manipulationsart

Daraus wird folgender MQTT Befehl abgeleitet:

Topic	Payload
lp/schlafzimmer/rollo/set	CLOSE

Steuerung der Vorhänge

Für die Steuerung der Vorhänge muss das Device "Vorhang" genannt werden zusammen mit einem beliebigen mit Vorhängen ausgestatteten Raum. Dabei können mit dem Stichwort "alle" auch alle Vorhänge gemeinsam angesprochen werden. Außerdem muss eine der Manipulationen "auf" oder "zu" genannt werden.

Beispiel eines Sprachbefehls für das Öffnen des Vorhanges im Flur:
"Mach den Vorhang im Flur auf."

Aus diesem Satz werden folgende Wörter extrahiert und kategorisiert:

Wort	Kategorie
"Flur"	Raum
"Vorhang"	Device
"auf"	Manipulationsart

Daraus wird folgender MQTT Befehl abgeleitet:

Topic	Payload
lp/flur/vorhang/set	OPEN

Steuerung der Kaffeemaschine

Für die Steuerung der Kaffeemaschine muss das Device "Kaffeemaschine", zusammen mit einer der Manipulationen "an" oder "aus" genannt werden.

Beispiel eines Sprachbefehls für das Anschalten der Kaffeemaschine:
"Mach die Kaffeemaschine an."

Aus diesem Satz werden folgende Wörter extrahiert und kategorisiert:

Wort	Kategorie
”Kaffeemaschine”	Device
”an”	Manipulationsart

Daraus wird folgender MQTT Befehl abgeleitet:

Topic	Payload
lp/kueche/kaffemaschine/set	ON

Steuerung des Lichts

Für die Steuerung des zentralen Deckenlichts der Wohnung muss das Device ”Licht”, zusammen mit einer der Manipulationen ”an” oder ”aus” genannt werden.

Beispiel eines Sprachbefehls für das Ausschalten des Deckenlichts:
”Mach das Licht aus.”

Aus diesem Satz werden folgende Wörter extrahiert und kategorisiert:

Wort	Kategorie
”Licht”	Device
”aus”	Manipulationsart

Daraus wird folgender MQTT Befehl abgeleitet:

Topic	Payload
lp/dali/set	OFF

3.3.5 Smart-Home Adapter

Die Bibliothek ”paho-mqtt” [3] stellt ein MQTT Interface zur Verfügung, welches dazu genutzt wird, die generierten MQTT Befehle an den Broker der Smart-Home Infrastruktur zu senden. Das Interface wird bei Systemstart initialisiert und die Verbindung zum Broker wird hergestellt. Nachfolgend werden die Befehle gesendet. Wird vom MQTT

Broker der Empfang des Befehls zurückgemeldet, so wird dies dem User durch ein positives Feedback mitgeteilt. Ist die Verbindung zum Broker abgebrochen oder der Empfang des Befehls nicht bestätigt, wird dies dem User über eine negative Rückmeldung kommuniziert.

3.3.6 Umsetzung User Feedback

Das User Feedback wird mittels zweier leicht unterscheidbarer Tonfolgen umgesetzt, welche bei Bedarf abgespielt werden. Dies geschieht über ein Interface, welches die zwei Optionen bietet, eine positive oder eine negative Rückmeldung zu senden. Das Abspielen der Audiodateien über das Standard-Audioausgabegerät erfolgt mit Hilfe der Bibliothek "playsound" [4].

4 Evaluation

Im Evaluationskapitel wird die Umsetzung des Systems in Hinblick auf die im Analysekapitel herausgearbeiteten Anforderungen (siehe Kap. 2.6) bewertet. Dazu werden zunächst Funktionalität und Leistung des Systems betrachtet, gefolgt von dessen Übertragbarkeit und Anpassbarkeit.

4.1 Funktionale Evaluation

Die funktionale Evaluation untersucht ausschließlich die technische Umsetzung und Funktionalität des Systems, beginnend mit der Ansteuerung der Geräte, gefolgt von der Sprecherunabhängigkeit und der Fehlerrate.

4.1.1 Steuerung der Devices und Interaktion

Alle in den Anforderungen genannten Devices der Smart-Home Umgebung können mit dem umgesetzten System angesteuert werden. Das System kommuniziert über das MQTT Protokoll mit der Smart-Home Infrastruktur. Durch Implementierung eines Wake Word werden durch den User nicht beabsichtigte Spracheingaben vermieden. Die Erkennung des Wake Word sowie der Status der Ausführung des Befehls werden dem User durch akustische Rückmeldungen kommuniziert, genauso wie nicht erkannte Spracheingaben.

4.1.2 Sprecherunabhängigkeit

Experimentelle Tests mit unterschiedlichen Probanden weisen darauf hin, dass die Sprecherunabhängigkeit des Mozilla DeepSpeech Modells durch das Einschränken des Vokabulars mittels external Scorer nicht beeinträchtigt wurde.

4.1.3 Fehlerrate

Um die Fehlerrate des Systems in der Anwendungsumgebung abzuschätzen, werden insgesamt 41 verschiedene Sprachbefehle über ein Headset eingesprochen und die Ergebnisse dokumentiert. Gleichzeitig wird die Zeit vom Ende der Befehlseingabe bis zum Senden des MQTT Befehls gemessen, um die Latenz des Systems festzustellen (siehe Kap. 4.2). Das Testsystem ist dasselbe, welches schon für den Vergleich der Speech-to-Text Models verwendet wurde (siehe Tab. 3.4), bei dem Headset handelt es sich um ein *Google Pixel Buds A-Series* In-Ear-Headset, welches über Bluetooth mit dem Testsystem verbunden ist. Die Testergebnisse sind in Tabelle 4.1 zusammengefasst.

Anzahl der Testbefehle	41
Davon korrekt ausgeführt	38
Davon nicht ausgeführt	3
Davon falsch ausgeführt	0
Fehlerrate der Testbefehle	7,3%
Gesamtanzahl der Wörter	275
Davon korrekt erkannt	270
WER	1,8%

Tabelle 4.1: Statistische Auswertung der Fehlerraten der Anwendungstests

Der Anwendungstest ergibt über alle Spracheingaben zusammen eine WER von 1,8 %, was im Vergleich zu anderen Sprachmodellen ohne Einschränkung des Wortschatzes eine deutliche Verbesserung darstellt (vgl. Abb. 2.4). Da sich die Sprachbefehle aus mehreren Wörtern zusammensetzen und meist ein falsches Wort reicht, damit der Befehl nicht erkannt wird, akkumuliert die WER mit jedem hinzugefügten Wort. Dies führt zu einer Fehlerquote von 7,3 % bei der Erkennung von Befehlen. Von den insgesamt 41 Testbefehlen wurden drei Befehle unvollständig oder fehlerhaft erkannt, was in allen Fällen zu einem ungültigen Befehl führte. Der in Kapitel 2.6.1 erwähnte zweite Fall, in dem der User mit einem nicht beabsichtigten Befehl konfrontiert wird, trat nicht auf.

4.2 Latenzzeit

Die Latenz des Systems wird vom Ende der Spracheingabe bis zum Senden des MQTT Befehls an den Broker gemessen. Die Messung erfolgt zeitgleich zur Messung der Fehlerrate, die Details der Durchführung sind in Kapitel 4.1.3 aufgeführt.

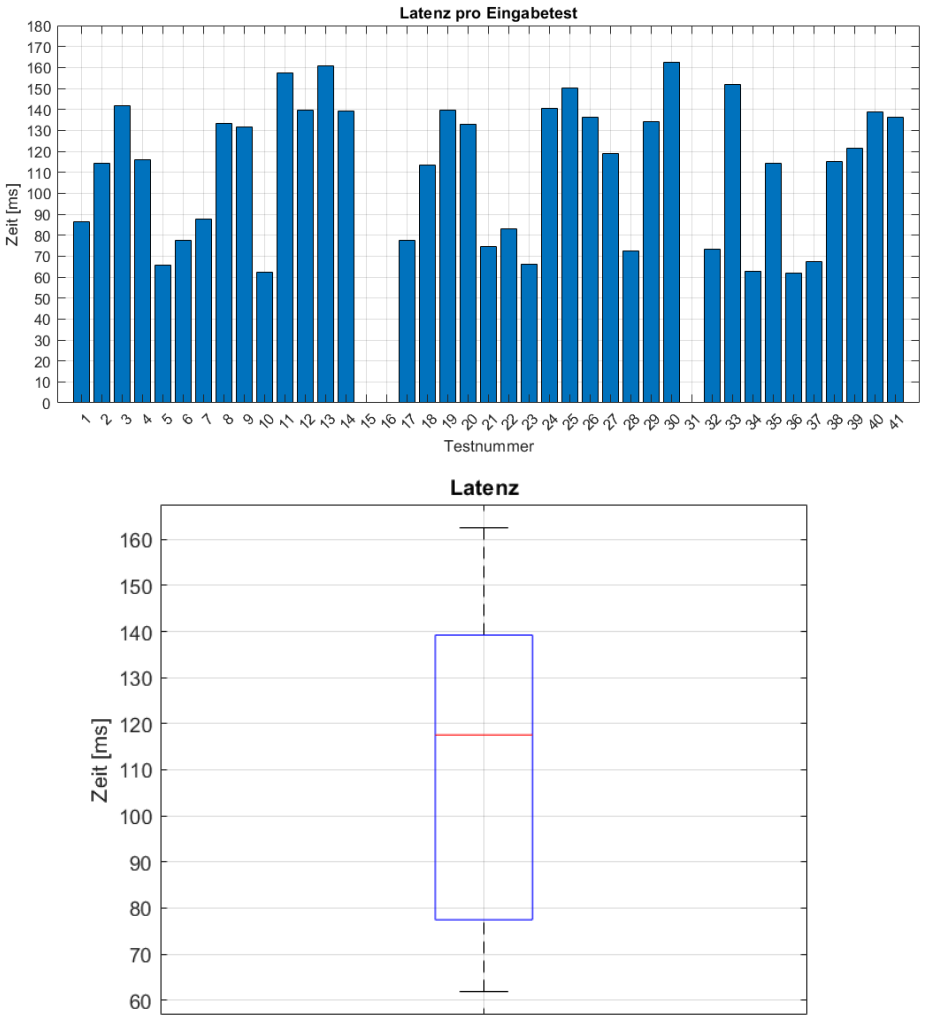


Abbildung 4.1: Ergebnisse des Systemtests, Latenzen

Die Latenzen der Testeingaben 15, 16 und 31 sind nicht aufgeführt, da diese fehlerhaft erkannt wurden und kein Befehl gesendet wurde. Die durchschnittliche Latenz der

Spracheingaben der Anwendungstests beträgt 104 ms, mit einer Standardabweichung von 33 ms. Die Latenz des Systems liegt damit noch deutlich unter der für den User kritischen Grenze von 200 ms. Die hier gemessene Latenz berücksichtigt allerdings nicht die Verzögerung durch die Smart-Home Infrastruktur. Deshalb liegt die reale Round-Trip-Time höher als die hier aufgeführte Latenz.

4.3 Übertragbarkeit und Anpassbarkeit

Hinzufügen neuer Devices

Um ein neues Device zur Sprachsteuerung hinzuzufügen, muss zunächst der external Scorer des DeepSpeech-Modells um die Bezeichnung des Devices, eventuell auch um den Raum und die möglichen Manipulationen erweitert werden. Außerdem muss die Textverarbeitung erweitert werden, sodass der gewünschte Befehl an die Smart-Home Infrastruktur generiert werden kann.

Integration in neue Smart-Home Umgebungen

Um das System in einer Smart-Home Umgebung zu integrieren, müssen alle Devices, Räume und Manipulationen, wie in Kapitel 4.3 beschrieben, in das System eingefügt werden. Dabei ermöglicht der Aufbau der Textverarbeitung auch die Integration in Umgebungen, welche nicht durch MQTT gesteuert werden.

Da es sich bei DeepSpeech um ein vergleichsweise ressourcenschonendes Speech-to-Text Modul handelt, kann das System auch auf weniger leistungsstarken Home Automation Systems implementiert werden.

5 Fazit und Ausblick

5.1 Fazit

Ergebnis dieser Arbeit ist ein Anwendung, die es ihrem User ermöglicht eine Smart-Home Umgebung mittels Sprachinteraktion zu steuern. Die Anwendung wird erfolgreich serverunabhängig und auf Basis von Open-Source Projekten implementiert. Tests der Anwendung ergeben eine vergleichsweise niedrige WER von 1,8 % (vgl. Abb. 2.4) und eine Fehlerrate bei der Befehlserkennung von 7,3 %. Die Latenz des Systems liegt mit durchschnittlich 104 ms in einem für User Interfaces akzeptablen Bereich [22].

Zunächst erfolgt im Analysekapitel (2) die Einordnung in den wissenschaftlichen Kontext der Arbeit: Cyber Physical Systems (2.1), Smart-Environments (2.1.1) und die Interaktion zwischen Mensch und Computer (2.1.2). Mit einem fiktiven Szenario wird anschließend an die Problemstellung der Arbeit herangeführt (2.2). Es folgen die Erläuterung des umgebenden Systems (2.3), eine Einführung in die technischen Ansätze der automatischen Spracherkennung (ASR) (2.4) und die Vorstellung einiger ausgewählter Open-Source ASR Frameworks (2.5). Abschließend werden die Anforderungen an das Zielsystem definiert (2.6).

Kapitel 3 umfasst Konzept und Umsetzung des Systems. Dieses wird zunächst in einzelne Komponenten aufgeteilt (siehe Abb.3.1), deren Funktionen und Interaktionen erläutert werden. Darauf aufbauend wird ein Systemablaufplan entwickelt, welcher als Vorlage für die Umsetzung dient (siehe Abb.3.3). Vor der Umsetzung werden in Kapitel 3.2 die im Analysekapitel vorgestellten ASR Frameworks verglichen, darauf aufbauend wird das Mozilla DeepSpeech Framework für die Anwendung ausgewählt. In Kapitel 3.3 wird die Umsetzung des konzipierten Systems erläutert. Kapitel 3.3.4 enthält eine Auflistung der Befehle, welche zur Steuerung des Smart-Homes eingesetzt werden können.

Zuletzt wird das entwickelte System in Hinblick auf die in Kapitel 2.6 definierten Anforderungen evaluiert (4). Die funktionale Evaluation zeigt, dass das entworfene System

mit einer Word-Error-Rate (WER) von 1,8% arbeitet. Die Fehlerrate der Befehlserkennung liegt bei 7,3 % (siehe Tab.4.1). Sämtliche in den Anforderungen genannten Devices können durch Sprachbefehle des Users angesteuert werden, wobei der User über den aktuellen Zustand des Systems durch akustische Rückmeldungen informiert wird. Die durchschnittliche Verzögerung von der Eingabe eines Befehls bis zur Weitergabe desselben an die Smart-Home Infrastruktur beträgt dabei 104 ms (siehe Abb. 4.1). Abschließend wurde die Übertragbarkeit und Anpassbarkeit erläutert (4.3).

5.2 Ausblick

Im Folgenden werden, aufbauend auf der in Kapitel 4 beschriebenen Evaluation, einige mögliche Erweiterungen und weiterführende Funktionalitäten für das umgesetzte Systems betrachtet.

In Bezug auf die Textverarbeitung sind einige Erweiterungen denkbar, beispielsweise würde die Umsetzung einer satzübergreifenden Semantik die Steuerung mehrerer Devices mit nur einem Befehl erlauben. Auch ließe sich die Steuerung durch Einführen von Synonymen erweitern, sodass beispielsweise neben dem Stichwort "Vorhang" auch "Gardine" verwendet werden könnte. Ebenso wäre die Implementierung einer relativen Ansteuerung der Devices als Systemerweiterung denkbar. Dies würde dem User erlauben, für Devices wie Heizungen, nicht nur eine absolute Temperatur, sondern auch eine Temperatur relativ zur aktuellen einzustellen.
Auch das User Feedback ließe sich erweitern. Denkbar ist hier eine Rückmeldung des erkannten Befehls über ein Text-to-Speech Interface.
Die Erweiterung des Systems um neue Sprachbefehle, Devices oder Räume, sowie die Integration des Systems in ein Smart-Home können mit einer Konfigurationsdatei vereinfacht werden, sodass der User nicht die Umsetzung des Systems anpassen muss.

Literaturverzeichnis

[1] *Living Place HAW Hamburg*. https://livingplace.haw-hamburg.de/. 2022. – Online; accessed 09.06.2022

[2] *NVIDIA NeMo Toolkit*. https://developer.nvidia.com/nvidia-nemo. 2022. – Online; accessed 22.05.2022

[3] *paho-mqtt*. https://www.eclipse.org/paho/. 2022. – Online; accessed 12.11.2022

[4] *playsound*. https://github.com/TaylorSMarks/playsound. 2022. – Online; accessed 12.11.2022

[5] *PortAudio*. http://www.portaudio.com/. 2022. – Online; accessed 12.11.2022

[6] *PyAudio*. https://people.csail.mit.edu/hubert/pyaudio/. 2022. – Online; accessed 12.11.2022

[7] *WebRTC*. https://webrtc.org/. 2022. – Online; accessed 12.11.2022

[8] Ardila, Rosana ; Branson, Megan ; Davis, Kelly ; Henretty, Michael ; Kohler, Michael ; Meyer, Josh ; Morais, Reuben ; Saunders, Lindsay ; Tyers, Francis M. ; Weber, Gregor: *Common Voice: A Massively-Multilingual Speech Corpus*. 2019

[9] Banks, Andrew ; Briggs, Ed ; Borgendale, Ken ; Gupta, Rahul: *MQTT Version 5.0*. https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html

[10] Baudisch, Patrick: Natural user interface hardware, 2013, S. xii–xii

[11] Bermuth, Daniel ; Poeppel, Alexander ; Reif, Wolfgang: *Scribosermo: Fast Speech-to-Text models for German and other Languages*. 2021

[12] Cook, Diane J. ; Das, Sajal K.: Overview. In: *Smart Environments*. John Wiley & Sons, Inc., 2005, S. 1–10

[13] Geisberger, Eva ; Broy, Manfred: *Agenda CPS, Integrierte Forschungsagenda Cyber-Physical Systems*. 2012

[14] Georgescu, Alexandru-Lucian ; Pappalardo, Alessandro ; Cucu, Horia ; Blott, Michaela: Performance vs. hardware requirements in state-of-the-art automatic speech recognition. In: *EURASIP Journal on Audio, Speech, and Music Processing* (2021)

[15] Griffor, Edward R. ; Greer, Chris ; Wollman, David A. ; Burns, Martin J.: Framework for cyber-physical systems: volume 1, overview. National Institute of Standards and Technology, 2017. – Forschungsbericht

[16] Gustafson, Sean ; Bierwirth, Daniel ; Baudisch, Patrick: Imaginary Interfaces: Spatial Interaction with Empty Hands and without Visual Feedback. In: *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology*, Association for Computing Machinery, 2010, S. 3–12

[17] Hannun, Awni ; Case, Carl ; Casper, Jared ; Catanzaro, Bryan ; Diamos, Greg ; Elsen, Erich ; Prenger, Ryan ; Satheesh, Sanjeev ; Sengupta, Shubho ; Coates, Adam ; Ng, Andrew Y.: *Deep Speech: Scaling up end-to-end speech recognition*. 2014

[18] Heafield, Kenneth ; Pouzyrevsky, Ivan ; Clark, Jonathan H. ; Koehn, Philipp: Scalable Modified Kneser-Ney Language Model Estimation. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Association for Computational Linguistics, 2013, S. 690–696

[19] Ishii, Hiroshi ; Ullmer, Brygg: Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms. In: *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, Association for Computing Machinery, 1997, S. 234–241

[20] Joshi, Vikas ; Zhao, Rui ; Mehta, Rupesh R. ; Kumar, Kshitiz ; Li, Jinyu: Transfer Learning Approaches for Streaming End-to-End Speech Recognition System. (2020)

[21] Juang, B. ; Rabiner, Lawrence: Automatic Speech Recognition - A Brief History of the Technology Development. (2005)

[22] Kohrs, Christin ; Angenstein, Nicole ; Brechmann, André: Delays in Human-Computer Interaction and Their Effects on Brain Activity. In: *PLOS ONE* 11 (2016)

[23] Kola, Jonathan ; Espy-Wilson, Carol ; Pruthi, Tarun: Voice activity detection. In: *Merit Bien* (2011), S. 1–6

[24] Kriman, Samuel ; Beliaev, Stanislav ; Ginsburg, Boris ; Huang, Jocelyn ; Kuchaiev, Oleksii ; Lavrukhin, Vitaly ; Leary, Ryan ; Li, Jason ; Zhang, Yang: QuartzNet: Deep Automatic Speech Recognition with 1D Time-Channel Separable Convolutions. (2019)

[25] Kunze, Julius ; Kirsch, Louis ; Kurenkov, Ilia ; Krug, Andreas ; Johannsmeier, Jens ; Stober, Sebastian: Transfer Learning for Speech Recognition on a Budget. In: *Proceedings of the 2nd Workshop on Representation Learning for NLP*, Association for Computational Linguistics, 2017, S. 168–177

[26] Li, Kin F. ; Tai, James S.: Dynamic Time Warping in Hardware. In: *Proceedings of the 14th International Conference on Information Integration and Web-Based Applications & Services*, Association for Computing Machinery, 2012, S. 132–137

[27] Milde, Benjamin ; Köhn, Arne: Open Source Automatic Speech Recognition for German. In: *Proceedings of ITG 2018*, 2018, S. 251–255

[28] Panayotov, Vassil ; Chen, Guoguo ; Povey, Daniel ; Khudanpur, Sanjeev: Librispeech: An ASR corpus based on public domain audio books. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, S. 5206–5210

[29] Permanasari, Yurika ; Harahap, Erwin H. ; Ali, Erwin P.: Speech recognition using Dynamic Time Warping (DTW). In: *Journal of Physics: Conference Series* 1366 (2019), S. 012091

[30] Poslad, Stefan: *Human–Computer Interaction.* S. 135–178. In: *Ubiquitous Computing: Smart Devices, Environments and Interactions*, 2009

[31] Poslad, Stefan: *Ubiquitous Computing: Basics and Vision.* S. 1–40. In: *Ubiquitous Computing: Smart Devices, Environments and Interactions*, 2009

[32] Povey, Daniel ; Ghoshal, Arnab ; Boulianne, Gilles ; Burget, Lukas ; Glembek, Ondrej ; Goel, Nagendra ; Hannemann, Mirko ; Motlicek, Petr ; Qian, Yanmin ; Schwarz, Petr ; Silovsky, Jan ; Stemmer, Georg ; Vesely, Karel: The

Kaldi Speech Recognition Toolkit. In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, IEEE Signal Processing Society, 2011

[33] Singh, Deepti ; Boland, Frank: Voice Activity Detection. In: *XRDS* 13 (2007), S. 7

[34] Veysov, Alexander ; Voronin, Dimitrii: One Voice Detector to Rule Them All. In: *The Gradient* (2022)

[35] Wang, Dong ; Wang, Xiaodong ; Lv, Shaohe: An Overview of End-to-End Automatic Speech Recognition. In: *Symmetry* 11 (2019), Nr. 8, S. 1018

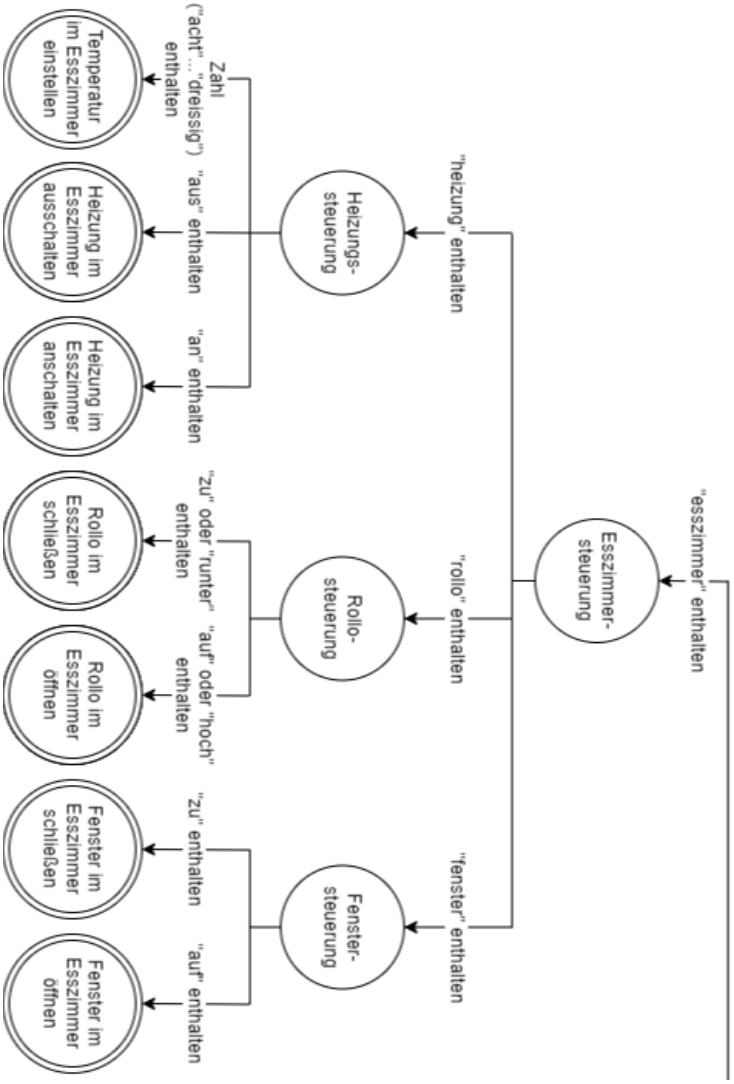
[36] Weiser, Mark: The Computer for the 21st Century. In: *SIGMOBILE Mob. Comput. Commun. Rev.* 3 (1999), S. 3–11

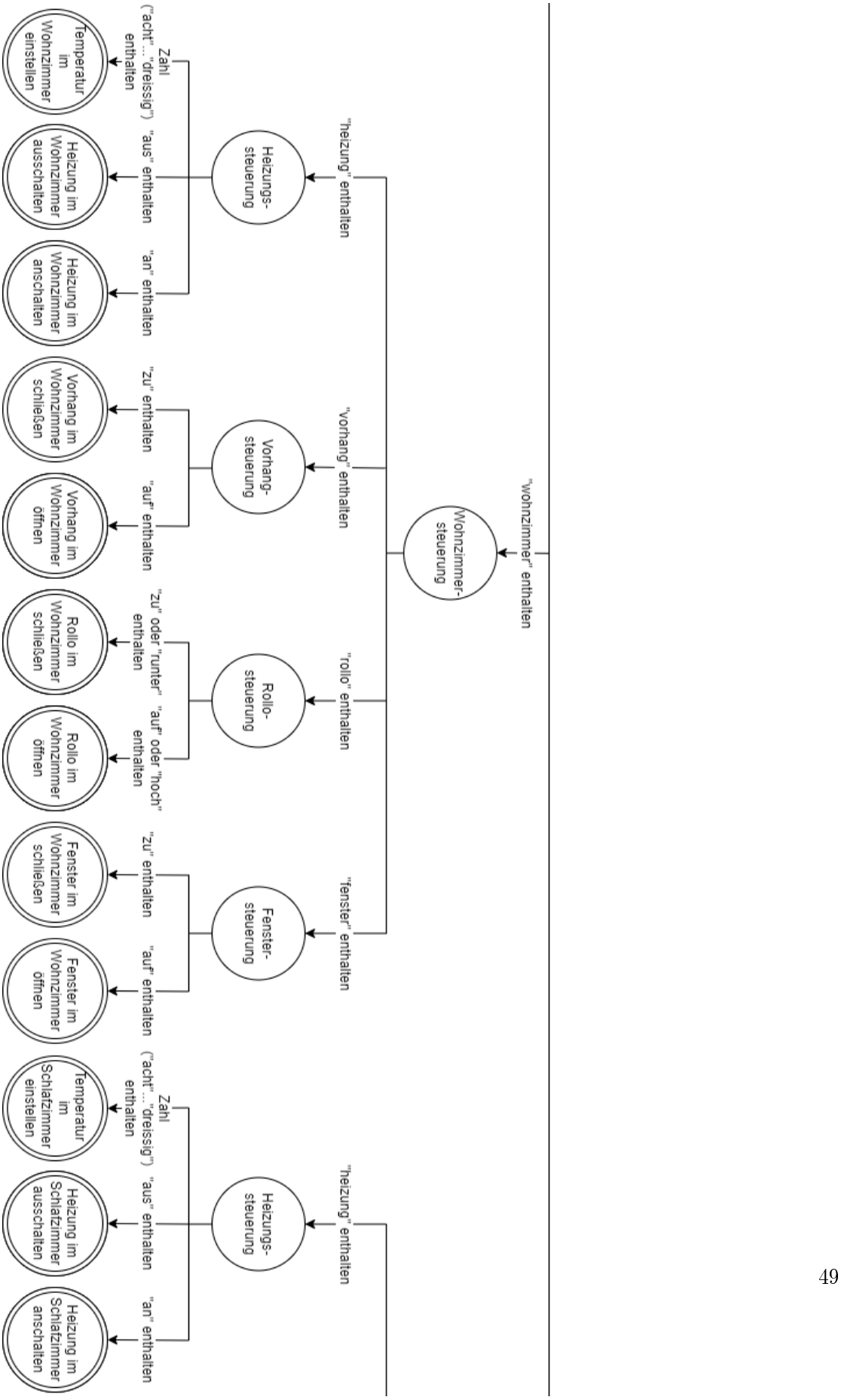
[37] Weiser, Mark ; Brown, John S.: Designing Calm Technology. (1995)

[38] Zilvan, Vicky ; Heryana, Ana ; Yuliani, Asri R. ; Krisnandi, Dikdik ; Yuwana, R. S. ; Pardede, Hilman F.: Front-End Based Robust Speech Recognition Methods: A Review. In: *The 2021 International Conference on Computer, Control, Informatics and Its Applications*, Association for Computing Machinery, 2021, S. 136–140

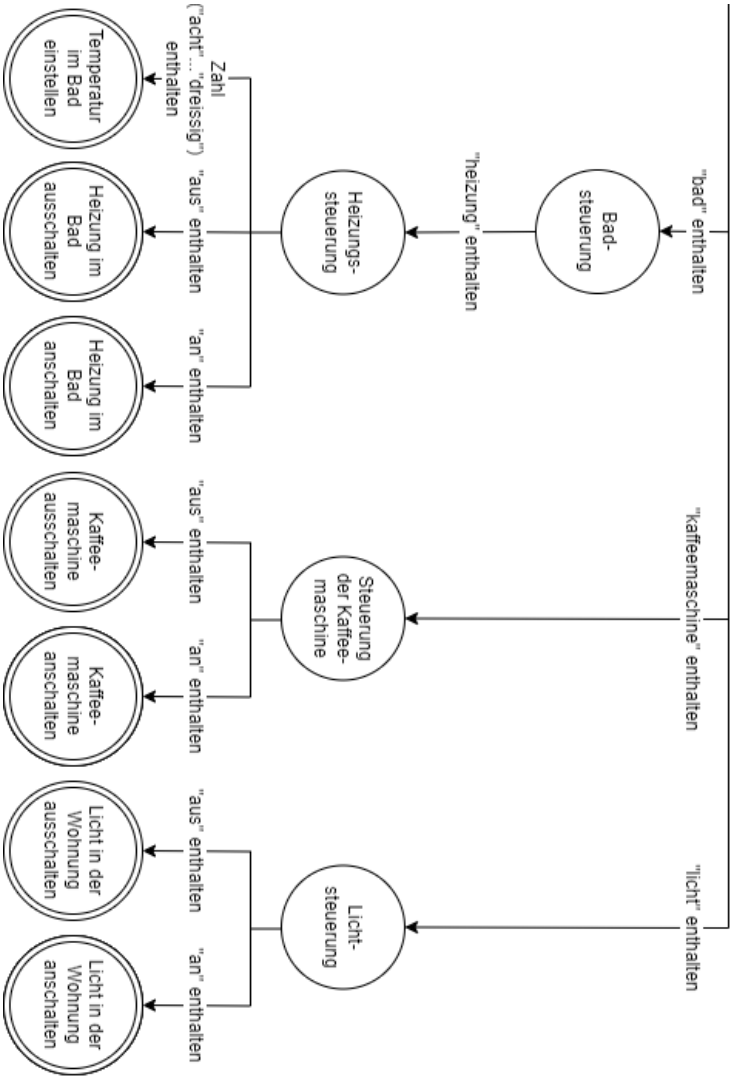
A Anhang

A.1 Finite-State-Machine der Sprachsteuerung




"wohnzimmer" enthalten
Wohnzimmer-steuerung
"heizung" enthalten
Heizungs-steuerung
Zahl ("acht"..."dreissig") enthalten
Temperatur im Wohnzimmer einstellen
"aus" enthalten
Heizung im Wohnzimmer ausschalten
"an" enthalten
Heizung im Wohnzimmer anschalten
"vorhang" enthalten
Vorhang-steuerung
"zu" enthalten
Vorhang im Wohnzimmer schließen
"auf" enthalten
Vorhang im Wohnzimmer öffnen
"rollo" enthalten
Rollo-steuerung
"zu" oder "runter" enthalten
Rollo im Wohnzimmer schließen
"auf" oder "hoch" enthalten
Rollo im Wohnzimmer öffnen
"fenster" enthalten
Fenster-steuerung
"zu" enthalten
Fenster im Wohnzimmer schließen
"auf" enthalten
Fenster im Wohnzimmer öffnen
"heizung" enthalten
Heizungs-steuerung
Zahl ("acht"..."dreissig") enthalten
Temperatur im Schlafzimmer einstellen
"aus" enthalten
Heizung im Schlafzimmer ausschalten
"an" enthalten
Heizung im Schlafzimmer anschalten

Neuer Inputstring
Keyword enthalten oder Eingabe noch aktiv
Text-verarbeitung starten
"schlafzimmer" enthalten
Schlafzimmer-steuerung
"vorhang" enthalten
Vorhang-steuerung
"zu" enthalten
Vorhang im Schlafzimmer schließen
"auf" enthalten
Vorhang im Schlafzimmer öffnen
"rollo" enthalten
Rollo-steuerung
"zu" oder "runter" enthalten
Rollo im Schlafzimmer schließen
"auf" oder "hoch" enthalten
Rollo im Schlafzimmer öffnen
"fenster" enthalten
Fenster-steuerung
"zu" enthalten
Fenster im Schlafzimmer schließen
"auf" enthalten
Fenster im Schlafzimmer öffnen
"kueche" enthalten
Küchen-steuerung
"rollo" enthalten
Rollo-steuerung
"zu" oder "runter" enthalten
Rollo in der Küche schließen
"auf" oder "hoch" enthalten
Rollo in der Küche öffnen
"fenster" enthalten
Fenster-steuerung
"zu" enthalten
Fenster in der Küche schließen
"auf" enthalten
Fenster in der Küche öffnen
"flur" enthalten
Flur-steuerung
"vorhang" enthalten
Vorhang-steuerung
"zu" enthalten
Vorhang im Flur schließen
"auf" enthalten
Vorhang im Flur öffnen



A.2 Vokabular des DeepSpeech Modells

Vollständiges Vokabular: {an, aus, im, auf, zu, hoch, runter, in, der, die, den, das, alle, grad, mach, stell, licht, vorhang, fenster, heizung, heizungen, rollo, rollos, tuer, kaffeemaschine, wohnzimmer, esszimmer, schlafzimmer, kueche, bad, flur, mark, zehn, elf, zwoelf, dreizehn, vierzehn, fuenfzehn, sechzehn, siebzehn, achtzehn, neunzehn, zwanzig, einundzwanzig, zweiundzwanzig, , dreiundzwanzig, vierundzwanzig, fuenfundzwanzig, sechsundzwanzig, siebenundzwanzig, achtundzwanzig, neunundzwanzig, dreissig}

Erklärung zur selbstständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original